



# **Red Hat OpenStack Red Hat OpenStack 3.0 (Grizzly) Installation and Configuration Guide**

---

Installing and Configuring OpenStack environments manually  
Edition 1

Steve Gordon  
Summer Long

Steve Gordon  
Scott Radvan

Tim Hildred  
Joshua Wulf



# Red Hat OpenStack Red Hat OpenStack 3.0 (Grizzly) Installation and Configuration Guide

---

## Installing and Configuring OpenStack environments manually

### Edition 1

Steve Gordon  
sgordon@redhat.com

Tim Hildred  
thildred@redhat.com

Summer Long  
slong@redhat.com

Scott Radvan  
sradvan@redhat.com

Joshua Wulf  
jwulf@redhat.com

## **Legal Notice**

Copyright 2013 Red Hat, Inc. The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at [http://creativecommons.org/licenses/by-sa/3.0/](#). In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version. Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law. Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, MetaMatrix, Fedora, the Infinity Logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries. Linux is the registered trademark of Linus Torvalds in the United States and other countries. Java is a registered trademark of Oracle and/or its affiliates. XFS is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries. MySQL is a registered trademark of MySQL AB in the United States, the European Union and other countries. All other trademarks are the property of their respective owners. 1801 Varsity Drive Raleigh, NC 27606-2072 USA Phone: +1 919 754 3700 Phone: 888 733 4281 Fax: +1 919 754 3701

## **Keywords**

## **Abstract**

Installing and Configuring OpenStack environments with Red Hat Enterprise Linux OpenStack Platform 3 (Grizzly).

# Table of Contents

<b>Preface</b> .....	<b>8</b>
1. Document Conventions	8
1.1. Report a Bug Links	8
1.2. Programming Language Selection	8
1.3. Typographic Conventions	8
1.4. Pull-quote Conventions	9
1.5. Notes and Warnings	10
2. Getting Help and Giving Feedback	10
<b>Part I. Introduction</b> .....	<b>12</b>
<b>Chapter 1. Product Introduction</b> .....	<b>13</b>
1.1. Overview	13
1.2. Architecture	13
1.3. Service Details	14
1.3.1. Dashboard Service	14
1.3.2. Identity Service	15
1.3.3. OpenStack Networking Service	16
1.3.4. Block Storage Service	16
1.3.5. Compute Service	17
1.3.6. Image Service	19
1.3.7. Object Storage Service	20
1.3.8. Metering (Technical Preview)	21
1.3.9. Orchestration (Technical Preview)	22
<b>Chapter 2. Prerequisites</b> .....	<b>23</b>
2.1. Software Requirements	23
2.1.1. Operating System Requirements	23
2.1.2. Software Repository Configuration	23
2.1.2.1. Register to Red Hat Network	23
2.1.2.2. Red Hat Enterprise Linux Repository Configuration	24
2.1.2.3. Red Hat Enterprise Linux OpenStack Platform Repository Configuration	25
2.2. Hardware Requirements	29
2.2.1. Compute Node Requirements	29
2.2.2. Network Node Requirements	30
2.2.3. Block Storage Node Requirements	30
<b>Part II. Installing OpenStack</b> .....	<b>32</b>
<b>Chapter 3. Installing the Database Server</b> .....	<b>33</b>
3.1. Installing the Packages	33
3.2. Configuring the Firewall	33
3.3. Starting the Database Service	33
3.4. Setting the Database Administrator Password	34
<b>Chapter 4. Installing the Message Broker</b> .....	<b>35</b>
4.1. Installing the Packages	35
4.2. Configuring the Message Broker	35
4.2.1. Simple Authentication and Security Layer - SASL	35
4.2.1.1. SASL - Simple Authentication and Security Layer	35
4.2.1.2. SASL Mechanisms	35
4.2.1.3. SASL Mechanisms and Packages	36
4.2.1.4. Configure SASL using a Local Password File	36

4.2.2. Configuring TLS/SSL	37
4.2.2.1. Encryption Using SSL	37
4.2.2.2. Enable SSL on the Broker	37
4.2.2.3. Export an SSL Certificate for Clients	38
4.3. Firewall Configuration	38
4.4. Starting the Messaging Server	39
<b>Chapter 5. Installing the OpenStack Identity Service</b>	<b>40</b>
5.1. Identity Service Requirements	40
5.2. Installing the Packages	40
5.3. Creating the Identity Database	40
5.4. Configuring the Service	41
5.4.1. Setting the Administration Token	41
5.4.2. Setting the Database Connection String	42
5.4.3. Configuring the Public Key Infrastructure	42
5.4.3.1. Public Key Infrastructure Overview	42
5.4.3.2. Creating the Public Key Infrastructure Files	43
5.4.4. Configuring for an LDAP Backend	43
5.4.5. Configuring the Firewall	46
5.4.6. Populating the Identity Service Database	47
5.5. Starting the Identity Service	47
5.6. Creating the Identity Service Endpoint	47
5.7. Creating an Administrator Account	48
5.8. Creating a Regular User Account	50
5.9. Creating the Services Tenant	51
5.10. Validating the Identity Service Installation	52
<b>Chapter 6. Installing the OpenStack Object Storage Service</b>	<b>54</b>
6.1. Services that Make Up the Object Storage Service	54
6.2. Architecture of the Object Storage Service	54
6.3. Object Storage Service Requirements	55
6.4. Installing the Object Storage Service Packages	56
6.5. Configuring the Object Storage Service	57
6.5.1. Configuring the Identity Service to work with the Object Storage Service	57
6.5.2. Configuring the Object Storage Service Storage Nodes	58
6.5.3. Configuring the Object Storage Service Proxy Service	59
6.5.4. Object Storage Service Rings	61
6.5.5. Building Object Storage Service Ring Files	61
6.6. Validating the Object Storage Service Installation	63
<b>Chapter 7. Installing the OpenStack Image Service</b>	<b>65</b>
7.1. Image Service Requirements	65
7.2. Installing the Image Service Packages	65
7.3. Creating the Image Service Database	65
7.4. Configuring the Image Service	66
7.4.1. Configuration Overview	66
7.4.2. Creating the Image Identity Records	66
7.4.3. Setting the Database Connection String	68
7.4.4. Configuring the Use of the Identity Service	68
7.4.5. Using the Object Storage Service for Image Storage	69
7.4.6. Configuring the Firewall	70
7.4.7. Populating the Image Service Database	70
7.5. Starting the Image API and Registry Services	71
7.6. Validating the Image Service Installation	71
7.6.1. Obtaining a Test Disk Image	71
7.6.2. Building a Custom Disk Image	71

7.6.3. Uploading a Disk Image	73
<b>Chapter 8. Installing OpenStack Block Storage</b>	<b>77</b>
8.1. Block Storage Installation Overview	77
8.2. Block Storage Prerequisite Configuration	78
8.2.1. Creating the Block Storage Database	78
8.2.2. Creating the Block Storage Identity Records	79
8.3. Common Block Storage Configuration	80
8.3.1. Installing the Block Storage Service Packages	80
8.3.2. Configuring Authentication	81
8.3.3. Setting the Message Broker	81
8.3.4. Setting the Database Connection String	83
8.3.5. Configuring the Firewall	83
8.3.6. Populating the Block Storage Database	84
8.4. Volume Service Specific Configuration	84
8.4.1. Block Storage Driver Support	84
8.4.2. Configuring for LVM Storage Backend	84
8.4.3. Configuring for NFS Storage Backend	85
8.4.4. Configuring for Red Hat Storage Backend	86
8.4.5. Configuring for Multiple Storage Backends	88
8.4.6. Configuring tgt	89
8.5. Starting the Block Storage Services	90
8.6. Validating the Block Storage Service Installation	91
<b>Chapter 9. Installing the OpenStack Networking Service</b>	<b>93</b>
9.1. OpenStack Networking Installation Overview	93
9.1.1. OpenStack Networking Architecture	93
9.1.2. OpenStack Networking API	93
9.1.3. OpenStack Networking API Extensions	94
9.1.4. OpenStack Networking Plug-ins	95
9.1.5. OpenStack Networking Agents	95
9.1.6. Recommended Networking Deployment	96
9.2. Networking Prerequisite Configuration	97
9.2.1. Creating the OpenStack Networking Database	97
9.2.2. Creating the OpenStack Networking Identity Records	98
9.3. Common Networking Configuration	99
9.3.1. Upgrading the Kernel	99
9.3.2. Disabling Network Manager	100
9.3.3. Installing the Packages	102
9.3.4. Configuring the Firewall	102
9.4. Configuring the Networking Service	103
9.5. Configuring the DHCP Agent	107
9.6. Configuring a Provider Network	108
9.7. Configuring the Plug-in Agent	110
9.7.1. Configuring the Open vSwitch Plug-in Agent	110
9.7.2. Configuring the Linux Bridge Plug-in Agent	111
9.8. Configuring the L3 Agent	112
9.9. Validating the OpenStack Networking Installation	114
<b>Chapter 10. Installing the OpenStack Compute Service</b>	<b>116</b>
10.1. Compute Service Requirements	116
10.1.1. Checking for Hardware Virtualization Support	116
10.2. Installing a Compute VNC Proxy	116
10.2.1. Installing the Compute VNC Proxy Packages	116
10.2.2. Configuring the Firewall	117
10.2.3. Controlling the VNC Proxy service	117

10.2.4. Accessing Instances with the Compute VNC Proxy	118
10.3. Installing a Compute Node	119
10.3.1. Creating the Compute Service Database	119
10.3.2. Creating the Compute Identity Records	120
10.3.3. Installing the Compute Service	121
10.3.4. Configuring the Compute Service	122
10.3.4.1. Configuring Authentication	122
10.3.4.2. Setting the Database Connection String	123
10.3.4.3. Setting the Message Broker	123
10.3.4.4. Configuring Resource Overcommitment	124
10.3.4.5. Reserving Host Resources	125
10.3.4.6. Configuring Compute Networking	125
10.3.4.6.1. Compute Networking Overview	125
10.3.4.6.2. Updating the Compute Configuration	126
10.3.4.6.3. Configuring the L2 Agent	127
10.3.4.6.4. Configuring Virtual Interface Plugging	127
10.3.4.7. Configuring the Firewall	128
10.3.5. Populating the Compute Service Database	128
10.3.6. Starting the Compute Services	129
<b>Chapter 11. Installing the Dashboard</b>	<b>131</b>
11.1. Dashboard Service Requirements	131
11.2. Installing the Dashboard Packages	131
11.3. Starting the Apache Web Service	132
11.4. Configuring the Dashboard	132
11.4.1. Configuring Connections and Logging	132
11.4.2. Configuring Secured Deployment (HTTPS)	133
11.4.3. Creating a Member Role	134
11.4.4. Configuring SELinux	135
11.4.5. Configuring the Dashboard Firewall	135
11.4.6. Session Storage Options	136
11.4.6.1. Configuring Local Memory Cache Session Storage	136
11.4.6.2. Configuring Memcached Session Storage	136
11.4.6.3. Configuring Database Session Storage	136
11.4.6.4. Configuring Cached Database Session Storage	138
11.4.6.5. Configuring Cookies Session Storage	138
11.5. Validating the Dashboard Installation	139
<b>Part III. Validating the Installation</b>	<b>140</b>
<b>Chapter 12. Working with Instances</b>	<b>141</b>
12.1. Uploading a Disk Image	141
12.2. Creating a Keypair	142
12.3. Creating a Network	143
12.4. Launching an Instance	144
12.5. Creating a Volume	146
12.6. Attaching a Volume to an Instance	147
12.7. Creating an Instance Snapshot	148
12.8. Controlling the State of an Instance (Pause, Suspend, Reboot)	149
<b>Chapter 13. Updating the Environment</b>	<b>151</b>
13.1. Defining a Floating IP-Address Pool	151
13.2. Creating a Router	152
13.3. Associating a Floating IP with the Instance	154
13.4. Adding a Rule to a Security Group	155
<b>Part IV. Monitoring the OpenStack Environment</b>	<b>157</b>



