Red Hat Ceph Storage 1.2.3 Installation Guide for Ubuntu (x86 64)

Installing Calamari and Ceph Storage on Ubuntu x86_64.

Red Hat Customer Content Services

Red Hat Ceph Storage 1.2.3 Installation Guide for Ubuntu (x86 64)

Installing Calamari and Ceph Storage on Ubuntu x86_64.

Legal Notice

Abstract

This document provides instructions for preparing nodes before installation, for downloading Red Hat Ceph Storage, for setting up a local Red Hat Ceph Storage repository, for configuring Calamari, and for creating an initial Ceph Storage Cluster on Ubuntu Precise and Ubuntu Trusty.

Table of Contents

PART I. INSTALLATION

Designed for cloud infrastructures and web-scale object storage, Red Hat® Ceph Storage is a massively scalable, open, software-defined storage platform that combines the most stable version of Ceph with a Ceph management platform, deployment tools, and support services. Providing the tools to flexibly and cost-effectively manage petabyte-scale data deployments in the enterprise, Red Hat Ceph Storage manages cloud data so enterprises can focus on managing their businesses.

This document provides procedures for installing Red Hat Ceph Storage v1.2.3 for x86_64 architecture on Ubuntu Precise and Ubuntu Trusty.

To simplify installation and to support deployment scenarios where security measures preclude direct Internet access, Red Hat Ceph Storage v1.2.3 is installed from a single software build delivered as an ISO with the **`ice_setup`** package, which installs the **`ice_setup`** script. When you execute the **`ice_setup`** script, it will install a local repository, the Calamari monitoring and administration server and the Ceph installation scripts, including a **`cephdeploy.conf`** file pointing **`ceph-deploy`** to the local repository.

We expect that you will have a dedicated administration node that will host the local repository and the Calamari monitoring and administration server. The following instructions assume you will install (or update) the repository on the dedicated administration node.

The administration/Calamari server hardware requirements vary with the size of your cluster. A minimum recommended hardware configuration for a Calamari server includes at least 4GB of RAM, a dual core CPU on x86_64 architecture and enough network throughput to handle communication with Ceph hosts. The hardware requirements scale linearly with the number of Ceph servers, so if you intend to run a fairly large cluster, ensure that you have enough RAM, processing power and network throughput.

CHAPTER 1. PRE-INSTALLATION REQUIREMENTS

If you are installing Red Hat Ceph Storage v1.2.3 for the first time, you should review the pre-installation requirements first. Depending on your Linux distribution, you may need to adjust default settings and install required software before setting up a local repository and installing Calamari and Ceph.

1.1. OPERATING SYSTEM

Red Hat Ceph Storage v1.2.3 and beyond requires a homogeneous operating system distribution and version e.g. Ubuntu Trusty on x86_64 architecture for all Ceph nodes, including the Calamari cluster. We do not support clusters with heterogeneous operating systems and versions.

1.2. DNS NAME RESOLUTION

Ceph nodes must be able to resolve short host names, not just fully qualified domain names. Set up a default search domain to resolve short host names. To retrieve a Ceph node's short host name, execute:

```
hostname -s
```

Each Ceph node MUST be able to ping every other Ceph node in the cluster by its short host name.

1.3. NICS

All Ceph clusters require a public network. You MUST have a network interface card configured to a public network where Ceph clients can reach Ceph Monitors and Ceph OSDs. You SHOULD have a network interface card for a cluster network so that Ceph can conduct heart-beating, peering, replication and recovery on a network separate from the public network.

We DO NOT RECOMMEND using a single NIC for both a public and private network.

1.4. FIREWALL

You **MUST** adjust your firewall settings on the Calamari node to allow inbound requests on port **80** so that clients in your network can access the Calamari web user interface.

Calamari also communicates with Ceph nodes via ports **2003**, **4505** and **4506**. You **MUST** open ports **80**, **2003**, and **4505-4506** on your Calamari node.

```
sudo iptables -I INPUT 1 -i <iface> -p tcp -s <ip-address>/<netmask> --
dport 80 -j ACCEPT
sudo iptables -I INPUT 1 -i <iface> -p tcp -s <ip-address>/<netmask> --
dport 2003 -j ACCEPT
sudo iptables -I INPUT 1 -i <iface> -m multiport -p tcp -s <ip-
address>/<netmask> --dports 4505:4506 -j ACCEPT
```

You **MUST** open port **6789** on your public network on **ALL Ceph monitor nodes**.

```
sudo iptables -I INPUT 1 -i <iface> -p tcp -s <ip-address>/<netmask> --
dport 6789 -j ACCEPT
```

Finally, you **MUST** also open ports for OSD traffic (e.g., **6800-7100**). **Each OSD on each Ceph node** needs three ports: one for talking to clients and monitors (public network); one for sending data to other OSDs (cluster network, if available; otherwise, public network); and, one for heartbeating (cluster network, if available; otherwise, public network). For example, if you have 4 OSDs, open **4 x 3** ports (**12**).

```
sudo iptables -I INPUT 1 -i <iface> -m multiport -p tcp -s <ip-
address>/<netmask> --dports 6800:6811 -j ACCEPT
```

Once you have finished configuring **iptables**, ensure that you make the changes persistent on each node so that they will be in effect when your nodes reboot.

