Hewlett Packard
Enterprise

OpenStack HPE StoreVirtual Block Storage Driver Configuration best practices

OpenStack Pike release

Technical white paper

Contents

Revision history

Rev.	Date	Description
1.0	8-Apr-2016	Initial release to support the OpenStack® Mitaka release • The **legacy mode** version of the driver which uses SSH/CLIQ to communicate with the HPE StoreVirtual Storage was removed this release • All driver paths and configuration options have been updated with the Hewlett Packard Enterprise (HPE) rebranding • Requires the **python-lefthandclient** version 2.X from PyPI • Configure the suppression of Python requests for library SSL certification • Added support for manage and unmanage of snapshots • Volume replication support • Image cache support
2.0	8-Nov-2016	Updated for the OpenStack Newton release
3.0	23-Feb-2017	Update for OpenStack Ocata release • Image-Volume cache functionality is supported in the HPE LeftHand driver
4.0	18-July-2017	Update for OpenStack Pike release • Generic Volume Group support • Known Limitations

Note

This document should be used for the OpenStack Mitaka, and later releases. An older version of the document exists for the Hewlett Packard Enterprise Cinder drivers for the OpenStack Liberty release and earlier, 4AA5-1966ENW.

Executive summary

Hewlett Packard Enterprise brings the power of OpenStack to the enterprise. HPE's commitment to the OpenStack community brings new and enhanced offerings that enable enterprises to increase agility, speed innovation, and lower costs.

Since the Grizzly release, Hewlett Packard Enterprise has been a top contributor to the advancement of the OpenStack project.[1] HPE's contributions have focused on continuous integration and quality assurance, which has supported the development of a reliable and scalable cloud platform that is equipped to handle production workloads.

To support the need that many larger organizations and service providers have for enterprise-class storage, Hewlett Packard Enterprise has developed the HPE StoreVirtual Block Storage Driver, which supports the OpenStack technology using the iSCSI protocol. This provides the flexibility and cost-effectiveness of a cloud-based open-source platform to customers with mission-critical environments and high resiliency requirements. Figure 1 shows the high-level components of a basic cloud architecture.

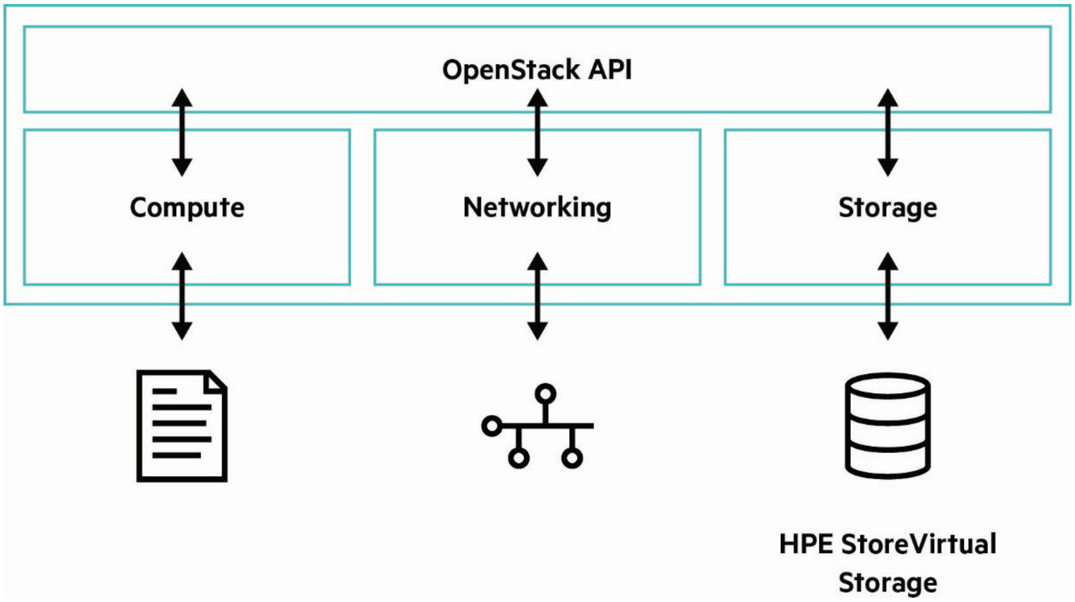


Figure 1. OpenStack cloud architecture

[1] Stackalytics.com, "OpenStack Pike Analysis," March 2016. stackalytics.com/?module=cinder-group&metric=commits&release=pike

Introduction

This document provides information on the new best practice features in the OpenStack Mitaka, Newton, and Ocata releases. These include configuring and using volume types, extra specs, consistency groups, oversubscription, and multiple backend support with the HPE StoreVirtual Block Storage Driver.

The HPE StoreVirtual iSCSI Driver is based on the Block Storage (Cinder) plug-in architecture. The driver executes the volume operations by communicating with the HPE StoreVirtual Storage system over HTTPS and Secure Shell (SSH) connections. The HTTPS communication uses the HPE LeftHand client, which is part of the Python Package Index (PyPI).

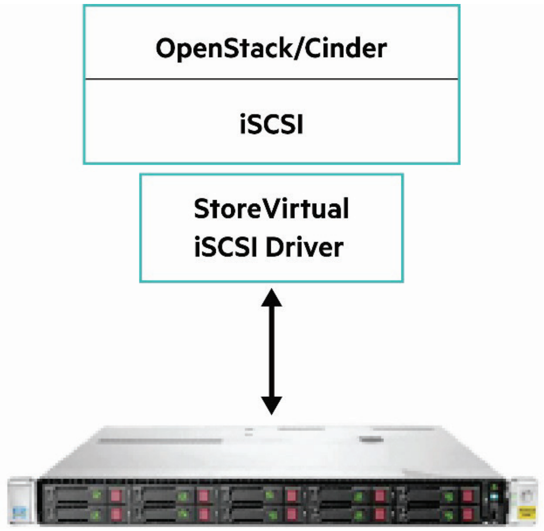


Figure 2. HPE StoreVirtual iSCSI Driver for OpenStack/Cinder

HPE StoreVirtual Storage

HPE StoreVirtual Storage uses a single architecture, shown in Figure 3, to deliver primary storage platforms for midrange, enterprise, and optimized all-flash arrays.

HPE StoreVirtual Storage, based on the HPE LeftHand operating system, is a scale-out storage platform that is designed to meet the fluctuating needs of virtualized environments. Intuitive, common management, and storage federation meet the need for simplicity and flexibility in today's virtual data centers.

It allows data mobility across tiers, locations, and between physical and virtual storage. HPE StoreVirtual is a versatile storage platform on the market today.[2]

Its software-defined storage VSA software and HPE ProLiant rack and BladeSystem-based hardware models provide options to fit any infrastructure and budget. Enterprise-class storage software functionality and leading virtualization software integration are built-in.

