# Red Hat OpenStack Platform 15

## Custom Block Storage Back End Deployment Guide

A Guide to Deploying a Custom Block Storage Back End in a Red Hat OpenStack Platform Overcloud

Last Updated: 2021-01-21

# Red Hat OpenStack Platform 15 Custom Block Storage Back End Deployment Guide

A Guide to Deploying a Custom Block Storage Back End in a Red Hat OpenStack Platform Overcloud

OpenStack Team
rhos-docs@redhat.com

## Legal Notice



## Abstract

This document describes how to deploy a custom, non-integrated back end for the Block Storage service (cinder) in a Red Hat OpenStack Platform 15 overcloud.

# Table of Contents

# CHAPTER 1. INTRODUCTION TO DEPLOYING A BLOCK STORAGE SERVICE CUSTOM BACK END

The Red Hat OpenStack Platform director installs and manages a complete, Enterprise-grade OpenStack deployment with minimal manual configuration. For more information about the director, see the Director Installation and Usage guide.

The Openstack environment that director creates is called the overcloud. The overcloud contains all the components that provide services to end users, including Block Storage. This document provides guidance about how to deploy a custom back end to the Block Storage service (cinder) on the overcloud. By default, the Block Storage service is installed on Controller nodes.

**Prerequisites**

- You have already deployed the overcloud with the director.
- The overcloud has a functioning Block Storage service.
- You are familiar with Block Storage concepts and configuration. For more information about Block Storage, see Block Storage and Volumes in the *Storage Guide*.

**WARNING**

This procedure has been tested successfully in limited use cases. Ensure that you test your planned deployment on a non-production environment first. If you have any questions, contact Red Hat support.

## 1.1. UNDERSTANDING CUSTOM BACK ENDS

A custom back end is a storage server, appliance, or configuration that is not yet fully integrated into the Red Hat OpenStack Platform director. Supported Block Storage back ends are already integrated and pre-configured with built-in director files. For example, Red Hat Ceph and single-back end configurations of Dell EMC PS Series, Dell Storage Center, and NetApp appliances.

Some integrated storage appliances support only a single-instance back end. For example, with the pre-configured director files for Dell Storage Center, you can only deploy a single back end. If you want to deploy multiple back end instances of this appliance, you need a custom configuration.

Although you can manually configure the Block Storage service by directly editing the **/etc/cinder/cinder.conf** file on the node where the Block Storage service is located, the director overwrites your configuration when you run the **openstack overcloud deploy** command. For more information, see Chapter 3, *Deploying the configured back ends* . Deploy the Block Storage back end with the director to ensure that your settings persist through overcloud deployments and updates.

If your back end configuration is fully integrated you can edit and invoke the packaged environment files. However, for custom back ends, you must write your own environment file. This document includes the annotated **/home/stack/templates/custom-env.yaml** file that you can edit for your deployment. This sample file is suitable for configuring the Block Storage service to use two NetApp back ends. For more information about environment files, see Including environment files in an overcloud deployment in the *Director Installation and Usage* guide.

## 1.2. REQUIREMENTS

The following additional prerequisite conditions must apply to your environment to configure custom Block Storage back ends:

- If you are using third-party back end appliances, you have configured them as storage repositories.
- You have deployed the overcloud with director with the instructions in Director Installation and Usage.
- You have the username and password of an account with elevated privileges. You can use the same **stack** user account that you created to deploy the overcloud.
- You have already planned the resulting configuration that you want for the Block Storage back end in **/etc/cinder/cinder.conf**.

## 1.3. UNDERSTANDING THE CONFIGURATION PROCESS

Configuring the Block Storage service to use custom back ends involves the following steps:

- Creating the environment file. For more information, see Chapter 2, *Creating the environment file*.
- Deploying the configured back ends. For more information, see Chapter 3, *Deploying the configured back ends*.
- Testing the configured back end. For more information, see Chapter 4, *Testing the configured back end*

# CHAPTER 2. CREATING THE ENVIRONMENT FILE

The environment file contains the settings for each back end that you want to define, and other relevant settings. For more information about environment files, see Environment Files in the *Advanced Overcloud Customization* guide.

The following sample environment file defines two NetApp back ends, **netapp1** and **netapp2**:

**/home/stack/templates/custom-env.yaml**

```
parameter_defaults: # 1
  CinderEnableIscsiBackend: false
  CinderEnableRbdBackend: false
  CinderEnableNfsBackend: false
  NovaEnableRbdBackend: false
  GlanceBackend: file # 2
  ControllerExtraConfig: # 3
    cinder::config::cinder_config:
      netapp1/volume_driver: # 4
        value: cinder.volume.drivers.netapp.common.NetAppDriver
      netapp1/netapp_storage_family:
        value: ontap_7mode
      netapp1/netapp_storage_protocol:
        value: iscsi
      netapp1/netapp_server_hostname:
        value: 10.35.64.11
      netapp1/netapp_server_port:
        value: 80
      netapp1/netapp_login:
        value: root
      netapp1/netapp_password:
        value: p@$$w0rd
      netapp1/volume_backend_name:
        value: netapp1
      netapp2/volume_driver: # 5
        value: cinder.volume.drivers.netapp.common.NetAppDriver # 6
      netapp2/netapp_storage_family:
        value: ontap_7mode
      netapp2/netapp_storage_protocol:
        value: iscsi
      netapp2/netapp_server_hostname:
        value: 10.35.64.11
      netapp2/netapp_server_port:
        value: 80
      netapp2/netapp_login:
        value: root
      netapp2/netapp_password:
        value: p@$$w0rd
      netapp2/volume_backend_name:
        value: netapp2
    cinder_user_enabled_backends: ['netapp1','netapp2'] # 7
```

1. The following parameters are set to **false**, which disables other back end types:

- **CinderEnableIscsiBackend**: other iSCSI back ends.
- **CinderEnableRbdBackend**: Red Hat Ceph.
- **CinderEnableNfsBackend**: NFS.
- **NovaEnableRbdBackend**: ephemeral Red Hat Ceph storage.

