



**Red Hat Reference Architecture Series**

# **Deploying and Using Red Hat OpenStack 3.0**

**Jacob Liberman, Principal Software Engineer**  
**RHCE**

**Version 0.3**

**August 2, 2013**





1801 Varsity Drive™  
Raleigh NC 27606-2072 USA  
Phone: +1 919 754 3700  
Phone: 888 733 4281  
Fax: +1 919 754 3701  
PO Box 13588  
Research Triangle Park NC 27709 USA

Linux is a registered trademark of Linus Torvalds. Red Hat, Red Hat Enterprise Linux and the Red Hat "Shadowman" logo are registered trademarks of Red Hat, Inc. in the United States and other countries.

UNIX is a registered trademark of The Open Group.

Intel, the Intel logo and Xeon are registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

OpenStack is a registered trademark of the OpenStack Foundation.

All other trademarks referenced herein are the property of their respective owners.

© 2013 by Red Hat, Inc. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, V1.0 or later (the latest version is presently available at <http://www.opencontent.org/openpub/>).

The information contained herein is subject to change without notice. Red Hat, Inc. shall not be liable for technical or editorial errors or omissions contained herein.

Distribution of modified versions of this document is prohibited without the explicit permission of Red Hat Inc.

Distribution of this work or derivative of this work in any standard (paper) book form for commercial purposes is prohibited unless prior permission is obtained from Red Hat Inc.

The GPG fingerprint of the [security@redhat.com](mailto:security@redhat.com) key is:  
CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E

Send feedback to [refarch-feedback@redhat.com](mailto: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](mailto: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.....	1
2 Component Overview.....	2
2.1 Red Hat OpenStack.....	2
2.2 Red Hat OpenStack 3.0 (“Grizzly”) Services.....	2
2.2.1 Identity Service (“Keystone”).....	3
2.2.2 Image Service (“Glance”).....	3
2.2.3 Compute Service (“Nova”).....	3
2.2.4 Block Storage (“Cinder”).....	4
2.2.5 Network Service (“Neutron”).....	4
2.2.6 Dashboard (“Horizon”).....	5
2.2.7 Services Not Covered in this Reference Architecture.....	6
2.3 Red Hat Enterprise Linux.....	6
2.4 Supporting Technologies.....	6
2.4.1 MySQL.....	7
2.4.2 QPID.....	7
2.4.3 KVM.....	7
2.4.4 Red Hat Storage Server (RHSS).....	7
2.4.5 Packstack.....	8
2.4.6 Supporting Technologies Not Used in this Reference Architecture.....	8
3 Reference Architecture Configuration Details.....	9
3.1 Environment.....	9
3.1.1 Network Topology.....	9
3.1.2 IP Addresses.....	10
3.2 Software and Security Reference.....	11
3.2.1 Software Versions.....	11
3.2.2 Security: iptables.....	13
3.2.3 Security: SELinux.....	13
3.2.4 Required Channels.....	13
3.3 Server Hardware Configuration.....	14
3.4 OpenStack Service Placement.....	15
4 Deploy Red Hat OpenStack 3.0 via Packstack.....	16
4.1 Architectural Overview.....	16
4.1.1 Cloud Controller.....	16



4.1.2 Compute Node.....	17
4.1.3 Network Node.....	17
4.1.4 Storage Server.....	17
4.1.5 Client.....	17
4.2 Prepare the Hosts.....	17
4.2.1 Install the Operating System.....	18
4.2.2 Register with Red Hat Network.....	18
4.2.3 Configure Name Resolution.....	18
4.2.4 Create an SSH Key on the Cloud Controller.....	19
4.3 Deploy via Packstack.....	21
4.3.1 Prepare the Cloud Controller for Packstack.....	21
4.3.2 Run Packstack on the Cloud Controller.....	22
4.3.3 Verify the configuration. ....	24
4.3.4 Verify Packstack Installation Completes Successfully.....	25
4.3.5 Perform Post-Packstack Customization.....	26
4.3.6 Reboot the Servers.....	27
4.4 Verify the Packstack Deployment.....	27
4.4.1 Initial Deployment Overview.....	28
4.4.2 Examine Deployment as Keystone Admin.....	28
5 Deploy Red Hat Storage Server.....	31
5.1 Build a Red Hat Storage Server.....	31
5.2 Configure the Red Hat Storage Clients.....	32
5.3 Add RHS to Glance and Cinder.....	36
6 Validate the Installation.....	39
6.1 Create the Demo Tenant, User, and Role.....	39
6.1.1 Import a Virtual Machine Disk Image.....	39
6.1.2 Create a Tenant.....	40
6.1.3 Add a User.....	41
6.1.4 Associate the User, Role, and Tenant.....	41
6.2 Configure the Network Server.....	42
6.2.1 Capture demo-tenant ID.....	42
6.2.2 Create a Network.....	42
6.2.3 Create a Subnet.....	43
6.3 Bring up Bridge Interface on All Compute and Network Nodes.....	44
6.4 Boot an Instance in the Tenant.....	45
6.4.1 Switch to demo User.....	45



6.4.2 Create Default Security Group Rules.....	48
6.5 Configure a Router.....	50
6.5.1 Prepare the Environment.....	50
6.5.2 Create the Router.....	51
6.5.3 Ping the Router's External Interface.....	52
6.5.4 Validated Installation.....	53
7 Deploy a Multi-tier Web Application.....	54
7.1 Overview.....	54
7.2 Deploy the Database Server.....	54
7.2.1 Prepare the Environment.....	55
7.2.2 Create the User Data Script.....	55
7.2.3 Create the Registration Script.....	55
7.2.4 Create the Database Volume.....	55
7.2.5 Boot the Database Instance .....	56
7.2.6 Verify the Database Server.....	58
7.3 Deploy the Web server.....	58
7.3.1 Create the User Data Script.....	59
7.3.2 Launch the Web Server.....	59
7.3.3 Associate a Public IP Address with the Web Server Instance.....	60
7.3.4 Verify the Web Server Installation.....	61
7.4 Test the Web Server from a Client.....	62
7.5 Complete Multi-tier Application.....	63
8 Conclusion.....	64
Appendix A: Red Hat Storage Setup.....	65
A.1 Overview.....	65
A.2 Prepare the RHS Server Nodes.....	65
A.2.1 Create a Virtual Disk (LUN) with Raid 6 using PERC Controller.....	65
A.2.2 Install RHS Server.....	68
A.3 Register and Update Storage Nodes.....	74
A.4 Tune Kernel with Tuned .....	75
A.5 Set up Trusted Storage Pools.....	76
A.6 Create and Mount XFS filesystems for Bricks.....	76
A.7 Create Replica Volumes.....	79
Appendix B: NFS Storage Setup.....	81



B.1 Create the NFS Server.....	81
B.2 Add the NFS Server to Cinder.....	81
Appendix C: Revision History.....	84
Appendix D: Contributors.....	85



# 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 OpenStack (RHOS) 3.0 addresses these challenges. Red Hat OpenStack 3.0 -- Red Hat's second official OpenStack release -- delivers a stable code base for production deployments backed by Red Hat's open source software expertise. Red Hat OpenStack 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 OpenStack 3.0 through a detailed use case:

- Installing Red Hat OpenStack 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, it is not required for a Red Hat OpenStack deployment. This document also includes instructions for NFS-backed persistent storage.





## 2 Component Overview

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

### 2.1 Red Hat OpenStack

Red Hat OpenStack 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.

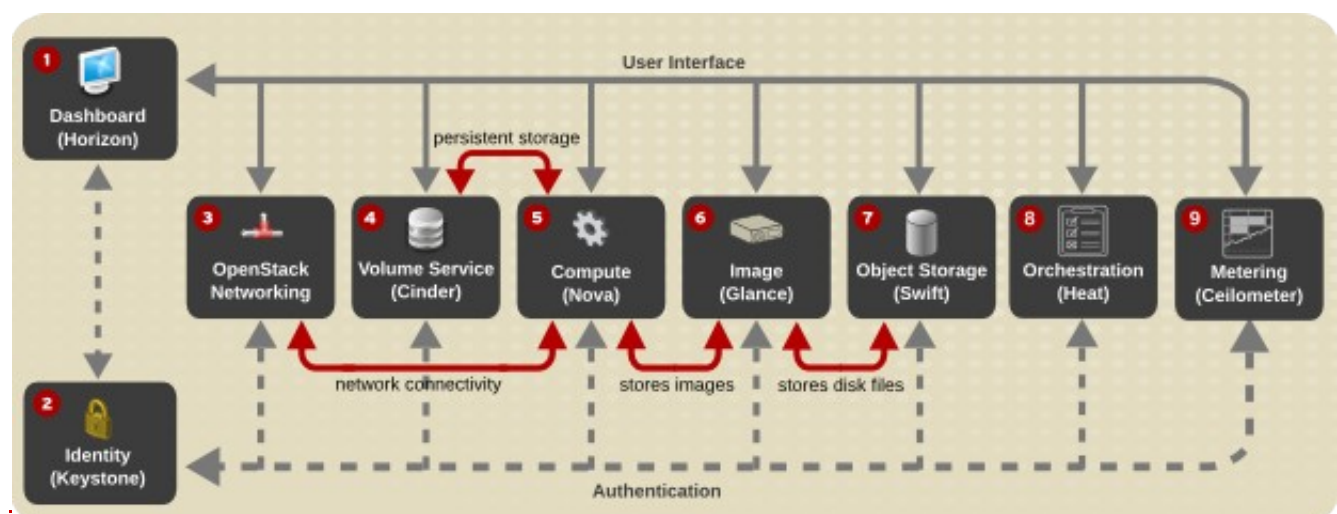
Red Hat OpenStack gives organizations a truly open framework for hosting cloud workloads, delivered by Red Hat subscription for maximum flexibility and cost effectiveness. In conjunction with other Red Hat technologies, Red Hat OpenStack allows organizations to move from traditional workloads to cloud-enabled 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 OpenStack 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 OpenStack 3.0 (“Grizzly”) Services

Red Hat OpenStack 3.0 is based on the upstream “Grizzly” OpenStack release. Red Hat OpenStack 3.0 is Red Hat’s second release. The first release was based on the “Folsom” OpenStack release. Folsom was the first release to include extensible block and volume storage capabilities. 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 enforces tenants by issuing 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 REST 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. The Glance API can also be configured to cache images locally on the hosts that use them 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, Red Hat Storage Server (RHSS), and RBD.

The Image service is composed of the **openstack-glance-api** that delivers image information from the registry service, and the **openstack-glance-registry** 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
- Security groups to flexibly assign instances to resource pools and control access to them
- 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 provisions 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.

