Installation runbook for Hedvig + Cinder Driver

Partner Name:	Hedvig Inc.
Product Name:	Hedvig Distributed Storage Platform
Product Version:	V-1.0.0
MOS Version:	Kilo on Ubuntu 14.04 (2015.1.0-7.0)
OpenStack version:	Kilo
Product Type:	Software-define storage

Contents

Document History

Version	Revision Date	Description
0.1	**09-12-2015**	**Initial Version**

1. Introduction

This document serves as a detailed Deployment Guide for the Hedvig cinder driver. The Hedvig Distributed Storage Platform provides a single, unified storage solution supporting block, file and object protocols. Hedvig software deploys on-premise and in public clouds, creating an implicitly hybrid storage system. A full set of APIs enable developers to build self-service portals and custom integrations. This document describes the reference architecture, installation steps for validation of MOS + Hedvig cinder driver, limitations and testing procedures.

1.1 Objective

The objective of Mirantis OpenStack validation is to provide Mirantis program partners with an consistent and unified approach for acceptance of their solution into the Mirantis Technology Partner Program.
Validation is designed within the context of Mirantis OpenStack infrastructure, including Mirantis Fuel deployment tool and supported cloud reference architectures.

1.2 Target Audience

This document serves as a guide to understand the reference architecture and deployment of the Hedvig cinder driver on Mirantis Fuel for anyone using the Hedvig Distributed Storage Platform.

2. Product Overview

The Hedvig Storage Service software transforms existing server and storage assets – including SSD/flash media and hard disk – into a full-featured elastic storage cluster. The software deploys in a private data center and in public clouds to create a single storage cluster that is implicitly hybrid. Every storage feature such as compression, deduplication, and replication can be switched on or off to fit the specific needs of any given workload. Granular selection of features empowers administrators to avoid the challenges and compromises of a “one size fits all” approach to storage.

The Hedvig Storage Service operates as an optimized key value store and is responsible for writing data directly to the storage media. It captures all random writes into the system, sequentially ordering them into a log structured format that flushes sequential writes to disk. This provides the ability to ingest data at a high rate, as well as optimize the disk utilization.

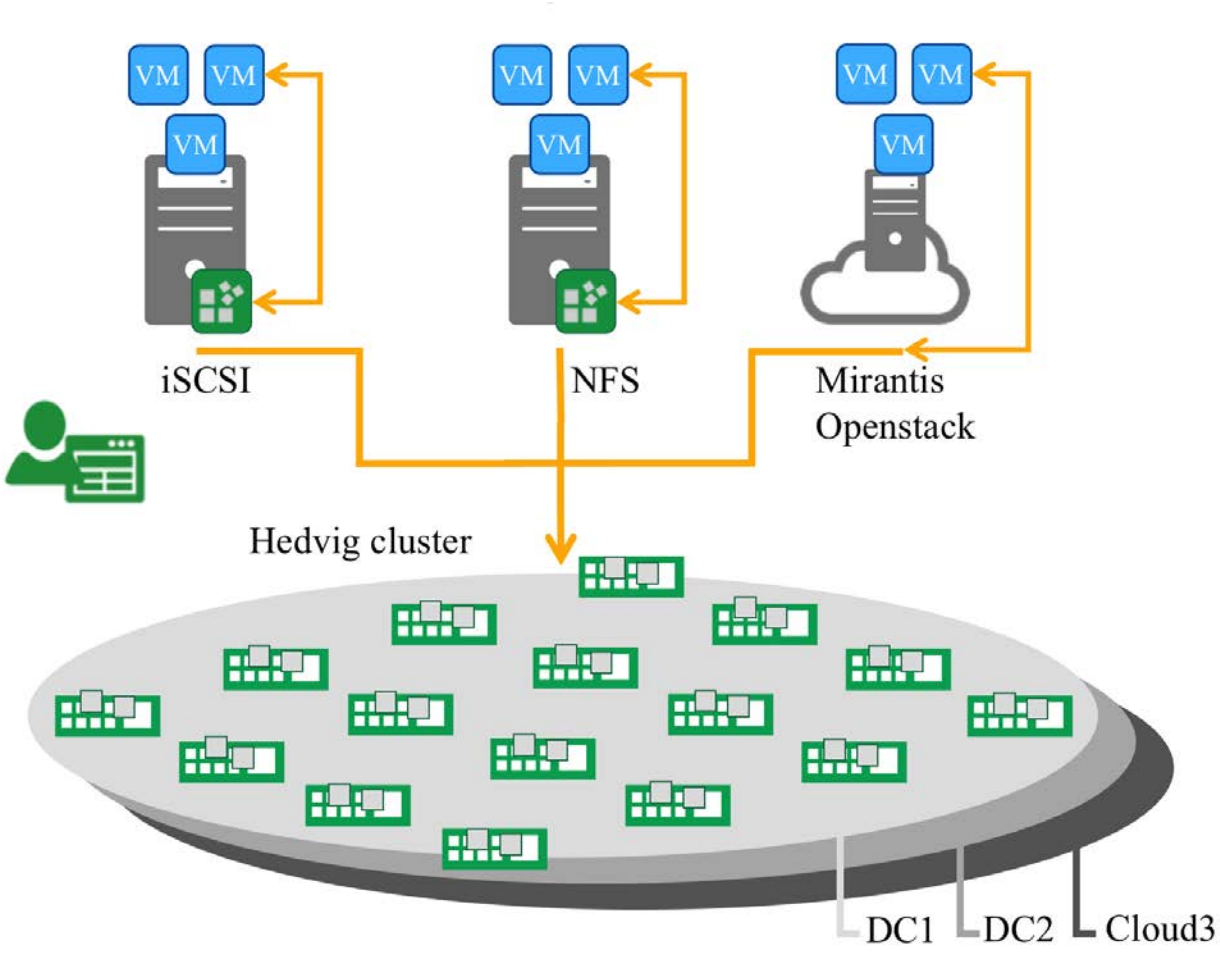
The Hedvig Storage Service consist of three primary components:

