Red Hat Reference Architecture Series

Deploying and Using Red Hat OpenStack 2.1

Jacob Liberman, Principal Software Engineer

RHCE

Version 1.0

May 1, 2013

redhat.

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

The GPG fingerprint of the 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

Comments and Feedback

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

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

Staying In Touch

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

Like us on Facebook:

https://www.facebook.com/rhrefarch

Follow us on Twitter:

https://twitter.com/RedHatRefArch

Plus us on Google+:

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

Table of Contents

1 Executive Summary

OpenStack is a free and open source Infrastructure-as-a-Service (IaaS) cloud computing project released under the Apache License. It enables enterprises and service providers to offer on-demand computing resources by provisioning and managing large networks of virtual machines. OpenStack boasts a massively scalable architecture that can control compute, storage, and networking resources through a unified web interface. The OpenStack development community operates on a six-month release cycle with frequent milestones. Their code base is composed of many loosely coupled projects supporting storage, compute, image management, identity, and networking services. OpenStack's rapid development cycle and architectural complexity create unique challenges for enterprise customers hoping to add OpenStack to their traditional IT portfolios.

Red Hat OpenStack (RHOS) 2.1 addresses these challenges. Red Hat OpenStack 2.1 delivers a stable code base for production OpenStack 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 2.1 through three detailed use cases:

- Installing an evaluation Red Hat OpenStack deployment consisting of a cloud controller and a single compute node
- Expanding the evaluation deployment to four compute nodes, multi-host networking, and NFS backed persistent storage
- Deploying a multi-tier web application to the Red Hat OpenStack infrastructure complete with post-boot customization

This paper contains step by step instructions for expanding an evaluation Red Hat OpenStack 2.1 deployment to a basic cloud architecture suitable for a small production environment. Every step was tested in Red Hat's engineering lab with production code.

2 Component Overview

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

2.1 Red Hat OpenStack

Red Hat OpenStack provides a foundation for organizations 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 OpenStack Services

Red Hat OpenStack 2.1 is based on the upstream OpenStack release code named “Folsom.” This is the sixth OpenStack release and the first deemed sufficiently hardened for production use. Folsom was the first release to include extensible block and volume storage capabilities. It also includes a dedicated advanced network automation platform with Layer 2 networking control, IP address management, and extensions for Layer 3 forwarding.

OpenStack services are modular. Some provide optional capabilities. **Figure 2.2.1: Used Services** depicts the OpenStack services used in this reference architecture. The Networking and Object Storage services are not used in this reference architecture. They are grayed-out in the diagram. These services are optional in the Folsom release and not necessary for the small production deployment described in this reference architecture. The diagram illustrates where they would fit into the overall infrastructure had they been included.

The following sections define the core OpenStack components utilized in this reference architecture. They include:

- Identity (code-named Keystone)
- Image (code-named Glance)
- Compute (code-named Nova)

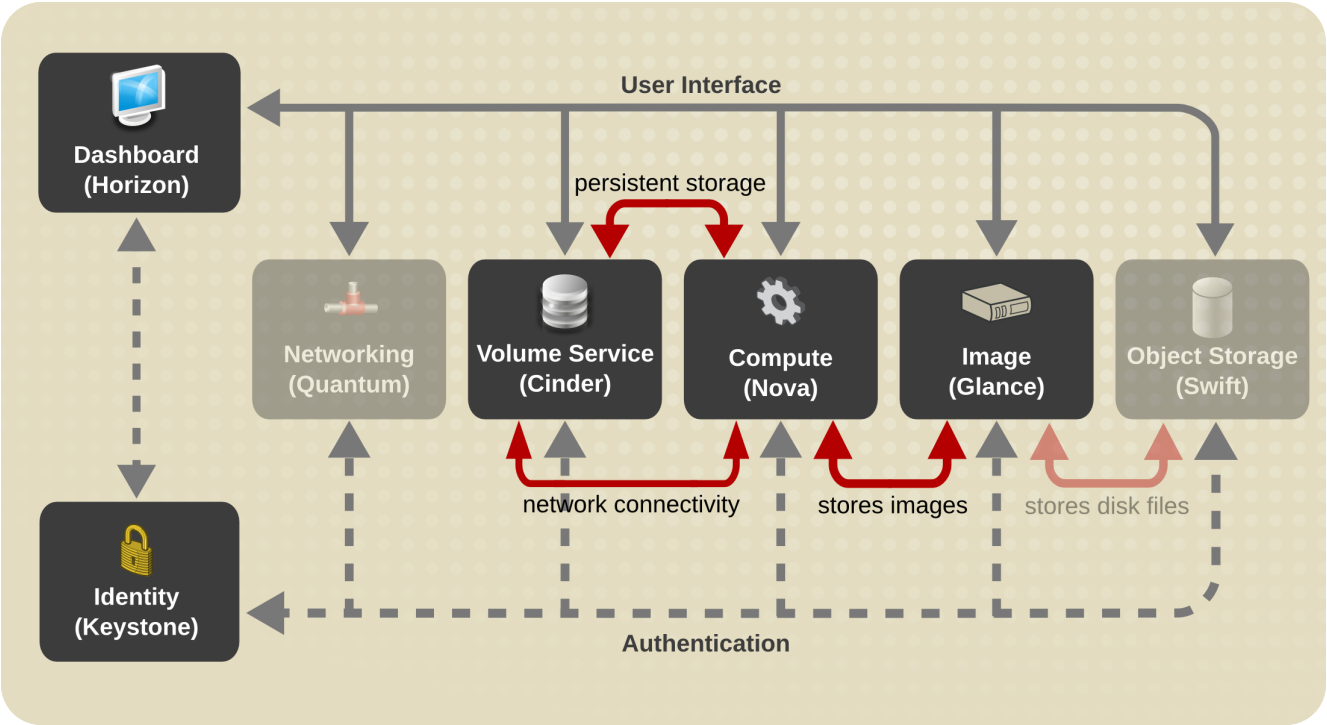


Figure 2.2.1: Used Services

2.2.1 Identity

The OpenStack Identity service is a central directory that maps users to the OpenStack services they can access. It acts as a common authentication system for all OpenStack users and services. The OpenStack Identity service can integrate with existing directory services such as LDAP. It also supports multiple forms of authentication including standard username and password credentials, token-based systems and AWS-style logins that use public/private key pairs.

The OpenStack Identity service catalog provides a way to query and list of all of the services deployed in an OpenStack cloud. Administrators can programmatically determine which resources users and third-party tools can access.

The Identity service authenticates users based on their access credentials. After the user is authenticated the user is issued a token. This token can be used to demonstrate that the user's identity has been authenticated when making subsequent requests. The token includes a list of roles the user may assume when performing a set of operations. This method of access control is commonly referred to as Role Based Access Control (RBAC). Administrators associate grant-or-deny access privileges with each role through either the Horizon dashboard interface or the Keystone command line interface.

The Identity service also manages authentication for OpenStack services through *Endpoints*. An Endpoint is a network-accessible address where a service may be accessed. The Identity

service provides each OpenStack service -- such as Image, Compute, or Block Storage -- with one or more endpoints through which is can be accessed to perform useful operations.

The Identity service uses tenants to group or isolate resources. User accounts in one tenant may not be able to access resources in another tenant even if they reside within the same OpenStack cloud deployment or physical host. Keystone enforces tenants by issuing tokens to authenticated users. The endpoint service validates the token before allowing user access.

2.2.2 Image

The OpenStack Image Service discovers, registers, and delivers disk and server images. The images 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 also stores and catalogs snap shots. The Image Service API provides a standard REST interface for querying information about the images.

The Image Service can store images in a variety of back-ends including OpenStack Object Storage. By default Glance stores images in the Image server's local filesystem in the */var/lib/glance/images* directory. The image store location is a configurable option. Glance supports several backend storage technologies including Swift (the OpenStack Object Storage service), Amazon S3, and RBD. The Glance API can also be configured to cache images locally on the hosts that use them in order to reduce image staging time.

2.2.3 Compute

OpenStack Compute is used to provision and manage large networks of virtual machines. It controls the computing fabric which is the backbone of OpenStack's IaaS functionality. Compute resources are accessible via APIs for developers and via web interfaces and command line tools for administrators and users. OpenStack compute scales horizontally on standard hardware enabling the favorable economics of cloud computing.

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
- Virtual Local Area Networking (LAN) management including VLANs, IP address assignment, and floating IP addresses
- Security groups to flexibly assign instances to resource pools and control access to them
- VNC access to instances via web browsers

2.2.4 Block Storage

The OpenStack Compute service provisions ephemeral storage for deployed instances based

on the instance's hardware profile. The OpenStack Block Storage service exposes block devices to compute instances for persistent storage. The Block Storage service can be backed by a variety of storage technologies that provide expanded capacity, better performance, or integration with enterprise storage platforms or simple NFS servers. Block Storage features include:

- Provides persistent block storage devices to compute instances
- Self-service users can manage the creation, attaching, and detaching of block devices to servers through the Dashboard
- Unified storage support for numerous storage platforms
- Block storage is appropriate for performance sensitive scenarios such as database storage or expandable file systems
- Snapshots can back up data stored on block storage volumes and restored to a new block storage volume

2.2.5 Dashboard

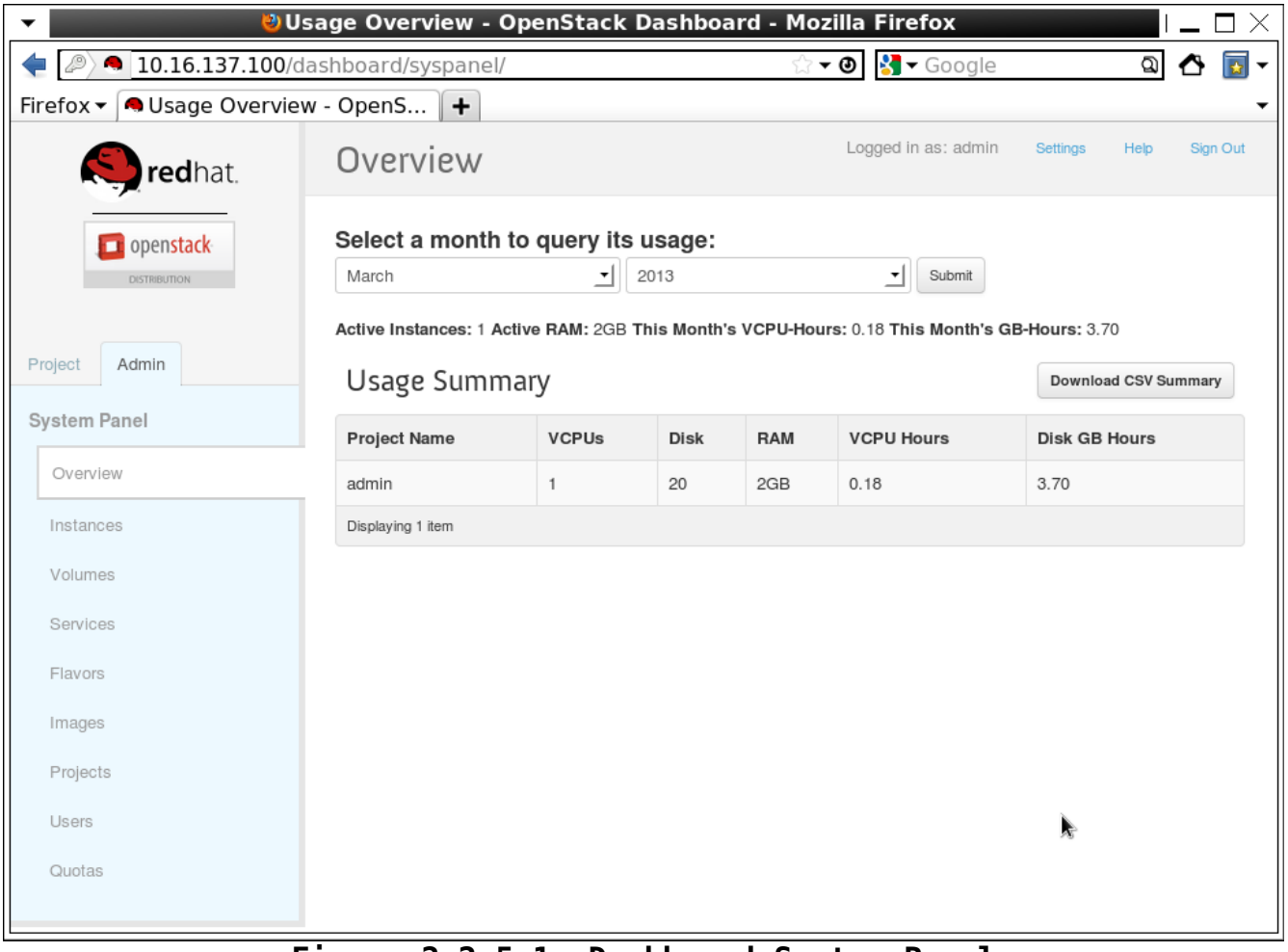


Figure 2.2.5.1: Dashboard System Panel

The OpenStack Dashboard is an extensible web app that allows cloud administrators and

users to control their compute, storage, and networking resources. It provides users with a self-service portal to provision their own resources. Administrators can use the Dashboard to view the state of the cloud, create users, assign them to tenants, and set resource limits. **Figure 2.2.5.1: Dashboard System Panel** depicts the Horizon Dashboard Overview screen.

2.2.6 Services Not Covered in this Reference Architecture

- **Network Service** -- OpenStack Networking service (code-named Quantum) is a pluggable, scalable and API-driven system for managing networks and IP addresses. OpenStack Networking gives users self-service control over their network configurations. Users can define, separate, and join networks and devices on demand. It also provides flexible network models that can be adapted to fit the requirements of different applications.

  NOTE: The OpenStack Network service is not discussed in this reference architecture. **`Nova-network`**, the basic networking functionality provided by the Compute service, was used instead.

- **Object Storage –** the OpenStack Object Storage service (code-named Swift) provides a fully distributed, API-accessible storage platform that can be integrated directly into applications or used for backup, archiving and data retention. OpenStack 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.

  NOTE: The OpenStack Object Storage service is not discussed in this reference architecture. Future Red Hat reference architectures describe object storage use cases and deployment strategies.

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.

Red Hat Enterprise Linux 6.4 provides the following features and capabilities:

Reliability, Availability, and Security (RAS):

- More sockets, more cores, more threads, and more memory
- RAS hardware-based hot add of CPUs and memory is enabled

- Memory pages with errors can be declared as “poisoned” and can be avoided

File Systems:

- ext4 is the default file system and scales to 16TB
- XFS is available as an add-on and can scale to 100TB
- Fuse allows file systems to run in user space allowing testing and development on newer fuse-based file systems (such as cloud file systems)

High Availability:

- Extends the current clustering solution to the virtual environment allowing for high availability of virtual machines and applications running inside those virtual machines
- Enables NFSv4 resource agent monitoring
- Introduction of Cluster Configuration System (CCS). CCS is a command line tool that allows for complete CLI administration of Red Hat's High Availability Add-On

Resource Management:

- *cgroups* organize system tasks so that they can be tracked and other system services can control the resources that cgroup tasks may consume
- *cpuset* applies CPU resource limits to cgroups, allowing processing performance to be allocated to tasks

There are many other feature enhancements to Red Hat Enterprise Linux 6. Refer to the Red Hat website for more information.

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

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

2.4.1 MySQL

In the OpenStack architecture the SQL database stores most of the build-time and run-time state for a cloud infrastructure. This includes 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 2.1 uses MySQL, a widely used open source database packaged with Red Hat Enterprise Linux.

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 instances and services. Red Hat OpenStack 2.1 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.

2.4.4 Packstack

Packstack is a utility to install Red Hat OpenStack 2.1 on one or more hosts via SSH. It is primarily useful for evaluation deployments. Packstack uses Puppet modules to install parts of OpenStack. Puppet modules ensure OpenStack can be installed and expanded in a consistent and repeatable manner.

2.4.5 Supporting Technologies Not Included 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 2.1 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.
- **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. Red Hat OpenStack 2.1 does not depend on Red Hat Storage Server. Strategies for integrating Red Hat Storage Server with Red Hat OpenStack is the topic of future reference architectures.

3 Reference Architecture Configuration Details

This section of the paper describes the hardware, software, and configuration steps used to stand up this configuration in the Red Hat reference architecture 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

Figure 3.1.1.1: Network Topology shows the network topology used in this reference architecture.

- All six servers and the client communicate via the public switch.
- All of the servers minus the client are also attached to the storage switch via a second interface. The storage network is primarily used for instance communication with the persistent block storage.

