Pure Storage FlashArray OpenStack® Cinder Volume Driver 6.0.0 Setup Guide

purestorage.com

Contents

Chapter 1. Introduction

Overview

OpenStack® is a set of related open source tools that support scalable cloud computing for both private and public clouds. As a cloud operating system, OpenStack controls a data center's pools of networking, storage, and compute resources and provides a web-based interface to manage these resources. OpenStack also provides a set of REST APIs accessible to command-line and other clients on remote machines.

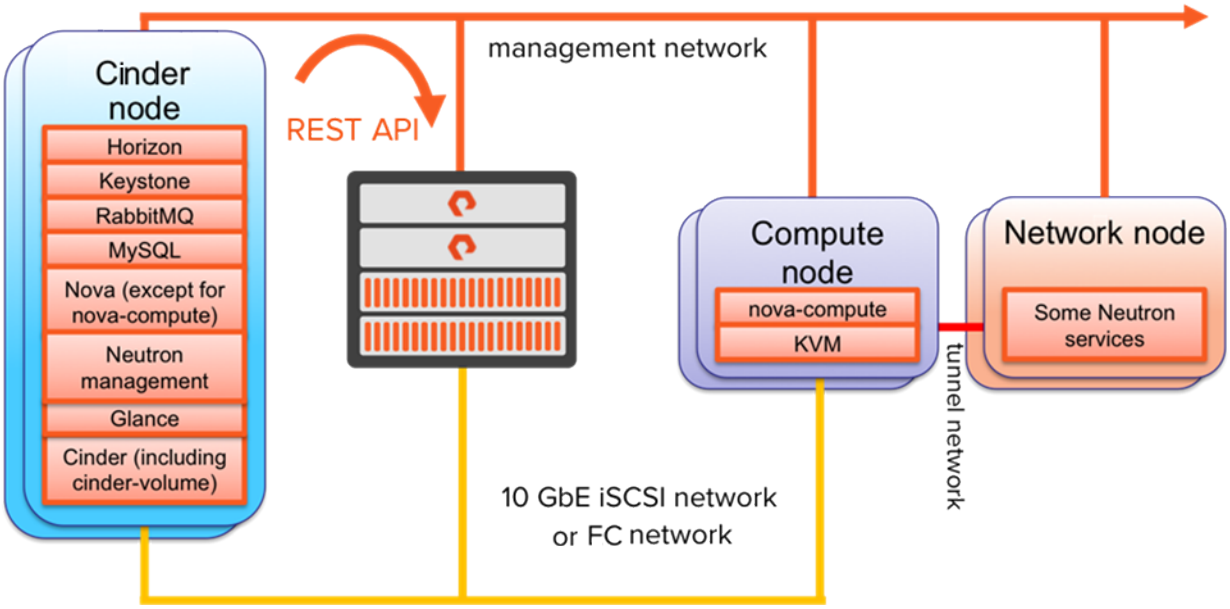
Using these interfaces, users directly provision virtual server instances, storage, and other resources, without requiring administrator involvement. As part of its management of cluster resources, OpenStack automatically selects physical resources to host users' VM instances.

The Cinder project is OpenStack's persistent block storage service. Cinder provides volume storage commands to manage volumes and snapshots of volumes (for example, to create and delete volumes, to create volumes from images and from snapshots, to attach volumes to instances, and to create and delete snapshots).

The Pure Storage Volume Driver allows Cinder to use a Pure Storage FlashArray as a block storage resource in the OpenStack cluster. Users' VM instances can be backed up entirely on solid-state FlashArray storage. The volume driver communicates with the FlashArray through its management ports and either its iSCSI ports or its Fibre Channel ports. The volume driver supports both Fibre Channel and iSCSI, but only one type can be used per backend definition.

The following image shows a block diagram of the driver architecture.

Figure 1.1: Architecture Diagram



Cinder services use Purity's REST API to request volume management actions from our volume driver. The Cinder services shown in the node on the left could be located in multiple nodes and in controller or non-controller nodes.

Pure Storage is a corporate sponsor of the OpenStack Foundation.

Versions

This release of the volume driver supports the OpenStack Ocata release. The following are the driver versions:

- Fibre Channel (cinder.volume.drivers.pure.PureFCDriver): 4.0.0
- iSCSI (cinder.volume.drivers.pure.PureISCSIDriver): 6.0.0

Audience

This guide describes how to set up and configure the volume driver on an OpenStack cluster. The intended audiences for this guide include the following:

- OpenStack administrators, who configure and use OpenStack Cinder
- Pure Storage system administrators, who configure and use Purity
- Pure Storage partners and solutions engineers, who support the Pure Storage FlashArray Volume Driver for OpenStack Cinder

This guide assumes familiarity with storage and networking concepts, OpenStack administration, Cinder administration, and the Purity system. (See *Resources* for information on FlashArrays, the Purity Operating Environment, and OpenStack integration.)

Process Overview

The following list presents a high-level overview of the steps to install and configure the Pure Storage Volume Driver. Refer to subsequent pages for detailed instructions on installation and configuration steps.

1. **Read this entire guide and the release notes before you begin**
2. **Review the prerequisites**
   System requirements and administrator privilege requirements are described in the *Prerequisite Tasks* chapter.
3. **Install and configure the `cinder-api` and `cinder-scheduler` services**
   Install and configure the **`cinder-api`** and **`cinder-scheduler`** services.
4. **Install and configure multipathing for the nova service**
   Install and configure the multipathing package on OpenStack compute nodes that will connect to the FlashArray.
5. **Install the Pure Storage SDK and packages required by the Pure Storage Volume Driver**

Install the Pure Storage SDK. Install the Cinder volume service, QEMU image handling, and multipathing packages.

Also install packages to support iSCSI or your Fibre Channel host bus adapters (HBAs).

6. **Configure OpenStack**
   Edit the OpenStack Cinder **`cinder.conf`** file to specify the Pure Storage Volume Driver and to enable multipathing in Cinder.

7. **Verify the configuration**
   In OpenStack, configure a volume on a Pure Storage FlashArray host and write to that volume from OpenStack.

   In Purity, list host information to confirm the multipathing configuration.

8. **Communicate restrictions to your FlashArray administrators**
   When you review the *Prerequisite Tasks* chapter, pay special attention to the list of actions that are disallowed in Purity for Cinder volumes and snapshots. Communicate these restrictions to your FlashArray administrators.

Chapter 2. Prerequisite Tasks

Before you install or configure the Pure Storage Volume Driver, review the release notes, verify that your system meets the minimum requirements, and verify that you have the appropriate privileges to perform the installation and configuration steps.

Verify System Requirements

OpenStack

- ☐ Verify that you are running an OpenStack cluster on the Ocata release.
- ☐ Verify that your OpenStack cluster uses a KVM or QEMU hypervisor.
- ☐ For OpenStack compute nodes that will connect with the FlashArray, verify that these nodes run the **`libvirt`** compute driver.
- ☐ Be aware that the following changes are required on your OpenStack cluster (installation and configuration steps are included in this guide):
  - ☐ Installation of the following packages on select OpenStack cluster nodes (if not already installed): **`cinder-api`**, **`cinder-scheduler`**, the Cinder volume service, QEMU image handling, and multipathing.
  - ☐ If you use the iSCSI protocol, also download and install the iSCSI package.
  - ☐ If you use the Fibre Channel protocol, also download and install any packages that are required by your Fibre Channel HBAs.

Purity, FlashArray, Connectivity, and Administrators

- ☐ Verify that you are running Purity version 4.0.20 (or later) on your FlashArray.
- ☐ Verify that you are running Purity REST API version 1.2 (or later) on your FlashArray.
- ☐ Verify that you are running Purity Python REST Client version 1.6 (or later).
  To verify your Python REST Client version, run either of the following:
  - ☐ Command:

    ```
    pip freeze | grep purestorage
    ```

    Example output: **`purestorage==1.6.1`**
  - ☐ Commands:

    ```
    python
    >>> import purestorage
    >>> print purestorage.VERSION
    ```

Example output: **`1.6.1`**

To install the Python REST Client, follow the steps in *Install the Pure Storage Python SDK*.

- ☐ To use the Cinder import feature, verify that you are running Purity REST API version 1.4 (or later) on your FlashArray. REST API version 1.4 is installed with Purity 4.1 and later releases.
- ☐ Verify that your FlashArray supports iSCSI or Fibre Channel ports.
- ☐ Verify that your FlashArray's Management ports, and iSCSI or Fibre Channel ports, are available on the network:
  - ☐ All OpenStack nodes where the Pure Storage Volume Driver will run need to access the FlashArray's management interface and also the FlashArray's iSCSI or Fibre Channel ports. From each of these nodes, verify that you can ping the FlashArray's management port and iSCSI or Fibre Channel ports.
  - ☐ OpenStack compute nodes hosting instances that connect to the array need to reach the iSCSI or Fibre Channel ports on the array. From each of these nodes, verify that you can ping the FlashArray's iSCSI or Fibre Channel ports.
- ☐ Verify that the OpenStack node where the Pure Storage Volume Driver will be installed connects to the FlashArray's iSCSI ports with 10GbE iSCSI networking or to the FlashArray's Fibre Channel ports with Fibre Channel networking.
- ☐ For production environments, if you choose to create a dedicated FlashArray administrator account to manage your Cinder volumes, verify that you can create an Active Directory user account in the array administrators security group.

Verify User Privileges for System Configuration

- ☐ Purity configuration steps require privileges associated with the array administrator role.
- ☐ OpenStack configuration steps require the cluster administrator role.

Plan Your OpenStack Node Usage

For evaluation purposes or in a simple OpenStack cluster configuration, you can install the volume driver and Cinder packages (**`cinder-api`**, **`cinder-scheduler`**, and **`cinder-volume`** or **`openstack-cinder`**) on one OpenStack controller node. This scenario simplifies some setup and configuration steps.

In other scenarios, decide on which OpenStack node to install the volume driver. This decision depends in part on load balancing and network issues. For example, the OpenStack node where the volume driver is installed should connect to the FlashArray's iSCSI-capable ports with 10GbE iSCSI networking or to the FlashArray's Fibre Channel ports with Fibre Channel networking. These connections are required to support the following driver actions:

- Copying an image to a volume
- Copying a volume to an image

- Migrating a volume

Notes and restrictions:

- The available disk space on the node where the volume driver is running must be greater than the size of any image file used to create a volume.
- Configuration steps are required on OpenStack nodes that run the volume driver and that connect to your FlashArray. (See *Configure the OpenStack Cinder Volume Service*).

Install the `cinder-api` and `cinder-scheduler` Packages

Install and configure the **cinder-api** and **cinder-scheduler** services on your OpenStack cluster.

Notes:

- The configuration of these services can be involved. Allow sufficient time to complete this step.
- The Pure Storage Volume Driver does not have any requirement on which node these services are installed.
- Later configuration steps ensure that the Cinder volume service instance that instantiates and manages the volume driver has access to the same database, RPC messaging, and authorization settings as the **cinder-api** and **cinder-scheduler** services.

(Optional) Create a Dedicated Purity User Account

A Purity array administrator account that is dedicated to Cinder is not a strict requirement but is strongly recommended for production environments. A dedicated array administrator account has these benefits:

- The account has the correct permission level for the volume driver. The volume driver cannot, for example, configure the array or change permissions of user accounts.
- The API token for the dedicated user is used by only the volume driver.
- Actions performed by the volume driver are easier to distinguish from other management actions in the audit trail.

See *Purity Administrator Accounts in Active Directory* for information on creating accounts for Purity.

Review Disallowed Actions with Your FlashArray Administrators

Your Cinder volumes should be managed only through OpenStack, not through Purity.

Purity administrators must not modify or delete any volumes or snapshots that Cinder manages. If objects that Cinder manages are changed through Purity, this interferes with the ability of Cinder to manage FlashArray objects. (Objects that are managed by the Cinder service have **-cinder** to their name.)

For Cinder volumes, use Purity only for read-only actions such as those to display, list, or monitor.

Disallowed Actions

The following actions are disallowed in Purity for a host with targets on the OpenStack cluster:

- Create the host
- Delete the host
- Connect a volume to the host
- Disconnect a volume from the host
- Rename the host

The following actions are disallowed in Purity for volumes and snapshots that Cinder manages:

- Connect a volume to a host or to a host group
- Disconnect a volume from a host
- Destroy a volume
- Rename a volume
- Extend a volume
- Truncate a volume
- Destroy a snapshot
- Rename a snapshot

The following actions are also disallowed in Purity for a host to which Cinder has connected a volume:

- Delete the host
- Set a host username or password for the host
- Set a target username or password for the host
- Add, remove, or change the IQN of the host
- Add any WWNs to the host

About Volumes and Snapshots Created in Purity

We do not recommend using Purity features to create a snapshot from a volume that Cinder manages. While this action does not interfere with the ability of Cinder to manage the original volume, Cinder does not automatically manage new snapshots that are created through Purity.

