Deploying Red Hat Ceph and OpenStack together with director

Configuring the director to deploy and use a Red Hat Ceph cluster
Configuring the director to deploy and use a Red Hat Ceph cluster

OpenStack Team
rhos-docs@redhat.com
Abstract

This guide provides information about using the Red Hat OpenStack Platform director to create an overcloud with a Red Hat Ceph Storage cluster. This includes instructions for customizing your Ceph cluster through the director.
Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE .................................................. 4

CHAPTER 1. DEPLOYING AN OVERCLOUD AND RED HAT CEPH STORAGE ......................... 5
1.1. RED HAT CEPH STORAGE DEPLOYMENT .................................................... 5
1.2. REQUIREMENTS TO DEPLOY RED HAT CEPH ............................................. 6
1.3. CEPH STORAGE CLUSTERS .................................................................. 6
1.4. CEPH STORAGE NODE REQUIREMENTS .................................................. 7
1.5. ANSIBLE PLAYBOOKS TO DEPLOY CEPH STORAGE .................................. 8

CHAPTER 2. PREPARING CEPH STORAGE NODES FOR DEPLOYMENT ......................... 10
2.1. CLEANING CEPH STORAGE NODE DISKS .................................................. 10
2.2. REGISTERING NODES ........................................................................... 10
2.3. VERIFYING AVAILABLE RED HAT CEPH STORAGE PACKAGES .................. 13
2.3.1. Verifying cephadm package installation ................................................. 13
2.4. DESIGNATING NODES FOR RED HAT CEPH STORAGE ............................... 13
2.5. DEFINING THE ROOT DISK FOR MULTI-DISK CLUSTERS ............................ 15
2.5.1. Properties that identify the root disk ...................................................... 16
2.6. USING THE OVERCLOUD-MINIMAL IMAGE TO AVOID USING A RED HAT SUBSCRIPTION ENTITLEMENT .................................................. 17

CHAPTER 3. DEPLOYING CEPH SERVICES ON DEDICATED NODES ........................... 19
3.1. CREATING A CUSTOM ROLES FILE ......................................................... 19

CHAPTER 4. DEPLOYING THE CEPH STORAGE CLUSTER ...................................... 20
4.1. DEPLOYMENT PREREQUISITES ............................................................... 20
4.2. CLUSTER NAME .................................................................................. 20
4.3. CONFIGURING NETWORK OPTIONS ....................................................... 21
4.4. CONFIGURING CRUSH HIERARCHIES ...................................................... 22
4.5. CONFIGURING SERVICE PLACEMENT OPTIONS ..................................... 23
4.6. CONFIGURING SSH USER OPTIONS ....................................................... 24
4.6.1. Creating the SSH user before cluster creation ........................................ 24
4.6.2. Disabling the SSH user ...................................................................... 25
4.7. CONFIGURING THE CONTAINER REGISTRY ............................................ 26

CHAPTER 5. CUSTOMIZING THE STORAGE SERVICE ........................................ 27
5.1. CONFIGURING A CUSTOM ENVIRONMENT FILE ........................................ 27
5.2. ENABLING CEPH METADATA SERVER ................................................... 28
5.3. CEPH OBJECT GATEWAY OBJECT STORAGE ........................................ 28
5.4. DEPLOYMENT OPTIONS FOR RED HAT OPENSTACK PLATFORM OBJECT STORAGE ............................................................ 29
5.4.1. RHOSP deployment using Ceph Object Gateway (RGW) ......................... 29
5.4.2. RHOSP deployment using Swift instead of RGW .................................... 29
5.4.3. RHOSP deployment without using object storage .................................. 30
5.5. CONFIGURING CEPH OBJECT STORE TO USE EXTERNAL CEPH OBJECT GATEWAY .......................................................... 30
5.6. CONFIGURING THE BLOCK STORAGE BACKUP SERVICE TO USE CEPH ........ 32
5.7. CONFIGURING MULTIPLE BONDED INTERFACES FOR CEPH NODES ............ 33
5.8. CONFIGURING BONDING MODULE DIRECTIVES .................................... 35

CHAPTER 6. CUSTOMIZING THE CEPH STORAGE CLUSTER .................................... 37
6.1. CONFIGURATION OPTIONS ................................................................. 37
6.2. GENERATING THE SERVICE SPECIFICATION (OPTIONAL) ......................... 38
6.3. CEPH CONTAINERS FOR RED HAT OPENSTACK PLATFORM WITH CEPH STORAGE .......................................................... 38
6.4. CONFIGURING ADVANCED OSD SPECIFICATIONS .................................. 39
6.5. MIGRATING FROM NODE-SPECIFIC OVERIDES ....................................... 39
6.6. ENABLING CEPH ON-WIRE ENCRYPTION

CHAPTER 7. USING DIRECTOR TO DEFINE PERFORMANCE TIERS FOR VARYING WORKLOADS
7.1. CONFIGURING PERFORMANCE TIERS
7.2. VERIFYING CRUSH RULES AND POOLS

CHAPTER 8. CREATING THE OVERCLOUD
8.1. ASSIGNING NODES AND FLAVORS TO ROLES
8.2. ASSIGNING NODES TO ROLES
8.3. INITIATING OVERCLOUD DEPLOYMENT
8.4. MONITORING THE STATUS OF THE OVERCLOUD CREATION PROCESS

CHAPTER 9. ADDING THE RED HAT CEPH STORAGE DASHBOARD TO AN OVERCLOUD DEPLOYMENT
9.1. TLS EVERYWHERE WITH CEPH DASHBOARD
9.2. INCLUDING THE NECESSARY CONTAINERS FOR THE CEPH DASHBOARD
9.3. DEPLOYING CEPH DASHBOARD
9.4. DEPLOYING CEPH DASHBOARD WITH A COMPOSABLE NETWORK
9.5. CHANGING THE DEFAULT PERMISSIONS
9.6. ACCESSING CEPH DASHBOARD

CHAPTER 10. POST-DEPLOYMENT OPERATIONS TO MANAGE THE CEPH STORAGE CLUSTER
10.1. ACCESSING THE OVERCLOUD
10.2. MONITORING CEPH STORAGE NODES
10.3. MAPPING A BLOCK STORAGE (CINDER) TYPE TO YOUR NEW CEPH POOL

CHAPTER 11. REBOOTING THE ENVIRONMENT
11.1. REBOOTING A CEPH STORAGE (OSD) CLUSTER
11.2. REBOOTING CEPH STORAGE OSDS TO ENABLE CONNECTIVITY TO THE CEPH MONITOR SERVICE

CHAPTER 12. SCALING THE CEPH STORAGE CLUSTER
12.1. SCALING UP THE CEPH STORAGE CLUSTER
12.2. SCALING DOWN AND REPLACING CEPH STORAGE NODES

CHAPTER 13. REPLACING A FAILED DISK
13.1. REPLACING A DISK

APPENDIX A. ADDITIONAL INFORMATION
A1. MULTIPLE BONDED INTERFACES
MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright’s message.
CHAPTER 1. DEPLOYING AN OVERCLOUD AND RED HAT CEPH STORAGE

WARNING

The content for this feature is available in this release as a Documentation Preview, and therefore is not fully verified by Red Hat. Use it only for testing, and do not use in a production environment.

You can use Red Hat OpenStack Platform (RHOSP) director to deploy your cloud environment, which director calls the overcloud, and Red Hat Ceph Storage. You manage and scale the cluster itself separate from the overcloud configuration.

For more information about Red Hat Ceph Storage, see the Red Hat Ceph Storage Architecture Guide.

This guide contains instructions for deploying a Red Hat Ceph Storage cluster with your overcloud. Director uses Ansible playbooks provided through the tripleo-ansible package to deploy the cluster. The director also manages the configuration and scaling operations of the cluster.

For more information about services in the Red Hat OpenStack Platform, see Configuring a basic overcloud with the CLI tools in Director Installation and Usage.

1.1. RED HAT CEPH STORAGE DEPLOYMENT

Red Hat Ceph Storage deployment is a two phase process:

- Ceph Cluster creation before the deployment of the overcloud.
- Ceph Cluster configuration during the deployment of the overcloud.

Cluster creation before the deployment of the overcloud requires the provisioning of networking resources and bare metal instances. For more information about the provisioning of networking resources, see Networking Guide. For more information about the provisioning of bare metal instances, see Bare Metal Provisioning.

During the cluster creation phase, a Red Hat Ceph cluster is deployed ready to serve the Ceph RADOS Block Device (RBD). The CephMon, CephMgr, and CephOSD services are running on all nodes that should have those services. Additionally, no pools or cephx keys have been created.

IMPORTANT

The CephDashboard, CephRGW, and CephMds are not available until after the cluster configuration phase during the deployment of the overcloud.

Cluster creation is performed with the openstack overcloud ceph deploy command.

Cluster configuration during the overcloud deployment finalizes the overall configuration of the Red Hat Ceph cluster. Additional daemons and services such as CephRGW and CephDashboard are deployed according to the overcloud definition. Additionally, the applicable Red Hat OpenStack services are
configured as clients for the cluster.

The Cluster bootstrap, with the initial configuration of the RGW, Dashboard and MDS services, is created with the `openstack overcloud deploy` command.

### 1.2. REQUIREMENTS TO DEPLOY RED HAT CEPH

Before you create a Red Hat Ceph cluster you must configure the following:

- Use the command `openstack overcloud node provision` to create networks using the `cli-overcloud-network-provision.yaml` ansible playbook with the network definitions provided in the `--network-file` argument. This playbook creates or updates the neutron networks on the undercloud and generates the `networks-deployed.yaml` environment file. This is included as a user environment when creating the overcloud heat stack.

- Use the command `openstack overcloud node provision` to provision bare metal instances using the `cli-overcloud-node-provision.yaml` ansible playbook with the bare metal instance definitions provided in the `--baremetal-deployment` argument in combination with the `--network-config` argument so that bare metal nodes are provisioned and network port resources are created. Also, run any arbitrary Complete configuration of overcloud node networking using the `tripleo_network_config` ansible role.

**IMPORTANT**

By default, the Ceph monitor service installs on the overcloud Controller nodes. You must provide adequate resources to avoid performance issues. See the Red Hat Ceph Storage Hardware Guide for Red Hat Ceph Storage resource allocation recommendations. See Recommendations for Large Deployments for Red Hat OpenStack Platform resource allocation recommendations. Controller nodes should have enough resources to support Ceph monitor service operation in addition to Red Hat OpenStack Platform operation.

When these steps are complete, you can proceed with creating and deploying the cluster.

Once the cluster is created and deployed, the configuration is finalized during the overcloud deployment. The overcloud environment must contain the following to proceed with finalizing the cluster configuration:

- An undercloud host with the Red Hat OpenStack Platform (RHOSP) director installed. See Installing director in Director Installation and Usage.

- Any additional hardware recommended for Red Hat Ceph Storage. For more information about recommended hardware, see the Red Hat Ceph Storage Hardware Guide.

### 1.3. CEPH STORAGE CLUSTERS

Red Hat Ceph Storage is a distributed data object store designed to provide excellent performance, reliability, and scalability. Distributed object stores accommodate unstructured data so clients can use modern object interfaces and legacy interfaces simultaneously. At the core of every Ceph deployment is the Ceph Storage cluster, which consists of several types of daemons, but primarily, these two:

**Ceph OSD (Object Storage Daemon)**

Ceph OSDs store data on behalf of Ceph clients. Additionally, Ceph OSDs use the CPU and memory of Ceph nodes to perform data replication, rebalancing, recovery, monitoring, and reporting functions.
Ceph monitor

A Ceph monitor maintains a master copy of the Ceph Storage cluster map with the current state of the storage cluster.

For more information about Red Hat Ceph, see the Red Hat Ceph Storage Architecture Guide.

1.4. CEPH STORAGE NODE REQUIREMENTS

If you use Red Hat OpenStack Platform (RHOSP) director to create Red Hat Ceph Storage nodes, there are additional requirements.

For information about how to select a processor, memory, network interface cards (NICs), and disk layout for Ceph Storage nodes, see Hardware selection recommendations for Red Hat Ceph Storage in the Red Hat Ceph Storage Hardware Guide.

Each Ceph Storage node also requires a supported power management interface, such as Intelligent Platform Management Interface (IPMI) functionality, on the motherboard of the server.

NOTE

RHOSP director uses cephadm, which does not support installing the OSD on the root disk of Ceph Storage nodes. This means that you need at least two disks for a supported Ceph Storage node.

Ceph Storage nodes and RHEL compatibility

- RHOSP 17.0-beta is supported on RHEL 9.0. However, hosts that are mapped to the Ceph Storage role update to the latest major RHEL release. Before upgrading to RHOSP 16.1 and later, review the Red Hat Knowledgebase article Red Hat Ceph Storage: Supported configurations.

Placement Groups (PGs)

Ceph Storage uses placement groups (PGs) to facilitate dynamic and efficient object tracking at scale. In the case of OSD failure or cluster rebalancing, Ceph can move or replicate a placement group and its contents, which means a Ceph Storage cluster can rebalance and recover efficiently.

