Installation Runbook for Violin® Memory Storage Driver for OpenStack®

Product Name:	Violin Memory Cinder Driver for OpenStack
Product Version:	4.0.0
MOS Version:	7.0
OpenStack Version:	Kilo
Product Type:	Storage Driver for Cinder

Violin Memory, Inc. | 4555 Great America Parkway, Santa Clara, CA 95054 USA | Tel: 1-650-396-1500 • Fax: 1-650-396-1543 | www.violin-memory.com

Table of Contents

Document History

Version	Revision Date	Description
0.1	23-05-2016	Initial Version
0.2	09-06-2016	Updated Section 1
0.3	04-07-2016	Updated hyperlinks

1. Abstract

The enterprise and service provider IT landscapes have been transformed by cloud architecture and the adoption of Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). These changes allow databases, applications, and services to take better advantage of the cloud's flexibility, cost- effectiveness, disaster-recovery options, and security.

With OpenStack, the leading open-source cloud platform, companies get a rich set of cloud features that go far beyond traditional compute, storage and networking. OpenStack works with popular enterprise and open source technologies and this makes it ideal for heterogeneous data center infrastructure. Contributions made by Mirantis and others can be leveraged with new and existing hardware in data centers or proof-of-concept labs.

Many of the world's largest companies use OpenStack in their data centers to reduce management costs and accelerate cloud provisioning. With a strong partner ecosystem, companies that want commercial support have access to different OpenStack-powered products and services from OpenStack partners.

Mirantis is the pure play OpenStack company, delivering all the software, services, training, and support needed for running OpenStack. More customers rely on Mirantis than on any other company to get to production deployment of OpenStack at scale. Mirantis is among the top three companies worldwide in contributing open source software to OpenStack, and has helped build and deploy some of the largest OpenStack clouds in the world.

1.1. What is OpenStack?

The OpenStack software platform is, essentially, a cloud operating system (OS) inended to control data center compute, storage, and networking IT infrastructure as resource pools that can be manages via a dashboard or an API (application programming interfaces). This provides data center administrators with the ability to provision IT infrastructure through a common interface regardless of vendor brands or hardware models. Since OpenStack works with popular enterprise products and open source technologies, it is ideal for use with heterogeneous IT infrastructure.

OpenStack is built and maintained by a community of developers and users and designed in the open at industry summits. While this is managed by the OpenStack Foundation (OpenStack.org) to promote the global development, distribution and adoption, OpenStack remains an open source project released under the Apache license.

According to the OpenStack Foundation, its goal is "to serve developers, users, and the entire ecosystem globally by providing a set of shared resources to grow the footprint of public and private OpenStack clouds, enable technology vendors targeting the platform and assist developers in producing the best cloud software in the industry. As the global independent home for OpenStack, the Foundation serves more than 30,000 Individual Members from over 170 countries around the world."

1.2. Why OpenStack?

Companies of all sizes and industries need to rapidly evolve in response to dynamic markets. For example, long established "brick-and-mortar" businesses find themselves in fierce competition with emerging online companies. Therefore, companies must transform their IT infrastructure to support an expansion of their online footprint and development of mobile-friendly applications to deliver the on-demand and self-service capabilities consumers increasingly expect.

OpenStack provides an open and flexible framework businesses can use as core technology to drive towards a cloud-based and software-defined data center (SDDC). With OpenStack, companies can more easily implement:

- Platform-as-a-Service (PaaS)
- Software-as-a-Service (SaaS)
- Infrastructure-as-a-Service (IaaS)
- Web-scale infrastructure
- Cloud architecture

Contributions made by Mirantis and others can be leveraged with existing hardware in your data center or proof-of-concept labs. This is because vendor drivers are provided Open Source and completely free of cost.

1.3. What is Violin Flash Storage Platform™?

The Violin Flash Storage Platform (FSP) is an all-flash storage solution leveraging unique innovations including Violin Flash Fabric Architecture™ (FFA) encompassing multiple layers of all-flash technologies, Violin Concerto OS 7™ next generation operating system software for all-flash storage, Violin Intelligent Memory Modules™ (VIMM) meshing individual flash dies into intelligent flash management units, and Violin Scale Smart™ scaling all-flash storage capacity, performance, and connectivity beyond the capabilities of scale-up and scale-out alternatives.

Violin FSP all flash storage is available in the following models:

Violin FSP 7000 Series

Violin FSP 7700 Series

Violin FSP all flash solutions include fully integrated systems occupying three rack units (3RU) of space and highly flexible modular systems occupying 9RU to 36RU of space. Storage capacities range from a few TBs and scale to several PBs with storage

performance ranging from hundreds of thousands to millions of IOPS with latencies as low as 150 microseconds. Violin FSP storage combines all flash storage with comprehensive business continuity, data protection, data efficiency, and data scaling data services.

Violin FSP all flash storage supports enterprise data services including:

Violin Management
Violin Symphony 3

Violin Operating System
Concerto OS 7

Violin Business Continuity
FSP Stretch Cluster
FSP Synchronous Mirroring
FSP Asynchronous Replication

Violin Data Protection
FSP Data At Rest Encryption
FSP Thick and Thin Clones
FSP Snapshots With Consistency Groups

Violin Data Efficiency
FSP Thick and Thin Provisioning
FSP Inline Data Compression
FSP Inline Data Deduplication

Violin Data Scaling
FSP Storage Pooling
FSP Online Capacity Expansion
FSP Online LUN Expansion

1.4. Why Violin Flash Storage Platform for OpenStack?

Violin FSP delivers all-flash storage with the extremely low latency, high IOPS and bandwidth, and large storage capacities needed to support consolidated workloads including databases, applications, and services. Mirantis OpenStack delivers data center software that controls pools of compute, storage, and networking resources. Both work with popular enterprise and open source technologies making them ideal complements for heterogeneous IT infrastructure throughout data centers.

Companies can deploy enterprise-class all-flash storage and achieve very favorable CAPEX and OPEX scenarios with the Violin Flash Storage Platform. Violin FSP facilitates the transition of storage from legacy disk and hybrid disk/flash to all-flash solutions. Mirantis OpenStack helps customers to enjoy the business benefits of cloud infrastructure. As the only pure-play provider of OpenStack, Mirantis can provide the appropriate support and development services customers need to be successful.

That's why many of the world's largest companies rely on Violin FSP or Mirantis OpenStack to transform their data center economics by reducing costs and accelerating business.

1.5. About Violin Memory

Violin Memory, the industry pioneer in All Flash Arrays, is revolutionizing how businesses operate, and by enabling IT to Be Instrumental to the organization, through unlocking the power of data. The consistent high-throughput and predictable low latency showcased by the Flash Storage Platform™ is combined with Concerto™ OS 7, a fully integrated storage operating system that enables complete data protection, business continuity, and data reduction services.

