Linux Academy

Study Guide

Linux Academy Red Hat Certified Engineer in Red Hat OpenStack

Contents

Introduction

RED HAT
OPENSTACK
PLATFORM

About This Course

This course has been created to help you prepare for the EX310K Red Hat Certified Engineer in Red Hat OpenStack exam.

Goals of This Course

- Learn concepts needed to be an Engineer of Red Hat OpenStack environments
- Install and configure the RHEL OpenStack Platform (v10.0)
- Perform common administrative tasks such as:
    - Manage projects, users, roles, and flavors
    - Set quotas
    - Manage images at instantiation
- Install and configure Red Hat Ceph storage in a multi-node environment
- Configure Ceph block devices
- Create snapshots of Ceph block devices
- Integrate Ceph block devices with OpenStack services such as Glance, Cinder, and Nova
- Create and manage virtual network devices
- Provision tenant networks with DHCP agents
- Manage network ports and VLANs
- Create instances and attach them to specific VLANs
- Create and manage floating IP addresses
- Configure Load Balancing as a Service (LBaaS)
- Troubleshoot virtual network devices

About the Red Hat EX310K Exam

The performance-based Red Hat Certified Engineer in Red Hat OpenStack exam tests the candidate's skill, knowledge, and ability to create, configure, and manage private cloud infrastructure using Red Hat Enterprise OpenStack Platform.

Requirements

- To earn the RHCE in Red Hat OpenStack certification, you must have a current RHCSA in Red Hat OpenStack certification.
- Red Hat recommends that candidates first earn the Red Hat Certified Systems Administrator (RHCSA - EX200) before attempting the RHCE in Red Hat OpenStack, but it is not required.
- Tests are taken in a Red Hat authorized on-site testing center.
- **Note:** Only certain locations are able to handle the EX310 individual exam at this time. As of creation of this course, the following locations were verified by Red Hat as equipped to deliver the EX310K:
    - Atlanta, Georgia, United States
    - Boston, Massachusetts, United States
    - Charlotte, North Carolina, United States
    - Charleston, South Carolina, United States
    - Columbia, Maryland, United States
    - Culver City, California, United States
    - Denver, Colorado, United States
    - Iselin, New Jersey, United States
    - Orlando, Florida, United States
    - Reston, Virigina, United States
    - San Antonio, Texas, United States
    - Austin, Texas, United States
    - San Jose, California, United States
    - Canberra, Australia
    - Lisboa, Portugal
    - Paris, France
    - Praha, Czech Republic

- Stockholm, Sweden
- Vienna, Austria

Course Prerequisites

Before starting this course, it is recommended that students have current Red Hat Certified Systems Administrator (EX200) and Red Hat Certified Systems Administrator in Red Hat OpenStack (EX210) certifications or equivalent experience.

Students should have an intermediate grasp of the following topics before beginning this course.

- Red Hat Enterprise Linux systems administration
  - RHEL Satellite
  - systemd
  - Package management (`dnf`/`yum`/`rpm`)
  - User management (creation, `sudo`)
  - Directory structure, creation, navigation
- OpenStack administration
  - CLI
  - Keystone

```
+ User management

+ Project management
```

  - Nova
  - Glance

```
+ Image management (types, creation, deletion)
```

  - Neutron

```
+ Networks and network types

+ VIPs (explained in course)
```

  - Swift

    - Cinder
- Networking
    - Virtual networks (namespaces, bridges, etc.)
    - VLANs
    - Routing
    - IP addresses (CIDRs)
    - DHCP

Recommended Courses

The following Linux Academy courses are recommended, but not required, before starting this course:

- OpenStack Essentials
- Linux KVM Virtualization Essentials
- OpenStack Foundation Certified OpenStack Administrator
- Linux Academy Red Hat Certified Systems Administrator prep course
- Linux Academy Red Hat OpenStack Administrator Certification Prep Course

Ceph

Introduction to Ceph

Initially created in 2007 by Sage Weil for his doctoral dissertation, with sponsorships from the US Department of Energy, Oak Ridge National Library, Intel, Microsoft, and others. If you're interested, that dissertation can be found in PDF format online.

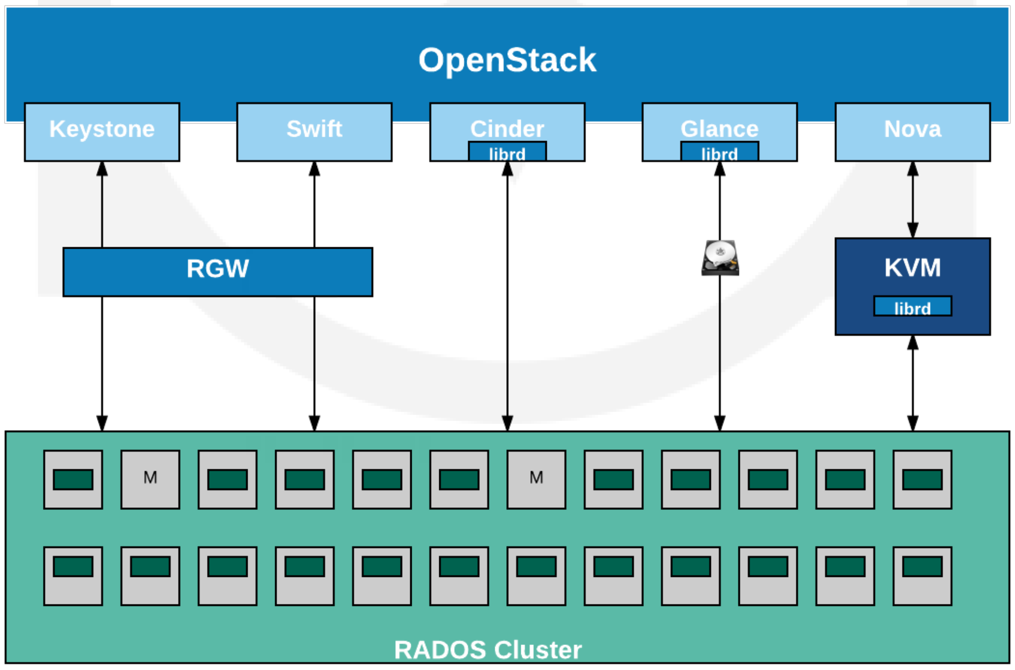
After graduating in 2007, Sage continued development and support of Ceph full time with Yehuda Weinraub and Gregory Farnum as the core development team. In 2012, Weil created Inktank Storage for professional services and support for Ceph.

Ceph is a self-healing and self-managing open source system that replicates data and makes it fault tolerant using commodity hardware and requiring no specific hardware support at the object, file, and block level under a unified system.

Merged into the Linux kernel by Linus Torvalds on March 19, 2010

What is Ceph?

Ceph is an open source software designed to provide highly scalable object, block, and file-based storage



under a unified system.

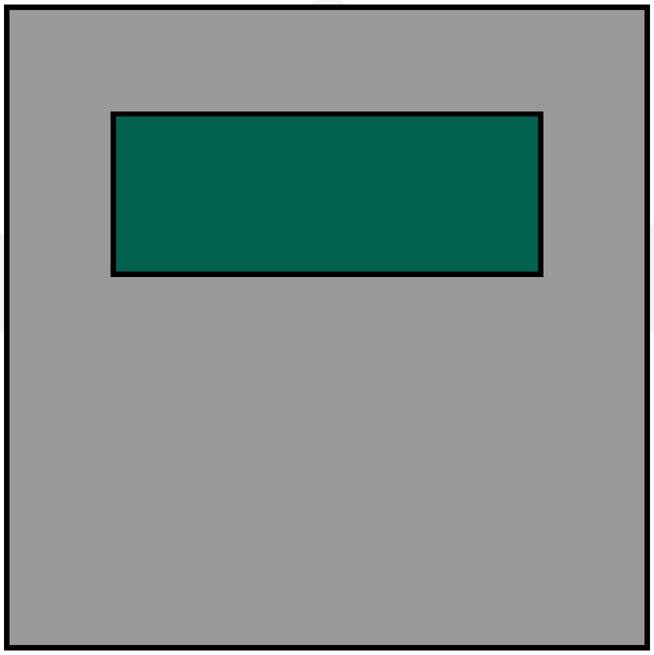
Architectural Overview

Requirements

- All Ceph Storage Cluster deployments begin with setting up each Ceph Node, your network, and the Ceph Storage Cluster.
- A Ceph Storage Cluster requires at least one Ceph Monitor and at least two Ceph OSD Daemons. The Ceph Metadata Server is essential when running Ceph Filesystem clients.

Ceph Components

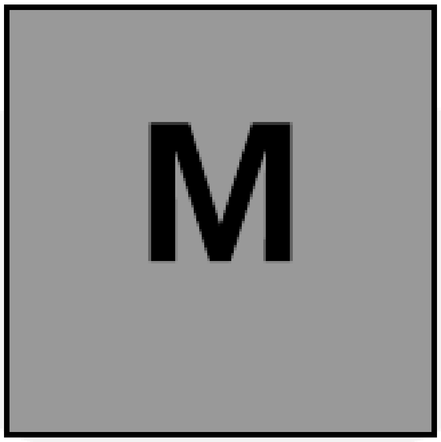
- **Ceph OSDs:** A Ceph OSD Daemon (Ceph OSD) stores data;handles data replication, recovery, backfilling, and rebalancing; and provides some monitoring information to Ceph Monitors by checking other Ceph OSD Daemons for a heartbeat. A Ceph Storage Cluster requires at least two Ceph OSD Daemons to achieve an active and clean state when the cluster makes two copies of your data (Ceph makes three copies by default, but you can adjust it).
- **Monitors:** A Ceph Monitor maintains maps of the cluster state, including the monitor map, the OSD map, the Placement Group (PG) map, and the CRUSH map. Ceph maintains a history (called an "epoch") of each state change in the Ceph Monitors, Ceph OSD Daemons, and PGs.
- **MDSs:** A Ceph Metadata Server (MDS) stores metadata on behalf of the Ceph Filesystem (i.e., Ceph Block Devices and Ceph Object Storage do not use MDS). Ceph Metadata Servers make it



feasible for POSIX file system users to execute basic commands like `ls`, `find`, etc. without placing an enormous burden on the Ceph Storage Cluster.