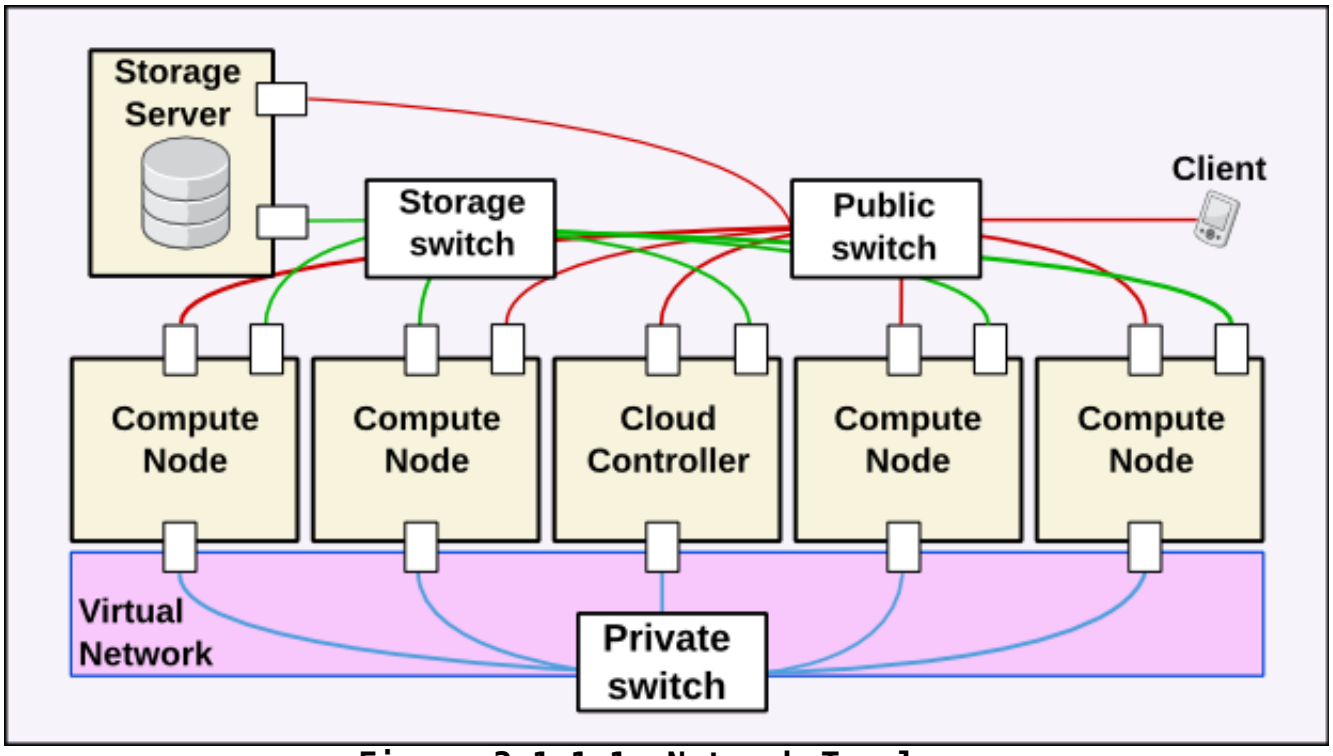


Figure 3.1.1.1: Network Topology

- The cloud controller and compute nodes communicate via the private switch. The private switch interface also acts as a bridge for the virtual machine network on the

compute nodes.

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 cloud controller and compute nodes have three interfaces on separate networks: public, private, and storage. The storage server is connected to the public and storage networks. The client is only connected to the public network. The client simulates a public system from which a self-service user would access the OpenStack infrastructure.

Host	Role	Network	IP Address
rhos0	Cloud Controller	Public	10.16.137.100
rhos0-priv	Cloud Controller	Private	192.168.137.100
rhos0-stor	Cloud Controller	Storage	172.31.139.100
rhos1	Storage Server	Public	10.16.137.101
rhos1-stor	Storage Server	Storage	172.31.139.101
rhos2	Compute Node	Public	10.16.137.102
rhos2-priv	Compute Node	Private	192.168.137.102
rhos2-stor	Compute Node	Storage	172.31.139.102
rhos3	Compute Node	Public	10.16.137.103
rhos3-priv	Compute Node	Private	192.168.137.103
rhos3-stor	Compute Node	Storage	172.31.139.103
rhos4	Compute Node	Public	10.16.137.104
rhos4-priv	Compute Node	Private	192.168.137.104
rhos4-stor	Compute Node	Storage	172.31.139.104
rhos5	Compute Node	Public	10.16.137.105
rhos5-priv	Compute Node	Private	192.168.137.105
rhos5-stor	Compute Node	Storage	172.31.139.105
rhos6	Client	Public	10.16.137.106

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) 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	python-qpid	0.14-11
	mysql-server	5.1.67-1
	openstack-packstack	2012.2.3-0.11
	openstack-keystone	2012.2.3-8
	openstack-nova-{api,cert,common,scheduler,console}	2012.2.3-9
	openstack-glance	2012.2.3-9
	openstack-cinder	2012.2.3-7
	openstack-dashboard	2012.2.3-9
Compute Node	openstack-nova-{api,common,compute,network}	2012.2.3-9
	openstack-utils	2013.1-2.2
Storage Node	nfs-utils	1.2.3-36
	rpcbind	0.2.0-11
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

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, role, and protocol.

Port	Host	Role	Protocol
22, 80, 443, 3306, 5000, 5672, 6080, 8773:8776, 9292, 35357	rhos0	Cloud Controller	TCP
22, 111, 662, 2049, 892, 32803	rhos1	Storage Server	TCP
111, 662, 892, 32769	rhos1	Storage Server	UDP
22, 80, 5900:5950, 8773:8776	rhos2-5	Compute nodes	TCP
22, 80, 443	rhos6	Client	TCP

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
selinux-policy	3.7.19-195
selinux-policy-targeted	3.7.19-195
openstack-selinux	0.1.2-10

Table 3.2.3.1: Supported SELinux Package Versions

3.2.4 Required Channels

Initially Red Hat OpenStack is only available via the Red Hat Network via the channels listed in **Table 3.2.4.1: Required Channels**.

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

Table 3.2.4.1: Required Channels

NOTE: This reference architecture used RHN Classic in all examples.

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-compute	rhos2-5	Compute node
openstack-nova-console	rhos0	Cloud Controller
openstack-nova-consoleauth	rhos0	Cloud Controller
openstack-nova-metadata-api	rhos2-5	Compute node
openstack-nova-network	rhos2-5	Compute node
openstack-nova-novncproxy	rhos0	Cloud Controller
openstack-nova-scheduler	rhos0	Cloud Controller
openstack-nova-xvpvncproxy	rhos0	Cloud Controller

Table 3.4.1: Service Placement

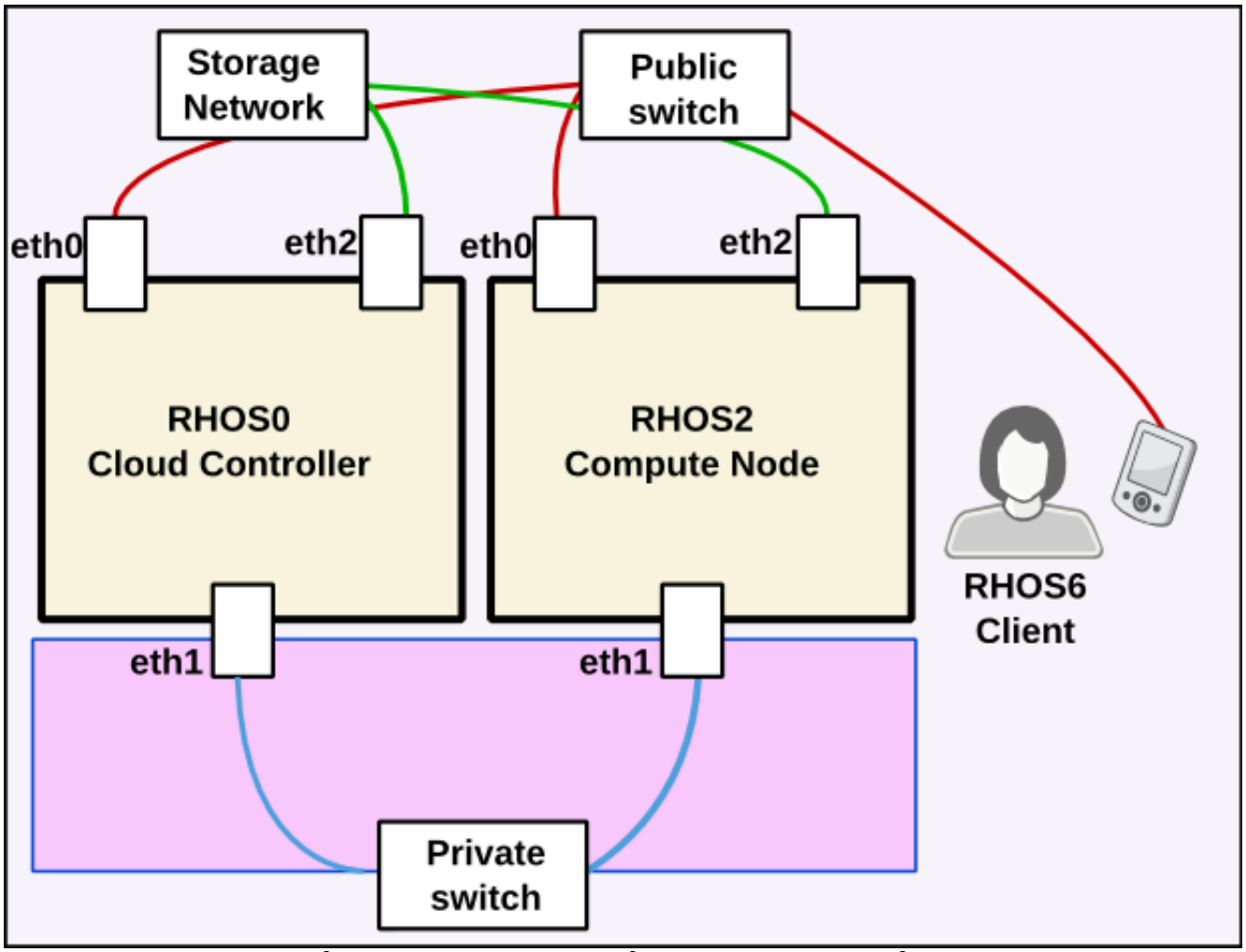
4 Deploy Cloud Controller, Compute Node, and Client via Packstack

This section of the paper describes the steps required to install a Red Hat OpenStack evaluation deployment via **`packstack`**.

The initial deployment is later expanded to demonstrate OpenStack's scale-out capability.

4.1 Architectural Overview

Figure 4.1.1: Architectural Overview shows the roles by host.



`Figure 4.1.1: Architectural Overview`

The initial reference architecture is limited to the following roles:

1. Cloud controller

2. Compute node
3. Client

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. In this reference architecture these services include Nova, Glance, Cinder, Keystone, and Horizon. 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 accept actions from the message queue, perform a series of commands, and then update the state database.

In this reference architecture the OpenStack deployment is scaled from one to four compute nodes to increase capacity and service availability.

4.1.3 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.

4.2 Prepare the Hosts

Perform the following steps on the cloud controller prior to deploying RHOS:

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 2.1 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
```

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
192.168.137.100  rhos0-priv
172.31.139.100   rhos0-stor
10.16.137.101    rhos1
172.31.139.101   rhos1-stor
10.16.137.102    rhos2
192.168.137.102  rhos2-priv
172.31.139.102   rhos2-stor
10.16.137.103    rhos3
192.168.137.103  rhos3-priv
172.31.139.103   rhos3-stor
10.16.137.104    rhos4
192.168.137.104  rhos4-priv
172.31.139.104   rhos4-stor
10.16.137.105    rhos5
192.168.137.105  rhos5-priv
172.31.139.105   rhos5-stor
10.16.137.106    rhos6
```

4.2.4 Create an SSH Key on the Cloud Controller

This section shows how to set up password-less SSH logins.

1. Create an SSH key on the cloud controller.

```
[root@rhos0 ~]# ssh-keygen -t rsa -N ""  -f /root/.ssh/id_rsa
Generating public/private rsa key pair.
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
```

```
e3:ed:34:50:11:b6:01:10:b7:d1:30:9d:c9:2f:ec:9d root@rhos0.example.com
The key's randomart image is:
+--[ RSA 2048]----+
|      oo**=+     |
|       . =*+     |
|        ..o.     |
|         .o .    |
|        S. o .   |
|       . +. E    |
|        . +      |
|         o .     |
|          .      |
+-----------------+

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

4.3 Deploy Cloud Controller and Compute Node via Packstack

Packstack is a utility that uses **Puppet** modules to deploy various parts of OpenStack on multiple preinstalled servers over SSH. **Packstack** was used in this reference architecture to deploy the cloud controller, the first compute node, and the OpenStack client tools.

4.3.1 Prepare the Cloud Controller for Packstack

Packstack is an interactive script whose answers define the Red Hat OpenStack configuration to be applied via Puppet modules. Red Hat OpenStack and **packstack** are distributed through RHN.

1. Add the OpenStack RHN Classic channel with the **rhn-channel** command. **rhel-x86_64-server-6** and **rhel-x86_64-server-6-ost-folsom** channels from RHN Classic are shown in this example.

```
[root@rhos0 ~]# rhn-channel -a -c rhel-x86_64-server-6-ost-folsom -u admin -p
password

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

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
This system is not registered to Red Hat Subscription Management. You can
use subscription-manager to register.
This system is receiving updates from RHN Classic or RHN Satellite.

Installed Packages
openstack-packstack.noarch
2012.2.3-0.11.dev475.el6ost
@openstack
```

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. The default answers to many of the questions assume **packstack** is running on the cloud controller.

Answers are stored in a time-stamped answer file. The answer file should be given to **packstack** as input for subsequent runs.

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

6. Place the **nova-compute** and **nova-network** services on the compute node.

   NOTE: In this reference architecture, **eth1** was used as the private interface on Nova compute and network servers.

```
Enter a comma separated list of IP addresses on which to install the Nova
Compute services  [10.16.137.100] : 10.16.137.102
Enter the Private interface for Flat DHCP on the Nova compute servers
[eth1] :
Enter the IP address of the Nova Network service  [10.16.137.100] :
10.16.137.102
Enter the Public interface on the Nova network server  [eth0] :
Enter the Private interface for Flat DHCP on the Nova network server
[eth1] :
```

7. The Flat DHCP range defines which IPv4 addresses are assigned to virtual machine instances at boot.

```
Enter the IP Range for Flat DHCP  [192.168.32.0/22] : 172.16.1.0/24
```

8. Floating IPs are typically a range of publicly accessible addresses assigned to virtual machine instances on an as-needed basis to communicate with the outside world.

   NOTE: In this reference architecture the floating IP addresses are routable to the public interface on the infrastructure servers. The range used in this example only creates two addresses. Use a larger range if more addresses are required.

```
Enter the IP Range for Floating IP's  [10.3.4.0/22] : 10.16.143.108/30
```

9. The **nova-scheduler** service determines how to dispatch compute requests such as where to boot a new virtual machine instance.

   By default **nova-scheduler** assigns instances to the compute node that meets the minimum hardware requirements and has the most free RAM.

The RAM and CPU over commitment ratios allow **nova-scheduler** to continue placing instances with aggregate hardware requirements on compute nodes until the over commitment ratio is reached.

```
Enter the IP address of the Nova Scheduler service  [10.16.137.100] :
Enter the CPU over commitment 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] :
```

10. The client server is the machine from which self-service users initiate OpenStack operations using command line utilities.

```
Enter the IP address of the client server  [10.16.137.100] : 10.16.137.106
```

11. Place Horizon on the client.

```
Enter the IP address of the Horizon server  [10.16.137.100] : 10.16.137.106
Would you like to set up Horizon communication over https [y|n]  [n] : y
```

12. The remaining questions provide additional customization and system registration options. Accept defaults.

```
Enter the path to a PEM encoded certificate to be used on the https server,
leave blank if one should be generated, this certificate should not require
a passphrase:
Enter the keyfile corresponding to the certificate if one was entered:
Enter a comma separated list of URLs to any additional yum repositories to
install:
```

13. Enter Red Hat registration information.

```
To subscribe each server to Red Hat enter a username here: rhnuser
To subscribe each server to Red Hat enter your password here : ********
```

14. The Red Hat Enterprise Linux 6 Server Beta channel is not used in this reference architecture.

```
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] :
```

15. Enter RHN credentials. The same credentials are used to register all systems.

```
To subscribe each server with RHN Satellite enter RHN Satellite server URL:
https://xmlrpc.rhn.redhat.com/XMLRPC
Enter RHN Satellite username or leave plain if you will use activation key
instead: rhnuser
Enter RHN Satellite password or leave plain if you will use activation key
instead : ********
Enter RHN Satellite activation key or leave plain if you used
username/password instead :
Specify a path or URL to a SSL CA certificate to use :
```

```
If required specify the profile name that should be used as an identifier
for the system in RHN Satellite :
Enter comma separated list of flags passed to rhnreg_ks : novirtinfo
Specify a HTTP proxy to use with RHN Satellite :
```

4.3.3 Verify the configuration.

Packstack displays a configuration summary at completion. Type **yes** to accept the configuration and install RHOS 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-horizon-install:             y
os-swift-install:               n
os-client-install:              y
ntp-severs:                     10.16.255.2,10.16.255.3
nagios-install:                 n
mysql-host:                     10.16.137.100
mysql-pw:                       ********
qpid-host:                      10.16.137.100
keystone-host:                  10.16.137.100
glance-host:                    10.16.137.100
cinder-host:                    10.16.137.100
cinder-volumes-create:          y
cinder-volumes-size:            20G
novaapi-host:                   10.16.137.100
novacert-host:                  10.16.137.100
novavncproxy-hosts:             10.16.137.100
novacompute-hosts:              10.16.137.102
novacompute-privif:             eth1
novanetwork-host:               10.16.137.102
novanetwork-pubif:              eth0
novanetwork-privif:             eth1
novanetwork-fixed-range:        172.16.1.0/24
novanetwork-floating-range:     10.16.143.108/30
novasched-host:                 10.16.137.100
novasched-cpu-allocation-ratio:16.0
novasched-ram-allocation-ratio:1.5
osclient-host:                  10.16.137.106
os-horizon-host:                10.16.137.106
os-horizon-ssl:                 y
os-ssl-cert:
os-ssl-key:
additional-repo:
rh-username:                    rhnuser
rh-password:                    ********
rh-beta-repo:                   n
rhn-satellite-server:           https://xmlrpc.rhn.redhat.com/XMLRPC
rhn-satellite-username:         rhnuser
rhn-satellite-password:         ********
rhn-satellite-activation-key:
```

```
rhn-satellite-cacert:
rhn-satellite-profile:
rhn-satellite-flags:           novirtinfo
rhn-satellite-proxy-host:
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-20130426-
124943.txt
 * To use the command line tools you need to source the file
/root/keystonerc_admin created on 10.16.137.106
 * NOTE : A certificate was generated to be used for ssl, You should change
the ssl certificate configured in /etc/httpd/conf.d/ssl.conf on
10.16.137.100 to use a CA signed cert.
 * To use the console, browse to https://10.16.137.106/dashboard
 * The installation log file is available at: /var/tmp/packstack/20130426-
124600-avpwiY/openstack-setup.log
```

4.4 Verify the Packstack Deployment

This section describes steps for verifying the **`packstack`** deployment. These steps include importing an image, booting an instance, and connecting to the instance via SSH and noVNC.