- **Data process.** The data process is responsible for the layout of data on raw disk. Hedvig storage nodes create two units of partition: storage pools, a logical grouping of three disks in the node, and containers, 16 GB chunks of data. Virtual Disks are divided into containers, each residing in a unique storage pool in a given storage node. Containers replicate based on the replication policy associated with the Virtual Disk.
- **Metadata process.** The metadata process is responsible for how and where data is written. Metadata also tracks all reads and guarantees all writes in the system, noting the container, storage pool, replica, and replica locations of all data. Metadata is a key component of the

underlying storage cluster, but is also cached by the Hedvig Storage Proxy enabling metadata queries from the application host tier without traversing the network.

- **The Hedvig Storage Proxy**. The Hedvig Storage Proxy is a lightweight abstraction hosted either in a VM or a container at the application tier, providing storage access to each physical host. The Storage Proxy ensures the Hedvig platform operates in your existing environment without requiring changes to hypervisors, guest VMs, OSes, or applications, preserving virtualization and storage administrator workflows to eliminate the need to adopt new processes and procedures to operate a distributed, elastic storage cluster.

3. Joint reference architecture



4. Networking

N.A

5. Installation and Configuration

5.1 Overview of MOS installation steps

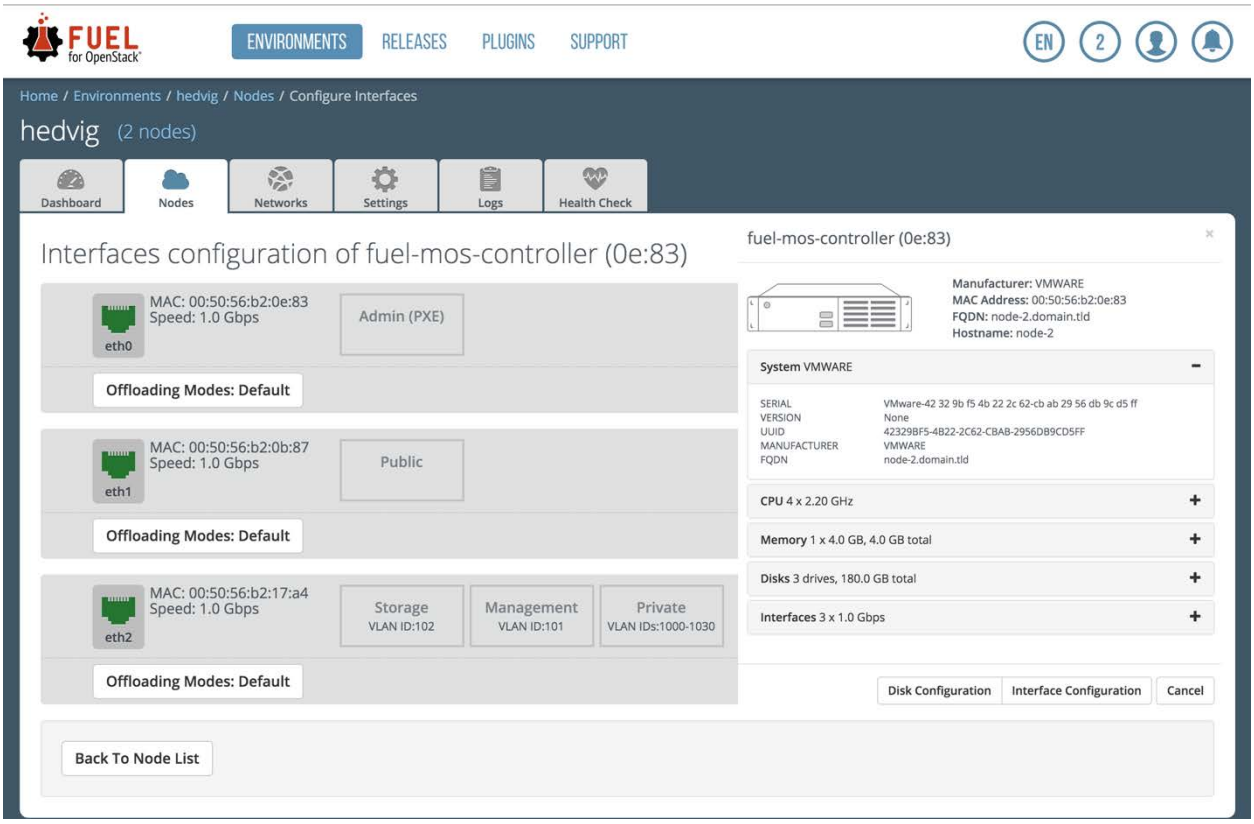
- Mirantis Openstack was setup on a virtualized environment within VMware.
- Mirantis Openstack 7.0 was downloaded and installed from the link - https://docs.mirantis.com/openstack/fuel/fuel-7.0/#downloads
- Three virtual machines were created – one fuel master node and two Openstack nodes.