OSDs

- 10s to 10000s in a cluster
- One per disk
    - Or one per SSD, RAID group, etc.
- Serve stored objects to clients



- Intelligently peer to perform replication and recovery tasks

Monitors

- Maintain cluster membership and state
- Provide consensus for distributed decision-making
- Small, odd number
- These do **NOT** serve stored objects to clients

Keyrings

When `cephx` is used, authentication is handled with keyrings generated by Ceph for each service.

- **ceph.mon.keyring** - Used by all Mon nodes to communicate with other Mon nodes
- **ceph.client.admin.keyring** - This keyring is what Ceph client commands, by default, to administer the Ceph cluster.
- **ceph.bootstrap-osd.keyring** - Used to generate `cephx` keyrings for OSD instances
- **ceph.bootstrap-rgw.keyring** - Used to generate `cephx` keyrings for RGW instances

Block, File, and Object Storage

- Ceph is a scalable, high performance distributed file system offering performance, reliability, and scalability.
    - Aims for a completely distributed operation without a single point of failure, is scalable to the exabyte-level, and is freely available
    - Ceph code is completely open source, released under the LPGL license
- Block storage - Physical storage media appears to computers as a series of sequential blocks of a uniform size (elder storage). Ceph object storage allows you to mount Ceph as a thin-provisioned block device. Writes to Ceph using a block device are automatically be striped and replicated across the cluster.
- File storage - File systems allow users to organize data stored in blocks using hierarchical folders and files. The Ceph file system runs on top of the same object storage system that provides object storage and block device interfaces.
- Object storage - Object stores distribute data algorithmically throughout a cluster of media without a rigid structure (NKOTB).

Ceph Concepts

- **Monitors** - Otherwise known as mons, maintain maps (CRUSH, PG, OSD, etc.) and cluster state. Monitors use Paxos to establish consensus.
- **Storage or OSD node** - Provides one or more OSDs. Each OSD represents a disk and has a running daemon process controlled by `systemctl`. Ceph has two disk types:
    - **data**
    - **journal** - Enables Ceph to commit small writes quickly and guarantees atomic compound operations.
- **Calamari (optional)** - Runs on one of the monitors to get statistics on Ceph cluster and provides a REST endpoint. RHSC talks to calamari.
- **Clients** - Each client requires authentication if `cephx` is enabled. `cephx` is based on Kerberos.
    - RBD client for block storage
    - CephFS for file storage
    - Fuse client
    - RADOS GW
- **Gateways (optional)** - Ceph is based on RADOS. The RADOS gateway is a web server that provides S3 and Swift endpoints and sends those requests to Ceph via RADOS. Similarly, there is an iSCSI gateway that provides iSCSI target to clients and talks to Ceph via RADOS.

CRUSH

Ceph Storage clusters are designed to run on commodity hardware, using the CRUSH (**C**ontrolled **R**eplication **U**nder **S**calable **H**ashing) algorithm to ensure even distribution of data across the cluster, and

$$\frac{(\text{ Target PGs per OSD }) \text{ x } (\text{ OSD \# }) \text{ x } (\text{ \%Data })}{(\text{ Size })}$$

that all cluster nodes can retrieve data quickly without any centralized bottlenecks.

CRUSH Algorithm

- Ceph stores a client's data as objects within storage pools. Using the CRUSH algorithm, Ceph calculates which placement group should contain the object and further calculates which Ceph OSD Daemon should store the placement group. The CRUSH algorithm enables the Ceph Storage Cluster to scale, rebalance, and recover dynamically.
- CRUSH **(Controlled Replication Under Scalable Hashing)** is a hash-based algorithm for calculating how and where to store and retrieve data in a distributed object–based storage cluster designed specifically for Ceph.
- CRUSH distributes data evenly across available object storage devices in what is often described as a pseudo-random manner.
- Distribution is controlled by a hierarchical cluster map called a CRUSH map. The map, which can be customized by the storage administrator, informs the cluster about the layout and capacity of nodes in the storage network and specifies how redundancy should be managed. By allowing cluster nodes to calculate where a data item has been stored, CRUSH avoids the need to look up data locations in a central directory. CRUSH also allows for nodes to be added or removed, moving as few objects as possible while still maintaining balance across the new cluster configuration.
- CRUSH was designed for Ceph, an open source software designed to provide object-, block- and file-based storage under a unified system. Because CRUSH allows clients to communicate directly with storage devices without the need for a central index server to manage data object locations, Ceph clusters can store and retrieve data very quickly and scale up or down quite easily.

CRUSH Maps

Source: Ceph Docs

- The CRUSH algorithm determines how to store and retrieve data by computing data storage locations. CRUSH empowers Ceph clients to communicate with OSDs directly rather than through a centralized server or broker.
- With an algorithmically determined method of storing and retrieving data, Ceph avoids a single point of failure, a performance bottleneck, and a physical limit to its scalability.

- CRUSH requires a map of your cluster, and uses the CRUSH map to pseudo-randomly store and retrieve data in `Object Storage Devices` with a uniform distribution of data across the cluster.

- CRUSH maps contain a list of OSDs, a list of 'buckets' for aggregating the devices into physical locations, and a list of rules that tell CRUSH how it should replicate data in a Ceph cluster's pools.

- By reflecting the underlying physical organization of the installation, CRUSH can model—and thereby address—potential sources of correlated device failures.

- Typical sources include physical proximity, a shared power source, and a shared network. By encoding this information into the cluster map, CRUSH placement policies can separate object replicas across different failure domains while still maintaining the desired distribution. For example, to address the possibility of concurrent failures, it may be desirable to ensure that data replicas are on devices using different shelves, racks, power supplies, controllers, and/or physical locations.

- When you create a configuration file and deploy Ceph with `ceph-deploy`, Ceph generates a default CRUSH map for your configuration. The default CRUSH map is fine for your Ceph sandbox environment; however, when you deploy a large-scale data cluster, you should give significant consideration to developing a custom CRUSH map, because it will help you manage your Ceph cluster, improve performance, and ensure data safety.

CRUSH Location

- The location of an OSD in terms of the CRUSH map's hierarchy is referred to as a 'crush location'. This location specifier takes the form of a list of key and value pairs describing a position.

- For example, if an OSD is in a particular row, rack, chassis and host and is part of the 'default' CRUSH tree, its crush location could be described as `root=default row=a rack=a2 chassis=a2a host=a2a1`

- The key name (left of `=`) must be a valid CRUSH type.

Default CRUSH Types

- root
- datacenter
- room
- row
- pod
- pdu
- rack
- chassis
- host

Default CRUSH types can be customized to be anything appropriate by modifying the CRUSH map.

Not all keys need to be specified. For example, by default, Ceph automatically sets a Ceph OSD Daemon's location to be `root=default host=HOSTNAME` (based on the output from `hostname -s`).

CRUSH Rules

Example policy:

```
rule flat   {
    ruleset 0
    type replicated
    min_size 1
    max_size 10
    step take root
    step choose firstn 0 type osd
    step emit
}
```

In the above example, `firstn` declares how to do a replacement. `type` has been specified to be `osd`. `firstn` is set to `0`, which allows access to however many the caller needs. The next example sets rules by host:

```
rule by-host    {
    ruleset 0
    type replicated
    min_size 1
    max_size 10
    step take root
    step choose firstn 0 type host
    step choose firstn 1 type osd
    step emit
}
```

This example chooses from a selection of hosts, which is set to `firstn 0`. We learned in the first example this means that the rule is able to choose however many the caller needs. The rule specifies that it is to choose `1 osd` for each host with `firstn 1 type osd`.

Cluster Expansion

- Stable mapping
  - Expansion by 2x = half of all objects will move
  - Expansion by 5% = ~5% of all objects will move
- Elastic placement
  - Expansion, failure, contraction - it's all the same

- CRUSH *always* rebalances on cluster expansion or contraction
  - Balanced placement --> balance load --> best performance
  - Rebalancing at scale is cheap

Weighted Devices

- OSDs may be different sizes
  - Different capacities
  - HDD or SSD
  - Clusters expand over time
  - Available devices changing constantly
- OSDs get PGs (and thus objects) proportional to their weight
- Standard practice
  - Weight = size in TB

Data Imbalance

- CRUSH placement is pseudo-random
  - Behaves like a random process
  - "Uniform distribution" in the statistical sense of the word
- Utilizations follow a **normal** distribution
  - More PGs --> tighter distribution
  - Bigger cluster --> more outliers
  - High outlier --> overfull OSD