Execute:

```
sudo apt-get install iptables-persistent
```

A terminal UI will open up. Select **yes** for the prompts to save current **IPv4** iptables rules to **/etc/iptables/rules.v4** and current **IPv6** iptables rules to **/etc/iptables/rules.v6**.

The **IPv4** iptables rules that you set in the earlier steps will be loaded in **/etc/iptables/rules.v4** and will be persistent across reboots.

If you add a new **IPv4** iptables rule after installing **iptables-persistent** you will have to add it to the rule file. In such case, execute the following as a **root** user:

```
iptables-save > /etc/iptables/rules.v4
```

1.5. NTP

You MUST install Network Time Protocol (NTP) on all Ceph monitor hosts and ensure that monitor hosts are NTP peers. You SHOULD consider installing NTP on Ceph OSD nodes, but it is not required. NTP helps preempt issues that arise from clock drift.

1. Install NTP

   ```
   sudo apt-get install ntp
   ```

2. Start the NTP service and ensure it’s running.

   ```
   sudo service ntp start
   sudo service ntp status
   ```

3. Ensure that NTP is synchronizing Ceph monitor node clocks properly.

   ```
   ntpq -p
   ```

1.6. INSTALL SSH SERVER

For **ALL** Ceph Nodes perform the following steps:

1. Install an SSH server (if necessary) on each Ceph Node:

```
sudo apt-get install openssh-server
```

2. Ensure the SSH server is running on **ALL** Ceph Nodes.

1.7. CREATE A CEPH USER

The **ceph-deploy** utility must login to a Ceph node as a user that has passwordless **sudo** privileges, because it needs to install software and configuration files without prompting for passwords.

ceph-deploy supports a **--username** option so you can specify any user that has password-less **sudo** (including **root**, although this is **NOT** recommended). To use **ceph-deploy --username <username>**, the user you specify must have password-less SSH access to the Ceph node, because **ceph-deploy** will not prompt you for a password.

We recommend creating a Ceph user on **ALL** Ceph nodes in the cluster. A uniform user name across the cluster may improve ease of use (not required), but you should avoid obvious user names, because hackers typically use them with brute force hacks (e.g., **root**, **admin**, **<productname>**). The following procedure, substituting **<username>** for the user name you define, describes how to create a user with passwordless **sudo** on a node called **ceph-server**.

1. Create a user on each Ceph Node. :

```
ssh user@ceph-server
sudo useradd -d /home/<username> -m <username>
sudo passwd <username>
```

2. For the user you added to each Ceph node, ensure that the user has **sudo** privileges. :

```
echo "<username> ALL = (root) NOPASSWD:ALL" | sudo tee
/etc/sudoers.d/<username>
sudo chmod 0440 /etc/sudoers.d/<username>
```

1.8. ENABLE PASSWORD-LESS SSH

Since **ceph-deploy** will not prompt for a password, you must generate SSH keys on the admin node and distribute the public key to each Ceph node. ceph-deploy will attempt to generate the SSH keys for initial monitors.

1. Generate the SSH keys, but do not use **sudo** or the **root** user. Leave the passphrase empty:

```
ssh-keygen

Generating public/private key pair.
Enter file in which to save the key (/ceph-admin/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /ceph-admin/.ssh/id_rsa.
Your public key has been saved in /ceph-admin/.ssh/id_rsa.pub.
```

2. Copy the key to each Ceph Node, replacing **<username>** with the user name you created with Create a Ceph User_. :

```
ssh-copy-id <username>@node1
ssh-copy-id <username>@node2
ssh-copy-id <username>@node3
```

3. (Recommended) Modify the **~/.ssh/config** file of your **ceph-deploy** admin node so that **ceph-deploy** can log in to Ceph nodes as the user you created without requiring you to specify **--username <username>** each time you execute **ceph-deploy**. This has the added benefit of streamlining **ssh** and **scp** usage. Replace **<username>** with the user name you created:

```
Host node1
   Hostname node1
   User <username>
Host node2
   Hostname node2
   User <username>
Host node3
   Hostname node3
   User <username>
```

1.9. DISABLE RAID

If you have RAID (not recommended), configure your RAID controllers to RAID 0 (JBOD).

1.10. ADJUST PID COUNT

Hosts with high numbers of OSDs (e.g., > 20) may spawn a lot of threads, especially during recovery and re-balancing. Many Linux kernels default to a relatively small maximum number of threads (e.g., **32768**). Check your default settings to see if they are suitable.

```
cat /proc/sys/kernel/pid_max
```

Consider setting **kernel.pid_max** to a higher number of threads. The theoretical maximum is 4,194,303 threads. For example, you could add the following to the **/etc/sysctl.conf** file to set it to the maximum:

```
kernel.pid_max = 4194303
```

To see the changes you made without a reboot, execute:

```
sudo sysctl -p
```

To verify the changes, execute:

```
sudo sysctl -a | grep kernel.pid_max
```

CHAPTER 2. SETTING UP YOUR ADMINISTRATION SERVER

Red Hat Ceph Storage uses an administration server for a Red Hat Ceph Storage repository, the Calamari monitoring and administration server, and your cluster's Ceph configuration and authentication keys.

