Red Hat OpenStack Red Hat OpenStack 2.1 (Folsom) Getting Started Guide

Getting Started with Red Hat OpenStack 2.1 (Folsom)

Red Hat Documentation Team

Red Hat OpenStack Red Hat OpenStack 2.1 (Folsom) Getting Started Guide

Getting Started with Red Hat OpenStack 2.1 (Folsom)

Red Hat Documentation Team

Keywords

Abstract

This manual covers the basic getting started tasks for OpenStack Folsom Preview.

Table of Contents

Preface

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the Liberation Fonts set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keys and key combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a key, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from an individual key by the plus sign that connects each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to a virtual terminal.

The first example highlights a particular key to press. The second example highlights a key combination: a set of three keys pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`mono-spaced bold`**. For example:

> File-related classes include **`filesystem`** for file systems, **`file`** for files, and **`dir`** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **`Buttons`** tab, click the **`Left-handed mouse`** check box and click **`Close`** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications** → **Accessories** →

Character Map from the main menu bar. Next, choose **Search** → **Find...** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

Mono-spaced Bold Italic or ***Proportional Bold Italic***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

To connect to a remote machine using ssh, type **ssh *username*@*domain.name*** at a shell prompt. If the remote machine is **example.com** and your username on that machine is john, type **ssh john@example.com**.

The **mount -o remount *file-system*** command remounts the named file system. For example, to remount the **/home** file system, the command is **mount -o remount /home**.

To see the version of a currently installed package, use the **rpm -q *package*** command. It will return a result as follows: ***package-version-release***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **mono-spaced roman** and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in **mono-spaced roman** but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }
}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

Part I. Introduction

Important

Red Hat OpenStack now includes 3 entitlements for Red Hat Enterprise Linux servers with unlimited guest subscriptions (both for guest and host OS). This trial is valid for a 90-day period.

OpenStack is a cloud-based collection of interacting services that control computing, storage, and networking resources. It is managed through a web-based interface which allows administrators to control, provision and automate the OpenStack resources.

This guide aims to provide you with the knowledge required to install Red Hat OpenStack on Red Hat Enterprise Linux systems in either a single or multiple node configuration with all core services of the OpenStack Folsom release enabled and working.

Additional Resources

Additional online resources are available to support users of Red Hat OpenStack.

Customer Portal

The Red Hat Customer Portal offers a wide range of resources to help guide you through planning, deploying, and maintaining your OpenStack deployment. Facilities available via the Customer Portal include:

- Knowledge base articles and solutions.
- Reference architectures.
- Technical briefs.
- Product documentation.
- Support case management.

Access the Customer Portal at https://access.redhat.com/.

Mailing Lists

Red Hat provides these public mailing lists that are relevant to OpenStack users:

- The **`rhos-list`** mailing list provides a forum for discussions about installing, running, and using OpenStack on Red Hat based distributions.

  Subscribe at https://www.redhat.com/mailman/listinfo/rhos-list.
- The **`rhsa-announce`** mailing list provides notification of the release of security fixes for all Red Hat products, including Red Hat OpenStack.

  Subscribe at https://www.redhat.com/mailman/listinfo/rhsa-announce.

Note

The full list of updates released for Red Hat OpenStack 2.0 and 2.1 (Folsom) is maintained at https://rhn.redhat.com/errata/rhel6-rhos-folsom-errata.html.

Community Documentation

Additional documentation provided by the wider OpenStack community is available at

http://docs.openstack.org.

Chapter 1. Architecture

OpenStack includes the following services:

Nova (Compute)

A service that manages networks of virtual machines running on nodes, providing virtual servers on demand. Nova is a distributed component and interacts with Keystone for authentication, Glance for images and Horizon for web interface. Nova is designed to scale horizontally on standard hardware, downloading images to launch instances as required.

Glance (Image)

A service that acts as a registry for virtual machine images, allowing users to copy server images for immediate storage. These images can be used as templates when setting up new servers. Usually the images are stored in the Swift (Object) service.

OpenStack Networking

OpenStack Networking provides connectivity between the interfaces of other OpenStack services, such as Nova. Due to OpenStack Networking's pluggable architecture, users can create their own networks, control traffic, and connect servers to other networks. Various networking technologies are supported.

Cinder (Volume)

A service that manages storage volumes for virtual machines. This is persistent block storage for the instances running in Nova. Snapshots can be taken for backing up and data, either for restoring data, or to be used to create new block storage volumes.

Swift (Object)

A service providing object storage which allows users to store and retrieve files. Swift architecture is distributed to allow for horizontal scaling, and to provide redundancy as failure-proofing. Data replication is manage by software, allowing greater scalability and redundancy than dedicated hardware.

Keystone (Identity)

A centralized identity service that provides authentication and authorization for other services. Keystone also provides a central catalog of services running in a particular OpenStack cloud. It supports multiple forms of authentication including user name and password credentials, token-based systems, and Amazon Web Services style logins.

Horizon (Dashboard)

A web-based interface for managing OpenStack services. It provides a graphical user interface for operations such as launching instances, managing networking and setting access controls. Its modular design allows interfacing with other products such as billing, monitoring and additional management tools.

The following diagram shows an overview of the services and how they interact:

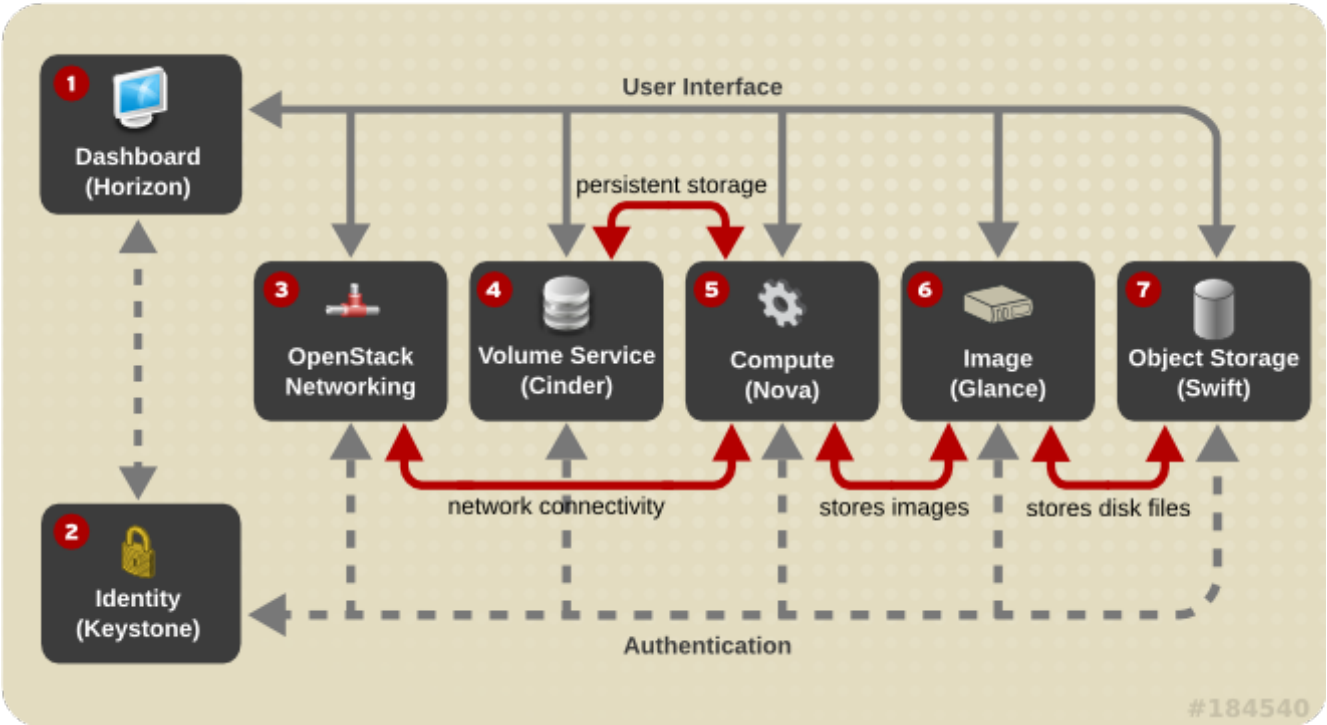


Figure 1.1. Relationships between OpenStack services

1. Horizon: Web browser user interface for creating and managing instances.
2. Keystone: Authentication and authorization framework.
3. OpenStack Networking: Network connectivity as a service.
4. Cinder: Persistent block storage for runtime instances.
5. Nova: Scheduler for networks of virtual machines running on nodes.
6. Glance: Registry for virtual machine images.
7. Swift: File storage and retrieval.

Chapter 2. Prerequisites

2.1. System Requirements

The system requirements for an OpenStack deployment vary based on the scale and workload of the environment being deployed.

This guide provides the recommended minimum system requirements for some common deployment scenarios.

Important

To verify that the processor of a system running Red Hat Enterprise Linux has the required CPU extensions and that they are enabled check the contents of the **/proc/cpuinfo** file:

```
# grep -E 'svm|vmx' /proc/cpuinfo | grep nx
```

If any output is shown, the processor is hardware virtualization capable. If no output is shown it is still possible that your processor supports hardware virtualization. In some circumstances manufacturers disable the virtualization extensions in the BIOS. Where you believe this to be the case consult the system's BIOS and the motherboard manual provided by the manufacturer.

2.1.1. Single Node ("All in One") Deployments

In this configuration all services are installed and run on a single system. This simplifies the deployment process and is suitable for evaluation purposes. Such a deployment is not however suitable for use in a production environment.

Processor

64-bit x86 processor with support for the Intel 64 or AMD64 CPU extensions, and the AMD-V or Intel VT hardware virtualization extensions enabled.

Memory

A minimum of 2 GB of RAM is recommended.

Add additional RAM to this requirement based on the amount of memory that you intend to make available to virtual machine instances.

Disk Space

A minimum of 50 GB of available disk space is recommended.

Add additional disk space to this requirement based on the amount of space that you intend to make available to virtual machine instances. This figure varies based on both the size of each disk image you intend to create and whether you intend to share one or more disk images between multiple instances.

1 TB of disk space is recommended for a realistic environment capable of hosting multiple instances of varying sizes.

Network Interface Cards

1 x 1 Gbps Network Interface Card.

2.1.2. Cloud Controller Deployment with One or More Compute Nodes

In this configuration one system acts as the cloud controller by hosting services including the compute database and API server.

Other available systems are used as compute nodes on which virtual machine instances are run. Support services such as image storage are provided on either the cloud controller or one or more of the compute nodes.

Cloud Controller

Processor

64-bit x86 processor with support for the Intel 64 or AMD64 CPU extensions, and the AMD-V or Intel VT hardware virtualization extensions enabled.

Memory

A minimum of 2 GB of RAM is recommended.

Disk Space

A minimum of 50 GB of available disk space is recommended.

Add additional disk space to this requirement based on the amount of space that you intend to make available to virtual machine instances. This figure varies based on both the size of each disk image you intend to create and whether you intend to share one or more disk images between multiple instances.

1 TB of disk space is recommended for a realistic environment capable of hosting multiple instances of varying sizes.

Network Interface Cards

2 x 1 Gbps Network Interface Cards.

Compute Nodes

Processor

64-bit x86 processor with support for the Intel 64 or AMD64 CPU extensions, and the AMD-V or Intel VT hardware virtualization extensions enabled.

Memory

A minimum of 2 GB of RAM is recommended.

Add additional RAM to this requirement based on the amount of memory that you intend to make available to virtual machine instances.

Disk Space

A minimum of 50 GB of available disk space is recommended.

Add additional disk space to this requirement based on the amount of space that you intend to make available to virtual machine instances. This figure varies based on both the size of each disk image you intend to create and whether you intend to share one or more disk images between multiple instances.

1 TB of disk space is recommended for a realistic environment capable of hosting multiple instances of varying sizes.

Network Interface Cards

2 x 1 Gbps Network Interface Cards.

2.2. Configuring Software Repositories

Red Hat OpenStack requires that each system in the OpenStack environment be running Red Hat Enterprise Linux 6.4 Server. It is recommended that freshly installed systems are used.

Additionally all systems must be subscribed to receive software updates for both Red Hat Enterprise Linux 6.4 Server and Red Hat OpenStack. Follow the instructions in Procedure 2.1, “Configuring Software Repositories” to ensure this is the case.

- For further information on installing Red Hat Enterprise Linux 6.4 Server refer to the Red Hat Enterprise Linux *Installation Guide*.
- For further information on managing Red Hat subscriptions refer to the Red Hat *Subscription Management Guide*.

Important

RHN Classic is intended to be used with legacy systems (Red Hat Enterprise Linux 6.0 or Red Hat Enterprise Linux 5.6 and earlier releases). It is strongly recommended that Red Hat Enterprise Linux 6.1/5.7 and later systems use Customer Portal Subscription Management, Subscription Asset Manager, or similar certificate-based subscription management service. As such these instructions are **not** intended for use on systems which have been registered to Red Hat Network using RHN Classic.

Procedure 2.1. Configuring Software Repositories

Unless otherwise mentioned all commands in this procedure must be run while logged in as the **`root`** user. You may choose to log in to the system as the **`root`** user directly or run each command via **sudo** if the system is configured to support it.

1. Use the **`subscription-manager register`** command to register the system to Red Hat Network. Enter your Red Hat Network user name and password when prompted.

```
# subscription-manager register
Username: administrator@example.com
Password:
```

When the system is registered to Red Hat Network successfully it will be assigned a unique identifier. This unique identifier will be displayed,

```
The system has been registered with id: IDENTIFIER
```

2. Locate the identifier for your Red Hat OpenStack subscription pool using the **subscription-manager *list*** command.

```
# subscription-manager list --available
+-------------------------------------------+
    Available Subscriptions
+-------------------------------------------+

Product Name:          Red Hat Enterprise Linux Server
Product Id:            69
Pool Id:               POOLID_1
Quantity:              1
Service Level:         None
Service Type:          None
Multi-Entitlement:     No
Expires:               01/01/2022
Machine Type:          physical

Product Name:          Red Hat OpenStack
Product Id:            SER0406
Pool Id:               POOLID_2
Quantity:              3
Service Level:         None
Service Type:          None
Multi-Entitlement:     No
Expires:               02/14/2013
Machine Type:          physical

Product Name:          Red Hat OpenStack
Product Id:            SER0406
Pool Id:               POOLID_3
Quantity:              unlimited
Service Level:         None
Service Type:          None
Multi-Entitlement:     No
Expires:               02/14/2013
Machine Type:          virtual
```

Note

There are two different types of Red Hat OpenStack subscriptions available. The first is for physical machines and the second is for virtual machines. You are limited to only three physical machines, presumably for compute nodes. There is no limit for the number of virtual machines that OpenStack components may be installed on.

3. Assign a subscription from a pool to the system using the **subscription-manager *attach*** command. You must run this command multiple times ensuring that you attach both a physical Red Hat OpenStack subscription and a Red Hat Enterprise Linux Server subscription to the system.

```
# subscription-manager attach --pool=POOLID
Successfully attached a subscription for PRODUCT.
```

PRODUCT will be replaced in the output with the name of the product you have attached a

subscription for.

4. Ensure that the **/etc/yum.repos.d/redhat.repo** is up to date by running the **yum *repolist*** command. This command also creates the file if it does not exist yet.

```
# yum repolist
```

Once repository metadata has been downloaded and examined, the list of repositories enabled will be displayed, along with the number of available packages.

```
repo id                  repo name
status
rhel-6-server-rpms       Red Hat Enterprise Linux 6 Server (RPMs)      8,816
repolist: 8,816
```

5. Install the *yum-utils* package. The *yum-utils* package provides the **yum-config-manager** utility. This utility will be used in subsequent steps of this procedure.

```
# yum install -y yum-utils
```

Note that depending on the options selected during Red Hat Enterprise Linux installation the *yum-utils* package may already be installed.

6. Use the **yum-config-manager** command to ensure that the correct software repositories are enabled. Each successful invocation of the command will display the updated repository configuration.
   a. Ensure that the repository for the previous Red Hat OpenStack release (Essex) has been disabled.

```
# yum-config-manager --disable rhel-server-ost-6-preview-rpms
Loaded plugins: product-id
==================== repo: rhel-server-ost-6-preview-rpms
====================
[rhel-server-ost-6-preview-rpms]
bandwidth = 0
base_persistdir = /var/lib/yum/repos/x86_64/6Server
baseurl =
https://cdn.redhat.com/content/beta/rhel/server/6/6Server/x86_64/openstack/essex/os
cache = 0
cachedir = /var/cache/yum/x86_64/6Server/rhel-server-ost-6-preview-rpms
cost = 1000
enabled = False
...
```

Note

Yum treats the values **False** and **0** as equivalent. As a result the output on your system may instead contain this string:

```
enabled = 0
```

Note

If you encounter this message in the output from **yum-config-manager** then the system has been registered to Red Hat Network using either RHN Classic or RHN Satellite.

```
This system is receiving updates from RHN Classic or RHN
Satellite.
```

Consult the Red Hat *Subscription Management Guide* for more information on managing subscriptions using RHN Classic or RHN Satellite.

b. Ensure that the repository for the current Red Hat OpenStack release (Folsom) has been enabled.

```
# yum-config-manager --enable rhel-server-ost-6-folsom-rpms
Loaded plugins: product-id
==================== repo: rhel-server-ost-6-folsom-rpms
=======================
[rhel-server-ost-6-folsom-rpms]
bandwidth = 0
base_persistdir = /var/lib/yum/repos/x86_64/6Server
baseurl =
https://cdn.redhat.com/content/dist/rhel/server/6/6Server/x86_64/opensta
ck/folsom/os
cache = 0
cachedir = /var/cache/yum/x86_64/6Server/rhel-server-ost-6-folsom-rpms
cost = 1000
enabled = True
...
```

Note

Yum treats the values **True** and **1** as equivalent. As a result the output on your system may instead contain this string:

```
enabled = 1
```

7. Run the **yum *repolist*** command to verify the correct software repositories are enabled. Note that when the command is run on your system the number of available packages listed may differ.

```
repo id                          repo name
status
rhel-6-server-rpms               Red Hat Enterprise Linux 6 Server (RPMs)
8,816
rhel-server-ost-6-folsom-rpms    Red Hat OpenStack Folsom (RPMs)
138
repolist: 10,058
```

8. Install the *yum-plugin-priorities* package. The *yum-plugin-priorities* package provides a **yum** plug-in allowing configuration of per-repository priorities.

```
# yum install -y yum-plugin-priorities
```

9. Use the **yum-config-manager** command to set the priority of the Red Hat OpenStack software repository to **1**. This is the highest priority value supported by the *yum-plugin-priorities* plug-in.

```
# yum-config-manager --enable rhel-server-ost-6-folsom-rpms \
      --setopt="rhel-server-ost-6-folsom-rpms.priority=1"
Loaded plugins: product-id
==================== repo: rhel-server-ost-6-folsom-rpms
=======================
[rhel-server-ost-6-folsom-rpms]
bandwidth = 0
base_persistdir = /var/lib/yum/repos/x86_64/6Server
baseurl =
https://cdn.redhat.com/content/dist/rhel/server/6/6Server/x86_64/openstack/fo
lsom/os
cache = 0
cachedir = /var/cache/yum/x86_64/6Server/rhel-server-ost-6-folsom-rpms
cost = 1000
enabled = True
...
priority = 1
...
```

10. Run the **yum *update*** command and reboot to ensure that the most up to date Red Hat Enterprise Linux packages, including the kernel, are installed and running.

```
# yum update -y
```

```
# reboot
```

You have successfully configured your system to receive Red Hat OpenStack packages. You may use the **yum *repolist*** to confirm the repository configuration again at any time.

2.3. Configuring sudo Access

The **sudo** command offers a mechanism for providing trusted users with administrative access to a system without sharing the password of the **root** user. When users given access via this mechanism precede an administrative command with **sudo** they are prompted to enter their own password. Once authenticated, and assuming the command is permitted, the administrative command is executed as if run by the **root** user.

Follow this procedure to create a normal user account and give it **sudo** access. You will then be able to use the **sudo** command from this user account to execute administrative commands without logging in to the account of the **root** user.

Procedure 2.2. Configuring sudo Access

1. Log in to the system as the **root** user.
2. Create a normal user account using the **useradd** command. Replace ***USERNAME*** with the user name that you wish to create.

```
# useradd USERNAME
```

3. Set a password for the new user using the **passwd** command.

```
# passwd USERNAME
Changing password for user USERNAME.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

4. Run the **visudo** to edit the **/etc/sudoers** file. This file defines the policies applied by the **sudo** command.

```
# visudo
```

5. Find the lines in the file that grant **sudo** access to users in the group **wheel** when enabled.

```
## Allows people in group wheel to run all commands
# %wheel        ALL=(ALL)       ALL
```

6. Remove the comment character (#) at the start of the second line. This enables the configuration option.
7. Save your changes and exit the editor.
8. Add the user you created to the **wheel** group using the **usermod** command.

```
# usermod -aG wheel USERNAME
```

9. Test that the updated configuration allows the user you created to run commands using **sudo**.
    a. Use the **su** to switch to the new user account that you created.

```
# su USERNAME -
```

    b. Use the **groups** to verify that the user is in the **wheel** group.

```
$ groups
USERNAME wheel
```

    c. Use the **sudo** command to run the **whoami** command. As this is the first time you have run a command using **sudo** from this user account the banner message will be displayed. You will be also be prompted to enter the password for the user account.

```
$ sudo whoami
We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for USERNAME:
root
```

    The last line of the output is the user name returned by the **whoami** command. If **sudo** is configured correctly this value will be **root**.

You have successfully configured a user with **sudo** access. You can now log in to this user account and use **sudo** to run commands as if you were logged in to the account of the **root** user.

2.4. Installing OpenStack Utilities

The *openstack-utils* package provides important utilities used when deploying and managing an OpenStack environment. These utilities include:

openstack-config
: Sets configuration parameters on various OpenStack config files.

openstack-db
: Sets up and initializes the database for OpenStack services.

openstack-status
: Displays service status information.

Procedure 2.3. Installing OpenStack Utilities

- Log in as a user with **sudo** access and install the *openstack-utils* package using the **yum** command:

```
$ sudo yum install -y openstack-utils dnsmasq-utils
```

You have successfully installed the *openstack-utils* package.

Part II. Upgrading an OpenStack Deployment

Chapter 3. Upgrading from Red Hat OpenStack 1.0 (Essex) Preview to Red Hat OpenStack 2.0 (Folsom) Preview

These upgrade steps are the basic requirements for upgrading the simple 2012.1.3 setup (all-in-one Nova + Glance + Keystone) to Folsom Preview.

If you are preparing a new installation of OpenStack, or want to use OpenStack Networking instead of Nova's networking, a fresh install is required. Skip this chapter and either use PackStack (see Part III, “Deploying OpenStack using PackStack” below) or for manual installation proceed to Part IV, “Deploying OpenStack Manually”.

Note

The commands in this section should be run when logged in as root. If not logged in as root, use "sudo" before each command.

1. Stop services:

```
service openstack-keystone stop
service openstack-glance-api stop
service openstack-glance-registry stop
service openstack-nova-compute stop
service openstack-nova-network stop
service openstack-nova-scheduler stop
service openstack-nova-api stop
```

2. If on a Red Hat Enterprise Linux based system, manually update Django:

```
rpm -e --nodeps Django
yum install -y Django14
```

3. Update RPMs:

```
yum update openstack\* python\*client
```

4. If using Nova volumes, update tgtd configuration:

```
sed -i '1iinclude /etc/nova/volumes/*' /etc/tgt/targets.conf
service tgtd restart
```

5. Merge configurations. They're customized when deployed, so you will get .rpmnew.

   In **`/etc/keystone/`**, save the old **`keystone.conf`** and make **`keystone.conf.rpmnew`** the configuration file:

```
mv keystone.conf keystone.conf.old
mv keystone.conf.rpmnew keystone.conf
```

6. Copy **`admin_token`** and the **`[sql]`** section's **`connection`** configurations from **`keystone.conf.old`** to **`keystone.conf`**.
7. In **`/etc/glance/`**, save the old **`glance-api.conf`** and **`glance-api-paste.ini`** and make **`glance-api.conf.rpmnew`** and **`glance-api-paste.ini.rpmnew`** the new .conf and .ini files, respectively.

Do the same for the old **glance-registry.conf** and **glance-registry-paste.ini** files:

```
mv glance-api.conf glance-api.conf.old
mv glance-api-paste.ini glance-api-paste.ini.old
mv glance-api.conf.rpmnew glance-api.conf
mv glance-api-paste.ini.rpmnew glance-api-paste.ini
mv glance-registry.conf glance-registry.conf.old
mv glance-registry-paste.ini glance-registry-paste.ini.old
mv glance-registry.conf.rpmnew glance-registry.conf
mv glance-registry-paste.ini.rpmnew glance-registry-paste.ini
```

Copy **admin_*** from **glance*paste.ini [filter:authtoken]** to **glance*.conf [keystone_authtoken]**,

and verify **sql_connection** in both **glance-registry.conf** and **glance-api.conf**.

8. In **/etc/nova/** , save the old **api-paste.ini** and **nova.conf** and make **api-paste.ini.rpmnew** and **nova.conf.rpmnew** the new .ini and .conf files, respectively:

```
mv api-paste.ini api-paste.ini.old
mv api-paste.ini.rpmnew api-paste.ini
mv nova.conf nova.conf.old
mv nova.conf.rpmnew nova.conf
```

Copy **admin_*** from **api-paste.ini [filter:authtoken]** to **nova.conf [keystone_authtoken]**,

and verify **sql_connection** in **nova.conf**.

9. Verify other configurations you might have customized, for example:

```
/etc/openstack-dashboard/local_settings
```

10. Update databases:

```
keystone-manage db_sync
glance-manage db_sync
nova-manage db_sync
```

11. Start services:

```
service openstack-keystone start
service openstack-glance-api start
service openstack-glance-registry start
service openstack-nova-api start
service openstack-nova-scheduler start
service openstack-nova-compute start
service openstack-nova-network start
```

12. Restart httpd:

```
service httpd restart
```

Chapter 4. Upgrading from Red Hat OpenStack 2.0 (Folsom) Preview to Red Hat OpenStack 2.1 (Folsom)

Users who installed Red Hat OpenStack 2.0 (Folsom) Preview after the release of the generally available version of Red Hat Enterprise Linux 6.4 may follow this procedure to upgrade their systems to Red Hat OpenStack 2.1 (Folsom).

This procedure must be followed on each system in the Red Hat OpenStack environment.

Important

Upgrading from Red Hat OpenStack 2.0 (Folsom) Preview to Red Hat OpenStack 2.1 (Folsom) is not formally supported at this time. This results from the fact that Red Hat OpenStack 2.0 (Folsom) Preview was originally released on the beta version of Red Hat Enterprise Linux 6.4 while Red Hat OpenStack 2.1 (Folsom) requires the generally available version of Red Hat Enterprise Linux 6.4.
Upgrading from beta releases of Red Hat Enterprise Linux to generally available release of Red Hat Enterprise Linux are not supported at this time.
Further details on the support status of systems upgraded from beta releases of Red Hat Enterprise Linux to generally available releases is available in this knowledge base article:

- https://access.redhat.com/knowledge/solutions/21531

Procedure 4.1. Upgrading from Red Hat OpenStack 2.0 (Folsom) Preview to Red Hat OpenStack 2.1 (Folsom)

1. Stop all OpenStack services that are currently active on the system.
    a. Use the **`openstack-status`** command to identify active OpenStack services.

