# Contents


- 1 Installing a ceph cluster and configuring rbd-backed cinder volumes.
  - 1.1 First steps
- 2 Choosing Your Configuration
  - 2.1 Ceph Standalone Node Deployment
  - 2.2 Ceph MON on the Controller Only and OSD on All Compute Nodes
  - 2.3 Ceph Multiple MON On Specified Controller and Compute Nodes, with OSD on Separate Compute Nodes
  - 2.4 Ceph MON and OSD on the Same Nodes
    - 2.4.1 Making a standalone OSD node in a combined node environment
  - 2.5 Deploying a Standalone Cinder Volume OSD node
  - 2.6 Configuring Glance to use Ceph
  - 2.7 Configuring Cinder to use Ceph
- 3 Ceph Node Installation and Testing
  - 3.1 Testing
    - 3.1.1 Testing Cinder
    - 3.1.2 Testing Glance


# Installing a ceph cluster and configuring rbd-backed cinder volumes.

## First steps

- Install your build server
- Run puppet_modules.py to download the necessary puppet modules
- Edit site.pp to fit your configuration.
- You must define one MON and at least one OSD to use Ceph.
- It is recommended that you zero the first several blocks on each disk that will be used for Ceph OSD data storage. This step is required if you're using disks that have been used in a previous Ceph deployment. The command "dd if=/dev/zero of=/dev/sdX bs=100M count=1" (with "sdX" replaced with an appropriate device name) will suffice.

# Choosing Your Configuration

Cisco COI Grizzly g.1 release only supports standalone ceph nodes. Please follow only those instructions. Cisco COI Grizzly g.2 release supports standalone and integrated. Integrated options allow you to run MON on control and compute servers, along with OSD on compute servers. You can also have standalone cinder-volume nodes as OSD servers.

For all ceph configurations, uncomment the following in site.pp and change the values as are appropriate for your deployment:

```
$ceph_auth_type         = 'cephx'
$ceph_monitor_fsid      = 'e80afa94-a64c-486c-9e34-d55e85f26406'
$ceph_monitor_secret    = 'AQAJzNxR+PNRIRAA7yUp9hJJdWZ3PVz242Xjiw=='
$ceph_monitor_port      = '6789'
$ceph_monitor_address   = $::ipaddress
$ceph_cluster_network   = '10.0.0.0/24'
$ceph_public_network    = '10.0.0.0/24'
$ceph_release           = 'cuttlefish'
$cinder_rbd_user        = 'admin'
$cinder_rbd_pool        = 'volumes'
```

```
$cinder_rbd_secret_uuid = 'e80afa94-a64c-486c-9e34-d55e85f26406'

Exec {
  path        => '/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin',
#  environment => "https_proxy=$::proxy",
}
```

## Ceph Standalone Node Deployment

Support for standalone Ceph nodes is available in g.1 and newer releases. Configure the cobbler node entries for your Ceph servers.

Uncomment or add the Puppet Ceph node entries as shown below. This entry is necessary on the first Mon node only:

```
  if !empty($::ceph_admin_key) {
  @@ceph::key { 'admin':
    secret       => $::ceph_admin_key,
    keyring_path => '/etc/ceph/keyring',
  }
  }

  class {'ceph_mon': id => 0 }
}
```

On any additional Mon nodes, you only need the following:

```
class {'ceph_mon': id => 1 }
```

YOU MUST INCREMENT THIS ID, AND IT MUST BE UNIQUE TO EACH MON!

Ceph osd nodes need the following:

```
  class { 'ceph::conf':
    fsid            => $::ceph_monitor_fsid,
    auth_type       => $::ceph_auth_type,
    cluster_network => $::ceph_cluster_network,
    public_network  => $::ceph_public_network,
  }
  class { 'ceph::osd':
    public_address  => '10.0.0.3',
    cluster_address => '10.0.0.3',
  }
  ceph::osd::device { '/dev/sdb': }
```

Change '/dev/sdb' to the disk you want to use. You may specify additional disks by duplicating the line and changing the device name.