[2] Stated on the HPE StoreVirtual webpage, hpe.com/storage/storevirtual

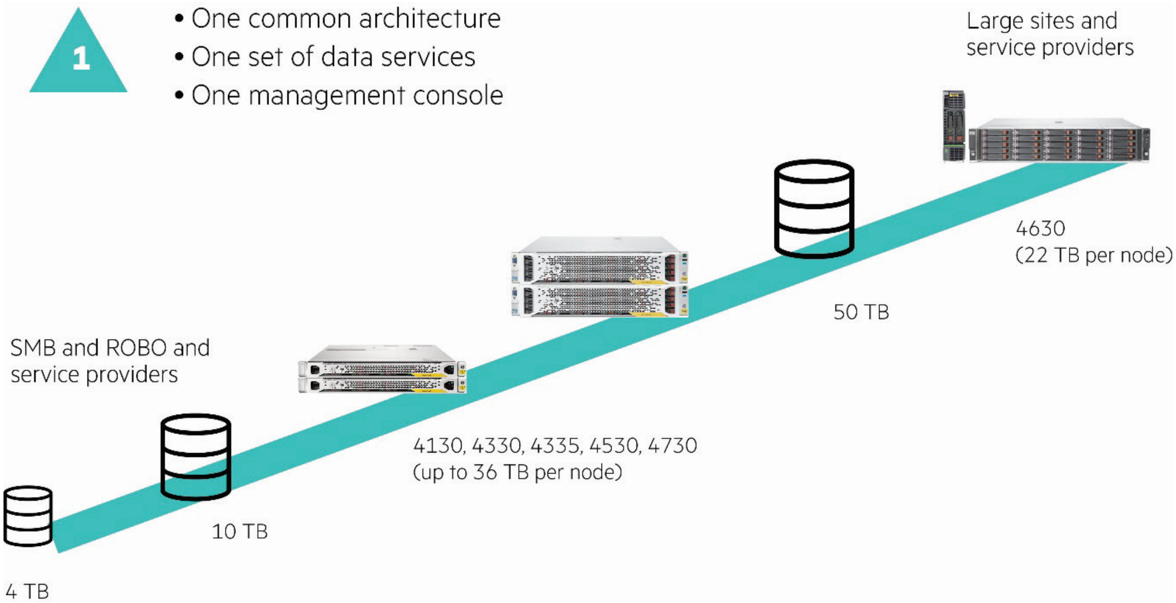

1
• One common architecture
• One set of data services
• One management console
Large sites and service providers
4630
(22 TB per node)
50 TB
SMB and ROBO and service providers
4130, 4330, 4335, 4530, 4730
(up to 36 TB per node)
10 TB
4 TB

Figure 3. HPE StoreVirtual Storage

Configuration

The HPE StoreVirtual Block Storage Drivers for iSCSI was introduced in the OpenStack Icehouse release. Since that release, several configuration improvements have been made that include, but are not limited to, the following:

- The array requires the HPE LeftHand/StoreVirtual Operating System software version 11.5 or higher.

Kilo

- The array requires the HPE LeftHand/StoreVirtual Operating System software version 12.0 or higher only if volume `manage` and `unmanage` is going to be used.

Liberty

- Cinder's `cinder.conf` settings of `max_over_subscription_ratio` is used for oversubscription of thin-provisioned volumes and `reserved_percentage` to prevent overprovisioning on the HPE StoreVirtual.
- The legacy mode version of the driver, which uses SSH/CLIQ, was marked as deprecated in the Liberty release.

Mitaka/Newton

- Requires the `python-lefthandclient` version 2.1.0 or newer from PyPI. The client was rebranded from the `lefthandclient`.
- HPE StoreVirtual Cinder driver's paths and configuration settings updated as part of the Hewlett Packard Enterprise rebranding.
- Added volume replication support to the HPE StoreVirtual driver.
- The legacy mode version of the driver that used SSH/CLIQ was removed in the Mitaka release.

Pike

- Generic volume group is supported for HPE StoreVirtual driver and support for consistency group is removed.
- Added few known limitations for HPE StoreVirtual driver.

Volume types creation

Block Storage volume types are a type or label that can be selected at volume create time in OpenStack. These types can be created either in the Admin Horizon UI or using the command line, as shown:

```
$cinder --os-username admin --os-tenant-name admin type-create <name>
```

The <name> is the name of the new volume type. After the volume type names have been created, you can assign extra specs or capabilities to these types. This will affect the Block Storage Driver's behavior, depending on which extra specs you assign to each type you created.

This example illustrates how to create three new volume type names with the names gold, silver, and bronze:

```
$cinder --os-username admin --os-tenant-name admin type-create gold
```

```
$cinder --os-username admin --os-tenant-name admin type-create silver
```

```
$cinder --os-username admin --os-tenant-name admin type-create bronze
```

Setting extra_specs or capabilities

The extra_specs or capabilities must be set/unset for a volume type. The extra_specs are set or unset either in the Admin Horizon UI or using the command line, as shown:

```
$cinder --os-username admin --os-tenant-name admin type-key <vtype><action> [<key=value>
[<key=value> ...]]
```

The argument <vtype> is the name or ID of the volume type. The argument <action> must be one of the actions: set or unset. The optional argument <key=value> is the extra_specs to set/unset the key. Only the key is necessary on unset. When extra_specs are set on a volume type, you must add the following key value pair to connect it to a particular Block Storage Driver.

The <volume backend name> must match the same value that is in the cinder.conf file for a particular Block Storage Driver. Each volume type must also have a volume_backend_name assigned to it.

volume_backend_name=<volume backend name>

Set examples

```
$cinder type-key gold set hpelh:data_pl=r-6 hpelh:provisioning=full hpelh:ao=true
```

```
$cinder type-key silver set hpelh:data_pl=r-5 hpelh:provisioning=full
```

```
$cinder type-key bronze set hpelh:data_pl=r-0 hpelh:provisioning=thin
```

Unset examples

```
$cinder type-key gold unset hpelh:ao
```

When you want to unset a particular key value pair from a volume type, the value is not required—only the key is.

The current HPE StoreVirtual extra_specs that can be specified in the Icehouse release require a scoping of the LeftHand keys. The filter scheduler uses the extra_specs data to determine capabilities and the backend. It also enforces strict checking.

Therefore, the scheduler requires scoping or adding, hpelh: to the LeftHand keys. The current list of supported LeftHand keys includes:

- hpelh:provisioning—The valid values are **thin** and **full**. Default **thin**.
- hpelh:ao—The valid values are **true** and **false**. Default **true**.

- `hpelh:data_pl`—The valid values are: Default **r-0**.
  - `r-0` Network RAID-0 (None)
  - `r-5` Network RAID-5 (Single Parity)
  - `r-10-2` Network RAID-10 (2-Way Mirror)
  - `r-10-3` Network RAID-10 (3-Way Mirror)
  - `r-10-4` Network RAID-10 (4-Way Mirror)
  - `r-6` Network RAID-6 (Dual Parity)

Any or all of the above capabilities can be set on a volume type. These are additional capabilities that the HPE StoreVirtual Storage array offers. To list all the volume types and `extra_specs` currently configured, use the following command:

```
$ cinder --os-username admin --os-tenant-name admin extra-specs-list
```

Multiple storage backend support and Block Storage configuration

Multiple backend support was added in the Grizzly release to OpenStack. Detailed instructions on setting up multiple backends can be found in the OpenStack Administrator Guide.

