Oracle Linux

Ceph Storage User's Guide

E96266-08
December 2024

Oracle Linux Ceph Storage User's Guide,

E96266-08

Contents

3 Using Ceph Storage for Oracle Linux

Preface

WARNING:

Oracle Linux 7 is now in Extended Support. See Oracle Linux Extended Support and Oracle Open Source Support Policies for more information.

Migrate applications and data to Oracle Linux 8 or Oracle Linux 9 as soon as possible.

This document contains information about Ceph Storage for Oracle Linux Release 3.0. It describes the differences from the upstream version, includes notes on installing and configuring Ceph Storage for Oracle Linux, and provides a statement of what is supported.

Document generated on: 2022-10-24 (revision: 13333)

Documentation License

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

Introduction to Ceph Storage for Oracle Linux Release 3.0

WARNING:

Oracle Linux 7 is now in Extended Support. See Oracle Linux Extended Support and Oracle Open Source Support Policies for more information.

Migrate applications and data to Oracle Linux 8 or Oracle Linux 9 as soon as possible.

This chapter provides information on Ceph Storage for Oracle Linux Release 3.0.

About Ceph Storage for Oracle Linux Release 3.0

Ceph Storage for Oracle Linux Release 3.0 presents a uniform view of object and block storage from a cluster of multiple physical and logical commodity-hardware storage devices. Ceph can provide fault tolerance and enhance I/O performance by replicating and striping data across the storage devices in a Ceph Storage Cluster. Ceph's monitoring and self-repair features minimize administration overhead. You can configure a Ceph Storage Cluster on non-identical hardware from different manufacturers.

Ceph Storage for Oracle Linux Release 3.0 is based on the Ceph Community Luminous release (v12.2.5). Differences between Oracle versions of the software and upstream releases are limited to Oracle specific fixes and patches for specific bugs.

Note:

The source RPMs for Ceph are available from Oracle Linux yum server at https://yum.oracle.com.

For a quick-start guide to using Ceph, see https://docs.ceph.com/en/latest/.

For more information about Ceph, go to https://docs.ceph.com/en/latest/.

Notable Updates and New Features

The following notable new features (from the Ceph Luminous upstream release) are included:

- A new Ceph Manager daemon, `ceph-mgr`, to monitor clusters
- Ceph Manager web-based dashboard
- New OSDs use the BlueStore backend to manage HDDs and SSDs
- Simplified OSD replacement process

The following upstream features are now available in this release:

- Ceph iSCSI Gateway
- Ceph file system (Ceph FS)
- Export Ceph file systems and block storage over NFS
- Ceph block devices with QEMU

2

Installing or Upgrading Ceph Storage for Oracle Linux

WARNING:

Oracle Linux 7 is now in Extended Support. See Oracle Linux Extended Support and Oracle Open Source Support Policies for more information.

Migrate applications and data to Oracle Linux 8 or Oracle Linux 9 as soon as possible.

This chapter discusses how to enable the repositories to install the Ceph Storage for Oracle Linux packages, how to perform an installation of those packages, and how to perform an upgrade.

Hardware and Network Requirements

Ceph Storage for Oracle Linux does not require specific hardware; however, certain Ceph operations are CPU and memory intensive. The X6 and X7 line of Oracle x86 Servers are suitable to host Ceph nodes. For more information on Oracle x86 Servers, see:

https://www.oracle.com/servers/x86/index.html

A minimum node configuration is:

- 4 CPU cores
- 4GB RAM
- 2 x 1GB Ethernet NICs
- 1 TB storage for object data

Your deployment needs may require nodes with a larger footprint. Additional considerations are detailed in the Ceph upstream documentation.

Operating System Requirements

Ceph Storage for Oracle Linux Release 3.0 is available for Oracle Linux 7 (x86_64) running the Unbreakable Enterprise Kernel Release 5 (UEK R5). A minimum of Oracle Linux 7.5 is required.

Enabling Access to the Ceph Storage for Oracle Linux Packages

The `ceph-deploy` package is available on the Oracle Linux yum server in the `ol7_ceph30` repository, or on the Unbreakable Linux Network (ULN) in the `ol7_x86_64_ceph30` channel, however there are also dependencies across other repositories and channels, and these must also be enabled on each system included in the Ceph Storage Cluster.

If you are using the Oracle Linux yum server, you must enable the following repositories:

- `ol7_ceph30`
- `ol7_addons`
- `ol7_latest`
- `ol7_optional_latest`
- `ol7_UEKR5`

If you are using ULN, you must enable the following channels:

- `ol7_x86_64_ceph30`
- `ol7_x86_64_addons`
- `ol7_x86_64_latest`
- `ol7_x86_64_optional_latest`
- `ol7_x86_64_UEKR5`

Enabling Repositories with ULN

If you are registered to use ULN, use the ULN web interface to subscribe the system to the appropriate channels:

1. Log in to https://linux.oracle.com with your ULN user name and password.
2. On the Systems tab, click the link named for the system in the list of registered machines.
3. On the System Details page, click **Manage Subscriptions**.
4. On the System Summary page, select each required channel from the list of available channels and click the right arrow to move the channel to the list of subscribed channels.

   Subscribe the system to the `ol7_x86_64_ceph30`, `ol7_x86_64_addons`, `ol7_x86_64_latest`, `ol7_x86_64_optional_latest` and `ol7_x86_64_UEKR5` channels.
5. Click **Save Subscriptions**.

Enabling Repositories with the Oracle Linux Yum Server

To enable the required repositories on the Oracle Linux yum server, ensure that your system is up to date and that you have transitioned to use the modular yum repository configuration by installing the `oraclelinux-release-el7` package and running the `/usr/bin/ol_yum_configure.sh` script.

```
# yum install oraclelinux-release-el7
# /usr/bin/ol_yum_configure.sh
```

Install the `oracle-ceph-release-el7` release package to install appropriate yum repository configuration.

```
# yum install oracle-ceph-release-el7
```

Enable the following repositories:

- `ol7_ceph30`
- `ol7_addons`
- `ol7_latest`
- `ol7_optional_latest`

- `ol7_UEKR5`

Use the `yum-config-manager` tool to update your yum configuration:

```
# yum-config-manager --enable ol7_ceph30 ol7_latest ol7_optional_latest
ol7_addons ol7_UEKR5
```

You can now prepare the Ceph Storage Cluster nodes for installation of the Ceph Storage for Oracle Linux packages. See Installing and Configuring a Ceph Storage Cluster.

Installing and Configuring a Ceph Storage Cluster

A Ceph Storage Cluster consists of several systems, known as *nodes*. The nodes run various software daemons:

- Every node runs the Ceph Object Storage Device (OSD) daemon.
- One or more nodes run the Ceph Monitor and Ceph Manager daemons. Ceph Monitor and Ceph Manager should run on the same nodes.
- Optionally, one or more nodes run the Ceph Object Gateway daemon.
- Optionally, one or more nodes run the Metadata Server daemon to use Ceph File System.

A node is selected as an *administration* node from which Ceph commands can be run to control and monitor the cluster. Typically the administration node is also used as the *deployment* node, from which other systems can automatically be set up and configured as nodes in the cluster.

For data integrity, a Ceph Storage Cluster should contain two or more nodes for storing copies of an object. For high availability, a Ceph Storage Cluster should contain three or more nodes that store copies of an object.

In the example used in the following steps, the administration and deployment node is `ceph-node1.example.com`. The nodes used in the Ceph Storage Cluster are `ceph-node2.example.com`, `ceph-node3.example.com`, and `ceph-node4.example.com`.

Preparing Ceph Storage Cluster Nodes

There are some basic requirements for each Oracle Linux system that you intend to use as a Ceph Storage Cluster node. These include the following items, for which some preparatory work may be required before you can begin your deployment.

1. Time must be accurate and synchronized across the nodes within the Ceph Storage Cluster. This is achieved by installing and configuring NTP on each system that you wish to run as a node in the cluster. If the NTP service if not already configured, install and start it. See Oracle® Linux 7: Administrator's Guide for more information on configuring NTP.

   Note:

   Use the `hwclock --show` command on a node in the cluster to ensure that all nodes agree on the time. If the clocks on the nodes differ by more than 50 milliseconds, the `ceph health` command displays the warning:

   ```
   health HEALTH_WARN clock skew detected on mon
   ```

2. Ceph Storage Cluster network communications must be able to take place between nodes within the cluster. If firewall software is running on any of the nodes, it must either be disabled or, preferably, configured to facilitate network traffic on the required ports.

   Preferably, leave the firewall running and configure the following rules:

   a. Allow TCP traffic on port 6789 to enable the Ceph Monitor:

      ```
      # firewall-cmd --zone=public --add-port=6789/tcp --permanent
      ```

   b. Allow TCP traffic for ports 6800 to 7300 to enable the traffic for the Ceph OSD daemon:

      ```
      # firewall-cmd --zone=public --add-port=6800-7300/tcp --permanent
      ```

   c. Allow TCP traffic on port 7480 to enable the Ceph Object Gateway:

      ```
      # firewall-cmd --zone=public --add-port=7480/tcp --permanent
      ```

   d. Allow TCP traffic on ports 3260 and 5000 on Ceph iSCSI Gateway nodes:

      ```
      # firewall-cmd --zone=public --add-port=3260/tcp --add-port=5000/tcp --
      permanent
      ```

   e. Allow TCP traffic on port 2049 on any NFS server nodes:

      ```
      # firewall-cmd --zone=public --add-port=2049/tcp --permanent
      ```

   f. After modifying firewall rules, reload and restart the firewall daemon service:

      ```
      # firewall-cmd --reload
      # systemctl restart firewalld.service
      ```

   Alternatively, stop and disable the firewall daemon on Oracle Linux 7:

   ```
   # systemctl stop firewalld
   # systemctl disable firewalld
   ```

3. Ceph Storage Cluster nodes must be able to resolve the fully qualified domain name for each node within the cluster. You may either use DNS for this purpose, or provide entries within `/etc/hosts` for each system. If you select to rely on DNS, it must have sufficient redundancy to ensure that the cluster is able to perform name resolution at any time. If you select to edit `/etc/hosts`, add entries for the IP address and host name of all of the nodes in the Ceph Storage Cluster, for example:

   ```
   192.168.1.51    ceph-node1.example.com ceph-node1
   192.168.1.52    ceph-node2.example.com ceph-node2
   192.168.1.53    ceph-node3.example.com ceph-node3
   192.168.1.54    ceph-node4.example.com ceph-node4
   ```

   Note:

   Although you can use DNS to configure host name to IP address mapping, Oracle recommends that you also configure `/etc/hosts` in case the DNS service becomes unavailable.

4. The Ceph Storage Cluster deployment node must be able to connect to each prospective node in the cluster over SSH, to facilitate deployment. To do this, you must generate an SSH key on the deployment node and copy the public key to each of the other nodes in the Ceph Storage Cluster.

   a. On the deployment node, generate the SSH key, specifying an empty passphrase:

```
# ssh-keygen
```

   b. From the deployment node, copy the key to the other nodes in the Ceph Storage Cluster, for example:

```
# ssh-copy-id root@ceph-node2
# ssh-copy-id root@ceph-node3
# ssh-copy-id root@ceph-node4
```

5. To prevent errors when running `ceph-deploy` as a user with passwordless `sudo` privileges, use `visudo` to comment out the `Defaults requiretty` setting in `/etc/sudoers` or change it to `Defaults:ceph !requiretty`.

You can now install and configure the Ceph Storage Cluster deployment node, which is usually the same system as the administration node. See Installing and Configuring a Deployment Node.

Installing and Configuring a Deployment Node

In the example used in the following steps, the deployment node is `ceph-node1.example.com` (192.168.1.51), which is the same as the administration node.

Perform the following steps on the deployment node:

1. Install the `ceph-deploy` package.

```
# yum install ceph-deploy
```

2. Create a Ceph configuration directory for the Ceph Storage Cluster and change to this directory, for example:

```
# mkdir /var/mydom_ceph
# cd /var/mydom_ceph
```

   This is the working configuration directory used by the deployment node to roll out configuration changes to the cluster and to client and gateway nodes. If you need to make changes to the Ceph configuration files in future, the changes should be made in this directory and then use the `ceph-deploy config push` command to update the configuration for other nodes in the cluster.

   All `ceph-deploy` commands should be run from this working directory. Future commands in this guide use the notation *`$CEPH_CONFIG_DIR`* to represent this directory.

3. Use the `ceph-deploy` command to define the members of the Ceph Storage Cluster, for example:

```
# $CEPH_CONFIG_DIR/ceph-deploy new ceph-node2
```

   To define multiple nodes, provide them in a space separated list, for example:

```
# $CEPH_CONFIG_DIR/ceph-deploy new ceph-node2 ceph-node3 ceph-node4
```

4. (Optional) The first time you run the `ceph-deploy` command a set of files are created to configure the Ceph Storage Cluster. You can edit *`$CEPH_CONFIG_DIR`*`/ceph.conf` to set options for Ceph features. For example, if you are setting up a test environment with only a few OSDs, you can reduce the default number of replicas (defaults to `3`), for example:

```
osd pool default size = 2
```

5. (Optional) BlueStore is the default backend for new OSDs. As well as BlueStore, Oracle supports FileStore, which uses either Btrfs or XFS file systems on OSDs. Oracle

recommends Btrfs or XFS file systems on OSDs for Oracle workloads. You can use a mixture of BlueStore and FileStore OSDs in a Ceph Storage Cluster.

To use Btrfs on OSDs, add the following entry to the Ceph configuration file:

```
enable experimental unrecoverable data corrupting features = btrfs
```

You can now install Ceph on the remaining Ceph Storage Cluster nodes. See Installing and Configuring Ceph Storage Cluster Nodes.

Installing and Configuring Ceph Storage Cluster Nodes

Having installed and configured the Ceph Storage for Oracle Linux deployment node, you can use this node to install Ceph Storage for Oracle Linux on the other nodes participating in the Ceph Storage Cluster.