```
$ sudo openstack-status
== Nova services ==
openstack-nova-api           active
openstack-nova-cert          active
openstack-nova-compute       inactive
openstack-nova-network       active
openstack-nova-scheduler     active
openstack-nova-volume        inactive (disabled on boot)
== Glance services ==
openstack-glance-api         active
openstack-glance-registry    active
== Keystone service ==
openstack-keystone           active
== Horizon service ==
openstack-dashboard          active
== Cinder services ==
openstack-cinder-api         active
openstack-cinder-scheduler   active
openstack-cinder-volume      inactive
== Support services ==
httpd:                       active
libvirtd:                    active
tgtd:                        active
qpidd:                       active
memcached:                   active
== Keystone users ==
Warning keystonerc not sourced
```

Note

When the **openstack-status** command is run while Keystone environment variables are set additional information will be displayed. This information is not required to complete this procedure.

b. Use the **service** command to stop each active service that has a name that starts with **openstack**.

```
$ sudo service openstack-COMPONENT stop
```

2. Use the **subscription-manager** command to verify that the system has subscriptions that provide both **Red Hat Enterprise Linux Server** and **Red Hat OpenStack** entitlements.

```
$ sudo subscription-manager list --consumed
+-------------------------------------------+
   Consumed Subscriptions
+-------------------------------------------+

Subscription Name:      Red Hat OpenStack
Provides:               Red Hat OpenStack
                        Red Hat Enterprise Linux Server
SKU:                    SER0406
Contract:               3169240
Account:                901578
Serial Number:          1667264867340998574
Active:                 True
Quantity Used:          1
Service Level:          None
Service Type:           None
Starts:                 08/12/2012
Ends:                   08/12/2013
```

If the **Red Hat Enterprise Linux Server** and **Red Hat OpenStack** entitlements are not shown then follow the steps shown in Section 2.2, "Configuring Software Repositories" to subscribe the system to them.

3. Use the **yum-config-manager** command to ensure that the system does not have the Red Hat Enterprise Linux 6 beta repositories enabled.

```
$ sudo yum-config-manager --disable rhel-6-server-beta-rpms
Loaded plugins: product-id
========== repo: rhel-6-server-beta-rpms ==========
[rhel-6-server-beta-rpms]
bandwidth = 0
base_persistdir = /var/lib/yum/repos/x86_64/6Server
baseurl = https://cdn.redhat.com/content/beta/rhel/server/6/6Server/x86_64/os
cache = 0
cachedir = /var/cache/yum/x86_64/6Server/rhel-6-server-beta-rpms
cost = 1000
enabled = False
...
```

Note

Yum treats the values **False** and **0** as equivalent. As a result the output on your system may instead contain this string:

```
enabled = 0
```

4. Use the **yum-config-manager** command to ensure that the system has the Red Hat Enterprise Linux 6 repositories enabled.

```
$ sudo yum-config-manager --enable rhel-6-server-rpms
Loaded plugins: product-id
========== repo: rhel-6-server-rpms ==========
[rhel-6-server-rpms]
bandwidth = 0
base_persistdir = /var/lib/yum/repos/x86_64/6Server
baseurl = https://cdn.redhat.com/content/dist/rhel/server/6/6Server/x86_64/os
cache = 0
cachedir = /var/cache/yum/x86_64/6Server/rhel-6-server-rpms
cost = 1000
enabled = True
...
```

Note

Yum treats the values **True** and **1** as equivalent. As a result the output on your system may instead contain this string:

```
enabled = 1
```

5. Use the **yum-config-manager** command to ensure that the system has the Red Hat OpenStack 2.1 (Folsom) software repositories enabled.

```
$ sudo yum-config-manager --enable rhel-server-ost-6-folsom-rpms
Loaded plugins: product-id
========== repo: rhel-server-ost-6-folsom-rpms ==========
[rhel-server-ost-6-folsom-rpms]
bandwidth = 0
base_persistdir = /var/lib/yum/repos/x86_64/6Server
baseurl =
https://cdn.redhat.com/content/dist/rhel/server/6/6Server/x86_64/openstack/fo
lsom/os
cache = 0
cachedir = /var/cache/yum/x86_64/6Server/rhel-server-ost-6-folsom-rpms
cost = 1000
enabled = True
...
```

6. Use the **yum** command to ensure that your system has the most up to date versions of all Red Hat Enterprise Linux and Red Hat OpenStack packages.

```
$ sudo yum update -y
```

7. When updating packages Yum attempts to also update their configuration files. In some cases when performing this task Yum finds a conflict which it is unable to resolve. In these cases Yum chooses one of the following actions depending on the options specified by the package for deploying its configuration files:
   - Creates a copy of the original configuration file from before the update commenced in a file with the suffix **.rpmsave**.

     Any user defined configuration values found in the **.rpmsave** file must be manually merged into the base configuration file.
   - Creates a copy of the new configuration file from after the update in a file with the suffix

.rpmnew.

Any package defined configuration values found in the **.rpmnew** file must be manually merged into the base configuration file.

Use the **find** command to identify any instances where the update identified conflicting changes to configuration files.

```
$ sudo find /etc/ -name '*.rpm?*'
```

Use the **diff** command to compare each file to the base configuration file. Then manually merge the conflicting changes from the **.rpmsave** and **.rpmnew** files into the base configuration file.

8. Use the **service** command to start all OpenStack services that were stopped in preparation for the upgrade.

```
$ sudo service openstack-COMPONENT start
```

You have successfully upgraded your environment to Red Hat OpenStack 2.1 (Folsom).

Part III. Deploying OpenStack using PackStack

PackStack is a command line utility that uses Puppet (http://www.puppetlabs.com/) modules to support rapid deployment of OpenStack on existing servers over an SSH connection. PackStack is suitable for deploying both single node proof of concept installations and more complex multi-node installations.

Deployment options are provided either interactively, via the command line, or via a text file containing preconfigured answers to the questions PackStack asks.

Chapter 5. Installing PackStack

The **packstack** utility is provided by the *openstack-packstack* package. Follow this procedure to install the *openstack-packstack* package.

Procedure 5.1. Installing PackStack

1. Use the **yum** command to install the *openstack-packstack* package.

```
$ sudo yum install -y openstack-packstack
```

2. Use the **which** command to verify that the **packstack** utility is now available.

```
$ which packstack
/usr/bin/packstack
```

The *openstack-packstack* package which provides the **packstack** utility is now installed. Proceed to Chapter 6, *Running PackStack* for information on prerequisites and running **packstack** for the first time.

Chapter 6. Running PackStack

There are several different methods of running **packstack** and using it to deploy OpenStack. If you just want to deploy an OpenStack environment as quickly as possible for evaluation purposes then refer to Section 6.1, “Quick Start Deployment using PackStack”.

These methods of running **packstack** are provided for more complex deployments:

Interactively

When run interactively **packstack** provides prompts for entry of each configuration value required to complete deployment. Alternatively the user may accept the provided default value.

Refer to Section 6.2, “Running PackStack Interactively” for more information on running **packstack** interactively.

Non-interactively

When run non-interactively **packstack** expects an "answer" file to be provided as a command line option. This file contains the desired settings for all configuration values that are required to complete deployment.

Refer to Section 6.3, “Running PackStack Non-interactively” for more information on generating an answer file and using it to run **packstack** non-interactively.

Important

To deploy OpenStack using **packstack** each machine targeted for deployment must be configured to allow access using the account of the **root** user over SSH on port **22**.

Important

By default **packstack** will configure a volume group named **cinder-volumes** on the system targeted for volume storage (Cinder) deployment if one does not already exist. This volume group will be backed by a loopback device and is not appropriate for production use.
If you intend to use physical storage for the **cinder-volumes** volume group then you must create the volume group in advance on the system to be used for Cinder.

Important

It is strongly recommended that each compute node has two network interfaces available. One for access to the public network and one for the internal Nova network. While it is possible to use a single interface for both purposes, this approach may result in virtual machine instances obtaining addresses from the wrong DHCP server.

6.1. Quick Start Deployment using PackStack

The quickest way to deploy an OpenStack environment using **packstack** is to provide a host, or list of

hosts, on the command line. The first host listed will be deployed as a compute controller node, subsequent hosts will be deployed as compute nodes.

When using this deployment method **packstack** will use default values for all other deployment options unless they are overridden on the command line.

For a list of available command line options refer to Table 6.1, “PackStack Configuration Keys”.

Procedure 6.1. Quick Start Deployment using PackStack

1. Run **packstack** with the ***--install-hosts*** parameter. The parameter expects a comma separated list of IP addresses.

   ```
   $ packstack --install-hosts=CONTROLLER_ADDRESS,NODE_ADDRESSES
   ```

   Replace ***CONTROLLER_ADDRESS*** with the IP address of the system that you intend to use as a compute controller. Replace ***NODE_ADDRESSES*** with IP addresses of the systems that you intend to use as compute nodes.

   Alternatively provide only a single IP address to deploy an "all in one" OpenStack installation, even on the system from which you are running **packstack**.

   > **Example 6.1. Single Node Deployment**
   >
   > In this example **packstack** is instructed to deploy an "all in one" installation to the system with IP address **192.0.43.10**.
   >
   > ```
   > $ packstack --install-hosts=192.168.1.0
   > ```

   > **Example 6.2. Multiple Node Deployment**
   >
   > In this example **packstack** is instructed to deploy a controller node on the system with IP address **192.168.1.0**.
   >
   > Additional compute nodes are deployed on the systems with IP addresses **192.168.1.1** and **192.168.1.2**.
   >
   > ```
   > $ packstack --install-hosts=192.168.1.0,192.168.1.1,192.168.1.2
   > ```

2. The **packstack** utility will prompt you to enter the password of the **root** user for each system in the deployment. This is required to connect to the system and install Puppet which is the tool used to facilitate the rest of the deployment.

   ```
   root@192.168.1.0's password:
   ```

3. The Puppet manifests used to deploy each component will be run on each of the target systems. The amount of time this takes to complete varies based on the hardware and existing workload of each system. It can be significant.

   When the deployment has successfully completed this message is displayed:

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

You have successfully deployed an OpenStack environment using **packstack**. Please note that:

- An answer file containing all chosen configuration options is saved to disk on the system from which you ran **packstack**. This file can be used to automate future deployments.

```
 * A new answerfile was created in: /root/packstack-answers-20130306-051043.txt
```

- A file containing the authentication details of the OpenStack **admin** user is saved to disk on the system to which the OpenStack client tools were deployed. You will need these details to manage the OpenStack environment.

```
 * To use the command line tools you need to source the file
/root/keystonerc_admin created on 192.168.1.0
```

Refer to Part V, “Using OpenStack” to begin using your OpenStack environment.

6.2. Running PackStack Interactively

Deploy OpenStack by running the **packstack** utility interactively. The **packstack** utility supports the creation of both single node and multiple node OpenStack deployments.

Procedure 6.2. Running PackStack Interactively

1. **Run packstack**

   Run the **packstack** command to commence the deployment process. Optionally append the ***--debug*** parameter to enable additional logging.

   ```
   $ packstack
   ```

   Important

   You are not required to log in as the **root** user to run the **packstack** command itself. However you will be required to provide **root** credentials for each machine to which you choose to deploy services.

2. **Configuring Public Key**

   Each server involved in the OpenStack deployment is configured for key-based authentication. If you already have a public key that you wish to use for this, enter the path to it. If you do not, then press **Enter** and the utility will generate one for you and save it to **~/.ssh/id_rsa.pub**.

   ```
   Enter the path to your ssh Public key to install on servers:
   ```

3. **Selecting the Services to Install**

   The **packstack** script will prompt you to select the OpenStack services that you want to install and configure. At each prompt enter **y** to install the service, enter **n** to skip the service, or press **Enter** to select the default option listed in square brackets (**[**, **]**).

   ```
   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] [y] :
   ```

   Each selected service can be deployed on either a local or remote system. Where each service deploys to will be determined based on the IP addresses you provide later in the deployment process.

4. OpenStack includes a number of client tools. Enter **y** to install the client tools. A file containing the authentication values of the administrative user will also be created.

```
Should Packstack install OpenStack client tools [y|n] [y] :
```

5. Optionally, the **packstack** script will configure all servers in the deployment to retrieve date and time information using Network Time Protocol (NTP). To use this facility enter a comma separated pool of NTP servers.

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

> **Example 6.3. Using the Default Red Hat Enterprise Linux NTP Servers**
>
> ```
> Enter list of NTP server(s). Leave plain if packstack should not
> install ntpd on instances.: 0.rhel.pool.ntp.org, 1.rhel.pool.ntp.org
> ```

6. Optionally, the **packstack** script will install and configure Nagios to provide advanced facilities for monitoring the nodes in the OpenStack environment.

```
Should Packstack install Nagios to monitor openstack hosts [y|n] [n] :
```

7. **Configuring the MySQL Instance**

   OpenStack services require a MySQL database to store data in. To configure the database:

   a. Enter the IP address of the server to deploy the MySQL database server on.

   ```
   Enter the IP address of the MySQL server [192.0.43.10] :
   ```

   b. Enter the password to use for the MySQL administrative user. If you do not enter a value it will be randomly generated. The generated password will be available both in the **~/.my.cnf** file of the current user and the answer file.

   ```
   Enter the password for the MySQL admin user :
   ```

8. **Configuring Qpid**

   OpenStack services use the Qpid (http://qpid.apache.org/) messaging system to communicate. Enter the IP address of the server to deploy Qpid on.

   ```
   Enter the IP address of the QPID service  [192.0.43.10] :
   ```

9. **Configuring Keystone**

   OpenStack uses Keystone (**openstack-keystone**) for identity, token, catalog, and policy services. If Keystone installation was selected then enter the IP address of the server to deploy Keystone on when prompted.

   ```
   Enter the IP address of the Keystone server  [192.0.43.10] :
   ```

10. **Configuring Glance**

    OpenStack uses Glance (**openstack-glance-***) to store, discover, and retrieve virtual machine images. If Glance installation was selected then enter the IP address of the server to deploy Glance on when prompted.

```
Enter the IP address of the Glance server  [192.0.43.10] :
```

11. **Configuring Cinder**

    OpenStack uses Cinder (**openstack-cinder-***) to provide volume storage services. Enter the IP address of the server to deploy Cinder on. If installation of the volume services was selected then these additional configuration prompts will be presented.

    ```
    Enter the IP address of the Cinder server  [192.0.43.10] :
    ```

    a. The **packstack** utility expects storage for use with Cinder to be available on a volume group named **cinder-volumes**. If this volume group does not already exist then you will be asked if you want it to be created automatically.

    Answering yes means that **packstack** will create a raw disk image in the **/var/lib/cinder** and mount it for use by Cinder using a loopback device.

    ```
    Should Cinder's volumes group be createdi (for proof-of-concept
    installation)? [y|n] [y]:
    ```

    b. If you elected to have **packstack** create the **cinder-volumes** volume group for you then you will be prompted to enter the size of it in gigabytes (GB).

    ```
    Enter Cinder's volume group size  [20G] :
    ```

    Important

    The amount of space selected must be available on the device used for **/var/lib/cinder**.
    Remember that the size of the Cinder volume group will restrict the amount of disk space that you can expose to compute instances.

12. **Configuring Nova**

    OpenStack uses Nova to provide compute services. Nova is itself made up of a number of complementary services that must be deployed. If installation of the compute services was selected then these additional configuration prompts will be presented.

    a. The Nova API service (**openstack-nova-api**) provides web service endpoints for authenticating and interacting with the OpenStack environment over HTTP or HTTPS. Enter the IP address of the server to deploy the Nova API service on.

    ```
    Enter the IP address of the Nova API service  [192.0.43.10] :
    ```

    b. Nova includes a certificate management service (**openstack-nova-cert**). Enter the IP address of the server to deploy the Nova certificate management service on.

    ```
    Enter the IP address of the Nova Cert service  [192.0.43.10] :
    ```

    c. The Nova VNC proxy provides facilities for connecting users of the Nova compute service to their instances running in the OpenStack cloud. Enter the IP address for the server to deploy the Nova VNC proxy on.

    ```
    Enter the IP address of the Nova VNC proxy  [192.0.43.10] :
    ```

d. The **packstack** script is able to deploy one or more compute nodes. Enter a comma separated list containing the IP addresses or hostnames of all of the nodes that you wish to deploy compute services on.

```
Enter a comma separated list of IP addresses on which to install the Nova
Compute services  [10.15.24.136] :
```

e. A private interface must be configured to provide DHCP services on the Nova compute nodes. Enter the name of the private interface to use.

```
Enter the Private interface for Flat DHCP on the Nova compute servers
[eth1] :
```

f. The Nova network service (**openstack-nova-network**) provides network services for compute instances. Enter the IP address of the server to deploy the Nova Network service on.

```
Enter the IP address of the Nova Network service  [192.0.43.10] :
```

Important

The Nova networking service is incompatible with the OpenStack Network Service added in the Folsom release.

g. A public interface must be configured to allow connections from other nodes and clients. Enter the name of the public interface to use.

```
Enter the Public interface on the Nova network server  [eth0] :
```

h. A private interface must be configured to provide DHCP services on the Nova network server. Enter the name of the private interface to use.

```
Enter the Private interface for Flat DHCP on the Nova network server
[eth1] :
```

i. All compute instances are automatically assigned a private IP address. Enter the range from which these private IP addresses must be assigned.

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

j. Compute instances can optionally be assigned publicly accessible *floating* IP addresses. Enter the range from which floating IP addresses will be assigned.

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

k. The Nova scheduler (**openstack-nova-scheduler**) is used to map compute requests to compute resources. Enter the IP address of the server to deploy the Nova scheduler on.

```
Enter the IP address of the Nova Scheduler service  [192.0.43.10] :
```

l. In the default configuration Nova allows for overcommitment of physical CPU and memory resources. This means that more of these resources are made available for running instances than actually physically exist on the compute node.

The amount of overcommitment that is permitted is configurable.

a. The default level of CPU overcommitment allows 16 virtual CPUs to be allocated for each physical CPU socket or core that exists on the physical compute node. Press **Enter** to accept the default or enter a different value if desired.

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

b. The default level of memory overcommitment allows up to 50% more virtual memory to be allocated than exists on the physical compute node. Press **Enter** to accept the default or enter a different value if desired.

```
Enter the RAM overcommitment ratio. Set to 1.0 to disable RAM
overcommitment [1.5] :
```

13. **Configuring Client Tools**

If installation of the client tools was selected then enter the IP address of the server to install the client tools on when prompted.

```
Enter the IP address of the client server  [192.0.43.10] :
```

14. **Configuring Horizon Dashboard**

OpenStack uses Horizon (**openstack-dashboard**) to provide a web-based user interface or dashboard for accessing OpenStack services including Cinder, Nova, Swift, and Keystone. If installation of Horizon dashboard was selected then these additional configuration values will be requested.

a. Enter the IP address of the server to deploy Horizon on.

```
Enter the IP address of the Horizon server  [192.0.43.10] :
```

b. To enable HTTPS communication with the dashboard enter **y** when prompted. Enabling this option ensures that user access to the dashboard is encrypted.

```
Would you like to set up Horizon communication over https [y|n] [n]
:
```

15. **Configuring Swift Object Storage**

If installation of Swift object storage was selected then these additional configuration values will be requested.

a. Enter the IP address of the server that is to act as the Swift proxy. This server will act as the public link between clients and the Swift object storage.

```
Enter the IP address of the Swift proxy service  [10.64.15.166] :
```

b. Enter a comma separated list of devices that Swift object storage will use to store objects. Each entry must be specified in ***HOST/DEVICE*** format where ***HOST*** is replaced by the IP address of the host the device is attached to and ***DEVICE*** is replaced by the path to the device.

```
Enter the Swift Storage servers e.g. host/dev,host/dev  [10.64.15.166] :
```

c. Swift object storage uses zones to ensure that each replica of a given object is stored separately. A zone might represent an individual disk drive or array, a server, all the servers

in a rack, or even an entire data center.

When prompted enter the number of Swift storage zones that must be defined. Note that the number provided must not be bigger than the number of individual devices specified.

```
Enter the number of swift storage zones, MUST be no bigger than the
number of storage devices configured  [1] :
```

d. Swift object storage relies on replication to maintain the state of objects even in the event of a storage outage in one or more of the configured storage zones. Enter the number of replicas that Swift must keep of each object when prompted.

A minimum of three (3) replicas is recommended to ensure a reasonable degree of fault tolerance in the object store. Note however that it the number of replicas specified must not be greater than the number of storage zones as this would result in one or more of the zones containing multiple replicas of the same object.

```
Enter the number of swift storage replicas, MUST be no bigger than the
number of storage zones configured  [1] :
```

e. Currently **packstack** supports the use of either Ext4 or XFS filesystems for object storage. The default and recommended choice is **ext4**. Enter the desired value when prompted.

```
Enter FileSystem type for storage nodes [xfs|ext4]  [ext4] :
```

16. **Configuring Software Sources**

The **packstack** utility allows you to configure the target servers to retrieve software packages from a number of sources.

a. **Enabling Custom Software Repositories**

The **packstack** utility allows you to optionally configure each server to retrieve updates from additional custom software repositories. Enter the URL for the directory containing the **repodata** folder of each desired repository at the prompt, separated by a comma.

```
Enter a comma separated list of URLs to any additional yum repositories
to install:
```

b. **Enabling Red Hat Network Subscription**

Enter your Red Hat Network account details when prompted. This will ensure each server involved in the deployment is subscribed to receive updates from Red Hat Network.

```
To subscribe each server to Red Hat enter a username here:
```

```
To subscribe each server to Red Hat enter your password here:
```

Important

The **packstack** utility registers systems to Red Hat Network using Subscription Manager or Red Hat Network Satellite. You may encounter problems if your systems have already been registered and subscribed to the Red Hat OpenStack channels using RHN Classic.

c. **Enabling the Red Hat Enterprise Linux Beta Channel**

To enable the Red Hat Enterprise Linux Beta channel enter **y** when prompted. Note that selecting this option is not recommended at this time but may be required by future Red Hat OpenStack preview releases.

```
 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]
:
```

d. **Enabling Red Hat Network Satellite**

The **packstack** utility allows you to optionally configure each server to retrieve updates from a Red Hat Network Satellite server.

Enter the URL of the Red Hat Network Satellite server that you wish to use when prompted. If you do not wish to use a Red Hat Satellite server then do not enter a value.

```
To subscribe each server with RHN Satellite enter RHN Satellite server
URL :
```

If an RHN Satellite URL is provided a number of follow up prompts will be displayed.

a. Red Hat Network Satellite supports authentication using a user name and password or an activation key. If your Satellite administrator provided you with a user name and password enter them when prompted. If your Satellite administrator provided you with an access key then do not enter a value.

```
Enter RHN Satellite username or leave plain if you will use
activation key instead :
```

```
Enter RHN Satellite password or leave plain if you will use
activation key instead :
```

b. If your Satellite administrator provided you with an access key then enter it when prompted. Otherwise do not enter a value.

```
Enter RHN Satellite activation key or leave plain if you used
username/password instead :
```

c. Specify the path to the certificate of the certificate authority that is used to verify that the connection with the Satellite server is secure.

```
Specify a path or URL to a SSL CA certificate to use :
```

d. Specify the profile name that must be used to identify the system in Red Hat Network. This is optional.

```
If required specify the profile name that should be used as an
identifier for the system in RHN Satellite :
```

e. Specify the HTTP proxy that must be used when connecting to the Satellite server. If no proxy is required then do not enter a value.

```
Specify a HTTP proxy to use with RHN Satellite :
```

f. Specify the user name for authenticating with the HTTP proxy that must be used when connecting to the Satellite server. If no proxy is required or the chosen proxy does not require authentication then do not enter a value.

```
Specify a username to use with an authenticated HTTP proxy :
```

g. Specify the password for authenticating with the HTTP proxy server that must be used when connecting to the Satellite server. If no proxy is required or the chosen proxy does not require authentication then do not enter a value.

```
Specify a password to use with an authenticated HTTP proxy. :
```

h. Specify any additional Satellite flags that you need to be passed to the **rhnreg_ks** command when it is run on each system. This configuration key accepts a comma separated list of flags. Valid flags are **novirtinfo**, **norhnsd**, and **nopackages**.

Refer to the Red Hat Satellite documentation for more information. If unsure do not enter a value.

```
Enter comma separated list of flags passed to rhnreg_ks :
```

17. At this point you will be asked to confirm the deployment details that you provided. Type **yes** and press **Enter** to continue with the deployment.

```
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-servers:
mysql-host:                      192.0.43.10
mysql-pw:                        ********
qpid-host:                       192.0.43.10
keystone-host:                  192.0.43.10
glance-host:                    192.0.43.10
cinder-host:                    192.0.43.10
cinder-volumes-create:          y
cinder-volumes-size:            20G
novaapi-host:                   192.0.43.10
novacert-host:                  192.0.43.10
novavncproxy-hosts:             192.0.43.10
novacompute-hosts:              192.0.43.10
novacompute-privif:             eth1
novanetwork-host:               192.0.43.10
novanetwork-pubif:              eth0
novanetwork-privif:             eth1
novanetwork-fixed-range:        192.168.32.0/22
novanetwork-floating-range:    10.3.4.0/22
novasched-host:                 192.0.43.10
novasched-cpu-allocation-ratio:16.0
novasched-ram-allocation-ratio:1.5
osclient-host:                  192.0.43.10
os-horizon-host:                192.0.43.10
os-horizon-ssl:                 n
os-swift-proxy:                 10.64.15.166
os-swift-storage:               10.64.15.166
os-swift-storage-zones:         1
os-swift-storage-replicas:      1
os-swift-storage-fstype:        ext4
additional-repo:
rh-username:                     admin@example.com
rh-password:                     ********
rhn-satellite-server:
Proceed with the configuration listed above? (yes|no): yes
```

18. At this point **packstack** will commence deployment. Note that when **packstack** is setting up SSH keys it will prompt you to enter the **root** password to connect to machines that are not already configured to use key authentication.

```
Installing:
Clean Up...                                             [ DONE ]
Running Pre install scripts...                          [ DONE ]
Setting Up ssh keys...                                  [ DONE ]
Create MySQL Manifest...                                [ DONE ]
Creating QPID Manifest...                               [ DONE ]
Creating Keystone Manifest...                           [ DONE ]
Adding Glance Keystone Manifest entries...              [ DONE ]
Creating Galnce Manifest...                             [ DONE ]
Adding Cinder Keystone Manifest entries...              [ DONE ]
Checking if the Cinder server has a cinder-volumes vg... [ DONE ]
Creating Cinder Manifest...                             [ DONE ]
Adding Nova API Manifest entries...                     [ DONE ]
Adding Nova Keystone Manifest entries...                [ DONE ]
Adding Nova Cert Manifest entries...                    [ DONE ]
Adding Nova Compute Manifest entries...                 [ DONE ]
Adding Nova Network Manifest entries...                 [ DONE ]
Adding Nova Scheduler Manifest entries...               [ DONE ]
Adding Nova VNC Proxy Manifest entries...               [ DONE ]
Adding Nova Common Manifest entries...                  [ DONE ]
Creating OS Client Manifest...                          [ DONE ]
Creating OS Horizon Manifest...                         [ DONE ]
Preparing Servers...                                    [ DONE ]
Running Post install scripts...                         [ DONE ]
Installing Puppet...                                    [ DONE ]
Copying Puppet modules/manifests...                     [ DONE ]
Applying Puppet manifests...
```

19. Applying the Puppet manifests to all machines involved in the deployment takes a significant amount of time. The **packstack** utility provides continuous updates indicating which manifests are being deployed as it progresses through the deployment process. Once the process completes a confirmation message will be displayed.

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

     (Please allow Installer a few moments to start up.....)


Additional information:
 * A new answerfile was created in: /root/packstack-answers-20130417-
042816.txt
 * Time synchronization was skipped. Please note that unsynchronized time on
server instances might be a problem for some OpenStack components.
 * To use the command line tools source the file /root/keystonerc_admin
created on 192.0.43.10
 * To use the console, browse to http://192.0.43.10/dashboard
 * The installation log file is available at: /var/tmp/packstack/20130417-
042816-TkY04B/openstack-setup.log
You have mail in /var/spool/mail/root
```

You have successfully deployed OpenStack using **packstack**.

The configuration details that you provided are also recorded in an answer file that can be used to recreate the deployment in future. The answer file is stored in **~/answers.txt** by default.

