Red Hat OpenStack Platform 13

Block Storage Backup Guide

Understanding, using, and managing the Block Storage backup service in OpenStack

Last Updated: 2021-03-05

Red Hat OpenStack Platform 13 Block Storage Backup Guide

Understanding, using, and managing the Block Storage backup service in OpenStack

OpenStack Team
rhos-docs@redhat.com

Abstract

This document describes how to deploy the OpenStack Block Storage Backup Service. The instructions herein are specific to an overcloud deployment. The OpenStack director can configure Red Hat Ceph Storage, NFS, and Object Storage (swift) as back ends.

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 .

CHAPTER 1. OVERVIEW OF THE RED HAT OPENSTACK PLATFORM BLOCK STORAGE BACKUP SERVICE

Red Hat OpenStack Platform (RHOSP) provides the foundation to build a private or public Infrastructure-as-a-Service (IaaS) cloud on top of Red Hat Enterprise Linux. It is a scalable, fault-tolerant platform for the development of cloud-enabled workloads.

You can manage most features of the backup service by using either the RHOSP dashboard or the command-line client methods, however you must use the command line to execute some of the more advanced procedures.

NOTE

For the complete suite of documentation for Red Hat OpenStack Platform, see Red Hat OpenStack Platform Documentation.

The Block Storage service (cinder) provides a horizontally scalable backup service that you can use to back up cinder volumes using diverse storage back ends. You can use the Block Storage backup service to create full or incremental backups and to restore these backups. The service is volume-array independent.

The Red Hat OpenStack Platform director is a toolset for installing and managing a complete OpenStack environment. The Red Hat OpenStack director orchestrates 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 deployed by the director is called the overcloud. The overcloud houses all the components that provide services to end users, including Block Storage. The Block Storage backup service is an optional service deployed on Controller nodes.

This document provides guidance about how to deploy the Block Storage backup service to use a specific back end. This guide describes planning, installing, configuring, and using the Block Storage backup service.

1.1. WHAT IS A BACKUP?

A volume backup is a persistent copy of the contents of a volume. Volume backups are typically created as object stores. By default, they are managed through the OpenStack Object Storage service (swift). Optionally, you can configure Red Hat Ceph Storage and NFS as alternative back ends for backups.

When you create a volume backup, all of the backup metadata is stored in the Block Storage service database. The cinder-backup service uses the metadata to restore a volume from the backup. When you recover data from catastrophic database loss, you must restore the Block Storage service database before restoring any volumes from backups. This recovery scenario presumes that the Block Storage service database is restored with all of the original volume backup metadata intact.

If you want to configure volume backups for only a subset of data, you must export the backup metadata for the volume. Volume metadata backups enable you to re-import the metadata to the Block Storage database, through the REST API or the cinder client, and restore the volume backup as normal.

Volume backups are different from snapshots. Backups preserve the data contained in the volume and are used to prevent data los. Snapshots preserve the state of a volume at a specific point in time and are used to facilitate cloning. You cannot delete a volume if it has existing snapshots.

To minimize latency during cloning, snapshot back ends are typically co-located with volume back ends. In a typical enterprise deployment, a backup repository is located in a separate location from the volume back end, such as on a different node, physical storage, or geographical location. This practice protects the backup repository from damage that might occur to the volume back end.

For more information about volume snapshots, see Create, Use, or Delete Volume Snapshots in the *Storage Guide*.