<b>Chapter 14. Monitoring OpenStack using Nagios</b> .....	<b>158</b>
14.1. Installing Nagios	158
14.1.1. Installing the Nagios Service	158
14.1.2. Installing the NRPE Addon	159
14.2. Configuring Nagios	159
14.2.1. Setting up Nagios	159
14.2.2. Configuring HTTPD	160
14.2.3. Configuring OpenStack Services	161
14.2.4. Configuring NRPE	162
14.2.5. Creating Host Definitions	163
14.2.6. Creating Service Definitions	163
14.2.7. Verifying the Configuration	164
<b>Chapter 15.. Installing and Configuring Remote Logging</b> .....	<b>166</b>
15.1. Introduction to Remote Logging	166
15.2. Installing rsyslog Server	166
15.3. Configuring rsyslog on the Centralized Logging Server	166
15.4. Configuring rsyslog on the Individual Nodes	167
15.5. Starting rsyslog Server	167
<b>Part V. Managing OpenStack Environment Expansion</b> .....	<b>169</b>
<b>Chapter 16.. Managing Compute Expansion</b> .....	<b>170</b>
16.1. Defining Regions	170
16.2. Adding Compute Resources	170
16.3. Safely Removing Compute Resources	171
16.4. Using Config Drive	172
16.4.1. Config Drive Overview	172
16.4.2. Setting Up Config Drive	173
16.4.2.1. Enabling Config Drive	173
16.4.2.2. Config Drive Options	173
16.4.3. Accessing Config Drive	174
16.4.4. Data Formats	175
16.4.4.1. OpenStack Metadata Format	175
16.4.4.2. EC2 Metadata Format	176
16.4.4.3. User Data Format	177
<b>Chapter 17.. Managing Quotas</b> .....	<b>178</b>
17.1. Viewing and Updating Quotas in the Dashboard	178
17.2. Updating Compute Service Quotas on the Command Line	179
17.3. Updating Block Storage Service Quotas on the Command Line	180
<b>Installation Checklist</b> .....	<b>182</b>
A.1. Installation Prerequisites Checklists	182
<b>Troubleshooting the OpenStack Environment</b> .....	<b>187</b>
B.1. No Networks or Routers Tab Appears in the Dashboard	187
B.2. Dashboard Reports ERROR When Launching Instances	187
B.3. Compute Instance Log Shows no Output	187
B.4. Identity Client (keystone) Reports "Unable to communicate with identity service"	188
<b>Service Log Files</b> .....	<b>190</b>
C.1. Block Storage Service Log Files	190
C.2. Compute Service Log Files	190
C.3. Dashboard Log Files	191
C.4. Identity Service Log Files	191

C.5. Image Service Log Files	191
C.6. Networking Service Log Files	191
C.7. Object Storage Service Log Files	192
<b>Example Configuration Files</b> .....	<b>193</b>
D.1. Dashboard Service Configuration Files	193
D.2. Block Storage Service Configuration Files	197
D.2.1. api-paste.ini	197
D.2.2. cinder.conf	198
D.2.3. policy.json	199
D.2.4. rootwrap.conf	199
D.3. Compute Service Configuration Files	200
D.3.1. api-paste.ini	200
D.3.2. nova.conf	202
D.3.3. policy.json	204
D.3.4. rootwrap.conf	204
D.4. Identity Service Configuration Files	204
D.4.1. keystone.conf	204
D.5. Image Service Configuration Files	210
D.5.1. glance-registry.conf	210
D.5.2. glance-registry-paste.ini	212
D.5.3. glance-api.conf	212
D.5.4. glance-api-paste.ini	218
D.5.5. glance-scrubber.conf	220
D.6. Networking Service Configuration Files:	220
D.6.1. api-paste.ini	220
D.6.2. dhcp_agent.ini	221
D.6.3. l3_agent.ini	221
D.6.4. lbaas_agent.ini	222
D.6.5. metadata_agent.ini	223
D.6.6. policy.json	223
D.6.7. quantum.conf	224
D.6.8. rootwrap.conf	230
D.7. Object Storage Service Configuration Files	230
D.7.1. account-server.conf	230
D.7.2. container-server.conf	231
D.7.3. object-server.conf	231
D.7.4. proxy-server.conf	231
<b>Revision History</b> .....	<b>233</b>



# Preface

## 1. Document Conventions

This manual has several features to aid you to discover relevant information and provide feedback.

### 1.1. Report a Bug Links

In the HTML version, each section in this manual has a link at the bottom on the right-hand margin. This link allows you to easily provide feedback about a specific section. Use this link when you encounter incorrect or incomplete information, or would like to suggest or request an alteration or addition.

### 1.2. Programming Language Selection

This manual may document examples using multiple programming languages. In that case, the HTML version of the manual may offer a selection of programming language samples in a tabbed dialog. Select the programming language that you wish to view. You can then choose to set this programming language as the default for the entire book by clicking the "Set as Default" link that appears.

### 1.3. Typographic Conventions

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

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.