Execute the following steps:

1. Visit https://access.redhat.com/articles/1554343 to obtain the Red Hat Ceph Storage installation ISO image files.
2. As per your distro download **`rhceph-1.2.3-ubuntu-12.04-x86_64.iso`** or **`rhceph-1.2.3-ubuntu-14.04-x86_64.iso`** file.
3. Using **`sudo`**, mount the image:

   ```
   sudo mount <path_to_iso>/rhceph-1.2.3-ubuntu-12.04-x86_64.iso
   /mnt
   ```

   Or:

   ```
   sudo mount <path_to_iso>/rhceph-1.2.3-ubuntu-14.04-x86_64.iso
   /mnt
   ```

4. Using **`sudo`**, install the setup script.

   ```
   sudo dpkg -i /mnt/ice-setup_*.deb
   ```

   Note

   If you receive an error about missing **`python-pkg-resources`**, run **`sudo apt-get -f install`** to install the missing **`python-pkg-resources`** dependency.

5. Create a working directory for your Ceph cluster configuration files and keys. Then, navigate to that directory. For example:

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

6. Using **`sudo`**, run the setup script in the working directory. **NOTE:** You cannot run the setup script in **`/mnt`** or a read-only directory or the script will crash. The script will output a **`cephdeploy.conf`** file, which **`ceph-deploy`** will use to point to the local repository.

   ```
   sudo ice_setup -d /mnt
   ```

   Follow the instructions in the interactive shell.

   The setup script performs the following operations:

   - It creates a local repo for the **`ceph-deploy`** and **`calamari`** packages

- It installs the Calamari server packages on the admin node and
- It installs the **ceph-deploy** package on the admin node

  Once the script completes the setup, you should have two new subdirectories under the **/opt** directory viz **/opt/ICE** and **/opt/calamari**. You should also have a cephdeploy.conf file in your **~/ceph-config** directory.

7. Using **sudo**, initialize the Calamari monitoring and administration server.

```
sudo calamari-ctl initialize
```

   This will start services, initialize the databases and prompt you for an administrator username, password and email address.

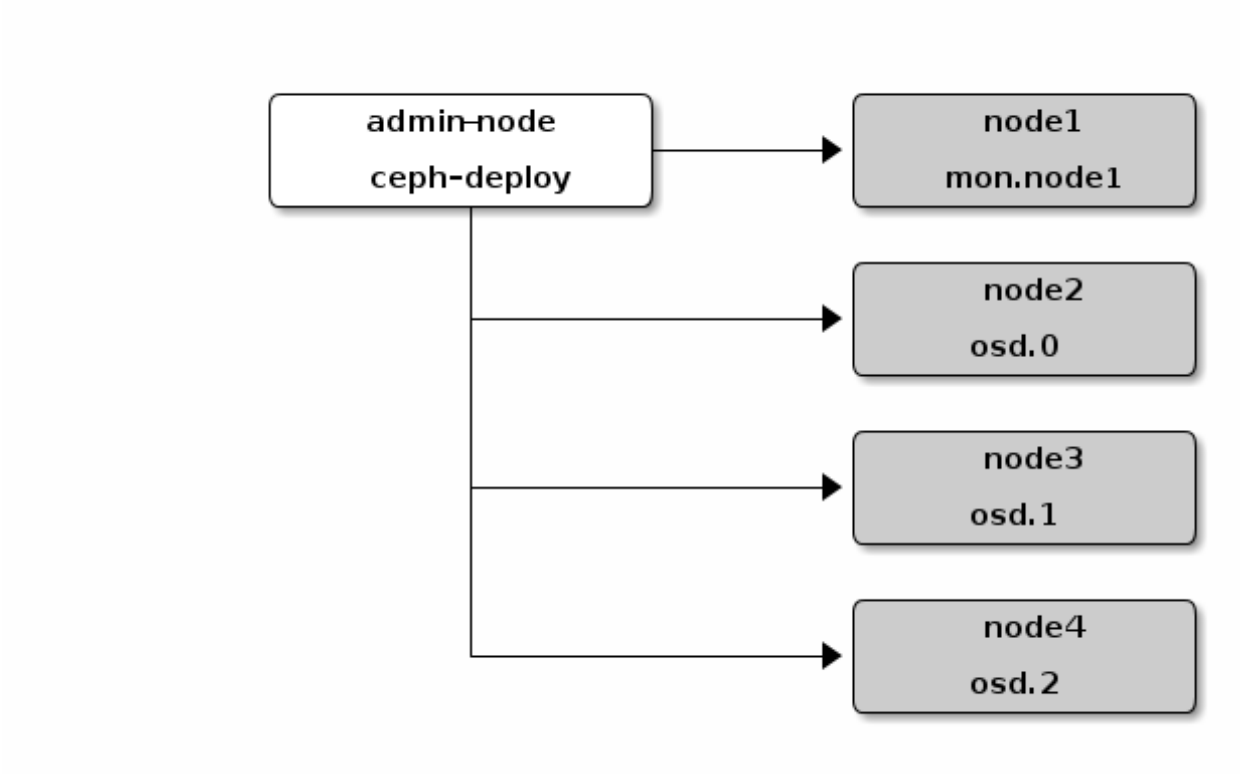
Note

The initialization script implies that you can only execute **ceph-deploy** when pointing to a remote site. You may also direct **ceph-deploy** to your Calamari admin node (e.g,. **ceph-deploy admin <admin-hostname>**). You can also use the Calamari admin node to run a Ceph daemon, although this is not recommended.

At this point, you should be able to access the Calamari web server via a web browser. Proceed to **Storage Cluster Quick Start**.