## Ceph MON on the Controller Only and OSD on All Compute Nodes

This option is available in g.2 and newer releases. Uncomment the following in your site.pp file:

```
$controller_has_mon = true
$osd_on_compute = true
```

Uncomment the following in your control server puppet node definition:

```
    if !empty($::ceph_admin_key) {
```

```
    @@ceph::key { 'admin':
      secret       => $::ceph_admin_key,
      keyring_path => '/etc/ceph/keyring',
    }
    }

    # each MON needs a unique id, you can start at 0 and increment as needed.
    class {'ceph_mon': id => 0 }
```

Add the following to each compute server puppet node definition

```
  class { 'ceph::conf':
    fsid            => $::ceph_monitor_fsid,
    auth_type       => $::ceph_auth_type,
    cluster_network => $::ceph_cluster_network,
    public_network  => $::ceph_public_network,
  }
  class { 'ceph::osd':
    public_address  => '10.0.0.3',
    cluster_address => '10.0.0.3',
  }
  # Specify the disk devices to use for OSD here.
  # Add a new entry for each device on the node that ceph should consume.
  # puppet agent will need to run four times for the device to be formatted,
  #   and for the OSD to be added to the crushmap.
  ceph::osd::device { '/dev/sdb': }
```

## Ceph Multiple MON On Specified Controller and Compute Nodes, with OSD on Separate Compute Nodes

This option is available in g.2 and newer releases. You cannot cohabitate mons and osds on the same server in this use case.

Uncomment the following:

```
$controller_has_mon = true
$computes_have_mons = false
```

Uncomment the following in your control server puppet node definition:

```
    if !empty($::ceph_admin_key) {
    @@ceph::key { 'admin':
      secret       => $::ceph_admin_key,
      keyring_path => '/etc/ceph/keyring',
    }
    }

    # each MON needs a unique id, you can start at 0 and increment as needed.
    class {'ceph_mon': id => 0 }
```

For each additional mon on a compute node, add the following:

```
  # each MON needs a unique id, you can start at 0 and increment as needed.
  class {'ceph_mon': id => 0 }

for each compute node that does NOT contain a mon, you can specify the OSD configuration
  class { 'ceph::conf':
    fsid            => $::ceph_monitor_fsid,
    auth_type       => $::ceph_auth_type,
```

```
    cluster_network => $::ceph_cluster_network,
    public_network  => $::ceph_public_network,
  }
  class { 'ceph::osd':
    public_address  => '10.0.0.3',
    cluster_address => '10.0.0.3',
  }
  # Specify the disk devices to use for OSD here.
  # Add a new entry for each device on the node that ceph should consume.
  # puppet agent will need to run four times for the device to be formatted,
  #   and for the OSD to be added to the crushmap.
  ceph::osd::device { '/dev/sdb': }
```

## Ceph MON and OSD on the Same Nodes

This feature will be available in g.3 and later releases. It is not supported in g.2 or ealier.

**WARNING: YOU MUST HAVE AN ODD NUMBER OF MON NODES.**

You can have as many OSD nodes as you like, but the MON nodes must be odd to reach a quorum.

First, uncomment the ceph_combo line in site.pp:

```
# Another alternative is to run MON and OSD on the same node. Uncomment
# $ceph_combo to enable this feature. You will NOT need to enabled
# $osd_on_compute, $controller_has_mon, or $computes_have_mon for this
# feature. You will need to specify the normal MON and OSD definitions
# for each puppet node as usual.
$ceph_combo = true
```

Do NOT uncomment these variables

```
# $controller_has_mon = true
# $osd_on_compute = true
# $computes_have_mons = false
```