## 1.4. 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"));
    }
}

```

## 1.5. 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.

## 2. Getting Help and Giving Feedback

If you experience difficulty with a procedure described in this documentation, visit the Red Hat Customer Portal at <http://access.redhat.com>. Through the customer portal, you can:

- ▶ search or browse through a knowledge base of technical support articles about Red Hat products.
- ▶ submit a support case to Red Hat Global Support Services (GSS).
- ▶ access other product documentation.

Red Hat also hosts a large number of electronic mailing lists for discussion of Red Hat software

and technology. You can find a list of publicly available mailing lists at <https://www.redhat.com/mailman/listinfo>. Click on the name of any mailing list to subscribe to that list or to access the list archives.

## **We Need Feedback**

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you. Please submit a report in Bugzilla: <http://bugzilla.redhat.com/> against the product **Red Hat OpenStack**.

When submitting a bug report, be sure to mention the manual's identifier: [\*doc-Installation and Configuration Guide\*](#)

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

# Part I. Introduction



# Chapter 1. Product Introduction

## 1.1. Overview

Red Hat Enterprise Linux OpenStack Platform provides the foundation to build a private or public Infrastructure-as-a-Service (IaaS) cloud on top of Red Hat Enterprise Linux. It offers a massively scalable, fault-tolerant platform for the development of cloud-enabled workloads.

The current Red Hat system is based on OpenStack Grizzly, and packaged so that available physical hardware can be turned into a private, public, or hybrid cloud platform including:

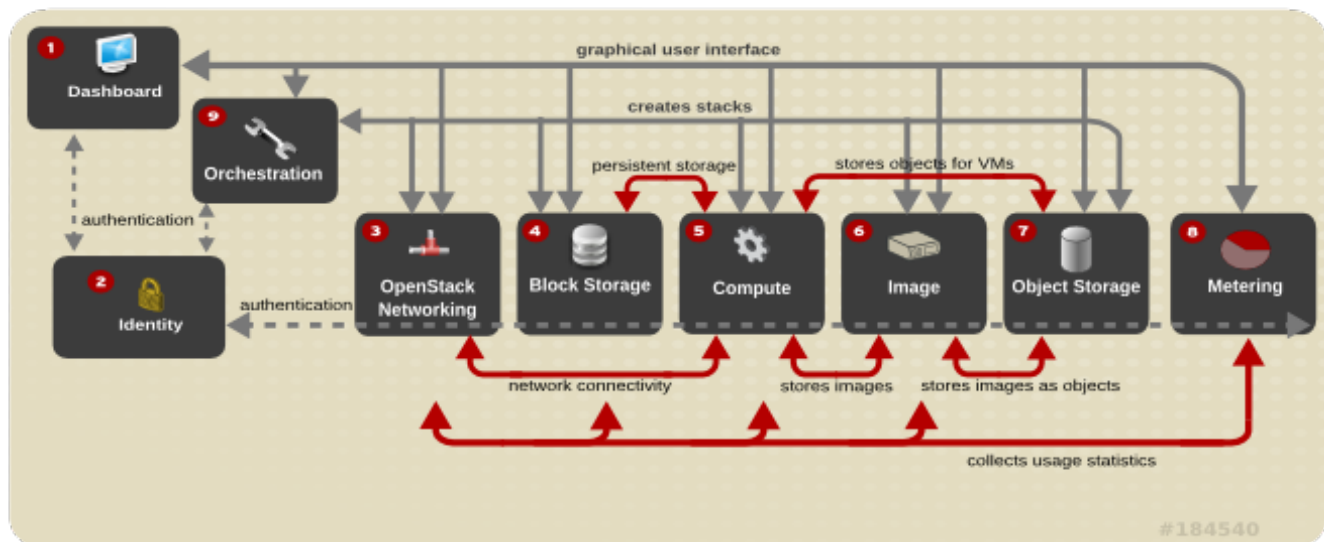
- ▶ Fully distributed object storage
- ▶ Persistent block-level storage
- ▶ Virtual-machine provisioning engine and image storage
- ▶ Authentication and authorization mechanism
- ▶ Integrated networking
- ▶ Web browser-based GUI for both users and administration.

The Red Hat Enterprise Linux OpenStack Platform IaaS cloud is implemented by a collection of interacting services that control its computing, storage, and networking resources. The cloud is managed using a web-based interface which allows administrators to control, provision, and automate OpenStack resources. Additionally, the OpenStack infrastructure is facilitated through an extensive API, which is also available to end users of the cloud.

[Report a bug](#)

## 1.2. Architecture

The following diagram provides a high-level overview of the OpenStack architecture.



Each OpenStack service has a code name, which is reflected in the names of configuration files and command-line utility programs. For example, the Identity service has a configuration file called `keystone.conf`.

**Table 1.1. Services**

	Service	Coden ame	Description
1	Dashboard	horizon	A web-based dashboard for managing OpenStack services.
2	Identity	keystone	A centralized identity service that provides authentication and authorization for other services, and manages users, tenants, and roles.
3	OpenStack Networking	quantum	A networking service that provides connectivity between the interfaces of other OpenStack services.
4	Block Storage	cinder	A service that manages persistent block storage volumes for virtual machines.
5	Compute	nova	A service that launches and schedules networks of machines running on nodes.
6	Image	glance	A registry service for virtual machine images.
7	Object Storage	swift	A service providing object storage which allows users to store and retrieve files (arbitrary data).
8	Metering (Technical Preview)	ceilometer	A service providing measurements of cloud resources.
9	Orchestration (Technical Preview)	heat	A service providing a template-based orchestration engine, which supports the automatic creation of resource stacks.

The Service Details section provides more detailed information about the Openstack service components. Each OpenStack service is comprised of a collection of Linux services, MySQL databases, or other components, which together provide a functional group. For example, the glance-api and glance-registry Linux services, together with a MySQL database, implement the Image service.



### Important

For more information on the support scope for features marked as technology previews, see <https://access.redhat.com/support/offerings/techpreview/>

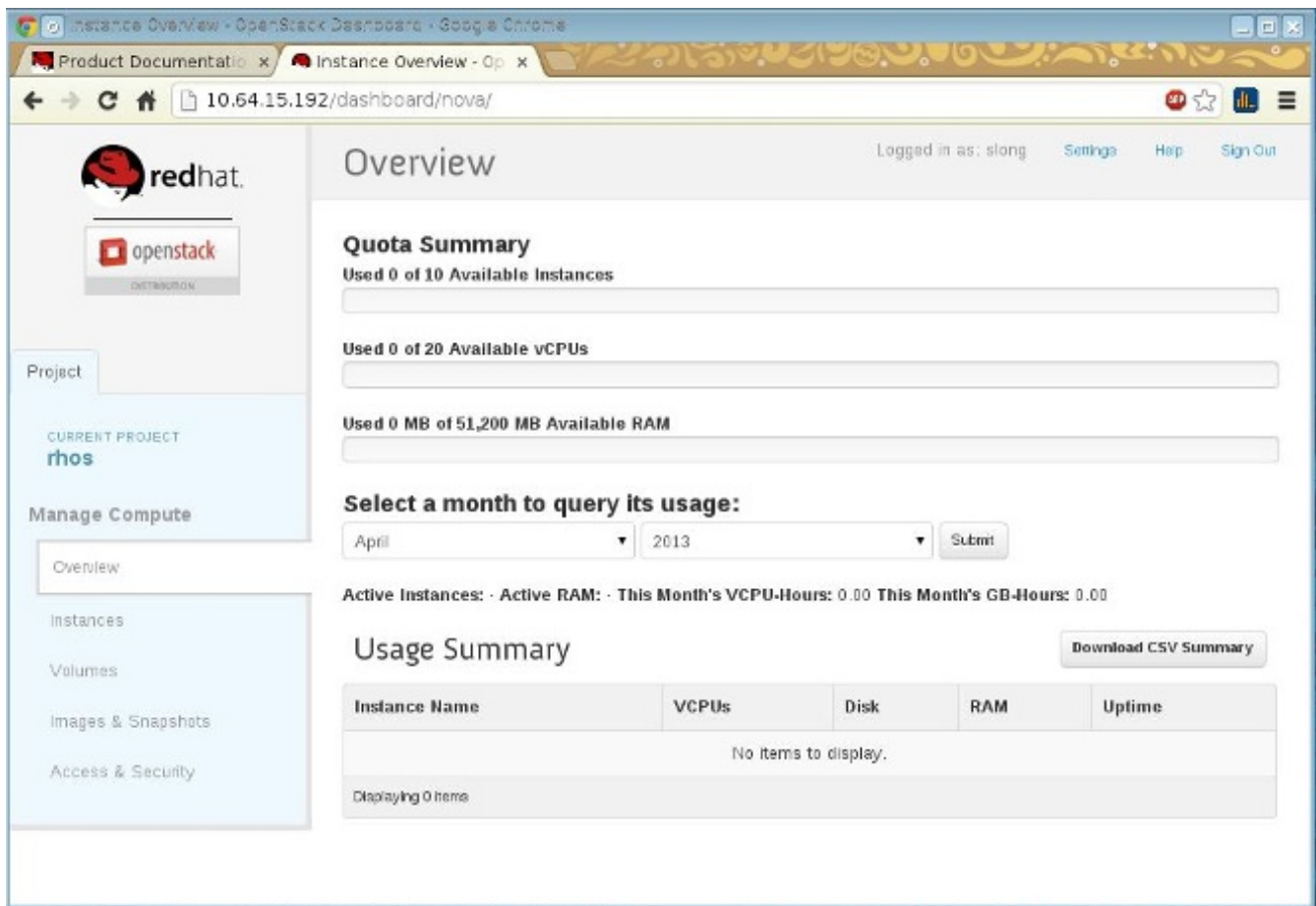
[Report a bug](#)

## 1.3. Service Details

### 1.3.1. Dashboard Service

The Dashboard service provides a graphical user interface for end users and administrators, allowing operations such as creating and 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 service provides three basic dashboards: user, system, and settings.

The following screenshot displays a user's dashboard after OpenStack is first installed:



The identity of the logged-in user determines the dashboards and panels that are visible in the dashboard.

**Table 1.2. Dashboard Service components**

Component	Description
openstack-dashboard	A Django (Python) web application, provides access to the dashboard using any web browser.
An Apache HTTP server (httpd service)	Hosts the application.
Database	For managing sessions.

### 1.3.2. Identity Service

[Report a bug](#)

The Identity service authenticates and authorizes OpenStack users (that is, keeps track of users and their permitted activities); the service is used by all OpenStack components. The service supports multiple forms of authentication including user name and password credentials, token-based systems, and AWS-style logins (Amazon Web Services).

The Identity service also provides a central catalog of services and endpoints running in a particular OpenStack cloud, which acts as a service directory for other OpenStack systems. Each endpoint is assigned:

- ▶ `adminURL`, the URL for the administrative endpoint for the service. Only the Identity service might use a value here that is different from `publicURL`; all other services will use the same value.
- ▶ `internalURL`, the URL of an internal-facing endpoint for the service (typically same as the `publicURL`).

- ▶ `publicURL`, the URL of the public-facing endpoint for the service.
- ▶ `region`, in which the service is located. By default, if a region is not specified, the 'RegionOne' location is used.

The Identity service uses the following concepts:

- ▶ Users, which have associated information (such as a name and password). In addition to custom users, a user must be defined for each cataloged service (for example, the 'glance' user for the Image service).
- ▶ Tenants, which are generally the user's group, project, or organization.
- ▶ Roles, which determine a user's permissions.

**Table 1.3. Identity Service components**

Component	Description
keystone	Provides the administrative and public APIs.
Databases	For each of the internal services.

### 1.3.3. OpenStack Networking Service

[Report a bug](#)

The OpenStack Networking service provides a scalable and API-driven system for managing the network connectivity, addressing, and services within an OpenStack IaaS cloud deployment. Because the OpenStack network is software-defined, it can easily and quickly react to changing network needs (for example, creating and assigning new IP addresses).

Advantages include:

- ▶ Users can create networks, control traffic, and connect servers and devices to one or more networks.
- ▶ OpenStack offers flexible networking models, so that administrators can change the networking model to adapt to their volume and tenancy.
- ▶ IPs can be dedicated or floating; floating IPs allow dynamic traffic rerouting.

**Table 1.4. Networking Service components**

Component	Description
quantum-server	A Python daemon, which manages user requests (and exposes the API). It is configured with a plugin that implements the OpenStack Networking API operations using a specific set of networking mechanisms. A wide choice of plugins are also available. For example, the <code>openvswitch</code> and <code>linuxbridge</code> plugins utilize native Linux networking mechanisms, while other plugins interface with external devices or SDN controllers.
quantum-l3-agent	An agent providing L3/NAT forwarding.
quantum-*-agent	A plug-in agent that runs on each node to perform local networking configuration for the node's VMs and networking services.
quantum-dhcp-agent	An agent providing DHCP services to tenant networks.
Database	Provides persistent storage.

### 1.3.4. Block Storage Service

[Report a bug](#)

The Block Storage (or volume) service provides persistent block storage management for virtual hard drives. The block storage system manages the creation of block devices to

servers. Block storage volumes are fully integrated into both the Compute and Dashboard services, which allows cloud users to manage their own storage needs (Compute handles the attaching and detaching of devices). Both regions and zones (for details, refer to the Object Storage section) can be used to handle distributed block storage hosts.

Block storage is appropriate for performance-sensitive scenarios such as database storage, expandable file systems, or providing a server with access to raw block-level storage. Additionally, snapshots can be taken to either restore data or to create new block storage volumes (snapshots are dependent upon driver support).

Basic operations include:

- ▶ Create, list, and delete volumes.
- ▶ Create, list, and delete snapshots.
- ▶ Attach and detach volumes to running virtual machines.

**Table 1.5. Block Storage Service components**

Component	Description
openstack-cinder-volume	Carves out storage for virtual machines on demand. A number of drivers are included for interaction with storage providers.
openstack-cinder-api	Responds to and handles requests, and places them in the message queue.
openstack-cinder-scheduler	Assigns tasks to the queue and determines the provisioning volume server.
Database	Provides state information.

**See Also:**

- ▶ [Section 1.3.5, “Compute Service”](#)

[Report a bug](#)

### 1.3.5. Compute Service

The Compute service is the heart of the OpenStack cloud by providing virtual machines on demand. Compute schedules virtual machines to run on a set of nodes by defining drivers that interact with underlying virtualization mechanisms, and exposing the functionality to the other OpenStack components.

Compute interacts with the Identity service for authentication, Image service for images, and the Dashboard service for the user and administrative interface. Access to images is limited by project and by user; quotas are limited per project (for example, the number of instances). The Compute service is designed to scale horizontally on standard hardware, and can download images to launch instances as required.

**Table 1.6. Ways to Segregate the Cloud**

Concept	Description
Regions	<p>Each service cataloged in the Identity service is identified by its region, which typically represents a geographical location, and its endpoint. In a cloud with multiple Compute deployments, regions allow for the discrete separation of services, and are a robust way to share some infrastructure between Compute installations, while allowing for a high degree of failure tolerance.</p>
Cells (Technology Preview)	<p>A cloud's Compute hosts can be partitioned into groups called cells (to handle large deployments or geographically separate installations). Cells are configured in a tree. The top-level cell ('API cell') runs the nova-api service, but no nova-compute services. In contrast, each child cell runs all of the other typical nova-* services found in a regular installation, except for the nova-api service. Each cell has its own message queue and database service, and also runs nova-cells, which manages the communication between the API cell and its child cells.</p> <p>This means that:</p> <ul style="list-style-type: none"> <li>▶ A single API server can be used to control access to multiple Compute installations.</li> <li>▶ A second level of scheduling at the cell level is available (versus host scheduling), which provides greater flexibility over the control of where virtual machines are run.</li> </ul>
