HP Helion OpenStack 2.0: Ceph

Contents

HP Helion OpenStack® 2.0: Ceph.. **3**

HP Helion OpenStack® 2.0: Ceph

What is Ceph?

Ceph is a Software Defined Storage (SDS) system which comprised of a block storage, object storage, and file system. It is is designed to deliver different types of storage interfaces to the end user in the same storage platform. Therefore ceph is termed as unified storage platform.

HP Helion OpenStack : Ceph Overview

Based on OpenStack® Ceph, HP Helion OpenStack®2.0 Ceph Storage Solution provides an unified scaleable and stable storage solution for the management of Helion OpenStack Volume Storage (Cinder persistent Volumes) service. The solution also supports user backup to the Object Storage (Swift) API service writing to the same unified Ceph storage platform.

This release supports Ceph Firefly 0.80.7 version, which runs on the hlinux kernel 3.14.44-1.

This page describes the integration of Ceph Block Storage with HP Helion OpenStack 2.0.

This guide focuses on installation, configuration and integration between HP Helion OpenStack: 2.0 and Ceph Firefly 0.80.7 running on the hlinux kernel 3.14.44-1.

This guide assumes that you are familiar with the concepts of OpenStack and Ceph. The main purpose of this guide is describe the integration of Ceph Block Storage with HP Helion OpenStack 2.0, detail steps to install dependencies, configure HP Helion OpenStack and Ceph Firefly, and provide troubleshooting guidance.

Installation

This section describes the integration of Ceph Block Storage with HP Helion OpenStack 2.0, detailed procedure to install dependencies, and configure HP Helion OpenStack and Ceph Firefly.

1. Login to the Deployer node.
2. Copy `helion/examples/` in the Deployed node.

   ```
   cp -r ~/helion/examples/ ~/helion-input/my_cloud/definition
   ```
3. List the folder in `~/helion-input/my_cloud/definition`.

   The configuration files for editing are available at `~/helion/my_cloud/definition/data`.
4. Edit the configuration files, based on your environment, to implement Ceph servers.
5. Execute the following command to ensure that the additional disks are available on the servers marked for OSD as specified in the `disks_osd.yml`.

   ```
   vi disks_osd.yml
   ```

   The sample file of `disks_osd.yml` is as follows:

   ```
     disk-models:
     - name: DISK_SET_OSD
       # two disk node; remainder of disk 1 and all of disk 2 combined in
    single VG
       # VG is used to create three logical vols for /var, /var/log, and /
   var/crash
       device-groups:
         - name: ceph-osd-data-and-journal
           devices:
             - name: /dev/sdb
           consumer:
              name: ceph
   ```

```
            attrs:
              usage: data
              journal_disk: /dev/sdc
       - name: ceph-osd-data-and-shared-journal-set-1
         devices:
           - name: /dev/sdd
         consumer:
            name: ceph
            attrs:
              usage: data
              journal_disk: /dev/sdf
       - name: ceph-osd-data-and-shared-journal-set-2
         devices:
           - name: /dev/sde
         consumer:
            name: ceph
            attrs:
              usage: data
              journal_disk: /dev/sdf
```

The above sample file contains three OSD nodes and two journal disk.

The disk model has the following fields:

device-groups	There can be several device groups. This allows different sets of disks to be used for different purposes.
name	This is an arbitrary name for the device group. The name must be unique.
devices	This is a list of devices allocated to the device group. A `name` field containing `/dev/sdb`, `/dev/sdd`, and `/dev/sde` indicates that the device group is used by Ceph.
consumer	This specifies the service that uses the device group. A `name` field containing **ceph** indicates that the device group is used by Ceph.
attrs	
usage	There can be several use of devices for a particular service.In the above sample, `usage` field contians **data** which indicates that the device is used for data storage.
journal_disk	what is its usage?? You can share the journal disk between two nodes.

Important: Minimum 3 nodes are required as OSD nodes.