Refer to Section 6.3, “Running PackStack Non-interactively” for more information on using answer files to automate deployment.

Warning

The answer file also contains a number of required configuration values that are automatically generated if you choose not to provide them including the administrative password for MySQL. It is recommended that you store the answer file in a secure location.

Important

Currently **packstack** does not support deployment of OpenStack networking. If you have deployed OpenStack using **packstack** and wish to use OpenStack Networking, then refer to Chapter 12, *Deploying OpenStack Networking Services* for more information.
It is important that such a conversion is performed **before** you begin creating and configuring compute instances with the Nova networking deployment created by **packstack**.

6.3. Running PackStack Non-interactively

PackStack supports being run non-interactively. When you run the **packstack** command non-interactively you must provide your configuration options via a text file, referred to as an answer file, instead of via standard input.

To do this you must:

- Use PackStack to generate a default answer file.
- Edit the answer file inserting your desired configuration values.
- Run the **packstack** command providing the completed answer file as a command line argument.

PackStack will then attempt to complete the deployment using the configuration options provided in the answer file.

6.3.1. Generating a PackStack Answer File

PackStack is able to generate a generic answer file which you are then able to customize to suit your specific deployment needs.

Procedure 6.3. Generating a PackStack Answer File

- Run the **packstack** command with the ***--gen-answer-file=FILE*** argument to generate an answer file. Replace ***FILE*** with the name of the file you wish to use to store the answer file.

```
$ sudo packstack --gen-answer-file=FILE
```

> **Example 6.4. Generating a PackStack Answer File**
>
> In this example a PackStack answer file is generated and saved to the file **~/answers.cfg**.
>
> ```
> $ sudo packstack --gen-answer-file=~/answers.cfg
> ```

You have successfully generated an answer file and are ready to begin customizing it for your deployment.

6.3.2. Editing a PackStack Answer File

PackStack answer files are editable in any text editor. Lines preceded with a # character are treated as comments and are ignored.

The table presented here lists the configuration keys available. Configuration values are provided in the answer files as key-value pairs of the form:

```
KEY=VALUE
```

Where a key accepts multiple comma separated values that is noted in the description of the configuration key. Some configuration keys also have command line equivalents, allowing them to be provided directly as arguments to the invocation of the **packstack** command. Where this is the case the command line argument is also listed in the table.

Table 6.1. PackStack Configuration Keys

Configuration Key	Command Line Argument	Default Value	Description
CONFIG_SSH_KEY	**--sh-public-key**	***/root/.ssh/id_rsa.pub***	Path to a Public key to install on servers. If a usable key has not been installed on the remote servers the user will be prompted for a password and this key will be installed so the password will not be required again.
CONFIG_GLANCE_INSTALL	**--os-glance-install**	***y***	Set to **y** if you would like Packstack to install Glance
CONFIG_CINDER_INSTALL	**--os-cinder-install**	***y***	Set to **y** if you would like Packstack to install Cinder
CONFIG_NOVA_INSTALL	**--os-nova-install**	***y***	Set to **y** if you would like Packstack to install Nova
CONFIG_HORIZON_INSTALL	**--os-horizon-install**	***y***	Set to **y** if you would like Packstack to install Horizon
CONFIG_SWIFT_INSTALL	**--os-swift-install**	***n***	Set to **y** if you would like Packstack to install Swift
CONFIG_CLIENT_INSTALL	**--os-client-install**	***y***	Set to **y** if you would like Packstack to install the openstack client packages. An admin "rc" file will also be installed
CONFIG_NTP_SERVERS	**--ntp-servers**		Comma separated list of NTP servers. Leave plain if packstack should not install ntpd on instances.
CONFIG_NAGIOS_INSTALL	**--nagios-install**	***n***	Set to **y** if you would like to install Nagios. Nagios provides additional tools for monitoring the OpenStack environment.
CONFIG_MYSQL_HOST	**--mysql-host**	***192.0.43.10***	The IP address of the server on which to install MySQL.

CONFIG_MYSQL_USER		***root***	Username for the MySQL administrative user.
CONFIG_MYSQL_PW	**--mysql-pw**		Password for the MySQL administrative user. This value is randomly generated if you do not provide it.
CONFIG_QPID_HOST	**--qpid-host**	***192.0.43.10***	The IP address of the server on which to install the QPID service.
CONFIG_KEYSTONE_HOST	**--keystone-host**	***192.0.43.10***	The IP address of the server on which to install Keystone.
CONFIG_KEYSTONE_DB_PW			The password to use for the Keystone to access database. This value is randomly generated if you do not provide it.
CONFIG_KEYSTONE_ADMINTOKEN			The token to use for the Keystone service API. This value is randomly generated if you do not provide it.
CONFIG_KEYSTONE_ADMINPASSWD			The password to use for the Keystone administrative user. This value is randomly generated if you do not provide it.
CONFIG_GLANCE_HOST	**--glance-host**	***192.0.43.10***	The IP address of the server on which to install Glance.
CONFIG_GLANCE_DB_PW			The password to use for the Glance to access database. This value is randomly generated if you do not provide it.
CONFIG_GLANCE_KS_PW			The password to use for the Glance to authenticate with Keystone. This value is randomly generated if you do not provide it.
CONFIG_CINDER_HOST	**--cinder-host**	***192.0.43.10***	The IP address of the server on which to install Cinder.
CONFIG_CINDER_DB_PW			The password to use for the Cinder to

			access database. This value is randomly generated if you do not provide it.
CONFIG_CINDER_KS_PW			The password to use for the Cinder to authenticate with Keystone. This value is randomly generated if you do not provide it.
CONFIG_CINDER_VOLUMES_CREATE	**--cinder-volumes-create**	***y***	The **packstack** utility expects storage for use with Cinder to be available on a volume group named **cinder-volumes**. If this volume group does not already exist then **packstack** is able to create it automatically. Selecting **y** means that **packstack** will create raw disk image in the **/var/lib/cinder** and mount it for use by Cinder using a loopback device.
CONFIG_CINDER_VOLUMES_SIZE	**--cinder-volumes-size**	***20G***	If you elected to have **packstack** create the **cinder-volumes** volume group for you then you will need to provide the desired size of it in gigabytes (GB).
CONFIG_NOVA_API_HOST	**--novaapi-host**	***192.0.43.10***	The IP address of the server on which to install the Nova API service.
CONFIG_NOVA_CERT_HOST	**--novacert-host**	***192.0.43.10***	The IP address of the server on which to install the Nova Certificate service.
CONFIG_NOVA_VNCPROXY_HOST	**--novavncproxy-hosts**	***192.0.43.10***	The IP address of the server on which to install the Nova VNC proxy.
CONFIG_NOVA_COMPUTE_HOSTS	**--novacompute-hosts**	***192.0.43.10***	A comma separated list of IP addresses on which to install the

			Nova Compute services.
CONFIG_NOVA_COMPUTE_PRIVIF	**--novacompute-privif**	***eth1***	Private interface for Flat DHCP on the Nova compute servers.
CONFIG_NOVA_NETWORK_HOST	**--novanetwork-host**	***192.0.43.10***	The IP address of the server on which to install the Nova Network service.
CONFIG_NOVA_DB_PW			The password to use for the Nova to access the database. This value is randomly generated if you do not provide it.
CONFIG_NOVA_KS_PW			The password to use for the Nova to authenticate with Keystone. This value is randomly generated if you do not provide it.
CONFIG_NOVA_NETWORK_PUBIF	**--novanetwork-pubif**	***eth0***	Public interface on the Nova network server.
CONFIG_NOVA_NETWORK_PRIVIF	**--novanetwork-privif**	***eth1***	Private interface for Flat DHCP on the Nova network server.
CONFIG_NOVA_NETWORK_FIXEDRANGE	**--novanetwork-fixed-range**	***192.168.32.0/22***	IP Range for Flat DHCP.
CONFIG_NOVA_NETWORK_FLOATRANGE	**--nova-network-floating-range**	***10.3.4.0/22***	IP Range for Floating IP addresses.
CONFIG_NOVA_SCHED_HOST	**--novasched-host**	***192.0.43.10***	The IP address of the server on which to install the Nova Scheduler service.
CONFIG_NOVA_SCHED_CPU_ALLOC_RATIO	**--novasched-cpu-allocation-ratio**	***16.0***	The overcommitment ratio for virtual to physical CPUs. Set to 1.0 to disable CPU overcommitment.
CONFIG_NOVA_SCHED_RAM_ALLOC_RATIO	**--novasched-ram-allocation-ratio**	***1.5***	The overcommitment ratio for virtual to physical RAM. Set to 1.0 to disable RAM overcommitment.
CONFIG_OSCLIENT_HOST	**--osclient-host**	***192.0.43.10***	The IP address of the server on which to install the openstack client packages. An admin "rc" file will also be installed.

CONFIG_HORIZON_HOST	**--os-horizon-host**	***192.0.43.10***	The IP address of the server on which to install Horizon.
CONFIG_HORIZON_SSL	**--os-horizon-ssl**	***n***	To set up Horizon communication over HTTPS set this to **y**.
CONFIG_SSL_CERT	**--os-ssl-cert**		PEM encoded certificate to be used for SSL connections to the HTTPS server, leave blank if one should be generated. This certificate must not require a passphrase.
CONFIG_SSL_KEY	**--os-ssl-key**		Keyfile corresponding to the certificate if one was provided.
CONFIG_SWIFT_PROXY_HOSTS	**--os-swift-proxy**	***192.0.43.10***	The IP address on which to install the Swift proxy service.
CONFIG_SWIFT_KS_PW			The password to use for the Swift to authenticate with Keystone. This value is randomly generated if you do not provide it.
CONFIG_SWIFT_STORAGE_HOSTS	**--os-swift-storage**	***192.0.43.10***	A comma separated list of IP addresses on which to install the Swift Storage services, each entry should take the format ***IP***[/***DEVICE***], for example **192.0.43.10/vdb** will install **/dev/vdb** on **192.0.43.10** as a swift storage device, if /***DEVICE*** is omitted Packstack will create a loopback device for a test setup.
CONFIG_SWIFT_STORAGE_ZONES	**--os-swift-storage-zones**	***1***	Number of swift storage zones, this number **must** be no bigger than the number of storage devices configured.
CONFIG_SWIFT_STORAGE_REPLICAS	**--os-swift-storage-replicas**	***1***	Number of swift storage replicas, this number **must** be no

			bigger than the number of storage zones configured.
CONFIG_SWIFT_STORAGE_FSTYPE	**--os-swift-storage-fstype**	***ext4***	FileSystem type for storage nodes. Supported values are **ext4** and **xfs** at this time.
CONFIG_REPO	**--additional-repo**		A comma separated list of URLs to any additional yum repositories to install.
CONFIG_RH_USER	**--rh-username**		To subscribe each server with Red Hat Subscription Manager, include this with **CONFIG_RH_PW**
CONFIG_RH_PW	**--rh-password**		To subscribe each server with Red Hat Subscription Manager, include this with **CONFIG_RH_USER**
CONFIG_RH_BETA_REPO	**--rh-beta-repo**	***n***	To subscribe each server to the Red Hat Enterprise Linux Beta repository set this configuration key to **y**. This is only required for preview releases of Red Hat OpenStack.
CONFIG_SATELLITE_URL	**--rhn-satellite-server**		To subscribe each server to receive updates from a Satellite server provide the URL of the Satellite server. You must also provide a user name (**CONFIG_SATELLITE_USERNAME**) and password (**CONFIG_SATELLITE_PASSWORD**) or an access key (**CONFIG_SATELLITE_AKEY**) for authentication.
CONFIG_SATELLITE_USERNAME	**--rhn-satellite-username**		Satellite servers require a user name for authentication. If using Satellite to distribute packages to your

		systems then you must set this configuration key to your Satellite username or provide an access key for authentication. If you intend to use an access key for Satellite authentication then leave this configuration key blank.
CONFIG_SATELLITE_PASSWORD	**--rhn-satellite-password**	Satellite servers require a password for authentication. If using Satellite to distribute packages to your systems then you must set this configuration key to your Satellite password or provide an access key for authentication. If you intend to use an access key for Satellite authentication then leave this configuration key blank.
CONFIG_SATELLITE_AKEY	**--rhn-satellite-activation-key**	Satellite servers are able to accept an access key for authentication. Set this configuration key to your Satellite access key if you have one. If you intend to use a user name and password for Satellite authentication then leave this configuration key blank.
CONFIG_SATELLITE_CACERT	**--rhn-satellite-cacert**	Specify the path to the certificate of the certificate authority that is used to verify that the connection with the Satellite server is secure. Leave this configuration key blank if you are not using Satellite in your

			deployment.
CONFIG_SATELLITE_PROFILE	**--rhn-satellite-profile**		Specify the profile name that must be used to identify the system in Red Hat Network, if you require one.
CONFIG_SATELLITE_FLAGS	**--rhn-satellite-flags**		Specify any additional Satellite flags that you need to be passed to the **rhnreg_ks** command. This configuration key accepts a comma separated list of flags. Valid flags are **novirtinfo**, **norhnsd**, and **nopackages**. Refer to the Red Hat Satellite documentation for more information.
CONFIG_SATELLITE_PROXY	**--rhn-satellite-proxy-host**		Specify the HTTP proxy that must be used when connecting to the Satellite server, if required.
CONFIG_SATELLITE_PROXY_USER	**--rhn-satellite-proxy-username**		Specify the user name for authenticating with the HTTP proxy that must be used when connecting to the Satellite server, if required.
CONFIG_SATELLITE_PROXY_PW	**--rhn-satellite-proxy-password**		Specify the password for authenticating with the HTTP proxy server that must be used when connecting to the Satellite server, if required.
CONFIG_NAGIOS_HOST	**--nagios-host**		The IP address of the server on which to install Nagios.
CONFIG_NAGIOS_PW	**--nagios-passwd**		The password of the **nagiosadmin** user on the Nagios server. This value will be randomly generated if it is not provided.

Important

The amount of space selected for **CINDER_VOLUMES_SIZE** must be available on the device used for **/var/lib/cinder**.

Important

Remember that the size of the Cinder volume group will restrict the amount of disk space that you can expose to compute instances.

Important

The **packstack** utility registers systems to Red Hat Network using Subscription Manager. You may encounter problems if your systems have already been registered and subscribed to the Red Hat OpenStack channels using RHN Classic.

6.3.3. Running PackStack with an Answer File

Once an answer file has been created and customized it can be used to run the **packstack** non-interactively.

Procedure 6.4. Running PackStack with an Answer File

1. Run the **packstack** with the ***--answer-file=FILE*** parameter to specify an answer file. Replace ***FILE*** with the path to the answer file.

```
$ sudo packstack --answer-file=FILE
```

> **Example 6.5. Running PackStack with an Answer File**
>
> In this example **packstack** is run using an answer file stored in **~/answers.cfg**.
>
> ```
> $ sudo packstack --answer-file=~/answers.cfg
> ```

2. PackStack will attempt to deploy OpenStack using Puppet manifests. This process may take a significant amount of time depending on the deployment options selected. When the deployment is complete PackStack will display this message:

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

   Additional information about your environment including the location of the **keystonerc** containing your OpenStack administrator authentication token and the URL of the dashboard, if configured, will also be displayed.

You have successfully deployed OpenStack using a PackStack answer file.

Important

Currently **packstack** does not support deployment of OpenStack networking. If you have deployed OpenStack using **packstack** and wish to use OpenStack Networking, then refer to Chapter 12, *Deploying OpenStack Networking Services* for more information.
It is important that such a conversion is performed **before** you begin creating and configuring compute instances with the Nova networking deployment created by **packstack**.

Note

A log file containing the details of each PackStack run is stored in a uniquely named folder in the **/var/tmp/packstack/** directory.

Part IV. Deploying OpenStack Manually

Chapter 7. Deploying Identity Services (Keystone)

This chapter covers the installation and configuration of Keystone, the Identity service.

7.1. Installation and Initial Configuration

Start by running the command that installs the **openstack-keystone** package:

```
$ sudo yum install -y openstack-keystone
```

Keystone uses a MySQL database. Use the **openstack-db** utility to create and initialize the tables for Keystone. If MySQL has not yet been installed on this server, the script will manage that as well. Ensure that you replace ***PASSWORD*** with a secure password to be sued by the service when connecting to the database:

```
$ sudo openstack-db --init --service keystone \
      --password PASSWORD
```

In order to administer Keystone, bootstrap the Keystone client with the **SERVICE_TOKEN** and **SERVICE_ENDPOINT** environment variables. Save the value of **SERVICE_TOKEN** in a file for later use:

```
$ export SERVICE_TOKEN=$(openssl rand -hex 10)
$ export SERVICE_ENDPOINT=http://192.0.43.10:35357/v2.0
$ echo $SERVICE_TOKEN > ~/ks_admin_token
```

The **SERVICE_TOKEN** must match the value of the **admin_token** option in the Keystone configuration file, **/etc/keystone/keystone.conf**. Set the **admin_token** option using this command:

```
$ sudo openstack-config --set /etc/keystone/keystone.conf \
  DEFAULT admin_token $SERVICE_TOKEN
```

Now start the Keystone service:

```
$ sudo service openstack-keystone start
$ sudo chkconfig openstack-keystone on
```

Verify that the Keystone service is running and that no errors are present in the Keystone log file:

```
$ ps -ef | grep -i keystone-all
keystone  8254     1  6 14:26 ?        00:00:00 /usr/bin/python /usr/bin/keystone-
all --config-file /etc/keystone/keystone.conf
osuser    8263  7795  0 14:26 pts/0    00:00:00 grep -i keystone-all
$ grep ERROR /var/log/keystone/keystone.log
```

Add Keystone as an API endpoint in the registry of endpoints in Keystone. Horizon (the web dashboard) requires this. Note that the **id** returned from the **service-create** command is then used as a part of the **endpoint-create** command:

```
$ keystone service-create --name=keystone --type=identity \
  --description="Keystone Identity Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |     Keystone Identity Service    |
| id          | a8bff1db381f4751bd8ac126464511ae |
| name        |             keystone             |
| type        |             identity             |
+-------------+----------------------------------+
$ keystone endpoint-create \
  --service_id a8bff1db381f4751bd8ac126464511ae \
  --publicurl 'http://192.0.43.10:5000/v2.0' \
  --adminurl 'http://192.0.43.10:35357/v2.0' \
  --internalurl 'http://192.0.43.10:5000/v2.0'
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| adminurl    |   http://192.0.43.10:35357/v2.0    |
| id          | 1295011fdc874a838f702518e95a0e13 |
| internalurl |   http://192.0.43.10:5000/v2.0     |
| publicurl   |   http://192.0.43.10:5000/v2.0     |
| region      |             regionOne            |
| service_id  | a8bff1db381f4751bd8ac126464511ae |
+-------------+----------------------------------+
```

Important

Ensure that the **publicurl**, **adminurl**, and **internalurl** parameters include the correct IP address for your Keystone identity server.

Allow incoming connections to Keystone by adding this firewall rule to the **/etc/sysconfig/iptables** configuration file:

```
-A INPUT -p tcp -m multiport --dports 5000,35357 -j ACCEPT
```

Important

This rule allows communication from all remote hosts to the system running the Keystone services on ports **5000** and **35357**. For information regarding the creation of more restrictive firewall rules refer to the Red Hat Enterprise Linux 6 *Security Guide*.

Use the **service** command to restart the **iptables** service for the new rule to take effect.

```
$ sudo service iptables restart
```

The following diagram gives an overview of what you have installed, configured, and running so far:

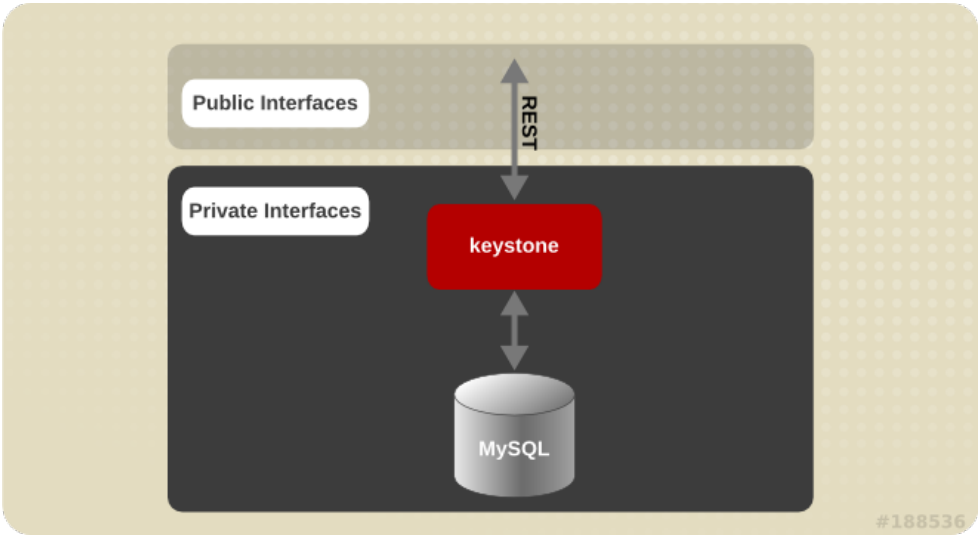


Figure 7.1. Keystone installed and configured

7.2. Creating Users

All OpenStack services will utilize Keystone for authentication. Start by creating an **admin** user, a tenant (a group of users), and role (an ID for a set of permissions).

```
$ keystone user-create --name admin --pass PASSWORD
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
| email    |                                  |
| enabled  |              True                |
| id       | 94d659c3c9534095aba5f8475c87091a |
| name     |              admin               |
| password |               ...                |
| tenantId |                                  |
+----------+----------------------------------+
$ keystone role-create --name admin
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
| id       | 78035c5d3cd94e62812d6d37551ecd6a |
| name     |              admin               |
+----------+----------------------------------+
$ keystone tenant-create --name admin
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |                                  |
| enabled     |              True                |
| id          | 6f8e3e36c4194b86b9a9b55d4b722af3 |
| name        |              admin               |
+-------------+----------------------------------+
```

Add the **admin** user to the **admin** tenant with a role of **admin**. (Note that the IDs used in this command come from the output of the previous three commands above.):

```
$ keystone user-role-add --user-id 94d659c3c9534095aba5f8475c87091a \
  --role-id 78035c5d3cd94e62812d6d37551ecd6a \
  --tenant-id 6f8e3e36c4194b86b9a9b55d4b722af3
```

The **admin** account is used to administer Keystone. To make it easy to set the **admin** user's credentials in the proper environment variables, create a **keystonerc_admin** file in your home directory with the following contents:

```
export OS_USERNAME=admin
export OS_TENANT_NAME=admin
export OS_PASSWORD=PASSWORD
export OS_AUTH_URL=http://192.0.43.10:35357/v2.0/
export PS1='[\u@\h \W(keystone_admin)]\$ '
```

Test the **keystonerc_admin** file you created by running the command to list users. Only an administrator can perform this action:

```
$ unset SERVICE_TOKEN
$ unset SERVICE_ENDPOINT
$ source ~/keystonerc_admin
$ keystone user-list
+----------------------------------+-------+---------+-------+
|                id                |  name | enabled | email |
+----------------------------------+-------+---------+-------+
| 94d659c3c9534095aba5f8475c87091a | admin |  True   |       |
+----------------------------------+-------+---------+-------+
```

So far you have been using the admin user. Now it is time to create a regular user, tenant, and role. In this example, it will have a username of **username**. Feel free to make it something else if you prefer.

```
$ keystone user-create --name username --pass PASSWORD
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
| email    |                                  |
| enabled  |               True               |
| id       | 1d59c0bfef9b4ea9ab63e2a058e68ae0 |
| name     |             username             |
| password |                ...               |
| tenantId |                                  |
+----------+----------------------------------+
$ keystone role-create --name user
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
| id       | 8261ac4eabcc4da4b01610dbad6c038a |
| name     |               user               |
+----------+----------------------------------+
$ keystone tenant-create --name rhos
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |                                  |
| enabled     |               True               |
| id          | 05816b0106994f95a83b913d4ff995eb |
| name        |               rhos               |
+-------------+----------------------------------+
```

Add the new user to the **rhos** tenant with a role of **user**. The IDs used in this command come from the output of the previous three commands:

```
$ keystone user-role-add --user-id 1d59c0bfef9b4ea9ab63e2a058e68ae0 \
  --role-id 8261ac4eabcc4da4b01610dbad6c038a \
  --tenant-id 05816b0106994f95a83b913d4ff995eb
```

To make it easy to use the **admin** user's credentials, you created the **~/keystonerc_admin** file. Now do the same thing for the new **username** user, create the file **keystonerc_username** in your home directory with the following contents:

```
export OS_USERNAME=username
export OS_TENANT_NAME=rhos
export OS_PASSWORD=PASSWORD
export OS_AUTH_URL=http://192.0.43.10:5000/v2.0/
export PS1='[\u@\h \W(keystone_username)]\$ '
```

Do a test using the new user. Source the **keystonerc_username** file and try some commands. The **user-list** command should fail since only an administrator can do that. However, retrieving a token should succeed.

```
$ source ~/keystonerc_username
$ keystone user-list
You are not authorized to perform the requested action: admin_required (HTTP 403)
$ keystone token-get
+-----------+----------------------------------+
|  Property |              Value               |
+-----------+----------------------------------+
| expires   |        2012-05-19T13:29:37Z      |
| id        | 0d709cb5840d4e53ba49fc0415b6a379 |
| tenant_id | 05816b0106994f95a83b913d4ff995eb |
| user_id   | 1d59c0bfef9b4ea9ab63e2a058e68ae0 |
+-----------+----------------------------------+
```

Chapter 8. Deploying Object Storage Services (Swift)

The next component to install and set up is Swift. The steps below explain how to create:

- A Swift proxy server.
- Six Swift storage devices (each storing accounts, containers and objects).
- Three zones (z1, z2, z3 with 2 storage servers each, that is z1d1, z1d2; z2d1, z2d2 etc.).

The replication level will be set to 3. Using this example setup, each object uploaded to Swift will be replicated (3 copies in total), with each zone storing one copy of the object.

The first step is to install the Swift packages:

```
$ sudo yum install -y openstack-swift openstack-swift-proxy \
openstack-swift-account openstack-swift-container openstack-swift-object \
openstack-utils memcached python-keystone
```

8.1. Creating the Swift Ring Files

Three ring files need to be created containing details of all the storage devices. These are used to deduce where a particular piece of data is stored.

In this example they are created with a "part power" of 12, as a result, each ring will contain 4096 partitions. Choose an appropriate value for this number if creating a ring file for a production deployment.

(more information can be found at: http://docs.openstack.org/developer/swift/deployment_guide.html , the *openstack.org 'Swift deployment guide'* in the section 'Preparing The Ring').