## 2.2.5 Network Service (“Neutron”)

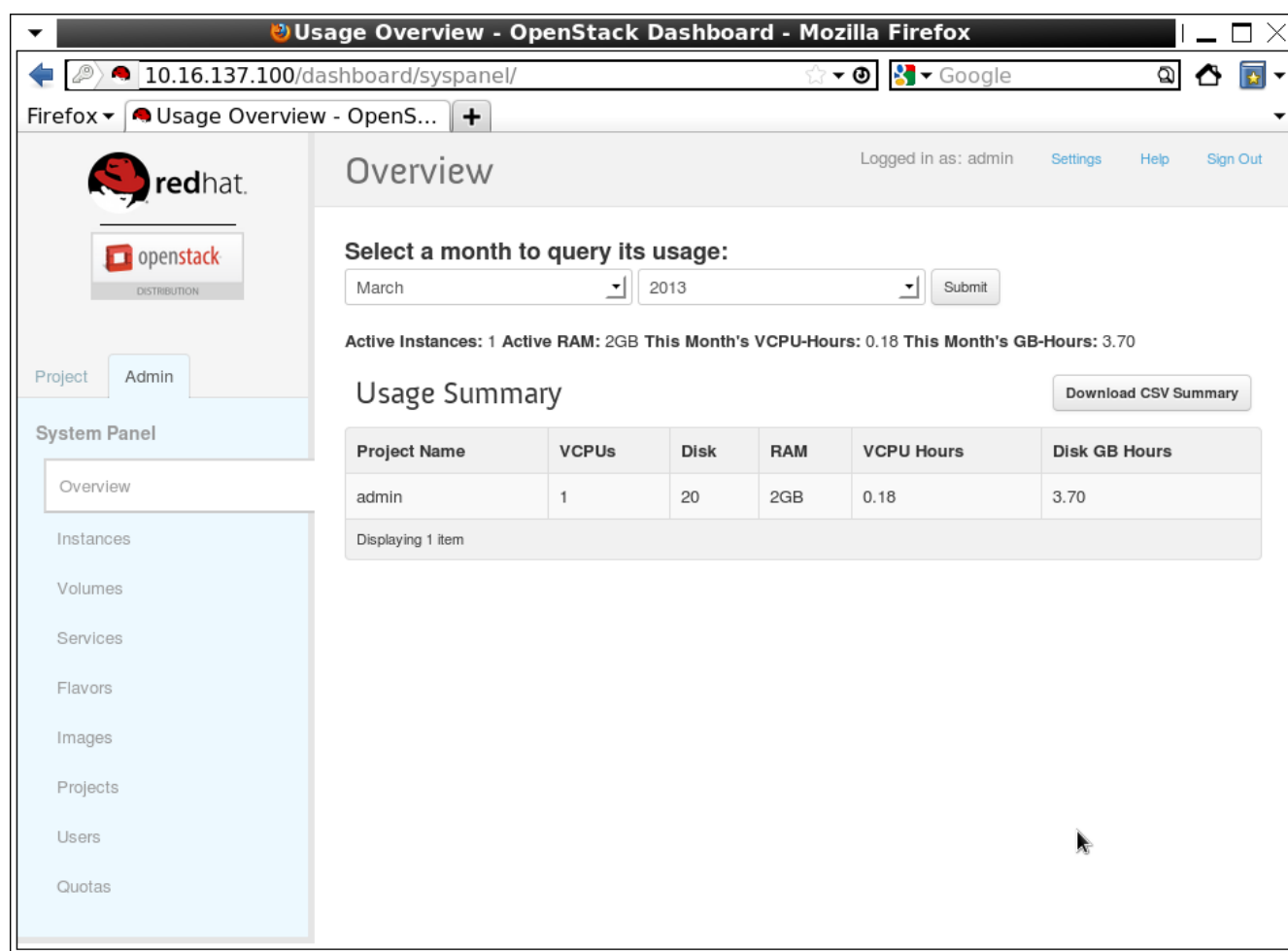
OpenStack Networking is a scalable API-driven service for managing networks and IP addresses. OpenStack Networking gives users self-service control over their network configurations. Users can define, separate, and join networks on demand. This allows for flexible network models that can be adapted to 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 as **openvswitch** and **linuxbridge**.



OpenStack Networking is composed of several services. The **quantum-server** exposes the API and responds to user requests. It also The **quantum-l3-agent** provides L3 functionality such as routing. 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”)



**Figure 1: Dashboard System Panel**

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. **Figure 1: Dashboard System Panel** depicts the Horizon Dashboard Overview screen.



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

**NOTE:** Both the Dashboard and command line tools can be 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 OpenStack 3.0, 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 OpenStack 3.0 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 OpenStack 3.0 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 OpenStack 3.0 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 OpenStack 3.0 uses QPID for open source enterprise messaging. QPID is an AMQP compliant, cross-platform enterprise messaging system developed for low latency based on Advanced Message Queuing Protocol (AMQP) -- an open standard for enterprise messaging. QPID is released under the Apache open source license and ships with Red Hat Enterprise Linux 6.4.

## 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. OpenStack Compute uses KVM as its underlying hypervisor to launch and control virtual machine instances.

## 2.4.4 Red Hat Storage Server (RHSS)

Red Hat Storage Server 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. In this reference architecture RHS is the back end storage for both the Block and Image Services. The RHS Fuse driver enables block storage support while Gluster volumes allow object storage for





virtual machine images. Red Hat OpenStack 3.0 does not depend on Red Hat Storage Server.

## 2.4.5 Packstack

Packstack is a Red Hat OpenStack 3.0 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. 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 OpenStack 3.0 does not depend on Red Hat Enterprise Virtualization. Although it is possible to combine Red Hat Enterprise Virtualization and Red Hat OpenStack 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.

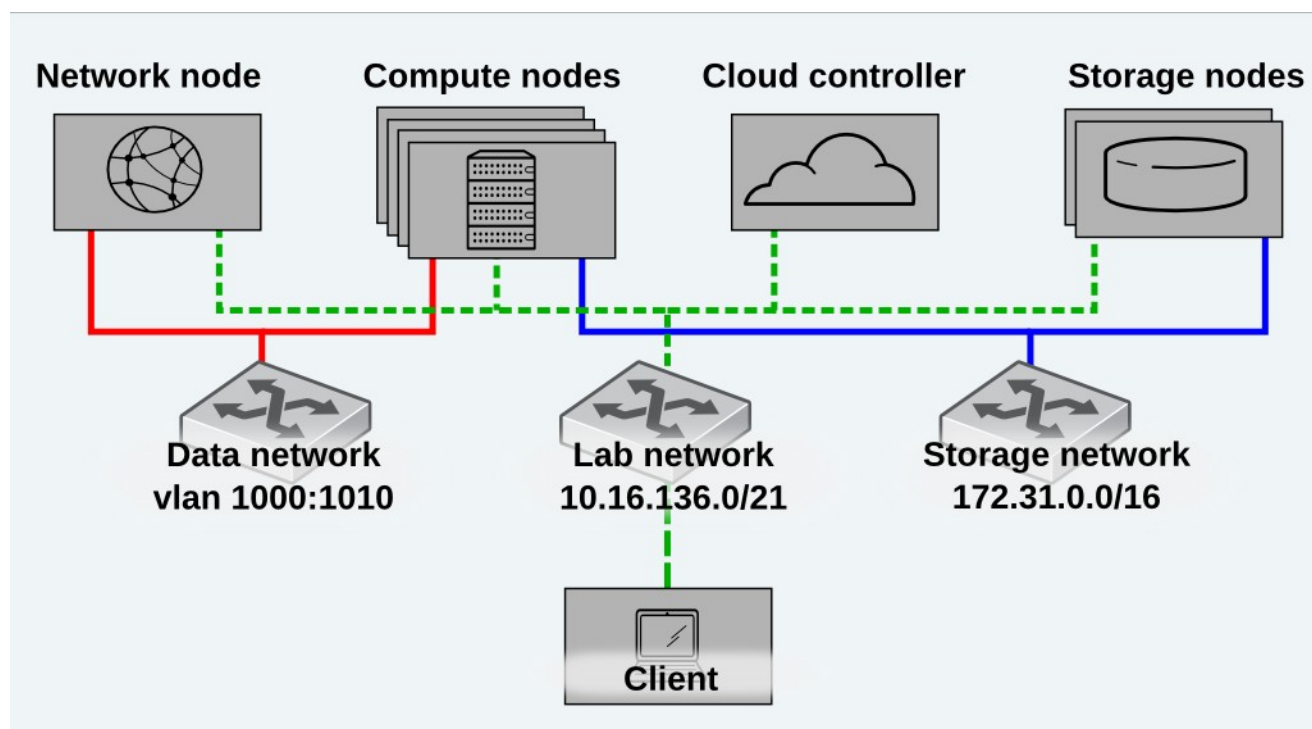
### 3.1 Environment

The reference architecture environment consists of the components required to build a small Red Hat OpenStack 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.

- 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 a series of tagged VLANs. The interfaces connected to this network are not given IPv4 addresses. Instead they act as bridges to software-defined network devices that exist in dedicated network namespaces.

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
rhos2	Compute	Storage	p3p2	172.31.139.102
rhos2	Compute	Data	p3p1	VLAN 1000:1010
rhos3	Compute	Lab	em1	10.16.137.103
rhos3	Compute	Storage	p3p2	172.31.139.103
rhos3	Compute	Data	p3p1	VLAN 1000:1010
rhos4	Compute	Lab	em1	10.16.137.104
rhos4	Compute	Storage	p3p2	172.31.139.104
rhos4	Compute	Data	p3p1	VLAN 1000:1010
rhos5	Compute	Lab	em1	10.16.137.105
rhos5	Compute	Storage	p3p2	172.31.139.105
rhos5	Compute	Data	p3p1	VLAN 1000:1010
rhos6	Network	Lab	em1	10.16.137.106
rhos6	Network	Data	p3p1	VLAN 1000:1010
ra-rhs-srv-10g-1	Storage	Storage	p3p2	172.31.143.91
ra-rhs-srv-10g-2	Storage	Storage	p3p2	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.

### 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	qpidd-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

**Table 3.2.1.1: Software Versions**

### 3.2.2 Security: iptables

Deploying Red Hat OpenStack 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, 38465:38469	rhos0	Cloud controller
22, 80	rhos1	Client
22, 53, 67, 5900:5950	rhos2-5	Compute nodes
22	rhos6	Network server
22, 111, 24007:24025	ra-rhs-srv{1,2}	Storage server

**Table 3.2.2.1: Allowed iptables Ports by Role**

### 3.2.3 Security: SELinux

Red Hat OpenStack 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.

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**

### 3.2.4 Required Channels

Red Hat OpenStack 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-rhscclient-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.

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

Hardware	Specifications
7 x DELL BladeServer – PowerEdge M520	2 x Intel(R) Xeon(R) CPU E5-2450 0 @ 2.10GHz GB (8 core)
	2 x Broadcom NetXtreme II BCM57810 10 Gigabit Ethernet 2 x Broadcom NetXtreme BCM5720 Gigabit Ethernet
	8 x DDR3 4096 MB @1600 MHZ DIMMs
	2 x 146GB 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 majority run on the cloud controller.

