Red Hat OpenStack Platform 16.1

Dell EMC PS Series Back End Guide

A Guide to Using Dell EMC PS Series Storage in a Red Hat OpenStack Platform Overcloud

Last Updated: 2020-10-18

Red Hat OpenStack Platform 16.1 Dell EMC PS Series Back End Guide

A Guide to Using Dell EMC PS Series Storage in a Red Hat OpenStack Platform Overcloud

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

This document describes how to deploy a single Dell EMC PS Series device as a back end to the Red Hat OpenStack Platform 15 Overcloud.

Table of Contents

CHAPTER 1. INTRODUCTION

This document describes how to configure OpenStack to use one or more Dell EMC PS Series back ends. It also includes instructions on addressing volume size discrepancies between Dell EMC PS Series devices and the OpenStack Block Storage service.

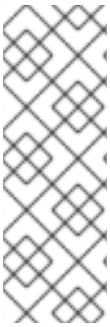
The following sections assume that:

- You intend to use only Dell EMC PS Series devices and drivers for Block Storage back ends
- The OpenStack Overcloud has already been deployed through Director, with a properly-functional Block Storage service
- The Dell storage device has already been deployed and configured as a storage repository
- A Dell EMC PS Series Group is already deployed and accessible through SSH
- You have the necessary credentials for connecting to the Group manager of the available Dell EMC PS Series Group (namely, CHAP and Group manager credentials)
- You have the username and password of an account with elevated privileges. You can use the same account that was created to deploy the Overcloud; in Creating a Director Installation User, we create and use the **stack** user for this purpose.

When RHEL OpenStack Platform is deployed through the Director, all major Overcloud settings (in particular, the Block Storage service back end) must be defined and orchestrated through the Director as well. This ensures that the settings will persist through any further Overcloud updates. For more information about deploying OpenStack through the Director, see Director Installation and Usage.

The purpose of this document is to explain how to orchestrate your desired Dell EqualLogic back end configuration to the Overcloud's Block Storage service. This document will not discuss the different deployment configurations possible with the back end. Rather, to learn more about the different available deployment configurations, see your device's product documentation.

Once you are familiar with the resulting back end configuration you want to deploy (and its corresponding settings), refer to this document for instructions on how to orchestrate it through the Director.



NOTE

At present, the Director only has the integrated components to deploy a **single** instance of a Dell EqualLogic back end. As such, this document only describes the deployment of a single back end.

Deploying multiple instances of a Dell EqualLogic back end requires a *custom back end configuration*. See the Custom Block Storage Back End Deployment Guide for instructions.

CHAPTER 2. PROCESS DESCRIPTION

RHEL OpenStack Platform includes all the drivers required for all Dell devices supported by the Block Storage service. In addition, the Director also has the puppet manifests, environment files, and Orchestration templates necessary for integrating the device as a back end to the Overcloud.

Configuring a **single** Dell device as a back end involves editing the default **environment file** and including it in the Overcloud deployment. This file is available locally on the Undercloud, and can be edited to suit your environment.

After editing this file, invoke it through the Director. Doing so ensures that it will persist through future Overcloud updates. The following sections describe this process in greater detail. In addition, the default environment file already contains enough information to call the necessary puppet manifests and Orchestration (Heat) templates that will configure the rest of the required Block Storage settings.

CHAPTER 3. DEFINE A SINGLE BACK END

IMPORTANT

This section describes the deployment of a single back end. Deploying multiple instances of a Dell EqualLogic back end requires a *custom back end configuration*. See the Custom Block Storage Back End Deployment Guide for instructions.

With a Director deployment, the easiest way to define a **single** Dell EMC PS Series back end is through the integrated environment file. This file is located in the following path of the Undercloud node:

/usr/share/openstack-tripleo-heat-templates/environments/cinder-dellps-config.yaml

Copy this file to a local path where you can edit and invoke it later. For example, to copy it to **~/templates/**:

```
$ cp /usr/share/openstack-tripleo-heat-templates/environments/cinder-dellps-config.yaml ~/templates/
```

Afterwards, open the copy (**~/templates/cinder-dellps-config.yaml**) and edit it as you see fit. The following snippet displays the default contents of this file:

```
# A Heat environment file which can be used to enable a
# a Cinder EMC PS Series backend, configured via puppet
resource_registry:
  OS::TripleO::Services::CinderBackendDellPs: ../puppet/services/cinder-backend-dellps.yaml # 1

parameter_defaults: # 2
  CinderEnableDellPsBackend: true # 3
  CinderDellPsBackendName: 'tripleo_dellps'
  CinderDellPsSanIp: ''
  CinderDellPsSanLogin: ''
  CinderDellPsSanPassword: ''
  CinderDellPsSanThinProvision: true
  CinderDellPsGroupname: 'group-0'
  CinderDellPsPool: 'default'
  CinderDellPsChapLogin: ''
  CinderDellPsChapPassword: ''
  CinderDellPsUseChap: false
```