PART II. STORAGE CLUSTER QUICK START

This **Quick Start** sets up a Red Hat Ceph Storage cluster using **`ceph-deploy`** on your Calamari admin node. Create a small Ceph cluster so you can explore Ceph functionality. As a first exercise, create a Ceph Storage Cluster with one Ceph Monitor and some Ceph OSD Daemons, each on separate nodes. Once the cluster reaches an **`active + clean`** state, you can use the cluster.



CHAPTER 3. EXECUTING CEPH-DEPLOY

When executing **ceph-deploy** with Red Hat Ceph Storage, **ceph-deploy** will need to retrieve Ceph packages from the **/opt/ICE** directory on your Calamari admin host, so you need to ensure that **ceph-deploy** has access to the **cephdeploy.conf** file that was written to your local working directory when you executed **calamari-ctl initialize**.

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

Important

The **ceph-deploy** utility does not issue **sudo** commands needed on the remote host. Execute **ceph-deploy** commands as a regular user (not as **root** or using **sudo**). The Create a Ceph User and Enable Password-less SSH steps enable **ceph-deploy** to execute as **root** without **sudo** and without connecting to Ceph nodes as the **root** user.

The **ceph-deploy** utility will output files to the current directory. Ensure you are in this directory when executing **ceph-deploy**, and ensure that **ceph-deploy** points to the **cephdeploy.conf** file generated by **calamari-ctl initialize** when installing Red Had Ceph Storage packages.

CHAPTER 4. CREATE A CLUSTER

If at any point you run into trouble and you want to start over, execute the following to purge the configuration:

```
ceph-deploy purgedata <ceph-node> [<ceph-node>]
ceph-deploy forgetkeys
```

To purge the Ceph packages too, you may also execute:

```
ceph-deploy purge <ceph-node> [<ceph-node>]
```

If you execute **purge**, you must re-install Ceph.

On your Calamari admin node from the directory you created for holding your configuration details, perform the following steps using **ceph-deploy**.

1. Create the cluster. :

   ```
   ceph-deploy new <initial-monitor-node(s)>
   ```

   For example:

   ```
   ceph-deploy new node1
   ```

   Check the output of **ceph-deploy** with **ls** and **cat** in the current directory. You should see a Ceph configuration file, a monitor secret keyring, and a log file of the **ceph-deploy** procedures.

   At this stage, you may begin editing your Ceph configuration file.

   Note

   If you choose not to use **ceph-deploy** you will have to deploy Ceph manually or refer to Ceph manual deployment documentation and configure a deployment tool (e.g., Chef, Juju, Puppet, etc.) to perform each operation **ceph-deploy** performs for you.

2. Add the **public_network** and **cluster_network** settings under the **[global]** section of your Ceph configuration file.

   ```
   public_network = <ip-address>/<netmask>
   cluster_network = <ip-address>/<netmask>
   ```

   These settings distinguish which network is public (front-side) and which network is for the cluster (back-side). Ensure that your nodes have interfaces configured for these networks. We do not recommend using the same NIC for the public and cluster networks.

3. Turn on IPv6 if you intend to use it.

   ```
   ms_bind_ipv6 = true
   ```

4. Add or adjust the **osd journal size** setting under the **[global]** section of your Ceph configuration file.

```
osd_journal_size = 10000
```

We recommend a general setting of 10GB. Ceph's default **osd_journal_size** is **0**, so you will need to set this in your **ceph.conf** file. A journal size should find the product of the **filestore_max_sync_interval** and the expected throughput, and multiply the product by two (2). The expected throughput number should include the expected disk throughput (i.e., sustained data transfer rate), and network throughput. For example, a 7200 RPM disk will likely have approximately 100 MB/s. Taking the **min()** of the disk and network throughput should provide a reasonable expected throughput.

5. Set the number of copies to store (default is **3**) and the default minimum required write data when in a **degraded** state (default is **2**) under the **[global]** section of your Ceph configuration file. We recommend the default values for production clusters.

```
osd_pool_default_size = 3
osd_pool_default_min_size = 2
```

For a quick start, you may wish to set **osd_pool_default_size** to **2**, and the **osd_pool_default_min_size** to 1 so that you can achieve and **active+clean** state with only two OSDs.

These settings establish the networking bandwidth requirements for the cluster network, and the ability to write data with eventual consistency (i.e., you can write data to a cluster in a degraded state if it has **min_size** copies of the data already).

6. Set the default number of placement groups (**osd_pool_default_pg_num**) and placement groups for placement (**osd_pool_default_pgp_num**) for a pool under the **[global]** section of your Ceph configuration file. The number you specify depends upon the number of OSDs in your cluster. For small clusters (< 5 OSDs) we recommend 128 placement groups per pool. The **osd_pool_default_pg_num** and **osd_pool_default_pgp_num** value should be equal.

```
osd_pool_default_pg_num = <n>
osd_pool_default_pgp_num = <n>
```

- Less than 5 OSDs set **pg_num** and **pgp_num** to 128
- Between 5 and 10 OSDs set **pg_num** and **pgp_num** to 512
- Between 10 and 50 OSDs set **pg_num** and **pgp_num** to 4096
- If you have more than 50 OSDs, you need to understand the tradeoffs and how to calculate the **pg_num** and **pgp_num** values. Generally, you may use the formula:

```
              (OSDs * 100)
Total PGs =  ------------
              pool size
```

Where the **pool size** in the formula above is the **osd_pool_default_size** value you set in the preceding step. For best results, round the result of this formula up to the nearest power of two. It is an optional step, but it will help CRUSH balance objects evenly across placement groups.