Host Aggregates and Availability Zones	<p>A single Compute deployment can be partitioned into logical groups (for example, into multiple groups of hosts that share common resources like storage and network, or which have a special property such as trusted computing hardware).</p> <p>If the user is:</p> <ul style="list-style-type: none"> <li>▶ An administrator, the group is presented as a Host Aggregate, which has assigned Compute hosts and associated metadata. An aggregate's metadata is commonly used to provide information for use with nova-scheduler (for example, limiting specific flavors or images to a subset of hosts).</li> <li>▶ A user, the group is presented as an Availability Zone. The user cannot view the group's metadata, nor which hosts make up the zone.</li> </ul> <p>Aggregates, or zones, can be used to:</p> <ul style="list-style-type: none"> <li>▶ Handle load balancing and instance distribution.</li> <li>▶ Provide some form of physical isolation and redundancy from other zones (such as by using a separate power supply or network equipment).</li> <li>▶ Identify a set of servers that have some common attribute.</li> <li>▶ Separate out different classes of hardware.</li> </ul>



### Important

For more information on the support scope for features marked as technology previews, refer to <https://access.redhat.com/support/offerings/techpreview/>

**Table 1.7. Compute Service components**

Component	Description
openstack-nova-api	Handles requests and provides access to the Compute services (such as booting an instance).
openstack-nova-cert	Provides the certificate manager.
openstack-nova-compute	Creates and terminates virtual instances. Interacts with the Hypervisor to bring up new instances, and ensures that the state is maintained in the Compute database.
openstack-nova-conductor	Provides database-access support for Compute nodes (thereby reducing security risks).
openstack-nova-consoleauth	Handles console authentication.
openstack-nova-network	Handles Compute network traffic (both private and public access). Handles such tasks as assigning an IP address to a new virtual instance, and implementing security group rules.
openstack-nova-novncproxy	Provides a VNC proxy for browsers (enabling VNC consoles to access virtual machines).
openstack-nova-scheduler	Dispatches requests for new virtual machines to the correct node.
Apache Qpid server (qpidd)	Provides the AMPQ message queue. This server (also used by Block Storage) handles the OpenStack transaction management, including queuing, distribution, security, management, clustering, and federation. Messaging becomes especially important when an OpenStack deployment is scaled and its services are running on multiple machines.
libvirt d	The driver for the hypervisor. Enables the creation of virtual machines.
KVM Linux hypervisor	Creates virtual machines and enables their live migration from node to node.
Database	Provides build-time and run-time infrastructure state.

### 1.3.6. Image Service

[Report a bug](#)

The Image service acts as a registry for virtual disk images. Users can add new images or take a snapshot (copy) of an existing server for immediate storage. Snapshots can be used as back up or as templates for new servers. Registered images can be stored in the Object Storage service, as well as in other locations (for example, in simple file systems or external web servers).

The following image formats are supported:

- ▶ raw (unstructured format)
- ▶ aki/ami/ari (Amazon kernel, ramdisk, or machine image)
- ▶ iso (archive format for optical discs; for example, CDROM)
- ▶ qcow2 (Qemu/KVM, supports *Copy on Write*)
- ▶ vhd (Hyper-V, common for virtual machine monitors from VMWare, Xen, Microsoft, VirtualBox, and others)
- ▶ vdi (Qemu/VirtualBox)
- ▶ vmdk (VMWare)

Container formats can also be used by the Image service; the format determines the type of metadata stored in the image about the actual virtual machine. The following formats are

supported.

- ▶ bare (no metadata is included)
- ▶ ovf (OVF format)
- ▶ aki/ami/ari (Amazon kernel, ramdisk, or machine image)

**Table 1.8. Image Service components**

Component	Description
openstack-glance-api	Handles requests and image delivery (interacts with storage back-ends for retrieval and storage). Uses the registry to retrieve image information (the registry service is never, and should never be, accessed directly).
openstack-glance-registry	Manages all metadata associated with each image. Requires a database.
Database	Stores image metadata.

### 1.3.7. Object Storage Service

[Report a bug](#)

The Object Storage service provides object storage in virtual containers, which allows users to store and retrieve files. The service's distributed architecture supports horizontal scaling; redundancy as failure-proofing is provided through software-based data replication.

Because it supports asynchronous eventual consistency replication, it is well suited to multiple data-center deployment. Object Storage uses the concept of:

- ▶ Storage replicas, which are used to maintain the state of objects in the case of outage. A minimum of three replicas is recommended.
- ▶ Storage zones, which are used to host replicas. Zones ensure that each replica of a given object can be 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.
- ▶ Storage regions, which are essentially a group of zones sharing a location. Regions can be, for example, groups of servers or server farms, usually located in the same geographical area. Regions have a separate API endpoint per Object Storage service installation, which allows for a discrete separation of services.



**Table 1.9. Object Storage Service components**

Component	Description
openstack-swift-proxy	Exposes the public API, and is responsible for handling requests and routing them accordingly. Objects are streamed through the proxy server to the user (not spooled). Objects can also be served out via HTTP.
openstack-swift-object	Stores, retrieves, and deletes objects.
openstack-swift-account	Responsible for listings of containers, using the account database.
openstack-swift-container	Handles listings of objects (what objects are in a specific container), using the container database.
Ring files	Contain details of all the storage devices, and are used to deduce where a particular piece of data is stored (maps the names of stored entities to their physical location). One file is created for each object, account, and container server.
Account database	
Container database	
ext4 (recommended) or XFS file system	Used for object storage.
Housekeeping processes	Replication and auditors.

### 1.3.8. Metering (Technical Preview)

[Report a bug](#)

The Metering service provides user-level usage data for OpenStack-based clouds, which can be used for customer billing, system monitoring, or alerts. Data can be collected by notifications sent by existing OpenStack components (for example, usage events emitted from Compute) or by polling the infrastructure (for example, libvirt).

Metering includes a storage daemon that communicates with authenticated agents via a trusted messaging system, to collect and aggregate data. Additionally, the service uses a plugin system, which makes it easy to add new monitors.

**Table 1.10. Metering Service components**

Component	Description
ceilometer-agent-compute	An agent that runs on each Compute node to poll for resource utilization statistics.
ceilometer-agent-central	An agent that runs on a central management server to poll for utilization statistics about resources not tied to instances or Compute nodes.
ceilometer-collector	An agent that runs on one or more central management servers to monitor the message queues. Notification messages are processed and turned into metering messages, and sent back out on to the message bus using the appropriate topic. Metering messages are written to the data store without modification.
Mongo database	For collected usage sample data.
API Server	Runs on one or more central management servers to provide access to the data store's data. Only the Collector and the API server have access to the data store.

### 1.3.9. Orchestration (Technical Preview)

The Orchestration service provides a template-based orchestration engine for the OpenStack cloud, which can be used to create and manage cloud infrastructure resources such as storage, networking, instances, and applications as a repeatable running environment.

Templates are used to create stacks, which are collections of resources (for example instances, floating IPs, volumes, security groups, or users). The service offers access to all OpenStack core services via a single modular template, with additional orchestration capabilities such as auto-scaling and basic high availability.

Features include:

- ▶ A single template provides access to all underlying service APIs.
- ▶ Templates are modular (resource oriented).
- ▶ Templates can be recursively defined, and therefore reusable (nested stacks). This means that the cloud infrastructure can be defined and reused in a modular way.
- ▶ Resource implementation is pluggable, which allows for custom resources.
- ▶ Autoscaling functionality (automatically adding or removing resources depending upon usage).
- ▶ Basic high availability functionality.

**Table 1.11. Orchestration Service components**

Component	Description
heat	A CLI tool that communicates with the heat-api to execute AWS CloudFormation APIs.
heat-api	An OpenStack-native REST API that processes API requests by sending them to the heat-engine over RPC.
heat-api-cfn	Provides an AWS-Query API that is compatible with AWS CloudFormation and processes API requests by sending them to the heat-engine over RPC.
heat-engine	Orchestrates the launching of templates and provide events back to the API consumer.
heat-api-cloudwatch	Provides monitoring (metrics collection) for the Orchestration service.
heat-cfnutils	A package of helper scripts (for example, cfn-hup, which handles updates to metadata and executes custom hooks).



#### Note

The `heat-cfnutils` package is only installed on images that are launched by heat into Compute servers.

# Chapter 2. Prerequisites

## 2.1. Software Requirements

### 2.1.1. Operating System Requirements

Red Hat Enterprise Linux OpenStack Platform requires Red Hat Enterprise Linux 6.4 Server. All systems in the environment must have Red Hat Enterprise Linux 6.4 Server installed and be subscribed to receive package updates from Red Hat Network or an equivalent source such as a Red Hat Network Satellite server.

For further information on installing Red Hat Enterprise Linux 6.4 Server refer to the Red Hat Enterprise Linux 6 *Installation Guide*.

#### See Also:

- ▶ [Section 2.1.2.1, “Register to Red Hat Network”](#)
- ▶ [Section 2.1.2.2, “Red Hat Enterprise Linux Repository Configuration”](#)
- ▶ [Section 2.1.2.3, “Red Hat Enterprise Linux OpenStack Platform Repository Configuration”](#)

### 2.1.2. Software Repository Configuration

#### 2.1.2.1. Register to Red Hat Network

Red Hat Enterprise Linux OpenStack Platform requires that each system in the OpenStack environment be running Red Hat Enterprise Linux Server and that all systems be signed up to receive updates from Red Hat Network using Subscription Manager. For further information on managing Red Hat subscriptions refer to the Red Hat *Subscription Management Guide*.

All steps in this procedure must be executed while logged in to the account of the root user on the system being registered.



#### 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.

1. Run the `subscription-manager register` command to register the system to Red Hat Network.