4.4.1 Initial Deployment Overview

Figure 4.4.1.1: Initial Deployment shows the initial **`packstack`** deployment with services and networks. Not all OpenStack services are depicted. The **`nova-network`** and **`nova-compute`** services are running on the compute node. The Horizon dashboard and client tools are installed on the client. All other installed OpenStack services are running on the cloud controller.

Private communication occurs via the private switch. The client accesses the services via the public network. At this stage the storage network is unused.

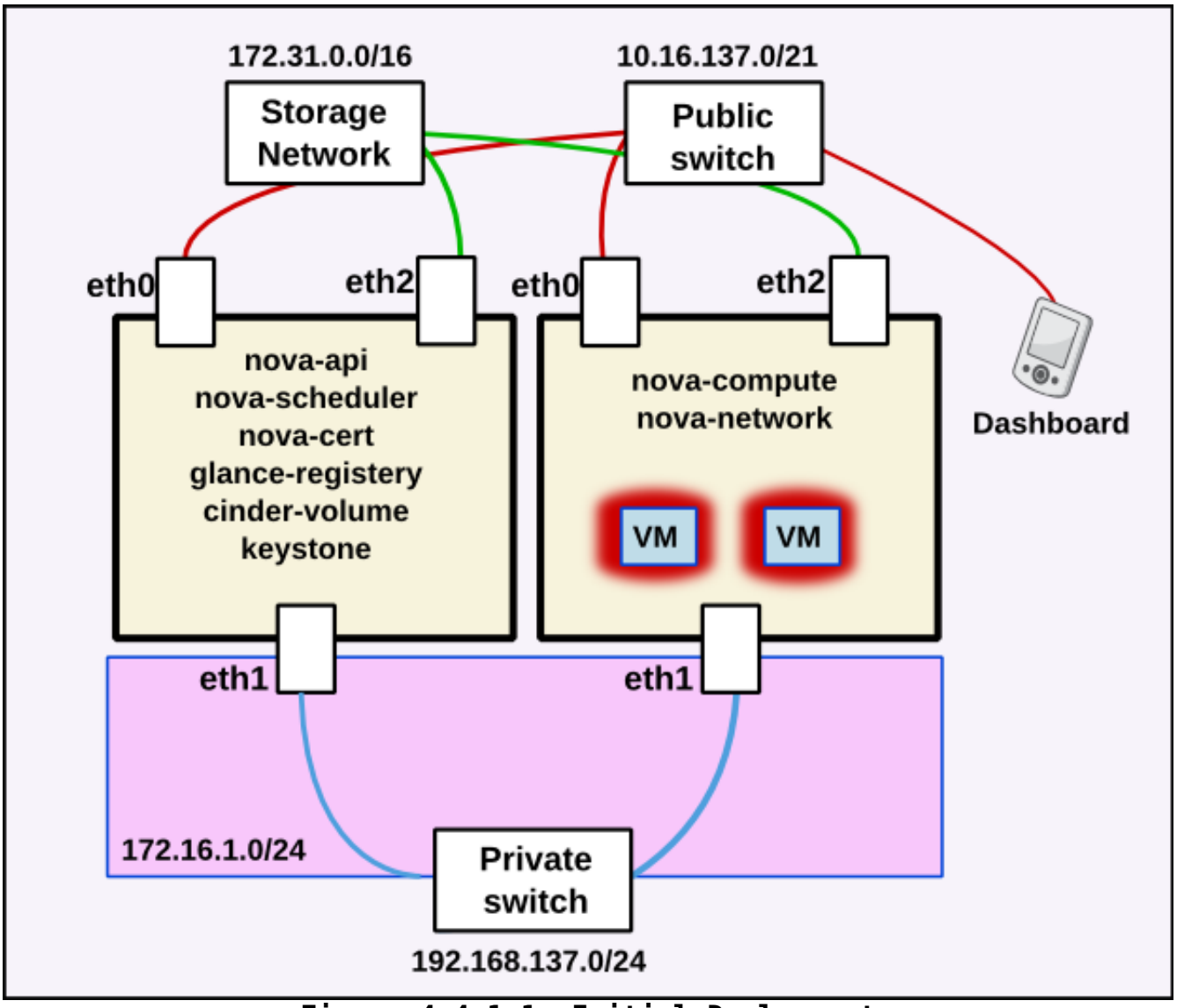


Figure 4.4.1.1: Initial Deployment

4.4.2 Examine Deployment as Keystone Admin

1. Verify the SSH keys were installed successfully.

```
[root@rhos0 ~]# for i in 0 2 6; do ssh rhos$i uptime; done
The authenticity of host 'rhos0 (10.16.137.100)' can't be established.
RSA key fingerprint is 36:74:20:b9:d4:9f:44:e5:58:7b:f5:6f:71:14:a9:bd.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'rhos0,10.16.137.100' (RSA) to the list of known
hosts.
 15:21:50 up  2:44,  1 user,  load average: 0.12, 0.09, 0.08
The authenticity of host 'rhos2 (10.16.137.102)' can't be established.
```

```
RSA key fingerprint is b5:12:0c:1f:1c:7d:99:98:4a:c8:2a:da:d1:dd:64:f3.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'rhos2,10.16.137.102' (RSA) to the list of known
hosts.
 15:21:52 up  2:44,  0 users,  load average: 0.09, 0.04, 0.02
The authenticity of host 'rhos6 (10.16.137.106)' can't be established.
RSA key fingerprint is db:a0:8d:9e:7f:d6:41:9d:13:35:f4:67:cf:99:1e:ab.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'rhos6,10.16.137.106' (RSA) to the list of known
hosts.
 15:21:54 up  2:44,  0 users,  load average: 0.08, 0.02, 0.01
```

2. Run **`openstack-status`** to check status of the non-Nova services installed on the cloud controller.

```
[root@rhos0 ~]# openstack-status
== Glance services ==
openstack-glance-api:         active
openstack-glance-registry:    active
== Keystone service ==
openstack-keystone:           active
== Cinder services ==
openstack-cinder-api:         active
openstack-cinder-scheduler:   active
openstack-cinder-volume:      active
== Support services ==
mysqld:                       active
tgtd:                         active
qpidd:                        active
== Keystone users ==
Warning keystonerc not sourced
```

NOTE: By default **`packstack`** installs the *keystonerc_admin* file on the client. Copy this file to the cloud controller and source it to display additional information.

3. View the status and placement of compute services with **`nova-manage`**.

```
[root@rhos0 ~(keystone_admin)]$ nova-manage service list
Binary           Host              Zone Status     State Updated_At
nova-consoleauth rhos0.example.com nova enabled    :-)   2013-03-14 20:42:41
nova-scheduler   rhos0.example.com nova enabled    :-)   2013-03-14 20:42:31
nova-cert        rhos0.example.com nova enabled    :-)   2013-03-14 20:42:35
nova-network     rhos2.example.com nova enabled    :-)   2013-03-14 20:42:39
nova-compute     rhos2.example.com nova enabled    :-)   2013-03-14 20:42:37
```

4. View the compute network.

```
[root@rhos0 ~(keystone_admin)]$ nova-manage network list
id    IPv4          IPv6 start address DNS1    DNS2 VlanID project uuid
1     172.16.1.0/24 None 172.16.1.2    8.8.4.4 None None   None
929a96b6-8ffc-4c62-b10e-eb0eff76a0ae
```

5. Verify the **ntpd** service is started and uses the correct servers.

```
[root@rhos0 ~]# for i in 0 2 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  8813) is running...
server 10.16.255.2
server 10.16.255.3
ntpd (pid  8537) is running...
server 10.16.255.2
server 10.16.255.3
ntpd (pid  8425) is running...
```

6. Verify that all three servers are subscribed to the appropriate RHN channel.

```
[root@rhos0 ~]# for i in 0 2 6; do ssh rhos$i rhn-channel -l; done
rhel-x86_64-server-6
rhel-x86_64-server-6
rhel-x86_64-server-6
```

4.4.3 Prepare to Boot an Instance

1. Connect to *rhos6* via SSH.

```
[root@rhos0 ~]# ssh rhos6
Last login: Tue Apr 30 16:08:50 2013 from rhos0
Kickstarted on 2013-04-29
```

2. Source */root/keystonerc_admin* to set Keystone environment variables and grant the user administrative privileges.

```
[root@rhos6 ~]# source /root/keystonerc_admin

[root@rhos6 ~(keystone_admin)]$ env | grep OS_
OS_PASSWORD=01d9c64149bb454a
OS_AUTH_URL=http://10.16.137.100:35357/v2.0/
OS_USERNAME=admin
OS_TENANT_NAME=admin
```

3. The Image service stores, indexes, and retrieves virtual machine images that compute nodes can use to build guest instances. Create a Glance image using a **qcow2** image file.

   NOTE: Red Hat Enterprise Linux subscribers can download a 6.4 KVM image: **https://rhn.redhat.com/rhn/software/channel/downloads/Download.do?cid=16952**

```
[root@rhos0 ~(keystone_admin)]$ glance image-create --name rhel-server2 \
--is-public true --disk-format qcow2 \
--container-format bare \
```

```
< /pub/rhel-server-x86_64-kvm-6.4_20130130.0-2-sda.qcow2

+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 5bacdb8ff85d25dbd1ae3db7b0bf57f0     |
| container_format | bare                                 |
| created_at       | 2013-04-03T16:11:43                  |
| deleted          | False                                |
| deleted_at       | None                                 |
| disk_format      | qcow2                                |
| id               | d6a8eca4-2413-4d9b-a689-a95ee1c83b8f |
| is_public        | True                                 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | rhel-server2                         |
| owner            | 89592f995560406a970876db4e6b64a5     |
| protected        | False                                |
| size             | 1974140928                           |
| status           | active                               |
| updated_at       | 2013-04-03T16:12:01                  |
+------------------+--------------------------------------+
```

4. View the image status. It should be **active** when the image build completes successfully.

```
[root@rhos6 ~(keystone_admin)]$ nova image-list
+--------------------------------------+--------------+--------+--------+
| ID                                   | Name         | Status | Server |
+--------------------------------------+--------------+--------+--------+
| cf2986cd-7675-4aa6-899b-7b3c702000c1 | rhel-server2 | ACTIVE |        |
+--------------------------------------+--------------+--------+--------+
```

7. Create a SSH key pair to use with the instances.

```
[root@rhos6 ~(keystone_admin)]$ nova keypair-add oskey > /root/oskey.priv

[root@rhos6 ~(keystone_admin)]$ chmod 600 /root/oskey.priv

[root@rhos6 ~(keystone_admin)]$ nova keypair-list
+-------+-------------------------------------------------+
| Name  | Fingerprint                                     |
+-------+-------------------------------------------------+
| oskey | 6c:86:ec:69:fd:f6:d8:33:a4:ff:ed:eb:1d:a8:71:5e |
+-------+-------------------------------------------------+
```

4.4.4 Boot an Instance

1. *FLAVORS* describe the compute, memory and storage capacity of Nova computing instances. They are hardware profiles for virtual machines. View available flavors with **`nova flavor-list`**.

```
[root@rhos6 ~(keystone_admin)]$ nova flavor-list
+----+-----------+-----------+------+-----------+------+-------------------
```

```
| ID | Name      | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor
| Is_Public | extra_specs |
+----+-----------+-----------+------+-----------+------+-------+-------------
| 1  | m1.tiny   | 512       | 0    | 0 | | 1  | 1.0  | True      | {}
| 2  | m1.small  | 2048      | 20   | 0 | | 1  | 1.0  | True      | {}
| 3  | m1.medium | 4096      | 40   | 0 | | 2  | 1.0  | True      | {}
| 4  | m1.large  | 8192      | 80   | 0 | | 4  | 1.0  | True      | {}
| 5  | m1.xlarge | 16384     | 160  | 0 | | 8  | 1.0  | True      | {}
+-------------------------------------------------------------------------
```

2. Boot an instance based on the **m1.small** flavor using the image and key pair created in the previous steps.

```
[root@rhos6 ~(keystone_admin)]$ nova boot --flavor 2 --image rhel-server2 \
--key_name oskey r2a1
+-------------------------------------+-------------------------------------+
| Property                            | Value
+-------------------------------------+-------------------------------------+
| status                              | BUILD
| updated                             | 2013-04-30T21:18:59Z
| OS-EXT-STS:task_state               | scheduling
| OS-EXT-SRV-ATTR:host                | None
| key_name                            | oskey
| image                               | rhel-server2
| hostId                              |
| OS-EXT-STS:vm_state                 | building
| OS-EXT-SRV-ATTR:instance_name       | instance-00000001
| OS-EXT-SRV-ATTR:hypervisor_hostname | None
| flavor                              | m1.small
| id                                  | 7eef733f-f987-4796-91e5-0471e6eb3d97
| security_groups                     | [{u'name': u'default'}]
| user_id                             | e7158493fcab45c0934e719d2351b9ac
| name                                | r2a1
| adminPass                           | r6xap69RSWHD
| tenant_id                           | ea6900dda1aa42d5a570e9712f57250b
| created                             | 2013-04-30T21:18:59Z
| OS-DCF:diskConfig                   | MANUAL
| accessIPv4                          |
| accessIPv6                          |
| progress                            | 0
| OS-EXT-STS:power_state              | 0
| metadata                            | {}
| config_drive                        |
+-------------------------------------+-------------------------------------+
```

3. Check the instance build status.

```
[root@rhos6 ~(keystone_admin)]$ nova list
+--------------------------------------+------+--------+----------------------+
| ID                                   | Name | Status | Networks
+--------------------------------------+------+--------+----------------------+
| 7eef733f-f987-4796-91e5-0471e6eb3d97 | r2a1 | BUILD  | novanet=172.16.1.2
+--------------------------------------+------+--------+----------------------+
```

4. View the instance after build completes. After a successful build the image status changes from **BUILD** to **ACTIVE**.

```
[root@rhos6 ~(keystone_admin)]$ nova show r2a1
+-------------------------------------+-------------------------------------+
| Property                            | Value
+-------------------------------------+-------------------------------------+
| status                              | ACTIVE
| updated                             | 2013-04-30T21:20:34Z
| OS-EXT-STS:task_state               | None
| OS-EXT-SRV-ATTR:host                | rhos2.example.com
| key_name                            | oskey
| image                               | rhel-server2
(cf2986cd-7675-4aa6-899b-7b3c702000c1)      |
| hostId                              |
7a7b8dcce8ed06d055fcadbc38441b58cc4a2eae630cfd2d2df82552 |
| OS-EXT-STS:vm_state                 | active
| OS-EXT-SRV-ATTR:instance_name       | instance-00000001
|
| OS-EXT-SRV-ATTR:hypervisor_hostname | rhos2.example.com
| flavor                              | m1.small (2)
| id                                  | 7eef733f-f987-4796-91e5-0471e6eb3d97
| security_groups                     | [{u'name': u'default'}]
| user_id                             | e7158493fcab45c0934e719d2351b9ac
| name                                | r2a1
| created                             | 2013-04-30T21:18:59Z
| tenant_id                           | ea6900dda1aa42d5a570e9712f57250b
| OS-DCF:diskConfig                   | MANUAL
| novanetwork network                 | 172.16.1.2
| accessIPv4                          |
| accessIPv6                          |
| progress                            | 0
| OS-EXT-STS:power_state              | 1
| metadata                            | {}
| config_drive                        |
+-------------------------+-------------------------------------------------+
```

5. Connect to the compute node in order to verify the instance. The compute node's bridged private interface acts as a default gateway for the instance.

   NOTE: The **`nova-network`** service assigns the bridge interface on the host compute node with an IP address on the guest's virtual network. The guest's default gateway is set to the bridge interface on the host. Only compute nodes with guests attached to the virtual network are able to access the guests over the network. Assign a floating IP address to the guest to reach it via a public network. Floating IP addresses are described further in section **Error: Reference source not found Error: Reference source not found**.

```
[root@rhos6 ~(keystone_admin)]$ ssh rhos2
The authenticity of host 'rhos2 (10.16.137.102)' can't be established.
```

```
RSA key fingerprint is ca:9c:ea:06:4c:3c:f1:bd:c7:d8:0c:a0:46:84:27:a3.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'rhos2,10.16.137.102' (RSA) to the list of known
hosts.
root@rhos2's password: ********
Kickstarted on 2013-04-29
```

6. View the networking configuration on the **nova-network** server.

```
[root@rhos2 ~]# ip a | grep 172.16.1
    inet 172.16.1.1/24 brd 172.16.1.255 scope global br100

[root@rhos2 ~]# brctl show
bridge name bridge id               STP enabled     interfaces
br100       8000.e0db5574412f       no              eth1  vnet0
virbr0      8000.5254005905b5       yes             virbr0-nic

[root@rhos2 ~]# iptables -L -nv -t nat | grep 172
3   180 ACCEPT     all  --  *      *  172.16.1.0/24        10.16.137.100
1   337 ACCEPT     all  --  *      *  172.16.1.0/24        172.16.1.0/24
! ctstate DNAT
0     0 SNAT       all  --  *       eth0 172.16.1.0/24 0.0.0.0/0
to:10.16.137.102
```

4.4.5 Connect to the Instance via SSH

1. Copy the SSH key from the client to the **nova-network** server and verify its permissions.

```
[root@rhos2 ~]# scp rhos6:/root/oskey.priv .
The authenticity of host 'rhos6 (10.16.137.106)' can't be established.
RSA key fingerprint is d2:98:4a:1f:7a:ee:d1:22:b2:c6:81:48:ea:5d:c9:48.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'rhos6,10.16.137.106' (RSA) to the list of known
hosts.
root@rhos6's password: ********
oskey.priv
100% 1676     1.6KB/s   00:00

[root@rhos2 ~]# ll oskey.priv
-rw-------. 1 root root 1676 Apr 30 16:28 oskey.priv
```

2. Connect to the instance and run **uptime**.

```
[root@rhos2 ~]# ssh -i oskey.priv 172.16.1.2 uptime
The authenticity of host '172.16.1.2 (172.16.1.2)' can't be established.
RSA key fingerprint is 06:e6:66:39:d7:8a:06:33:20:06:99:8c:45:0e:9d:b5.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.16.1.2' (RSA) to the list of known hosts.
 21:29:02 up 8 min,  0 users,  load average: 0.00, 0.01, 0.00
```

3. View the default gateway on the instance. It should be set to the bridge interface address on the **nova-network** server.

```
[root@rhos2 ~]# ssh -i oskey.priv 172.16.1.2 ip route
172.16.1.0/24 dev eth0  proto kernel  scope link  src 172.16.1.2
169.254.0.0/16 dev eth0  scope link  metric 1002
default via 172.16.1.1 dev eth0
```

4.4.6 Connect to the Instance via noVNC Console

1. noVNC is a Virtual Network Client (VNC) server that can be accessed directly via a web browser. OpenStack instances are accessible via noVNC if this feature is enabled in */etc/nova/nova.conf* on the host compute node.