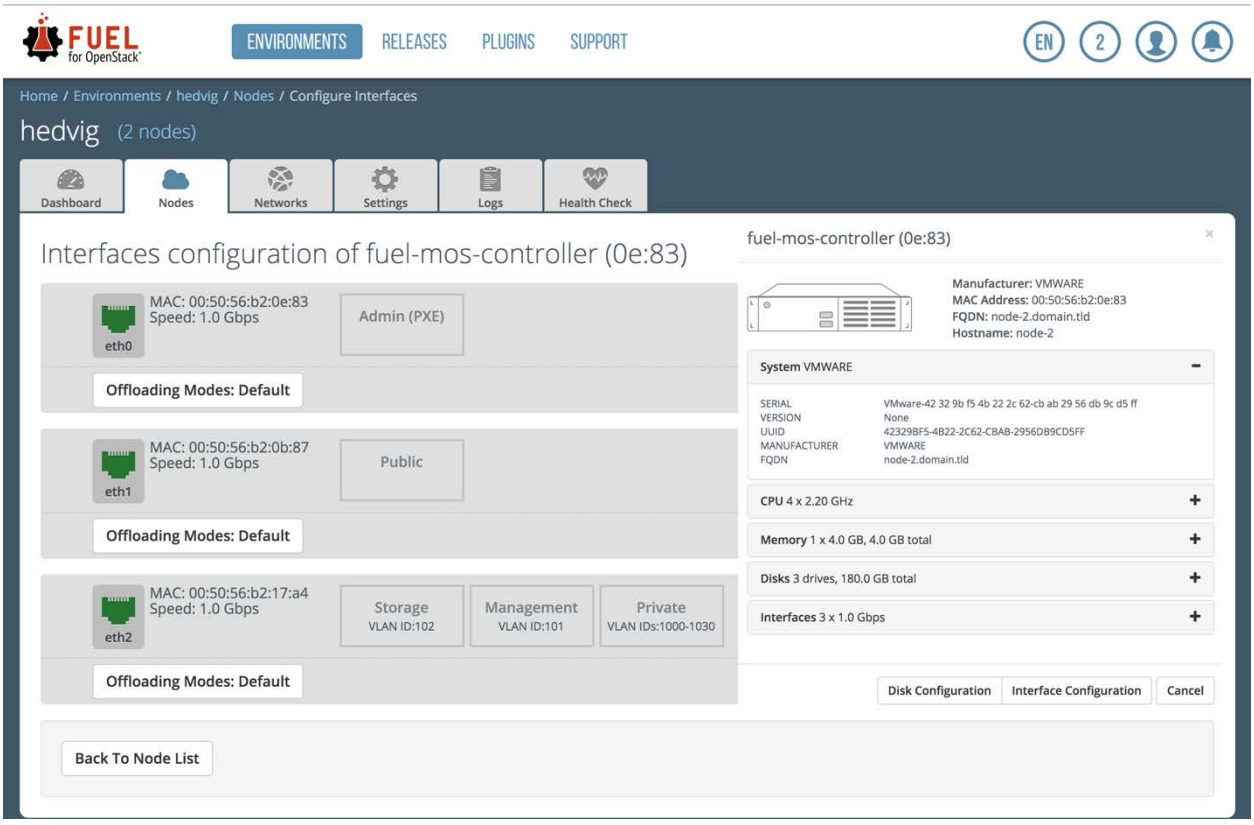
- Setup MOS according to the following user guide - https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