Reweighting

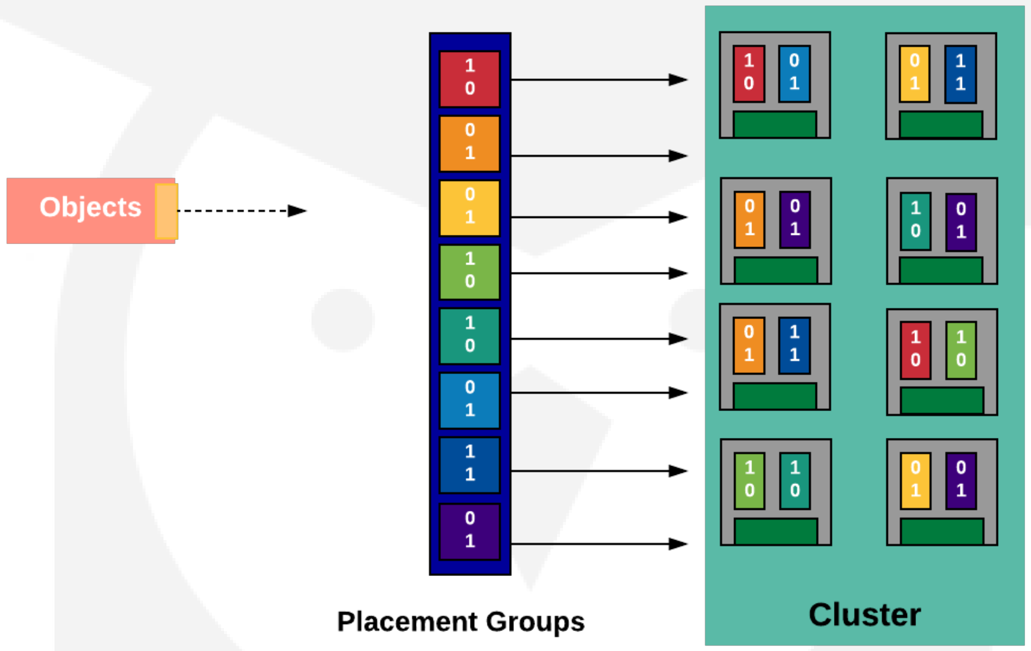
- OSDs get data proportional to their weight – unless it fails
  - No data received
  - CRUSH will attempt an internal "retry" if it encounters a failed device
- Reweight treats failure as non-binary
  - 1 = device is OK
  - 0 = always reject this device

- .9 = reject it 10% of the time

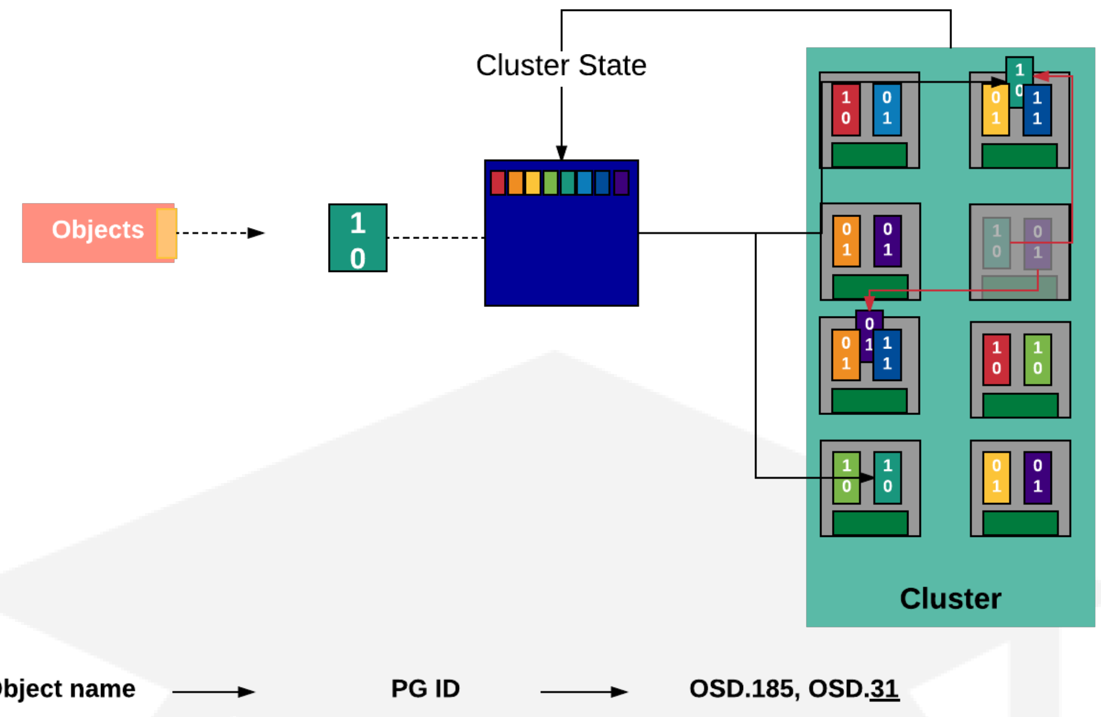
Reweight-by-Utilization

- Find OSDs with highest utilization
  - Reweight proportional to their distance from average
- Find OSDs with lowest utilization
  - If previously reweighted down, reweight them back up
- Run periodically, automatically
- Make small, regular adjustments to data balance

How It Works



- Object is given a name based on PG and replicated at least 3 times in binary matching PG for chosen placement group
- Placement is a function
- Hash (object name) PG ID --> CRUSH (PG ID, Cluster topology) --> OSD.165, OSD.67
- Replicas for each device are spread around
- Failure repair is distributed and parallelized
- Recovery does not bottleneck on a single device



- In the case of a device failure, Ceph avoids the failed device and rebalances itself, moving the replicated data to another, functional OSD
- Crush generates *n* distinct target devices, or OSDs. These may be replicas or erasure coding shards.
- Set replicas across domains
- A single failure should only compromise one replica
- Size of failure domain depends on cluster size (disk), host (NIC, RAM, PS), rack (ToR switch, PDU), row (distribution switch)
- Based on types in CRUSH hierarchy

Ceph Placement Group Calculators

- Ceph PGCalc
    - http://ceph.com/pgcalc/
- Red Hat placement group calculator
    - https://access.redhat.com/labs/cephpgc/

Ceph Troubleshooting

Health Monitoring

It is best practice to check the health of your cluster before performing any reads or writes of data. This can be done with the command `ceph health`.

Watching a Cluster

You can watch ongoing events with the command `ceph -w`, which prints each event and contains the following information:

- Cluster ID
- Cluster health status
- The monitor map epoch and the status of the monitor quorum
- The OSD map epoch and the status of OSDs
- The placement group map version
- The number of placement groups and pools
- The notional amount of data stored and the number of objects stored
- The total amount of data stored

Checking Cluster Usage Stats

The used value reflects the actual amount of raw storage used. The `[num]GB / [num] GB` value means the amount available (the lesser number) of the overall storage capacity of the cluster. The notional number reflects the size of the stored data before it is replicated, cloned or snapshot; therefore, the amount of data actually stored typically exceeds the notional amount stored, because Ceph creates replicas of the data and may also use storage capacity for cloning and snapshotting.

Handy Commands - CRUSH

View OSD tree details:

```
sudo ceph osd tree
```

Create basic rules:

```
ceph osd crush rule create-simple by-rack default rack
```

Adjust weights:

```
ceph osd crush reweight osd.7 4.0
ceph osd crush add-bucket b rack
ceph osd crush move b root=default
ceph osd crush move ceph1 rack=b
```