```
# subscription-manager register
```

2. Enter your Red Hat Network user name when prompted.

```
Username: admin@example.com
```



## Important

Your Red Hat Network account must have Red Hat Enterprise Linux OpenStack Platform entitlements. If your Red Hat Network account does not have Red Hat Enterprise Linux OpenStack entitlements then you may register for access to the evaluation program at <http://www.redhat.com/openstack/>.

3. Enter your Red Hat Network password when prompted.

Password:

4. When registration completes successfully system is assigned a unique identifier.

The system has been registered with id: *IDENTIFIER*

The system has been registered to Red Hat Network and is ready to be attached to specific software subscriptions.

### 2.1.2.2. Red Hat Enterprise Linux Repository Configuration

[Report a bug](#)

Follow the steps in this procedure to register a Red Hat Enterprise Linux system to receive updates from Red Hat Network. These steps must be run while logged in as the root user. Repeat these steps on each system in the OpenStack environment.

1. Use the `subscription-manager list` command to locate the pool identifier of the Red Hat Enterprise Linux subscription.

```
# subscription-manager list --available
+-----+
| Available Subscriptions |
+-----+
Product Name:      Red Hat Enterprise Linux Server
Product Id:        69
Pool Id:           POOLID
Quantity:          1
Service Level:     None
Service Type:      None
Multi-Entitlement: No
Expires:           01/01/2022
Machine Type:      physical
...
```

The pool identifier is indicated in the `Pool Id` field associated with the Red Hat Enterprise Linux Server product. The identifier will be unique to your subscription. Take note of this identifier as it will be required to perform the next step.



## Note

The output displayed in this step has been truncated to conserve space. All other available subscriptions will also be listed in the output of the command.

2. Use the `subscription-manager attach` command to attach the subscription identified in the previous step.

```
# subscription-manager attach --pool=POOLID
Successfully attached a subscription for Red Hat Enterprise Linux Server.
```

Replace *POOLID* with the unique identifier associated with your Red Hat Enterprise Linux Server subscription. This is the identifier that was located in the previous step.

3. Run the `yum repolist` command. This command ensures that the repository configuration file `/etc/yum.repos.d/redhat.repo` exists and is up to date.

```
# 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		



### Note

The output displayed in this step may differ from that which appears when you run the `yum repolist` command on your system. In particular the number of packages listed will vary if or when additional packages are added to the `rhel-6-server-rpms` repository.

You have successfully configured your system to receive Red Hat Enterprise Linux updates from Red Hat Network.

### 2.1.2.3. Red Hat Enterprise Linux OpenStack Platform Repository Configuration

[Report a bug](#)

Follow the steps in this procedure to configure a Red Hat Enterprise Linux system to receive OpenStack packages and updates from Red Hat Network. Access to a Red Hat software entitlement that includes Red Hat Enterprise Linux OpenStack Platform is required, such entitlements include:

- ▶ Red Hat Cloud Infrastructure
- ▶ Red Hat Cloud Infrastructure (without Guest OS)
- ▶ Red Hat Enterprise Linux OpenStack Platform
- ▶ Red Hat Enterprise Linux OpenStack Platform Preview
- ▶ Red Hat Enterprise Linux OpenStack Platform (without Guest OS)

These steps must be run while logged in as the root user. Repeat these steps on each system in the environment.

1. Use the `subscription-manager list` command to locate the pool identifier of the relevant Red Hat Cloud Infrastructure or Red Hat Enterprise Linux OpenStack Platform entitlement.

```
# subscription-manager list --available
+-----+
| Available Subscriptions |
+-----+
...
Product Name:      ENTITLEMENT
Product Id:        ID_1
Pool Id:           POOLID_1
Quantity:          3
Service Level:     None
Service Type:      None
Multi-Entitlement: No
Expires:           02/14/2013
Machine Type:      physical

Product Name:      ENTITLEMENT
Product Id:        ID_2
Pool Id:           POOLID_2
Quantity:          unlimited
Service Level:     None
Service Type:      None
Multi-Entitlement: No
Expires:           02/14/2013
Machine Type:      virtual
...
```

Locate the entry in the list where the Product Name matches the name of the entitlement that will be used to access Red Hat Enterprise Linux OpenStack Platform packages. Take note of the pool identifier associated with the entitlement, this value is indicated in the Pool Id field. The pool identifier is unique to your subscription and will be required to complete the next step.



## Note

The output displayed in this step has been truncated to conserve space. All other available subscriptions will also be listed in the output of the command.

2. Use the `subscription-manager attach` command to attach the subscription identified in the previous step.

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

Replace *POOLID* with the unique identifier associated with your Red Hat Cloud Infrastructure or Red Hat Enterprise Linux OpenStack Platform entitlement. This is the identifier that was located in the previous step.

3. Install the `yum-utils` package. The `yum-utils` package is provided by the Red Hat Enterprise Linux subscription but provides the `yum-config-manager` utility required to complete configuration of the Red Hat Enterprise Linux OpenStack Platform software repositories.

```
# 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.

4. 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 Red Hat OpenStack 1.0 (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/opensta
ck/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 Red Hat OpenStack 2.1 (Folsom) is disabled.

```
# yum-config-manager --disable 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/beta/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 = False
...
```

- c. Ensure that the repository for Red Hat Enterprise Linux OpenStack Platform 3 (Grizzly) has been enabled.

```
# yum-config-manager --enable rhel-server-ost-6-3-rpms
Loaded plugins: product-id
==== repo: rhel-server-ost-6-3-rpms ====
[rhel-server-ost-6-3-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/3/os
cache = 0
cachedir = /var/cache/yum/x86_64/6Server/rhel-server-ost-6-3-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
```

- Run the `yum repolist` command. This command ensures that the repository configuration file `/etc/yum.repos.d/redhat.repo` exists and is up to date.

```
# 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
rhel-server-ost-6-3-rpms	Red Hat OpenStack 3 (RPMs)	138
repolist: 10,058		



## Note

The output displayed in this step may differ from that which appears when you run the `yum repolist` command on your system. In particular the number of packages listed will vary if or when additional packages are added to the repositories.

- 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
```

- Use the `yum-config-manager` command to set the priority of the Red Hat Enterprise Linux OpenStack Platform 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-6-3-rpms \
  --setopt="rhel-server-ost-6-3-rpms.priority=1"
Loaded plugins: product-id
==== repo: rhel-server-ost-6-3-rpms ====
[rhel-server-ost-6-3-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/3/
os
cache = 0
cachedir = /var/cache/yum/x86_64/6Server/rhel-server-ost-6-3-rpms
cost = 1000
enabled = True
...
priority = 1
...
```

8. Run the `yum update` command and reboot to ensure that the most up to date packages, including the kernel, are installed and running.

```
# yum update -y
```

```
# reboot
```

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

[Report a bug](#)

## 2.2. Hardware Requirements

### 2.2.1. Compute Node Requirements

Compute nodes are responsible for running virtual machine instances once they have been launched. Compute nodes must have hardware virtualization support. In addition they must have enough available RAM and disk space to support the requirements of the virtual machine instances that they will host.

#### 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.

## 2.2.2. Network Node Requirements

[Report a bug](#)

Network nodes are responsible for hosting the services that provided networking functionality to compute instances. In particular they host the DHCP agent, layer 3 agent, and metadata proxy services. Like all systems that handle networking in an OpenStack environment they also host an instance of the layer 2 agent.

The hardware requirements of network nodes vary widely depending on the networking workload of the environment. The requirements listed here are intended as a guide to the minimum requirements of a network node.

### Processor

No specific CPU requirements are imposed by the networking services.

### Memory

A minimum of 2 GB of RAM is recommended.

### Disk Space

A minimum of 10 GB of available disk space is recommended.

No additional disk space is required by the networking services other than that required to install the packages themselves. Some disk space however must be available for log and temporary files.

### Network Interface Cards

2 x 1 Gbps Network Interface Cards.

## 2.2.3. Block Storage Node Requirements

[Report a bug](#)

Block storage nodes are those nodes that will host the volume service (`openstack-cinder-volume`) and provide volumes for use by virtual machine instances or other cloud users. The block storage API (`openstack-cinder-api`) and scheduling services (`openstack-cinder-scheduler`) may run on the same nodes as the volume service or separately. In either case the primary hardware requirement of the block storage nodes is that there is enough block storage available to serve the needs of the OpenStack environment.

The amount of block storage required in an OpenStack environment varies based on:

- ▶ The number of volumes that will be created in the environment.
- ▶ The average size of the volumes that will be created in the environment.
- ▶ Whether or not the storage backend will be configured to support redundancy.
- ▶ Whether or not the storage backend will be configured to create sparse volumes by default.

Use this formula to assist with estimating the initial block storage needs of an OpenStack environment.

$$VOLUMES * SIZE * REDUNDANCY * UTILIZATION = TOTAL$$

- ▶ Replace *VOLUMES* with the number of volumes that it is expected will exist in the environment at any one time.
- ▶ Replace *SIZE* with the expected average size of the volumes that will exist in the environment at any one time.
- ▶ Replace *REDUNDANCY* with the expected number of redundant copies of each volume the backend storage will be configured to keep. Use 1 or skip this multiplication operation if no redundancy will be used.
- ▶ Replace *UTILIZATION* with the expected percentage of each volume that will actually be used. Use 1, indicating 100%, if the use of sparse volumes will not be enabled.

The resultant figure represents an **estimate** of the block storage needs of your OpenStack environment in gigabytes. It is recommended that some additional space is allowed for future growth. Addition of further block storage once the environment has been deployed is facilitated by adding more block storage providers and, if necessary, additional instances of the volume service.

## Part II. Installing OpenStack

[Report a bug](#)

# Chapter 3. Installing the Database Server

## 3.1. Installing the Packages

The steps listed in this procedure install the packages required by the MySQL database server. The packages that will be installed are:

### ***mysql-server***

Provides the MySQL database server.

### ***mysql***

Provides the MySQL client tools and libraries. Installed as a dependency of the *mysql-server* package.

All steps listed in this procedure must be performed while logged in as the root user.

- ▶ Install the required packages using the yum command:

```
# yum install -y mysql-server
```

The MySQL database server is installed and ready to be configured.

[Report a bug](#)

## 3.2. Configuring the Firewall

As the database service is used by all of the components in the OpenStack environment it must be accessible by them.

To allow this the firewall on the system hosting the database service must be altered to allow network traffic on the required port. All steps in this procedure must be run while logged in to the server hosting the database service as the root user.

1. Open the `/etc/sysconfig/iptables` file in a text editor.
2. Add an INPUT rule allowing TCP traffic on port 3306 to the file. The new rule must appear before any INPUT rules that REJECT traffic.

```
-A INPUT -p tcp -m multiport --dports 3306 -j ACCEPT
```

3. Save the changes to the `/etc/sysconfig/iptables` file.
4. Restart the iptables service to ensure that the change takes effect.

```
# service iptables restart
```

The iptables firewall is now configured to allow incoming connections to the MySQL database service on port 3306.

[Report a bug](#)

## 3.3. Starting the Database Service

All steps in this procedure must be performed while logged in to the server hosting the database service as the root user.