When Ceph is initially deployed with the openstack overcloud ceph deploy command, the PG and replica count settings are not changed from the defaults unless the parameters osd_pool_default_size, osd_pool_default_pg_num, osd_pool_default_pgp_num are included in an initial Ceph configuration file. This configuration file is passed using the --config option. They can also be modified after openstack overcloud ceph deploy is run.

When the openstack overcloud deploy command is used, a pool is created for each OpenStack Platform service that is enabled by including the associated Heat environment. For example, a command such as the following will create pools for Nova (vms), Cinder (volumes) and Glance (images):

```
openstack overcloud deploy --templates -e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm-rbd-only.yaml
```

If the -e environments/cinder-backup.yaml option was included in the above command, then a pool called backups would also be created.

By default, each pool has pg_autoscale_mode enabled. It is not necessary to directly set a PG number per pool. However, even with this mode enabled, it is recommended to set the target_size_ratio (or...
pg_num) per pool to minimize data rebalancing. To control the target_size_ratio per pool, create a Heat environment file that defines the configuration.

The following provides the example contents of such a file.

```
CephPools:
- name: volumes
  target_size_ratio: 0.4
  application: rbd
- name: images
  target_size_ratio: 0.1
  application: rbd
- name: vms
  target_size_ratio: 0.3
  application: rbd
```

If this file was saved as pools.yaml, it would be deployed with the openstack overcloud deploy command with the -e pools.yaml option.

In the above example, it is assumed that the percentage of data used per service will be:

- Cinder volumes - 40%
- Glance images - 10%
- Nova vms - 30%
- Free space for other pools - 20%

Set these values based on your expected usage. If you do not override the CephPools parameter, each pool has Ceph’s default PG number. Though the autoscaler will adjust this number automatically over time based on usage, the data will be moved within the cluster. This uses computational resources.

If you prefer to set a PG number instead of a target size ratio, replace target_size_ratio in the example above with pg_num. Use a different integer per pool based on your expected usage.

See the Red Hat Ceph Storage Hardware Guide for Red Hat Ceph Storage processor, network interface card, and power management interface recommendations.

### 1.5. ANSIBLE PLAYBOOKS TO DEPLOY CEPH STORAGE

Director deploys a Ceph cluster ready to serve RBD and CephFS using tripleo-ansible roles executed by the cephadm command. After director completes the deployment, the following should be in place:

**WARNING**

Although tripleo-ansible uses playbooks to deploy Ceph, do not edit these files to customize your deployment. Instead, use heat environment files to override the defaults set by these playbooks. If you edit the tripleo-ansible playbooks directly, your deployment fails.
SSH access to a node with the CephMon service to use the `sudo cephadm shell` command.

All OSDs should be running unless environmental issues, such as disks not being cleaned, prevent it.

A Ceph configuration file and client administration keyring file in the `/etc/ceph` directory of nodes with the CephMon service.

The Ceph cluster is ready to serve RBD.

After director completes the deployment, the should not be in place:

- No pools or cephx keys for OpenStack.
- No CephDashboard or CephRGW services will be running.

Pools, cephx keys, CephDashboard, and CephRGW are configured during overcloud deployment by the `openstack overcloud deploy` command. This is for two reasons:

- The Dashboard and RGW services must integrate with haproxy. This is deployed with the overcloud.
- The list of pools to create and their respective cephx keys are dependent on which OpenStack clients, such as Nova and Cinder, are deployed.

During the overcloud deployment, these resources will be created in Ceph using the client administration keyring file and the `~/deployed_ceph.yaml` file output by the `openstack overcloud ceph deploy` command.

See Red Hat Ceph Storage Installation Guide for more information about Cephadm.
CHAPTER 2. PREPARING CEPH STORAGE NODES FOR DEPLOYMENT

All nodes are bare metal systems that use IPMI for power management. These nodes do not require an operating system because director copies a Red Hat Enterprise Linux image to each node. Additionally, the Ceph Storage services on these nodes are containerized. Director communicates to each node through the Provisioning network during the introspection and provisioning processes. All nodes connect to this network through the native VLAN.

See Provisioning bare metal nodes before deploying the overcloud in Director Installation and Usage for additional information on bare metal provisioning before overcloud deployment. See Bare Metal Provisioning for a complete guide to bare metal provisioning.

2.1. CLEANING CEPH STORAGE NODE DISKS

Ceph Storage OSDs and journal partitions require factory clean disks. This means the additional disks on Ceph Storage should be wiped by the Bare Metal Provisioning service (ironic) before installing the Ceph OSD services. You must delete all metadata from the disks or OSDs will not be created on them.

You can configure director to delete all disk metadata by default. With this option, the Bare Metal Provisioning service runs an additional step to boot the nodes and clean the disks each time the node is set to available. This process adds an additional power cycle after the first introspection and before each deployment. The Bare Metal Provisioning service uses the `wipefs --force --all` command to perform the clean.

Procedure

1. Add the following setting to your `/home/stack/undercloud.conf` file:

   ```
   clean_nodes=true
   ```

2. After you set this option, run the `openstack undercloud install` command to execute this configuration change.

   **WARNING**

   The `wipefs --force --all` command deletes all data and metadata on the disk, but it does not perform a secure erase. A secure erase takes much longer.

2.2. REGISTERING NODES

You must register the nodes to enable director to communicate with them.

Procedure

1. Create a node inventory file in JSON format in the `home` directory of the `stack` user:
2. Configure the inventory file with the hardware and power management details that director can use to register nodes:

```json
{
  "nodes": [
    {
      "mac": [
        "b1:b1:b1:b1:b1:b1"
      ],
      "cpu": "4",
      "memory": "6144",
      "disk": "40",
      "arch": "x86_64",
      "pm_type": "ipmi",
      "pm_user": "admin",
      "pm_password": "p@55w0rd!",
      "pm_addr": "192.0.2.205"
    },
    {
      "mac": [
        "b2:b2:b2:b2:b2:b2"
      ],
      "cpu": "4",
      "memory": "6144",
      "disk": "40",
      "arch": "x86_64",
      "pm_type": "ipmi",
      "pm_user": "admin",
      "pm_password": "p@55w0rd!",
      "pm_addr": "192.0.2.206"
    },
    {
      "mac": [
        "b3:b3:b3:b3:b3:b3"
      ],
      "cpu": "4",
      "memory": "6144",
      "disk": "40",
      "arch": "x86_64",
      "pm_type": "ipmi",
      "pm_user": "admin",
      "pm_password": "p@55w0rd!",
      "pm_addr": "192.0.2.207"
    },
    {
      "mac": [
        "c1:c1:c1:c1:c1:c1"
      ],
      "cpu": "4",
      "memory": "6144",
      "disk": "40",
      "arch": "x86_64",
      "pm_type": "ipmi",
      "pm_user": "admin",
      "pm_password": "p@55w0rd!",
      "pm_addr": "192.0.2.208"
    }
  ]
}
```
"pm_user":"admin",
"pm_password":"p@55w0rd!",
"pm_addr":"192.0.2.208"
},
{
  "mac": [
    "c2:c2:c2:c2:c2:c2"
  ],
  "cpu":"4",
  "memory":"6144",
  "disk":"40",
  "arch":"x86_64",
  "pm_type":"ipmi",
  "pm_user":"admin",
  "pm_password":"p@55w0rd!",
  "pm_addr":"192.0.2.209"
},
{
  "mac": [
    "c3:c3:c3:c3:c3:c3"
  ],
  "cpu":"4",
  "memory":"6144",
  "disk":"40",
  "arch":"x86_64",
  "pm_type":"ipmi",
  "pm_user":"admin",
  "pm_password":"p@55w0rd!",
  "pm_addr":"192.0.2.210"
},
{
  "mac": [
    "d1:d1:d1:d1:d1:d1"
  ],
  "cpu":"4",
  "memory":"6144",
  "disk":"40",
  "arch":"x86_64",
  "pm_type":"ipmi",
  "pm_user":"admin",
  "pm_password":"p@55w0rd!",
  "pm_addr":"192.0.2.211"
},
{
  "mac": [
    "d2:d2:d2:d2:d2:d2"
  ],
  "cpu":"4",
  "memory":"6144",
  "disk":"40",
  "arch":"x86_64",
  "pm_type":"ipmi",
  "pm_user":"admin",
  "pm_password":"p@55w0rd!",
  "pm_addr":"192.0.2.212"
},
3. Initialize the stack user, then import the `instackenv.json` inventory file into director:

```bash
$ source ~/stackrc
$ openstack overcloud node import ~/instackenv.json
```

The `openstack overcloud node import` command imports the inventory file and registers each node with director.

4. Assign the kernel and ramdisk images to each node:

```bash
$ openstack overcloud node configure <node>
```

The nodes are registered and configured in director.

### 2.3. VERIFYING AVAILABLE RED HAT CEPH STORAGE PACKAGES

To help avoid overcloud deployment failures, verify that the required packages exist on your servers.

#### 2.3.1. Verifying cephadm package installation

The `cephadm` package must be installed on at least one overcloud node to bootstrap the first node of the Ceph Storage cluster. The `cephadm` package is included in the `overcloud-full` image. The `tripleo_cephadm` role uses the Ansible package module to ensure it is present in the image.

### 2.4. DESIGNATING NODES FOR RED HAT CEPH STORAGE

To designate nodes for Red Hat Ceph Storage, you must create a new role file to configure the `CephStorage` role, and configure the bare metal nodes with a resource class for `CephStorage`.

**Procedure**

1. Log in to the undercloud as the `stack` user.

2. Source the `stackrc` file:

   ```bash
   [stack@director ~]$ source ~/stackrc
   ```
3. Generate a new roles data file named `roles_data.yaml` that includes the **Controller**, **Compute**, and **CephStorage** roles:

   ```bash
   (undercloud)$ openstack overcloud roles \
       generate -o /home/stack/templates/roles_data.yaml \
   ```

4. Open `roles_data.yaml` and edit or add the following parameters and sections:

<table>
<thead>
<tr>
<th>Section/Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role comment</td>
<td>Role: CephStorage</td>
</tr>
<tr>
<td>Role name</td>
<td>name: CephStorage</td>
</tr>
<tr>
<td>description</td>
<td>Ceph node role</td>
</tr>
<tr>
<td>HostnameFormatDefault</td>
<td>%stackname%-novaceph-%index%</td>
</tr>
<tr>
<td>deprecated_nic_config_name</td>
<td>ceph.yaml</td>
</tr>
</tbody>
</table>

5. Register the Ceph nodes for the overcloud by adding them to your node definition template, `node.json` or `node.yaml`. For more information, see [Registering nodes for the overcloud](../DirectorInstallationAndUsageGuide) in the Director Installation and Usage guide.

6. Inspect the node hardware:

   ```bash
   (undercloud)$ openstack overcloud node introspect --all-manageable --provide
   ```

   For more information, see [Creating an inventory of the bare-metal node hardware](../DirectorInstallationAndUsageGuide) in the Director Installation and Usage guide.

7. Tag each bare metal node that you want to designate for Ceph with a custom Ceph resource class:

   ```bash
   (undercloud)$ openstack baremetal node set \
       --resource-class baremetal.CEPH <node>
   ```

   Replace `<node>` with the ID of the bare metal node.

8. Add the **CephStorage** role to your `overcloud-baremetal-deploy.yaml` file, and define any predictive node placements, resource classes, or other attributes that you want to assign to your nodes:

   ```yaml
   - name: Controller
     count: 3
   - name: Compute
     count: 3
   - name: CephStorage
     count: 1
     defaults:
       resource_class: baremetal.CEPH
   ```

9. Run the provisioning command:
(undercloud)$ openstack overcloud node provision \ 
--stack stack \ 
--output /home/stack/templates/overcloud-baremetal-deployed.yaml \ 
/home/stack/templates/overcloud-baremetal-deploy.yaml

10. Monitor the provisioning progress in a separate terminal. When provisioning is successful, the node state changes from available to active:

(undercloud)$ watch openstack baremetal node list

2.5. DEFINING THE ROOT DISK FOR MULTI-DISK CLUSTERS

Most Ceph Storage nodes use multiple disks. When nodes use multiple disks, director must identify the root disk. By default, director writes the overcloud image to the root disk during the provisioning process.

Use this procedure to identify the root device by serial number. For more information about other properties you can use to identify the root disk, see Section 2.5.1, "Properties that identify the root disk".