The multi-backend configuration is done in the `cinder.conf` file. The `enabled_backends` flag has to be set up. This flag defines the names (separated by a comma) of the configuration groups for the different backends. One name is associated to one configuration group for a backend (e.g., [lefthand-1]). Each group must have a full set of the driver-required configuration options. The list shown in Figure 4 shows the entries in `cinder.conf` for the HPE StoreVirtual Storage array backends.

```
# List of backends that will be served by this node
enabled_backends=lefthand-rest-1,lefthand-rest-2

# [lefthand-rest-1]
volume_driver=cinder.volume.drivers.hpe.hpe_lefthand_iscsi.HPELeftHandISCSIDriver
volume_backend_name=lefthand-rest
hpelefthand_api_url=https://10.10.22.241:8081/lhos
hpelefthand_username=<username>
hpelefthand_password=<password>
hpelefthand_clustername=lefthandSmallCluster
max_over_subscription_ratio=10.0
reserved_percentage=15
image_volume_cache_enabled = True

# [lefthand-rest-2]
volume_driver=cinder.volume.drivers.hpe.hpe_lefthand_iscsi.HPELeftHandISCSIDriver
volume_backend_name=lefthand-rest
hpelefthand_api_url=https://10.10.22.244:8081/lhos
hpelefthand_username=<username>
hpelefthand_password=<password>
hpelefthand_clustername=lefthandLargeCluster
image_volume_cache_enabled = True
```

Figure 4. Sample cinder.conf file

In this configuration, both the lefthand-rest-1 and lefthand-rest-2 have the same `volume_backend_name`. When a volume request comes in with the LeftHand backend name, the scheduler must choose which one is most suitable. This is done with the capacity filter scheduler detailed in Block Storage scheduler configuration with multi-backend section. This example also includes a single, legacy-configured LeftHand Cinder driver with a different `volume_backend_name`.

The oversubscription feature adds two additional parameters that must be set at the global level to apply to all drivers or under each driver instance in Mitaka. The first new parameter, `max_over_subscription_ratio`, is the ratio of oversubscription when thin-provisioned volumes are involved. The default ratio is 20.0; this means that a provisioned capacity can be 20 times of the total physical capacity. The second is `reserved_percentage`; this option represents the percentage of reserved backend capacity. The default percentage is zero. The capacity filter scheduler takes these values into account during its decision for a backend to host a particular volume request.

Block Storage scheduler configuration with multi-backend

Multi-backend **must be** used with filter scheduler enabled. Filter scheduler acts in two steps:

1. Filter scheduler filters the available backends. By default, `AvailabilityZoneFilter`, `CapacityFilter`, and `CapabilitiesFilter` are enabled.
2. Filter scheduler weighs the previously filtered backends. By default, `CapacityWeigher` is enabled. The `CapacityWeigher` attributes high scores to backends with the most available space.

According to the filtering and weighing, the scheduler is able to pick “the best” backend to handle the request. In that way, filter scheduler achieves the goal of explicitly creating volumes on specific backends using volume types.

Block Storage scheduler configuration with driver filter and weigher

The driver filter and weigher for the Block Storage scheduler is a feature (new in Kilo) that, when enabled, allows a filter and goodness function to be defined in your `cinder.conf` file. The two functions are used during volume creation time by the Block Storage scheduler to determine which backend is the ideal for the volume. The filter function is used to filter out backend choices that should not be considered at all. The goodness function is used to rank the filtered backends from zero to 100. This feature should be used when the default Block Storage scheduling does not provide enough control for where volumes are being created.

Enable the usage of the driver filter for the scheduler by adding `DriverFilter` to the `scheduler_default_filters` property in your `cinder.conf` file. Enabling the driver weigher is similar. Add `GoodnessWeigher` to the `scheduler_default_weighers property` in your `cinder.conf` file. If you want to include other OpenStack filters and weighers in your setup, make sure to add those to the `scheduler_default_filters` and `scheduler_default_weighers` properties as well.

Note

You can choose to have only the `DriverFilter` or `GoodnessWeigher` enabled in your `cinder.conf` file depending on how much customization you want.

OpenStack supports various math operations that can be used in the filter and goodness functions. The currently supported list of math operations can be seen in Table 1.

Table 1. Supported math operations for filter and goodness functions

Operations	Type
+, -, *, /, ^	standard math
not, and, or, &, |, !	logic
>, >=, <, <=, ==, <>, !=	equality
+, -	sign
x ? a: b	ternary
abs(x), max(x,y), min(x,y)	math helper functions

Several driver-specific properties are available for use in the filter and goodness functions for an HPE StoreVirtual backend. The currently supported list of HPE StoreVirtual specific properties includes:

- **capacity_utilization**—Percent of total space used on the HPE StoreVirtual cluster
- **total_volumes**—The total number of volumes on the HPE StoreVirtual cluster

Additional generic volume properties are available from OpenStack for use in the filter and goodness functions. These properties can be seen in the OpenStack Administrator Guide.

Note
Access the HPE LeftHand specific properties by using the following format in your filter or goodness functions: `capabilities.<property>`.

The sample **cinder.conf** file in Figure 5 shows an example of how several HPE StoreVirtual backends could be configured to use the driver filter and weigher from the Block Storage scheduler.

```
[default]
cinder_internal_tenant_project_id = PROJECT_ID
cinder_internal_tenant_user_id = USER_ID
scheduler_default_filters = DriverFilter
scheduler_default_weighers = GoodnessWeigher
enabled_backends = lefthand-rest-1, lefthand-rest-2, lefthand-rest-3
[lefthand-rest-1]
hpelefthand_api_url = <api_url>
hpelefthand_username = <username> hpelefthand_password = <password>
hpelefthand_clustername = cluster-1
volume_backend_name = lefthand
volume_driver = cinder.volume.drivers.san.hpe.hpe_lefthand_iscsi.HPELeftHandISCSIDriver
filter_function = “capabilities.total_volumes< 10”
goodness_function = “(capabilities.capacity_utilization< 75)? 90 : 50”
image_volume_cache_enabled = True
[lefthand-rest-2]
hpelefthand_api_url = <api_url>
hpelefthand_username = <username>
hpelefthand_password = <password>
hpelefthand_clustername = cluster-2
volume_backend_name = lefthand
volume_driver = cinder.volume.drivers.hpe.hpe_lefthand_iscsi.HPELeftHandISCSIDriver
filter_function = “capabilities.total_volumes< 10”
goodness_function = “(capabilities.capacity_utilization< 50)? 95 : 45”
image_volume_cache_enabled = True
[lefthand-rest-3]
hpelefthand_api_url = <api_url>
hpelefthand_username = <username>
hpelefthand_password = <password>
hpelefthand_clustername = cluster-3
volume_backend_name = lefthand
```

```
volume_driver = cinder.volume.drivers.hpe.hpe_lefthand_iscsi.HPELeftHandISCSIDriver
filter_function = “capabilities.total_volumes< 20”
goodness_function = “(capabilities.capacity_utilization< 90)? 75 : 40”
image_volume_cache_enabled = True
```

Figure 5. Sample cinder.conf file showing driver filter and weigher usage

In Figure 5, there are three HPE StoreVirtual backends enabled in the cinder.conf file. The sample shows how you can use HPE StoreVirtual specific properties to distribute volumes with more control than the default Block Storage scheduler.