   On the client, generate a URL for the noVNC console. Access it through a web browser.

```
[root@rhos6 ~(keystone_admin)]$ nova get-vnc-console r2a1 novnc
+-------+-----------------------------------------------------------------+
| Type  | Url
+-------+-----------------------------------------------------------------+
| novnc | http://10.16.137.100:6080/vnc_auto.html?token=fc492b01-eb67-4161-
b97a-5e98f9b3c5fe |
+-------+-----------------------------------------------------------------+
```

2. Enter the resulting URL into a browser to open a noVNC console to the instance. **Figure 4.4.6.1: noVNC Console** depicts the initial noVNC console screen.

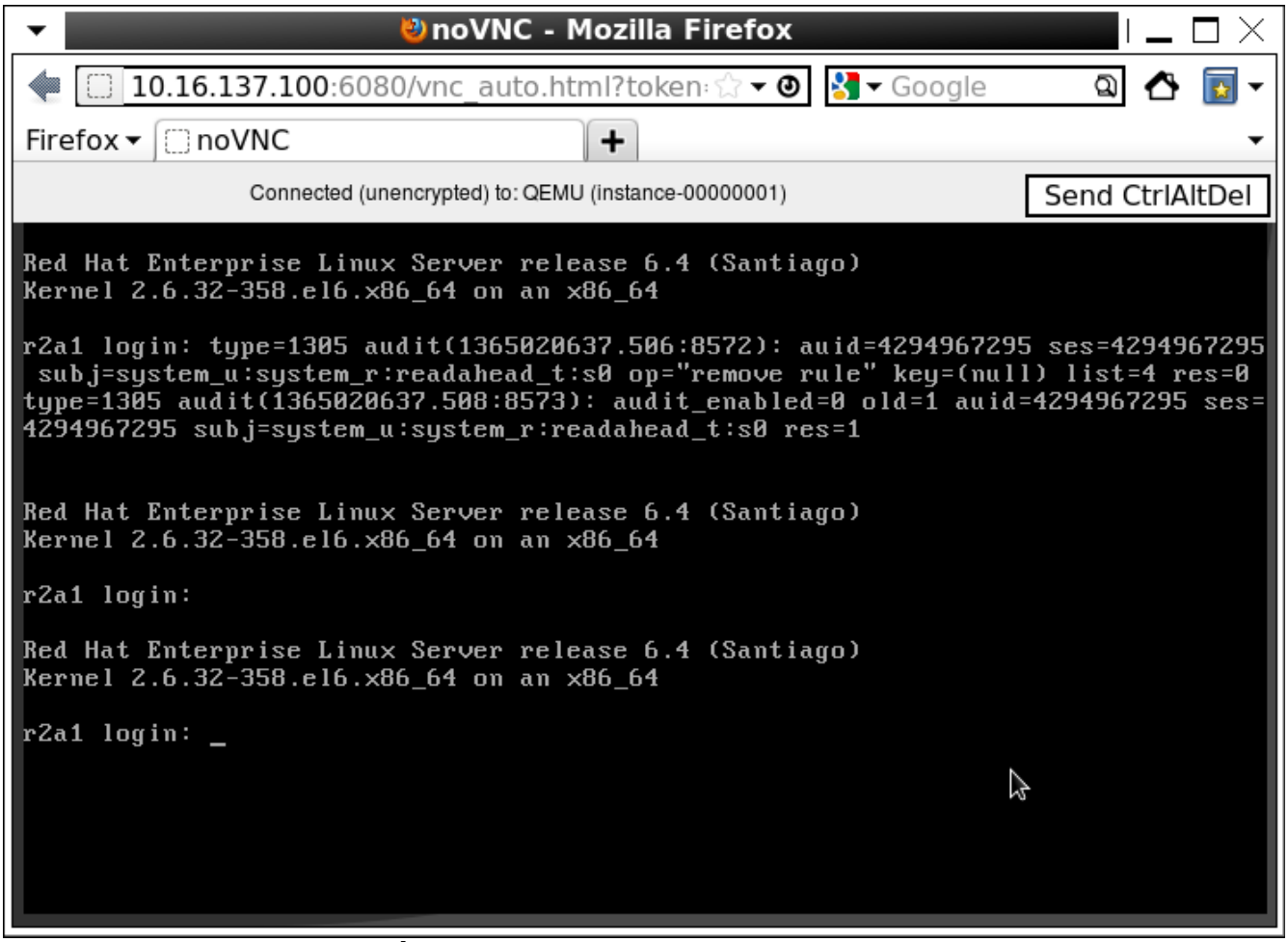


Figure 4.4.6.1: noVNC Console

4.4.7 View Instance from Horizon Dashboard

The Horizon Dashboard is a web-based user interface to OpenStack.

1. Retrieve the OpenStack password from the environment.

```
[root@rhos6 ~(keystone_admin)]$ env | grep PASSWORD
OS_PASSWORD=01d9c64149bb454a
```

2. Sign in to the Horizon Dashboard with the OpenStack administrator account. **Figure 4.4.7.1: OpenStack Dashboard Login** shows the Dashboard sign in screen.

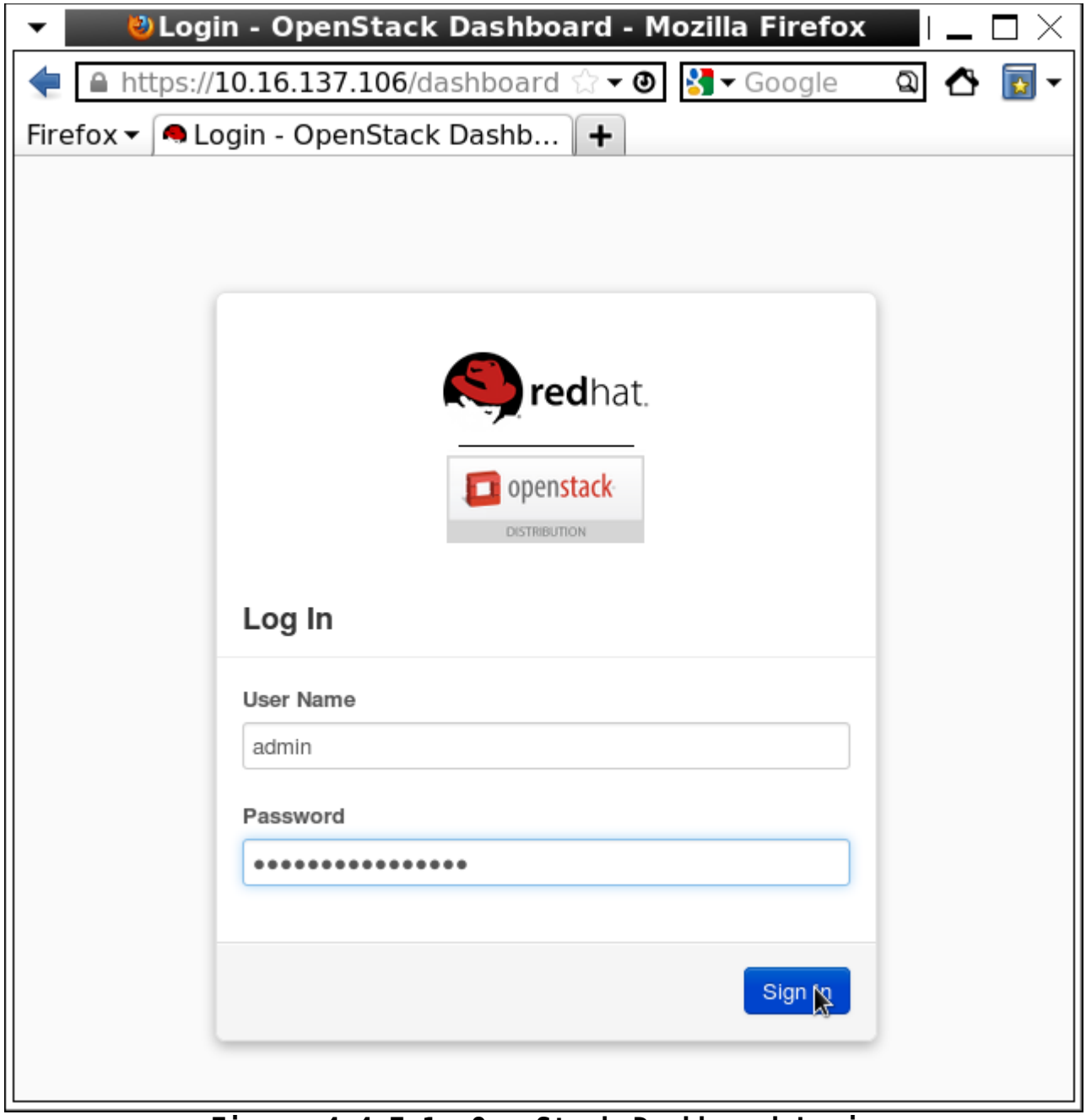


Figure 4.4.7.1: OpenStack Dashboard Login

3. On the **Overview** screen, select **Instances** in the **System Panel** on the left-hand pane, as shown in **Figure 4.4.7.2: OpenStack Dashboard System Panel**.

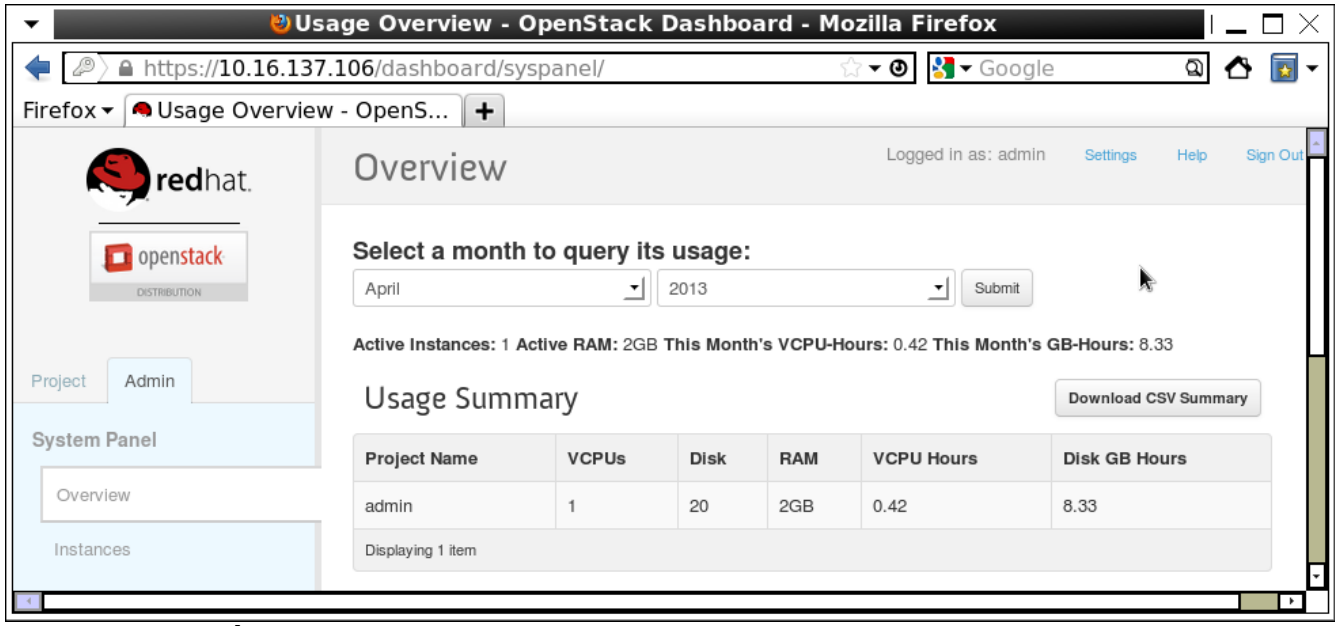


Figure 4.4.7.2: OpenStack Dashboard System Panel

4. Select the instance **r2a1** in the middle of the **Instances** screen. **Figure 4.4.7.3: OpenStack Dashboard Instances** shows this screen.

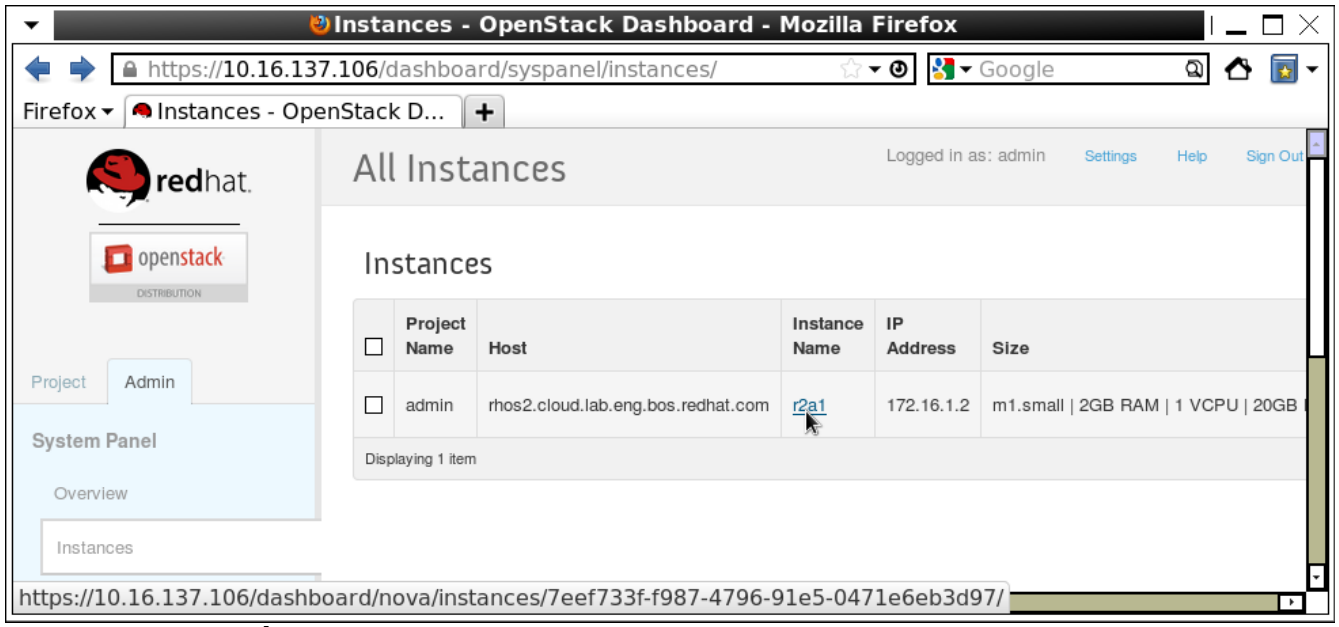


Figure 4.4.7.3: OpenStack Dashboard Instances

5. Select the instance name. Review the instance detail on the **Instance Detail** screen shown in **Figure 4.4.7.4: OpenStack Dashboard Instance Detail**.