To install Ceph Storage for Oracle Linux on all the Ceph Storage Cluster nodes, run the following command on the deployment node:

```
# $CEPH_CONFIG_DIR/ceph-deploy install ceph-node2
```

To install the software on multiple nodes, provide them in a space separated list, for example:

```
# $CEPH_CONFIG_DIR/ceph-deploy install ceph-node2 ceph-node3 ceph-node4
```

After installing the Ceph packages on each node, you may need to add the Ceph services to the firewall rules on each node, for example.

```
# sudo firewall-cmd --zone=public --add-service=ceph-mon --permanent
# sudo firewall-cmd --zone=public --add-service=ceph --permanent
# sudo firewall-cmd --reload
# sudo systemctl restart firewalld.service
```

To configure the Ceph Storage Cluster, perform the following steps on the administration node:

1. Initialize Ceph Monitor:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy mon create-initial
   ```

2. Deploy a Ceph Monitor on one or more nodes in the Ceph Storage Cluster, for example:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy mon create ceph-node2
   ```

   To deploy a Ceph Monitor on multiple nodes, provide them in a space separated list, for example:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy mon create ceph-node2 ceph-node3 ceph-node4
   ```

   For high availability, Oracle recommends that you configure at least three nodes as Ceph Monitors.

3. Deploy the Ceph Manager daemon to one or more nodes in the Ceph Storage Cluster. Oracle recommends you deploy the Ceph Manager daemon to the same nodes you use as Ceph Monitors.

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy mgr create ceph-node2
   ```

   To deploy Ceph Manager on multiple nodes, provide them in a space separated list, for example:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy mgr create ceph-node2 ceph-node3 ceph-node4
   ```

4. Gather the monitor keys and the OSD and MDS bootstrap keyrings from one of the Ceph Monitors, for example:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy gatherkeys ceph-node3
   ```

5. Create the OSD nodes to use in the Ceph Storage Cluster. For information on creating OSDs, see Creating an Object Storage Device (OSD) Node.
6. Create an administration node to use the Ceph monitoring Command Line Interface (CLI), `ceph`, to manage the Ceph Storage Cluster. For information on creating an administration host, see Creating a Ceph Storage Cluster Administration Host.
7. You can check the status and health of the cluster using `ceph` command. On the administration node, enter:

   ```
   # ceph health
   # ceph status
   ```

   It usually takes several minutes for the cluster to stabilize before its health is shown as `HEALTH_OK`. You can also check the cluster quorum status to get an indication on the quorum status of the cluster monitors:

   ```
   # ceph quorum_status --format json-pretty
   ```

   Refer to the upstream Ceph documentation for help troubleshooting any issues with the health or status of your cluster.
8. (Optional) If you want to use Ceph File System (Ceph FS), deploy a Ceph Metadata Server (MDS). You can install the MDS service on an existing Monitor node within your Ceph Storage Cluster, as the service does not have significant resource requirements. For more information on setting up Ceph FS, see Setting Up and Using Ceph FS.

Creating an Object Storage Device (OSD) Node

This section discusses creating Object Storage Device (OSD) nodes in a Ceph Storage Cluster.

BlueStore is the default backend for OSDs. As well as BlueStore, Oracle supports the FileStore option, which provides XFS and Btrfs file systems on OSDs. Oracle recommends and XFS or Btrfs file systems on OSDs for Oracle workloads.

Creating a BlueStore OSD

Before you add a BlueStore OSD node to a Ceph Storage Cluster, you should first delete all data on the specified device.

```
# $CEPH_CONFIG_DIR/ceph-deploy disk zap node device
```

Replace *node* with the node name or host name where the disk is located. Replace *device* with the path to the device on the host where the disk is located. For example, to delete the data on a device named `/dev/sdb` on a node named `ceph-node2` in the Ceph Storage Cluster:

```
# $CEPH_CONFIG_DIR/ceph-deploy disk zap ceph-node2 /dev/sdb
```

To create a BlueStore OSD, enter:

```
# $CEPH_CONFIG_DIR/ceph-deploy osd create --data device node
```

This command creates a volume group and logical volume using the disk you specify. Data and journal reside on the same logical volume. Replace *node* with the node name or host name where the disk is located. Replace *device* with the path to the device on the host where the disk is located. For example, run:

```
# $CEPH_CONFIG_DIR/ceph-deploy osd create --data /dev/sdb ceph-node2
```

Creating a FileStore OSD

To create a FileStore OSD, you should first create the required partitions: one for data, and one for journal. This should be done on the OSD node. This example creates a data partition on `/dev/sdb1` with a size of 40GB, and a journal partition on `/dev/sdb2` with a size of 12GB:

```
# parted /dev/sdb --script -- mklabel gpt
# parted --script /dev/sdb mkpart primary 0MB 40000MB
# parted --script /dev/sdb mkpart primary 42000MB 55000MB
# dd if=/dev/zero of=/dev/sdb1  bs=1M count=1000
# sgdisk --zap-all --clear --mbrtogpt -g -- /dev/sdb2
# ceph-volume lvm zap /dev/sdb2
```

On the deployment node, create the FileStore OSD. To specify the OSD file type, use the `--filestore` and `--fs-type` parameters when creating OSDs. This example shows how to create a FileStore OSD with XFS:

```
# CEPH_CONFIG_DIR/ceph-deploy osd create --filestore --fs-type xfs \
   --data /dev/sdb1 --journal /dev/sdb2 ceph-node2
```

To use Btrfs on OSDs, use the `--fs-type btrfs` option when creating OSDs, for example:

```
# CEPH_CONFIG_DIR/ceph-deploy osd create --filestore --fs-type btrfs \
   --data /dev/sdb1 --journal /dev/sdb2 ceph-node2
```

Creating a Ceph Storage Cluster Administration Host

When you have set up a Ceph Storage Cluster, you should provide the client admin key and the Ceph configuration file to another host so that a user on the host can use the `ceph` command line as an administrative user. In this example, *ceph-node1* is the host name of the client system configured to manage the cluster using the Ceph Command Line Interface (CLI). This node is also used as the deployment node in examples.

On the deployment node, use `ceph-deploy` to install the Ceph CLI, for example:

```
# $CEPH_CONFIG_DIR/ceph-deploy admin ceph-node1
```

If you are using a separate host (not the deployment node as used in these examples) there are a few more steps.

1. On the deployment node, copy the SSH key to the host you want to use for Ceph administration, for example:

   ```
   # ssh-copy-id root@ceph-client
   ```

   This example assumes that you have configured entries for the Ceph client system in DNS and/or in `/etc/hosts`.

2. On the deployment node, use `ceph-deploy` to install the Ceph Storage for Oracle Linux packages, for example:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy install ceph-client
   ```

3. On the deployment node, copy the Ceph configuration file and Ceph keyring, and install the Ceph CLI (**ceph**), for example:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy admin ceph-client
   ```

You can now use the Ceph CLI (`ceph`) to manage the Ceph Storage Cluster.

Deploying the Ceph Manager Dashboard

The Ceph Manager service (`ceph-mgr`) includes a web-based user interface to view monitoring information about a Ceph Storage Cluster. The Ceph Manager dashboard is read-only.

If you want to increase security of the Ceph Manager dashboard, you should configure the module to listen on a local port, and use a proxy server to provide TLS or access control for remote access.

To deploy and connect to the Ceph Manager dashboard:

1. Enable the Ceph Manager dashboard:

   ```
   # ceph mgr module enable dashboard
   ```

   If you want to enable the dashboard when performing a Ceph deployment, add the following to the *$CEPH_CONFIG_DIR*/`ceph.conf` file:

   ```
   [mon]
           mgr initial modules = dashboard
   ```

2. Run the `ceph mgr services` command on the administration node to display the URL to access the dashboard, for example:

   ```
   # ceph mgr services
   {
       "dashboard": "http://ceph-node1.example.com:7000/"
   }
   ```

   The Ceph Manager dashboard runs on port `7000` by default. Connect to the URL displayed using a web browser to access the dashboard.

Removing a Ceph Storage Cluster Node

This section shows you how to remove OSD and a Ceph Manager nodes from a Ceph Storage Cluster. For information on removing other node types, see the upstream Ceph documentation.

NOT_SUPPORTED:

Performing the steps in this section removes all data on the Ceph Storage Cluster node.

Removing an OSD Node

This section shows you how to remove an OSD node from a Ceph Storage Cluster.

Removing an OSD node

1. On the OSD node, find the ID:

```
# systemctl status ceph-osd@*
```

The results list the ID of the node in a format similar to:

```
ceph-osd@1.service - Ceph object storage daemon osd.1
```

In this case the ID is 1.

2. On the administration node, remove the OSD from the cluster. For example, run:

   ```
   # ceph osd out osd.1
   ```

3. On the OSD node, stop and disable the service using the ID. For example, run:

   ```
   # systemctl stop ceph-osd@1.service
   # systemctl disable ceph-osd@1.service
   ```

4. On the administration node, remove the OSD from the CRUSH map, remove the authorization keys, and delete the OSD from the cluster. For example, run:

   ```
   # ceph osd crush remove osd.1
   # ceph auth del osd.1
   # ceph osd rm 1
   ```

5. To remove the data in the `/var/lib/ceph` directory, and uninstall the Ceph packages, from the deployment node, run:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy purge hostname
   ```

   Alternatively you can remove all data from the `/var/lib/ceph` directory, but leave the Ceph packages installed. To do this, from the deployment node, run:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy purgedata hostname
   ```

6. On the OSD node, delete the volume groups and volume labels used for OSDs on the disk. For example, to remove a volume label, use the `lvdisplay` command to list the volume labels, and delete any labels:

   ```
   # lvdisplay
   ...
   # lvremove --force /dev/ceph-dc39f7cc-e423-48d3-a466-9701e7bf972a/osd-block-
   f7db38d2-...
   ```

   Use the `vgdisplay` command to list the volume groups, and delete any groups:

   ```
   # vgdisplay
   ...
   # vgremove --force ceph-dc39f7cc-e423-48d3-a466-9701e7bf972a
   ```

7. If there are any configuration entries specific to the OSD you are removing in the Ceph configuration file, you should remove them. On the deployment node, edit the `$CEPH_CONFIG_DIR/ceph.conf` file to remove any entries for the node.

   On the deployment node, push the configuration change to all remaining OSD nodes. For example:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy --overwrite-conf config push ceph-node2 ceph-
   node3 ceph-node4
   ```

   On each remaining OSD node, restart the `ceph-osd` daemon:

   ```
   # systemctl restart ceph-osd.target
   ```

Removing a Ceph Monitor Node

This section shows you how to remove a Ceph Monitor node from a Ceph Storage Cluster.

To remove a Ceph Monitor node:

1. On the deployment node, remove the Ceph Monitor node from the cluster. For example, run:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy mon destroy hostname
   ```

2. On the Ceph Monitor node, stop and disable the service. For example, run:

   ```
   # systemctl stop ceph-mon@hostname.service
   # systemctl disable ceph-mon@hostname.service
   ```

3. To remove the data in the `/var/lib/ceph` directory, and uninstall the Ceph packages, from the deployment node, run:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy purge hostname
   ```

   Alternatively you can remove all data from the `/var/lib/ceph` directory, but leave the Ceph packages installed. To do this, from the deployment node, run:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy purgedata hostname
   ```

4. If there are any configuration entries specific to the Ceph Monitor node you are removing in the Ceph configuration file, you should remove them. On the deployment node, edit the `$CEPH_CONFIG_DIR/ceph.conf` file to remove any entries for the node.

   On the deployment node, push the configuration change to all remaining Ceph Monitor nodes. For example, run:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy --overwrite-conf config push ceph-node2 ceph-
   node3 ceph-node4
   ```

   On each remaining Ceph Monitor node, restart the `ceph-mon` daemon:

   ```
   # systemctl restart ceph-mon.target
   ```

Upgrading Ceph Storage Cluster Nodes

This section describes how to upgrade the Oracle Linux operating system, and update to the supported Unbreakable Enterprise Kernel. When the operating system on each node in the Ceph Storage Cluster has been upgraded, you can upgrade Ceph Storage for Oracle Linux on each node.

Upgrading Oracle Linux

The operating system must be at least Oracle Linux 7.5. Follow the steps to upgrade your operating system on each node in the Ceph Storage Cluster.

1. If you are using ULN, subscribe to the Oracle Linux 7 `ol7_x86_64_latest` channel. Alternatively, if you are using the Oracle Linux yum server, enable access to the Oracle Linux 7 `ol7_latest` repository.
2. Run the `yum update` command to update the packages.

   ```
   # yum update
   ```

3. When the upgrade has completed, reboot the system.

   ```
   # systemctl reboot
   ```

4. Select the new Oracle Linux 7 kernel version during the system startup if it is not the default boot kernel.

More detailed information on upgrading Oracle Linux 7 is available in Oracle® Linux 7: Installation Guide

Upgrading Unbreakable Enterprise Kernel

The Unbreakable Enterprise Kernel must be at least UEK R5. Follow the steps to upgrade to UEK R5 on each node in the Ceph Storage Cluster.

1. If you are using ULN, subscribe to the UEK R5 channel `ol7_x86_64_UEKR5`. Alternatively, if you are using the Oracle Linux yum server, enable access to the UEK R5 repository `ol7_UEKR5`. Make sure you also disable the UEK R4 channel and yum repository if they are enabled.
2. Run the `yum update` command to update the packages.

   ```
   # yum update
   ```

3. When the upgrade has completed, reboot the system.

   ```
   # systemctl reboot
   ```

4. Select the UEK R5 kernel version during the system startup if it is not the default boot kernel.

More detailed information on upgrading to UEK R5 is available in the Unbreakable Enterprise Kernel: Release Notes for Unbreakable Enterprise Kernel Release 5. Also take care to look at the release notes for the latest update release in Unbreakable Enterprise Kernel Documentation.

Upgrading Ceph Storage for Oracle Linux

This section discusses upgrading the Ceph Storage for Oracle Linux Release 2.0 components to the current release. During the upgrade, you do not have to stop the Ceph daemons, but Oracle recommends the Ceph Storage Cluster not be in use during the upgrade. To protect data integrity during the upgrade, Oracle recommends:

1. Remove the Ceph Storage Cluster from use.
2. Unmount all file systems in the cluster, including those for Ceph Block Device, Ceph Object Gateway, and Ceph File System (Ceph FS).
3. Upgrade the cluster using the instructions in this section.
4. Confirm the cluster is healthy.
5. Remount all file-systems.
6. Return the cluster to use.

Where component upgrade is required, it is recommended that the upgrades are performed in the following order:

1. Ceph Deploy Package
2. Ceph Monitors
3. Ceph Managers (deploy new daemon)

4. Ceph OSDs
5. Ceph Metadata Servers
6. Ceph Object Gateways

Oracle recommends that all daemons of a specific type are upgraded together to ensure that they are all on the same release, and that all of the components within a Ceph Storage Cluster are upgraded before you attempt to configure or use any new functionality in the current release.

The following instructions provide an outline of some of the common steps required to perform an upgrade for a Ceph Storage Cluster.

1. To begin the upgrade process, the yum configuration on all systems that are part of the Ceph Storage Cluster must be updated to provide access to the appropriate yum repositories and channels as described in Enabling Access to the Ceph Storage for Oracle Linux Packages.
2. From the administration node, check the Ceph Storage Cluster status with the following command:

   ```
   # ceph health
   ```

   If the `health` of the cluster is `HEALTH_OK`, continue with the upgrade. If not, make sure all cluster nodes are stable and healthy.

   Do not create any new erasure-code pools while upgrading the monitors.

   You can monitor the progress of your upgrade at any time using the `ceph versions` command, which lists the Ceph version for each daemon.
3. From the administration node, set the `sortbitwise` option to change the internal sorting algorithm. This option is used by the new object enumeration API and for the new BlueStore backend:

   ```
   # ceph osd set sortbitwise
   ```
4. From the administration node, set the `noout` option to prevent the CRUSH algorithm from attempting to rebalance the cluster during the upgrade:

   ```
   # ceph osd set noout
   ```
5. Edit the Ceph configuration file and **remove** the workaround required in Ceph Storage for Oracle Linux Release 2.0:

   ```
   rbd default features = 3
   ```
6. From the deployment node, push the updated Ceph configuration file to all the nodes in the cluster, for example:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy --overwrite-conf config push ceph-node2 ceph-
   node3 ceph-node4
   ```
7. From the deployment node (usually the same as the administration node), upgrade the `ceph-deploy` package:

   ```
   # yum update ceph-deploy
   ```
8. On the deployment node, use `ceph-deploy` to upgrade the Ceph Command Line Interface (CLI) on the administration node, for example:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy admin ceph-node1
   ```

In this example, *ceph-node1* is the host name of the client system configured to manage the cluster using the Ceph CLI. This node is also used as the deployment node in examples.

9. From the deloyment node, use `ceph-deploy` to upgrade the packages on each node in the Ceph Storage Cluster for example:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy install ceph-node2 ceph-node3 ceph-node4
   ```

   > **Note:**
   >
   > The upstream documentation mentions the `--release` switch, which is meant to allow you to control which release you are updating to; however, this switch does not have any effect when used in an Oracle Linux environment, and the packages are simply installed to the latest version using yum.

10. On each Ceph Monitor node, restart the Ceph Monitor daemon:

    ```
    # systemctl restart ceph-mon.target
    ```

    When all Ceph Monitor daemons are upgraded, on the administration node, verify the monitor upgrade is complete:

    ```
    # ceph mon feature ls
    ...
    on current monmap (epoch 1)
        persistent: [kraken,luminous]
        required: [kraken,luminous]
    ```

    The output should include `luminous` under `persistent` features in the `monmap` list.

11. From the deployment node, gather the monitor keys and the OSD and MDS bootstrap keyrings from one of the Ceph Monitor nodes, for example:

    ```
    # $CEPH_CONFIG_DIR/ceph-deploy gatherkeys ceph-node2
    ```

12. From the deployment node, deploy new Ceph Manager daemons on each node that runs Ceph Monitor daemons, for example:

    ```
    # $CEPH_CONFIG_DIR/ceph-deploy --overwrite-conf mgr create ceph-node2 ceph-
    node3 ceph-node4
    ```

    On the administration node, you can verify the Ceph Manager daemons are running with the command:

    ```
    # ceph -s
    ...
      services:
        mon: 3 daemons, quorum ceph-node2,ceph-node3,ceph-node4
        mgr: ceph-node2(active), standbys: ceph-node3, ceph-node4
        osd: 3 osds: 3 up, 3 in
    ...
    ```

    This command lists the `mon`, `mgr` and `osd` daemons.

    You can also verify the `ceph-mgr` daemon is running on each Ceph Manager node. For example, on a node named `ceph-node2`, which includes the Ceph Manager daemon::

    ```
    [ceph-node2 ~]# systemctl status ceph-mgr@ceph-node2
    ● ceph-mgr@ceph-node2.service - Ceph cluster manager daemon
    ```

```
   Loaded: loaded (/usr/lib/systemd/system/ceph-mgr@.service; enabled; vendor
preset: disabled)
   Active: active (running) since <date>
 Main PID: 4902 (ceph-mgr)
   CGroup: /system.slice/system-ceph\x2dmgr.slice/ceph-mgr@ceph-node2.service
           └─4902 /usr/bin/ceph-mgr -f --cluster ceph --id ceph-node2 --setuser ceph
--setgroup ceph...
...
```

13. On each OSD node, restart the `ceph-osd` daemon:

    ```
    # systemctl restart ceph-osd.target
    ```

    From the administration node, you can monitor the progress of the OSD upgrades with the `ceph osd versions` command, for example:

    ```
    # ceph osd versions
    {
       "ceph version 12.2.4 (...) luminous (stable)": 2,
       "ceph version 10.2.6 (...)": 1,
    }
    ```

    The output shows two OSDs upgraded to Luminous, and one OSD not yet upgraded.

14. On each node that runs the Ceph Metadata Server daemon, restart the `ceph-mds` daemon:

    ```
    # systemctl restart ceph-mds.target
    ```

15. On each node that runs the Ceph Object Gateway daemon, restart the `radosgw` daemon:

    ```
    # systemctl restart radosgw.target
    ```

16. On the administration node, prevent old (pre-Luminous) OSD nodes from joining the Ceph Storage Cluster:

    ```
    # ceph osd require-osd-release luminous
    ```

17. Check the Ceph Storage Cluster status with the following command:

    ```
    # ceph health
    ```

    If the health of the cluster is `HEALTH_OK`, re-enable the CRUSH algorithm's ability to balance the cluster by unsetting the `noout` option:

    ```
    # ceph osd unset noout
    ```

18. (Optional) BlueStore is the default backend for new OSDs in Ceph Storage for Oracle Linux Release 3.0. The FileStore backend can be used with upgraded OSDs and a mixture of BlueStore and FileStore OSDs is supported. If you want to migrate OSDs from FileStore to BlueStore, see the upstream documentation at:

    https://docs.ceph.com/en/latest/rados/operations/bluestore-migration/

3 Using Ceph Storage for Oracle Linux

WARNING:

Oracle Linux 7 is now in Extended Support. See Oracle Linux Extended Support and Oracle Open Source Support Policies for more information.

Migrate applications and data to Oracle Linux 8 or Oracle Linux 9 as soon as possible.

This chapter discusses setting up a Ceph storage pool, block device, block device image and setting up a Ceph iSCSI Gateway to access block devices. This chapter also discusses using the Ceph Object Gateway, and the Ceph File System.

Setting Up a Ceph Block Device

Ceph block devices are thin-provisioned, resizable and store data striped over multiple OSDs in a Ceph Storage Cluster. Ceph block devices leverage RADOS capabilities such as snapshotting, replication and consistency. Ceph's RADOS Block Devices (RBD) interact with OSDs using kernel modules, or the librbd library. You can also use Kernel Virtual Machines (KVMs) to host Ceph block devices.

This section discusses setting up a node to use to create a Ceph storage pool, and to host a Ceph block device.

This section also discusses setting up and using the Ceph iSCSI Gateway to enable access to clients, such as Microsoft Windows, to the Ceph block devices.

Creating and Removing Ceph Block Devices and Storage Pools

This section discusses setting up a node to use to create a Ceph storage pool, and to host a Ceph block device. The node used in this section is referred to as a *Ceph Client*. The Ceph Client must have the Ceph CLI installed in order to create the Ceph storage pool and Ceph block device. For information on installing the Ceph CLI, see Creating a Ceph Storage Cluster Administration Host.

Creating a Block Device and Storage Pool

This section discusses creating a Ceph storage pool and an associated Ceph block device. Make sure the cluster is active and healthy before configuring a block device. The steps shown here should be perfomed on the Ceph Client node.

To create a storage pool and associated block device:

1. On a Ceph Client, create a storage pool using the following command:

   ```
   # ceph osd pool create pool_name pg_num pgp_num
   ```

For example, create a pool named `datastore`, with 128 placement groups for the storage pool (*pg_num*), and 128 placement groups to be considered for placement by the CRUSH algorithm (*pgp_num*). The *pgp_num* value should be equal to the *pg_num* value.

```
# ceph osd pool create datastore 128 128
```

Data durability and even distribution among all OSDs requires more placement groups, but creating too many may utilize unnecessary CPU and memory. It is important to calculate appropriate values for the placement groups for the number of OSDs in your environment. See the Ceph upstream documentation for more information on Ceph placement groups, and how to calculate the values required when creating a storage pool.

Tip:

You can use the upstream Ceph PGs per Pool Calculator to calculate the placement group sizes for your pool based on the number of OSDs in your environment.

2. Associate the storage pool with the `rbd` application.

```
# ceph osd pool application enable datastore rbd
```

3. Initialize the pool with the `rbd` application.

```
# rbd pool init datastore
```

4. Use the `rbd` command to create a block device image in the storage pool, for example:

```
# rbd create --size 4096 --pool datastore vol01
```

This example creates a 4096 MB volume named `vol01` in the `datastore` pool.

If you do not specify a storage pool, `rbd` uses the default `rbd` pool:

```
# rbd create --size 4096 vol01
```

5. Set the RBD features you want to enable or disable for the block device. You must disable the `object-map`, `fast-diff`, and `deep-flatten` features as they are not supported in UEK R5. For example, run:

```
# rbd feature disable datastore/vol01 object-map fast-diff deep-flatten
```

6. Use the `rbd` command to map the image to a block device, for example:

```
# rbd map vol01 --pool datastore
```

Ceph creates the block device under `/dev/rbd/`*`pool`*`/`*`volume`* .

The RBD kernel module is not loaded until you run this command. You can check that it is loaded after running the command by doing the following:

```
# lsmod|grep rbd
                                rbd      73304   1
 libceph  235751  2  rbd,ceph
```

The `rbd ls` command lists the images that you have mapped for a storage pool, for example:

```
# rbd ls -p datastore
                               vol01
```

7. Create a file system on the block device, for example:

```
# mkfs.xfs /dev/rbd/datastore/vol01
```

Oracle recommends using Btrfs or XFS as the file system type for Oracle workloads.

8. Mount this file system, for example:

```
# mkdir /var/vol01
# mount /dev/rbd/datastore/vol01 /var/vol01
```

Removing a Block Device and its Storage Pool

This section discusses removing a Ceph storage pool and an associated Ceph block device. The steps shown here should be perfomed on the Ceph Client node.

To remove a block device and its associated storage pool:

1. On the Ceph Client node, unmount any file system that is using the block device, for example:

   ```
   # umount /var/vol01
   ```

2. Unmap the block device from its image, for example:

   ```
   # rbd unmap /dev/rbd/datastore/vol01
   ```

3. Remove the block device image, for example:

   ```
   # rbd rm vol01 -p datastore
   ```

4. Remove the storage pool, for example:

   ```
   # ceph osd pool delete datastore datastore --yes-i-really-really-mean-it
   ```

   If you have not allowed the removal of a storage pool in the Ceph configuration, you can temporarily allow it and return to the original configuration using the following commands:

   ```
   # ceph tell mon.* injectargs --mon-allow-pool-delete=true
   # ceph osd pool delete datastore datastore --yes-i-really-really-mean-it
   # ceph tell mon.* injectargs --mon-allow-pool-delete=false
   ```

Setting Up Ceph iSCSI Gateway

This section contains information on setting up and using the Ceph iSCSI Gateway. The iSCSI gateway provides a High Availability iSCSI target that exports Ceph block device images as SCSI disks. This allows for clients, such as Microsoft Windows, to access the Ceph Storage Cluster. It is recommended to use two to four iSCSI gateway nodes for Ceph iSCSI Gateway High Availability.

A Ceph iSCSI Gateway node can be a standalone node or be co-located on a Ceph OSD node.

More detailed information on the Ceph iSCSI Gateway is available in the upstream documentation at:

https://docs.ceph.com/en/latest/rbd/iscsi-overview/

Setting Up Ceph iSCSI Gateway Nodes

This section shows you how to set up one or more Ceph iSCSI Gateway nodes. You can set up a Ceph iSCSI Gateway node on an OSD node, or on a non-OSD node. The steps shown here are for setting up Ceph iSCSI Gateway nodes on OSD nodes. For information on setting up gateway nodes when they are not on an OSD node, see the upstream documentation.

To set up an Ceph iSCSI Gateway node (co-located on an OSD node):

1. On each Ceph iSCSI Gateway node, open TCP ports 3260 and 5000:

   ```
   # firewall-cmd --permanent --zone=public --add-port=3260/tcp --add-
   port=5000/tcp
   # firewall-cmd --reload
   # systemctl restart firewalld.service
   ```

2. Lower the default timers for detecting down OSDs to reduce the possibility of iSCSI initiator timeouts. This should be performed for every OSD node in the cluster. There are a number of methods to change the configuration.

   - Make the configuration change in the Ceph configuration file. Add the following to the *$CEPH_CONFIG_DIR*/`ceph.conf` file:

     ```
     [osd]
     osd heartbeat grace = 20
     osd heartbeat interval = 5
     ```

     Push the configuration change to all OSD nodes. To push the configuration to multiple nodes, provide them in a space separated list, for example:

     ```
     # $CEPH_CONFIG_DIR/ceph-deploy --overwrite-conf config push ceph-node2
     ceph-node3 ceph-node4
     ```

     On each OSD node, restart the `ceph-osd` daemon:

     ```
     # systemctl restart ceph-osd.target
     ```

   - Make the configuration change at runtime by sending the change to each OSD in the cluster from the administration node. The changes made using this method are persistent. For example, run:

     ```
     # ceph tell osd.0 config set osd_heartbeat_grace 20
     # ceph tell osd.0 config set osd_heartbeat_interval 5
     ```

     Replace `0` with the ID of an OSD. Use the `ceph osd ls` command on the administration node to get a list of all OSD IDs in the cluster.