7. Set the maximum number of placement groups per OSD. The Ceph Storage Cluster has a default maximum value of 300 placement groups per OSD. You can set a different maximum value in your Ceph configuration file.

   ```
   mon_pg_warn_max_per_osd
   ```

   Multiple pools can use the same CRUSH ruleset. When an OSD has too many placement groups associated to it, Ceph performance may degrade due to resource use and load. This setting warns you, but you may adjust it to your needs and the capabilities of your hardware.

8. Set a CRUSH leaf type to the largest serviceable failure domain for your replicas under the **`[global]`** section of your Ceph configuration file. The default value is **`1`**, or host, which means that CRUSH will map replicas to OSDs on separate separate hosts. For example, if you want to make three object replicas, and you have three racks of chassis/hosts, you can set **`osd_crush_chooseleaf_type`** to **`3`**, and CRUSH will place each copy of an object on OSDs in different racks. For example:

   ```
   osd_crush_chooseleaf_type = 3
   ```

   The default CRUSH hierarchy types are:

   - type 0 osd
   - type 1 host
   - type 2 chassis
   - type 3 rack
   - type 4 row
   - type 5 pdu
   - type 6 pod
   - type 7 room
   - type 8 datacenter
   - type 9 region
   - type 10 root

9. Set **`max_open_files`** so that Ceph will set the maximum open file descriptors at the OS level to help prevent Ceph OSD Daemons from running out of file descriptors.

   ```
   max_open_files = 131072
   ```

10. We recommend having settings for clock drift in your Ceph configuration in addition to setting up NTP on your monitor nodes, because clock drift is a common reason monitors fail to achieve a consensus on the state of the cluster. We recommend having the report time out and down out interval in the Ceph configuration file so you have a reference point for how long an OSD can be down before the cluster starts re-balancing.

    ```
    mon_clock_drift_allowed = .15
    mon_clock_drift_warn_backoff = 30
    mon_osd_down_out_interval = 300
    mon_osd_report_timeout = 300
    ```

11. Set the *full_ratio* and *near_full_ratio* to acceptable values. They default to full at 95% and near full at 85% by default. You may also set *backfill_full_ratio* so that OSDs don't accept backfill requests when they are already near capacity.

```
mon_osd_full_ratio = .75
mon_osd_nearfull_ratio = .65
osd_backfill_full_ratio = .65
```

Consider the amount of storage capacity that would be unavailable during the failure of a large-grained failure domain such as a rack (e.g., the failure of a power distribution unit or a rack switch). You should consider the cost/benefit tradeoff of having that amount of extra capacity available for the failure of a large-grained failure domain if you have stringent high availability requirements. As a best practice, as you get close to reaching the full ratio, you should start receiving "near full" warnings so that you have ample time to provision additional hardware for your cluster. "Near full" warnings may be annoying, but they are not as annoying as an interruption of service.

Important

When your cluster reaches its full ratio, Ceph prevents clients from accessing the cluster to ensure data durability. This results in a service interruption, so you should carefully consider the implications of capacity planning and the implications of reaching full capacity—especially in view of failure.

In summary, your initial Ceph configuration file should have at least the following settings with appropriate values assigned after the **=** sign:

```
[global]
fsid = <cluster-id>
mon_initial_members = <hostname>[, <hostname>]
mon_host = <ip-address>[, <ip-address>]
public_network = <network>[, <network>]
cluster_network = <network>[, <network>]
ms_bind_ipv6 = [true | false]
max_open_files = 131072
auth_cluster_required = cephx
auth_service_required = cephx
auth_client_required = cephx
osd_journal_size = <n>
filestore_xattr_use_omap = true
osd_pool_default_size = <n>  # Write an object n times.
osd_pool_default_min_size = <n> # Allow writing n copy in a degraded
state.
osd_pool_default_pg_num = <n>
osd_pool_default_pgp_num = <n>
osd_crush_chooseleaf_type = <n>
mon_osd_full_ratio = <n>
mon_osd_nearfull_ratio = <n>
osd_backfill_full_ratio = <n>
mon_clock_drift_allowed = .15
mon_clock_drift_warn_backoff = 30
mon_osd_down_out_interval = 300
mon_osd_report_timeout = 300
```

CHAPTER 5. INSTALL CEPH

Ensure that **ceph-deploy** is pointing to the **cephdeploy.conf** file generated by **calamari-ctl initialize** (e.g., in the exemplary **~/ceph-config** directory, the **/opt/ICE** directory, etc.). Otherwise, you may not receive packages from the local repository. Ideally, you should run **ceph-deploy** from the directory where you keep your configuration (e.g., the examplary **~/ceph-config**) so that you can maintain a **{cluster-name}.log** file with all the commands you have executed with **ceph-deploy**. To install Ceph on remote nodes, first use **--repo** option with **ceph-deploy** to install the repo files on remote nodes.