To import a volume or snapshot into the Cinder service, see *Import a Volume or Snapshot into Cinder*.

Use OpenStack and Cinder Interfaces

The disallowed actions rule applies both to the dedicated Cinder administrator account in Purity (if you use a dedicated account) and to other FlashArray administrators.

The dedicated Cinder administrator has array administrator permissions in Purity, but those permissions are intended for the Cinder volume service to access the FlashArray and should not be used directly in Purity. Instead practice the discipline of using only OpenStack and Cinder interfaces for actions that modify or delete volumes and snapshots that Cinder manages.

Chapter 3. Best Practices

This section contains our recommendations for setup, configuration, and ongoing practices.

Virtual IP Address for `san_ip`

When you specify the FlashArray in the file **`/etc/cinder/cinder.conf`**, the recommended practice is to set **`san_ip`** to a virtual IP address that is assigned to both controllers in the FlashArray, or to a DNS name that resolves to such a virtual IP address. This setting is required for both high availability and non-disruptive upgrades.

(For configuration steps, see *Edit the Cinder Volume Service Configuration File*.)

Multipathing

It is important that multipathing be enabled both in the nova configuration and in the configuration of the Cinder volume service that manages the FlashArray. By default, OpenStack does not use multipathing and therefore would use only one of the FlashArray ports.

Multipathing offers these benefits:

- Increased performance
- High availability
- Non-disruptive upgrades

We strongly recommend that you enable multipathing in both your nova configuration and in the Cinder volume service that controls your FlashArray. (Steps are described later in *Install and Configure Multipathing for the Nova Service*.)

Confirm Your Multipathing Configuration

After you have attached a volume to an instance, use a Purity **`purehost list`** command to confirm that multipathing is configured properly. See *Confirm Multipathing in Purity*.

Fibre Channel or iSCSI Best Practices

Please review and follow Pure Storage's best practices recommendations for either Fibre Channel or iSCSI:

- *Introduction to Fibre Channel and Zoning* [https://support.purestorage.com/Solutions/SAN/Configuration/Introduction_to_Fibre_Channel_and_Zoning]
  Configuring iSCSI on Linux for the Pure FlashArray [https://support.purestorage.com/Solutions/Operating_Systems/Linux/ Configuring_iSCSI_on_Linux_for_the_Pure_FlashArray]

Network Configuration

The following OpenStack nodes should be as similar as possible in network setup, especially in regard to network subnets:

- The nodes that the driver is installed on
- The compute nodes that host instances that will connect to volumes in the FlashArray

The Pure Storage Volume Driver selects a FlashArray iSCSI or Fibre Channel port that it can reach and always uses that port when first establishing a connection with the array. If the OpenStack compute nodes have access to the same SAN subnet as the node where the driver is running, then the compute nodes are able to connect to that port as well.

Confirm the following:

- From each OpenStack nodes where the volume driver will run, confirm that you can ping the array's management port and the array's iSCSI or Fibre Channel ports.
- From each of the OpenStack compute nodes that will connect to the FlashArray, confirm that you can ping the array's iSCSI or Fibre Channel ports.

Filter and Goodness Functions

We recommend that you add a custom filter function and, if needed, a goodness (weighing) function to determine volume placement, as described in *Backend Scheduling*.

These functions optimize volume placement in multi-backend environments and also have utility in single array environments, for instance to avoid exceeding array limits. Through a goodness function, the driver selects a backend based on the arrays' real-time status for user-defined properties such as capacity levels, IOPS, latency, numbers of volumes, hosts, snapshots, or protection groups, or other storage metrics. A goodness function selects the best backend in real time by assigning each backend a score from 0 to 100 based on storage metrics.

A filter function is also used to determine volume placement, but is simpler and returns only whether a backend satisfies the filter or not.

Scheduler Filter

If you do not define a different filter function, we recommend that you add the Cinder scheduler default filter function:

```
[DEFAULT]
scheduler_default_filters = AvailabilityZoneFilter,CapacityFilter,Capabiliti
esFilter,DriverFilter
[each-named-backend]
filter_function = "capabilities.total_volumes < 500"
```

Dedicated Purity User Account

We recommend that you create a dedicated Purity array administrator, with a descriptive name such as **`cinder`**, to administrator your Cinder objects in Purity. The benefits of a dedicated user account include:

- The account has the correct permission level (array administrator) for the volume driver.
- The API token for the dedicated user is used for only the volume driver.
- Audit trails for the volume driver are easy to distinguish from other management actions in the audit log.

Manage Your Cinder Objects *Only* through OpenStack

Manage your Cinder volumes and snapshots ***only*** through OpenStack and ***not*** through Purity. Do not allow Purity administrators (including the dedicated Cinder administrator, if any) to modify or delete Cinder volumes or snapshots. These actions in Purity interfere with the ability of the Cinder service to manage these objects on the FlashArray.

For Cinder volumes and snapshots, use Purity only for read-only actions such as those to display, list, or monitor.

See *Review Disallowed Actions with Your FlashArray Administrators* for a list of Purity actions that are not allowed with the volumes and snapshots that Cinder manages.

Educate FlashArray Administrators about Objects Managed by Cinder

It is very important that FlashArray administrators do not create, modify, or delete host objects that are managed by the Cinder volume service.

Objects that are managed by the Cinder service have a **`-cinder`** suffix in their name.

Create Multiple Volumes from an Image

The following command is recommended only for the first copy of a volume from an image:

```
cinder create --image-id ...
```

This command is not efficient to create additional volumes from the same image. See *Efficiently Create Multiple Volumes from an Image* for recommendations.

Review Deprecated Configuration Entries

See *Deprecated Configuration Items* for a summary of recently-deprecated OpenStack configurations, including the following:

- Volume driver configuration entries in the [DEFAULT] section
- Consistency groups

Chapter 4. Install and Configure Multipathing for the Nova Service

Multipathing is required for optimal performance, high availability, and non-disruptive upgrades.

Install the Multipathing Package on Compute Nodes

Install multipathing on OpenStack compute nodes that will connect with the FlashArray. On Red Hat Enterprise Linux or CentOS, also enable and start the multipathing service.

1. Run the appropriate commands for your operating system:
   - For Ubuntu:
     ```
     sudo apt-get install multipath-tools
     ```
   - For Red Hat Enterprise Linux or CentOS:
     a. `sudo yum install device-mapper-multipath`
     b. `sudo mpathconf --enable`
     c. `sudo service multipathd start`
2. Repeat step 1 on each OpenStack compute node that will connect to the FlashArray.

Enable Multipathing in the Nova Service

This configuration enables multipathing for the nova service and is required on each OpenStack compute node that will connect to the FlashArray.

To enable multipathing for the nova service:

1. Add the following entry to **`[libvirt]`** section of the file **`/etc/nova/nova.conf`**:
   ```
   iscsi_use_multipath = True
   ```
   Note: Do not leave an entry **`iscsi_use_multipath = False`** in the file.
2. Restart the nova service on the compute node:
   ```
   service nova-compute restart
   ```
3. Repeat this configuration on all OpenStack compute nodes that will host compute instances that volumes on the FlashArray are attached to.

Chapter 5. Install Packages and the Pure Storage Python SDK

The Pure Storage Python SDK and several packages are required on the OpenStack node where the Pure Storage Volume Driver will run.

Install the Pure Storage Python SDK

These steps install the Pure Storage Python SDK and the Pure Storage Python REST Client.

To install:

1. **`pip install requests`**
2. **`pip install purestorage`**

The following command updates the SDK and Python REST Client:

```
pip install -U purestorage
```

Note: The SDK and Python REST Client are part of PyPI [http://pypi.python.org/pypi/purestorage].

Install Packages

Install these packages on the OpenStack node where the Pure Storage Volume Driver will run.

Remove the **`open-iscsi`** or **`iscsi-initiator-utils`** packages if you will not use iSCSI. If you use Fibre Channel, add any packages required by your Fibre Channel HBAs.

1. Run the one of the following commands or set of commands:
   - For Ubuntu:
     ```
     sudo apt-get install cinder-volume open-iscsi qemu-utils \
     multipath-tools
     ```
   - For Red Hat Enterprise Linux or CentOS:
     a. `sudo yum install openstack-cinder iscsi-initiator-utils \ qemu-img device-mapper-multipath`
     b. `sudo mpathconf --enable`
     c. `sudo service multipathd start`
2. Repeat step 1 on each OpenStack node where the Pure Storage Volume Driver will run.

Background

The package names vary depending on your operating system's package manager. The following is a brief description of the packages:

The Cinder volume service
Includes the service that instantiates and manages the Pure Storage Volume Driver

- Ubuntu: **`cinder-volume`**
- Red Hat, CentOS: **`openstack-cinder`**

Fibre Channel
Confirm if any packages are required for your Fibre Channel HBAs

iSCSI
Required for iSCSI support

- Ubuntu: **`open-iscsi`**
- Red Hat, CentOS: **`iscsi-initiator-utils`**

QEMU image handling
Required for image-to-volume and volume-to-image copying

- Ubuntu: **`qemu-utils`**
- Red Hat, CentOS: **`qemu-img`**

Multipathing
Required for optimal performance, high availability, and non-disruptive upgrades

- For Ubuntu: **`multipath-tools`**
- For Red Hat, CentOS: **`device-mapper-multipath`**

Chapter 6. Configure the OpenStack Cinder Volume Service

The following configuration options are supported:

- *Required*
  - Copy the API authorization token from Purity
  - Register the Pure Storage FlashArray Volume Driver for OpenStack Cinder (either the Fibre Channel or iSCSI version)
  - Register the Pure Storage FlashArrays as backends
  - Enable multipathing
- *Strongly recommended*
  - Configure backend scheduling functions and parameters
    - Set an oversubscription ratio
    - Create a filter function
    - Create a goodness (weighing) function
  - Enable a Fibre Channel Zone Manager (if using Fibre Channel)
  - Enable Glance image caching
    - Enable TRIM, discard, and UNMAP commands
    - Create a tenant project and user
- *Optional*
  - Verify HTTPS requests with an SSL certificate
  - Avoid "Unverified HTTPS request" log entries
  - Configure replication (Cinder Cheesecake)
  - Configure generic volume groups (these replace consistency groups, which are being deprecated)
  - Configure CHAP support
  - Enable automatic eradication on object destroy

Prerequisite

Before configuring the **`cinder.conf`** file, set up the **`cinder-api`** and **`cinder-scheduler`** services with a database, RPC messaging system, and authentication endpoint.

These instructions assume that the **`cinder-api`** and **`cinder-scheduler`** services are installed and configured in your OpenStack cluster.

Copy API Tokens from Purity

For each FlashArray, the Cinder volume service configuration requires an API token from a Purity array administrator. Actions performed by the volume driver use this token for authorization. Also, Purity logs the volume driver's actions as being performed by the user who owns this API token.

If you created a Purity user account that is dedicated to managing your OpenStack Cinder volumes, copy the API token from that user account.

Each FlashArray backend uses a different API token.

Use the appropriate create or list command below to display and copy the Purity API token.

To create a new API token:

```
pureadmin create --api-token USER
```

The following is example output. Copy the API token (902fdca3-7e3f-d2e4-d6a6-24c2285fe1d9 in this example) to use in the next step.

```
pureadmin create --api-token pureuser
Name    API Token                                Created
pureuser  902fdca3-7e3f-d2e4-d6a6-24c2285fe1d9  2014-08-04 14:50:30
```

To list an existing API token:

```
pureadmin list --api-token --expose USER
```

The output is the same as shown for the `pureadmin create` example above.

Notes about Purity API tokens:

- In multi-array environments, each backend uses a different API token.
- In replication environments, only the API tokens from the target FlashArrays are used.
- The API token or tokens are used in the **`cinder.conf`** file as shown below.
- (*Optional*) For GUI steps, see *Create a Purity API Token (GUI)* or *Copy an Existing Purity API Token (GUI)*.

Configuration File Entries

This section describes entries to be added to the Cinder volume configuration file. Some entries are optional or not required in all configurations.

Deprecated Volume Driver Configuration Settings in the DEFAULT Section

Beginning with the OpenStack Newton release, volume driver configuration settings are deprecated in the **`[DEFAULT]`** section of the Cinder volume configuration. These must instead be placed in named backend sections.