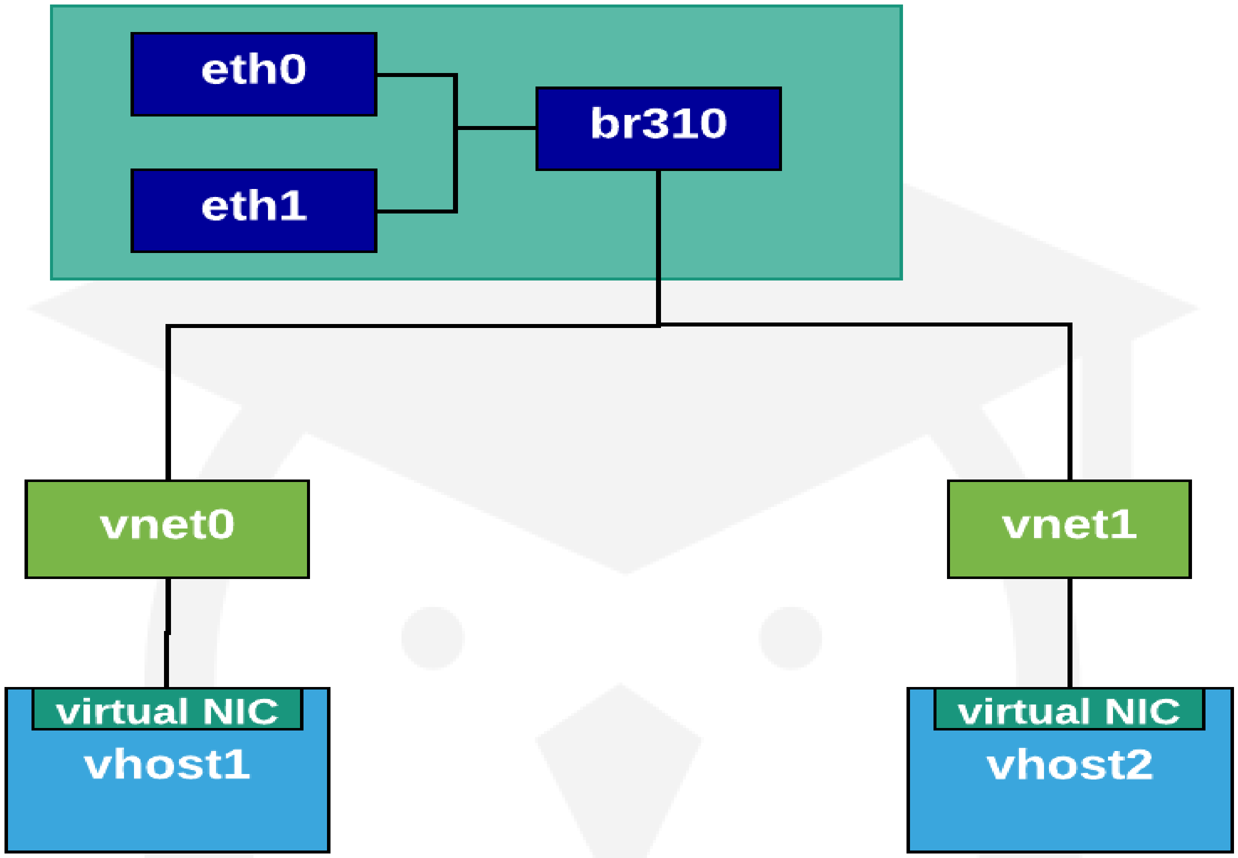
Glossary

- **iSCSI** - An extension of the standard SCSI storage interface that allows SCSI commands to be sent over an IP-based network.
- **Linux Bridge** - A way to connect two Ethernet segments together in a protocol independent way.
- **Ceph Platform** - All Ceph software, which includes any piece of code hosted in the Ceph GitHub repository
    - **Ceph Node** - Any single machine or server in a Ceph system
    - **Ceph Cluster Map** - The set of maps comprising of the monitor map, OSD map, PG map, MDS map, and CRUSH map.
    - **Cephx** - Ceph authentication protocol. Cephx operates like Kerberos but has no single point of failure
    - **Ceph OSD Daemons** - The Ceph OSD software which interacts with a logical disk (OSD). Sometimes "OSD" is used to refer to the "Ceph OSD Daemon", but the proper term is "Ceph OSD".
    - **Object Storage Device (OSD)** - A physical or logical storage unit. Sometimes confused with Ceph OSD Daemon.
    - **Ceph Monitor (MON)** - Ceph monitor software
    - **Cloud Platforms [Stacks]** - Third-party cloud provisioning platforms such as OpenStack, CloudStack, OpenNebula, ProxMox, etc.
- **Cluster** - A set of monitors
- **Quorum** - An active set of monitors consisting of a majority of the cluster
- **RADOS (Reliable Atomic Distributed Object Store) Cluster** - The core set of storage software which stores the user's data (Mon+OSD)
    - **Rados Block Device (RBD)** - The block storage component of Ceph
    - **RADOS Gateway (RGW)** - The S3/Swift gateway component of Ceph
- **Controlled Replication Under Scalable Hashing (CRUSH)** - The algorithm Ceph uses to compute object storage locations.
    - **Ruleset** - A set of CRUSH data placement rules that applies to a particular pool(s).
    - **Pool** - Logical partitions for storing objects.

Lab Setup

Linux Bridges

Bridging is the process of transparently connecting two networks segments together, so that packets can



pass between the two as if they were a single logical network. Bridging is performed on the data link layer; hence, it is independent of the network protocol being used as the bridge operates upon the raw ethernet packets. Typically, in a non-bridged situation, a computer with two network cards would be connected to a separate network on each; while the computer itself may or may not route packets between the two, in the IP realm, each network interface would have a different address and different network number. When bridging is used, however, each network segment is effectively part of the same logical network, the two network cards are logically merged into a single bridge device and devices connected to both network segments are assigned addresses from the same network address range.

KVM Virtual Network Setup

Open Virtual Machine Manager

```
sudo virt-manager
```

Create a new virtual network under **Menu** --> **Edit** --> **Connection Details**.

Enter the following settings:

Step 1

- **Network Name**: shadowman

Step 2

- Enable IPv4 network address space definition
- **Network**: 192.168.10.0/24
- Enable DHCPv4
- **Start**: AUTOMATIC
- **End**: AUTOMATIC

Step 3

- Skip this step. We will not be using IPv6.

Step 4

- Forwarding to a **physical network**:
- **Destination**: eth0 (or whatever your main network is named)
- **Mode**: NAT
- **DNS Domain Name**: shadowman (leave as default)

Create a New Virtual Machine

Download Red Hat Enterprise Linux 7.3 (or higher) minimal DVD image before beginning.

In the **Virtual Machine Manager** menu, go to **File** --> **New virtual machine**.

Step 1

- Local install media (ISO image or CDROM)

Step 2

- Use ISO image: Browse to your hard drive to select the downloaded RHEL7.2 ISO

Step 3

- **Memory (RAM)**: 4096 MiB *minimum*
- **CPUs**: 1

Step 4

- Select or create custom storage, then click **Manage**
- Choose storage pool, then click the blue **+** by **Volumes** to create a new volume
    - **Name**: [compute1 | controller | network | ceph{1,2,3}]
    - **Format**: qcow2
    - **Max Capacity**: Controller, Compute: 40GiB, Network: 20GiB, Ceph{1,2,3}: 238GiB (30GB root disk, 100x2 ephermal disk, 8GB *optional* swap)

Step 5

- **Name**: [compute1 | controller | network | ceph{1,2,3}]
- **Network Selection**: Virtual network 'shadowman': NAT to eth0
- Hit **Done** to begin OS installation

OS Installation

- Select regional settings, then continue
- Localization, set date and time to your local settings
- Software:
    - **Installation Source**: Local media (auto-selected)
    - **Software selection**: Minimal install (auto-selected)
- System
    - **Installation Destination**: Automatic partitioning
    - **Network and host name**:
        - **Host name**: [controller | compute1 | network | ceph{1,2,3}].$FQDN
        - **Ethernet**: Set to ON to allow DHCP to assign an IP
        - **Configure**:
            - **Method**: Manual

        - **Addresses**:
            - Enter IP assigned by DHCP
            - **Netmask**: 24 (probably)
            - **Gateway**: enter gateway provided by DHCP
            - **Additional DNS servers**: Use gateway provided by DHCP
- Begin installation
    - While install runs, set root password and create a user with admin privileges
    - After install completes, select the **Reboot** option, then log in through a terminal

OpenStack Controller and Network Node Configuration

Attach subscription:

```
subscription-manager register
```

List available subscription pools:

```
subscription-manager list --available | grep -A29 "Red Hat OpenStack
Platform"
```

Attach subscription pools for RHEL7-openstack:

```
subscription manager attach --pool=$POOL-ID
```

Disable all repositories:

```
subscription-manager repos --disable=*
```

Re-enable RHEL7, RHEL-optional, and RHELOSP10 repositories:

```
subscription-manager repos --enable=rhel-7-server-rpms \
--enable=rhel-7-server-optional-rpms \
--enable=rhel-7-server-rh-common-rpms \
--enable=rhel-7-server-extras-rpms \
--enable=rhel-7-server-openstack-10-rpms \
--enable=rhel-7-server-openstack-10-devtools-rpms
```

Install `yum-utils` package:

```
yum -y install yum-utils
```

Update system:

```
yum -y update
```

Disable `NetworkManager`, `firewalld`, and `selinux`:

```
systemctl disable NetworkManager firewalld
sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config
```

Reboot to set changes:

```
systemctl reboot
```

Verify SELinux is disabled:

```
sestatus  # should return disabled
```

Controller

Install `openstack-packstack`:

```
yum -y install openstack-packstack
```

Generate an answer file:

```
packstack -gen-answer-file=/root/answers.txt
```

Edit the following values in `answers.txt`:

```
CONFIG_DEFAULT_PASSWORD=openstack
CONFIG_NETWORK_HOST=$NETWORK_IP
CONFIG_KEYSTONE_ADMIN_TOKEN=linuxacademy123
CONFIG_KEYSTONE_ADMIN_PW=openstack
CONFIG_KEYSTONE_DEMO_PW=openstack
CONFIG_PROVISION_DEMO=n
CONFIG_NEUTRON_ML2_TYPE_DRIVERS=vxlan,flat
CONFIG_LBAAS_INSTALL=y
```

OPTIONAL: Install heat:

```
CONFIG_HEAT_INSTALL=y
```

```
CONFIG_HEAT_CLOUDWATCH_INSTALL=y
CONFIG_HEAT_CFN_INSTALL=y
```

Populate `/etc/hosts`:

```
$IP controller  controller.$FQDN
$IP network network.$FQDN
```

Network Node

Attach subscription:

```
subscription-manager register
```

List available subscription pools:

```
subscription-manager list --available | grep -A29 "Red Hat OpenStack
Platform"
```

Attach subscription pools for RHEL7-openstack:

```
subscription manager attach --pool=$POOL-ID
```

Disable all repositories:

```
subscription-manager repos --disable=*
```

Re-enable RHEL7, RHEL-optional, and RHELOSP10 repos:

```
subscription-manager repos --enable=rhel-7-server-rpms \
--enable=rhel-7-server-optional-rpms \
--enable=rhel-7-server-rh-common-rpms \
--enable=rhel-7-server-extras-rpms \
--enable=rhel-7-server-openstack-10-rpms \
--enable=rhel-7-server-openstack-10-devtools-rpms
```

Install `yum-utils` package:

```
yum -y install yum-utils
```

Update system:

```
yum -y update
```

Disable `NetworkManager`, `firewalld`, and `selinux`:

```
systemctl disable NetworkManager firewalld
sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config
```

Reboot to set changes:

```
systemctl reboot
```

Verify SELinux is disabled:

```
sestatus  # should return disabled
```

Populate `/etc/hosts`:

```
$IP controller  controller.$FQDN
$IP network network.$FQDN
```

Controller Node

Kick off installation:

```
packstack -answer-file=/root/answers.txt
```

NOTE: PackStack installations can take upwards of 30 minutes to complete depending on system resources.

Ceph Lab Environment Components

You need at least three virtual server nodes to follow along with this course. Each node should have a 20GB root disk and 100GB data disk.

- **Admin console** - This is the node that hosts the UI and CLI used for managing the Ceph cluster.
- **Monitors** - Monitor health of the Ceph cluster. One or more monitors forms a Paxos Part-Time Parliament, providing extreme reliability and durability of cluster membership. Monitors maintain monitor, OSD, placement group (PG), and CRUSH maps and are installed on all nodes.
- **OSD** - The object storage daemon handles storing data, recovery, backfiling, rebalanicng, and replication. OSDs sit on top of a disk or filesystem. Bluestore enables OSDs to bypass the filesystem but is not an option in Ceph 1.3. Installed on all nodes.

Create and Attach Data Disks

Data disks need to be attached using the virsh command line interface. You can use either `dd` or `fallocate` to generate an empty 100GB file.

NOTE: Verify that SELinux permissions are correct on the folder storing the disk files.