For **admin node**, execute:

```
ceph-deploy install <ceph-node>
```

For example:

```
ceph-deploy install admin-node
```

For other nodes, execute:

```
ceph-deploy install --repo <ceph-node> [<ceph-node> ...]
ceph-deploy install <ceph-node> [<ceph-node> ...]
```

For example:

```
ceph-deploy install --repo node1 node2 node3 node4
ceph-deploy install node1 node2 node3 node4
```

The **ceph-deploy** utility will install Ceph on each node. **NOTE**: If you use **ceph-deploy purge**, you must re-execute these steps to re-install Ceph.

CHAPTER 6. ADD INITIAL MONITORS

Add the initial monitor(s) and gather the keys.

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

Once you complete the process, your local directory should have the following keyrings:

- **<cluster-name>.client.admin.keyring**
- **<cluster-name>.bootstrap-osd.keyring**
- **<cluster-name>.bootstrap-mds.keyring**

CHAPTER 7. CONNECT MONITOR HOSTS TO CALAMARI

Once you have added the initial monitor(s), you need to connect the monitor hosts to Calamari.

```
ceph-deploy calamari connect <ceph-node>[<ceph-node> ...]
```

For example, using the exemplary **node1** from above, you would execute:

```
ceph-deploy calamari connect node1
```

If you expand your monitor cluster with additional monitors, you will have to connect the hosts that contain them to Calamari, too.

CHAPTER 8. MAKE YOUR CALAMARI ADMIN NODE A CEPH ADMIN NODE

After you create your initial monitors, you can use the Ceph CLI to check on your cluster. However, you have to specify the monitor and admin keyring each time with the path to the directory holding your configuration, but you can simplify your CLI usage by making the admin node a Ceph admin client.

Note

You will also need to install **ceph** or **ceph-common** on the Calamari node.

```
ceph-deploy admin <node-name>
```

For example:

```
ceph-deploy admin admin-node
```

The **ceph-deploy** utility will copy the **ceph.conf** and **ceph.client.admin.keyring** files to the **etc/ceph** directory. When **ceph-deploy** is talking to the local admin host (**admin-node**), it must be reachable by its hostname (e.g., **hostname -s**). If necessary, modify **/etc/hosts** to add the name of the admin host. If you do not have an **/etc/ceph** directory, you should install **ceph-common**.

You may then use the Ceph CLI.

Once you have added your new Ceph monitors, Ceph will begin synchronizing the monitors and form a quorum. You can check the quorum status by executing the following:

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

Ensure that you have acceptable permissions for the **/etc/ceph/ceph.client.admin.keyring**. You can use **sudo** when executing the **ceph** command, or your can change the keyring permissions to enable a specific user or group. Keyring permissions provide administrative capability to the Red Hat Ceph Storage cluster. So exercise caution if many users have access to the Ceph nodes and admin node.

```
sudo chmod +r /etc/ceph/ceph.client.admin.keyring
```

Note

Your cluster will not achieve an **active + clean** state until you add enough OSDs to facilitate object replicas. This is inclusive of CRUSH failure domains.

CHAPTER 9. ADD OSDS

Before creating OSDs, consider the following:

- We recommend using the XFS filesystem (default).
- We recommend using SSDs for journals. It is common to partition SSDs to serve multiple OSDs. Ensure that the number of SSD partitions does not exceed your SSD's sequential write limits. Also, ensure that SSD partitions are properly aligned, or their write performance will suffer.
- We recommend using **`ceph-deploy disk zap`** on a Ceph OSD drive before executing **`ceph-deploy osd create`**. For example:

```
ceph-deploy disk zap <ceph-node>:<data-drive>
```

From your admin node, use **`ceph-deploy`** to prepare the OSDs.

```
ceph-deploy osd prepare <ceph-node>:<data-drive>:<journal-partition>
[<ceph-node>:<data-drive>:<journal-partition>]
```

For example:

```
ceph-deploy osd prepare node2:sdb:ssdb node3:sdd:ssdb node4:sdd:ssdb
```

In the foregoing example, **`sdb`** is a spinning hard drive. Ceph will use the entire drive for OSD data. **`ssdb`** is a partition on an SSD drive, which Ceph will use to store the journal for the OSD.

Once you prepare OSDs, use **`ceph-deploy`** to activate the OSDs.

```
ceph-deploy osd activate <ceph-node>:<data-drive>:<journal-partition>
[<ceph-node>:<data-drive>:<journal-partition>]
```

For example:

```
ceph-deploy osd activate node2:sdb:ssdb node3:sdd:ssdb node4:sdd:ssdb
```

To achieve an **`active + clean`** state, you must add as many OSDs as the value of **`osd pool default size = <n>`** from your Ceph configuration file.

CHAPTER 10. CONNECT OSD HOSTS TO CALAMARI

Once you have added the initial OSDs, you need to connect the OSD hosts to Calamari.

```
ceph-deploy calamari connect <ceph-node>[<ceph-node> ...]
```

For example, using the exemplary **node2**, **node3** and **node4** from above, you would execute:

```
ceph-deploy calamari connect node2 node3 node4
```

As you expand your cluster with additional OSD hosts, you will have to connect the hosts that contain them to Calamari, too.