You will need to specify the normal MON and OSD definitions for each puppet node as usual:

```
node 'compute-server01' inherits os_base {
  class { 'compute':
    internal_ip        => '192.168.242.21',
    #enable_dhcp_agent => true,
    #enable_l3_agent   => true,
    #enable_ovs_agent  => true,
  }

  # If you want to run ceph mon0 on your controller node, uncomment the
  # following block. Be sure to read all additional ceph-related
  # instructions in this file.
  # Only mon0 should export the admin keys.
  # This means the following if statement is not needed on the additional
  # mon nodes.
  if !empty($::ceph_admin_key) {
    @@ceph::key { 'admin':
      secret       => $::ceph_admin_key,
      keyring_path => '/etc/ceph/keyring',
    }
  }

  # Each MON needs a unique id, you can start at 0 and increment as needed.
```

```
  class {'ceph_mon': id => 0 }

  class { 'ceph::osd':
    public_address  => '192.168.242.21',
    cluster_address => '192.168.242.21',
  }

  # Specify the disk devices to use for OSD here.
  # Add a new entry for each device on the node that ceph should consume.
  # puppet agent will need to run four times for the device to be formatted,
  #   and for the OSD to be added to the crushmap.
  ceph::osd::device { '/dev/sdb': }
}
```

**Making a standalone OSD node in a combined node environment**

Add the following to your puppet OSD node in site.pp

```
  class { 'ceph::conf':
    fsid => $::ceph_monitor_fsid,
  }
```

## Deploying a Standalone Cinder Volume OSD node

This option is available in g.1 and newer releases. For each puppet node definition, add the following:

```
  # if you are using rbd, uncomment the following ceph classes
  class { 'ceph::conf':
    fsid            => $::ceph_monitor_fsid,
    auth_type       => $::ceph_auth_type,
    cluster_network => $::ceph_cluster_network,
    public_network  => $::ceph_public_network,
  }
  class { 'ceph::osd':
    public_address  => '192.168.242.22',
    cluster_address => '192.168.242.22',
  }
```

## Configuring Glance to use Ceph

This option is available in g.1 and newer releases. Uncomment the following:

```
# $glance_ceph_enabled = true
# $glance_ceph_user    = 'admin'
# $glance_ceph_pool    = 'images'
```

and change $glance_backend to 'rbd':

```
# Glance backend configuration, supports 'file', 'swift', or 'rbd'.
$glance_backend      = 'rbd'
```

## Configuring Cinder to use Ceph

This option is available in g.1 or newer releases. Uncomment the following:

```
# The cinder_ceph_enabled configures cinder to use rbd-backed volumes.
# $cinder_ceph_enabled           = true
```

Then change $cinder_storage_driver to 'rbd'

```
$cinder_storage_driver         = 'rbd'
```

# Ceph Node Installation and Testing

If you do not set puppet to autostart in the site.pp, you will have to run the agent manually as shown here. Regardless of the start method, the agent must run at least four times on each node running any Ceph services in order for Ceph to be properly configured.

- First bring up the mon0 node and run:

```
apt-get update
run 'puppet agent -t -v --no-daemonize' at least four times
```

- Then bring up the OSD node(s) and run:

```
 apt-get update
run 'puppet agent -t -v --no-daemonize' at least four times
```

- The ceph cluster will now be up. You can verify by logging in to the mon0 node and running the 'ceph status' command. The "monmap" line should show 1 more more mons (depending on the number you configured). The osdmap shoudl show 1 or more OSD's (depending on the number you configured) and the OSD should be marked as "up". There will be one OSD per disk configured eg. if you have a single OSD node with three disks available for ceph, you will have 3 OSDs show up in your 'ceph status'.

```
$ ceph status
health HEALTH_WARN 320 pgs degraded; 320 pgs stuck unclean; recovery 2/4 degraded (50.000%)
monmap e1: 1 mons at {0=192.168.2.71:6789/0}, election epoch 2, quorum 0 0
osdmap e7: 1 osds: 1 up, 1 in
pgmap v17: 320 pgs: 320 active+degraded; 138 bytes data, 4131 MB used, 926 GB / 930 GB avail; 0B/s
mdsmap e1: 0/0/1 up
```

- If your OSD is not marked as up, you will NOT be able to create block storage until it is.