1. The **OS::TripleO::Services::CinderBackendDellPs** parameter in the **resource_registry** section refers to a composable service template named **cinder-backend-dellps.yaml**. The director uses this template to load the necessary resources for configuring the back end. By default, the parameter specifies the path to **cinder-backend-dellps.yaml** relatively. As such, update this parameter with the absolute path to the file:

   ```
   resource_registry:
     OS::TripleO::ControllerExtraConfigPre: /usr/share/openstack-tripleo-heat-templates/puppet/services/cinder-backend-dellps.yaml
   ```

2. The **parameter_defaults** section contains your back end definition. Specifically, it contains the parameters that the Director should pass to the resources defined in **cinder-backend-dellps.yaml**.

3 The **CinderEnableDellPsBackend: true** line instructs the Director to use the puppet manifests necessary for the default configuration of a Dell EMC PS Series back end. This includes defining

To define your Dell EMC PS Series back end, edit the settings in the **parameter_defaults** section as you see fit. The following table explains each parameter, and also lists its corresponding **/etc/cinder/cinder.conf** setting.

Table 3.1. Dell EMC PS Series settings

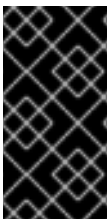
Parameter	/etc/cinder/cinder.conf setting	Description
CinderDellPsBackendName	volume_backend_name	An arbitrary name to identify the volume back end.
CinderDellPsSanIp	san_ip	The IP address used to reach the Dell EMC PS Series Group through SSH.
CinderDellPsSanLogin	san_login	The user name to login to the Group manager via SSH at the **CinderDellPsSanIp**. The default user name is **grpadmin**.
CinderDellPsSanPassword	san_password	The corresponding password of **CinderDellPsSanLogin**. The default password is **password**.
CinderDellPsSanThinProvision	san_thin_provision	Sets whether thin provisioning for SAN volumes is enabled (**true**), as is required for this setup.
CinderDellPsGroupname	eqlx_group_name	The group to be used for a pool where the Block Storage service will create volumes and snapshots. The default group is **group-0**.
CinderDellPsPool	eqlx_pool	The pool where the Block Storage service will create volumes and snapshots. This option cannot be used for multiple pools utilized by the Block Storage service on a single Dell EMC PS Series Group. The default pool is **default**.
CinderDellPsChapLogin	eqlx_chap_login	The CHAP login account for each volume in a pool. The default account name is **chapadmin**.
CinderDellPsChapPassword	eqlx_chap_password	The corresponding password of **CinderDellPsChapLogin**. The default password is randomly generated in hexadecimal, so you must set this password manually.
CinderDellPsUseChap	eqlx_use_chap	Sets whether CHAP authentication is disabled (**false** by default) or enabled (**true**).

CHAPTER 4. DEPLOY THE CONFIGURED BACK END

The Director installation uses a non-root user to execute commands, which includes orchestrating the deployment of the Block Storage back end. In Creating a Director Installation User , a user named **stack** is created for this purpose. This user is configured with elevated privileges.

To deploy the lone back end configured in Chapter 3, *Define a Single Back End* , first log in as the **stack** user to the Undercloud. Then, deploy the back end (defined in the edited **~/templates/cinder-dellps-config.yaml**) by running the following:

```
$ openstack overcloud deploy --templates -e ~/templates/cinder-dellps-config.yaml
```



IMPORTANT

If you passed any extra environment files when you created the overcloud, pass them again here using the **-e** option to avoid making undesired changes to the overcloud. For more information, see Modifying the overcloud environment in the *Director Installation and Usage* guide.

CHAPTER 5. TEST THE CONFIGURED BACK END

After deploying the back end, test whether you can successfully create volumes on it. Doing so will require loading the necessary environment variables first. These variables are defined in **/home/stack/overcloudrc** by default.

To load these variables, run the following command as the **stack** user:

```
$ source /home/stack/overcloudrc
```

NOTE

For more information, see Accessing the Overcloud.

You should now be logged in to the Controller node. From there, you can create a *volume type*, which can be used to specify the back end you want to use (in this case, the newly-defined back end in Chapter 3, *Define a Single Back End*). This is required in an OpenStack deployment where you have other back ends enabled (preferably, also through Director).