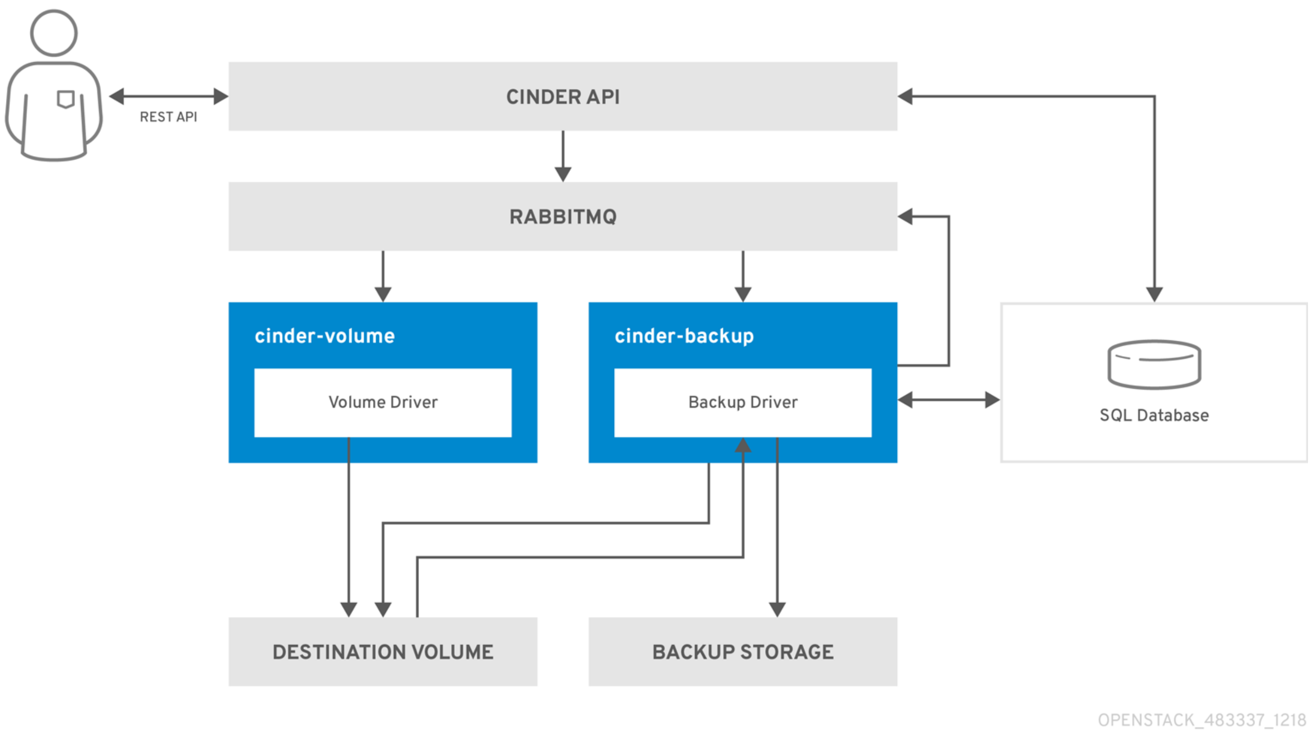
1.2. HOW DO BACKUPS AND RESTORES WORK?

Volume backups and restores have similar workflows.

1.2.1. Volume backup workflow

When the Block Storage backup service performs a back up, it receives a request from the cinder API to back up a targeted volume and store the backup content on the back end.

The following diagram illustrates how the request interacts with cinder services to perform the backup.



1. The client issues a request to back up a cinder volume by invoking the cinder REST API.
2. The cinder API service receives the request from HAProxy and validates the request, user credentials, and so forth.
    a. Creates the backup record in the SQL database.
    b. Makes an asynchronous RPC call to the cinder-backup service via AMQP to back up the volume.
    c. Returns the current backup record, with an ID, to the API caller.
3. The RPC create message arrives on one of the backup services.
4. The cinder-backup service makes a synchronous RPC call to get_backup_device.

5. The cinder-volume service ensures the correct device is returned to the caller. Normally, it is the same volume, but if the volume is in use, the service returns a temporary cloned volume or a temporary snapshot instead, depending on the configuration.

6. The cinder-backup service makes another synchronous RPC to cinder-volume to expose the source device.

7. The cinder-volume service exports and maps the source device (volume or snapshot) returning the appropriate connection information.

8. The cinder-backup service attaches source volume using connection information.

9. The cinder-backup service calls the backup driver with the device already attached, which begins the data transfer to the backup destination.