In the Ocata and later releases, having volume driver configuration entries in the **[DEFAULT]** section generates the following error:

```
Configuration for cinder-volume does not specify "enabled_backends".
Using the DEFAULT section to configure drivers is not supported since Ocata.
```

See *Parameters for Backend Stanzas* for a list of parameters that may appear only in a named backend section.

Named Backend Stanzas

A backend stanza contains the volume driver configuration entries for a single FlashArray. Volume driver entries are deprecated in the **[DEFAULT]** section and must appear only in named backend stanzas.

The following is a simplified example of two backend stanzas, named example1 and example2:

```
[example1]
volume_driver = cinder.volume.drivers.pure.PureFCDriver
san_ip = MANAGEMENT-INTERFACE-1
pure_api_token = API-TOKEN-1
use_multipath_for_image_xfer = True
volume_backend_name = example1
use_chap_auth = False

[example2]
volume_driver = cinder.volume.drivers.pure.PureISCSIDriver
san_ip = MANAGEMENT-INTERFACE-2
pure_api_token = API-TOKEN-2
volume_backend_name = example2
use_chap_auth = True
```

Register the Pure Storage Volume Driver

Use only one of the following entries per individual backend definition:

- For Fibre Channel:

  ```
  volume_driver=cinder.volume.drivers.pure.PureFCDriver
  ```

- For iSCSI:

  ```
  volume_driver=cinder.volume.drivers.pure.PureISCSIDriver
  ```

The Pure Storage volume driver supports mixing FC backends and iSCSI backends in the same environment.

The Pure Storage volume driver depends on the following entries. Add these entries to each named backend section.

```
san_ip=MANAGEMENT-INTERFACE
```

```
pure_api_token=API-TOKEN
use_multipath_for_image_xfer = True
```

where:

MANAGEMENT-INTERFACE:
The IP address of the FlashArray's management interface or a domain name that resolves to that IP address.

API-TOKEN:
The Purity authorization token that the volume driver uses to perform volume management on this FlashArray. See *Copy API Tokens from Purity*.

The **`san_ip`** entry registers the FlashArray as a Cinder backend.

We recommend that the **`san_ip`** entry be set to a virtual IP address that is assigned to both controllers for one FlashArray, or to a DNS name that resolves to such a virtual IP address. This setting is required to support high availability and non-disruptive upgrade.

The **`use_multipath_for_image_xfer = True`** entry enables multipathing.

Multipathing is required to support high availability, non-disruptive upgrades, and improved performance.

The multipathing setting here affects only operations performed directly by the driver, such as copying an image to a volume or a volume to an image. Steps for enabling multipathing for the compute notes where instances are hosted are described in *Enable Multipathing in the Nova Service*.

Backend Scheduling

Backend scheduling can be controlled through a user-defined filter function (or the Cinder scheduler default filter function), a user-defined goodness (weighing) function, and overprovisioning parameters.

Backend scheduling configurations are optional but we very strongly recommend setting some combination of backend scheduling options. At a minimum, the Cinder scheduler default filter function with the automatic overprovisioning ratio parameter are recommended.

User-defined filter and goodness functions are recommended, since when only the default Cinder scheduler filter function is used, thin provisioning and high oversubscription ratios can make arrays appear to have near-unlimited capacity. Using only the capacity filter can cause unbalanced scheduling between arrays with regards to IOPS and volume performance. Thin provisioning and high oversubscription ratios can make an array appear to have almost unlimited capacity.

As a result, we recommend the following:

- Using either the Cinder schedule default capacity filter (see *(Optional) Enable the Cinder Scheduler Default Filter Function*) or your own customer filter function
- Enabling the automatic overprovisioning ratio
- In multi-array environments, using a custom goodness function with the automatic overprovisioning ratio (this combination helps maintain a balance between different backends)

Thin Provisioning and Cinder Backend Scheduling

Thin provisioning can "confuse" Cinder, as the amount of storage actually used can be near zero for newly provisioned volumes. This would cause Cinder to keep adding volumes to an array with little data on it, even though the volumes already on the array, when fully provisioned, could exceed the array's capacity.

We recommend enabling the Cinder scheduler default filter function, at a minimum.

(Optional) Enable the Cinder Scheduler Default Filter Function

This configuration is optional but very strongly recommended.

We recommend setting the filter to less than 500, as shown in this example:

```
[DEFAULT]
scheduler_default_filters = AvailabilityZoneFilter,CapacityFilter,Capabiliti
esFilter,DriverFilter
[each-named-backend]
filter_function = "capabilities.total_volumes < 500"
```

These settings prevent Cinder requests from failing if Cinder attempts to create too many volumes on the array. These settings are recommended in addition to any custom goodness function you configure.

(Optional) Set an Oversubscription Ratio

This configuration is optional but very strongly recommended.

There are two configuration options related to the oversubscription ratio:

- **max_over_subscription_ratio**:
  The **`max_over_subscription_ratio`** is a Cinder parameter that limits how much a backend can be oversubscribed. The driver honors this limit when the **`pure_automatic_max_oversubscription_ratio`** parameter is not set or is set to False.

  An appropriate setting for **`max_over_subscription_ratio`** depends on array-specific conditions such as workload, data sets, and data reduction rates.

  The following are example **`max_over_subscription_ratio`** settings and their impact on thin provisioning.

  - **1.0**: This setting turns off thin provisioning.
  - **2.0**: This setting allows an array to be over-provisioned by a factor of two. This setting is very restrictive and typically not recommended for FlashArrays.
  - **20.0**: This setting is the Cinder default.
  - **100.0**: This setting provides minimal real-world restriction on thin provisioning. As always, storage administrators have the responsibility to monitor array capacities.

  Note: Values less than 1.0 are ignored and the default of 20.0 is used instead.
- **pure_automatic_max_oversubscription_ratio**:

When set to True, the driver calculates the oversubscription ratio as total provisioned/actual used. This setting causes the **max_over_subscription_ratio** to be ignored.

When set to False, the driver uses the value set by the configuration option **max_over_subscription_ratio**, which sets a hard limit on oversubscription.

This option defaults to False, but True is recommended.

Note: The **pure_automatic_max_oversubscription_ratio** is recommended for most use. There are a couple limited scenarios in which to consider setting this parameter to False. One scenario is in a large cloud environment. The scheduler could report thinly-provisioned arrays as having thousands of terabytes of free space. Later, provisioning a large amount of storage in a short time on such an array could be problematic. The other scenario is if the parameter is incompatible with an unusual data set and unusual workload.

To enable the automatically-calculated oversubscription ratio, add the following entry to **cinder.conf** file:

```
pure_automatic_max_oversubscription_ratio = True
```

To set the Cinder maximum oversubscription ratio, add a **max_over_subscription_ratio** entry: Also remove the **pure_automatic_max_oversubscription_ratio** parameter or set it to False. For example:

```
pure_automatic_max_oversubscription_ratio = False
max_over_subscription_ratio = 30.0
```

Place these oversubscription entries in the appropriate backend stanzas.

(Optional) Create Filter and Goodness Functions

In a multi-array environments, user-defined filtering and goodness functions give you control over how volumes are assigned to specific backends. These functions instruct the driver based on real-time metrics from the array or arrays.

You can define one function of each type per backend section.

If you do not create other filter or goodness functions, at a minimum we recommend the Cinder scheduler default filter function (see *(Optional) Enable the Cinder Scheduler Default Filter Function*). This setting prevents Cinder requests from failing if Cinder attempts to create too many volumes on the array.

However, you may replace it with your own filter function.

A filter function is a simple check that makes sure an array is not out of space or has hit an array limit. A goodness function makes more sophisticated use of arrays' status to help select the best array for the job.

Place custom filter and goodness functions in each named backend section.

Storage Metrics Table

The metrics available for these functions include the following:

Metric	Description
total_capacity_gb	Total amount of space (in GiB) available on the array
free_capacity_gb	Free capacity (in GiB) on the array
provisioned_capacity	Total amount of provisioned space (in GiB) on the array
total_volumes	Total count of volumes provisioned on the array, including volumes pending eradication
total_snapshots	Total count of snapshots provisioned on the array, including snapshots pending eradication
total_hosts	Total count of hosts on the array
total_pgroups	Total count of protection groups on the array
writes_per_sec	Total number of write operations per second currently being processed on the array
reads_per_sec	Total number of read operations per second currently being processed on the array
input_per_sec	Total input (in bytes) per second for the array
output_per_sec	Total output (in bytes) per second for the array
usec_per_read_op	Current latency per read operation for the array
usec_per_write_op	Current latency per write operation for the array
queue_depth	Current latency per read operation for the array

Note: These values include all objects on the array, not only those that are managed by Cinder.

Create a Goodness Function

Goodness functions, also called weighing functions, employ a user-defined formula based on real-time array status to assign a number from 1 to 100 to each backend. The driver then determines volume placement based on the assigned scores, with 100 being the best score and most likely to receive the new volume, and 0 being the least appropriate to receive the new volume. Through goodness functions, the driver "weighs" the real-time status of multiple backends, based on user-defined criteria.

The user-defined formula can be based on array metrics such as those listed above (*Storage Metrics Table*). With many of these functions, the actual values in the formulas would need to be adjusted on a case by case basis, depending on the environment, workload, and data characteristics.

Notes on these examples:

- Each example function would be written in the configuration file in one line. Some are shown here in multiple lines only to avoid wrap-around lines.
- These examples use **`pure-2`** as an example backend name.
- These examples use the max and min functions, division into 1, and multiplying by 100 to avoid dividing by zero and to keep the return value between 0 and 100, as required by Cinder. Make sure your functions do not return a negative value.

- More advanced scheduling techniques can be used as needed.

Example 1

This example shows a goodness function based on write latency. The driver would schedule to the array with the lowest write latency.

```
[pure-2]
goodness_function = "100 * (1 / max(capabilities.usec_per_write_op, 1))"
```

Example 2

This example goodness function considers read latency, write latency, and queue depth. The driver would schedule to the array with the best score based equally on these three considerations.

```
[pure-2]
goodness_function = "100 * (1 / max(capabilities.usec_per_read_op, 1))
                    * (1 / max(capabilities.usec_per_write_op, 1))
                    * (1 / max(capabilities.queue_depth, 1))"
```

Example 3

This example still considers read latency, write latency, and queue depth, but puts more weight (25% more) on queue depth:

```
[pure-2]
goodness_function = "100 * (1 / max(capabilities.usec_per_read_op, 1))
                    * (1 / max(capabilities.usec_per_write_op, 1))
                    * (1.25 * max(1 / max(capabilities.queue_depth, 1), 1))"
```

Using "0.75" instead of "1.25" would change the function to put less importance on queue depth than on the latencies.

Example 4

This example returns a goodness score of 100 while the array's IOPS are below 25K (300000 - 275000). From IOPS of 25K to 300K, the goodness score drops linearly. At 300K or more, the goodness score is 0.

```
[pure-2]
goodness_function = "max(min(100 * (300000 - capabilities.reads_per_sec) /
                     275000), 100), 0)"
```

Example 5

This example gives lower weighting to arrays that already have greater numbers of protection groups, hosts, and snapshots, and schedules to arrays that have smaller numbers of those objects.

```
[pure-2]
goodness_function = 100 * (1 / (50 - max(capabilities.total_pgroups, 1)) *
                           (1 / (50 - max(capabilities.total_hosts, 1))) *
```

```
                          (1 / (5000 - max(capabilities.total_snapshots, 1)))"
```

Create a Filter Function

A filter function is used to protect against exceeding the limitations of an array.

Compared to a goodness function, a filter function is more hard-coded and results in a "yes" or "no" determination of each array, until a passing array is found. With filter functions, there is no comparison across multiple backends.

For example, this example filter function prevents creating a new volume on an array that already has 500 volumes:

```
[pure-2]
filter_function = "capabilities.total_volumes < 500"
```

This example helps prevent scheduling new volumes onto an array that is under heavy load:

```
[pure-2]
filter_function = "capabilities.usec_per_read_op < 3000 and
                   capabilities.usec_per_write_op < 3000 and
                   capabilities.queue_depth < 100"
```

(This example would be written in the configuration file in one line. It is shown here in multiple lines only to avoid line wrapping.)

This example prevents scheduling new volumes on a backend with a latency greater then 3 ms and a queue depth greater than 100. However, this example function is not very flexible. Temporary spikes could mean failed volume creations, if no arrays had low latency at that particular moment. A goodness function would provide more flexibility.

Fibre Channel Zone Manager

If you are using Fibre Channel, we recommend configuring a Fibre Channel Zone Manager. Brocade and Cisco Zone Managers are supported. Each Fibre Channel fabric requires its own Zone Manager configuration entry.