html

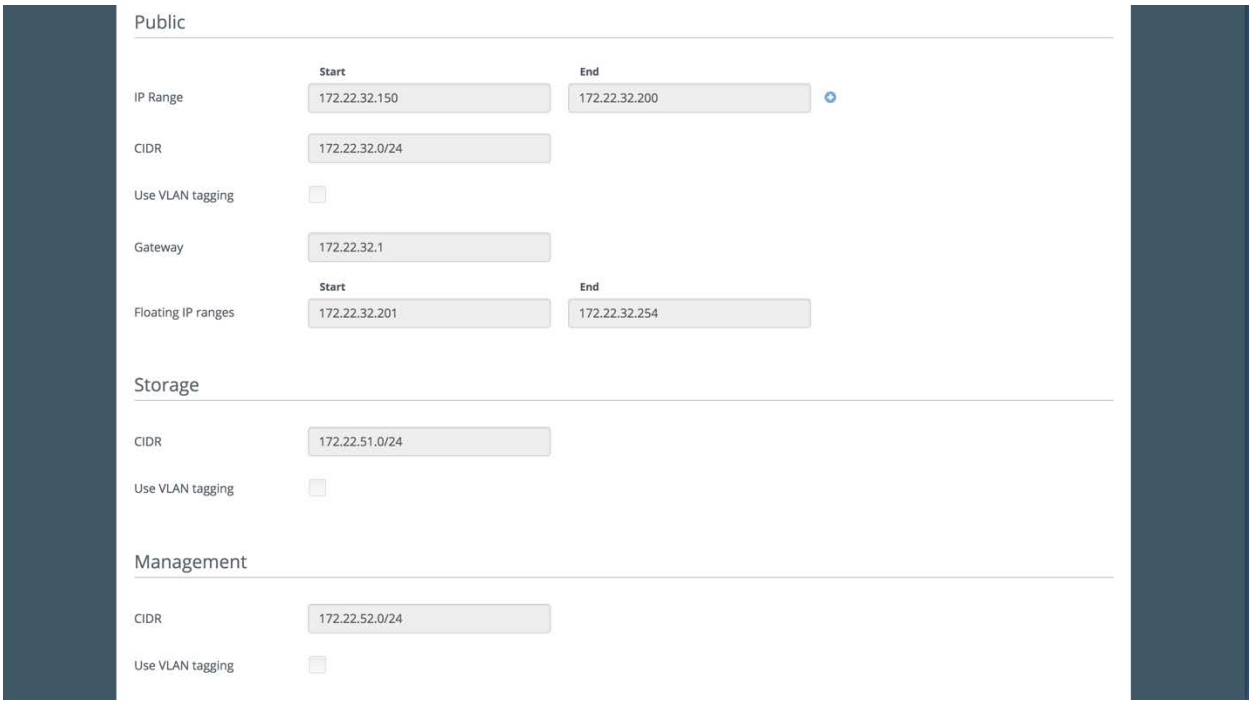
5.2 MOS Installation in detail - Creation of OpenStack environment