Note
Remember that you can combine the HPE StoreVirtual specific properties with the generic volume properties provided by OpenStack. Also, the values used in the previous sample are only examples. In your own environment, you have full control over the filter and goodness functions that you create. For more details and examples, see the OpenStack Administrator Guide.

Volume types assignment

Use the following command or the Admin Horizon UI (new in Juno) to specify a `volume_backend_name` for each volume type you create. This links the volume type to a backend name.

```
$ cinder --os-username admin --os-tenant-name admin type-key gold set
volume_backend_name=lefthand_gold
```

The second volume type could be for an iSCSI Driver volume type named silver.

```
$ cinder --os-username admin --os-tenant-name admin type-key silver set
volume_backend_name=lefthand_silver
```

Multiple key value pairs can be specified when running the previous command. For example, the following command is used to create a volume type named “gold,” with a data protection level of “r-6,” Network RAID-6 (Dual Parity), and with “full” provisioning.

```
$ cinder --os-username admin --os-tenant-name admin type-key gold set
volume_backend_name=lefthand_goldhpelh:data_pl=r-6 hpelh:provisioning=full
```

Volume manage and unmanage

Starting in the Kilo release, HPE StoreVirtual volumes can be managed and unmanaged. This allows importing non-OpenStack volumes already on an HPE StoreVirtual Storage array into OpenStack/Cinder or exporting, which removes them from the OpenStack/Cinder perspective. However, the volume on the HPE StoreVirtual Storage array is left intact. From the command line, you can see the available driver instances represented as “hosts” within the `cinder.conf` file. This host is where the HPE StoreVirtual volume that you want to manage resides. To see the list of hosts, use the following command:

```
$ cinder-manage host list

mystack

mystack@lefthand

mystack@lefthand-1

mystack@lefthand-2
```

To manage volumes that exist on the HPE LeftHand but are not already managed by OpenStack/Cinder, use the command:

```
$ cinder manage --name <cinder name><host>#<pool><source-name>
```

Where <source-name> represents the name of the volume to manage and <cinder name> is optional but represents the OpenStack name and <host> represents the driver instance. The <pool> value is required. For the HPE StoreVirtual drivers, this is just a repeat of the driver backend name. The manage volume command also accepts an optional <--volume-type> parameter that performs a retype of the virtual volume after being managed. The following is an example of the manage command:

```
$ cinder manage --name vol-lhmystack@lefthand#lefthand volume12345
```

Note
Cinder manage renames the volume on the HPE StoreVirtual Storage array to a name that starts with volume- followed by a UUID as this is required for OpenStack/Cinder to locate the volume under its management.

To unmanage a volume from OpenStack/Cinder and leave the volume intact on the HPE StoreVirtual Storage array, use the command:

```
$ cinder unmanage <volume_id>
```

Where <volume_id> is the ID of the OpenStack/Cinder volume to unmanage. The following is an example of the unmanage command:

```
$ cinder unmanage 647bd7d5-e8e1-4701-9038-7b4de14fda18
```

Note
Cinder unmanage removes the OpenStack/Cinder volume from OpenStack but the volume remains intact on the HPE StoreVirtual Storage array. The volume name has unm- prefixed to it, followed by an encoded UUID. This is required because the HPE StoreVirtual has name length and character limitations.

Snapshot manage and unmanage

Starting in the Mitaka release, volume snapshots can be managed and unmanaged from OpenStack. Unmanaging a volume snapshot allows the snapshot to be removed from the OpenStack perspective but remain intact on the HPE StoreVirtual Storage array. This can be useful when volumes are being unmanaged from OpenStack, as the volume's snapshots will not be automatically unmanaged. Bringing volume snapshots back into the OpenStack perspective involves using the manage command for snapshots. The process for both managing and unmanaging snapshots is similar to the managing and unmanaging of volumes.

To manage a volume snapshot from an HPE StoreVirtual Storage array, use the command:

```
$ cinder snapshot-manage <parent_volume_name> <snapshot_name>
```

Where <parent_volume_name> represents the name of the parent volume for the snapshot being managed and <snapshot_name> is the name of the snapshot on the HPE StoreVirtual Storage array:

```
$ cinder snapshot-manage 16fn6873-eb09-49h2-8c1f-91ccb83be95k ums-647kx7d5-e8e1-4701-9374-7b5nb14ffh18
```

Note
Cinder snapshot-manage renames the snapshot on the HPE StoreVirtual Storage array to a name that starts with snapshot- followed by a UUID, as this is required for OpenStack/Cinder to locate the volume under its management.

To unmanage a volume snapshot from OpenStack/Cinder and leave the snapshot intact on the HPE StoreVirtual Storage array, use the command:

```
$ cinder snapshot-unmanage <snapshot_name>
```

Where <snapshot_id> is the ID of the OpenStack/Cinder snapshot to unmanage:

```
$ cinder snapshot-unmanage 16ab6873-eb09-4522-8d0f-91ccb83be56k
```

Note
Cinder `snapshot-unmanage` removes the OpenStack/Cinder snapshot from OpenStack but the snapshot remains intact on the HPE StoreVirtual Storage array.

The snapshot name has `ums-` prefixed to it, followed by an encoded UUID. This is required because the HPE StoreVirtual has name length and character limitations.

Consistency groups

Support for Consistency group is available for Liberty to Ocata release. Prior to consistency groups, every operation in Cinder happened at the volume level. Grouping like volumes allows improved data protection, paves the way to maintaining consistency of data across multiple different volumes, and allows operations to be performed on groups of volumes. The fundamental supported operations include creating a consistency group, deleting a consistency group (and all volumes inside of it), adding volumes to a consistency group, removing volumes from a consistency group, and snapshotting a consistency group.

Note
The LeftHand Cinder driver does not currently support creating a consistency group from a source consistency group or a source consistency group snapshot.

