Red Hat Reference Architecture Series

Deploying and Using Red Hat Enterprise Linux OpenStack Platform 3

Jacob Liberman, Principal Software Engineer

RHCE

Version 1.0

August 2013

Send feedback to refarch-feedback@redhat.com

Comments and Feedback

In the spirit of open source, we invite anyone to provide feedback and comments on any reference architectures. Although we review our papers internally, sometimes issues or typographical errors are encountered. Feedback allows us to not only improve the quality of the papers we produce, but allows the reader to provide their thoughts on potential improvements and topic expansion to the papers.

Feedback on the papers can be provided by emailing refarch-feedback@redhat.com. Please refer to the title within the email.

Staying In Touch

Join us on some of the popular social media sites where we keep our audience informed on new reference architectures as well as offer related information on things we find interesting.

Like us on Facebook:

https://www.facebook.com/rhrefarch

Follow us on Twitter:

https://twitter.com/RedHatRefArch

Plus us on Google+:

https://plus.google.com/u/0/b/114152126783830728030/

Table of Contents

1 Executive Summary

OpenStack is a free and open source Infrastructure-as-a-Service (IaaS) cloud computing project released under the Apache License. It enables enterprises and service providers to offer on-demand computing resources by provisioning and managing large networks of virtual machines. OpenStack boasts a massively scalable architecture that can control compute, storage, and networking resources through a unified web interface.

The OpenStack development community operates on a six-month release cycle with frequent milestones. Their code base is composed of many loosely coupled projects supporting storage, compute, image management, identity, and networking services. OpenStack's rapid development cycle and architectural complexity create unique challenges for enterprise customers adding OpenStack to their traditional IT portfolios.

Red Hat's OpenStack technology addresses these challenges. Red Hat Enterprise Linux OpenStack Platform (RHEL OSP) 3, Red Hat's third OpenStack release, delivers a stable code base for production deployments backed by Red Hat's open source software expertise. Red Hat Enterprise Linux OpenStack Platform 3 adopters enjoy immediate access to bug fixes and critical security patches, tight integration with Red Hat's enterprise security features including SELinux, and a steady release cadence between OpenStack versions. This allows Red Hat customers to adopt OpenStack with confidence, at their own pace, and on their own terms.

This reference architecture introduces Red Hat Enterprise Linux OpenStack Platform 3 through a detailed use case:

- Installing Red Hat's OpenStack technology via Packstack
- Adding Red Hat Storage (RHS) Server persistent storage
- Configuring a Neutron software-based network
- Deploying a multi-tier web application complete with post-boot customization

Every step of this use case was tested in Red Hat's engineering lab with production code.

NOTE: Although this use case includes Red Hat Storage Server, Red Hat Storage Server is not required for a Red Hat Enterprise Linux OpenStack Platform deployment. This document also includes instructions for NFS-backed storage.

2 Component Overview

This section describes the software components used to develop this reference architecture.

2.1 Red Hat Enterprise Linux OpenStack Platform

Red Hat Enterprise Linux OpenStack Platform provides the foundation to build private or public Infrastructure-as-a-Service (IaaS) for cloud-enabled workloads. It allows organizations to leverage OpenStack, the largest and fastest growing open source cloud infrastructure project, while maintaining the security, stability, and enterprise readiness of a platform built on Red Hat Enterprise Linux.

Red Hat Enterprise Linux OpenStack Platform gives organizations a truly open framework for hosting cloud workloads, delivered by Red Hat subscription for maximum flexibility and cost effectiveness. In conjunction with other Red Hat technologies, Red Hat Enterprise Linux OpenStack Platform allows organizations to move from traditional workloads to cloud-enabled workloads on their own terms and timelines, as their applications require. Red Hat frees organizations from proprietary lock-in, and allows them to move to open technologies while maintaining their existing infrastructure investments.

Unlike other OpenStack distributions, Red Hat Enterprise Linux OpenStack Platform provides a certified ecosystem of hardware, software, and services, an enterprise lifecycle that extends the community OpenStack release cycle, and award-winning Red Hat support on both the OpenStack modules and their underlying Linux dependencies. Red Hat delivers long-term commitment and value from a proven enterprise software partner so organizations can take advantage of the fast pace of OpenStack development without risking the stability and supportability of their production environments.

2.2 Red Hat Enterprise Linux OpenStack Platform 3 (“Grizzly”) Services

Red Hat Enterprise Linux OpenStack Platform 3 is based on the upstream “Grizzly” OpenStack release. Red Hat Enterprise Linux OpenStack Platform 3 is Red Hat’s third release. The first release was based on the “Essex” OpenStack release. Red Hat’s second release was based on the “Folsom” OpenStack release. It was the first release to include extensible block and volume storage services. Grizzly includes all of Folsom’s features along with a more robust network automation platform and support for metering and orchestration.

Figure 2.2.1: OpenStack Services depicts the services described in this section.

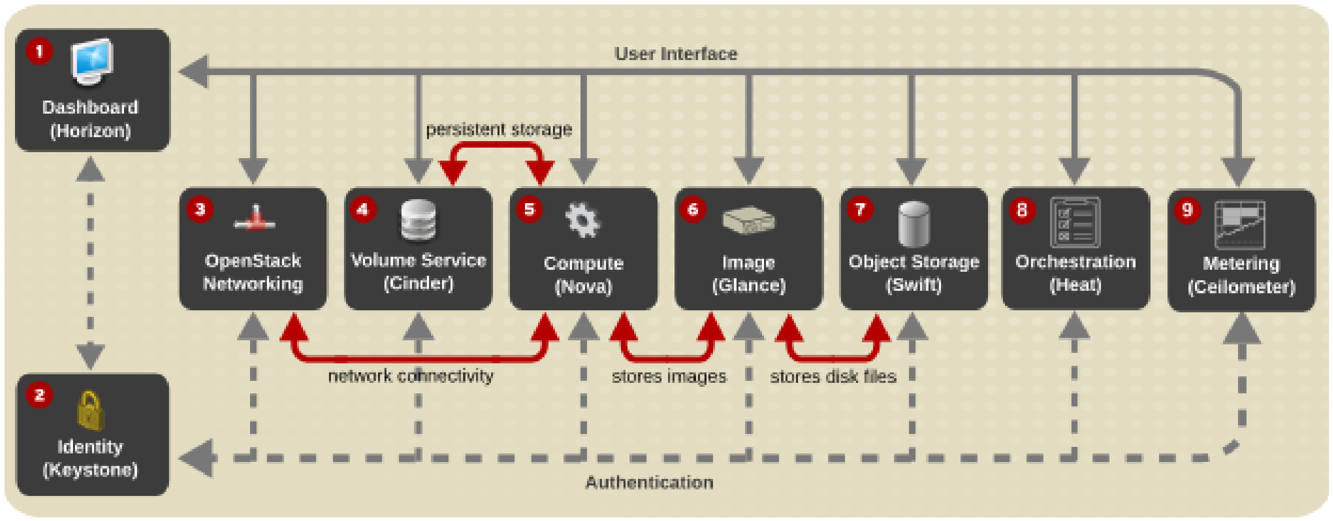


Figure 2.2.1: OpenStack Services

2.2.1 Identity Service ("Keystone")

This is a central authentication and authorization mechanism for all OpenStack users and services. It supports multiple forms of authentication including standard username and password credentials, token-based systems and AWS-style logins that use public/private key pairs. It can also integrate with existing directory services such as LDAP.

The Identity service catalog lists all of the services deployed in an OpenStack cloud and manages authentication for them through *endpoints*. An endpoint is a network address where a service listens for requests. The Identity service provides each OpenStack service -- such as Image, Compute, or Block Storage -- with one or more endpoints.

The Identity service uses *tenants* to group or isolate resources. By default users in one tenant can't access resources in another even if they reside within the same OpenStack cloud deployment or physical host. The Identity service issues tokens to authenticated users. The endpoints validate the token before allowing user access. User accounts are associated with *roles* that define their access credentials. Multiple users can share the same role within a tenant.

The Identity Service is comprised of the **`keystone`** service, which responds to service requests, places messages in queue, grants access tokens, and updates the state database.

2.2.2 Image Service ("Glance")

This service discovers, registers, and delivers virtual machine images. They can be copied via snapshot and immediately stored as the basis for new instance deployments. Stored images allow OpenStack users and administrators to provision multiple servers quickly and consistently. The Image Service API provides a standard RESTful interface for querying information about the images.

By default the Image Service stores images in the */var/lib/glance/images* directory of the local server’s filesystem where Glance is installed. The Glance API can also be configured to cache images in order to reduce image staging time. The Image Service supports multiple back end storage technologies including Swift (the OpenStack Object Storage service), Amazon S3, and Red Hat Storage Server.

The Image service is composed of the **`openstack-glance-api`** that delivers image information from the registry service, and the **`openstack-glance-registery`** which manages the metadata associated with each image.

2.2.3 Compute Service (“Nova”)

OpenStack Compute provisions and manages large networks of virtual machines. It is the backbone of OpenStack’s IaaS functionality. OpenStack Compute scales horizontally on standard hardware enabling the favorable economics of cloud computing. Users and administrators interact with the compute fabric via a web interface and command line tools.

Key features of OpenStack Compute include:

- Distributed and asynchronous architecture, allowing scale out fault tolerance for virtual machine instance management
- Management of commoditized virtual server resources, where predefined virtual hardware profiles for guests can be assigned to new instances at launch
- Tenants to separate and control access to compute resources
- VNC access to instances via web browsers

OpenStack Compute is composed of many services that work together to provide the full functionality. The **`openstack-nova-cert`** and **`openstack-nova-consoleauth`** services handle authorization. The **`openstack-nova-api`** responds to service requests and the **`openstack-nova-scheduler`** dispatches the requests to the message queue. The **`openstack-nova-conductor`** service updates the state database which limits direct access to the state database by compute nodes for increased security. The **`openstack-nova-compute`** service creates and terminates virtual machine instances on the compute nodes. Finally, **`openstack-nova-novncproxy`** provides a VNC proxy for console access to virtual machines via a standard web browser.

2.2.4 Block Storage (“Cinder”)

While the OpenStack Compute service provisions ephemeral storage for deployed instances based on their hardware profiles, the OpenStack Block Storage service provides compute instances with persistent block storage. Block storage is appropriate for performance sensitive scenarios such as databases or frequently accessed file systems. Persistent block storage can survive instance termination. It can also be moved between instances like an any external storage device. This service can be backed by a variety of enterprise storage platforms or simple NFS servers. This service’s features include:

- Persistent block storage devices for compute instances
- Self-service user creation, attachment, and deletion
- A unified interface for numerous storage platforms

- Volume snapshots

The Block Storage service is comprised of **`openstack-cinder-api`** which responds to service requests and **`openstack-cinder-scheduler`** which assigns tasks to the queue. The **`openstack-cinder-volume`** service interacts with various storage providers to allocate block storage for virtual machines. By default the Block Storage server shares local storage via the ISCSI **`tgtd`** daemon.

2.2.5 Network Service ("Neutron")

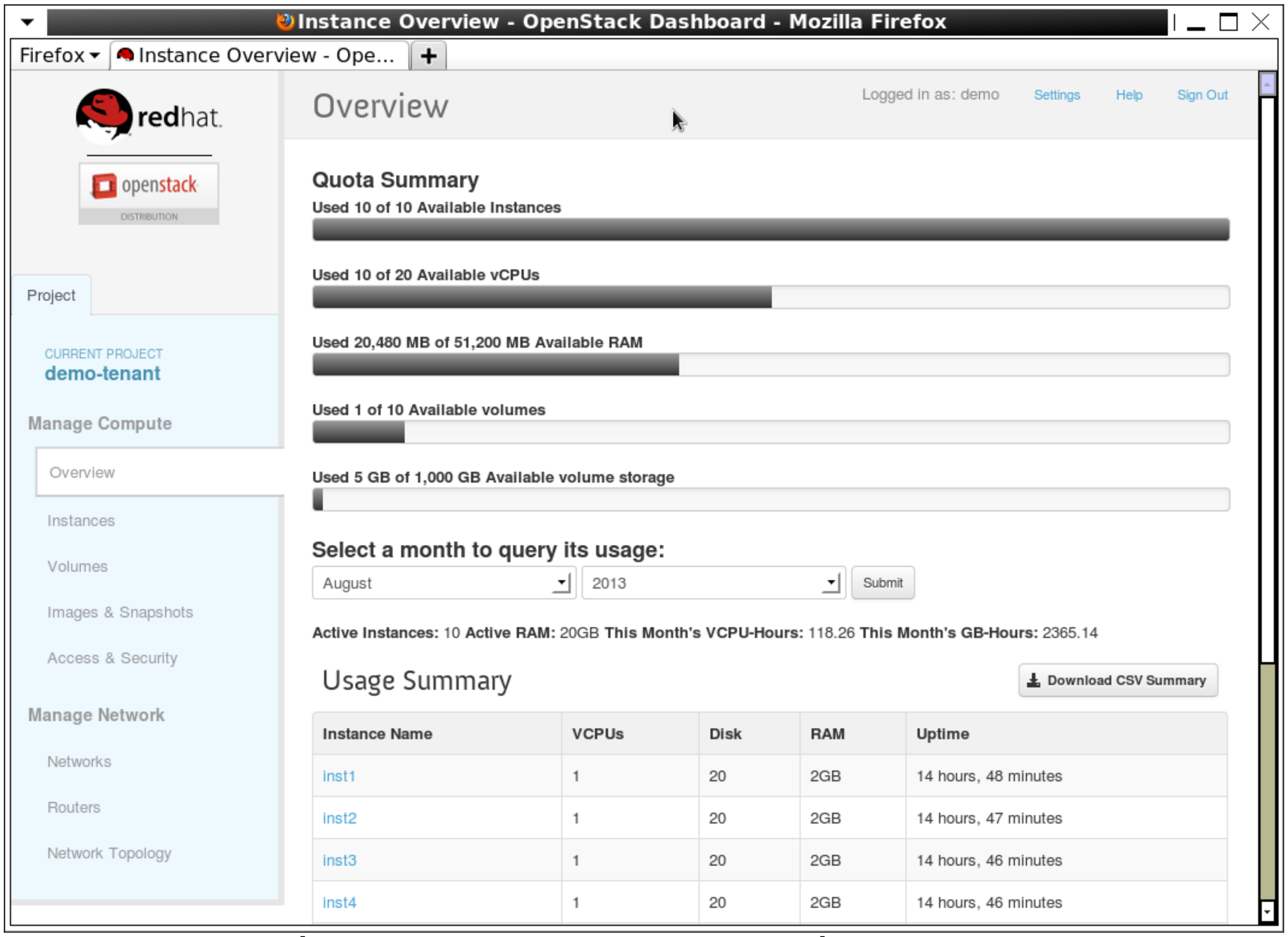
OpenStack Networking is a scalable API-driven service for managing networks and IP addresses. OpenStack Networking gives users self-service control over their network configurations. Users can define, separate, and join networks on demand. This allows for flexible network models that can be adapted to fit the requirements of different applications. OpenStack Networking has a pluggable architecture that supports numerous physical networking technologies as well as native Linux networking mechanisms including **`openvswitch`** and **`linuxbridge`**.

OpenStack Networking is composed of several services. The **`quantum-server`** exposes the API and responds to user requests. The **`quantum-l3-agent`** provides L3 functionality, such as routing, through interaction with the other networking plugins and agents. The **`quantum-dhcp-agent`** provides DHCP to tenant networks. There are also a series of network agents that perform local networking configuration for the node's virtual machines.

NOTE: In previous OpenStack versions the Network Service was named Quantum. In the Grizzly release Quantum was renamed to Neutron. However, many of the command line utilities in RHOS 3.0 retain the legacy name.

2.2.6 Dashboard ("Horizon")

The OpenStack Dashboard is an extensible web-based application that allows cloud administrators and users to control and provision compute, storage, and networking resources. Administrators can use the Dashboard to view the state of the cloud, create users, assign them to tenants, and set resource limits. **Illustration 2.2.6.1 Dashboard ("Horizon") System Panel** depicts the Horizon Dashboard Overview screen.



Quota Summary

Used 10 of 10 Available Instances

Used 10 of 20 Available vCPUs

Used 20,480 MB of 51,200 MB Available RAM

Used 1 of 10 Available volumes

Used 5 GB of 1,000 GB Available volume storage

Select a month to query its usage:

August | 2013

Active Instances: 10 Active RAM: 20GB This Month's VCPU-Hours: 118.26 This Month's GB-Hours: 2365.14

Usage Summary

Instance Name	VCPUs	Disk	RAM	Uptime
inst1	1	20	2GB	14 hours, 48 minutes
inst2	1	20	2GB	14 hours, 47 minutes
inst3	1	20	2GB	14 hours, 46 minutes
inst4	1	20	2GB	14 hours, 46 minutes

Illustration 2.2.6.1: Dashboard ("Horizon") System Panel

The OpenStack Dashboard runs as an Apache HTTP server via the **`httpd`** service.

> **NOTE:** Both the Dashboard and command line tools be can used to manage an OpenStack environment. This document focuses on the command line tools because they offer more granular control and insight into OpenStack's functionality.

2.2.7 Services Not Covered in this Reference Architecture

- **Metering Service ("Ceilometer")**

This service provides the infrastructure to collect measurements for monitoring and metering within OpenStack. Metering is necessary to enforce service level agreements and assess usage. In Red Hat Enterprise Linux OpenStack Platform 3, this service is tech preview, and not included in this reference architecture.

- **Orchestration Service ("Heat")**

  This service provides a REST API to orchestrate multiple composite cloud applications through a single template file. These templates allow for the creation of most OpenStack resource types such as virtual machine instances, floating IPs, volumes, and users. The Orchestration service is also tech preview in Red Hat Enterprise Linux OpenStack Platform 3 and not included in this reference architecture.

- **Object Storage Service ("Swift")**

  The OpenStack Object Storage service provides a fully distributed, API-accessible storage platform that can be integrated directly into applications or used for backup, archiving and data retention. It provides redundant, scalable object storage using clusters of standardized servers capable of storing petabytes of data. Object Storage is not a traditional file system, but rather a distributed storage system for static data. Objects and files are written to multiple disks spread throughout the data center. Storage clusters scale horizontally simply by adding new servers. The OpenStack Object Storage service is not discussed in this reference architecture. Red Hat Storage Server offers many of the core functionalities of this service.

2.3 Red Hat Enterprise Linux

Red Hat Enterprise Linux 6, the latest release of Red Hat's trusted datacenter platform, delivers advances in application performance, scalability, and security. With Red Hat Enterprise Linux 6, physical, virtual, and cloud computing resources can be deployed within the data center.

NOTE: This reference architecture is based on Red Hat Enterprise Linux 6.4. However, Red Hat Enterprise Linux OpenStack Platform 3 uses a non-standard kernel version **`2.6.32-358.114.1.openstack`** in order to support *NETWORK NAMESPACES*. Many of the robust features of OpenStack networking such as duplicate IP address ranges across tenants require network namespaces.

2.4 Supporting Technologies

This section describes the supporting technologies used to develop this reference architecture beyond the OpenStack services and core operating system. Supporting technologies include:

- MySQL
- Qpid
- KVM
- Packstack

- Red Hat Storage Server

The following sections describe each of these supporting technologies in greater detail.

2.4.1 MySQL

A state database resides at the heart of an OpenStack deployment. This SQL database stores most of the build-time and run-time state information for the cloud infrastructure including available instance types, networks, and the state of running instances in the compute fabric. Although OpenStack theoretically supports any SQL-Alchemy compliant database, Red Hat Enterprise Linux OpenStack Platform 3 uses MySQL, a widely used open source database packaged with Red Hat Enterprise Linux 6.

2.4.2 Qpid

Enterprise messaging systems let programs communicate by exchanging messages. OpenStack services use enterprise messaging to communicate tasks and state changes between clients, service endpoints, service schedulers, and instances. Red Hat Enterprise Linux OpenStack Platform 3 uses Qpid for open source enterprise messaging. Qpid is an Advanced Message Queuing Protocol (AMQP) compliant, cross-platform enterprise messaging system developed for low latency based on an open standard for enterprise messaging. Qpid is released under the Apache open source license.

2.4.3 KVM

Kernel-based Virtual Machine (KVM) is a full virtualization solution for Linux on x86 and x86_64 hardware containing virtualization extensions for both Intel and AMD processors. It consists of a loadable kernel module that provides the core virtualization infrastructure. Red Hat Enterprise Linux OpenStack Platform Compute uses KVM as its underlying hypervisor to launch and control virtual machine instances.

2.4.4 Red Hat Storage Server

Red Hat Storage Server (RHSS) is an enterprise storage solution that enables enterprise-wide storage sharing with a single access point across data storage locations. It has a scale-out, network-attach architecture to accommodate exponential data growth. Red Hat Enterprise Linux OpenStack Platform 3 does not depend on Red Hat Storage Server, but in this reference architecture RHSS is the back end storage for both the Block and Image Services. The Red Hat Storage client driver enables block storage support. Gluster volumes are used to store virtual machine images.

2.4.5 Packstack

Packstack is a Red Hat Enterprise Linux OpenStack Platform 3 installer. Packstack uses Puppet modules to install parts of OpenStack via SSH. Puppet modules ensure OpenStack can be installed and expanded in a consistent and repeatable manner. This reference architecture uses Packstack for a multi-server deployment. Through the course of this reference architecture, the initial Packstack installation is modified with OpenStack Network and Storage service enhancements.

2.4.6 Supporting Technologies Not Used in this Reference Architecture

- **Red Hat Enterprise Virtualization (RHEV)** -- The Red Hat Enterprise Virtualization portfolio is an end-to-end virtualization solution, with use cases for both servers and desktops, designed to overcome current IT challenges for consolidation, enable pervasive data center virtualization, and unlock unprecedented capital and operational efficiency. The Red Hat Enterprise Virtualization portfolio builds upon the Red Hat Enterprise Linux platform that is trusted by thousands of organizations on millions of systems around the world for their most mission-critical workloads. Red Hat Enterprise Linux OpenStack Platform 3 does not depend on Red Hat Enterprise Virtualization. Although it is possible to combine Red Hat Enterprise Virtualization and Red Hat Enterprise Linux OpenStack Platform in a single deployment, that course of action is not covered in this reference architecture.

3 Reference Architecture Configuration Details

This section of the paper describes the hardware, software, and procedures used to configure this reference architecture in the lab. Best practices learned in the lab are shared throughout this document.

3.1 Environment

The reference architecture environment consists of the components required to build a small Red Hat Enterprise Linux OpenStack Platform cloud infrastructure.

3.1.1 Network Topology

Illustration 3.1.1.1 Network Topology shows the network topology of this reference architecture.

- All eight servers and the client communicate via the lab network switch on the 10.16.136.0/21 network. This network is used for both client requests to the API and service communication between the OpenStack services. This topology mimics a private cloud where client and service communication do not need to be segmented for security reasons.

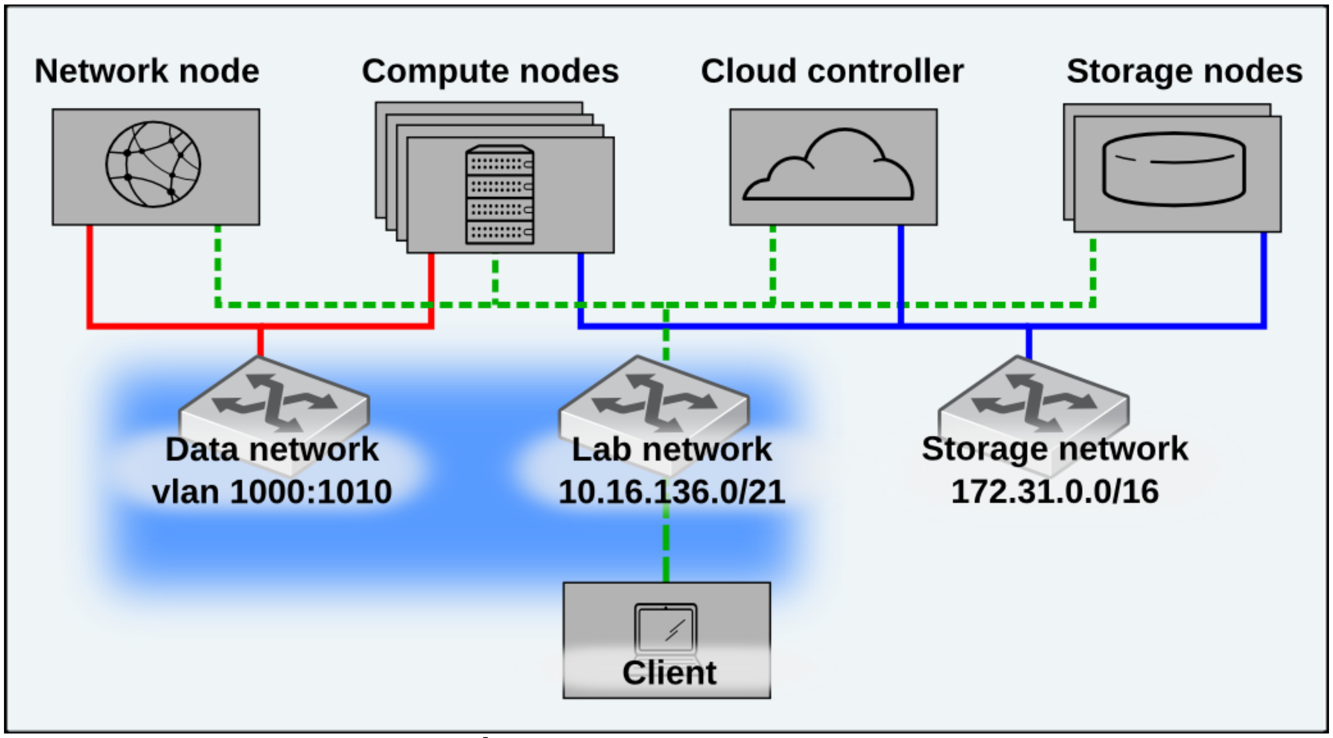


Illustration 3.1.1.1: Network Topology

- The network node and compute nodes are connected via a 10Gb switch on the Data network. This network carries the communication between virtual machines in the cloud and communications between the software-defined networking components.

Although the name Data network sounds as though it would carry storage traffic, this is the common OpenStack Networking term for the private network over which the instances communicate.

> **NOTE:** The Data network and Lab network are actually connected to the same physical switch, as depicted by the outline joining them. The Data network acts as a software-defined external network in Open vSwitch. Hosts on the Lab network can access instances on the Data network via floating IP address.

- The storage and compute nodes are also connected to a 10Gb storage network switch on the 172.31.0.0/16 network. This network delivers the Image service virtual machine disk images as well as the persistent block storage for the Block Storage service.

3.1.2 IP Addresses

Table 3.1.2.1: Host IP Addresses lists the IPv4 Addresses used in this reference architecture by server host name and role. The compute nodes have three interfaces on separate networks: lab, data, and storage. The storage server is connected to the lab and storage networks. The client is only connected to the lab network. The client simulates a system from which a self-service user would access the OpenStack infrastructure.

> **NOTE:** The data network carries virtual machine and software-defined network device traffic over tagged VLANs. The interfaces connected to this network are not configured with IPv4 addresses by the OpenStack administrator. Instead they act as bridges to software-defined network devices in network name spaces.