This example shows the configuration of a Cisco Fibre Channel Zone Manager.

```
[DEFAULT]
mode=fabric

[fc-zone-manager]         # this name, fc-zone-manager, is required by Cinder
fc_fabric_names = fabric-1
fc_san_lookup_service = cinder.zonemanager.drivers.cisco.cisco_fc_san_lookup_service.CiscoFCS
zone_driver = cinder.zonemanager.drivers.cisco.cisco_fc_zone_driver.CiscoFCZoneDriver
zoning_policy = initiator-target
cisco_sb_connector = cinder.zonemanager.drivers.cisco.cisco_fc_zone_client_cli.CiscoFCZoneCli

[fabric-1]
cisco_fc_fabric_address = IP_ADDRESS_OF_SWITCH
```

```
cisco_fc_fabric_user = USERNAME_OF_SWITCH
cisco_fc_fabric_password = PASSWORD_OF_SWITCH
cisco_fc_fabric_port = 22
cisco_zone_activate = True
cisco_zoning_vsan = VSAN_ON_SWITCH
```

Here are the long entries that tend to scroll off the end of the page. Each name begins with **`cinder.zonemanager.drivers.cisco`** (not shown here).

```
.cisco_fc_san_lookup_service.CiscoFCSanLookupService
.cisco_fc_zone_driver.CiscoFCZoneDriver
.cisco_fc_zone_client_cli.CiscoFCZoneClientCLI
```

Each fabric name listed in the **`fc_fabric_names`** entry must have its own section such as the **`[fabric_1]`** section shown in the example.

(Optional) Generic Volume Groups

Generic volume groups are a core cinder feature that is replacing consistency groups.

Use a volume group type that has this extra specification (syntax and case are important):

```
set consistent_group_snapshot_enabled='<is> True'
```

The Pure Storage volume drivers support **`consistent_group_snapshot_enabled`** type groups that work the same way as consistency groups did previously and are interpreted as representing Purity protection groups.

(Optional) Glance Image Caching

Glance image caching is a Cinder feature that prevents the repeated downloading of images. Combined with Purity's support for very fast volume cloning, enabling this feature offers significant efficiency savings.

Enable Glance Image Caching

The following are example cinder.conf entries to enable and configure glance image caching:

```
[DEFAULT]
cinder_internal_tenant_project_id = b7455b8974bb4064ad247c8f375eae6c
cinder_internal_tenant_user_id = f46924c112a14c80ab0a24a613d95eef
[named-backends]
image_volume_cache_enabled = True
image_volume_cache_max_size_gb = 200
image_volume_cache_max_count = 50
```

Notes about image caching entries:

- **`image_volume_cache_max_size_gb`** is the maximum size of all images in the cache.
- **`image_volume_cache_max_count`** is the maximum number of images that can be cached.

- Set either of these values to **0** to remove any limitation on that parameter.
- **image_volume_cache_max_size_gb** and **image_volume_cache_max_count** are optional.
- Place the **image_volume_cache_*** settings in the appropriate backend sections.
- In multiple backend environments, all backends must use the same tenant project and user settings (which are placed in the [DEFAULT] section).

Enable the TRIM Command for Glance Images

Support for TRIM commands is available on volumes booted from Glance images that are created with properties described in this section.

Add the following Glance image properties:

- hw_scsi_model=virtio-scsi
- hw_disk_bus=scsi

These properties can be added on the command line or through the Horizon OpenStack Dashboard (not in the **cinder.conf** configuration file.

The following is an example Glance command (enter as a single line):

```
glance image-update --properties hw_scsi_model=virtio-scsi
                    --properties hw_disk_bus=scsi imgName
```

Notes about image caching:

- These commands are supported for volumes that are attached to a VM, including VMs on the root disk that the VM is booted from.
- The volumes must be newly attached to a VM.
  Volumes that are already attached to a VM must be detached from the VM and reattached.
- The VM must be using **virtio-scsi**. The virtual storage driver must be configured to use the virtio scsi controller.
- As always, the Cinder service must be restarted after the parameters are added.

Create the Tenant Project and User (for Glance Image Support)

A tenant project and user are required only if Glance image caching is enabled.

Creating the tenant project and user involves these two steps:

1. Running keystone commands to create the tenant project and user
2. Taking the tenant and user **id** values provided in the keystone output, to use in **cinder_internal_tenant_project_id** and **cinder_internal_tenant_user_id** entries in the Cinder configuration file

Keystone Commands

Run the following keystone commands on the OpenStack node where the volume driver is installed:

- **keystone tenant-create --name *TENANT-NAME***
- **keystone user-create --name *USER-NAME*** --tenant ***TENANT-NAME*** --pass ***PASSWORD***

The following are example keystone commands, using the project name and user name **cinder_internal**:

```
keystone tenant-create --name cinder_internal
keystone user-create --name cinder_internal --tenant cinder_internal --pass s3cret!
```

The following includes example output of these commands:

```
# keystone tenant-create --name cinder_internal
+-------------+----------------------------------+
| Property|           Value                   |
+-------------+----------------------------------+
| description |                                  |
| enabled |            True                   |
|  id     | 3bf76f01a3ab40dea9cfaf9638aaef64 |
| name    |         cinder_internal           |
+-------------+----------------------------------+
# keystone user-create --name cinder_internal --tenant cinder_internal --pass s3cret!
+----------+----------------------------------+
| Property |          Value                   |
+----------+----------------------------------+
|email |                                      |
| enabled|             True                  |
|id    | de613425cac64a36b86dcf51fe0dc968    |
| name |         cinder_internal             |
| tenantId | 3bf76f01a3ab40dea9cfaf9638aaef64|
| username |     cinder_internal             |
+----------+----------------------------------+
```

Tenant Configuration File Entries

Create **cinder_internal_tenant_project_id** and **cinder_internal_tenant_user_id** entries, using the values given in the **id** rows in the keystone output. For example:

```
cinder_internal_tenant_project_id = 3bf76f01a3ab40dea9cfaf9638aaef64
cinder_internal_tenant_user_id = de613425cac64a36b86dcf51fe0dc968
```

Create the Keystone tenant entries in the **[DEFAULT]** section, even if you use multiple named backend definitions.

(Optional) CHAP Support

The volume driver supports the iSCSI Challenge Handshake Authentication Protocol (CHAP).

When CHAP support is enabled, the volume driver does the following for host objects that it creates:

- Generates a CHAP username
- Generates a random password
- Sets the host_user and host_password
- Stores CHAP information in Cinder's database for continued use

The following parameters in the Cinder configuration file enable or disable CHAP support:

use_chap_auth=True
Enables CHAP support

use_chap_auth=False
Disables CHAP support

Notes about CHAP support:

- Place CHAP settings in the appropriate named backend sections.
- If you use multiple backends, CHAP support can be enabled for some backends and disabled for others.
- The driver only modifies credentials for hosts that it manages. It does not modify credentials for a manually-created host object.

(Optional) Enable Automatic Eradication on Destroy

By default, Purity holds deleted (or destroyed) volumes, snapshots, and volume groups for 24 hours in an eradication pending state. During the 24-hour eradication pending period, the deleted items can be recovered. The same is true for items deleted by the driver.

To instead have items be eradicated on deletion, add the following entry to the appropriate named backend sections:

```
pure_eradicate_on_delete = True
```

When this parameter is set to True, the 24-hour eradication pending state does not apply and deleted items cannot be recovered.

When this parameter is not present or is set to False, deleted items are available for recovery during the 24-hour eradication pending state.

(Optional) Configure Replication

The Pure Storage driver supports asynchronous replication between connected FlashArrays.

The supported use case is to recover a FlashArray, for instance after a catastrophic data center event. This requires that the OpenStack administrator configure a second FlashArray as a full replica of the first, and allows either the full array or specific volumes to be replicated.

Three steps are required to configure replication: the source and target arrays are connected arrays in Purity, the target arrays are defined with the **`replication_device`** parameter, and volumes to be replicated are of a volume-type with the specification **`replication_enabled`**.

These configurations are described in the next sections.

Notes about replication:

- Replication failover is supported from the primary array to any of the target arrays. However, a subsequent replication failover is only supported back to the original primary array.
- The target array should be accessible by the same Nova availability zone as the source array.
- The target array should be use the same storage protocol as the source array.

Configure Replication Target Arrays

Target arrays are defined with the **`replication_device`** parameter, which contains the following key-value pairs:

- backend_id:***NAME_OF_REPLICATION_TARGET_BACKEND***
- san_ip:***IP_MANAGEMENT_ADDRESS_OF_TARGET_ARRAY***
- api_token:***API_TOKEN_FOR_TARGET_ARRAY***

The following example shows two target arrays (the source array is not used with the **`replication_device`** parameter, and the API tokens are truncated):

```
replication_device = backend_id:pure-2,san_ip:10.10.10.101,api_token:d89e29d9-b3fe-
replication_device = backend_id:pure-3,san_ip:10.10.10.102,api_token:f29643cf-bf70-
```

Notes about replication target arrays:

- API tokens in these examples are truncated for space considerations.
- The **`replication_device`** parameter differs from other configuration parameters in that multiple entries of this parameter are supported. Subsequent occurrences of this parameter do not overwrite previous occurrences.
- This parameter defines the replication target array or arrays. The source array does not appear in this parameter.
- Place replication parameters in the appropriate named backend sections.

Configure Optional Replication Parameters

The following optional parameters are supported for replication:

Option	Description	Default
pure_replica_interval_default	Snapshot retention interval (in seconds)	900
pure_replica_retention_short_term_default	Retain all snapshots on target array for this time (in seconds)	14400
pure_replica_retention_long_term_per_day_default	Retain this many snapshots for each day	3

Option	Description	Default
pure_replica_retention_long_term_default	Retain per-day snapshots on target array for this number of days	7

If entered in the **`cinder.conf`** file, these values override the values configured in Purity.

Configure One or More Replication Volume Types

Volumes to be replicated must be of a volume-type that has the extra specification **`set replication_enabled='<is> True'`**.

These Cinder commands create a volume type that supports replication:

```
cinder type-create "ReplicationType"
cinder type-key "ReplicationType" set replication_enabled='<is> True'
```

Note: Copy the syntax exactly as shown, including the angle brackets and the capitalization of "True".

Configure Connected Arrays

The source and target arrays must be connected in Purity, for instance in the Purity GUI system > Connected Arrays pane or with the **`purearray connect`** command.

See the FlashArray User Guide for a description of these commands and of replication.

(Optional) Verifying HTTPS Requests

The driver supports to separate configurations related to verifying HTTPS requests. One deals with avoiding "Unverified HTTPS request" log entries (without using an SSL certificate to verify HTTPS requests). The other configures an SSL certificate to verify HTTPS requests.

Both of these configurations require version 1.6.0 or later of the Pure Storage FlashArray REST Client.

The client is available for download from the ***Python Software Foundation site*** [https://pypi.python.org/]. At the time of this writing, this is the link to version 1.6.0 of the ***Pure Storage FlashArray REST Client package*** [https://pypi.python.org/pypi/purestorage/1.6.0].

Unverified HTTPS Requests Log Entries

Multiple "Unverified HTTPS request" entries can be seen in cinder-volume and nova-compute log files. To avoid these log entries, use version 1.6.0 or later of the Pure Storage FlashArray REST Client.

Verifying HTTPS Requests with SSL Certificates

To use the FlashArray's SSL certificates to verify HTTPS requests, copy the certificates from the FlashArray and add the following entries to the appropriate named backend sections of the **`cinder.conf`** file:

- driver_ssl_cert_verify: Set to **`True`** to enable validating HTTPS request certificates
- driver_ssl_cert_path: Set to the path to the CA bundle to verify HTTPS certificates

For example:

```
driver_ssl_cert_verify = True
driver_ssl_cert_path = /usr/local/share/ca-certificates
```

The following are the steps to apply the FlashArray certificate to the Cinder host (root access might be required on the Cinder host).

1. Download the certificate from Purity, for instance with the System > Configuration > SSL Certificate > options icon > Export Certificate > Download option.
2. Download the certificate to your local machine.
3. Copy the certificate to the **`/usr/local/share/ca-certificates`** directory on the Cinder host.
4. On the Cinder host, run the command **`update-ca-certificates`** or **`sudo update-ca-certificates`** to update the CA certificate store.

In summary, there are four parts to this configuration: the new configuration options in the **`cinder.conf`** file, copying the certificate, updating the CA certificate store, and using the 1.6.0 or later REST Client.

(Optional) Enable Verbose Mode

Verbose mode is recommended for the initial setup, to help debug configuration issues. Verbose mode can be turned off whenever the setup is stable.