Violin Memory's innovative single storage platform solution delivers transformative performance for cloud, enterprise, virtualized business and mission-critical storage applications. The Violin Flash Storage Platform is designed to consolidate high performance and primary storage workloads onto a flexible, uniquely scalable solution called Scale Smart™ while achieving substantive CAPEX and OPEX savings. Founded in 2005, Violin Memory is headquartered in Santa Clara, California. For more information, visit www.violin-memory.com.

1.6. About Mirantis

Mirantis is the pure play OpenStack company, delivering all the software, services, training, and support needed for running OpenStack. More customers rely on Mirantis than on any other company to get to production deployment of OpenStack at scale. Mirantis is among the top three companies worldwide in contributing open source software to OpenStack, and has helped build and deploy some of the largest OpenStack clouds in the world, at companies such as Cisco, Comcast, Ericsson, NASA, Samsung and Symantec.

Mirantis is venture-backed by August Capital, Dell Ventures, Ericsson, Goldman Sachs, Intel, Insight Venture Partners, Sapphire Ventures, Siguler Guff & Co., and WestSummit Capital, with headquarters in Sunnyvale, California.

1.7. About this Reference Architecture

This document is to serve as a detailed Deployment Guide for the Violin Memory Cinder Driver for OpenStack Kilo release with MOS7.0. Violin Memory offers a Cinder Driver to integrate with its high-performance Flash Storage Platform™ (FSP) storage solution provided by the Violin all-flash storage platform. This document describes the reference architecture, installation steps for validating MOS7.0 with the Violin Memory Cinder Driver; it's limitations and testing procedures.

2. Target Audience

The target audience for the Violin Memory Cinder Driver is anyone interested in a high performance, low latency, low cost solution for all their primary storage needs.

3. Product Overview

Violin Memory provides full OpenStack Cinder block storage support for its FSP 7300 and 7700 storage arrays for iSCSI and Fibre channel protocols. Mirantis OpenStack combined with the Violin Memory Cinder driver supporting OpenStack Block Storage will give enterprise businesses and service providers, the ability to leverage Violin's Flash Storage Platform™ (FSP) in private and public cloud computing environments.

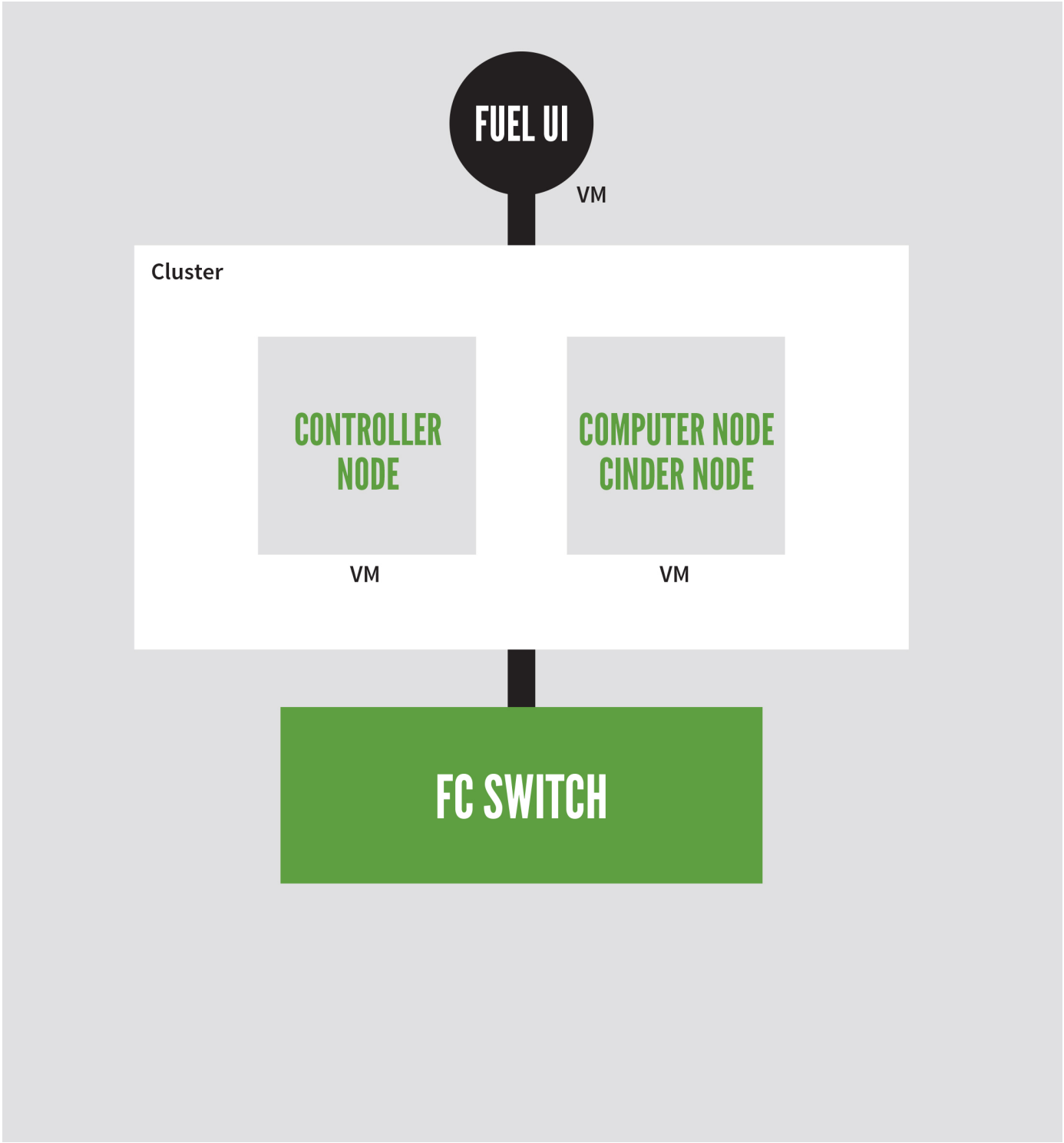
The Volume Driver package for OpenStack Cinder from Violin Memory adds block-storage service support for Violin Flash Storage Platforms. The package is implemented as a storage "plug-in" using the standard Cinder storage driver API, and facilitates the creation, attachment, and management of volumes (LUNs) between the Violin Flash Storage Platform and different host servers. All required Cinder volume features are supported, including volume, snapshot, and clone operations.