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

**Table 3.4.1: Service Placement**

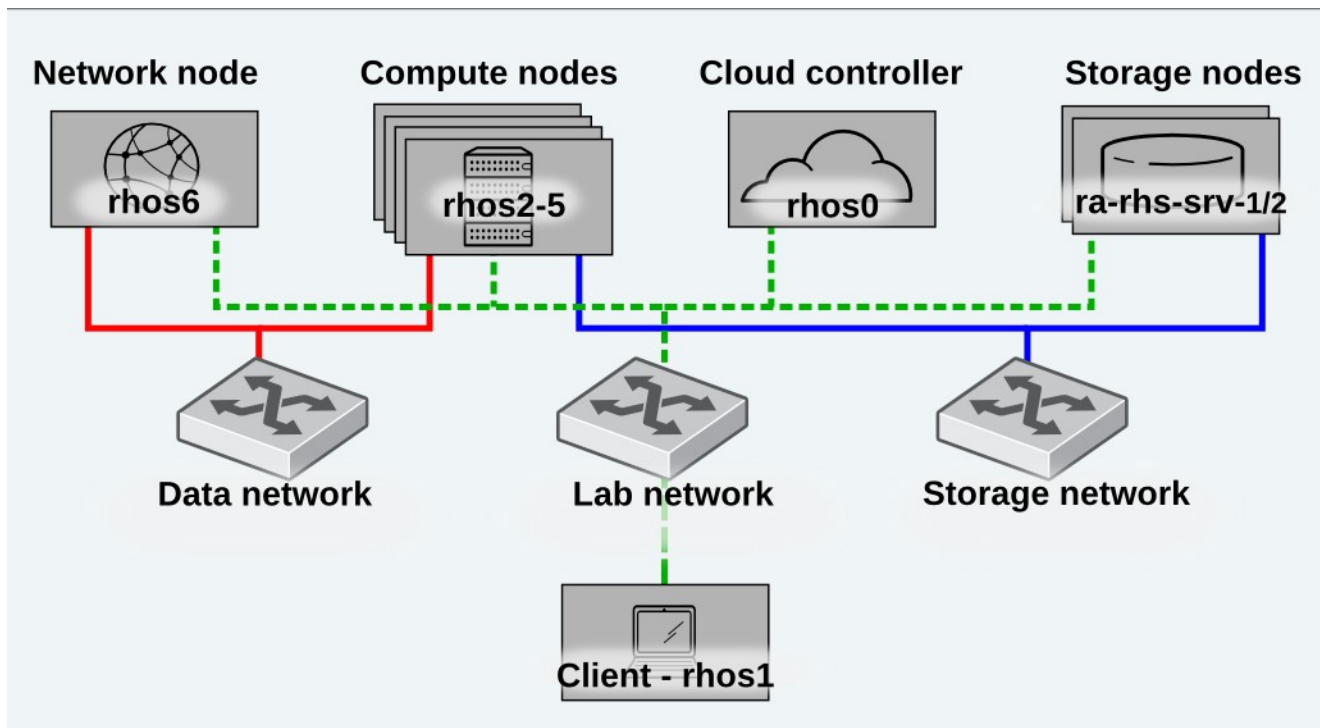


## 4 Deploy Red Hat OpenStack 3.0 via Packstack

This section of the paper describes the steps required to install Red Hat OpenStack 3.0 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.

### 4.1 Architectural Overview

**Illustration 4.1.1 Architectural Overview** shows the Red Hat OpenStack server roles by host.



**Illustration 4.1.1: Architectural Overview**

In this reference architecture Red Hat OpenStack 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, Identity, and Network services. The cloud controller also updates the state database and runs the messaging system.

Due to 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 responsible for creating virtual machine instances. These worker nodes field requests from the message queue, perform a series of commands, and then update the state database. The **openstack-nova-scheduler** service will load balance instance deployments across compute nodes. By default it will assign new instances to the nodes with the most free memory in a round robin fashion.

## 4.1.3 Network Node

In the Red Hat OpenStack 2.1 reference architecture networking was provided by the **openstack-nova-network** service. This service is a sub-service of OpenStack Compute that controls basic networking functionality. Each compute node ran a copy of the network service in multi host mode for redundancy and localization.

In the Red Hat OpenStack 3.0 reference architecture, OpenStack Compute networking has been replaced with OpenStack Networking. 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 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.

## 4.2 Prepare the Hosts

Perform the following steps on the cloud controller prior to deploying Red Hat OpenStack 3.0:

1. Install the operating system
2. Register system with RHN





3. Configure networking
4. Configure name resolution
5. Configure the firewall
6. Configure SSH keys

## 4.2.1 Install the Operating System

Red Hat OpenStack 3.0 requires Red Hat Enterprise Linux 6.4 with Eratta or higher.

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

[root@rhos0 ~]# uname -a
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 Register with Red Hat Network

Register the cloud controller with the Red Hat Network.

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

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

**NOTE:** It is not necessary to register the cloud controller with both Red Hat Network and Red Hat Certificate Server. This document shows how to do both for completeness.

## 4.2.3 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.

```
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
172.31.139.101 rhos1-stor
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
```

## 4.2.4 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 will prompt 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**=+          |
|      . =*+          |
|      ..0.           |
|      .0 .           |
|      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 script whose answers define the Red Hat OpenStack 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**.

### 4.3.1 Prepare the Cloud Controller for Packstack

Red Hat OpenStack and **packstack** are distributed through RHN and Subscription Manager.

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

```
[root@rhos0 ~]# rhn-channel -u sreichar -p rhnpasswd -a -c rhel-x86_64-server-6-ost-3
```

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

Alternately, add the OpenStack subscription with the **subscription-manager** command.

```
[root@rhos0 ~]# subscription-manager register --username sreichar --password rhnpasswd --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 OpenStack 3.0 (RPMs)          590
```

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

```
[root@rhos0 ~]# yum install -y -q 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 **4.2.4 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. It will be deleted in subsequent steps.

```
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. Enter the CPU and RAM overcommitment ratios.

```
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. Enter the IP address of the Network server.

```
Enter the IP address of the Quantum server [10.16.137.100] : 10.16.137.106  
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. Enter the external bridge name. The external bridge passes instance traffic to an external network interface on the Compute node. This bridge will be removed in subsequent steps and a custom bridge will be created.

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

12. The DHCP agent should also be on the Network server.

```
Enter a comma separated list of IP addresses on which to install Quantum  
DHCP agent [10.16.137.100] : 10.16.137.106  
Enter the name of the L2 plugin to be used with Quantum [linuxbridge|  
openvswitch] [openvswitch] :
```

13. Install the metadata agent on the Network server. The metadata agent listens for boot-time customization requests from installing instances 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
```

14. Allocate local networks for tenant networks.

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

15. Assign a VLAN range for the **openvswitch** plugin. This range must be configured on the physical switches that will carry the OpenStack Network service traffic.

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

16. Enter the bridge mapping for the **openvswitch** plugin.

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

17. Accept the default for bridge:interface pairs.

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

18. Install the client tools and Dashboard web interface on the client system.

```
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] :
```

19. Enter the update account information. It will be propagated to the servers. 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 OpenStack 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: *****
qpidd-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: 10.16.137.106
quantum-use-namespaces: y
quantum-l3-hosts: 10.16.137.106
quantum-l3-ext-bridge: removeme
quantum-dhcp-hosts: 10.16.137.106
quantum-l2-plugin: openvswitch
quantum-metadata-hosts: 10.16.137.106
quantum-ovs-tenant-network-type:local
quantum-ovs-vlan-ranges: physnet1:1000:1010
quantum-ovs-bridge-mappings: physnet1:br-p3p1
quantum-ovs-bridge-interfaces:
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 Perform Post-Packstack Customization

First, remove the external bridge from the Network server.

1. Connect to the Network server via SSH.

```
[root@rhos0 ~]# ssh -l root rhos6
```

2. View the *removeme* bridge.

```
[root@rhos6 ~]# grep removeme /etc/quantum/l3_agent.ini
external_network_bridge = removeme
```

```
[root@rhos6 ~]# ovs-vsctl list-br
br-int
br-p3p1
removeme
```

3. Remove the bridge from the configuration file with **sed**.

```
[root@rhos6 ~]# sed -i.orig 's/external_network_bridge =
removeme/external_network_bridge =/' /etc/quantum/l3_agent.ini
```



4. Delete the existing bridge with **ovs-vsctl**.

```
[root@rhos6 ~]# ovs-vsctl del-br removeme
```

5. Restart the network services.

```
[root@rhos6 ~]# for i in $(chkconfig --list | grep 3:on | awk ' /quantum/ { print $1 } ' ); do service $i restart; done
Stopping quantum-dhcp-agent: [ OK ]
Starting quantum-dhcp-agent: [ OK ]
Stopping quantum-l3-agent: [ OK ]
Starting quantum-l3-agent: [ OK ]
Stopping quantum-metadata-agent: [ OK ]
Starting quantum-metadata-agent: [ OK ]
Stopping quantum-openvswitch-agent: [ OK ]
Starting quantum-openvswitch-agent: [ OK ]
Stopping quantum-server: [ OK ]
Starting quantum-server: [ OK ]
```

6. Verify the bridge has been delete from the openvswitch configuration and all references to the bridge are removed from the L3 agent configuration file.

```
[root@rhos6 ~]# ovs-vsctl list-br
br-int
br-p3p1
[root@rhos6 ~]# grep removeme /etc/quantum/l3_agent.ini
```

## 4.3.6 Reboot the Servers

1. Reboot the OpenStack servers to boot 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
```

