PURESTORAGE

Pure Storage OpenStack (Liberty) Cinder Driver Best Practices

Simon Dodsley, OpenStack Solutions Architect

Version 2.0, 30 November 2015

Contents

Executive Summary

OpenStack is an open source cloud computing platform designed as an Infrastructure as a Service (IaaS) implementation. It has the ability to control large numbers of compute, network and storage resources.

The block storage portion of OpenStack is referred to as the Cinder project and aims to provide persistent block level storage to the OpenStack compute (Nova) instances. The Pure Storage Cinder driver is a Python-based module integrated into the main OpenStack distributions.

This document describes the best practices for using the Pure Storage OpenStack Liberty Cinder driver and details specific configuration recommendations to get the best from the Pure Storage Cinder driver. It is a subset of the **Pure Storage OpenStack Cinder Volume Driver Setup Guide** available from the Pure Storage Community website.

This document assumes the use of the OpenStack Liberty 2015.2.0.

Audience

This document describes how to configure an OpenStack Cinder node with a Pure Storage FlashArray providing the shared storage. The intended audiences for this document include the following:

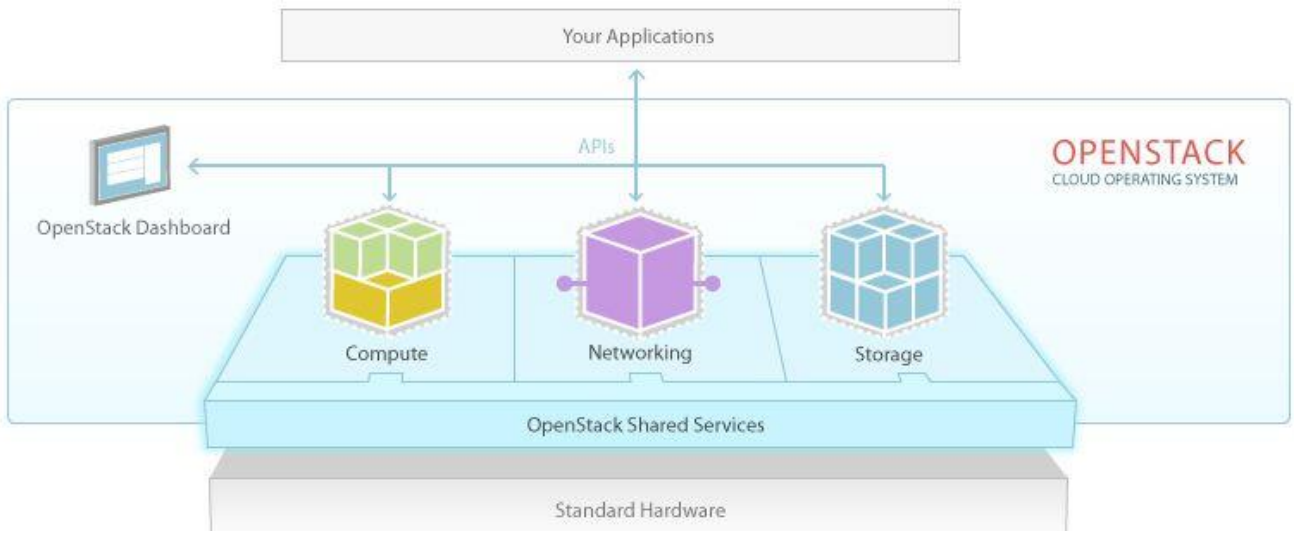
- OpenStack administrators, who configure and use OpenStack Cinder
- Pure Storage system administrators, who configure and use the Purity Operating Environment
- Pure Storage partners and solutions engineers, who support the Pure Storage FlashArray Volume Driver for OpenStack Cinder

This guide assumes familiarity with storage and networking concepts, OpenStack administration, Cinder administration, the Purity Operating Environment, and Pure Storage's iSCSI and Fibre Channel best practices

OpenStack Architecture

Openstack provides services to allow storage, compute and network resources to be connected together to form a cloud using independent programs.