Host	Role	Network	Interface	Network Address
rhos0	Cloud Control	Lab	em1	10.16.137.100
rhos1	Client	Lab	em1	10.16.137.101
rhos2	Compute	Lab	em1	10.16.137.102
		Storage	p3p2	172.31.139.102
		Data	p3p1	VLAN 1000:1010
rhos3	Compute	Lab	em1	10.16.137.103
		Storage	p3p2	172.31.139.103
		Data	p3p1	VLAN 1000:1010
rhos4	Compute	Lab	em1	10.16.137.104
		Storage	p3p2	172.31.139.104
		Data	p3p1	VLAN 1000:1010
rhos5	Compute	Lab	em1	10.16.137.105
		Storage	p3p2	172.31.139.105
		Data	p3p1	VLAN 1000:1010

rhos6	Network	Lab	em1	10.16.137.106
		Data	p3p1	VLAN 1000:1010
rhos7	NFS	Lab	em1	10.16.137.107
ra-rhs-srv-10g-1	Storage	Storage	p1p2	172.31.143.91
ra-rhs-srv-10g-2	Storage	Storage	p1p2	172.31.143.92

Table 3.1.2.1: Host IP Addresses

3.2 Software and Security Reference

This section of the reference architecture lists the required software revisions. It also lists software configuration details related to security including **`SELinux`** and **`iptables`**. Customers who use the correct OpenStack and Red Hat Enterprise Linux channels on Red Hat Network (RHN) or Subscription Manager meet the minimum required software versions.

NOTE: At the time of writing, Red Hat Storage was only available via RHN or ISO download, not via Subscription Manager.

3.2.1 Software Versions

Table 3.2.1.1: Software Versions lists the software versions used to develop this reference architecture.

Host	Software	Version
Cloud Controller	qpid-cpp-server	0.14-22
	mysql-server	5.1.69-1
	openstack-packstack	2013.1.1-0.23.dev642
	openstack-keystone	2013.1.2-2
	openstack-nova-{api,cert,common,conductor,scheduler,console}	2013.1.2-4
	openstack-nova-novncproxy	0.4-6
	openstack-glance	2013.1.2-1
	openstack-cinder	2013.1.2-3
	openstack-dashboard	2012.2.3-9
	kernel{-firmware}	2.6.32-358.114.1.openstack
	openstack-utils	2013.1-8.1
	openstack-quantum	2013.1.2-4

	openstack-quantum-openvswitch	2013.1.2-4
	openstack-selinux	0.1.2-10
Compute Nodes	openstack-nova-{common,compute}	2013.1.2-4
	openstack-utils	2013.1-8.1
	kernel{-firmware}	2.6.32-358.114.1.openstack
	openstack-selinux	0.1.2-10
	openstack-quantum	2013.1.2-4
	openstack-quantum-openvswitch	2013.1.2-4
Client	python-keystoneclient	0.2.0-4
	python-glanceclient	0.8.0-4
	python-cinderclient	1.0.1-3
	python-novaclient	2.10.0-8
Network Server	openstack-utils	2013.1-8.1
	kernel{-firmware}	2.6.32-358.114.1.openstack
	openstack-selinux	0.1.2-10
	openstack-quantum	2013.1.2-4
	openstack-quantum-openvswitch	2013.1.2-4
NFS Server	kernel	2.6.32-358.114.1.openstack
	nfs-utils	1.2.3-36
RHS Server	glusterfs-{server,fuse}	3.3.0.7rhs-1
	kernel	2.6.32-220.32.1

Table 3.2.1.1: Software Versions

3.2.2 Security: iptables

Deploying Red Hat Enterprise Linux OpenStack Platform via **`packstack`** makes the appropriate firewall rules. **Table 3.2.2.1: Allowed iptables Ports by Role** lists the allowed ports by host, and role.

Port	Host	Role
22, 111, 3260, 3306, 5000, 5672, 6080, 8773:8776, 9292, 9696, 24007:24011, 35357	rhos0	Cloud controller
22, 80	rhos1	Client
22, 53, 67, 5900:5950	rhos2-5	Compute nodes
22, 443	rhos6	Network server
22, 2049	rhos7	NFS server
22, 111, 24007:24011, 38465-38469	ra-rhs-srv{1,2}	Storage server

Table 3.2.2.1: Allowed iptables Ports by Role

3.2.3 Security: SELinux

Red Hat Enterprise Linux OpenStack Platform supports **SELinux** in **enforcing** mode in Red Hat Enterprise Linux 6.4. **Table 3.2.3.1: Supported SELinux Package Versions** lists the required packages. **Table 3.2.3.2 SELinux Boolean Values** shows the SELinux Boolean values that must be set to **True** by server type.

Package	Version
libselinux	2.0.94-5.3.el6_4.1
selinux-policy	3.7.19-195.el6_4.12
selinux-policy-targeted	3.7.19-1953.7.19-195.el6_4.12
openstack-selinux	0.1.2-10

Table 3.2.3.1: Supported SELinux Package Versions

Server	Version
RHS Server	virt_use_fusefs
NFS Server	virt_use_nfs

Table 3.2.3.2: SELinux Boolean Values

3.2.4 Required Channels

Red Hat Enterprise Linux OpenStack Platform is available via the Red Hat Network channels and RHN Certificate Server repositories listed in **Table 3.2.4.1: Required Channels.**

Channel	Source
rhel-x86_64-server-6	RHN Classic
rhel-x86_64-server-6-ost-3	RHN Classic
rhel-x86_64-server-rhsclient-6	RHN Classic
rhel-6-server-rpms	RHN Certificate
rhel-server-ost-6-3-rpms	RHN Certificate

Table 3.2.4.1: Required Channels

NOTE: This reference architecture uses RHN Classic in all examples via a lab satellite server.

3.3 Server Hardware Configuration

Table 3.3.1: Server Hardware lists the hardware specifications for the servers used in this reference architecture. The bottom row lists the specifications for the Red Hat Storage servers. The top row lists the hardware specifications for all remaining OpenStack servers.

NOTE: Red Hat Enterprise Linux OpenStack Platform servers do not require identical hardware. The hardware used in this reference architecture meets the minimum requirements outlined in the OpenStack documentation.

Hardware	Specifications
8 x DELL BladeServer – PowerEdge M520	2 x Intel(R) Xeon(R) CPU E5-2450 0 @ 2.10GHz (8 core)
	2 x Broadcom NetXtreme II BCM57810 10 Gigabit Ethernet 4 x Broadcom NetXtreme BCM5720 Gigabit Ethernet
	8 x DDR3 4096 MB @1600 MHZ DIMMs
	2 x 146GB SAS internal disk drives
2 x DELL PowerEdge R510	2 x Intel Xeon(R) CPU 5650 @ 2.67GHz (6 core)
	2 x Broadcom NetXtreme II BCM5716 GB Ethernet
	6 x DDR3 8192 MB @1333 MHz DIMMs
	12 x 1TB SAS internal disk drives 2 x 500 GB SAS internal disk drives

Table 3.3.1: Server Hardware

3.4 OpenStack Service Placement

Table 3.4.1: Service Placement shows the final service placement for all OpenStack services. The API-listener services (including **`quantum-server`**) run on the cloud controller in order to field client requests. The Network node runs all other Network services except for those necessary for Nova client operations, which also run on the Compute nodes. The Dashboard runs on the client system to prevent self-service users from accessing the cloud controller directly.

Host	Role	Service
rhos0	Cloud Controller	openstack-cinder-api
		openstack-cinder-scheduler
		openstack-cinder-volume
		openstack-glance-api
		openstack-glance-registry
		openstack-glance-scrubber
		openstack-keystone
		openstack-nova-api
		openstack-nova-cert
		openstack-nova-conductor
		openstack-nova-consoleauth
		openstack-nova-novncproxy
		openstack-nova-scheduler
		quantum-server
rhos1	Client	httpd (Dashboard)
rhos2-5	Compute node	openstack-nova-compute
rhos2-6	Network / Compute node	quantum-openvswitch-agent
		quantum-ovs-cleanup
rhos6	Network node	quantum-l3-agent
		quantum-dhcp-agent
		quantum-metadata-agent

Table 3.4.1: Service Placement

4 Deploy Red Hat Enterprise Linux OpenStack Platform 3 via Packstack

This section of the paper describes the steps required to install Red Hat Enterprise Linux OpenStack Platform 3 via **`packstack`**. The initial deployment is later expanded to include Red Hat Storage Server. The default OpenStack Network service configuration is also modified to include additional networks.

> **NOTE:** Manual modifications to the **`packstack`** deployment are supported by Red Hat.

4.1 Architectural Overview

Illustration 4.1.1 Initial Deployment shows the Red Hat Enterprise Linux OpenStack Platform server roles by host.

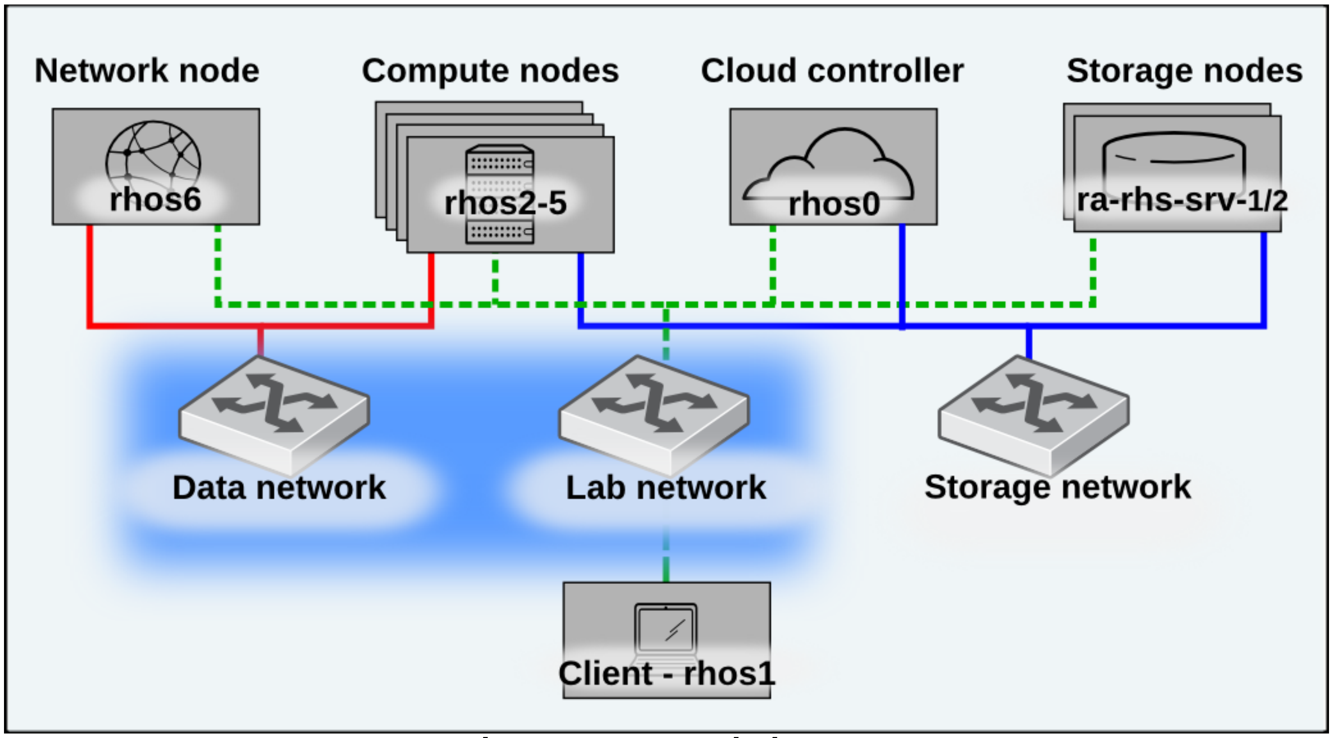


Illustration 4.1.1: Initial Deployment

In this reference architecture Red Hat Enterprise Linux OpenStack Platform servers are defined by the following roles:

1. Cloud controller
2. Compute nodes
3. Network node

4. Storage servers

4.1.1 Cloud Controller

Cloud controller is the designation used in this reference architecture for the server that provides the endpoint for the REST-based API queries. These include the Compute, Image, Block, and Identity. Even though there is a dedicated Network node, the cloud controller also runs the **`quantum-server`** service which listens for connection requests to the OpenStack Networking service. This isolates the network server by preventing access directly to it by external clients. The cloud controller also updates the state database and hosts the messaging system. With OpenStack's scale-out architecture these services can be shared across one or more servers.

4.1.2 Compute Node

Compute Node refers to an OpenStack server that runs a KVM hypervisor. It is responsible for creating virtual machine instances. These worker nodes field requests from the message queue, perform a series of commands, and update the state database. The **`openstack-nova-scheduler`** service balances instance deployments across compute nodes. The scheduler load balancing policy is configurable. By default it assigns new instances to the nodes with the most free memory in a round robin fashion.

4.1.3 Network Node

In Red Hat's Folsom-based release of OpenStack 2.1, networking was provided by the **`openstack-nova-network`** service. **`Openstack-nova-network`** is a sub-service of OpenStack Compute that controls basic networking functionality. Compute nodes could run local copies of the network service in multi-host mode for redundancy and localization.

In Red Hat Enterprise Linux OpenStack Platform 3, the Neutron network service replaces Nova Networking which has been deprecated. This change affords users with the ability to define their own networks on demand. It also allows for a range of previously unsupported network technologies and topologies. The OpenStack Network services run on a dedicated server that provides centralized networking control for all compute nodes and tenants.

4.1.4 Storage Server

The storage servers present the underlying storage used by the Block Storage and Image services. In this reference architecture the storage is delivered by a redundant pair of Red Hat storage servers. The shares are mounted to the compute and controller nodes via software clients.

4.1.5 Client

System administrators and self-service users access OpenStack service APIs via the *Client*. It only has access to the cloud controller via the public network. **`Packstack`** installs command line tools on the client. In this reference architecture the OpenStack dashboard is also installed on the client, although it could theoretically be installed to any system that can communicate with the Identity Service endpoints. Installing the Dashboard on a client system provides an additional layer of security for the cloud controller.

4.2 Prepare the Hosts

Perform the following steps on the cloud controller prior to deploying Red Hat Enterprise Linux OpenStack Platform 3:

1. Install Red Hat Enterprise Linux 6.4
2. Configure name resolution
3. Configure SSH keys

4.2.1 Install Red Hat Enterprise Linux 6.4

Red Hat Enterprise Linux OpenStack Platform 3 requires Red Hat Enterprise Linux 6.4 with Eratta.

```
[root@rhos0 ~]# cat /etc/redhat-release
Red Hat Enterprise Linux Server release 6.4 (Santiago)

[root@rhos0 ~]# uname --all
Linux rhos0.example.com 2.6.32-358.0.1.el6.x86_64 #1 SMP Wed Feb 20 11:05:23
EST 2013 x86_64 x86_64 x86_64 GNU/Linux
```

NOTE: Instructions for registering servers with RHN via the **`rhn_register`** command can be found in chapter 5 of the Red Hat Network Satellite Reference Guide.

4.2.2 Configure Name Resolution

The examples used in this reference architecture require forward and reverse name resolution either via DNS or a static hosts file. A static file was used while developing this reference architecture. Every interface on every host is resolvable by name except the interfaces on the Compute nodes used by the OpenStack Networking service. In the lab environment DNS was used. This example file is provided for completeness.

```
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1  localhost localhost.localdomain localhost6 localhost6.localdomain6
10.16.137.100    rhos0
172.31.139.100   rhos0-stor
10.16.137.101    rhos1
10.16.137.102    rhos2
172.31.139.102   rhos2-stor
10.16.137.103    rhos3
172.31.139.103   rhos3-stor
10.16.137.104    rhos4
172.31.139.104   rhos4-stor
10.16.137.105    rhos5
172.31.139.105   rhos5-stor
```

```
10.16.137.106    rhos6
172.31.143.91    ra-rhs-srv1-10g
172.31.143.92    ra-rhs-srv2-10g
```

4.2.3 Create an SSH Key on the Cloud Controller

This section shows how to set up password-less SSH logins. Packstack uses SSH for authentication and file transfer.

> **NOTE:** This step is not necessary as Packstack prompts for SSH credentials during installation. However, it is easier to prepare the servers if SSH keys are already in place.

1. Create an SSH key on the cloud controller.

```
[root@rhos0 ~]# ssh-keygen -t rsa -N ""  -f /root/.ssh/id_rsa
Generating public/private rsa key pair.
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
e3:ed:34:50:11:b6:01:10:b7:d1:30:9d:c9:2f:ec:9d root@rhos0.example.com
The key's randomart image is:
+--[ RSA 2048]----+
|      oo**=+     |
|       . =*+     |
|        ..o.     |
|         .o .    |
|        S. o .   |
|       . +. E    |
|        . +      |
|         o .     |
|          .      |
+-----------------+

[root@rhos0 ~]# ls -al .ssh
total 28
drwx------. 2 root root 4096 Mar 14 11:46 .
dr-xr-x---. 4 root root 4096 Mar 14 11:45 ..
-rw-------. 1 root root 1675 Mar 14 11:47 id_rsa
-rw-r--r--. 1 root root  421 Mar 14 11:47 id_rsa.pub
-rw-r--r--. 1 root root  403 Mar 14 11:17 known_hosts
```

2. Disable SSH strict host key checking.

```
[root@rhos0 ~]# echo "StrictHostKeyChecking no" >> /root/.ssh/config
```

> **NOTE:** Strict host key checking should remain enabled in production environments. It was disabled in this reference architecture in order to streamline remote operations.

3. Copy the SSH key to the servers with **`ssh-copy-id`**.

```
[root@rhos0 ~]# i=6; while [ $i -ge 0 ]; do ssh-copy-id -i
```

```
/root/.ssh/id_rsa.pub rhos$i; ((i--)); done
Warning: Permanently added 'rhos6,10.16.137.106' (RSA) to the list of known
hosts.
root@rhos6's password: ********
Now try logging into the machine, with "ssh 'rhos6'", and check in:

  .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.

Warning: Permanently added 'rhos5,10.16.137.105' (RSA) to the list of
knownhosts.
root@rhos5's password: ********
Now try logging into the machine, with "ssh 'rhos5'", and check in:

  .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.

Warning: Permanently added 'rhos4,10.16.137.104' (RSA) to the list of known
hosts.
root@rhos4's password: ********
Now try logging into the machine, with "ssh 'rhos4'", and check in:

  .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.

Warning: Permanently added 'rhos3,10.16.137.103' (RSA) to the list of known
hosts.
root@rhos3's password: ********
Now try logging into the machine, with "ssh 'rhos3'", and check in:

  .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.

Warning: Permanently added 'rhos2,10.16.137.102' (RSA) to the list of known
hosts.
root@rhos2's password: ********
Now try logging into the machine, with "ssh 'rhos2'", and check in:

  .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.

Warning: Permanently added 'rhos1,10.16.137.101' (RSA) to the list of known
hosts.
root@rhos1's password: ********
Now try logging into the machine, with "ssh 'rhos1'", and check in:

  .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.
```

```
Warning: Permanently added 'rhos0,10.16.137.100' (RSA) to the list of known
hosts.
Now try logging into the machine, with "ssh 'rhos0'", and check in:

  .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.
```

4.3 Deploy via Packstack

`Packstack` is an interactive or scriptable installer whose answers define the Red Hat Enterprise Linux OpenStack Platform configuration to be applied via Puppet modules. It was used in this reference architecture to deploy the cloud controller, the Compute nodes, the network server, and the OpenStack client tools. The storage servers were not deployed with **`packstack`** because it does not include this capability.

NOTE: Manual changes to the **`packstack`** deployment, including adding storage servers, are overwritten if **`packstack`** is run again.

4.3.1 Prepare the Cloud Controller for Packstack

Red Hat Enterprise Linux OpenStack Platform and **`packstack`** are distributed through RHN and Subscription Manager.

NOTE: Register the cloud controller with either Red Hat Network and Red Hat Certificate Server. This document shows how to do both for completeness.

1. Register the cloud controller with the Red Hat Network.

```
[root@rhos0 ~]# rhnreg_ks --force --username admin --password password

[root@rhos0 ~]# rhn-channel --list
rhel-x86_64-server-6
```

2. Add the OpenStack RHN Classic channel with the **`rhn-channel`** command.

```
[root@rhos0 ~]# rhn-channel --user labadmin --password labpasswd --add
--channel rhel-x86_64-server-6-ost-3

[root@rhos0 ~]# rhn-channel --list
rhel-x86_64-server-6
rhel-x86_64-server-6-ost-3
```

3. Alternately, add the OpenStack subscription with **`subscription-manager`**.

```
[root@rhos0 ~]# subscription-manager register --username labadmin --password
labpasswd --auto-attach
```

```
[root@rhos0 ~]# yum-config-manager --enable rhel-server-ost-6-3-rpms

[root@rhos0 ~]# yum repolist | grep ost
rhel-server-ost-6-3-rpms   Red Hat Enterprise Linux OpenStack Platform 3
(RPMs)        590
```

4. Install **openstack-packstack** on the cloud controller with **yum**.

```
[root@rhos0 ~]# yum install --assumeyes --quiet openstack-packstack

[root@rhos0 ~]# yum list openstack-packstack
Loaded plugins: product-id, refresh-packagekit, rhnplugin, security,
subscription-manager
Installed Packages
openstack-packstack.noarch 2013.1.1-0.23.dev642.el6ost
@rhel-server-ost-6-3-rpms
```

4.3.2 Run Packstack on the Cloud Controller

Run **packstack** on the cloud controller. Accept all defaults aside from those highlighted in the example output. Answers are stored in a time-stamped answer file. This answer file must be used as input for subsequent runs in order to retain access credentials. Otherwise the entire infrastructure must be completely reinstalled.

NOTE: The default answers assume **packstack** is running on the cloud controller. Inline comments describe the non-default answers used in this example.

1. Run **packstack**.

```
[root@rhos0 ~]# packstack
Welcome to Installer setup utility
```

2. Accept the default. The SSH Public key was installed in step **4.2.3 Create an SSH Key on the Cloud Controller**.

```
Enter the path to your ssh Public key to install on servers
[/root/.ssh/id_rsa.pub] :
```

3. Select services to install. Accept the defaults.

```
Should Packstack install Glance image service [y|n]  [y] :
Should Packstack install Cinder volume service [y|n]  [y] :
Should Packstack install Nova compute service [y|n]  [y] :
Should Packstack install Quantum network service [y|n]  [y] :
Should Packstack install Horizon dashboard [y|n]  [y] :
Should Packstack install Swift object storage [y|n]  [n] :
Should Packstack install OpenStack client tools [y|n]  [y] :
```

4. List **`ntpd`** servers.

```
Enter a comma separated list of NTP server(s). Leave plain if Packstack
Should not install ntpd on instances.: 10.16.255.2,10.16.255.3
```

5. Define service placement. The first few services are installed on the cloud controller.

```
Should Packstack install Nagios to monitor openstack hosts [y|n]  [n] :
Enter the IP address of the MySQL server  [10.16.137.100] :
Enter the password for the MySQL admin user : ********
Enter the IP address of the Qpid service  [10.16.137.100] :
Enter the IP address of the Keystone server  [10.16.137.100] :
Enter the IP address of the Glance server  [10.16.137.100] :
Enter the IP address of the Cinder server  [10.16.137.100] :
```

6. Create a 1G Cinder volume group to provide a default when Cinder is installed. This is deleted in subsequent steps when Red Hat Storage is used in place of the volume group.

```
Should Cinder's volumes group be created (for proof-of-concept installation)?
[y|n]  [y] :
Enter Cinder's volumes group size  [20G] : 1G
```

7. Place the **`nova-compute`** service on the compute nodes and all other Compute services on the cloud controller.

```
Enter the IP address of the Nova API service  [10.16.137.100] :
Enter the IP address of the Nova Cert service  [10.16.137.100] :
Enter the IP address of the Nova VNC proxy  [10.16.137.100] :
Enter a comma separated list of IP addresses on which to install the Nova
Compute services  [10.16.137.100] :
10.16.137.102,10.16.137.103,10.16.137.104,10.16.137.105
Enter the IP address of the Nova Conductor service  [10.16.137.100] :
Enter the IP address of the Nova Scheduler service  [10.16.137.100] :
```

8. Accept the default CPU and RAM overcommitment ratios. The KVM hypervisor supports over committing CPUs and memory. Over committing is the process of allocating more virtualized CPUs or memory than there are physical resources on the system. CPU over commit allows under-utilized virtualized servers to run on fewer servers.

