Red Hat OpenStack Platform 13

Integrating an Overcloud with an Existing Red Hat Ceph Cluster

Configuring an Overcloud to Use Stand-Alone Red Hat Ceph Storage

Last Updated: 2018-07-11

Red Hat OpenStack Platform 13 Integrating an Overcloud with an Existing Red Hat Ceph Cluster

Configuring an Overcloud to Use Stand-Alone Red Hat Ceph Storage

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

This guide describes how to use Red Hat OpenStack Platform director to integrate an Overcloud with an existing, stand-alone Red Hat Ceph cluster.

Table of Contents

CHAPTER 1. INTRODUCTION

Red Hat OpenStack Platform director creates a cloud environment called the *overcloud*. The director provides the ability to configure extra features for an Overcloud. One of these extra features includes integration with Red Hat Ceph Storage. This includes both Ceph Storage clusters created with the director or existing Ceph Storage clusters.

1.1. DEFINING CEPH STORAGE

Red Hat Ceph Storage is a distributed data object store designed to provide excellent performance, reliability, and scalability. Distributed object stores are the future of storage, because they accommodate unstructured data, and because clients can use modern object interfaces and legacy interfaces simultaneously. At the heart of every Ceph deployment is the **Ceph Storage Cluster**, which consists of two types of daemons:

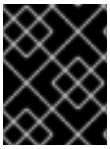
Ceph OSD (Object Storage Daemon)

Ceph OSDs store data on behalf of Ceph clients. Additionally, Ceph OSDs utilize 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 Storage, see the Red Hat Ceph Storage Architecture Guide.



IMPORTANT

This guide only integrates Ceph Block storage and the Ceph Object Gateway (RGW). It does not include Ceph File (CephFS) storage.

1.2. DEFINING THE SCENARIO

This guide provides instructions on integrating an existing Ceph Storage cluster with an overcloud. This means the director configures the overcloud to use the Ceph Storage cluster for storage needs. You manage and scale the cluster itself outside of the Overcloud configuration.

CHAPTER 2. PREPARING OVERCLOUD NODES

The scenario described in this chapter consists of six nodes in the Overcloud:

- Three Controller nodes with high availability.
- Three Compute nodes.

The director will integrate a separate Ceph Storage Cluster with its own nodes into the Overcloud. You manage this cluster independently from the Overcloud. For example, you scale the Ceph Storage cluster using the Ceph management tools and not through the OpenStack Platform director. Consult the Red Hat Ceph documentation for more information.

2.1. CONFIGURING THE EXISTING CEPH STORAGE CLUSTER

1. Create the following pools in your Ceph cluster relevant to your environment:
   - **`volumes`**: Storage for OpenStack Block Storage (cinder)
   - **`images`**: Storage for OpenStack Image Storage (glance)
   - **`vms`**: Storage for instances
   - **`backups`**: Storage for OpenStack Block Storage Backup (cinder-backup)
   - **`metrics`**: Storage for OpenStack Telemetry Metrics (gnocchi)
     Use the following commands as a guide:

     ```
     [root@ceph ~]# ceph osd pool create volumes PGNUM
     [root@ceph ~]# ceph osd pool create images PGNUM
     [root@ceph ~]# ceph osd pool create vms PGNUM
     [root@ceph ~]# ceph osd pool create backups PGNUM
     [root@ceph ~]# ceph osd pool create metrics PGNUM
     ```

     Replace *PGNUM* with the number of *placement groups*. We recommend approximately 100 per OSD. For example, the total number of OSDs multiplied by 100 divided by the number of replicas (**`osd pool default size`**). You can also use the Ceph Placement Groups (PGs) per Pool Calculator to determine a suitable value.
2. Create a **`client.openstack`** user in your Ceph cluster with the following capabilities:
   - **`cap_mon: allow r`**
   - **`cap_osd: allow class-read object_prefix rbd_children, allow rwx pool=volumes, allow rwx pool=vms, allow rwx pool=images, allow rwx pool=backups, allow rwx pool=metrics`**
     Use the following command as a guide:

     ```
     [root@ceph ~]# ceph auth add client.openstack mon 'allow r' osd
     'allow class-read object_prefix rbd_children, allow rwx
     pool=volumes, allow rwx pool=vms, allow rwx pool=images, allow
     rwx pool=backups, allow rwx pool=metrics'
     ```
3. Next, note the *Ceph client key* created for the **`client.openstack`** user:
   -

```
[root@ceph ~]# ceph auth list
...
client.openstack
 key: AQDLOh1VgEp6FRAAFzT7Zw+Y9V6JJExQAsRnRQ==
 caps: [mon] allow r
 caps: [osd] allow class-read object_prefix rbd_children, allow rwx
pool=volumes, allow rwx pool=vms, allow rwx pool=images, allow rwx
pool=backups, allow rwx pool=metrics
...
```

The **key** value here (**AQDLOh1VgEp6FRAAFzT7Zw+Y9V6JJExQAsRnRQ==**) is your Ceph client key.

4. Finally, note the *file system ID* of your Ceph Storage cluster. This value is specified with the **fsid** setting in the configuration file of your cluster (under the **[global]** section):

```
[global]
fsid = 4b5c8c0a-ff60-454b-a1b4-9747aa737d19
...
```

NOTE

For more information about the Ceph Storage cluster configuration file, see Configuration Reference (from the Red Hat Ceph Storage Configuration Guide).

The Ceph client key and file system ID will both be used later in Chapter 3, *Integrating with the Existing Ceph Cluster*.

2.2. INITIALIZING THE STACK USER

Log into the director host as the **stack** user and run the following command to initialize your director configuration:

```
$ source ~/stackrc
```

This sets up environment variables containing authentication details to access the director's CLI tools.

2.3. REGISTERING NODES

A node definition template (**instackenv.json**) is a JSON format file and contains the hardware and power management details for registering nodes. For example:

```
{
    "nodes":[
        {
            "mac":[
                "bb:bb:bb:bb:bb:bb"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"pxe_ipmitool",
```

```
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.205"
        },
        {
            "mac":[
                "cc:cc:cc:cc:cc:cc"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"pxe_ipmitool",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.206"
        },
        {
            "mac":[
                "dd:dd:dd:dd:dd:dd"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"pxe_ipmitool",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.207"
        },
        {
            "mac":[
                "ee:ee:ee:ee:ee:ee"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"pxe_ipmitool",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.208"
        }
        {
            "mac":[
                "ff:ff:ff:ff:ff:ff"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"pxe_ipmitool",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.209"
        }
```

```
        {
            "mac":[
                "gg:gg:gg:gg:gg:gg"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"pxe_ipmitool",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.210"
        }
    ]
}
```

After creating the template, save the file to the stack user's home directory (**/home/stack/instackenv.json**). Initialize the stack user, then import **instackenv.json** into the director:

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

This imports the template and registers each node from the template into the director.

Assign the kernel and ramdisk images to each node:

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

The nodes are now registered and configured in the director.

2.4. MANUALLY TAGGING THE NODES

After registering each node, you will need to inspect the hardware and tag the node into a specific profile. Profile tags match your nodes to flavors, and in turn the flavors are assigned to a deployment role.

To inspect and tag new nodes, follow these steps:

1. Trigger hardware introspection to retrieve the hardware attributes of each node:

   ```
   $ openstack overcloud node introspect --all-manageable --provide
   ```

   - The **--all-manageable** option introspects only nodes in a managed state. In this example, it is all of them.
   - The **--provide** option resets all nodes to an **active** state after introspection.

   IMPORTANT

   Make sure this process runs to completion. This process usually takes 15 minutes for bare metal nodes.

