



Red Hat OpenStack Platform 12

Custom Block Storage Back End Deployment Guide

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

Red Hat OpenStack Platform 12 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

Copyright © 2019 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This document describes how to deploy a custom, non-integrated back end for the Block Storage service in a Red Hat OpenStack Platform 12 Overcloud.

Table of Contents

CHAPTER 1. INTRODUCTION	3
1.1. CUSTOM BACK ENDS	3
1.2. REQUIREMENTS	4
CHAPTER 2. PROCESS DESCRIPTION	5
CHAPTER 3. CREATE THE ENVIRONMENT FILE	6
CHAPTER 4. DEPLOY THE CONFIGURED BACK ENDS	8
CHAPTER 5. TEST THE CONFIGURED BACK END	9
APPENDIX A. APPENDIX	10
A.1. THE STACK USER	10
A.2. RESULTING CONFIGURATION FROM SAMPLE ENVIRONMENT FILE	10

CHAPTER 1. INTRODUCTION

The Red Hat OpenStack Platform Director is a toolset for installing and managing a complete OpenStack environment. It is based primarily on the upstream TripleO (*OpenStack-on-OpenStack*) project. The Director's primary objective is to fully orchestrate a functional, Enterprise-grade OpenStack deployment with minimal manual configuration. It helps address many of the issues inherent in manually configuring individual OpenStack components.

The end-result OpenStack deployment provided by the Director is called the *Overcloud*. The Overcloud houses all the components that provide services to end users, including Block Storage. This document provides guidance on how to deploy custom back ends to the Overcloud's Block Storage service.

This document presumes existing knowledge of concepts relating to manual Block Storage configuration. In a test deployment of OpenStack (for example, through Packstack), configuring this service involves editing its host node's `/etc/cinder/cinder.conf`. Most of the Block Storage settings in that file are documented in better detail elsewhere; this document describes how to apply those same settings to the Overcloud in order to attach a *custom back end*.



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. CUSTOM BACK ENDS

For the purposes of this document, a custom back end is defined as a storage server/appliance or configuration that has yet to be integrated fully into the Red Hat OpenStack Platform Director. Some supported Block Storage back ends are already integrated into the Director; this means that pre-configured Director files are already provided out-of-the-box. An integrated back end can be configured and deployed to the Overcloud through these files. Examples of integrated back ends include Red Hat Ceph and single-back end configurations of Dell EMC PS Series, Dell Storage Center, and NetApp appliances.

Further, some storage appliances already integrated into Director only support a single-instance back end. For example, the pre-configured Director files for Dell Storage Center only allow for the deployment of a single back end. Deploying multiple back end instances of this appliance requires a custom configuration, as demonstrated in this document.

While you can manually configure the Block Storage service by directly editing its node's `/etc/cinder/cinder.conf`, the Director will overwrite your configuration once you run **openstack overcloud deploy** (as you do later on in [Chapter 4, Deploy the Configured Back Ends](#)). As such, the recommended method for deploying a Block Storage back end is through the Director, as doing so will ensure that your settings persist through Overcloud deployments and updates.

If a back end configuration is already fully integrated, you can simply edit and invoke its packaged environment files. With custom back ends, however, you need to write your own [environment file](#). This document includes an annotated sample that you can edit for your own deployment, namely [/home/stack/templates/custom-env.yaml](#). This sample file is suitable for configuring the Block Storage service to use two NetApp back ends.

1.2. REQUIREMENTS

Other than prior knowledge on manually configuring Block Storage and the back end you want to deploy, this document also assumes that:

- If you are using third-party back end appliances, then they must already be properly configured as storage repositories.
- The Overcloud has already been deployed through Director, per instructions in [Director Installation and Usage](#).
- 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](#), the **stack** user is created for this purpose.
- You have already mapped out the resulting configuration you want for the Block Storage back end in `/etc/cinder/cinder.conf`. With this, all that remains is the orchestration of your planned configuration through the Director.

CHAPTER 2. PROCESS DESCRIPTION

The Block Storage service's settings are stored in `/etc/cinder/cinder.conf`; these settings include *back end definitions*. Most third-party back ends usable with (or even supported by) the Block Storage service provide setup instructions that involve editing `/etc/cinder/cinder.conf` settings. As mentioned in [Chapter 1, Introduction](#), doing so will configure the Block Storage service; however, those settings will get overwritten in future Overcloud updates.