   NOTE: Although the defaults were accepted in the lab environment, do not over commit memory or CPUs in a production environment without extensive testing. Applications which use 100% of memory or processing resources may become unstable in over committed environments. Test before deploying.

```
Enter the CPU overcommitment ratio. Set to 1.0 to disable CPU overcommitment
[16.0] :
Enter the RAM overcommitment ratio. Set to 1.0 to disable RAM overcommitment
```

```
[1.5] :
```

9. Install the Quantum Server on the cloud controller. It should be co-resident with the other API listeners.

```
Enter the IP address of the Quantum server  [10.16.137.100] : 10.16.137.100
Should Quantum use network namespaces? [y|n]  [y] : y
```

10. Install the L3 and DHCP agents on the network server.

```
Enter a comma separated list of IP addresses on which to install the Quantum
L3 agent  [10.16.137.100] : 10.16.137.106
```

11. The Quantum L3 agent should use a provider network for external traffic. A provider networks maps an external network directly to a physical network. This gives tenants direct access to a public network.

```
Enter the bridge the Quantum L3 agent will use for external traffic, or
'provider' if using provider networks  [br-ex] : provider
```

12. The DHCP agent should also be on the network server. This agent assigns IP addresses to instances via DHCP.

```
Enter a comma separated list of IP addresses on which to install Quantum DHCP
agent  [10.16.137.100] : 10.16.137.106
```

13. Accept the default Open vSwitch plugin. Open vSwitch runs as a software -defined switch within KVM on the Compute nodes. It provides robust networking capability to the instances.

```
Enter the name of the L2 plugin to be used with Quantum [linuxbridge|
openvswitch]  [openvswitch] :
```

14. Install the metadata agent on the network server. The metadata agent listens for customization requests and forwards them to **`openstack-nova-api`**.

```
Enter a comma separated list of IP addresses on which to install the Quantum
metadata agent  [10.16.137.100] : 10.16.137.106
```

15. Allocate VLAN networks for tenant networks. Tenant flows are separated internally by an internally assigned VLAN ID. GRE is not supported in this version of Red Hat Enterprise Linux OpenStack Platform.

```
Enter the type of network to allocate for tenant networks [local|vlan|gre]
[local] : vlan
```

16. Assign a VLAN range for the **`openvswitch`** plugin. This range of tagged VLANs must be enabled on the physical switches that carry the OpenStack Network service traffic. Tenant traffic is converted to a physical VLAN ID as it traverses external bridge

interface. For example, the VLAN range must be configured on the switch that carries communication between instances in the same tenant that reside on different Compute nodes.

```
Enter a comma separated list of VLAN ranges for the Quantum openvswitch
plugin: physnet:1000:1010
```

17. Enter the bridge mapping for the **`openvswitch`** plugin. An **`openvswitch`** bridge acts like a virtual switch. Network interface devices connect to **`openvswitch`** bridge's ports. The ports can be configured like a physical switch's ports including VLAN configurations.

```
Enter a comma separated list of bridge mappings for the Quantum openvswitch
plugin: physnet:br-p3p1
```

18. Add the **`p3p1`** interface to the **`br-p3p1`** bridge. This maps the **`openvswitch`** bridge to the physical interface on the server.

```
Enter a comma separated list of OVS bridge:interface pairs for the Quantum
openvswitch plugin: br-p3p1:p3p1
```

19. Install the client tools and Dashboard web interface on the client system. HTTPS is not required for Horizon communication. This reference architecture assumes OpenStack is deployed as a private cloud. Enable HTTPS for a public cloud.

```
Enter the IP address of the client server  [10.16.137.100] : 10.16.137.101
Enter the IP address of the Horizon server  [10.16.137.100] : 10.16.137.101
Would you like to set up Horizon communication over https [y|n]  [n] :
```

20. Enter RHN account information if the servers are not already registered. This account information is propagated to the servers by **`packstack`**. In this reference architecture the servers were registered to a RHN classic satellite server during a kickstart installation.

```
To subscribe each server to EPEL enter "y" [y|n]  [n] :
Enter a comma separated list of URLs to any additional yum repositories to
install:
To subscribe each server to Red Hat enter a username here:
To subscribe each server to Red Hat enter your password here :
To subscribe each server to Red Hat Enterprise Linux 6 Server Beta channel
(only needed for Preview versions of RHOS) enter "y" [y|n]  [n] :
To subscribe each server with RHN Satellite enter RHN Satellite server URL:
```

4.3.3 Verify the configuration.

`Packstack` displays a configuration summary at completion. Type **yes** to accept the configuration and install Red Hat Enterprise Linux OpenStack Platform components.

```
Installer will be installed using the following configuration:
==============================================================
ssh-public-key:                /root/.ssh/id_rsa.pub
os-glance-install:             y
os-cinder-install:             y
os-nova-install:               y
os-quantum-install:            y
os-horizon-install:            y
os-swift-install:              n
os-client-install:             y
ntp-severs:                    10.16.255.2,10.16.255.3
nagios-install:                n
mysql-host:                    10.16.137.100
mysql-pw:                      ********
qpid-host:                     10.16.137.100
keystone-host:                 10.16.137.100
glance-host:                   10.16.137.100
cinder-host:                   10.16.137.100
cinder-volumes-create:         y
cinder-volumes-size:           1G
novaapi-host:                  10.16.137.100
novacert-host:                 10.16.137.100
novavncproxy-hosts:            10.16.137.100
novacompute-hosts:
10.16.137.102,10.16.137.103,10.16.137.104,10.16.137.105
novaconductor-host:            10.16.137.100
novasched-host:                10.16.137.100
novasched-cpu-allocation-ratio:16.0
novasched-ram-allocation-ratio:1.5
quantum-server-host:           10.16.137.100
quantum-use-namespaces:        y
quantum-l3-hosts:              10.16.137.106
quantum-l3-ext-bridge:         provider
quantum-dhcp-hosts:            10.16.137.106
quantum-l2-plugin:             openvswitch
quantum-metadata-hosts:        10.16.137.106
quantum-ovs-tenant-network-type:vlan
quantum-ovs-vlan-ranges:       physnet1:1000:1010
quantum-ovs-bridge-mappings:   physnet1:br-p3p1
quantum-ovs-bridge-interfaces: br-p3p1:p3p1
osclient-host:                 10.16.137.101
os-horizon-host:               10.16.137.101
os-horizon-ssl:                n
use-epel:                      n
additional-repo:
rh-username:
rh-password:
rh-beta-repo:                  n
rhn-satellite-server:
Proceed with the configuration listed above? (yes|no): yes
```

NOTE: `packstack` requires root authentication to copy SSH keys. Enter the root password when prompted.

4.3.4 Verify Packstack Installation Completes Successfully

A successful **`packstack`** installation displays the following output. The bottom of the command output includes tips for accessing OpenStack via the command line and Dashboard.

```
 **** Installation completed successfully ******


Additional information:
 * A new answerfile was created in: /root/packstack-answers-20130722-
141712.txt
 * To use the command line tools you need to source the file
/root/keystonerc_admin created on 10.16.137.101
 * To use the console, browse to http://10.16.137.101/dashboard
 * Kernel package with netns support has been installed on host
10.16.137.100. Because of the kernel update host mentioned above requires
reboot.
 * Kernel package with netns support has been installed on host
10.16.137.106. Because of the kernel update host mentioned above requires
reboot.
 * Kernel package with netns support has been installed on host
10.16.137.103. Because of the kernel update host mentioned above requires
reboot.
 * Kernel package with netns support has been installed on host
10.16.137.105. Because of the kernel update host mentioned above requires
reboot.
 * Kernel package with netns support has been installed on host
10.16.137.101. Because of the kernel update host mentioned above requires
reboot.
 * Kernel package with netns support has been installed on host
10.16.137.104. Because of the kernel update host mentioned above requires
reboot.
 * Kernel package with netns support has been installed on host
10.16.137.102. Because of the kernel update host mentioned above requires
reboot.
 * The installation log file is available at: /var/tmp/packstack/20130722-
141350-nLirm2/openstack-setup.log
```

4.3.5 Configure p3p1 to Start Automatically at Boot

Configure the *p3p1* interface on the cloud controller, Network, and Compute nodes to start on boot. This ensures the bridge devices are available after reboot.

1. Change **ONBOOT** from **no** to **yes** and **BOOTPROTO** from **dhcp** to **none**.

```
[root@rhos0 scripts]# for i in 6 5 4 3 2 0; do ssh rhos$i “sed -i
-e's/BOOTPROTO=.*dhcp.*/BOOTPROTO=none/' -e's/ONBOOT=no/ONBOOT=yes/'
/etc/sysconfig/network-scripts/ifcfg-p3p1”; done
```

2. Verify the changes.

```
[root@rhos0 ~]# for i in 6 5 4 3 2 0; do ssh rhos$i grep BOOT
/etc/sysconfig/network-scripts/ifcfg-p3p1; done
ONBOOT=yes
```

```
BOOTPROTO=none
ONBOOT=yes
BOOTPROTO=none
ONBOOT=yes
BOOTPROTO=none
ONBOOT=yes
BOOTPROTO=none
ONBOOT=yes
BOOTPROTO=none
ONBOOT=yes
BOOTPROTO=none
```

4.3.6 Reboot the Servers

1. Reboot the OpenStack servers into the netns-aware kernel.

```
[root@rhos0 ~]# for i in 6 5 4 3 2 1 0; do ssh rhos$i shutdown -r now; done

Broadcast message from root@rhos0.cloud.lab.eng.bos.redhat.com
      (unknown) at 14:57 ...

The system is going down for reboot NOW!
```

2. Verify the servers are booted to the OpenStack kernel.

```
[root@rhos0 ~]# for i in 6 5 4 3 2 1 0; do ssh rhos$i uname -r; done
2.6.32-358.114.1.openstack.el6.x86_64
2.6.32-358.114.1.openstack.el6.x86_64
2.6.32-358.114.1.openstack.el6.x86_64
2.6.32-358.114.1.openstack.el6.x86_64
2.6.32-358.114.1.openstack.el6.x86_64
2.6.32-358.114.1.openstack.el6.x86_64
2.6.32-358.114.1.openstack.el6.x86_64
```

4.4 Verify the Packstack Deployment

This section describes steps for verifying the **`packstack`** deployment.

4.4.1 Initial Deployment Overview

Illustration 4.4.1.1 Initial Deployment shows the service placement after deployment.

The **`nova-compute`** service is running on the compute nodes. The Horizon dashboard and client tools are installed on the client server. The network services are running on the network server. All other installed OpenStack services are running on the cloud controller.

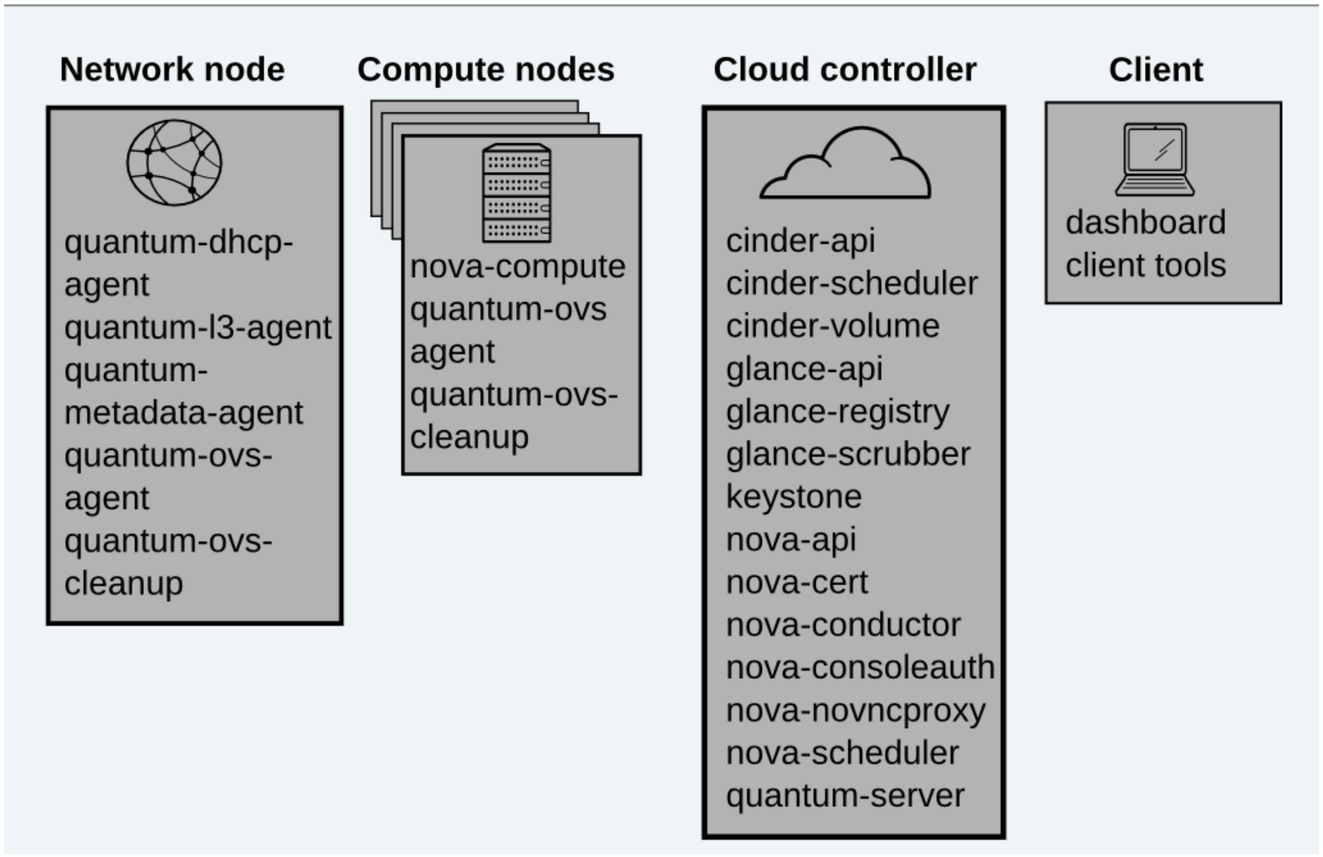


Illustration 4.4.1.1: Initial Deployment

4.4.2 Examine Deployment as Keystone Admin

1. Verify all servers are subscribed to the appropriate software channels.

```
[root@rhos0 ~]# for i in 0 1 2 3 4 5 6; do ssh rhos$i rhn-channel --list; done
rhel-x86_64-server-6
rhel-x86_64-server-6-ost-3
rhel-x86_64-server-6
rhel-x86_64-server-6-ost-3
rhel-x86_64-server-6
rhel-x86_64-server-6-ost-3
rhel-x86_64-server-6
rhel-x86_64-server-6-ost-3
rhel-x86_64-server-6
rhel-x86_64-server-6-ost-3
rhel-x86_64-server-6
rhel-x86_64-server-6-ost-3
rhel-x86_64-server-6
rhel-x86_64-server-6-ost-3
```

2. Verify **ntpd** is configured and running on all servers.

```
[root@rhos0 ~]# for i in 0 1 2 3 4 5 6; do ssh rhos$i "grep ^server
/etc/ntp.conf && service ntpd status"; done
server 10.16.255.2
server 10.16.255.3
ntpd (pid  7313) is running...
server 10.16.255.2
server 10.16.255.3
ntpd (pid  6954) is running...
server 10.16.255.2
server 10.16.255.3
ntpd (pid  8127) is running...
server 10.16.255.2
server 10.16.255.3
ntpd (pid  8127) is running...
server 10.16.255.2
server 10.16.255.3
ntpd (pid  8058) is running...
server 10.16.255.2
server 10.16.255.3
ntpd (pid  7983) is running...
server 10.16.255.2
server 10.16.255.3
ntpd (pid  6917) is running...
```

3. Run **openstack-status** to check status of the services on the cloud controller.

```
[root@rhos0 ~]# openstack-status
== Nova services ==
openstack-nova-api:           active
openstack-nova-cert:          active
openstack-nova-compute:       dead (disabled on boot)
openstack-nova-network:       dead (disabled on boot)
openstack-nova-scheduler:     active
openstack-nova-volume:        dead (disabled on boot)
openstack-nova-conductor:     active
== Glance services ==
openstack-glance-api:         active
openstack-glance-registry:    active
== Keystone service ==
openstack-keystone:           active
== Cinder services ==
openstack-cinder-api:         active
openstack-cinder-scheduler:   active
openstack-cinder-volume:      active
== Support services ==
mysqld:                       active
tgtd:                         active
qpidd:                        active
== Keystone users ==
Warning keystonerc not sourced
```

NOTE: By default **packstack** installs the *keystonerc_admin* file on the client server. Copy this file to the cloud controller and source it before running **openstack-status** to display additional information. The **openstack-status** command is not installed with the client tools. It is also helpful to have this file on the network server.

4. Connect to the client via SSH and source the *keystonerc_admin* file.

```
[root@rhos0 ~]# ssh rhos1
Last login: Mon Jul 22 15:21:36 2013 from rhos0
Kickstarted on 2013-07-22

[root@rhos1 ~]# source /root/keystonerc_admin

[root@rhos1 ~(keystone_admin)]# env | grep OS_
OS_PASSWORD=ed7c0a23121e4983
OS_AUTH_URL=http://10.16.137.100:35357/v2.0/
OS_USERNAME=admin
OS_TENANT_NAME=admin
```

5. View the status and placement of compute services with **nova service-list**.

```
[root@rhos1 ~(keystone_admin)]# nova service-list
+------------------+------------------------------------+------------------+
| Binary           | Host   | Zone     | Status   | State | Updated_at
+------------------+------------------------------------+------------------+
| nova-cert        | rhos0.cloud.lab.eng.bos.redhat.com | internal | enabled
| up    | 2013-07-22T20:24:29.000000 |
| nova-compute     | rhos2.cloud.lab.eng.bos.redhat.com | nova     | enabled
| up    | 2013-07-22T20:24:29.000000 |
| nova-compute     | rhos3.cloud.lab.eng.bos.redhat.com | nova     | enabled
| up    | 2013-07-22T20:24:33.000000 |
| nova-compute     | rhos4.cloud.lab.eng.bos.redhat.com | nova     | enabled
| up    | 2013-07-22T20:24:35.000000 |
| nova-compute     | rhos5.cloud.lab.eng.bos.redhat.com | nova     | enabled
| up    | 2013-07-22T20:24:35.000000 |
| nova-conductor   | rhos0.cloud.lab.eng.bos.redhat.com | internal | enabled
| up    | 2013-07-22T20:24:29.000000 |
| nova-consoleauth | rhos0.cloud.lab.eng.bos.redhat.com | internal | enabled
| up    | 2013-07-22T20:24:29.000000 |
| nova-scheduler   | rhos0.cloud.lab.eng.bos.redhat.com | internal | enabled
| up    | 2013-07-22T20:24:29.000000 |
+------------------+------------------------------------+------------------+
```

6. Verify the *p3p1* link is UP on the controller, compute, and network servers.

```
[root@rhos0 ~]# for i in 6 5 4 3 2 0; do ssh rhos$i ip link show p3p1 | grep
UP; done
2: p3p1:<BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
2: p3p1:<BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
2: p3p1:<BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
2: p3p1:<BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
2: p3p1:<BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
```

```
2: p3p1:<BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
```

5 Deploy Red Hat Storage Server

Cinder is the OpenStack block storage service. It provides volume and snapshot creation, deletion, and attachment. Cinder volumes play an important role in an OpenStack deployment. They provide persistent storage that can survive instance deletion.

> **NOTE:** Red Hat Storage is not required with Red Hat Enterprise Linux OpenStack Platform. The Image and Block Storage services support many backend storage drivers. **Appendix B: NFS Storage Setup** includes steps for configuring NFS-backed services.

By default **`packstack`** creates a volume group on the Block Storage server's local storage that is shared as an iSCSI target via **`tgtd`**. This section of the reference architecture shows how to convert the local Block Storage server to an RHS-backed server using a dedicated storage network. The cloud controller runs the core Block Storage services including **`cinder-api`**, **`cinder-scheduler`**, and **`cinder-volume`**.

> **NOTE:** These customizations are over-written and must be manually recreated if **`packstack`** is run again on the same servers.

5.1 Build a Red Hat Storage Server

The RHS cluster is composed of two servers. Each server contains two local XFS file systems called bricks. One brick from each RHS Server is combined with a corresponding brick on the other RHS Server to make a replicated volume. Therefore, the RHS Servers present two replicated volumes – one for the Image Service and one for Block Storage Service – composed of four bricks. Both volumes are synchronously replicated. If either RHS Server becomes unavailable, all data is still available via the remaining node.

Illustration 5.1.1: Red Hat Storage Cluster shows the storage architecture presented to the Red Hat Enterprise Linux OpenStack Platform servers.

Refer to **Appendix A: Red Hat Storage Setup** for complete instructions how to install the Red Hat Storage Servers and present the volumes.

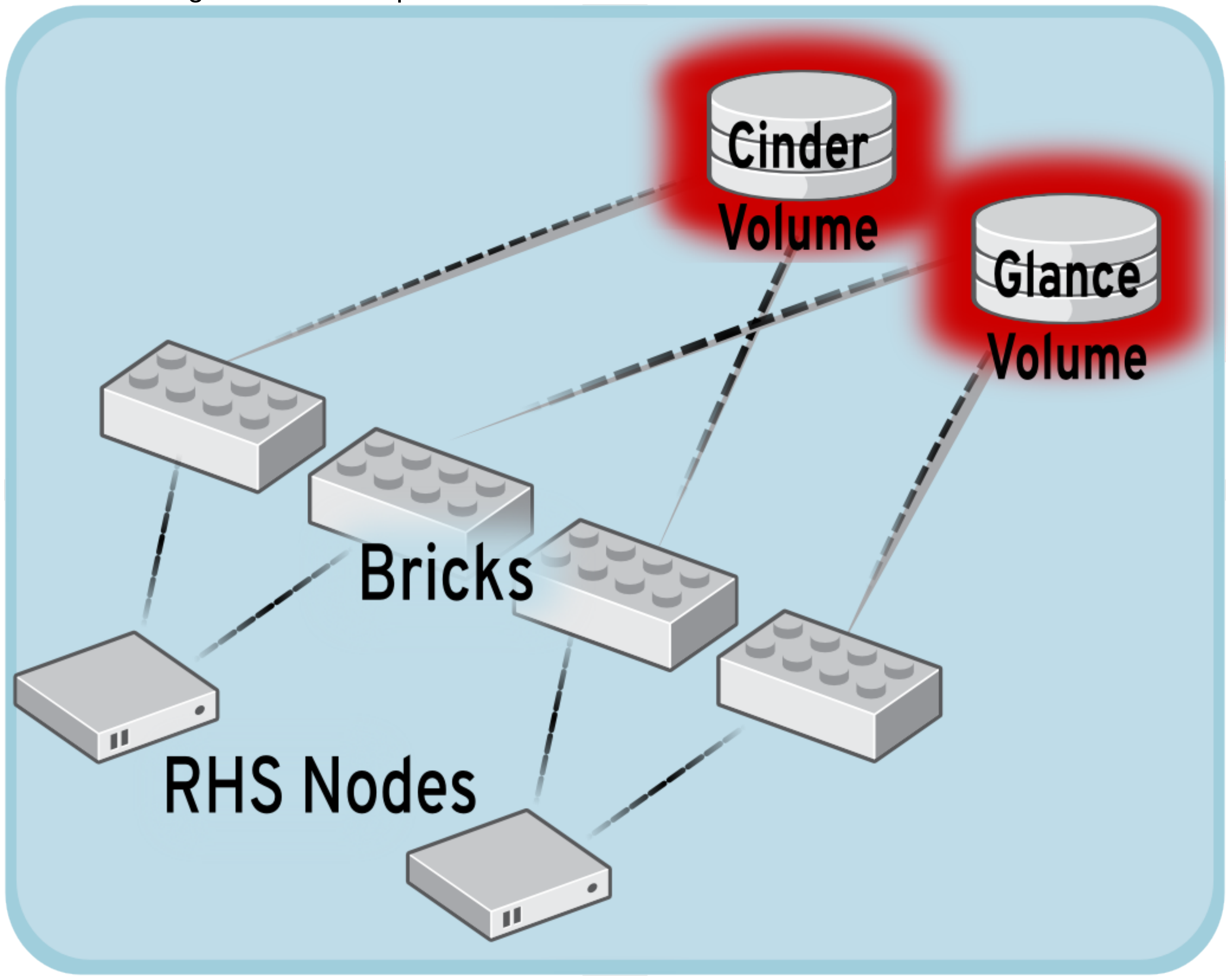


Illustration 5.1.1: Red Hat Storage Cluster

5.2 Configure the Red Hat Storage Clients

1. Add the RHN Red Hat Storage client channel. It is required for all compute nodes and the cloud controller.

```
[root@rhos0 ~]# i=0; while [ $i -lt 7 ]; do ssh rhos$i rhn-channel --user admin --password labpasswd --add --channel rhel-x86_64-server-rhsclient-6; ((i++)); done

[root@rhos0 ~]# i=0; while [ $i -lt 7 ]; do ssh rhos$i rhn-channel --list; ((i++)); done
rhel-x86_64-server-6
rhel-x86_64-server-6-ost-3
rhel-x86_64-server-rhsclient-6
rhel-x86_64-server-6
rhel-x86_64-server-6-ost-3
```

```
rhel-x86_64-server-rhsclient-6
rhel-x86_64-server-6
rhel-x86_64-server-6-ost-3
rhel-x86_64-server-rhsclient-6
rhel-x86_64-server-6
rhel-x86_64-server-6-ost-3
rhel-x86_64-server-rhsclient-6
rhel-x86_64-server-6
rhel-x86_64-server-6-ost-3
rhel-x86_64-server-rhsclient-6
rhel-x86_64-server-6
rhel-x86_64-server-6-ost-3
rhel-x86_64-server-rhsclient-6
rhel-x86_64-server-6
rhel-x86_64-server-6-ost-3
rhel-x86_64-server-rhsclient-6
```

1. Install the Red Hat Storage driver and **`tuned`** on the compute nodes and cloud controller.

```
[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i "yum install --assumeyes
--quiet glusterfs-fuse.x86_64 tuned"; done

[root@rhos0 ~]# yum install --quiet --assumeyes glusterfs-fuse.x86_64 tuned
```

2. Verify the packages installed successfully.

```
[root@rhos0 ~]# for i in 0 2 3 4 5; do ssh rhos$i "rpm --query --all | grep
--extended-regexp 'glusterfs-fuse|tuned'"; done
tuned-0.2.19-11.el6.1.noarch
glusterfs-fuse-3.3.0.11rhs-1.el6.x86_64
glusterfs-fuse-3.3.0.11rhs-1.el6.x86_64
tuned-0.2.19-11.el6.1.noarch
tuned-0.2.19-11.el6.1.noarch
glusterfs-fuse-3.3.0.11rhs-1.el6.x86_64
tuned-0.2.19-11.el6.1.noarch
glusterfs-fuse-3.3.0.11rhs-1.el6.x86_64
glusterfs-fuse-3.3.0.11rhs-1.el6.x86_64
tuned-0.2.19-11.el6.1.noarch
```

3. Create an Image Storage server mount point on all compute nodes.

```
[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i "mkdir –parents --verbose
/var/lib/glance; chown 161.161 /var/lib/glance"; done
mkdir: created directory `/var/lib/glance'
mkdir: created directory `/var/lib/glance'
mkdir: created directory `/var/lib/glance'
mkdir: created directory `/var/lib/glance'
```

4. Add the Image Storage server mount point to */etc/fstab* on the compute nodes so it is mounted automatically at boot.

```
[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i cp /etc/fstab{,.orig}; done
```

```
[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i 'echo "ra-rhs-srv1-10g:/RHOSglance /var/lib/glance glusterfs  _netdev,backupvolfile-server=ra-rhs-srv2-10g,selinux" 0 0 >> /etc/fstab'; done

[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i tail -1 /etc/fstab; done
ra-rhs-srv1-10g:/RHOSglance /var/lib/glance glusterfs  _netdev,backupvolfile-server=ra-rhs-srv2-10g,selinux 0 0
ra-rhs-srv1-10g:/RHOSglance /var/lib/glance glusterfs  _netdev,backupvolfile-server=ra-rhs-srv2-10g,selinux 0 0
ra-rhs-srv1-10g:/RHOSglance /var/lib/glance glusterfs  _netdev,backupvolfile-server=ra-rhs-srv2-10g,selinux 0 0
ra-rhs-srv1-10g:/RHOSglance /var/lib/glance glusterfs  _netdev,backupvolfile-server=ra-rhs-srv2-10g,selinux 0 0
```

5. Mount */var/lib/glance*.

```
[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i mount -a; done

[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i mount|grep glance; done
ra-rhs-srv1-10g:/RHOSglance on /var/lib/glance type fuse.glusterfs
(rw,default_permissions,allow_other,max_read=131072)
ra-rhs-srv1-10g:/RHOSglance on /var/lib/glance type fuse.glusterfs
(rw,default_permissions,allow_other,max_read=131072)
ra-rhs-srv1-10g:/RHOSglance on /var/lib/glance type fuse.glusterfs
(rw,default_permissions,allow_other,max_read=131072)
ra-rhs-srv1-10g:/RHOSglance on /var/lib/glance type fuse.glusterfs
(rw,default_permissions,allow_other,max_read=131072)
```