CHAPTER 11. CREATE A CRUSH HIERARCHY

You can run a Ceph cluster with a flat node-level hierarchy (default). This is NOT RECOMMENDED. We recommend adding named buckets of various types to your default CRUSH hierarchy. This will allow you to establish a larger-grained failure domain, usually consisting of racks, rows, rooms and data centers.

```
ceph osd crush add-bucket <bucket-name> <bucket-type>
```

For example:

```
ceph osd crush add-bucket dc1 datacenter
ceph osd crush add-bucket room1 room
ceph osd crush add-bucket row1 row
ceph osd crush add-bucket rack1 rack
ceph osd crush add-bucket rack2 rack
ceph osd crush add-bucket rack3 rack
```

Then, place the buckets into a hierarchy:

```
ceph osd crush move dc1 root=default
ceph osd crush move room1 datacenter=dc1
ceph osd crush move row1 room=room1
ceph osd crush move rack1 row=row1
ceph osd crush move node2 rack=rack1
```

CHAPTER 12. ADD OSD HOSTS/CHASSIS TO THE CRUSH HIERARCHY

Once you have added OSDs and created a CRUSH hierarchy, add the OSD hosts/chassis to the CRUSH hierarchy so that CRUSH can distribute objects across failure domains. For example:

```
ceph osd crush set osd.0 1.0 root=default datacenter=dc1 room=room1
row=row1 rack=rack1 host=node2
ceph osd crush set osd.1 1.0 root=default datacenter=dc1 room=room1
row=row1 rack=rack2 host=node3
ceph osd crush set osd.2 1.0 root=default datacenter=dc1 room=room1
row=row1 rack=rack3 host=node4
```

The foregoing example uses three different racks for the exemplary hosts (assuming that is how they are physically configured). Since the exemplary Ceph configuration file specified "rack" as the largest failure domain by setting **`osd_crush_chooseleaf_type = 3`**, CRUSH can write each object replica to an OSD residing in a different rack. Assuming **`osd_pool_default_min_size = 2`**, this means (assuming sufficient storage capacity) that the Ceph cluster can continue operating if an entire rack were to fail (e.g., failure of a power distribution unit or rack router).

CHAPTER 13. CHECK CRUSH HIERARCHY

Check your work to ensure that the CRUSH hierarchy is accurate.

```
ceph osd tree
```

If you are not satisfied with the results of your CRUSH hierarchy, you may move any component of your hierarchy with the **`move`** command.

```
ceph osd crush move <bucket-to-move> <bucket-type>=<parent-bucket>
```

If you want to remove a bucket (node) or OSD (leaf) from the CRUSH hierarchy, use the **`remove`** command:

```
ceph osd crush remove <bucket-name>
```

CHAPTER 14. CHECK CLUSTER HEALTH

To ensure that the OSDs in your cluster are peering properly, execute:

```
ceph health
```

You may also check on the health of your cluster using the Calamari dashboard.

CHAPTER 15. LIST/CREATE A POOL

You can manage pools using Calamari, or using the Ceph command line. Verify that you have pools for writing and reading data:

```
ceph osd lspools
```

You can bind to any of the pools listed using the **admin** user and **client.admin** key. To create a pool, use the following syntax:

```
ceph osd pool create <pool-name> <pg-num> [<pgp-num>] [replicated]
[crush-ruleset-name]
```

For example:

```
ceph osd pool create mypool 512 512 replicated replicated_ruleset
```

Note

To find the rule set names available, execute **ceph osd crush rule list**.

CHAPTER 16. STORING/RETRIEVING OBJECT DATA

To perform storage operations with Ceph Storage Cluster, all Ceph clients regardless of type must:

1. Connect to the cluster.
2. Create an I/O contest to a pool.
3. Set an object name.
4. Execute a read or write operation for the object.

The Ceph Client retrieves the latest cluster map and the CRUSH algorithm calculates how to map the object to a placement-group, and then calculates how to assign the placement group to a Ceph OSD Daemon dynamically. Client types such as Ceph Block Device and the Ceph Object Gateway perform the last two steps transparently.

To find the object location, all you need is the object name and the pool name. For example:

```
ceph osd map <poolname> <object-name>
```

Note

The **rados** CLI tool in the following example is for Ceph administrators only.

Exercise: Locate an Object

As an exercise, lets create an object. Specify an object name, a path to a test file containing some object data and a pool name using the **rados put** command on the command line. For example:

```
echo <Test-data> > testfile.txt
rados put <object-name> <file-path> --pool=<pool-name>
rados put test-object-1 testfile.txt --pool=data
```

To verify that the Ceph Storage Cluster stored the object, execute the following:

```
rados -p data ls
```

Now, identify the object location:

```
ceph osd map <pool-name> <object-name>
ceph osd map data test-object-1
```

Ceph should output the object's location. For example:

```
osdmap e537 pool 'data' (0) object 'test-object-1' -> pg 0.d1743484
(0.4) -> up [1,0] acting [1,0]
```

To remove the test object, simply delete it using the **rados rm** command. For example:

```
rados rm test-object-1 --pool=data
```

As the cluster size changes, the object location may change dynamically. One benefit of Ceph's dynamic rebalancing is that Ceph relieves you from having to perform the migration manually.

PART III. UPGRADING CEPH

You may upgrade your administration server and your Ceph Storage cluster when Red Hat provides fixes or delivers a major release.

CHAPTER 17. UPGRADING YOUR CLUSTER FROM V1.2.2 TO V1.2.3

Upgrade the administration server first to support upgrading other daemons in the cluster.

Remove **calamari-server**, **ceph**, and **ceph-deploy** repositories under **/etc/apt/sources.list.d/** directory from previous installation, download and mount the latest Ceph ISO, run **ice_setup**, re-initialize Calamari and upgrade Ceph.