6. Commit your configuration to a *local repo*:

```
cd ~/helion/hos/ansible
git add -A
git commit -m "<commit message>"
```

Note: Enter your commit message <commit message>

7. Run the configuration procesor

```
cd ~/helion/hos/ansible
```

```
ansible-playbook -i hosts/localhost config-processor-run.yml
```

8. Use `ansible-playbook -i hosts/localhost ready-deployment.yml` file to create a deployment directory.
9. Run `verb_host` commands from the following directory:

   ```
    ~/scratch/ansible/next/hos/ansible
   ```

10. Modify `./helion/hlm/ansible/hlm-deploy.yml` to uncomment the line containing `ceph-deploy.yml`.
11. Run the following ansible playbook:

    ```
    ansible-playbook -i hosts/verb_hosts site.yml
    ```

Ceph Monitor service is deployed on the Controller Nodes and OSD's are deployed as separate nodes (Resource Nodes).

Run Ceph Client Packages

Execute the following command to install the ceph client packages on controller nodes and create users and ceph pools on the resource nodes:

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts ceph-client-prepare.yml
```

Configure Ceph as a Cinder backend

Perform the following procedure on the Deployer node to configure Ceph as a Cinder backend:

1. Edit `~/helion/hos/ansible/roles/_CND-CMN/templates/cinder.conf.j2` to add ceph configuration data as shown below:

   ```
   enabled_backends=ceph1
   ```

2. Copy the following configurations:

   ```
   [ceph1]
   rbd_max_clone_depth = 5
   rbd_flatten_volume_from_snapshot = False
   rbd_uuid = 457eb676-33da-42ec-9a8c-9293d545c337
   rbd_user = cinder
   rbd_pool = volumes
   rbd_ceph_conf = /etc/ceph/ceph.conf
   volume_driver = cinder.volume.drivers.rbd.RBDDriver
   volume_backend_name = ceph
   ```

   Note: The rbd_uuid is available in "/home/stack/helion/hos/ansible/roles/ceph-client-prepare/vars/ceph_user_model.yml"
3. Modify `cinder.conf.j2` at `~/helion/hos/ansible/roles/_CND-CMN/templates/cinder.conf.j2` with the following values:

   ```
   backup_driver = cinder.backup.drivers.ceph
   backup_ceph_conf = /etc/ceph/ceph.conf
   backup_ceph_user = cinder-backup
   backup_ceph_chunk_size = 134217728