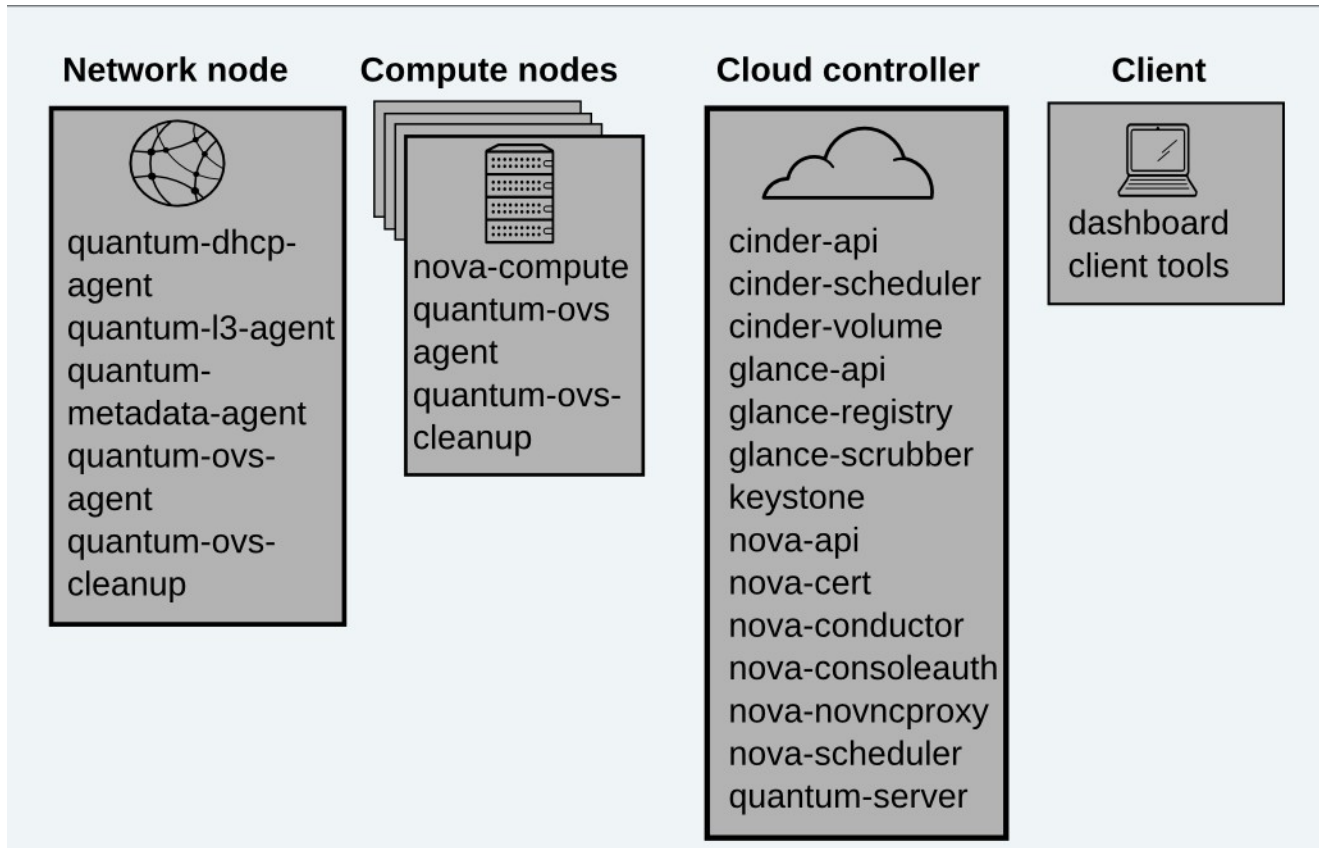
## 4.4 Verify the Packstack Deployment

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



## 4.4.1 Initial Deployment Overview

**Error: Reference source not found** shows the initial **packstack** deployment with services and networks. The **nova-compute** service is running on the compute nodes. The Horizon dashboard and client tools are installed on the client. 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 -l; 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 keystoneadmin not sourced
```

**NOTE:** By default **packstack** installs the *keystoneadmin* file on the client. Copy this file to the cloud controller and source it before running **openstack-status** to display additional information.

4. Connect to the client via SSH and source the *keystoneadmin* 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/keystoneadmin

[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



## 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.

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

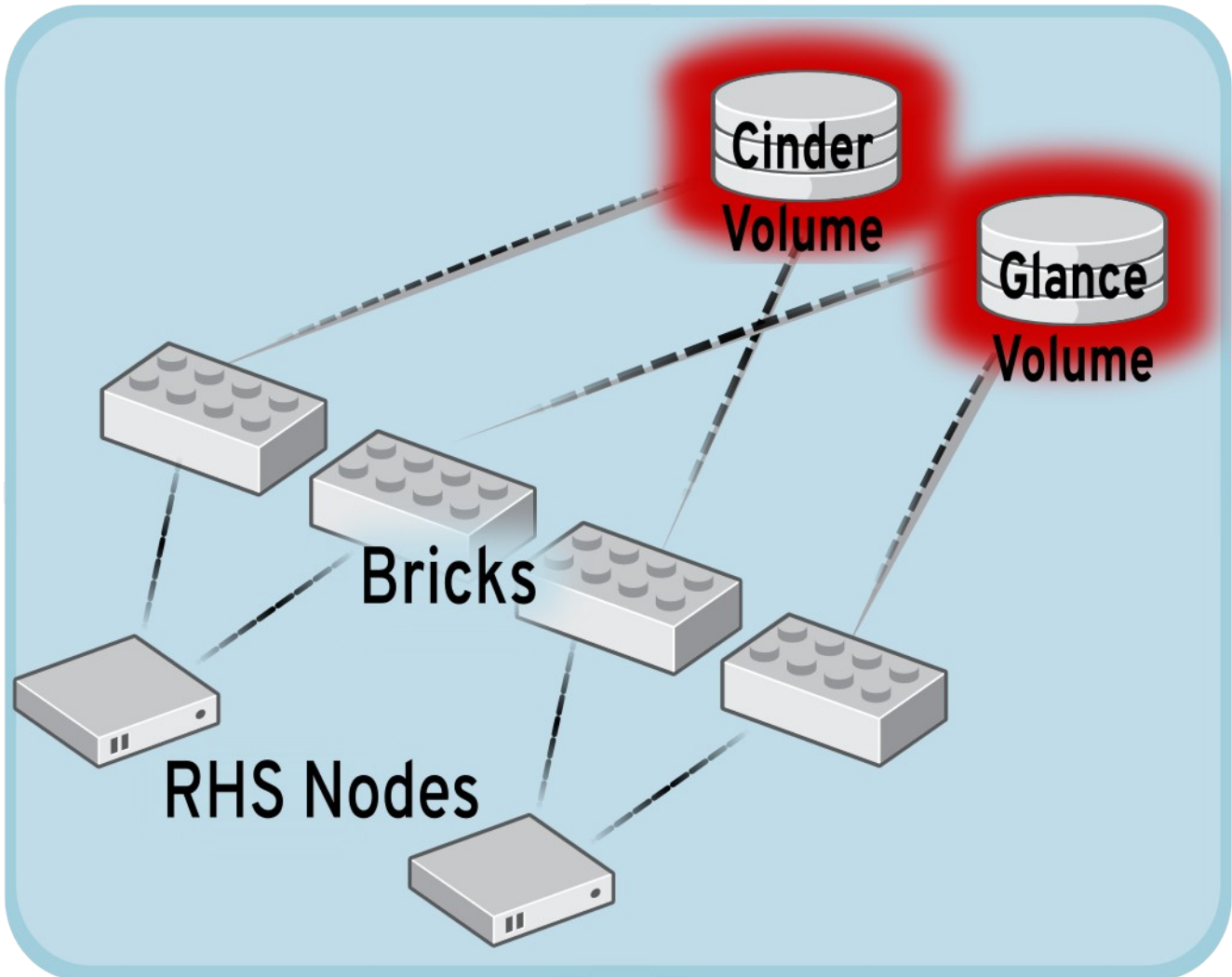
### 5.1 Build a Red Hat Storage Server

The RHS Server is composed of two servers. Each server exports 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 are still available via the remaining node.

**Illustration 5.1.1: Red Hat Storage Server** shows the storage architecture presented to the Red Hat OpenStack 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 Server*

## 5.2 Configure the Red Hat Storage Clients

1. Open the RHS-required ports on the cloud controller.

```
[root@rhos0 ~]# iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 111 -j ACCEPT
[root@rhos0 ~]# iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 24007 -j ACCEPT
[root@rhos0 ~]# iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 24008 -j ACCEPT
[root@rhos0 ~]# iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 24009 -j ACCEPT
[root@rhos0 ~]# iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 24010 -j ACCEPT
[root@rhos0 ~]# iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 24011 -j ACCEPT
```





```
[root@rhos0 ~]# iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 38465:38469 -j ACCEPT
```

```
[root@rhos0 ~]# service iptables save
iptables: Saving firewall rules to /etc/sysconfig/iptables:[ OK ]
```

2. 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 -u admin -p 100yard- -a -c rhel-x86_64-server-rhsclient-6; ((i++)); done
```

```
[root@rhos0 ~]# i=0; while [ $i -lt 7 ]; do ssh rhos$i rhn-channel -l; ((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
```

3. 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 -y -q glusterfs-fuse.x86_64 tuned"; done
```

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

4. Verify the packages installed successfully.

```
[root@rhos0 ~]# for i in 0 2 3 4 5; do ssh rhos$i "rpm -qa | grep -E '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
```

5. Create a Object Storage server mount point on all compute nodes.

```
[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i "mkdir -pv /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'
```

6. Add the Object 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 '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
```

7. 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)
```

8. 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 ]

[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 Object Storage server to use the Identity Service for authentication and allow direct URL naming 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/glance/glance-api.conf DEFAULT
flavor keystone

[root@rhos0 ~]# grep ^flavor /etc/glance/glance-api.conf
flavor = keystone
flavor=keystone

[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 share as a backup for the first.

```
[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 -pv /var/lib/cinder/mnt
mkdir: created directory `/var/lib/cinder/mnt'

[root@rhos0 ~]# chown -v 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
ra-rhs-srv2-10g:/RHOScinder

[root@rhos0 ~]# grep -i gluster /etc/cinder/cinder.conf | grep -v \#
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
```

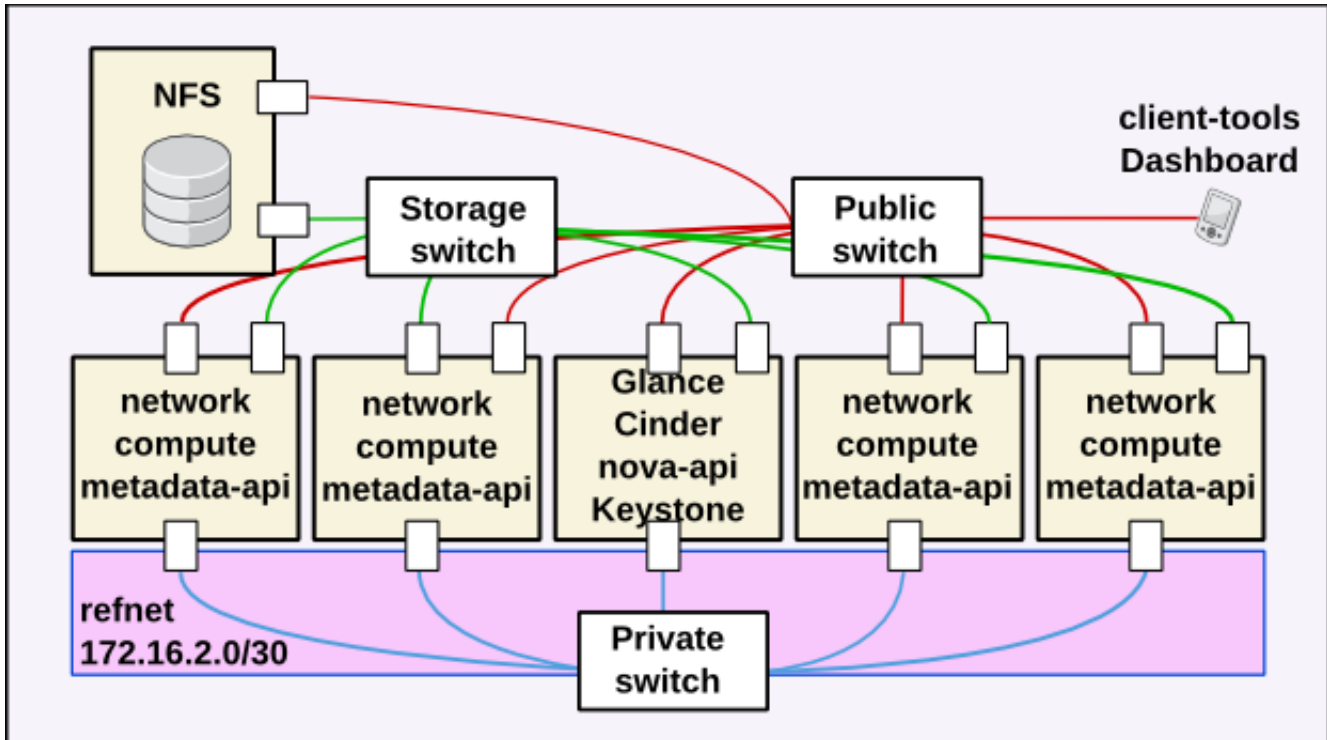


Figure 3: Complete OpenStack Deployment



## 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 Keystone 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 will be



used to deploy virtual machines.

```
[root@rhos1 ~(keystone_admin)]# glance image-create --name rhel-server2 --is-public true --disk-format qcow2 --container-format bare </pre>
</div>


```
/pub/tmp/jliberma/rhos/save/rhel-server-x86_64-kvm-6.4_20130130.0-2-sda.qcow2
```



| Property         | Value                                |
|------------------|--------------------------------------|
| checksum         | 5bacdb8ff85d25dbd1ae3db7b0bf57f0     |
| container_format | bare                                 |
| created_at       | 2013-07-23T04:55:19                  |
| deleted          | False                                |
| deleted_at       | None                                 |
| disk_format      | qcow2                                |
| id               | cb80a3c7-fa17-45d5-aa00-2618c18fb7ec |
| is_public        | True                                 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | rhel-server2                         |
| owner            | 988a0c24ff4f4d7ea2842ed642324f1b     |
| protected        | False                                |
| size             | 1974140928                           |
| status           | active                               |
| updated_at       | 2013-07-23T04:55:37                  |



NOTE: <SOURCE OF IMAGE >



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



```
[root@rhos1 ~(keystone_admin)]# glance image-list
```



| ID                                   | Name         | Disk Format |
|--------------------------------------|--------------|-------------|
| cb80a3c7-fa17-45d5-aa00-2618c18fb7ec | rhel-server2 | qcow2       |
| 1974140928                           | active       | bare        |



## 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.



www.redhat.com



40



refarch-feedback@redhat.com


```



```
[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 in *demo-tenant* 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*.