Create three empty 100GB IMG files:

```
for i in {1,2,3};
    do
    fallocate -l 100G /var/lib/libvirt/qemu/disk$i.img;
done
```

Create an XML file for each disk:

```
<disk type='file' device='disk'> <driver name='qemu' type='raw'
cache='none'/> <source file='/var/lib/libvirt/images/disk$i.img'/>
<target dev='sdb'/> </disk>
```

Set *qemu* as the owner of the IMG files:

```
chown -R qemu:qemu disk*
```

Attach one volume to each node using virsh CLI:

```
for i in {1,2,3};
    do
    virsh attach-device --config ceph$i ~/disk$i.xml;
done
```

Reboot each instance to complete mount of data disk:

```
systemctl reboot
```

Ceph Node Configuration

On All Ceph Nodes

Populate hosts file on each node:

```
$CONTROLLER_IP  $IP     controller.$FQDN
$COMPUTE_IP     $IP     compute1.$FQDN
$NETWORK_IP     $IP     network.$FQDN

# ceph nodes
$IP1        ceph1   ceph1.$FQDN
$IP2        ceph2   ceph2.$FQDN
$IP3        ceph3   ceph3.$FQDN
```

Configure Red Hat repositories:

- List available subscription pools:

```
    subscription-manager list --available
```

- Locate and attach subscription pool for RHEL7-OPenStack:

```
    subscription-manager attach --pool=$(subscription-manager list
--available | grep -A28 "Red Hat OpenStack Platform" | grep -i "pool id"
| awk '{print $3}'
    subscription manager attach --pool=$POOL-ID
```

- Disable all repos:

```
    subscription-manager repos --disable=*
```

- Enable Red Hat Enterprise Linux 7 server repos:

```
    subscription-manager repos --enable=rhel-7-server-rpms
--enable=rhel-7-server-extras-rpms
```

firewalld Configuration

Set `firewalld` to start at boot:

```
systemctl enable firewalld.service
```

Start `firewalld` (if not already started):

```
systemctl start firewalld.service
```

Permanently open ports 80, 2003, 4505, 4506, 6789, and 6800-7300:

```
#!/bin/bash
#
# easy firewalld script for OpenStack Ceph to accompany the
# Linux Academy Red Hat Certified Engineer in Red Hat OpenStack prep
course
# Opens ports 80, 2003, 4505, 4506, 6789, 6800-7300
# June 13, 2017
# This script comes with no guarantees.
#
systemctl enable firewalld && systemctl start firewalld
sleep 3
firewall-cmd --zone=public --add-port=80/tcp --permanent
firewall-cmd --zone=public --add-port=2003/tcp --permanent
```

```
firewall-cmd --zone=public --add-port=4505-4506/tcp --permanent
firewall-cmd --zone=public --add-port=6789/tcp --permanent
firewall-cmd --zone=public --add-port=6800-7300/tcp --permanent
sleep 3
firewall-cmd --reload
```

NTP Configuration

Install NTP:

```
yum -y install ntp
```

Set NTP to start at boot:

```
systemctl enable ntpd.service
```

Start NTP:

```
systemctl start ntpd.service
```

Verify synchronization of NTP service (on all nodes):

```
ntpq -p
```

Create *ceph* user for deployer:

```
useradd ceph
echo openstack | passwd --stdin ceph
```

Grant *ceph* user sudo privileges without password:

```
cat /etc/sudoers.d/ceph
ceph ALL = (root) NOPASSWD:ALL
Defaults:ceph !requiretty
EOF
```

Secure `sudoers.d` file for *ceph* user:

```
chmod 440 /etc/sudoers.d/ceph
```

Drop to *ceph* user:

```
su - ceph
```

On each node, create an ssh key for *ceph* user:

```
ssh-keygen
```

Load ssh keys for Ceph1, Ceph2, and Ceph3 on each node:

```
for i in {1,2,3};
    do
ssh-copy-id ceph@ceph$1;
    done
```

Modify your `~/.ssh/config` file of your admin node so that it defaults to logging in as the *ceph* user across your Ceph environment when no username is specified:

```
Host ceph1
        Hostname ceph-server.fqdn-or-ip-address.com
        User ceph
Host ceph2
        Hostname ceph-server.fqdn-or-ip-address.com
        User ceph
Host ceph3
        Hostname ceph-server.fqdn-or-ip-address.com
        User ceph
```

Set SELinux to disabled:

```
sudo sed -i "s/SELINUX=enforcing/SELINUX=disabled/g" /etc/selinux/config
```

Create `ceph-config` directory:

```
mkdir ~/ceph-config
cd ceph-config
```

Monitor Nodes (Ceph2)

Update system:

```
sudo yum -y update
```

Install the Extra Packages for Enterprise Linux (EPEL) repository:

```
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.
rpm
```

```
rpm -ivh epel-release-latest-7.noarch.rpm
```

Enable the RHEL7 Extras repository:

```
subscription-manager repos --enable=rhel-7-server-extras-rpms
```

OSD Nodes (Ceph3)

Update system:

```
sudo yum -y update
```

Install the Extra Packages for Enterprise Linux (EPEL) repository:

```
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.
rpm
rpm -ivh epel-release-latest-7.noarch.rpm
```

Enable the RHEL7 Extras repository:

```
subscription-manager repos --enable=rhel-7-server-extras-rpms
```

Install Ceph with ceph-ansible

On Ceph1 Admin Node

Install Ansible and git:

```
yum -y install ansible git
```

Declare IPs for mon, rgw, and osd nodes in `/etc/ansible/hosts`:

```
[mons]
$IP or hostname
[osds]
$ip or hostname
$IP or hostname
$IP or hostname
[rgws]
$IP or hostname
```

Copy SSH keys for new user to all nodes:

```
ssh-copy-id ceph@ceph1
ssh-copy-id ceph@ceph2
ssh-copy-id ceph@ceph3
```

Verify connectivity to all nodes:

```
ansible all -m ping
```

Clone Ceph repo:

```
git clone https://github.com/ceph/ceph-ansible/
```

Move into `group_vars` directory:

```
cd ~/ceph-ansible/group_vars
```

Copy sample config files for all, mon, and osds:

```
cp all.yml.sample all.yml
cp mons.yml.sample mons.yml
cp osds.yml.sample osds.yml
```

Add or uncomment the following values in `all.yml`:

```
...
ceph_origin: upstream
....
#   COMMUNITY VERSION
ceph_stable: true
ceph_mirror: http://download.ceph.com
```

ceph_stable_key: https://download.ceph.com/keys/release.asc

```
ceph_stable_release: kraken
ceph_stable_repo: https://download.ceph.com/rpm-kraken/el7/x86_64
...
ceph_stable_redhat_distro: el7
...
cephx_requires_signature: false
...
#   CEPH CONFIGURATION #
cephx: true
...
## Monitor Options
monitor_interface: $NETWORK
monitor_address: $SERVER_IP
```

```
ip_version: ipv4
...
## OSD options
journal_size: 10240
public_network: $SERVER_IP/24
```

Add or edit the following in `osds.yml`:

```
...
########################
# CEPH OPTIONS
########################
devices:
  - dev/vdb
...
journal_collocation: true
```

Move back to `/ceph-ansible/`:

```
cd ..
```

Start installation:

```
ansible-playbook site.yml
```

Post Installation

Cephx Keyrings

If Cephx is enabled, ceph-ansible is going to generate initial auth files for your user, which can be retrieved in a new directory created under the `/ceph-ansible/fetch/` directory. If gatherkeys complains about locating `ceph.keyring` files on ceph1, you can manually relocate the initial keyrings created by ceph-ansible from the `fetch` directory. Under `fetch`, there is a directory named as a UUID, and under this are `/etc/` and `/var/` directories. The `/var/` directory contains the missing keyrings. Just manually copy these to the `/etc/ceph/` directory.

Install `ceph-deploy`:

```
yum -y install ceph-deploy
```

Change to *ceph* user:

```
su - ceph
```

Move to `ceph-config`:

```
cd ceph-config
```

Collect initial keyring files:

```
sudo ceph-deploy gatherkeys ceph1 ceph2 ceph3
```

Integrate Ceph with OpenStack

Integrate Ceph with Glance (Image Service)

![glance-mascot](images/OpenStack_Project_Glance_mascot.png)

Install Ceph client used by Glance on controller node:

```
yum -y install python-rbd
```

Create Ceph RBD pool for Glance images on ceph1:

```
su - ceph
sudo ceph osd pool create images 64
```

Create the keyring that will allow Glance access to pool:

```
sudo ceph auth get-or-create client.images mon 'allow r' osd 'allow
class-read object_prefix rdb_children, allow rwx pool=images' -o /etc/
ceph/ceph.client.images.keyring
```

Copy the keyring to `/etc/ceph` on the OpenStack controller:

```
scp /etc/ceph/ceph.client.images.keyring root@controller:/etc/ceph
```

Copy the Ceph configuration file to the controller:

```
scp /etc/ceph/ceph.conf root@controller:/etc/ceph
```

Set permissions on controller so Glance can access Ceph keyring:

```
chgrp glance /etc/ceph/ceph.client.images.keyring
chmod 0640 /etc/ceph/ceph.client.images.keyring
```

Add keyring file to Ceph config:

```
vim /etc/ceph/ceph.conf
...
[client.images]
keyring = /etc/ceph/ceph.client.images.keyring
```

Back up original Glance config:

```
cp /etc/glance/glance-api.conf /etc/glance/glance-api.conf.orig
```

Update `/etc/glance/glance-api.conf`:

```
...
 [glance_store]
stores = glance.store.rbd.Store
default_store = rbd
rbd_store_pool = images
rbd_store_user = images
rbd_store_ceph_conf = /etc/ceph/ceph.conf
```

Restart Glance:

```
systemctl restart openstack-glance-api
```

Download CirrOS image:

```
wget http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img
```

Convert from QCOW2 to RAW. It is recommended that Ceph use RAW format:

```
qemu-img convert cirros-0.3.4-x86_64-disk.img cirros-0.3.4-x86_64-disk.
raw
```

Add image to Glance:

```
openstack image create Cirros-0.3.4 --disk-format raw --container-format
bare --public --file cirros-0.3.4-x86_64-disk.raw
```

Verify that the image exists in Ceph:

```
su - ceph
sudo rbd ls images
$UUID       RESPONSE
sudo rbd info images/$UUID
```

Integrate Ceph with Cinder

Cinder is the block storage service in OpenStack. Cinder provides an abstraction around block storage and allows vendors to integrate by providing a driver. In Ceph, each storage pool can be mapped to a different Cinder backend. This allows for creating storage services such as gold, silver, or bronze. You can decide, for example, that gold should be fast SSD disks that are replicated three times, while silver only should be replicated two times and bronze should use slower disks with erasure coding.