The high level architecture is shown here.



At the time of writing these are the OpenStack services and associated project names:

Services	OpenStack Project
Object Storage	Swift
Compute	Nova
Image	Glance
Dashboard	Horizon
Identity	Keystone
Network	Neutron
Block Storage	Cinder
Telemetry	Ceilometer
Orchestration	Heat
Database	Trove
Data Processing	Sahara
Bare Metal	Ironic

Cinder Architecture

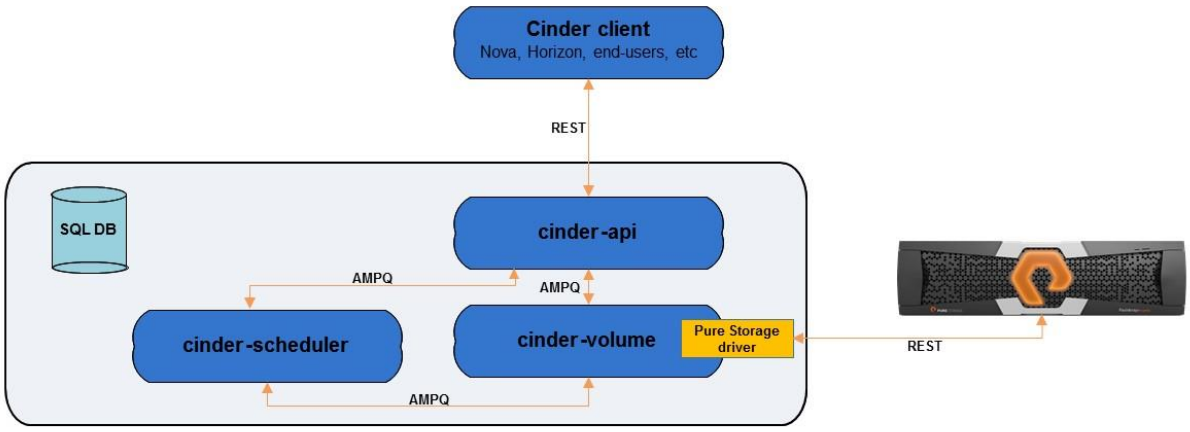
Cinder is the Block Storage provider to OpenStack and relies on 3 main components: Cinder-API, Cinder-Scheduler and Cinder-Volume. Communication between these components is achieved using and advanced message queue (AMQP). Other projects within OpenStack communicate with Cinder using REST protocols. Cinder communicates with other projects using their own REST protocol.

The Cinder-API service provides the RESTful API for the major volume operations, such as create, delete, snapshot, expand, etc.

The API service sends requests using AMQP to the Scheduler service. This then ensures that sufficient resources are available to service before dispatching the requests to the Volume service. There will be a Volume service instance for each Cinder backend configured in the environment.

In the case of a backend array from Pure Storage the Cinder-Volume instance will load the Pure Storage Cinder driver to control and communicate with the backend array.

A high level view of this can be seen below:



Pure Storage Cinder Driver

Supported Versions

The Pure Storage Cinder driver is supported for all Pure Storage Flash Array models running Purity 4.0.0 or higher.

Getting the Pure Storage Cinder driver

Pure Storage have contributed a Cinder driver since the official Juno release of OpenStack project. Any OpenStack release from Juno onwards will incorporate the Pure Storage Cinder driver.

Installation

The Pure Storage Cinder driver is available as a standard driver under the Cinder Volume service. Please refer to the installation guide of your specific OpenStack deployment, or the main OpenStack website (http://www.openstack.org), to ensure you have the Cinder project correctly installed.

After ensuring that your OpenStack deployment has the Cinder Volume service correctly installed it is necessary to additionally install the Pure Storage Python SDK kit. This is publically available and is installed using the following command:

```
pip install purestorage
```

Ensure that you have the *python-pip* package installed prior to running this command, using the appropriate package installation procedures for your Linux distribution. The Pure Storage Python SDK has a pre-requisite of the *requests* Python package, so ensure that this is available.