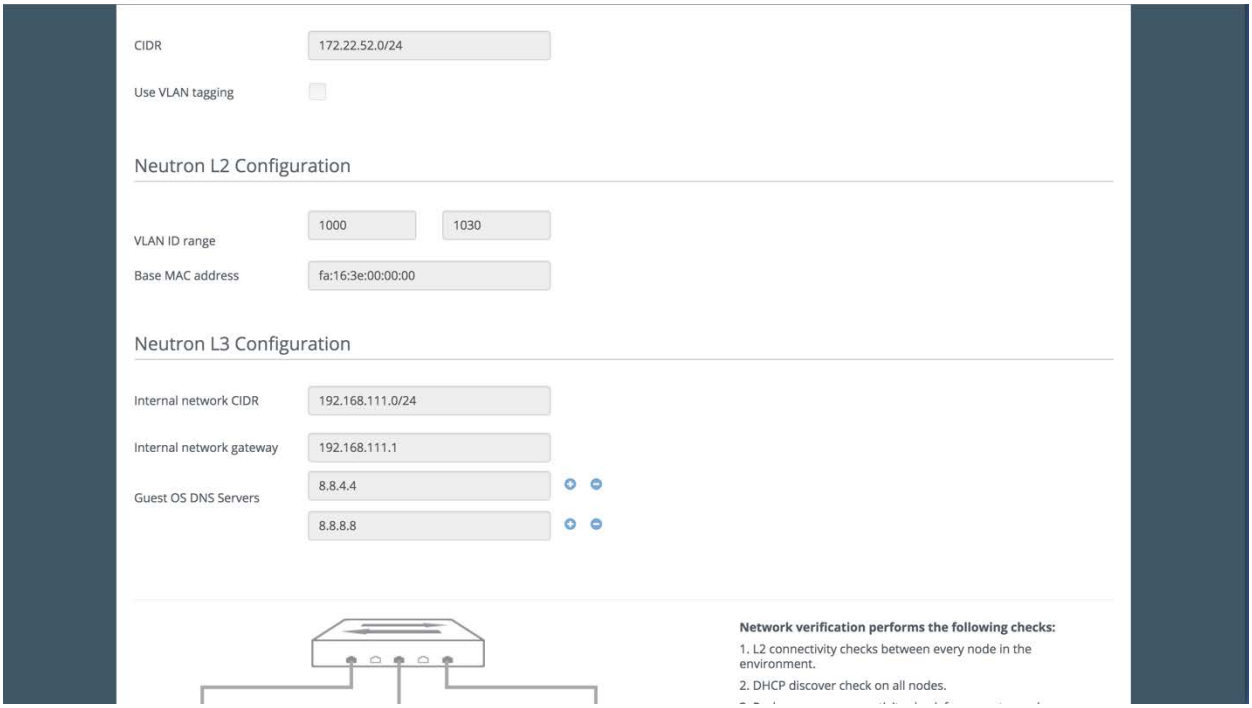
Environment name: Hedvig
Openstack Release: Kilo on Ubuntu 14.04
Compute: QEMU
Network: Neutron with VLAN Segmentation
Storage Backends: Cinder LVM over iSCSI for volumes

The following screenshots show the interface configuration of the nodes and also the network settings of the Openstack environment –