To enable verbose mode, place the following entry in the **`[DEFAULT]`** section:

```
Verbose = True
```

To disable verbose mode, remove the entry or set it to **`False`**:

```
Verbose = False
```

Edit the Cinder Volume Service Configuration File

This section assumes that you have already set up **`cinder-api`** and **`cinder-scheduler`** with a database, RPC messaging system, and authentication endpoint. The Cinder volume service also uses these configuration settings and must have the same database, RPC messaging, and authorization settings as the **`cinder-api`** and **`cinder-scheduler`** services. Installing these services on the same node ensures that the Cinder volume service instance that instantiates and manages the driver has access to the same database, RPC messaging, and authorization settings as the **`cinder-api`** and **`cinder-scheduler`** services.

If you installed these services on the same node, copy the file **`/etc/cinder/cinder.conf`** from the node where the **`cinder-api`** and **`cinder-scheduler`** services are installed, to the node where the Cinder volume service is installed.

Notes:

- Create a backup copy of the file **`/etc/cinder/cinder.conf`** before you begin.

- To configure multiple FlashArrays, see the appendix *Configure Multiple Pure Storage FlashArrays* for configuration information.
- The entries listed in the instructions below may already exist in your **`cinder.conf`** file. Make sure you replace or delete any existing entries, rather than leaving conflicting settings in the file.
- The **`volume_driver`**, **`san_ip`**, and **`pure_api_token`** entries register the volume driver as default volume driver for Cinder.
- We recommend that the **`san_ip`** entry be set to a virtual IP address that is assigned to both controllers for one FlashArray, or to a DNS name that resolves to such a virtual IP address. This setting is required to support high availability and non-disruptive upgrade.
- Multipathing is required to support high availability, non-disruptive upgrades, and improved performance.
  The multipathing setting here affects only operations performed directly by the driver, such as copying an image to a volume or a volume to an image. Steps for enabling multipathing for the compute notes where instances are hosted are described in *Enable Multipathing in the Nova Service*.

Editing steps:

Edit the file **`/etc/cinder/cinder.conf`** on the OpenStack node where the volume driver is installed:

- Create a named backend stanza for each backend. At least one named backend stanza is required. See *Named Backend Stanzas* and *Configure Multiple Pure Storage FlashArrays*.
- ***Required parameters***
  - **API token**: If you have not already done so, copy your Purity API authorization token (described in *Copy API Tokens from Purity*). You need the appropriate API token for each FlashArray backend.
  - **Volume driver**: Register either the Fibre Channel or iSCSI volume driver per backend. Use *one* of the following:
    - `volume_driver=cinder.volume.drivers.pure.PureFCDriver`
    - `volume_driver=cinder.volume.drivers.pure.PureISCSIDriver`

    Add the appropriate volume driver entry to each named backend section.
  - **Array parameters**: The volume driver depends on the following three entries. Place these entries directly after the volume driver entry:

    ```
    san_ip=MANAGEMENT-INTERFACE
    pure_api_token=API-TOKEN
    use_multipath_for_image_xfer = True
    ```

    For information on these entries, including ***MANAGEMENT-INTERFACE*** and ***API-TOKEN***, see *Register the Pure Storage Volume Driver*).

- ***Recommended parameters***
  - **Verbose logging**:
    The following entry is recommended during initial setup, in the **[DEFAULT]** section:

    ```
    Verbose = True
    ```

  - **Backend scheduling**:
    Customize user-defined filter and goodness functions and enable the driver's automatic oversubscription ratio. See *Backend Scheduling*.

    Backend scheduling is optional but recommended. At a minimum, the following are recommended:

    ```
    [DEFAULT]
    scheduler_default_filters = AvailabilityZoneFilter,CapacityFilter,Cap
    abilitiesFilter,DriverFilter
    [each-named-backend]
    filter_function = "capabilities.total_volumes < 500"
    ```

  - **Fibre Channel Zone Manager**: If using Fibre Channel, add the entries described in Fibre Channel Zone Manager.
  - **Glance image caching**: To take advantage of Glance image caching, add the entries described in *(Optional) Glance Image Caching*.
    Glance image caching is optional.
  - **Glance imaging parameters**: If Glance image caching support is enabled, add the entries to create the tenant project and user, as described in *Create the Tenant Project and User (for Glance Image Support)*.
    These parameters are required when Glance image caching is enabled.
- ***Optional parameters***
  - **SSL verification**:
    To use the FlashArray's SSL certificates to verify HTTPS requests, see *(Optional) Verifying HTTPS Requests*.

    Version 1.6.0 or later of the Pure Storage FlashArray REST Client is required.
  - **"Unverified HTTPS request" log entries**:
    To avoid "Unverified HTTPS request" log entries, see *(Optional) Verifying HTTPS Requests*.
  - **Configure replication (Cinder Cheesecake)**:
    To configure asynchronous replication between FlashArrays, including required steps in Purity, see *(Optional) Configure Replication*.
  - **CHAP support**: Optionally enable CHAP authorization.
    The entry is placed in each appropriate named backend section. CHAP support can be enabled for some backends and disabled for others.

See *(Optional) CHAP Support*.

- **Automatic eradication**: To disable Purity's 24-hour eradication pending period for deleted objects, see *(Optional) Enable Automatic Eradication on Destroy*.
  With this configuration, deleted objects cannot be recovered.

- **Save changes**
  Save the `cinder.conf` file.

- **Other steps**:
  See the next section to restart the cinder service (required) and optionally to enable TRIM commands for volumes booted from Glance images and to avoid "Unverified HTTPS request" log entries.

Restart the Cinder Service

Restart the Cinder volume service for your new settings to take effect, then confirm that the service is running:

- For Ubuntu:
  1. `service cinder-volume restart`
  2. `service cinder-volume status`
- For Red Hat Enterprise Linux or CentOS:
  1. `service openstack-cinder-volume stop`
  2. `service openstack-cinder-volume start`
  3. `service openstack-cinder-volume status`

Other Configuration Steps

Configure a Replication Volume Type

If replication is enabled, remember to create a volume-type that supports replication. Only volumes of such a volume type are replicated.

See *Configure One or More Replication Volume Types*.

Configure Connected Arrays for Replication

If replication is enabled, the source and target arrays must be connected in Purity.

See *Configure Connected Arrays*.

Configure a Generic Volume Group Type

To use generic volume groups to map to Purity protection groups, see *(Optional) Generic Volume Groups*.

The use of a generic volume group is optional.

Enable the TRIM Command for Glance Volumes

If Glance imaging is enabled, optionally use a **`glance image-update`** command with **`hw_scsi_model`** and **`hw_disk_bus`** properties to enable support for TRIM commands on volumes booted from those Glance images.

See *(Optional) Glance Image Caching*.

Example Cinder Volume Service Configuration File

The following is an example completed configuration file. Your values and options will be different. Not all configurations included in this file are required.

```
[DEFAULT]
# Enable backends, required
enabled_backends = pure-1-iscsi

# Enables verbose logging, optional, recommended for initial setup
Verbose = True

# Required only if CHAP authorization is enabled
cinder_internal_tenant_project_id = 3bf76f01a3ab40dea9cfaf9638aaef64
cinder_internal_tenant_user_id = de613425cac64a36b86dcf51fe0dc968

# Enables the Cinder Scheduler Default Filter Function, optional, recommended
scheduler_default_filters = AvailabilityZoneFilter,CapacityFilter,Capabiliti
esFilter,DriverFilter

# Defines an iSCSI backend named pure-1-iscsi
[pure-1-iscsi]
volume_backend_name = pure-1-iscsi
volume_driver=cinder.volume.drivers.pure.PureISCSIDriver
san_ip=10.101.10.10
pure_api_token=f29643cf-bf70-a1c5-222a-a3015f86d7ea
use_multipath_for_image_xfer = True

# Enable CHAP authorization, optional
use_chap_auth = True

# Enables Glance image caching, optional
image_volume_cache_enabled = True

# Glance image caching parameters, optional
image_volume_cache_max_size_gb = 200
image_volume_cache_max_count = 50
```

```
# Verify HTTPS requests with SSL certificates, optional
driver_ssl_cert_verify = true
driver_ssl_cert_path = /usr/local/share/ca-certificates

# Enables the automatic maximum oversubscription ratio, optional
pure_automatic_max_oversubscription_ratio = True

# Disables automatic eradication (keeping the 24-hour eradication pending
# period), optional
pure_eradicate_on_delete = false

# Defines two replication target FlashArrays (pure-2, pure-3), optional
# On array pure-2, retain per-day snapshots for 14 days (instead of the
# default of 7 days), optional (enter as a single line)
replication_device = backend_id:pure-2,san_ip:10.10.10.101,api_token:d89e29d,
pure_replica_retention_long_term_default:14
replication_device = backend_id:pure-3,san_ip:10.10.10.102,api_token:f29643c

# Sets a custom filter_function for this backend, optional, recommended
filter_function = "capabilities.total_volumes < 500"
```

See *Configure Multiple Pure Storage FlashArrays* for information on multiple backend stanzas.

Chapter 7. Verify the Configuration

This chapter describes the following actions to verify the correct configuration:

- Use the Cinder volume service to write to the Pure Storage FlashArray
- Write to the Pure Storage FlashArray directly through the volume driver on an OpenStack node
- Use the Purity command `purehost list --all` to verify multipathing

Write to the FlashArray from an OpenStack Instance

This section describes an example of confirming your setup by using OpenStack to create a volume, attach it to an instance, and create a file system on the volume. These steps cause OpenStack's Cinder volume service to write to the Pure Storage FlashArray through the volume driver.

This example uses the command line and can be performed on any machine with an OpenStack CLI client. These steps can also be done in the OpenStack GUI.

To write to the FlashArray from an OpenStack instance:

1. Create a volume with the `cinder create` command. An example command and output are shown here:

```
$ cinder create --display-name pureVol 1
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|     attachments     |                  []                  |
|  availability_zone  |                 nova                 |
|       bootable      |                false                 |
|      created_at     |      2014-07-28T21:06:20.400601      |
| display_description |                 None                 |
|     display_name    |               pureVol                |
|      encrypted      |                False                 |
|          id         | 8b7bc53a-0380-47a6-9ae0-00cf11d8c43a |
|       metadata      |                  {}                  |
|         size        |                  1                   |
|     snapshot_id     |                 None                 |
|     source_volid    |                 None                 |
|        status       |              creating                |
|     volume_type     |                 None                 |
+---------------------+--------------------------------------+
```

2. To confirm that the volume is created, run the `purevol list` command on the Purity command line:

```
pureuser@pure01> purevol list
Name                                                Size Source Created
                    Serial
volume-8b7bc53a-0380-47a6-9ae0-00cf11d8c43a-cinder  1G    -      2014-
07-28 13:34:05 PDT  ED596ADA70A2F9B70001236E
```

3. On your OpenStack CLI client, attach this volume to an instance:

```
$ nova volume-attach INSTANCE VOLUME-ID
+----------+--------------------------------------+
| Property | Value                                |
+----------+--------------------------------------+
| device   | /dev/vdb                             |
| id       | 8b7bc53a-0380-47a6-9ae0-00cf11d8c43a |
| serverId | 75b4f50c-ac4a-4059-b17e-f4dbeea61762 |
| volumeId | 8b7bc53a-0380-47a6-9ae0-00cf11d8c43a |
+----------+--------------------------------------+
```

4. (*Optional*) The following are example commands to mount the volume in the instance's file system. This example sets up a file system and generates a spike of write activity that is displayed in the Purity GUI's Dashboard tab. Run the following commands inside the instance:

   a. `sudo mkfs.ext3 /dev/vdb`

   b. `sudo mkdir -p /mnt/myVolume`

   c. `sudo mount /dev/vdb /mnt/myVolume`

Write to the FlashArray Directly from the Volume Driver

The steps described in this section test a different function of the driver. For a volume that is not attached to an instance, the following actions write to the FlashArray directly through the volume driver:

- Create a volume from an image
- Create an image from a volume
- Migrate a volume between backends

This example uses **`glance`** to find an image number, then **`cinder create`** to create a volume from the image, and finally **`cinder list`** to check the new volume's status.