   backup_ceph_pool = backups
   backup_ceph_stripe_unit = 0
   backup_ceph_stripe_count = 0
   restore_discard_excess_bytes = true
   ```

   Parameter description required

4. On all the Controller nodes copy the folling packages:

   ```
   cp /usr/lib/python2.7/dist-packages/rbd.py /opt/stack/venv/
   cinder-20150827T030317Z/lib/python2.7/site-packages/
   cp /usr/lib/python2.7/dist-packages/rados.py /opt/stack/venv/
   cinder-20150827T030317Z/lib/python2.7/site-packages/
   ```

   Note: Currently RBD volume attach to a nova instance is **NOT** working.

5. Copy `ceph.client.cinder.keyring` to the controller nodes:

   a. Login to controller node as a root user and execute the following command.

      ```
      # ceph auth get-or-create client.cinder | tee /etc/ceph/
      ceph.client.cinder.keyring
      ```

      OR

      You can execute the following command from the deployer node.

      ```
      # cp /etc/ceph/ceph.client.cinder.keyring to the /etc/ceph folder on the
       controller nodes.
      ```

6. Commit your confiugration to the local repo to configure cinder on the deployer node

   ```
   cd /home/stack/helion/hos/ansible
   git add -A
   git commit -m "<your commit message>"
   ```

   Note: Enter your commit message <commit message>

7. Use `ansible-playbook -i hosts/localhost ready-deployment.yml` file to create a deployment directory.
8. Run `verb_host` commands from the following directory:

   ```
    ~/scratch/ansible/next/hos/ansible
   ```

9. Run the following ansible playbook:

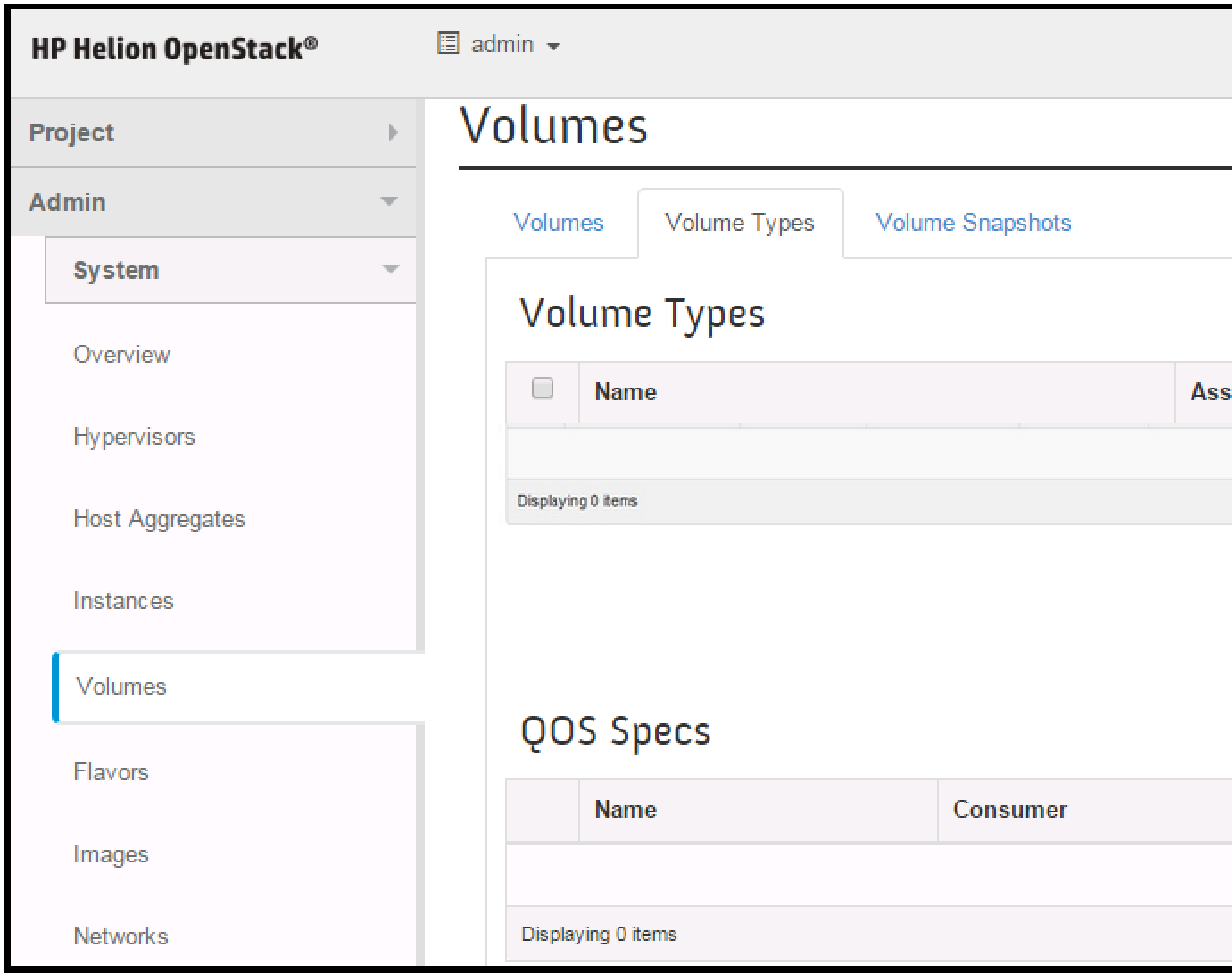
   ```
   ansible-playbook -i hosts/verb_hosts cinder-reconfigure.yml
   ```

   Once cinder is configured, launch the Horizon dashboard to create a cinder volume type.

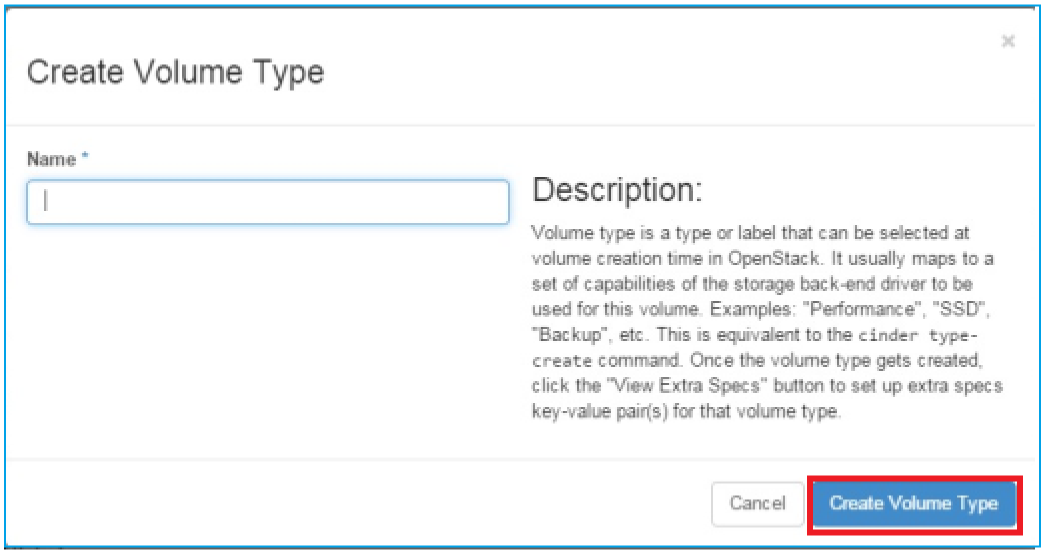
Creating Cinder Volume Type

To create a volume type using the Horizon dashboard, do the following:

1. Log into the Horizon dashboard. The Horizon dashboard displays with the options in the left panel.
2. From the left panel, click the **Admin** tab and then click the **Volumes** tab to display the Volumes page.



3. Click **Create Volume Type** to display a dialog box.