- Note: If You are using a disk that was previously used as an osd device, you must write zeros to the drive. Do this by running

```
dd if=/dev/zero of=/dev/DISK bs=1M count=100';
```

If you do not zero the disk, your OSD installation will fail.

- Installing your compute nodes will run the necessary commands to create the volumes pool and the client.volumes account.

- Your compute nodes will be automatically configured to use ceph for block storage.

## Testing

### Testing Cinder

- Once these steps are complete, you should be able to create a rbd-backed volume and attach it to an instance as normal.

```
root@control-server01:~# cinder create --display-name myvol 1
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|     attachments     |                  []                  |
|  availability_zone  |                 nova                 |
|       bootable      |                false                 |
|      created_at     |      2013-09-25T14:49:17.484197      |
| display_description |                 None                 |
|     display_name    |                 myvol                |
|          id         | b6d78eb1-e7d7-453d-969b-1b8a23afdd38 |
|       metadata      |                  {}                  |
|         size        |                  4                   |
|     snapshot_id     |                 None                 |
|     source_volid    |                 None                 |
|        status       |               creating               |
|     volume_type     |                 None                 |
+---------------------+--------------------------------------+
root@control-server01:~#
root@control-server01:~# cinder list
+--------------------------------------+-----------+--------------+------+-------------+----------
|                  ID                  |   Status  | Display Name | Size | Volume Type | Bootable
+--------------------------------------+-----------+--------------+------+-------------+----------
| b6d78eb1-e7d7-453d-969b-1b8a23afdd38 | available |    myvol     |  4   |     None    |  false
+--------------------------------------+-----------+--------------+------+-------------+----------
root@control-server01:~#
```

Note the volume's ID in the output above.

Check Ceph to see that the new volume exists

```
root@control-server01:~# rbd --pool volumes ls
volume-b6d78eb1-e7d7-453d-969b-1b8a23afdd38
root@control-server01:~#
```

This command should return a list of UUIDs, of which you will see the one matching the output of cinder commands above. This is your volume.

- For a moment, depending on the speed of your ceph cluster, "cinder list" will show the volume status as "creating".
- After it's created, it should mark the volume "available".
- Failure states will either be "error" or a indefinite "creating" status. If this is the case, check the /var/log/cinder/cinder-volume.log for any errors.

Next, you can attach the volume to a running instance. First, use the "nova list" command to find the UUID of the instance to which you want to attach the volume. Then use the "nova volume-attach \[instance id\] \[volume id\] auto" command to attach the volume to the instance.

```
root@control-server01:~# nova list
+--------------------------------------+---------------+--------+------------+-------------+------
| ID                                   | Name          | Status | Task State | Power State | Netwo
+--------------------------------------+---------------+--------+------------+-------------+------
| 68070cdd-8953-4307-99dc-6f346c876f65 | cirros_test1  | ACTIVE | None       | Running     | net10
| 7876fb8c-a202-421e-8711-c15362c9699f | precise_test1 | ACTIVE | None       | Running     | net10
+--------------------------------------+---------------+--------+------------+-------------+------
root@control-server01:~# nova volume-attach 68070cdd-8953-4307-99dc-6f346c876f65 b6d78eb1-e7d7-453
+----------+--------------------------------------+
| Property | Value                                |
```

```
+----------+--------------------------------------+
| device   | /dev/vdb                             |
| serverId | 68070cdd-8953-4307-99dc-6f346c876f65 |
| id       | b6d78eb1-e7d7-453d-969b-1b8a23afdd38 |
| volumeId | b6d78eb1-e7d7-453d-969b-1b8a23afdd38 |
+----------+--------------------------------------+
root@control-server01:~#
```