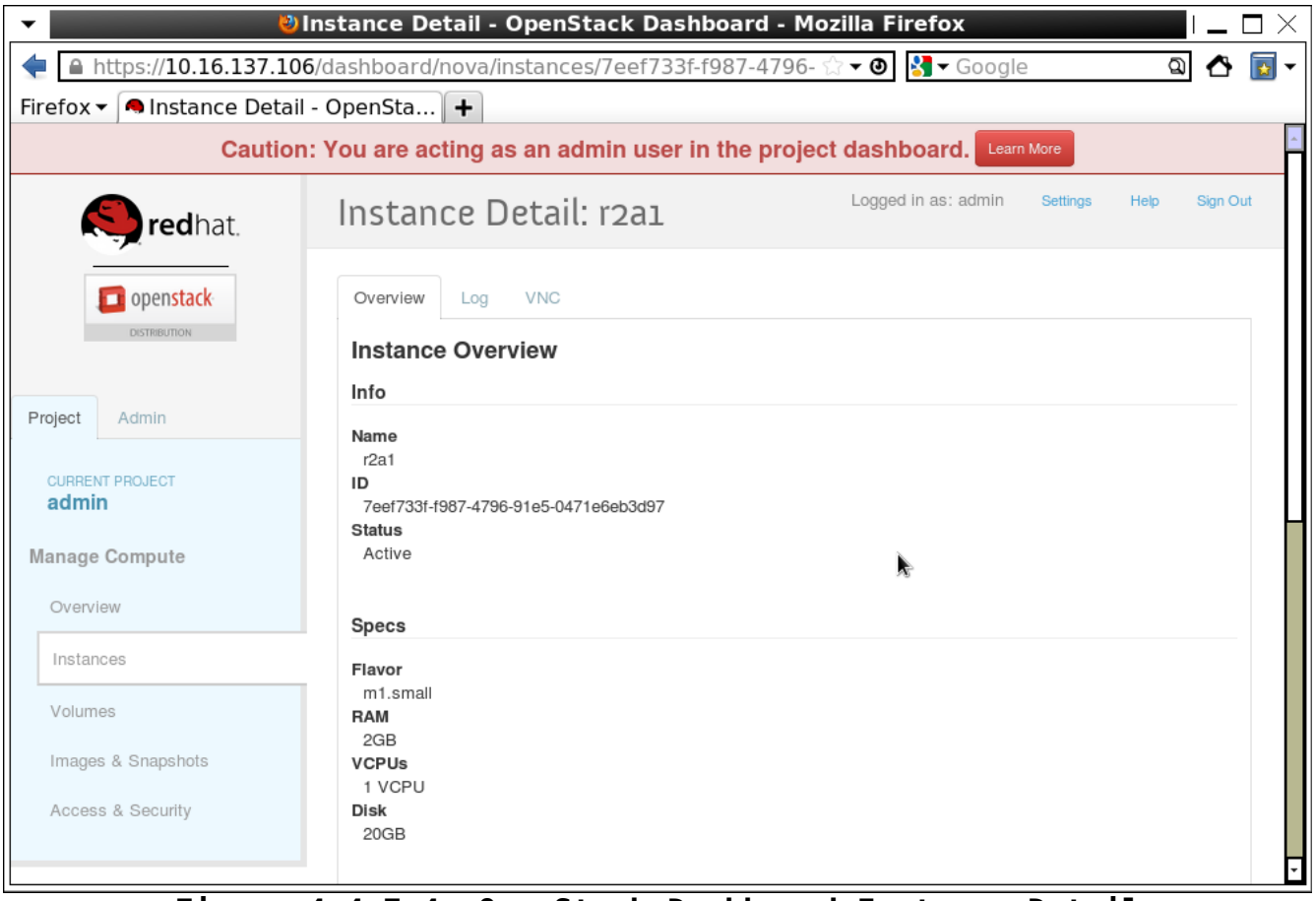


Figure 4.4.7.4: OpenStack Dashboard Instance Detail

5 Expand the Compute Infrastructure

The basic **packstack** deployment is complete and verified. In this section the initial deployment is expanded from one to four compute nodes in order to increase capacity and availability.

Each compute node also acts as an additional **nova-network** server to provide fault tolerance for this critical service.

Finally, this section also implements an NFS-backed Cinder server for persistent volume storage. The cloud controller continues to field Cinder API connections and schedule storage requests. Guest instances on all four compute nodes attach to persistent volumes on a remote NFS server. Moving the persistent volume storage to a dedicated NFS server relieves the cloud controller of the performance overhead associated with fielding network storage access.

5.1 Re-run Packstack to Add Compute Nodes

This section describes the process of re-running **packstack** to modify the initial deployment.

> **NOTE:** Always use the original answer file when re-running **packstack.** Otherwise new passwords are generated for all users and services.

5.1.1 Modify the Packstack Answer File

1. Make a copy of the original **packstack** answer file.

```
[root@rhos0 ~]# cp packstack-answers-20130430-143425.txt packstack-answers-
20130430-143425.txt.orig
```

2. Add the public IP addresses for the three new compute nodes to the answer file.

```
[root@rhos0 ~]#  vim packstack-answers-20130430-143425.txt
```

Here is a snippet from the edited answer file. Changes are in bold.

```
# The IP address of the server on which to install the Nova API
# service
CONFIG_NOVA_API_HOST=10.16.137.100

# The IP address of the server on which to install the Nova Cert
# service
CONFIG_NOVA_CERT_HOST=10.16.137.100

# The IP address of the server on which to install the Nova VNC proxy
CONFIG_NOVA_VNCPROXY_HOST=10.16.137.100

# A comma separated list of IP addresses on which to install the Nova
# Compute services
```

```
CONFIG_NOVA_COMPUTE_HOSTS=10.16.137.102,10.16.137.103,10.16.137.104,10.16.13
7.105

# Private interface for Flat DHCP on the Nova compute servers
CONFIG_NOVA_COMPUTE_PRIVIF=eth1

# The IP address of the server on which to install the Nova Network
# service
CONFIG_NOVA_NETWORK_HOST=10.16.137.102
```

3. Verify the changes using **diff**.

```
[root@rhos0 ~]# diff -y packstack-answers-20130430-143425.txt packstack-
answers-20130430-143425.txt.orig  | grep \|
CONFIG_NOVA_COMPUTE_HOSTS=10.16.137.102,10.16.137.103,10.16.1 |
      CONFIG_NOVA_COMPUTE_HOSTS=10.16.137.102
```

5.1.2 Re-run Packstack with the New Answer File

1. Re-run **packstack** with the modified answer file.

```
[root@rhos0 ~]# packstack --answer-file=packstack-answers-20130430-
143425.txt
Welcome to Installer setup utility
```

2. Enter the root passwords for the new compute nodes when prompted.

```
Installing:
Clean Up...                                             [ DONE ]
Setting up ssh keys...root@10.16.137.103's password: ********
root@10.16.137.104's password: ********
root@10.16.137.105's password: ********
                                     [ DONE ]
```

3. Verify the installation completes successfully.

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


Additional information:
 * Did not create a cinder volume group, one already existed
 * To use the command line tools you need to source the file
/root/keystonerc_admin created on 10.16.137.106
 * NOTE : A certificate was generated to be used for ssl, You should change
the ssl certificate configured in /etc/httpd/conf.d/ssl.conf on
10.16.137.106 to use a CA signed cert.
 * To use the console, browse to https://10.16.137.106/dashboard
 * The installation log file is available at: /var/tmp/packstack/20130501-
123750-bJ5w7Q/openstack-setup.log
```

5.1.3 Verify the New Compute Nodes

Verify the three new **nova-compute** service instances are seen in **nova-manage**. This means they are communicating via **messagebus** and available to field service requests.

```
[root@rhos0 ~]# nova-manage service list
Binary           Host                 Zone  Status   State Updated_At
nova-cert        rhos0.example.com    nova  enabled  :-)   2013-04-03 17:00:38
nova-scheduler   rhos0.example.com    nova  enabled  :-)   2013-04-03 17:00:37
nova-consoleauth rhos0.example.com    nova  enabled  :-)   2013-04-03 17:00:37
nova-compute     rhos2.example.com    nova  enabled  :-)   2013-04-03 17:00:38
nova-network     rhos2.example.com    nova  enabled  :-)   2013-04-03 17:00:38
nova-compute     rhos4.example.com    nova  enabled  :-)   2013-04-03 17:00:37
nova-compute     rhos3.example.com    nova  enabled  :-)   2013-04-03 17:00:41
nova-compute     rhos5.example.com    nova  enabled  :-)   2013-04-03 17:00:43
```

5.2 Create the Tenant, Users, and Roles

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

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 *refarch* tenant defines a hypothetical organization. A user named *refarch_user* is granted user-level access to the tenant.

5.2.1 Create a Tenant

1. Define a new tenant called *refarch*.

```
[root@rhos6 ~(keystone_admin)]$ keystone tenant-create --name refarch
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |                                  |
|   enabled   |               True               |
|      id     | b1c1febbb08a422d89357349c71e1543 |
|     name    |             refarch              |
+-------------+----------------------------------+
```

2. Verify the new tenant is seen by Keystone.

```
[root@rhos6 ~(keystone_admin)]$ keystone tenant-list
+----------------------------------+----------+---------+
|                id                |   name   | enabled |
+----------------------------------+----------+---------+
| 462913efebf04fce9f54fa2bd4030715 | services |   True  |
| 4a958724b956481c9de42ddae7d0d7f1 |  admin   |   True  |
| b1c1febbb08a422d89357349c71e1543 | refarch  |   True  |
+----------------------------------+----------+---------+
```

5.2.2 Add a User

1. Create a user in the *refarch* tenant named *refarch_user*.

```
[root@rhos6 ~(keystone_admin)]$ keystone user-create --name refarch_user \
--tenant-id b1c1febbb08a422d89357349c71e1543 --pass password –email \
refarch_user@example.com
+----------+----------------------------------------------------------------+
| Property | Value
+----------+----------------------------------------------------------------+
|  email   | refarch_user@example.com
| enabled  | True
|    id    | 043914f4ff124305a8ac0389aeb02d35
|   name   | refarch_user
| password |
$6$rounds=40000$d9R4asobKc1ePMGy$rTcZH77E681JS77ZVgN.IIdJjL5MU5BFIsjNOddFtPM
GXVOXKdidfCpMisP/FlRFdJTxLJJMk6XnBSZaOfC2K/ |
| tenantId | b1c1febbb08a422d89357349c71e1543
+----------+----------------------------------------------------------------+
```

2. View the new user.

```
[root@rhos6 ~(keystone_admin)]$ keystone user-list
+----------------------------------+--------------+---------+--------------+
|                id                |     name     | enabled |    email
+----------------------------------+--------------+---------+--------------+
| 043914f4ff124305a8ac0389aeb02d35 | refarch_user | True  |
refarch_user@example.com
| 35ec00c862e74ea19b02f8d501cb6e29 |   nova   |  True  | nova@localhost
| 52a4c1b335b04f62bc15faec83ab62ee |  admin   |  True  | test@test.com
| ad648000e5cd4741bbc7149451426fcc |  glance  |  True  | glance@localhost
| b6eb42cf1a514870aa4ba951f77b0647 |  cinder  |  True  | cinder@localhost
+----------------------------------+--------------+---------+--------------+
```

5.2.3 Add a Role

1. Create a new role named *refarch_user_role*.

```
[root@rhos6 ~(keystone_admin)]$ keystone role-create --name
refarch_user_role
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|    id    | 77c3220b791644b18535f54cad6d71b6 |
|   name   |        refarch_user_role         |
+----------+----------------------------------+
```

2. View the role.

```
[root@rhos6 ~(keystone_admin)]$ keystone role-list
+----------------------------------+------------------+
|                id                |       name       |
+----------------------------------+------------------+
| 77c3220b791644b18535f54cad6d71b6 | refarch_user_role |
| 920b6a06ac064e9398b00e66cb74c330 |       Member       |
```

```
| fbd1e47f6515462aafa46f5c1dc445e5 |       admin       |
+----------------------------------+-------------------+
```

5.2.4 Associate the User, Role, and Tenant

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

```
[root@rhos6 ~(keystone_admin)]$ keystone user-role-add \
--user-id=043914f4ff124305a8ac0389aeb02d35 \
--role-id=77c3220b791644b18535f54cad6d71b6 \
--tenant-id=b1c1febbb08a422d89357349c71e1543
```

> **NOTE:** The **`user-role-list`** command does not return correct output. See **BZ905541** for details.

2. Check the state database to verify user-role mapping on the cloud controller.

```
[root@rhos0 ~]# mysql -u root -p keystone -e "select * from metadata where
user_id='043914f4ff124305a8ac0389aeb02d35';"
Enter password: ********
+----------------------------------+----------------------------------+----+
| user_id                          | tenant_id                        | data
+----------------------------------+----------------------------------+----+
| 043914f4ff124305a8ac0389aeb02d35 | b1c1febbb08a422d89357349c71e1543 |
{"roles": ["77c3220b791644b18535f54cad6d71b6"]} |
+----------------------------------+----------------------------------+----+
```

5.3 Configure Nova Networking for Multi-host Mode

In the current configuration guest instances deployed on any of the four compute nodes use *rhos2* as their **`nova-network`** server. These instances use the bridge interface on *rhos2* as a default gateway to outside networks.

This configuration has several drawbacks. First, all routed traffic from instances traverses *rhos2* which introduces an extra hop for guests on *rhos3-5*. It also increases the network utilization on *rhos2*. This configuration also creates a single point of failure for the redundant **`nova-compute`** servers. If the **`nova-network`** server fails traffic from all of the instances stops.

In this section the **`nova-network`** service is converted to *multi_host* mode. Each compute node acts as its own default gateway for local instances.

5.3.1 Disable Default Virtual Networks

1. Disable the default virtual interfaces on all compute nodes from the cloud controller.

```
[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i "virsh net-destroy default
&& virsh net-autostart default --disable"; done
Network default destroyed

Network default unmarked as autostarted
```

```
Network default destroyed

Network default unmarked as autostarted

Network default destroyed

Network default unmarked as autostarted

Network default destroyed

Network default unmarked as autostarted
```

2. Verify the default virtual network is disabled on all compute nodes.

```
[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i "virsh net-list --all"; done
Name                 State      Autostart     Persistent
----------------------------------------------------------
default              inactive   no            yes
Name                 State      Autostart     Persistent
----------------------------------------------------------
default              inactive   no            yes
Name                 State      Autostart     Persistent
----------------------------------------------------------
default              inactive   no            yes
Name                 State      Autostart     Persistent
----------------------------------------------------------
default              inactive   no            yes
```

5.3.2 Install Networking Services on the Compute Nodes

1. Install the additional Nova services on the compute nodes.

```
[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i yum install -q -y openstack-
nova-network openstack-nova-api; done
```

NOTE: The **`openstack-nova-api`** package provides the **`nova-metadata-api`** service. **5.3.4 Customize Each Compute Node** describes this service.

2. Verify the same packages are installed on all nodes.

```
[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i "rpm -qa | grep -E 'nova-
api|nova-network'"; done
openstack-nova-network-2012.2.3-9.el6ost.noarch
openstack-nova-api-2012.2.3-9.el6ost.noarch
openstack-nova-network-2012.2.3-9.el6ost.noarch
openstack-nova-api-2012.2.3-9.el6ost.noarch
openstack-nova-network-2012.2.3-9.el6ost.noarch
openstack-nova-api-2012.2.3-9.el6ost.noarch
openstack-nova-network-2012.2.3-9.el6ost.noarch
openstack-nova-api-2012.2.3-9.el6ost.noarch
```

5.3.3 Modify Nova Configuration

The initial changes are made on *rhos2*. Copy */etc/nova/nova.conf* to the new compute nodes to propagate the changes.

1. Set ***multi_host = true*** in */etc/nova/nova.conf*.

```
[root@rhos0 ~]# ssh rhos2 “openstack-config --set /etc/nova/nova.conf DEFAULT
multi_host True”

[root@rhos0 ~(keystone_admin)]$ ssh rhos2 grep multi_host /etc/nova/nova.conf

# Default value for multi_host in networks (boolean value)
#multi_host=false
multi_host = True
```

2. Change the ***fixed_range*** in */etc/nova/nova.conf*. This creates a new network address range that can be assigned to the *refarch* tenant.

```
[root@rhos0 ~]# ssh rhos2 “openstack-config --set /etc/nova/nova.conf DEFAULT
fixed_range 172.16.2.0/24”

[root@rhos0 ~(keystone_admin)]$ ssh rhos2 “grep fixed_range
/etc/nova/nova.conf”
#fixed_range=10.0.0.0/8
fixed_range=172.16.2.0/24
```

3. Copy */etc/nova/nova.conf* from *rhos2* to the other compute nodes.

```
[root@rhos0 ~]# for i in 3 4 5; do scp rhos2:/etc/nova/nova.conf
rhos$i:/etc/nova/nova.conf; done
The authenticity of host 'rhos3 (10.16.137.103)' can't be established.
RSA key fingerprint is e7:25:29:3b:90:a7:8e:88:4b:c5:7d:b7:51:75:cd:27.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'rhos3,10.16.137.103' (RSA) to the list of known
hosts.
root@rhos3's password: ********
nova.conf
100%   61KB  60.7KB/s   00:00
Connection to rhos2 closed.
The authenticity of host 'rhos4 (10.16.137.104)' can't be established.
RSA key fingerprint is b0:bb:5f:8d:05:1e:13:e7:f1:5d:1e:8e:7e:33:a1:24.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'rhos4,10.16.137.104' (RSA) to the list of known
hosts.
root@rhos4's password: ********
nova.conf
100%   61KB  60.7KB/s   00:00
Connection to rhos2 closed.
The authenticity of host 'rhos5 (10.16.137.105)' can't be established.
RSA key fingerprint is 90:3b:c0:97:94:67:61:e2:38:c7:78:d9:01:8a:67:b4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'rhos5,10.16.137.105' (RSA) to the list of known
hosts.
```

```
root@rhos5's password: ********
nova.conf
100%   61KB  60.7KB/s   00:00
Connection to rhos2 closed.
```

5.3.4 Customize Each Compute Node

Two additional customizations should be made beyond configuring each node as a **nova-network** server.

- Configure each node to provide VNC access to its guest instances.
- Configure each node as a **metadata-api** server. This allows guest instances to retrieve instance-specific data from their local hosts during boot.

1. Customize each compute node as its own network and VNC server in */etc/nova/nova.conf*.

```
[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i sed -i
"s/network_host=10.16.137.102/network_host=10.16.137.10$i/"
/etc/nova/nova.conf; done

[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i sed -i
"s/vncserver_listen=10.16.137.102/vncserver_listen=10.16.137.10$i/"
/etc/nova/nova.conf; done

[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i sed -i
"s/vncserver_proxyclient_address=10.16.137.102/vncserver_proxyclient_address
=10.16.137.10$i/" /etc/nova/nova.conf; done

[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i grep network_host
/etc/nova/nova.conf; done
#network_host=nova
network_host=10.16.137.102
#network_host=nova
network_host=10.16.137.103
#network_host=nova
network_host=10.16.137.104
#network_host=nova
network_host=10.16.137.105
```

2. Configure */etc/nova/nova.conf* so each compute node is its own **metadata-api** server.

```
[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i "openstack-config --set
/etc/nova/nova.conf DEFAULT metadata_listen 0.0.0.0"; done

[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i "openstack-config --set
/etc/nova/nova.conf DEFAULT metadata_manager
nova.api.manager.MetadataManager"; done

[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i "openstack-config --set
/etc/nova/nova.conf DEFAULT metadata_listen_port 8775"; done
```

3. Configure each compute node to provide its own metadata.

```
[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i "openstack-config --set
/etc/nova/nova.conf DEFAULT metadata_host 10.16.137.10$i"; done
```

4. Verify that each compute node is configured to provide its own **metadata-api** service.