### 6.2.1 Capture demo-tenant ID

1. Capture the *demo-tenant* id to an environment variable for use in subsequent steps.

```
[root@rhos1 ~(keystone_admin)]# demo_id=$(keystone tenant-get demo-tenant |  
awk ' /id/ { print $4 } ' )
```

```
[root@rhos1 ~(keystone_admin)]# echo -e "$demo_id"  
ba9579abd57b41879ce62af9785298f4
```

### 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 will be available to instances booted in *demo-tenant*.

```
[root@rhos1 ~(keystone_admin)]# quantum net-create --tenant-id $demo_id 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_admin)]# 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 a V4 IP address range of 10.0.5.0/24.

```
[root@rhos1 ~(keystone_admin)]# quantum subnet-create --tenant-id $demo_id --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_admin)]# 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_admin)]# 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_admin)]# demo_net=$(quantum net-list | awk ' /net1/ {
print $2 } ')
```

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

## 6.3 Bring up Bridge Interface on All Compute and Network Nodes

The software defined bridge interface is associated with a physical network interface in the compute and network nodes. In this reference architecture physical interface **p3p1** is bridged to the software defined networks and carries the network traffic. In this section the bridge interface is configured on all compute and network nodes.

**NOTE:** Issue the commands in this section from the cloud controller.

1. Add the **p3p1** interface to the **br-p3p1** bridge on all compute and network nodes.

```
[root@rhos0 ~]# for i in 2 3 4 5 6
do
    ssh rhos$i ovs-vsctl add-port br-p3p1 p3p1
    ssh rhos$i ovs-vsctl list port | grep \"p3p1\"
done
name           : \"p3p1\"
name           : \"p3p1\"
name           : \"p3p1\"
name           : \"p3p1\"
name           : \"p3p1\"
```

2. “Up” the physical interface link and verify that it is UP on all compute and network nodes.

```
[root@rhos0 ~]# for i in 2 3 4 5 6
do
    ssh rhos$i ip link set p3p1 up
    ssh rhos$i ip addr show dev 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
```



1000

## 6.4 Boot an Instance in the Tenant

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

### 6.4.1 Switch to demo User

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*.

```
[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
```

4. Create a key pair as *demo* user. This key pair will be 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
```

5. Boot four instances as *demo* specifying the new key and network.

```
[root@rhos1 ~(demo_member)]# nova boot --flavor 2 --image rhel-server2 --key-name demokp --nic net-id=$demo_net 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	[{'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 --nic net-id=$demo_net 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	[{'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 --nic net-id=$demo_net 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 --nic net-id=$demo_net 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	

2. Verify the new instances are in an **ACTIVE** state.



```
[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

## 6.4.2 Create Default Security Group Rules

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. In Red Hat OpenStack 3.0 the default security groups are very restrictive. In this section the default security group for *demo-tenant* will be 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": null, "port_range_min": null, "ethertype": null, "id": null }





```

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
+-----+

```

## 6.5 Configure a Router

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.

### 6.5.1 Prepare the Environment

These steps should be performed as the **keystone\_admin** user on the client.

1. Connect to the client via SSH.

```

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

```

2. Source **keystonerc\_admin**.

```

[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
```

## 6.5.2 Create the Router

1. The **service** tenant is reserved for services and endpoints. The router – which may service several tenants – should be created in the **service** tenant. Find the Identity Service id of the **service** tenant and save it to an environment variable.

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

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

[root@rhos1 ~(keystone_admin)]# echo -e "$service_id"
eac6d62ba1c6466bad44ce8df05efcc4
```

2. Create a router named **route1** in the **service** tenant.

```
[root@rhos1 ~(keystone_admin)]# quantum router-create --tenant-id $service_id
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	eac6d62ba1c6466bad44ce8df05efcc4

3. Display **route1** configuration details.

```
[root@rhos1 ~(keystone_admin)]# quantum router-show route1
```

Field	Value
admin_state_up	True
external_gateway_info	
id	b334a0b9-3001-4cea-a00e-053f3f6aaede
name	route1
routes	
status	ACTIVE
tenant_id	eac6d62ba1c6466bad44ce8df05efcc4



```
+-----+-----+-----+
```

4. Find the **private** subnet's Identity Service id.

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

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

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

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

### 6.5.3 Ping the Router's External Interface

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

```
[root@rhos1 ~(keystone_admin)]# route_id=$(quantum router-list | awk '
/router1/ { print $2 } ')
```

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

2. Find the IP address of the router's external interface and set it to an environment variable.

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

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

3. Find the network name space of the router on the network node.

```
[root@rhos1 ~(keystone_admin)]# qroute_id=$(ssh rhos6 ip netns list | grep
qrouter)
```

```
root@rhos6's password: *****
```

```
[root@rhos1 ~(keystone_admin)]# 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



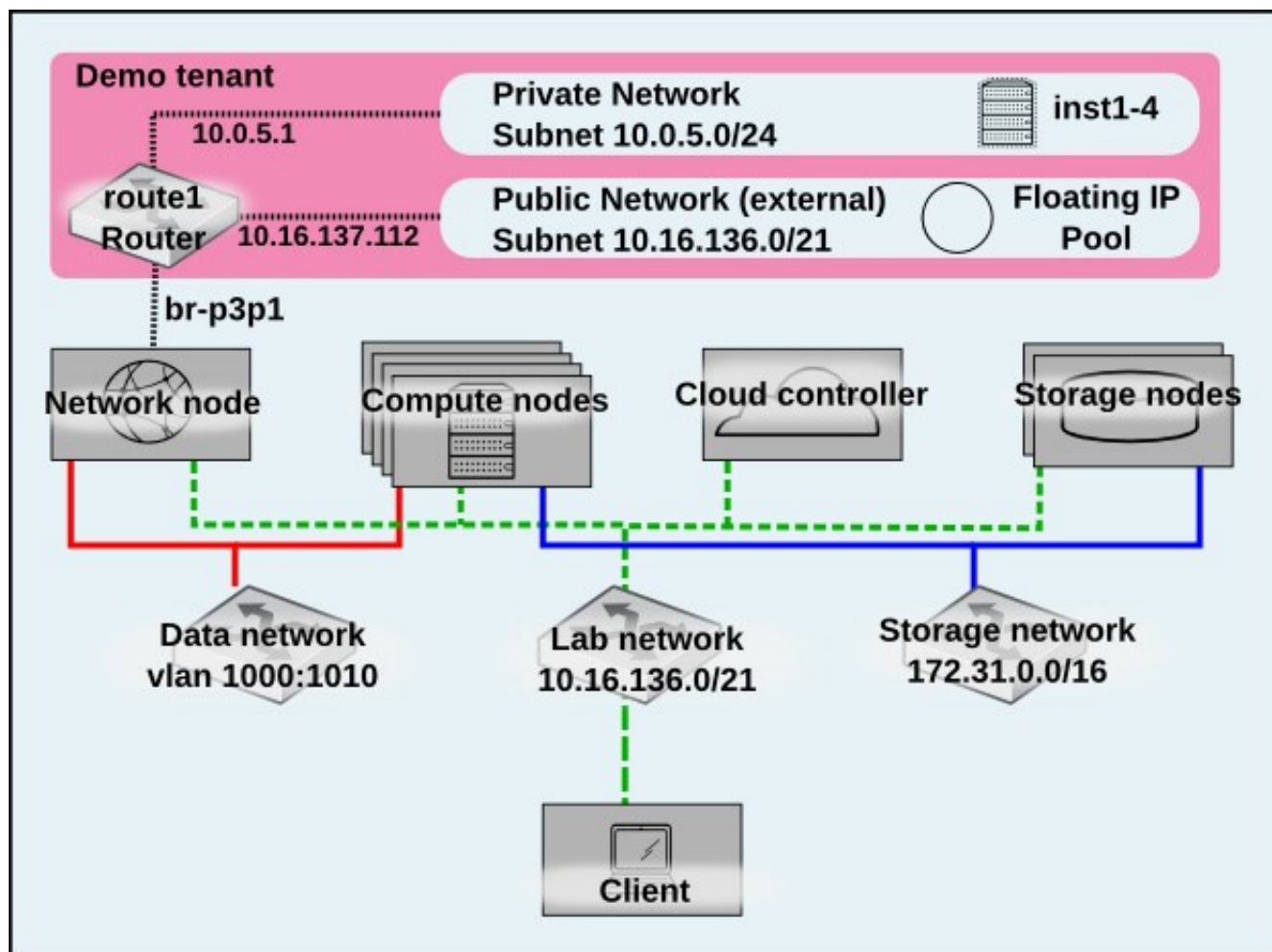
node. This demonstrates network connectivity between the network server and the software router.

```
[root@rhos1 ~(keystone_admin)]# ssh rhos6 ip netns exec $qrouted_id ping -c 3 $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
64 bytes from 10.0.5.1: icmp_seq=3 ttl=64 time=0.040 ms

--- 10.0.5.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.040/0.048/0.059/0.011 ms
```

## 6.5.4 Validated Installation

**Illustration 6.5.4.1 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 Network server's bridge interface.



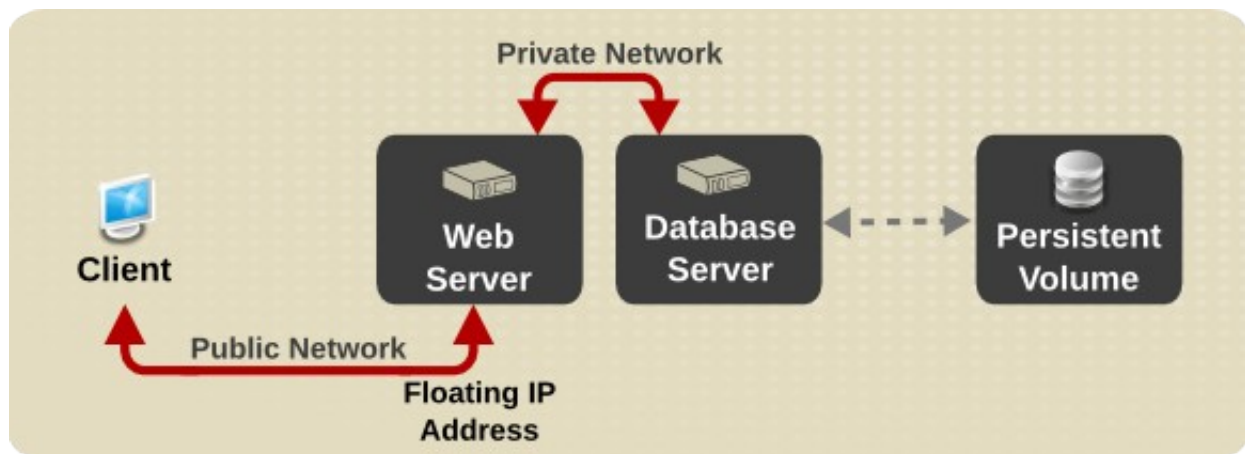
*Illustration 6.5.4.1: Validated Installation*



# 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 virtual network. The database server stores its database on a persistent volume provided by the Block Storage service.



**Figure 4: 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 needed. Keeping the database server off a publicly accessible network provides an extra layer of security. **Figure 4: Multi-tier Application** depicts the end state of the multi-tier application deployment.

## 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 and EPEL so it can install necessary packages.
- 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 Prepare the Environment

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. This will be pushed to the */etc/resolv.conf* file on instances booted on this subnet.