2. Retrieve a list of your nodes to identify their UUIDs:

```
$ openstack baremetal node list
```

3. Add a **profile** option to the **properties/capabilities** parameter for each node to manually tag a node to a specific profile.
   For example, to tag three nodes to use the **control** profile and another three nodes to use the **compute** profile, run:

```
$ ironic node-update 1a4e30da-b6dc-499d-ba87-0bd8a3819bc0 add
properties/capabilities='profile:control,boot_option:local'
$ ironic node-update 6faba1a9-e2d8-4b7c-95a2-c7fbdc12129a add
properties/capabilities='profile:control,boot_option:local'
$ ironic node-update 5e3b2f50-fcd9-4404-b0a2-59d79924b38e add
properties/capabilities='profile:control,boot_option:local'
$ ironic node-update 484587b2-b3b3-40d5-925b-a26a2fa3036f add
properties/capabilities='profile:compute,boot_option:local'
$ ironic node-update d010460b-38f2-4800-9cc4-d69f0d067efe add
properties/capabilities='profile:compute,boot_option:local'
$ ironic node-update d930e613-3e14-44b9-8240-4f3559801ea6 add
properties/capabilities='profile:compute,boot_option:local'
```

The addition of the **profile** option tags the nodes into each respective profiles.

NOTE

As an alternative to manual tagging, use the Automated Health Check (AHC) Tools to automatically tag larger numbers of nodes based on benchmarking data.

CHAPTER 3. INTEGRATING WITH THE EXISTING CEPH CLUSTER

The Heat template collection provided by the director already contains the necessary templates and environment files to deploy an overcloud.

This environment file is invoked during deployment (Section 3.3, “Deploying the Overcloud”) to integrate an existing Ceph cluster to the overcloud being deployed.

- **`/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible-external.yaml`**

The director uses **`ceph-ansible`** to integrate with an existing Ceph cluster, but **`ceph-ansible`** is not installed by default on the undercloud. Run the following command to install the ceph-ansible package on the undercloud:

```
sudo yum install -y ceph-ansible
```

To configure the integration, you must supply the details of your Ceph cluster to the director. To do this, use a *custom environment file*, which will allow you to override the default settings used by **`/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible-external.yaml`**:

1. Create the following custom environment file: **`/home/stack/templates/ceph-config.yaml`**
2. Add a **`parameter_defaults:`** header to this file:

   ```
   parameter_defaults:
   ```

3. Under this header, set all the parameters you want to override in **`/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible-external.yaml`**. At a minimum, you need to set:

   - **`CephClientKey`**: the Ceph client key of your Ceph Storage cluster. This is the value of **`key`** you retrieved earlier in Section 2.1, “Configuring the Existing Ceph Storage Cluster” (for example, **`AQDLOh1VgEp6FRAAFzT7Zw+Y9V6JJExQAsRnRQ==`**).
   - **`CephClusterFSID`**: the file system ID of your Ceph Storage cluster. This is the value of **`fsid`** in your Ceph Storage cluster configuration file, which you retrieved earlier in Section 2.1, “Configuring the Existing Ceph Storage Cluster” (for example, **`4b5c8c0a-ff60-454b-a1b4-9747aa737d19`**).
   - **`CephExternalMonHost`**: a comma-delimited list of the IPs of all MON hosts in your Ceph Storage cluster (for example, **`172.16.1.7, 172.16.1.8`**).

For example:

```
parameter_defaults:
  CephClientKey: AQDLOh1VgEp6FRAAFzT7Zw+Y9V6JJExQAsRnRQ==
  CephClusterFSID: 4b5c8c0a-ff60-454b-a1b4-9747aa737d19
  CephExternalMonHost: 172.16.1.7, 172.16.1.8
```

If necessary, also set the name of the OpenStack pools and the client user using the following parameters and values:

- **CephClientUserName: openstack**
- **NovaRbdPoolName: vms**
- **CinderRbdPoolName: volumes**
- **GlanceRbdPoolName: images**
- **CinderBackupRbdPoolName: backups**
- **GnocchiRbdPoolName: metrics**