Remove the ceph related repositories under **/etc/apt/sources.list.d/**.

For example:

```
cd /etc/apt/sources.list.d/
sudo rm calamari-server.list ceph.list ceph-deploy.list
```

To obtain the Red Hat Ceph Storage installation ISO image files for the newer version, visit https://access.redhat.com/articles/1554343.

Execute the following steps:

1. As per your distro download **rhceph-1.2.3-ubuntu-12.04-x86_64.iso** or **rhceph-1.2.3-ubuntu-14.04-x86_64.iso** file.
2. Using **sudo**, mount the image:

   ```
   sudo mount <path_to_iso>/rhceph-1.2.3-ubuntu-12.04-x86_64.iso
   /mnt
   ```

   Or:

   ```
   sudo mount <path_to_iso>/rhceph-1.2.3-ubuntu-14.04-x86_64.iso
   /mnt
   ```

3. Using **sudo**, install the setup script.

   ```
   sudo dpkg -i /mnt/ice-setup_*.deb
   ```

   Note

   if you receive an error about missing **python-pkg-resources**, run **sudo apt-get -f install** to install the missing **python-pkg-resources** dependency.

4. Back up your Ceph configuration, log and key files.
5. Change to your working directory. For example:

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

6. Run **ice_setup**:

```
sudo ice_setup -d /mnt
```

The **ice_setup** program will install upgraded version of **ceph-deploy**, **calamari** server, create new local repositories and a **cephdeploy.conf** file.

7. Restart Apache:

```
sudo service apache2 restart
```

8. Restart Calamari:

```
sudo calamari-ctl initialize
```

9. Upgrade Ceph:

```
ceph-deploy install <admin node>
```

10. To upgrade the Ceph daemons running on your cluster hosts, see **Storage Cluster Upgrade** for details.

CHAPTER 18. STORAGE CLUSTER UPGRADE

Upgrading Ceph daemons involves installing the upgraded packages, and restarting each Ceph daemon. We recommend upgrading in this order:

- Ceph Monitors
- Ceph OSD Daemons
- Ceph Object Gateways

- To upgrade monitors, first remove existing **Ceph** repos.

  From monitor node, execute:

  ```
  cd /etc/apt/sources.list.d/
  sudo rm -rf calamari-minion.list ceph.list
  ```

  Then, execute the following from your admin node:

  ```
  ceph-deploy install --repo <ceph-node>[<ceph-node> ...]
  ceph-deploy install <ceph-node>[<ceph-node> ...]
  ```

  ceph-deploy will install the latest version of Ceph.

  Although your monitor node is already connected with Calamari node from the previous installation, you need to reconnect your monitor node to the Calamari to get the latest **salt-minion** package.

  Execute from your admin node:

  ```
  ceph-deploy calamari connect <ceph-node>
  ```

  Restart your monitors one at a time. Give each daemon time to come **up** and **in**, rejoining the quorum before you restart the next instance. To restart a monitor, execute:

  ```
  sudo /etc/init.d/ceph [options] restart mon.[id]
  ```

  Or:

  ```
  sudo restart ceph-mon id={hostname}
  ```

- To upgrade OSDs, first remove existing **Ceph** repos.

  From OSD node, execute:

  ```
  cd /etc/apt/sources.list.d/
  sudo rm -rf calamari-minion.list ceph.list
  ```

  Then, execute the following from your admin node:

  ```
  ceph-deploy install --repo <ceph-node>[<ceph-node> ...]
  ceph-deploy install <ceph-node>[<ceph-node> ...]
  ```

ceph-deploy will install the latest version of Ceph.

Although your OSD node is already connected with Calamari node from the previous installation, you need to reconnect your OSD node to the Calamari node to get the latest **salt-minion** package.

Execute from your admin node:

```
ceph-deploy calamari connect <ceph-node>
```

We recommend upgrading OSDs by CRUSH hierarchy—i.e., by failure domain or performance domain. Give each daemon time to come **up** and **in** with the cluster reaching a **HEALTH_OK** state before proceeding to the next CRUSH hierarchy. To restart an OSD, execute:

```
sudo /etc/init.d/ceph [options] restart osd.[id]
```

Or:

```
sudo restart ceph-osd id={id}
```

- To upgrade a Ceph Object Gateway daemon, first remove existing **Ceph** repos (if any).

  From gateway node, execute:

  ```
  cd /etc/apt/sources.list.d/
  sudo rm -rf ceph.list
  ```

  Then, execute the following from your admin node:

  ```
  ceph-deploy install --repo <gateway-node>
  ```

  Then upgrade the **radosgw** package. Execute from your gateway node:

  ```
  sudo apt-get install radosgw
  ```

  To upgrade the Ceph Object Gateway synchronization agent, execute the following:

  ```
  sudo apt-get install radosgw-agent
  ```

  Restart each Ceph Object gateway daemon. To do so, execute the following on each host:

  ```
  sudo service radosgw restart
  ```

  If you are running a federated architecture, restart your sync agent(s). For data replication agents, go to the terminal and execute **ctrl + c**; then, execute:

  ```
  radosgw-agent -c [config-file]
  ```

  For metadata replication agents, go to the terminal and execute **ctrl + c**; then, execute:

  ```
  radosgw-agent -c [config-file] --metadata-only
  ```