Consistency group CLI support is defaulted to **off**. To access the consistency group related CLI commands, modify `/etc/cinder/policy.conf by removing group: nobody` from the following lines:

```
"consistencygroup:create" : "",

"consistencygroup:delete": "",

"consistencygroup:update": "",

"consistencygroup:get": "",

"consistencygroup:get_all": "",

"consistencygroup:create_cgsnapshot" : "",

"consistencygroup:delete_cgsnapshot": "",

"consistencygroup:get_cgsnapshot": "",

"consistencygroup:get_all_cgsnapshots": "",
```

Creating a consistency group

Once the `policy.conf` file is correctly modified, you can create a consistency group either in the Horizon UI or using the command line, as shown:

```
$cinder consisgroup-create [--name <name>] [--description <description>] <volume-type>
```

Where `<volume-type>` is the OpenStack/Cinder volume type name and `--name` and `--description` are optional:

```
$cinder consisgroup-create --name "MyCG" --description "lefthand cg" lefthand
```

To view the newly created consistency group, use the command:

```
$cinder consisgroup-list

| 831b2099-d5ba-4b92-a097-8c08f9a8404f | available | MyCG |
```

Deleting a consistency group

An empty consistency group can be deleted either in the Horizon UI or using the command line, as shown:

```
$cinder consisgroup-delete <consisgroup-id>
```

Where `<consisgroup-id>` represents to consistency group ID:

```
$cinder consisgroup-delete 831b2099-d5ba-4b92-a097-8c08f9a8404f
```

If the group has volumes, add the force flag (Note: This will fully delete all volumes in the group.):

```
$cinder consisgroup-delete <consisgroup-id> --force
```

Where `<consisgroup-id>` represents to consistency group ID:

```
$cinder consisgroup-delete 831b2099-d5ba-4b92-a097-8c08f9a8404f --force
```

Adding volumes

Add volumes to the consistency group by either using the Horizon UI or using the command line, as shown:

```
$cinder consisgroup-update <consisgroup-id> --add-volumes <uuid1,uuid2,......>
```

Where `<consisgroup-id>` represents the consistency group ID and `<uuid1,uuid2,......>` is a comma-separated list of OpenStack/Cinder volume IDs:

```
$cinder consisgroup-update 831b2099-d5ba-4b92-a097-8c08f9a8404f --add-volumes 87ac88d4-e360-4bdb-b888-
2208fbe282dd,9ce09c09-0a20-4bcd-bd1a-0eaa95dbe0cd,a4563466-61d4-4018-b586-f07f84c4010c
```

Removing volumes

Remove volumes from the consistency group by either using the Horizon UI or using the command line, as shown:

```
$cinder consisgroup-update <consisgroup-id> --remove-volumes <uuid1,uuid2,......>
```

Where `<consisgroup-id>` represents the consistency group ID and `<uuid1,uuid2,......>` is a comma-separated list of OpenStack/Cinder volume IDs:

```
$cinder consisgroup-update 831b2099-d5ba-4b92-a097-8c08f9a8404f --remove-volumes 87ac88d4-e360-4bdb-
b888-2208fbe282dd
```

Note

This does not delete the volume; it only removes it from a group.

Creating a cgsnapshot

To snapshot a consistency group, use the command:

```
$cinder cgsnapshot-create [--name <name>] [--description <description>] <consisgroup-id>
```

Where `<consisgroup-id>` represents the consistency group ID and `--name` and `--description` are optional:

```
$cinder cgsnapshot-create --name “MyCGSnap” --description “Snapshot of MyCg”
831b2099-d5ba-4b92-a097- 8c08f9a8404f
```

To view the newly created `cgsnapshot`, use the command:

```
$cinder cgsnapshot-list
```

```
| 70c266bb-8255-4f2b-83cf-87f79d54dfb4 | creating | MyCGSnap |
```

Deleting a cgsnapshot

To delete a `cgsnapshot`, use the command:

```
$cinder cgsnapshot-delete <cgsnapshot-id>
```

Where `<cgsnapshot-id>` is the consistency group snapshot ID:

```
$cinder cgsnapshot-delete 70c266bb-8255-4f2b-83cf-87f79d54dfb4
```

Generic volume groups

Consistency group feature of cinder is replaced with Generic volume groups in Pike release. Consistency groups in cinder only support consistent group snapshot. It cannot be extended easily to serve other purposes. A tenant may want to put volumes used in the same application together in a group so that it is easier to manage them together, and this group of volumes may or may not support consistent group snapshot. Generic volume group is introduced to solve this problem. By decoupling the tight relationship between the group construct and the consistency concept, generic volume groups can be extended to support other features in the future.

Creating a generic volume group

You can create a generic volume group either in the Horizon UI or using the command line, as shown:

```
$cinder --os-volume-api-version 3.13 group-create [--name <name>] [--description <description>]
GROUP_TYPE VOLUME_TYPES
```

Where `GROUP_TYPE` is the name or UUID of a group type, `VOLUME_TYPES` is the OpenStack/Cinder volume type name and --name and --description are optional:

```
$cinder --os-volume-api-version 3.13 group-create --name MyGVG --description “VSA gvg” GrpType
VSA-1,VSA-2
```

To view the newly created group:

```
$cinder --os-volume-api-version 3.13 group-list
```

```
| 831b2099-d5ba-4b92-a097-8c08f9a8404f | available | MyGVG |
```

Deleting a generic volume group

An empty generic volume group can be deleted either in the Horizon UI or using the command line, as shown:

```
$cinder --os-volume-api-version 3.13 group-delete <group>
```

Where `<group>` represents to generic volume group UUID/name:

```
$cinder --os-volume-api-version 3.13 group-delete MyGVG
```

If the group has volumes, the `--delete-volumes` flag can be added.

Note

This will fully delete all volumes in the group.

```
$cinder --os-volume-api-version 3.13 group-delete --delete-volumes MyGVG
```

Create a volume and add it to a group

To create a volume and add it to a group, issue the following command:

```
$cinder --os-volume-api-version 3.13 create --volume-type VOLUME_TYPE --group-id GROUP_ID SIZE
```

Note

When creating a volume and adding it to a group, the parameters `VOLUME_TYPE` and `GROUP_ID` must be provided. This is because a group can support more than one volume type.

Adding and Removing volumes

Add volumes to the group or remove volume from the group by either using the Horizon UI or using the command line, as shown:

```
$cinder --os-volume-api-version 3.13 group-update --add-volumes <uuid1,uuid2,......> --remove-volumes
<uuid3,uuid4,......> <group>
```

Where `<group>` represents the group UUID/name and `<uuid1,uuid2,......>` is a comma-separated list of OpenStack/Cinder volume IDs:

```
$cinder  --os-volume-api-version 3.13 group-update --add-volumes 87ac88d4-e360-4bdb-b888-
2208fbe282dd,9ce09c09-0a20-4bcd-bd1a-0eaa95dbe0cd --remove-volumes a4563466-61d4-4018-b586-
f07f84c4010c MyGVG
```

Note

Removing volume does not delete the volume; it only removes it from the generic volume group.

Creating a snapshot for a group

Snapshotting a generic volume group can be accomplished with the following command:

```
$cinder --os-volume-api-version 3.14 group-snapshot-create [--name <name>] [--description <description>]

<group>
```

Where `<group>` represents the group UUID/name and `--name` and `--description` are optional:

```
$cinder --os-volume-api-version 3.14 group-snapshot-create --name “MyGVGSnap” --description “Snapshot
of MyGVG” MyGVG
```

To view the newly created group snapshot:

```
$cinder --os-volume-api-version 3.14 group-snapshot-list

| 70c266bb-8255-4f2b-83cf-87f79d54dfb4 | creating | MyGVGSnap |
```

Deleting group snapshot

To delete a group snapshot:

```
$cinder --os-volume-api-version 3.14 group-snapshot-delete <group_snapshot>
```

Where `<group_snapshot>` is the group snapshot ID:

```
$cinder --os-volume-api-version 3.14 group-snapshot-delete MyGVGSnap
```

Create a group from a group snapshot

To create a group from a group snapshot:

```
$cinder --os-volume-api-version 3.14 group-create-from-src --group-snapshot <group_snapshot> --name <name>
```

Where `<group_snapshot>` is the group snapshot UUID/name:

```
$cinder --os-volume-api-version 3.14 group-create-from-src --group-snapshot MyGVGSnap --name MyGVG2
```

Cloning a group

To clone a group, issue this command:

```
$cinder --os-volume-api-version 3.14 group-create-from-src --source-group <source_group> --name <name>
```

Where `<source_group>` is the group we want to clone.