6. Apply the virtual host tuning on the compute nodes with **`tuned-adm`**.

```
[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i tuned-adm profile virtual-host; done
Reverting to saved sysctl settings: [  OK  ]
Calling '/etc/ktune.d/tunedadm.sh stop': [  OK  ]
Reverting to cfq elevator: dm-0 sda [  OK  ]
Stopping tuned: [  OK  ]
Switching to profile 'virtual-host'
Applying ktune sysctl settings:
/etc/ktune.d/tunedadm.conf: [  OK  ]
Calling '/etc/ktune.d/tunedadm.sh start': [  OK  ]
Applying sysctl settings from /etc/sysctl.conf
Applying deadline elevator: dm-0 sda [  OK  ]
Starting tuned: [  OK  ]
Reverting to saved sysctl settings: [  OK  ]
Calling '/etc/ktune.d/tunedadm.sh stop': [  OK  ]
Reverting to cfq elevator: dm-0 sda [  OK  ]
Stopping tuned: [  OK  ]
Switching to profile 'virtual-host'
Applying ktune sysctl settings:
/etc/ktune.d/tunedadm.conf: [  OK  ]
Calling '/etc/ktune.d/tunedadm.sh start': [  OK  ]
Applying sysctl settings from /etc/sysctl.conf
Applying deadline elevator: dm-0 sda [  OK  ]
```

```
Starting tuned: [  OK  ]
Reverting to saved sysctl settings: [  OK  ]
Calling '/etc/ktune.d/tunedadm.sh stop': [  OK  ]
Reverting to cfq elevator: dm-0 sda [  OK  ]
Stopping tuned: [  OK  ]
Switching to profile 'virtual-host'
Applying ktune sysctl settings:
/etc/ktune.d/tunedadm.conf: [  OK  ]
Calling '/etc/ktune.d/tunedadm.sh start': [  OK  ]
Applying sysctl settings from /etc/sysctl.conf
Applying deadline elevator: dm-0 sda [  OK  ]
Starting tuned: [  OK  ]
Reverting to saved sysctl settings: [  OK  ]
Calling '/etc/ktune.d/tunedadm.sh stop': [  OK  ]
Reverting to cfq elevator: dm-0 sda [  OK  ]
Stopping tuned: [  OK  ]
Switching to profile 'virtual-host'
Applying ktune sysctl settings:
/etc/ktune.d/tunedadm.conf: [  OK  ]
Calling '/etc/ktune.d/tunedadm.sh start': [  OK  ]
Applying sysctl settings from /etc/sysctl.conf
Applying deadline elevator: dm-0 sda [  OK  ]
Starting tuned: [  OK  ]

[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i tuned-adm active; done
Current active profile: virtual-host
Service tuned: enabled, running
Service ktune: enabled, running
Current active profile: virtual-host
Service tuned: enabled, running
Service ktune: enabled, running
Current active profile: virtual-host
Service tuned: enabled, running
Service ktune: enabled, running
Current active profile: virtual-host
Service tuned: enabled, running
Service ktune: enabled, running
```

5.3 Add RHS to Glance and Cinder

1. Configure the Image Storage server to use the Identity Service for authentication and allow direct URL access of images.

```
[root@rhos0 ~]# openstack-config --set /etc/glance/glance-api.conf DEFAULT
show_image_direct_url True

[root@rhos0 ~]# grep ^show_image_direct /etc/glance/glance-api.conf
show_image_direct_url = True

[root@rhos0 ~]# openstack-config --set /etc/nova/nova.conf DEFAULT
allowed_direct_url_schemes [file]

[root@rhos0 ~]# grep ^allowed /etc/nova/nova.conf
allowed_direct_url_schemes = [file]
```

2. Configure the Block Storage server to use the Red Hat Storage share. Run these commands on the cloud controller.

```
[root@rhos0 ~]# openstack-config --set /etc/cinder/cinder.conf DEFAULT
volume_driver cinder.volume.drivers.glusterfs.GlusterfsDriver
```

3. Configure Cinder to use the */etc/cinder/glusterfs.shares* file.

```
[root@rhos0 ~]# openstack-config --set /etc/cinder/cinder.conf DEFAULT
glusterfs_shares_config /etc/cinder/glusterfs.shares
```

4. Add the Red Hat Storage share to the file. The *backupvolfile* option adds the second server as a backup for the first to obtain volume information. Volume access is detailed in the volume information. Therefore access is not controlled by the server specified in the mount.

```
[root@rhos0 ~]# echo -e "ra-rhs-srv1-10g:/RHOScinder -o backupvolfile-
server=ra-rhs-srv2-10g,selinux" > /etc/cinder/glusterfs.shares

[root@rhos0 ~]# cat /etc/cinder/glusterfs.shares
ra-rhs-srv1-10g:/RHOScinder -o backupvolfile-server=ra-rhs-srv2-10g,selinux
```

5. Create the mount point.

```
[root@rhos0 ~]# mkdir --parents --verbose /var/lib/cinder/mnt
mkdir: created directory `/var/lib/cinder/mnt'

[root@rhos0 ~]# chown --verbose cinder.cinder /var/lib/cinder/mnt
changed ownership of `/var/lib/cinder/mnt' to cinder:cinder
```

6. Verify the changes.

```
[root@rhos0 ~]# cat /etc/cinder/glusterfs.shares
ra-rhs-srv1-10g:/RHOScinder -o backupvolfile-server=ra-rhs-srv2-10g,selinux

[root@rhos0 ~]# grep --ignore-case gluster /etc/cinder/cinder.conf | grep
--verbose \#
volume_driver = cinder.volume.drivers.glusterfs.GlusterfsDriver
glusterfs_shares_config = /etc/cinder/glusterfs.shares
```

7. Restart the Block Storage services on the cloud controller.

```
[root@rhos0 ~]# for i in $(chkconfig --list | awk ' /cinder/ { print $1 } ');
do service $i restart; done
Stopping openstack-cinder-api:                                 [  OK  ]
Starting openstack-cinder-api:                                 [  OK  ]
Stopping openstack-cinder-scheduler:                           [  OK  ]
Starting openstack-cinder-scheduler:                           [  OK  ]
Stopping openstack-cinder-volume:                              [  OK  ]
Starting openstack-cinder-volume:                              [  OK  ]
```

```
[root@rhos0 ~]# for i in $(chkconfig --list | awk ' /cinder/ { print $1 } ');
do service $i status; done
openstack-cinder-api (pid  9836) is running...
openstack-cinder-scheduler (pid  9851) is running...
openstack-cinder-volume (pid  9866) is running...
```

8. Remove the volume group created by **packstack**. It is no longer necessary.

```
[root@rhos0 ~]# vgremove cinder-volumes
Volume group "cinder-volumes" successfully removed

[root@rhos0 ~]# vgs
VG    #PV #LV #SN Attr   VSize   VFree
myvg    1   1   0 wz--n- 134.94g    0
```

9. Configure SELinux to allow virtual machines to use **fusefs**.

```
[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i "setsebool -P
virt_use_fusefs 1"; done

[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i "getsebool virt_use_fusefs
"; done
virt_use_fusefs --> on
virt_use_fusefs --> on
virt_use_fusefs --> on
virt_use_fusefs --> on
```

10. Restart the compute nodes in order to relabel the file system with SELinux changes.

```
root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i "touch ./autorelabel; shutdown
-r now"; done
```

6 Validate the Installation

The basic **`packstack`** deployment is complete. In this section the initial deployment is verified by configuring a demonstration environment that exercises the core functionality.

This section covers the following steps:

1. Create a demo tenant and user
2. Create a Neutron network for the tenant
3. Import a virtual machine image
4. Create an SSH key pair for authentication
5. Launch multiple virtual machines in the tenant
6. Verify network connectivity between the host and guests
7. Verify network connectivity between guests across compute nodes
8. Attach a persistent volume to a virtual machine

6.1 Create the Demo Tenant, User, and Role

A *TENANT* is a logical division of users and resources within OpenStack. Tenants enforce organizational boundaries. They allow multiple organizations or logical divisions to co-exist within the same cloud infrastructure. Resources created within a tenant are not available to users in another tenant.

Users and roles are defined on a per-tenant basis. Multiple users can be associated with the same role. The Identity service enforces tenant and role access for each user.

In this reference architecture the *demo-tenant* defines a hypothetical organization. A user named *demo* is granted user-level access to the tenant.

6.1.1 Import a Virtual Machine Disk Image

1. Connect to the client system via SSH and source the *keystonerc_admin* file.

```
[root@rhos0 ~]# ssh rhos1
Last login: Mon Jul 22 23:52:45 2013 from rhos0
Kickstarted on 2013-07-22

[root@rhos1 ~]# source /root/keystonerc_admin

[root@rhos1 ~(keystone_admin)]# env | grep OS_
OS_PASSWORD=ed7c0a23121e4983
OS_AUTH_URL=http://10.16.137.100:35357/v2.0/
OS_USERNAME=admin
OS_TENANT_NAME=admin
```

2. Import a virtual machine disk image. This Red Hat Enterprise Linux 6.4 image is used to deploy virtual machines. The image is public so it is available to all tenants.

```
[root@rhos1 ~(keystone_admin)]# glance image-create --name rhel-server2 --is-
public true --disk-format qcow2 --container-format bare --file
/pub/images/rhel-server-x86_64-kvm-6.4_20130130.0-4.qcow2

+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | e793566cf8aa170db033e37467334ecd     |
| container_format | bare                                 |
| created_at       | 2013-08-08T21:44:40                  |
| deleted          | False                                |
| deleted_at       | None                                 |
| disk_format      | qcow2                                |
| id               | 15f4d810-a30a-4be6-bdf0-b2968d96d6f2 |
| is_public        | True                                 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | rhel-server2                     |
| owner            | 3db07eecaa7f405c9ad35d4df6828bdb     |
| protected        | False                                |
| size             | 699592704                            |
| status           | active                               |
| updated_at       | 2013-08-08T21:44:49                  |
+------------------+--------------------------------------+
```

NOTE: The KVM image used in this example is available for download at *http://access.redhat.com*. From the Downloads tab, select the **Downloads** link under **Red Hat Enterprise**. Expand **Red Hat Enterprise Linux Server (v. 6 for 64-bit x86_64)** then select **Red Hat Common (for RHEL 6 Server x86_64)**. The image is named **KVM Guest image**. This customized image includes packages such as **cloud-init** that make it easier to interact with in a cloud environment.

3. List the image. The image status should be **active**.

```
[root@rhos1 ~(keystone_admin)]# glance image-list
+--------------------------------------+------------------+-------------+----+
| ID                                   | Name             | Disk Format |
Container Format | Size      | Status |
+--------------------------------------+------------------+-------------+----+
| 15f4d810-a30a-4be6-bdf0-b2968d96d6f2 | rhel-server2 | qcow2       | bare
| 699592704 | active |
+--------------------------------------+------------------+-------------+----+
```

6.1.2 Create a Tenant

1. Define a new tenant called *demo-tenant*.

```
[root@rhos1 ~(keystone_admin)]# keystone tenant-create --name demo-tenant
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |                                  |
|   enabled   |               True               |
|      id     | ba9579abd57b41879ce62af9785298f4 |
|     name    |           demo-tenant            |
+-------------+----------------------------------+
```

2. Verify the new tenant is seen by the Identity Service.

```
[root@rhos1 ~(keystone_admin)]# keystone tenant-list
+----------------------------------+-------------+---------+
|                id                |     name    | enabled |
+----------------------------------+-------------+---------+
| 988a0c24ff4f4d7ea2842ed642324f1b |    admin    |   True  |
| ba9579abd57b41879ce62af9785298f4 | demo-tenant |   True  |
| eac6d62ba1c6466bad44ce8df05efcc4 |   services  |   True  |
+----------------------------------+-------------+---------+
```

6.1.3 Add a User

1. Create a user named *demo*.

```
[root@rhos1 ~(keystone_admin)]# keystone user-create --name demo --pass demo
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |                                  |
| enabled  |               True               |
|    id    | 4534e4c52aee4a6b8178807c4ce869ec |
|   name   |               demo               |
| tenantId |                                  |
+----------+----------------------------------+
```

2. View the new user.

```
[root@rhos1 ~(keystone_admin)]# keystone user-list
+----------------------------------+---------+---------+-------------------+
|                id                |   name  | enabled |       email       |
+----------------------------------+---------+---------+-------------------+
| 238a9d18fbc84b08ba69876fcb3e110a |  admin  |   True  |   test@test.com   |
| f855e819b3e54879a52dc646a869dbf9 |  cinder |   True  |  cinder@localhost |
| 4534e4c52aee4a6b8178807c4ce869ec |   demo  |   True  |                   |
| c735baadecfb4391b114e9d79c5d5254 |  glance |   True  |  glance@localhost |
| 0736b5379d204ab497268fb85bd415f7 |   nova  |   True  |   nova@localhost  |
| 9f2591634d6b456f8e97b864c4a27f47 | quantum |   True  | quantum@localhost |
+----------------------------------+---------+---------+-------------------+
```

6.1.4 Associate the User, Role, and Tenant

1. Create a new user-role mapping within *demo-tenant*.

```
[root@rhos1 ~(keystone_admin)]# keystone user-role-add --user-id demo
--tenant-id demo-tenant --role-id Member
```

2. Verify the user-role mapping.

```
[root@rhos1 ~(keystone_admin)]# keystone user-role-list --user-id=demo
--tenant-id=demo-tenant
+----------------------------------+--------+----------------------------------+
|                id                |  name  |             user_id
|            tenant_id             |
+----------------------------------+----------------------------------+
| c2734d66fe5241bb937ad03d0d1b18c0 | Member |
4534e4c52aee4a6b8178807c4ce869ec | ba9579abd57b41879ce62af9785298f4 |
+----------------------------------+----------------------------------+
```

6.2 Configure the Network Server

Guest instances deployed on any of the compute nodes use *rhos6* as their network server. In this section a unique network is configured for guests instances in *demo-tenant*. Networks and instances created by the *demo* user are automatically added to *demo-tenant*.

6.2.1 Create the demo user keystonerc File

1. Create an Identity Service ID file for *demo* user on the client named *keystonerc_demo.*

```
export OS_USERNAME=demo
export OS_TENANT_NAME=demo-tenant
export OS_PASSWORD=demo
export OS_AUTH_URL=http://10.16.137.100:35357/v2.0/
export PS1='[\u@\h \W(demo_member)]\# '
```

2. Source *keystonerc_demo*. This ensures resources created by this user inherit the correct Keystone identifiers and are available only within this tenant unless explicitly made public by the user.

```
[root@rhos1 ~(keystone_admin)]# source /root/keystonerc_demo
```

3. Verify the environment variables are exported.

```
[root@rhos1 ~(demo_member)]# env | grep OS_
OS_PASSWORD=demo
OS_AUTH_URL=http://10.16.137.100:35357/v2.0/
OS_USERNAME=demo
OS_TENANT_NAME=demo-tenant
```

6.2.2 Create a Network

In OpenStack networking terminology, a *NETWORK* is an isolated L2 network segment similar to a VLAN. Networks are the basic building blocks of OpenStack networking.

1. Create a *demo-tenant* network. This network is available to instances booted in *demo-tenant*. This network carries all communication between instances in this tenant.

```
[root@rhos1 ~(keystone_member)]# quantum net-create net1
Created a new network:
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 75dbbf35-cae7-4f40-9b33-7e0e8b6a8358 |
| name                      | net1                                 |
| provider:network_type     | local                                |
| provider:physical_network |                                      |
| provider:segmentation_id  |                                      |
| router:external           | False                                |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | ba9579abd57b41879ce62af9785298f4     |
+---------------------------+--------------------------------------+
```

2. Display **net1**. The tenant ID should be set to *demo-tenant*.

```
[root@rhos1 ~(keystone_demo)]# quantum net-show net1
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 75dbbf35-cae7-4f40-9b33-7e0e8b6a8358 |
| name                      | net1                                 |
| provider:network_type     | local                                |
| provider:physical_network |                                      |
| provider:segmentation_id  |                                      |
| router:external           | False                                |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   | 4e721701-e97d-4ab5-94eb-5e7900185524 |
| tenant_id                 | ba9579abd57b41879ce62af9785298f4     |
+---------------------------+--------------------------------------+
```

6.2.3 Create a Subnet

In OpenStack networking terminology, a *SUBNET* associates a block of IP addresses and other network configuration details such as default gateway and DNS servers with an OpenStack network. Each network can contain multiple subnets.

1. Create a subnet named private in **net1** with an IPV4 address range of 10.0.5.0/24. This subnet carries layer-2 and broadcast traffic on the **net1** network.

```
[root@rhos1 ~(keystone_demo)]# quantum subnet-create  --name private net1
10.0.5.0/24
Created a new subnet:
+------------------+--------------------------------------------+
| Field            | Value                                      |
+------------------+--------------------------------------------+
| allocation_pools | {"start": "10.0.5.2", "end": "10.0.5.254"} |
| cidr             | 10.0.5.0/24                                |
| dns_nameservers  |                                            |
| enable_dhcp      | True                                       |
| gateway_ip       | 10.0.5.1                                   |
| host_routes      |                                            |
| id               | 4e721701-e97d-4ab5-94eb-5e7900185524       |
| ip_version       | 4                                          |
| name             | private                                    |
| network_id       | 75dbbf35-cae7-4f40-9b33-7e0e8b6a8358       |
| tenant_id        | ba9579abd57b41879ce62af9785298f4           |
+------------------+--------------------------------------------+
```

2. List the installed networks to verify the association between the **net1** network and **private** subnet.

```
[root@rhos1 ~(keystone_demo)]# quantum net-list
+--------------------------------------+------+-----------------------------+
| id                                   | name | subnets
+--------------------------------------+------+-----------------------------+
| 75dbbf35-cae7-4f40-9b33-7e0e8b6a8358 | net1 | 4e721701-e97d-4ab5-94eb-
5e7900185524 10.0.5.0/24 |
+--------------------------------------+------+-----------------------------+
```

3. View the subnet to verify the starting address and allocation range.

```
[root@rhos1 ~(keystone_demo)]# quantum subnet-list
+--------------------------------------+---------+------------+-----------+
| id                                   | name    | cidr        |
allocation_pools                            |
+--------------------------------------+---------+------------+-----------+
| 4e721701-e97d-4ab5-94eb-5e7900185524 | private | 10.0.5.0/24 | {"start":
"10.0.5.2", "end": "10.0.5.254"} |
+--------------------------------------+---------+------------+-----------+
```

4. Capture the Identity Service id for **net1** for use in subsequent steps.

```
[root@rhos1 ~(keystone_demo)]# demo_net=$(quantum net-list | awk ' /net1/
{ print $2 } ')

[root@rhos1 ~(keystone_demo)]# echo -e "$demo_net"
75dbbf35-cae7-4f40-9b33-7e0e8b6a8358
```

6.3 Boot an Instance in the Tenant

This section describes the steps to configure a network and launch an instance in the *demo* tenant.

6.3.1 Create a Key Pair and Boot Instances

1. Create a key pair as *demo* user. This key pair is used to authenticate SSH sessions to virtual machine instances.

```
[root@rhos1 ~(demo_member)]# nova keypair-add demokp > /root/demokp.pem

[root@rhos1 ~(demo_member)]# chmod 600 /root/demokp.pem

[root@rhos1 ~(demo_member)]# ll /root/demokp.pem
-rw-------. 1 root root 1676 Jul 23 01:05 /root/demokp.pem
```

2. Boot four instances as *demo* specifying the new key.

```
[root@rhos1 ~(demo_member)]# nova boot --flavor 2 --image rhel-server2 --key-
name demokp inst1
```

```
+------------------------------+--------------------------------------+
| Property                     | Value                                |
+------------------------------+--------------------------------------+
| status                       | BUILD                                |
| updated                      | 2013-07-23T06:08:08Z                 |
| OS-EXT-STS:task_state        | None                                 |
| key_name                     | demokp                               |
| image                        | rhel-server2                         |
| hostId                       |                                      |
| OS-EXT-STS:vm_state          | building                             |
| flavor                       | m1.small                             |
| id                           | 4103a0d6-d2ad-4c1b-b905-44bb61bd1286 |
| security_groups              | [{u'name': u'default'}]              |
| user_id                      | 4534e4c52aee4a6b8178807c4ce869ec     |
| name                         | inst1                                |
| adminPass                    | cTcxMRPYVGQ9                         |
| tenant_id                    | ba9579abd57b41879ce62af9785298f4     |
| created                      | 2013-07-23T06:08:07Z                 |
| OS-DCF:diskConfig            | MANUAL                               |
| metadata                     | {}                                   |
| accessIPv4                   |                                      |
| accessIPv6                   |                                      |
| progress                     | 0                                    |
| OS-EXT-STS:power_state       | 0                                    |
| OS-EXT-AZ:availability_zone  | nova                                 |
| config_drive                 |                                      |
+------------------------------+--------------------------------------+

[root@rhos1 ~(demo_member)]# nova boot --flavor 2 --image rhel-server2 --key-
name demokp inst2
+------------------------------+--------------------------------------+
| Property                     | Value                                |
+------------------------------+--------------------------------------+
| status                       | BUILD                                |
| updated                      | 2013-07-23T06:08:23Z                 |
| OS-EXT-STS:task_state        | None                                 |
| key_name                     | demokp                               |
| image                        | rhel-server2                         |
| hostId                       |                                      |
| OS-EXT-STS:vm_state          | building                             |
| flavor                       | m1.small                             |
| id                           | c5be56d5-8411-46e2-a9cc-b3688c2ecfaf |
| security_groups              | [{u'name': u'default'}]              |
| user_id                      | 4534e4c52aee4a6b8178807c4ce869ec     |
| name                         | inst2                                |
| adminPass                    | fWYHK9zCqj58                         |
| tenant_id                    | ba9579abd57b41879ce62af9785298f4     |
| created                      | 2013-07-23T06:08:22Z                 |
| OS-DCF:diskConfig            | MANUAL                               |
| metadata                     | {}                                   |
| accessIPv4                   |                                      |
| accessIPv6                   |                                      |
| progress                     | 0                                    |
| OS-EXT-STS:power_state       | 0                                    |
| OS-EXT-AZ:availability_zone  | nova                                 |
| config_drive                 |                                      |
+------------------------------+--------------------------------------+

[root@rhos1 ~(demo_member)]# nova boot --flavor 2 --image rhel-server2 --key-
name demokp inst3
+---------------+------------------------------------------------------+
| Property                     | Value
+---------------+------------------------------------------------------+
| status                       | BUILD
| updated                      | 2013-07-23T06:08:34Z
| OS-EXT-STS:task_state        | None
| key_name                     | demokp
| image                        | rhel-server2
| hostId                       | 5e4ca5aee70805f6b2b2eac41400bc917095269714d9 |
| OS-EXT-STS:vm_state          | building
| flavor                       | m1.small
| id                           | 9236f8fe-20af-4145-98ff-cbf8b738b366
| security_groups              | [{u'name': u'default'}]
| user_id                      | 4534e4c52aee4a6b8178807c4ce869ec
| name                         | inst3
| adminPass                    | sMMJwC3e6aER
| tenant_id                    | ba9579abd57b41879ce62af9785298f4
| created                      | 2013-07-23T06:08:33Z
| OS-DCF:diskConfig            | MANUAL
| metadata                     | {}
| accessIPv4                   |
| accessIPv6                   |
| progress                     | 0
| OS-EXT-STS:power_state       | 0
| OS-EXT-AZ:availability_zone  | nova
| config_drive                 |
+------------------------------+----------------------------------------------+

[root@rhos1 ~(demo_member)]# nova boot --flavor 2 --image rhel-server2 --key-
name demokp inst4
+------------------------------+--------------------------------------+
| Property                     | Value                                |
+------------------------------+--------------------------------------+
| status                       | BUILD                                |
| updated                      | 2013-07-23T06:08:53Z                 |
| OS-EXT-STS:task_state        | None                                 |
| key_name                     | demokp                               |
| image                        | rhel-server2                         |
| hostId                       |                                      |
| OS-EXT-STS:vm_state          | building                             |
| flavor                       | m1.small                             |
| id                           | c19d9e9c-a5a0-458b-9a3d-c87ab2a1857d |
| security_groups              | [{u'name': u'default'}]              |
| user_id                      | 4534e4c52aee4a6b8178807c4ce869ec     |
| name                         | inst4                                |
| adminPass                    | Qnaof36iSe2G                         |
| tenant_id                    | ba9579abd57b41879ce62af9785298f4     |
| created                      | 2013-07-23T06:08:52Z                 |
| OS-DCF:diskConfig            | MANUAL                               |
| metadata                     | {}                                   |
| accessIPv4                           |                                      |
| accessIPv6                           |                                      |
| progress                             | 0                                    |
| OS-EXT-STS:power_state               | 0                                    |
| OS-EXT-AZ:availability_zone          | nova                                 |
| config_drive                         |                                      |
+------------------------------+---------------------------------------+
```

3. Verify the new instances are in an **ACTIVE** state. They may stay in **BUILD** state for several minutes.

```
[root@rhos1 ~(demo_member)]# nova list
+--------------------------------------+-------+--------+---------------+
| ID                                   | Name  | Status | Networks      |
+--------------------------------------+-------+--------+---------------+
| 4103a0d6-d2ad-4c1b-b905-44bb61bd1286 | inst1 | ACTIVE | net1=10.0.5.2 |
| c5be56d5-8411-46e2-a9cc-b3688c2ecfaf | inst2 | ACTIVE | net1=10.0.5.4 |
| 9236f8fe-20af-4145-98ff-cbf8b738b366 | inst3 | ACTIVE | net1=10.0.5.5 |
| c19d9e9c-a5a0-458b-9a3d-c87ab2a1857d | inst4 | ACTIVE | net1=10.0.5.6 |
+--------------------------------------+-------+--------+---------------+
```

6.3.2 Create Default Security Group Rules

So far the following steps have been performed:

1. Create a demo tenant and user
2. Create a Neutron network for the tenant
3. Import a virtual machine image
4. Create an SSH key pair for authentication
5. Launch multiple virtual machines in the tenant

The remaining steps in this section include setting up a security group, verifying the network configuration, and attaching a persistent volume to an instance.

A security group defines the network traffic allowed into and out of a network. Each tenant has a default security group which defines allowed traffic for all instances booted in that tenant. It serves as a second firewall beyond the instance's own firewall if any. In Red Hat Enterprise Linux OpenStack Platform 3 the default security groups are very restrictive. In this section the default security group for *demo-tenant* is modified to allow ICMP, HTTP, and SSH traffic.