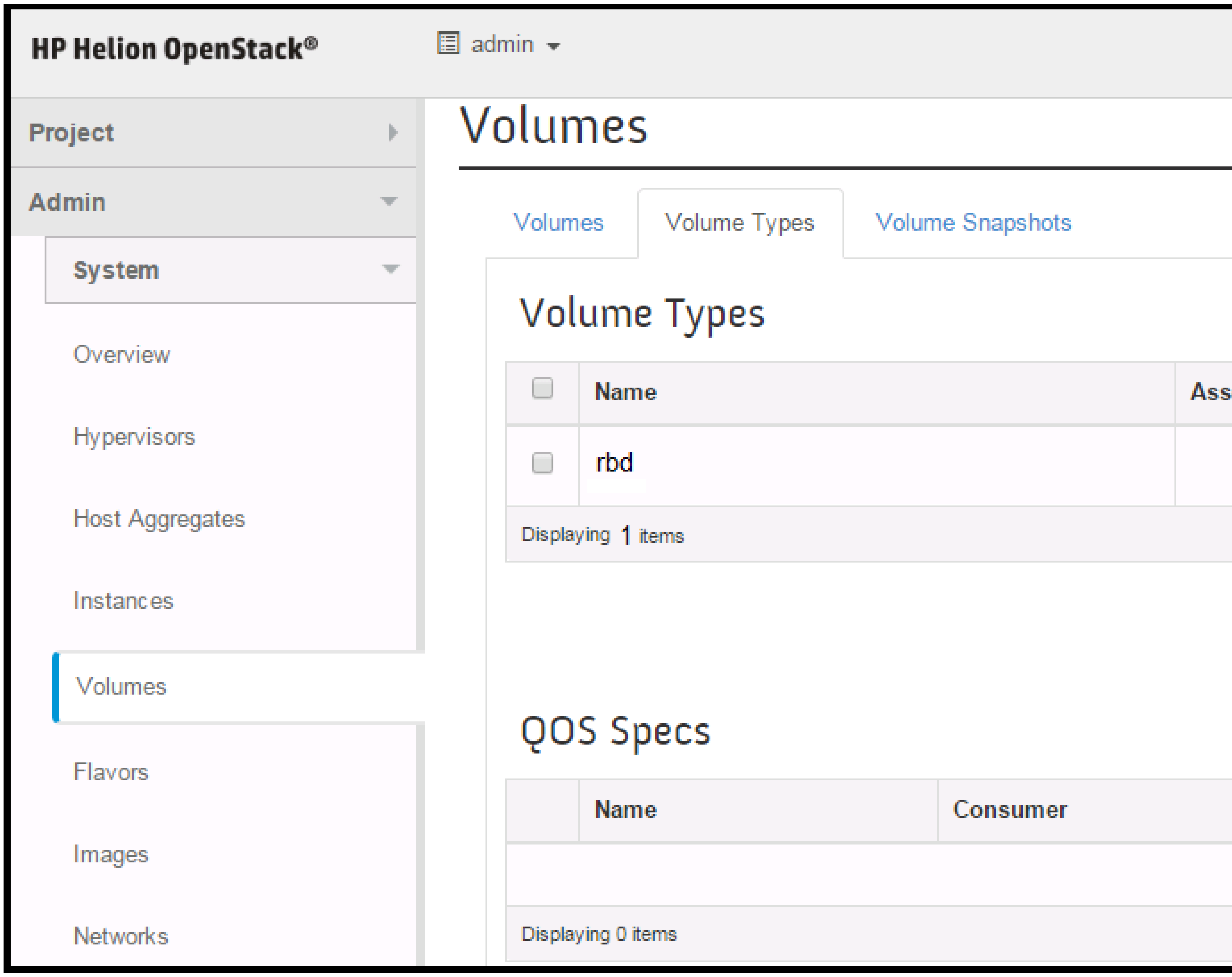

Create Volume Type

Name *

Description:

Volume type is a type or label that can be selected at volume creation time in OpenStack. It usually maps to a set of capabilities of the storage back-end driver to be used for this volume. Examples: "Performance", "SSD", "Backup", etc. This is equivalent to the `cinder type-create` command. Once the volume type gets created, click the "View Extra Specs" button to set up extra specs key-value pair(s) for that volume type.

4. Enter the name of the volume type.
5. Click **Create Volume Type**. The newly created volume displays in the Volumes page.



Volume Types

	Name	Ass
☐	rbd	

Displaying 1 items

QOS Specs

	Name	Consumer

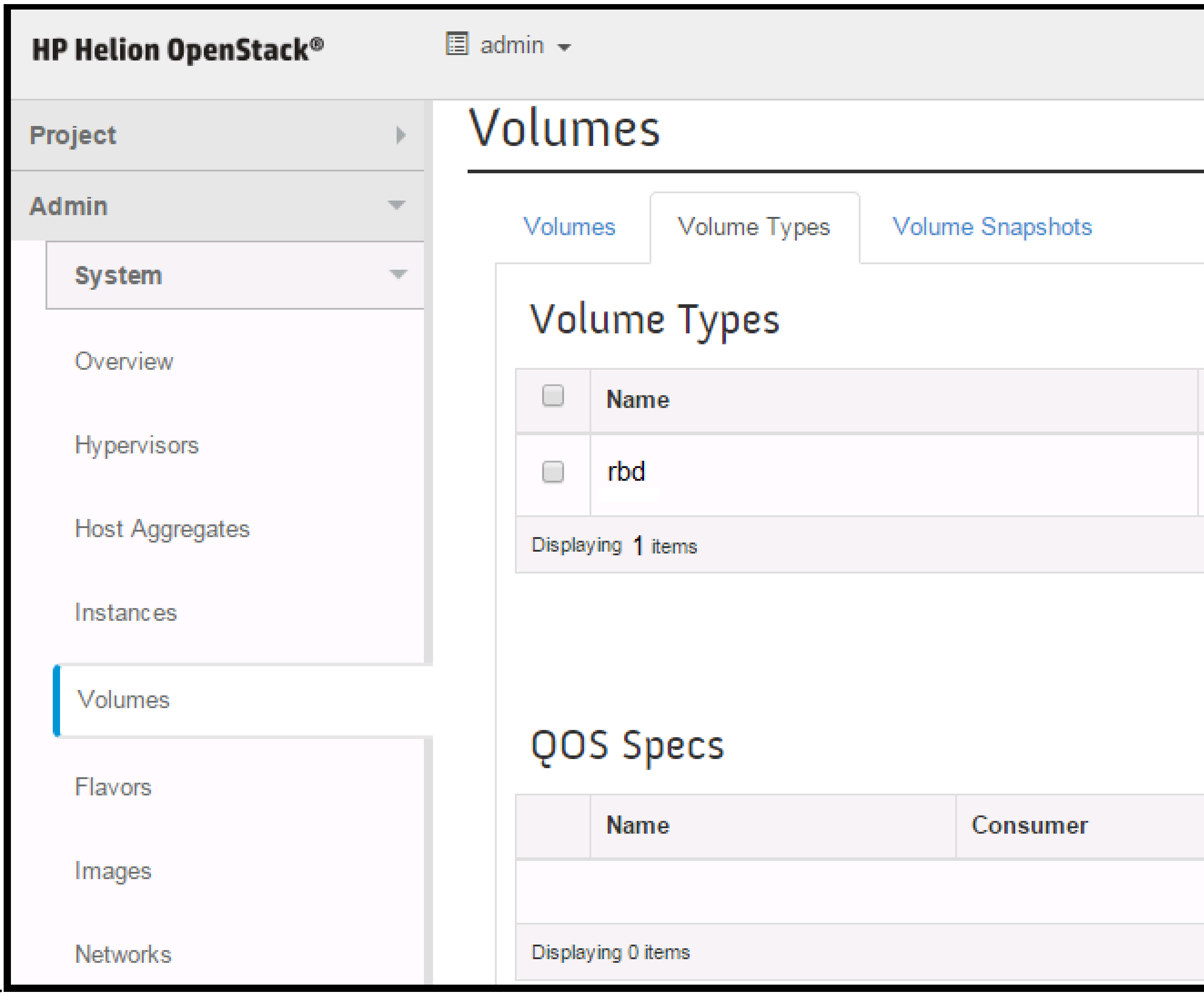
Displaying 0 items

Associate the volume type to a backend

To map a volume type to a backend, do the following:

1. Login to the Overcloud Horizon dashboard. The Overcloud dashboard displays with the options in the left panel.

2. From the left panel, click the **Admin** tab and then click the **Volumes** tab to display the Volumes page.