Procedure

1. Verify the disk information from the hardware introspection of each node. The following command to displays the disk information of a node:

(undercloud)$ openstack baremetal introspection data save 1a4e30da-b6dc-499d-ba87-0bd8a3819bc0 | jq ".inventory.disks"

For example, the data for one node might show three disks:

```json
[
  {
    "size": 299439751168,
    "rotational": true,
    "vendor": "DELL",
    "name": "/dev/sda",
    "wwn_vendor_extension": "0x1ea4dcc412a9632b",
    "wwn_with_extension": "0x61866da04f3807001ea4dcc412a9632b",
    "model": "PERC H330 Mini",
    "wwn": "0x61866da04f380700",
    "serial": "61866da04f3807001ea4dcc412a9632b"
  },
  {
    "size": 299439751168,
    "rotational": true,
    "vendor": "DELL",
    "name": "/dev/sdb",
    "wwn_vendor_extension": "0x1ea4e13c12e36ad6",
    "wwn_with_extension": "0x61866da04f3807001ea4e13c12e36ad6",
    "model": "PERC H330 Mini",
    "wwn": "0x61866da04f380700",
    "serial": "61866da04f3807001ea4e13c12e36ad6"
  },
  {
    "size": 299439751168,
    "rotational": true,
    "vendor": "DELL",
    "name": "/dev/sdc",
    "wwn_vendor_extension": "0x1ea4e13c12e36ad6",
    "wwn_with_extension": "0x61866da04f3807001ea4e13c12e36ad6",
    "model": "PERC H330 Mini",
    "wwn": "0x61866da04f380700",
    "serial": "61866da04f3807001ea4e13c12e36ad6"
  }
]```
2. On the undercloud, set the root disk for a node. Include the most appropriate hardware attribute value to define the root disk.

```bash
(undercloud)$ openstack baremetal node set --property root_device='{"serial": "<serial_number>"}'} <node-uuid>
```

For example, to set the root device to disk 2, which has the serial number `61866da04f380d001ea4e13c12e36ad6`, enter the following command:

```bash
(undercloud)$ openstack baremetal node set --property root_device='{"serial": "61866da04f380d001ea4e13c12e36ad6"}'} 1a4e30da-b6dc-499d-ba87-0bd8a3819bc0
```

**NOTE**

Configure the BIOS of each node to boot from the root disk that you choose. Configure the boot order to boot from the network first, then from the root disk.

Director identifies the specific disk to use as the root disk. When you run the `openstack overcloud node provision` command, director provisions and writes the overcloud image to the root disk.

### 2.5.1. Properties that identify the root disk

There are several properties that you can define to help director identify the root disk:

- **model** (String): Device identifier.
- **vendor** (String): Device vendor.
- **serial** (String): Disk serial number.
- **hctl** (String): Host:Channel:Target:Lun for SCSI.
- **size** (Integer): Size of the device in GB.
- **wwn** (String): Unique storage identifier.
- **wwn_with_extension** (String): Unique storage identifier with the vendor extension appended.
- **wwn_vendor_extension** (String): Unique vendor storage identifier.
- **rotational** (Boolean): True for a rotational device (HDD), otherwise false (SSD).
- **name** (String): The name of the device, for example: `/dev/sdb1`.

16
Important

Use the `name` property only for devices with persistent names. Do not use `name` to set the root disk for any other devices because this value can change when the node boots.

2.6. Using the Overcloud-Minimal Image to Avoid Using a Red Hat Subscription Entitlement

The default image for a Red Hat OpenStack Platform (RHOSP) deployment is `overcloud-hardened-uefi-full`. The `overcloud-hardened-uefi-full` image uses a valid Red Hat OpenStack Platform (RHOSP) subscription. You can use the `overcloud-minimal` image when you do not want to consume your subscription entitlements, to avoid reaching the limit of your paid Red Hat subscriptions. This is useful, for example, when you want to provision nodes with only Ceph daemons, or when you want to provision a bare operating system (OS) where you do not want to run any other OpenStack services. For information about how to obtain the `overcloud-minimal` image, see Obtaining images for overcloud nodes.

Note

The `overcloud-minimal` image supports only standard Linux bridges. The `overcloud-minimal` image does not support Open vSwitch (OVS) because OVS is an OpenStack service that requires a Red Hat OpenStack Platform subscription entitlement. OVS is not required to deploy Ceph Storage nodes. Use `linux_bond` instead of `ovs_bond` to define bonds. For more information about `linux_bond`, see Linux bonding options.

Procedure

1. Open your `overcloud-baremetal-deploy.yaml` file.

2. Add or update the `image` property for the nodes that you want to use the `overcloud-minimal` image. You can set the image to `overcloud-minimal` on specific nodes, or for all nodes for a role:

Specific nodes

- name: Ceph
count: 3
instances:
- hostname: overcloud-ceph-0
  name: node00
  image:
    href: overcloud-minimal
- hostname: overcloud-ceph-1
  name: node01
  image:
    href: overcloud-full-custom
- hostname: overcloud-ceph-2
  name: node02
  image:
    href: overcloud-full-custom

All nodes for a role

- name: Ceph
3. In the `roles_data.yaml` role definition file, set the `rhsm_enforce` parameter to **False**.

   ```yaml
   rhsm_enforce: False
   ```

4. Run the provisioning command:

   ```bash
   (undercloud)$ openstack overcloud node provision \
   --stack stack \
   --output /home/stack/templates/overcloud-baremetal-deployed.yaml \
   /home/stack/templates/overcloud-baremetal-deploy.yaml
   ```

5. Pass the `overcloud-baremetal-deployed.yaml` environment file to the `openstack overcloud deploy` command.
CHAPTER 3. DEPLOYING CEPH SERVICES ON DEDICATED NODES

By default, director deploys the Ceph MON and Ceph MDS services on the Controller nodes. This is suitable for small deployments. However, with larger deployments, Red Hat recommends that you deploy the Ceph MON and Ceph MDS services on dedicated nodes to improve the performance of your Ceph cluster.

Director uses the following file as a default reference for all overcloud roles:

- `/usr/share/openstack-tripleo-heat-templates/roles_data.yaml`

3.1. CREATING A CUSTOM ROLES FILE

Create a custom role for services that you want to isolate on dedicated nodes.

Procedure

1. Make a copy of the `roles_data.yaml` file in `/home/stack/templates/` so that you can add custom roles:

   ```
   $ cp /usr/share/openstack-tripleo-heat-templates/roles_data.yaml /home/stack/templates/roles_data_custom.yaml
   ```

2. Include the new custom role file in the deployment command with the `-r` option when you deploy the overcloud:

   ```
   $ openstack overcloud deploy -r roles_data_custom.yaml
   ```
CHAPTER 4. DEPLOYING THE CEPH STORAGE CLUSTER

This chapter describes how to configure and deploy the Ceph Storage cluster.

4.1. DEPLOYMENT PREREQUISITES

Confirm the following has been performed before attempting to configure and deploy the cluster:

- Provision of bare metal instances and their networks using the Bare Metal Provisioning service (ironic). For more information about the provisioning of bare metal instances, see Bare Metal Provisioning.

4.2. CLUSTER NAME

The Ceph cluster can be deployed with a name other than the default of ceph.

Procedure

1. Log in to the undercloud node as the stack user.

2. Use the command `openstack overcloud ceph deploy --cluster <cluster_name>` to configure the name.

   $ openstack overcloud ceph deploy --cluster central

Result

Ceph configuration is created for the new cluster name.

**NOTE**

Keyring files are not created at this time. When they are created during the overcloud deployment, they will have the cluster name configured here.

In the example above, the cluster was named central. The configuration and keyring files for the central cluster would be created in /etc/ceph.

```
[root@oc0-controller-0 ~]# ls -l /etc/ceph/
total 16
-rw-------. 1 root root  63 Mar 26 21:49 central.client.admin.keyring
-rw-------. 1  167  167 201 Mar 26 22:17 central.client.openstack.keyring
-rw-------. 1  167  167 134 Mar 26 22:17 central.client.radosgw.keyring
-rw-r--r--. 1 root root 177 Mar 26 21:49 central.conf
```
The error `monclient: get_monmap_and_config cannot identify monitors to contact because` may be displayed if the default `ceph` cluster name is not used. If this error is displayed, execute the `cephadm shell --config <configuration_file> --keyring <keyring_file>` after Ceph deployment.

For the example above, the command will look like the following:

```
cephadm shell --config /etc/ceph/central.conf \
    --keyring /etc/ceph/central.client.admin.keyring
```

Alternatively, the following command could be also used for the above example:

```
cephadm shell --mount /etc/ceph:/etc/ceph
export CEPH_ARGS='--cluster central'
```

## 4.3. CONFIGURING NETWORK OPTIONS

The `network_data.yaml` file describes the networks used by the Red Hat Ceph Storage cluster. Using network isolation, the standard network deployments consists of two storage networks which map to the two Ceph networks in the following way:

- The storage network `storage` maps to the Ceph network `public_network`. This network handles storage traffic such as the RBD traffic from the Nova compute nodes to the Ceph cluster.
- The storage network `storage_mgmt` maps to the Ceph network `cluster_network`. This network handles storage management traffic such as data replication between Ceph OSDs.

The `openstack overcloud ceph deploy` command uses the `network_data.yaml` file specified by the `--network-data` option to determine the networks to be used as the `public_network` and `cluster_network`. The command assumes these networks are named `storage` and `storage_mgmt` in `network_data.yaml` unless a different name is specified by the `--public-network-name` and `--cluster-network-name` options.

### IMPORTANT

You must use the `--network-data` option when deploying with network isolation. The default undercloud (192.168.24.0/24) will be used for both the `public_network` and `cluster_network` if you do not use this option.

### Procedure

1. Log in to the undercloud node as the `stack` user.
2. Create a standard format initialization (ini) file to configure the cluster. If you have already created a file to include other configuration options, the network options can be added to it.
3. Add the following parameters to the `[global]` section of the file:
   - `public_network`
   - `cluster_network`
   - `ms_bind_ipv4`
• **ms_bind_ipv6**
  The following is an example of a configuration file entry for a network configuration with multiple subnets and custom networking names:

  ```
  [global]
  public_network = '172.16.14.0/24,172.16.15.0/24'
  cluster_network = '172.16.12.0/24,172.16.13.0/24'
  ms_bind_ipv4 = True
  ms_bind_ipv6 = False
  ```

  The following is an example of a configuration file entry for an IPv6 network:

  ```
  [global]
  public_network = fd00:fd00:fd00:3000::/64
  cluster_network = fd00:fd00:fd00:4000::/64
  ms_bind_ipv4 = False
  ms_bind_ipv6 = True
  ```

4. Use the command `openstack overcloud ceph deploy` with the `--config` option to deploy the configuration file.

   ```bash
   $ openstack overcloud ceph deploy --config initial-ceph.conf --network-data network_data.yaml
   ```

**IMPORTANT**

Ensure the `public_network` and `cluster_network` map to the same networks as `storage` and `storage_mgmt` as outlined above.

**IMPORTANT**

Using this method to configure network options overwrites automatically generated values in `network_data.yaml`. Ensure you set all four values when using this network configuration method.

### 4.4. CONFIGURING CRUSH HIERARCHIES

A custom CRUSH hierarchy can be defined during OSD deployment to add the `location` attribute to the Hosts specification. The `location` attribute defines where the OSD is placed within the CRUSH hierarchy. See Red Hat Ceph Storage workload considerations in the Red Hat Ceph Storage Installation Guide for additional information.

**NOTE**

The `location` attribute only affects the initial CRUSH location. Subsequent changes of the attribute are ignored.

Removing a host does not remove the CRUSH-generated bucket.

**Procedure**

1. Log in to the undercloud node as the `stack` user.
Create a YAML format file that defines the custom hierarchy. The following is an example of a custom hierarchy file named `crush_hierarchy.yaml`.

```yaml
ceph_crush_hierarchy:
  ceph-0:
    root: default
    rack: r0
  ceph-1:
    root: default
    rack: r1
  ceph-2:
    root: default
    rack: r2
```

1. Use the `openstack overcloud ceph deploy` command with the `--crush-hierarchy` option to deploy the file.

```bash
openstack overcloud ceph deploy \
  deployed_metal.yaml \
  -o deployed_ceph.yaml \
  --osd-spec osd_spec.yaml \
  --crush-hierarchy crush_hierarchy.yaml
```

The Ceph cluster is created with the custom OSD layout.

The example file above would result in the following OSD layout.