To write to your FlashArray directly through the volume driver:

1. Use `glance` to find an image number. An example command and output are shown here.

```
$ glance image-list
+--------------------------------------+----------+-------------+-----
| ID                                   | Name     | Disk Format | Cont
```

```
+--------------------------------------+---------+------------+-----
| 7d03ff64-193f-4718-a720-f0e815fafeb5 | example1 | qcow2       | bare
+--------------------------------------+---------+------------+-----
```

2. Create a volume with the **`cinder create`** command. Example command and output:

```
$ cinder create --image-id 7d03ff64-193f-4718-a720-f0e815fafeb5 1
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|     attachments     |                  []                  |
|  availability_zone  |                 nova                 |
|       bootable      |                false                 |
|      created_at     |      2014-08-15T00:47:24.699223      |
| display_description |                 None                 |
|     display_name    |                 None                 |
|      encrypted      |                False                 |
|          id         | 06b06fb2-d200-4b8c-8c4f-0ed698a78313 |
|       image_id      | 7d03ff64-193f-4718-a720-f0e815fafeb5 |
|       metadata      |                  {}                  |
|         size        |                  1                   |
|     snapshot_id     |                 None                 |
|     source_volid    |                 None                 |
|        status       |              creating                |
|     volume_type     |                 None                 |
+---------------------+--------------------------------------+
```

3. The time required to create the volume depends on the size of the source image. Use a command such as **`cinder list`** to check the volume's status.
   A status of **`available`** confirms that this portion of the setup is correct:

```
$ cinder list
+--------------------------------------+-------------+--------------+--
|                  ID                  |    Status   | Display Name | S
+--------------------------------------+-------------+--------------+--
| 06b06fb2-d200-4b8c-8c4f-0ed698a78313 | downloading |     None     |
+--------------------------------------+-------------+--------------+--
$
$
$ cinder list
+--------------------------------------+-------------+--------------+--
|                  ID                  |   Status    | Display Name | S
+--------------------------------------+-------------+--------------+--
| 06b06fb2-d200-4b8c-8c4f-0ed698a78313 | available   |     None     |
+--------------------------------------+-------------+--------------+--
```

   The **`Bootable`** field also changes to **`true`**:

```
----------+--------------+------+-------------+----------+-------------+
  Status  | Display Name | Size | Volume Type | Bootable | Attached to |
----------+--------------+------+-------------+----------+-------------+
available |     None     |  1   |     None    |   true   |             |
----------+--------------+------+-------------+----------+-------------+
```

If you encounter an error, check the Cinder logs (in **`/var/log/cinder/`**) on the node where the volume driver is installed.

Confirm Multipathing in Purity

These steps confirm that multipathing is enabled and require that you have already attached a volume to an instance.

Run the following command in Purity:

```
purehost list --all
```

Example output:

```
purehost list --all
Name  WWN IQN                       Host Group  LUN  Vol      Target Port
host1  -  iqn.2014-08.com.example:target0  clus1  10  vol1  CT0.ETH0
                                                             CT0.ETH2
                                                             CT1.ETH0
                                                             CT1.ETH2
host2  -  iqn.2014-08.com.example:target1  clus1  11  vol2  CT0.ETH0
                                                             CT0.ETH2
                                                             CT1.ETH0
                                                             CT1.ETH2
```

Non-disruptive upgrades and high availability require at least two FlashArray controllers (**`CT0`** and **`CT1`** in this example) per host.

Chapter 8. Notes about The Volume Driver and Purity

How the Volume Driver Selects an iSCSI or Fibre Channel Port

At Cinder volume service initialization, the Pure Storage Volume Driver goes through Purity's list of FlashArray ports. For each port, the driver runs a discovery command to check if that port is reachable. The driver selects the first port that is reachable and uses that port for communication with the volume.

It is important that multipathing be enabled to ensure optimal FlashArray performance and to support for non-disruptive upgrades and high availability.

Notes:

- The order in which ports are checked is the same as the order that ports are returned by the **`pureport list`** CLI command and a **`GET port`** REST API request.
- Every time a volume is attached to an instance, the driver checks if the FlashArray port that it selected is still reachable. If it cannot reach the port, the driver again goes through Purity's list of ports to select a reachable port.

Cinder Volume and Snapshot Names in Purity

UUIDs

The Cinder volume service generates a UUID for each volume and each snapshot that it creates. An example UUID is shown here:

```
00114832-b093-426c-8f15-b5a903bd5b26
```

Volume Names

By default, the Cinder volume service generates volume names based on the UUID. For example:

```
volume-00114832-b093-426c-8f15-b5a903bd5b26
```

The volume driver appends **`-cinder`** to this ID. This is an example of how a Cinder volume appears in Purity (in the UI **Storage > Volumes** tab or in the output of a CLI command such as **`purevol list`**):

```
volume-00114832-b093-426c-8f15-b5a903bd5b26-cinder
```

Snapshot Names

By default, the Cinder volume service also generates snapshot names based on the UUID. For example:

```
snapshot-e0fd258a-7bd8-43f4-9900-3a123c5fd5d6
```

The PureStorage Volume Driver appends a period to the Cinder volume name, followed by the Cinder snapshot name. In Purity, the pattern for snapshot names managed by Cinder is the following:

```
volume-<volume-UUID>-cinder.snapshot-<snapshot-UUID>
```

For example:

```
volume-00114832-b093-426c-8f15-b5a903bd5b26-cinder.snapshot-
e0fd258a-7bd8-43f4-9900-3a123c5fd5d6
```

Capacity Reporting

The driver takes advantage of updated Cinder filters that are aware of thin provisioning, deduplication, and compression. Capacity is reported based on current data reduction rates on the array.

Support for Multiple Pure Storage FlashArrays

See *Configure Multiple Pure Storage FlashArrays* to have your Cinder volume service manage more than one FlashArray.

Chapter 9. Logs, Error Messages, and Troubleshooting

See also this Knowledge Base [http://support.purestorage.com] article: Troubleshooting the Pure Storage OpenStack® Cinder Driver [https://support.purestorage.com/Solutions/Cloud/OpenStack/Troubleshooting_Pure_Storage_OpenStack%C2%AE_Cinder_Driver] (login required).

The Knowledge Base article has the most recent troubleshooting information and additional tips.

Logging Level

The Cinder volume service generates a log file, by default in the **`/var/log/cinder/`** directory. The log file is typically named **`cinder-volume.log`** or **`volume.log`**.

Settings in the file **`/etc/cinder/cinder.conf`** on the OpenStack node where the PureStorage Volume Driver is installed control the logging level. By default, logging includes messages of level WARNING or higher (including ERROR and CRITICAL).

To include INFO or DEBUG log messages (*optional*):

1. Edit the file **`cinder.conf`** to add one of the following lines in the **`[DEFAULT]`** section:

   `verbose=True`
   Log all messages of level INFO or higher

   `debug=True`
   Log all messages of level DEBUG or higher (including INFO and others)

2. Restart the Cinder volume service for your changes to take effect.

We recommend setting **`verbose=True`** to make debugging easier during configuration. The setting **`debug=True`** results in log files that are very verbose, and is not necessary for debugging most problems. The setting **`debug=True`** overrides **`verbose=True`**, if both entries are present in the file.

These settings affect all Cinder services on that node (after those services are restarted).

Log Messages

Driver operations failures generate ERROR messages in the Cinder volume service log file. Pure Storage Volume Driver failures generate the following exception types:

`PureHTTPError`
Failures reported by the Pure Storage REST API

`PureDriverException`
Failures in establishing connections, most commonly from incorrect configuration

These exceptions appear in the log alongside stack traces and contain messages that explain the failure.

Debugging

This section describes several error scenarios, with probable causes and steps for diagnosis and recovery. Error recovery steps typically require storage administrator permission.

Deprecated Volume Driver Configuration in [DEFAULT] Section

In the OpenStack Ocata and later releases, having volume driver configuration entries in the **`[DEFAULT]`** section generates an error and prevents the cinder service from starting properly.

Symptoms

The volume service is down and the cinder service fails to start.

The following error message is seen in the volume service log, immediately after startup:

```
Configuration for cinder-volume does not specify "enabled_backends".
Using the DEFAULT section to configure drivers is not supported since Ocata.
```

Admin information

Search for **`enabled_backends`** in the Cinder volume service's logs in **`/var/log/cinder/`** on the node where the driver is running.

Probable cause

A volume driver configuration entry is present in the **`[DEFAULT]`** section.

Recovery

Remove volume driver configuration entries from the **`[DEFAULT]`** section of the configuration file. Instead place volume driver configuration entries in named backend stanzas.

See *Named Backend Stanzas* and *Configure Multiple Pure Storage FlashArrays* for information on backend stanzas.

See *Parameters for Backend Stanzas* for a list of parameters that may appear only in a named backend section.

Restart the cinder service.

Initialization Errors and `DriverNotInitialized` Exceptions

Driver initialization errors and **`DriverNotInitialized`** Exceptions occur if the volume driver encounters an exception during startup.

Background

After a driver initialization error, the Cinder volume service does not attempt any further actions with the driver. When you perform a volume management action such as creating a new volume, the action may fail with an exception of type **`DriverNotInitialized`**. This exception in the log indicates that there was an earlier failure during driver initialization.

Symptoms

All attempts to perform volume management actions using one driver instance fail. The following may occur:

- New volumes immediately appear with status **`error`**.

- Volumes that are deleted do not disappear but appear with status **`error_deleting`**.

Admin information

The **`DriverNotInitialized`** exception appears in the Cinder volume service's log in **`/var/log/cinder/`** on the node where the driver is running. Look earlier in the log for **`PureHTTPError`** and **`PureDriverException`**.

To search for the root cause of the problem, look in the log history around the time of the most recent Cinder volume service restart.

Possible causes

- Unable to reach the Pure Storage REST API at the location specified by **`san_ip`**.
- Unable to establish a REST API session due to an invalid **`pure_api_token`** setting.
- The OpenStack node which is running the volume driver cannot reach an iSCSI-enabled port or Fibre Channel-enabled port on the FlashArray.

Recovery

- Examine Cinder logs for error messages.
- Verify that **`san_ip`** is set correctly in **`/etc/cinder/cinder.conf`**, and that the location indicated by **`san_ip`** is reachable.
- Verify that **`pure_api_token`** is set to a valid API token.

After resolving the problem, restart the Cinder volume service.

Note: For instructions to create or copy an API token, see *Copy API Tokens from Purity*. API tokens are also described in the FlashArray Users Guide.

Volume Creation Errors

Volume creation errors can occur after an attempt to create a new volume, for example after a command such as the following:

`cinder create <VOLUME-ID>` ...

Symptoms

The status of the volume appears as **`Error`**.

Attempts to attach the volume to any instances fail.

Admin information

Search for **`PureHTTPError`** in the Cinder volume service's logs in **`/var/log/cinder/`** on the node where the driver is running.

Possible causes

- The Cinder **`volume_name_template`** entry in **`/etc/cinder/cinder.conf`** is set to a value that results in names that are too long or contain invalid characters for Purity.
- An administrator deleted the source of the volume (a snapshot or another volume) in Purity.

Recovery

Diagnose the problem on the node where the driver is running. Look for the error message in a **`PureHTTPError`** in the logs in **`/var/log/cinder/`**.

If volume names are invalid, change the **`volume_name_template`** entry in **`/etc/cinder/cinder.conf`** to a valid value and restart the Cinder volume service.

Snapshot Creation Errors

Snapshot creation errors can occur after a snapshot is created from a volume, for example after a command such as the following:

```
cinder snapshot-create ...
```

Symptoms

The status of the snapshot appears as **`Error`**.

Admin information

Search for **`PureHTTPError`** in the Cinder volume service's logs in **`/var/log/cinder/`** on the node where the driver is running.

Possible causes

- The Cinder **`snapshot_name_template`** entry in **`/etc/cinder/cinder.conf`** is set to a value that results in names that are too long or contain invalid characters for Purity.
- An administrator deleted the source volume for the snapshot in Purity.

Recovery

Diagnose the problem by finding the error message in a **`PureHTTPError`** in the logs in **`/var/log/cinder/`** on the node where the driver is running.

- If snapshot names are invalid, change **`snapshot_name_template`** in **`/etc/cinder/cinder.conf`** to a valid value and restart the Cinder volume service.
- If there are too many snapshots, eradicate volumes in Purity to free space.

Retry the command or action.

Volume Attachment Errors

Volume attachment errors can occur after a volume is attached to an instance, for example after commands such as the following:

```
nova volume-attach ...
```
or
```
nova boot --boot-volume ...
```

Symptoms

The following may occur:

- A request to attach the volume fails, the volume remains in status **`available`** as if the request to attach the volume never occurred.
- After an attempt to boot a new instance from a volume, the instance's status is **`error`**.

Admin information

Search for **`PureHTTPError`** in the Cinder volume service's logs in **`/var/log/cinder/`** on the node where the driver is running.

Also check the logs in **`/var/log/nova/`** on the compute node that hosts the instances.