```
[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i "grep metadata
/etc/nova/nova.conf | grep -v \#"; done
metadata_host=10.16.137.102
metadata_manager = nova.api.manager.MetadataManager
metadata_listen = 0.0.0.0
metadata_listen_port = 8775
metadata_host=10.16.137.103
metadata_listen = 0.0.0.0
metadata_manager = nova.api.manager.MetadataManager
metadata_listen_port = 8775
metadata_host=10.16.137.104
metadata_manager=nova.api.manager.MetadataManager
metadata_listen=0.0.0.0
metadata_listen_port=8775
metadata_host=10.16.137.105
metadata_listen = 0.0.0.0
metadata_manager = nova.api.manager.MetadataManager
metadata_listen_port = 8775
```

5. Restart services for changes to take effect.

```
[root@rhos0 ~]#  for i in 2 3 4 5; do ssh rhos$i 'for j in compute network
metadata-api; do service openstack-nova-$j restart; done'; done
Stopping openstack-nova-compute: [  OK  ]
Starting openstack-nova-compute: [  OK  ]
Stopping openstack-nova-network: [  OK  ]
Starting openstack-nova-network: [  OK  ]
Stopping openstack-nova-metadata-api: [FAILED]
Starting openstack-nova-metadata-api: [  OK  ]
Stopping openstack-nova-compute: [  OK  ]
Starting openstack-nova-compute: [  OK  ]
Stopping openstack-nova-network: [FAILED]
Starting openstack-nova-network: [  OK  ]
Stopping openstack-nova-metadata-api: [FAILED]
Starting openstack-nova-metadata-api: [  OK  ]
Stopping openstack-nova-compute: [  OK  ]
Starting openstack-nova-compute: [  OK  ]
Stopping openstack-nova-network: [FAILED]
Starting openstack-nova-network: [  OK  ]
Stopping openstack-nova-metadata-api: [FAILED]
Starting openstack-nova-metadata-api: [  OK  ]
Stopping openstack-nova-compute: [  OK  ]
Starting openstack-nova-compute: [  OK  ]
Stopping openstack-nova-network: [FAILED]
Starting openstack-nova-network: [  OK  ]
Stopping openstack-nova-metadata-api: [FAILED]
```

```
Starting openstack-nova-metadata-api: [  OK  ]
```

6. Start services at boot.

```
[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i chkconfig openstack-nova-
metadata-api on; done

[root@rhos0 ~]# for i in 2 3 4 5; do ssh rhos$i chkconfig openstack-nova-
network on; done
```

7. Verify the new network configuration.

```
[root@rhos0 ~]# nova-manage service list
Binary           Host                 Zone Status  State Updated_At
nova-cert        rhos0.example.com    nova enabled  :-)   2013-04-03 20:44:53
nova-scheduler   rhos0.example.com    nova enabled  :-)   2013-04-03 20:44:53
nova-consoleauth rhos0.example.com    nova enabled  :-)   2013-04-03 20:44:52
nova-compute     rhos2.example.com    nova enabled  :-)   2013-04-03 20:44:56
nova-network     rhos2.example.com    nova enabled  :-)   2013-04-03 20:44:50
nova-compute     rhos4.example.com    nova enabled  :-)   2013-04-03 20:44:52
nova-compute     rhos3.example.com    nova enabled  :-)   2013-04-03 20:44:50
nova-compute     rhos5.example.com    nova enabled  :-)   2013-04-03 20:44:54
nova-network     rhos3.example.com    nova enabled  :-)   2013-04-03 20:44:55
nova-network     rhos4.example.com    nova enabled  :-)   2013-04-03 20:44:56
nova-network     rhos5.example.com    nova enabled  :-)   2013-04-03 20:44:58
```

5.4 Boot an Instance in the Tenant

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

5.4.1 Add a Network to the Tenant

Make a network in the *refarch* tenant. Use the new *fixed_range* specified in */etc/nova/nova.conf* and the tenant ID from **`keystone tenant-list`**.

1. Get the tenant id.

```
[root@rhos6 ~(keystone_admin)]$ keystone tenant-list
+----------------------------------+----------+---------+
|                id                |   name   | enabled |
+----------------------------------+----------+---------+
| 462913efebf04fce9f54fa2bd4030715 | services |   True  |
| 4a958724b956481c9de42ddae7d0d7f1 |  admin   |   True  |
| b1c1febbb08a422d89357349c71e1543 | refarch  |   True  |
+----------------------------------+----------+---------+
```

2. Create a network in the tenant specifying the new fixed address range.

NOTE: Create the new network on the cloud controller. The **project_id** is the tenant id.

```
[root@rhos0 ~]# nova-manage network create refnet 172.16.2.0/24 1 256 \
--bridge=br100 --bridge_interface=eth1 --multi_host=T \
--project_id b1c1febbb08a422d89357349c71e1543
```

3. Display the new network.

```
[root@rhos0 ~]# nova-manage network list
id IPv4  IPv6 start address DNS1 DNS2 VlanID project  uuid
1  172.16.1.0/24 None 172.16.1.2 8.8.4.4 None None None d6351b4c-962b-4c3f-
bd1c-ca4e6cdbfb76
2 172.16.2.0/24 None 172.16.2.2 8.8.4.4 None None
b1c1febbb08a422d89357349c71e1543  1826f214-63e8-45b7-bd82-f1c7bb7dffe3
```

5.4.2 Boot a New Instance as the Tenant User

1. Create a Keystone file for *refarch_user* on the client named *keystonerc_ra*.

```
export OS_USERNAME=refarch_user
export OS_TENANT_NAME=refarch
export OS_PASSWORD=password
export OS_AUTH_URL=http://10.16.137.100:35357/v2.0/
export PS1="[\u@\h \W(refarch_user)]$ "
```

2. Source it.

```
[root@rhos6 ~]# source keystonerc_ra
```

3. Verify the environment variables are exported.

```
[root@rhos6 ~(refarch_user)]$ env | grep OS_
OS_PASSWORD=password
OS_AUTH_URL=http://10.16.137.100:35357/v2.0/
OS_USERNAME=refarch_user
OS_TENANT_NAME=refarch
```

4. Create a key pair as the *refarch_user*.

```
[root@rhos6 ~(refarch_user)]$ nova keypair-add rakey > /root/rakey.priv

[root@rhos6 ~(refarch_user)]$ chmod 600 rakey.priv

[root@rhos6 ~(refarch_user)]$ ll rakey.priv
-rw-------. 1 root root 1676 Apr  3 16:23 rakey.priv
```

5. Boot an instance as *refarch_user* specifying the new key and network. The *–nic net-id* parameter forces the instance to use a virtual interface on the *refarch* network.

```
[root@rhos6 ~(refarch_user)]$ nova boot --flavor 2 --key_name rakey \
--image rhel-server2 \
--nic net-id=1826f214-63e8-45b7-bd82-f1c7bb7dffe3 ra1
+------------------------+--------------------------------------+
| Property               | Value                                |
```

```
+-------------------------+--------------------------------------+
| status                  | BUILD                                |
| updated                 | 2013-05-01T18:43:48Z                 |
| OS-EXT-STS:task_state   | scheduling                           |
| key_name                | rakey                                |
| image                   | rhel-server2                         |
| hostId                  |                                      |
| OS-EXT-STS:vm_state     | building                             |
| flavor                  | m1.small                             |
| id                      | 54ff0d3f-443f-47e6-9ba5-27ce6a77aee7 |
| security_groups         | [{u'name': u'default'}]              |
| user_id                 | 043914f4ff124305a8ac0389aeb02d35     |
| name                    | ra1                                  |
| adminPass               | PYcPb6zb9ooD                         |
| tenant_id               | b1c1febbb08a422d89357349c71e1543     |
| created                 | 2013-05-01T18:43:48Z                 |
| OS-DCF:diskConfig       | MANUAL                               |
| accessIPv4              |                                      |
| accessIPv6              |                                      |
| progress                | 0                                    |
| OS-EXT-STS:power_state  | 0                                    |
| metadata                | {}                                   |
| config_drive            |                                      |
```

8. Verify the new instance is in an **ACTIVE** state.

```
[root@rhos6 ~(refarch_user)]$ nova list
+--------------------------------------+------+--------+-------------------+
| ID                                   | Name | Status | Networks          |
+--------------------------------------+------+--------+-------------------+
| 54ff0d3f-443f-47e6-9ba5-27ce6a77aee7 | ra1  | ACTIVE | refnet=172.16.2.2 |
+--------------------------------------+------+--------+-------------------+
```

5.4.3 Assign a Floating IP Address to the Instance

1. A *floating IP address* refers to a IP address that can be dynamically assigned to a running instance. Floating IP addresses are frequently public.

   On the cloud controller list available floating IP addresses. These were created during the initial **packstack** deployment.

```
[root@rhos0 ~]# nova-manage floating list
None  10.16.143.109    None  nova  eth0
None  10.16.143.110    None  nova  eth0
```

2. Create a floating IP address on the client as the *refarch* user.

```
[root@rhos6 ~(refarch_user)]$ nova floating-ip-create
+---------------+-------------+----------+------+
| Ip            | Instance Id | Fixed Ip | Pool |
+---------------+-------------+----------+------+
| 10.16.143.109 | None        | None     | nova |
+---------------+-------------+----------+------+
```

3. Add the floating IP address to the instance.

```
[root@rhos6 ~(refarch_user)]$ nova add-floating-ip ra1 10.16.143.109

[root@rhos6 ~(refarch_user)]$ nova floating-ip-list
+---------------+--------------------------------------+------------+------+
| Ip            | Instance Id                          | Fixed Ip   | Pool |
+---------------+--------------------------------------+------------+------+
| 10.16.143.109 | 54ff0d3f-443f-47e6-9ba5-27ce6a77aee7 | 172.16.2.2 | nova |
+---------------+--------------------------------------+------------+------+
```

4. A *SECURITY GROUP* is a named collection of network access rules. They specify which incoming network traffic should be delivered to all instances in the group. All other incoming traffic is discarded.

   Add a rule allowing SSH traffic from the public network.

```
[root@rhos6 ~(refarch_user)]$ nova secgroup-add-rule default tcp 22 22
10.16.137.0/24
+-------------+-----------+---------+----------------+--------------+
| IP Protocol | From Port | To Port | IP Range       | Source Group |
+-------------+-----------+---------+----------------+--------------+
| tcp         | 22        | 22      | 10.16.137.0/24 |              |
+-------------+-----------+---------+----------------+--------------+
```

5. View the instance from the cloud controller to determine its host compute node. This compute node has an address on the **refnet** network.

5.4.4 Connect to the Instance via SSH

1. Connect to the instance via SSH over the floating IP address.

```
[root@rhos6 ~(refarch_user)]$ ssh -i rakey.priv 10.16.143.109 uptime
The authenticity of host '10.16.143.109 (10.16.143.109)' can't be established.

RSA key fingerprint is cd:03:d0:8b:e0:c4:91:3d:46:ef:9e:36:ee:2f:3a:ba.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.16.143.109' (RSA) to the list of known hosts.
 19:59:39 up  1:14,  0 users,  load average: 0.01, 0.01, 0.00
```

2. Verify its default gateway is set to the bridge interface on *refnet*.

```
[root@rhos6 ~(refarch_user)]$ ssh -i rakey.priv 10.16.143.109 ip route
172.16.2.0/24 dev eth0  proto kernel  scope link  src 172.16.2.2
169.254.0.0/16 dev eth0  scope link  metric 1002
default via 172.16.2.3 dev eth0
```

5.5 Test Nova Service Availability

Fault tolerance is one advantage of OpenStack's scale out architecture. Redundant **`nova-network`** and **`nova-compute`** servers eliminates the compute node as a single point of failure.

Temporarily disable the Nova services on the original compute node. Boot another instance and verify that it is accessible and routable.

5.5.1 Disable Nova Services on the Original Compute Node

1. From the cloud controller disable the **`nova-compute`** and **`nova-network`** services on on the original compute node. In this example it is *rhos2*.

```
[root@rhos0 ~]# nova-manage service disable \ --host=rhos2.example.com
--service=nova-compute

[root@rhos0 ~]# nova-manage service disable \ --host=rhos2.example.com
--service=nova-network
```

2. Verify that the services are disabled with **`nova-manage service list`**.

```
[root@rhos0 ~]# nova-manage service list
Binary           Host                 Zone  Status   State Updated_At
nova-cert        rhos0.example.com    nova  enabled  :-)   2013-05-01 20:04:37
nova-scheduler   rhos0.example.com    nova  enabled  :-)   2013-05-01 20:04:38
nova-consoleauth rhos0.example.com    nova  enabled  :-)   2013-05-01 20:04:37
nova-compute     rhos2.example.com    nova  disabled :-)   2013-05-01 20:04:37
nova-network     rhos2.example.com    nova  disabled :-)   2013-05-01 20:04:38
nova-compute     rhos5.example.com    nova  enabled  :-)   2013-05-01 20:04:33
nova-compute     rhos3.example.com    nova  enabled  :-)   2013-05-01 20:04:31
nova-compute     rhos4.example.com    nova  enabled  :-)   2013-05-01 20:04:32
nova-network     rhos3.example.com    nova  enabled  :-)   2013-05-01 20:04:33
nova-network     rhos4.example.com    nova  enabled  :-)   2013-05-01 20:04:37
nova-network     rhos5.example.com    nova  enabled  :-)   2013-05-01 20:04:38
```

5.5.2 Boot a Test Instance

1. From the client boot a new instance as *refarch_user*.

```
[root@rhos6 ~(refarch_user)]$ nova boot --flavor 2 --key_name rakey --image
rhel-server2 --nic net-id=1826f214-63e8-45b7-bd82-f1c7bb7dffe3 ra2
+------------------------+--------------------------------------+
| Property               | Value                                |
+------------------------+--------------------------------------+
| status                 | BUILD                                |
| updated                | 2013-05-01T20:09:20Z                 |
| OS-EXT-STS:task_state  | scheduling                           |
| key_name               | rakey                                |
| image                  | rhel-server2                         |
| hostId                 |                                      |
| OS-EXT-STS:vm_state    | building                             |
| flavor                 | m1.small                             |
| id                     | c0e9f13d-18e0-4f7e-aa38-c8734d5d77f7 |
| security_groups        | [{u'name': u'default'}]              |
| user_id                | 043914f4ff124305a8ac0389aeb02d35     |
| name                   | ra2                                  |
| adminPass              | iRyU7LdDK3W3                         |
| tenant_id              | b1c1febbb08a422d89357349c71e1543     |
| created                | 2013-05-01T20:09:20Z                 |
```

```
| OS-DCF:diskConfig      | MANUAL                                 |
| accessIPv4             |                                        |
| accessIPv6             |                                        |
| progress               | 0                                      |
| OS-EXT-STS:power_state | 0                                      |
| metadata               | {}                                     |
| config_drive           |                                        |
```

2. Verify the new instance boots to **ACTIVE** state.

```
[root@rhos6 ~(refarch_user)]$ nova list
+--------------------------------------+------+--------+------------------+
| ID                                   | Name | Status | Networks
+--------------------------------------+------+--------+------------------+
| 54ff0d3f-443f-47e6-9ba5-27ce6a77aee7 | ra1  | ACTIVE | refnet=172.16.2.2,
10.16.143.109 |
| c0e9f13d-18e0-4f7e-aa38-c8734d5d77f7 | ra2  | ACTIVE | refnet=172.16.2.4
+--------------------------------------+------+--------+------------------+
```

3. Copy the SSH key to the original instance.

```
[root@rhos6 ~(refarch_user)]$ scp -i rakey.priv rakey.priv
10.16.143.109:/root
rakey.priv
100% 1676     1.6KB/s   00:00
```

4. Connect to the new instance from the original instance to verify network connectivity. This demonstrates Nova fault tolerance in *multi-host* mode.

```
[root@rhos6 ~(refarch_user)]$ ssh -i rakey.priv 10.16.143.109

[root@ra1 ~]# ssh -i rakey.priv 172.16.2.4 uptime
The authenticity of host '172.16.2.4 (172.16.2.4)' can't be established.
RSA key fingerprint is 81:76:58:27:14:5f:73:df:cc:94:2b:6d:63:55:db:f9.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.16.2.4' (RSA) to the list of known hosts.
 20:15:44 up 4 min,  0 users,  load average: 0.07, 0.04, 0.01
```

5.5.3 Re-enable Nova Services on the Original Compute Node

1. From the cloud controller, re-enable the disabled Nova services on *rhos2*.

```
[root@rhos0 ~]# nova-manage service enable –host=rhos2.example.com \
--service=nova-network

[root@rhos0 ~]# nova-manage service enable –host=rhos2.example.com \
--service=nova-compute
```

2. Verify the services have been re-enabled.

```
[root@rhos0 ~]# nova-manage service list
Binary           Host                Zone  Status   State Updated_At
nova-cert        rhos0.example.com   nova  enabled  :-)   2013-05-01 20:18:37
nova-scheduler   rhos0.example.com   nova  enabled  :-)   2013-05-01 20:18:38
```

```
nova-consoleauth rhos0.example.com  nova  enabled  :-)   2013-05-01 20:18:37
nova-compute     rhos2.example.com  nova  enabled  :-)   2013-05-01 20:18:37
nova-network     rhos2.example.com  nova  enabled  :-)   2013-05-01 20:18:38
nova-compute     rhos5.example.com  nova  enabled  :-)   2013-05-01 20:18:33
nova-compute     rhos3.example.com  nova  enabled  :-)   2013-05-01 20:18:31
nova-compute     rhos4.example.com  nova  enabled  :-)   2013-05-01 20:18:32
nova-network     rhos3.example.com  nova  enabled  :-)   2013-05-01 20:18:33
nova-network     rhos4.example.com  nova  enabled  :-)   2013-05-01 20:18:37
nova-network     rhos5.example.com  nova  enabled  :-)   2013-05-01 20:18:38
```

5.6 Deploy Cinder NFS 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 Cinder server's local storage that is shared as an iSCSI target via **tgtd**. This section of the reference architecture describes the steps necessary to convert the local Cinder server to an NFS-backed Cinder server that uses a dedicated storage network. The cloud controller runs the core Cinder services including **cinder-api**, **cinder-scheduler**, and **cinder-volume**.

5.6.1 Build a NFS Server

The first step in deploying an NFS-backed Cinder server is to build an NFS server. Server *rhos1* was used in this reference architecture. It has a dedicated 10Gb interface on the storage network and ample internal disk space for persistent storage volumes.

1. Connect to to the NFS server.