To create a volume type named **dellps**, run:

```
$ cinder type-create dellps
```

Next, map this volume type to the back end defined in Chapter 3, *Define a Single Back End* . Given the back end name **tripleo_dellps** (as defined through the **CinderDellPsBackendName** parameter, in Chapter 3, *Define a Single Back End* , run:

```
$ cinder type-key dellps set volume_backend_name=tripleo_dellps
```

You should now be able to create a 2GB volume on the newly defined back end by invoking its volume type. To do so, run:

```
$ cinder create --volume-type dellps 2
```

CHAPTER 6. ADDRESS VOLUME SIZE DISCREPANCIES WITH DELL EQUALLOGIC BACK ENDS

When reporting volume sizes, Dell EqualLogic (EQL) back ends also account for additional storage to be used for internal volume metadata. This size will be slightly larger than the volume size reported by the Block Storage services. However, the volume size reported by an EQL back end is the same one used by the Image service.

As a result, when creating an image-backed volume on an EQL back end, check the size of the image first. If the image was originally volume-backed, then EQL (and, by extension, the Image service) will be reporting a volume size slightly larger than what is reported by the Block Storage service.

If the image size reported by EQL is slightly larger, then you need to take the size discrepancy into consideration when creating volumes backed by this image.

6.1. EXAMPLE

To illustrate, when you create a 1GB volume:

 # cinder create --display-name vol1 1

```
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|     attachments     |                  []                  |
|  availability_zone  |                 nova                 |
|       bootable      |                false                 |
|      created_at     |      2014-12-19T03:57:47.730359      |
| display_description |                 None                 |
|     display_name    |                 vol1                 |
|      encrypted      |                False                 |
|          id         | 6bdace69-bd41-42fc-a63a-f834fb65a2e4 |
|       metadata      |                  {}                  |
|         size        |                  1                   |
|     snapshot_id     |                 None                 |
|     source_volid    |                 None                 |
|        status       |               creating               |
|     volume_type     |                 None                 |
+---------------------+--------------------------------------+
```

The Block Storage service will report a volume size of 1GB, but on the EQL array the size (VolReserve) will be slightly bigger:

eql> volume select volume-6bdace69-bd41-42fc-a63a-f834fb65a2e4

eql (volume-6bdace69-bd41-42fc-a63a-f834fb65a2e4)> show

```
_____________________________ Volume Information _____...
Name: volume-6bdace69-bd41-42fc-a63a-f834fb65a2e4
Size: 1GB
VolReserve: 1.01GB
...
```

When you create a new image from this volume, **cinder** will report a correct volume size of 1GB:

 # cinder upload-to-image --disk-format raw --container-format bare vol1 image_vol1

```
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|   container_format  |                bare                  |
|     disk_format     |                raw                   |
| display_description |                None                  |
|          id         | 6bdace69-bd41-42fc-a63a-f834fb65a2e4 |
|       image_id      | c65f7eae-e2c1-44ba-8af1-e33695897559 |
|      image_name     |              image_vol1              |
|         size        |                  1                   |
|        status       |             uploading                |
|      updated_at     |      2014-12-19T03:57:48.000000      |
|     volume_type     |                None                  |
+---------------------+--------------------------------------+
```

However, the Image service will report a slightly larger size:

 # glance image-list

```
...+------------+-------------+------------------+------------+--------+
...| Name       | Disk Format | Container Format | Size       | Status |
...+------------+-------------+------------------+------------+--------+
...| image_vol1 | raw         | bare             | 1085276160 | active |
...+------------+-------------+------------------+------------+--------+
```

The **glance** tool reports an image size of approximately 1.01GB. As a result, creating a new 1GB volume backed by this image will fail:

 # cinder create --display-name vol2 --image-id c65f7eae-e2c1-44ba-8af1-e33695897559 1

```
ERROR: Invalid input received: Size of specified image 2 is larger than volume size 1
```

6.2. WORKAROUND

As mentioned earlier, you need to consider the discrepancy between the volume sizes reported by the Image and the Block Storage services when specifying the size of image-backed volumes. This means that when specifying the size of the image-backed volume, use the **next whole number after the image size reported by glance**.

Using the previous example, **glance** reported an image size of 1.01GB. This means that when you create a volume, you need to specify a volume size of 2GB instead of 1GB:

 # cinder create --display-name vol2 --image-id c65f7eae-e2c1-44ba-8af1-e33695897559 2

```
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|     attachments     |                  []                  |
|  availability_zone  |                nova                  |
|       bootable      |               false                  |
|      created_at     |      2014-12-19T04:54:07.036260      |
| display_description |                None                  |
```

```
|     display_name    |                vol2                |
|      encrypted      |               False                |
|          id         | fcf49715-094d-4bba-9f05-8b7fa6deffce |
|       image_id      | c65f7eae-e2c1-44ba-8af1-e33695897559 |
|       metadata      |                {}                  |
|         size        |                2                   |
|     snapshot_id     |               None                 |
|     source_volid    |               None                 |
|        status       |             creating               |
|     volume_type     |               None                 |
+---------------------+------------------------------------+
```