```
ID  CLASS  WEIGHT       TYPE NAME                  STATUS  REWEIGHT  PRI-AFF
-1         0.02939      root default
-3         0.00980      rack r0
-2         0.00980          host ceph-node-00
 0    hdd  0.00980              osd.0                 up   1.00000   1.00000
-5         0.00980      rack r1
-4         0.00980          host ceph-node-01
 1    hdd  0.00980              osd.1                 up   1.00000   1.00000
-7         0.00980      rack r2
-6         0.00980          host ceph-node-02
 2    hdd  0.00980              osd.2                 up   1.00000   1.00000
```

NOTE

Device classes are automatically detected by Ceph but CRUSH rules are associated with pools. Pools are still defined and created using the `CephCrushRules` parameter during the overcloud deployment. See Defining performance tiers for varying workloads in a Ceph Storage cluster with director

4.5. CONFIGURING SERVICE PLACEMENT OPTIONS

Service definitions in the `roles_data.yaml` file determine which bare metal instance runs which service. By default, the Controller role has the CephMon and CephMgr service while the CephStorage role has the CephOSD service. Unlike most composable services, Ceph services do not require Heat output to determine how services are configured. The `roles_data.yaml` file always determines Ceph service placement even though the deployed Ceph process occurs before Heat runs.
A custom roles file is only necessary when default role assignments are not used because of the environment. For example, when deploying hyperconverged nodes, the predeployed compute nodes should be labeled as `osd` with a service type of `osd` to have a placement list containing a list of compute instances.

**Procedure**

1. Log in to the undercloud node as the `stack` user.

2. Create a YAML format file that defines the custom roles.

3. Use the command `openstack overcloud ceph deploy` with the `--roles-data` option to deploy the configuration file.

   ```bash
   $ openstack overcloud ceph deploy \
   deployed_metal.yaml \
   -o deployed_ceph.yaml \
   --roles-data custom_roles.yaml
   ```

   **NOTE**

   The `--stack` option is used in the above example to find the working directory containing the Ansible inventory created when `openstack overcloud node provision` was run to initially provision the bare metal instances.

### 4.6. CONFIGURING SSH USER OPTIONS

Cephadm connects to all managed remote Ceph hosts using SSH. An account and SSH key pair is created on all Ceph nodes in the overcloud during cluster creation. That information is then given to Cephadm so it can communicate with the nodes. The `openstack overcloud ceph deploy` command automatically creates the user and keys and distributes them to the hosts so it is not necessary to follow the procedure below. However, it is a supported option.

#### 4.6.1. Creating the SSH user before cluster creation

You can create the SSH user before cluster creation with the `openstack overcloud ceph user enable` command.

**Procedure**

1. Log in to the undercloud node as the `stack` user.

2. Use the command `openstack overcloud ceph user enable` to create the SSH user.

   ```bash
   $ openstack overcloud ceph user enable
   ```
NOTE

The default user name is `ceph-admin`. To specify a different user name, use the `-cephadm-ssh-user` option to specify a different one.

```
openstack overcloud ceph user enable --cephadm-ssh-user <custom_user_name>
```

If you specify a custom user name, it must be used consistently where necessary. It is recommended to use the default name and not pass `--cephadm-ssh-user`.

If the user is created in advanced, use the parameter `--skip-user-create` when executing `openstack overcloud ceph deploy`.

NOTE

This command requires the path to a Ceph specification file to determine:

- Which hosts require the SSH user.
- Which hosts have the `_admin` label and require the private SSH key.
- Which hosts require the public SSH key.

For more information about specification files and how to generate them, see Generating the service specification.

4.6.2. Disabling the SSH user

Disabling the SSH user will disable Cephadm. Disabling Cephadm removes the ability of the service to administer the Ceph cluster and no associated commands will work. It also will prevent Ceph node overcloud scaling operations. It will also remove all public and private SSH keys.

Procedure

1. Log in to the undercloud node as the `stack` user.

2. Use the command `openstack overcloud ceph user disable --fsid <FSID> ceph_spec.yaml` to disable the SSH user.

NOTE

The FSID is located in the `deployed_ceph.yaml` environment file.

IMPORTANT

Disabling cephadm will disable all Ceph management features. The `openstack overcloud ceph user disable` command is not recommended unless it is necessary to disable Cephadm.

IMPORTANT

Use the `openstack overcloud ceph user enable --fsid <FSID> ceph_spec.yaml` command to enable the SSH user and Cephadm service after being disabled.
NOTE

This command requires the path to a Ceph specification file to determine:

- Which hosts require the SSH user.
- Which hosts have the _admin label and require the private SSH key.
- Which hosts require the public SSH key.

For more information about specification files and how to generate them, see Generating the service specification.

4.7. CONFIGURING THE CONTAINER REGISTRY

The undercloud can be configured to serve as a container registry for Ceph containers. By default, `openstack overcloud ceph deploy` pulls the Ceph container in the default `container_image_prepare_defaults.yaml` file. If the `push_destination` attribute is defined in the file, the overcloud is configured to access the local registry to download the Ceph container. This modifies the overcloud `/etc/hosts` and `/etc/containers/registries.conf` files unless the `--skip-hosts-config` and `-skip-container-registry-config` options are used or a `push_destination` is not defined.

NOTE

Red Hat Ceph version changes in each Red Hat OpenStack release. The current Ceph version can be viewed using this command:

```bash
$ egrep "ceph_namespace|ceph_image|ceph_tag" \
/usr/share/tripleo-common/container-images/container_image_prepare_defaults.yaml
```

Procedure

1. Log in to the undercloud node as the stack user.

2. Create a YAML format file that defines the custom container registry. The following is an example of a custom container registry file called `custom_container_image_prepare.yaml`.

   ```yaml
   ContainerImageRegistryCredentials:
   quay.io/ceph-ci:
   quay_username: quay_password
   
   custom_container_image_prepare.yaml
   ```

3. Use the `openstack overcloud ceph deploy` command with the `--container-image-prepare` option to define the custom container registry.

   ```bash
   openstack overcloud ceph deploy \
   deployed_metal.yaml \
   -o deployed_ceph.yaml \
   --container-image-prepare custom_container_image_prepare.yaml
   ```
CHAPTER 5. CUSTOMIZING THE STORAGE SERVICE

The Red Hat OpenStack Platform (RHOSP) director heat template collection already contains the necessary templates and environment files to enable a basic Ceph Storage configuration.

Director uses the `/usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm.yaml` environment file to add additional configuration to the Ceph cluster deployed by `openstack overcloud ceph deploy` and integrate it with your overcloud during deployment.

For more information about containerized services in RHOSP, see Configuring a basic overcloud with the CLI tools in Director Installation and Usage.

5.1. CONFIGURING A CUSTOM ENVIRONMENT FILE

Director applies basic, default settings to the deployed Ceph Storage cluster. You must define any additional configurations in a custom environment file.

Procedure

1. Create the file `storage-config.yaml` in `/home/stack/templates/`. In this example, the `~/templates/storage-config.yaml` file contains most of the overcloud-related custom settings for your environment. Parameters that you include in the custom environment file override the corresponding default settings from the `/usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm.yaml` file.

2. Add a `parameter_defaults` section to `~/templates/storage-config.yaml`. This section contains custom settings for your overcloud. For example, to set `vxlan` as the network type of the networking service (neutron), add the following snippet to your custom environment file:

   ```yaml
   parameter_defaults:
     NeutronNetworkType: vxlan
   ```

3. Optional: Set the following options under `parameter_defaults` according to your requirements:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CinderEnableIscsiBackend</td>
<td>Enables the iSCSI backend</td>
<td>false</td>
</tr>
<tr>
<td>CinderEnableRbdBackend</td>
<td>Enables the Ceph Storage backend</td>
<td>true</td>
</tr>
<tr>
<td>CinderBackupBackend</td>
<td>Sets ceph or swift as the backend for volume backups. For more information, see Section 5.6, “Configuring the Block Storage Backup Service to use Ceph”.</td>
<td>ceph</td>
</tr>
<tr>
<td>NovaEnableRbdBackend</td>
<td>Enables Ceph Storage for Nova ephemeral storage</td>
<td>true</td>
</tr>
</tbody>
</table>
### Option Description Default value

| GlanceBackend | Defines which back end the Image service should use: rbd (Ceph), swift, or file | rbd |

**NOTE**

You can omit an option from `~/templates/storage-config.yaml` if you intend to use the default settings.

The contents of your custom environment file change depending on the settings that you apply in the following sections. See for a completed example.

### 5.2. ENABLING CEPH METADATA SERVER

The Ceph Metadata Server (MDS) runs the ceph-mds daemon, which manages metadata related to files stored on CephFS. CephFS can be consumed through NFS. For more information about using CephFS through NFS, see File System Guide and Deploying the Shared File Systems service with CephFS through NFS.

**NOTE**

Red Hat supports deploying Ceph MDS only with the CephFS through NFS back end for the Shared File Systems service.

**Procedure**

- To enable the Ceph Metadata Server, invoke the following environment file when you create your overcloud: `/usr/share/openstack-tripleo-heat-templates/environments/cephadm/ceph-mds.yaml`

For more information, see Chapter 8, Creating the overcloud. For more information about the Ceph Metadata Server, see The Ceph File System Metadata Server in the File System Guide.

**NOTE**

By default, the Ceph Metadata Server is deployed on the Controller node. You can deploy the Ceph Metadata Server on its own dedicated node.

### 5.3. CEPH OBJECT GATEWAY OBJECT STORAGE

The Ceph Object Gateway (RGW) provides applications with an interface to access object storage capabilities within a Ceph Storage cluster.

When you use director to deploy Ceph, director automatically enables RGW, which is a direct replacement for the Object Storage service (swift). All other services that normally use Swift can start using RGW instead without additional configuration. Swift remains available as an object storage option for upgraded clusters.
Now that RGW is enabled by default, there is no requirement for a separate RGW environment file to enable it. For more information about environment files that you can pass to the `openstack overcloud deploy` command for other object storage options, see Section 5.4, “Deployment options for Red Hat OpenStack Platform object storage”.

By default, Ceph Storage allows 250 placement groups per Object Storage Daemon (OSD). When you enable RGW, Ceph Storage creates the following six additional pools that are required by RGW:

- .rgw.root
- default.rgw.control
- default.rgw.meta
- default.rgw.log
- default.rgw.buckets.index
- default.rgw.buckets.data

**NOTE**

In your deployment, `default` is replaced with the name of the zone to which the pools belong.

Additional resources

- For more information about RGW, see the Object Gateway Configuration and Administration Guide.
- For more information about using RGW instead of Swift, see the Block Storage Backup Guide.

### 5.4. DEPLOYMENT OPTIONS FOR RED HAT OPENSTACK PLATFORM OBJECT STORAGE

When you have deployed a Ceph cluster using the `openstack overcloud ceph deploy` command, you deploy the overcloud using the `openstack overcloud deploy` command. When you have deployed the overcloud, you can pass different environment files to the `openstack overcloud deploy` command for different object storage options.

#### 5.4.1. RHOSP deployment using Ceph Object Gateway (RGW)

To deploy RGW as described in Section 5.3, “Ceph Object Gateway object storage”, include the following environment file:

```
-e environments/cephadm/cephadm.yaml
```

This environment configures both Ceph block storage (RBD) and RGW.

#### 5.4.2. RHOSP deployment using Swift instead of RGW

To deploy Ceph with Swift object storage instead of RGW, include the following environment file:

```
-e environments/cephadm/cephadm-rbd-only.yaml
```
The **cephadm-rbd-only.yaml** file configures Ceph RBD but not RGW. If you use this file, object storage is still installed by default using Swift.

NOTE

If you used Swift object storage before upgrading your cluster, you can continue to use Swift instead of RGW by replacing the **environments/ceph-ansible/ceph-ansible.yaml** file with the **environments/cephadm/cephadm-rbd-only.yaml** during the upgrade. For more information, see the upgrade documentation. RHOSP does not support migration from Swift to RGW.

5.4.3. RHOSP deployment without using object storage

To deploy Ceph with RBD but not with RGW or Swift, include the following environment files:

```
-e environments/cephadm/cephadm-rbd-only.yaml
-e environments/disable-swift.yaml
```

The **cephadm-rbd-only.yaml** file configures RBD but not RGW. The **disable-swift.yaml** file ensures that the Object Storage service (swift) does not deploy.

5.5. CONFIGURING CEPH OBJECT STORE TO USE EXTERNAL CEPH OBJECT GATEWAY

Red Hat OpenStack Platform (RHOSP) director supports configuring an external Ceph Object Gateway (RGW) as an Object Store service. To authenticate with the external RGW service, you must configure RGW to verify users and their roles in the Identity service (keystone).

For more information about how to configure an external Ceph Object Gateway, see Configuring the Ceph Object Gateway to use Keystone authentication in the Using Keystone with the Ceph Object Gateway Guide.

Procedure

1. Add the following **parameter_defaults** to a custom environment file, for example, **swift-external-params.yaml**, and adjust the values to suit your deployment:

   ```yaml
   parameter_defaults:
     ExternalSwiftPublicUrl: 'http://<Public RGW endpoint or loadbalancer>:8080/swift/v1/AUTH_%(project_id)s'
     ExternalSwiftInternalUrl: 'http://<Internal RGW endpoint>:8080/swift/v1/AUTH_%(project_id)s'
     ExternalSwiftAdminUrl: 'http://<Admin RGW endpoint>:8080/swift/v1/AUTH_%(project_id)s'
     ExternalSwiftUserTenant: 'service'
     SwiftPassword: 'choose_a_random_password'
   ```
NOTE

The example code snippet contains parameter values that might differ from values that you use in your environment:

- The default port where the remote RGW instance listens is **8080**. The port might be different depending on how the external RGW is configured.

- The **swift** user created in the overcloud uses the password defined by the `SwiftPassword` parameter. You must configure the external RGW instance to use the same password to authenticate with the Identity service by using the `rgw_keystone_admin_password`.

2. Add the following code to the Ceph config file to configure RGW to use the Identity service. Replace the variable values to suit your environment:

```yaml
rgw_keystone_api_version = 3
rgw_keystone_url = http://<public Keystone endpoint>:5000/
rgw_keystone_accepted_roles = member, Member, admin
rgw_keystone_accepted_admin_roles = ResellerAdmin, swiftoperator
rgw_keystone_admin_domain = default
rgw_keystone_admin_project = service
rgw_keystone_admin_user = swift
rgw_keystone_admin_password =
<password_as_defined_in_the_environment_parameters>
rgw_keystone_implicit_tenants = true
rgw_keystone_revocation_interval = 0
rgw_s3_auth_use_keystone = true
rgw_swift_versioning_enabled = true
rgw_swift_account_in_url = true
```

NOTE

Director creates the following roles and users in the Identity service by default:

- `rgw_keystone_accepted_admin_roles`: ResellerAdmin, swiftoperator
- `rgw_keystone_admin_domain`: default
- `rgw_keystone_admin_project`: service
- `rgw_keystone_admin_user`: swift

3. Deploy the overcloud with the additional environment files with any other environment files that are relevant to your deployment:

```bash
openstack overcloud deploy --templates \  
-e <your_environment_files> 
-e /usr/share/openstack-tripleo-heat-templates/environments/swift-external.yaml 
-e swift-external-params.yaml
```

Verification

1. Log in to the undercloud as the **stack** user.
2. Source the `overcloudrc` file:

   ```bash
   $ source ~/stackrc
   ```

3. Verify that the endpoints exist in the Identity service (keystone):

   ```bash
   $ openstack endpoint list --service object-store
   +---------+-----------+-------+-------+---------+-----------+---------------+
   | ID      | Region    | Service Name | Service Type | Enabled | Interface | URL                     |
   +---------+-----------+-------+-------+---------+-----------+---------------+
   | 233b7ea32aaf40c1ad782c696128aa0e | regionOne | swift | object-store | True    | admin     | http://192.168.24.3:8080/v1/AUTH_%(project_id)s |
   | 4ccde35ac76444d7bb82c5816a97abd8 | regionOne | swift | object-store | True    | public    | https://192.168.24.2:13808/v1/AUTH_%(project_id)s |
   | b4ff283f445348639864f560aa2b2b41 | regionOne | swift | object-store | True    | internal  | http://192.168.24.3:8080/v1/AUTH_%(project_id)s |
   +---------+-----------+-------+-------+---------+-----------+---------------+
   ```

4. Create a test container:

   ```bash
   $ openstack container create <testcontainer>
   +----------------+---------------+------------------------------------+
   | account | container | x-trans-id |
   +----------------+---------------+------------------------------------+
   | AUTH_2852da3cf2fc49008114c434d1fc157 | testcontainer | tx6f5253e710a2449b8ef7e-005f2d29e8 |
   +----------------+---------------+------------------------------------+
   ```

5. Create a configuration file to confirm that you can upload data to the container:

   ```bash
   $ openstack object create testcontainer undercloud.conf
   +-----------------+---------------+----------------------------------+
   | object          | container     | etag                             |
   +-----------------+---------------+----------------------------------+
   | undercloud.conf | testcontainer | 09fcffe126cac1dbac7b89b8fd7a3e4b |
   +-----------------+---------------+----------------------------------+
   ```

6. Delete the test container:

   ```bash
   $ openstack container delete -r <testcontainer>
   ```

5.6. CONFIGURING THE BLOCK STORAGE BACKUP SERVICE TO USE CEPH

The Block Storage Backup service (``cinder-backup``) is disabled by default. To enable the Block Storage Backup service, complete the following steps:

**Procedure**

- Invoke the following environment file when you create your overcloud: `/usr/share/openstack-tripleo-heat-templates/environments/cinder-backup.yaml`. 

5.7. CONFIGURING MULTIPLE BONDED INTERFACES FOR CEPH NODES

Use a bonded interface to combine multiple NICs and add redundancy to a network connection. If you have enough NICs on your Ceph nodes, you can create multiple bonded interfaces on each node to expand redundancy capability.

You can then use a bonded interface for each network connection that the node requires. This provides both redundancy and a dedicated connection for each network.

The simplest implementation of bonded interfaces involves the use of two bonds, one for each storage network used by the Ceph nodes. These networks are the following:

**Front-end storage network (StorageNet)**

The Ceph client uses this network to interact with the corresponding Ceph cluster.

**Back-end storage network (StorageMgmtNet)**

The Ceph cluster uses this network to balance data in accordance with the placement group policy of the cluster. For more information, see [Ceph placement groups](#) in the [Red Hat Ceph Architecture Guide](#).

To configure multiple bonded interfaces, you must create a new network interface template, as the director does not provide any sample templates that you can use to deploy multiple bonded NICs. However, the director does provide a template that deploys a single bonded interface. This template is `/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml`. You can define an additional bonded interface for your additional NICs in this template.

The following snippet contains the default definition for the single bonded interface defined in the `/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml` file:

```yaml
- type: ovs_bridge // 1
  name: br-bond
  members:
    - type: ovs_bond // 2
      name: bond1 // 3
      ovs_options: {get_param: BondInterfaceOvsOptions} // 4
      members:
        - type: interface
          name: nic2
          primary: true
        - type: interface
          name: nic3
    - type: vlan // 6
      device: bond1 // 7
      vlan_id: {get_param: StorageNetworkVlanID}
      addresses:
        - ip_netmask: {get_param: StorageIpSubnet}
    - type: vlan
```
A single bridge named `br-bond` holds the bond defined in this template. This line defines the bridge type, namely OVS.

The first member of the `br-bond` bridge is the bonded interface itself, named `bond1`. This line defines the bond type of `bond1`, which is also OVS.

The default bond is named `bond1`.

The `ovs_options` entry instructs director to use a specific set of bonding module directives. Those directives are passed through the `BondInterfaceOvsOptions`, which you can also configure in this file. For more information about configuring bonding module directives, see Section 5.8, “Configuring bonding module directives”.

The `members` section of the bond defines which network interfaces are bonded by `bond1`. In this example, the bonded interface uses `nic2` (set as the primary interface) and `nic3`.

The `br-bond` bridge has two other members: a VLAN for both front-end (`StorageNetwork`) and back-end (`StorageMgmtNetwork`) storage networks.

The `device` parameter defines which device a VLAN should use. In this example, both VLANs use the bonded interface, `bond1`.

With at least two more NICs, you can define an additional bridge and bonded interface. Then, you can move one of the VLANs to the new bonded interface, which increases throughput and reliability for both storage network connections.

When you customize the `/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml` file for this purpose, Red Hat recommends that you use Linux bonds (`type: linux_bond`) instead of the default OVS (`type: ovs_bond`). This bond type is more suitable for enterprise production deployments.

The following edited snippet defines an additional OVS bridge (`br-bond2`) which houses a new Linux bond named `bond2`. The `bond2` interface uses two additional NICs, `nic4` and `nic5`, and is used solely for back-end storage network traffic:
name: nic3
- type: vlan
device: bond1
  vlan_id: {get_param: StorageNetworkVlanID}
  addresses:
    - ip_netmask: {get_param: StorageIpSubnet}

- type: ovs_bridge
name: br-bond2
members:
  - type: linux_bond
    name: bond2
    bonding_options: {get_param: BondInterfaceOvsOptions}
members:
  - type: interface
    name: nic4
    primary: true
  - type: interface
    name: nic5

- type: vlan
device: bond1
  vlan_id: {get_param: StorageMgmtNetworkVlanID}
  addresses:
    - ip_netmask: {get_param: StorageMgmtIpSubnet}

As bond1 and bond2 are both Linux bonds (instead of OVS), they use bonding_options instead of ovsg_options to set bonding directives. For more information, see Section 5.8, “Configuring bonding module directives”.

For the full contents of this customized template, see Section A.1, “Multiple bonded interfaces”.

5.8. CONFIGURING BONDING MODULE DIRECTIVES

After you add and configure the bonded interfaces, use the BondInterfaceOvsOptions parameter to set the directives that you want each bonded interface to use. You can find this information in the parameters: section of the /usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml file. The following snippet shows the default definition of this parameter (namely, empty):

BondInterfaceOvsOptions:
  default: "
  description: The ovsg_options string for the bond interface. Set things like lacp=active and/or bond_mode=balance-slb using this option.
  type: string
Define the options you need in the **default** line. For example, to use 802.3ad (mode 4) and a LACP rate of 1 (fast), use `mode=4 lacp_rate=1`:

```yaml
BondInterfaceOvsOptions:
  default: 'mode=4 lacp_rate=1'
  description: The bonding_options string for the bond interface. Set things like lacp=active and/or bond_mode=balance-slb using this option.
  type: string
```

Additional resources

- For the full contents of the customized `/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml` template, see Section A.1, "Multiple bonded interfaces".
CHAPTER 6. CUSTOMIZING THE CEPH STORAGE CLUSTER

Red Hat OpenStack Platform (RHOSP) director uses a default configuration to deploy containerized Red Hat Ceph Storage. You can customize Ceph Storage by overriding the default settings.

Prerequisites

- The servers should be deployed and their storage networks configured.
- The deployed bare metal file as output by `openstack overcloud node provision -o ~/deployed_metal.yaml` ...

6.1. CONFIGURATION OPTIONS

Red Hat Ceph Storage provides several options for customizing the initial configuration of the cluster.

Initial Red Hat Ceph Storage configuration can be applied using a standard format initialization (ini) file with a `.conf` extension.

Procedure

1. Log in to the undercloud node as the `stack` user.
2. Optional: Use a standard format initialization (ini) file to configure the cluster.
   a. Create the file with configuration options. The following is an example of a simple configuration file:

   ```
   [global]
   osd crush chooseleaf type = 0
   log_file = /var/log/ceph/$cluster-$type.$id.log
   
   [mon]
   mon_cluster_log_to_syslog = true
   ```
   b. Use the `openstack overcloud ceph deploy --config <configuration_file_name>` command to deploy the configuration.

   ```
   $ openstack overcloud ceph deploy --config initial-ceph.conf
   ```

   **IMPORTANT**

   The `deployed_ceph.yaml` environment file output by the `openstack overcloud ceph deploy` command has the `ApplyCephConfigOverridesOnUpdate` attribute set to `true`. This value allows services not covered by during the initial cluster creation, such as RGW, to be set during the overcloud deployment. After the overcloud deployment is complete, update the `deployed_ceph.yaml` file, or the file used for a similar purpose in your environment, to set the `ApplyCephConfigOverridesOnUpdate` to `false`. Subsequent Ceph configuration changes made during the operation of your environment should be made with the `ceph config` command. For more information on the `ApplyCephConfigOverridesOnUpdate` and `CephConfigOverrides` parameters see Overcloud Parameters.
3. Optional: Use the command `openstack overcloud ceph deploy --single-host-defaults` to configure the Ceph cluster to run on a single instance.

4. Optional: Use the command `openstack overcloud ceph deploy --force --cephadm-extra-args "optional_arguments"` to pass any additional configuration values to the `cephadm bootstrap` command. If the arguments `--log-to-file` and `--skip-prepare-host` are required, the command `openstack overcloud ceph deploy --force --cephadm-extra-args "--log-to-file --skip-prepare-host"` would be used. The underlying `cephadm bootstrap` command `cephadm bootstrap --log-to-file --skip-prepare-host` would be executed.

**NOTE**
The `--force` option is required when using `--cephadm-extra-args` because not all possible options ensure a functional deployment.

### 6.2. GENERATING THE SERVICE SPECIFICATION (OPTIONAL)

The Red Hat Ceph Storage cluster service specification is a YAML file that describes the deployment of Ceph services. It is automatically generated by tripleo before the cluster is deployed and does not typically have to be generated separately.

A custom service specification can be created to customize the cluster.

**Procedure**

1. Log in to the undercloud node as the `stack` user.

2. Use the command `openstack overcloud ceph spec` command to generate the specification file. The following example would output the specification file at location specified by the `-o` switch.

   ```bash
   openstack overcloud ceph spec -o '~/ceph_spec.yaml'
   ```

3. Edit the generated file with the required configuration.

4. Use the command `openstack overcloud ceph deploy` command to deploy the custom service specification.

   ```bash
   openstack overcloud ceph deploy \
   deployed_metal.yaml \
   -o deployed_ceph.yaml --ceph-spec '~/ceph_spec.yaml'
   ```

### 6.3. CEPH CONTAINERS FOR RED HAT OPENSTACK PLATFORM WITH CEPH STORAGE

A Ceph container is required to configure RHOSP to use Ceph, even with an external Ceph cluster. To be compatible with Red Hat Enterprise Linux 8, Red Hat OpenStack Platform (RHOSP) 16 requires Red Hat Ceph Storage 4. The Ceph Storage 4 container is hosted at `registry.redhat.io`, a registry that requires authentication.

You can use the heat parameter `ContainerImageRegistryCredentials` to authenticate at `registry.redhat.io`. For more information, see Container image preparation parameters.
6.4. CONFIGURING ADVANCED OSD SPECIFICATIONS

All disks, except for the one where the operating system is installed, are used as OSDs by default. This is because the default OSD specification file has the following definition.