The newly created generic volume group can be treated as a new, completely separate group with no ties to its parent group or group snapshot. It can be deleted at any time, as can its source group.

For more details on usage of commands, refer this link: docs.openstack.org/cinder/latest/admin/blockstorage-groups.html.

Multiple backend requirements

The `hpelefthand_clustername` is required and is case-sensitive.

Backend assisted volume migration

Backend assisted volume migration is another new capability when the driver is configured to execute in standard mode. When a volume is migrated using backend assisted volume migration, both source and destination clusters must be in the same HPE StoreVirtual management group. The HPE StoreVirtual Storage array uses native LeftHand APIs to migrate the volume. The volume cannot be attached or have snapshots to migrate.

Volume retype

Volume retype was available in the Icehouse release. The retype only works if the volume is on the same HPE StoreVirtual Storage array. This allows the volume retype from a “silver” volume type to a “gold” volume type. The HPE StoreVirtual OpenStack drivers modify the volume’s data protection level, provisioning type, and Adaptive Optimization settings, as needed, to make the volume behave appropriately for the new volume type. Use caution when using the optional `–migration-policy on-demand` as this falls back to copying the entire volume (using `dd` over the network) to the Cinder node and then to the destination HPE StoreVirtual Storage array. The Cinder node also must have enough space available to store the entire volume during the migration. We recommend that you use the default `–migration-policy never` when retype is used.

Note
The `volume_backend_name in cinder.conf` must be the same between the source and destination volume types when `-migration-policy is set to never`, which is the default and recommend retype method.

Volume replication

Cinder’s replication is host-driven, meaning all `volume_type` replicated volumes created on a host are automatically replicated to the configured secondary target(s). LeftHand only supports periodic replication, meaning the volumes are only synchronized every X amount of seconds. You can configure the sync period via `extra_specs` as shown in the Volume type extra_specs section. To begin replicating, `cinder.conf` entries need to be created. You also must create a replicated volume type that points to the newly configured backend.

As shown here, the only additional field required to enable replication on a backend is to provide a minimum of one `replication_device` entry. If multiple replication targets are desired, adding additional `replication_device` entries for each target is satisfactory. The following fields are expected for each `replication_device`:

- `backend_id` must be specified but can be any value. It is designed to be a unique identifier for the configured array.
- `hpelefthand_api_url` should be the API URL for the secondary system.
- `hpelefthand_username` is the username for the secondary system.
- `hpelefthand_password` is the password for the secondary system.
- `hpelefthand_clustername` is the cluster for the secondary system.

```
[lefthandrep]
hpelefthand_api_url = https://10.50.3.50:8081/lhos
hpelefthand_username = <username>
hpelefthand_password = <password>
hpelefthand_clustername = vsa-12-5-mgmt1-vip
volume_backend_name = lefthandrep
volume_driver = cinder.volume.drivers.hpe.hpe_lefthand_iscsi.HPELeftHandISCSIDriver
image_volume_cache_enabled = True
replication_device = backend_id:lh-id,
                      hpelefthand_api_url:https://10.50.3.54:8081/lhos,
                      hpelefthand_username:USERNAME,
                      hpelefthand_password:PASSWORD,
                      hpelefthand_clustername:vsa-12-5-mgmt2-vip
```

Figure 6. Sample cinder.conf file showing replication devices

Volume type extra_specs

Required extra_spec values:

- replication_enabled = <is> True

Warning: Use the syntax above exactly, including case, or the volume replication will not work.

Optional extra_spec values:

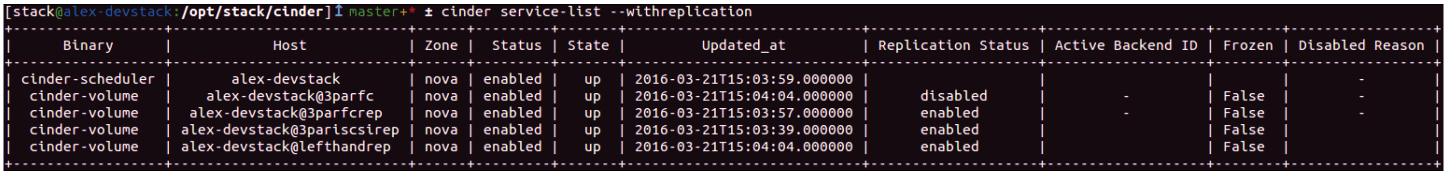
- replication:sync_period = 1800 (in seconds, specifies the time interval for synchronization. Defaults to 1800)
- replication:retention_count = 5 (how many snapshots will be saved on the primary array. Defaults to 5)
- replication:remote_retention_count = 5 (how many snapshots will be saved on the secondary arrays. Defaults to 5)

After modifying cinder.conf and creating a volume type that supports LeftHand replication, any volumes created under that type are automatically replicated.

Replication status of a host

Since Cinder's implementation of replication is host-based, you can view the replication state of the host to see if it is enabled, disabled, failed-over, or in error state as shown below with the --withreplication flag.

```
$ cinder service-list --withreplication
```



```
[stack@alex-devstack:/opt/stack/cinder]↥ master+* ± cinder service-list --withreplication
+------------------+----------------------------+------+---------+-------+----------------------------+--------------------+-------------------+--------+-----------------+
|      Binary      |            Host            | Zone |  Status | State |         Updated_at         | Replication Status | Active Backend ID | Frozen | Disabled Reason |
+------------------+----------------------------+------+---------+-------+----------------------------+--------------------+-------------------+--------+-----------------+
| cinder-scheduler |       alex-devstack        | nova | enabled |   up  | 2016-03-21T15:03:59.000000 |                    |                   |        |        -        |
|  cinder-volume   |    alex-devstack@3parfc    | nova | enabled |   up  | 2016-03-21T15:04:04.000000 |      disabled      |         -         | False  |        -        |
|  cinder-volume   |  alex-devstack@3parfcrep   | nova | enabled |   up  | 2016-03-21T15:03:57.000000 |      enabled       |         -         | False  |        -        |
|  cinder-volume   | alex-devstack@3pariscsirep | nova | enabled |   up  | 2016-03-21T15:03:39.000000 |      enabled       |                   | False  |                 |
|  cinder-volume   | alex-devstack@lefthandrep  | nova | enabled |   up  | 2016-03-21T15:04:04.000000 |      enabled       |                   | False  |                 |
+------------------+----------------------------+------+---------+-------+----------------------------+--------------------+-------------------+--------+-----------------+
```

Figure 7. Sample output from the "$cinder service-list—withreplication" command

This shows the alex-devstack@lefthandrep host has a replication status of "enabled."