4. Joint Reference Architecture

Overview:

This reference architecture describes how to integrate Mirantis OpenStack 7.0 (using OpenStack Kilo) with Violin Cinder 4.0.0, utilizing Violin FSP as a backend storage. The following are components of the reference architecture:

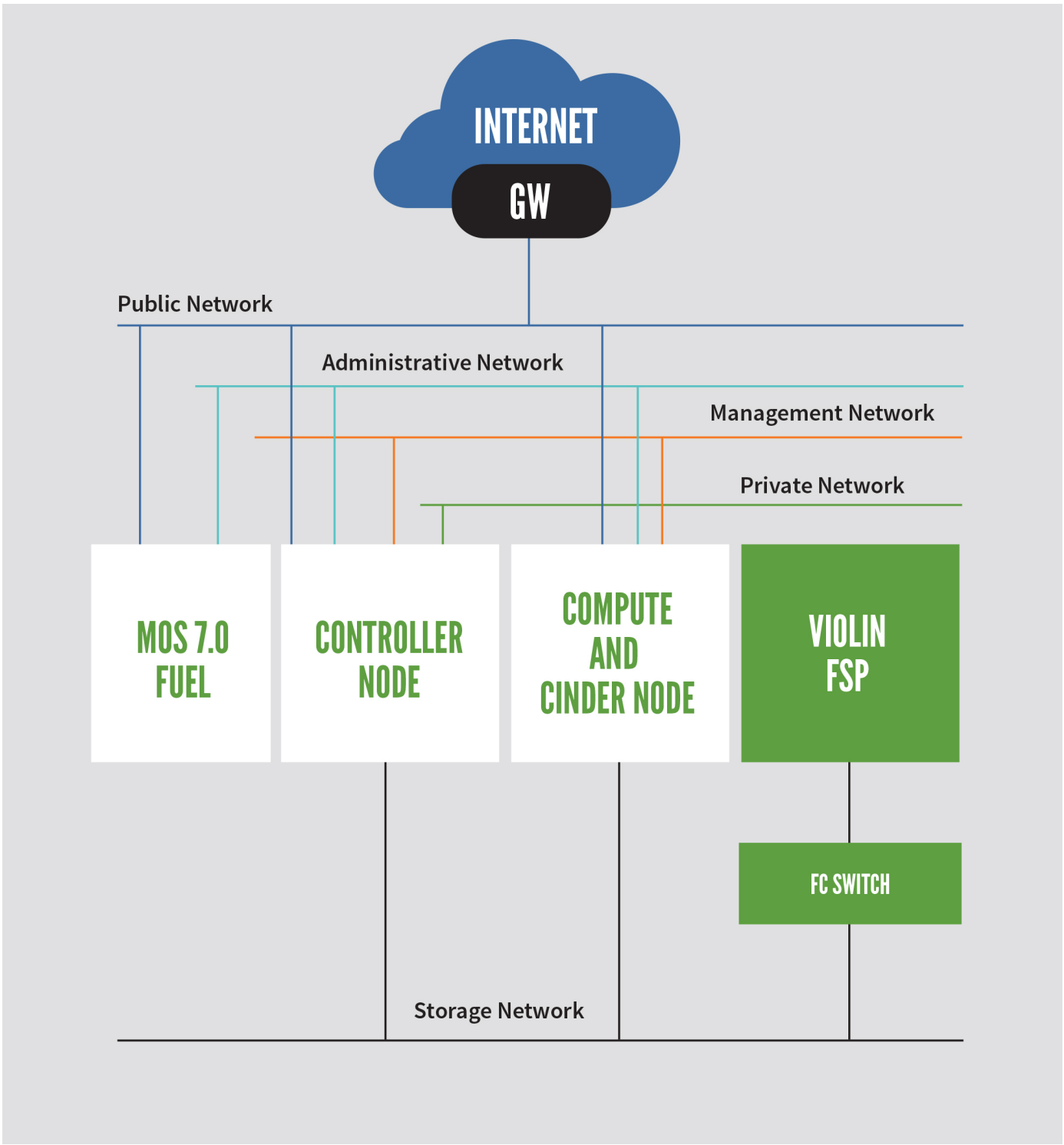
1. **Violin FSP** - To use high-performance and high-reliability backend storage features.
2. **Controller nodes** - Servers running OpenStack controller elements.
3. **Compute nodes** - Servers running OpenStack compute elements.
4. **Cinder node** - Server running OpenStack cinder elements.
5. **Fuel** - Infrastructure running OpenStack deployment and management tool.



4.1. Physical & Logical network topology

In the Mirantis OpenStack (MOS) reference architecture, five different logical networks are used as follows:

1. **Administrative /PXE (Fuel) network:** untagged. Provides DHCP services from the MOS/Fuel server to all nodes
2. **Public/ Floating Network:** untagged. Provides external communications (e.g. Internet) for cloud resources
3. **Management Network:** VLAN 101. Carries OpenStack API traffic and related communications.
4. **Storage Network:** VLAN 102.
5. **Private Network:** VLANs 1000-1030



4.2. Storage Infrastructure: Violin FSP

Historically, achieving high throughput and lower latency meant over-provisioning hardware or using software to mitigate, but not fully eliminate, performance issues. These workarounds have high CAPEX and OPEX costs and still don't meet the requirements for cost-effective storage in 21st century businesses. Violin FSP provides an approach that can support growth, improve efficiency and manageability, and deliver consistent and predictable service levels without breaking the IT budget.

With the Violin Flash Storage Platform, you can deploy enterprise-class all-flash storage and achieve very favorable CAPEX and OPEX scenarios. This set of capabilities developed from a vertically integrated design of software, firmware, and hardware enable the transition of storage from legacy solutions to all-flash.

Additionally, Violin Flash Storage Platforms can address your use case today: whether it be Performance, Primary or more Capacity focused storage needs, the Flash Storage Platform has the feature set with Concerto Enterprise Data Services to fit the bill. Its management suite, including Symphony and the Simple Setup installation utility, make management tasks a breeze and eliminate many of the headaches associated with managing a storage infrastructure.

5. Installation and Configuration

5.1. Environment preparation

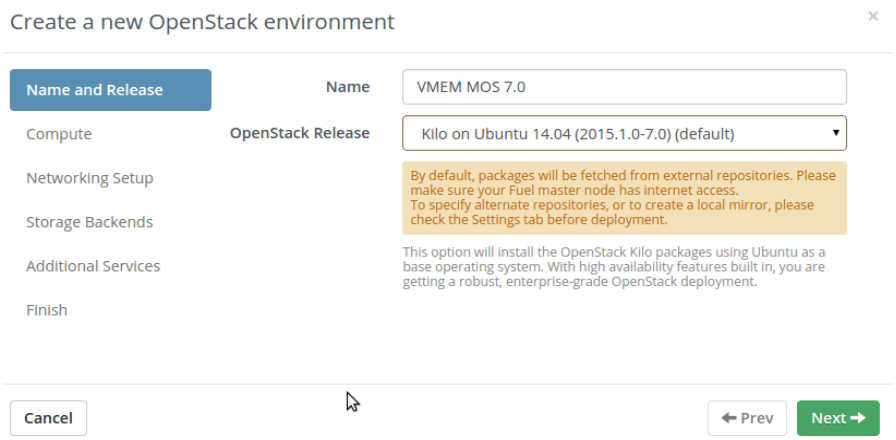
- The Mirantis OpenStack Planning Guide is a recommended resource for the planning phase.
- The following set of hardware should be provided:
    - VM running Mirantis OpenStack / Fuel node
    - One VM to serve as a controller node
    - One VM to serve as a cinder/compute node
    - One Violin FSP
    - Racks, network equipment, power as needed
- While Mirantis OpenStack provides network verification, it is highly recommended to double check IP ranges to ensure a smooth deployment. IP ranges cannot be changed later without redeployment.
- Mirantis recommends adhering closely to the Mirantis OpenStack reference architecture for general MOS configuration and deployment. Deviations from the reference architecture are feasible and in some cases required, but add complexity and require additional testing as well as operational procedures to support them.
- Changes after Fuel deployment should be kept to a minimum.