```yaml
data_devices:
  all: true
```

The `data_devices` attribute is one of several OSD-related attributes used to configure OSDs in the Ceph service specification. See OSD Service Specification lists the attributes available for OSD configuration.

Procedure

1. Log in to the undercloud node as the `stack` user.

2. Create a YAML format file called that defines the OSD specification. The following is an example of a custom OSD specification.

```yaml
data_devices:
  rotational: 1
db_devices:
  rotational: 0
```

This example would create a OSD specification where all rotating devices will be data devices and all non-rotating devices will be used as shared devices. This is because when the dynamic Ceph service specification is built, whatever is in the specification file is appended to the section of the specification if the service_type is `osd`.

3. Use the command `openstack overcloud ceph deploy \ --osd-spec <osd_specification_file>` to deploy the configuration.

```
$ openstack overcloud ceph deploy \ --osd_spec osd_spec.yaml \
```

**NOTE**

The Provisioning of the node-specific Hieradata feature is not supported by cephadm integration. The OSD Service Specification has a `host_pattern` attribute that specifies which host to target for certain `data_devices` definitions.

The following is another example of an `osd_spec.yaml` file:

```yaml
data_devices:
  model: 'SAMSUNG'
osds_per_device: 2
```

In this example, the `model` parameter is used with `data_devices` to only create OSDs on that model of device, with two OSDs per device. The `osds_per_device` parameter is not under `data_devices`.

6.5. MIGRATING FROM NODE-SPECIFIC OVERRIDES

Prior to Red Hat Openstack Platform 17.0, node-specific overrides were used to manage non-homogenous server hardware. This is now accomplished using a custom OSD specification file. See
Configuring advanced OSD specifications for information on how to create a custom OSD specification file.

6.6. ENABLING CEPH ON-WIRE ENCRYPTION

You can enable encryption for all Ceph traffic over the network with the introduction of the messenger version 2 protocol. The secure mode setting for messenger v2 encrypts communication between Ceph daemons and Ceph clients, giving you end-to-end encryption.

Procedure

1. Configure the initial ceph.conf with the directives described in Enabling the messenger v2 protocol in the Red Hat Ceph Storage Data Security and Hardening Guide.

For more information about Ceph on-wire encryption, see Ceph on-wire encryption in the Red Hat Ceph Storage Architecture Guide.
CHAPTER 7. USING DIRECTOR TO DEFINE PERFORMANCE TIERS FOR VARYING WORKLOADS

You can use Red Hat OpenStack Platform (RHOSP) director to deploy different Red Hat Ceph Storage performance tiers. You can combine Ceph CRUSH rules and the `CephPools` director parameter to use the device classes feature and build different tiers to accommodate workloads that have different performance requirements. For example, you can define a HDD class for normal workloads and an SSD class that distributes data only over SSDs for high performance loads. In this scenario, when you create a new Block Storage volume, you can choose the performance tier, either HDDs or SSDs.

**NOTE**

See [Configuring CRUSH hierarchies](#) for information on CRUSH rule creation.

**WARNING**

Defining performance tiers in an existing environment might result in massive data movement in the Ceph cluster. `cephadm`, which director triggers during the stack update, does not have logic to verify whether a pool is already defined in the cluster and if it contains data. This means that defining performance tiers in an existing environment can be dangerous because the change of the default CRUSH rule that is associated with a pool results in data movement. If you require assistance or recommendations for adding or removing nodes, contact Red Hat support.

**NOTE**

Ceph autodetects the disk type and assigns it to the corresponding device class, either HDD, SSD, or NVMe based on the hardware properties exposed by the Linux kernel.

**Prerequisites**

- For new deployments, Red Hat Ceph Storage (RHCS) version 5.2 or later.

### 7.1. CONFIGURING PERFORMANCE TIERS

To deploy different Red Hat Ceph Storage performance tiers, create a new environment file that contains the CRUSH map details and include it in the deployment command. Director does not expose specific parameters for this feature, but you can generate the `tripleo-ansible` expected variables.

**NOTE**

Performance tier configuration can be combined with CRUSH hierarchies. See [Configuring CRUSH hierarchies](#) for information on CRUSH rule creation.

In the example procedure, each Ceph Storage node contains three OSDs: `sdb` and `sdc` are spinning disks and `sdc` is a SSD. Ceph automatically detects the correct disk type. You then configure two CRUSH rules, HDD and SSD, to map to the two respective device classes.
NOTE

The HDD rule is the default and applies to all pools unless you configure pools with a different rule.

Finally, you create an extra pool called **fastpool** and map it to the SSD rule. This pool is ultimately exposed through a Block Storage (cinder) back end. Any workload that consumes this Block Storage back end is backed by SSD for fast performances only. You can leverage this for either data or boot from volume.

**WARNING**

Defining performance tiers in an existing environment might result in massive data movement in the Ceph cluster. **cephadm**, which director triggers during the stack update, does not have logic to verify whether a pool is already defined in the cluster and if it contains data. This means that defining performance tiers in an existing environment can be dangerous because the change of the default CRUSH rule that is associated with a pool results in data movement. If you require assistance or recommendations for adding or removing nodes, contact Red Hat support.

**Procedure**

1. Log in to the undercloud node as the **stack** user.

2. Create an environment file, such as `/home/stack/templates/ceph-config.yaml`, to contain the Ceph config parameters and the device classes variables. Alternatively, you can add the following configurations to an existing environment file.

3. Add the **CephCrushRules** parameters. The **crush_rules** parameter must contain a rule for each class that you define or that Ceph detects automatically. When you create a new pool, if no rule is specified, the rule that you want Ceph to use as the default is selected.

   ```yaml
   CephCrushRules:
   crush_rules:
   - name: HDD
     root: default
     type: host
     class: hdd
     default: true
   - name: SSD
     root: default
     type: host
     class: ssd
     default: false
   ```

4. Add the **CephPools** parameter:

   - Use the **rule_name** parameter to specify the tier for each pool that does not use the default rule. In the following example, the **fastpool** pool uses the SSD device class that is configured as a fast tier, to manage Block Storage volumes.

   ```yaml
   - name: fastpool
     role: block
     device_class: ssd
     rule_name: SSD
   ```

   - Replace `<appropriate_PG_num>` with the appropriate number of placement groups (PGs). Alternatively, use the placement group auto-scaler to calculate the number of PGs for the Ceph pools.

   - Use the **CinderRbdExtraPools** parameter to configure **fastpool** as a Block Storage back end.
CephPools:
- name: fastpool
  rule_name: SSD
  application: rbd
CinderRbdExtraPools: fastpool

5. Use the following example to ensure that your environment file contains the correct values:

```yaml
parameter_defaults:
  crush_rules:
    - name: HDD
      root: default
      type: host
      class: hdd
      default: true
    - name: SSD
      root: default
      type: host
      class: ssd
      default: false
  CinderRbdExtraPools: fastpool
  CephPools:
    - name: fastpool
      rule_name: SSD
      application: rbd
```

6. Include the new environment file in the `openstack overcloud deploy` command.

```bash
$ openstack overcloud deploy \\
  --templates \\
  ... \\
  -e <other_overcloud_environment_files> \\
  -e /home/stack/templates/ceph-config.yaml \\
  ...
```

Replace `<other_overcloud_environment_files>` with the list of other environment files that are part of your deployment.
IMPORTANT

If you apply the environment file to an existing Ceph cluster, the pre-existing Ceph pools are not updated with the new rules. For this reason, you must enter the following command after the deployment completes to set the rules to the specified pools.

```
$ ceph osd pool set <pool> crush_rule <rule>
```

- Replace `<pool>` with the name of the pool that you want to apply the new rule to.
- Replace `<rule>` with one of the rule names that you specified with the `crush_rules` parameter.

For every rule that you change with this command, update the existing entry or add a new entry in the `CephPools` parameter in your existing templates:

```
CephPools:
  - name: <pool>
    rule_name: <rule>
    application: rbd
```

7.2. VERIFYING CRUSH RULES AND POOLS

Verify your CRUSH rules and pools settings.

WARNING

Defining performance tiers in an existing environment might result in massive data movement in the Ceph cluster. `tripleo-ansible`, which director triggers during the stack update, does not have logic to check if a pool is already defined in the cluster and if it contains data. This means that defining performance tiers in an existing environment can be dangerous because the change of the default CRUSH rule that is associated with a pool results in data movement. If you require assistance or recommendations for adding or removing nodes, contact Red Hat support.

Procedure

1. Log in to the overcloud Controller node as the `heat-admin` user.

2. To verify that your OSD tiers are successfully set, enter the following command.

   ```
   $ sudo cephadm shell ceph osd tree
   ```

3. In the resulting tree view, verify that the `CLASS` column displays the correct device class for each OSD that you set.

4. Also verify that the OSDs are correctly assigned to the device classes with the following command.

   ```
   $ sudo cephadm shell ceph osd crush tree --show-shadow
   ```

5. Compare the resulting hierarchy with the results of the following command to ensure that the same values apply for each rule.

   ```
   $ sudo cephadm shell ceph osd crush rule dump <rule_name>
   ```
6. Verify that the rules name and ID that you created are correct according to the `crush_rules` parameter that you used during deployment.

   ```
   $ sudo cephadm shell ceph osd crush rule dump | grep -E "rule_(id|name)"
   ```

7. Verify that the Ceph pools are tied to the correct CRUSH rule ID that you retrieved in Step 3.

   ```
   $ sudo cephadm shell -- ceph osd dump | grep pool
   ```

8. For each pool, ensure that the rule ID matches the rule name that you expect.
CHAPTER 8. CREATING THE OVERCLOUD

When your custom environment files are ready, you can specify the nodes that each role uses and then execute the deployment.

8.1. ASSIGNING NODES AND FLAVORS TO ROLES

Planning an overcloud deployment involves specifying how many nodes and which flavors to assign to each role. Like all heat template parameters, these role specifications are declared in the `parameter_defaults` section of your custom environment file, in this case, `~/home/stack/templates/ceph-config.yaml`.

Use the parameters in the following table to configure roles:

<table>
<thead>
<tr>
<th>Heat template parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ControllerCount</td>
<td>The number of Controller nodes to scale out</td>
</tr>
<tr>
<td>OvercloudControlFlavor</td>
<td>The flavor to use for Controller nodes (control)</td>
</tr>
<tr>
<td>ComputeCount</td>
<td>The number of Compute nodes to scale out</td>
</tr>
<tr>
<td>OvercloudComputeFlavor</td>
<td>The flavor to use for Compute nodes (compute)</td>
</tr>
</tbody>
</table>

For example, to configure the overcloud to deploy three nodes for each role, Controller and Compute, add the following to `parameter_defaults`:

```
parameter_defaults:
  ControllerCount: 3
  ComputeCount: 3
  OvercloudControlFlavor: control
  OvercloudComputeFlavor: compute
```

NOTE

For more information and a list of heat template parameters, see Creating the Overcloud with the CLI Tools in the Director Installation and Usage guide.

8.2. ASSIGNING NODES TO ROLES

Planning an overcloud deployment involves specifying how nodes are assigned to each role. Like all Heat template parameters, these role specifications are declared in the `parameter_defaults` section of your environment file (in this case, `~/templates/storage-config.yaml`).

See Designating nodes for Red Hat Ceph Storage for information on deploying Ceph nodes.

See Configuring service placement options for information on deploying a custom roles file.
8.3. INITIATING OVERCLOUD DEPLOYMENT

To implement the changes you made to your Red Hat OpenStack Platform (RHOSP) environment, you must deploy the overcloud.

Prerequisites

- During undercloud installation, set `generate_service_certificate=false` in the `undercloud.conf` file. Otherwise, you must inject a trust anchor when you deploy the overcloud.

NOTE

If you want to add Ceph Dashboard during your overcloud deployment, see Chapter 9, *Adding the Red Hat Ceph Storage Dashboard to an overcloud deployment*.

Procedure

- Deploy the overcloud. The deployment command requires additional arguments, for example:

  ```
  $ openstack overcloud deploy --templates -r /home/stack/templates/roles_data_custom.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/ceph-mds.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/cinder-backup.yaml \
  -e /home/stack/templates/storage-config.yaml \
  -e /home/stack/templates/deployed-ceph.yaml \
  --ntp-server pool.ntp.org
  ```

  The example command uses the following options:

  - `-templates` - Creates the overcloud from the default heat template collection, `/usr/share/openstack-tripleo-heat-templates/`.

  - `-r /home/stack/templates/roles_data_custom.yaml` - Specifies the customized roles definition file from Chapter 3, *Deploying Ceph services on dedicated nodes*, which adds custom roles for either Ceph MON or Ceph MDS services. These roles allow either service to be installed on dedicated nodes.

  - `-e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm.yaml` - Sets the director to finalize the previously deployed Ceph Storage cluster. This environment file deploys RGW by default. It also creates pools, keys, and daemons. If you do not want to deploy RGW or object storage, see the options described in Section 5.4, “Deployment options for Red Hat OpenStack Platform object storage”

  - `-e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/ceph-mds.yaml` - Enables the Ceph Metadata Server, as described in Section 5.2, “Enabling Ceph Metadata Server”.  