10. The volume is detached from the backup host.

11. The cinder-backup service makes a synchronous RPC to cinder-volume to disconnect the source device.

12. The cinder-volume service unmaps and removes the export from the device.

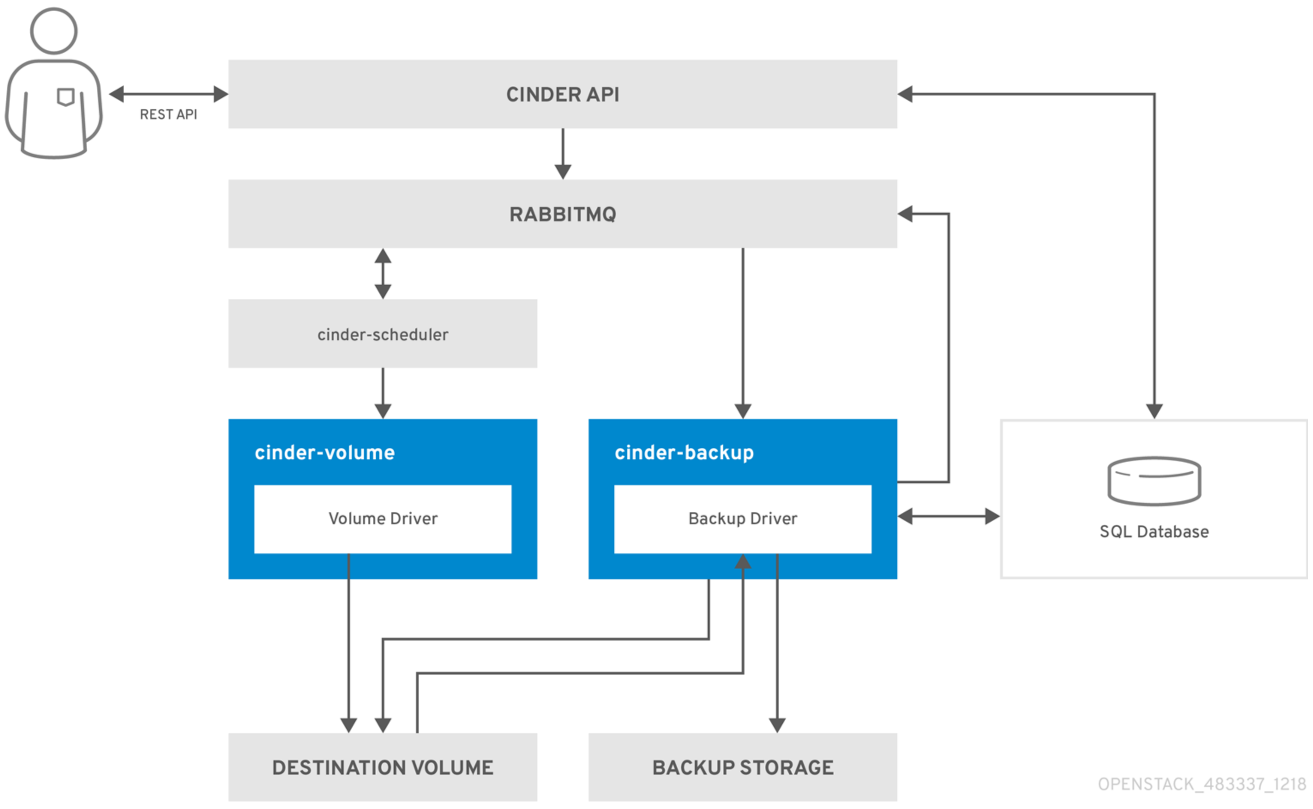
13. If a temporary volume or temporary snapshot was created, cinder-backup calls cinder-volume to remove it.

14. The cinder-volume service removes the temporary volume.

15. After the backup completed, the backup record is updated in the database.

1.2.2. Volume restore workflow

The following diagram illustrates the steps that occur when a user restores a Block Storage backup.



1. The client issues a request to restore a cinder backup by invoking the CinderREST API.
2. The cinder API receives the request from HAProxy and validates the request, user credentials, and so forth.
3. If the request did not contain an existing volume as the destination, the API makes an asynchronous RPC call to create a new volume and polls the status of the volume until it becomes available.
4. The cinder-scheduler selects a volume service and makes the RPC call to create the volume.
5. The selected cinder-volume service creates the volume.
6. After the cinder-api detects the available volume, the backup record is created in the database.
7. The cinder-api makes an asynchronous RPC call to the backup service via AMQP to restore the backup and returns the current volume ID, backup ID, and volume name to the API caller.
8. The RPC create message arrives on one of the backup services.
9. The cinder-backup service makes a synchronous RPC call to cinder-volume to expose the destination volume.
10. The cinder-volume service exports and maps the destination volume and returns the appropriate connection information.
11. The cinder-backup service attaches the source volume using connection information.
12. The cinder-backup service calls the driver with the device already attached, which begins the data restoration to the volume destination.
13. The volume is detached from the backup host.
14. The cinder-backup service issues a synchronous RPC to cinder-volume to disconnect the source device.
15. The cinder-volume service unmaps and removes the export from the device.
16. After the backup completes, the backup record is updated in the database.

CHAPTER 2. BLOCK STORAGE BACKUP SERVICE DEPLOYMENT

The Block Storage backup service is optional. It is not installed by default and must be added to the overcloud deployment.