1. Find the Identity Service id of the **default** security group.

```
[root@rhos1 ~(demo_member)]# default_id=$(quantum security-group-list | awk '
/default/ { print $2 } ')

[root@rhos1 ~(demo_member)]# echo -e "$default_id"
 2a7ddc8a-90c0-4ba0-bbc1-080273448f24
```

2. Create security group rules allowing SSH, ICMP, and HTTP traffic to instances in this security group.

```
[root@rhos1 ~(demo_member)]# quantum security-group-rule-create --direction
```

```
ingress --protocol icmp  $default_id

Created a new security_group_rule:
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| direction         | ingress                              |
| ethertype         | IPv4                                 |
| id                | f9a62d52-35e6-480e-a243-04998112cf13 |
| port_range_max    |                                      |
| port_range_min    |                                      |
| protocol          | icmp                                 |
| remote_group_id   |                                      |
| remote_ip_prefix  |                                      |
| security_group_id | 2a7ddc8a-90c0-4ba0-bbc1-080273448f24 |
| tenant_id         | ba9579abd57b41879ce62af9785298f4     |
+-------------------+--------------------------------------+

[root@rhos1 ~(demo_member)]# quantum security-group-rule-create --direction
ingress --protocol tcp --port_range_min 22 --port_range_max 22 $default_id

Created a new security_group_rule:
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| direction         | ingress                              |
| ethertype         | IPv4                                 |
| id                | 0ecf41a5-a014-45fa-ae53-13132afc5792 |
| port_range_max    | 22                                   |
| port_range_min    | 22                                   |
| protocol          | tcp                                  |
| remote_group_id   |                                      |
| remote_ip_prefix  |                                      |
| security_group_id | 2a7ddc8a-90c0-4ba0-bbc1-080273448f24 |
| tenant_id         | ba9579abd57b41879ce62af9785298f4     |
+-------------------+--------------------------------------+

[root@rhos1 ~(demo_member)]# quantum security-group-rule-create --direction
ingress --protocol tcp --port_range_min 80 --port_range_max 80 $default_id

Created a new security_group_rule:
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| direction         | ingress                              |
| ethertype         | IPv4                                 |
| id                | 53d74064-41ba-4832-9f40-8384155cc4f3 |
| port_range_max    | 80                                   |
| port_range_min    | 80                                   |
| protocol          | tcp                                  |
| remote_group_id   |                                      |
| remote_ip_prefix  |                                      |
| security_group_id | 2a7ddc8a-90c0-4ba0-bbc1-080273448f24 |
| tenant_id         | ba9579abd57b41879ce62af9785298f4     |
+-------------------+--------------------------------------+
```

3. View the default security group rules for *demo-tenant*.

```
[root@rhos1 ~(demo_member)]# quantum security-group-show $default_id
+----------------------+--------------------------------------------------------+
| Field                | Value
+--------------------------------------------------------------------------------+
| description          | default
| id                   | 2a7ddc8a-90c0-4ba0-bbc1-080273448f24
| name                 | default
| security_group_rules | {"remote_group_id": null, "direction": "ingress",
"remote_ip_prefix": null, "protocol": "tcp", "tenant_id":
"ba9579abd57b41879ce62af9785298f4", "port_range_max": 22,
"security_group_id": "2a7ddc8a-90c0-4ba0-bbc1-080273448f24",
"port_range_min": 22, "ethertype": "IPv4", "id": "0ecf41a5-a014-45fa-ae53-
13132afc5792"}                                              |
|                      | {"remote_group_id": null, "direction": "ingress",
"remote_ip_prefix": null, "protocol": "tcp", "tenant_id":
"ba9579abd57b41879ce62af9785298f4", "port_range_max": 80,
"security_group_id": "2a7ddc8a-90c0-4ba0-bbc1-080273448f24",
"port_range_min": 80, "ethertype": "IPv4", "id": "53d74064-41ba-4832-9f40-
8384155cc4f3"}                                              |
|                      | {"remote_group_id": "2a7ddc8a-90c0-4ba0-bbc1-
080273448f24", "direction": "ingress", "remote_ip_prefix": null, "protocol":
null, "tenant_id": "ba9579abd57b41879ce62af9785298f4", "port_range_max":
null, "security_group_id": "2a7ddc8a-90c0-4ba0-bbc1-080273448f24",
"port_range_min": null, "ethertype": "IPv6", "id": "6b233ae7-7d2b-4fff-a551-
7208983ea643"} |
|                      | {"remote_group_id": "2a7ddc8a-90c0-4ba0-bbc1-
080273448f24", "direction": "ingress", "remote_ip_prefix": null, "protocol":
null, "tenant_id": "ba9579abd57b41879ce62af9785298f4", "port_range_max":
null, "security_group_id": "2a7ddc8a-90c0-4ba0-bbc1-080273448f24",
"port_range_min": null, "ethertype": "IPv4", "id": "7ccd7dea-541c-4215-910d-
49579b82c2b5"} |
|                      | {"remote_group_id": null, "direction": "egress",
"remote_ip_prefix": null, "protocol": null, "tenant_id":
"ba9579abd57b41879ce62af9785298f4", "port_range_max": null,
"security_group_id": "2a7ddc8a-90c0-4ba0-bbc1-080273448f24",
"port_range_min": null, "ethertype": "IPv4", "id": "7f0b4038-b8d9-4a91-a9a6-
f92f884bcb12"}                                            |
|                      | {"remote_group_id": null, "direction": "egress",
"remote_ip_prefix": null, "protocol": null, "tenant_id":
"ba9579abd57b41879ce62af9785298f4", "port_range_max": null,
"security_group_id": "2a7ddc8a-90c0-4ba0-bbc1-080273448f24",
"port_range_min": null, "ethertype": "IPv6", "id": "e2ee5ba5-c7f4-4d24-a986-
3c67fa60d72f"}                                            |
|                      | {"remote_group_id": null, "direction": "ingress",
"remote_ip_prefix": null, "protocol": "icmp", "tenant_id":
"ba9579abd57b41879ce62af9785298f4", "port_range_max": null,
"security_group_id": "2a7ddc8a-90c0-4ba0-bbc1-080273448f24",
"port_range_min": null, "ethertype": "IPv4", "id": "f9a62d52-35e6-480e-a243-
04998112cf13"}                                         |
| tenant_id            | ba9579abd57b41879ce62af9785298f4
+----------------------+--------------------------------------------------------+
```

6.4 Configure a Layer 3 Network in the Environment

This section describes the steps necessary to create a router to pass L3 traffic between OpenStack and physical networks. OpenStack Networking routers can connect multiple L2 networks, and can also provide a gateway that connects private L2 networks to a shared external network. The L3 router provides basic NAT capabilities on gateway ports that uplink the router to external networks.

This section of the reference architecture has three goals. First, connect to the private subnet directly from a Compute node via the tenant's network namespace. Second, create an external network in Neutron that maps to the lab physical network. Third, connect the software defined private subnet with the new external public network via the software defined router.

6.4.1 Create the Router

1. Create a router named **route1**.

```
[root@rhos1 ~(keystone_demo)]# quantum router-create  route1
Created a new router:
+-----------------------+--------------------------------------+
| Field                 | Value                                |
+-----------------------+--------------------------------------+
| admin_state_up        | True                                 |
| external_gateway_info |                                      |
| id                    | b334a0b9-3001-4cea-a00e-053f3f6aaede |
| name                  | route1                               |
| status                | ACTIVE                               |
| tenant_id             | ba9579abd57b41879ce62af9785298f4    |
+-----------------------+--------------------------------------+
```

2. Find the **private** subnet's Identity Service ID.

```
root@rhos1 ~(keystone_demo)]# subnet_id=$(quantum subnet-list | awk ' /
10.0.5.0/ { print $2 } ')

[root@rhos1 ~(keystone_demo)]# echo -e "$subnet_id"
4e721701-e97d-4ab5-94eb-5e7900185524
```

3. Attach the **private** subnet to the router.

```
[root@rhos1 ~(keystone_demo)]# quantum router-interface-add route1 $subnet_id
Added interface to router route1
```

6.4.2 Ping the Router's External Interface

1. Set an environment variable to the Identity Service ID of the router.

```
[root@rhos1 ~(keystone_demo)]# route_id=$(quantum router-list | awk '
/route1/ { print $2 } ')

[root@rhos1 ~(keystone_demo)]# echo -e "$route_id"
b334a0b9-3001-4cea-a00e-053f3f6aaede
```

2. Set an environment variable to the IP address of the router's external interface.

```
[root@rhos1 ~(keystone_demo)]# quantum port-list --device-id $route_id
+--------------------------------+------+-------------------+---------------+
| id                             | name | mac_address       | fixed_ips
--------------------------------+------+-------------------+---------------+
| 1a598720-6fa3-41ae-ad80-40355700b718 |      | fa:16:3e:58:ce:56 |
{"subnet_id": "4e721701-e97d-4ab5-94eb-5e7900185524", "ip_address":
"10.0.5.1"} |
--------------------------------+------+-------------------+---------------+

[root@rhos1 ~(keystone_demo)]# router_ip=$(quantum subnet-show $subnet_id |
awk ' /gateway_ip/ { print $4 } ')

[root@rhos1 ~(keystone_demo)]# echo -e "$router_ip"
10.0.5.1
```

3. Find the network name space of the router on the network node. In this reference architecture, **rhos6** is the network node.

```
[root@rhos1 ~(keystone_demo)]# qroute_id=$(ssh rhos6 ip netns list | grep
qrouter)
root@rhos6's password: ******

[root@rhos1 ~(keystone_demo)]# echo $qroute_id
qrouter-b334a0b9-3001-4cea-a00e-053f3f6aaede
```

4. Ping the router via its external interface within the network name space on the network node. This demonstrates network connectivity between the server and the router.

```
[root@rhos1 ~(keystone_demo)]# ssh rhos6 ip netns exec $qroute_id ping -c 2
$router_ip
root@rhos6's password: ********
PING 10.0.5.1 (10.0.5.1) 56(84) bytes of data.
64 bytes from 10.0.5.1: icmp_seq=1 ttl=64 time=0.059 ms
64 bytes from 10.0.5.1: icmp_seq=2 ttl=64 time=0.047 ms

--- 10.0.5.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.040/0.048/0.059/0.011 ms
```

6.4.3 Create an External Network

An *external network* is a Neutron construct that maps to a physical network in the data center. In this section the lab network is added to the router as an external network so packets can be routed between the *public* and *private* subnets.

Create an external network named *public* that maps to the lab network.

1. Source *keystonerc_admin*.

   Create the external network outside of *demo-tenant* so instances launched in this tenant are not automatically attached to this network. Instead they are attached selectively by floating IP addresses.

```
[root@rhos1 ~(keystone_demo)]# source /root/keystonerc_admin

[root@rhos1 ~(keystone_admin)]# env | grep OS_
OS_PASSWORD=ed7c0a23121e4983
OS_AUTH_URL=http://10.16.137.100:35357/v2.0/
OS_USERNAME=admin
OS_TENANT_NAME=admin
```

2. Create an external network in the **service** tenant named **ext_net**. This is a provider network that maps to the physical lab network.

   A provider network maps directly to a physical network in the data center. They are used to give tenants direct access to public networks.

   The *network_type flat* option species that all devices in this network are directly connected to one another via a switch without the need for intermediate devices such as routers.

   These commands must be run as the Keystone admin.

```
[root@rhos1 ~(keystone_admin)]# service_id=$(keystone tenant-list | awk '
/service/ { print $2 } ')

[root@rhos1 ~(keystone_admin)]# quantum net-create --tenant-id $service_id
ext_net --router:external=True --provider:network_type flat
--provider:physical_network physnet1
Created a new network:
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 8f773c4e-54d4-4d2a-9ec6-990787ffee89 |
| name                      | ext_net                              |
| provider:network_type     | flat                                 |
| provider:physical_network | physnet1                             |
| provider:segmentation_id  |                                      |
| router:external           | True                                 |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | 0f4a450c76e34aae9163646f41999e59     |
+---------------------------+--------------------------------------+
```

3. Create a subnet on **ext_net** named **public** that includes a range of floating IP addresses and a default gateway. The default gateway should be the actual default gateway for the external network.

   Instances attached to this network use the gateway address as their default gateway.

   The *enable_dhcp* option is set to *False*. This ensures that instances booted in this network are not assigned an IP address via DHCP. Instances in this network are assigned floating IP addresses from the allocation pool in order to communicate with physical systems in the lab and reach the gateway.

```
[root@rhos1 ~(keystone_admin)]# quantum subnet-create --tenant-id $service_id
--name public --gateway 10.16.143.254 --allocation-pool
start=10.16.137.112,end=10.16.137.114 ext_net 10.16.136.0/21
--enable_dhcp=False
Created a new subnet:
+------------------+----------------------------------------------------+
| Field            | Value                                              |
+------------------+----------------------------------------------------+
| allocation_pools | {"start": "10.16.137.112", "end": "10.16.137.114"} |
| cidr             | 10.16.136.0/21                                     |
| dns_nameservers  |                                                    |
| enable_dhcp      | False                                              |
| gateway_ip       | 10.16.143.254                                      |
| host_routes      |                                                    |
| id               | b057e0bc-7b80-4687-b0cb-db3b6fdaea19               |
| ip_version       | 4                                                  |
| name             | public                                             |
| network_id       | 8f773c4e-54d4-4d2a-9ec6-990787ffee89               |
| tenant_id        | 0f4a450c76e34aae9163646f41999e59                   |
+------------------+----------------------------------------------------+
```

4. The previous command assigns the router with an address on the *public* subnet. Set the router's default gateway to the external network. These commands allow the router to act as a NAT gateway for external connectivity.

```
[root@rhos1 ~(keystone_admin)]# route_id=$(quantum router-list | awk '
/route1/ { print $2 }')

[root@rhos1 ~(keystone_admin)]# ext_net_id=$(quantum net-list | awk '
/ext_net/ { print $2 } ')

[root@rhos1 ~(keystone_admin)]# quantum router-gateway-set $route_id
$ext_net_id
Set gateway for router c36a1917-9e77-4901-99ed-181f59813dee
```

6.4.4 Verify Routing

In section **6.4.2 Ping the Router's External Interface**, connectivity to the router was verified from the network server. The network server owns the network namespace for the router. Connect to an instance on the private subnet from a physical server on the lab network by its private IP address to verify routing.

1. From the client server, the gateway IP address on the Neutron router. 113 is the floating IP address. The router interface is 112.

```
[root@rhos1 ~]# ssh rhos6 'ip netns exec $(ip netns | grep qrouter) ip a |
grep 10.16.137'
root@rhos6's password: ******
inet 10.16.137.112/21 brd 10.16.143.255 scope global qg-1c07308a-2d
inet 10.16.137.113/32 brd 10.16.137.113 scope global qg-1c07308a-2d
```

2. Add a route to the **`private`** network via the router's interface on the **`public`** network.

```
[root@rhos1 ~]# route add -net 10.0.5.0 netmask 255.255.255.0 gateway
10.16.137.112
```

3. SSH directly to the instance from the client server.

```
[root@rhos1 ~]# ssh -i demokp.pem 10.0.5.2 uptime
Warning: Permanently added '10.0.5.2' (RSA) to the list of known hosts.
04:37:16 up  1:33,  0 users,  load average: 0.00, 0.00, 0.00
```

6.4.5 Validated Installation

Illustration 6.4.5.1 Validated Installation depicts the validated installation including the software defined networks and virtual machines. The software defined router connects the Private network in the Demo tenant to the external Public network through the bridge.

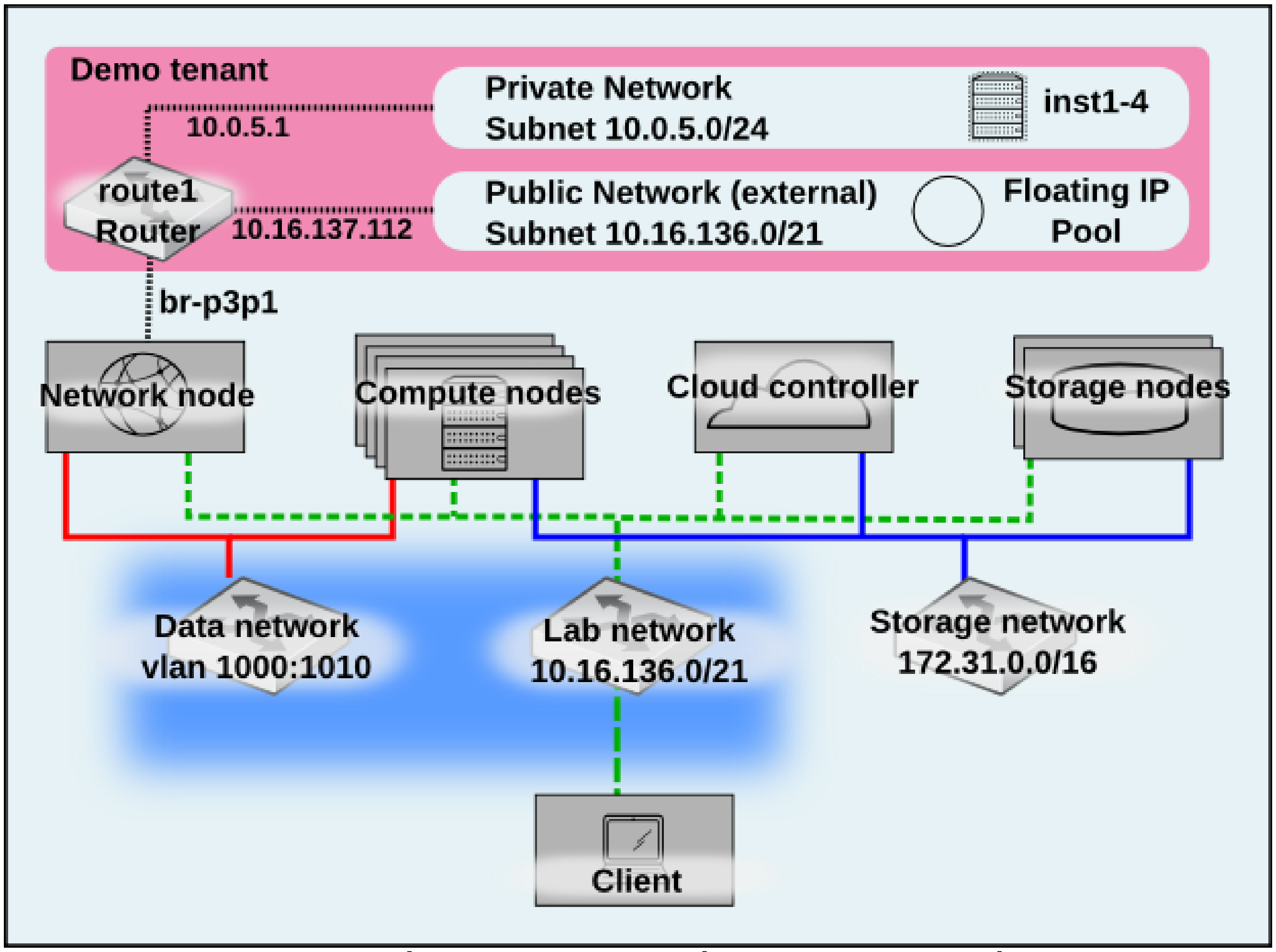


Illustration 6.4.5.1: Validated Installation

NOTE: The *p3p1* interface on the Network node is attached to the same physical network as the *em1* interface. Tenant traffic traverses the VLANs configured on the switch. External traffic also traverses the *p3p1* interface but is not tagged.

7 Deploy a Multi-tier Web Application

7.1 Overview

This section of the reference architecture describes how to install a multi-tier web application consisting of a web server and a database server. The example uses a simple WordPress deployment. The web server is accessible via a public floating IP address on the client network. It connects to a remote MySQL database server via the tenant network. The database server stores its database on a persistent volume provided by the Block Storage service.

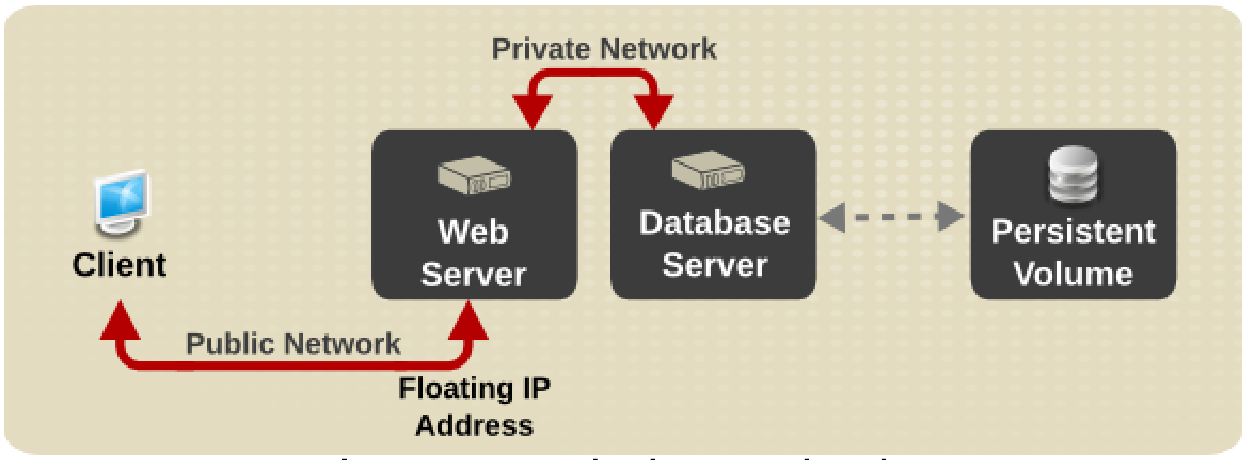


Figure 1: Multi-tier Application

The process for deploying a multi-tier application demonstrates OpenStack's complete functionality as described in this reference architecture. It is a realistic use case that touches every service discussed in the preceding pages. Multi-tier web applications are often deployed in IaaS clouds so they can be scaled as required. Keeping the database server off a publicly accessible network provides an extra layer of security. **Figure 1: Multi-tier Application** depicts the end state of the multi-tier application deployment.

> **NOTE:** The OpenStack Orchestration service ("Heat") also provides this functionality. However, it is tech preview in RHOS 3.0 and not covered in this reference architecture.

Appendix C: Full Network Diagram presents a complete view of the deployment including host names and IP addresses.

7.2 Deploy the Database Server

This section describes the steps for installing the database server as a virtual machine instance in OpenStack. There are four steps:

- Create a user data script. This script performs post-boot customizations to the instance such as installing software packages.
- Create a registration script. This script registers the instance with RHN.
- Boot the instance. The scripts are executed during boot. The database is installed on a persistent volume passed as a parameter to the instance at boot.
- Verify the installation.

7.2.1 Add a DNS Server to the private Subnet

Add a DNS server to the **private** subnet. The DNS server address is pushed to the */etc/resolv.conf* file on instances booted on this subnet. The instance customization scripts require DNS name resolution.

1. Source the environment for the *demo* user on the client system.

```
[root@rhos1 ~]# source /root/keystonerc_demo
```

2. Save the **private** subnet ID to an environment variable.

```
[root@rhos1 ~(demo_member)]# subnet_id=$(quantum subnet-list | awk '
/private/ { print $2 } ')

[root@rhos1 ~(demo_member)]# echo $subnet_id
970c21ae-87f1-4b8d-bb96-38bba1ffebf4
```

3. Update the **private** subnet to add a DNS server.

```
[root@rhos1 ~(demo_member)]# quantum subnet-update $subnet_id
--dns_nameservers list=true 10.16.138.14
```

4. Display the DNS server for the **private** subnet.

```
[root@rhos1 ~(demo_member)]# quantum subnet-show $subnet_id | grep dns
| dns_nameservers  | 10.16.138.14                               |
```

7.2.2 Create the User Data Script

The user data script employed in this section can be downloaded by registered Red Hat customers here: **https://access.redhat.com/site/node/455603/40/1**. The script is named *wp-backend.sh*. It installs and configures the MySQL server software. The MySQL database is installed to a persistent volume if one is available.

7.2.3 Create the Registration Script

The registration script employed in this reference architecture is named *sysreg.sh*. It registers the instance with RHN channels required to deploy the database server. This script is also included in the script download.

7.2.4 Create the Database Volume

1. Create a 5 GB persistent volume to store the MySQL database.

```
[root@rhos1 ~(demo_member)]# cinder create --display-name mysql 5
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|     attachments     |                  []                  |
|  availability_zone  |                 nova                 |
|       bootable      |                false                 |
|      created_at     |      2013-07-25T18:09:58.927522      |
| display_description |                 None                 |
|     display_name    |                mysql                 |
|          id         | 9df83cd2-7400-4e1a-b1d4-d4d27625df38 |
|       metadata      |                  {}                  |
|         size        |                  5                   |
|     snapshot_id     |                 None                 |
|     source_volid    |                 None                 |
|        status       |               creating               |
|     volume_type     |                 None                 |
+---------------------+--------------------------------------+
```

2. Display the new volume.

```
[root@rhos1 ~(demo_member)]# nova volume-list
+--------------------------------------+--------+--------------+-----+-----+
| ID                                   | Status | Display Name | Size |
Volume Type | Attached to                          |
+--------------------------------------+--------+--------------+-----+-----+
| 9df83cd2-7400-4e1a-b1d4-d4d27625df38 | Available | mysql        | 5    |
None        | None |
+-------------------------------------+--------+--------------+-----+------+
```

3. Save the volume id of the **mysql** volume to an environment variable.

```
[root@rhos1 ~(demo_member)]# volid=$(nova volume-list | awk ' /mysql/ { print
$2 }')

[root@rhos1 ~(demo_member)]# echo $volid
9df83cd2-7400-4e1a-b1d4-d4d27625df38
```

4. Save the **private** network id to an environment variable.

```
[root@rhos1 ~(demo_member)]# netid=$(quantum net-list | awk ' /10.0.5.0/
{ print $2 } ')

[root@rhos1 ~(demo_member)]# echo $netid
9a12ca89-d5f3-4490-8b29-f08fb7a96386
```

7.2.5 Boot the Database Instance

1. Boot a new instance specifying the network and new volume. Also point to the post-boot configuration scripts.

```
[root@rhos1 ~(demo_member)]# nova boot --config-drive=true \
--image=rhel-server2 \
--user-data /pub/scripts/wp-backend.sh \
--flavor 2 --key-name demokp --file /etc/hosts=/etc/hosts \
--file sysreg.sh=/pub/scripts/sysreg.sh \
--meta sysregopts="-u admin -p labpasswd -s https://ra-
ns1.cloud.lab.eng.bos.redhat.com/XMLRPC -r http://ra-
ns1.cloud.lab.eng.bos.redhat.com/pub/rhn-org-trusted-ssl-cert-1.0-
1.noarch.rpm" \
--meta rhncreds="-u admin -p labpasswd" \
--meta wp_http_ip="10.0.5.201" \
--block-device-mapping vdb=$volid:::0 \
--nic net-id=$netid,v4-fixed-ip="10.0.5.200" ra1

+------------------------------+---------------------------------------------+
| Property                     | Value
+------------------------------+---------------------------------------------+
| status                       | BUILD
| updated                      | 2013-07-25T18:10:05Z
| OS-EXT-STS:task_state        | None
| key_name                     | demokp
| image                        | rhel-server2
| hostId                       |
| OS-EXT-STS:vm_state          | building
| flavor                       | m1.small
| id                           | 71396a1b-dc3b-4b18-bf29-239b5260a424
| security_groups              | [{u'name': u'default'}]
| user_id                      | 365cea099c874ff3b53b94ead32e93da
| name                         | ra1
| adminPass                    | vT2h2xKWADrD
| tenant_id                    | 4f2f60304f574ad4bb40920a39bb8396
| created                      | 2013-07-25T18:10:04Z
| OS-DCF:diskConfig            | MANUAL
| metadata                     | {u'sysregopts': u'-u admin -p labpasswd -s
https://ra-ns1.cloud.lab.eng.bos.redhat.com/XMLRPC -r http://ra-
ns1.cloud.lab.eng.bos.redhat.com/pub/rhn-org-trusted-ssl-cert-1.0-
1.noarch.rpm', u'wp_http_ip': u'10.0.5.201', u'rhncreds': u'-u admin -p
labpasswd'} |
| accessIPv4                   |
| accessIPv6                   |
| progress                     | 0
| OS-EXT-STS:power_state       | 0
| OS-EXT-AZ:availability_zone  | nova
| config_drive                 | 1
+-----------------------------+----------------------------------------------+
```

2. The command line arguments to **`nova boot`** have the following meanings:

- *--config-drive=true*

  OpenStack can write metadata to a configuration drive attached to the instance at boot. The instance retrieves metadata by mounting this drive as an ISO9660 filesystem and reading files from it.

- *--user-data /pub/scripts/wp-backend.sh*

  The metadata service allows instances to retrieve instance-specific data. These data are put in a file on the local system and passed to the instance during boot with the flag *--user-data*. This script orchestrated the configuration.

- *--file sysreg.sh=/pub/scripts/sysreg.sh*

  This option injects a file into the instance at launch. The file is accessed by the user-data scripts or executed through regular initialization scripts. In this example the instance is injected with a script that registers the instance with content providers.

- *--meta sysregopts="-u admin -p password -s https://ra-sat-server.example.com/XMLRPC -r http://ra-sat-server.example.com/pub/rhn-org-trusted-ssl-cert-1.0-1.noarch.rpm"*

  This option inserts metadata to the instance as a key/value pair. In this example, the system registration command line arguments are passed with the key *sysregopts*.

- *--meta rhncreds="-u admin -p password"*

  RHN credentials are also passed as metadata.