1. Use the service command to start the mysqld service.

```
# service mysqld start
```

2. Use the chkconfig command to ensure that the mysqld service will be started automatically in the future.

```
# chkconfig mysqld on
```

The mysqld service has been started.

[Report a bug](#)

## 3.4. Setting the Database Administrator Password

The MySQL database server maintains its own list of user accounts and authentication details including a root user account. This account acts as the database administrator account.

For security reasons it is recommended that you set a password for the root database user once the database service has been started for the first time.

All steps in this procedure must be run while logged in to the system hosting the MySQL database as the root user.

1. Use the mysqladmin command to set the password for the root database user.

```
# /usr/bin/mysqladmin -u root password "PASSWORD"
```

Replace *PASSWORD* with the intended password.

2. The mysqladmin command can also be used to change the password of the root database user if required.

```
# /usr/bin/mysqladmin -u root -p OLDPASS NEWPASS
```

Replace *OLDPASS* with the existing password and *NEWPASS* with the password that is intended to replace it.

The MySQL administrator, or root, password has been set. This password will be required when logging in to create databases and database users.

# Chapter 4. Installing the Message Broker

## 4.1. Installing the Packages

The steps listed in this procedure install the packages required by the Qpid message broker. The packages that will be installed are:

### ***qpid-cpp-server***

Provides the Qpid message broker.

### ***qpid-cpp-server-ssl***

Provides the Qpid plug-in enabling support for SSL as a transport later for AMQP traffic. This package is optional but recommended to support secure configuration of Qpid.

All steps listed in this procedure must be performed while logged in as the root user.

- Install the required packages using the yum command:

```
# yum install -y qpid-cpp-server qpid-cpp-server-ssl
```

The Qpid message broker is installed and ready to be configured.

## 4.2. Configuring the Message Broker

### 4.2.1. Simple Authentication and Security Layer - SASL

#### 4.2.1.1. SASL - Simple Authentication and Security Layer

Qpid is optionally able to use Simple Authentication and Security Layer (SASL) for identifying and authorizing incoming connections to the broker, as mandated in the AMQP specification. SASL provides a variety of authentication methods. Clients can negotiate with the Messaging Broker to find a SASL mechanism that both can use.

#### 4.2.1.2. SASL Mechanisms

The SASL authentication mechanisms allowed by the broker are controlled by the file `/etc/sasl2/qpid.conf` on the broker. To narrow the allowed mechanisms to a smaller subset, edit this file and remove mechanisms.



### Important

The PLAIN authentication mechanism sends passwords in cleartext. If using this mechanism, for complete security using Security Services Library (SSL) is recommended.

### SASL Mechanisms

#### **ANONYMOUS**

Clients are able to connect anonymously.

Note that when the broker is started with `auth=no`, authentication is disabled. PLAIN and ANONYMOUS authentication mechanisms are available as *identification mechanisms*, but they have no authentication value.

### PLAIN

Passwords are passed in plain text between the client and the broker. This is not a secure mechanism, and should be used in development environments only. If PLAIN is used in production, it should only be used over SSL connections, where the SSL encryption of the transport protects the password.

Note that when the broker is started with `auth=no`, authentication is disabled. The PLAIN and ANONYMOUS authentication mechanisms are available as *identification mechanisms*, but they have no authentication value.

### DIGEST-MD5

MD5 hashed passwords are exchanged using HTTP headers. This is a medium strength security protocol.

#### 4.2.1.3. SASL Mechanisms and Packages

[Report a bug](#)

The following table lists the `cyrus-sasl-*` package(s) that need to be installed on the server for each authentication mechanism to be available.

**Table 4.1.**

Method	Package	/etc/sasl2/qpidd.conf entry
ANONYMOUS	-	-
PLAIN	<code>cyrus-sasl-plain</code>	<code>mech_list: PLAIN</code>
DIGEST-MD5	<code>cyrus-sasl-md5</code>	<code>mech_list: DIGEST-MD5</code>

[Report a bug](#)

#### 4.2.1.4. Configure SASL using a Local Password File

The local SASL database is used by the PLAIN and DIGEST-MD5 authentication mechanisms.

##### Procedure 4.1. Configure SASL using a Local Password File

To use the default SASL PLAIN authentication mechanism implemented by Qpid, either use the default username and password of *guest*, which are included in the database at `/var/lib/qpidd/qpidd.sasl` on installation, or add your own accounts.

1. Add new users to the database by using the `saslpasswd2` command. The User ID for authentication and ACL authorization uses the form `user-id@domain`.

Ensure that the correct realm has been set for the broker. This can be done by editing the configuration file or using the `-u` option. The default realm for the broker is *QPID*.

```
# saslpasswd2 -f /var/lib/qpidd/qpidd.sasl -u QPID new_user_name
```

2. Existing user accounts can be listed by using the `-f` option:

```
# sasldblistusers2 -f /var/lib/qpidd/qpidd.sasl
```



**Note**

The user database at `/var/lib/qpidd/qpidd.sasl` is readable only by the `qpidd` user. If you start the broker from a user other than the `qpidd` user, you will need to either modify the configuration file, or turn authentication off. Note also that this file must be readable by the `qpidd` user. If you delete and recreate this file, make sure the `qpidd` user has read permissions, or authentication attempts will fail.

- To switch authentication on or off, add the appropriate line to the `/etc/qpidd.conf` configuration file:

```
auth=no
auth=yes
```

The SASL configuration file is in `/etc/sasl2/qpidd.conf` for Red Hat Enterprise Linux.

## 4.2.2. Configuring TLS/SSL

[Report a bug](#)

### 4.2.2.1. Encryption Using SSL

Encryption and certificate management for `qpidd` is provided by Mozilla's Network Security Services Library (NSS).

### 4.2.2.2. Enable SSL on the Broker

[Report a bug](#)

- You will need a certificate that has been signed by a Certification Authority (CA). This certificate will also need to be trusted by your client. If you require client authentication in addition to server authentication, the client certificate will also need to be signed by a CA and trusted by the broker.

In the broker, SSL is provided through the `ssl` module. This module is installed and loaded by default in MRG Messaging. To enable the module, you need to specify the location of the database containing the certificate and key to use. This is done using the `ssl-cert-db` option.

The certificate database is created and managed by the Mozilla Network Security Services (NSS) `certutil` tool. Information on this utility can be found on the [Mozilla website](#), including tutorials on setting up and testing SSL connections. The certificate database will generally be password protected. The safest way to specify the password is to place it in a protected file, use the password file when creating the database, and specify the password file with the `ssl-cert-password-file` option when starting the broker.

The following script shows how to create a certificate database using `certutil`:

```
mkdir ${CERT_DIR}
certutil -N -d ${CERT_DIR} -f ${CERT_PW_FILE}
certutil -S -d ${CERT_DIR} -n ${NICKNAME} -s "CN=${NICKNAME}" -t "CT,," -x -
f ${CERT_PW_FILE} -z /usr/bin/certutil
```

When starting the broker, set `ssl-cert-password-file` to the value of `${CERT_PW_FILE}`, set `ssl-cert-db` to the value of `${CERT_DIR}`, and set `ssl-cert-name` to the value of `${NICKNAME}`.

- The following SSL options can be used when starting the broker:
  - `--ssl-use-export-policy`  
Use NSS export policy

`--ssl-cert-password-file PATH`

Required. Plain-text file containing password to use for accessing certificate database.

`--ssl-cert-db PATH`

Required. Path to directory containing certificate database.

`--ssl-cert-name NAME`

Name of the certificate to use. Default is `localhost.localdomain`.

`--ssl-port NUMBER`

Port on which to listen for SSL connections. If no port is specified, port 5671 is used. If the SSL port chosen is the same as the port for non-SSL connections (i.e. if the `--ssl-port` and `--port` options are the same), both SSL encrypted and unencrypted connections can be established to the same port. However in this configuration there is no support for IPv6.

`--ssl-require-client-authentication`

Require SSL client authentication (i.e. verification of a client certificate) during the SSL handshake. This occurs before SASL authentication, and is independent of SASL.

This option enables the EXTERNAL SASL mechanism for SSL connections. If the client chooses the EXTERNAL mechanism, the client's identity is taken from the validated SSL certificate, using the CN, and appending any DC's to create the domain. For instance, if the certificate contains the properties `CN=bob, DC=acme, DC=com`, the client's identity is `bob@acme.com`.

If the client chooses a different SASL mechanism, the identity taken from the client certificate will be replaced by that negotiated during the SASL handshake.

`--ssl-sasl-no-dict`

Do not accept SASL mechanisms that can be compromised by dictionary attacks. This prevents a weaker mechanism being selected instead of EXTERNAL, which is not vulnerable to dictionary attacks.

`--require-encryption`

This will cause `qpidd` to only accept encrypted connections. This means only clients with EXTERNAL SASL on the SSL-port, or with GSSAPI on the TCP port.

#### 4.2.2.3. Export an SSL Certificate for Clients

[Report a bug](#)

When SSL is enabled on a server, the clients require a copy of the SSL certificate to establish a secure connection.

The following example commands can be used to export a client certificate and the private key from the broker's NSS database:

```
pk12util -o <p12exportfile> -n <certname> -d <certdir> -w <p12filepwfile>
openssl pkcs12 -in <p12exportfile> -out <clcertname> -nodes -clcerts -passin
pass:<p12pw>
```

For more information on SSL commands and options, refer to the [OpenSSL Documentation](#). On Red Hat Enterprise Linux type: `man openssl`.

[Report a bug](#)

## 4.3. Firewall Configuration

You must allow incoming connections on the port used by the Messaging System. The default

port for AMQP traffic is 5672.

On a Red Hat Enterprise Linux system, the firewall is provided by `iptables`. Commands modifying the firewall configuration must be run with root privileges. The following sequence of commands will configure `iptables` to allow incoming connections on port 5672.

```
iptables -I INPUT -p tcp -m tcp --dport 5672 -j ACCEPT
service iptables save
service iptables restart
```

To view the currently configured firewall rules, issue the command:

```
service iptables status
```

[Report a bug](#)

## 4.4. Starting the Messaging Server

The `qpidd` service must be started before the broker can commence sending and receiving messages.

1. Use the `service` command to start the service.

```
# service qpidd start
```

2. Use the `chkconfig` command to enable the service permanently.

```
# chkconfig qpidd on
```

The `qpidd` service has been started.

# Chapter 5. Installing the OpenStack Identity Service

## 5.1. Identity Service Requirements

The system hosting the identity service must have:

- ▶ Access to Red Hat Network or equivalent service provided by a tool such as Satellite.
- ▶ A network interface that is addressable by all other systems that will host OpenStack services.
- ▶ Network access to the database server.
- ▶ Network access to the directory server if using an LDAP backend.