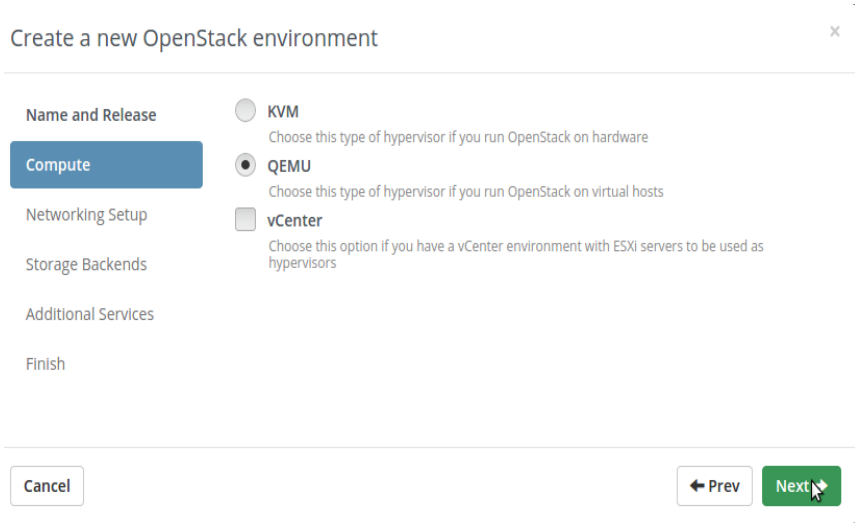
For further details on these and other best practices, please refer to the Mirantis OpenStack 7.0 documentation.

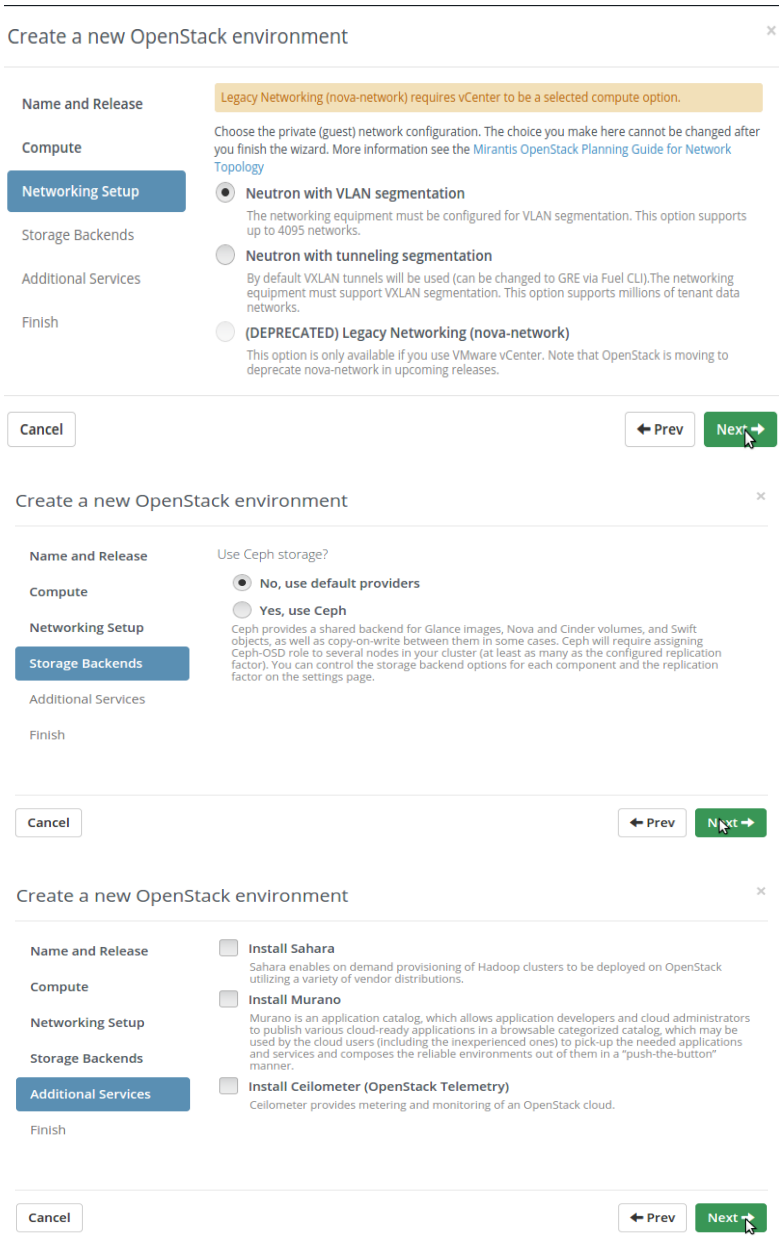
Creation of OpenStack environment:

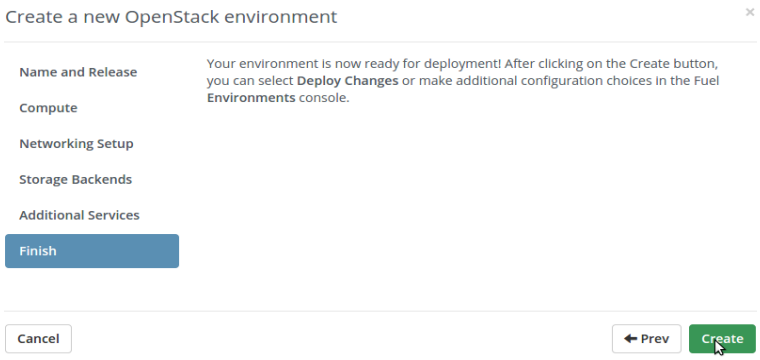
- Launch Wizard to Create New Environment.
- Click on the "New OpenStack environment" icon to launch the wizard that creates a new OpenStack environment.

- Give the environment a name and select the Linux distribution from the drop-down list as, Kilo on Ubuntu 14.04 (2015.1.0-7.0) (default)
- This is the operating system that will be installed on the target nodes in the environment.