     The configuration changes using this method are made to the osd.*0* process and are enabled immediately. You do not need to restart the OSD service on nodes when you use this method.

   - Make the configuration change at runtime on each OSD node. The changes made using this method are persistent.

     ```
     # ceph daemon osd.0 config set osd_heartbeat_grace 20
     # ceph daemon osd.0 config set osd_heartbeat_interval 5
     ```

     Replace `0` with the ID of the OSD. Use the `ceph osd tree` command on the administration node to get a list of all OSD IDs in the cluster and the nodes on which they reside.

     The configuration changes using this method are made to the osd.*0* process, but may not be enabled until you restart the OSD service. To enable the changes, on each OSD node, restart the `ceph-osd` daemon:

     ```
     # systemctl restart ceph-osd.target
     ```

3. From the deployment node, install the Ceph CLI (`ceph` command) and add the keyring on the Ceph iSCSI Gateway node. For example, run:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy admin ceph-node1 ceph-node2
   ```

   For more information on installing the Ceph CLI on a node, see Creating a Ceph Storage Cluster Administration Host.

4. From the administration node, check if a storage pool exists with the name `rbd`.

   ```
   # ceph osd lspools
   ```

   If the `rbd` pool does not exist, create it. For example, run:

   ```
   # ceph osd pool create rbd 128 128
   ```

   Associate the `rbd` pool with the RBD application. For example, run:

   ```
   # ceph osd pool application enable rbd rbd
   ```

   Initialize the `rbd` pool with the RBD application. For example, run:

   ```
   # rbd pool init rbd
   ```

5. On each Ceph iSCSI Gateway node, install the required packages:

   ```
   # yum install ceph-iscsi-cli tcmu-runner
   ```

6. On a Ceph iSCSI Gateway node, create a file named `iscsi-gateway.cfg` in the `/etc/ceph/` directory and add the following to the file. Edit the contents of the file to suit your environment.

   ```
   [config]
   # Name of the Ceph storage cluster. A suitable Ceph configuration file allowing
   # access to the Ceph storage cluster from the gateway node is required, if not
   # colocated on an OSD node.
   cluster_name = ceph

   # Place a copy of the ceph cluster's admin keyring in the gateway's /etc/ceph
   # directory and reference the filename here
   gateway_keyring = ceph.client.admin.keyring

   # API settings.
   # The API supports a number of options that allow you to tailor it to your
   # local environment. If you want to run the API under https, you will need to
   # create cert/key files that are compatible for each iSCSI gateway node, that is
   # not locked to a specific node. SSL cert and key files *must* be called
   # 'iscsi-gateway.crt' and 'iscsi-gateway.key' and placed in the '/etc/ceph/'
   directory
   # on *each* gateway node. With the SSL files in place, you can use 'api_secure =
   true'
   # to switch to https mode.

   # To support the API, the bear minimum settings are:
   api_secure = false

   # Additional API configuration options are as follows, defaults shown.
   # api_user = admin
   # api_password = admin
   # api_port = 5001
   trusted_ip_list = 192.168.0.10,192.168.0.11
   ```

Replace *trusted_ip_list* with a comma separated list of the IP addresses for each Ceph iSCSI Gateway node. These IP addresses are used for management operations like target creation, LUN exporting, and so on.

Copy the `/etc/ceph/iscsi-gateway.cfg` file to all Ceph iSCSI Gateway nodes in the cluster.

7. On each Ceph iSCSI Gateway node in the cluster, enable and start the API service:

```
# systemctl daemon-reload
# systemctl enable rbd-target-api
# systemctl start rbd-target-api
```

You can check the status of the API service on a node using:

```
# systemctl status rbd-target-api
```

Configuring iSCSI Targets

This section shows you how to set up and configure the iSCSI targets on Ceph iSCSI Gateway nodes.

To configure iSCSI targets:

1. Use the iSCSI gateway command line interface (`gwcli`) to configure iSCSI targets and RBD images. On a Ceph iSCSI Gateway node, start gwcli:

```
# gwcli
```

2. Change to the `iscsi-target` directory and create a target with the name `iqn.1988-12.com.oracle.iscsi-gw:iscsi-igw`:

```
> /> cd /iscsi-target
> /iscsi-target> create iqn.1988-12.com.oracle.iscsi-gw:iscsi-igw
```

3. Change to the `gateways` directory and create the iSCSI gateways.

```
> /iscsi-target> cd iqn.1988-12.com.oracle.iscsi-gw:iscsi-igw/gateways
> /iscsi-target...-igw/gateways> create gateway_hostname ip_address
```

In this example, *gateway_hostname* is the host name for the gateway node and *ip_address* is an IP used for iSCSI data like READ and WRITE commands. These IPs can be the same as used for management operations listed in `trusted_ip_list` in the `iscsi-gateway.cfg` file, but it is recommended you use different IP addresses here.

For example, to create two iSCSI gateways, run:

```
> /iscsi-target...-igw/gateways> create ceph-node1 203.0.113.150
> /iscsi-target...-igw/gateways> create ceph-node2 203.0.113.151
```

You can list the gateways (and other objects you create) using the `ls` command:

```
> /iscsi-target...-igw/gateways> ls
```

4. Add an RBD image to the `rbd` storage pool, for example:

```
> /iscsi-target...-igw/gateways>  cd /disks
> /disks> create pool=rbd image=mydisk-1 size=90G
```

5. Create a client initiator. The client initiator name used in this example is `iqn.1988-12.com.oracle:ol7-client`:

```
> /disks> cd /iscsi-target/iqn.1988-12.com.oracle.iscsi-gw:iscsi-igw/hosts
> /iscsi-target...csi-igw/hosts> create iqn.1988-12.com.oracle:ol7-client
```

6. Set the client's CHAP username and password:

   ```
   > /iscsi-target...le:ol7-client> auth chap=cephiscsiuser/cephiscsipass
   ```

7. Add the disk to the client. For example, run:

   ```
   > /iscsi-target...le:ol7-client> disk add rbd.mydisk-1
   ```

Configuring iSCSI Initiators

This section shows you how to configure iSCSI initiators to access the iSCSI targets on Ceph iSCSI Gateway nodes. The client node does not need to be part of the Ceph Storage Cluster; it is a client outside the cluster accessing the Ceph iSCSI Gateway.

To configure iSCSI Initiators:

1. On an iSCSI client, install the iSCSI initiator and multipath tools:

   ```
   # yum install iscsi-initiator-utils device-mapper-multipath
   ```

2. Create the default `/etc/multipath.conf` file and enable the multipathd service:

   ```
   # mpathconf --enable --with_multipathd y
   ```

3. Add the following to `/etc/multipath.conf` file:

   ```
    devices {
              device {
                      vendor                 "LIO-ORG"
                      hardware_handler       "1 alua"
                      path_grouping_policy   "failover"
                      path_selector          "queue-length 0"
                      failback               60
                      path_checker           tur
                      prio                   alua
                      prio_args              exclusive_pref_bit
                      fast_io_fail_tmo       25
                      no_path_retry          queue
              }
       }
   ```

4. Restart the multipathd service:

   ```
   # systemctl reload multipathd
   ```

5. Set up the iSCSI target. Edit the `/etc/iscsi/iscsid.conf` file and add the CHAP login credentials, for example:

   ```
   node.session.auth.authmethod = CHAP
   node.session.auth.username = cephiscsiuser
   node.session.auth.password = cephiscsipass
   ```

   Edit the `/etc/iscsi/initiatorname.iscsi` file and replace the content with the client initiator name you created on the target. In this example it is `iqn.1988-12.com.oracle:ol7-client`.

   ```
   InitiatorName=iqn.1988-12.com.oracle:ol7-client
   ```

6. Enable and start iSCSI client service:

   ```
   # systemctl restart iscsid.service
   # systemctl enable iscsid.service
   ```

7. Discover the target portals. For example, run:

   ```
   # iscsiadm --mode discovery --type sendtargets --portal 203.0.113.150
   ```

8. Log into the target:

```
$ iscsiadm -m node -T  iqn.1988-12.com.oracle.iscsi-gw:iscsi-igw -l
```

9. The multipath daemon (multipathd) automatically sets up devices based on the settings in the `/etc/multipath.conf` file. Running the `multipath` command shows devices set up in a failover configuration with a priority group for each path.

```
# multipath -ll
```

   You can now use the RBD image as you would any other multipath iSCSI disk.

Setting Up Ceph Object Gateway

A Ceph Object Gateway provides a REST interface to the Ceph Storage Cluster to facilitate Amazon S3 and OpenStack Swift client access. The Ceph Object Gateway is described in more detail in the upstream documentation.

There are two deployment configuration options for Ceph Object Gateways, depending on requirements:

- **Simple Ceph Object Gateway**

  A simple Ceph Object Gateway configuration is used where the Ceph Object Storage service runs in a single data center and there is no requirement to define regions and zones.

- **Multi-site Ceph Object Gateway**

  A multi-site Ceph Object Gateway configuration is used where the Ceph Object Storage service is geographically distributed within a federated architecture and provides the facility to configure storage for different regions and to further distinguish separate zones per region. Data synchronization agents allow the service to maintain multiple copies of the data across a widely distributed environment, helping to provide better data resilience for failover, backup and disaster recovery. The feature facilitates the ability to write to non-master zones and maintain synchronization of data changes between different zones.

Setting Up Simple Ceph Object Gateway

The Ceph Object Gateway is a client of the Ceph Storage Cluster, but may be hosted on a node within the cluster if required. The Ceph Object Gateway has the following requirements:

- A running Ceph Storage Cluster
- A public facing network interface that allows traffic on the network port used to serve HTTP or HTTPS requests (7480 by default)
- A name for the Ceph Object Gateway instance
- A username with appropriate permissions in a keyring
- Pools to store its data
- A data directory for the gateway instance
- An instance entry in the Ceph configuration file

The following sections describe installation and deployment steps to get you started using Ceph Object Gateway.

Installing the Simple Ceph Object Gateway

To install the Ceph Object Gateway software on a node within your Ceph Storage Cluster, you can run the following command from the deployment node within your environment:

```
# $CEPH_CONFIG_DIR/ceph-deploy install --rgw ceph-node1
```

Substitute *ceph-node1* with the resolvable host name of the node where you wish to install the software. Note that the target node must have the appropriate Yum channels configured, as described in Enabling Access to the Ceph Storage for Oracle Linux Packages.

To create a Ceph Object Gateway within the Ceph configuration, use the following command:

```
# $CEPH_CONFIG_DIR/ceph-deploy --overwrite-conf rgw create ceph-node1
```

Updating Firewall Requirements

If you are running a firewall service, make sure that the port where the Ceph Object Gateway is running is open. For example, run:

```
# firewall-cmd --zone=public --add-port 7480/tcp --permanent
# firewall-cmd --reload
```

Note that if you change the default port for the Ceph Object Gateway at a later stage, you may need to repeal this firewall rule and add a new rule for the new port number.

Creating Users and Keys

Before you are able to use the Ceph Object Gateway, users must be created to allow access to the different APIs exposed by the gateway.

To create a user for S3 access, run the following command on the gateway node:

```
# radosgw-admin user create --uid="testuser" --display-name="First User"
```

The command returns JSON formatted output describing the newly created user. This output includes a listing of the keys that are automatically generated when you create a new user. Take note of the `access_key` and `secret_key` for the user that you have created. You require these when connecting to the gateway from an S3 client application.

If you wish to provide access to the Swift API, a subuser can be created against the original user. To create a user for Swift access, run the following command on the gateway host:

```
# radosgw-admin subuser create --uid=testuser --subuser=testuser:swift \
    --access=full --access-key=$SYSTEM_ACCESS_KEY --secret=$SWIFT_KEY --default
--key-type=swift
```

Details for the user are displayed. The `swift_keys` key in the JSON output displays a `user` and `secret_key` that can be used for Swift access validation.

At any point, if you need to see this user information again to obtain keys or to check other information, such as permissions, you can use the `radosgw-admin user info` command. For example, run:

```
# radosgw-admin user info --uid=testuser
```

You can list users with the `radosgw-admin user list` command.

Testing Access

To test S3 access, you require the `python-boto` package and you must create a simple Python script that can be used to create a new bucket. The Amazon S3 API uses the term *bucket* to describe a data container. The term *bucket* is the equivalent of the term *container*. A bucket can hold a collection of data objects.

1. Install the `python-boto` package if it is not already installed:

   ```
   # yum install python-boto
   ```

2. Create a Python script that can be used to test S3 access. Using a text editor, create a file called `s3test.py` and insert the following code:

   ```
   #!/usr/bin/env python
   import boto
   import boto.s3.connection

   access_key = 'SZUP3NC5P7452N1HQT4B'
   secret_key = 'v0Ok4YK0MtcSZURk6vwCwRtQnB3vAW2G8TjrAlIj'
   conn = boto.connect_s3(
           aws_access_key_id = access_key,
           aws_secret_access_key = secret_key,
           host = 'ceph-node1.example.com', port = 7480,
           is_secure=False, calling_format = boto.s3.connection.OrdinaryCallingFormat(),
           )

   bucket = conn.create_bucket('my-bucket')
   for bucket in conn.get_all_buckets():
               print "{name} {created}".format(
                       name = bucket.name,
                       created = bucket.creation_date,
               )
   ```

   Replace the `access_key` value, *SZUP3NC5P7452N1HQT4B*, with the access key for the *testuser* user that you created for S3 access. Replace the `secret_key` value, *v0Ok4YK0MtcSZURk6vwCwRtQnB3vAW2G8TjrAlIj*, with the secret key that was generated for the *testuser* user that you created for S3 access. Replace *ceph-node1.example.com* with the host name or fully qualified domain name where the gateway host is located. Replace the port number *7480*, if you have configured an alternate port to the default.

   If you want to create multiple buckets in this script, add them to the bucket array, for example:

   ```
   ...