To deploy the backup service, you need:

- A new or existing OpenStack installation
- An available storage source with a compatible backup driver: Object Storage (swift), Red Hat Ceph Storage, or NFS.

The examples in this section assume that you are deploying the back end service in a standard OpenStack environment that uses the default Pacemaker installation.

2.1. CONFIGURING BACKEND OPTIONS

The backup service is enabled by including the **cinder-backup.yaml** environment file in the **/usr/share/openstack-tripleo-heat-templates/environments/** directory.

The default settings in this file configure a swift back end for the Block Storage backup service with Pacemaker.

Procedure

Create a custom environment file, for example **cinder-backup-settings.yaml**, that contains the parameter settings for the backup service and configuration options for the driver:

1. Create a copy of the **cinder-backup.yaml** file and store it in the same location as other custom templates.

   ```
   cp /usr/share/openstack-tripleo-heat-templates/environments/cinder-backup.yaml
   /home/stack/templates/cinder-backup-settings.yaml
   ```

2. Modify the appropriate options for the backup back end that you are using (see instructions in the sections below).
3. Save the changes to the file.

2.1.1. Object storage (swift)

Swift is the default value for the CinderBackupBackend option. If you are using swift, no additional changes are required.

Example

```
resource_registry:
  OS::TripleO::Services::CinderBackup: /usr/share/openstack-tripleo-heat-
templates/docker/services/pacemaker/cinder-backup.yaml
  # For non-pcmk managed implementation
  # OS::TripleO::Services::CinderBackup: /usr/share/openstack-tripleo-heat-
templates/docker/services/cinder-backup.yaml
```

```
parameter_defaults:
  CinderBackupBackend: swift
```

Setting	Options	Values
CinderBackupBackend	swift (default)	Swift is the default selection in the **cinder-backup.yaml** template.

2.1.2. Red Hat Ceph Storage

If you are using Red Hat Ceph Storage as a backup back end, then you may change the RBD pool name used for the backup. The default value is **backups**.

Example

```
resource_registry:
  OS::TripleO::Services::CinderBackup: /usr/share/openstack-tripleo-heat-
templates/docker/services/pacemaker/cinder-backup.yaml
  # For non-pcmk managed implementation
  # OS::TripleO::Services::CinderBackup: /usr/share/openstack-tripleo-heat-
templates/docker/services/cinder-backup.yaml

parameter_defaults:
  CinderBackupBackend: ceph
  CinderBackupRbdPoolName: backups
```

Setting	Options	Values
CinderBackupBackend	ceph	Required. Change the value to **ceph**.
CinderBackupRbdPoolName	backups (default name)	Optional. No other settings must be updated unless you are using a custom RBD pool name.

2.1.3. NFS

To use NFS as a back end for the backup service, you must provide the NFS share to be mounted.

Example

```
resource_registry:
  OS::TripleO::Services::CinderBackup: /usr/share/openstack-tripleo-heat-
templates/docker/services/pacemaker/cinder-backup.yaml
  # For non-pcmk managed implementation
  # OS::TripleO::Services::CinderBackup: /usr/share/openstack-tripleo-heat-
templates/docker/services/cinder-backup.yaml
```

```
parameter_defaults:
  CinderBackupBackend: nfs
  CinderBackupNfsShare: '192.168.122.1:/export/cinder/backups'
  CinderBackupNfsMountOptions: ''
```

Setting	Options	Values
CinderBackupBackend	nfs	Required. Set **nfs** as the value.
CinderBackupNfsShare		Required. Enter the NFS share to be mounted. Default value is empty.
CinderBackupNfsMountOptions		Optional. Backup NFS mount options can be left blank. If you must specify mount options, include them here.