FUEL for OpenStack
ENVIRONMENTS
RELEASES
PLUGINS
SUPPORT
EN
2
Home / Environments / hedvig / Nodes / Configure Interfaces
hedvig (2 nodes)
Dashboard
Nodes
Networks
Settings
Logs
Health Check
Interfaces configuration of fuel-mos-controller (0e:83)
MAC: 00:50:56:b2:0e:83
Speed: 1.0 Gbps
eth0
Admin (PXE)
Offloading Modes: Default
MAC: 00:50:56:b2:0b:87
Speed: 1.0 Gbps
eth1
Public
Offloading Modes: Default
MAC: 00:50:56:b2:17:a4
Speed: 1.0 Gbps
eth2
Storage
VLAN ID:102
Management
VLAN ID:101
Private
VLAN IDs:1000-1030
Offloading Modes: Default
Back To Node List
fuel-mos-controller (0e:83)
Manufacturer: VMWARE
MAC Address: 00:50:56:b2:0e:83
FQDN: node-2.domain.tld
Hostname: node-2
System VMWARE
SERIAL VMware-42 32 9b f5 4b 22 2c 62-cb ab 29 56 db 9c d5 ff
VERSION None
UUID 42329BF5-4B22-2C62-CBAB-2956DB9CD5FF
MANUFACTURER VMWARE
FQDN node-2.domain.tld
CPU 4 x 2.20 GHz
Memory 1 x 4.0 GB, 4.0 GB total
Disks 3 drives, 180.0 GB total
Interfaces 3 x 1.0 Gbps
Disk Configuration
Interface Configuration
Cancel


Public
Start
End
IP Range
172.22.32.150
172.22.32.200
CIDR
172.22.32.0/24
Use VLAN tagging
Gateway
172.22.32.1
Start
End
Floating IP ranges
172.22.32.201
172.22.32.254
Storage
CIDR
172.22.51.0/24
Use VLAN tagging
Management
CIDR
172.22.52.0/24
Use VLAN tagging



5.3 MOS Deployment

Following 3 nodes were deployed –

- Fuel node – A virtual machine was deployed that served as the Fuel master node, which bootstrapped and deployed the Openstack nodes.
- Openstack controller – A virtual machine was deployed that served as the Openstack controller and the cinder node.
- Openstack compute – A virtual machine was deployed that served as the Openstack compute node.

Fuel node was deployed with the following network interfaces –

- VLAN-PXE – This interface was on the same VLAN as the Admin PXE interface of the fuel master.
- VLAN-Public – This interface was on a VLAN that had access to the internet.

Both the Openstack nodes were deployed with the following network interfaces –

- VLAN-PXE – This interface was on the same VLAN as the Admin PXE interface of the fuel master.
- VLAN-Public – This interface was on a VLAN that had access to the internet.
- VLAN-Trunked – A trunked VLAN was used for Openstack Storage, Management and Private networks.

5.4 Hedvig Cinder Driver Installation steps

Prerequisite – In order to use the hedvig cinder driver, you need to have a hedvig cluster setup.

Steps to manually install the hedvig cinder driver on the cinder node (or the controller node, if the controller node is also assigned the role of cinder)

- **Step 1** – Download the hedvig-cinder.tar which is hosted on Hedvig FTP Server (User will need credentials to download it provided by Hedvig) onto the openstack cinder node (We can make this publically available on the hedvig github)
- **Step 2** – Untar hedvig-cinder.tar. You should now see three directories created with names "libs", "python" and "python-libs".
- **Step 3** – Copy the contents of the "libs" directory to /usr/local/lib. If "/usr/local/lib" does not exists, please create this directory.
- **Step 4** – Copy the contents of the "python-libs" directory to /usr/lib/python2.7/dist-packages.
- **Step 5** – Copy the contents of the "python" directory to /usr/lib/python2.7/dist-packages/cinder/volume/drivers/hedvig. If this directory does not exist, please create this directory.
- **Step 6** – Install the following dependencies for the hedvig cinder driver using the apt-get command – libboost-all-dev, libsnappy-dev and libgoogle-glog-dev.
- **Step 7** – Run the command "ldconfig /usr/local/lib"

Steps to configure the hedvig cinder driver on the cinder node (or the controller node, if the controller node is also assigned the role of cinder)

- **Step 1** – Run the following command on the cinder node to create a volume type for hedvig – **cinder type-create hedvig**
- **Step 2** – Add the following line under the [DEFAULT] section in /etc/cinder/cinder.conf – **enabled_backends=hedvig**
- **Step 3** – Add the following section for hedvig cinder driver in /etc/cinder/cinder.conf –

  [hedvig]
  volume_driver=cinder.volume.drivers.hedvig.hedvig-cinder.HedvigISCSIDriver
  hedvig_metadata_server=<IP Address or hostname of a hedvig server node in the cluster>

After this is done, please restart the cinder-volume service to apply the changes and initialize the hedvig cinder driver. When you try to create a new cinder volume using the Openstack UI, you should see the "hedvig" option appear in the "Type" drop down box.