Possible causes

One of the following issues generates an error message in the logs in **`/var/log/cinder/`** on the node where the driver is running:

- There is no host object in Purity with the initiator IQN of the compute node hosting the instance.
- A storage administrator deleted the volume or host object in Purity.
- The volume and host object are already connected in Purity.
- The host credentials for CHAP were modified outside of Cinder.
- In Fibre Channel environments, there is a Fibre Channel network error such as a mis-configured zone or cables not connected to the correct ports.

Note that Cinder cannot detect the following issues and explanatory error messages are not logged in **`/var/log/cinder/`**:

- The iSCSI or Fibre Channel port that the volume driver selected on the FlashArray and returned to nova is not reachable from the compute node on which the instance is hosted. This problem is likely a network issue. For example, the iSCSI or Fibre Channel port the volume driver could reach is on a subnet that the compute node cannot reach. (See *Network Configuration*.)
- It is not possible to connect to a volume because multipathing services are not configured correctly on the OpenStack compute node on which the instance is hosted. (See the *Install and Configure Multipathing for the Nova Service* chapter.)

Recovery

Check the following:

- Ensure that Purity does not contain a host that uses CHAP but is not managed by Cinder.
- The volume still exists in Purity.
- The volume is not already connected to the host object.
- The same iSCSI-capable 10 Gbps Ethernet ports or Fibre Channel ports on the array are reachable from the compute node as are available from the node where the driver is running.
- In Fibre Channel environments, confirm the following:
  - The initiator has an online Fibre Channel HBA.
  - The initiator and the FlashArray's Fibre Channel ports have connectivity to the same Fibre Channel fabric or fabrics.

- The Fibre Channel fabric is configured correctly.

- Run the following command on the compute node hosting the instance to which you tried to attach the volume:

  ```
  multipath
  ```

  If the output includes warnings that the DM-multipath kernel driver is not loaded or that a configuration file is missing, verify that multipathing is installed and enabled on OpenStack compute nodes that connect to the FlashArray. See the *Install and Configure Multipathing for the Nova Service* chapter.

- There should not be stale iSCSI sessions that are in an error state. The system log should not contain any 1020 connection errors, and the LUN(s) should *not* be visible in **`/dev/disk/by-path/`** if the attach failed. If any of these occur, logout of the iSCSI session, delete the node, and remove any multipath device for them.

Retry the command or action.

Volume Manage or Unmanage Errors

Volume manage or unmanage errors can occur after Cinder commands such as the following:

`cinder manage ... name=myVol` or

`cinder ... unmanage myVol`

Symptoms

The following may occur:

- Cinder **`manage`** creates a volume but in an error state.
- Cinder **`unmanage`** does not remove a volume from Cinder and that volume is now in an error state.

Admin information

Look for errors, exceptions, and stack traces in the Cinder volume service's logs in **`/var/log/cinder/`** on the node where the driver is running.

Possible causes

- The key-value pair **`name=<`*`volumeName`*`>`** was not provided (as required by the Pure Storage driver).
- The volume could not be found on the array.
- The Cinder command fails because of a name collision (a volume already exists with the name of the final output of the **`manage`** or **`unmanage`** command).
- The volume type or backend name was not correctly passed in to the manage command for the Cinder scheduler to place the volume on the correct backend.

Recovery

Check the following:

- Look for errors, exceptions, and stack traces in the Cinder volume service's logs in **`/var/log/cinder/`** on the node where the driver is running.
- Verify that the command has the correct format, and includes a **`name=`**<***volumeName***> key-value pair.
- Verify that the name passed in the Cinder command exactly matches the volume name in Purity.
- Ensure there is not a volume with the same name as the final target's name, with **`-cinder`** or **`-cinder-unmanaged`** appended. (The final name strings appear in the volume log.)
- Verify the volume type will correctly schedule the "new" managed volume to be on the same backend.
- Verify the backend name is correctly specified.

  The format is **`host@backend#pool`**. For the PureISCSIDriver, the backend and pool names are always the same.

Retry the command.

Import Errors

REST API 1.4 or later version is required to import a new volume or snapshot into the Cinder service. The use of the **`cinder manage`** command with a FlashArray that is not running at least REST API 1.4 results in an error.

Symptoms

- The Cinder **`manage`** command fails to import the item.
- An error similar to the following is seen in the Cinder volume service log:

  `Unable to do manage snapshot operations with Purity REST API version 1.2, requires 1.4`.

Probable cause

The FlashArray is not running at least REST API 1.4.

Recovery

- Install Purity 4.1.0 or later release.
- Retry the command or action.

Appendix A. Configure Multiple Pure Storage FlashArrays

Overview

This appendix describes the use of multiple backend definitions to manage multiple FlashArrays.

Each named stanza contains settings for one driver instance, which manages one FlashArray. Each backend stanza must have unique **`pure_api_token`** and **`san_ip`** settings. The **`volume_backend_name`** setting, if used, also must be unique across all backend stanzas.

It is supported for some backends to use iSCSI while other backends use Fibre Channel (through different **`volume_driver`** settings).

Each backend definition can have different settings for the following optional parameters:

- **`use_chap_auth`**
- **`use_multipath_for_image_xfer`**
- **`replication_driver`**
- **`image_volume_cache_*`**
- **`pure_automatic_max_oversubscription_ratio`**
- **`max_over_subscription_ratio`** (a Cinder parameter)
- **`pure_eradicate_on_delete`**
- **`driver_ssl_cert_*`**
- **`filter_function`**
- **`goodness_function`**

If you are using Fibre Channel, all Fibre Channel backends must use the same Fibre Channel Zone Manager settings.

Example Configuration File Entries

The following example shows entries to manage three FlashArrays as Cinder backends. This example shows backend definitions for one Fibre Channel FlashArray and two iSCSI FlashArrays.

These entries appear in the **`/etc/cinder/cinder.conf`** configuration file on the OpenStack node where the Pure Storage Volume Driver is installed:

```
[other sections]
...
[DEFAULT]
...
enabled_backends = pure1_fc,pure2_iscsi,pure3_iscsi
```

```
...
verbose = True
...
cinder_internal_tenant_project_id = 123456...
cinder_internal_tenant_user_id = abcdef...

[pure1_fc]
volume_driver = cinder.volume.drivers.pure.PureFCDriver
san_ip = MANAGEMENT-INTERFACE-1
pure_api_token = API-TOKEN-1
use_multipath_for_image_xfer = True
volume_backend_name = pure1_fc
use_chap_auth = False
filter_function = "capabilities.total_volumes < 500"
image_volume_cache_enabled = True
image_volume_cache_max_size_gb = 20
image_volume_cache_max_count = 5
goodness_function = "100 * (1 / max(capabilities.usec_per_read_op, 1)) * (1
/ max(capabilities.usec_per_write_op, 1)) * (1 / max(capabilities.queue_dep
th, 1))"
pure_automatic_max_oversubscription_ratio = True

[pure2_iscsi]
volume_driver = cinder.volume.drivers.pure.PureISCSIDriver
san_ip = MANAGEMENT-INTERFACE-2
pure_api_token = API-TOKEN-2
use_multipath_for_image_xfer = True
volume_backend_name = pure2_iscsi
use_chap_auth = True
filter_function = "capabilities.total_volumes < 500"
image_volume_cache_enabled = True
image_volume_cache_max_size_gb = 200
image_volume_cache_max_count = 50
goodness_function = "100 * (1 / max(capabilities.usec_per_read_op, 1)) * (1
/ max(capabilities.usec_per_write_op, 1)) * (1 / max(capabilities.queue_dep
th, 1))"
pure_automatic_max_oversubscription_ratio = True
zoning_mode = ...
[fczonemanager]
...

[fabricfab1]
...

[fabricfab2]
...
```

```
[pure3_iscsi]
volume_driver = cinder.volume.drivers.pure.PureISCSIDriver
san_ip = MANAGEMENT-INTERFACE-3
pure_api_token = API-TOKEN-3
use_multipath_for_image_xfer = True
volume_backend_name = pure3_iscsi
use_chap_auth = True
filter_function = "capabilities.total_volumes < 500"
goodness_function = "100 * (1 / max(capabilities.usec_per_read_op, 1)) * (1
/ max(capabilities.usec_per_write_op, 1)) * (1 / max(capabilities.queue_dep
th, 1))"
image_volume_cache_enabled = False
pure_automatic_max_oversubscription_ratio = False
max_over_subscription_ratio = 30.0

...
[other sections]
```

where:

- ***MANAGEMENT-INTERFACE*** is the IP address of the FlashArray's management interface or a domain name that resolves to that IP address.
- ***API-TOKEN*** is the Purity authorization token of the FlashArray's dedicated array administrator for Cinder volumes or of a different Purity administrator. For more information, see *Copy API Tokens from Purity*.

Notes about backend stanzas:

- The **`enabled_backends`** entry names the backend stanzas in the **`cinder.conf`** file. The section names (**`pure1_fc`**, **`pure2_iscsi`**, and **`pure3_iscsi`** in this example) need to match only the **`enabled_backends`** entries.

  The **`enabled_backends`** names are not required to match any part of the FlashArray configuration.
- Each of the named sections contains settings for one driver instance which manages one FlashArray. Each section must have unique **`pure_api_token`** and **`san_ip`** settings.
- The **`volume_backend_name`** setting is optional and defaults to the **`volume_driver`** object name, such as **`PureISCSIDriver`** or **`PureFCDriver`**.
- The **`verbose = True`** entry can be removed or changed to **`False`** once the configuration is working well.
- The **`fc-zone-manager`** entry name is required when Fibre Channel is used.
- The **`volume_backend_name`** names **`pure1_fc`**, **`pure2_iscsi`**, and **`pure3_iscsi`** are examples only. Yours can be different.

- In multiple-array environments, a user-defined goodness function is recommended. The goodness_function seen here is an example only. See *Create a Goodness Function*.
- If Glance image support is enabled, all backends must use the same tenant project and user settings.

Parameters for Backend Stanzas

The following parameters are deprecated in the [DEFAULT] section:

- **volume_backend_name**
- **volume_driver**
- **san_ip**
- **pure_api_token**
- **use_multipath_for_image_xfer**
- **max_over_subscription_ratio**
- **pure_automatic_max_oversubscription_ratio**
- **filter_function**
- **goodness_function**
- **image_volume_cache*** Glance image settings
- **pure_eradicate_on_delete**
- **replication_device**
- **driver_ssl_cert_verify**, **driver_ssl_cert_path**
- **use_chap_auth**

These parameters related to the driver are allowed in the [DEFAULT] section:

- **verbose**
- **enabled_backends**
- **scheduler_default_filters**
- **cinder_internal_tenant_user_id**, **cinder_internal_tenant_project_id**
- **fc-zone-manager** and FC fabric settings

Appendix B. Configure Tiered Volume Types

The section describes how to define tiered volume types. The example shows the configuration to manage two different volume drivers as Cinder backends.

First the Cinder configuration for the backends is shown. Then the Cinder commands to create the tiers are shown.

Pure Storage FlashArray Backend

The following entries appear in the **`/etc/cinder/cinder.conf`** configuration file on the OpenStack node where the Pure Storage Volume Driver is installed:

```
[DEFAULT]
...
enabled_backends = pure1
...
verbose = True
...
[pure1]
volume_driver = cinder.volume.drivers.pure.PureISCSIDriver
san_ip = MANAGEMENT-INTERFACE
pure_api_token = API-TOKEN
use_multipath_for_image_xfer = True
volume_backend_name = Pure_iSCSI

...
[other sections]
```

where:

- ***MANAGEMENT-INTERFACE*** is the IP address of the FlashArray's management interface or a domain name that resolves to that IP address.
- ***API-TOKEN*** is the Purity authorization token of the FlashArray's dedicated array administrator for Cinder volumes or of a different Purity administrator. For more information, see *Copy API Tokens from Purity*.

Notes:

- The **`enabled_backends`** entry name (**`pure1`** in this example) must match a section name (**`[pure1]`**) in the **`cinder.conf`** file.
- The section name and the **`enabled_backends`** entry must match each other but do not need to match any part of the FlashArray configuration.