Ensure that these requirements are met before proceeding with installation and configuration of the identity service.

## 5.2. Installing the Packages

The steps listed in this procedure install the packages required by the OpenStack Identity service. The packages that will be installed are:

### ***openstack-keystone***

Provides the OpenStack Identity service.

### ***openstack-utils***

Provides supporting utilities to assist with a number of tasks including the editing of configuration files.

### ***openstack-selinux***

Provides OpenStack specific SELinux policy modules.

All steps listed in this procedure must be performed while logged in as the root user.

- ▶ Install the required packages using the yum command:

```
# yum install -y openstack-keystone \  
openstack-utils \  
openstack-selinux
```

The OpenStack Identity service is installed and ready to be configured.

## 5.3. Creating the Identity Database

In this procedure the database and database user that will be used by the identity service will be created. These steps must be performed while logged in to the database server as the root user (or at least as a user with the correct permissions: create db, create user, grant

permissions).

1. Connect to the database service using the `mysql` command.

```
# mysql -u root -p
```

2. Create the keystone database.

```
mysql> CREATE DATABASE keystone;
```

3. Create a keystone database user and grant it access to the keystone database.

```
mysql> GRANT ALL ON keystone.* TO 'keystone'@'%' IDENTIFIED BY 'PASSWORD';
```

```
mysql> GRANT ALL ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY 'PASSWORD';
```

Replace `PASSWORD` with a secure password that will be used to authenticate with the database server as this user.

4. Flush the database privileges to ensure that they take effect immediately.

```
mysql> FLUSH PRIVILEGES;
```

5. Exit the `mysql` client command.

```
mysql> quit
```

The database has been created. The database will be populated during service configuration.

[Report a bug](#)

## 5.4. Configuring the Service

### 5.4.1. Setting the Administration Token

Before the identity service is started for the first time you must define an administrative token in an environment variable. This value will be used to authenticate before user and service accounts have been defined using the identity service.

All steps listed in this procedure must be performed while logged in to the server that will host the identity service as the root user.

1. Use OpenSSL to generate an initial service token and save it in the `SERVICE_TOKEN` environment variable.

```
# export SERVICE_TOKEN=$(openssl rand -hex 10)
```

2. Store the value of the administration token in a file for future use.

```
# echo $SERVICE_TOKEN > ~/ks_admin_token
```

3. Use the `openstack-config` tool to set the value of the `admin_token` configuration key to that of the newly created token.

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

The administration token for the identity service has been created. This value will be used in subsequent identity configuration procedures.

## 5.4.2. Setting the Database Connection String

The database connection string used by the identity service is defined in the `/etc/keystone/keystone.conf` file. It must be updated to point to a valid database server before starting the service.

All commands in this procedure must be run while logged in as the root user on the server hosting the identity service.

- ▶ Use the `openstack-config` command to set the value of the connection configuration key.

```
# openstack-config --set /etc/keystone/keystone.conf \
  sql connection mysql://USER:PASS@IP/DB
```

Replace:

- *USER* with the database user name the identity service is to use, usually keystone.
- *PASS* with the password of the chosen database user.
- *IP* with the IP address or host name of the database server.
- *DB* with the name of the database that has been created for use by the identity service, usually keystone.

The database connection string has been set and will be used by the identity service.

## 5.4.3. Configuring the Public Key Infrastructure

### 5.4.3.1. Public Key Infrastructure Overview

The identity service generates tokens which are cryptographically signed documents users and other services use for authentication. The tokens are signed using a private key while the public key is made available in an X509 certificate.

The certificates and relevant configuration keys are automatically generated by the `keystone-manage pki_setup` command. It is however possible to manually create and sign the required certificates using a third party certificate authority. If using third party certificates the identity service configuration must be manually updated to point to the certificates and supporting files.

The configuration keys relevant to PKI setup appear in the `[signing]` section of the `/etc/keystone/keystone.conf` configuration file that are relevant to the PKI setup are:

#### **ca\_certs**

Specifies the location of the certificate for the authority that issued the certificate denoted by the `certfile` configuration key. The default value is `/etc/keystone/ssl/certs/ca.pem`.

#### **ca\_key**

Specifies the key of the certificate authority that issued the certificate denoted by the `certfile` configuration key. The default value is `/etc/keystone/ssl/certs/cakey.pem`.

#### **ca\_password**

Specifies the password, if applicable, required to open the certificate authority file. The default action if no value is specified is not to use a password.

#### **certfile**

Specifies the location of the certificate that must be used to verify tokens. The default value of `/etc/keystone/ssl/certs/signing_cert.pem` is used if no value is specified.

**keyfile**

Specifies the location of the private key that must be used when signing tokens. The default value of `/etc/keystone/ssl/private/signing_key.pem` is used if no value is specified.

**token\_format**

Specifies the algorithm to use when generating tokens. Possible values are UUID and PKI. The default value is PKI.

**5.4.3.2. Creating the Public Key Infrastructure Files**[Report a bug](#)

This procedure documents the steps required to configure the PKI files for use by the identity service. All steps listed in this procedure must be performed while logged into the system hosting the identity service as the root user.

1. Run the `keystone-manage pki_setup` command.

```
# keystone-manage pki_setup \  
--keystone-user keystone \  
--keystone-group keystone
```

2. Ensure that the keystone user owns the `/var/log/keystone/` and `/etc/keystone/ssl/` directories.

```
# chown -R keystone:keystone /var/log/keystone \  
/etc/keystone/ssl/
```

The identity service PKI infrastructure files have been created and will be used when generating and signing tokens.

**5.4.4. Configuring for an LDAP Backend**[Report a bug](#)

The default identity service deployment is backed by an SQL database. As an alternative the identity service is able to use an existing directory server.

**Important**

For the Identity service to access an LDAP backend, SELinux requires the `authlogin_nsswitch_use_ldap` Boolean enabled on any client machine accessing the LDAP backend. Run the following command on each client machine as the root user to enable the Boolean and make it persistent across reboots:

```
# setsebool -P authlogin_nsswitch_use_ldap
```

This is an example directory server schema intended to support the identity service:

```

dn: cn=example,cn=org
dc: openstack
objectClass: dcObject
objectClass: organizationalUnit
ou: openstack

dn: ou=Groups,cn=example,cn=org
objectClass: top
objectClass: organizationalUnit
ou: groups

dn: ou=Users,cn=example,cn=org
objectClass: top
objectClass: organizationalUnit
ou: users

dn: ou=Roles,cn=example,cn=org
objectClass: top
objectClass: organizationalUnit
ou: roles

```

The corresponding entries in the identity service configuration file, `/etc/keystone/keystone.conf` are:

```

[ldap]
url = ldap://localhost
user = dc=Manager,dc=openstack,dc=org
password = badpassword
suffix = dc=openstack,dc=org
use_dumb_member = False
allow_subtree_delete = False

user_tree_dn = ou=Users,dc=openstack,dc=com
user_objectclass = inetOrgPerson

tenant_tree_dn = ou=Groups,dc=openstack,dc=com
tenant_objectclass = groupOfNames

role_tree_dn = ou=Roles,dc=example,dc=com
role_objectclass = organizationalRole

```

The default object classes and attributes defined in this example are intentionally simplistic. They reflect the common standard objects as described in the LDAP RFCs.

These attributes are merely suggestions and can be overridden to support a preexisting, more complex schema. As an example within the user object, the `objectClassposixAccount` described in RFC2307 is commonly found in directory server implementations.

If this is the underlying `objectclass`, then the `uid` field is likely to be named `uidNumber` and the `username` field is likely to be named either `uid` or `cn`. To change these two fields, the corresponding entries in the identity service configuration file are:

```

[ldap]
user_id_attribute = uidNumber
user_name_attribute = cn

```

There is also a set of allowed actions per object type that can be defined and modified depending on your specific deployment needs.

For example, the users are managed by another tool and you have only read access, in such case the configuration is:



```
[ldap]
user_allow_create = False
user_allow_update = False
user_allow_delete = False

tenant_allow_create = True
tenant_allow_update = True
tenant_allow_delete = True

role_allow_create = True
role_allow_update = True
role_allow_delete = True
```

There are also configuration options for filtering users, tenants and roles, if the backend is providing too much output. In such cases the configuration values that must be modified are:

```
[ldap]
user_filter = (memberof=CN=openstack-users,OU=workgroups,DC=openstack,DC=com)
tenant_filter =
role_filter =
```

Some directory servers do not expose a boolean attribute indicating whether a user account is enabled or disabled. Where this is the case it is possible to configure the Identity service to extract an equivalent value from an integer attribute such as that used by Active Directory:

```
[ldap]
user_enabled_attribute = userAccountControl
user_enabled_mask      = 2
user_enabled_default   = 512
```

In this example the attribute `userAccountControl` is an integer and the enabled attribute is listed in the first bit (bit 1). The values of the `user_enabled_mask` and the `user_enabled_attribute` are added together. If the resultant value matches the mask then the account is disabled.

The Identity service also saves the value without mask to the user identity in the attribute `enabled_nomask`. This is required to allow the restoration of the value when enabling or disabling a user. This needs to be done because the value contains more than just the status of the user. Setting the value of the `user_enabled_mask` configuration key is required in order to create a default value on the integer attribute (512 = NORMAL ACCOUNT on Active Directory).

In case of Active Directory the classes and attributes could not match the specified classes in the LDAP module so you can configure them like so:

```
[ldap]
user_objectclass      = person
user_id_attribute     = cn
user_name_attribute   = cn
user_mail_attribute   = mail
user_enabled_attribute = userAccountControl
user_enabled_mask     = 2
user_enabled_default  = 512
user_attribute_ignore = tenant_id, tenants
tenant_objectclass    = groupOfNames
tenant_id_attribute   = cn
tenant_member_attribute = member
tenant_name_attribute = ou
tenant_desc_attribute = description
tenant_enabled_attribute = extensionName
tenant_attribute_ignore =
role_objectclass      = organizationalRole
role_id_attribute     = cn
role_name_attribute   = ou
role_member_attribute = roleOccupant
role_attribute_ignore =
```

If you are connecting the identity service to a directory server, it is strongly recommended that you secure the connection between the two. In addition to supporting LDAPS, the identity service also provides Transport Layer Security (TLS) support.

Configuring TLS involves identifying the file or directory containing the certificates of the certificate authorities to recognize and trust when verifying the identity of the directory server. Additionally it is necessary to declare which checks the LDAP client must perform when verifying the directory server certificate. This functionality is configured as follows:

```
[ldap]
use_tls = True
tls_cacertfile = /etc/keystone/ssl/certs/cacert.pem
tls_cacertdir = /etc/keystone/ssl/certs/
tls_req_cert = demand
```

If both `tls_cacertfile` and `tls_cacertdir` are set then `tls_cacertfile` will be used and `tls_cacertdir` is ignored. Furthermore, valid options for `tls_req_cert` are `demand`, `never`, and `allow`. These correspond to the standard options permitted by the `TLS_REQCERT` TLS option.