Creating a replicated volume

This example assumes a volume type of `lefthandrep` was created and the proper `extra_specs` were added in order to support replication.

```
$ cinder create --volume-type lefthandrep 10
```

```
|              id              |    808482df-3bae-47d1-8c66-6a41c6cdf796    |
```

...

```
$ cinder show 808482df-3bae-47d1-8c66-6a41c6cdf796
```

...

```
|      replication_status      |                 enabled                    |
```

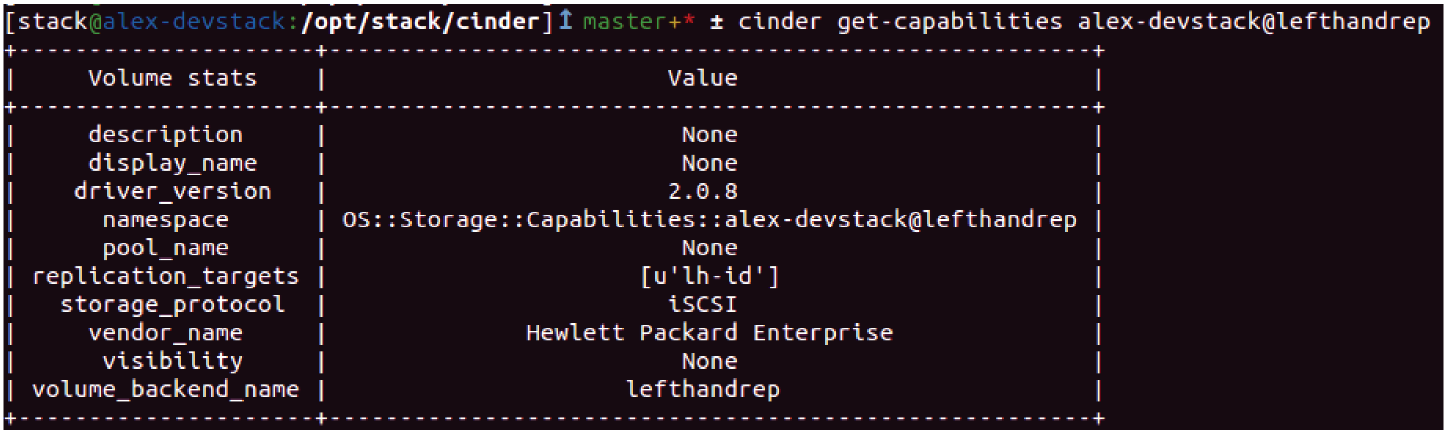
...

The `cinder show` command on a volume allows you to see if a volume is being replicated. The default synchronization period is 1800 seconds. If `extra_spec replication:sync_period` was specified, the sync period is overwritten with its value.

Listing valid replication targets

Replication targets are reported as a host capability under the `replication_targets` field. These targets are helpful when an administrator wants to see the number of targets they are replicating to, or when they need to specify where to fail the host over to.

```
$ cinder get-capabilities alex-devstack@lefthandrep
```



```
[stack@alex-devstack:/opt/stack/cinder]↥ master+* ± cinder get-capabilities alex-devstack@lefthandrep
+---------------------+-----------------------------------------------------+
|     Volume stats    |                        Value                        |
+---------------------+-----------------------------------------------------+
|     description     |                         None                        |
|     display_name    |                         None                        |
|    driver_version   |                        2.0.8                        |
|      namespace      | OS::Storage::Capabilities::alex-devstack@lefthandrep |
|      pool_name      |                         None                        |
| replication_targets |                      [u'lh-id']                     |
|   storage_protocol  |                        iSCSI                        |
|     vendor_name     |             Hewlett Packard Enterprise              |
|      visibility     |                         None                        |
| volume_backend_name |                     lefthandrep                     |
+---------------------+-----------------------------------------------------+
```

Figure 8. Sample output from the "$cinder get-capabilities <host>" command

Failing over a host

When a host is failed over through Cinder's `failover-host` command, all replicated volumes on the host are individually transitioned to the specified secondary target. If the primary system is entirely offline, the failover still works as intended. If there are any non-replicated volumes on the host at the time failover is requested, they will not be available on the secondary system. These volumes are forced into error state, but will once again be available upon failing back.

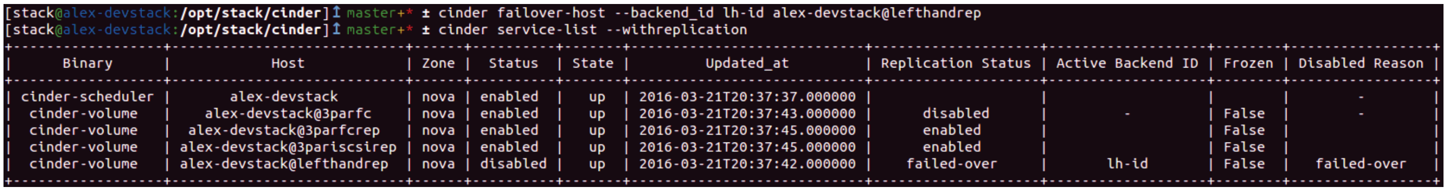
```
$ cinder failover-host -backend_id <lefthand-target> <primary@host>
```

Where `<lefthand-target>` is a value obtained from the `replication_targets` field of Cinder's `get-capabilities` command and `<primary@host>` is the host that is intended to be failed over.

```
$ cinder service-list -withreplication
```

Shows the new status of the replicated host.

Figure 9 shows an example of failing over a host using the `get-capabilities` command.



```
[stack@alex-devstack:/opt/stack/cinder]↥ master+* ± cinder failover-host --backend_id lh-id alex-devstack@lefthandrep
[stack@alex-devstack:/opt/stack/cinder]↥ master+* ± cinder service-list --withreplication
+------------------+----------------------------+------+----------+-------+----------------------------+--------------------+-------------------+--------+-----------------+
|      Binary      |            Host            | Zone |  Status  | State |         Updated_at         | Replication Status | Active Backend ID | Frozen | Disabled Reason |
+------------------+----------------------------+------+----------+-------+----------------------------+--------------------+-------------------+--------+-----------------+
| cinder-scheduler |       alex-devstack        | nova | enabled  |   up  | 2016-03-21T20:37:37.000000 |                    |                   |        |        -        |
|   cinder-volume  |    alex-devstack@3parfc    | nova | enabled  |   up  | 2016-03-21T20:37:43.000000 |      disabled      |         -         | False  |        -        |
|   cinder-volume  |  alex-devstack@3parfcrep   | nova | enabled  |   up  | 2016-03-21T20:37:45.000000 |      enabled       |                   | False  |                 |
|   cinder-volume  | alex-devstack@3pariscsirep | nova | enabled  |   up  | 2016-03-21T20:37:45.000000 |      enabled       |                   | False  |                 |
|   cinder-volume  | alex-devstack@lefthandrep  | nova | disabled |   up  | 2016-03-21T20:37:42.000000 |    failed-over     |       lh-id       | False  |   failed-over   |
+------------------+----------------------------+------+----------+-------+----------------------------+--------------------+-------------------+--------+-----------------+
```

Figure 9. Sample output showing a host in the "failed-over" state

Note

If StoreVirtual backend is in failed over state, new volume creation on secondary StoreVirtual system is not allowed. Trying to create a new volume will end it in error state as Remote Copy association is either broken or not available.