```
[user@ra-users ~]$ ssh -l root rhos1
Warning: Permanently added 'rhos1,10.16.137.101' (RSA) to the list of known
hosts.
root@rhos1's password:
Kickstarted on 2013-04-03
```

2. Install the **nfs-utils** package if it is not already installed.

```
[root@rhos1 ~]# yum install -y -q nfs-utils
Package 1:nfs-utils-1.2.3-36.el6.x86_64 already installed and latest version
```

3. Start NFS and configure it to start at boot.

```
[root@rhos1 ~]# service nfs start

[root@rhos1 ~]# chkconfig nfs on
```

4. Create a share and export it. In this example the share is named */share*.

```
[root@rhos1 ~]# mkdir /share

[root@rhos1 ~]# cp /etc/exports /etc/exports.orig
```

```
[root@rhos1 ~]# echo "/share *(rw,sync,no_root_squash)" > /etc/exports

[root@rhos1 ~]# exportfs -rav
exporting *:/share
```

5. Add the firewall rules for NFS.

```
[root@rhos1 ~]# iptables -I INPUT -p tcp --dport 2049 -j ACCEPT

[root@rhos1 ~]# service iptables save
iptables: Saving firewall rules to /etc/sysconfig/iptables:[  OK  ]
[root@rhos1 ~]# service iptables restart
iptables: Flushing firewall rules:                         [  OK  ]
iptables: Setting chains to policy ACCEPT: filter          [  OK  ]
iptables: Unloading modules:                               [  OK  ]
iptables: Applying firewall rules:                         [  OK  ]
```

6. Verify that the share is being exported.

```
[root@rhos1 ~]# showmount -e localhost
Export list for localhost:
/share *
```

5.6.2 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.nfs.NfsDriver

[root@rhos0 ~]# echo "rhos1-stor:/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
rhos1-stor:/share

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

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 setsebool
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  13488) is running...
openstack-cinder-scheduler (pid  13503) is running...
openstack-cinder-volume (pid  13520) is running...
```

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

```
[root@rhos0 ~]# vgs
  VG             #PV #LV #SN Attr   VSize   VFree
  cinder-volumes   1   0   0 wz--n-  20.00g 20.00g
  myvg             1   1   0 wz--n- 134.94g     0

[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
```

5.6.3 Create a Persistent Volume

1. Create a Cinder volume as the *refarch* user.

```
[root@rhos6 ~(refarch_user)]$ cinder create --display-name ravo1_1 5
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|     attachments     |                  []                  |
|  availability_zone  |                 nova                 |
|      created_at     |      2013-05-01T21:07:40.152862      |
| display_description |                 None                 |
|     display_name    |               ravo1_1                |
|          id         | b6096868-1770-4d55-88cc-9b0eab53b0de |
```

```
|       metadata      |                  {}                  |
|         size        |                  5                   |
|     snapshot_id     |                None                  |
|        status       |              creating                |
|     volume_type     |                None                  |
+---------------------+--------------------------------------+
```

2. Display the new volume.

```
[root@rhos6 ~(refarch_user)]$ cinder list
+--------------------------------------+-----------+--------------+------+-+
|                  ID                  |   Status  | Display Name | Size |
Volume Type | Bootable | Attached to |
+--------------------------------------+-----------+--------------+------+-+
| b6096868-1770-4d55-88cc-9b0eab53b0de | available |   ravo1_1    |  5   |
None    |          |             |
+--------------------------------------+-----------+--------------+------+-+
```

3. Attach the volume to an instance. In this example the volume is attached to **ra1**.

```
[root@rhos6 ~(refarch_user)]$ nova list
+--------------------------------------+------+--------+-------------------+
| ID                                   | Name | Status | Networks
+--------------------------------------+------+--------+-------------------+
| 54ff0d3f-443f-47e6-9ba5-27ce6a77aee7 | ra1  | ACTIVE | refnet=172.16.2.2,
10.16.143.109 |
| c0e9f13d-18e0-4f7e-aa38-c8734d5d77f7 | ra2  | ACTIVE | refnet=172.16.2.4
+--------------------------------------+------+--------+-------------------+

[root@rhos6 ~(refarch_user)]$ nova volume-list
+------------------------------------+-----------+---------+------+-----+-------+
| ID                                 | Status    | Display Name | Size |
Volume Type | Attached to |
+---------------+-----------+--------------+------+-------------+----------+
| b6096868-1770-4d55-88cc-9b0eab53b0de | available | ravo1_1      | 5    |
None        |             |
+------------+-----------+--------------+------+-------------+-------------+

[root@rhos6 ~(refarch_user)]$ nova volume-attach ra1 b6096868-1770-4d55-88cc-
9b0eab53b0de auto
+----------+--------------------------------------+
| Property | Value                                |
+----------+--------------------------------------+
| device   | /dev/vdb                             |
| serverId | 54ff0d3f-443f-47e6-9ba5-27ce6a77aee7 |
| id       | b6096868-1770-4d55-88cc-9b0eab53b0de |
| volumeId | b6096868-1770-4d55-88cc-9b0eab53b0de |
+----------+--------------------------------------+
```

4. View the volume.

```
[root@rhos6 ~(refarch_user)]$ nova volume-list
+-----------+--------+--------------+------+-------------+-----------------+
| ID | Status | Display Name | Size | Volume Type | Attached to
```

```
+-----------+--------+--------------+------+------------+-----------------+
| b6096868-1770-4d55-88cc-9b0eab53b0de | in-use | ravo1_1      | 5    | None
| 54ff0d3f-443f-47e6-9ba5-27ce6a77aee7 |
+-----------+--------+--------------+------+------------+-----------------+
```

The compute node requests the persistent volume from the Cinder server via the **`cinder-api`** service. The **`cinder-scheduler`** presents the NFS share to the compute node. **Figure 5.6.3.1: NFS-Backed Cinder** depicts this relationship.

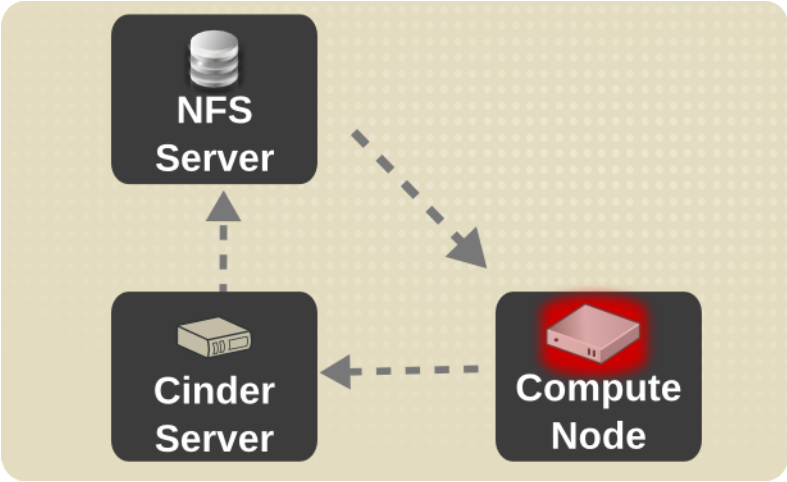


Figure 5.6.3.1: NFS-Backed Cinder

5.6.4 Attach the Volume to an Instance

1. SSH to the instance attached to the volume. Discover the volume.

   NOTE: Connect to the instance from a compute node with a bridge interface on the *refarch* virtual network. Only a compute node with an instance deployed to the *refarch* tenant has a bridge interface on the *refarch* virtual network.

```
[root@rhos6 ~(refarch_user)]$ ssh -i rakey.priv 10.16.143.109
Last login: Wed May  1 20:14:56 2013 from 10.16.137.106

[root@ra1 ~]# grep vdb /proc/partitions
 252       16    5242880 vdb
```

2. Label and create a partition on the volume.

```
[root@ra1 ~]# parted /dev/vdb mklabel gpt
Information: You may need to update /etc/fstab.

[root@ra1 ~]# parted /dev/vdb mkpart test 1 1000
```

```
Information: You may need to update /etc/fstab.
```

3. Format the new partition.

```
[root@ra1 ~]# mkfs.ext3 /dev/vdb1
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
61056 inodes, 243968 blocks
12198 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=251658240
8 block groups
32768 blocks per group, 32768 fragments per group
7632 inodes per group
Superblock backups stored on blocks:
      32768, 98304, 163840, 229376

Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 37 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

4. Mount the filesystem and create a new file.

```
[root@ra1 ~]# mount /dev/vdb1 /mnt

[root@ra1 ~]# touch /mnt/test

[root@ra1 ~]# ls /mnt
lost+found  test
```

5. Unmount the filesystem and exit.

```
[root@ra1 ~]# umount /mnt

[root@ra1 ~]# exit
```

6. Detach the volume from the host.

```
[root@rhos6 ~(refarch_user)]$ nova volume-detach ra1 b6096868-1770-4d55-
88cc-9b0eab53b0de

[root@rhos6 ~(refarch_user)]$ nova volume-list
+------------+-----------+--------------+------+-------------+-------------+
| ID                                   | Status    | Display Name | Size |
Volume Type | Attached to |
+------------+-----------+--------------+------+-------------+-------------+
| b6096868-1770-4d55-88cc-9b0eab53b0de | available | ravo1_1      | 5    |
```

```
None        |             |
+------------+-----------+--------------+------+-------------+-------------+
```

5.6.5 Attach the Volume to the Second Instance

1. Attach the volume to the second instance created in **5.5.2 Boot a Test Instance.** In this example the second host is named *ra2***.**

```
[root@rhos6 ~(refarch_user)]$ nova volume-attach ra2 b6096868-1770-4d55-
88cc-9b0eab53b0de auto
+----------+--------------------------------------+
| Property | Value                                |
+----------+--------------------------------------+
| device   | /dev/vdb                             |
| serverId | c0e9f13d-18e0-4f7e-aa38-c8734d5d77f7 |
| id       | b6096868-1770-4d55-88cc-9b0eab53b0de |
| volumeId | b6096868-1770-4d55-88cc-9b0eab53b0de |
+----------+--------------------------------------+

[root@rhos6 ~(refarch_user)]$ nova volume-list
+------------+-----------+--------------+------+-------------+-------------+
| ID                                   | Status | Display Name | Size |
Volume Type | Attached to                          |
+------------+-----------+--------------+------+-------------+-------------+
| b6096868-1770-4d55-88cc-9b0eab53b0de | in-use | ravo1_1      | 5    | None
| c0e9f13d-18e0-4f7e-aa38-c8734d5d77f7 |
+------------+-----------+--------------+------+-------------+-------------+
```

2. SSH to the host and mount the volume.

```
[root@rhos6 ~(refarch_user)]$ ssh -i rakey.priv 10.16.143.109
Last login: Wed May  1 21:29:50 2013 from 10.16.137.106

[root@ra1 ~]# ssh -i rakey.priv 172.16.2.4

[root@ra2 ~]# mount /dev/vdb1 /mnt
```

3. Verify the file is seen then unmount the partition.

```
[root@ra2 ~]# ls /mnt
lost+found  test

[root@ra2 ~]# umount /dev/vdb1

[root@ra2 ~]# exit
log out
Connection to 172.16.2.4 closed.

[root@ra1 ~]# exit
logout
Connection to 10.16.143.109 closed.
```

4. Detach the volume from the second host.

```
[root@rhos6 ~(refarch_user)]$ nova volume-detach ra2 b6096868-1770-4d55-
88cc-9b0eab53b0de

[root@rhos6 ~(refarch_user)]$ nova volume-list
+------------+-----------+--------------+------+-------------+-------------+
| ID                                     | Status    | Display Name | Size |
Volume Type | Attached to |
+------------+-----------+--------------+------+-------------+-------------+
| b6096868-1770-4d55-88cc-9b0eab53b0de | available | ravo1_1      | 5    |
None        |             |
+------------+-----------+--------------+------+-------------+-------------+
```

5.6.6 Complete OpenStack Deployment

Figure 5.6.6.1: Complete OpenStack Deployment depicts the complete OpenStack deployment.

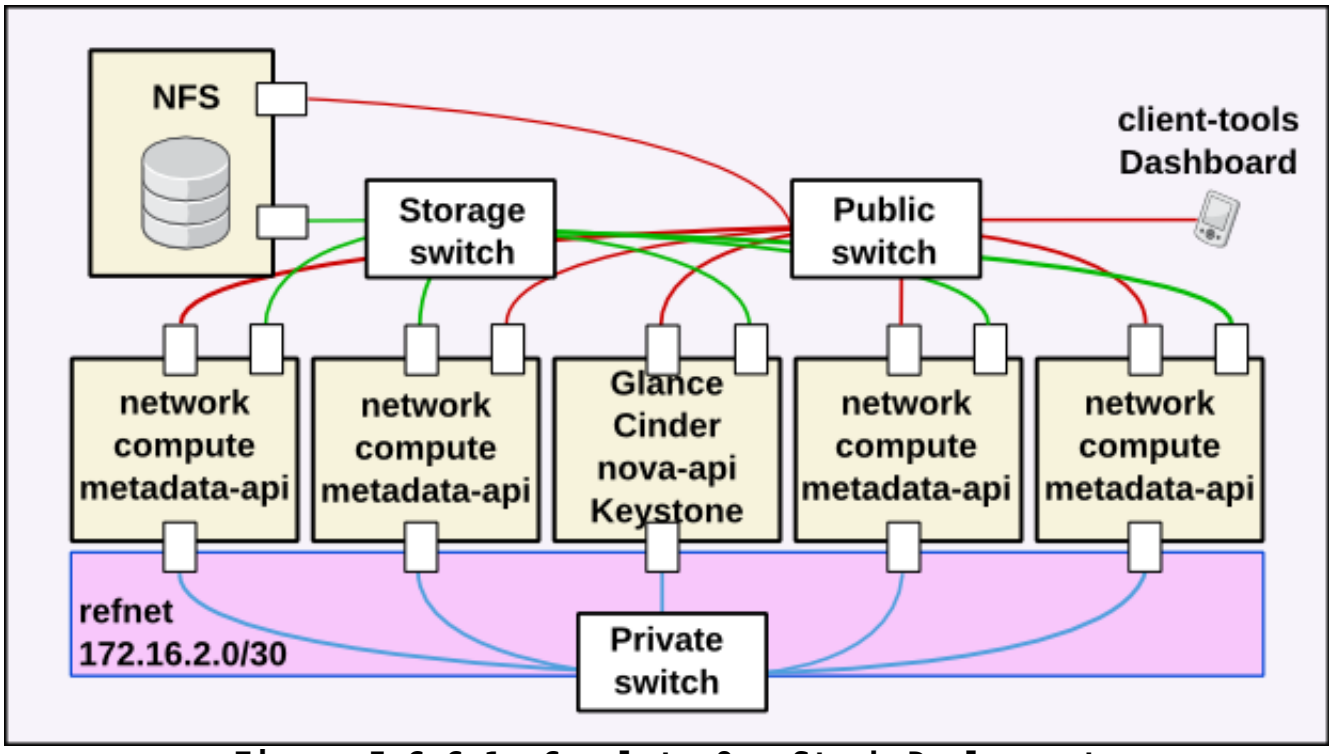


Figure 5.6.6.1: Complete OpenStack Deployment

NOTE: Not all services are shown in this graphic. **Table 3.4.1: Service Placement** lists server placement for all services.

6 Deploy a Multi-tier Web Application

6.1 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. This example uses a simple WordPress deployment. The web server is accessible via a public 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 Cinder.

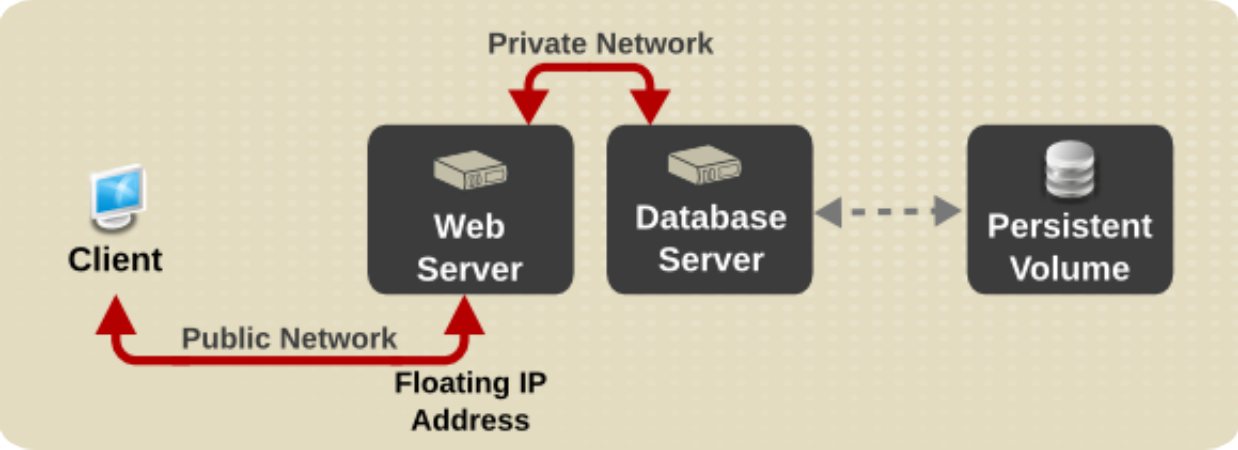


Figure 6.1.1: Multi-tier Application

The process for deploying a multi-tier application demonstrates OpenStack's complete functionality as described in this reference architecture. It is a realistic use case that touches every service discussed in the preceding pages. Multi-tier web applications are often deployed in IaaS clouds so they can be scaled as needed. Keeping the database server off a publicly accessible network provides an extra layer of security. **Figure 6.1.1: Multi-tier Application** depicts the end state of the multi-tier application deployment.

6.2 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.

6.2.1 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.

6.2.2 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.

6.2.3 Create the Database Volume

1. Create the persistent volume to store the MySQL database.

