Red Hat OpenStack Platform 17.1

Deploying a custom Block Storage back end

Deploy a custom, non-integrated, Block Storage back end in a Red Hat OpenStack Platform overcloud

Red Hat OpenStack Platform 17.1 Deploying a custom Block Storage back end

end

Deploy a custom, non-integrated, Block Storage back end in a Red Hat OpenStack Platform overcloud

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

Deploy a back end for the Block Storage service (cinder) in a Red Hat OpenStack Platform (RHOSP) overcloud, which is not yet fully integrated into the RHOSP director.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message .

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Tell us how we can make it better.

Providing documentation feedback in Jira

Use the Create Issue form to provide feedback on the documentation. The Jira issue will be created in the Red Hat OpenStack Platform Jira project, where you can track the progress of your feedback.

1. Ensure that you are logged in to Jira. If you do not have a Jira account, create an account to submit feedback.

2. Click the following link to open a the **Create Issue** page: Create Issue

3. Complete the **Summary** and **Description** fields. In the **Description** field, include the documentation URL, chapter or section number, and a detailed description of the issue. Do not modify any other fields in the form.

4. Click **Create**.

CHAPTER 1. 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 Installing and managing Red Hat OpenStack Platform with director 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 *Configuring persistent 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.

CHAPTER 2. 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 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, see Configuration from sample environment file .

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 *Installing and managing Red Hat OpenStack Platform with director* guide.

2.1. 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 Installing and managing Red Hat OpenStack Platform with director.

- 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**.

2.2. 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 Creating the custom back end environment file.

- Deploying the configured back ends. For more information, Deploying the configured back ends.

- Testing the configured back end. For more information, Testing the configured back ends .

CHAPTER 3. CREATING THE CUSTOM BACK END ENVIRONMENT FILE

The environment file that you create to configure custom back ends contains the settings for each back end that you want to define. It also contains other settings that are relevant to the deployment of a custom back end. For more information about environment files, see Environment Files in the *Customizing your Red Hat OpenStack Platform deployment* 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 Deploying Red Hat Ceph Storage and Red Hat OpenStack Platform together with director.

To see the resulting **/etc/cinder/cinder.conf** settings from **/home/stack/templates/custom-env.yaml**, see Configuration from sample environment file .

CHAPTER 4. DEPLOYING THE CONFIGURED BACK ENDS

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

Procedure

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 *Installing and managing Red Hat OpenStack Platform with director* guide.
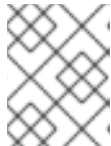
CHAPTER 5. TESTING THE CONFIGURED BACK ENDS

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
   ```

   > **NOTE**
   >
   > For more information, see Accessing the overcloud in the *Installing and managing Red Hat OpenStack Platform with director* guide.

2. 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.

3. 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
   ```

4. 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
   ```

5. 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
   ```

CHAPTER 6. CONFIGURATION FROM SAMPLE ENVIRONMENT FILE

The environment file that you created in Creating the custom back end 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
```