The "cinder list" command should now show the volume's status as "in-use" and it should show the UUID of the instance in the "Attached to" field:

```
root@control-server01:~# cinder list
+--------------------------------------+--------+--------------+------+-------------+----------+--
|                  ID                  | Status | Display Name | Size | Volume Type | Bootable |
+--------------------------------------+--------+--------------+------+-------------+----------+--
| b6d78eb1-e7d7-453d-969b-1b8a23afdd38 | in-use |    myvol     |  4   |     None    |  false   | 6
+--------------------------------------+--------+--------------+------+-------------+----------+--
root@control-server01:~#
```

You can now log in to the instance, partition the volume, and create a filesystem on the volume. First we'll need to SSH into the instance. We know it's IP address from the "nova list" output above. Use the "quantum router-list" command to find out the namespace we'll need to use when calling SSH.

```
root@control-server01:~# quantum router-list
+--------------------------------------+---------+-----------------------------------------------
| id                                   | name    | external_gateway_info
+--------------------------------------+---------+-----------------------------------------------
| fc39be0f-b963-45ba-9da8-9874d2924d08 | router1 | {"network_id": "e0692be1-af70-478f-9dc9-c550f8a
+--------------------------------------+---------+-----------------------------------------------
root@control-server01:~# ip netns exec qrouter-fc39be0f-b963-45ba-9da8-9874d2924d08 ssh -i ~/.ssh/
The authenticity of host '10.10.10.4 (10.10.10.4)' can't be established.
RSA key fingerprint is 70:ba:13:5e:cf:15:92:8f:30:dd:7d:aa:ac:fa:9f:48.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.10.4' (RSA) to the list of known hosts.
$
```

Note from the output of "nova volume-attach" above that the volume was attached as device "/dev/vdb". We can treat that as an ordinary hard drive by partitioning it, creating a filesystem on it, and mounting it:

```
$ sudo su -
# fdisk /dev/vdb
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0x07505fb0.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1):
Using default value 1
First sector (2048-8388607, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-8388607, default 8388607):
Using default value 8388607
```

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
# mkfs.ext4 /dev/vdb1
mke2fs 1.42.2 (27-Mar-2012)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
262144 inodes, 1048320 blocks
52416 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=1073741824
32 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

# mkdir /tmp/mount
# mount /dev/vdb1 /tmp/mount
# df -h
Filesystem                Size      Used Available Use% Mounted on
/dev                    242.4M         0    242.4M   0% /dev
/dev/vda1                23.2M     17.9M      4.1M  81% /
tmpfs                   245.9M         0    245.9M   0% /dev/shm
tmpfs                   200.0K     76.0K    124.0K  38% /run
/dev/vdb1                 3.9G     72.0M      3.7G   2% /tmp/mount
#
```

### Testing Glance

Download an image and add it to glance:

```
wget http://cloud-images.ubuntu.com/precise/current/precise-server-cloudimg-amd64-disk1.img
glance add name="precise-x86_64" is_public=true container_format=ovf disk_format=qcow2 < precise-s
wget http://download.cirros-cloud.net/0.3.1/cirros-0.3.1-x86_64-disk.img
glance add name="cirros-x86_64" is_public=true disk_format=qcow2 container_format=ovf < cirros-0.3
```

Check that the image is known to glance and to Ceph:

```
root@control-server01:~# glance image-list
+--------------------------------------+----------------+-------------+------------------+--------
| ID                                   | Name           | Disk Format | Container Format | Size
+--------------------------------------+----------------+-------------+------------------+--------
| 02315102-8422-44bb-84d2-a7e28029e152 | cirros-x86_64  | qcow2       | ovf              | 1314764
| bee406a9-1570-4ed6-b0d4-d1394297ae58 | precise-x86_64 | qcow2       | ovf              | 2531000
+--------------------------------------+----------------+-------------+------------------+--------
root@control-server01:~# rbd --pool images ls
02315102-8422-44bb-84d2-a7e28029e152
bee406a9-1570-4ed6-b0d4-d1394297ae58
root@control-server01:~#
```

As with Cinder, you should see a matching UUID in the glance volume-list and rbd commands. This is your image stored in Ceph.