LVM Backend

The following entries appear in the **/etc/cinder/cinder.conf** configuration file on the OpenStack node where Cinder's default logical volume management (LVM) driver is installed:

```
[DEFAULT]
...
enabled_backends = LVM
...
[LVM]
volume_driver = cinder.volume.drivers.lvm.LVMISCSIDriver
volume_backend_name = LVM_iSCSI
[other entries as required for the LVM driver]
...
[other sections]
```

Cinder Commands

These examples create two tiers and set the tiers to the pre-configured FlashArray and LVM backends.

```
$  cinder type-create ssd-tier
+--------------------------------------+-------------------------------+
|                  ID                  |    Name                       |
+--------------------------------------+-------------------------------+
| 2c57e348-8ac5-43d5-8c25-e31aaa902ec9 | ssd-tier                      |
+--------------------------------------+-------------------------------+
$  cinder type-key ssd-tier set volume_backend_name=Pure_iSCSI
$
$  cinder type-create lvm-tier
+--------------------------------------+-------------------------------+
|                  ID                  |     Name                      |
+--------------------------------------+-------------------------------+
| aa6e11dd-5e05-4818-ab19-b11504289037 | lvm-tier                      |
+--------------------------------------+-------------------------------+
$  cinder type-key lvm-tier set volume_backend_name=LVM_iSCSI
```

Notes:

- The tier name in a **type-create** command must match a tier name following the **type-key** attribute in a **set volume_backend_name** command.
- The **volume_backend_name** in a **set volume_backend_name** command must match a **volume_backend_name** entry for a valid backend in a Cinder configuration file.

Appendix C. Purity Administrator Accounts in Active Directory

This section describes example security group names in Active Directory for the three levels of Purity administrators.

You can skip this section if your environment already has Purity administrator accounts configured or if you use only the built-in **`pureuser`** administrator account.

Note: If you create a dedicated account for your Cinder objects (as recommended), create that account in the *array administrator* security group.

Background

The built-in **`pureuser`** administrator account does not provide adequate support for production environments. Separate administrator accounts allow for different roles and provide a clear record in Purity's audit log of which administrator performed various actions.

In the current Purity release, administrator accounts cannot be created through Purity and must be configured in Active Directory.

Example Active Directory Names

Each administrator level (read-only, storage, and array) requires a different security group Common Name (CN) in Active Directory. All CNs for Purity administrators must exist in the same Organization Unit (OU).

Example Distinguished Names for security groups for the three levels of Purity administrators are shown below. The names of these groups are examples only.

- **`CN=purereadonly,OU=PureGroups,OU=SAN,OU=IT,OU=US,DC=sample,DC=com`**
- **`CN=purestorageadmins,OU=PureGroups,OU=SAN,OU=IT,OU=US,DC=sample,DC=com`**
- **`CN=purearrayadmins,OU=PureGroups,OU=SAN,OU=IT,OU=US,DC=sample,DC=com`**

Note that all three groups are contained within the same OU (**`OU=PureGroups,OU=SAN,OU=IT,OU=US,DC=sample,DC=com`** in this example).

Dedicated Array Administrator

To create a dedicated account for your Cinder objects, create that account in the *array administrator* security group (**`CN=puresarrayadmins`** in the example above).

Appendix D. Configuration Instructions in the Purity GUI

This sections lists the steps in the Purity GUI. These steps are equivalent to the CLI steps described in the configuration chapter.

Create a Purity API Token (GUI)

1. In Purity, select **System > Users > Local Users**.
2. Hover over the username and click the settings icon.
3. Select **Create API Token**.
4. The API Token pop-up displays the new token. Copy the token and click **OK**.

Link back to *Copy API Tokens from Purity*.

Copy an Existing Purity API Token (GUI)

1. In Purity, select **System > Users > API Tokens**.
2. Hover over the username and click the settings icon.
3. Select **Show API Token**. Copy the token from the API Token pop-up.

Link back to *Copy API Tokens from Purity*.

Appendix E. Efficiently Create Multiple Volumes from an Image

Create the First Copy of a Volume

The following Cinder command creates a volume from an image:

```
cinder create –-image-id ...
```

This command is recommended only for the first copy of a volume from an image but is not efficient to create additional volumes from the same image. Each additional volume would take the same amount of time as the initial volume.

After the first volume is created from the image, a snapshot or clone is more efficient to create additional volumes from the same image. The following sections describe recommended ways to create additional volumes from the original image.

Clone the First Volume

One alternative is to clone the first volume. Run the following command to create the clone:

```
cinder create VOLUME-ID ...
```

Repeat to create as many volumes as needed.

Disadvantage

In order to be a copy of the original image, the clone must be created before any changes to the original volume.

Advantage

Cloning completes very quickly and does not require copying the image's or volume's data to a new location.

Create a Snapshot of the First Volume

This alternative uses a snapshot of the first volume. Run the following commands to create the snapshot and create volumes from the snapshot:

1. `cinder snapshot-create ...` (once)
2. `cinder create --snapshot-id ...` (as needed)

Repeat step 2 to create as many volumes as needed, for the life of the snapshot.

Disadvantage

As long as the original snapshot is need for additional copies, it is occupying space on the volume.

Advantages

- With this approach, volume creation completes very quickly and does not require copying the image's or snapshot's data within the array.
- For the lifetime of the snapshot, you can continue to create volumes that are identical to the original image.
- The original volume can be used and modified as required.

Appendix F. Import a Volume or Snapshot into Cinder

If you use Purity to copy a volume or snapshot that the Cinder service manages, by default the Cinder service does not manage the new volume or snapshot. Optionally use the **`cinder manage`** command to import the new volume or snapshot into the Cinder service.

Partial examples:

- **`cinder --id-type name manage`** ***`BACKEND VOL-NAME`*** **`...`**
- **`cinder --id-type name manage`** ***`BACKEND SNAP-NAME`*** **`...`**

See OpenStack documentation for complete syntax: *Block Storage command-line client* [http://docs.openstack.org/cli-reference/content/cinderclient_commands.html#cinderclient_subcommand_manage_v2]. The commands are listed in the "Block Storage API v2 commands" section.

Our volume driver requires the **`name=`*`VOL-NAME`*** or **`name=`*`SNAP-NAME`* key-value pair as a parameter.

Notes:

- When a volume or snapshot is imported into the Cinder service, it is renamed in the Cinder service's format. Example of an imported volume: **`volume-`**<***`volume-UUID`***>**`-cinder`**.
- The **`cinder unmanage`** command removes a volume or snapshot from the control of the Cinder service.
- An unmanaged volume's name has the format **`volume-`**<***`volume-UUID`***>**`-cinder-unmanaged`**.
- The Purity REST API 1.4 or later release is required for the Cinder import feature. An error is seen if an import request is attempted with an earlier version of the REST API.
- The command **`cinder get-pools`** displays backend names:

```
cinder get-pools
+----------+--------------------------------------------------------------+
| Property | Value                                                        |
+----------+--------------------------------------------------------------+
| name     | cinder-init1.dev.purestorage.com@cinder-fad1#cinder-fad1 |
+----------+--------------------------------------------------------------+
```

- An example **`cinder manage`** command and its output:

```
cinder manage --id-type name --name managed-vol cinder-init1.dev.purestorage.co
m@cinder-fad1#cinder-fad1 cinder-unmanaged
+--------------------------------+-----------------------------------------
| Property                       | Value
+--------------------------------+-----------------------------------------
| attachments                    | []
| availability_zone              | nova
```

```
| bootable                       | false
| consistencygroup_id            | None
| created_at                     | 2017-05-30T19:01:32.000000
| description                    | None
| encrypted                      | False
| id                             | 578e740f-aa0a-4d06-9fd2-d46ca06e2cbf
| metadata                       | {}
| migration_status               | None
| multiattach                    | False
| name                           | managed-vol
| os-vol-host-attr:host          | cinder-init1.dev.purestorage.com@cinder-fad1
                                   #cinder-fad1
| os-vol-mig-status-attr:migstat | None
| os-vol-mig-status-attr:name_id | None
| os-vol-tenant-attr:tenant_id   | 0fd0fd3739aa4529baa2587b5cc69eac
| replication_status             | None
| size                           | 0
| snapshot_id                    | None
| source_volid                   | None
| status                         | creating
| updated_at                     | None
| user_id                        | 48eb701cfeb54a259b6a74d1665fddb7
| volume_type                    | None
+--------------------------------+--------------------------------------
```

Appendix G. Deprecated Configuration Items

This appendix lists configurations that have been deprecated in recent releases of OpenStack.

Volume driver configuration in the [DEFAULT] section

Volume driver configuration entries are no longer supported in the **`[DEFAULT]`** section of the **`cinder.conf`** configuration file.

Beginning in the Newton release, these entries generate a warning.

Beginning in the Ocata release, these entries generate the following error and prevent the cinder service from starting.

```
Configuration for cinder-volume does not specify "enabled_backends".
Using the DEFAULT section to configure drivers is not supported since Ocata.
```

Consistency groups

Consistency groups are being replaced by generic volume groups that have a consistent snapshot type.

See the OpenStack article ***Migrate CGs to Generic Volume Groups*** [http://docs.openstack.org/developer/cinder/devref/groups.html#migrate-cgs-to-generic-volume-groups] for information on this change.

Existing consistency groups are supported in the Ocata release but are expected to generate an error in future releases.

Appendix H. Controller? Host?

This section describes terminology conflicts around *controller* and *host* and how these terms are used in OpenStack, in Purity, and in this document. The OpenStack usage is taken from the OpenStack Glossary [http://docs.openstack.org/glossary/content/glossary.html].

Controller

OpenStack

In OpenStack, the terms *cloud controller node* and *controller node* refer to an OpenStack cluster node that runs network, volume, API, scheduler, and image services. Each service may be broken out into separate nodes for scalability or availability.

Purity

Purity uses the term *controller* to refer to one of a FlashArray hardware component. Each FlashArray contains two controllers, which run the Purity Operating Environment.

This document

This document uses the terms *cloud controller node* and *OpenStack controller node* to refer to a controller node on the OpenStack cluster.

This document uses the terms *FlashArray controller* or *controller in the FlashArray* to refer to the FlashArray hardware component.

Host

OpenStack

In OpenStack usage, a *host* is a physical computer, as opposed to a VM instance.

In Nova usage, a *host* typically is a compute node that hosts VMs.

In Cinder usage, a *host* typically is an available backend with a volume driver instance.

Purity

Purity uses the term *host* to refer to a server or desktop computer that makes use of the FlashArray's data storage services. Hosts may be physical or virtual, and are identified to Purity by the port names used to connect them to the FlashArray system.

This document

This document uses the noun *host* only in the Purity sense, as a Purity host object.

Appendix I. Resources

OpenStack Documentation

- ***OpenStack Cinder project wiki*** [https://wiki.openstack.org/wiki/Cinder]
- ***OpenStack documentation*** [http://docs.openstack.org/]
- ***Generic Volume Groups*** [http://docs.openstack.org//developer/cinder/devref/groups.html#migrate-cgs-to-generic-volume-groups]

Pure Storage Documentation and Online Resources

The following resources (and more) are available from Pure Storage:

- ***Knowledge Base*** [http://support.purestorage.com]
  - ***OpenStack-related Knowledge Base articles*** [https://support.purestorage.com/Solutions/Cloud/OpenStack]
    For Linux users, we recommend the following Knowledge Base articles:
    - *Linux Recommended Settings* [https://support.purestorage.com/Solutions/Operating_Systems/Linux/Linux_Recommended_Settings]
    - *Configuring iSCSI on Linux for the Pure FlashArray* [https://support.purestorage.com/Solutions/Operating_Systems/Linux/Configuring_iSCSI_on_Linux_for_the_Pure_FlashArray]

    FlashArray and Purity Knowledge base articles [https://support.purestorage.com/FlashArray]
  - Documentation and user guides, for instance: ***Purity FlashArray User Guides*** [https://support.purestorage.com/FlashArray/PurityOE/FlashArray_User_Guide]
- ***Pure Storage Community*** [http://community.purestorage.com]
  - ***User forums*** [http://community.purestorage.com/t5/Forums/ct-p/forums]

A Note about Document Conventions

This document follows Purity CLI man page conventions:

- Constant-width (Courier) font is used for text that is to be entered verbatim (e.g., command and subcommand names such as **`pureadmin list`**).
- Uppercase italic text represents entered text whose content is based on the nature of the subcommand (e.g., --api-token ***USER***, where ***USER*** represents the value to be entered, in this case "pureuser" or other Purity administrator account name).

Documentation Feedback

We welcome your feedback about Pure Storage documentation and encourage you to send your questions and comments to **<DocumentFeedback@purestorage.com>**. We would love to hear from you.

Legal Information

End-User Agreement

End User Agreement [http://www.purestorage.com/agreements/Pure_enduser_agreement.pdf]

Notices

Trademarks

Copyright

Pure Storage, Inc.
Twitter: @purestorage

650 Castro Street, Suite 260
Mountain View, CA 94041
800-379-7873

Sales: sales@purestorage.com
Support: support@purestorage.com
Media: pr@purestorage.com
General: info@purestorage.com