Important

This example assumes that you have set the environment variables containing the authentication information of the OpenStack administrator by loading them from the **keystonerc_admin** file.

```
$ source ~/keystonerc_admin
```

```
$ sudo swift-ring-builder /etc/swift/account.builder create 12 3 1
$ sudo swift-ring-builder /etc/swift/container.builder create 12 3 1
$ sudo swift-ring-builder /etc/swift/object.builder create 12 3 1
```

Once the ring file is created you need to add the storage devices to each ring, starting with the accounts. Use the IP address of the Swift object storage server in the account identifiers:

```
$ sudo swift-ring-builder /etc/swift/account.builder add
z1-192.0.43.10:6002/z1d1 100
$ sudo swift-ring-builder /etc/swift/account.builder add
z1-192.0.43.10:6002/z1d2 100
$ sudo swift-ring-builder /etc/swift/account.builder add
z2-192.0.43.10:6002/z2d1 100
$ sudo swift-ring-builder /etc/swift/account.builder add
z2-192.0.43.10:6002/z2d2 100
$ sudo swift-ring-builder /etc/swift/account.builder add
z3-192.0.43.10:6002/z3d1 100
$ sudo swift-ring-builder /etc/swift/account.builder add
z3-192.0.43.10:6002/z3d2 100
```

then the containers:

```
$ sudo swift-ring-builder /etc/swift/container.builder add
z1-192.0.43.10:6001/z1d1 100
$ sudo swift-ring-builder /etc/swift/container.builder add
z1-192.0.43.10:6001/z1d2 100
$ sudo swift-ring-builder /etc/swift/container.builder add
z2-192.0.43.10:6001/z2d1 100
$ sudo swift-ring-builder /etc/swift/container.builder add
z2-192.0.43.10:6001/z2d2 100
$ sudo swift-ring-builder /etc/swift/container.builder add
z3-192.0.43.10:6001/z3d1 100
$ sudo swift-ring-builder /etc/swift/container.builder add
z3-192.0.43.10:6001/z3d2 100
```

and then the objects:

```
$ sudo swift-ring-builder /etc/swift/object.builder add
z1-192.0.43.10:6000/z1d1 100
$ sudo swift-ring-builder /etc/swift/object.builder add
z1-192.0.43.10:6000/z1d2 100
$ sudo swift-ring-builder /etc/swift/object.builder add
z2-192.0.43.10:6000/z2d1 100
$ sudo swift-ring-builder /etc/swift/object.builder add
z2-192.0.43.10:6000/z2d2 100
$ sudo swift-ring-builder /etc/swift/object.builder add
z3-192.0.43.10:6000/z3d1 100
$ sudo swift-ring-builder /etc/swift/object.builder add
z3-192.0.43.10:6000/z3d2 100
```

To distribute the partitions across the drives in the ring, enter the following commands:

```
$ sudo swift-ring-builder /etc/swift/account.builder rebalance
$ sudo swift-ring-builder /etc/swift/container.builder rebalance
$ sudo swift-ring-builder /etc/swift/object.builder rebalance
```

Check to see that you now have 3 ring files in the directory **/etc/swift**. The command:

```
$ ls /etc/swift/*gz
```

should reveal:

```
  /etc/swift/account.ring.gz  /etc/swift/container.ring.gz
/etc/swift/object.ring.gz
```

Ensure that all files in the **/etc/swift/** directory including those that you have just created are owned by the **root** user and **swift** group.

```
$ sudo chown -R root:swift /etc/swift
```

You also need to create a private hash key for the Swift hashing algorithm:

```
$ sudo openstack-config --set /etc/swift/swift.conf swift-hash \
swift_hash_path_suffix $(openssl rand -hex 10)
```

8.2. Configuring Keystone

In this example we are assuming Keystone has already been created with an admin user. We need to create a tenant named 'services' (if it does not already exist), a user called **swift**and then give the swift user the admin role in the services tenant.

Set up the shell to access Keystone as the admin user:

```
$ source ~/keystonerc_admin
```

Check if there is a services tenant:

```
$ keystone tenant-list
```

If not, create one:

```
$ keystone tenant-create --name services
```

Create a Swift user and set its password by replacing ***PASSWORD*** with your chosen password:

```
$ keystone user-create --name swift --pass PASSWORD
```

Get the uuid of the admin role:

```
$ keystone role-list | grep admin
```

If no admin role exists, create one:

```
$ keystone role-create --name admin
```

Add the swift user to the service tenant with the admin role:

```
$ keystone user-role-add --role-id <roleid> --tenant-id <tenantid> --user-id
<userid>
```

Create swift as an endpoint in Keystone:

```
$ keystone service-list
```

If the object-store service does not already exist, create one:

```
$ keystone service-create --name swift --type object-store \
   --description "Swift Storage Service"
$ keystone endpoint-create --service_id <serviceid> \
   --publicurl "http://192.0.43.10:8080/v1/AUTH_\$(tenant_id)s" \
   --adminurl "http://192.0.43.10:8080/v1/AUTH_\$(tenant_id)s" \
   --internalurl "http://192.0.43.10:8080/v1/AUTH_\$(tenant_id)s"
```

Important

Ensure that the **publicurl**, **adminurl**, and **internalurl** parameters include the correct IP address for your Swift object storage server.

8.3. Configuring the Swift Proxy

Update the configuration file for the Swift proxy server with the correct authentication details for the appropriate Keystone user.

```
$ openstack-config --set /etc/swift/proxy-server.conf \
     filter:authtoken admin_tenant_name services
$ openstack-config --set /etc/swift/proxy-server.conf \
     filter:authtoken admin_user swift
$ openstack-config --set /etc/swift/proxy-server.conf \
     filter:authtoken admin_password PASSWORD
```

Start the **memcached** and **openstack-swift-proxy** services using the **service** command.

```
$ sudo service memcached start
$ sudo service openstack-swift-proxy start
```

Enable the **memcached** and **openstack-swift-proxy** services permenantly using the **chkconfig** command.

```
$ sudo chkconfig memcached on
$ sudo chkconfig openstack-swift-proxy on
```

Allow incoming connections to the Swift proxy server by adding this firewall rule to the **/etc/sysconfig/iptables** configuration file:

```
-A INPUT -p tcp -m multiport --dports 8080 -j ACCEPT
```

Important

This rule allows communication from all remote hosts to the system hosting the Swift proxy on port **8080**. For information regarding the creation of more restrictive firewall rules refer to the Red Hat Enterprise Linux 6 *Security Guide*.

Use the **service** command to restart the **iptables** service for the new rule to take effect.

```
$ sudo service iptables restart
```

8.4. Configuring Swift Storage Nodes

Object storage must be backed by on disk storage volumes. This is where objects will actually be stored. To provide adequate backing storage for the installation discussed so far in this chapter you will require six (6) devices.

If you have enough physical devices available to support this setup then ensure they are mounted and proceed. Note that the instructions in this section assume the devices are mounted to the **/srv/node/** directory.

If you do not have enough physical devices available to support this setup but still wish to test the object storage facilities provided by OpenStack then it is possible to create test devices using the loopback facility. This is not recommended for production environments.

Example 8.1. Creating Object Storage Devices using Loopback

This example script creates three (3) zones and two (2) devices backed by loopback devices for each zone for a total of six (6) devices. The volume images are created in the **/data/** directory and mounted under the **/srv/node/** directory using the loopback facility.

Each device is 50 GB in size. You may wish to lower this number.

This script must be run with **root** privileges.

```
#!/bin/sh

# Device size in GB
SIZE=50
DATA=/data
MOUNT=/srv/node

mkdir ${DATA}

for ZONE in 1 2 3; do
    for DEVICE in 1 2; do
        truncate ${DATA}/swift-z${ZONE}d${DEVICE} --size ${SIZE}G
        LOOPDEVICE=`losetup --show -f ${DATA}/swift-z${ZONE}d${DEVICE}`
        mkfs.ext4 -I 1024 ${LOOPDEVICE}
        mkdir -p ${MOUNT}/z${ZONE}d${DEVICE}
        mount -o noatime,nodiratime,nobarrier,user_xattr ${LOOPDEVICE} \
              ${MOUNT}/z${ZONE}d${DEVICE}
    done
done
```

Warning

This example setup for object storage is only intended as a quick and easy way to try out persistent volumes in a testing environment. This method must not be used in a production environment.
Object storage created in this manner will not persist across system reboots automatically.
To restore the object storage from the files created in this procedure following a reboot you must:

- Use the **losetup** command to recreate the loopback device for each file in the **/data/** directory.
- Use the **mount** command to remount the loopback device to the appropriate **/srv/node/** directory.
- Use the **service** command to restart the **openstack-swift-account**, **openstack-swift-container**, and **openstack-swift-object** services.

The recommended procedure for a production environment is to instead use physical storage and persist the storage mounts in the **/etc/fstab** file. You must also use the **chkconfig** to ensure that the object storage services start automatically on boot.

Important

By default a Red Hat Enterprise Linux system has eight (8) loopback devices available for use. If you are already using some of these devices then you may encounter this message when running the script:

```
mount: could not find any free loop device
```

If this occurs use the **losetup -a** command to list the status of all loopback devices. You can then either release some of the devices using the **umount** and **losetup -d** commands or alter the script to require less devices.
Alternatively the command **MAKEDEV -v /dev/loop** will create a large number of additional loopback devices. Note however that the number of loopback devices available will revert to the default on the next boot.

```
$ sudo chown -R swift:swift /srv/node/
```

Use the **restorecon** command to ensure that all files in the **/srv** directory have the correct SELinux security contexts defined.

```
$ sudo restorecon -R /srv
```

Failure to set the security contexts correctly will result in AVC denials being recorded when Swift attempts to access the data stored in the **/srv/** directory.

Start the account, container and object storage services:

```
$ sudo service openstack-swift-account start
$ sudo service openstack-swift-container start
$ sudo service openstack-swift-object start
```

If deploying a setup backed by physical storage configure the services to start automatically in future:

```
$ sudo chkconfig openstack-swift-account on
$ sudo chkconfig openstack-swift-container on
$ sudo chkconfig openstack-swift-object on
```

Allow incoming connections to Swift by adding this firewall rule to the **/etc/sysconfig/iptables** configuration file:

```
-A INPUT -p tcp -m multiport --dports 6000,6001,6002,873 -j ACCEPT
```

Important

This rule allows communication from all remote hosts to the system running Swift on ports **6000** to **6002** and **873**. For information regarding the creation of more restrictive firewall rules refer to the Red Hat Enterprise Linux 6 *Security Guide*.

Use the **service** command to restart the **iptables** service for the new rule to take effect.

```
$ sudo service iptables restart
```

8.5. Testing Swift

Set up the shell to access Keystone as a user that has either the admin or SwiftOperator role. The admin user is shown in this example:

```
$ source ~/keystonerc_admin
$ swift list
$ head -c 1024 /dev/urandom > data.file ; swift upload c1 data.file
$ head -c 1024 /dev/urandom > data2.file ; swift upload c1 data2.file
$ head -c 1024 /dev/urandom > data3.file ; swift upload c2 data3.file
$ swift list
$ swift list c1
$ swift list c2
```

You have now uploaded 3 files into 2 containers. If you look in the various storage devices you should see 9 **.data** files (each file has 3 copies):

```
$ find /srv/node/ -type f -name "*data"
```

Chapter 9. Deploying Image Services (Glance)

Now that Keystone has been installed and configured, the next component to set up is Glance. The first step is to install the **openstack-glance** package:

```
$ sudo yum install -y openstack-glance
```

Ensure that you have set the environment variables containing the authentication information of the OpenStack administrator by loading them from the **keystonerc_admin** file.

```
$ source ~/keystonerc_admin
```

Glance makes use of MySQL to store metadata about images. Use **openstack-db** to initialize the database for use with Glance. Ensure that you replace ***PASSWORD*** with a secure password to be used by the service when connecting to the database:

```
$ sudo openstack-db --init --service glance \
      --password PASSWORD
```

Keystone is used to manage authentication. Run the following commands to update the Glance configuration files for Keystone use:

```
$ sudo openstack-config --set /etc/glance/glance-api.conf \
  paste_deploy flavor keystone
$ sudo openstack-config --set /etc/glance/glance-api.conf \
  keystone_authtoken admin_tenant_name admin
$ sudo openstack-config --set /etc/glance/glance-api.conf \
  keystone_authtoken admin_user admin
$ sudo openstack-config --set /etc/glance/glance-api.conf \
  keystone_authtoken admin_password PASSWORD
$ sudo openstack-config --set /etc/glance/glance-registry.conf \
  paste_deploy flavor keystone
$ sudo openstack-config --set /etc/glance/glance-registry.conf \
  keystone_authtoken admin_tenant_name admin
$ sudo openstack-config --set /etc/glance/glance-registry.conf \
  keystone_authtoken admin_user admin
$ sudo openstack-config --set /etc/glance/glance-registry.conf \
  keystone_authtoken admin_password PASSWORD
```

In the default configuration Glance uses the **file** storage backend. This means that uploaded disk images are stored in the **/var/lib/glance/images/** directory on the local file system of the Glance server.

To configure Glance to use the Swift storage backend instead you must set several additional configuration values in the **/etc/glance/glance-api.conf** file. If you have chosen not to deploy Swift or do not intend to use it for disk image storage then skip these steps.

- Set the **default_store** configuration key to **swift**.

  ```
  $ sudo openstack-config --set /etc/glance/glance-api.conf \
        DEFAULT default_store swift
  ```

- Set the **swift_store_auth_address** configuration key to the full authentication URL for your Swift deployment. This is the public URL of the Swift service entry.

```
$ sudo openstack-config --set /etc/glance/glance-api.conf \
       DEFAULT swift_store_auth_address http://192.0.43.10:8080/v1/
```

- Set the **swift_store_user** configuration key to contain the tenant and user to use for authentication in the format ***TENANT*:*USER***. If you followed the instructions in this guide to deploy Swift these values must be replaced with the **service** tenant and the **swift** user respectively. If you do not follow the instructions in this guide to deploy Swift then you must replace these values with the appropriate Swift tenant and user for your environment.

```
$ sudo openstack-config --set /etc/glance/glance-api.conf \
       DEFAULT swift_store_user service:swift
```

- Set the **swift_store_key** configuration key to the password of the user to be used for authentication. This is the password that you set for the **swift** user when deploying Swift.

```
$ sudo openstack-config --set /etc/glance/glance-api.conf \
       DEFAULT swift_store_key PASSWORD
```

Now that Glance has been configured, start the **glance-api** and **glance-registry** services:

```
$ sudo service openstack-glance-registry start
$ sudo service openstack-glance-api start
$ sudo chkconfig openstack-glance-registry on
$ sudo chkconfig openstack-glance-api on
```

Allow incoming connections to Glance by adding this firewall rule to the **/etc/sysconfig/iptables** configuration file:

```
-A INPUT -p tcp -m multiport --dports 9292 -j ACCEPT
```

Important

This rule allows communication from all remote hosts to the system hosting the Glance services on port **9292**. For information regarding the creation of more restrictive firewall rules refer to the Red Hat Enterprise Linux 6 *Security Guide*.

Use the **service** command to restart the **iptables** service for the new rule to take effect.

```
$ sudo service iptables restart
```

In addition to providing authentication, Keystone also maintains a registry of available OpenStack services and how to reach them. This is referred to as the service catalog. After installing Glance, add Glance as an entry in Keystone's service catalog. This must be performed as the Keystone administrator:

```
$ source ~/keystonerc_admin
$ keystone service-create --name=glance --type=image --description="Glance
Image Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |       Glance Image Service       |
| id          | 1447f490e9784aa8890f9e39ddaa8d44 |
| name        |              glance              |
| type        |              image               |
+-------------+----------------------------------+
$ keystone endpoint-create --service_id 1447f490e9784aa8890f9e39ddaa8d44 \
  --publicurl http://192.0.43.10:9292 \
  --adminurl http://192.0.43.10:9292 \
  --internalurl http://192.0.43.10:9292
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| adminurl    |      http://192.0.43.10:9292     |
| id          | 9693e1af560843f68e9e91f2b3664a19 |
| internalurl |      http://192.0.43.10:9292     |
| publicurl   |      http://192.0.43.10:9292     |
| region      |             regionOne            |
| service_id  | 1447f490e9784aa8890f9e39ddaa8d44 |
+-------------+----------------------------------+
```

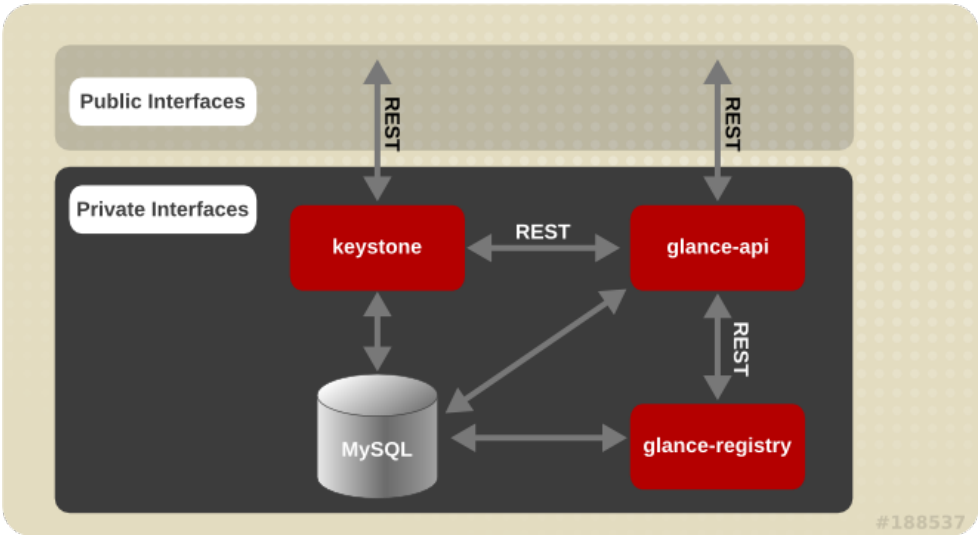
Important

Ensure that the **publicurl**, **adminurl**, and **internalurl** parameters include the correct IP address for your Glance image server.

Now it's time to test out the **glance** client application. First set up your environment to use the credentials for the regular account. Then run the **glance image-list** command. This will list the images that are currently available. If no errors occur, this command will produce no output since you have not registered an image:

```
$ source ~/keystonerc_username
$ glance image-list
```

The following diagram shows the services that have been installed and configured so far, which include Keystone and Glance.



Note that the direct communication between the **`glance-api`** service and MySQL database (shown by the diagonal arrow) only occurs if the v2 API is used. For v1, all the interaction with the image metadata store occurs via the **`glance-registry`** service.

Figure 9.1. Keystone and Glance installed and configured

9.1. Building Images using Oz

Red Hat OpenStack includes Oz, a set of libraries and utilities for performing automated operating system installations with limited input from the user. Oz is also useful for building virtual machine images that can be uploaded to Glance and used to launch virtual machine instances.

Template Description Language Files

Oz accepts input in the form of Template Description Language (TDL) files. These files contain XML that describes the operating system being installed, where the installation media is available from, and any additional packages or customization changes that must be applied to the image.

An example TDL file is provided in Procedure 9.1, “Building Images using Oz”. For more detailed information about the TDL file format refer to the Cloud Forms *Cloud Engine User Guide*.

The `virt-sysprep` Command

It is also recommended that the **`virt-sysprep`** command is run on Linux-based virtual machine images prior to uploading them to Glance. The **`virt-sysprep`** command re-initializes a disk image in preparation for use in a virtual environment. Operations it performs by default include removal of SSH keys, removal of persistent MAC addresses, and removal of user accounts.

The **`virt-sysprep`** command is provided by the *libguestfs-tools* package.

Important

Oz makes use of the **`default`** Libvirt network. It is recommended that you do not build images using Oz on a system that is running either the **`nova-network`** service or any of the OpenStack Networking components.

Procedure 9.1. Building Images using Oz

1. Use the **yum** command to install the *oz* and *libguestfs-tools* packages.

```
$ yum install -y oz libguestfs-tools
```

2. Download the Red Hat Enterprise Linux 6 Server installation DVD ISO file from https://rhn.redhat.com/rhn/software/channel/downloads/Download.do?cid=10486.
   While Oz supports the use of network-based installation media in this procedure a Red Hat Enterprise Linux 6 ISO will be used.
3. Use a text editor to create a TDL file for use with Oz. Refer to the example provided here for a basic TDL file.

   Example 9.1. TDL File

   This is a basic TDL file to be used for the creation of a Red Hat Enterprise Linux 6 disk image. In particular take note of the use of the **rootpw** element to set the password for the **root** user and the **iso** element to set the path to the DVD ISO.

```
<template>
 <name>rhel64_x86_64</name>
 <description>Red Hat 6.4 x86_64 template</description>
 <os>
  <name>RHEL-6</name>
  <version>4</version>
  <arch>x86_64</arch>
  <rootpw>PASSWORD</rootpw>
  <install type='iso'>
    <iso>file:///home/shadowman/rhel-server-6.4-x86_64-dvd.iso</iso>
  </install>
 </os>
 <commands>
   <command name='console'>
sed -i 's/ rhgb//g' /boot/grub/grub.conf
sed -i 's/ quiet//g' /boot/grub/grub.conf
sed -i 's/ console=tty0 / serial=tty0 console=ttyS0,115200n8 /g'
/boot/grub/grub.conf
   </command>
 </commands>
</template>
```

4. Run the **oz-install** to build an image. Use the **-u** argument to ensure any required customization changes to the image are applied after guest operating installation. Use the **-d3** argument to enable the display of errors, warnings, and informational messages.
   Provide the path to your TDL file as the final argument to the **oz-install** command.

```
$ oz-install -u -d3 TDL_FILE
```

   By default Oz will store the resultant image in the **/var/lib/libvirt/images/** directory. This location is configurable by editing the **/etc/oz/oz.cfg** configuration file.
5. Run the **virt-sysprep** command on the image to re-initialize it in preparation for upload to Glance. Replace ***FILE*** with the path to the disk image.

```
$ virt-sysprep --add FILE
```

Refer to the **virt-sysprep** manual page by running the **man virt-sysprep** command for information on enabling and disabling specific operations.

You have successfully created a Red Hat Enterprise Linux based image that is ready to be added to Glance.

9.2. Adding Images to Glance

To launch instances based on images stored in Glance you must first add some images. You must first have created or downloaded suitable images to use in an OpenStack environment. If you have not then refer to Section 9.1, “Building Images using Oz” and the Red Hat Enterprise Linux *Virtualization_Host_Configuration_and_Guest_Installation_Guide* for more information.

Important

It is recommended that the **virt-sysprep** command is run on Linux-based virtual machine images prior to uploading them to Glance. The **virt-sysprep** command re-initializes a disk image in preparation for use in a virtual environment. Operations it performs by default include removal of SSH keys, removal of persistent MAC addresses, and removal of user accounts. The **virt-sysprep** command is available in the *libguestfs-tools* package in Red Hat Enterprise Linux.

```
$ sudo yum install -y libguestfs-tools
$ virt-sysprep --add FILE
```

Refer to the **virt-sysprep** manual page by running the **man virt-sysprep** command for information on enabling and disabling specific operations.

Procedure 9.2. Adding an Image to Glance

1. Ensure that you have set the environment variables used for authenticating with Keystone by loading them from the **keystonerc** file associated with your user. Note that an administrative account is not required.

   ```
   $ source ~/keystonerc_user
   ```

2. Use the **glance image-create** command to import your disk image:

   ```
   $ glance image-create --name "NAME" \
         --is-public IS_PUBLIC \
         --disk-format DISK_FORMAT \
         --container-format CONTAINER_FORMAT \
         --file IMAGE
   ```

   Replace the command line arguments to **glance image-create** with the appropriate values for your environment and disk image:

   - Replace ***NAME*** with the name that users will refer to the disk image by.
   - Replace ***IS_PUBLIC*** with **true** or **false**. Setting this value to **true** means that all users will be able to view and use the image.
   - Replace ***DISK_FORMAT*** with the format of the virtual machine disk image. Valid values include **raw**, **vhd**, **vmdk**, **vdi**, **iso**, **qcow2**, **aki**, **ami**, and **ami**.

If the format of the virtual machine disk image is unknown then use the **qemu-img info** command to try and identify it.

Example 9.2. Using qemu-img info

In this example the **qemu-img info** is used to determine the format of a disk image stored in the file **./RHEL64.img**.

```
$ qemu-img info ./RHEL64.img
image: ./RHEL64.img
file format: qcow2
virtual size: 5.0G (5368709120 bytes)
disk size: 136K
cluster_size: 65536
```

- Replace ***CONTAINER_FORMAT*** with the container format of the image. The container format is **bare** unless the image is packaged in a file format such as **ovf** or **ami** that includes additional metadata related to the image.
- Replace ***IMAGE*** with the local path to the image file to upload.

Refer to the output of the **glance help image-create** command for more information about supported arguments to **glance image-create**.

Note

If the image being uploaded is not locally accessible but is available using a remote URL then provide it using the **--location** parameter instead of using the **--file** parameter. Note however that unless you also specify the **--copy-from** argument then Glance will not copy the image into the object store but instead it will be accessed remotely each time it is required.

Example 9.3. Uploading an Image to Glance

In this example the **qcow2** format image in the file named **rhel-64.qcow2** is uploaded to Glance. It is created in Glance as a publicly accessible image named **RHEL 6.4**.

```
$ glance image-create --name "RHEL 6.4" --is-public true --disk-format
qcow2 \
          --container-format bare \
          --file rhel-64.qcow2
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 2f81976cae15c16ef0010c51e3a6c163     |
| container_format | bare                                 |
| created_at       | 2013-01-25T14:45:48                  |
| deleted          | False                                |
| deleted_at       | None                                 |
| disk_format      | qcow2                                |
| id               | 0ce782c6-0d3e-41df-8fd5-39cd80b31cd9 |
| is_public        | True                                 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | RHEL 6.4                             |
| owner            | b1414433c021436f97e9e1e4c214a710     |
| protected        | False                                |
| size             | 25165824                             |
| status           | active                               |
| updated_at       | 2013-01-25T14:45:50                  |
+------------------+--------------------------------------+
```

3. Use the **glance image-list** command to verify that your image was successfully uploaded.

```
 $ glance image-list
+--------------+----------+-------------+------------------+----------+----
----+
| ID           | Name     | Disk Format | Container Format |Size      |
Status |
+--------------+----------+-------------+------------------+----------+----
----+
| 0ce782c6-... | RHEL 6.4 | qcow2       | bare             |213581824 |
active |
+--------------+----------+-------------+------------------+----------+----
----+
```

Use the **glance image-show** command to view more detailed information about an image. Use the identifier of the image to specify the image that you wish to view.

```
$ glance image-show 0ce782c6-0d3e-41df-8fd5-39cd80b31cd9
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 2f81976cae15c16ef0010c51e3a6c163     |
| container_format | bare                                 |
| created_at       | 2013-01-25T14:45:48                  |
| deleted          | False                                |
| disk_format      | qcow2                                |
| id               | 0ce782c6-0d3e-41df-8fd5-39cd80b31cd9 |
| is_public        | True                                 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | RHEL 6.4                             |
| owner            | b1414433c021436f97e9e1e4c214a710     |
| protected        | False                                |
| size             | 25165824                             |
| status           | active                               |
| updated_at       | 2013-01-25T14:45:50                  |
+------------------+--------------------------------------+
```

You have successfully uploaded a disk image to Glance. This disk image can now be used as the basis for launching virtual machine instances in your OpenStack environment.

Chapter 10. Deploying Volume Services (Cinder)

The next component to configure is Cinder, which is responsible for volume storage of virtual machines' data. The first step is to install the **openstack-cinder** package:

```
$ sudo yum install -y openstack-cinder
```

Ensure that you have set the environment variables containing the authentication information of the OpenStack administrator by loading them from the **keystonerc_admin** file.

```
$ source ~/keystonerc_admin
```

Cinder makes use of MySQL just like the previous OpenStack services. Use the **openstack-db** utility to initialize the database for Cinder. Ensure that you replace ***PASSWORD*** with a secure password to be used by the service when connecting to the database:

```
$ sudo openstack-db --init --service cinder \
      --password PASSWORD
```

You must explicitly configure Cinder to make use of Keystone for authentication. To do so, run the following commands to update the Cinder configuration files:

```
$ sudo openstack-config --set /etc/cinder/cinder.conf DEFAULT auth_strategy
keystone
$ sudo openstack-config --set /etc/cinder/cinder.conf \
   keystone_authtoken admin_tenant_name admin
$ sudo openstack-config --set /etc/cinder/cinder.conf \
   keystone_authtoken admin_user admin
$ sudo openstack-config --set /etc/cinder/cinder.conf \
   keystone_authtoken admin_password PASSWORD
```

OpenStack deployments are made up of multiple services. As an OpenStack deployment is scaled, these services will be running on multiple machines. The OpenStack services utilize AMQP (Advanced Message Queuing Protocol) to communicate amongst themselves. You will use Qpid to provide AMQP services for OpenStack. Run the following commands to install, configure, and run Qpid:

Warning

The following commands disable Qpid authentication. This is acceptable for getting started with a simple test deployment. When you are doing a production deployment, however, you should leave authentication enabled. Consult Qpid documentation to see methods for adding users. Update **/etc/nova/nova.conf** with the **qpid_username** and **qpid_password** options so that Nova is able to authenticate with Qpid. For more detailed documentation about Nova's configuration related to Qpid, see http://docs.openstack.org/folsom/openstack-compute/admin/content/configuration-qpid.html.

```
$ sudo yum install -y qpid-cpp-server
$ sudo sed -i -e 's/auth=.*/auth=no/g' /etc/qpidd.conf
$ sudo service qpidd start
$ sudo chkconfig qpidd on
```

Cinder is the OpenStack service which is responsible for handling persistent storage for virtual

machines. There are multiple backends for **cinder**. The one that is enabled by default utilizes an LVM volume group called **cinder-volumes** to carve out storage for virtual machines on demand. For testing purposes, you may create a simple file-backed volume group.