```
[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/364184/40/1>. The script is named *wp-backend.sh*. It installs and configures the MySQL server software. The MySQL database is installed to a persistent volume.

## 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. Find the volume id of the **mysql** volume and save it 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
```

2. Save the **private** network id to an environment variable. This may have been accomplished in a previous step.

```
[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
```

3. Create the 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

4. 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		

## 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/projects/rhos/grizzly/scripts/jliberma/wp-mysql.sh
--flavor 2 --key-name demokp --file /etc/hosts=/etc/hosts --file
sysreg.sh=/pub/projects/rhos/grizzly/scripts/jliberma/sysreg.sh --meta
sysregopts="-u admin -p 100yard- -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 100yard-" --meta
wp_http_ip="10.0.5.201" --block-device-mapping vdd=$volid:::0 --nic net-
id=$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	[{'name': 'default'}]
user_id	365cea099c874ff3b53b94ead32e93da
name	ra1
adminPass	vT2h2xKWADrD
tenant_id	4f2f60304f574ad4bb40920a39bb8396
created	2013-07-25T18:10:04Z
OS-DCF:diskConfig	MANUAL
metadata	{'sysregopts': 'u-admin -p 100yard- -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', 'wp_http_ip': '10.0.5.201', 'rhncreds': 'u-admin -p 100yard-'}
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**.

- **--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>**





[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 will contain the database.

## 7.2.6 Verify the Database Server

1. Save the network namespace `qrouter` to an environment variable.

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

[root@rhos1 ~(demo_member)]# echo $qroute
qrouter-a89302d2-307d-477d-bc01-9bbd2244c05a
```

2. 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.

```
[root@rhos1 ~(demo_member)]# ssh rhos6 ip netns exec $qroute ssh -i
/root/demokp.pem 10.0.5.200 service mysqld status
root@rhos6's password: *****
mysqld (pid 1840) is running...
```

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

```
[root@rhos1 ~(demo_member)]# ssh rhos6 ip netns exec $qroute ssh -i
/root/demokp.pem 10.0.5.200 mount | grep vd
root@rhos6's password: *****
/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/364184/40/1>

**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/projects/rhos/grizzly/scripts/jliberma/wp-
frontend.sh --flavor 2 --key-name demokp --file /etc/hosts=/etc/hosts --file
sysreg.sh=/pub/projects/rhos/grizzly/scripts/jliberma/sysreg.sh --meta
sysregopts="-u admin -p 100yard- -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 100yard-" --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 100yard- -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 100yard-'}
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.

```
[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. **Figure 5: Wordpress Admin Screen** is displayed after a successful installation.



**Figure 5: Wordpress Admin Screen**



## 7.5 Complete Multi-tier Application

**Figure 6: Complete Multi-tier Application** depicts the complete multi-tier web application deployed in this section.

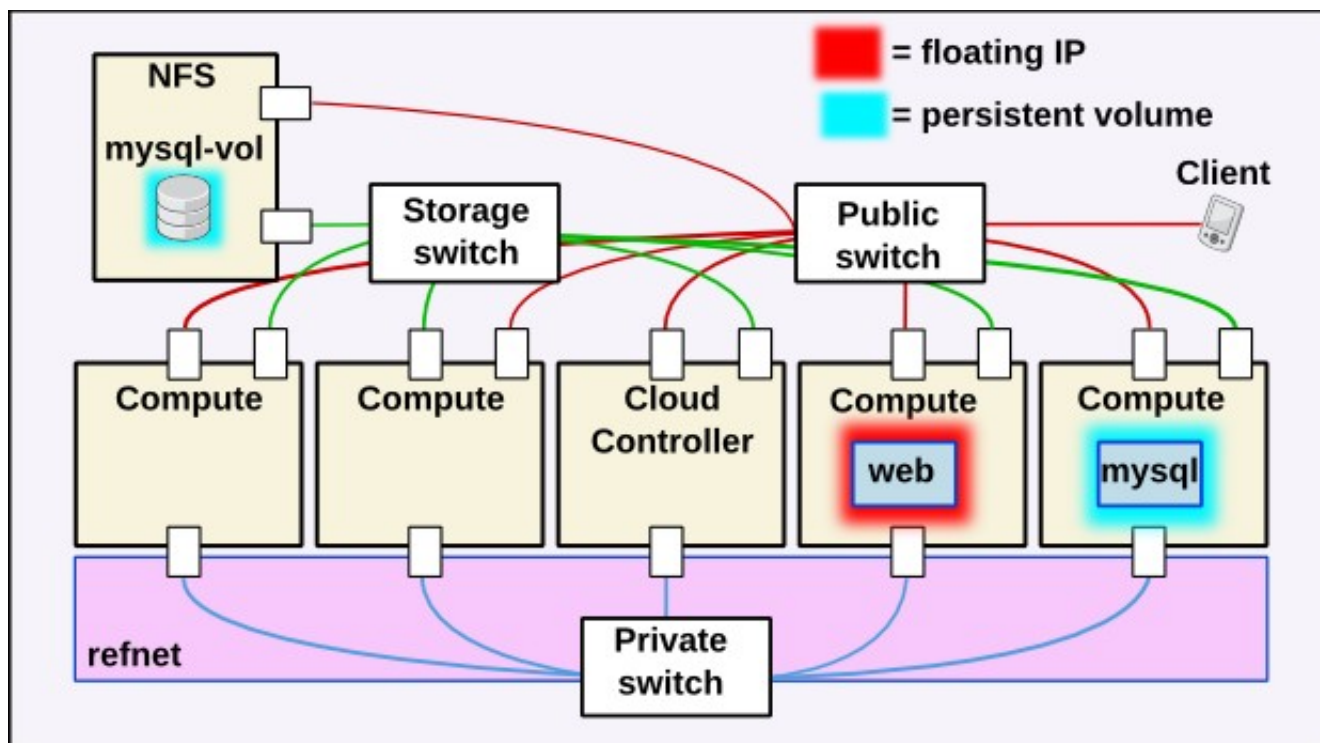


Figure 6: Complete Multi-tier Application



## 8 Conclusion

Red Hat OpenStack 3.0 provides a flexible cloud framework built on a stable code base and backed by Red Hat's open source software expertise. Red Hat OpenStack 3.0 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 OpenStack 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 OpenStack 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 exports 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 are 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:

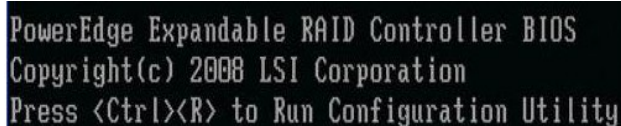
Stripe unit – 256K

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.





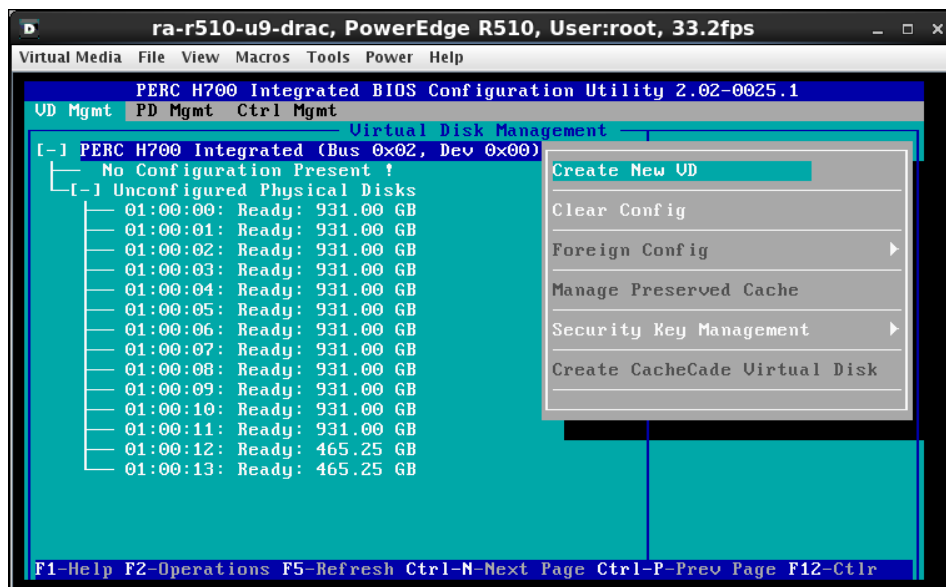
This will launch the PERC Integrated BIOS Configuration Utility. This utility will have the following three TABs on the top.

- VD Mgmt – Virtual Disk Management, which will be 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.



**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.

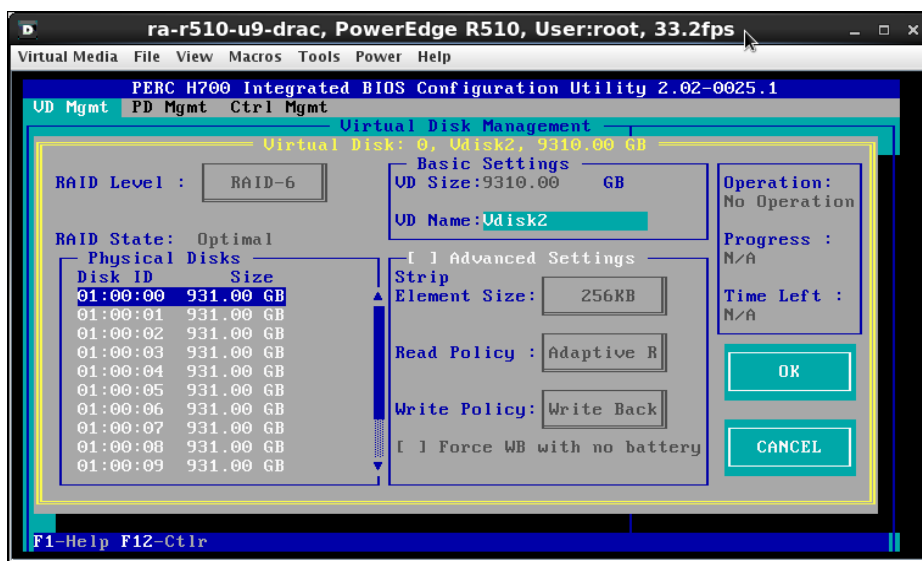


7. Use the arrow key to highlight a physical disk and press the space bar, <Alt>, or **Enter** to select the disk.
8. Select additional disks, if desired.
9. Press <Tab> to move the cursor to the box *Basic Settings*.
10. Press <Tab> to access the **VD Size** field, and type a virtual disk name (**Vdisk2**). The virtual disk size Vdisk2 displays the final pool size (**9310 GB**) 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.



**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**)

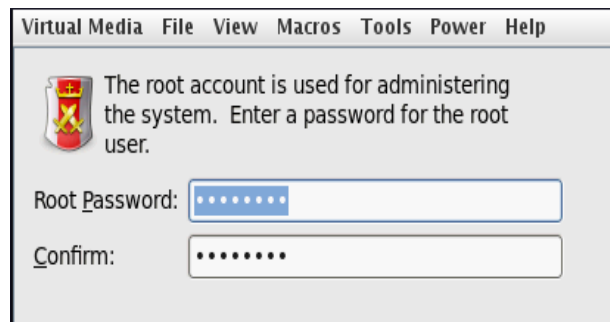
## A.2.2 Install RHS Server

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/](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](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 is a Dell Server. 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.