6. Testing

6.1 Test tools

No third party test tools were used for testing the hedvig cinder driver. All the tests run were manual integration tests.

6.2 Test cases

- Create a volume of type 'hedvig' and verify whether an underlying vdisk is created in the hedvig cluster for this volume.
- Create a snapshot of a volume and verify whether a snapshot of the underlying vdisk is created in the hedvig cluster.
- Create a volume from the snapshot and verify whether a clone vdisk corresponding to the snapshot is created in the hedvig cluster.
- Attach a volume to a test instance and verify whether the volume appears on the test instance under the specified path.
- Integrity tests for writes to the volume in the test instance.
- Integrity tests for reads to the volume in the test instance.

6.2.1 Target Use case(s)

The main use case for the hedvig cinder driver is to enable the users to interoperate between the hedvig distributed storage platform and Mirantis Openstack with hedvig serving as a backend for cinder volumes.

6.2.2 Deployment modes and configuration options

Currently, Mirantis Openstack 7.0 is only supported on Ubuntu 14.04. All the testing was done on controller and compute nodes running Ubuntu 14.04. The only limitation today is that the cinder driver has to run on the controller node therefore the cinder role has to be assigned to the controller node as well.

6.2.3 Functional testing

The following functional tests were executed

- Creation and deletion of a volume (passed)
- Attaching a volume to a compute instance and performing IO on that volume (passed)
- Extend a volume (passed)
- Creation and deletion of snapshots (passed)
- Create a volume from snapshot (passed)
- Clone a volume (passed)
- Copy image to volume (passed)

- Boot an instance from volume (passed)
- Boot an instance from a volume snapshot (passed)

6.2.4 Performance testing

- Performance tests were run using fio and vdbench tools on cinder volumes attached to compute instances in Openstack.
- Performance tests were run on cinder volumes with different QoS specs assigned to them.

6.2.5 Negative testing

- Hedvig cluster node failure tests for cinder volume create/attach/extend/snapshot/clone operations.
- Hedvig cluster node failures tests for IO operations after the cinder volume is attached to the compute instance.
- Hedvig datacenter failure tests for cinder volume create/attach/extend/snapshot/clone operations.
- Hedvig datacenter failure tests for IO operations after the cinder volume is attached to the compute instance.

6.3 Test results (if FUEL HealthCheck is used)

Fuel HealthCheck isn't used to test the Hedvig cinder driver.

7. Troubleshooting

1. How do I debug a volume creation failure?
    - Any errors at the time of volume creation will be logged in /var/log/cinder/cinder-volume.log.
    - If the error is related to the hedvig cinder driver, then you will see a HedvigDriverException in the logs. When this happens, a corresponding error can be found under /var/log/cinder/cinder/cinder.ERROR.

2. How do I debug the situation where volume creation succeeds but I cannot locate the corresponding vdisk on the hedvig cluster?
    - The main reason for this can be a failure in communication between the hedvig cinder driver and the hedvig server nodes.
    - Open the /etc/cinder/cinder.conf file and make a note of the value associated with the parameter 'hedvig_metadata_server' under the 'hedvig' section. Check whether you can ping this hostname from the openstack controller node.

3. When I attach a volume to a compute instance, I don't see device path listed for that volume. How do I debug this?
    - This can happen when iscsi utils are not installed on the openstack controller node where the hedvig cinder driver runs.

4. When I attach a volume to a compute instance, I get an error. How do I debug this?

- To interoperate with the hedvig cluster, every openstack compute host must be associated with a iscsi target provided by hedvig.
- If the attach volume operation fails, the openstack compute node where the corresponding instance is running might not have an iscsi target associated with it.

5. I cannot perform an I/O on the cinder volume from the compute instance. How do I debug this?
    - This indicates that the iscsi target associated with the compute host (where the instance is running) is inoperational.
    - Restart the iscsi target can fix this issue.