Create a Ceph pool for Cinder volumes:

```
sudo ceph osd pool create volumes 32
```

Create a keyring to grant Cinder access:

```
sudo ceph auth get-or-create client.volumes mon 'allow r' osd \
    'allow class-read object_prefix rbd_children, allow rwx \
    pool=volumes, allow rx pool=images' \
    -o /etc/ceph/ceph.client.volumes.keyring
```

Copy the keyring to OpenStack Controller(s):

```
scp /etc/ceph/ceph.client.volumes.keyring root@controller:/etc/ceph
```

Create a file that contains only the authentication key on OpenStack controller(s):

```
sudo ceph auth get-key client.volumes | ssh controller tee client.
volumes.key
```

Set needed permissions on keyring file to allow access by Cinder:

```
chgrp cinder /etc/ceph/ceph.client.volumes.keyring
chmod 0640 /etc/ceph/ceph.client.volumes.keyring
```

Add the keyring to the Ceph configuration file on OpenStack controller(s).

Edit `/etc/ceph/ceph.conf`:

```
...
[client.volumes]
keyring = /etc/ceph/ceph.client.volumes.keyring
```

Give KVM hypervisor access to Ceph:

```
uuidgen |tee /etc/ceph/cinder.uuid.txt
```

`scp` contents of `/etc/ceph/` to compute node:

```
scp /etc/ceph/* root@compute1:/etc/ceph/
```

On the compute node, create a secret in virsh so KVM can access the Ceph pool for Cinder volumes.

Edit `/etc/ceph/cinder.xml`:

```
ce6d1549-4d63-476b-afb6-88f0b196414f
client.volumes secret
virsh secret-define --file /etc/ceph/cinder.xml
virsh secret-set-value --secret ce6d1549-4d63-476b-afb6-88f0b196414f \
    --base64 $(cat /etc/ceph/client.volumes.key)
```

Add Ceph backend for Cinder.

Edit `/etc/cinder/cinder.conf`:

```
...
[DEFAULT]
...
enabled_backends = ceph
[rbd]
volume_driver = cinder.volume.drivers.rbd.RBDDriver
rbd_pool = volumes
rbd_ceph_conf = /etc/ceph/ceph.conf
rbd_flatten_volume_from_snapshot = false
rbd_max_clone_depth = 5
rbd_store_chunk_size = 4
rados_connect_timeout = -1
glance_api_version = 2
rbd_user = volumes
rbd_secret_uuid = 00000000-00000000-00000000 # This should be the same
as the secret set in /etc/ceph/cinder.xml
```

Restart Cinder on controller(s):

```
openstack-service restart cinder
```

Create Ceph volume backend:

```
openstack volume type create --property volume_backend_name=RBD CEPH
```

Create volume:

```
openstack volume create --type CEPH --display-name="cephtest01" 1
```

List volume using the Cinder CLI:

```
cinder list
```

List volume using the rbd CLI from the Ceph admin node (node1):

```
sudo rbd -p volumes ls
sudo rbd info volumes/vol-UUID
```

Integrate Ceph with Nova Compute

Nova is the compute service within OpenStack. Nova stores virtual disks images associated with running VMs by default, locally on the hypervisor under `/var/lib/nova/instances`. There are a few drawbacks to using local storage on compute nodes for virtual disk images:

- Images are stored under the root filesystem. Large images can cause the file system to fill up, thus crashing compute nodes.

- A disk crash on the compute node could cause loss of virtual disk, and as such, a VM recovery would be impossible.

Ceph is one of the storage backends that can integrate directly with Nova. In this section, we see how to configure that.

Ceph Admin Node (node1)

On the Ceph admin node (node1), create a Ceph pool for Nova:

```
sudo ceph osd pool create vms 64
```

Create an authentication ring for Nova on the Ceph admin node:

```
sudo ceph auth get-or-create client.nova mon 'allow r' osd \
    'allow class-read object_prefix rbd_children, allow rwx pool=vms, \
    allow rx pool=images' -o /etc/ceph/ceph.client.nova.keyring
```

Copy the keyring to OpenStack controller(s):

```
scp /etc/ceph/ceph.client.nova.keyring root@controller:/etc/ceph
```

Create key file on OpenStack controller(s):

Creates as *ceph* user:

```
sudo ceph auth get-key client.nova |ssh controller tee client.nova.key
```

OpenStack Controller

Set permissions of the keyring file to allow access by Nova:

```
chgrp nova /etc/ceph/ceph.client.nova.keyring
chmod 0640 /etc/ceph/ceph.client.nova.keyring
```

Verify correct packages are installed:

```
yum list installed python-rbd ceph-common
```

Update Ceph config.

Edit `/etc/ceph/ceph.conf`:

```
...
[client.nova]
keyring = /etc/ceph/ceph.client.nova.keyring
```

Give KVM access to Ceph:

```
uuidgen |tee /etc/ceph/nova.uuid.txt
```

Create a secret in virsh so KVM can access the Ceph pool for Cinder volumes.

Generate a new secret with `uuidgen`.

Edit `/etc/ceph/nova.xml`:

```
c89c0a90-9648-49eb-b443-c97adb538f23
client.volumes secret
```

Run:

```
virsh secret-define --file /etc/ceph/nova.xml
virsh secret-set-value --secret c89c0a90-9648-49eb-b443-c97adb538f23 \
    --base64 $(cat /etc/ceph/client.nova.key)
```

Create a backup of `nova.conf`:

```
cp /etc/nova/nova.conf /etc/nova/nova.conf.orig
```

Update `nova.conf` to use Ceph backend.

Edit `/etc/nova/nova.conf`:

```
...
force_raw_images = True
disk_cachemodes = writeback
...
[libvirt]
images_type = rbd
images_rbd_pool = vms
images_rbd_ceph_conf = /etc/ceph/ceph.conf
rbd_user = nova
rbd_secret_uuid = c89c0a90-9648-49eb-b443-c97adb538f23
```

Restart Nova.

```
systemctl restart openstack-nova-compute
```

Launch a new ephemeral instance using the image uploaded during Glance/Ceph integration.

```
openstack network list
openstack image list
openstack flavor list
openstack server create cephvm --flavor m1.small --nic net-id=$UUID \
    --image='Cirros 0.3.4'
```

After build completes, list the images in the Ceph VMS pool from the Ceph admin node (node1):

```
sudo rbd -p vms ls
```

Optional - Ceph Object Gateway Node (ceph4)

Node Requirements

- RGW node (ceph4)
- 4096MB RAM
- 30GB root disk
- 100GB data disk
- 1 vCPU
- Network
    - Shadowman
- Subscription
    - Red Hat OpenStack Platform
- Repositories
    - rhel-7-server-rpms

Repositories

Register with Red Hat Subscription Manager:

```
subscription-manager register
```

List subscription pools:

```
subscription-manager list --available
```

Attach RHOSP subscription pool:

```
subscription-manager attach --pool=$POOL-UUID
```

Disable all repos:

```
subscription-manager repos --disable=*
```

Enable Red Hat Enterprise Linux 7 repository:

```
subscription-manager repos --enable=rhel-7-server-rpms --enable=rhel-7-
server-extras-rpms
```

NTP and SELinux

Update and Install `ntp`:

```
yum -y update && yum -y install ntp
```

Set `ntp` to start at boot:

```
systemctl enable ntpd.service && systemctl start ntpd.service
```

Verify synchronization:

```
ntpq -p
```

Disable SELinux:

```
sed -i "s/SELINUX=enforcing/SELINUX=disabled/g" /etc/selinux/config
```

FirewallD

```
Permanently open ports 80, 2003, 4505-4506, 6789, 6800-7300, 7480:
firewall-cmd --zone=public --add-port=80/tcp --permanent
firewall-cmd --zone=public --add-port=2003/tcp --permanent
firewall-cmd --zone=public --add-port=4505-4506/tcp --permanent
firewall-cmd --zone=public --add-port=6789/tcp --permanent
firewall-cmd --zone=public --add-port=6800-7300/tcp --permanent
firewall-cmd --zone=public --add-port=7400/tcp --permanent
firewall-cmd --reload
```

Ceph User

Create Ceph user:

```
useradd ceph
echo openstack | passwd --stdin ceph
```

Grant *ceph* user passwordless sudo privileges:

```
Create /etc/sudoers.d/ceph
ceph ALL = (root) NOPASSWD:ALL
Defaults:ceph !requiretty
```

Secure *ceph* `sudoers.d` file:

```
chmod 440 /etc/sudoers.d/ceph
```

SSH Keys

Drop to *ceph* user:

```
su - ceph
```

Create an SSH key for the *ceph* user:

```
ssh-keygen
```

Copy SSH keys to admin, mon, and OSD nodes:

```
ssh-copy-id ceph@ceph{1-3}
```

EPEL Repository

Install Extra Packages for Enterprise Linux (EPEL) repository:

```
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.
rpm
rpm -ivh epel-release-latest-7.noarch.rpm
```

ceph_stable Repository

Manually create `/etc/yum.repos.d/ceph_stable.repo`:

```
[ceph_stable]
```

```
baseurl = http://download.ceph.com/rpm-kraken/el7/$basearch
gpgcheck = 1
gpgkey = https://download.ceph.com/keys/release.asc
name = Ceph Stable repo
```