**Figure A-4: RHS Installation-1**

4. Select a layout on the partitioning screen.

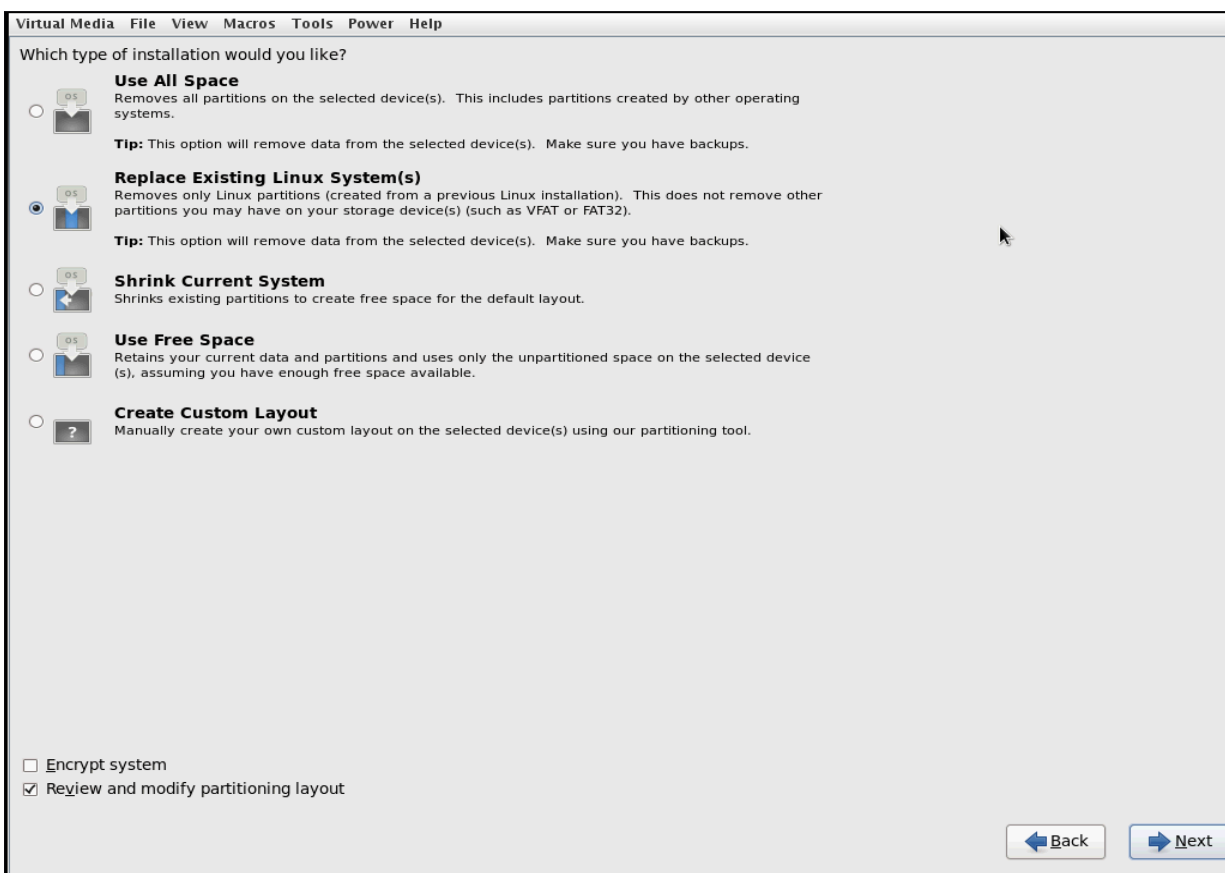


Figure A-5: RHS Installation-2



5. Available disks are displayed.

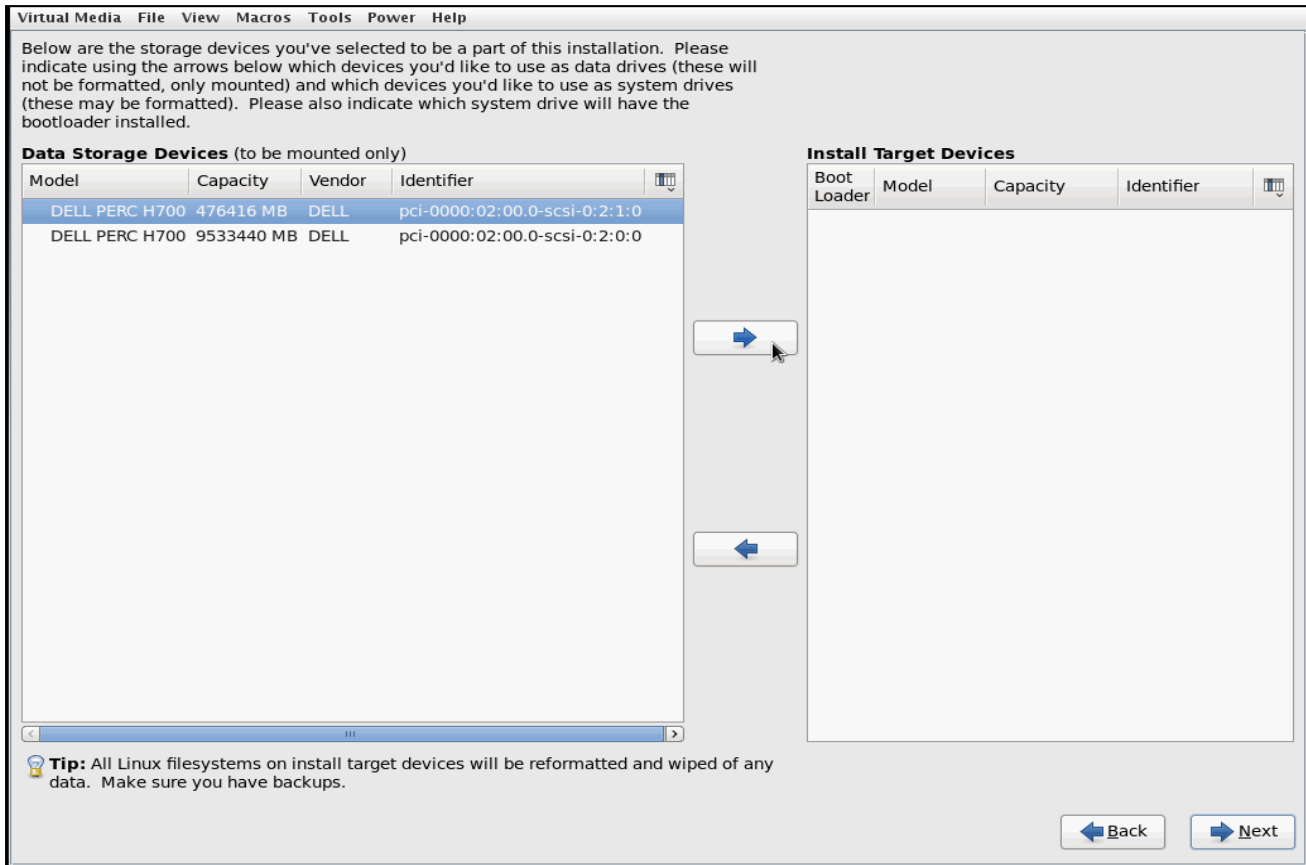
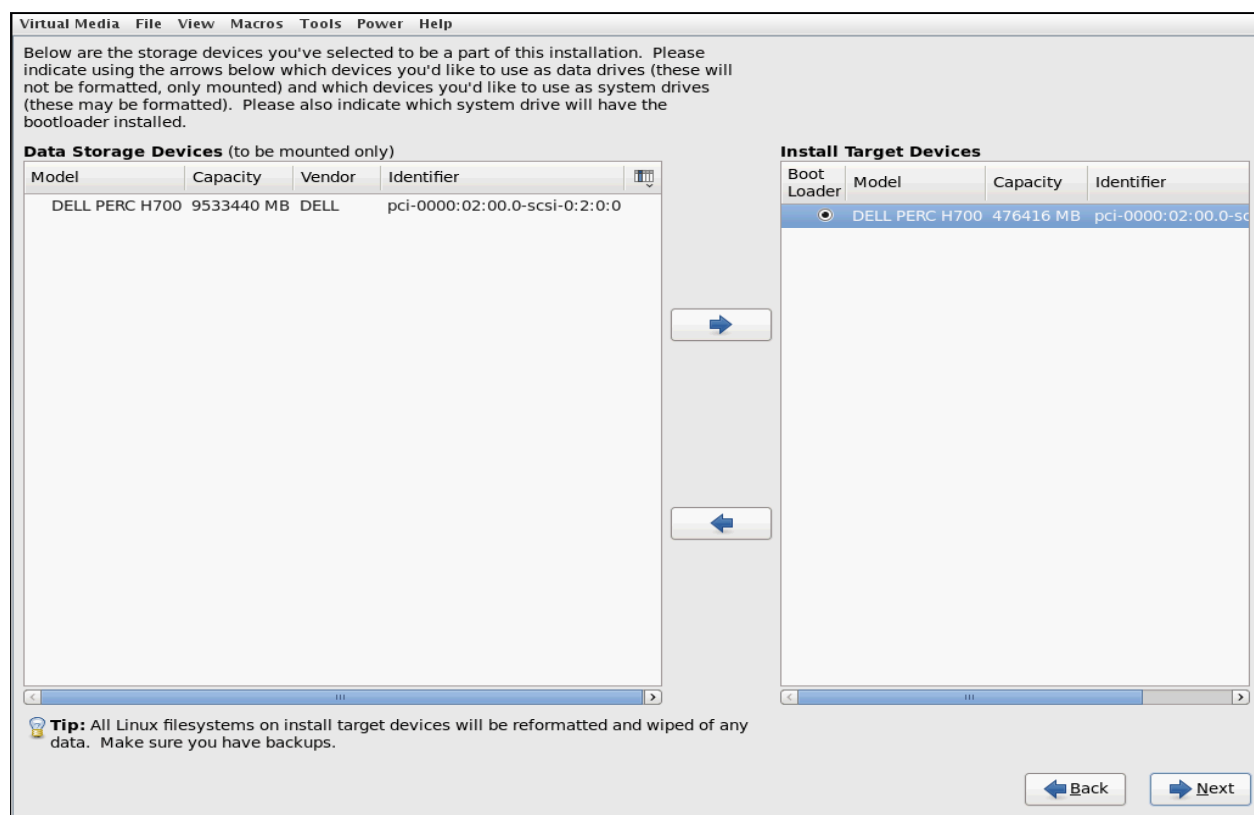