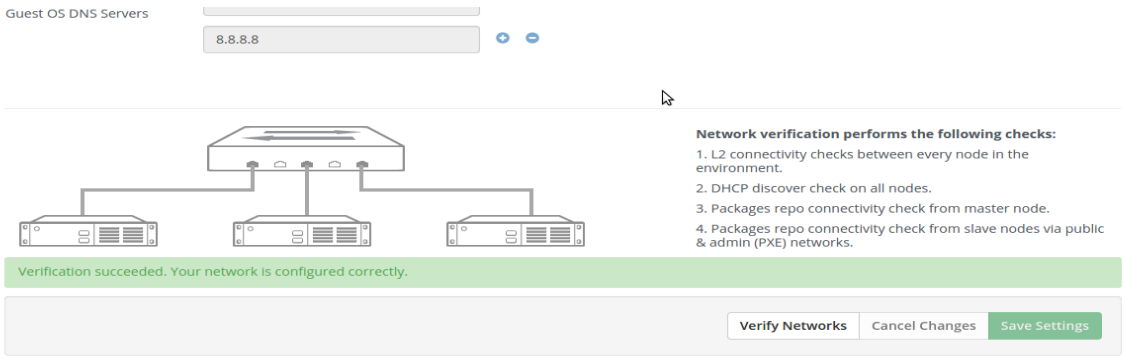
5.2. MOS Installation

The MOS deployment will consist of:

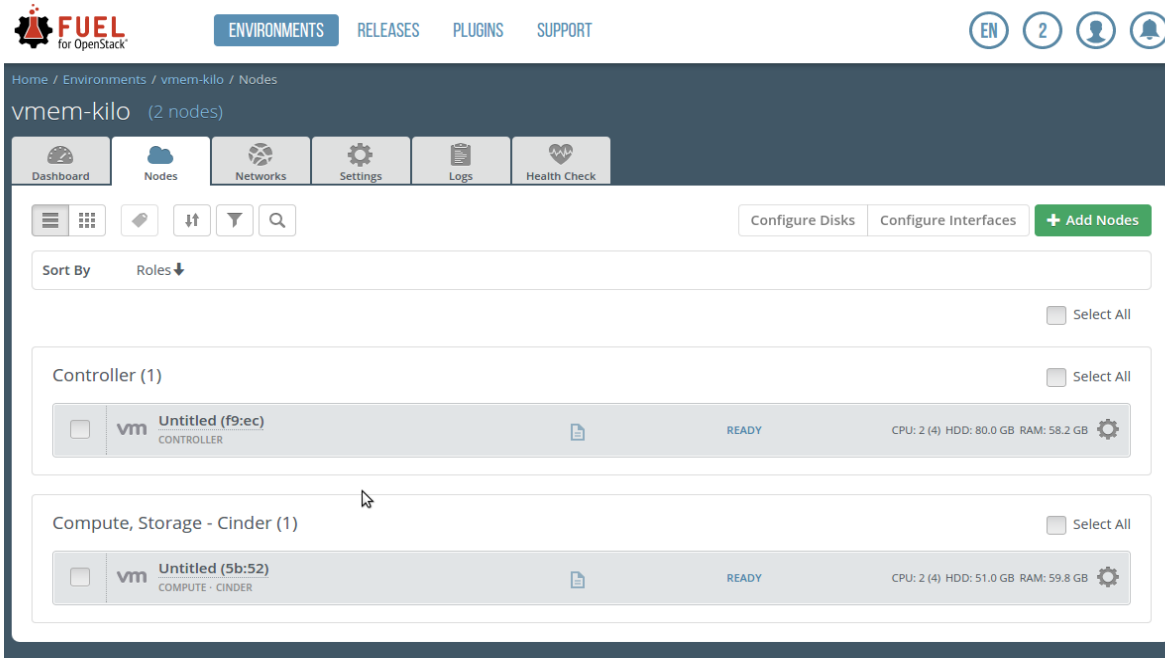
1. One Fuel server.
2. At least one MOS controller
3. Neutron VLAN based configuration is recommended.
4. Storage backend as default providers [Cinder LVM over iSCSI for volumes] is mandatory.

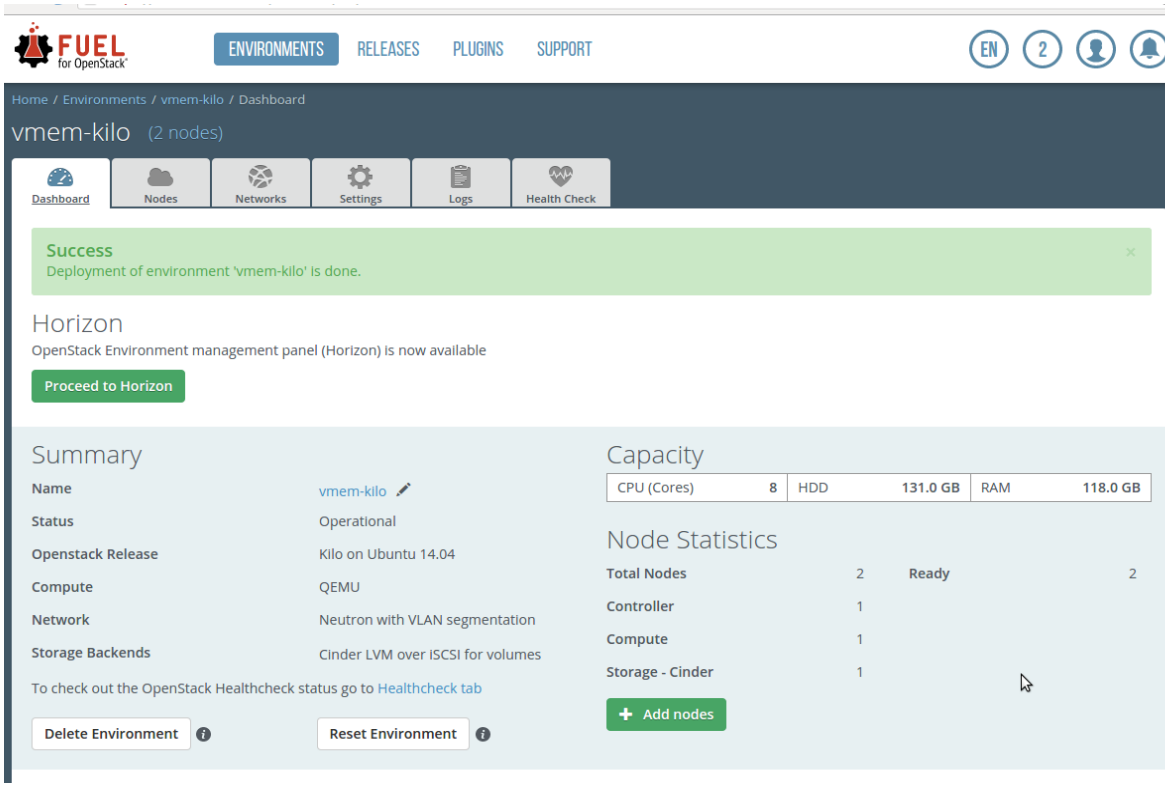
Please follow Mirantis documentation on bringing up a fuel node and discovering nodes on which OpenStack controller/ compute services shall run.