- *--block-device-mapping vdd=23d8a454-27ea-4a74-bed1-19b72da42df2:::0*

  This option maps a volume to a local block device. In this example a persistent volume is passed to the instance as */dev/vdd*. This volume contains the database.

7.2.6 Verify the Database Server

1. The database server is not assigned a publicly accessible floating IP as an added layer of security. Connect to the instance via SSH through the network server and verify that **mysqld** is running.

   The static route defined in section **6.4.4 Verify Routing** should still be in place. Otherwise, recreate the route via the **qrouter's public** network address.

```
[root@rhos1 ~(demo_member)]# ssh -i /root/demokp.pem 10.0.5.200 service mysqld status
mysqld (pid  1840) is running...
```

2. Verify the volume is mounted and stores the database directory.

```
[root@rhos1 ~(demo_member)]# ssh -i /root/demokp.pem 10.0.5.200 mount | grep vd
/dev/vdc on /rhos type iso9660 (rw)
/dev/vdb on /var/lib/mysql type ext4 (rw)
```

7.3 Deploy the Web server

This section describes how to install the web server as a virtual machine instance. There are five steps:

- Create a user data script that installs software packages and connects to the database post-boot.

- Create a script that registers the instance with RHN.
- Boot the instance and execute the scripts passing metadata where appropriate.
- Verify the installation.
- Assign a floating IP address to the web server so it is publicly accessible.

7.3.1 Create the User Data Script

The user data script employed in this section is named *wp-frontend.sh*. Registered Red Hat customers can download the script here: **https://access.redhat.com/site/node/455603/40/1**

> **NOTE:** Use the same system registration script described in section **7.2.3 Create the Registration Script**.

7.3.2 Launch the Web Server

1. Launch an instance for the WordPress web server. Specify the database IP address as a metadata variable.

```
[root@rhos1 ~(demo_member)]# nova boot --config-drive=true \
--image rhel-server2 \
--user-data /pub/scripts/wp-frontend.sh \
--flavor 2 --key-name demokp \
--file /etc/hosts=/etc/hosts \
--file sysreg.sh=/pub/scripts/sysreg.sh \
--meta sysregopts="-u admin -p labpasswd -s https://ra-
ns1.cloud.lab.eng.bos.redhat.com/XMLRPC -r http://ra-
ns1.cloud.lab.eng.bos.redhat.com/pub/rhn-org-trusted-ssl-cert-1.0-
1.noarch.rpm" \
--meta rhncreds="-u admin -p labpasswd" \
--meta wp_mysql_ip="10.0.5.200" \
--nic net-id=$netid,v4-fixed-ip="10.0.5.201" ra2

+------------------------------+---------------------------------------------+
| Property                     | Value
+------------------------------+---------------------------------------------+
| status                       | BUILD
| updated                      | 2013-07-25T18:10:12Z
| OS-EXT-STS:task_state        | None
| key_name                     | demokp
| image                        | rhel-server2
| hostId                       |
| OS-EXT-STS:vm_state          | building
| flavor                       | m1.small
| id                           | 002c7eb3-f0c0-4121-b14e-497af7d06330
| security_groups              | [{u'name': u'default'}]
| user_id                      | 365cea099c874ff3b53b94ead32e93da
| name                         | ra2
| adminPass                    | aykntfF4o6Z8
| tenant_id                    | 4f2f60304f574ad4bb40920a39bb8396
| created                      | 2013-07-25T18:10:11Z
| OS-DCF:diskConfig            | MANUAL
| metadata                     | {u'sysregopts': u'-u admin -p labpasswd -s
https://ra-ns1.cloud.lab.eng.bos.redhat.com/XMLRPC -r http://ra-
ns1.cloud.lab.eng.bos.redhat.com/pub/rhn-org-trusted-ssl-cert-1.0-
1.noarch.rpm', u'wp_mysql_ip': u'10.0.5.200', u'rhncreds': u'-u admin -p
labpasswd'} |
| accessIPv4                   |
| accessIPv6                   |
| progress                     | 0
| OS-EXT-STS:power_state       | 0
| OS-EXT-AZ:availability_zone | nova
| config_drive                 | 1
+------------------------------+----------------------------------------------+

[root@rhos1 ~(demo_member)]# nova list
+------------------------------+---------------------------------------------+
| ID                                   | Name  | Status | Networks
+------------------------------+---------------------------------------------+
| 04134a71-da11-4890-9368-840a4b946931 | inst1 | ACTIVE | net1=10.0.5.2,
10.16.137.113   |
| f8a96aac-736f-43eb-a5fd-21bab6732b2b | inst2 | ACTIVE | net1=10.0.5.4
| a467e775-5ac9-46e5-95c8-b73700c80187 | inst3 | ACTIVE | net1=10.0.5.5
| 84276f80-c43b-426d-b34c-fd4313afbb2c | inst4 | ACTIVE | net1=10.0.5.6
| 71396a1b-dc3b-4b18-bf29-239b5260a424 | ra1   | ACTIVE | net1=10.0.5.200
| 002c7eb3-f0c0-4121-b14e-497af7d06330 | ra2   | ACTIVE | net1=10.0.5.201
+------------------------------+---------------------------------------------+
```

2. Several of the command line arguments to **`nova boot`** were described in **7.2.5 Boot the Database Instance** . The remaining parameters are described here.

- *--meta wp_mysql_ip="10.0.5.200"*

  This parameter passes the database server private IP address to the web server.

- *--nic net-id=$netid,v4-fixed-ip="10.0.5.201"*

  This assigns the private interface to the *private* network and specifies its private IP address.

7.3.3 Associate a Public IP Address with the Web Server Instance

1. Find the router port for the web server and save it to an environment variable.

```
[root@rhos1 ~(demo_member)]# ra2_port=$(quantum port-list | awk ' /
10.0.5.201/ { print $2 } ')

[root@rhos1 ~(demo_member)]# echo $ra2_port
9589e93d-9e5d-4098-b75e-4a38eae96f01
```

2. Create the floating IP address and save its Identity Server id to an environment variable.

```
[root@rhos1 ~(demo_member)]# floatip_id=$(quantum floatingip-create ext_net |
awk ' / id/ { print $4 } ')

[root@rhos1 ~(demo_member)]# echo $floatip_id
7381bacb-2f59-44b5-8180-b0d1e3ab78a7
```

3. Associate the floating IP id with the **ra2** router port. It may take one or two minutes before the floating IP address shows up in **`nova list`**.

```
[root@rhos1 ~(demo_member)]# quantum floatingip-associate $floatip_id
$ra2_port
Associated floatingip 7381bacb-2f59-44b5-8180-b0d1e3ab78a7
```

4. Save the floating IP address to an environment variable.

```
[root@rhos1 ~(demo_member)]# floatip=$(quantum floatingip-list | grep
$floatip_id | awk ' { print $6 } ')

[root@rhos1 ~(demo_member)]# echo $floatip
10.16.137.114
```

5. Ping the floating IP address.

```
[root@rhos1 ~(demo_member)]# ping -c 3 $floatip
PING 10.16.137.114 (10.16.137.114) 56(84) bytes of data.
64 bytes from 10.16.137.114: icmp_seq=1 ttl=64 time=1.92 ms
64 bytes from 10.16.137.114: icmp_seq=2 ttl=64 time=0.211 ms
64 bytes from 10.16.137.114: icmp_seq=3 ttl=64 time=0.172 ms

--- 10.16.137.114 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.172/0.768/1.921/0.815 ms
```

7.3.4 Verify the Web Server Installation

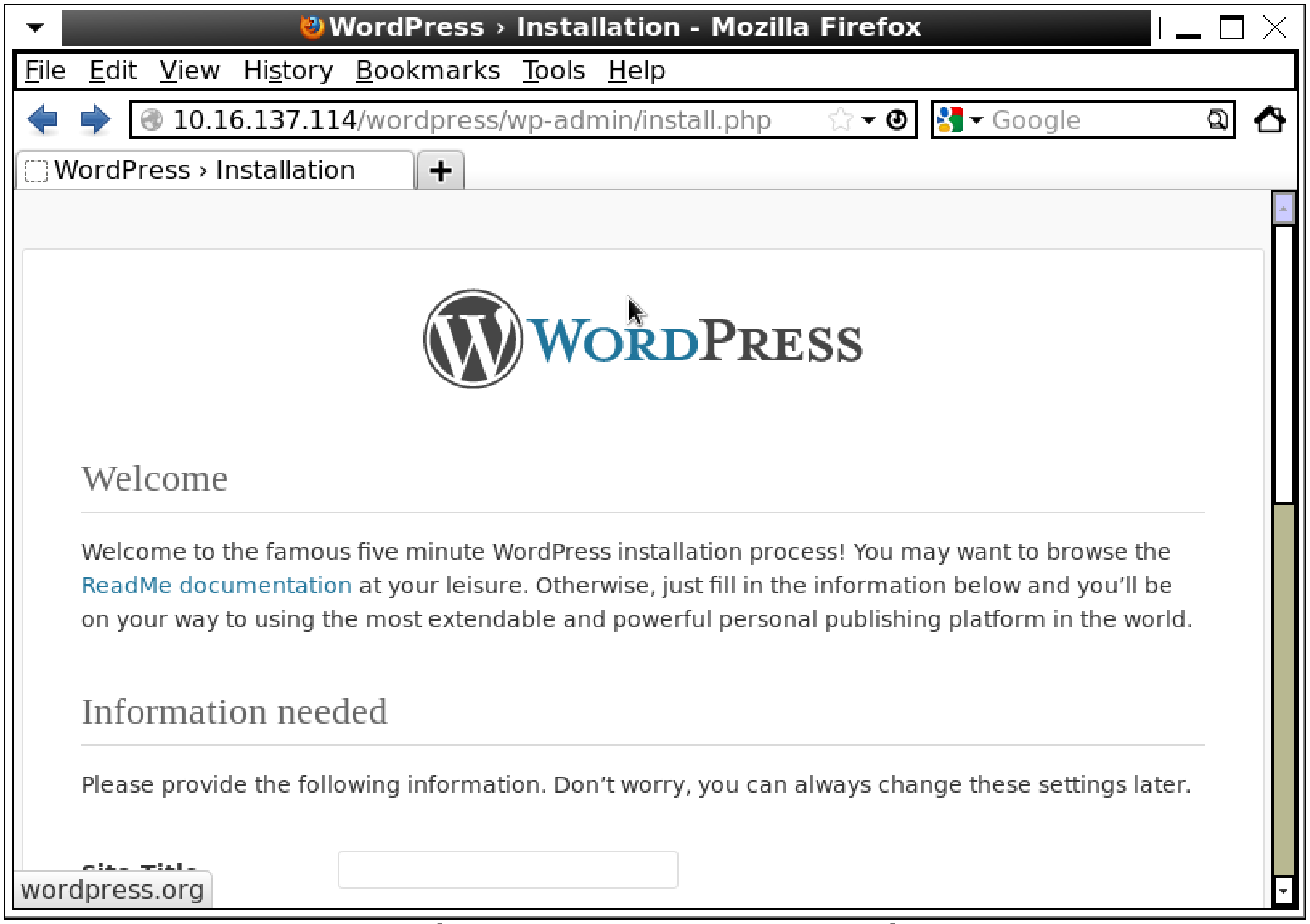
Connect to the instance. Verify **`httpd`** is running and that it can **`ping`** the database server.

```
[root@rhos1 ~(demo_member)]# ssh -i demokp.pem 10.16.137.114
Last login: Fri Jul 26 21:34:05 2013 from rhos1.

[root@ra2 ~]# rpm -qa | grep http
httpd-2.2.15-28.el6_4.x86_64
httpd-tools-2.2.15-28.el6_4.x86_64
lighttpd-1.4.31-1.el6.x86_64

[root@ra2 ~]# service httpd status
httpd (pid  1850) is running...

[root@ra2 ~]# rpm -q wordpress
wordpress-3.5.2-1.el6.noarch

[root@ra2 ~]# ping -c 3 10.0.5.200
PING 10.0.5.200 (10.0.5.200) 56(84) bytes of data.
64 bytes from 10.0.5.200: icmp_seq=1 ttl=64 time=2.64 ms
64 bytes from 10.0.5.200: icmp_seq=2 ttl=64 time=0.782 ms
```

```
64 bytes from 10.0.5.200: icmp_seq=3 ttl=64 time=0.782 ms

--- 10.0.5.200 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.782/1.403/2.647/0.880 ms
```

7.4 Test the Web Server from a Client

Connect to the WordPress page in a browser via the floating IP address. **Illustration 7.4.1WordPress Admin Screen** is displayed after a successful installation.


WordPress › Installation - Mozilla Firefox
File Edit View History Bookmarks Tools Help
10.16.137.114/wordpress/wp-admin/install.php
Google
WordPress › Installation
WordPress
wordpress.org

Welcome

Welcome to the famous five minute WordPress installation process! You may want to browse the ReadMe documentation at your leisure. Otherwise, just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Don't worry, you can always change these settings later.

Site Title

Illustration 7.4.1: WordPress Admin Screen

7.5 Complete Multi-tier Application

Illustration 7.5.1 Multi-tier Application depicts the complete multi-tier web application deployed in this section.

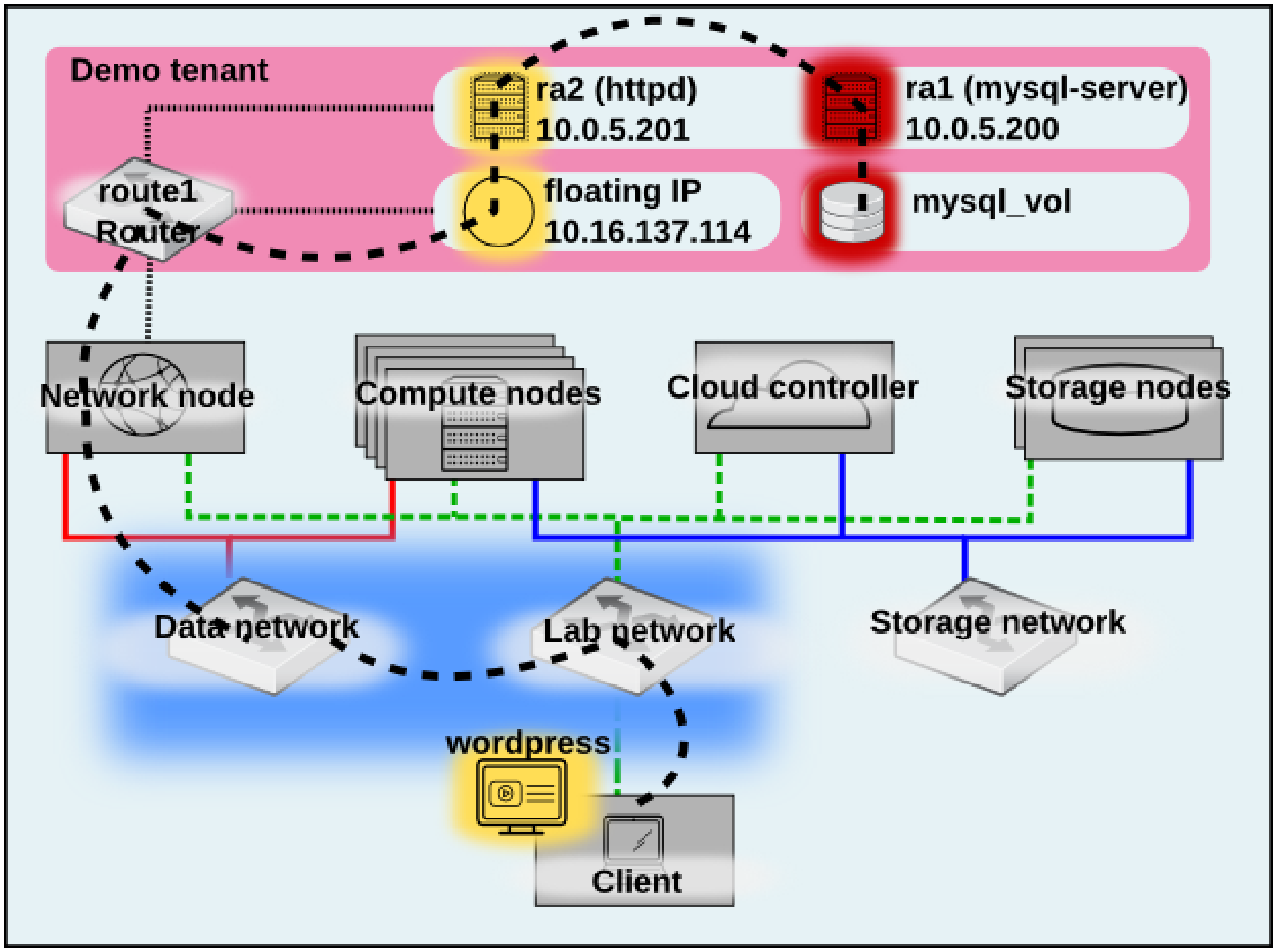


Illustration 7.5.1: Multi-tier Application

The black dashed line traces connection points from the client accessing the WordPress application to the instance resident on the compute node through to the database volume. The **ra2** instance with floating IP 10.16.137.114 serves the WordPress application. The **ra1** instance mounts the Cinder volume which stores the MySQL database. The instances communicate via the **private** Neutron subnet. Traffic to physical machines on the Lab network, such as the client, traverses the software router via the Network node's external bridge interface.

Appendix C: Full Network Diagram expands the view of the complete network diagram to include host names and IP addresses.

8 Conclusion

Red Hat Enterprise Linux OpenStack Platform 3 provides a flexible cloud framework built on a stable code base and backed by Red Hat's open source software expertise. Red Hat Enterprise Linux OpenStack Platform 3 allows administrators to build an IaaS cloud quickly and easily. Self service users can deploy their own instances to the cloud using the dashboard interface or command line tools.

The goal of this reference architecture is to provide guidance on how to complete the following tasks:

- Deploy the cloud controller via **`packstack`**
- Configure Red Hat Storage backed Block Storage and Image Services
- Define a custom OpenStack Network Service network using the openvswitch plugin
- Deploy a multi-tier application

This document covers each of these topics in sufficient detail to allow Red Hat customers to reproduce them in their own environments. The Red Hat Enterprise Linux OpenStack Platform configuration described in this document can be customized and expanded to meet specific requirements. Future reference architectures build on the features exposed in this document add additional OpenStack services as well as strategies for integrating Red Hat Enterprise Linux OpenStack Platform with other Red Hat products.

Appendix A: Red Hat Storage Setup

This appendix details installation of Red Hat Storage Server and the creation of two distributed Gluster volumes, one for the Image Service and another for the Block Storage Service. Setting up Red Hat Storage Server requires the following steps:

- Prepare the Red Hat Storage Server nodes
- Register the Nodes with RHN and Update
- Synchronize Time Service
- Configure DNS
- Update Host File
- Kernel Tuning using Tuned
- Create Gluster Volume

A.1 *Overview*

The RHS Server is composed of two servers. Each server hosts two local XFS file systems called bricks. One brick from each RHS Server is combined with a corresponding brick on the other RHS Server to make a replicated volume. Therefore, the RHS Servers present two replicated volumes – one for the Image Service and one for Block Storage Service – composed of four bricks. Both volumes are synchronously replicated. If either RHS Server becomes unavailable, all data is still available via the remaining node.

A.2 *Prepare the RHS Server Nodes*

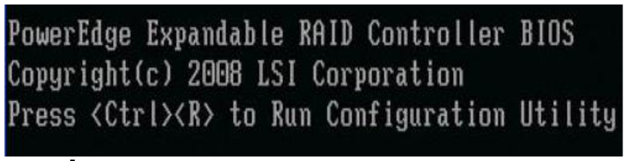
Prior to server installation, a virtual disk must be configured on each RHS Server node.

The virtual disk has the following parameters set:
Stripe unit – 256K
Stripe width – 6 disks with RAID 6 (perceived width is 4)

Note: A disk pool with more disks is preferred for better performance.

A.2.1 Create a Virtual Disk (LUN) with Raid 6 using PERC Controller

1. During host system bootup, press <Ctrl><R> when the BIOS banner displays.



```
PowerEdge Expandable RAID Controller BIOS
Copyright(c) 2008 LSI Corporation
Press <Ctrl><R> to Run Configuration Utility
```

Figure A-1: Launch PERC BIOS Configuration Utility

The *Virtual Disk Management* screen displays.

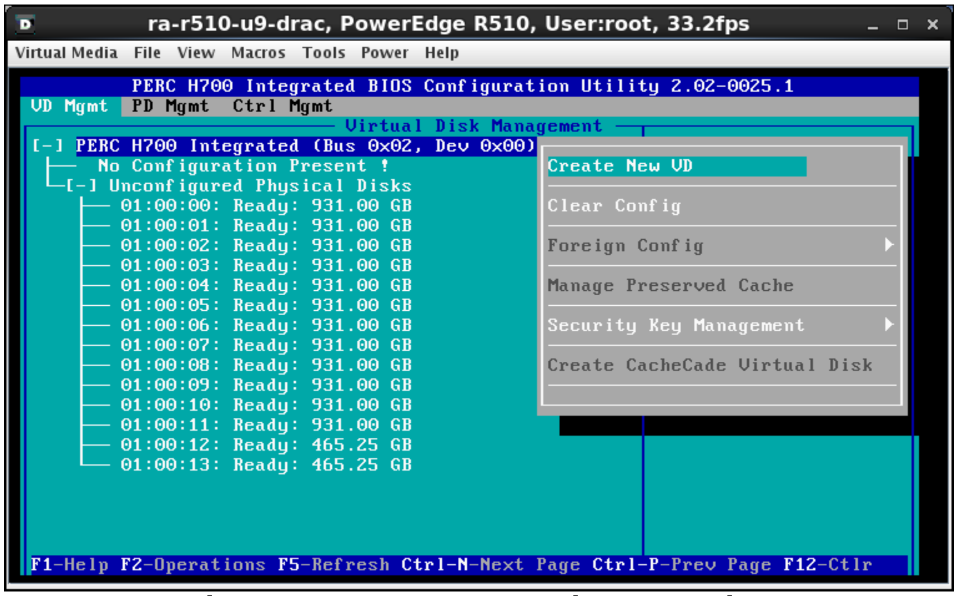
This launches the PERC Integrated BIOS Configuration Utility. This utility has the following three TABs on the top.

- VD Mgmt – Virtual Disk Management, which is selected by default.
- PD Mgmt – Physical Disk Management
- Ctrl Mgmt – Controller Management

If there is more than one controller, the main menu screen displays. Select a controller, and press **Enter**. The *Virtual Disk Management* screen displays for the selected controller.

NOTE: The PERC does not support creation of a virtual disk that combines SAS and SATA disks.

2. Use the arrow keys to highlight *Controller #* or *Disk Group #*. Press <F2> to display the actions you can perform.



```
ra-r510-u9-drac, PowerEdge R510, User:root, 33.2fps
Virtual Media  File  View  Macros  Tools  Power  Help
PERC H700 Integrated BIOS Configuration Utility 2.02-0025.1
VD Mgmt  PD Mgmt  Ctrl Mgmt
                        Virtual Disk Management
[-] PERC H700 Integrated (Bus 0x02, Dev 0x00)
    ├── No Configuration Present !
    └─[-] Unconfigured Physical Disks
       ├── 01:00:00: Ready: 931.00 GB
       ├── 01:00:01: Ready: 931.00 GB
       ├── 01:00:02: Ready: 931.00 GB
       ├── 01:00:03: Ready: 931.00 GB
       ├── 01:00:04: Ready: 931.00 GB
       ├── 01:00:05: Ready: 931.00 GB
       ├── 01:00:06: Ready: 931.00 GB
       ├── 01:00:07: Ready: 931.00 GB
       ├── 01:00:08: Ready: 931.00 GB
       ├── 01:00:09: Ready: 931.00 GB
       ├── 01:00:10: Ready: 931.00 GB
       ├── 01:00:11: Ready: 931.00 GB
       ├── 01:00:12: Ready: 465.25 GB
       └── 01:00:13: Ready: 465.25 GB
F1-Help F2-Operations F5-Refresh Ctrl-N-Next Page Ctrl-P-Prev Page F12-Ctlr
```

Figure A-2: Create Virtual Disk

3. Select *Create New VD* and press **Enter**. The *Create New VD* screen displays.

4. Press **Enter** to display the possible RAID levels, based on the physical disks available.

5. Press the down arrow key to select a RAID level (**RAID 6**)and press **Enter**.

6. Press <Tab> to move the cursor to the list of physical disks.

7. Use the arrow key to highlight a physical disk and press the space bar, <Alt>, or **Enter** to select the disk.

8. Select additional disks, if desired.

9. Press <Tab> to move the cursor to the box *Basic Settings*.

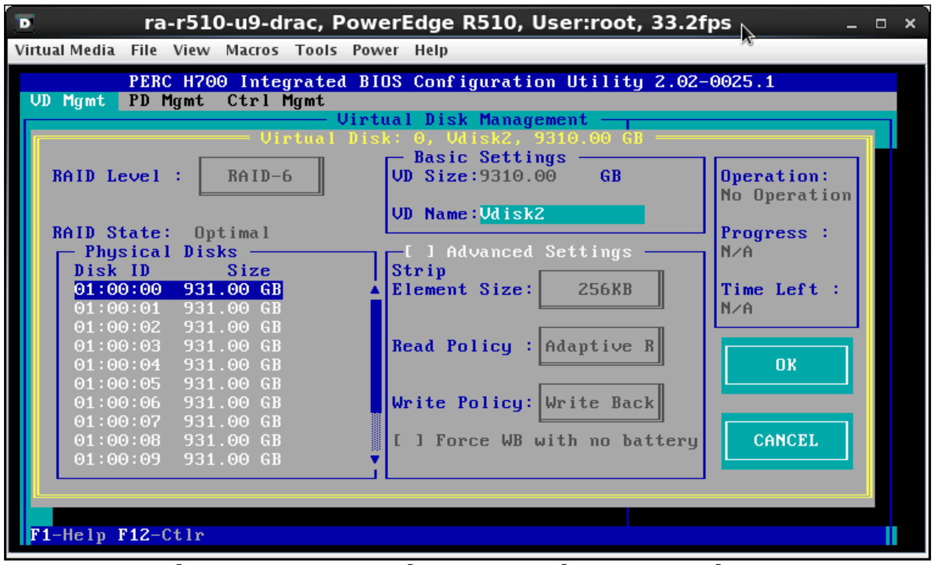
10. Press <Tab> to access the **VD Size** field, and type a virtual disk name (**Vdisk2)**. The virtual disk size Vdisk2 displays the final pool size based on the total disk space, number of disks and type of RAID selected.