2.2. DEPLOYING THE BLOCK STORAGE BACKUP SERVICE

After you create the environment file in **/home/stack/templates/**, log in as the stack user and deploy the configuration by running:

```
$ openstack overcloud deploy --templates \
-e /home/stack/templates/cinder-backup-settings.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 the Including Environment Files in Overcloud Creation in the *Director Installation and Usage Guide* and the Environment Files section of the *Advanced Overcloud Customization Guide*.

CHAPTER 3. USING THE BLOCK STORAGE BACKUP SERVICE

This chapter explains how to use the Block Storage backup service to perform full or incremental backups, how to restore a backup to a volume, and basic troubleshooting tips.

3.1. FULL BACKUPS

3.1.1. Creating a full volume backup

To back up a volume, use the **cinder backup-create** command. By default, this command creates a full backup of the volume. If the volume has existing backups, you can create an incremental backup instead. For information about creating an incremental backup, see Section 2.4.1.2, "Create an Incremental Volume Backup" in the *Storage Guide*.

You can create backups of volumes to which you have access. This means that users with administrative privileges can back up any volume, regardless of owner. For more information, see Section 3.1.2, "Creating a volume backup as an admin".

Procedure

1. View the ID or Display Name of the volume you want to back up:

   ```
   # cinder list
   ```

2. Back up the volume:

   ```
   # cinder backup-create --name <full_backup_name> _VOLUME_
   ```

   Replace *VOLUME* with the **ID** or **Display Name** of the volume you want to back up. For example:

   ```
   +-----------+--------------------------------------+
   |  Property |                Value                 |
   +-----------+--------------------------------------+
   |     id    | e9d15fc7-eeae-4ca4-aa72-d52536dc551d |
   |    name   |                 None                 |
   | volume_id | 5f75430a-abff-4cc7-b74e-f808234fa6c5 |
   +-----------+--------------------------------------+
   ```

   The **volume_id** of the resulting backup is identical to the ID of the source volume.

3. Verify that the volume backup is complete:

   ```
   # cinder backup-list
   ```

   The volume backup is complete when the **Status** of the backup entry is **available**.

3.1.2. Creating a volume backup as an admin

Users with administrative privileges, such as the default admin account, can back up any volume managed by OpenStack. When an admin backs up a volume owned by a non-admin user, the backup is hidden from the volume owner by default.

Procedure

To back up a volume and make the backup available to a specific tenant:

```
# cinder --os-tenant-name _TENANTNAME_ backup-create _VOLUME_
```

Where:

- *TENANTNAME* is the name of the tenant where you want to make the backup available.
- *VOLUME* is the name or ID of the volume you want to back up.

When performing this operation, the size of the backup counts against the quota of *TENANTNAME* rather than the admin’s tenant.

3.1.3. Exporting backup metadata from a volume

You can export and store the backup metadata from a volume. Exporting and storing the backup metadata from a volume allows you to restore a volume even if the Block Storage database suffers a catastrophic loss. This is useful when you require full backups of only a small subset of volumes.

When you back up a volume and export its metadata, you can restore the volume on a different Block Storage database or cloud service. Volume encryption is retained in the backup metadata if you specify UUID encryption during volume creation. The restored, encrypted volume is then accessible with your credentials.

Procedure

Run the following command to export and store backup metadata:

```
# cinder backup-export _BACKUPID_
```

Where *BACKUPID* is the ID or name of the volume backup. For example,

```
+----------------+------------------------------------------+
|    Property    |                Value                     |
+----------------+------------------------------------------+
| backup_service |     cinder.backup.drivers.swift          |
|   backup_url   | eyJzdGF0dXMiOiAiYXZhaWxhYmxlIiwgIm9iam...|
|                | ...4NS02ZmY4MzBhZWYwNWUiLCAic2l6ZSI6IDF9 |
+----------------+------------------------------------------+
```

The volume backup metadata consists of the **backup_service** and **backup_url** values.

3.1.4. Backing up an in-use volume

You can use the **cinder backup-create** command to create a backup of an in-use volume by adding the **--force** option.

NOTE

The **--force** option relies on Block Storage back end snapshot support and should be supported by most drivers. You can verify snapshot support by checking the documentation for the back end you are using.

By using the **--force** option, you acknowledge that you are not quiescing the drive before performing the backup. Using this method creates a crash-consistent, but not application-consistent, backup. In other words, the backup will not have an awareness of which applications were running when the backup was performed. However, the data will be intact.

Procedure

To create a backup of an in-use volume, run:

```
# cinder backup-create _VOLUME_ --incremental --force
```

3.1.5. Backing up a snapshot

You can create a full backup from a snapshot by using the volume ID that is associated with the snapshot.

Procedure

1. Locate the snapshot ID of the snapshot to back up using **cinder snapshot list**.

   ```
   # cinder snapshot-list --volume-id _VOLUME_ID_
   ```

2. If the snapshot is named, you can use this example to locate the **ID**:

   ```
   # cinder snapshot-show _SNAPSHOT_NAME_
   ```

3. Create the backup of a snapshot:

   ```
   # cinder backup-create _VOLUME_ --snapshot-id=_SNAPSHOT_ID_
   ```

NOTE

Snapshot-based backups of NFS volumes fail when you use the **--snapshot-id** option. This is a known issue.

3.2. INCREMENTAL BACKUPS

The Block Storage backup service provides the option of performing incremental backups.

3.2.1. Performance considerations

Some backup features, such as incremental and data compression, may impact performance. Incremental backups have a performance impact because all of the data in a volume must be read and checksummed for both the full backup and each incremental backup.

Data compression can be used with non-Ceph backends. Enabling data compression requires additional CPU power but uses less network bandwidth and storage space overall.

Multipathing configuration also impacts performance. If multiple volumes are attached without multipathing enabled, you might not be able to connect to the volumes or experience full network capabilities, which impacts performance.

You can use the advanced configuration options (see Appendix A, *Advanced Block Storage backup configuration options*) to enable or disable compression, define the number of processes, and add additional CPU resources.

3.2.1.1. Backup from a snapshot impact

Some back ends support creating a backup from a snapshot. A driver that supports this feature can directly attach a snapshot to a volume, which is faster than cloning the snapshot into a volume. In general, this feature can affect performance because of the extra step of creating the volume from a snapshot.

3.2.2. Performing incremental backups

By default, the **cinder backup-create** command creates a full backup of a volume. However, if the volume has existing backups, you can choose to create an incremental backup.

Incremental backups are fully supported on NFS, Object Storage (swift), and Red Hat Ceph Storage backup repositories. Ceph backups currently ignore the **--incremental** option. Ceph backups attempt to perform incremental backups when the source is a Ceph volume, therefore it is not possible to perform multiple full backups.

NOTE

Incremental Ceph backups cannot be performed for non-Ceph volumes.

An incremental backup captures any changes to the volume since the last backup (full or incremental). Performing numerous, regular, full backups of a volume can become resource-intensive as the volume's size increases over time. Incremental backups allow you to capture periodic changes to volumes while minimizing resource usage.

Procedure

To create an incremental volume backup, use the **--incremental** option:

```
# cinder backup-create --name <incremental_backup_name> --incremental _VOLUME_
```

Replace *VOLUME* with the **ID** or **Display Name** of the volume you want to back up.

NOTE

You cannot delete a full backup if it already has an incremental backup. If a full backup has multiple incremental backups, you can only delete the latest one.

3.3. CANCELING A BACKUP

To cancel a backup, an admin must request a force delete on the backup:

```
# cinder backup-delete --force BACKUP ID
```

Even if the backup is immediately deleted and no longer appears in the listings, the cancellation operation may still be running. Check the status of the source resource to verify when the status is no longer "backing-up".

NOTE

Before Red Hat OpenStack Platform version 12, the “backing-up” status was stored in the volume, even when backing up a snapshot. Therefore, when backing up a snapshot, any delete operation on the snapshot that followed a cancellation could result in an error if the snapshot was still mapped. In Red Hat OpenStack Platform version 13 and later, ongoing restoration operations can be canceled on any of the supported backup drivers.

3.4. VIEWING AND MODIFYING THE BACKUP QUOTA OF A TENANT

You can use the dashboard to modify tenant storage quotas, for example, the number of volumes, volume storage, snapshots, or other operational limits that a tenant can have. However, you cannot modify backup quotas through the dashboard.

You must use the command-line interface to modify backup quotas by using the **cinder quota-update** command.

Procedure

1. To view the storage quotas of a specific tenant (*TENANT_ID*), run:

```
# cinder quota-show TENANT_ID
```

2. To update the maximum number of backups (*MAXNUM*) that can be created in a specific tenant, run:

```
# cinder quota-update --backups MAXNUM TENANT_ID
```

3. To update the maximum total size of all backups (*MAXGB*) within a specific tenant, run:

```
# cinder quota-update --backup-gigabytes MAXGB TENANT_ID
```

4. To view the storage quota usage of a specific tenant, run:

```
# cinder quota-usage TENANT_ID
```

3.5. RESTORING FROM BACKUPS

3.5.1. Restoring a volume from a backup

These steps create a new volume from a backup.

Procedure

1. Find the ID of the volume backup you want to use:

```
# cinder backup-list
```

The Volume ID must match the ID of the volume you want to restore.

2. Restore the volume backup:

```
# cinder backup-restore _BACKUP_ID_
```

Where *BACKUP_ID* is the ID of the volume backup you want to use.

3. If you no longer need the backup, delete it:

```
# cinder backup-delete _BACKUP_ID_
```

4. If you must restore a backed up volume to a volume of a particular type, use the **--volume** option to restore a backup to a specific volume:

```
# cinder backup-restore _BACKUP_ID --volume VOLUME_ID_
```

NOTE

If you restore a volume from an encrypted backup, then the destination volume type must also be encrypted.

3.5.2. Restoring a volume after a Block Storage database loss

Typically, a Block Storage database loss prevents you from restoring a volume backup. This is because the Block Storage database contains metadata required by the volume backup service **openstack-cinder-backup**. This metadata consists of backup_service and backup_url values, which you can export after creating the volume backup. For more information, see Section 3.1.1, "Creating a full volume backup".

If you exported and stored this metadata, you can import it to a new Block Storage database, which enables you to restore the volume backup.

NOTE

For incremental backups, you must import all exported data before you can restore one of the incremental backups.

Procedure

1. As a user with administrative privileges, run:

```
# cinder backup-import _backup_service_ _backup_url_
```

Where *backup_service* and *backup_url* are from the metadata you exported. For example, using the exported metadata from Section 3.1.1, "Creating a full volume backup" :

```
# cinder backup-import cinder.backup.drivers.swift eyJzdGF0dXMi...c2l6ZSI6IDF9
+----------+--------------------------------------+
| Property |                Value                 |
+----------+--------------------------------------+
|    id    | 77951e2f-4aff-4365-8c64-f833802eaa43 |
|   name   |                 None                 |
+----------+--------------------------------------+
```

2. After the metadata is imported into the Block Storage service database, you can restore the volume as normal. For more information, see Section 3.5.1, "Restoring a volume from a backup" .

3.5.3. Canceling a restoration

Procedure

1. To cancel the restoration of a backup, alter its status to anything other than **restoring**. You can use the **error** state to minimize confusion regarding whether the restore was successful or not. Alternatively, you can change the value to **available**.

   ```
   $ openstack volume backup set --state error BACKUP_ID
   ```

NOTE

Backup cancellation is an asynchronous action, because the backup driver must detect the status change before it cancels the backup. When the status changes to **available** in the destination volume, the cancellation is complete.

NOTE

This feature is not currently available on RBD backups.

WARNING

If a restore operation is canceled after it starts, the destination volume is useless, because there is no way of knowing how much data, if any, was actually restored.

3.6. TROUBLESHOOTING

You can diagnosed many issues by verifying that services are available and by looking in log files for error messages.

Two scenarios account for many issues with the backup service:

- When the cinder-backup service starts up, it connects to its configured back end and uses the back end as a target for backups. Problems with this connection can cause a service to be down.
- When backups are requested, the backup service connects to the volume service and attaches the requested volume. You will not notice connection problems until backup time.

In either case, review the logs for error messages.

For more information about log locations and troubleshooting suggestions, see Logging, Monitoring and Troubleshooting Guide. Log files and services are listed in the Log Files for OpenStack Services section.

3.6.1. Verifying services

Verify that the necessary services are available and check the logs for error messages. Section 1.2, “How do backups and restores work?” illustrates the key services and their interaction.

NOTE

When troubleshooting an active/passive configuration, make sure to review the log files on the active controller node.

After you verify the status of the services, check the **cinder-backup.log** file. The Block Storage Backup service log is located in **/var/log/containers/cinder/cinder-backup.log**.

Procedure

1. Run **cinder show** on the volume to see if the volume is stored by the host:

```
# cinder show
```

2. Run **cinder service-list** to view running services:

```
# cinder service-list
+------------------+-------------------+------+---------+-------+----------------------------+--------------
---+
| Binary           | Host              | Zone | Status  | State | Updated_at                 | Disabled
Reason |
+------------------+-------------------+------+---------+-------+----------------------------+--------------
---+
| cinder-backup    | hostgroup         | nova | enabled | up    | 2017-05-15T02:42:25.000000 | -
|
| cinder-scheduler | hostgroup         | nova | enabled | up    | 2017-05-15T02:42:25.000000 | -
|
| cinder-volume    | hostgroup@sas-pool | nova | enabled | down  | 2017-05-
14T03:04:01.000000 | -               |
| cinder-volume    | hostgroup@ssd-pool | nova | enabled | down  | 2017-05-
14T03:04:01.000000 | -               |
+------------------+-------------------+------+---------+-------+----------------------------+--------------
---+
```

3. Run **pcs resource** to verify the controller on which the cinder backup resource is running:

```
$ pcs resource

openstack-cinder-volume (systemd:openstack-cinder-volume): Started controller-0
```

3.6.2. Troubleshooting tips

Backups are asynchronous. The Block Storage backup service performs a small number of static checks upon receiving an API request, such as checking for an invalid volume reference (missing) or a volume that is **in-use** or attached to an instance. The in-use case requires you to use the **--force** option.

NOTE

Using the **--force** option means that I/O will not be quiesced and the resulting volume image may be corrupt.

If the API accepts the request, the backup occurs in the background. The CLI returns immediately, even if the backup has failed or is about to fail. The status of a backup can be queried using the cinder backup API. If an error occurs, review the logs for the cause.

3.6.3. Pacemaker

By default, the Block Storage backup service is deployed with Pacemaker in active/passive mode. By configuring virtual IP addresses, containers, services, and other features as resources in a cluster, Pacemaker ensures that the defined set of OpenStack cluster resources are running and available. When a service or an entire node in a cluster fails, Pacemaker can restart the resource, take the node out of the cluster, or reboot the node. Requests to most of those services are done through HAProxy.

For information on using Pacemaker for troubleshooting, refer to the Managing high availability services with Pacemaker of the *High Availability Deployment and Usage* guide.

APPENDIX A. ADVANCED BLOCK STORAGE BACKUP CONFIGURATION OPTIONS

Prior to director-deployed installations, the cinder.conf file was used to configured the Block Storage service and the backup service. When a value from cinder.conf does not have a Heat template equivalent, the values can still be passed from director using a custom environment. The values are added to an **ExtraConfig** section in the **parameter_defaults** section of a custom environment file (like **cinder-backup-settings.yaml**).

ExtraConfig provides a method for additional hiera configuration to inject into the cluster on all nodes. These settings are included on a dedicated backup node, for example, if you used **ExtraConfig**. If you used **ControllerExtraConfig** instead of **ExtraConfig**, then those settings would only be installed on controller nodes and not on a dedicated backup node.

You can substitute **DEFAULT/[cinder.conf setting]** for the setting that would be used in the **DEFAULT** section of the cinder.conf file. The example below shows how the **ExtraConfig** and entries appear in a YAML file.

```
parameter_defaults:
 ExtraConfig:
   cinder::config::cinder_config:
     DEFAULT/backup_compression_algorithm:
       value: None
```

The options below provide a sample of the backup-related options.

Table A.1. Block Storage backup service configuration options

Option	Type	Default Value	Description
backup_service_inithost_offload	Optional	True	Offload pending backup delete during backup service startup. If false, the backup service remains down until all pending backups are deleted.
use_multipath_for_image_xfer	Optional	False	Attach volumes using multipath, if available, during backup and restore procedures. This affects all cinder attach operations, such as create volume from image, generic cold migrations, and so forth.
num_volume_device_scan_tries	Optional	3	The maximum number of times to rescan targets to find volume during attach.

Option	Type	Default Value	Description
backup_workers	Optional	1	Number of backup processes to run. Performance gains will be significant when running multiple concurrent backups or restores with compression.
backup_native_threads_pool_size	Optional	60	Size of the native threads pool for the backups. Most backup drivers rely heavily on this. The value can be decreased for specific drivers that don’t rely on this option.
backup_share	Required		Set to *HOST*:_EXPORT_PATH_.
backup_container	Optional	None	(String) Custom directory to use for backups.
backup_enable_progress_timer	Optional	True	Enable (true) or disable (false) the timer to send the periodic progress notifications to Ceilometer when backing up the volume to the backend storage.
backup_mount_options	Optional		Comma-separated list of options to be specified when mounting the NFS export specified in backup_share.
backup_mount_point_base	Optional	$state_path/backup_mount	(String) Base directory containing mount point for NFS share.

Option	Type	Default Value	Description
backup_compression_algorithm	Optional	zlib	The compression algorithm to be used when sending backup data to the repository. Valid values are **zlib**, **bz2**, and **None**.
backup_file_size	Optional	1999994880	Data from cinder volumes larger than this will be stored as multiple files in the backup repository. This option must be a multiple of backup_sha_block_size_bytes.
backup_sha_block_size_bytes	Optional	32768	Size of cinder volume blocks for digital signature calculation