1. Add nodes to the environment.
2. Assign a role or roles to each node server.
3. Do the required Network settings.
4. Mapping logical networks to physical interfaces on servers [if required].
5. Verify Networks (The network verification check should succeed as shown below)
6. Deploy Changes



Here are the role assignments for the nodes:





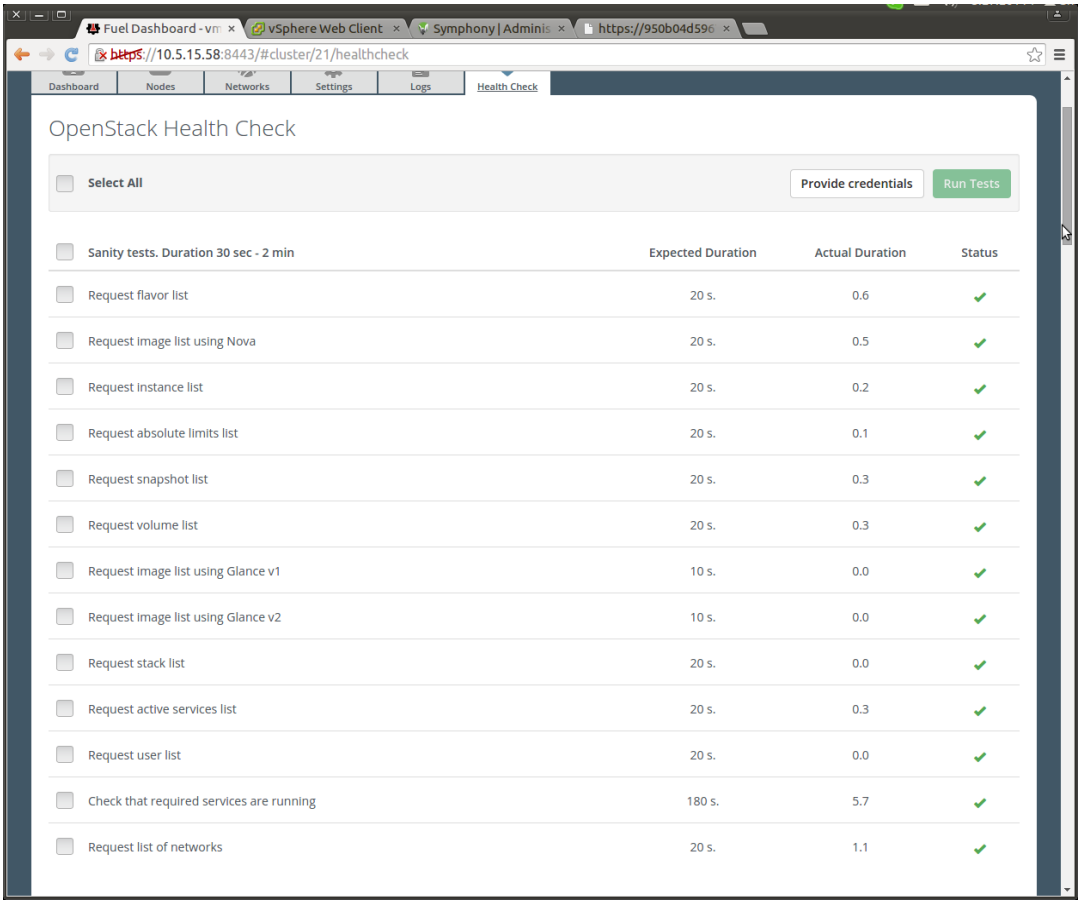
The Fuel UI should confirm successful deployment of the environment.

5.2.1. Health Check Results

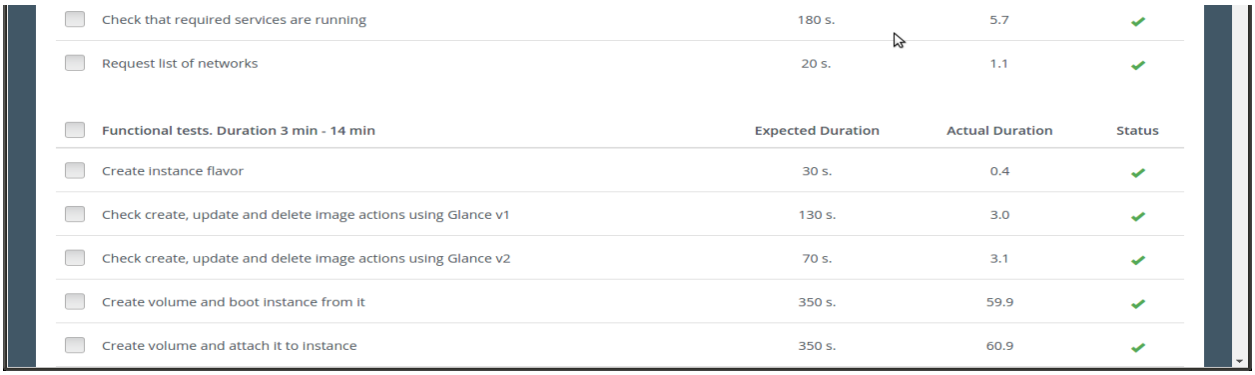
Validating the installation:

- After the configuration has been completed, it should be validated using the automated health check capabilities of Mirantis Fuel.
- Doing this will catch most errors before trying to deploy production workloads.
- All Cinder related tests should pass with no errors.
- The Health Check is initiated from the Mirantis Fuel console (within the context of the relevant OpenStack cloud).
- All of the Sanity Tests should pass, and it is important that the "Create Volume… " related Functional Tests also pass.
- If any of these basic tests fail, the cause should be determined and corrected before proceeding to deploy a workload on these systems.

Here is the output of the Sanity Tests:



Here is the output of the Volume Tests in the Functional Tests:



Check that required services are running	180 s.	5.7	✔
Request list of networks	20 s.	1.1	✔
Functional tests. Duration 3 min - 14 min	**Expected Duration**	**Actual Duration**	**Status**
Create instance flavor	30 s.	0.4	✔
Check create, update and delete image actions using Glance v1	130 s.	3.0	✔
Check create, update and delete image actions using Glance v2	70 s.	3.1	✔
Create volume and boot instance from it	350 s.	59.9	✔
Create volume and attach it to instance	350 s.	60.9	✔