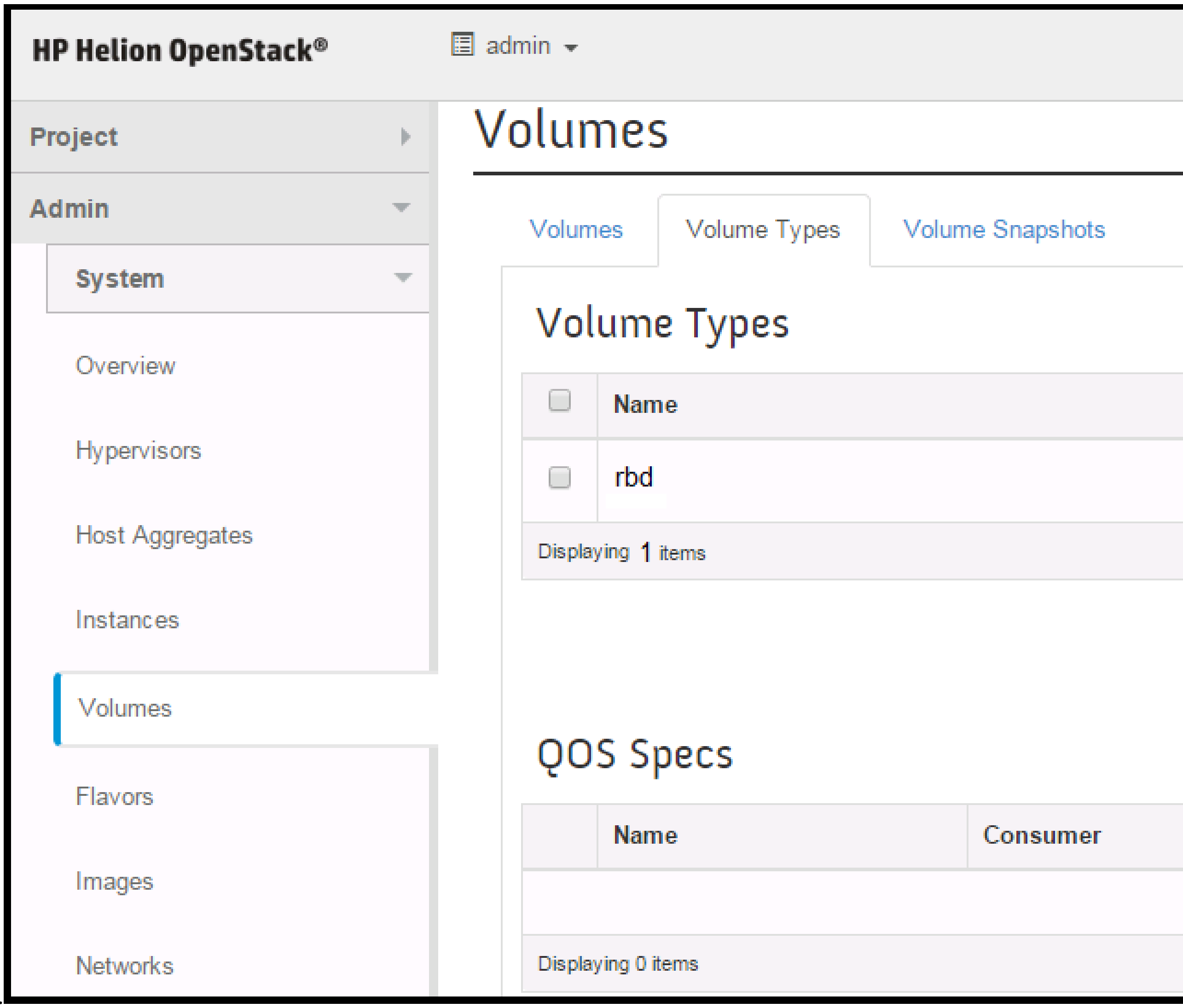
	Name
☐	rbd

Displaying 1 items

	Name	Consumer

Displaying 0 items

3. Click **View Extra Specs** displayed against the volume type which you want to associate to the backend.



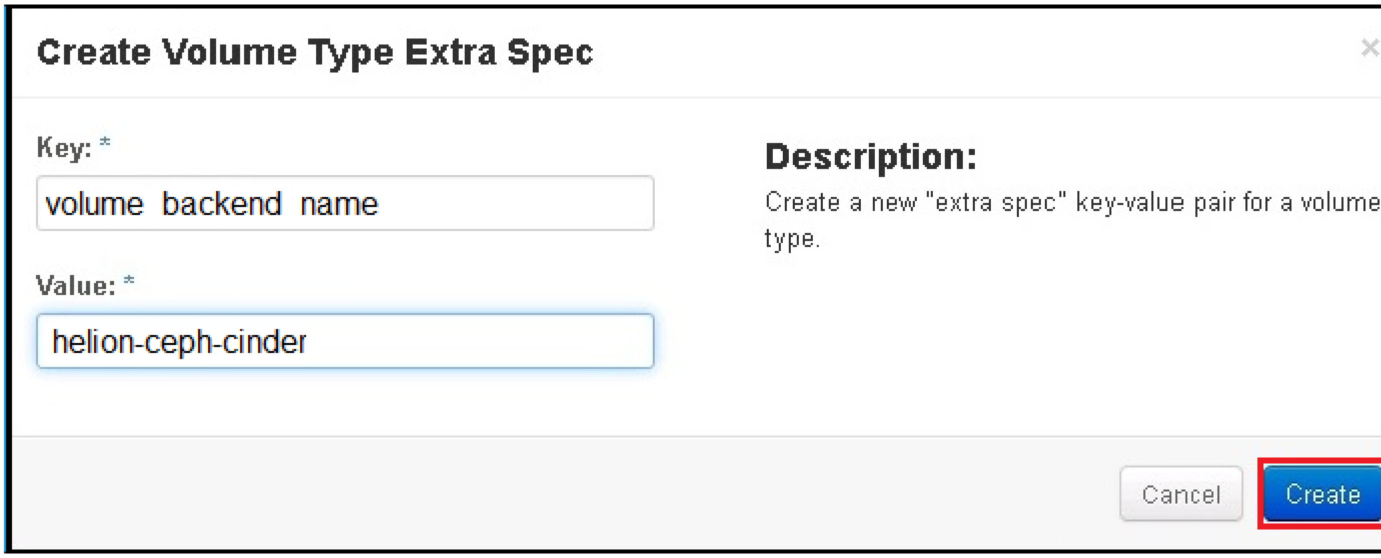
	Name
	rbd

Displaying 1 items

	Name	Consumer

Displaying 0 items

The **Create Volume Type Extra Specs** dialog box displays.



Key: * volume_backend_name

Value: * helion-ceph-cinder

Description:

Create a new "extra spec" key-value pair for a volume type.

4. In the **Key** box, enter *volume_backend_name*. This is the name of the key used to specify the storage backend when provisioning volumes of this volume type.
5. In the **Value** box, enter the name of the backend to which you want to associate the volume type. For example:*helion-ceph-cinder*.
6. Click **Create** to create the extra volume type specs.Note: Once the volume type is mapped to the backend, you can create volumes.

   Note: Once the volume type is mapped to the backend, you can create volumes.