Failing back a host

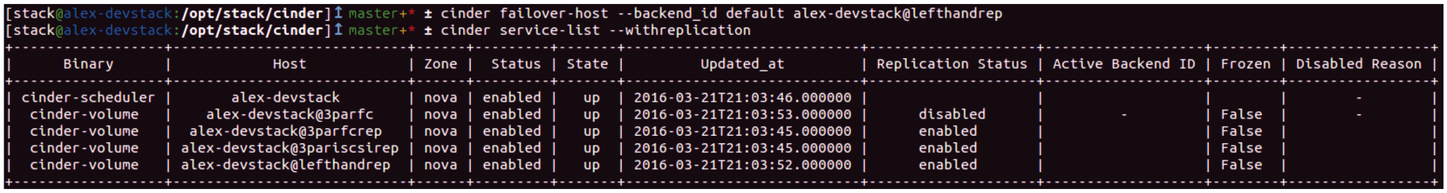
All replicated volumes need to be returned to normal. To return a host's replication status back to normal, steps need to be performed on the LeftHand backends. For details, see the Failing back a LeftHand backend section in the Appendix. After the host's replication status is back to normal, the failback command can be issued.

Note

If these steps are not performed on the backend prior to failing back, the command will not be successful. The host will remain in a failed over state.

Using a `--backend_id` value of "default" triggers the failback command as follows:

```
$ cinder failover-host -backend_id default <primary@host>
```



```
[stack@alex-devstack:/opt/stack/cinder]↥ master+* ± cinder failover-host --backend_id default alex-devstack@lefthandrep
[stack@alex-devstack:/opt/stack/cinder]↥ master+* ± cinder service-list --withreplication
+------------------+----------------------------+------+---------+-------+----------------------------+--------------------+-------------------+--------+-----------------+
|      Binary      |            Host            | Zone |  Status | State |         Updated_at         | Replication Status | Active Backend ID | Frozen | Disabled Reason |
+------------------+----------------------------+------+---------+-------+----------------------------+--------------------+-------------------+--------+-----------------+
| cinder-scheduler |       alex-devstack        | nova | enabled |   up  | 2016-03-21T21:03:46.000000 |                    |                   |        |        -        |
|   cinder-volume  |    alex-devstack@3parfc    | nova | enabled |   up  | 2016-03-21T21:03:53.000000 |      disabled      |         -         | False  |        -        |
|   cinder-volume  |  alex-devstack@3parfcrep   | nova | enabled |   up  | 2016-03-21T21:03:45.000000 |      enabled       |                   | False  |                 |
|   cinder-volume  | alex-devstack@3pariscsirep | nova | enabled |   up  | 2016-03-21T21:03:45.000000 |      enabled       |                   | False  |                 |
|   cinder-volume  | alex-devstack@lefthandrep  | nova | enabled |   up  | 2016-03-21T21:03:52.000000 |      enabled       |                   | False  |                 |
+------------------+----------------------------+------+---------+-------+----------------------------+--------------------+-------------------+--------+-----------------+
```

Figure 10. Sample output from the "$cinder failover-host" and "$cinder service-list—withreplication" command

If all the proper backend steps are performed, the host's `replication_status` returns to an enabled state, and can be failed back over at a later date. The volumes will resume replication and all data will be duplicated to the secondary targets.

Support for Containerized Stateful Services

The HPE LeftHand drivers are supported by the `ClusterHQ Flocker` open source technology. Details on the setup and usage of Flocker can be found at clusterhq.com/docs/.

Suppress requests library SSL certificate warnings

The HPE LeftHand client uses the Python request library to talk to the array, the following warning message is logged to the Cinder's volume log on every client request, if the SSL certificate is invalid or out of date.

```
WARNING py.warnings /usr/local/lib/python2.7/dist-
packages/requests/packages/urllib3/connectionpool.py:791: InsecureRequestWarning: Unverified HTTPS
request is being made. Adding certificate verification is strongly advised. See:
urllib3.readthedocs.org/en/latest/security.html InsecureRequestWarning.
```

Cinder has added the `suppress_requests_ssl_warnings` option to the `cinder.conf` configuration file to suppress these warning messages. The default value is false, but the option can be set to true to suppress the above warnings. This is a valid warning and probably should not be disabled in a production environment.

Known Limitations

1. Volume encryption is not supported for HPE StoreVirtual Block Storage Driver.
2. Cinder Cascade deletion operation would not work where snapshot1 is created from volume1 and again volume2 is created from snapshot1.
3. Volume Retype with Replication is not supported. Changing the volume-type of a volume to a volume-type that includes ‘`replication_enabled = <is> True`’ (and didn’t have it before) would not result in adding a secondary copy to a volume. Changing the volume-type of a volume to a volume-type that no longer includes ‘`replication_enabled = <is> True`’ would not result in removing the secondary copy.

Summary

Hewlett Packard Enterprise is a Platinum member of The OpenStack Foundation. HPE has integrated OpenStack open source cloud platform technology into its enterprise solutions to enable customers and partners to build enterprise-grade private, public, and hybrid clouds.

The Mitaka release continues HPE’s contributions to the Cinder project, enhancing core Cinder capabilities as well as extending HPE StoreVirtual Block Storage Driver. The focus continues to be on adding enterprise functionality such as volume replication, image caching, and managing of snapshots, etc. The HPE StoreVirtual Block Storage Drivers support the OpenStack technology across the iSCSI protocol.

Appendix

Failing back a LeftHand backend

To properly resynchronize a LeftHand volume, the data written to the secondary array while the primary was down needs to be written back to the primary. Also, the original schedule needs to be resumed in order to allow primary-to-secondary replication once again.

Perform the following steps for each volume that needs resynchronization:

1. The original schedule is Paused.
2. The original volume is set to “Remote”.
3. Server connections to original volume are set to “Read Only” access.
4. Server connections to secondary volume are set to “Read Only” access.
5. A new schedule is created to copy from secondary volume to primary volume:
   a. This could take a while.
6. The new schedule is paused.
7. The original volume is set to “Primary”:
   a. Server connections to the original volume are set to “Read/Write” access.
8. Validate that the original volume is fully functional.
9. The secondary volume is set to “Remote”.
10. The original schedule is resumed.

Once every volume on the Cinder host has been resynchronized and the original remote snapshot schedule has been resumed, the `failback` command can be successfully issued.

For more information

HPE Cloud

HPE Helion

HPE Helion CloudSystem

OpenStack

OpenStack website

OpenStack Documentation

OpenStack Administrator Guide

HPE StoreVirtual array

HPE StoreVirtual Storage Family

HPE StoreVirtual iSCSI Driver

Learn more at
hpe.com/helion

Sign up for updates

4AA6-5362ENW, October 2017, Rev. 3