Figure A-6: RHS Installation - Disk selection 1



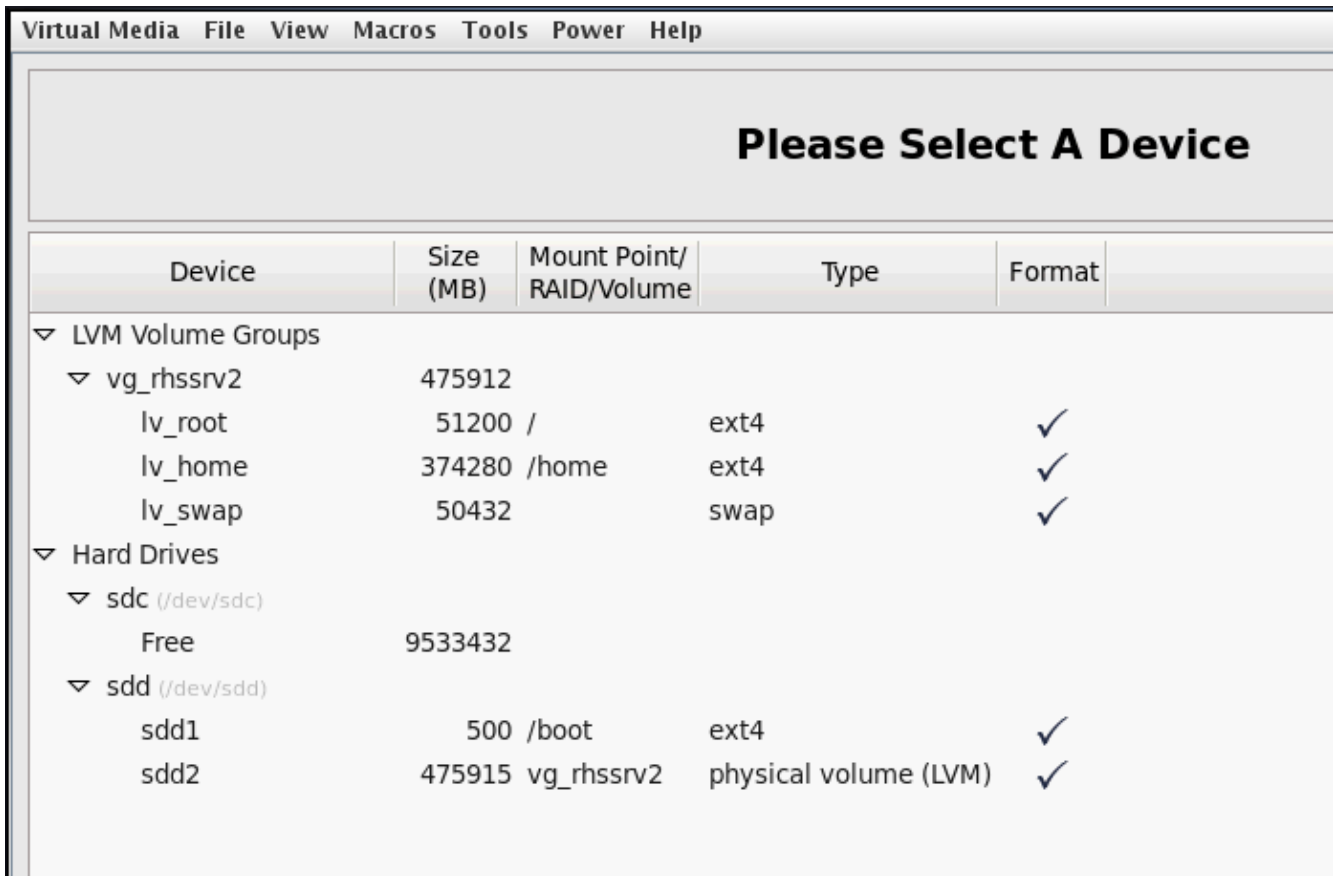
6. Ensure that the right disk and size is allocated to the operating system. Select **Next** to proceed with the installation.



**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.



**Figure A-8: RHS Installation - Root Filesystem**

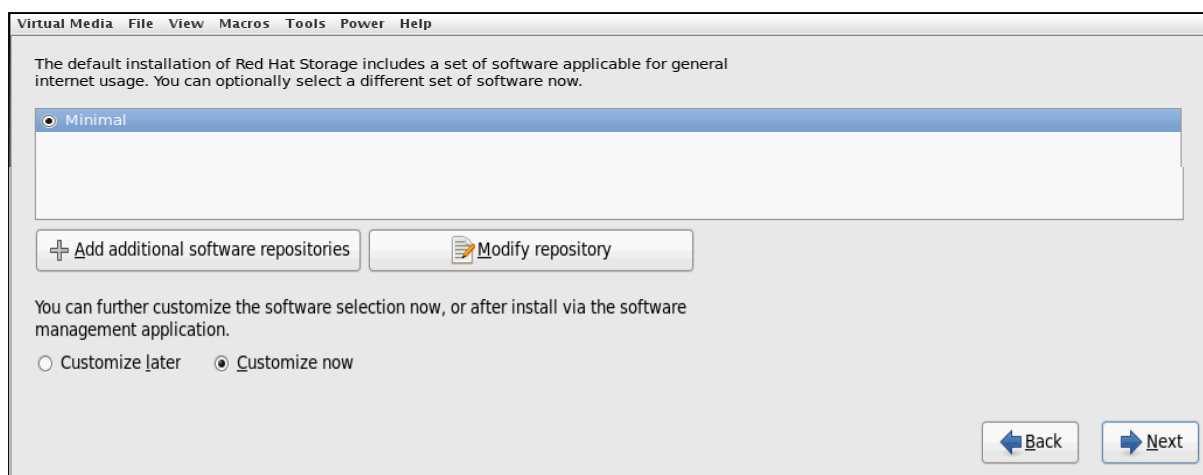
8. Confirm formatting of the right disks.



**Figure A-9: RHS Installation - Disk formatting**

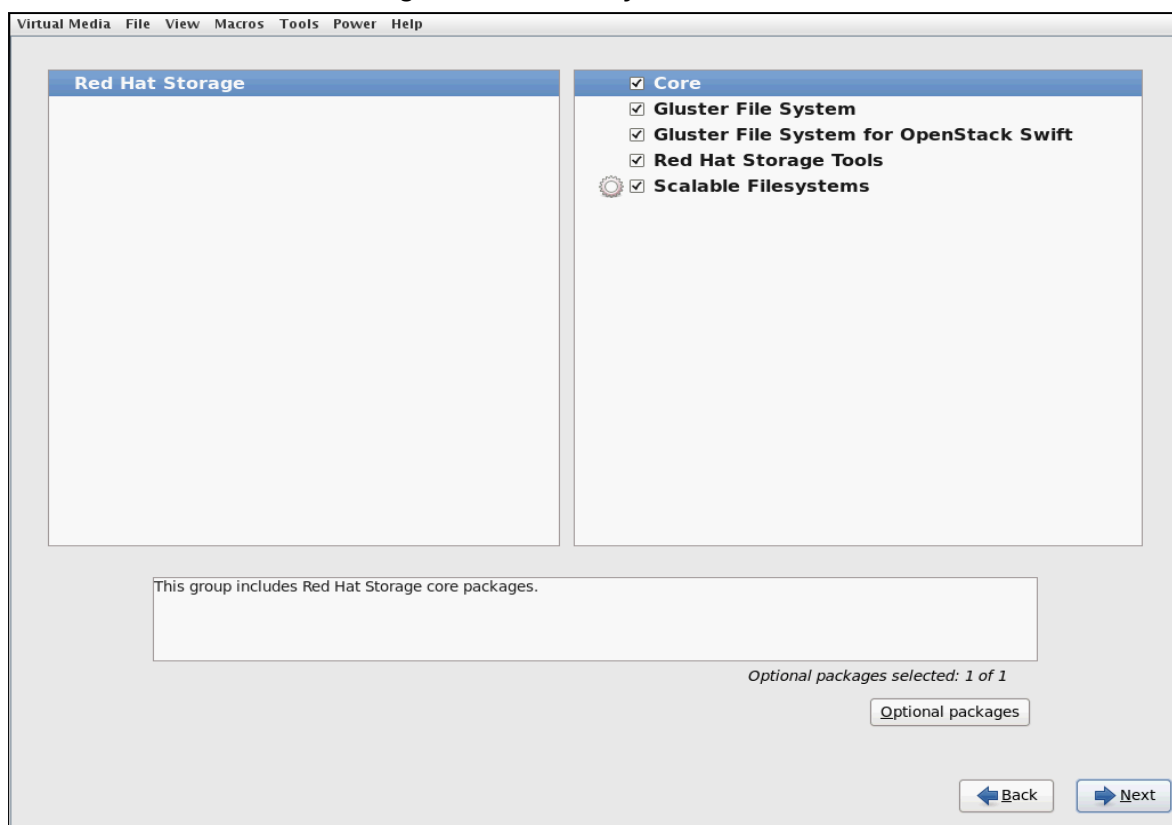


9. Upon confirmation, the installer creates the file systems and proceed to install the default package set for Red Hat Storage Server. By selecting **customize now** the software subsets can be viewed/modified.



**Figure A-10: RHS Installation - Software Selection 1**

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

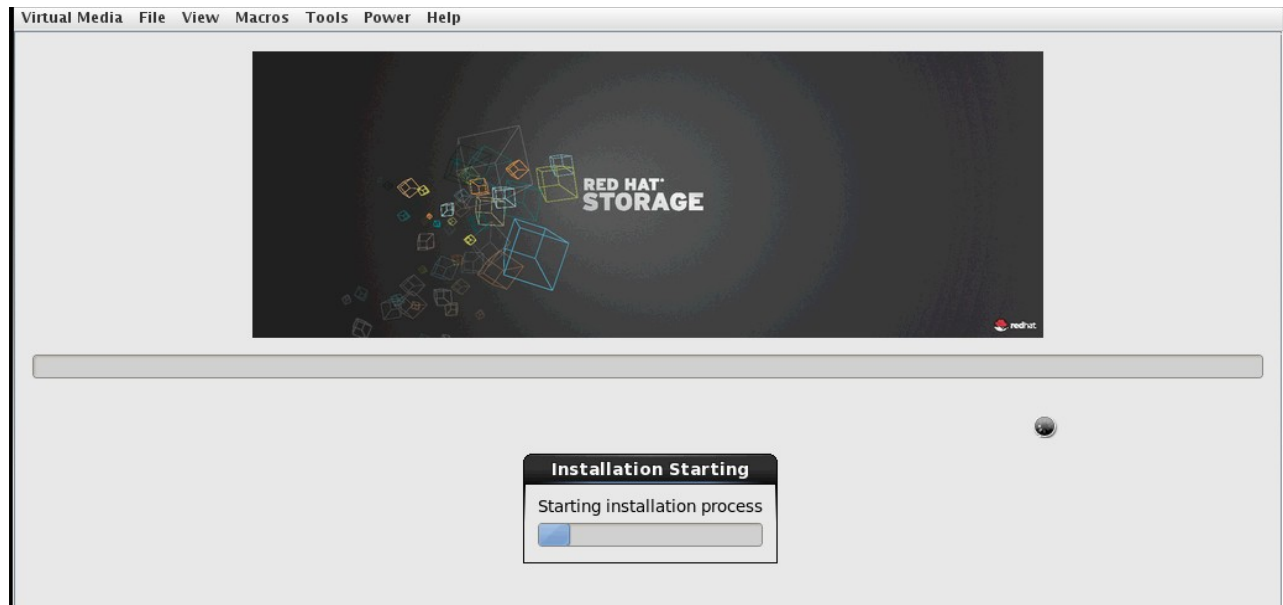


**Figure A-11: RHS Installation - Software Selection 2**





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.



**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 Nodes

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 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 brick
devices: 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 s[ OK ]dc
Starting tuned: [ OK ]
```

This profile performs the following:

- Attempts to increase 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 swappiness 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.5 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.6 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 **pvccreate**.

```
# pvccreate --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 *datavg* 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
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.7 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**.

### 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: Revision History

Revision 0.3	August 2, 2013	Jacob Liberman
Second draft		
<ul style="list-style-type: none"><li>Validated steps, all content</li><li>First review draft</li></ul>		
Revision 0.2	July 19, 2013	Jacob Liberman
First draft		
<ul style="list-style-type: none"><li>Style template</li><li>Graphics</li><li>Appendices</li></ul>		
Revision 0.1	May 12, 2013	Jacob Liberman
Initial Release		
<ul style="list-style-type: none"><li>Title</li><li>Abstract</li><li>Outline</li></ul>		



## Appendix D: Contributors

1. Steve Reichard, content and scripts, technical and content review