   bucket = conn.create_bucket('my-bucket-1')
   bucket = conn.create_bucket('my-bucket-2')
   bucket = conn.create_bucket('my-bucket-3')
   bucket = conn.create_bucket('my-bucket-4')
   for bucket in conn.get_all_buckets():
               print "{name} {created}".format(
                       name = bucket.name,
                       created = bucket.creation_date,
               )
   ```

3. Change permissions on the script so that you can run the script:

   ```
   # chmod 776 ./s3test.py
   ```

4. Run the script:

```
# ./s3test.py my-bucket 2018-06-05T04:05:10.130Z
```

The script should return the name of the new bucket and the date and timestamp for when it was created.

If you want to list all buckets, use the `radosgw-admin bucket list` command.

If you need to test Swift access, install the Swift command-line client and use the secret key that was generated for the subuser that you created for this purpose.

1. Install the `python-swiftclient` package and its dependencies:

   ```
   # yum install python-swiftclient
   ```

2. Run the client from the command line, providing the appropriate credentials and connection information:

   ```
   # swift -A http://ceph-node1.example.com:7480/auth/1.0 -U testuser:swift \
      -K '2DHaQknPsc5XsYEmHQ0mWCGLnoGnaCr4VUd62czm' list my-bucket
   ```

Replace *ceph-node1.example.com* with the host name or fully qualified domain name where the gateway host is located. Replace the port number *7480*, if you have configured an alternate port to the default. Replace *testuser:swift*, with the *testuser* user subuser that you created for Swift access. Replace *2DHaQknPsc5XsYEmHQ0mWCGLnoGnaCr4VUd62czm*, with the secret Swift key that was generated for the Swift subuser. Run `man swift` for more information about this command and the options available.

The command should list any existing buckets within Ceph, including the bucket created when you tested S3 access.

Changing Port Numbering

The port number that is used for the Ceph Object Gateway HTTP interface can be updated or changed in the Ceph configuration file on the deployment node. To change the port number used by the Ceph Object Gateway, edit the `$CEPH_CONFIG_DIR/ceph.conf` file on the deployment node. Add the following lines to the end of the configuration file:

```
[client.rgw.ceph-node1]
rgw_frontends = "civetweb port=80"
```

Replace *ceph-node1* with the resolvable host name that you used when you deployed the gateway. Replace *80* with the port number that you wish to use for the HTTP port.

To push the configuration change to the gateway node and to the other nodes in the cluster, run the following command:

```
# $CEPH_CONFIG_DIR/ceph-deploy --overwrite-conf config push ceph-node2
```

To push the configuration to multiple nodes, provide them in a space separated list, for example:

```
# $CEPH_CONFIG_DIR/ceph-deploy --overwrite-conf config push ceph-node2 ceph-
node3 ceph-node4
```

On the gateway node, restart the Ceph Object Gateway for the settings to take effect:

```
# systemctl restart ceph-radosgw@*
```

Enabling Transport Layer Security

Oracle recommends using certificates signed by a recognized Certificate Authority (CA) in production environments. In a test environment, you can create and use a self-signed certificate. Certificates must be in PEM format (X.509v3).

To create a self-signed certificate:

1. To enable Transport Layer Security (TLS) on the Ceph Object Gateway service, you must install the OpenSSL packages on the gateway host, if they are not installed already:

   ```
   # yum install -y openssl mod_ssl
   ```

2. In the working directory where you are generating the key and certificate, create a copy of the template OpenSSL configuration file:

   ```
   # cp /etc/pki/tls/openssl.cnf ./
   ```

3. Modify the configuration file template at `./openssl.cnf` and make the following changes:

   a. In the section `[ req ]` make sure that the following line is uncommented and not preceded with a # character:

      ```
      req_extensions = v3_req # The extensions to add to a certificate request
      ```

   b. In the section `[ v3_req ]`, add the following line to the end of the parameters in this section:

      ```
      subjectAltName = @alt_names
      ```

   c. Add a section to the end of the configuration file:

      ```
      [ alt_names ]
      DNS.1 = hostname.example.com
      ```

      Replace *hostname.example.com* with the fully qualified domain name for the host that you are creating the certificate for.

4. Generate your certificate key, as normal:

   ```
   # openssl genrsa -out hostname.example.com.key 2048
   ```

5. Use the certificate key and the new `openssl.cnf` file to create a Certificate Signing Request (CSR):

   ```
   # openssl req -new -key hostname.example.com.key \
   -out hostname.example.com.csr -extensions v3_req -config openssl.cnf
   ```

6. You may either use the generated CSR to obtain a signed certificate from a recognized Certificate Authority (CA). Or, for testing purposes, you may use this to generate a self-signed certificate as follows:

   a. Create a new configuration file, `v3.cnf`, that can host the information for the v3 requirements. Edit it to contain the following lines:

      ```
      [v3_req]
      subjectAltName = @alt_names
      [alt_names]
      DNS.1 = hostname.example.com
      ```

   b. Run the following OpenSSL command to generate a self-signed certificate using the CSR and your local key:

```
# openssl x509 -req -days 365 -in hostname.example.com.csr -signkey
hostname.example.com.key \
-out hostname.example.com.crt -extensions v3_req -extfile v3.cnf
```

7. Copy the key, CSR and certificate to the usable location on the host:

   ```
   # cp -f hostname.example.com.crt /etc/pki/tls/certs/
   # cp -f hostname.example.com.csr /etc/pki/tls/private/
   # cp -f hostname.example.com.key /etc/pki/tls/private/
   ```

8. Create a single PEM file containing both the key and certificate, that can be used by the Ceph Object Gateway when it is started:

   ```
   # cp hostname.example.com.crt hostname.example.com.pem
   # cat hostname.example.com.key >> hostname.example.com.pem
   # cp hostname.example.com.pem /etc/pki/tls/
   ```

9. If you use a self-signed certificate, you may encounter TLS certificate verification or validation errors when you attempt to access the service in TLS mode, particularly when using the example Python script provided in this document.

   If you choose to use a self-signed certificate, you can copy the CA certificate to the client system's certificate bundle to avoid any errors. For example, run:

   ```
   # cat custom.crt >> /etc/pki/tls/certs/ca-bundle.crt
   ```

   Alternatively, use the client program or script's environment to specify the path to additional trusted CA certificates in PEM format. The environment variables `SSL_CERT_FILE` and `SSL_CERT_DIR` can be used to specify additional trusted CA certificates. For example, run:

   ```
   # SSL_CERT_FILE=/root/ceph/custom.pem python script.py
   ```

   Note:

   Oracle does not recommend the use of self-signed certificates in production environments.

The certificate in PEM format should be copied into the `/etc/pki/tls/` directory on each node in the cluster.

Update the `$CEPH_CONFIG_DIR/ceph.conf` file on the deployment node of your cluster. If there is an existing entry for `[client.rgw.gateway]`, modify it to look similar to the following example. Alternatively add an entry that looks similar to the following:

```
[client.rgw.ceph-node1]
rgw_frontends = "civetweb port=443s ssl_certificate=/etc/pki/tls/ceph-
node1.example.com.pem
```

Replace *ceph-node1* with the resolvable host name that you used when you deployed the gateway. Replace *443* with the port number that you wish to use for the HTTPS port. Note that the port number must have the letter `s` affixed to indicate to the embedded Civetweb web server that HTTPS should be used on this port. Replace */etc/pki/tls/ceph-node1.example.com.pem* with the path to a PEM formatted file that contains both the certificate and key file.

To push the configuration change to the gateway node and to the other nodes in the cluster, run the following command:

```
# $CEPH_CONFIG_DIR/ceph-deploy --overwrite-conf config push ceph-node2
```

To push the configuration to multiple nodes, provide them in a space separated list, for example:

```
# $CEPH_CONFIG_DIR/ceph-deploy --overwrite-conf config push ceph-node2 ceph-node3 ceph-node4
```

On the gateway node, restart the Ceph Object Gateway for the settings to take effect:

```
# systemctl restart ceph-radosgw@*
```

If you are running firewall software on the gateway node, make sure that a rule exists to allow traffic on the port defined in your configuration file. For instance:

```
 # firewall-cmd --zone=public --add-port=443/tcp --permanent
```

Setting Up Multi-site Ceph Object Gateway

A multi-site Ceph Object Gateway configuration can be deployed to achieve synchronization between different zones within a zone group. The multi-site configuration replaces the federated Ceph Object Gateway configuration described in previous releases of Ceph.

Oracle has tested a multi-site configuration consisting of a single zone group containing multiple zones distributed across separate Ceph Storage Clusters. No sync agent is configured to mirror data changes between gateways, which allows for a simpler active-active configuration. All metadata operations, such as the creation of new users, must be made via the master zone, however data operations, such as the creation of buckets and objects can be handled by any zone in the deployment.

The following configuration steps describe a basic multi-site configuration consisting of two Ceph Storage Clusters in a single zone group containing three separate zones that actively sync data between them. The example setup is deployed on three servers with local storage available. It is assumed that the systems do not have an existing Ceph configuration and that they have access to the Ceph packages and their dependencies, as described in Enabling Access to the Ceph Storage for Oracle Linux Packages.

The following naming conventions are used in the example configuration:

- Realm: `gold`
- Master Zone Group: `us`
- Master Zone: `us-east-1`
- Secondary Zone: `us-east-2`
- Secondary Zone: `us-west`

The zones `us-east-1` and `us-east-2` are part of the same Ceph Storage Cluster. The zone `us-west` is installed on a second Ceph Storage Cluster.

Setting Up the First Ceph Storage Cluster

1. Install the `ceph-deploy` tool on one of the systems that is intended to be part of the first cluster:

   ```
   # yum install ceph-deploy
   ```

   This system is referred to as the 'first cluster deployment node' through the rest of these instructions.

2. Create a clean working Ceph configuration directory for the Ceph Storage Cluster and change to this directory, for example:

   ```
   # rm -rf /var/mydom_ceph
   # mkdir /var/mydom_ceph
   # cd /var/mydom_ceph
   ```

   The `/var/mydom_ceph` directory created here is referred to as *`$CEPH_CONFIG_DIR`* in the remaining commands in this example.

3. Clear the systems of any pre-existing Ceph configuration information or data:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy purge ceph-node1 ceph-node2
   # $CEPH_CONFIG_DIR/ceph-deploy purgedata ceph-node1 ceph-node2
   ```

   Replace the node names with the host names of the systems taking part in the Ceph Storage Cluster.

4. Deploy the Ceph Storage Cluster configuration:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy new ceph-node1 ceph-node2
   ```

   Replace the node names with the host names of the systems taking part in the Ceph Storage Cluster.

5. Update the configuration template with required configuration variables. For example, run:

   ```
   # echo "osd pool default size = 2" >> ceph.conf
   ```

6. Install the Ceph packages on the nodes:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy install ceph-node1 ceph-node2
   ```

   Replace the node names with the host names of the systems taking part in the Ceph Storage Cluster.

7. Deploy a Ceph Monitor on one or more of the nodes:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy mon create-initial
   # $CEPH_CONFIG_DIR/ceph-deploy mon create ceph-node1 ceph-node2
   # $CEPH_CONFIG_DIR/ceph-deploy gatherkeys ceph-node1
   ```

   Replace the node names with the host names of the nodes to designate as a Ceph Monitor nodes.

8. Deploy a Ceph Manager on the nodes that are running Ceph Monitor:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy mgr create ceph-node1 ceph-node2
   ```

   Replace the node names with the host names of the nodes to designate as a Ceph Manager nodes.

9. Prepare an available disk on each node to function as an Object Storage Device (OSD):

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy disk zap ceph-node1 /dev/sdb
   # $CEPH_CONFIG_DIR/ceph-deploy osd create --data /dev/sdb ceph-node1
   # $CEPH_CONFIG_DIR/ceph-deploy disk zap ceph-node2 /dev/sdc
   # $CEPH_CONFIG_DIR/ceph-deploy osd create --data /dev/sdc ceph-node2
   ```

   Replace the node name with the host names of the systems taking part in the Ceph Storage Cluster. Replace *sdb* and *sdc* with the appropriate device names for available disks on each host. Note that these disks are repartitioned and formatted, destroying any existing data on them.

For more information on creating OSD nodes, see Creating an Object Storage Device (OSD) Node.

10. Check the status of the Ceph Cluster to make sure that the Ceph Storage Cluster is healthy and that the OSDs are available. On the administration node, enter:

```
# ceph status
```

Setting Up Ceph Object Gateway Instances on the First Ceph Storage Cluster

1. From the first Ceph Storage Cluster deployment node, install the Ceph Object Gateway software on each of the nodes in the cluster:

```
# $CEPH_CONFIG_DIR/ceph-deploy install --rgw ceph-node1 ceph-node2
# $CEPH_CONFIG_DIR/ceph-deploy rgw create ceph-node1 ceph-node2
```

   Replace the node names with the host names of the systems taking part in the Ceph Storage Cluster to install the Ceph Object Gateway software.

2. Edit the template configuration in `$CEPH_CONFIG_DIR/ceph.conf` on the first Ceph Storage Cluster deployment node and add the following lines to the end of the configuration file:

```
[client.rgw.ceph-node1]
rgw_frontends = "civetweb port=80"

[client.rgw.ceph-node2]
rgw_frontends = "civetweb port=80"
```

   Replace *ceph-node1* and *ceph-node2* with the host names of the gateway systems.

3. Push the configuration to each of the nodes in the Ceph Storage Cluster, for example:

```
# $CEPH_CONFIG_DIR/ceph-deploy --overwrite-conf config push ceph-node1 ceph-
node2
```

4. On *each* of the nodes, restart the Ceph Object Gateway service and check its status to ensure that it is running correctly:

```
# systemctl restart ceph-radosgw@*
# systemctl status ceph-radosgw@*
```

Creating the Required Pools on the First Ceph Storage Cluster

Ceph Object Gateways require several pools to store gateway related data. Where gateways are configured as zones, it is typical to create pools particular to a zone using the naming convention: `zone.pool-name`. For this reason, it is best to manually create all of the required pools for each of the zones within the Ceph Storage Cluster.

On the first Ceph Storage Cluster deployment node, run the commands to create all of the pools required for the zones that are hosted in this cluster. For example, run:

```
# ceph osd pool create ceph-us-east-1.rgw.control 4 4
# ceph osd pool create ceph-us-east-1.rgw.data.root 4 4
# ceph osd pool create ceph-us-east-1.rgw.gc 4 4
# ceph osd pool create ceph-us-east-1.rgw.log 4 4
# ceph osd pool create ceph-us-east-1.rgw.intent-log 4 4
# ceph osd pool create ceph-us-east-1.rgw.usage 4 4
# ceph osd pool create ceph-us-east-1.rgw.users.keys 4 4
# ceph osd pool create ceph-us-east-1.rgw.users.email 4 4
# ceph osd pool create ceph-us-east-1.rgw.users.swift 4 4
# ceph osd pool create ceph-us-east-1.rgw.users.uid 4 4
# ceph osd pool create ceph-us-east-1.rgw.buckets.index 8 8
# ceph osd pool create ceph-us-east-1.rgw.buckets.data 64 64
# ceph osd pool create ceph-us-east-1.rgw.meta 4 4

# ceph osd pool create ceph-us-east-2.rgw.control 4 4
# ceph osd pool create ceph-us-east-2.rgw.data.root 4 4
# ceph osd pool create ceph-us-east-2.rgw.gc 4 4
# ceph osd pool create ceph-us-east-2.rgw.log 4 4
# ceph osd pool create ceph-us-east-2.rgw.intent-log 4 4
# ceph osd pool create ceph-us-east-2.rgw.usage 4 4
# ceph osd pool create ceph-us-east-2.rgw.users.keys 4 4
# ceph osd pool create ceph-us-east-2.rgw.users.email 4 4
# ceph osd pool create ceph-us-east-2.rgw.users.swift 4 4
# ceph osd pool create ceph-us-east-2.rgw.users.uid 4 4
# ceph osd pool create ceph-us-east-2.rgw.buckets.index 8 8
# ceph osd pool create ceph-us-east-2.rgw.buckets.data 64 64
# ceph osd pool create ceph-us-east-2.rgw.meta 4 4
```

Tip:

You can use the upstream Ceph PGs per Pool Calculator to calculate the page sizes for your pool based on the number of OSDs in your environment.

Creating System Keys

While configuring zones, each gateway instance requires a system user with credentials set up to allow for S3-like access. This allows each gateway instance to pull the configuration remotely using the access and secret keys. To make sure that the same keys are configured on each gateway instance, it is best to define these keys beforehand and to set them manually when the zones and users are created.

It is good practice to set these as reusable environment variables while you are setting up your configuration and to randomize the keys as much as possible:

```
# SYSTEM_ACCESS_KEY=$(cat /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w 20 | head
-n 1)
# SYSTEM_SECRET_KEY=$(cat /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w 40 | head
-n 1)
```

You can check that these keys are set and contain good content:

```
# echo SYSTEM_ACCESS_KEY=$SYSTEM_ACCESS_KEY
# echo SYSTEM_SECRET_KEY=$SYSTEM_SECRET_KEY
```

Keep a record of the output from these commands. You need to export the same environment variables when you set up the second Ceph Storage Cluster and the secondary zone located here.

Configuring the Realm and Master Zone

The multi-site configuration is built around a single realm, named `gold`, with a single zone group called `us`. Within this zone group is a master zone named `ceph-us-east-1`. The following steps describe what must be done to create and configure these components:

1. Create the realm and make it the default:

```
# radosgw-admin realm create --rgw-realm=gold --default
```

2. Delete the default zone group which is created as part of the simple installation of the Ceph Object Gateway software.

   ```
   # radosgw-admin zonegroup delete --rgw-zonegroup=default
   ```

3. Create a new master zone group. The master zone group is in control of the zone group map and propagates changes across the system. This zone group should be set as the default zone group to allow you to run commands for it in future without having to explicitly identify is using the `--rgw-zonegroup` switch.

   ```
   # radosgw-admin zonegroup create --rgw-zonegroup=us \
    --endpoints=http://ceph-node1.example.com:80 --master --default
   ```

4. Create the master zone and make it the default zone. Note that for metadata operations, such as user creation, you must use this zone. You can also add the zone to the zone group when you create it, and specify the access and secret key that should be used for this zone:

   ```
   # radosgw-admin zone create --rgw-zonegroup=us --rgw-zone=ceph-us-east-1 \
    --endpoints=http://ceph-node1.example.com:80 --access-
   key=$SYSTEM_ACCESS_KEY \
    --secret=$SYSTEM_SECRET_KEY --default --master
   ```

5. Create a system user that can be used to access the zone pools. The keys for this user must match the keys used by each each of the zones that are being configured:

   ```
   # radosgw-admin user create --uid=zone.user --display-name="ZoneUser" \
     --access-key=$SYSTEM_ACCESS_KEY --secret=$SYSTEM_SECRET_KEY --system
   ```

6. The realm period holds the entire configuration structure for the current state of the realm. When realm information, such as configuration for the zone groups and zones, is modified, the changes must be updated for the period. This is achieved by committing the changes:

   ```
   # radosgw-admin period update --commit
   ```

Configuring the Secondary Zone

The following commands can be executed on the first Ceph Storage Cluster deployment node, but are used to update the zone group and realm configuration to add the secondary zone hosted on the other node within the cluster.

1. Create the secondary zone, making sure that you specify the same access and secret key as used for the master zone:

   ```
   # radosgw-admin zone create --rgw-zonegroup=us --rgw-zone=ceph-us-east-2 \
       --endpoints=http://ceph-node2.example.com:80 \
       --access-key=$SYSTEM_ACCESS_KEY --secret=$SYSTEM_SECRET_KEY
   ```

2. Update the realm period with the new configuration information:

   ```
   # radosgw-admin period update --commit
   ```

Updating Ceph Configuration and Restarting the Gateways

Edit the template configuration in the working directory on the first Ceph Storage Cluster deployment node to map the zone names to each gateway configuration. This is done by adding a line for the `rgw_zone` variable to the gateway configuration entry for each node:

```
[client.rgw.ceph-node1]
rgw_frontends = "civetweb port=80"
```

```
rgw_zone=ceph-us-east-1

[client.rgw.ceph-node2]
rgw_frontends = "civetweb port=80"
rgw_zone=ceph-us-east-2
```

When you have updated the template configuration, push the changes to each of the nodes in the Ceph Storage Cluster:

```
# $CEPH_CONFIG_DIR/ceph-deploy --overwrite-conf config push ceph-node1 ceph-
node2
```

On *each* of the nodes, restart the Ceph Object Gateway service and check its status to ensure that it is running correctly:

```
# systemctl restart ceph-radosgw@*
# systemctl status ceph-radosgw@*
```

Setting Up the Second Ceph Storage Cluster

Install and deploy a second Ceph Storage Cluster in much the same way as you did the first. In this example, the second cluster consists of a single node, although you may add more nodes if you require. This node ultimately hosts the gateway for the `ceph-us-west` zone.

The following commands, recap on the steps to deploy the Ceph Storage Cluster and to configure an OSD that can be used for storage. These commands must be issued on a new server, *`ceph-node3`*, outside of the first cluster:

```
# mkdir -p /var/mydom_ceph; cd /var/mydom_ceph
# yum install ceph-deploy
# $CEPH_CONFIG_DIR/ceph-deploy new ceph-node3
# $CEPH_CONFIG_DIR/echo "osd pool default size = 2" >> ceph.conf
# $CEPH_CONFIG_DIR/ceph-deploy install ceph-node3
# $CEPH_CONFIG_DIR/ceph-deploy mon create-initial
# $CEPH_CONFIG_DIR/ceph-deploy mon create ceph-node3
# $CEPH_CONFIG_DIR/ceph-deploy mgr create ceph-node3
# $CEPH_CONFIG_DIR/ceph-deploy gatherkeys ceph-node3
# $CEPH_CONFIG_DIR/ceph-deploy disk zap ceph-node3 /dev/sdb
# $CEPH_CONFIG_DIR/ceph-deploy osd create --data /dev/sdb ceph-node3
```

Setting Up the Ceph Object Gateway Instance on the Second Ceph Storage Cluster

1. Install the Ceph Object Gateway software on the newly deployed node in the Ceph Storage Cluster:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy install --rgw ceph-node3
   # $CEPH_CONFIG_DIR/ceph-deploy rgw create ceph-node3
   ```

   Replace *ceph-node3* with the host name of the node where you wish to install the Ceph Object Gateway software.

2. Edit the template configuration in `$CEPH_CONFIG_DIR/ceph.conf` on the second Ceph Storage Cluster deployment node and add the following lines to the end of the configuration file:

   ```
   [client.rgw.ceph-node3]
   rgw_frontends = "civetweb port=80"
   ```

   Replace *ceph-node3* with the host name of the gateway system.

3. Push the configuration to each of the nodes in the Ceph Storage Cluster:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy --overwrite-conf config push ceph-node3
   ```

4. Restart the Ceph Object Gateway service on the gateway node and check its status to ensure that it is running correctly:

   ```
   # systemctl restart ceph-radosgw@*
   # systemctl status ceph-radosgw@*
   ```

Creating the Required Pools on the Second Ceph Storage Cluster

Create the required pools for the Ceph Object Gateway on the second Ceph Storage Cluster by running the following commands:

```
# ceph osd pool create ceph-us-west.rgw.control 4 4
# ceph osd pool create ceph-us-west.rgw.data.root 4 4
# ceph osd pool create ceph-us-west.rgw.gc 4 4
# ceph osd pool create ceph-us-west.rgw.log 4 4
# ceph osd pool create ceph-us-west.rgw.intent-log 4 4
# ceph osd pool create ceph-us-west.rgw.usage 4 4
# ceph osd pool create ceph-us-west.rgw.users.keys 4 4
# ceph osd pool create ceph-us-west.rgw.users.email 4 4
# ceph osd pool create ceph-us-west.rgw.users.swift 4 4
# ceph osd pool create ceph-us-west.rgw.users.uid 4 4
# ceph osd pool create ceph-us-west.rgw.buckets.index 8 8
# ceph osd pool create ceph-us-west.rgw.buckets.data 64 64
# ceph osd pool create ceph-us-west.rgw.meta 4 4
```

Updating Realm Configuration

1. Export the same `SYSTEM_ACCESS_KEY` and `SYSTEM_SECRET_KEY` environment variables that you set up on the first Ceph Storage Cluster. For example, run:

   ```
   # SYSTEM_ACCESS_KEY=OJywnXPrAA4uSCgv1UUs
   # SYSTEM_SECRET_KEY=dIpf1FRPwUYcXfswYx6qjC0eSuHEeHy0I2f9vHFf
   ```

2. Using these keys, pull the realm configuration directly from the first Ceph Storage Cluster, via the node running the master zone, by issuing the following command:

   ```
   # radosgw-admin realm pull --url=http://ceph-node1.example.com:80 \
     --access-key=$SYSTEM_ACCESS_KEY --secret=$SYSTEM_SECRET_KEY
   ```

3. Pull the period state directly from the first Ceph Storage Cluster, via the node running the master zone:

   ```
   # radosgw-admin period pull --url=http://ceph-node1.example.com:80 \
     --access-key=$SYSTEM_ACCESS_KEY --secret=$SYSTEM_SECRET_KEY
   ```

4. Set the default realm for the gateway instance to `gold`:

   ```
   # radosgw-admin realm default --rgw-realm=gold
   ```

5. Set the default zone group to `us`:

   ```
   # radosgw-admin zonegroup default --rgw-zonegroup=us
   ```

Configuring the Secondary Zone

1. Create the new secondary zone, *ceph-us-west*, and add it to the *us* zone group. Make sure that when you create the zone you use the same access and secret keys as were used on the original configuration on the first Ceph Storage Cluster:

   ```
   # radosgw-admin zone create --rgw-zonegroup=us --rgw-zone=ceph-us-west \
    --endpoints=http://ceph-node3.example.com:80 \
    --default --access-key=$SYSTEM_ACCESS_KEY --secret=$SYSTEM_SECRET_KEY
   ```

2. Commit the zone group changes to update the period state:

   ```
   # radosgw-admin period update --commit --rgw-zone=ceph-us-west
   ```

3. Edit the template configuration in the working directory at `$CEPH_CONFIG_DIR/ceph.conf` to map the zone names to the gateway configuration. This is done by adding a line for the `rgw_zone` variable to the gateway configuration entry:

   ```
   [client.rgw.ceph-node3]
   rgw_frontends = "civetweb port=80"
   rgw_zone=ceph-us-west

   [client.rgw.ceph-node2]
   rgw_frontends = "civetweb port=80"
   rgw_zone=ceph-us-east-2
   ```

4. When you have updated the template configuration, push the changes to each of the nodes in the Ceph Storage Cluster:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy --overwrite-conf config push ceph-node3
   ```

   Restart the Ceph Object Gateway service and check its status to ensure that it is running correctly:

   ```
   # systemctl restart ceph-radosgw@*
   # systemctl status ceph-radosgw@*
   ```

Testing Zone Synchronization

At this point all zones should be running and synchronizing.

To test synchronization, you can create a bucket in any of the zones and then list the buckets on any alternative zone. You should discover that the newly created bucket is visible within any of the zones.

This test can be performed using a simple Python script. Copy the example script, included below, into a file called `$HOME/s3zone_test.py` on any host that is able to access each of the nodes where the zones are running:

```
#!/usr/bin/env python

import boto
import boto.s3.connection
from optparse import OptionParser

parser = OptionParser()
parser.add_option("--access_key", dest="access_key", default="OJywnXPrAA4uSCgv1UUs")
parser.add_option("--secret_key", dest="secret_key",
default="dIpf1FRPwUYcXfswYx6qjC0eSuHEeHy0I2f9vHFf")
parser.add_option("-H","--host", dest="host", default="ceph-node1.example.com")
parser.add_option("-s","--secure",dest="is_secure", action="store_true", default=False)
```

```
parser.add_option("-c","--create", dest="bucket")

(options,args) = parser.parse_args()

conn = boto.connect_s3(
 aws_access_key_id = options.access_key,
 aws_secret_access_key = options.secret_key,
 host = options.host,
 is_secure=options.is_secure,
 calling_format = boto.s3.connection.OrdinaryCallingFormat(),
)
if options.bucket:
    bucket = conn.create_bucket(options.bucket)

for bucket in conn.get_all_buckets():
        print "{name}\t{created}".format(
                name = bucket.name,
                created = bucket.creation_date,
)
```

Substitute the default values for the `access_key`, `secret_key` and `host` in the script to better suit your own environment.