You can also add overcloud parameters to your custom environment file. For example, to set **vxlan** as the **neutron** network type, add the following to **parameter_defaults**:

```
NeutronNetworkType: vxlan
```

3.1. BACKWARDS COMPATIBILITY WITH OLDER VERSIONS OF RED HAT CEPH STORAGE

If you are integrating Red Hat OpenStack Platform with an external Ceph Storage Cluster from an earlier version (that is, Red Hat Ceph Storage 1.3), you need to enable backwards compatibility. To do so, add the following line to the **parameter_defaults** of your custom environment file (in this case, **/home/stack/templates/ceph-config** from Chapter 3, *Integrating with the Existing Ceph Cluster*):

```
parameter_defaults:
  RbdDefaultFeatures: 1
```

3.2. 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** from Chapter 3, *Integrating with the Existing Ceph Cluster*):

For this purpose, use the following parameters:

Table 3.1. Roles and Flavors for Overcloud Nodes

Heat Template Parameter	Description
ControllerCount	The number of Controller nodes to scale out
OvercloudControlFlavor	The flavor to use for Controller nodes (**control**)
ComputeCount	The number of Compute nodes to scale out
OvercloudComputeFlavor	The flavor to use for Compute nodes (**compute**)

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

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

NOTE

See Creating the Overcloud with the CLI Tools from the Director Installation and Usage guide for a more complete list of Heat template parameters.

3.3. DEPLOYING THE OVERCLOUD

The creation of the Overcloud requires additional arguments for the **openstack overcloud deploy** command. For example:

```
$ openstack overcloud deploy --templates \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-
ansible/ceph-ansible-external.yaml \
  -e /home/stack/templates/ceph-config.yaml \
  -e --ntp-server pool.ntp.org \
```

The above command uses the following options:

- **--templates** - Creates the Overcloud from the default Heat template collection (namely, **/usr/share/openstack-tripleo-heat-templates/**).
- **-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible-external.yaml** - Sets the director to integrate an existing Ceph cluster to the overcloud.
- **-e /home/stack/templates/ceph-config.yaml** - Adds a custom environment file to override the defaults set by **-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible-external.yaml**. In this case, it is the custom environment file you created in Chapter 3, *Integrating with the Existing Ceph Cluster*.
- **--ntp-server pool.ntp.org** - Sets our NTP server.

TIP

You can also use an *answers file* to invoke all your templates and environment files. For example, you can use the following command to deploy an identical overcloud:

```
$ openstack overcloud deploy \
  --answers-file /home/stack/templates/answers.yaml \
  --ntp-server pool.ntp.org
```

In this case, the answers file **`/home/stack/templates/answers.yaml`** contains:

```
templates: /usr/share/openstack-tripleo-heat-templates/
environments:
  - /usr/share/openstack-tripleo-heat-templates/environments/ceph-
ansible/ceph-ansible-external.yaml \
  - /home/stack/templates/ceph-config.yaml \
```

See Including Environment Files in Overcloud Creation for more details.

For a full list of options, run:

```
$ openstack help overcloud deploy
```

For more information, see Creating the Overcloud with the CLI Tools in the Director Installation and Usage guide.

The Overcloud creation process begins and the director provisions your nodes. This process takes some time to complete. To view the status of the Overcloud creation, open a separate terminal as the **`stack`** user and run:

```
$ source ~/stackrc
$ openstack stack list --nested
```

This configures the Overcloud to use your external Ceph Storage cluster. Note that you manage this cluster independently from the Overcloud. For example, you scale the Ceph Storage cluster using the Ceph management tools and not through the OpenStack Platform director.

CHAPTER 4. ACCESSING THE OVERCLOUD

The director generates a script to configure and help authenticate interactions with your Overcloud from the director host. The director saves this file (**overcloudrc**) in your **stack** user's home directory. Run the following command to use this file:

```
$ source ~/overcloudrc
```

This loads the necessary environment variables to interact with your Overcloud from the director host's CLI. To return to interacting with the director's host, run the following command:

```
$ source ~/stackrc
```