All Nodes: Hosts

Add entries to `/etc/hosts` for ceph4 on Ceph nodes 1-4, OpenStack controller, and network node:

```
$IP controller  controller.$fqdn
$IP network network.$fqdn
$IP ceph1   ceph1.$fqdn
$IP ceph2   ceph2.$fqdn
$IP ceph3   ceph3.$fqdn
$IP ceph4   ceph4.$fqdn
```

Ceph Admin Node

Username Resolution

Add entries for ceph4 to '~/.ssh/config':

```
Host ceph4
Hostname ceph4.fqdn
User ceph
```

Copy SSH keys to RGW node:

```
ssh-copy-id ceph@ceph4
```

Ceph Object Gateway Installation

Check `ceph-deploy` version:

```
ceph-deploy --version
```

If Running ceph-deploy Version Earlier Than 1.5.36

Version v1.5.25 of `ceph-deploy` is available in the Ceph Kraken repository but has a few bugs that will interrupt the build of your Ceph RGW node. This has been resolved in a later release. Before we can install Ceph OGW on ceph4, we need to update `ceph-deploy` to `v1.5.36` or later using `python2-pip`.

Install `python2-pip`:

```
sudo yum -y install python2-pip
```

Uninstall `ceph-deploy-1.5.25`:

```
sudo yum erase ceph-deploy
```

Upgrade `ceph-deploy`:

```
sudo pip install ceph-deploy
```

Install Ceph Object Gateway:

```
sudo ceph-deploy install --rgw ceph4
```

Make the ceph4 node an admin node:

```
ceph-deploy admin ceph4
```

Verify installation:

```
curl http://ceph4:7480
```

If the gateway is operational, the response should look a bit like this:

```
<?xml version="1.0" encoding="UTF-8"?> <ListAllMyBucketsResult
xmlns="http://s3.amazonaws.com/doc/2006-03-01/"> <Owner> <ID>anonymous</
ID> <DisplayName></DisplayName> </Owner> <Buckets> </Buckets> </
ListAllMyBucketsResult>
```

Change Default Port

As of Ceph Firefly (v0.80), Ceph Object Gateway is running on Civetweb (embedded into the `ceph-radosgw` daemon) instead of Apache and FastCGI. Using Civetweb simplifies the Ceph Object Gateway installation and configuration.

Civetweb runs on port 7480, by default. To change the default port (e.g., to port 80), modify your Ceph configuration file in the working directory of your administration server. Add a section entitled `[client.rgw.]`, replacing with the short node name of your Ceph Object Gateway node (i.e., `hostname -s`).

Admin Node (ceph4)

Edit `ceph.conf`:

```
[client.rgw.ceph4]
rgw_frontends = "civetweb port=80"
```

Push updated `ceph.conf` to RGW node:

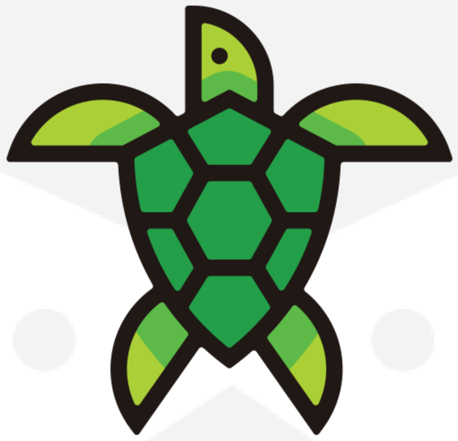
```
ceph-deploy --overwrite-conf config push ceph4
```

RGW Node (ceph4)

Restart Ceph Object Gateway:

```
sudo systemctl restart ceph-radosgw@rgw.ceph4
```

Projects, Users, and Roles



Managing Projects

Projects, previously known as tenants or accounts, are organizational units in the cloud comprised of zero or more users. Projects own specific resources in the OpenStack environment. Projects, users, and roles can be managed independently of one another, and, during setup, the operator defines at least one project, user, and role.

List current projects:

```
openstack project list
```

Create a project:

```
openstack project create --description "Red Hat Ex310K" shadowman
```

Disable/enable a project:

```
openstack project set $PROJECT_ID [--disable | --enable ]
```

Delete a project:

```
openstack project delete $PROJECT_ID
```

Managing Users

List users:

```
openstack user list
```

Create a new user:

```
openstack user create --project admin --password openstack bigfin
```

Disable/enable a user:

```
openstack user set bigfin [--disable | --enable]
```

Update user project

```
openstack user set bigfin --project shadowman --email bigfin@example.com
```

Managing Roles

Keystone roles are a personality, including a set of rights and privileges, that a user assumes to perform a specific set of operations. Roles permit users to be members of multiple projects. Users are assigned to more than one project by defining a role, then assigning said role to a user/project pair.

OpenStack identity (Keystone) defines a user's role on a project, but it is up to the individual service (policy) to define what that role means. To change a policy, edit the file `/etc/$service/policy.json`.

Create role:

```
openstack role create ex310
```

List user details:

```
openstack user list | grep bigfin
openstack project list
openstack role list
```

Assign role to the user/project pair:

```
openstack role add --user bigfin --project shadowman ex310
```

Verify role assignment:

```
openstack role assignment list --user bigfin --project shadowman --names
```

View role details:

```
openstack role show ex310
```

Remove a role:

```
openstack role remove --user bigfin --project shadowman ex310
```

Neutron

Neutron is an OpenStack project to provide "network connectivity as a service" between interface devices (e.g., vNICs) managed by other OpenStack services (e.g., Nova). The Networking service, code-named Neutron, provides an API that lets you define network connectivity and addressing in the cloud.

A standalone component in OpenStack modular architecture positioned alongside other OpenStack services, Neutron uncouples network dependencies from all of the sub-components previously required by nova-network (deprecated). The Neutron abstraction layer allows projects (tenants) to create their own rich network topologies and configure advanced network policies.

Advanced network services such as FWaaS, LBaaS, and VPNaaS can be inserted either as VMs that route between networks or as API extensions.

Creating Internal Networks

List networks:

```
openstack network list
openstack subnet list
```

Create a new, private network named `ex310k-private`:

```
openstack network create ex310k-private
```

Create new subnet for the `ex310k-private` network named `ex310k-sub` on 192.168.1.0/24:

```
openstack subnet create --network ex310k-private --subnet-range
192.168.1.0/24 ex310k-sub
```

Creating External Networks

Create an external network:

```
openstack network create shadownet --share -external
```

Create subnet:

```
openstack subnet create \
--allocation-pool start=172.24.10.8,end=172.24.10.55 \
--gateway 172.24.10.1 --no-dhcp --network shadownet \
--subnet-range 172.24.10.0/24 --dns-nameserver 192.168.10.1 shadownet-
sub
```

Creating Network Ports

List subnets:

```
openstack subnet list
```

Create port:

```
openstack port create --network shadownet \
--fixed-ip subnet=$SUBNET,ip-address=172.24.10.12 \ ex310-port
```

Create new server on shadowman network:

```
openstack server create ex310 --image cirros-0.3.4 --flavor c2.web \
--nic port-id=$UUID --wait
```

Floating IPs

Create new server:

```
openstack server create mantaray --image cirros-0.3.4 \
--flavor m1.tiny --nic net-id=$PRIVATE-UUID
```

Create a floating IP address:

```
openstack floating ip create shadownet \
--floating-ip-address=172.24.10.20
```

Add floating IP to a server:

```
openstack server add floating ip mantaray 172.24.10.20
```

Security Groups

Neutron Security Groups are for filtering both ingress and egress traffic at the hypervisor level. For ingress traffic (to an instance), only traffic matched with security group rules are allowed. When there is no rule defined, all traffic is dropped. For egress traffic (from an instance) only traffic matched with security group rules are allowed. When there is no rule defined, all egress traffic is dropped.

When a new security group is created, rules to allow all egress traffic are automatically added and the "default" security group is defined for each tenant.

For the default security group, a rule which allows intercommunication among hosts associated with the default security group is defined by default. As a result, all egress traffic and intercommunication in the default group are allowed and all ingress from outside of the default group is dropped by default (in the default security group).