To test the script, make sure that it is executable by modifying the permissions on the script:

```
# chmod 775 $HOME/s3zone_test.py
```

Check that you have all the appropriate libraries installed to properly run the script:

```
# yum install python-boto
```

You can use the options in the script to set different variables such as which zone host you wish to run the script against, and to determine whether or not to create a new bucket. Create a new bucket in the first zone by running the following command:

```
# $HOME/s3zone_test.py --host ceph-node1 -c my-bucket-east-1
my-bucket-east-1 2016-09-21T09:16:14.894Z
```

Now check the other two nodes to see that the new bucket is synchronized:

```
# $HOME/s3zone_test.py --host ceph-node2
my-bucket-east-1        2016-09-21T09:16:16.932Z
# $HOME/s3zone_test.py --host ceph-node3
my-bucket-east-1        2016-09-21T09:16:15.145Z
```

Note that the timestamps are different due to the time that it took to synchronize the data. You may also test creating a bucket in the zone located on the second Ceph Storage Cluster:

```
# $HOME/s3zone_test.py --host ceph-node3 -c my-bucket-west-1
my-bucket-east-1        2016-09-21T09:16:15.145Z
my-bucket-west-1        2016-09-21T09:22:15.456Z
```

Check that this bucket is synchronized into the other zones:

```
# $HOME/s3zone_test.py --host ceph-node1
my-bucket-east-1        2016-09-21T09:16:14.894Z
my-bucket-west-1        2016-09-21T09:22:15.428Z
# $HOME/s3zone_test.py --host ceph-node2
my-bucket-east-1        2016-09-21T09:16:16.932Z
my-bucket-west-1        2016-09-21T09:22:17.488Z
```

Configuring Transport Layer Security

Oracle recommends using certificate signed by a recognized Certificate Authority (CA) in production environments. In a test environment, you can create and use a self-signed certificate. Certificates must be in PEM format (X.509v3).

When configuring Transport Layer Security (TLS) for a multi-site Ceph Object Gateway deployment, it is critical that each zone is capable of validating and verifying the certificates. This means that if you choose to use self-signed certificates, each zone must have a copy of all of the certificates already within its recognized CA bundle. Alternatively, make sure that you use certificates signed by a recognized Certificate Authority.

The steps provided in this example show how to create self-signed certificates for each gateway node, how to share the generated certificates between the nodes to ensure that they can be validated and how to change the existing multi-site configuration to enable TLS.

To set up TLS:

1. Create certificates for each gateway node in the deployment.

   On each gateway node, run the following commands:

   ```
   # cd /etc/ceph
   # openssl genrsa -out ca.key 2048
   # openssl req -new -key ca.key -out ca.csr -subj \
       "/C=US/ST=California/L=RedWoodShore/O=Oracle/OU=Linux/CN=`hostname`"
   # openssl x509 -req -days 365 -in ca.csr -signkey ca.key -out ca.crt
   # cp -f ca.crt /etc/pki/tls/certs/`hostname`.crt
   # cp -f ca.key /etc/pki/tls/private/`hostname`.key
   # cp -f ca.csr /etc/pki/tls/private/`hostname`.csr
   # cp ca.crt /etc/pki/tls/`hostname`.pem
   # cat ca.key >> /etc/pki/tls/`hostname`.pem
   ```

   You may replace the values for the certificate subject line with values that are more appropriate to your organization, but ensure that the CommonName (CN) of the certificate resolves to the host name that you use for the endpoint URLs in your zone configuration.

2. Copy the certificates to from each gateway node to the collection of recognized CAs in `/etc/pki/tls/certs/ca-bundle.crt` on each node.

   For example, on *ceph-node1*, run:

   ```
   # cat /etc/ceph/ca.crt >> /etc/pki/tls/certs/ca-bundle.crt
   # cat /etc/ceph/ca.crt | ssh root@ceph-node2 "cat >> /etc/pki/tls/certs/ca-
   bundle.crt"
   # cat /etc/ceph/ca.crt | ssh root@ceph-node3 "cat >> /etc/pki/tls/certs/ca-
   bundle.crt"
   ```

   On *ceph-node2*:

   ```
   # cat /etc/ceph/ca.crt >> /etc/pki/tls/certs/ca-bundle.crt
   # cat /etc/ceph/ca.crt | ssh root@ceph-node1 "cat >> /etc/pki/tls/certs/ca-
   bundle.crt"
   # cat /etc/ceph/ca.crt | ssh root@ceph-node3 "cat >> /etc/pki/tls/certs/ca-
   bundle.crt"
   ```

   On *ceph-node3*:

   ```
   # cat /etc/ceph/ca.crt >> /etc/pki/tls/certs/ca-bundle.crt
   # cat /etc/ceph/ca.crt | ssh root@ceph-node1 "cat >> /etc/pki/tls/certs/ca-
   ```

```
bundle.crt"
# cat /etc/ceph/ca.crt | ssh root@ceph-node2 "cat >> /etc/pki/tls/certs/ca-
bundle.crt"
```

3. If you are running a firewall service on any of the nodes in your environment, make sure that traffic is permitted on port 443. For example, run:

   ```
   # firewall-cmd --zone=public --add-port=443/tcp --permanent
   # systemctl restart firewalld.service
   ```

4. Modify the existing zone group information to use HTTPS to access any zone endpoints.

   To do this run the following commands:

   ```
   # radosgw-admin zonegroup get|sed -r 's/http(.*):80/https\1:443/g' >/tmp/
   zonegroup.json
   # radosgw-admin zonegroup set --infile /tmp/zonegroup.json
   # radosgw-admin period update --commit
   ```

5. Redeploy the Ceph Object Gateway on each node in your environment, to reset any previous configuration and to ensure that the nodes are deployed using the full host name matching the CN used for the certificates.

   Run the following command from the first Ceph Storage Cluster deployment node:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy --overwrite-conf rgw create ceph-
   node1.example.com \
       ceph-node2.example.com
   ```

   Substitute *ceph-node1.example.com* and *ceph-node2.example.com* with the full host names of the nodes that are running the gateway software, so that these match the CN used on their certificates.

   Run the following command from the second Ceph Storage Cluster deployement node:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy --overwrite-conf rgw create ceph-
   node3.example.com
   ```

   Substitute *ceph-node3.example.com* with the full host name of the node that is running the gateway software, so that it matches the CN used on the certificate that you generated for this node.

   At this point, all of the gateway services should have restarted running on the default port 7480.

6. On the deployment node on each Ceph Storage Cluster, edit the template configuration to change the port number and to identify the location of the TLS certificate PEM file for each gateway.

   For example, on *ceph-node1*, edit `$CEPH_CONFIG_DIR/ceph.conf` and modify the gateway configuration entries. Make sure that the full host name is used in the entry label and that you modify the port and add an entry pointing to the SSL certificate path:

   ```
   ...
   osd pool default pg num = 100
   osd pool default pgp num = 100
   mon pg warn max per osd = 2100

   [client.rgw.ceph-node1.example.com]
   rgw_frontends = "civetweb port=443s ssl_certificate=/etc/pki/tls/ceph-
   node1.example.com.pem"
   rgw_zone=ceph-us-east-1

   [client.rgw.ceph-node2.example.com]
   ```

```
rgw_frontends = "civetweb port=443s ssl_certificate=/etc/pki/tls/ceph-
node2.example.com.pem"
rgw_zone=ceph-us-east-2
```

Push the modified configuration to each gateway node:

```
# $CEPH_CONFIG_DIR/ceph-deploy --overwrite-conf config push ceph-node1 ceph-
node2
```

On *each* of the nodes, restart the Ceph Object Gateway service and check its status to ensure that it is running correctly:

```
# systemctl restart ceph-radosgw@*
# systemctl status ceph-radosgw@*
```

Repeat this step on the second Ceph Storage Cluster deployment node, to update the configuration for *ceph-node3.example.com*.

7. At this point, each gateway entry should be configured to use TLS. You can test that the zones are continuing to synchronize correctly and are using TLS, by using the test script at `$HOME/s3zone_test.py`, remembering to use the `-s` switch to enable TLS.

   For example, run:

   ```
   # $HOME/s3zone_test.py -s --host ceph-node2.example.com -c my-bucket-east-2
   my-bucket-east-1        2016-09-21T09:16:16.932Z
   my-bucket-east-2        2016-09-21T14:09:51.287Z
   my-bucket-west-1        2016-09-21T09:22:17.488Z
   # $HOME/s3zone_test.py -s --host ceph-node3.example.com
   my-bucket-east-1        2016-09-21T09:16:15.145Z
   my-bucket-east-2        2016-09-21T14:09:58.783Z
   my-bucket-west-1        2016-09-21T09:22:15.456Z
   ```

   If you attempt to use the script without the `-s` switch set, the script attempts to connect without TLS on port 80 and fails to connect, ultimately terminating with a socket error:

   ```
   socket.error: [Errno 111] Connection refused
   ```

8. If you use a self-signed certificate, you may encounter TLS certificate verification or validation errors when you attempt to access the service in TLS mode, particularly when using the example Python script provided in this document.

   If you choose to use a self-signed certificate, you can copy the CA certificate to the client system's certificate bundle to avoid any errors. For example, run:

   ```
   # cat custom.crt >> /etc/pki/tls/certs/ca-bundle.crt
   ```

   Alternatively, use the client program or script's environment to specify the path to additional trusted CA certificates in PEM format. The environment variables `SSL_CERT_FILE` and `SSL_CERT_DIR` can be used to specify additional trusted CA certificates. For example, run:

   ```
   # SSL_CERT_FILE=/root/ceph/custom.pem python script.py
   ```

   Note:

   Oracle does not recommend the use of self-signed certificates in production environments.

Exporting Ceph Object Gateway Over NFS

Ceph Object Gateway namespaces can be exported over NFS using NFS-Ganesha, a user space file server that supports the NFS protocol. More information on using NFS-Ganesha with Ceph Object Gateway is available in the upstream documentation at:

https://docs.ceph.com/en/latest/radosgw/nfs/

To set up the NFS server:

1. Make sure the NFS server host is connected to the Ceph public network and part of the Ceph Storage Cluster.
2. Install the nfs-ganesha-ceph and nfs-ganesha-rgw packages:

   ```
   # yum install nfs-ganesha-ceph nfs-ganesha-rgw
   ```

3. Start the RPC bind service:

   ```
   # systemctl start rpcbind
   ```

4. Stop and disable nfs-server service:

   ```
   # systemctl stop nfs-server.service
   # systemctl disable nfs-server.service
   ```

5. Edit the `/etc/ganesha/ganesha.conf` file to suit your requirements. For example:

   ```
   EXPORT
   {
   # Export Id (mandatory, each EXPORT must have a unique Export_Id)
   Export_Id = 2;

   # Use NFSv4
   Protocols = 4;

   # NFSv4 does not allow UDP transport
   Transports = TCP;

   # Path into the cephfs tree. For now, FSAL_CEPH doesn't support
   # having more than one filesystem per running ganesha daemon.
   #
   # Note that FSAL_CEPH does not support subtree checking, so there is
   # no way to validate that a filehandle presented by a client is
   # reachable via an exported subtree.
   #
   # For that reason, we just export "/" here.
   Path = /;

   # Pseudo Path (required for NFS v4)
   # The pseudoroot path. This is where the export will appear in the
   # NFS pseudoroot namespace.
   Pseudo = /;

   # Required for access (default is None)
   # Could use CLIENT blocks instead
   Access_Type = RW;
   # Squash = no_root_squash;
   SecType = sys;

   # Exporting FSAL
   ```

```
FSAL {
    Name = RGW;
    User_Id = "user";
    Access_Key_Id ="access_key_id";
    Secret_Access_Key = "secret_access_key";
  }
}

RGW {
    ceph_conf = "/etc/ceph/ceph.conf";
    name = "client.rgw.hostname";
    cluster = "ceph";
    init_args = "-d --debug-rgw=16";
}
```

Replace the values for *user*, *access_key_id* and *secret_access_key* with the user and keys created in Setting Up Simple Ceph Object Gateway. Replace *hostname* with a node in your Ceph Storage Cluster where Ceph Object Gateway has been installed and configured. For example:

```
...
# Exporting FSAL
FSAL {
    Name = RGW;
    User_Id = "testuser";
    Access_Key_Id ="SZUP3NC5P7452N1HQT4B";
    Secret_Access_Key = "v0Ok4YK0MtcSZURk6vwCwRtQnB3vAW2G8TjrAlIj";
  }
}

RGW {
    ceph_conf = "/etc/ceph/ceph.conf";
    name = "client.rgw.ceph-node1";
    cluster = "ceph";
    init_args = "-d --debug-rgw=16";
}
```

6. Restart the nfs-ganesha service:

```
# systemctl restart nfs-ganesha.service
```

7. Make the directory mount point, if required:

```
# mkdir -p /mnt/nfs-ganesha
```

8. Mount NFS ganesha file system using the syntax:

```
mount -t nfs -o nfsvers=4.1,proto=tcp ganesha-host-name:ganesha-pseudo-path mount-
point
```

For example, run:

```
# mount -t nfs -o nfsvers=4.1,noauto,soft,sync,proto=tcp ceph-node1:/ /mnt/
nfs-ganesha
```

After setting up the NFS server with Ceph Object Gateway, you cannot write to the mount point without first creating a bucket. For information on creating a test bucket, see Testing Access.

When a bucket exists, it is listed in the mount point you created, which in this example is `/mnt/nfs-ganesha`. If you have a bucket named `my-bucket`, you see a directory named `/mnt/nfs-ganesha/my-bucket`. You can perform read/write operations on the `my-bucket` directory. For example:

```
# cd /mnt/nfs-ganesha/my-bucket/
# touch test.txt
```

Setting Up and Using Ceph FS

This section contains information on setting up and using the Ceph File System (Ceph FS). This section also contains information on exporting a Ceph file system over NFS. More detailed information on Ceph FS is available in the upstream documentation at:

https://docs.ceph.com/en/latest/cephfs/

Setting Up Ceph FS

This section contains information on setting up and using the Ceph FS.