Warning

This example setup for the **cinder-volumes** group is only intended as a quick and easy way to try out persistent volumes in a testing environment. This method must not be used in a production environment.
Volumes created in this manner will not persist across system reboots. To restore the **cinder-volumes** volume group from the **~/cinder-volumes** file created in this procedure following a reboot, issue these commands:

```
$ sudo losetup -fv ~/cinder-volumes
```

```
$ sudo service openstack-cinder-volume restart
```

```
$ sudo grep -q /etc/cinder/volumes /etc/tgt/targets.conf || sudo sed -i
'1iinclude /etc/cinder/volumes/*' /etc/tgt/targets.conf
$ sudo service tgtd start
$ sudo chkconfig tgtd on
$ sudo truncate --size 20G ~/cinder-volumes
$ sudo losetup -fv ~/cinder-volumes
$ sudo vgcreate cinder-volumes /dev/loop0
$ sudo vgdisplay cinder-volumes
--- Volume group ---
  VG Name               cinder-volumes
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               20.00 GiB
  PE Size               4.00 MiB
  Total PE              5119
  Alloc PE / Size       0 / 0
  Free  PE / Size       5119 / 20.00 GiB
  VG UUID               9ivjbZ-wmwI-xXQ4-YSon-mnrC-MMwZ-dU53L5
```

With the prerequisite steps completed, run the following commands to start up the Cinder services:

```
$ for srv in api scheduler volume ; do \
   sudo service openstack-cinder-$srv start ; \
  done
$ for srv in api scheduler volume ; do \
   sudo chkconfig openstack-cinder-$srv on ; \
  done
```

Make sure that the Cinder log files do not contain any errors:

```
$ grep -i ERROR /var/log/cinder/*
$ grep CRITICAL /var/log/cinder/*
```

Now that the Cinder services are running, you may find it beneficial to monitor Cinder while completing system setup. To do so you can open another terminal and tail the files in **/var/log/cinder**

```
$ tail -f /var/log/cinder/*.log
```

Register the Cinder API as an endpoint with Keystone. Note that the **service_id** passed to the **endpoint-create** command comes from the output of the **service-create** command.

```
$ source ~/keystonerc_admin
$ keystone service-create --name=cinder --type=volume --description="Cinder
Volume Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |      Cinder Volume Service       |
| id          | 94b6247ac66643a88af2185ad7738edf |
| name        |              cinder              |
| type        |              volume              |
+-------------+----------------------------------+
$ keystone endpoint-create --service_id 94b6247ac66643a88af2185ad7738edf \
  --publicurl "http://192.0.43.10:8776/v1/\$(tenant_id)s" \
  --adminurl "http://192.0.43.10:8776/v1/\$(tenant_id)s" \
  --internalurl "http://192.0.43.10:8776/v1/\$(tenant_id)s"
+-------------+-----------------------------------------+
|   Property  |                  Value                  |
+-------------+-----------------------------------------+
| adminurl    | http://192.0.43.10:8776/v1/$(tenant_id)s |
| id          |     247a56ec4fa94afca231aa0c304c7049     |
| internalurl | http://192.0.43.10:8776/v1/$(tenant_id)s |
| publicurl   | http://192.0.43.10:8776/v1/$(tenant_id)s |
| region      |                regionOne                |
| service_id  |     94b6247ac66643a88af2185ad7738edf     |
+-------------+-----------------------------------------+
```

Important

Ensure that the **publicurl**, **adminurl**, and **internalurl** parameters include the correct IP address for your Cinder volume server.

Allow incoming connections to Cinder by adding this firewall rule to the **/etc/sysconfig/iptables** configuration file:

```
-A INPUT -p tcp -m multiport --dports 3260,8776 -j ACCEPT
```

Important

This rule allows communication from all remote hosts to the system hosting the Cinder services on ports **3260** and **8776**. For information regarding the creation of more restrictive firewall rules refer to the Red Hat Enterprise Linux 6 *Security Guide*.

Use the **service** command to restart the **iptables** service for the new rule to take effect.

```
$ sudo service iptables restart
```

Set up your environment to use the credentials for your regular user. Then test out the **cinder** client application. The **list** command shows configured volumes. Since you have not created any volumes no output will be returned:

```
$ source ~/keystonerc_username
$ cinder list
```

Chapter 11. Deploying Compute Services (Nova)

So far you have installed Keystone, Glance, and Cinder. Now you will install Nova, which is responsible for the life cycle of virtual machines. The first step is to install the *openstack-nova* package. As we will be using the Cinder volume service for storage, we also need to ensure that the *python-cinderclient* package is explicitly included when installing Nova.

```
$ sudo yum install -y openstack-nova python-cinderclient
```

Storage volumes are provided by the **cinder-volumes** service. Install the *python-cinderclient* package on all of the compute nodes in your environment.

```
$ sudo yum install -y python-cinderclient
```

Ensure that you have set the environment variables containing the authentication information of the OpenStack administrator by loading them from the **keystonerc_admin** file.

```
$ source ~/keystonerc_admin
```

Nova makes use of MySQL just like Keystone and Glance. Use the **openstack-db** utility to initialize the database for Nova. Ensure that you replace ***PASSWORD*** with a secure password to be used by the service when connecting to the database:

```
$ sudo openstack-db --init --service nova \
      --password PASSWORD
```

You must explicitly configure Nova to make use of Keystone for authentication. To do so, run the following commands to update the Nova configuration files:

```
$ sudo openstack-config --set /etc/nova/nova.conf DEFAULT auth_strategy
keystone
$ sudo openstack-config --set /etc/nova/api-paste.ini \
  filter:authtoken admin_token $(cat ~/ks_admin_token)
```

You must also configure Nova to use the correct network interfaces. The interfaces to use are specified by the values of keys in the **/etc/nova/nova.conf** configuration file. The relevant configuration keys are:

flat_interface

Specifies the interface to use as the private interface for the node providing DHCP services to running instances.

public_interface

Specifies the interface to use as the public interface for the node as seen by clients and other nodes in the environment.

Use the **openstack-config** command to update the keys with the interfaces to use, replacing ***INTERFACE*** with a valid network interface.

```
$ sudo openstack-config --set /etc/nova/nova.conf DEFAULT flat_interface
INTERFACE
$ sudo openstack-config --set /etc/nova/nova.conf DEFAULT public_interface
INTERFACE
```

Important

In Red Hat OpenStack the Nova Networking service is configured to use the **FlatDHCPManager** network manager by default. Upstream OpenStack releases default to using the **VlanManager** network manager.

- When using the **FlatDHCPManager** network manager each compute node has a network bridge. An instance of the **dnsmasq** service listens on the bridge interface and handles DHCP requests from virtual machine instances running on the node. The bridge acts as the default network gateway for all instances running on the compute node.
- When using the **VlanManager** network manager virtual machine instances are divided into, and assigned address from, different subnets based on the tenants that they belong to using VLAN tagging. This provides additional isolation in multi-tenant environments but also requires more initial configuration. In particular each node must be configured for IP forwarding, all network switches must support VLAN tagging, and all network switches must be configured to enable the VLAN tags that you intend to use with your Compute configuration.

The network manager in use is determined by the value of the **network_manager** configuration key in the **/etc/nova/nova.conf** configuration file.

Note

Use the **ip link** command to identify the network interfaces attached to the system:

```
# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UP qlen 1000
   link/ether 00:1a:4a:0f:18:d2 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UP qlen 1000
   link/ether 00:1a:4a:0f:18:d5 brd ff:ff:ff:ff:ff:ff
```

Nova is made up of multiple services. As an OpenStack deployment is scaled, these services will be running on multiple machines. The Nova services utilize AMQP (Advanced Message Queuing Protocol) to communicate amongst themselves. You will use Qpid to provide AMQP for OpenStack. Run the following commands to install, configure, and run Qpid:

Warning

The following commands disable Qpid authentication. This is acceptable for getting started with a simple test deployment. When you are doing a production deployment, however, you should leave authentication enabled. Consult Qpid documentation to see methods for adding users. Update **/etc/nova/nova.conf** with the **qpid_username** and **qpid_password** options so that Nova is able to authenticate with Qpid. For more detailed documentation about Nova's configuration related to Qpid, see http://docs.openstack.org/folsom/openstack-compute/admin/content/configuration-qpid.html.

```
$ sudo yum install -y qpid-cpp-server
$ sudo sed -i -e 's/auth=.*/auth=no/g' /etc/qpidd.conf
$ sudo service qpidd start
$ sudo chkconfig qpidd on
```

Allow incoming connections to Qpid by adding this firewall rule to the **/etc/sysconfig/iptables** configuration file:

```
-A INPUT -p tcp -m multiport --dports 5672 -j ACCEPT
```

Important

This rule allows communication from all remote hosts to the system hosting Qpid on port **5672**. For information regarding the creation of more restrictive firewall rules refer to the Red Hat Enterprise Linux 6 *Security Guide*.

Use the **service** command to restart the **iptables** service for the new rules to take effect.

```
$ sudo service iptables restart
```

The messagebus service for **dbus** is used by **libvirt**. Run the following commands to ensure it is enabled.

```
$ sudo service messagebus start
$ sudo chkconfig messagebus on
```

When Nova creates a virtual machine, it uses **libvirt** to do so. Run the following commands to start up **libvirtd**:

```
$ sudo service libvirtd start
$ sudo chkconfig libvirtd on
```

You previously configured the volume service, Cinder, so configure Nova to use Cinder for volumes:

```
$ sudo openstack-config --set /etc/nova/nova.conf DEFAULT \
  volume_api_class nova.volume.cinder.API
$ sudo openstack-config --set /etc/nova/nova.conf DEFAULT \
  enabled_apis ec2,osapi_compute,metadata
```

You have completed the prerequisite steps for being able to start the Nova services.

Important

Do **not** use the **service** or **chkconfig** commands to start the **openstack-nova-network** service if you intend to install and configure the OpenStack Network service to manage the networking for your environment.

Start the services using the **service** command:

```
$ sudo service openstack-nova-api start
$ sudo service openstack-nova-cert start
$ sudo service openstack-nova-network start
$ sudo service openstack-nova-objectstore start
$ sudo service openstack-nova-scheduler start
$ sudo service openstack-nova-compute start
```

Configure the services to start automatically in future using the **chkconfig** command:

```
$ sudo chkconfig openstack-nova-api on
$ sudo chkconfig openstack-nova-cert on
$ sudo chkconfig openstack-nova-network on
$ sudo chkconfig openstack-nova-objectstore on
$ sudo chkconfig openstack-nova-scheduler on
$ sudo chkconfig openstack-nova-compute on
```

Make sure that the Nova log files do not contain any errors:

```
$ grep -i ERROR /var/log/nova/*
```

Now that the nova services are running, you may find it beneficial to monitor the Nova logs while completing system setup, to do so you can open another terminal and tail the files in **/var/log/nova**:

```
$ tail -f /var/log/nova/*.log
```

Register the Nova compute API as an endpoint with Keystone. Note that the **service_id** passed to the **endpoint-create** command comes from the output of the **service-create** command:

```
$ source ~/keystonerc_admin
$ keystone service-create --name=nova --type=compute --description="Nova
Compute Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |        Nova Compute Service      |
| id          | 9f004f52a97e469b9983d5adefe9f6d0 |
| name        |               nova               |
| type        |              compute             |
+-------------+----------------------------------+
$ keystone endpoint-create --service_id 9f004f52a97e469b9983d5adefe9f6d0 \
  --publicurl "http://192.0.43.10:8774/v1.1/\$(tenant_id)s" \
  --adminurl "http://192.0.43.10:8774/v1.1/\$(tenant_id)s" \
  --internalurl "http://192.0.43.10:8774/v1.1/\$(tenant_id)s"
+-------------+-----------------------------------------+
|   Property  |                  Value                  |
+-------------+-----------------------------------------+
| adminurl    |http://192.0.43.10:8774/v1.1/$(tenant_id)s|
| id          |     247a56ec4fa94afca231aa0c304c7049     |
| internalurl |http://192.0.43.10:8774/v1.1/$(tenant_id)s|
| publicurl   |http://192.0.43.10:8774/v1.1/$(tenant_id)s|
| region      |                regionOne                |
| service_id  |     9f004f52a97e469b9983d5adefe9f6d0     |
+-------------+-----------------------------------------+
```

Important

Ensure that the **publicurl**, **adminurl**, and **internalurl** parameters include the correct IP address for your Nova compute server.

Allow incoming connections to Nova by adding these firewall rules to the **/etc/sysconfig/iptables** configuration file:

```
-A INPUT -p tcp -m multiport --dports 5900:5999 -j ACCEPT
-A INPUT -p tcp -m multiport --dports 6080 -j ACCEPT
-A INPUT -p tcp -m multiport --dports 8773,8774,8775 -j ACCEPT
```

Important

These rules allow communication from all remote hosts to the system hosting the Nova services on ports **5900** to **5999**, the Nova VNC proxy on port **6080**, and the Nova API on ports **8773** to **8775**. For information regarding the creation of more restrictive firewall rules refer to the Red Hat Enterprise Linux 6 *Security Guide*.

Use the **service** command to restart the **iptables** service for the new rules to take effect.

```
$ sudo service iptables restart
```

Set up your environment to use the credentials for your regular user. Then test out the **nova** client application. The **list** command shows running instances. Since no instances have been started yet the output will be empty. The **image-list** command should show that the image you added to Glance is

available for use:

```
$ source ~/keystonerc_username
$ nova list
$ nova image-list
+--------------------------------------+-----------+--------+--------+
|                  ID                  |    Name   | Status | Server |
+--------------------------------------+-----------+--------+--------+
| 17a34b8e-c573-48d6-920c-b4b450172b41 | RHEL 6.2  | ACTIVE |        |
+--------------------------------------+-----------+--------+--------+
```

Use the **nova-manage** command to create a network to use when allocating IP addresses for instances. Note that if you intend to use OpenStack Network for network services then you must skip this step.

```
$ sudo nova-manage network create demonet 10.0.0.0/24 1 256 --
bridge=demonetbr0
```

Warning

The **nova-manage** command does not use the API to manipulate the compute services. The command provides administrators with direct access to manipulate the compute services without authenticating to Keystone,
Additionally the **nova-manage** command does not perform the same level of input validation as commands that access the compute services using the API.
As a result it is recommended that you use alternatives such as the **nova** comamnd to access and manipulate compute services where possible. Where you must use the **nova-manage** then it is recommended that you take care to verify that your inputs are valid and correct **before** running the command.

You can use Nova to create an ssh key pair. When you create an instance, you can specify the name of this key pair and the public key will be placed in the instance so that you can log in using ssh. The output of this command is the private key, save it to a file for later use:

```
$ nova keypair-add oskey > oskey.priv
$ chmod 600 oskey.priv
```

Chapter 12. Deploying OpenStack Networking Services

12.1. Introduction to OpenStack Networking

OpenStack Networking is a virtual network service that aims to provide a rich interface for defining network connectivity and addressing in the OpenStack environment. It does this while providing plug-ins that provide administrators with the flexibility they require to leverage different networking technologies and strategies using plug-ins.

In Red Hat OpenStack 2.1 (Folsom) the OpenStack networking service is offered as an alternative to the Nova networking service.

12.1.1. OpenStack Networking Architecture

OpenStack Networking provides cloud administrators with flexibility in deciding which individual services should run on which physical systems. All service daemons can be run on a single physical host for evaluation purposes. Alternatively each service can have its own physical host or even be replicated across multiple hosts for redundancy.

This chapter focuses on an architecture that combines the role of cloud controller with that of the network host while allowing for multiple compute nodes on which virtual machines instances run. The networking services deployed on the cloud controller in this chapter may also be deployed on a separate network host entirely. This is in fact recommended for environments where it is expected that significant amounts of network traffic will need to be routed from virtual machine instances to external networks.

12.1.2. OpenStack Networking API

OpenStack Networking provides a powerful API to define the network connectivity and addressing used by devices from other services, such as OpenStack Compute.

The OpenStack Compute API has a virtual server abstraction to describe compute resources. Similarly, the OpenStack Networking API has virtual network, subnet, and port abstractions to describe network resources. In more detail:

Network

An isolated L2 segment, analogous to VLAN in the physical networking world.

Subnet

A block of v4 or v6 IP addresses and associated configuration state.

Port

A connection point for attaching a single device, such as the NIC of a virtual server, to a virtual network. Also describes the associated network configuration, such as the MAC and IP addresses to be used on that port.

You can configure rich network topologies by creating and configuring networks and subnets, and then instructing other OpenStack services like OpenStack Compute to attach virtual devices to ports on these networks. In particular, OpenStack Networking supports each tenant having multiple private networks, and allows tenants to choose their own IP addressing scheme, even if those IP addresses overlap with those used by other tenants. This enables very advanced cloud networking use cases, such as building multi-tiered web applications and allowing applications to be migrated to the cloud without changing IP addresses.

Even if a cloud administrator does not intend to expose these capabilities to tenants directly, the OpenStack Networking API can be very useful even when used as an administrative API because it provides significantly more flexibility for the cloud administrator to customize network offerings.

12.1.3. OpenStack Networking Plug-ins

The original OpenStack Compute network implementation assumed a basic networking model where all network isolation was performed through the use of Linux VLANs and IP tables. OpenStack Networking introduces the concept of a *plug-in*, which is a pluggable back-end implementation of the OpenStack Networking API. A plug-in can use a variety of technologies to implement the logical API requests. Some OpenStack Networking plug-ins might use basic Linux VLANs and IP tables, while others might use more advanced technologies, such as L2-in-L3 tunneling or OpenFlow, to provide similar benefits.

Plug-ins for these networking technologies are currently tested and supported for use with Red Hat OpenStack:

- Open vSwitch (*openstack-quantum-openvswitch*)
- Linux Bridge (*openstack-quantum-linuxbridge*)

Other plug-ins that are also packaged and available include:

- Cisco (*openstack-quantum-cisco*)
- NEC OpenFlow (*openstack-quantum-nic*)
- Nicira (*openstack-quantum-nicira*)
- Ryu (*openstack-quantum-ryu*)

Plug-ins enable the cloud administrator to weigh different options and decide which networking technology is right for the deployment.

Important

To use Open vSwitch the latest Red Hat Enterprise Linux kernel must be installed and running. Open vSwitch will not work with a kernel versions lower than version **2.6.32-343.el6.x86_64**. Use the **uname** command to determine the version of the kernel in use.

```
$ uname --kernel-release
2.6.32-343.el6.x86_64
```

If the kernel version displayed is less than **2.6.32-343.el6.x86_64** then use the **yum *update*** command to update the system. Once the update completes successfully reboot the system for the kernel update to take effect.

```
$ sudo yum update -y
```

```
$ sudo reboot
```

12.2. Configuring Keystone for OpenStack Networking

Configuring Keystone to support OpenStack Network involves:

- Creating a OpenStack Network service entry.

- Creating a OpenStack Network endpoint.
- Creating a OpenStack Network service user.

These configuration steps must be performed for OpenStack Network to function correctly.

Procedure 12.1. Configuring Keystone for OpenStack Networking

1. **Authenticating as the OpenStack Administrator**
   Ensure that you have set the environment variables containing the authentication information of the OpenStack administrator by loading them from the **keystonerc_admin** file.

   ```
   $ source ~/keystonerc_admin
   ```

   The **keystonerc_admin** is created during identity services deployment both when performed manually and when performed using **packstack**.
2. **Creating a OpenStack Network Service Entry**
   Use the **keystone *service-create*** command to create the service entry in the catalog.

   ```
   $ keystone service-create --name openstack_network --type network \
         --description 'OpenStack Networking Service'
   ```

   Take note of the service identifier returned, it will be required in subsequent steps.

   > **Example 12.1. OpenStack Network Service Creation**
   > In this example the ***openstack_network*** service and an associated endpoint are created.
   >
   > ```
   > $ keystone service-create --name openstack_network --type network \
   >       --description 'OpenStack Networking Service'
   > +-------------+----------------------------------+
   > |  Property   |              Value               |
   > +-------------+----------------------------------+
   > | description |   OpenStack Networking Service   |
   > | id          | 26a55b340e254ad5bb78c0b14391e153 |
   > | name        |             openstack_network              |
   > | type        |             network              |
   > +-----------+----------------------------------+
   > ```

3. **Creating a OpenStack Network Endpoint**
   Use the **keystone *endpoint-create*** command to create the endpoint entry in the catalog. Replace ***SERVICE_ID*** with the service identifier from the previous step, and replace ***IP*** with the IP address of the OpenStack Network server.

   ```
   $ keystone endpoint-create --service-id SERVICE_ID \
         --publicurl 'http://IP:9696/' \
         --adminurl 'http://IP:9696/' \
         --internalurl 'http://IP:9696/'
   ```

Example 12.2. OpenStack Network Endpoint Creation

```
$  keystone endpoint-create --service-id
26a55b340e254ad5bb78c0b14391e153 \
        --publicurl 'http://192.0.43.10:9696/' \
        --adminurl 'http://192.0.43.10:9696/' \
        --internalurl "http://192.0.43.10:9696/'
+-------------+-----------------------------------------+
|   Property  |                  Value                  |
+-------------+-----------------------------------------+
| adminurl    |          http://192.0.43.10:9696/       |
| id          |     29bd0a74249db11fad3bb22efa96a52a    |
| internalurl |          http://192.0.43.10:9696/       |
| publicurl   |          http://192.0.43.10:9696/       |
| region      |               regionOne                 |
| service_id  |     26a55b340e254ad5bb78c0b14391e153    |
+-------------+-----------------------------------------+
```

4. **Creating a OpenStack Network Service User**

   To enable Nova and some internal components of OpenStack Network to communicate with the OpenStack Network API, you must provide them with administrative user credentials. The suggested approach is to create a service tenant, create a **openstack_network** user within this tenant, and assign this user an **admin** role.

   a. Use the **keystone *tenant-create*** command to create a tenant for use by OpenStack Network. Take note of the tenant identifier returned, it will be required in subsequent steps.

   ```
   $ keystone tenant-create --name openstack_network \
          --description 'OpenStack Network Tenant'
   ```

   Example 12.3. OpenStack Network Tenant Creation

   In this example the **openstack_network** tenant is created.

   ```
   $ keystone tenant-create --name openstack_network \
          --description 'OpenStack Network Tenant'
   +-------------+----------------------------------+
   |   Property  |              Value               |
   +-------------+----------------------------------+
   | description |     OpenStack Network Tenant     |
   |   enabled   |               True               |
   |      id     | c0aa3887450646638335f3ad01bc699  |
   |     name    |         openstack_network        |
   +-------------+----------------------------------+
   ```

   b. Use the **keystone *user-create*** command to create a user for OpenStack Network to use. Replace ***PASSWORD*** with the password that you wish to use for the user, and replace ***TENANT_ID*** with the identifier of the **openstack_network** tenant. Take note of the user identifier returned, it will be required in subsequent steps.

   ```
   $ keystone user-create --name openstack_network \
          --pass 'PASSWORD' \
          --tenant-id TENANT_ID
   ```

Example 12.4. OpenStack Network User Creation

In this example the **openstack_network** user is created.

```
$ keystone user-create --name openstack_network \
       --pass 'PASSWORD' \
       --tenant-id c0aa38874506466383335f3ad01bc699
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |                                  |
| enabled  |               True               |
|    id    | 02425a0a159047c684765479e24361f4 |
|   name   |            openstack_network             |
| password |         PASSWORD_HASH            |
| tenantId | c0aa38874506466383335f3ad01bc699 |
+----------+----------------------------------+
```

c. Use the **keystone** ***role-list*** command to determine the role identifier of the **admin** role. It will be required in subsequent steps.

```
$ keystone role-list
```

Example 12.5. List Keystone Roles

```
$ keystone role-list
+----------------------------------+--------+
|                id                |  name  |
+----------------------------------+--------+
| 6e21529587304dd3837169beb6d0cab5 | admin  |
| ebcaaa525caf4f6b8a599e741b160a10 | Member |
+----------------------------------+--------+
```

d. Use the **keystone** ***user-role-add*** command to add the **admin** role to the **openstack_network** user. Replace ***USER_ID*** with the identifier for the **openstack_network** user, replace ***ROLE_ID*** with the identifier for the **admin** role, and replace ***TENANT_ID*** with the identifier of the **openstack_network** tenant.

```
$ keystone user-role-add --user-id USER_ID \
      --role-id ROLE_ID \
      --tenant-id TENANT_ID
```

Example 12.6. Adding the Administrator Role to the OpenStack Network User

```
$ keystone user-role-add --user-id
02425a0a159047c684765479e24361f4 \
       --role-id 6e21529587304dd3837169beb6d0cab5 \
       --tenant-id c0aa38874506466383335f3ad01bc699
```

You have successfully configured Keystone for OpenStack Network. Refer to Chapter 7, *Deploying Identity Services (Keystone)* for more information about creating service entries and service users.

12.3. Configuring the Cloud Controller or Network Node for OpenStack Networking

Deploying OpenStack Networking involves installing and configuring the OpenStack Networking service on the cloud controller or a separate node which will act as the network node. OpenStack Networking agents on each compute node communicate with the network node to provide network services to virtual machine instances.

12.3.1. Installing the OpenStack Networking Service

The OpenStack Networking service and associated plug-ins must be installed and configured on the system that is to act as the controller or network node.

The OpenStack Networking command line client, **quantum** will also be installed.

Warning

Environments that have been configured to use Nova networking, either using the **packstack** utility, or manually, can be reconfigured to use OpenStack networking. This is however currently **not** recommended for environments where compute instances have already been created and configured to use Nova networking.
If you wish to proceed with such a conversion then you must ensure that you stop the **openstack-nova-network** service on each compute node using the **service** command before proceeding.

```
$ service openstack-nova-network stop
```

You must also disable the **openstack-nova-network** service permanently on each node using the **chkconfig** command.

```
$ chkconfig openstack-nova-network off
```

Procedure 12.2. Installing the OpenStack Networking Service

1. Use the **yum** command to install the *openstack-quantum* package.

   ```
   $ sudo yum install openstack-quantum
   ```

2. Use the **yum** command to install the plug-in that you wish to use.

   ```
   $ sudo yum install openstack-quantum-PLUG-IN
   ```

   Replace ***PLUG-IN*** with the desired plug-in, for example *openvswitch*.

You have successfully installed the OpenStack Networking service and a plug-in. Both the service and the plug-in must be configured before use.

12.3.2. Installing the Message Broker

The OpenStack Network service uses Remote Procedure Calls (RPC) to allow DHCP agents and other plug-in agents to communicate with the **quantum-server** process. To support this a messaging system that supports the Advanced Message Queuing Protocol (AMQP) must be installed and available. The message broker recommended for use with Red Hat OpenStack is QPID.

If QPID was already installed in your environment, for example when configuring compute services, then the same QPID instance can be used by OpenStack Networking. If not then you will need to install QPID on one of the systems in your environment.

To use QPID as the messaging system for RPC you must ensure that QPID is installed on a host reachable via the management network.