2 The **GlanceBackend** parameter sets what the Image service uses to store images. The following values are supported:

- **file**: store images on **/var/lib/glance/images** on each Controller node.
- **swift**: use the Object Storage service for image storage.
- **cinder**: use the Block Storage service for image storage.

3 **ControllerExtraConfig** defines custom settings that are applied to all Controller nodes. The **cinder::config::cinder_config** class means the settings must be applied to the Block Storage (cinder) service.

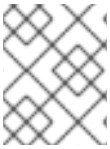
4 The **netapp1/volume_driver** and **netapp2/volume_driver** settings follow the *section/setting* syntax. With the Block Storage service, each back end is defined in its own section in **/etc/cinder/cinder.conf**. Each setting that uses the **netapp1** prefix is defined in a new **[netapp1]** back end section.

5 **netapp2** settings are defined in a separate **[netapp2]** section.

6 The **value** prefix configures the preceding setting.

7 The **cinder_user_enabled_backends** class sets and enables custom back ends. Use this class only for user-enabled back ends, specifically, those defined in the **cinder::config::cinder_config** class.

Do not use **cinder_user_enabled_backends** to list back ends that you can enable natively with director. These include Red Hat Ceph, NFS, and single back ends for supported NetApp or Dell appliances. For example, if you enable a Red Hat Ceph back end, do not list it in **cinder_user_enabled_backends**, enable it by setting **CinderEnableRbdBackend** to **true**.



NOTE

For more information about defining a Red Hat Ceph back end for OpenStack Block Storage, see the Deploying an Overcloud with Containerized Red Hat Ceph guide.

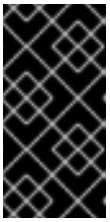
To see the resulting **/etc/cinder/cinder.conf** settings from **/home/stack/templates/custom-env.yaml**, see Appendix A, *Appendix*

# CHAPTER 3. DEPLOYING THE CONFIGURED BACK ENDS

To deploy the configured back ends, complete the following steps:

1. Log in as the **stack** user.
2. Run the following command to deploy the custom back end configuration:

```
$ openstack overcloud deploy --templates -e /home/stack/templates/custom-env.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.

Test the back end after director orchestration is complete.

# CHAPTER 4. TESTING THE CONFIGURED BACK END

After you deploy the back ends to the overcloud, test that you can successfully create volumes on them.

**Procedure**

1. Run the following command as the **stack** user to load the environment variables defined in **home/stack/overcloudrec**:

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

For more information, see Accessing the overcloud in the *Director Installation and Usage* guide.

1. Create a volume type for each back end. Log in to the Controller node of the overcloud as the **stack** user and run the following command:

```
$ cinder type-create backend1
$ cinder type-create backend2
```

These commands create the volume types **backend1** and **backend2**, one for each back end that is defined with the **cinder::config::cinder_config** class of the environment file that you created.

2. Map each volume type to the **volume_backend_name** of a back end that is enabled with the **cinder_user_enabled_backends** class of the environment file that you created The following commands map the volume type **backend1** to **netapp1** and **backend2** to **netapp2**:

```
$ cinder type-key backend1 set volume_backend_name=netapp1
$ cinder type-key backend2 set volume_backend_name=netapp2
```

3. Run the following command to test that it is possible to create a back end on **netapp1** by invoking the **backend1** volume type:

```
$ cinder create --volume-type backend1 --display_name netappvolume_1 1
```

4. Create a similar volume on the **netapp2** back end by invoking the **backend2** volume type:

```
$ cinder create --volume-type backend2 --display_name netappvolume_2 1
```

# APPENDIX A. APPENDIX

## Configuration from sample environment file

The environment file that you created in Chapter 2, *Creating the environment file* configures the Block Storage service to use two NetApp back ends. The following snippet displays the relevant settings:

```
enabled_backends = netapp1,netapp2

[netapp1]
volume_backend_name=netapp_1
volume_driver=cinder.volume.drivers.netapp.common.NetAppDriver
netapp_login=root
netapp_storage_protocol=iscsi
netapp_password=p@$$w0rd
netapp_storage_family=ontap_7mode
netapp_server_port=80
netapp_server_hostname=10.35.64.11

[netapp2]
volume_backend_name=netapp_2
volume_driver=cinder.volume.drivers.netapp.common.NetAppDriver
netapp_login=root
netapp_storage_protocol=iscsi
netapp_password=p@$$w0rd
netapp_storage_family=ontap_7mode
netapp_server_port=80
netapp_server_hostname=10.35.64.11
```