```
- `e /usr/share/openstack-tripleo-heat-templates/environments/cinder-backup.yaml` - Enables the Block Storage Backup service (*cinder-backup*), as described in Section 5.6, “Configuring the Block Storage Backup Service to use Ceph”.

- `e /home/stack/templates/storage-config.yaml` - Adds the environment file that contains your custom Ceph Storage configuration.

- `e /home/stack/templates/deployed-ceph.yaml` - Adds the environment file that contains your Ceph cluster settings, as output by the `openstack overcloud ceph deploy` command run earlier.

- `--ntp-server pool.ntp.org` - Sets the NTP server.

**NOTE**

For a full list of options, run the `openstack help overcloud deploy` command.

For more information, see Configuring a basic overcloud with the CLI tools in the Director Installation and Usage guide.

### 8.4. Monitoring the Status of the Overcloud Creation Process

After you initiate the overcloud creation process, director provisions your nodes. This process takes some time to complete, but you can monitor the status of the process in a separate terminal.

**Procedure**

- To view the status of the overcloud creation, open a separate terminal as the *stack* user and enter the following commands:

  ```bash
  $ source ~/stackrc
  $ openstack stack list --nested
  ```
Red Hat Ceph Storage Dashboard is disabled by default but you can enable it in your overcloud with the Red Hat OpenStack Platform (RHOSP) director. The Ceph Dashboard is a built-in, web-based Ceph management and monitoring application that administers various aspects and objects in your cluster. Red Hat Ceph Storage Dashboard comprises the following components:

- The Ceph Dashboard manager module provides the user interface and embeds the platform front end, Grafana.
- Prometheus, the monitoring plugin.
- Alertmanager sends alerts to the Dashboard.
- Node Exporters export cluster data to the Dashboard.

**NOTE**

This feature is supported with Ceph Storage 4.1 or later. For more information about how to determine the version of Ceph Storage installed on your system, see Red Hat Ceph Storage releases and corresponding Ceph package versions.

**NOTE**

The Red Hat Ceph Storage Dashboard is always colocated on the same nodes as the other Ceph manager components.

**NOTE**

If you want to add Ceph Dashboard during your initial overcloud deployment, complete the procedures in this chapter before you deploy your initial overcloud in Chapter 8, Creating the overcloud.

The following diagram shows the architecture of Ceph Dashboard on Red Hat OpenStack Platform:
For more information about the Dashboard and its features and limitations, see Dashboard features in the Red Hat Ceph Storage Dashboard Guide.

9.1. TLS EVERYWHERE WITH CEPH DASHBOARD

The Dashboard front end is fully integrated with the TLS everywhere framework. You can enable TLS
everywhere provided that you have the required environment files and they are included in the overcloud deploy command. This triggers the certificate request for both Grafana and the Ceph Dashboard and the generated certificate and key files are passed to cephadm during the overcloud deployment. For instructions and more information about how to enable TLS for the Dashboard as well as for other RHOSP services, see the following topics in the Advanced Overcloud Customization guide:

NOTE

The port to reach the Ceph Dashboard remains the same even in the TLS-everywhere context.

9.2. INCLUDING THE NECESSARY CONTAINERS FOR THE CEPH DASHBOARD

Before you can add the Ceph Dashboard templates to your overcloud, you must include the necessary containers by using the containers-prepare-parameter.yaml file. To generate the containers-prepare-parameter.yaml file to prepare your container images, complete the following steps:

Procedure

1. Log in to your undercloud host as the stack user.

2. Generate the default container image preparation file:

   ```bash
   $ sudo openstack tripleo container image prepare default \
   --local-push-destination \
   --output-env-file containers-prepare-parameter.yaml
   ```

3. Edit the containers-prepare-parameter.yaml file and make the modifications to suit your requirements. The following example containers-prepare-parameter.yaml file contains the image locations and tags related to the Dashboard services including Grafana, Prometheus, Alertmanager, and Node Exporter. Edit the values depending on your specific scenario:

   ```yaml
   parameter_defaults:
   ContainerImagePrepare:
   - push_destination: true
     set:
     - ceph_alertmanager_image: ose-prometheus-alertmanager
       ceph_alertmanager_namespace: registry.redhat.io/openshift4
       ceph_alertmanager_tag: v4.1
       ceph_grafana_image: rhceph-5-dashboard-rhel8
       ceph_grafana_namespace: registry.redhat.io/rhceph
       ceph_grafana_tag: 4
       ceph_image: rhceph-5-rhel8
       ceph_namespace: registry.redhat.io/rhceph
       ceph_node_exporter_image: ose-prometheus-node-exporter
       ceph_node_exporter_namespace: registry.redhat.io/openshift4
       ceph_node_exporter_tag: v4.1
       ceph_prometheus_image: ose-prometheus
       ceph_prometheus_namespace: registry.redhat.io/openshift4
       ceph_prometheus_tag: v4.1
       ceph_tag: latest
   ```
For more information about registry and image configuration with the `containers-prepare-parameter.yaml` file, see Container image preparation parameters in the Transitioning to Containerized Services guide.

9.3. DEPLOYING CEPH DASHBOARD

Include the ceph-dashboard environment file to deploy the Ceph Dashboard.

**NOTE**

If you want to deploy Ceph Dashboard with a composable network, see Section 9.4, “Deploying Ceph Dashboard with a composable network”.

**NOTE**

The Ceph Dashboard admin user role is set to read-only mode by default. To change the Ceph Dashboard admin default mode, see Section 9.5, “Changing the default permissions”.

Procedure

1. Log in to the undercloud node as the stack user.

2. Optional: The Ceph Dashboard network is set by default to the provisioning network. If you want to deploy the Ceph Dashboard and access it through a different network, create an environment file, for example: `ceph_dashboard_network_override.yaml`. Set `CephDashboardNetwork` to one of the existing overcloud routed networks, for example `ctlplane`:

   ```yaml
   parameter_defaults:
       CephDashboardNetwork: ctlplane
   ```

3. Include the following environment files in the openstack overcloud deploy command. Include all environment files that are part of your deployment, and the `ceph_dashboard_network_override.yaml` file if you chose to change the default network:

   ```bash
   $ openstack overcloud deploy \
   --templates \
   -e <overcloud_environment_files> \
   -e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm.yaml \
   -e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/ceph-dashboard.yaml \
   -e ceph_dashboard_network_override.yaml
   ```

   Replace `<overcloud_environment_files>` with the list of environment files that are part of your deployment.

Result

The resulting deployment comprises an external stack with the grafana, prometheus, alertmanager, and node-exporter containers. The Ceph Dashboard manager module is the back end for this stack, and it embeds the grafana layouts to provide ceph cluster specific metrics to the end users.
9.4. DEPLOYING CEPH DASHBOARD WITH A COMPOSABLE NETWORK

You can deploy the Ceph Dashboard on a composable network instead of on the default Provisioning network. This eliminates the need to expose the Ceph Dashboard service on the Provisioning network. When you deploy the Dashboard on a composable network, you can also implement separate authorization profiles.

You must choose which network to use before you deploy because you can apply the Dashboard to a new network only when you first deploy the overcloud. Use the following procedure to choose a composable network before you deploy.

Procedure

1. Log in to the undercloud as the stack user.

2. Generate the Controller specific role to include the Dashboard composable network:

   ```bash
   $ openstack overcloud roles generate -o /home/stack/roles_data_dashboard.yaml
   ControllerStorageDashboard Compute BlockStorage ObjectStorage CephStorage
   ```

   **Result**

   - A new `ControllerStorageDashboard` role is generated inside the `roles_data.yaml` defined as the output of the command. You must include this file in the template list when you use the overcloud deploy command.

     **NOTE:** The `ControllerStorageDashboard` role does not contain `CephNFS` nor `network_data_dashboard.yaml`.

   - Director provides a network environment file where the composable network is defined. The default location of this file is `/usr/share/openstack-tripleo-heat-templates/network_data_dashboard.yaml`. You must include this file in the overcloud template list when you use the overcloud deploy command.

3. Include the following environment files, with all environment files that are part of your deployment, in the `openstack overcloud deploy` command:

   ```bash
   $ openstack overcloud deploy \
   --templates \ 
   -r /home/stack/roles_data.yaml \ 
   -n /usr/share/openstack-tripleo-heat-templates/network_data_dashboard.yaml \ 
   -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \ 
   -e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \ 
   -e <overcloud_environment_files> \ 
   -e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm.yaml \ 
   -e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/ceph-dashboard.yaml
   ```

   Replace `<overcloud_environment_files>` with the list of environment files that are part of your deployment.

   **Result**

   The resulting deployment comprises an external stack with the grafana, prometheus, alertmanager, and node-exporter containers. The Ceph Dashboard manager module is the
back end for this stack, and it embeds the grafana layouts to provide Ceph cluster-specific metrics to the end users.

9.5. CHANGING THE DEFAULT PERMISSIONS

The Ceph Dashboard admin user role is set to read-only mode by default for safe monitoring of the Ceph cluster. To permit an admin user to have elevated privileges so that they can alter elements of the Ceph cluster with the Dashboard, you can use the `CephDashboardAdminRO` parameter to change the default admin permissions.

**WARNING**

A user with full permissions might alter elements of your cluster that director configures. This can cause a conflict with director-configured options when you run a stack update. To avoid this problem, do not alter director-configured options with Ceph Dashboard, for example, Ceph OSP pools attributes.

**Procedure**

1. Log in to the undercloud as the **stack** user.
2. Create the following `ceph_dashboard_admin.yaml` environment file:

   ```
   parameter_defaults:
     CephDashboardAdminRO: false
   ```
3. Run the overcloud deploy command to update the existing stack and include the environment file you created with all other environment files that are part of your existing deployment:

   ```
   $ openstack overcloud deploy \
   --templates \
   -e <existing_overcloud_environment_files> \
   -e ceph_dashboard_admin.yml
   ```
   Replace `<existing_overcloud_environment_files>` with the list of environment files that are part of your existing deployment.

9.6. ACCESSING CEPH DASHBOARD

To test that Ceph Dashboard is running correctly, complete the following verification steps to access it and check that the data it displays from the Ceph cluster is correct.

**Procedure**

1. Log in to the undercloud node as the **stack** user.
2. Retrieve the dashboard admin login credentials:
3. Retrieve the VIP address to access the Ceph Dashboard:

   ```bash
   [stack@undercloud ~]$ grep tripleo_cephadm_dashboard_admin_password <config-download>/<stack>/cephadm/cephadm-extra-vars-heat.yml
   ```

4. Use a web browser to point to the front end VIP and access the Dashboard. Director configures and exposes the Dashboard on the provisioning network, so you can use the VIP that you retrieved to access the Dashboard directly on TCP port 8444. Ensure that the following conditions are met:

   - The Web client host is layer 2 connected to the provisioning network.
   - The provisioning network is properly routed or proxied, and it can be reached from the web client host. If these conditions are not met, you can still open a SSH tunnel to reach the Dashboard VIP on the overcloud:

     ```bash
     client_host$ ssh -L 8444:<dashboard_vip>:8444 stack@<your undercloud>
     ```

     Replace `<dashboard_vip>` with the IP address of the control plane VIP that you retrieved.

5. To access the Dashboard, go to: `http://localhost:8444` in a web browser and log in with the following details:

   - The default user that `cephadm` creates: `admin`.

   **Results**

   - You can access the Ceph Dashboard.
   - The numbers and graphs that the Dashboard displays reflect the same cluster status that the CLI command, `ceph -s`, returns.

For more information about the Red Hat Ceph Storage Dashboard, see the *Red Hat Ceph Storage Administration Guide*. 

---

**CHAPTER 9. ADDING THE RED HAT CEPH STORAGE DASHBOARD TO AN OVERCLOUD DEPLOYMENT**
CHAPTER 10. POST-DEPLOYMENT OPERATIONS TO MANAGE THE CEPH STORAGE CLUSTER

After you deploy your Red Hat OpenStack Platform (RHOSP) environment with containerized Red Hat Ceph Storage, there are some operations you can use to manage the Ceph Storage cluster.

10.1. ACCESSING THE OVERCLOUD

Director generates a script to configure and help authenticate interactions with your overcloud from the undercloud. Director saves this file, `overcloudrc`, in the home directory of the stack user.

Procedure

1. Run the following command to source the file:
   ```
   $ source ~/overcloudrc
   ```
   This loads the necessary environment variables to interact with your overcloud from the undercloud CLI.