NTP Configuration

It is recommended that the Pure Storage Flash Array and all your OpenStack controller and compute nodes are configured to use NTP.

If the array and OpenStack nodes are significantly out of sync you may experience error messages in the Cinder volume controller and REST logs that show as 401 errors.

MPIO Configuration

To ensure multipath volume accessibility for all volumes presented from the Pure Storage Flash Array it is essential to perform the following steps:

- Install the `device-mapper-multipath` or `multipath-tools` package (depending on your Linux distribution) on nodes running the Cinder Volume service and the Nova Compute service
- Install `sysfsutils` and `sg3_utils` on nodes running both the Cinder Volume service and the Nova Compute service
- Set `iscsi_use_multipath = True` in `/etc/nova/nova.conf` for any nodes using iSCSI as the Cinder block protocol driver
- Set `use_multipath_for_image_xfer = True` in `/etc/cinder/cinder.conf` for each

Cinder volume service node

Modify the multipathing software configuration file contains information required for the Pure Storage Flash Array. Example entries in the `/etc/multipath.conf` configuration file are shown here:

```
defaults {
    polling_interval      10
}

devices {
    device {
        vendor                "PURE"
        path_selector         "round-robin 0"
        path_grouping_policy  multibus
        rr_min_io_rq          1
        path_checker          tur
        fast_io_fail_tmo      10
        dev_loss_tmo          60
        no_path_retry         0
    }
}
```

Finally ensure that the multi-pathing software service is enabled and running.

Cinder policy.json Configuration

To enable the use of Cinder volume consistency groups, which Pure Storage leverage using our Protection Groups, it is necessary to change policies in the consistency groups APIs in the `/etc/cinder/policy.json` file which, by default are disabled. To enable this functionality edit the policy file and remove `group:nobody` from the following lines:

```
"consistencygroup:create": "group:nobody",
"consistencygroup:delete": "group:nobody",
"consistencygroup:get": "group:nobody",
"consistencygroup:get_all": "group:nobody",
"consistencygroup:create_cgsnapshot" : "group:nobody",
"consistencygroup:delete_cgsnapshot": "group:nobody",
"consistencygroup:get_cgsnapshot": "group:nobody",
"consistencygroup:get_all_cgsnapshots": "group:nobody",
```

So that the edited result is:

```
"consistencygroup:create": "",
"consistencygroup:delete": "",
"consistencygroup:update": "",
"consistencygroup:get": "",
"consistencygroup:get_all": "",
"consistencygroup:create_cgsnapshot" : "",
"consistencygroup:delete_cgsnapshot": "",
"consistencygroup:get_cgsnapshot": "",
"consistencygroup:get_all_cgsnapshots": "",
```


Consistency group operations, along with a few other newer cinder commands can only be performed from the command line using version 2 of the Cinder API. To ensure that these operations are successful add the following line to a login script, or your `~/keystonerc_<user>` file:

```
export OS_VOLUME_API_VERSION=2
```

Modify the Cinder Configuration File

As detailed in the main **Pure Storage Cinder Driver Setup Guide** the `/etc/cinder/cinder.conf` file must include settings to allow the Cinder Volume service to communicate with and control the Pure Storage Flash Array. To enable this to happen add the following entries to the `[DEFAULT]` section

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

san_ip=<MANAGEMENT-INTERFACE>
pure_api_token=<API-TOKEN>
```

where

MANAGEMENT-INTERFACE :
The IP address for the FlashArray's management interface virtual IP address or the fully qualified domain name assigned to both array controllers. This is required to maintain high availability and support non-disruptive firmware updates to the array.

API-TOKEN :
The Purity API token that the volume driver uses to perform volume management on this FlashArray. More details on how to obtain the API Token are given in the **Configuring OpenStack (Kilo) with the Pure Storage Flash Array** document available from the Pure Storage website (http://www.purestorage.com/resources/).

NOTE: If you are using the Fibre Channel driver then it will also be necessary to add the following line to the `[DEFAULT]` section of the cinder configuration file:

```
mode=fabric
```

and also add `[fc-zone-manager]` and `[fabric-x]` sections to the configuration file as well. For more details on this please refer to the OpenStack documentation from your fabric switch vendor.

Restarting OpenStack Services

After modifying any OpenStack configuration file it is required to restart the associated service to ensure your modifications are accepted and implemented.

Other Configuration Requirements

Pure Storage also recommend additional configuration changes to HBA and Queue setting to optimize for high performance flash-based storage.

For more details on these setting and also more details on the Multipath configuration mentioned above please refer to the **Linux Initiator Setting – Best Practices** article available in the Pure1 Community website (http://community.purestorage.com).

Advanced Functionality

Thin and Over-Provisioning Support

As the Pure Storage Flash Array is fully thin-aware and uses industry leading data reduction technologies we can allow for massive over-provisioning of the array.

Due to the architectural design of the Pure Storage Flash Array we recommend leaving the `reserved_percentage` at the default value of 0 in the `/etc/cinder/cinder.conf` file.

Using Multiple Cinder Backends

The previous sections have assumed that there is only one Cinder backend. In reality there may be a number of backends available to the OpenStack administrator, either of the same type, or from different vendors.

If the multiple backends are exclusively using the same Pure Storage Cinder driver then only the following modifications would need to be made to the above recommendations:

Instead of putting the array configuration parameters in the `[DEFAULT]` section, each backend requires its own section with the `/etc/cinder/cinder.conf` file. The following example shows two backend arrays, each with its own section.

```
[Pure-1]
volume_backend_name=pure-1
volume_driver=cinder.volume.drivers.pure.PureISCSIDriver
san_ip=10.209.112.203
pure_api_token=960e185a-255f-4bdb-3b03-3b9eb1e5cf1f

[Pure-2]
volume_backend_name=pure-2
volume_driver=cinder.volume.drivers.pure.PureFCDriver
san_ip=10.209.112.201
pure_api_token=12de56fa-04d5-fde4-22ed-23ed409fab65
```

Notice that each section has a new parameter, `volume_backend_name`. These are unique to each backend and are used to tell Cinder which backends are enabled (you may have backends listed which are not enabled for use if required).

To inform Cinder which backends are actually enabled modify the `/etc/cinder/cinder.conf` file and set the parameter `enabled_backends` with comma-separated backend names. For example:

```
enabled_backends=pure-1, pure-2
```

If you have backends using different drivers, i.e. different vendor arrays, you need to check whether certain configuration options are supported, or are going to be implemented across one or all backends.

Any parameter set globally in the [DEFAULT] section will be interpreted by all drivers, unless overridden in their section. Examples of parameters that may, or may not, be supported by different vendors, and therefore may need be added to the section for the specific backend are:

- use_multipath_for_image_xfer
- use_chap_auth
- reserved_capacity

It is recommended that any backend sections are added to the end of the cinder.conf file thereby ensuring that no default values are inadvertently read into the backend driver section, effectively making them invisible to the default section.

Enabling TRIM / SCSI UNMAP

Currently OpenStack can support TRIM / SCSI UNMAP on Pure Storage hosted Cinder boot devices created with appropriately configured Glance images.

To enable this functionality it is necessary to perform two additional configuration changes:

- Modify the Nova Configuration File

  Within the [libvirt] section of the /etc/nova/nova.conf configuration modify the hw_disk_discard entry to be the following:

  ```
  hw_disk_discard = unmap
  ```

- Modifying Glance Images

  To allow the Nova instances to use the virtio-scsi block storage driver and provide discard support it is necessary to set properties for the Glance images to be used. This is performed by issueing the following command against each Glance image:

  ```
  glance image-update --property hw_scsi_model=virtio-scsi --property
  hw_disk_bus=scsi <glance_image_id>
  ```

  The glance image ID can be obtained from the command glance image-list.

It is important to note that although these settings will enable the ability to use the discard feature for the boot device, subsequent devices attached to the Nova instance will not be automatically mounted with the discard flag, even if the devices come from a TRIM capable Cinder backend array. This functionality is expected to become available for the Mikata release of OpenStack.

Glance Image Cache for Cinder

The Glance Image Cache for Cinder is by default disabled, therefore it is recommended to enable this when using a Pure Storage array as a Cinder backend and there are significant improvements in instance creation times that can be achieved by using this functionality.

A pre-requisite for this feature is the creation of an OpenStack project/tenant and user to act as the internal owner of the image cache volumes. This is done with the following commands:

```
# keystone tenant-create --description "Cinder Internal Tenant Project" --name
cinder_internal_tenant
# keystone user-create --pass USER_PASS --name cinder_internal_user
```

Once these are created use the created project/tenant and user IDs to add the following (example) entries to the `cinder.conf` file

```
[DEFAULT]
...
cinder_internal_tenant_project_id = PROJECT_ID
cinder_internal_tenant_user_id = USER_ID

[PURE]
...
image_volume_cache_enabled = True
image_volume_cache_max_size_gb = 200
image_volume_cache_max_count = 50
```

Note that a value of 0 for these parameters mean unlimited.

More details on the benefits and use of this feature with a Pure Storage array are detailed in a whitepaper specifically dealing with this subject entitled **OpenStack Liberty: A Look at the Glance Image-Cache for Cinder** that is available from the Pure Storage website (http://www.purestorage.com/resources/).

Storage Quality of Service

Whilst Pure Storage arrays inherently provide volumes with very high performance in respect of IOPs and bandwidth capabilities, there are certain circumstances where an administrator may wish to apply Quality of Service limitations to specific volumes presented from a Pure Storage array to an instance. Natively there is no Quality of Service tuning available within the Pure Storage array, but within OpenStack there is Storage Quality of Service functionality which can be leveraged to achieve the desired result, although this currently only works when using KVM or QEMU as your hypervisor. The options available for QoS limits are:

- `total_bytes_sec`: the total allowed bandwidth for the guest per second
- `read_bytes_sec`: sequential read limitation
- `write_bytes_sec`: sequential write limitation
- `total_iops_sec`: the total allowed IOPS for the guest per second
- `read_iops_sec`: random read limitation
- `write_iops_sec`: random write limitation

To apply QoS to Pure Storage volumes perform the following steps:

1. Create a Cinder Quality of Service for each QoS level you require

```
# cinder qos-create low-iops consumer="front-end" read_iops_sec=2000
write_iops_sec=1000
+----------+------------------------------------------------------------+
| Property |                          Value                             |
+----------+------------------------------------------------------------+
| consumer |                        front-end                           |
|    id    |          bd9c4200-d216-4320-a3f0-675f92a7b7c7              |
|   name   |                         low-iops                           |
|  specs   | {u'write_iops_sec': u'1000', u'read_iops_sec': u'2000'}    |
+----------+------------------------------------------------------------+
```


2. Create a new volume type for each QoS level

```
# cinder type-create low-iops
+--------------------------------------+----------+-------------+-----------+
|                  ID                  |   Name   | Description | Is_Public |
+--------------------------------------+----------+-------------+-----------+
| 26baaa59-24ca-43ad-b3c7-94cab636f36b | low-iops |      -      |    True   |
+--------------------------------------+----------+-------------+-----------+
```

3. Associate the new volume type to the appropriate QoS level