Warning

The following commands disable Qpid authentication. This is acceptable for getting started with a simple test deployment. When you are doing a production deployment, however, you should leave authentication enabled.
Consult Qpid documentation to see methods for adding users. Update **`/etc/quantum/quantum.conf`** with the **`qpid_username`** and **`qpid_password`** options so that OpenStack Networking is able to authenticate with Qpid.

Procedure 12.3. Installing the Message Broker

1. Use the **`yum`** command to install the *qpid-cpp-server* package. This package provides the QPID server.

```
$ sudo yum install -y qpid-cpp-server
```

2. Disable QPID authentication by updating the value of the **`auth`** configuration key in the **`/etc/qpidd.conf`** file.

```
$ sudo sed -i -e 's/auth=.*/auth=no/g' /etc/qpidd.conf
```

3. Use the **`service`** command to start the **`qpidd`** service.

```
$ sudo service qpidd start
```

4. Use the **`chkconfig`** command to ensure that the **`qpidd`** service is automatically started in the future.

```
$ sudo chkconfig qpidd on
```

5. Allow incoming connections to Qpid by adding this firewall rule to the **`/etc/sysconfig/iptables`** configuration file:

```
-A INPUT -p tcp -m multiport --dports 5672 -j ACCEPT
```

Important

This rule allows communication from all remote hosts to the system hosting Qpid on port **`5672`**. For information regarding the creation of more restrictive firewall rules refer to the Red Hat Enterprise Linux 6 *Security Guide*.

6. Use the **`service`** command to restart the **`iptables`** service for the new rules to take effect.

```
$ sudo service iptables restart
```

You have successfully installed the QPID messaging system.

12.3.3. Configuring the OpenStack Networking Service

The OpenStack Networking service must be configured to connect to QPID and to setup the chosen plug-in before use.

Note

A number of scripts are provided to assist with deploying OpenStack Network in a variety of configurations. Each of these scripts configures the relevant identity parameters on your behalf. Unlike other components it is not necessary to manually edit the identity parameters.

Procedure 12.4. Configuring the OpenStack Networking Service

1. Ensure that you have set the environment variables containing the authentication information of the OpenStack administrator by loading them from the **keystonerc_admin** file.

   ```
   $ source ~/keystonerc_admin
   ```

2. The OpenStack Networking Service stores its configuration in the **/etc/quantum/quantum.conf** file. Use the **openstack-config** to specify the RPC implementation that the service is to use and where it is hosted.
   a. The ***rpc_backend*** key specifies which RPC implementation to use. Ensure that it is set to **quantum.openstack.common.rpc.impl_qpid**.

      ```
      $ sudo openstack-config --set /etc/quantum/quantum.conf \
      DEFAULT rpc_backend quantum.openstack.common.rpc.impl_qpid
      ```

      This is the default ***rpc_backend*** value for Red Hat OpenStack and may already be set for your installation.
   b. The ***qpid_hostname*** key specifies the management IP address or host name of the server on which QPID is running.

      ```
      $ sudo openstack-config --set /etc/quantum/quantum.conf \
      DEFAULT qpid_hostname IP
      ```

      Important

      If you have configured QPID to require authentication then you must also provide the appropriate values for the ***qpid_username*** and ***qpid_password*** configuration keys.

3. Update the **admin_tenant_name**, **admin_user**, and **admin_password** configuration keys to use the **openstack_network** keystone user created earlier in this chapter.
   a. Set the **admin_tenant_name** configuration key to refer to the **openstack_network** tenant.

```
$ sudo openstack-config --set /etc/quantum/quantum.conf \
      keystone_authtoken admin_tenant_name openstack_network
```

b. Set the **admin_user** configuration key to refer to the **openstack_network** user.

```
$ sudo openstack-config --set /etc/quantum/quantum.conf \
      keystone_authtoken admin_user openstack_network
```

c. Set the **admin_password** configuration key to refer to the password that you defined for the **openstack_network** user.

```
$ sudo openstack-config --set /etc/quantum/quantum.conf \
      keystone_authtoken admin_password PASSWORD
```

4. OpenStack Networking plug-ins require a database server to be installed and configured using their plug-in configuration file.

   OpenStack Networking includes a server setup utility script (**quantum-server-setup**) to automate plug-in configuration. This utility will also update the compute configuration if compute services are found on the system.

   Follow these steps to enable the **openvswitch** OpenStack Networking plug-in:

   a. The OpenStack Networking setup scripts expect the hostname of the local machine to be a resolvable network address. Where this is not the case add an entry to the **/etc/hosts** file mapping the hostname of the system to the loopback IP address (**127.0.0.1**).

   > **Example 12.7. Adding a Mapping to /etc/hosts**
   >
   > In this example the **hostname** command is used to determine the hostname of the system. This value is then written to the **/etc/hosts** file as a valid mapping for the loopback IP address (**127.0.0.1**).
   >
   > ```
   > $ hostname
   > quantum.example.com
   > $ sudo sh -c 'echo 127.0.0.1 `hostname` >> /etc/hosts'
   > $ cat /etc/hosts
   > 127.0.0.1        localhost.localdomain localhost
   > ::1        localhost6.localdomain6 localhost6
   > 127.0.0.1 quantum.example.com
   > ```

   b. Run the **quantum-server-setup** command with the ***--plugin openvswitch*** parameters to configure the ***openvswitch*** plug-in.

   ```
   $ sudo quantum-server-setup --plugin openvswitch
   ```

   c. Enter the password for the MySQL **root** user when prompted.

   ```
   OpenStack Network plugin: openvswitch
   Plugin: openvswitch => Database: ovs_quantum
   Please enter the password for the 'root' MySQL user:
   ```

   d. Enter **y** when prompted to automatically update the Nova compute configuration files. Alternatively enter **n** to skip this step and configure the compute services to use OpenStack Network manually.

```
Verified connectivity to MySQL.
Would you like to update the nova configuration files? (y/n):
y
Configuration updates complete!
```

5. Start the OpenStack Networking server for the first time using the **service** command.

```
$ sudo service quantum-server start
```

6. Enable the OpenStack Networking server to be started automatically in the future using the **chkconfig** command.

```
$ sudo chkconfig quantum-server on
```

7. Allow incoming connections to OpenStack Networking by adding this firewall rule to the **/etc/sysconfig/iptables** configuration file:

```
-A INPUT -p tcp -m multiport --dports 9696 -j ACCEPT
```

Important

This rule allows communication from all remote hosts to the system running the OpenStack Networking services on port **9696**. For information regarding the creation of more restrictive firewall rules refer to the Red Hat Enterprise Linux 6 *Security Guide*.

8. Use the **service** command to restart the **iptables** service for the new rule to take effect.

```
$ sudo service iptables restart
```

You have successfully configured the OpenStack Networking service. You must now configure, or reconfigure, all compute nodes in the environment to use the OpenStack Networking service instead of Nova Networking.

Note

Log files are stored in **/var/log/quantum**.
Configuration files are stored in **/etc/quantum**.

12.4. Configuring Compute Nodes for OpenStack Networking

Unlike Nova-only deployments, when OpenStack Network is in use, Nova should not run the **nova-network** service. Instead, Nova delegates almost all of the network related decisions to the OpenStack Networking Service. This means many of the network related command line and configuration options do not work with OpenStack Networking.

Therefore, it is very important that you refer to this guide when configuring networking, rather than relying on Nova networking documentation or past experience with Nova. If a Nova CLI command or configuration option related to networking is not mentioned in this guide, it is probably not supported for use with OpenStack Networking. In particular, using CLI tools like **nova-manage** and **nova** to manage networks or IP addressing, including both fixed and floating IPs, is not supported with OpenStack

Networking.

Important

It is strongly recommended that you uninstall **nova-network** and reboot any physical nodes that were running nova-network before using them to run OpenStack Network. Inadvertently running the **nova-network** process while using OpenStack Networking service can cause problems, as can stale **iptables** rules pushed down by a previously running **nova-network**.

12.4.1. Configuring Nova to reach the OpenStack Network API

Each time an instance is provisioned or deprovisioned in Nova, Nova communicates with OpenStack Networking via the standard API. To do so, it requires the following items in the **nova.conf** used by each **nova-compute** instance:

network_api_class
: Must be modified from default to **nova.network.quantumv2.api.API** to indicate that OpenStack Network should be used rather than the traditional **nova-network** networking model.

quantum_url
: Must include the host name/IP and port of the OpenStack Network server for this deployment.

quantum_auth_strategy
: Should be kept as default **keystone** for all production deployments.

quantum_admin_tenant_name
: Must be modified to be the name of the service tenant created during the Keystone configuration.

quantum_admin_username
: Must be modified to be the name of the user created during the Keystone configuration.

quantum_admin_password
: Must be modified to be the password of the user created during the Keystone configuration.

quantum_admin_auth_url
: Must be modified to point to the Keystone server IP and port. This is the Identity (Keystone) admin API server IP and port value, and not the Identity service API IP and port.

The **openstack-nova-compute** service must be restarted for changes to these configuration values to take effect.

12.4.2. Configuring Vif-plugging in Nova

When nova-compute creates an instance, it must 'plug' each of the instance's vNICs into a OpenStack Network controlled virtual switch, and inform the virtual switch about the OpenStack Network port-id

associated with each vNIC. This is done by specifying a field in the **`nova.conf`** of the **`nova-compute`** instance indicating what type of **`vif-plugging`** should be used. The exact field(s) you need to set depend on your plug-in. For plug-ins not listed below, see the plug-in specific documentation.

Vif-plugging with Open vSwitch Plugin

The choice of vif-plugging for the Open vSwitch plugin depends on what version of libvirt you are using, as well as whether you are using Nova security filtering (that is: security groups, provider firewall, or VM spoofing prevention).

- When using libvirt (any version) with Nova security filtering: libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybirdOVSBridgeDriver.
- When using libvirt (a version earlier than 0.9.11) without Nova security filtering: libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtOpenVswitchDriver.
- When using libvirt (version 0.9.11 or later) without Nova security groups: libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtOpenVswitchVirtualPortDriver.

Vif-plugging with Linux Bridge Plugin

- When using libvirt (any version): libvirt_vif_driver=nova.virt.libvirt.vif.QuantumLinuxBridgeVIFDriver.

Vif-plugging with Nicira NVP Plugin

The choice of vif-plugging for the NVP Plugin depends on what version of libvirt you are using (this assumes you are using NVP for security groups and VM spoof prevention).

- When using libvirt (a version earlier than 0.9.11): libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtOpenVswitchDriver.
- When using libvirt (version 0.9.11 or later): libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtOpenVswitchVirtualPortDriver.

Vif-plugging with NEC OpenFlow Plugin

The choice of vif-plugging for the NEC plugin depends on what version of libvirt you are using, as well as whether you are using Nova security filtering (that is: security groups, provider firewall, or VM spoofing prevention).

- When using libvirt (any version) with Nova security filtering: libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybirdOVSBridgeDriver.
- When using libvirt (a version earlier than 0.9.11) without Nova security filtering: libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtOpenVswitchDriver.
- When using libvirt (version 0.9.11 or later) without Nova security groups: libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtOpenVswitchVirtualPortDriver.

12.4.3. Example nova.conf (for nova-compute and nova-api)

Below are example values for the above settings, assuming a cloud controller node running Keystone and OpenStack Network with an IP address of 192.168.1.2 and vif-plugging using the LibvirtOpenVswitchDriver with virtio enabled:

```
network_api_class=nova.network.quantumv2.api.API
quantum_url=http://192.0.43.10:9696
quantum_auth_strategy=keystone
quantum_admin_tenant_name=service
quantum_admin_username=openstack_network
quantum_admin_password=PASSWORD
quantum_admin_auth_url=http://192.0.43.10:35357/v2.0

# needed only for nova-compute
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtOpenVswitchDriver
libvirt_use_virtio_for_bridges=True
```

12.5. Installing OpenStack Networking Agents

12.5.1. Installing the Open vSwitch Agent

Many of the OpenStack Networking plug-ins, including Open vSwitch, utilize their own agent. The plug-in specific agent runs on each node that manages data packets. This includes all compute nodes as well as nodes running the dedicated agents **quantum-dhcp-agent** and **quantum-l3-agent**.

Procedure 12.5. Installing the Open vSwitch Agent

1. Use the **yum** to install the *openstack-quantum-openvswitch* package.

   ```
   $ sudo yum install -y openstack-quantum-openvswitch
   ```

2. Ensure that you have set the environment variables containing the authentication information of the OpenStack administrator by loading them from the **keystonerc_admin** file.

   ```
   $ source ~/keystonerc_admin
   ```

3. Run the **quantum-node-setup** script with the ***--plugin openvswitch*** arguments to configure the plug-in. This utility will also update the compute configuration if compute services are found on the system.

   ```
   $ sudo quantum-node-setup --plugin openvswitch
   ```

4. The setup script prompts you to enter a hostname. Enter the IP address or hostname of the QPID server for your environment.

   ```
   Please enter the Quantum hostname:
   ```

5. Enter **y** at the prompt to update the compute configuration files.

   ```
   Would you like to update the nova configuration files? (y/n):
   ```

   A confirmation message is displayed once the required updates have been completed.

   ```
   Configuration updates complete!
   ```

6. Use the **service** command to start the **openvswitch** service.

   ```
   $ sudo service openvswitch start
   ```

7. Use the **chkconfig** command to ensure that the **openvswitch** service is started automatically in the future.

```
$ sudo chkconfig openvswitch on
```

8. Each host running the Open vSwitch agent also requires an Open vSwitch bridge named **br-int**. This bridge is used for private network traffic. Use the **ovs-vsctl** command to create this bridge before starting the agent.

```
$ sudo ovs-vsctl add-br br-int
```

Warning

The **br-int** bridge is required for the agent to function correctly. Once created do not remove or otherwise modify the **br-int** bridge.

9. Use the **service** command to start the **quantum-openvswitch-agent** service.

```
$ sudo service quantum-openvswitch-agent start
```

10. Use the **chkconfig** command to ensure that the **quantum-openvswitch-agent** service is started automatically in the future.

```
$ sudo chkconfig quantum-openvswitch-agent on
```

11. Use the **chkconfig** command to ensure that the **quantum-ovs-cleanup** service is started automatically in the future. When started at boot time this service ensures that the OpenStack Networking agents maintain full control over the creation and management of tap devices.

```
$ sudo chkconfig quantum-ovs-cleanup on
```

You have successfully installed and configured the Open vSwitch agent. Repeat this procedure on all compute nodes in your OpenStack environment and any network nodes running the **quantum-dhcp-agent** or **quantum-l3-agent** services.

12.5.2. Installing the OpenStack Networking DHCP Agent (quantum-dhcp-agent)

The OpenStack Networking DHCP agent is capable of allocating IP addresses to virtual machines running on the network. If the agent is enabled and running and a subnet is created then by default that subnet has DHCP enabled.

Warning

The DHCP agent attempts to use Linux network namespaces in order to support overlapping IP addresses. Red Hat Enterprise Linux does not currently support Linux network namespaces. As such the ***use_namespaces*** configuration key in the agent configuration file is set to **False**.

Procedure 12.6. Installing the OpenStack Networking DHCP Agent (quantum-dhcp-agent)

1. Ensure that you have set the environment variables containing the authentication information of the OpenStack administrator by loading them from the **keystonerc_admin** file.

```
$ source ~/keystonerc_admin
```

2. Run the agent setup script:

```
$ sudo quantum-dhcp-setup --plugin openvswitch
```

3. The setup script prompts you to enter a hostname. Enter the IP address or hostname of the QPID server for your environment.

```
Please enter the Quantum hostname:
```

4. Enter **y** at the prompt to update the compute configuration files.

```
Would you like to update the nova configuration files? (y/n):
```

   A confirmation message is displayed once the required updates have been completed.

```
Configuration updates complete!
```

5. Enable and start the agent:

```
$ sudo service quantum-dhcp-agent start
$ sudo chkconfig quantum-dhcp-agent on
```

12.5.3. Installing the OpenStack Networking L3 Agent (quantum-l3-agent)

The L3 agent is part of the *openstack-quantum* package. It acts as an abstract L3 router that can connect to and provide gateway services for multiple L2 networks.

The nodes on which the L3 agent is to be hosted must not have a manually configured IP address on a network interface that is connected to an external network. Instead there must be a range of IP addresses from the external network that are available for use by OpenStack Networking. These IP addresses will be assigned to the routers that provide the link between the internal and external networks.

The range selected must be large enough to provide a unique IP address for each router in the deployment as well as each desired floating IP.

Procedure 12.7. Installing the OpenStack Networking L3 Agent (quantum-l3-agent)

1. Create both the integration bridge (**br-int**) that will be used for private network traffic and the external bridge (**br-ex**) that will be used by the L3 agent for traffic from external networks.
   a. Create the **br-int** bridge using the **ovs-vsctl** command.

```
$ sudo ovs-vsctl add-br br-int
```

   b. Create the **br-ex** bridge using the **ovs-vsctl** command.

```
$ sudo ovs-vsctl add-br br-ex
```

   Add a network interface that is connected to the external network to the **br-ex** bridge.

```
$ sudo ovs-vsctl add-port br-ex INTERFACE
```

Replace ***INTERFACE*** with the name of the network interface to use, for example ***eth1***.

Warning

The **br-ex** and **br-int** bridges are required for the agent to function correctly. Once created do not remove or otherwise modify the **br-ex** or **br-int** bridge.

2. Ensure that you have set the environment variables containing the authentication information of the OpenStack administrator by loading them from the **keystonerc_admin** file.

```
$ source ~/keystonerc_admin
```

3. Use the **yum** to install the *openstack-quantum* package. The *openstack-quantum* package includes the agent setup script.

```
$ sudo yum install -y openstack-quantum
```

4. Run the agent setup script (**quantum-l3-setup**).

```
$ sudo quantum-l3-setup --plugin openvswitch
```

5. The setup script prompts you to enter a hostname. Enter the IP address or hostname of the QPID server for your environment.

```
Please enter the Quantum hostname:
```

6. Enter **y** at the prompt to update the compute configuration files.

```
Would you like to update the nova configuration files? (y/n):
```

A confirmation message is displayed once the required updates have been completed.

```
Configuration updates complete!
```

7. As Red Hat Enterprise Linux does not yet support network namespaces each L3 agent must have its own router defined. The router must be created in OpenStack Networking and the L3 agent configured to refer to the router using the unique identifier associated with it.

   Additionally this means that it is not currently possible to run OpenStack Networking L3 agents on the same nodes as DHCP agents.

   a. Use the **quantum router-create** command to create a router.

```
$ quantum router-create NAME
Created a new router:
+-----------------------+--------------------------------------+
| Field                 | Value                                |
+-----------------------+--------------------------------------+
| admin_state_up        | True                                 |
| external_gateway_info |                                      |
| id                    | 3129097f-4d02-4610-b77a-47dc7a2904ea |
| name                  | NAME                                 |
| status                | ACTIVE                               |
| tenant_id             | e83f7b909b044419961cc6698fea89e6     |
+-----------------------+--------------------------------------+
```

Replace ***NAME*** with a descriptive name for the new router.

b. It is also necessary to configure the L3 agent to use the new router.

To configure the L3 agent to use the new router open the **/etc/quantum/l3_agent.ini** file with a text editor. Find the line that defines the **router_id** used by the L3 agent.

```
# router_id =
```

Remove the comment character (#) and set the value of the **router_id** configuration key to the identifier of the new router created in the previous step.

```
router_id = 3129097f-4d02-4610-b77a-47dc7a2904ea
```

Save the file and exit your text editor.

8. Enable and start the agent:

```
$ sudo service quantum-l3-agent start
$ sudo chkconfig quantum-l3-agent on
```

12.6. Installing the Client (quantum)

The **quantum** command provides a command line client for interacting with OpenStack Networking. The client is automatically installed on systems where the *quantum-server* package is installed. It can also be installed separately.

Procedure 12.8. Installing the Client (quantum)

- Use the **yum** to install the *python-quantumclient* package.

```
$ sudo yum install -y python-quantumclient
```

Chapter 13. Deploying the Dashboard (Horizon)

The Horizon dashboard provides a web browser accessible interface to an OpenStack environment. It allows users and administrators of the environment to interact with and manage the various functional components without having to install any local client tools other than a web browser.

13.1. Installing Horizon

To install the Horizon dashboard and prepare it for use you must:

- Install the *openstack-dashboard* package.
- Create a **Member** role in Keystone.
- Start the **httpd** service.
- Configure SELinux to allow the **httpd** service to make outbound network connections, allowing it to connect to the Keystone server.
- Configure the firewall to allow incoming connections to the **httpd** service.

Unless otherwise noted all steps in this procedure must be performed while logged in as the **root** user or a user with **sudo** access.

Procedure 13.1. Installing Horizon

1. **Installing the *openstack-dashboard* Package**

   Use **yum install** to install the *openstack-dashboard* package.

   ```
   $ sudo yum install -y openstack-dashboard
   ```

   Important

   Users access the dashboard using HTTP in the default configuration. For security reasons it is recommended that HTTPS is enabled and used to encrypt communications with the dashboard. To support HTTPS you must install the *mod_ssl* package.

   ```
   $ sudo yum install -y mod_ssl
   ```

2. **Creating a Member Role in Keystone**

   Horizon requires a Keystone role named the **Member** role. You must create this role in Keystone prior to using the dashboard.

   a. Log in to the system on which your **keystonerc_admin** file resides and authenticate as the Keystone administrator.

   ```
   $ source ~/keystonerc_admin
   ```

   b. Use the **keystone *role-create*** command to create the **Member** role.

```
$ keystone role-create --name Member
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
| id       | 8261ac4eabcc4da4b01610dbad6c038a |
| name     |              Member              |
+----------+----------------------------------+
```

Note

To configure Horizon to use a role other than the **Member** role change the value of the **OPENSTACK_KEYSTONE_DEFAULT_ROLE** configuration key.
The **OPENSTACK_KEYSTONE_DEFAULT_ROLE** configuration key is stored in the **/etc/openstack-dashboard/local_settings** file.
The **httpd** service must be restarted for the change to take effect.

3. **Configuring httpd**
   a. Use the **service** command to start the **httpd** service.

```
$ sudo service httpd start
```

   b. Use the **chkconfig** command to ensure the **httpd** service starts automatically in future.

```
$ sudo chkconfig httpd on
```

4. **Configuring SELinux**

   Use the **getenforce** command to check the status of SELinux on the system. Possible return values are **Enforcing**, **Permissive**, and **Disabled**.

```
$ getenforce
Enforcing
```

   If SELinux is configured in **Enforcing** mode then you must modify the SELinux policy to allow connections from the **httpd** service to the Keystone server. This is also recommended if SELinux is configured in **Permissive** mode.

   Use **setsebool** command to modify the SELinux policy to allow the **httpd** service to connect to the Keystone server.

```
$ sudo setsebool -P httpd_can_network_connect on
```

5. **Configuring the Firewall**

   To allow users to connect to the dashboard you must configure the system firewall to allow connections. The **httpd** service, and the dashboard, support both HTTP and HTTPS connections. To protect authentication credentials and other data it is highly recommended that you only enable HTTPS connections.

   A. **Allowing HTTPS Connections (Recommended)**

   Allow incoming connections to Horizon using HTTPS by adding this firewall rule to the **/etc/sysconfig/iptables** configuration file:

```
-A INPUT -p tcp -m multiport --dports 443 -j ACCEPT
```

B. **Allowing HTTP Connections**

Allow incoming connections to Horizon using HTTP by adding this firewall rule to the **/etc/sysconfig/iptables** configuration file:

```
-A INPUT -p tcp -m multiport --dports 80 -j ACCEPT
```

Important

These rules allows communication from all remote hosts to the system running the Horizon services on ports **80** or **443**. For information regarding the creation of more restrictive firewall rules refer to the Red Hat Enterprise Linux 6 *Security Guide*.

You must restart the **iptables** service for the changes to take effect.

```
$ sudo service iptables restart
```

You have successfully installed the Horizon dashboard. Use your browser to open the appropriate link for your configuration to access the dashboard for the first time. Replace ***HOSTNAME*** with the host name or IP address of the server on which you installed Horizon:

HTTPS:

https://*HOSTNAME*/dashboard/

HTTP:

http://*HOSTNAME*/dashboard/

When prompted log in using the credentials of your OpenStack user. Refer to Chapter 7, *Deploying Identity Services (Keystone)* for information on creating an OpenStack user.

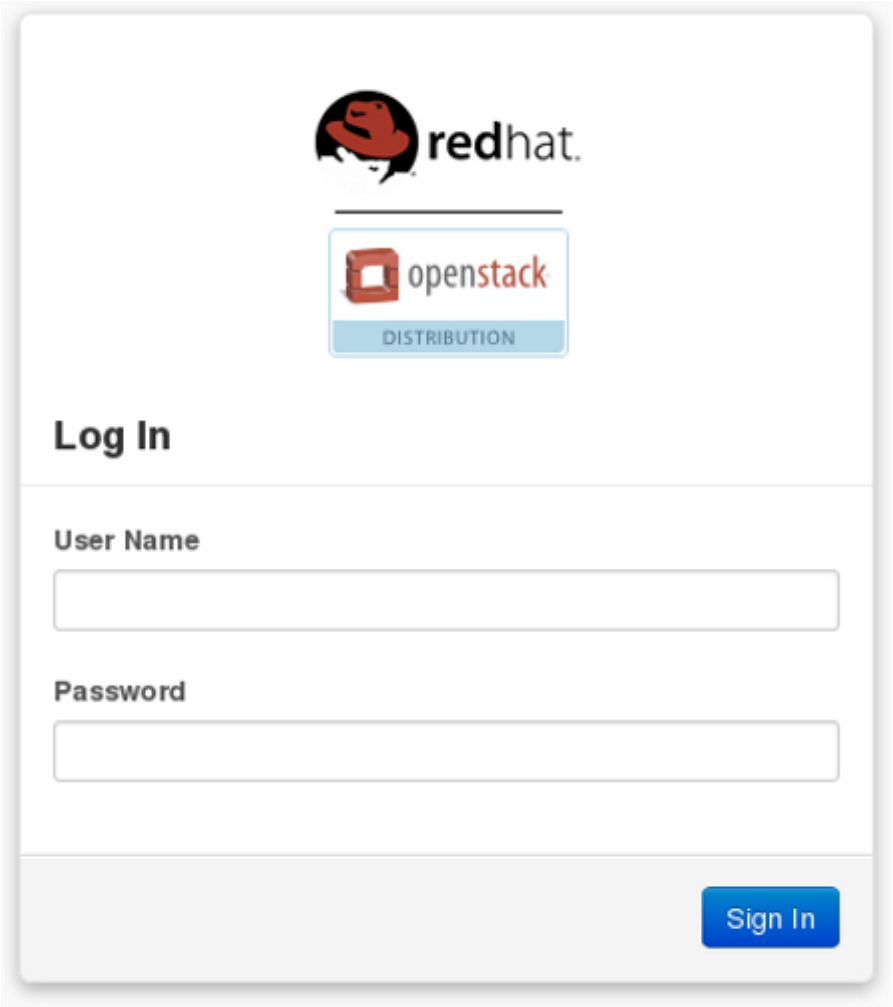


Figure 13.1. The Horizon dashboard login page

13.2. Enabling Console Access

OpenStack includes a VNC proxy. The VNC proxy allows users to connect to the consoles of their running compute instances remotely. Horizon includes support for initiating these connections.

To enable this functionality you must:

- Install and configure an instance of the **`nova-consoleauth`** service.
- Install and configure one or more instances of the **`nova-novncproxy`** service.
- Configure each of the compute hosts in the environment.

Procedure 13.2. Enabling Console Access

1. **Installing and Configuring the Nova VNC Proxy Service**

   The Nova VNC proxy service is provided by the *openstack-nova-novncproxy* package. Use **`yum install`** to install the *openstack-nova-novncproxy* package on the system that is to act as the VNC proxy for the environment.