11. Press <Tab> to move the cursor to *Advanced Settings.* Enter

    It is important that the following settings are selected in the *Advanced Settings*:

    - Stripe Element Size: **256KB**
    - Read Policy: **Adaptive**
    - Write Policy: **Write Back**

NOTE: The Write Policy has to be set to Write Back for performance reasons. It is implied that the RAID Controller has *Battery Backup Unit BBU.* If this is not the case, Write Policy is set to *Write Through* by default and caching is disabled. Please verify with the vendor before proceeding further.


ra-r510-u9-drac, PowerEdge R510, User:root, 33.2fps
Virtual Media File View Macros Tools Power Help

```
PERC H700 Integrated BIOS Configuration Utility 2.02-0025.1
VD Mgmt  PD Mgmt  Ctrl Mgmt
Virtual Disk Management
Virtual Disk: 0, Vdisk2, 9310.00 GB

RAID Level : RAID-6            Basic Settings              Operation:
                               VD Size:9310.00    GB       No Operation
                               VD Name:Vdisk2
RAID State:  Optimal                                       Progress :
 Physical Disks                [ ] Advanced Settings       N/A
  Disk ID       Size           Strip
  01:00:00   931.00 GB         Element Size:  256KB        Time Left :
  01:00:01   931.00 GB                                     N/A
  01:00:02   931.00 GB
  01:00:03   931.00 GB         Read Policy : Adaptive R
  01:00:04   931.00 GB                                     OK
  01:00:05   931.00 GB
  01:00:06   931.00 GB         Write Policy: Write Back
  01:00:07   931.00 GB
  01:00:08   931.00 GB         [ ] Force WB with no battery  CANCEL
  01:00:09   931.00 GB

F1-Help F12-Ctlr
```

Figure A-3: Virtual Disk Settings

12. Select OK to accept the settings and press Enter to exit this window.

13. Initialize the Virtual Disk by selecting the right Virtual Disk on the VD Mgmt Tab -> **F2** -> **Initialization-> Start Init** (or Select **Fast Init**)

A.2.2 *Install RHS Server*

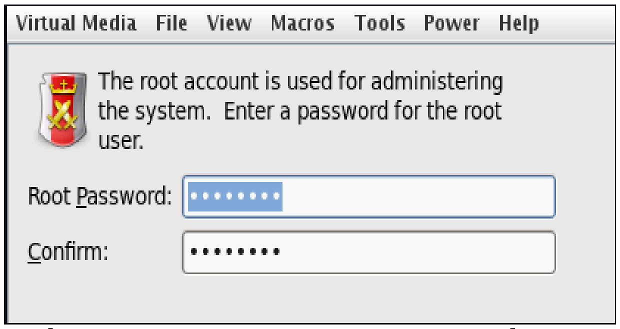
This section provides a brief overview of how to install Red Hat Storage Server on the storage nodes. For detailed description please refer to the Red Hat Storage Documentation: *https://access.redhat.com/site/documentation/Red_Hat_Storage/*

Installing Red Hat Storage Server is similar to installing Red Hat Enterprise Linux, but with far fewer choices to make during the installation process. The basic procedure is outlined below.

1. Obtain an installation ISO for Red Hat Storage Server from https://access.redhat.com The image used in this reference architecture is: **RHS-2.0-20130320.2-RHS-x86_64-DVD1.iso**

2. Mount the ISO image before the new hardware is booted up. In this case, the hardware are Dell Servers. The virtual CD/DVD can be mounted on the IDRAC console. Alternatively, this image can be burned as a physical DVD and used for installation.

3. Boot the server. Enter the root password and click on **Next** to continue.



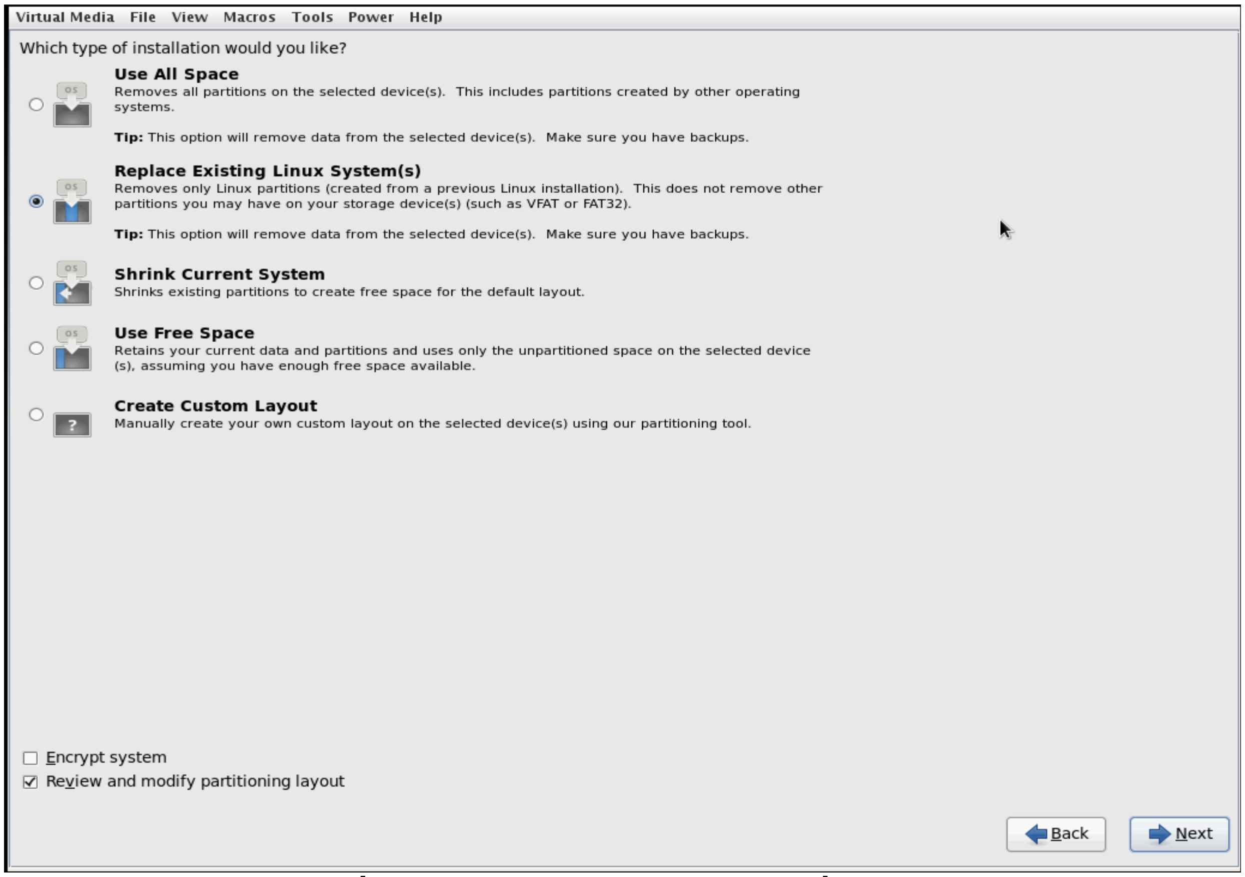
The root account is used for administering the system. Enter a password for the root user.

Root Password: ••••••••

Confirm: ••••••••

Figure A-4: RHS Installation-1

4. Select a layout on the partitioning screen.



Which type of installation would you like?

( ) **Use All Space**
Removes all partitions on the selected device(s). This includes partitions created by other operating systems.

Tip: This option will remove data from the selected device(s). Make sure you have backups.

(●) **Replace Existing Linux System(s)**
Removes only Linux partitions (created from a previous Linux installation). This does not remove other partitions you may have on your storage device(s) (such as VFAT or FAT32).

Tip: This option will remove data from the selected device(s). Make sure you have backups.

( ) **Shrink Current System**
Shrinks existing partitions to create free space for the default layout.

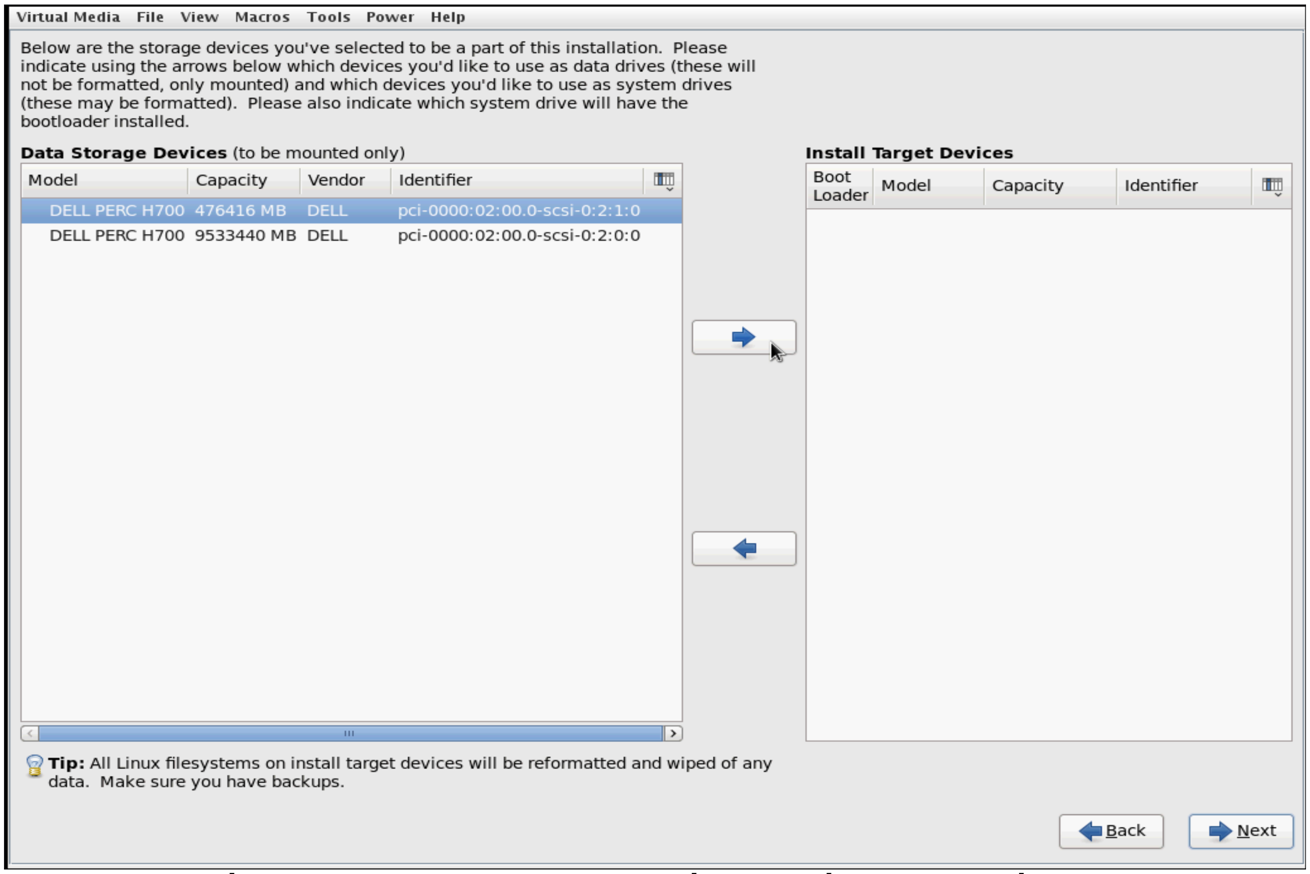
( ) **Use Free Space**
Retains your current data and partitions and uses only the unpartitioned space on the selected device (s), assuming you have enough free space available.

( ) **Create Custom Layout**
Manually create your own custom layout on the selected device(s) using our partitioning tool.

☐ Encrypt system
☑ Review and modify partitioning layout

Figure A-5: RHS Installation-2

5. Available disks are displayed.



Below are the storage devices you've selected to be a part of this installation. Please indicate using the arrows below which devices you'd like to use as data drives (these will not be formatted, only mounted) and which devices you'd like to use as system drives (these may be formatted). Please also indicate which system drive will have the bootloader installed.

Data Storage Devices (to be mounted only)

Model	Capacity	Vendor	Identifier
DELL PERC H700	476416 MB	DELL	pci-0000:02:00.0-scsi-0:2:1:0
DELL PERC H700	9533440 MB	DELL	pci-0000:02:00.0-scsi-0:2:0:0

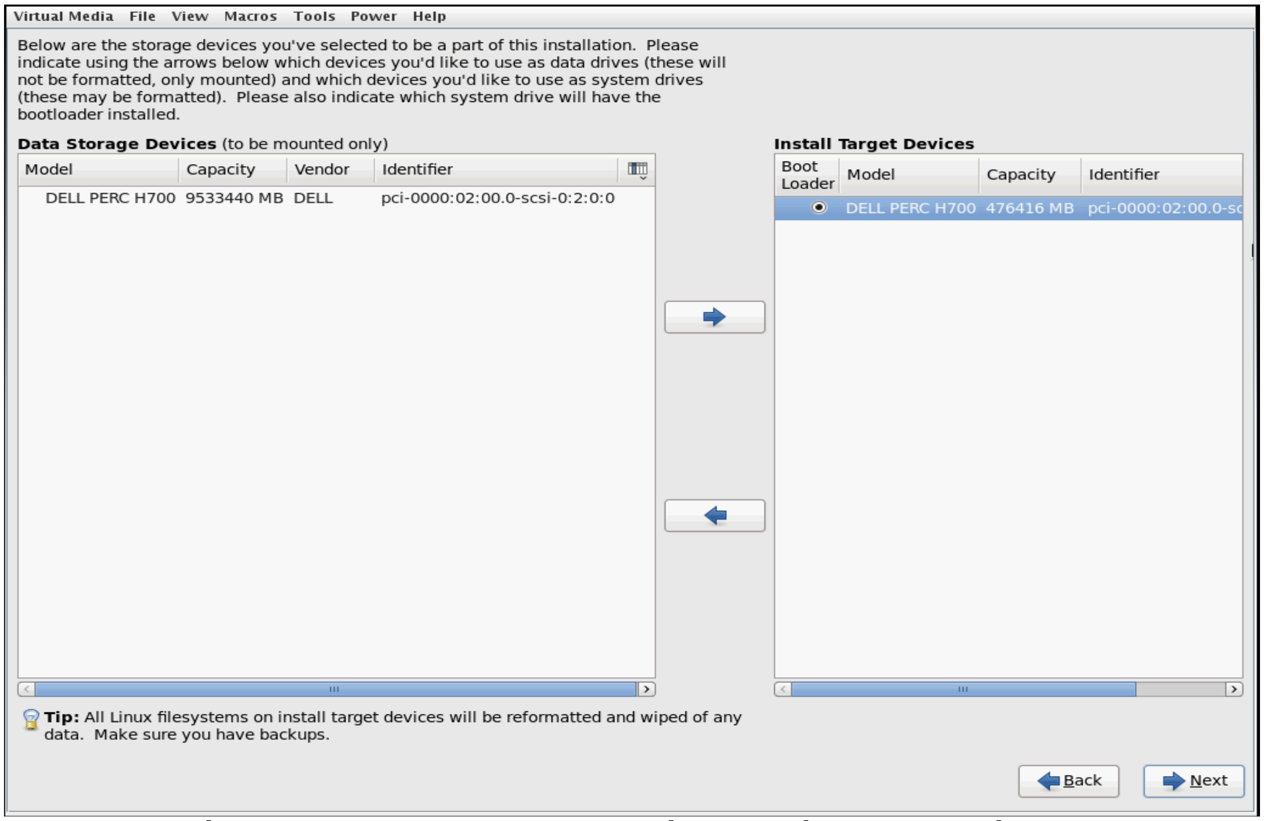
Install Target Devices

Boot Loader	Model	Capacity	Identifier

Tip: All Linux filesystems on install target devices will be reformatted and wiped of any data. Make sure you have backups.

Figure A-6: RHS Installation - Disk selection 1

6. Ensure that the right disk and size is allocated to the operating system. Select **Next** to proceed with the installation.



Below are the storage devices you've selected to be a part of this installation. Please indicate using the arrows below which devices you'd like to use as data drives (these will not be formatted, only mounted) and which devices you'd like to use as system drives (these may be formatted). Please also indicate which system drive will have the bootloader installed.

Data Storage Devices (to be mounted only)

Model	Capacity	Vendor	Identifier
DELL PERC H700	9533440 MB	DELL	pci-0000:02:00.0-scsi-0:2:0:0

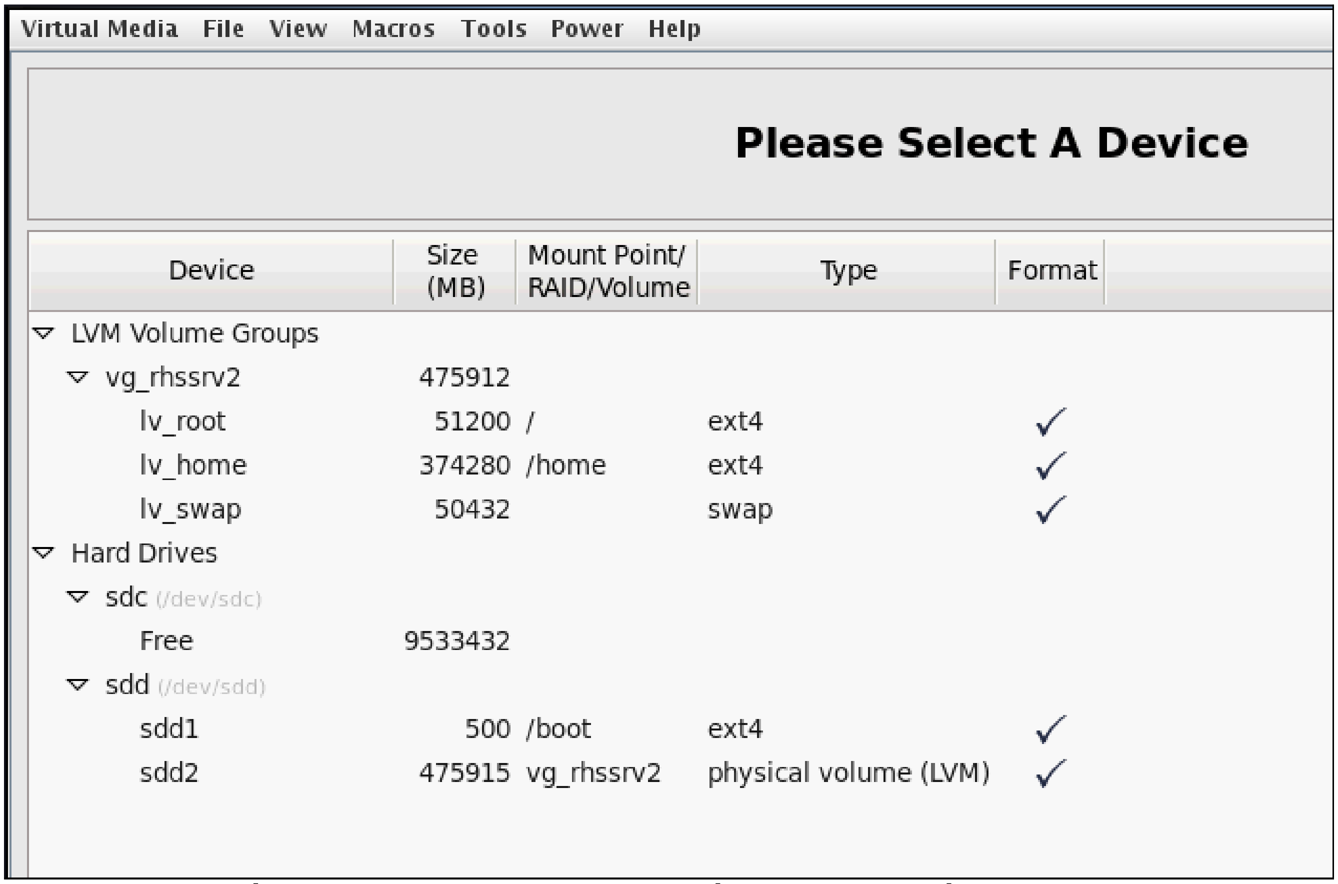
Install Target Devices

Boot Loader	Model	Capacity	Identifier
●	DELL PERC H700	476416 MB	pci-0000:02:00.0-sc

Tip: All Linux filesystems on install target devices will be reformatted and wiped of any data. Make sure you have backups.

Figure A-7: RHS Installation - Disk selection 2

7. Ensure root partition has been allocated at least 2048 MB of disk space. Delete the **lv_home** volume and add this freed space to the root volume.

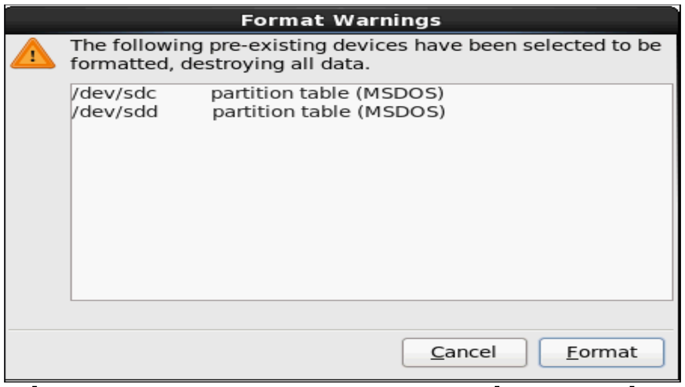


Please Select A Device

Device	Size (MB)	Mount Point/ RAID/Volume	Type	Format
LVM Volume Groups				
vg_rhssrv2	475912			
lv_root	51200	/	ext4	✓
lv_home	374280	/home	ext4	✓
lv_swap	50432		swap	✓
Hard Drives				
sdc (/dev/sdc)				
Free	9533432			
sdd (/dev/sdd)				
sdd1	500	/boot	ext4	✓
sdd2	475915	vg_rhssrv2	physical volume (LVM)	✓

Figure A-8: RHS Installation - Root Filesystem

8. Confirm formatting of the right disks.

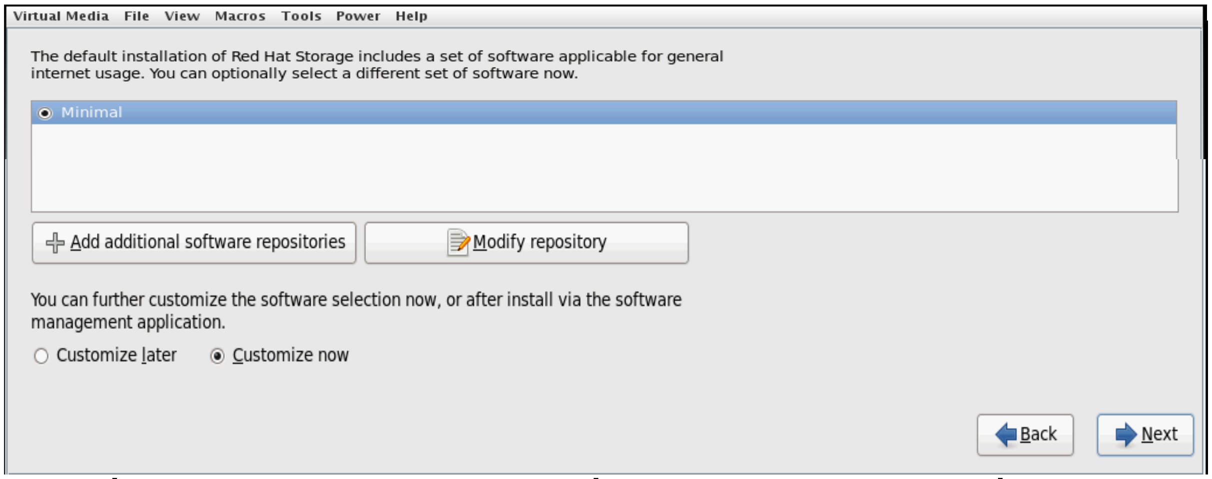


The following pre-existing devices have been selected to be formatted, destroying all data.

```
/dev/sdc        partition table (MSDOS)
/dev/sdd        partition table (MSDOS)
```

Figure A-9: RHS Installation - Disk formatting

9. Upon confirmation, the installer creates the file systems and proceed to install the default package set for Red Hat Storage Server. By selecting **customize now** the software subsets can be viewed/modified.


Virtual Media File View Macros Tools Power Help

The default installation of Red Hat Storage includes a set of software applicable for general internet usage. You can optionally select a different set of software now.

[✓ Minimal]

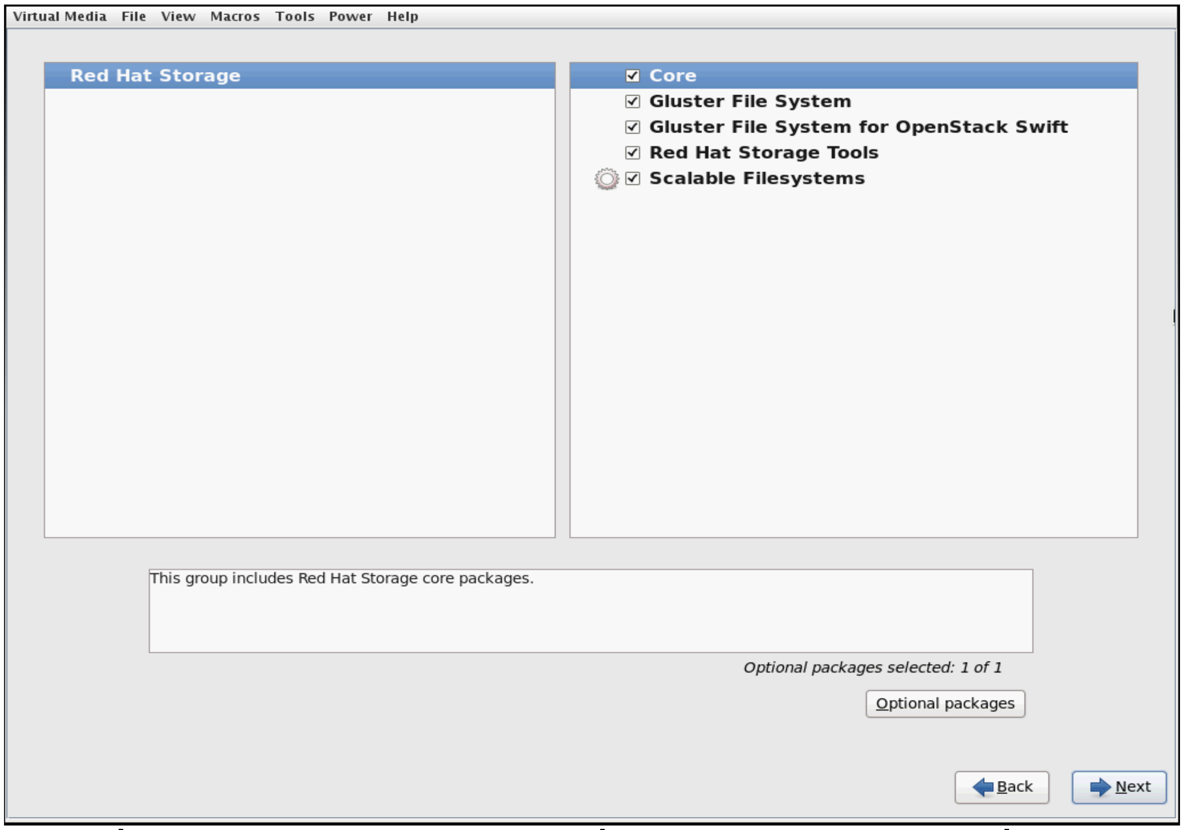
Add additional software repositories
Modify repository

You can further customize the software selection now, or after install via the software management application.

Customize later [✓ Customize now]

Figure A-10: RHS Installation - Software Selection 1

10. Select **Next** without having to deselect any of these subsets.



- ✓ Core
- ✓ Gluster File System
- ✓ Gluster File System for OpenStack Swift
- ✓ Red Hat Storage Tools
- ✓ Scalable Filesystems

This group includes Red Hat Storage core packages.

Optional packages selected: 1 of 1

Figure A-11: RHS Installation - Software Selection 2

11. Once the installer finishes, select **Reboot** button to finish the installation. After the server reboots, log in as root using the password specified earlier.