List all security groups:

```
openstack security group list
```

List rules in a specific security group:

```
openstack security group rule list $NAME
```

Create a new security group:

```
openstack security group create redhat --description "ex310k security
group"
```

Add a TCP rule allowing access to port 22 to redhat security group:

```
openstack security group rule create redhat --protocol tcp -dst-port
22:22 --src-ip 0.0.0.0/0
```

Associate security group with 'cephvm' server:

```
openstack server add security group cephvm redhat
```

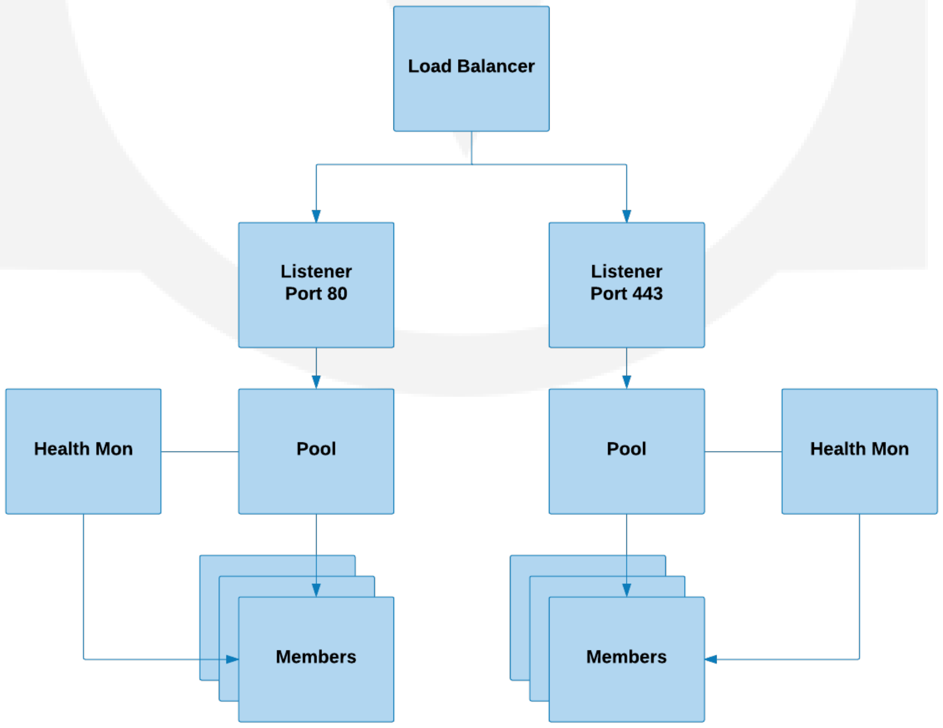
Remove a security group:

```
openstack server remove security group cephvm default
```

Delete an unwanted security group:

```
openstack security group delete default
```

LBaaS



Neutron Load balancing as a service, or LBaaS, is an advanced service in Neutron that allows for proprietary and open source load balancing technologies to drive the actual load balancing of requests, and enables Neutron networking to distribute incoming requests evenly among designated instances. Neutron LBaaS ensures that the workload is shared predictably among instances and enables more effective use of system resources.

- **Monitors**: Implemented to determine whether pool members are available to handle requests
- **Connection limits**: Allows shaping of ingress traffic with connection limits
- **Session persistence**: Supports routing decisions based on cookies & source IP addresses
- **Management**: LBaaS is managed using a variety of tool sets. The REST API is available for programmatic administration and scripting. Users perform administrative management of load balancers through either the CLI (neutron) or the OpenStack Dashboard.

LBaaS Algorithms

- **Round Robin**: Rotates requests evenly between multiple instances
- **Source IP**: Requests from a unique source IP address are consistently directed to the same instance
- **Least Connections**: Allocates requests to the instance with the least number of active connections

Pre-configuration

Install dependencies for OpenStackCLI and Horizon:

```
yum -y install openstack-neutron-lbaas openstack-neutron-lbaas-ui
```

Edit Horizon settings to enable LBaaS UI:

```
/etc/openstack-dashboard/local_settings
'enable_lb': True
```

Restart Apache:

```
systemctl restart httpd
```

Creating a Neutron Load Balancer using LBaaS V2

Create a load balancer:

```
openstack subnet list
neutron lbaas-loadbalancer-create --name ex310 ex310k-sub
```

Create LB listener:

```
neutron lbaas-listener-create --name ex310-listener \
--loadbalancer ex310 --protocol HTTP --protocol-port 80
```

Create a load balancer pool

```
neutron lbaas-pool-create --name ex310-pool \
--lb-algorithm ROUND_ROBIN --listener ex310-listener \
--protocol HTTP
```

Add members to ex310-pool:

```
openstack server list
neutron lbaas-member-create -subnet ex310k-sub \
--address 192.168.1.3 --protocol-port 80 ex310-pool
```

Add a floating IP to a load balancer:

```
openstack floating ip create shadownet
openstack port list
neutron floatingip-associate $FLOAT-ID $LB-PORT-ID
```

Create health monitor:

```
neutron lb-healthmonitor-create --delay 5 --type HTTP \
--max-retries 3 --timeout 10 --type HTTP --pool  ex310-pool
```

Check load balancer stats:

```
neutron lbaas-loadbalancer-stats ex310
```

Basic Neutron Troubleshooting

OVS Bridges

List OVS database contents:

```
ovs-vsctl show
```

List all OVS bridges:

```
ovs-vsctl list-br
```

List interfaces connected to OVS bridge:

```
ovs-vsctl list-ifaces $BRIDGE
```

Reset OVS to last working state:

```
ovs-vsctl emer-reset
```

Network Namespaces

Linux network namespaces are a kernel feature the networking service uses to support multiple isolated layer-2 networks with overlapping IP address ranges. The support may be disabled, but it is on by default. If it is enabled in your environment, your network nodes will run their dhcp-agents and l3-agents in isolated namespaces. Network interfaces and traffic on those interfaces will not be visible in the default namespace.

List namespaces on the network node:

```
ip netns
```

Interact with virtual network:

```
ip netns exec $(ip netns | grep qrouter) $COMMAND
```

Login to a virtual machine through a network namespace:

```
ip netns exec $(ip netns | grep qrouter) ssh cirros@172.24.10.20
```

Nova Compute

OpenStack Nova source software is designed to provision and manage large networks of virtual machines, create redundant and scalable cloud computing platform, and allow you to control an IaaS cloud computing platform. Nova defines drivers that interact with underlying virtualization mechanisms that run on your host OS and exposes functionality over a web-based API. Hardware and hypervisor agnostic, Nova currently supports a variety of standard hardware configurations and seven major hypervisors.

Server Flavors

List flavors:

```
openstack flavor list
```

Create a new, public `c1.web` flavor:

```
openstack flavor create c1.web --ram 512 --disk 1 --vcpus 1
```

Create a new, public `c2.web` flavor with ephemeral disk:

```
openstack flavor create c2.web --ram 1024 --disk 2 \
    --ephemeral 10 --vcpus 1
```

Edit existing flavor:

```
openstack flavor set c2.web --property hw:cpu_cores=5
```

Glance

OpenStack Compute (Nova) relies on an external image service to store virtual machine images and maintain a catalog of available images. Glance is a core OpenStack service that accepts API requests for disk or server images, metadata definitions from end users or OpenStack Compute components, and provides a service that allows users to upload and discover data assets, including images and metadata definitions, meant to be used with other services Glance supports several backends. This is including but not limited to:

- Regular filesystems
- Vmware
- Swift object storage
- Rados block devices

Create a Fedora-Atomic Image

List images:

```
openstack image list
```

Create Fedora-Atomic image:

List images:

```
openstack image list
```

Create Fedora-Atomic image:

```
wget -O Fedora-Atomic-25-20170522.0.x86_64.raw.xz \ https://goo.gl/
LBBjBm
unxz Fedora-Atomic-25-20170522.0.x86_64.raw.xz
```

Upload image to Glance:

```
openstack image create fedora-atomic --disk-format raw \
--container-format bare --file \
Fedora-Atomic-25-20170522.0.x86_64.raw --public
```

Converting Images with qemu-img

Download CentOS-Atomic image:

```
wget -O CentOS-Atomic-Host-7-GenericCloud.qcow2.xz https://goo.gl/O0g3oR
unxz CentOS-Atomic-Host-7-GenericCloud.qcow2.xz
```

Convert the QCOW2 image to RAW:

```
qemu-img convert CentOS-Atomic-Host-7-GenericCloud.qcow2 \ CentOS-
Atomic-Host-7-GenericCloud.raw
```

Upload image to Glance:

```
openstack image create centos-atomic --disk-format raw \
--container-format bare --file \
CentOS-Atomic-Host-7-GenericCloud.raw --public
```

Ceph Troubleshooting

Check Ceph cluster health:

```
ceph health
```

Watch ongoing events in a Ceph cluster:

```
ceph -w
```

Check overall status:

```
ceph status
```

Check cluster usage stats:

```
ceph df
```

Ceph Mon status

If your cluster has multiple monitors, you should check the monitor quorum status before reading and/or writing data. A quorum must be present when multiple monitors are running. You should also check monitor status periodically to ensure that they are running.

Display the monitor map:

```
ceph mon stat
```

Check Ceph quorum status in environments with multiple mon nodes:

```
sudo ceph quorum_status --format json-pretty
```

Output:

```
{
 "election_epoch": 6,
 "quorum": [
 0,
 1,
 2
 ],
 "quorum_names": [
 "ceph1",
 "ceph2",
 "ceph3"
 ],
 "quorum_leader_name": "ceph1",
 "monmap": {
 "epoch": 1,
 "fsid": "188aff9b-7da5-46f3-8eb8-465e014a472e",
 "modified": "0.000000",
 "created": "0.000000",
 "mons": [
 {
 "rank": 0,
 "name": "ceph1",
 "addr": "192.168.0.31:6789\/0"
 },
 {
 "rank": 1,
 "name": "ceph2",
 "addr": "192.168.0.32:6789\/0"
 },
 {
 "rank": 2,
 "name": "ceph3",
 "addr": "192.168.0.33:6789\/0"
 }
 ]
 }
}
```

Ceph OSD Monitoring

Print Ceph OSD position in CRUSH map:

```
ceph osd tree
```

Repair an OSD:

```
ceph osd repair osd$i
```

Start a downed OSD:

```
systemctl start ceph-osd@$i
```

Delete an OSD:

```
ceph osd rm osd.$i
```

Deleting Unwanted OSD Pools

Edit Ceph config file on all mon nodes.

In `/etc/ceph/ceph.conf`:

```
[mon]
mon_allow_pool_delete = True
```

Reboot monitor nodes:

```
systemctl reboot
```

Delete unwanted pools:

```
ceph osd pool delete [name] [name] --yes-i-really-really-mean-it
```

Starting Over

If you want to redeploy Ceph without needing to reinstall the OS, the `ceph-deploy` command line client provides three easy commands to refresh your environment.

Uninstall Ceph packages from nodes:

```
ceph-deploy purge ceph1 ceph2 ceph3
```

Purge configuration files:

```
ceph-deploy purgedata ceph1 ceph2 ceph3
```

Remove keyrings:

```
ceph-deploy forgetkeys
```