   ```
   $ sudo yum install -y openstack-nova-novncproxy
   ```

   a. Ensure that you have set the environment variables containing the authentication

information of the OpenStack administrator by loading them from the **keystonerc_admin** file.

```
$ source ~/keystonerc_admin
```

b. The ***novncproxy_host*** configuration key sets the address or host name for the service to bind to when listening for VNC connections. The default is **0.0.0.0** which allows the service to bind on all of the addresses associated with the system.

```
$ sudo openstack-config --set /etc/nova/nova.conf \
DEFAULT novncproxy_host 0.0.0.0
```

c. The ***novncproxy_port*** configuration key sets the port for the service to bind to when listening for VNC connections. The default port is **6080**.

```
$ sudo openstack-config --set /etc/nova/nova.conf \
DEFAULT novncproxy_port 6080
```

2. **Configuring the Nova Compute Nodes**

The Nova configuration is stored in the **/etc/nova/nova.conf** file of each compute node. The changes described here must be applied to each compute node in the environment.

a. The ***novncproxy_base_url*** configuration key sets the public base URL to which client systems will connect. Replace ***FQDN*** with the fully qualified domain name of the VNC proxy server for your environment.

```
$ sudo openstack-config --set /etc/nova/nova.conf \
DEFAULT novncproxy_base_url http://FQDN:6080/vnc_auto.html
```

b. Ensure that the ***vnc_enabled*** configuration key is set to **True**. If this configuration key is set to ***False*** then instances will be launched without VNC support.

```
$ sudo openstack-config --set /etc/nova/nova.conf \
DEFAULT vnc_enabled true
```

c. The ***vncserver_listen*** and ***vncserver_proxyclient_address*** configuration keys determine the address that VNC is actually listening on.

```
$ sudo openstack-config --set /etc/nova/nova.conf \
DEFAULT vncserver_listen 127.0.0.1
```

```
$ sudo openstack-config --set /etc/nova/nova.conf \
DEFAULT vncserver_proxyclient_address 127.0.0.1
```

3. **Starting the Nova VNC Proxy Service**

Both the **openstack-nova-novncproxy** and **openstack-nova-consoleauth** services must be started for the VNC proxy configuration to work.

a. Use the **service** command to start the **openstack-nova-novncproxy** and **openstack-nova-consoleauth** services.

```
$ sudo service openstack-nova-novncproxy start
```

```
$ sudo service openstack-nova-consoleauth start
```

b. Use the **chkconfig** command to ensure that the **openstack-nova-novncproxy** and **openstack-nova-consoleauth** services are started automatically in future.

```
$ sudo chkconfig openstack-nova-novncproxy on
```

```
$ sudo chkconfig openstack-nova-consoleauth on
```

4. Allow incoming connections to the VNC proxy by adding this firewall rule to the **/etc/sysconfig/iptables** configuration file:

```
-A INPUT -p tcp -m multiport --dports 6080 -j ACCEPT
```

Important

This rule allows communication from all remote hosts to the system running the Nova VNC proxy on port **6080**. For information regarding the creation of more restrictive firewall rules refer to the Red Hat Enterprise Linux 6 *Security Guide*.

5. Use the **service** command to restart the **iptables** service for the new rule to take effect.

```
$ sudo service iptables restart
```

6. **Restarting the Nova Compute Service**

   You must restart the **openstack-nova-compute** service on each compute node for the changes to take effect.

```
$ sudo service openstack-nova-compute restart
```

You have successfully configured the compute services to support VNC proxying. Test the VNC proxy by attempting to connect to the VNC console of a running instance from the dashboard.

Part V. Using OpenStack

Once OpenStack is deployed interacting with the environment is primarily done using either the dashboard or the command line interface. This part of the guide provides procedures for performing some common tasks using either of these interfaces.

Note

Commands that begin with `vm$` as opposed to just `$` are commands that should be run inside a virtual machine.

Chapter 14. Launching an Instance

OpenStack Nova supports the launching of compute instances using both the dashboard and the **nova** command line client.

14.1. Launching an Instance using the Dashboard

To launch an instance from the dashboard you must have first:

- Installed the dashboard.
- Created a user with the **Member** role.
- Uploaded an image to use as the basis for your instances.

Procedure 14.1. Launching an Instance using the Dashboard

1. **Log In**
   Log in to the dashboard. Use a user that has the **Member** role.
2. **Click Instances**
   Click **Instances** in the side menu.
3. **Click Launch Instance**
   Click the **Launch Instance** button. The **Launch Instance** dialog is displayed.

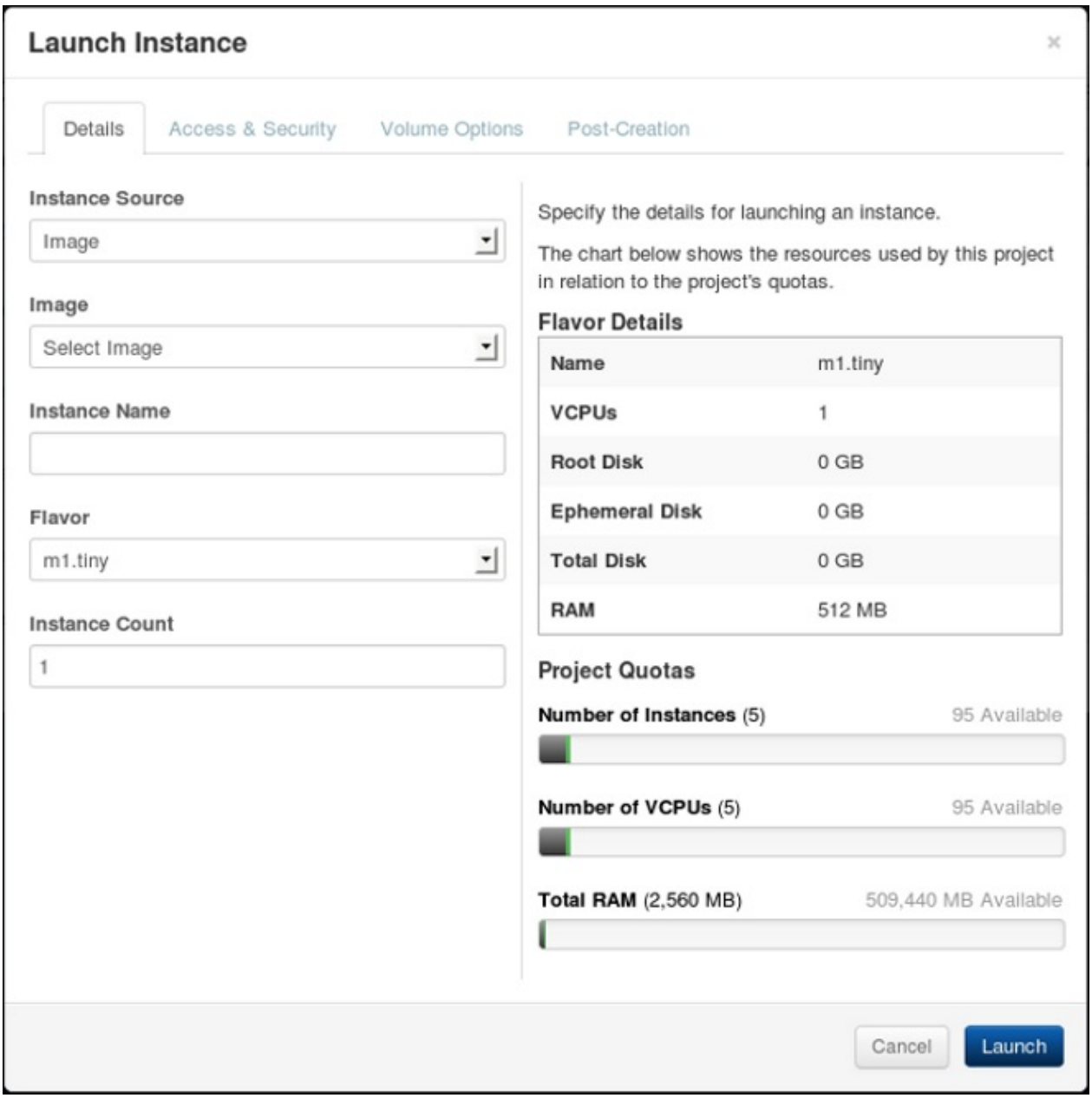


Figure 14.1. Launch Instance Dialog

4. **Configure Details**

   Configure the settings that define your instance on the **`Details`** tab.

   a. Select an **`Instance Source`** for your instance. Available values are:

      - **`Image`**
      - **`Snapshot`**

   b. Select an **`Image`** or **`Snapshot`** to use when launching your instance. The image selected defines the operating system and architecture of your instance.

   c. Enter an **`Instance Name`** to identify your instance.

   d. Select a **`Flavor`** for your instance. The flavor selected determines the compute resources available to your instance. The specific resources for the flavor selected are displayed in the **`Flavor Details`** pane for you to preview.

   e. Enter an **`Instance Count`**. This determines how many instances to launch using the selected options.

5. **Configure Access & Security**

   Click the **Access & Security** tab and configure the security settings for your instance.

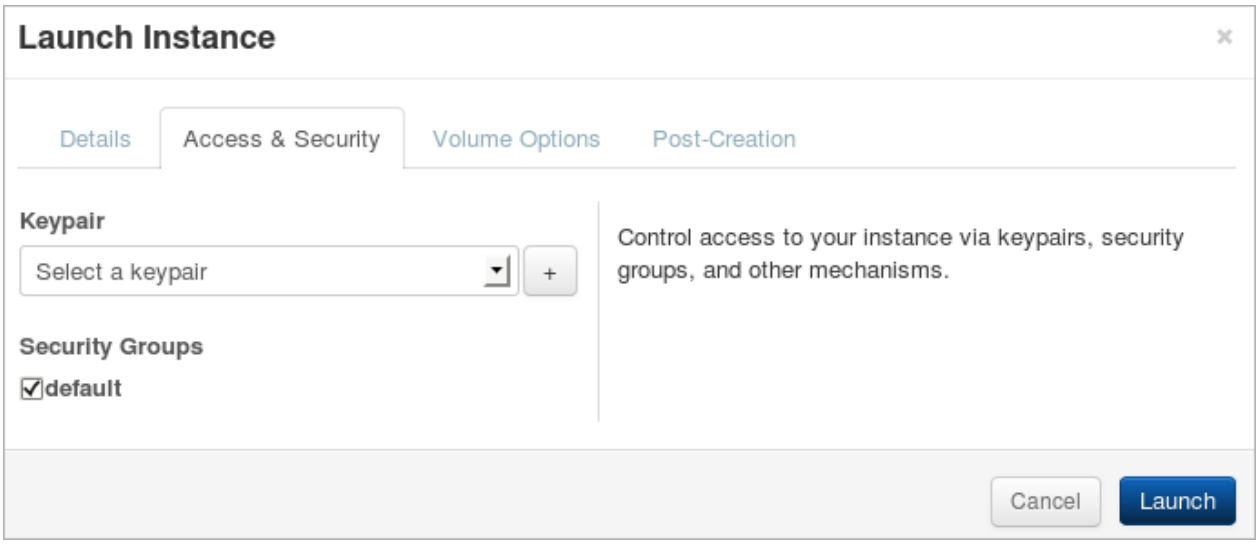


Figure 14.2. Access & Security Tab

   a. A. Select an existing keypair from the **Keypair** drop down box; or

      B. Click the **+** button to upload a new keypair.

   b. Select the **Security Groups** that you wish to apply to your instances. By default only the **default** security group will be available.

6. **Click Launch**

   Click the **Launch** button.

You have created a compute instance. From the **Instances** tab in the dashboard click the name of your instance and then the **VNC** tab on the resultant page to access the console.

14.2. Launching an Instance using the Command Line Interface

When launching an instance using OpenStack, you must specify the ID for the flavor you want to use for the instance. A flavor is a resource allocation profile. For example, it specifies how many virtual CPUs and how much RAM your instance will get. To see a list of the available profiles, run the **nova flavor-list** command.

```
$ nova flavor-list
+----+-----------+-----------+------+-----------+------+-------+-------------+
| ID |    Name   | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor |
+----+-----------+-----------+------+-----------+------+-------+-------------+
| 1  | m1.tiny   | 512       | 0    | 0         |      | 1     | 1.0         |
| 2  | m1.small  | 2048      | 10   | 20        |      | 1     | 1.0         |
| 3  | m1.medium | 4096      | 10   | 40        |      | 2     | 1.0         |
| 4  | m1.large  | 8192      | 10   | 80        |      | 4     | 1.0         |
| 5  | m1.xlarge | 16384     | 10   | 160       |      | 8     | 1.0         |
+----+-----------+-----------+------+-----------+------+-------+-------------+
```

Get the ID of the image you would like to use for the instance using the **nova image-list** command. Create the instance using **nova boot**. If there is not enough RAM available to start an instance, Nova will not do so. Create an instance using flavor **1** or **2**. Once the instance has been created, it will show up in the output of **nova list**. You can also retrieve additional details about the specific instance using

the **nova show** command.

```
$ nova image-list
+--------------------------------------+-----------+--------+--------+
|                  ID                  |    Name   | Status | Server |
+--------------------------------------+-----------+--------+--------+
| 17a34b8e-c573-48d6-920c-b4b450172b41 | RHEL 6.2  | ACTIVE |        |
+--------------------------------------+-----------+--------+--------+
$ nova boot --flavor 2 --key_name oskey --image 17a34b8e-c573-48d6-920c-
b4b450172b41 rhel
+------------------------+--------------------------------------+
|        Property        |                Value                 |
+------------------------+--------------------------------------+
| OS-DCF:diskConfig      | MANUAL                               |
| OS-EXT-STS:power_state | 0                                    |
| OS-EXT-STS:task_state  | scheduling                           |
| OS-EXT-STS:vm_state    | building                             |
| accessIPv4             |                                      |
| accessIPv6             |                                      |
| adminPass              | QVAmyS5i5etE                         |
| config_drive           |                                      |
| created                | 2012-05-18T13:41:40Z                 |
| flavor                 | m1.small                             |
| hostId                 |                                      |
| id                     | 0e4011a4-3128-4674-ab16-dd1b7ecc126e |
| image                  | RHEL 6.2                             |
| key_name               | oskey                                |
| metadata               | {}                                   |
| name                   | rhel                                 |
| progress               | 0                                    |
| status                 | BUILD                                |
| tenant_id              | 05816b0106994f95a83b913d4ff995eb     |
| updated                | 2012-05-18T13:41:40Z                 |
| user_id                | 1d59c0bfef9b4ea9ab63e2a058e68ae0     |
+------------------------+--------------------------------------+
$ nova list
+--------------------------------------+--------+--------+------------------+
|                  ID                  |  Name  | Status |     Networks     |
+--------------------------------------+--------+--------+------------------+
| 0e4011a4-3128-4674-ab16-dd1b7ecc126e | rhel   | BUILD  | demonet=10.0.0.2 |
+--------------------------------------+--------+--------+------------------+
$ nova list
+--------------------------------------+--------+--------+------------------+
|                  ID                  |  Name  | Status |     Networks     |
+--------------------------------------+--------+--------+------------------+
| 0e4011a4-3128-4674-ab16-dd1b7ecc126e | rhel   | ACTIVE | demonet=10.0.0.2 |
+--------------------------------------+--------+--------+------------------+
$ nova show 0e4011a4-3128-4674-ab16-dd1b7ecc126e
```

Once enough time has passed so that the instance is fully booted and initialized, you can ssh into the instance. You can obtain the IP address of the instance from the output of **nova list**.

```
$ ssh -i oskey.priv root@10.0.0.2
```

Chapter 15. Creating a Volume

15.1. Creating a Volume using the Dashboard

Nova compute instances support the attachment and detachment of Cinder storage volumes. This procedure details the steps involved in creating a logical volume in the **cinder-volumes** volume group using the dashboard.

Procedure 15.1. Creating a Volume using the Dashboard

1. **Log In**

   Log in to the dashboard. Use a user that has the **Member** role.
2. **Click Volumes**

   Click **Volumes** in the side menu.
3. **Click Create Volume**

   Click the **Create Volume** button. The **Create Volume** dialog is displayed.

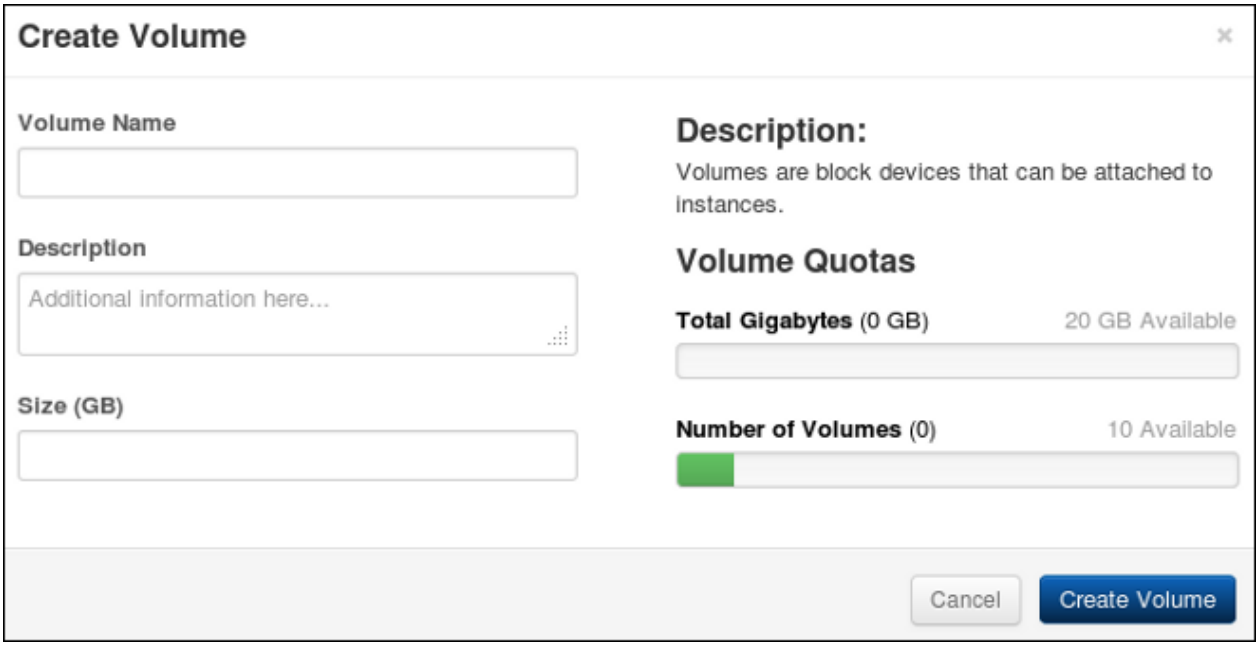


Figure 15.1. Create Volume Dialog

4. **Configure Volume**

   Configure the values that will be used to define your new volume.
   a. Enter a **Volume Name** to identify your new volume by.
   b. Enter a **Description** to further describe your new volume.
   c. Enter the **Size** of your new volume in gigabytes (GB).

Important

Your new volume will be allocated from the **cinder-volumes** volume group. There must be enough free disk space in the **cinder-volumes** volume group for your new volume to be allocated.

5. **Click `Create Volume`**

   Click the **`Create Volume`** button to create the new volume.

You have successfully created a Cinder volume using the dashboard.

15.2. Creating a Volume using the Command Line Interface

Nova compute instances support the attachment and detachment of Cinder storage volumes. This procedure details the steps involved in creating a logical volume in the **`cinder-volumes`** volume group using the **`cinder`** command line interface.

Procedure 15.2. Creating a Volume using the Command Line Interface

1. **Authenticate**

   Use the **`keystonerc_admin`** file to authenticate with Keystone.

   ```
   $ source ~/keystonerc_admin
   ```

2. **Create Cinder Volume**

   Use the **`cinder`** ***`create`*** command to create a new volume.

   ```
   $ cinder create --display_name NAME SIZE
   ```

   Replace ***`NAME`*** with a name to identify your new volume and ***`SIZE`*** with the desired size for the new volume in gigabytes (GB).

You have successfully created a Cinder volume using the command line interface.

Chapter 16. Attaching a Volume

16.1. Attaching a Volume using the Dashboard

This procedure details the steps involved in attaching a Cinder volume to an existing compute instance using the dashboard.

Procedure 16.1. Attaching a Volume using the Dashboard

1. **Log In**

   Log in to the dashboard. Use a user that has the **`Member`** role.

2. **Click `Volumes`**

   Click **`Volumes`** in the side menu.

3. **Click `Edit Attachments`**

   Click the **`Edit Attachments`** button on the row associated with the volume that you want to attach to an instance. The **`Manage Volume Attachments`** dialog is displayed.

4. **Select Instance**

   Select the instance to attach the volume to from the **`Attach to Instance`** box.

5. **Attach Volume**

   Click the **`Attach Volume`** button to attach the volume to the selected instance.

You have successfully attached a Cinder volume to an instance using the dashboard. The volume will appear as a physical hard disk drive to the guest operating system.

16.2. Attaching a Volume using the Command Line Interface

This procedure details the steps involved in attaching a Cinder volume to an existing compute instance using the **`cinder`** and **`nova`** command line interfaces.

Procedure 16.2. Attaching a Volume using the Command Line Interface

1. **Authenticate**

   Use the **`keystonerc_admin`** file to authenticate with Keystone.

   ```
   $ source ~/keystonerc_admin
   ```

2. **Identify the Volume**

   Use the **`cinder list`** command to find available volumes.

   ```
   $ cinder list
   +--------------------------------------+---------+------------+----+---------
   --+
   |                  ID                  | Status  |Display Name|Size|Volume
   Type|
   +--------------------------------------+---------+------------+----+---------
   --+
   |15a9f901-ba9d-45e1-8622-a5438473ae76|available|    NAME    | 1  |
   |
   +--------------------------------------+---------+------------+----+---------
   --+
   ```

   Take note of the **`ID`** of the volume you wish to use. You will need it when attaching the volume to

an instance.

Note

The **Attached to** column has been intentionally omitted from this example output.

3. **Identify the Instance**

   Use the **nova *list*** command to find running instances.

```
$ nova list
+--------------------------------------+------+--------+------------------
---+
|                  ID                  | Name | Status |      Networks
|
+--------------------------------------+------+--------+------------------
---+
| 6842461c-973d-f91b-170a-07324034fbb9 | NAME | ACTIVE | private=192.0.43.10
|
+--------------------------------------+------+--------+------------------
---+
```

   Take note of the **ID** of the instance you wish to use. You will need it when attaching the volume.

4. **Attach the Volume**

   Use the **nova *volume-attach*** to attach the volume to the instance. Replace ***INSTANCE_ID*** with the identifier of the instance and replace ***VOLUME_ID*** with the identifier of the volume.

```
$ nova volume-attach INSTANCE_ID VOLUME_ID auto
```

   The ***auto*** parameter indicates that Nova must attempt to automatically assign a device identifier to the volume within the guest. Manual allocation of specific device identifiers within the guest is not supported on KVM hypervisors at this time.

You have successfully attached a Cinder volume to an instance using the command line interface. The volume will appear as a physical hard disk drive to the guest operating system.

16.3. Accessing a Volume from a Running Instance

Once you attach a volume to an instance a new device will appear to the guest operating system. The device is accessible both using a traditional device label such as **/dev/vd*c*** and in the **/dev/disk/by-id/** tree.

Volumes appear in **/dev/disk/by-id/** with identifiers of the form **virtio-*ID*** where ***ID*** is a subset of the volume identifier assigned when the volume was defined in Cinder.

For example a disk with the identifier **15a9f901-ba9d-45e1-8622-a5438473ae76** in Cinder appears as **/dev/disk/by-id/virtio-15a9f901-ba9d-45e1-8** when viewed from within a virtual machine instance that it is attached to.

```
vm$ ls /dev/disk/by-id/
virtio-15a9f901-ba9d-45e1-8
```

Create a filesystem on the device and mount it in the virtual machine:

```
vm$ mkfs.ext4 /dev/disk/by-id/virtio-15a9f901-ba9d-45e1-8
vm$ mkdir -p /mnt/volume
vm$ mount /dev/disk/by-id/virtio-15a9f901-ba9d-45e1-8 /mnt/volume
```

Write some data to the mounted volume:

```
vm$ echo "Red Hat OpenStack" > /mnt/volume/test.txt
```

Unmount the volume inside the virtual machine.

```
vm$ umount /mnt/volume
```

From the host running Nova, detach the volume from the instance. The **volume-detach** command requires an instance ID and the volume ID you would like to detach from that instance:

```
$ nova volume-detach <instanceid> <volumeid>
```

To verify that the data written to the volume has persisted, you can start up a new instance. Once the new instance is in the **ACTIVE** state, attach the volume to that instance, and then mount the volume in the instance:

```
$ nova boot --image <imageid> --flavor 2 --key_name oskey rhel2
+------------------------+--------------------------------------+
|        Property        |                Value                 |
+------------------------+--------------------------------------+
| OS-DCF:diskConfig      | MANUAL                               |
| OS-EXT-STS:power_state | 0                                    |
| OS-EXT-STS:task_state  | scheduling                           |
| OS-EXT-STS:vm_state    | building                             |
| accessIPv4             |                                      |
| accessIPv6             |                                      |
| adminPass              | uPnzQhpdZZf9                         |
| config_drive           |                                      |
| created                | 2012-05-18T13:45:56Z                 |
| flavor                 | m1.small                             |
| hostId                 |                                      |
| id                     | b8d5c952-f2fc-4556-83f2-57c79378d867 |
| image                  | RHEL 6.2                             |
| key_name               | oskey                                |
| metadata               | {}                                   |
| name                   | rhel2                                |
| progress               | 0                                    |
| status                 | BUILD                                |
| tenant_id              | 05816b0106994f95a83b913d4ff995eb     |
| updated                | 2012-05-18T13:45:56Z                 |
| user_id                | 1d59c0bfef9b4ea9ab63e2a058e68ae0     |
+------------------------+--------------------------------------+
$ nova list
+--------------------------------------+---------+--------+------------------+
|                  ID                  |   Name  | Status |     Networks     |
+--------------------------------------+---------+--------+------------------+
| 0e4011a4-3128-4674-ab16-dd1b7ecc126e | rhel    | ACTIVE | demonet=10.0.0.2 |
| b8d5c952-f2fc-4556-83f2-57c79378d867 | rhel2   | BUILD  | demonet=10.0.0.3 |
+--------------------------------------+---------+--------+------------------+
$ nova list
+--------------------------------------+---------+--------+------------------+
|                  ID                  |   Name  | Status |     Networks     |
+--------------------------------------+---------+--------+------------------+
| 0e4011a4-3128-4674-ab16-dd1b7ecc126e | rhel    | ACTIVE | demonet=10.0.0.2 |
| b8d5c952-f2fc-4556-83f2-57c79378d867 | rhel2   | ACTIVE | demonet=10.0.0.3 |
+--------------------------------------+---------+--------+------------------+
$ nova volume-attach b8d5c952-f2fc-4556-83f2-57c79378d867 15a9f901-ba9d-45e1-
8622-a5438473ae76 auto
$ ssh -i oskey.priv root@10.0.0.3
```

```
vm2$ mkdir -p /mnt/volume
vm2$ mount /dev/disk/by-id/virtio-15a9f901-ba9d-45e1-8 /mnt/volume
vm2$ cat /mnt/volume/test.txt
Red Hat OpenStack
vm2$ umount /mnt/volume
```

And now detach the volume, where the first id is the instance id (Nova) and the second id is the volume id (Cinder):

```
$ nova volume-detach b8d5c952-f2fc-4556-83f2-57c79378d867 15a9f901-ba9d-45e1-
8622-a5438473ae76
```

Chapter 17. Creating a Snapshot

It it possible to create a snapshot of a running instance. This may be done for backup purposes or for creating a base image to create other instances from after applying some customizations to it.

17.1. Creating a Snapshot using the Dashboard

This procedure details the steps involved in creating a snapshot based on a running instance using the dashboard.