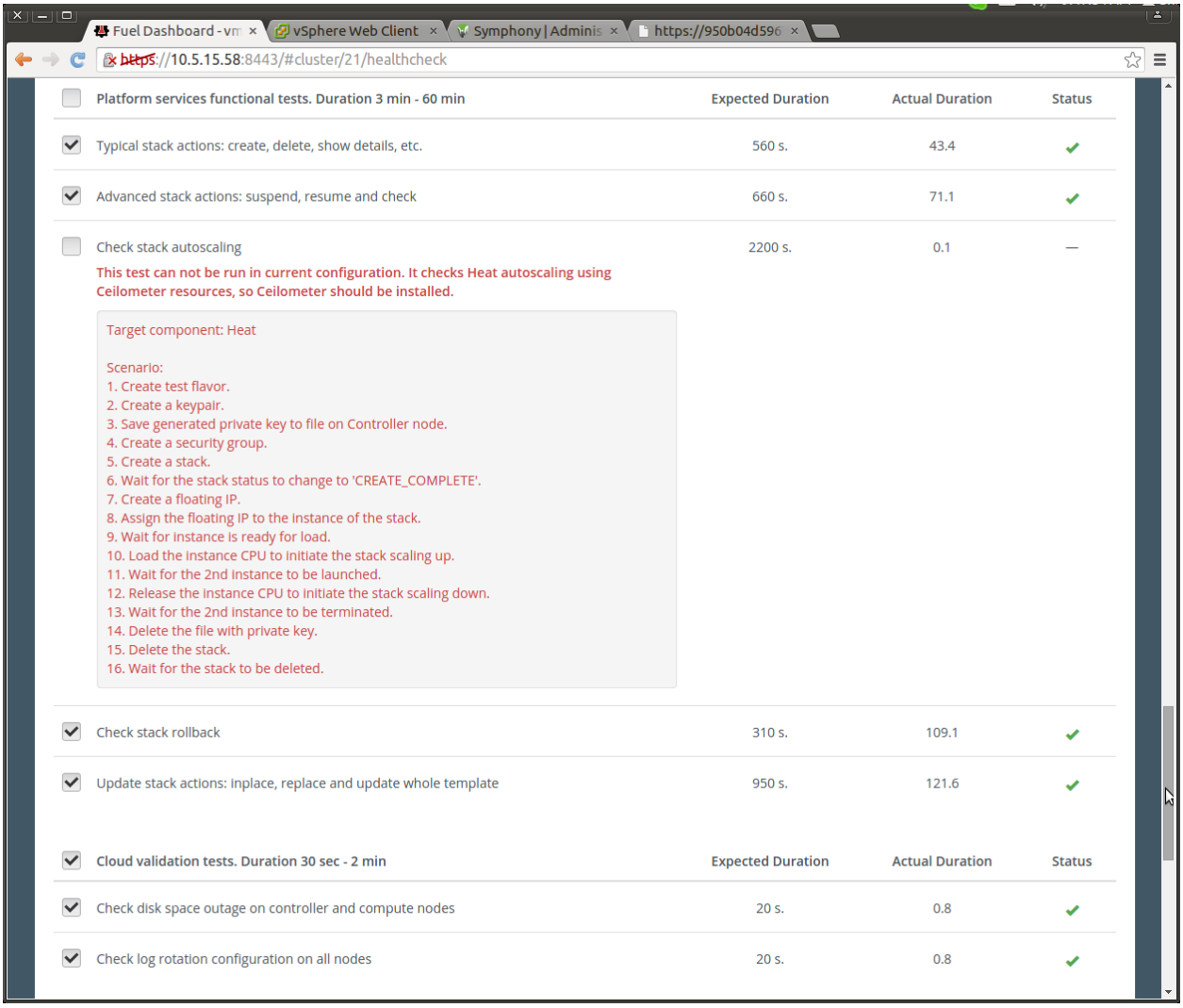
In Health Check - Functional test, the " Check network connectivity from instance via floating IP" was skipped not only because the target component is Neutron hence not required for cinder certification, but also because of the following step in the test:

Check that public IP 8.8.8.8 can be pinged from instance. 8. Check that public IP 8.8.8.8 can be pinged from instance.

In the above mentioned test, this step will check whether public IP 8.8.8.8 can be pinged or not. As the environment built for this certification does not contain 8.8.8.8 in DNS list [Available in "Mirantis OpenStack Environment - Settings tab - Host OS DNS Servers"], the pinging will not happen. Hence this step has been skipped. We use a proxy server IP to establish connectivity between instance and public connectivity.

HA tests were skipped because we did not configure the environment for HA.

This is the output of the Platform Services Functional tests:



As noted, the “Check Autoscaling” test was skipped because we did not use Cielometer.

5.3. Violin Memory Cinder Driver Installation Steps

To install the Violin Cinder driver, do the following:

1. Go to http://www.violin-memory.com/support/
2. Log in to Customer Support using your Violin Memory Customer Portal user name and password. (Contact Customer Support if you do not have an account.)
3. Click the Software Downloads tab.

4. From the OpenStack Cinder folder, download the cinder driver tarball file to a client computer (laptop).

Follow the instructions below, depending on your environment.

The released software is available as an installable tarball. Software and support for existing Violin Memory customers is available from the Violin Memory Support portal at:

http://www.violin-memory.com/support

The Violin OpenStack driver has two components:

1. The Violin Cinder Driver: This driver implements the Cinder API calls.
2. The vmemclient library: This is the Violin Array Communications library to the Flash Storage Platform through a REST like interface.

Installing vmemclient

The version of vmemclient recommended for VMEM OpenStack 3.0 driver is vmemclient 1.1.4. You can download the driver from the python web site: https://pypi.python.org/pypi/vmemclient/.

To install vmemclient driver, do the following:

1. Download, unzip, and untar the vmemclient-1.1.4.tar.gz file.
2. Login as root and navigate to the vmemclient-1.1.4 directory.
3. Run the following command: python setup.py install

To install the tarball, do the following:

1. Unzip the tarball on the machine(s) running Cinder's volume service (cinder-volume).
   For example: tar -zxvf openstack-cinder-vmemdriver-x.x.x.tar.gz
2. Recursively copy the Cinder directory to the same directory as your Cinder code installation.
3. cp -r cinder /usr/local/lib/python2.7/dist-packages/cinder
4. Ensure that FibreChannel is enabled and the HBAs are configured on the FSP. See the 7300 Flash Storage Platform Installation Guide for more information.
5. Configure Cinder to use one of the Violin drivers.
6. Restart cinder-volume.

5.4. Violin Memory Cinder Driver Configuration Options

Basic Configuration

The parameters to be configured for cinder can be referenced by logging into the Violin Memory Support site and obtaining the OpenStack 4.0 Installation and Configuration Guide, Section 2 on Page 11 describes the configuration parameters.

The Cinder configuration parameters are defined in /etc/cinder/cinder.conf. To use the Violin Cinder Driver, this is the only configuration that needs to be changed.

The following table lists the basic options you have to be set prior to running cinder.

IP address or hostname of the Violin 7300 Memory Gateway (string value – Can be # ipaddress or hostname) san_ip=
Log-in credentials for the Violin 7300 Memory Gateway or 7700 FSP controller (string value) san_login= san_password=
The path to the Violin Cinder FCP driver volume_driver=cinder.volume.drivers.violin.v7000_fcp.V7000FCPDriver

The following table lists optional parameters you can set at your preference.

Use thin provisioning for SAN volumes – default is True? # (boolean value, enter true for thin provision and false if not) san_thin_provision=
Name of the Volume backend (string value) volume_backend_name=
User defined capabilities, a JSON formatted string # specifying key/value pairs. (string value) # The ones particularly supported are dedup and thin. Only these two capabilities are # listed here in cinder.conf, will this backend be selected for creating luns which have # volume type associated with them that have 'dedup' or 'thin' extra_specs specified extra_capabilities={}

The 7700 FSP controller, also known as an external head setup, requires the following additional configuration.