To set up Ceph FS:

1. Deploy a Ceph Metadata Server (MDS).

   At least one MDS must be active within your environment to use Ceph FS. If you do not have a dedicated server available for this purpose, you can install the MDS service on an existing Monitor node within your Ceph Storage Cluster, as the service does not have significant resource requirements. To deploy MDS in your environment, execute the following command from your deployment node:

   ```
   # $CEPH_CONFIG_DIR/ceph-deploy mds create ceph-node3
   ```

2. Deploy the Ceph CLI.

   The Ceph CLI must be deployed on the system where you intend to mount the Ceph FS and this system must have the appropriate network access and authentication keyring to access the Ceph Storage Cluster. See Creating a Ceph Storage Cluster Administration Host for more information on deploying and setting up the Ceph CLI and keyring.

3. Create storage pools and a new Ceph file system.

   A Ceph file system requires at least two storage pools to function. The first pool is used to store actual data, while the second is used to store metadata. Although the Ceph CLI includes commands for creating and removing Ceph file systems, only one file system can exist at the one time.

   To create the storage pools, run the following commands using the Ceph CLI:

   ```
   # ceph osd pool create cephfs_data 1
   # ceph osd pool create cephfs_metadata 2
   ```

   To create the new Ceph file system, run the following command using the Ceph CLI:

   ```
   # ceph fs new cephfs cephfs_metadata cephfs_data
   ```

   To display a list of Ceph file systems, use the command:

   ```
   # ceph fs ls
   ```

4. Check the status of the Ceph MDS.

   After a file system is created, the Ceph MDS enters into an *active* state. You are only able to mount the file system once the MDS is active. To check its status:

   ```
   # ceph mds stat
   e5: 1/1/1 up {0=ceph-node3=up:active}
   ```

5. Store the secret key used for admin authentication into a file that can be used for mounting the Ceph file system.

   The Ceph Storage Cluster admin key is stored in `/etc/ceph/ceph.client.admin.keyring` on each node in the cluster and also on the administration node. When mounting a Ceph file system the key is required to authenticate the mount request. To prevent this key from being visible in the process list, it is best practice to copy it into a file that is secured with the appropriate permissions to keep this information safe.

   For example, run:

   ```
   # echo $(sed -n 's/.*key *= *\([^ ]*.*\)/\1/p' < /etc/ceph/
   ceph.client.admin.keyring) > \
       /etc/ceph/admin.secret
   # chmod 600 /etc/ceph/admin.secret
   # cat /etc/ceph/ceph.client.admin.keyring
   [client.admin]
   key = AQDIvtZXzBriJBAA+3HmoYkUmPFnKljqhxlYGw==
   # cat /etc/ceph/admin.secret
   AQDIvtZXzBriJBAA+3HmoYkUmPFnKljqhxlYGw==
   ```

Mounting Ceph FS

This section shows you how to mount a Ceph file system using the Ceph kernel module.

Note:

Mounting a Ceph file system using FUSE is not supported in this release.

To mount Ceph FS:

1. Make the directory mount point, if required:

   ```
   # mkdir -p /mnt/cephfs
   ```

2. Mount the file system:

   ```
    # mount -t ceph ceph-node2:6789:/ /mnt/cephfs/ \
       -o name=admin,secretfile=/etc/ceph/admin.secret
   ```

   Replace *ceph-node2* with the host name or IP address of a Ceph Monitor node within your Ceph Storage Cluster. Multiple monitor addresses can be specified, separated by commas, although only one active monitor is needed to successfully mount the file system. If you do not know what Monitors are available, you can run `ceph mon stat` to get a listing of available Monitors, their IP addresses and port numbers.

   Mounting a Ceph file system automatically loads the `ceph` and `libceph` kernel modules.

3. To unmount the file system:

   ```
   # umount /mnt/cephfs
   ```

Exporting Ceph FS Over NFS

Ceph FS namespaces can be exported over NFS using NFS-Ganesha, a user space file server with support for the NFS protocol. More information on using NFS-Ganesha to export Ceph FS over NFS is available in the upstream documentation at:

https://docs.ceph.com/en/latest/cephfs/nfs/

To set up the NFS server:

1. Make sure the NFS server host is connected to the Ceph public network and part of the Ceph Storage Cluster.
2. Install the libcephfs2, nfs-ganesha and nfs-ganesha-ceph packages:

   ```
   # yum install libcephfs2 nfs-ganesha nfs-ganesha-ceph
   ```

3. Edit the `/etc/ganesha/ganesha.conf` file to suit your requirements, for example:

   ```
   EXPORT
   {
   # Export Id (mandatory, each EXPORT must have a unique Export_Id)
   Export_Id = 1;

   # Use NFSv4
   Protocols = 4;

   # NFSv4 does not allow UDP transport
   Transports = TCP;

   # Path into the cephfs tree. For now, FSAL_CEPH doesn't support
   # having more than one file system per running ganesha daemon.
   #
   # Note that FSAL_CEPH does not support subtree checking, so there is
   # no way to validate that a filehandle presented by a client is
   # reachable via an exported subtree.
   #
   # For that reason, we just export "/" here.
   Path = /;

   # Pseudo Path (required for NFS v4)
   # The pseudoroot path. This is where the export will appear in the
   # NFS pseudoroot namespace.
   Pseudo = /mnt/cephfs;

   # Required for access (default is None)
   # Could use CLIENT blocks instead
   Access_Type = RW;
        Squash = no_root_squash;
   SecType = sys;

   # Exporting FSAL
   FSAL {
     Name = CEPH;
     }
   }
   ```

4. Restart the nfs-ganesha service:

   ```
   # systemctl restart nfs-ganesha.service
   ```

5. Mount the Ceph file system. See Mounting Ceph FS for information on mounting a Ceph file system.

Mounting Ceph FS over NFS

This section shows you how to mount a Ceph file system on an Oracle Linux client using NFS. Follow the steps in Exporting Ceph FS Over NFS to set up a Ceph file system over NFS before you mount it.

To mount Ceph FS over NFS:

1. If not installed, install the nfs-utils package:

   ```
   # yum install nfs-utils
   ```

2. Make the directory mount point, if required:

   ```
   # mkdir -p /mnt/nfs-ganesha
   ```

3. Mount the NFS ganesha file system using the syntax:

   ```
   mount -t nfs -o nfsvers=4.1,proto=tcp ganesha-host-name:ganesha-pseudo-path mount-
   point
   ```

   For example, run:

   ```
   # mount -t nfs -o nfsvers=4.1,noauto,soft,sync,proto=tcp ceph-node2:/mnt/
   cephfs /mnt/nfs-ganesha
   ```

4 Known Issues

WARNING:

Oracle Linux 7 is now in Extended Support. See Oracle Linux Extended Support and Oracle Open Source Support Policies for more information.

Migrate applications and data to Oracle Linux 8 or Oracle Linux 9 as soon as possible.

The following sections describe known issues in this release.

`ceph-deploy` tool not compatible with previous releases

The `ceph-deploy` tool provided in this release is not compatible with previous Ceph Storage for Oracle Linux releases. Using the `ceph-deploy` provided in this release to manage deployments from previous releases may result in errors and unexpected behavior. Make sure you upgrade your environment using the steps set out in Upgrading Ceph Storage Cluster Nodes to avoid using the latest `ceph-deploy` tool with older deployments.

(Bug 28049396)

`ceph-deploy purge` command does not clean up OSD disk volumes or labels

By design, the `ceph-deploy purge` command does not remove existing OSD volume groups or labels. This is to avoid unintentional data destruction. After purging an OSD, the volume remains in use, and any attempt to reuse it results in an error similar to the following:

```
# ceph-deploy osd create ceph-1 --data /dev/sdb
...
# ceph-deploy purge ceph-1
...
# ceph-deploy osd create ceph-1 --data /dev/sdb
...
[DEBUG ]  stderr: Can't open /dev/sdb exclusively.  Mounted filesystem?
[DEBUG ] --> Was unable to complete a new OSD, will rollback changes
[DEBUG ] --> OSD will be fully purged from the cluster, because
the ID was generated
[DEBUG ] Running command: ceph osd purge osd.0 --yes-i-really-mean-it
[DEBUG ]  stderr: purged osd.0
[ERROR ] RuntimeError: command returned non-zero exit status: 1
[ceph_deploy.osd][ERROR ] Failed to execute command: /usr/sbin/ceph-volume
--cluster ceph lvm create --bluestore --data /dev/sdb
[ceph_deploy][ERROR ] GenericError: Failed to create 1 OSDs
```

Workaround: Remove the volume group and volume label used for the OSD on the disk. The disk can then be used to create a new volume the next time you use the `ceph-deploy osd create` command. For example, to remove a volume label, use the `lvdisplay` command to list the volume labels, and delete the label:

```
# lvdisplay
...
# lvremove --force /dev/ceph-dc39f7cc-e423-48d3-a466-9701e7bf972a/osd-block-
f7db38d2-...
```

Use the `vgdisplay` command to list the volume groups, and delete the group:

```
# vgdisplay
...
# vgremove --force ceph-dc39f7cc-e423-48d3-a466-9701e7bf972a
```

(Bug 27748402)

Some RBD features not supported in UEK R5

An error may be displayed when mapping an image to a block device using the RBD kernel module, for example:

```
# rbd map vol01 --pool datastore
rbd: sysfs write failed
RBD image feature set mismatch. You can disable features unsupported by the
kernel with "rbd feature disable datastore/vol01 object-map fast-diff
deep-flatten".
In some cases useful info is found in syslog - try "dmesg | tail".
rbd: map failed: (6) No such device or address
```

UEK R5 does not support all RBD features. Unsupported features must be disabled.

Workaround: Disable the `object-map`, `fast-diff`, and `deep-flatten` RBD features *before* mapping an image to a block device, for example:

```
# rbd create --size 4096 --pool datastore vol01
# rbd feature disable datastore/vol01 object-map fast-diff deep-flatten
# rbd map vol01 --pool datastore
```

(Bug 28028199)

Ceph Object Gateway does not support HTTP and HTTPS concurrently

Although upstream documentation suggests that it is possible to configure the Ceph Object Gateway to support both HTTP and HTTPS at the same time, by specifying two ports concatenated with the `+` symbol and by appending the `s` character to the port number where SSL/TLS should be used, this functionality does not work properly and only the first port specified is used.

Only configure for one protocol at a time, until this issue is resolved.

(Bug 24422558)

NFS exports using nfs-ganesha are read-only if SELinux enabled

If SELinux is enabled, NFS exports using nfs-ganesha are read-only. This affects Ceph FS and Ceph Object Gateway over NFS using nfs-ganesha.

Workaround: Disable SELinux. Edit the `/etc/sysconfig/selinux` file and change `SELINUX=enforcing` to `SELINUX=disabled`. Reboot the host.

(Bug 28036218)

TLS SecurityWarning: Certificate has no subjectAltName

When you configure a Ceph Object Gateway instance and enable Transport Layer Security (TLS) you must create or use a certificate. If the certificate does not have the v3 extension enabled and the subjectAltName set within the certificate, a warning message is displayed when a client such as the Swift client attempts to access the gateway:

```
/usr/lib/python2.7/site-packages/urllib3/connection.py:251: SecurityWarning:
Certificate has no `subjectAltName`, falling back to check for a `commonName`
for now. This feature is being removed by major browsers and deprecated by
RFC 2818. (See https://github.com/shazow/urllib3/issues/497 for details.)
```

If a `subjectAltName` extension of type `dNSName` is present, this is used as the identity. Otherwise, the `Common Name` field in the `Subject` field of the certificate is used. Although the use of the Common Name is existing practice, it is deprecated and Certification Authorities are encouraged to use the dNSName instead.

Workaround: Use a certificate in PEM format with the v3 extension enabled (X.509v3). See Enabling Transport Layer Security and Configuring Transport Layer Security for information on creating and using self-signed certificates.

(Bug 24424028)

5

Ceph Terminology

WARNING:

Oracle Linux 7 is now in Extended Support. See Oracle Linux Extended Support and Oracle Open Source Support Policies for more information.

Migrate applications and data to Oracle Linux 8 or Oracle Linux 9 as soon as possible.

Block Device

A Ceph component that provides access to Ceph storage as a thinly provisioned block device. When an application writes to a Block Device, Ceph implements data redundancy and enhances I/O performance by replicating and striping data across the Ceph Storage Cluster.

Also known as a *RADOS Block Device* or *RBD*.

BlueStore

The default storage backend for Ceph OSDs.

Ceph Client

A host that can contains the Ceph Command Line Interface libraries to configure and deploy a Ceph Storage Cluster. A Ceph Client need not be a member node of a Ceph Storage Cluster.

Also known as *administration node*, or *admin node*.

Ceph Manager (ceph-mgr)

A Ceph component used by external monitoring tools to report on nodes in a Ceph Storage Cluster. Runs on the same nodes that run Ceph Monitor on Ceph Storage Cluster nodes

Also known as a *Manager*.

Ceph Monitor (ceph-mon)

A Ceph component used for tracking active and failed nodes in a Ceph Storage Cluster.

Also known as a *Monitor* or *MON*.

Ceph OSD (ceph-osd)

A Ceph component that provides access to an OSD.

Also known as a *Ceph OSD Daemon*.

Ceph Storage Cluster

A Ceph component that stores MON and OSD data across cluster nodes.

Also known as a *Ceph Object Store*, *RADOS Cluster*, or *Reliable Autonomic Distributed Object Store*.

Node

A host system that is a member of a Ceph Storage Cluster.

Object Gateway

A Ceph component that provides a RESTful gateway that can use the Amazon S3 and OpenStack Swift compatible APIs to present OSD data to Ceph Clients, OpenStack, and Swift clients. An Object Gateway is configured on a node of a Ceph Storage Cluster.

Also known as a *RADOS Gateway* or *RGW*.

Object Storage Device (OSD)

Storage on a physical device or logical unit (LUN). The default management system is BlueStore. Oracle recommends data on an OSD is configured as a Btrfs file system to take advantage of its snapshot features. However, XFS and BlueStore can also be used.