Regardless, any documentation relating to manual configuration through `/etc/cinder/cinder.conf` is still useful for Overcloud deployments. The Director, after all, applies the same configuration to `/etc/cinder/cinder.conf`, albeit through `heat`. As such, planning the back end configuration requires that you:

- Thoroughly plan the Block Storage back end configuration you want, and
- Map out the resulting `/etc/cinder/cinder.conf` file for this configuration.

Once you map out the resulting `/etc/cinder/cinder.conf` file, create the environment file that will orchestrate the back end settings. [environment file](#) describes this step in greater detail, using the sample file `/home/stack/templates/custom-env.yaml`. Having the environment file handy will help ensure that the back end settings persist through future Overcloud updates.

CHAPTER 3. CREATE THE ENVIRONMENT FILE

The environment file contains the settings for each back end you want to define. It also contains other settings relevant to the deployment of a custom back end. 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, namely **netapp1** and **netapp2**:

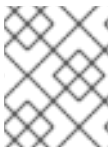
/home/stack/templates/custom-env.yaml

```
parameters: # 1
  CinderEnableIscsiBackend: false
  CinderEnableRbdBackend: false
  CinderEnableNfsBackend: false
  NovaEnableRbdBackend: false
  GlanceBackend: file # 2

parameter_defaults:
  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**, and thereby disable 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 should use 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 will be applied to all Controller nodes. The **cinder::config::cinder_config** class means the settings should be applied to the Block Storage (**cinder**) service. This, in turn, means that the back end settings will ultimately end in the **/etc/cinder/cinder.conf** file of each Controller node.
- 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 will be defined in a new **[netapp1]** back end section.
- 5 Likewise, **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. As the name implies, this class should only be used 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 you can enable natively through Director. These include Red Hat Ceph, NFS, and single back ends for supported NetApp or Dell appliances. For example, if you are also enabling a Red Hat Ceph back end, do not list it in **cinder_user_enabled_backends**; rather, enable it using **CinderEnableRbdBackend: true**.



NOTE

For more information on defining a Red Hat Ceph back end for OpenStack Block Storage, see [Deploying an Overcloud with Containerized Red Hat Ceph](#).

Chapter 4, *Deploy the Configured Back Ends* describes how to use the environment file **/home/stack/templates/custom-env.yaml** to orchestrate the custom back end's deployment. To see the resulting **/etc/cinder/cinder.conf** settings from **/home/stack/templates/custom-env.yaml**, see [Section A.2, "Resulting Configuration from Sample Environment File"](#).

CHAPTER 4. DEPLOY THE CONFIGURED BACK ENDS

Once you have created the [custom-env.yaml](#) file in `/home/stack/templates/`, log in as the **stack** user. Then, deploy the custom back end configuration by running:

```
$ 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](#) (from [Director Installation and Usage](#)).

Once the Director completes the orchestration, test the back end. See [Chapter 5, Test the Configured Back End](#) for instructions.

CHAPTER 5. TEST THE CONFIGURED BACK END

After deploying the back ends to the Overcloud, test whether you can successfully create volumes on them. 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 Basic Overcloud](#).

Next, 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:

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

These commands will create the volume types **backend1** and **backend2**, one for each back end defined through the **cinder::config::cinder_config** class of `xref:envfile`.

Finally, map each volume type to the **volume_backend_name** of a back end enabled through the **cinder_user_enabled_backends** class of `xref:envfile`. The following commands will 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
```

At this point, you should now be ready to test each back end. To do start, create a 1GB volume named **netapp_volume_1** on the **netapp1** back end by invoking the **backend1** volume type:

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

Likewise, 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

A.1. THE STACK USER

The [Creating a Director Installation User](#) section instructs readers to create a user named **stack**, which in turn will be used to deploy the Overcloud. You can use the **stack** account to run commands that require elevated privileges, such as deploying the back end ([Chapter 4, Deploy the Configured Back Ends](#)) or loading the necessary environment variables for accessing the Overcloud ([Chapter 5, Test the Configured Back End](#)).

Presumably, if you followed the instructions in [Director Installation and Usage](#), the **stack** user would already exist. As such, it would be convenient to use it as needed.

A.2. RESULTING CONFIGURATION FROM SAMPLE ENVIRONMENT FILE

The environment file in `xref:envfile` will configure 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
```