Procedure 17.1. Creating a Snapshot using the Dashboard

1. **Log In**

   Log in to the dashboard. Use a user that has the **`Member`** role.
2. **Click `Volumes`**

   Click **`Instances`** in the side menu.
3. **Click `Edit Attachments`**

   Click the **`Create Snapshot`** button on the row associated with the instance that you want to create a snapshot of.

   The **`Create Snapshot`** dialog is displayed.

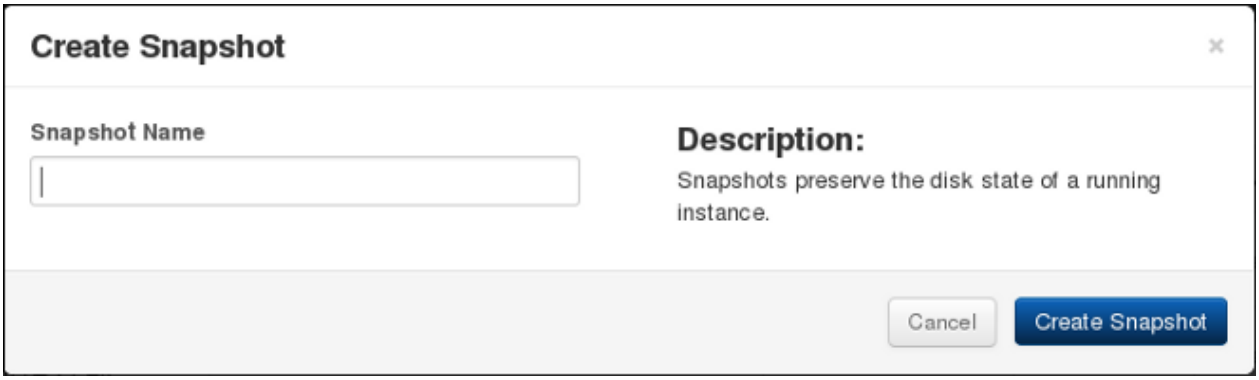


Figure 17.1. Create Snapshot Dialog

4. Enter a descriptive name for your snapshot in the **`Snapshot Name`** field.
5. Click the **`Create Snapshot`** to create the snapshot.
6. The **`Images & Snapshots`** screen is displayed. Your new snapshot will appear in the **`Image Snapshots`** table.

You have successfully created a snapshot of your instance which can be used to restore instance state or as a basis for spawning new instances.

17.2. Creating a Snapshot using the Command Line Interface

As an example, you may want every instance to have a user called **`projectuser`**. Create that user in the virtual machine and then create a snapshot. That snapshot can be used as the base for new instances.

Start by applying some sort of customization to the virtual machine. These commands could be used to create a user and set its password:

```
vm$ useradd projectuser
vm$ passwd projectuser
```

Now create a snapshot of that running instance:

```
$ nova image-create <instanceid> "snapshot 1"
```

The snapshot is complete when its status in **nova image-list** changes from **SAVING** to **ACTIVE**.

```
$ nova image-list
+--------------------------------------+------------+--------+--------+
|                  ID                  |    Name    | Status | Server |
+--------------------------------------+------------+--------+--------+
| 17a34b8e-c573-48d6-920c-b4b450172b41 | RHEL 6.2   | ACTIVE |        |
| 84420f08-1e4b-499a-837a-5c6c1b9903d0 | snapshot 1 | SAVING | ...... |
+--------------------------------------+------------+--------+--------+
$ nova image-list
+--------------------------------------+------------+--------+--------+
|                  ID                  |    Name    | Status | Server |
+--------------------------------------+------------+--------+--------+
| 17a34b8e-c573-48d6-920c-b4b450172b41 | RHEL 6.2   | ACTIVE |        |
| 84420f08-1e4b-499a-837a-5c6c1b9903d0 | snapshot 1 | ACTIVE | ...... |
+--------------------------------------+------------+--------+--------+
```

Once the snapshot's status is active, you can start up a new instance using this image:

```
$ nova boot --image 84420f08-1e4b-499a-837a-5c6c1b9903d0 --flavor 2 --key_name
oskey \
  snapshot-instance
+------------------------+--------------------------------------+
|        Property        |                Value                 |
+------------------------+--------------------------------------+
| OS-DCF:diskConfig      | MANUAL                               |
| OS-EXT-STS:power_state | 0                                    |
| OS-EXT-STS:task_state  | scheduling                           |
| OS-EXT-STS:vm_state    | building                             |
| accessIPv4             |                                      |
| accessIPv6             |                                      |
| adminPass              | QASX3r8jKzVd                         |
| config_drive           |                                      |
| created                | 2012-05-18T13:49:07Z                 |
| flavor                 | m1.small                             |
| hostId                 |                                      |
| id                     | ac9e6a9f-58c3-47c3-9b4c-485aa421b8a8 |
| image                  | snapshot 1                           |
| key_name               | oskey                                |
| metadata               | {}                                   |
| name                   | snapshot-instance                    |
| progress               | 0                                    |
| status                 | BUILD                                |
| tenant_id              | 05816b0106994f95a83b913d4ff995eb     |
| updated                | 2012-05-18T13:49:08Z                 |
| user_id                | 1d59c0bfef9b4ea9ab63e2a058e68ae0     |
+------------------------+--------------------------------------+
 $ nova list
+--------------------------------------+-------------------+--------+-----------
-------+
|                  ID                  |        Name       | Status |
Networks     |
+--------------------------------------+-------------------+--------+-----------
-------+
| 0e4011a4-3128-4674-ab16-dd1b7ecc126e | rhel              | ACTIVE |
demonet=10.0.0.2 |
| ac9e6a9f-58c3-47c3-9b4c-485aa421b8a8 | snapshot-instance | BUILD  |
demonet=10.0.0.4 |
| b8d5c952-f2fc-4556-83f2-57c79378d867 | rhel2             | ACTIVE |
demonet=10.0.0.3 |
+--------------------------------------+-------------------+--------+-----------
-------+
```

Finally, test that the new instance contains the expected customizations made earlier in this exercise. If you followed the example, the instance should have a user named **projectuser**.

```
$ ssh -i oskey.priv root@10.0.0.4
```

```
vm$ su projectuser
```

Chapter 18. Modifying Security Groups

Security groups are used to specify what IP traffic is allowed to reach an instance on its public IP address. The rules defined by security groups are processed before network traffic reaches any firewall rules defined within the guest itself.

18.1. Adding a Rule to a Security Group using the Dashboard

The dashboard interface provides facilities for adding rules to security groups.

Procedure 18.1. Adding a Rule to a Security Group using the Dashboard

1. Log in to the dashboard. Use a user that has the **Member** role.
2. Click **Access & Security** in the side menu.
3. In the **Security Groups** pane click the **Edit Rules** button on the row for the **default** security group.

   The **Edit Security Group Rules** window is displayed.
4. Select the protocol that the rule must apply to from the **IP Protocol** list.
5. Define the range of ports to allow connections on.
   a. Enter the port that defines the start of the range that the rule is to apply to in the **From Port** field.
   b. Enter the port that defines the end of the range that the rule is to apply to in the **To Port** field.

   Note

   A port range of **-1** to **-1** is taken to mean that all valid ports are included.
6. Enter the IP address to accept connections from using Classless Inter-Domain Routing (CIDR) notation. A value of **0.0.0.0/0** allows connections from all IP addresses.

   Alternatively select an existing security group from the **Source Group** list to use the same IP address range selection for this entry.
7. Click the **Add Rule** button to add the new rule to the security group.

You have successfully added a rule to a security group using the dashboard. It is now possible to connect to instances that use the altered security group from the specified IP address block and using the specified ports and protocol.

18.2. Adding a Rule to a Security Group using the Command Line Interface

The **nova** command line interface provides facilities for adding rules to security groups.

Procedure 18.2. Adding a Rule to a Security Group using the Command Line Interface

1. Use the **nova secgroup-list** command to list the security groups that have been defined.

```
$ nova secgroup-list
+---------+-------------+
|   Name  | Description |
+---------+-------------+
| default | default     |
+---------+-------------+
```

On an installation where no security groups have been created yet only the **default** security group will be defined.

2. Use the **nova secgroup-add-rule** command to add a new rule to a security group. The syntax of the **nova secgroup-add-rule** command is:

```
$ nova secgroup-add-rule GROUP \
       PROTOCOL \
       FROM \
       TO \
       CIDR
```

The arguments that the **nova secgroup-add-rule** command expects represent:

GROUP
The identifier of the security group to add the rule to.

PROTOCOL
The IP protocol that the group applies to. Valid values are **icmp**, **tcp**, and **udp**.

FROM
The port that defines the start of the range of ports to allow network traffic on. Valid values are in the range -1 to 65535 for TCP and UDP, -1 to 255 for ICMP.

TO
The port that defines the end of the range of ports to allow network traffic on. Valid values are in the range -1 to 65535 for TCP and UDP, -1 to 255 for ICMP.

CIDR
The Classless Inter-Domain Routing (CIDR) notation defining the IP addresses to accept connections from. A value of **0.0.0.0/0** allows connections from any IP address.

Note

A port range of **-1** to **-1** is taken to mean that all valid ports are included.

3. Use the **nova secgroup-list-rules** to verify that your new rule has been added to the selected security group.

```
$ nova secgroup-list-rules GROUP
```

Replace ***GROUP*** with the identifier of the security group that you added the rule to.

You have successfully added a rule to a security group using the command line interface. It is now possible to connect to instances that use the altered security group from the specified IP address block and using the specified ports and protocol.

Example 18.1. Adding a Security Rule to Allow SSH Connections

In this example a rule is added to the **default** security group to allow SSH access from machines in the IP address block **172.31.0.224/28**.

```
$ nova secgroup-add-rule default tcp 22 22 172.31.0.224/28
+-------------+-----------+---------+-----------------+--------------+
| IP Protocol | From Port | To Port |     IP Range    | Source Group |
+-------------+-----------+---------+-----------------+--------------+
| tcp         | 22        | 22      | 172.31.0.224/28 |              |
+-------------+-----------+---------+-----------------+--------------+
```

Chapter 19. Adding Floating IP Addresses

When an instance is created it is automatically assigned a fixed IP address this IP address is permanently associated with the instance until such time as the instance is terminated.

OpenStack also provides floating IP addresses. Floating IP addresses are associated with instances in addition to their fixed IP addresses. Unlike fixed IP addresses floating IP addresses are able to have their associations modified at any time regardless of the state of the instances involved.

- Define a pool of floating IP addresses.
- Reserve a specific floating IP address from the pool.
- Associate the reserved floating IP address with the instance.

Defining a pool of floating IP addresses is currently only possible using the command line interface. Reserving of addresses and association of addresses with specific instances is possible using both the command line interface and the dashboard.

19.1. Adding Floating IP Addresses using the Dashboard

This procedure details the reservation of a floating IP address from an existing pool of addresses and the association of that address with a specific compute instance. This assumes that a pool of floating IP addresses has already been defined.

Refer to Section 19.2, “Adding Floating IP Addresses using the Command Line Interface” for information about defining a pool of floating IP addresses.

Procedure 19.1. Adding Floating IP Addresses using the Dashboard

1. Log in to the dashboard. Use a user that has the **`Member`** role.
2. Click **`Access & Security`** in the side menu.
3. Click the **`Allocate IP To Project`** button. The **`Allocate Floating IP`** window is displayed.
4. Select a pool of addresses from the **`pool`** list.
5. Click the **`Allocate IP`** button. The allocated IP address will appear in the **`Floating IPs`** table.
6. Locate the newly allocated IP address in the **`Floating IPs`** table. On the same row click the **`Associate Floating IP`** button to assign the IP address to a specific instance.

   The **`Manage Floating IP Associations`** window is displayed.
7. The **`IP Address`** field is automatically set to the selected floating IP address.

   Select the instance to associate the floating IP address with from the **`Instance`** list.
8. Click the **`Associate`** button to associate the IP address with the selected instance.

Note

To disassociate a floating IP address from an instance when it is no longer required use the **`Disassociate Floating IP`** button.

You have successfully associated a floating IP address with an instance using the dashboard.

19.2. Adding Floating IP Addresses using the Command Line Interface

This procedure details the definition of a pool of floating IP addresses. It also covers the reservation of a floating IP address from the pool and the association of that address with a specific compute instance.

Procedure 19.2. Adding Floating IP Addresses using the Command Line Interface

1. Use the **nova-manage floating create** command to define a pool of floating IP addresses.

```
$ sudo nova-manage floating create IP_BLOCK
```

   Replace ***IP_BLOCK*** with the block of IP addresses to use. This value is expressed using CIDR notation.

   Example 19.1. Defining a Pool of Floating IP Addresses

```
$ sudo nova-manage floating create 172.31.0.224/28
```

2. Use the **nova floating-ip-create** command to reserve a floating IP address from the available blocks of public IP addresses.

```
$ nova floating-ip-create
+--------------+-------------+----------+------+
|      Ip      | Instance Id | Fixed Ip | Pool |
+--------------+-------------+----------+------+
| 172.31.0.225 |             |          | nova |
+--------------+-------------+----------+------+
```

3. Use the **nova list** command to identify running instances and select an instance to assign the floating IP address to.

```
$ nova list
+--------------------------------------+------------------+--------+------
------------+
|                  ID                  |        Name        | Status |
Networks     |
+--------------------------------------+------------------+--------+------
------------+
| 0e4011a4-3128-4674-ab16-dd1b7ecc126e | rhel               | ACTIVE |
demonet=10.0.0.1 |
+--------------------------------------+------------------+--------+------
------------+
```

4. Use the **nova add-floating-ip** command to assign the floating IP address that reserved in an earlier step to the selected instance.

```
$ nova add-floating-ip INSTANCE IP
```

   Replace ***INSTANCE*** with the identifier of the selected instance and replace ***IP*** with the floating IP address being assigned to it.

Example 19.2. Assigning a Floating IP Address to an Instance

```
$ nova add-floating-ip 0e4011a4-3128-4674-ab16-dd1b7ecc126e
172.31.0.225
```

5. Periodically check the output of the **nova list** until the floating IP address appears in the output for the selected instance. Once this occurs the instance is accessible using the floating IP address.

Note

To disassociate a floating IP address from an instance when it is no longer required use the **nova remove-floating-ip**.

```
$ nova remove-floating-ip INSTANCE IP
```

Replace ***INSTANCE*** with the identifier of the instance and replace ***IP*** with the floating IP address to remove from it.

You have successfully associated a floating IP address with an instance using the command line interface.

Chapter 20. Controlling Instance State (Suspend, Resume, Reboot, Terminate)

Up to this point you have only booted up instances. There are some other commands that can be used to adjust instance state. You can suspend, resume, reboot, and terminate an instance. The following commands show some examples of doing a suspend, resume, and reboot. Terminating instances is covered in Chapter 21, *Deleting Instances*.

```
$ nova list
+--------------------------------------+------------------+--------+-----------
-------+
|                  ID                  |        Name        | Status |
Networks     |
+--------------------------------------+------------------+--------+-----------
-------+
| 0e4011a4-3128-4674-ab16-dd1b7ecc126e | rhel               | ACTIVE |
demonet=10.0.0.2 |
| ac9e6a9f-58c3-47c3-9b4c-485aa421b8a8 | snapshot-instance | ACTIVE |
demonet=10.0.0.4 |
| b8d5c952-f2fc-4556-83f2-57c79378d867 | rhel2              | ACTIVE |
demonet=10.0.0.3 |
+--------------------------------------+------------------+--------+-----------
-------+
$ nova suspend ac9e6a9f-58c3-47c3-9b4c-485aa421b8a8
$ ping 10.0.0.4   # should not get a response
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
Ctrl+c
--- 10.0.0.4 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2879ms
$ nova resume ac9e6a9f-58c3-47c3-9b4c-485aa421b8a8
$ ping 10.0.0.4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=1685 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=685 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.451 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.394 ms
Ctrl+c
--- 10.0.0.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3607ms
$ nova reboot ac9e6a9f-58c3-47c3-9b4c-485aa421b8a8
$ ssh -i oskey.priv root@10.0.0.4
Last login: Fri May 18 09:50:38 2012 from 10.0.0.1
vm$ uptime
 09:59:09 up 0 min,  1 user,  load average: 0.15, 0.03, 0.01
```

Chapter 21. Deleting Instances

A running instance can be deleted using **`nova delete`**. The following example shows how to delete all running instances:

```
$ nova list
+--------------------------------------+------------------+--------+-----------
-------+
|                  ID                  |        Name        | Status |
Networks     |
+--------------------------------------+------------------+--------+-----------
-------+
| 0e4011a4-3128-4674-ab16-dd1b7ecc126e | rhel               | ACTIVE |
demonet=10.0.0.2 |
| ac9e6a9f-58c3-47c3-9b4c-485aa421b8a8 | snapshot-instance | ACTIVE |
demonet=10.0.0.4 |
| b8d5c952-f2fc-4556-83f2-57c79378d867 | rhel2              | ACTIVE |
demonet=10.0.0.3 |
+--------------------------------------+------------------+--------+-----------
-------+
$ nova delete 0e4011a4-3128-4674-ab16-dd1b7ecc126e
$ nova delete ac9e6a9f-58c3-47c3-9b4c-485aa421b8a8
$ nova delete b8d5c952-f2fc-4556-83f2-57c79378d867
$ nova list
+----+------+--------+----------+
| ID | Name | Status | Networks |
+----+------+--------+----------+
+----+------+--------+----------+
```

Revision History

Revision 1.0-31 Wed Apr 24 2013 Steve Gordon

BZ#927526 - Replaced **admin** user with **openstack_network** user for consistency in the OpenStack Networking content.
BZ#950350 - Added additional upgrade step detailing the comparison and cleanup of **.rpmnew** and **.rpmsave** files.
BZ#911459 - Cleaned up unnecessary and insecure use of **/tmp** directory for temporary file storage.
BZ#921395 - Updated PackStack documentation to include Red Hat Enterprise Linux beta options.
BZ#923017 - Added explicit installation of *python-cinderclient* package for Nova compute nodes.

Revision 1.0-30 Tue Apr 09 2013 Steve Gordon

Updated subtitle to reflect status of release.

Revision 1.0-29 Tue Apr 09 2013 Steve Gordon

BZ#923845 - Added steps for using a Swift storage backend to Glance chapter.
BZ#915788 - Added basic firewall configuration information to each chapter.
BZ#927063 - Updated to remove reference to **./answers.txt** file.
BZ#928348 - Added informational and warning text around Nova to OpenStack Networking conversion.

Revision 1.0-28 Wed Mar 27 2013 Steve Gordon

BZ#895699 - Added additional documentation covering the creation of images using Oz.
BZ#923021 - Moved QPID configuration from the Nova chapter to the Cinder chapter.
BZ#896197 - Updated **packstack** answer file variables and configuration options.
BZ#927526 - Added missing configuration keys to **/etc/quantum/quantum.conf** configuration steps.
BZ#927520 - Added missing **sudo** call to **openstack-config** calls.
BZ#920397 - Updated permissions applied to **/etc/swift/**.
BZ#921395 - Updated **packstack** installation material in response to Quality Engineering review.
BZ#923423 - Removed explicit enablement of Red Hat Enterprise Linux 6 repository.
BZ#922787 - Added missing call to **restorecon** in Swift chapter.

Revision 1.0-27 Fri Mar 22 2013 Steve Gordon

BZ#921782 - Removed API version from Glance endpoint URLs.
BZ#903271 - Removed errant trademark symbols.
BZ#907990 - Corrected errors in volume attachment procedure in response to QE review.
BZ#920457 - Added a caption to Figure 8.1 clarifying the links between the services illustrated.
BZ#923017 - Removed references to the use of **nova-volume**.
BZ#923020 - Corrected ordering of starting the **libvirt** and **messagebus** services as a result of dependencies.
BZ#920456 - Updated to consistently use the **source** instead of the period shortcut.

Revision 1.0-26 Thu Mar 14 2013 Steve Gordon

BZ#920466 - Reduced size of table displayed in Glance chapter to avoid overflow of page.
BZ#918809 - Updated to use **/dev/disk/by-id/** structure to access attached Cinder volumes.
BZ#911101 - Added steps required to switch from Nova Networking to OpenStack Networking.
BZ#892383 - Corrected explanation of ***--location*** parameter to **glance**.
BZ#888773 - Added pointer to create an entry in **/etc/hosts** file before running **quantum-server-setup**.

Revision 1.0-25 **Wed Mar 13 2013** **Steve Gordon**

BZ#920456 - Updated Glance and Cinder chapters to use ***PASSWORD*** placeholder for consistency with other chapters..
BZ#918809 - Updated output of all Glance examples.
BZ#918809 - Removed statements about manual device assignment when attaching volumes. This feature does not work with KVM at this time.
BZ#918615 - Added an overview of what the "all in one" installation actually entails.
BZ#918582 - Added a note detailing the need to install *mod_ssl* for a secure dashboard installation.
BZ#920351 - Added missing **`chkconfig`** commands to Swift chapter.
BZ#920283 - Removed redundant *python-keystone-auth-token* package installation from Swift chapter.
BZ#903321 - Added a note explaining that Red Hat OpenStack has a different default network manager to the OpenStack upstream releases.
BZ#892383 - Updated Glance chapter to provide more detailed information regarding uploading images.
BZ#915461 - Updated all Keystone endpoints to use a "real" IP address instead of loopback.
BZ#917645 - Added admonitions explaining that the **`br-ex`** and **`br-int`** network bridges must not be manually removed or modified.
BZ#888045 - Updated **`subscription-manager`** output to include a valid Red Hat Enterprise Linux subscription (not a beta subscription).
BZ#917326 - Removed email address from OpenStack Networking service definition.
BZ#918992 - Updated **`sudo`** configuration instructions to correctly add the **`wheel`** group to the user.

Revision 1.0-24 **Wed Mar 6 2013** **Steve Gordon**

BZ#903271 - Added system requirements.
BZ#910873 - Moved Keystone service and endpoint creation to correct location in procedure.
BZ#916066 - Removed systemd specific service suffix from volume storage configuration procedure.
BZ#913138 - Updated object storage instructions for using a loopback device.
BZ#903843 - Added additional materials to introduction.
BZ#905944 - Added initial RHOS 2.0 to RHOS 2.1 upgrade instructions.
BZ#889581, BZ#917466 - Updated diagrams for Glance and Keystone services.
BZ#907990 - Expanded "Using OpenStack" material to include more examples of using the dashboard.
BZ#896197 - Documented the ***`--install-hosts`*** argument for **`packstack`**.

Revision 1.0-23 **Tue Feb 26 2013** **Steve Gordon**

BZ#910717 - Added warning regarding nova-manager usage.
BZ#905160 - Updated PackStack material to note automatic creation of answer file.
BZ#889113 - Added step for sourcing keystonerc_admin file to all procedures that require it.
BZ#911194 - Added parameters for Cinder volume creation.
BZ#911349 - Added PackStack options for Satellite based deployments.
BZ#913283 - Changed chapter titles in manual deployment flow to better illustrate contents.

Revision 1.0-22 **Fri Feb 15 2013** **Bruce Reeler**

BZ#888402 - Restructured Nova VNC proxy configuration to make it clear where each step needs to be applied.
BZ#876763 - Updated openstack-config commands for Glance and Cinder configuration.
BZ#889613 - Improved commands in Ch 2 Upgrading from Essex to Folsom.

Revision 1.0-20 **Tue Feb 13 2013** **Steve Gordon**

BZ#910444, BZ#910447 - Added instructions to work around temporary issue with Nova and Cinder management utilities.

Revision 1.0-18 **Mon Feb 12 2013** **Steve Gordon**

BZ#876763 - Updated openstack-config commands for Glance configuration.

BZ#902469 - Added informational message instructing users to install *python-cinderclient* on their Nova compute nodes if using the Cinder volume service.
BZ#888812 - Added warning message instructing users wishing to use the OpenStack Network Service to skip network creation using the **nova-manage** utility.

Revision 1.0-16 Thu Feb 7 2013 Bruce Reeler

BZ#906081 - Updated Figure 1.1. Relationships between OpenStack services.

Revision 1.0-15 Wed Feb 6 2013 Bruce Reeler

BZ#906081 - Renamed "Quantum" to "OpenStack Network".

Revision 1.0-14 Fri Feb 1 2013 Stephen Gordon

BZ#896197 - Added documentation of PackStack non-interactive use case.

Revision 1.0-13 Tue Jan 29 2013 Stephen Gordon

BZ#876763 - Updated authtoken configuration for Nova, Glance and Cinder.
BZ#888343 - Fixed various issues in the OpenStack Network chapter
BZ#888496 - Updated to use Keystone regions consistently.
BZ#891407 - Added information about kernel requirements for Open vSwitch.

Revision 1.0-12 Tue Jan 29 2013 Bruce Reeler

BZ#889306 Fixed typo.
BZ#881869 Replaced deprecated commands 'glance add' and 'glance show'.

Revision 1.0-11 Fri Jan 25 2013 Stephen Gordon

BZ#886178 - Added steps to configure yum-plugin-priorities.
BZ#888073 - Replaced usage of "glance index" which is deprecated.
BZ#888336 - Updated instructions for configuring network interfaces to be more generic.
BZ#888553 - Updated OpenStack Network instructions to start correct services and note need for root access where required.
BZ#889105 - Expanded warning associated with temporary cinder-volumes volume group creation.
BZ#889106 - sections 7.2.2 and 7.2.3 should become subtopics of 7.2.1
BZ#889118 - Added step to Horizon instructions detailing required firewall rules.
BZ#889224 - Added step to Horizon instructions detailing need to persist SELinux change.
BZ#890510 - Added step to Horizon instructions detailing need to add "Member" role to Keystone.
BZ#903920 - Corrected argument list for "glance image-create" example.

Revision 1.0-10 Thu Jan 24 2013 Bruce Reeler

Updated architecture section and replaced architecture diagram.

Revision 1.0-9 Wed Jan 23 2013 Stephen Gordon

BZ#889306 - Updated repository configuration information.
BZ#889526 - Appended "-y" argument to all yum install commands.
BZ#888045 - Updated subscription manager output examples.
BZ#888060 - Updated keystone output examples.
BZ#888087 - Updated description of expected "cinder list" output.
BZ#891370 - Added section detailing expected sudo configuration.
BZ#888064 - Updated example keystonerc files to use correct PS1 values.

Revision 1.0-8 Tue Jan 22 2013 Stephen Gordon

Updated web_version_label.

Revision 1.0-7 **Tue Jan 22 2013** **Bruce Reeler**

BZ888061 RH Summit namings removed, RHEL spelled out, minor edits.
BZ895236 OpenStack Network description added to Intro.

Revision 1.0-6 **Fri Dec 21 2012** **Bruce Reeler**

BZ885070 Missing packages in Folsom added.
BZ889160 Old Essex URL in Nova chapter replaced with Folsom URL.
BZ888061 RH summit refs in example tenant names removed.

Revision 1.0-5 **Wed Dec 12 2012** **Bruce Reeler**

BZ884766 Several commands in OpenStack Network packages installation section replaced.

Revision 1.0-4 **Tue Dec 11 2012** **Bruce Reeler**

BZ884932 Command to subscribe to RHEL beta added.
BZ871703 Broken hyperlink in Horizon chapter to reach dashboard replaced with example URL.

Revision 1.0-3 **Fri Dec 7 2012** **Bruce Reeler**

BZ884762 Cinder Keystone service-create cmd screen output example corrected.
BZ881844 & BZ876775 typos.
BZ877289 subscription update Essex to Folsom.

Revision 1.0-2 **Thu Nov 29 2012** **Bruce Reeler**

Information on subscription added to Section 1.1 Repository Config and section 1.2 Getting Started.

Revision 1.0-1 **Mon Nov 12 2012** **Bruce Reeler**

Edits to Guide for Folsom Preview release.

Revision 1.0-0 **Sat Nov 10 2012** **Bruce Reeler**

First version of the Folsom Preview guide.