```
# cinder qos-associate bd9c4200-d216-4320-a3f0-675f92a7b7c7 26baaa59-24ca-43ad-b3c7-
94cab636f36b
# cinder create --display-name low-iops --volume-type low-iops 1
+---------------------------------------+--------------------------------------+
|                Property               |                Value                 |
+---------------------------------------+--------------------------------------+
|              attachments              |                  []                  |
|           availability_zone           |                 nova                 |
|               bootable                |                false                 |
|          consistencygroup_id          |                 None                 |
|              created_at               |      2015-11-30T14:48:36.000000      |
|             description               |                 None                 |
|              encrypted                |                False                 |
|                  id                   | 89f3d3d4-e498-46f3-8dc1-358923457481 |
|              metadata                 |                  {}                  |
|             multiattach               |                False                 |
|                  name                 |               low-iops               |
|         os-vol-host-attr:host         |                 None                 |
|     os-vol-mig-status-attr:migstat    |                 None                 |
|     os-vol-mig-status-attr:name_id    |                 None                 |
|      os-vol-tenant-attr:tenant_id     |   e6b9d4b0a58e4586af88cbfff6d5b02c   |
|   os-volume-replication:driver_data   |                 None                 |
| os-volume-replication:extended_status |                 None                 |
|           replication_status          |               disabled               |
|                 size                  |                  1                   |
|             snapshot_id               |                 None                 |
|             source_volid              |                 None                 |
|                status                 |               creating               |
|               user_id                 |   f1528c46b89a43ea80414d7b63932bc5   |
|             volume_type               |               low-iops               |
+---------------------------------------+--------------------------------------+
```

When this volume is now attached to a Nova instance the hypervisor will control the IOPs and bandwidth to this specific disk.

You can have multiple disks, each with a different QoS level attached to a single instance if required.

If the administrator needs to change the QoS level of a volume then this can be dome using the `cinder retype` command available within OpenStack.

Troubleshooting

If you are experiencing any issues in OpenStack, whether it be with a Cinder driver or another module with the project, the first and best method of root cause analysis is by investigating the OpenStack log files.

Enhanced logging is available in all modules of OpenStack by setting the `verbose` and `debug` parameters to `true` in the module configuration files.

Examples of configuration files that can be relevant to issues with Cinder drivers and their interaction with other modules are `/etc/cinder/cinder.conf`, `/etc/nova/nova.conf`, `/etc/glance/glance-api.conf` and `/etc/glance/glance-registry.conf`. You must restart these services if you modify the configuration files.

All logs will be created in a module specific subdirectory from `/var/log`. For example, the Cinder log files will be found in `/var/log/cinder`.

Additional troubleshooting documentation can be found on the Pure Storage Community website.

References

The following references were used as part of the development of this document.

- Pure Storage FlashArray OpenStack Cinder Volume Driver 2.0.0 Setup Guide
  [http://community.purestorage.com/ekgav24373/attachments/ekgav24373/pure-storage-knowledge/267/1/FAVolumeDriverForCinder_SetupGuide_2.0.0.pdf]
- Configuring OpenStack (Kilo) with the Pure Storage FlashArray
  [http://info.purestorage.com/ConfiguringOpenStackwiththePureStorageFlashArray_Request.html]
- Pure Storage Linux Initiator iSCSI Best Practises
  [http://community.purestorage.com/t5/Pure-Customer-Knowledge-Base/Linux-Initiator-Settings-Best-Practices/ta-p/2]
- OpenStack Liberty: A Look at the Glance image-Cache for Cinder
  [insert link here]

About the Author

As Global Solutions Architect, Simon is helping implement OpenStack technologies on Pure Storage. Core items include best practices, reference architectures and configuration guides.

With over 25 years of storage experience across all aspects of the discipline, from administration to architectural design, Simon has worked with all major storage vendors' technologies and organizations, large and small, across Europe and the USA as both customer and service provider. He also specializes in Data Migration methodologies assisting customers in their Pure Storage transition.

Blog: http://www.purestorage.com/blog/author/simon

PURESTORAGE

Pure Storage, Inc.
Twitter: @purestorage
www.purestorage.com

650 Castro Street, Suite #260
Mountain View, CA 94041

T: 650-290-6088
F: 650-625-9667

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