2. To return to interacting with the undercloud, run the following command:
   ```
   $ source ~/stackrc
   ```

10.2. MONITORING CEPH STORAGE NODES

After you create the overcloud, check the status of the Ceph Storage cluster to confirm that it works correctly.

Procedure

1. Log in to a Controller node as the heat-admin user:
   ```
   $ nova list
   $ ssh heat-admin@192.168.0.25
   ```

2. Check the health of the cluster:
   ```
   $ sudo cephadm shell -- ceph health
   ```
   If the cluster has no issues, the command reports back `HEALTH_OK`. This means the cluster is safe to use.

3. Log in to an overcloud node that runs the Ceph monitor service and check the status of all OSDs in the cluster:
   ```
   $ sudo cephadm shell -- ceph osd tree
   ```

4. Check the status of the Ceph Monitor quorum:
   ```
   $ sudo cephadm shell -- ceph quorum_status
   ```
This shows the monitors participating in the quorum and which one is the leader.

5. Verify that all Ceph OSDs are running:

$ sudo cephadm shell -- ceph osd stat

For more information on monitoring Ceph Storage clusters, see Monitoring a Ceph Storage cluster in the Red Hat Ceph Storage Administration Guide.

10.3. MAPPING A BLOCK STORAGE (CINDER) TYPE TO YOUR NEW CEPH POOL

After you complete the configuration steps, make the performance tiers feature available to RHOSP tenants by using Block Storage (cinder) to create a type that is mapped to the fastpool tier that you created.

Procedure

1. Log in to the undercloud node as the stack user.

2. Source the overcloudrc file:

   $ source overcloudrc

3. Check the Block Storage volume existing types:

   $ cinder type-list

4. Create the new Block Storage volume fast_tier:

   $ cinder type-create fast_tier

5. Check that the Block Storage type is created:

   $ cinder type-list

6. When the fast_tier Block Storage type is available, set the fastpool as the Block Storage volume back end for the new tier that you created:

   $ cinder type-key fast_tier set volume_backend_name=tripleo_ceph_fastpool

7. Use the new tier to create new volumes:

   $ cinder create 1 --volume-type fast_tier --name fastdisk

NOTE

The Red Hat Ceph Storage documentation provides additional information and procedures for the ongoing maintenance and operation of the Ceph Storage cluster. See Product Documentation for Red Hat Ceph Storage 5 for this documentation.
CHAPTER 11. REBOOTING THE ENVIRONMENT

It might become necessary to reboot the environment. For example, when you need to modify physical servers or recover from a power outage. In these types of situations, it is important to make sure your Ceph Storage nodes boot correctly.

You must boot the nodes in the following order:

1. **Boot all Ceph Monitor nodes first** - This ensures the Ceph Monitor service is active in your high availability cluster. By default, the Ceph Monitor service is installed on the Controller node. If the Ceph Monitor is separate from the Controller in a custom role, make sure this custom Ceph Monitor role is active.

2. **Boot all Ceph Storage nodes** - This ensures the Ceph OSD cluster can connect to the active Ceph Monitor cluster on the Controller nodes.

### 11.1. REBOOTING A CEPH STORAGE (OSD) CLUSTER

Complete the following steps to reboot a cluster of Ceph Storage (OSD) nodes.

**Procedure**

1. Log in to a Ceph MON or Controller node and disable Ceph Storage cluster rebalancing temporarily:
   
   ```shell
   $ sudo cephadm shell -- ceph osd set noout
   $ sudo cephadm shell -- ceph osd set norebalance
   ```

   **NOTE**

   If you have a multistack or distributed compute node (DCN) architecture, you must specify the cluster name when you set the `noout` and `norebalance` flags. For example:
   ```shell
   sudo cephadm shell -c /etc/ceph/<cluster>.conf -k /etc/ceph/<cluster>.client.keyring
   ```

2. Select the first Ceph Storage node that you want to reboot and log in to the node.

3. Reboot the node:
   
   ```shell
   $ sudo reboot
   ```

4. Wait until the node boots.

5. Log in to the node and check the cluster status:
   
   ```shell
   $ sudo cephadm -- shell ceph status
   ```

   Check that the `pgmap` reports all `pgs` as normal (active+clean).

6. Log out of the node, reboot the next node, and check its status. Repeat this process until you have rebooted all Ceph Storage nodes.

7. When complete, log in to a Ceph MON or Controller node and re-enable cluster rebalancing:
$ sudo cephadm shell -- ceph osd unset noout
$ sudo cephadm shell -- ceph osd unset norebalance

NOTE
If you have a multistack or distributed compute node (DCN) architecture, you must specify the cluster name when you unset the noout and norebalance flags. For example: `sudo cephadm shell -c /etc/ceph/<cluster>.conf -k /etc/ceph/<cluster>.client.keyring`

8. Perform a final status check to verify that the cluster reports HEALTH_OK:

$ sudo cephadm shell ceph status

11.2. REBOOTING CEPH STORAGE OSDS TO ENABLE CONNECTIVITY TO THE CEPH MONITOR SERVICE

If a situation occurs where all overcloud nodes boot at the same time, the Ceph OSD services might not start correctly on the Ceph Storage nodes. In this situation, reboot the Ceph Storage OSDs so they can connect to the Ceph Monitor service.

Procedure

- Verify a HEALTH_OK status of the Ceph Storage node cluster:

  $ sudo ceph status
CHAPTER 12. SCALING THE CEPH STORAGE CLUSTER

You can scale the size of your Ceph Storage cluster by adding or removing storage nodes.

12.1. SCALING UP THE CEPH STORAGE CLUSTER

As capacity and performance requirements change, you can scale up your Ceph Storage cluster to meet increased demands. Before doing so, ensure that you have enough nodes for the updated deployment. Then you can register and tag the new nodes in your Red Hat OpenStack Platform (RHOSP) environment.

To register new Ceph Storage nodes with director, complete this procedure.

Procedure

1. Log in to the undercloud node as the stack user.

2. Modify the ~/overcloud_baremetal_deploy.yaml to add the CephStorage nodes to the deployment.
   The following example file represents an original deployment with three CephStorage nodes.

   ```yaml
   - name: CephStorage
     count: 3
     instances:
       - hostname: ceph-0
         name: ceph-0
       - hostname: ceph-1
         name: ceph-2
       - hostname: ceph-2
         name: ceph-2
   ```
   
The following example modifies this file to add three additional nodes.

   ```yaml
   - name: CephStorage
     count: 6
     instances:
       - hostname: ceph-0
         name: ceph-0
       - hostname: ceph-1
         name: ceph-2
       - hostname: ceph-2
         name: ceph-2
       - hostname: ceph-3
         name: ceph-3
       - hostname: ceph-4
         name: ceph-4
       - hostname: ceph-5
         name: ceph-5
   ```

3. Use the openstack overcloud node provision command with the updated ~/overcloud_baremetal_deploy.yaml file.

   ```bash
   openstack overcloud node provision --stack overcloud
   ```
NOTE
This command will provision the configured nodes and output an updated copy of ~/overcloud-baremetal-deployed.yaml. The new version updates the CephStorageCount. The DeployedServerPortMap and HostnameMap also contain the new storage nodes.

4. Use the `openstack overcloud deploy` command with the updated ~/overcloud-baremetal-deployed.yaml file.

    openstack overcloud deploy --templates \
    -e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm.yaml \
    -e deployed_ceph.yaml \
    -e overcloud_baremetal_deploy.yaml

Result
The following actions occur when the `openstack overcloud deploy` command runs:

- The storage networks and firewall rules are configured on the new CephStorage nodes.
- The ceph-admin user is created on the new CephStorage nodes.
- The ceph-admin user public SSH key is distributed to the new CephStorage nodes so that cephadm can use SSH to add extra nodes.
- If a new CephMon or CephMgr node is added, the ceph-admin private SSH key is also distributed to that node.
- An updated Ceph specification is generated and installed on the bootstrap node. This updated specification will typically be available in /home/ceph-admin/specs/ceph_spec.yaml on the bootstrap node.
- The cephadm bootstrap process is skipped because cephadm ls indicates the Ceph containers are already running.
- The updated Ceph specification is applied and cephadm schedules the new nodes to join the cluster.

12.2. SCALING DOWN AND REPLACING CEPH STORAGE NODES

In some cases, you might need to scale down your Ceph Storage cluster or replace a Ceph Storage node. In either situation, you must disable and rebalance the Ceph Storage nodes that you want to remove from the overcloud to prevent data loss.
PROCEDURE

Do not proceed with this procedure if the Ceph Storage cluster does not have the capacity to lose OSDs.

1. Log in to the overcloud Controller node as the **heat-admin** user.

2. Use the **cephadm shell** command to start a Ceph shell.

3. Use the **ceph osd tree** command to identify OSDs to be removed by server.
   In the following example we want to identify the OSDs of **ceph-2** host.

   ```
   [ceph: root@oc0-controller-0 /]# ceph osd tree
   ID  CLASS  WEIGHT   TYPE NAME            STATUS  REWEIGHT  PRI-AFF
   -1         0.58557  root default
   -7         0.19519  host ceph-2
   5    hdd  0.04880       osd.5           up      1.00000  1.00000
   7    hdd  0.04880       osd.7           up      1.00000  1.00000
   9    hdd  0.04880       osd.9           up      1.00000  1.00000
   11   hdd  0.04880       osd.11          up      1.00000  1.00000
   ```

4. Use the command **ceph orch osd rm --zap <osd_list>** to remove the OSDs.

   ```
   [ceph: root@oc0-controller-0 /]# ceph orch osd rm --zap 5 7 9 11
   Scheduled OSD(s) for removal
   [ceph: root@oc0-controller-0 /]# ceph orch osd rm status
   OSD_ID HOST   STATE    PG_COUNT REPLACE  FORCE  DRAIN_STARTED_AT
   7      ceph-2 draining 27       False    False  2021-04-23 21:35:51.215361
   9      ceph-2 draining 8        False    False  2021-04-23 21:35:49.111500
   11     ceph-2 draining 14       False    False  2021-04-23 21:35:50.243762
   ```

5. Use the command **ceph orch osd status** to check the status of OSD removal.

   ```
   [ceph: root@oc0-controller-0 /]# ceph orch osd rm status
   OSD_ID HOST   STATE    PG_COUNT REPLACE  FORCE  DRAIN_STARTED_AT
   7      ceph-2 draining 34       False    False  2021-04-23 21:35:51.215361
   11     ceph-2 draining 14       False    False  2021-04-23 21:35:50.243762
   ```

6. **WARNING**

   Do not proceed with the next step until this command returns no results.

   ```
   [ceph: root@oc0-controller-0 /]# ceph orch host rm ceph-2
   ```
7. End the Ceph shell session.

8. Log out of the **heat-admin** account.

9. Log in to the undercloud node as the **stack** user.

10. Modify the `~/overcloud_baremetal_deploy.yaml` in the following ways:

    - Decrease the `count` attribute in the roles to be scaled down.

    - Add an `instances` entry for each node being unprovisioned. Each entry must contain the following:
      - The **name** of the baremetal node.
      - The **hostname** assigned to that node.
      - A `provisioned: false` value.

    The following example would remove the node **overcloud-compute-1**.

      ```yaml
      - name: Compute count: 1 instances:
        - hostname: overcloud-compute-0
          name: node10
          # Removed from deployment due to disk failure
          provisioned: false
        - hostname: overcloud-compute-1
          name: node11
      ```

11. Use the `openstack overcloud node delete` command to remove the node.

    ```bash
    openstack overcloud node delete
    --stack overcloud
    --baremetal-deployment
    ~/overcloud_baremetal_deploy.yaml
    ```

    **NOTE**
    
    A list of nodes to delete will be provided with a confirmation prompt before the nodes are deleted.

    **NOTE**
    
    If scaling down the cluster is temporary and the nodes removed will be restored later, the scaling up action can increment the `count` and set `provisioned: true` on nodes that were previously set `provisioned: false`. If the node will never be reused, it can be set `provisioned: false` indefinitely and the scaling up action can specify a new `instances` entry.

    + The following file sample provides some examples of each instance.

      ```yaml
      - name: Compute count: 2 instances:
        - hostname: overcloud-compute-0
          name: node10
          # Removed from deployment due to disk failure
          provisioned: false
        - hostname: overcloud-compute-1
          name: node11
        - hostname: overcloud-compute-2
          name: node12
      ```
CHAPTER 13. REPLACING A FAILED DISK

If a disk in your Ceph Storage cluster fails, you can replace it.

13.1. REPLACING A DISK

See Adding OSDs in the Red Hat Ceph Storage Installation Guide for information on replacing a failed disk.
This appendix contains additional information about how to create some of the files required to deploy an overcloud with containerized Red Hat Ceph Storage.

A.1. MULTIPLE BONDED INTERFACES

Multiple bonded interfaces are a networking configuration often used with Red Hat Ceph Storage. Configuration of multiple bonded interfaces is described in Configuring multiple bonded interfaces for Ceph nodes.

For additional information see, Director Installation and Usage and the Networking Guide.