```
[root@rhos6 ~(refarch_user)]$ nova volume-create --display-name mysql-vol 5
+---------------------+--------------------------------------+
| Property            | Value                                |
+---------------------+--------------------------------------+
| status              | creating                             |
| display_name        | mysql-vol                            |
| attachments         | []                                   |
| availability_zone   | nova                                 |
| created_at          | 2013-04-19T18:56:32.126873           |
| display_description | None                                 |
| volume_type         | None                                 |
| snapshot_id         | None                                 |
| size                | 5                                    |
| id                  | 23d8a454-27ea-4a74-bed1-19b72da42df2 |
| metadata            | {}                                   |
+---------------------------------------------------------------+
```

2. Display the new volume.

```
[root@rhos6 ~(refarch_user)]$ nova volume-list
+------------+-----------+--------------+------+-------------+-------------+
| ID                                   | Status    | Display Name | Size |
Volume Type | Attached to |
+------------+-----------+--------------+------+-------------+-------------+
| 23d8a454-27ea-4a74-bed1-19b72da42df2 | available | mysql-vol    | 5    |
None        |             |
| b6096868-1770-4d55-88cc-9b0eab53b0de | available | ravo1_1      | 5    |
None        |             |
+------------+-----------+--------------+------+-------------+-------------+
```

6.2.4 Boot the Instance

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

```
[root@rhos6 ~(refarch_user)]$ nova boot --config-drive=true \
```

```
--image=rhel-server2 \
--user-data /pub/scripts/wp-backend.sh \
--flavor 2 \
--key-name rakey \
--file sysreg.sh=/pub/scripts/sysreg.sh \
--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" \
--meta rhncreds="-u admin -p password" \
--block-device-mapping vdd=23d8a454-27ea-4a74-bed1-19b72da42df2:::0 \
--nic net-id=1826f214-63e8-45b7-bd82-f1c7bb7dffe3 ra3
```

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"*

  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.

NOTE: Per **BZ893374** the block device mapping device name may be ignored.

6.2.5 Verify the Database Server

1. The database server is not assigned a publicly accessible floating IP as an added layer

of security. Connect to the instance via SSH from its host compute node. Verify that **mysqld** is running.

```
[root@rhos4 ~]# ssh -i rakey.priv 172.16.2.6
The authenticity of host '172.16.2.6 (172.16.2.6)' can't be established.
RSA key fingerprint is 8d:eb:0e:f0:a6:4b:1b:4b:c7:dc:35:db:13:2c:1a:2c.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.16.2.6' (RSA) to the list of known hosts.

[root@ra3 ~]# rpm -qa | grep mysql
mysql-5.1.69-1.el6_4.x86_64
mysql-server-5.1.69-1.el6_4.x86_64
mysql-libs-5.1.69-1.el6_4.x86_64

[root@ra3 ~]# service mysqld status
mysqld (pid  1799) is running...
```

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

```
[root@ra3 ~]# mount | grep vd
/dev/vdc on /rhos type iso9660 (rw)
/dev/vdb on /var/lib/mysql type ext4 (rw)
```

3. View the WordPress database has been created on the MySQL server.

```
[root@ra3 ~]# mysql -u root -e 'show databases;'
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| test               |
| wordpress          |
+--------------------+
```

6.3 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.

6.3.1 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 **6.2.2 Create the Registration Script**.

6.3.2 Launch the Web Server

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

```
[root@rhos6 ~(refarch_user)]$ nova boot --config-drive=true \
--image=rhel-server2 \
--user-data /pub/projects/rhos/folsom/scripts/wp-frontend.sh \
--flavor 2 --key-name rakey \
--file sysreg.sh=/pub/projects/rhos/folsom/scripts/sysreg.sh \
--meta sysregopts="-u admin -p 100yard- -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" \
--meta rhncreds="-u admin -p 100yard-" \
--meta wp_mysql_ip="172.16.2.6" \
--nic net-id=1826f214-63e8-45b7-bd82-f1c7bb7dffe3,v4-fixed-ip="172.16.2.110"
ra4
+-------------------------+--------------------------------------------------+
| Property                | Value
+-------------------------+--------------------------------------------------+
| status                  | BUILD
| updated                 | 2013-05-02T03:53:59Z
| OS-EXT-STS:task_state   | scheduling
| key_name                | rakey
| image                   | rhel-server2
| hostId                  |
| OS-EXT-STS:vm_state     | building
| flavor                  | m1.small
| id                      | 2bd838f7-34f9-444f-a1d6-15e34c5cc8a5
| security_groups         | [{u'name': u'default'}]
| user_id                 | 043914f4ff124305a8ac0389aeb02d35
| name                    | ra4
| adminPass               | noeNDpgGdE5o
| tenant_id               | b1c1febbb08a422d89357349c71e1543
| created                 | 2013-05-02T03:53:58Z
| OS-DCF:diskConfig       | MANUAL
| accessIPv4              |
| accessIPv6              |
|
| progress                | 0
|
| OS-EXT-STS:power_state | 0
|
| metadata                | {u'sysregopts': u'-u admin -p 100yard- -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',
u'wp_mysql_ip': u'172.16.2.6', u'rhncreds': u'-u admin -p 100yard-'} |
| config_drive            | 1
```

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

[root@rhos6 ~(refarch_user)]$ nova list
+-----------------------+-------------------------------------------------+
| ID                                   | Name | Status | Networks
+-----------------------+-------------------------------------------------+
| 54ff0d3f-443f-47e6-9ba5-27ce6a77aee7 | ra1  | ACTIVE | refnet=172.16.2.2,
10.16.143.109 |
| c0e9f13d-18e0-4f7e-aa38-c8734d5d77f7 | ra2  | ACTIVE | refnet=172.16.2.4
| ae2c1bcd-0113-4640-8a38-18b5ac85847e | ra3  | ACTIVE | refnet=172.16.2.6
| 2bd838f7-34f9-444f-a1d6-15e34c5cc8a5 | ra4  | ACTIVE | refnet=172.16.2.110
+-----------------------+-------------------------------------------------+
```

2. Several of the command line arguments to **nova boot** were described in **6.2.4 Boot the Instance** . The remaining parameters are described here.

- *--meta wp_mysql_ip="172.16.2.6"*

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

- *--nic net-id=1826f214-63e8-45b7-bd82-f1c7bb7dffe3,v4-fixed-ip="172.16.2.110"*

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

6.3.3 Associate a Public IP Address with the Web Server Instance

1. Create a floating IP address.

```
[root@rhos6 ~(refarch_user)]$ nova floating-ip-create
+---------------+-------------+----------+------+
| Ip            | Instance Id | Fixed Ip | Pool |
+---------------+-------------+----------+------+
| 10.16.143.110 | None        | None     | nova |
+---------------+-------------+----------+------+
```

2. Add the floating IP address to the WordPress server.

```
[root@rhos6 ~(refarch_user)]$ nova add-floating-ip ra4 10.16.143.110

[root@rhos6 ~(refarch_user)]$ nova floating-ip-list
+---------------+--------------------------------------+-------+------+----+
| Ip            | Instance Id                          | Fixed Ip     | Pool
+---------------+--------------------------------------+--------------+----+
| 10.16.143.109 | 54ff0d3f-443f-47e6-9ba5-27ce6a77aee7 | 172.16.2.2   | nova
| 10.16.143.110 | 1d05b6fe-33a1-45d3-a6d1-1c916c292359 | 172.16.2.110 | nova
+---------------+--------------------------------------+--------------+----+
```

3. Add rules allowing HTTP and ICMP from anywhere to the default security group.

```
[root@rhos6 ~(refarch_user)]$ nova secgroup-add-rule default tcp 80 80
0.0.0.0/0
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
```

```
+-------------+-----------+---------+-----------+--------------+
| tcp         | 80        | 80      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+

[root@rhos6 ~(refarch_user)]$ nova secgroup-add-rule default icmp -1 -1
0.0.0.0/0
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| icmp        | -1        | -1      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
```

4. Display the default security group rules.

```
[root@rhos6 ~(refarch_user)]$ nova secgroup-list-rules default
+-------------+-----------+---------+----------------+--------------+
| IP Protocol | From Port | To Port | IP Range       | Source Group |
+-------------+-----------+---------+----------------+--------------+
| icmp        | -1        | -1      | 0.0.0.0/0      |              |
| tcp         | 22        | 22      | 10.16.137.0/24 |              |
| tcp         | 80        | 80      | 0.0.0.0/0      |              |
+-------------+-----------+---------+----------------+--------------+
```

6.3.4 Verify the Web Server Installation

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

```
[root@rhos6 ~(refarch_user)]$ ssh -i rakey.priv 10.16.143.110
The authenticity of host '10.16.143.110 (10.16.143.110)' can't be
established.
RSA key fingerprint is 92:37:75:e2:91:b4:4d:33:c9:b5:d7:f2:2d:a8:51:ab.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.16.143.110' (RSA) to the list of known hosts.

[root@ra4 ~]# rpm -qa | grep http
httpd-2.2.15-26.el6.x86_64
httpd-tools-2.2.15-26.el6.x86_64
lighttpd-1.4.31-1.el6.x86_64

[root@ra4 ~]# service httpd status
httpd (pid  1863) is running...
```

6.4 Test the Web Server from a Client

Connect to the WordPress page in a browser via the floating IP address. **Figure 6.4.1: WordPress Installation** is displayed after a successful installation.

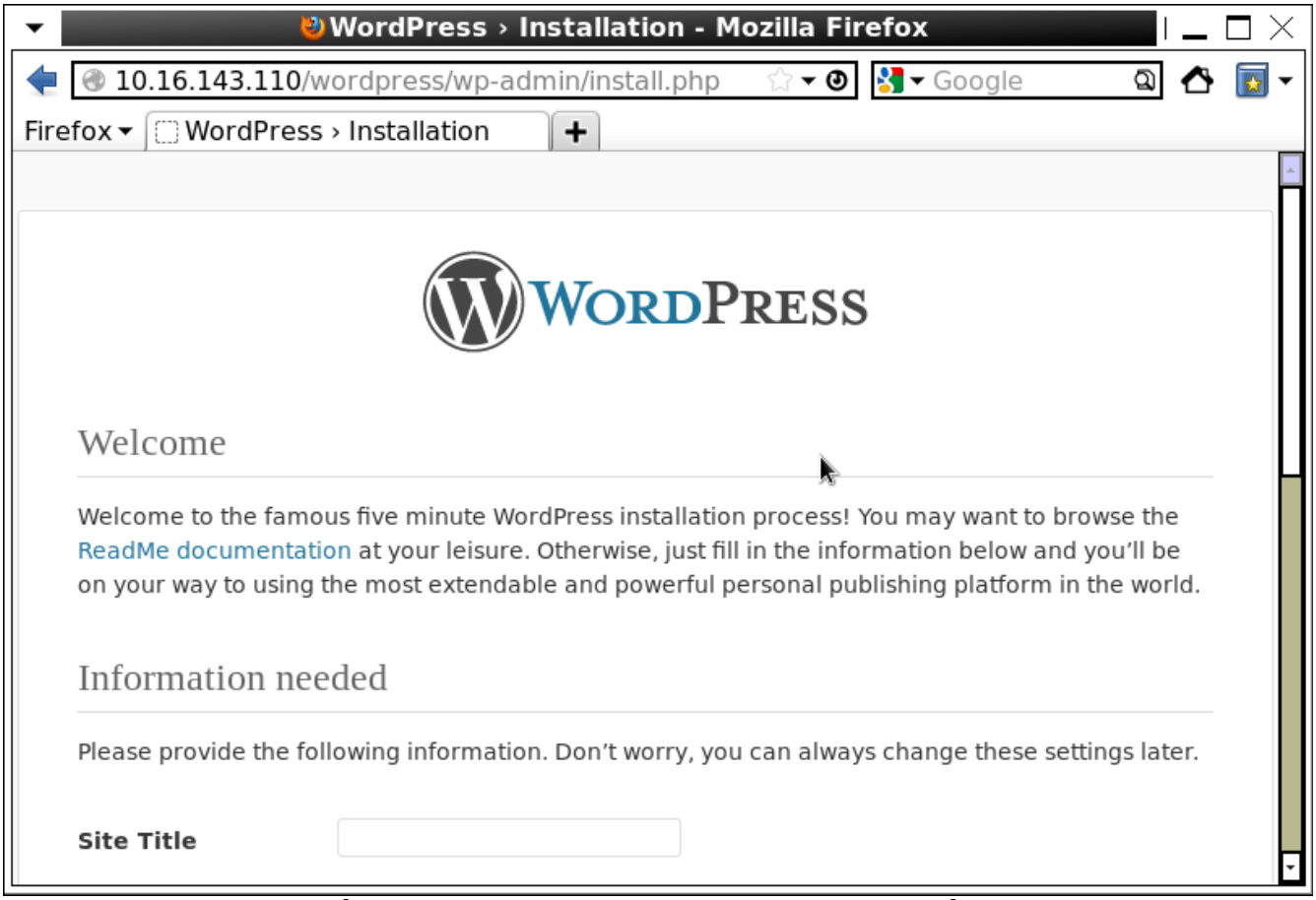


Figure 6.4.1: WordPress Installation

6.5 Complete Multi-tier Application

Figure 6.5.1: Complete Multi-tier Application depicts the complete multi-tier web application deployed in this section.

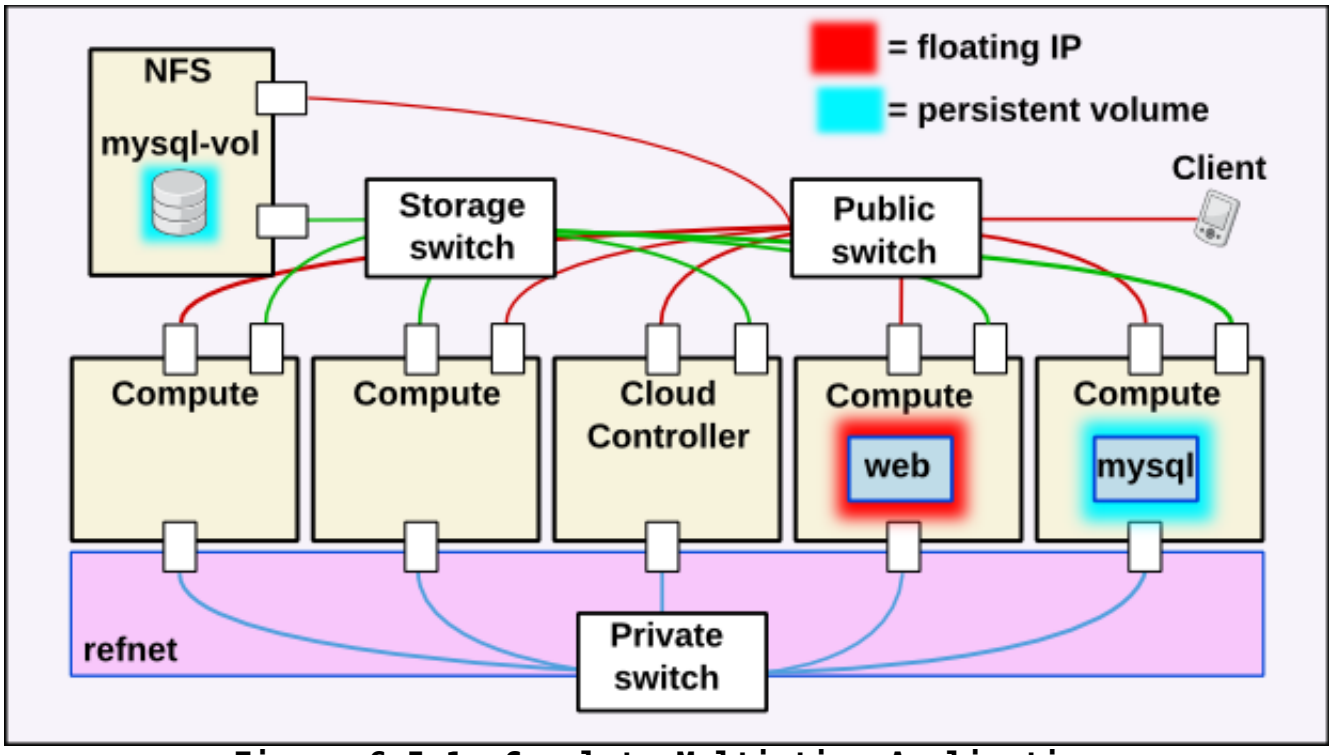


Figure 6.5.1: Complete Multi-tier Application

7 Conclusion

Red Hat OpenStack 2.1 provides a flexible cloud framework built on a stable code base and backed by Red Hat's open source software expertise. Red Hat OpenStack 2.1 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`**
- Expand the compute node infrastructure
- 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: Revision History

Revision 1.0	May 1, 2013	Jacob Liberman

Final

- Links to supporting materials
- Final graphic placement

Revision 0.4	April 29, 2013	Jacob Liberman

Third draft

- Engineering review change commit
- Graphic conversion
- Final text and screen captures

Revision 0.3	April 25, 2013	Jacob Liberman

Second draft

- Format
- Spacing

Revision 0.2	April 15, 2013	Jacob Liberman

First draft

- Style template
- Graphics
- Appendices

Revision 0.1	March 11, 2013	Jacob Liberman

Initial Release

- Title
- Abstract
- Outline

Appendix B: Contributors

1. Steve Reichard, content and scripts, technical and content review
2. Scott Collier, content review
3. Derek Higgins, engineering review
4. Padraig Brady, engineering review