Storage pools that support Dedup luns only (list value) dedup_only_pools=
Storage pools that are capable of supporting Dedup in addition to other lun types (list value) dedup_capable_pools=
The method of choosing a suitable storage pool for creating a lun. The recognized # methods are random, largest and smallest. When none specified, random method # is used.

```
pool_allocation_method=
```

Multi-Backend Configuration

Cinder can be configured to use multiple backend storage devices that are all controlled by a single Cinder Volume node. When used in this mode, each of the backend storage must have its own section in cinder.conf. Each section identifies the name of the backend. Multiple backends can share the same name. In such a case, cinder scheduler takes care of selecting the most appropriate backend for the current request based on the user-specified volume-type. In addition to a name, each backend configuration should minimally have the following in its configuration section:

```
# A configuration block name that uniquely identifies the backend
[Block-name]
```

```
# Name of the Volume backend (string value)
volume_backend_name=
```

```
# Path to the Violin FCP cinder driver
volume_driver=
```

```
# Log in credentials to the backend storage device
san_ip=
san_password=
```

```
# Should luns be setup as thin luns by default (Boolean value)
san_thin_provision=
```

```
# Do we attach/detach volumes in cinder using multipath for volume to image and
# image to volume transfers? (boolean value) – Note: While we recommend setting
# this to true, MOS deployment failed saying multipathd was not running
use_multipath_for_image_xfer=false
```

For a complete list of Block storage options configurable through Cinder, refer to:

http://docs.openstack.org/juno/config-reference/content/section_cinder.conf.html

5.5. Limitations

1. iscsi target discovery has to be run after the slaves are delpoyed and iscscid service has to be restarted to get cinder-volume to see the backend. Otherwise systool-c fc_host reports no FC ports.

5.6. Testing

5.6.1. Test cases

In addition to running Health Check from the Fuel UI, the following API functions were also tested from Horizon and matched with the results of the operation from Symphony, Violin's UI for FSP:

Feature	Justification	Description
Setup	Basic API operation.	Initializes the driver. This function creates the control plane connections to the backend and gathers critical setup information (like container name) from the backend.
Check for Setup Errors	Basic API operation.	Checks for errors that occurred during initialization. It checks for things like existence of connections, containers, and verifies FCP configuration bits. Cinder-volume will not allow use of the backend if this function returns any setup errors.
Create Volume	Basic API operation.	Creates a new volume (lun).
Delete Volume	Basic API operation.	Deletes a specified volume.
Get Volume Stats	Basic API operation.	Gathers Violin Array stats/status. For the most part this is checking to see how much free space the array has, gathers some 'capability' data, and verifies that the array is still available. Cinder-scheduler utilizes this data when attempting to allocate space for new volumes.
Create Export	Basic API operation.	Functionality provided by export volume lives in the initialize_connection call for SAN drivers.
Remove Export	Basic API operation.	Functionality provided by remove_export lives in the initialize_connection call for SAN drivers.
Initialize Connection	Basic API operation.	Creates targets and exports the volume. If igroups are being used, the targets and initiators are added to an appropriate igroup. Also provides cinder-volume with connection info that can be shared with other services to complete the connection process.
Terminate Connection	Basic API operation.	Unexports the volume and deletes targets if necessary. If igroups are being used, the targets and initiators may be removed from the appropriate igroup.
Create Snapshot	Basic API operation.	Creates a new snapshot from an existing volume.
Delete Snapshot	Basic API operation.	Deletes a snapshot.
Create Volume From Snapshot	Basic API operation.	Creates a new volume using an existing snapshot as a template. This involves creating a new volume, exporting both the volume and snapshot, and copying the data from the snapshot to the volume.

Create Cloned Volume	Basic API operation.	Creates a new volume using an existing volume as a template. This involves creating a new volume, exporting both volumes, and copying the data from the snapshot to the volume.
Copy Volume to Image	Basic API operation.	Creates a new image using an existing volume as a template. An 'image' is a Glance bootable disk image. Assuming an OS is installed on the volume, it will be exported and copied into a raw disk image that is uploaded to the Glance disk image repository.
Copy Image to Volume	Basic API operation.	Creates a new volume using an existing image as a template. An 'image' is a Glance bootable disk image. A new volume is created and exported and then the disk image will be downloaded from the Glance image repository into the volume.
Migrate volume	Basic API operation.	Copies contents of a volume to a new volume on a different backend storage array. Similar in basic operation to "Create Cloned Volume" except the new volume is created on a different storage device.
Extend Volume	Basic API operation.	Increase the size of an existing volume for all volume types.
FCP Support	Key OpenStack customers use Fibrechannel protocol.	The driver must be able to configure volumes on arrays that use Fibrechannel HBAs.
iSCSI Support	Majority of OpenStack users use iSCSI protocol. So it is definitely must-have.	The driver must be able to configure volumes on 7300/7700 arrays that use iSCSI HBAs.
Multibackend Support	Must-have to support customers with multiple arrays	Cinder-volume can be configured to manage several active arrays at once. In order to schedule volumes on each backend, the driver must be able to collect data from the arrays to identify capabilities and free space for each.
Extra_specs support	Needed to apply custom array settings provided by volume_types	Extra_specs are customer configuration settings that can be associated with volume_types to configure the driver and backend array when provisioning storage. For instance, lun_type (thick, thin, or dedupe) and igroup_name would be supported extra_specs.
HA support	Must-have for reliability	The drivers need to be cluster-aware and gracefully deal with mastership changes and gateway failures/reboots and must be able to operate seamlessly through Gateway/external head failover and failback.

5.6.2. Test results

The tempest suite of tests was also run with the following results:

```
======
Totals
======
Ran: 253 tests in 594.0000 sec.
 - Passed: 249
 - Skipped: 4
 - Expected Fail: 0
 - Unexpected Success: 0
 - Failed: 0
Sum of execute time for each test: 637.4629 sec.
```

All the tests run through Horizon also passed.

5.7. Troubleshooting

For a general guideline on installing/troubleshooting Violin Memory FSP and for the Best Practices Guide, please refer to http://docs.violin-memory.com.

Here are some issues that could occur while trying to run the Violin Memory Cinder driver:

1. If the host is not visible on the compute node, please check to see if PCIe Passthrough has been enforced to pass on control of the ports to it.
2. The WWPNs should also be visible from the storage node.
3. In the case of FC clients, the SAN client corresponding to the compute node should be created with the corresponding WWPNs using Violin Symphony. There is a section in the Best Practices Guide that describes how to do this.
4. If hostnames are used in cinder.conf, verify that DHCP is running and that the hostnames are resolved correctly.
5. In case the backend array is not visible from the storage node, verify that there are no network errors and that the san_ip provided in cinder.conf is pingable.
6. In case of authentication errors to the backend array, verify that the username/password provided to access the Violin FSP in cinder.conf is correct.
7. In some cases, isciadm discovery has to be run to see the ports. After discovering the ports, restart the iscsi daemon.