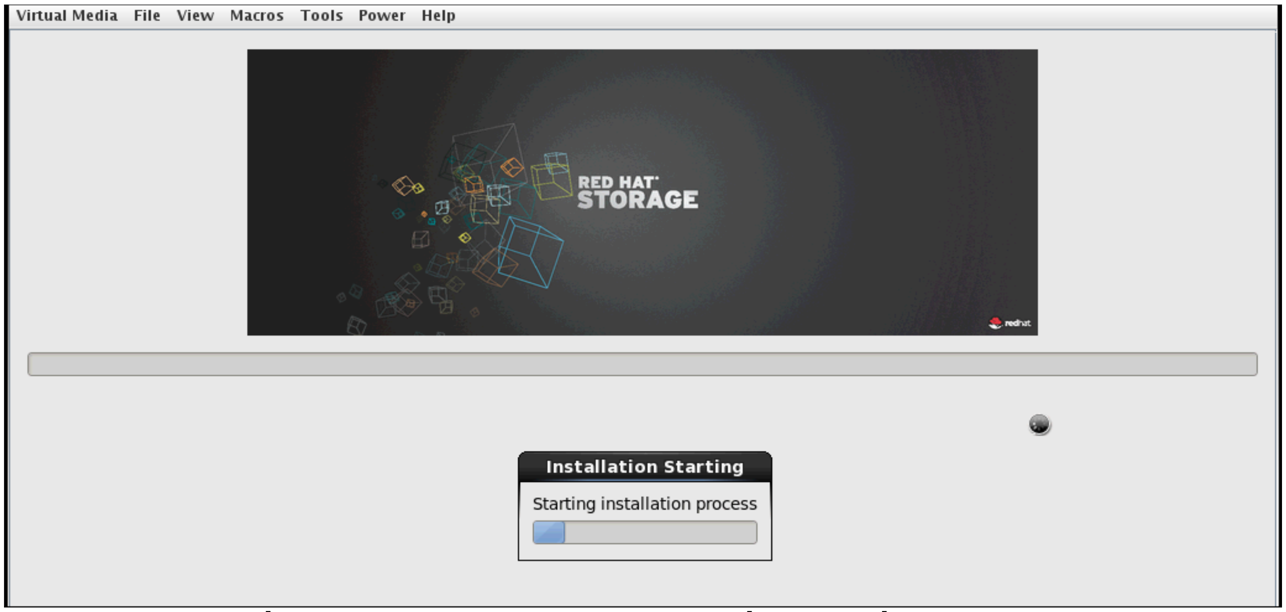

Virtual Media File View Macros Tools Power Help
RED HAT STORAGE
redhat
Installation Starting
Starting installation process

Figure A-12: RHS Installation - Final Step

12. Verify the glusterd service has been automatically started.

```
# service glusterd status
glusterd (pid  1434) is running...
```

A.3 *Register and Update Storage Servers*

1. Run the **rhn_register** command to register the system with Red Hat Network. Supply the Red Hat Network username and password and follow the on-screen prompts to complete registration.

```
# rhn_register
```

2. Subscribe to the required channels using **`rhn-channel`**.

```
# rhn-channel --add --channel=rhel-x86_64-server-6-rhs-2.0
# rhn-channel --add --channel=rhel-x86_64-server-sfs-6.2.z
```

3. List the channels to ensure the system is registered successfully.

```
# rhn-channel -l
rhel-x86_64-server-6-rhs-2.0
rhel-x86_64-server-6.2.z
rhel-x86_64-server-sfs-6.2.z
```

4. Run **yum** to update and ensure the Red Hat Storage servers are kept up-to-date with security patches and bug fixes.

```
# yum update
```

A.4 *Apply iptables Changes*

1. Open the RHS-required ports on each storage node.

```
# iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 111 -j ACCEPT

# iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 24007 -j ACCEPT

# iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 24008 -j ACCEPT

# iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 24009 -j ACCEPT

# iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 24010 -j ACCEPT

# iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 24011 -j ACCEPT

# iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 38465:38469 -j
ACCEPT

# service iptables save
iptables: Saving firewall rules to /etc/sysconfig/iptables:[  OK  ]
```

A.5 *Tune Kernel with Tuned*

On each node, perform kernel tuning using the tuned profile *rhs-virtualization* as follows:

```
# tuned-adm profile rhs-virtualization
Switching to profile 'rhs-virtualization'
Applying ktune sysctl settings:
/etc/ktune.d/tunedadm.conf:                                [  OK  ]
Calling '/etc/ktune.d/tunedadm.sh start': setting readahead to 4096 on
brickdevices: readlink: missing operand
Try `readlink --help' for more information.
basename: missing operand
Try `basename --help' for more information.
 /etc/ktune.d/tunedadm.sh: line 42: /sys/block//queue/read_ahead_kb: No such
file or directory
readlink: missing operand
Try `readlink --help' for more information.
basename: missing operand
Try `basename --help' for more information.
 /etc/ktune.d/tunedadm.sh: line 42: /sys/block//queue/read_ahead_kb: No such
file or directory
readlink: missing operand
Try `readlink --help' for more information.
basename: missing operand
Try `basename --help' for more information.
 /etc/ktune.d/tunedadm.sh: line 42: /sys/block//queue/read_ahead_kb: No such
```

```
file or directory
readlink: missing operand
Try `readlink --help' for more information.
basename: missing operand
Try `basename --help' for more information.
 /etc/ktune.d/tunedadm.sh: line 42: /sys/block//queue/read_ahead_kb: No such
file or directory

                                                           [  OK  ]
Applying sysctl settings from /etc/sysctl.conf
Applying deadline elevator: dm-0 dm-1 dm-2 dm-3 dm-4 dm-5 sdc [  OK  ]
Starting tuned:                                            [  OK  ]
```

This profile performs the following:

- Attempts to increases read-ahead to 64 MB. (BZ 910566)
- Configure the cpuspeed daemon for performance
- Changes I/O scheduler to "deadline"
- Change kernel parameters
  - set the preemption granularity
  - set the wakeup granularity
  - set VM swapiness to 10
  - set the VM writeback dirty ratio to 1%
  - several network settings
- Turns off write barriers (assumes RAID controller has non-volatile writeback caching and that disks are set to - writeback caching off)

A.6 *Set up Trusted Storage Pools*

Before creating a Red Hat Storage volume, a trusted storage pool must be created, consisting of the storage servers that provide bricks to a volume. The storage pool is a trusted network of Red Hat Storage servers. In this reference architecture the servers communicate via 10GbE NICs on a VLAN dedicated for network storage traffic. From the first node other peers can be added to the trusted storage pool by using the **probe** command.

1. Execute the following command on ra-*rhs-srv1* to probe for additional nodes.

```
# gluster peer probe ra-rhs-srv2-10g
Probe successful
```

2. Verify the trusted pool.

```
# gluster peer status
Number of Peers: 1

Hostname: ra-rhs-srv2-10g
Port: 24007
Uuid: 47d7bcfe-9117-48f7-9c3f-72b1aab9ff54
State: Peer in Cluster (Connected)
```

NOTE: While in this case the probe command is being executed from the first node, it could be issued from any node in the network that has RHS software installed. However while adding an additional node to an existing pool, the probe command has to originate from one of the trusted nodes in the pool. This ensures integrity. Join requests must originate with a trusted node.

A.7 *Create and Mount XFS filesystems for Bricks*

In this section, file systems are created on the Raid 6 virtual disks described under **A.2 Prepare the RHS Server Nodes**. These steps must be performed on both nodes.

1. Display the size of the discovered disk with **`fdisk`**.

```
# fdisk -l /dev/sdb

Disk /dev/sdb: 3998.6 GB, 3998614552576 bytes
255 heads, 63 sectors/track, 486137 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

Disk /dev/sdb doesn't contain a valid partition table
```

2. Initialize the disk using **pvcreate**.

```
# pvcreate --dataalignment 1024k /dev/sdb
  Writing physical volume data to disk "/dev/sdb"
  Physical volume "/dev/sdb" successfully created
```

3. Confirm the location of the first Physical Extent of physical volume */dev/sdb*. This is a multiple of the requested data alignment - 1024k, calculated from the product of 256k stripe size with an active stripe width of 4.

```
# pvs -o +pe_start --units k
  PV         VG           Fmt  Attr PSize          PFree          1st PE
  /dev/sda2  vg_rarhssrv1 lvm2 a--    487333888.00k             0k 1024.00k
  /dev/sdb                lvm2 a--   3904831488.00k  683606016.00k 1024.00k
```

4. Create a volume group named *RHOSvg* with 64mb extent size using LUN */dev/sdb*.

```
# vgcreate RHOSvg /dev/sdb
Volume group "RHOSvg" successfully created
Using volume group(s) on command line
Finding volume group "RHOSvg"
```

5. Create logical volumes using **`lvcreate`**.

```
# lvcreate -L 1T -n glance_lv RHOSvg
  Logical volume "glance_lv" created
```

```
# lvcreate -L 1T -n cinder_lv RHOSvg
  Logical volume "cinder_lv" created
```

6. Display the logical volumes.

```
# lvs
  LV            VG           Attr     LSize   Pool Origin Data%  Move Log 
Copy%  Convert
  cinder_lv     RHOSvg       -wi-ao--   1.00t
  glance_lv     RHOSvg       -wi-ao--   1.00t
  lv_root       vg_rarhssrv1 -wi-ao-- 415.51g
  lv_swap       vg_rarhssrv1 -wi-ao--  49.25g
```

7. Create XFS file systems on the new logical volumes using **`mkfs.xfs`** with the following parameters:

   - Inode size = 512 bytes
   - Stripe unit = 256K
   - Stripe width =4

```
# mkfs.xfs -i size=1024,maxpct=0 -n size=8192 -d su=256k,sw=4 
/dev/RHOSvg/glance_lv
meta-data=/dev/mapper/RHOSvg-glance_lv isize=1024    agcount=300, 
agsize=894848 blks
         =                       sectsz=512   attr=2
data     =                       bsize=4096   blocks=268435456, imaxpct=0
         =                       sunit=64     swidth=256 blks
naming   =version 2              bsize=8192   ascii-ci=0
log      =internal               bsize=4096   blocks=131072, version=2
         =                       sectsz=512   sunit=64 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0

# mkfs.xfs -i size=1024,maxpct=0 -n size=8192 -d su=256k,sw=4 
/dev/RHOSvg/cinder_lv
meta-data=/dev/mapper/RHOSvg-cinder_lv isize=1024    agcount=300, 
agsize=894848 blks
         =                       sectsz=512   attr=2
data     =                       bsize=4096   blocks=268435456, imaxpct=0
         =                       sunit=64     swidth=256 blks
naming   =version 2              bsize=8192   ascii-ci=0
log      =internal               bsize=4096   blocks=131072, version=2
         =                       sectsz=512   sunit=64 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

8. Mount the XFS file systems with the following mount options:

   - inode64
   - allocsize=4k
   - logbsize=256knodiratime
   - noatime

- nodiratime

```
# mkdir -p /.rhs/RHOScinder

# mkdir -p /.rhs/RHOSglance

# echo "/dev/RHOSvg/cinder_lv   /.rhs/RHOScinder        xfs
inode64,allocsize=4k,logbsize=256k,noatime,nodiratime 1 3” >> /etc/fstab

# echo “/dev/RHOSvg/glance_lv   /.rhs/RHOSglance        xfs
inode64,allocsize=4k,logbsize=256k,noatime,nodiratime 1 3 ” >> /etc/fstab

# mount -a
```

9. Verify the file systems and mount options.

```
# mount -v | grep RHOSvg
/dev/mapper/RHOSvg-cinder_lv on /.rhs/RHOScinder type xfs
(rw,noatime,nodiratime,inode64,allocsize=4k,logbsize=256k,nobarrier)
/dev/mapper/RHOSvg-glance_lv on /.rhs/RHOSglance type xfs
(rw,noatime,nodiratime,inode64,allocsize=4k,logbsize=256k,nobarrier)
```

10. Repeat the steps in this section on both nodes.

A.8 *Create Replica Volumes*

Replica volumes ensure data consistency across volumes for fault tolerance.

1. Create a brick sub-directory under both mount points.

```
# mkdir /.rhs/RHOScinder/RHOScinder_brick
# mkdir /.rhs/RHOSglance/RHOSglance_brick
```

   Repeat this step on each node.

2. On only one node, create a replica volume for each gluster volume.

```
# gluster volume create RHOScinder replica 2 ra-rhs-srv1
10g:/.rhs/RHOScinder/RHOScinder_brick ra-rhs-srv2-
10g:/.rhs/RHOScinder/RHOScinder_brick
Creation of volume RHOScinder has been successful. Please start the volume to
access data.

# gluster volume create RHOSglance replica 2 ra-rhs-srv1
10g:/.rhs/RHOSglance/RHOSglance_brick ra-rhs-srv2
10g:/.rhs/RHOSglance/RHOSglance_brick
Creation of volume RHOSglance has been successful. Please start the volume to
access data.
```

3. Tune the Block Storage volume for virtualization.

```
# gluster volume set RHOScinder group virt
Set volume successful
```

4. Start the gluster volumes.

```
# gluster volume start RHOScinder
Starting volume RHOScinder has been successful
# gluster volume start RHOSglance
Starting volume RHOSglance has been successful
```

5. Verify the gluster replica volumes started successfully.

```
# gluster
gluster> volume status RHOScinder
Status of volume: RHOScinder
Gluster process                              Port  Online     Pid
--------------------------------------------------------------------------
Brick ra-rhs-srv1-10g:/.rhs/RHOScinder/RHOScinder_brick 24009 Y     29611
Brick ra-rhs-srv2-10g:/.rhs/RHOScinder/RHOScinder_brick 24009 Y     2253
NFS Server on localhost                           38467 Y     10464
Self-heal Daemon on localhost                     N/A   Y     12774
NFS Server on ra-rhs-srv2-10g                     38467 Y     24345
Self-heal Daemon on ra-rhs-srv2-10g               N/A   Y     4125

gluster> volume info RHOScinder

Volume Name: RHOScinder
Type: Replicate
Volume ID: f52a926b-9093-4e51-b2d6-090cba0454c3
Status: Started
Number of Bricks: 1 x 2 = 2
Transport-type: tcp
Bricks:
Brick1: ra-rhs-srv1-10g:/.rhs/RHOScinder/RHOScinder_brick
Brick2: ra-rhs-srv2-10g:/.rhs/RHOScinder/RHOScinder_brick
Options Reconfigured:
network.remote-dio: on
cluster.eager-lock: enable
performance.stat-prefetch: off
performance.io-cache: off
performance.read-ahead: off
performance.quick-read: off
gluster> volume status RHOSglance
Status of volume: RHOSglance
Gluster process                              Port  Online     Pid
--------------------------------------------------------------------------
Brick ra-rhs-srv1-10g:/.rhs/RHOSglance/RHOSglance_brick 24010 Y     29699
Brick ra-rhs-srv2-10g:/.rhs/RHOSglance/RHOSglance_brick 24010 Y     2248
NFS Server on localhost                           38467 Y     10464
Self-heal Daemon on localhost                     N/A   Y     12774
NFS Server on ra-rhs-srv2-10g                     38467 Y     24345
Self-heal Daemon on ra-rhs-srv2-10g               N/A   Y     4125
```

```
gluster> volume info RHOSglance

Volume Name: RHOSglance
Type: Replicate
Volume ID: ac0e1051-b394-4024-8417-b0eb82e08752
Status: Started
Number of Bricks: 1 x 2 = 2
Transport-type: tcp
Bricks:
Brick1: ra-rhs-srv1-10g:/.rhs/RHOSglance/RHOSglance_brick
Brick2: ra-rhs-srv2-10g:/.rhs/RHOSglance/RHOSglance_brick
gluster> exit
```

Appendix B: NFS Storage Setup

This appendix describes the steps for creating an NFS server and using it to back the OpenStack Block Storage service. After completing these steps resume the reference architecture at **6 Validate the Installation**.

With this architecture the Storage network is removed. Clients access the NFS server directly via the Lab network.

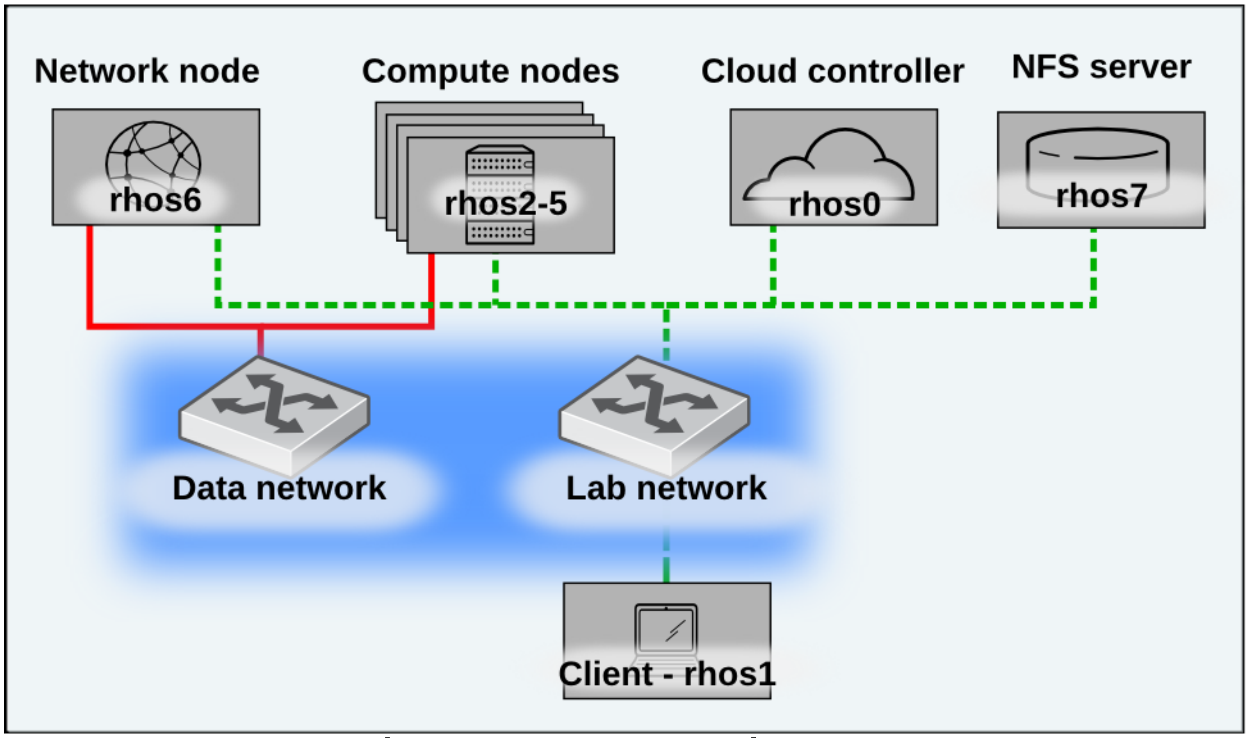


Figure B-1: NFS Architecture

B.1 Create the NFS Server

```
[root@rhos0 ~(keystone_admin)]# ssh -l root rhos7
Warning: Permanently added 'rhos7,10.16.137.107' (RSA) to the list of known
hosts.
root@rhos7's password: ********
Kickstarted on 2013-07-19

[root@rhos7 ~]# yum install -y -q nfs-utils

[root@rhos7 ~]# service nfs start
Starting NFS services:                                    [  OK  ]
Starting NFS quotas:                                      [  OK  ]
Starting NFS mountd:                                      [  OK  ]
Stopping RPC idmapd:                                      [  OK  ]
```

```
Starting RPC idmapd:                                        [  OK  ]
Starting NFS daemon:                                        [  OK  ]

[root@rhos7 ~]# chkconfig nfs on

[root@rhos7 ~]# mkdir /share

[root@rhos7 ~]# cp /etc/exports /etc/exports.orig

[root@rhos7 ~]# echo "/share *(rw,sync,no_root_squash)" > /etc/exports

[root@rhos7 ~]# exportfs -rav
exporting *:/share

[root@rhos7 ~]# iptables -I INPUT -p tcp --dport 2049 -j ACCEPT

[root@rhos7 ~]# service iptables save
iptables: Saving firewall rules to /etc/sysconfig/iptables:[  OK  ]

[root@rhos7 ~]# showmount -e localhost
Export list for localhost:
/share *
```

B.2 Add the NFS Server to Cinder

1. Configure Cinder to use the NFS share. Run these commands on the cloud controller.

```
[root@rhos0 ~]# openstack-config --set /etc/cinder/cinder.conf DEFAULT
nfs_shares_config /etc/cinder/shares.conf

[root@rhos0 ~]# openstack-config --set /etc/cinder/cinder.conf DEFAULT
volume_driver cinder.volume.drivers.nfs.NfsDriver

[root@rhos0 ~]# echo "rhos7:/share" > /etc/cinder/shares.conf

[root@rhos0 ~]# mkdir /var/lib/cinder/mnt

[root@rhos0 ~]# chown -v cinder.cinder /var/lib/cinder/mnt
changed ownership of `/var/lib/cinder/mnt' to cinder:cinder
```

2. Verify the changes.

```
[root@rhos0 ~]# cat /etc/cinder/shares.conf
rhos7:/share

[root@rhos0 ~]# grep -i nfs /etc/cinder/cinder.conf | grep -v \#
volume_driver = cinder.volume.nfs.NfsDriver
nfs_shares_config = /etc/cinder/shares.conf
```

3. Configure SELinux on the compute nodes to allow the virtual machines to use NFS.

```
[root@rhos0 ~]# i=2; while [ $i -lt 6 ]; do ssh rhos$i setsebool -P
virt_use_nfs on; ((i++)); done
```

```
[root@rhos0 ~]# i=2; while [ $i -lt 6 ]; do ssh rhos$i getsebool
virt_use_nfs; ((i++)); done
virt_use_nfs --> on
virt_use_nfs --> on
virt_use_nfs --> on
virt_use_nfs --> on
```

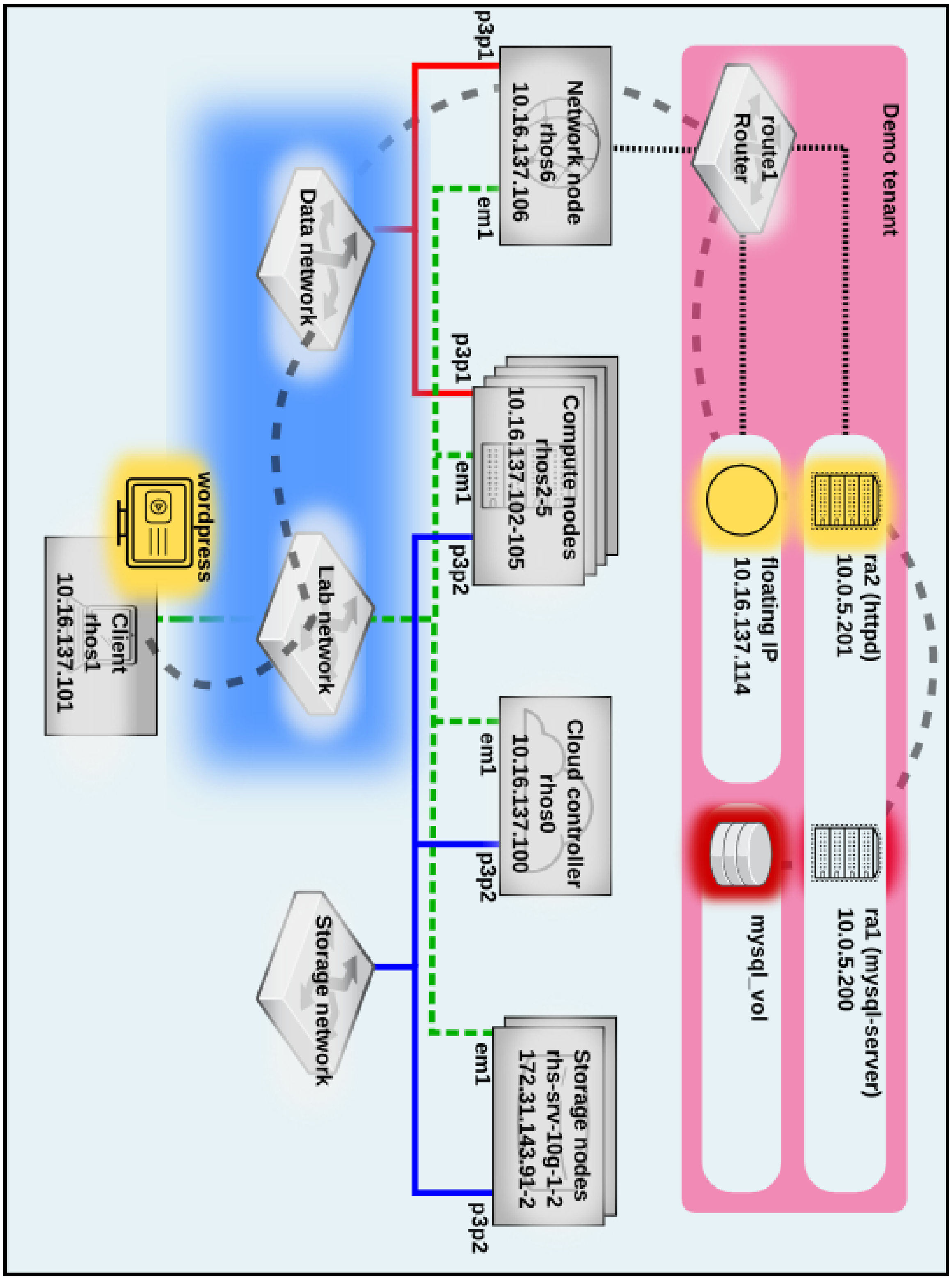
4. Restart the Cinder services and verify they are running.

```
[root@rhos0 ~]# for i in $(chkconfig --list | awk ' /cinder/ { print $1 } ');
do service $i restart; done
Stopping openstack-cinder-api:                             [  OK  ]
Starting openstack-cinder-api:                             [  OK  ]
Stopping openstack-cinder-scheduler:                       [  OK  ]
Starting openstack-cinder-scheduler:                       [  OK  ]
Stopping openstack-cinder-volume:                          [  OK  ]
Starting openstack-cinder-volume:                          [  OK  ]

[root@rhos0 ~]# for i in $(chkconfig --list | awk ' /cinder/ { print $1 } ');
do service $i status; done
openstack-cinder-api (pid  19386) is running...
openstack-cinder-scheduler (pid  19401) is running...
openstack-cinder-volume (pid  19416) is running...
```

5. Delete the **cinder-volumes** volume group on the cloud controller. The NFS server now provides the persistent volumes.

```
[root@rhos0 ~]# vgremove cinder-volumes
Volume group "cinder-volumes" successfully removed

[root@rhos0 ~]# vgs
VG    #PV #LV #SN Attr   VSize   VFree
myvg    1   1   0 wz--n- 134.94g    0
```

Appendix C: Full Network Diagram



Appendix D: Revision History

Revision 0.5 August 16, 2013 Jacob Liberman

- Vinny Valdez review
- Scott Collier review

Revision 0.4 August 12, 2013 Jacob Liberman

Third draft

- Steve Reichard review

Revision 0.3 August 2, 2013 Jacob Liberman

Second draft

- Validated steps, all content
- First review draft

Revision 0.2 July 19, 2013 Jacob Liberman

First draft

- Style template
- Graphics
- Appendices

Revision 0.1 May 12, 2013 Jacob Liberman

Initial Release

- Title
- Abstract
- Outline

Appendix E: Contributors

1. Steve Reichard, content and scripts, technical and content review
2. Scott Collier, technical review
3. Vinny Valdez, technical review
4. Robert Kukura, technical review, expert advisor (Neutron)