### 5.4.5. Configuring the Firewall

[Report a bug](#)

As the identity service is used by all of the components in the OpenStack environment for authentication it must be accessible by them.

To allow this the firewall on the system hosting the identity service must be altered to allow network traffic on the required ports. All steps in this procedure must be run while logged in to the server hosting the identity service as the root user.

1. Open the `/etc/sysconfig/iptables` file in a text editor.
2. Add an INPUT rule allowing TCP traffic on ports 5000 and 35357 to the file. The new rule must appear before any INPUT rules that REJECT traffic.

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

3. Save the changes to the `/etc/sysconfig/iptables` file.
4. Restart the iptables service to ensure that the change takes effect.

```
# service iptables restart
```

The iptables firewall is now configured to allow incoming connections to the identity service on ports 5000 and 35357.

### 5.4.6. Populating the Identity Service Database

[Report a bug](#)

This procedure describes the steps required to populate the identity service database. These steps must be performed while logged in to the system hosting the identity service. The database connection string must already be defined in the configuration of the service.

- ▶ Use the `su` command to switch to the keystone user and run the `keystone-manage db_sync` command to initialize and populate the database identified in `/etc/keystone/keystone.conf`.

```
# su keystone -s /bin/sh -c "keystone-manage db_sync"
```

The identity service database has been initialized and populated.

[Report a bug](#)

## 5.5. Starting the Identity Service

All steps in this procedure must be performed while logged in to the server hosting the identity service as the root user.

1. Use the `service` command to start the `openstack-keystone` service.

```
# service openstack-keystone start
```

2. Use the `chkconfig` command to ensure that the `openstack-keystone` service will be started automatically in the future.

```
# chkconfig openstack-keystone on
```

The `openstack-keystone` service has been started.

[Report a bug](#)

## 5.6. Creating the Identity Service Endpoint

Once the identity service has been started its API endpoint must be defined. Some OpenStack services including the dashboard will not work unless this record is present.

All steps listed in this procedure must be performed while logged in to the identity server as the root user.

1. **Set the SERVICE\_TOKEN Environment Variable**

Set the `SERVICE_TOKEN` environment variable to the administration token. This is done by reading the token file created when setting the administration token.

```
# export SERVICE_TOKEN=`cat ~/ks_admin_token`
```

2. **Set the SERVICE\_ENDPOINT Environment Variable**

Set the `SERVICE_ENDPOINT` environment variable to point to the server hosting the identity service.

```
# export SERVICE_ENDPOINT="http://IP:35357/v2.0"
```

Replace `IP` with the IP address or host name of your identity server.

3. **Create a Service Entry**

Create a service entry for the identity service using the `keystone service-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 |
+-----+-----+
```

Take note of the unique identifier assigned to the entry. This value will be required in subsequent steps.

#### 4. Create an Endpoint for the API

Create an endpoint entry for the v2.0 API identity service using the `keystone endpoint-create` command.

```
# keystone endpoint-create \
  --service_id ID \
  --publicurl 'http://IP:5000/v2.0' \
  --adminurl 'http://IP:35357/v2.0' \
  --internalurl 'http://IP:5000/v2.0'
+-----+-----+
| Property | Value |
+-----+-----+
| adminurl | http://IP:35357/v2.0 |
| id | 1295011fdc874a838f702518e95a0e13 |
| internalurl | http://IP:5000/v2.0 |
| publicurl | http://IP:5000/v2.0 |
| region | regionOne |
| service_id | ID |
+-----+-----+
```

Replace *ID* with the service identifier returned in the previous step. Replace *IP* with the IP address or host name of the identity server.



#### Important

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



#### Note

By default, the endpoint is created in the default region, `regionOne`. If you need to specify a different region when creating an endpoint use the `--region` argument to provide it.

The identity service and endpoint entry has been created. The final step in identity service configuration is the creation of the default user accounts, roles, and tenants.

[Report a bug](#)

## 5.7. Creating an Administrator Account

Executing the following procedure will result in the creation of an administrative user as well as an associated tenant and role.

The steps listed in this procedure must be performed while logged in to the system hosting the Identity service as a user who has access to a file containing the administration token.

1. Set the `SERVICE_TOKEN` environment variable to the value of the administration token. This is done by reading the token file created when setting the administration token:

```
# SERVICE_TOKEN=`cat ~/ks_admin_token`
```

2. Set the `SERVICE_ENDPOINT` environment variable to point to the server hosting the identity service:

```
# export SERVICE_ENDPOINT="http://IP:35357/v2.0"
```

Replace *IP* with the IP address or host name of your identity server.

3. Use the `keystone user-create` command to create an admin user:

```
# keystone user-create --name admin --pass PASSWORD
+-----+-----+
| Property |          Value          |
+-----+-----+
| email    |                          |
| enabled  |             True        |
| id       | 94d659c3c9534095aba5f8475c87091a |
| name     |             admin       |
| tenantId |                          |
+-----+-----+
```

Replace *PASSWORD* with a secure password for the account. Take note of the created user's ID as it will be required in subsequent steps.

4. Use the `keystone role-create` command to create an admin role:

```
# keystone role-create --name admin
+-----+-----+
| Property |          Value          |
+-----+-----+
| id       | 78035c5d3cd94e62812d6d37551ecd6a |
| name     |             admin       |
+-----+-----+
```

Take note of the admin user's ID as it will be required in subsequent steps.

5. Use the `keystone tenant-create` command to create an admin tenant:

```
# keystone tenant-create --name admin
+-----+-----+
| Property |          Value          |
+-----+-----+
| description |                          |
| enabled    |             True        |
| id        | 6f8e3e36c4194b86b9a9b55d4b722af3 |
| name      |             admin       |
+-----+-----+
```

Take note of the admin tenant's ID as it will be required in the next step.

6. Now that the user account, role, and tenant have been created, the relationship between them must be explicitly defined using the `keystone user-role-add`:

```
# keystone user-role-add --user-id USERID --role-id ROLEID --tenant-id TENANTID
```

Replace the user, role, and tenant IDs with those obtained in the previous steps.

7. The newly created admin account will be used for future management of the identity service. To facilitate authentication, create a `kestonerc_admin` file in a secure location such as the home directory of the root user.

Add these lines to the file to set the environment variables that will be used for authentication:

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

Replace *PASSWORD* with the password of the admin user and replace *IP* with the IP address or host name of the identity server.

8. Run the source command on the file to load the environment variables used for authentication:

```
# source ~/keystonerc_admin
```

An administration user account, role, and tenant have been defined in the Identity server. The `keystonerc_admin` file has also been created for authenticating as the admin user.

[Report a bug](#)

## 5.8. Creating a Regular User Account

Executing the following procedure will result in the creation of a regular user as well as an associated tenant and role.

The steps listed in this procedure must be performed while logged in to the system hosting the Identity service as a user that has access to a file containing the administration token.

1. Load identity credentials from the `~/keystonerc_admin` file that was generated when the administrative user was created:

```
# source ~/keystonerc_admin
```

2. Use the `keystone user-create` to create a regular user:

```
# keystone user-create --name USER --pass PASSWORD
+-----+-----+
| Property | Value |
+-----+-----+
| email    |      |
| enabled  | True  |
| id       | b8275d7494dd4c9cb3f69967a11f9765 |
| name     | USER |
| tenantId |      |
+-----+-----+
```

Replace *USER* with the user name that you would like to use for the account. Replace *PASSWORD* with a secure password for the account. Take note of the created user's ID as it will be required in subsequent steps.

3. Use the `keystone role-create` command to create an Member role. The Member role is the default role required for access to the dashboard:

```
# keystone role-create --name Member
+-----+-----+
| Property | Value |
+-----+-----+
| id       | 78035c5d3cd94e62812d6d37551ecd6a |
| name     | Member |
+-----+-----+
```

Take note of the created role's ID as it will be required in subsequent steps.

4. Use the keystone `tenant-create` command to create a tenant:

```
# keystone tenant-create --name TENANT
+-----+-----+
| Property | Value |
+-----+-----+
| description | |
| enabled | True |
| id | 6f8e3e36c4194b86b9a9b55d4b722af3 |
| name | TENANT |
+-----+-----+
```

Replace *TENANT* with the name that you wish to give to the tenant. Take note of the created tenant's ID as it will be required in the next step.

5. Now that the user account, role, and tenant have been created, the relationship between them must be explicitly defined using the keystone `user-role-add`:

```
# keystone user-role-add --user-id USERID --role-id ROLEID --tenant-id
TENANTID
```

Replace the user, role, and tenant IDs with those obtained in the previous steps.

6. To facilitate authentication create a `keystonerc_user` file in a secure location such as the home directory of the root user.

Set these environment variables that will be used for authentication:

```
export OS_USERNAME=USER
export OS_TENANT_NAME=TENANT
export OS_PASSWORD=PASSWORD
export OS_AUTH_URL=http://IP:5000/v2.0/
export PS1='[\u@\h \w(keystone_user)]\$ '
```

Replace *USER* and *TENANT* with the name of the new user and tenant respectively. Replace *PASSWORD* with the password of the user and replace *IP* with the IP address or host name of the identity server.

A regular user account, role, and tenant have been defined in the identity server. A `keystonerc_user` file has also been created for authenticating as the created user.

[Report a bug](#)

## 5.9. Creating the Services Tenant

Tenants are used to aggregate service resources (tenants are also known as projects). Per tenant, quota controls can be used to limit the numbers of resources (refer to *Managing Quotas*).

Each user is assigned to a tenant. For regular users, their tenant typically represents their group, project, or organisation. For service users (the entity accessing the Identity service on behalf of the service), the tenant represents a service's geographical region. This means that if your cloud's services are:

- ▶ Distributed, typically one service tenant is created for each endpoint on which services are running (excepting the Identity and Dashboard services).
- ▶ Deployed on a single node, only one service tenant is required (but of course this is just one option; more can be created for administrative purposes).

This guide's installation examples are based on all services being deployed on one node, therefore only one service tenant is required; all guide examples use the services tenant.



## Note

Because administrators, regular users, and service users all need a tenant, at least three tenants are typically created, one for each group. To create administrative and regular users and tenants, refer to *Creating an Administrator Account* and *Creating a Regular User Account*.

To create the services tenant:

1. Run the source command on the file containing the environment variables used to identify the Identity service administrator.

```
# source ~/keystonerc_admin
```

2. Create the services tenant in the Identity service:

```
# keystone tenant-create --name services --description "Services Tenant"
+-----+-----+
| Property | Value |
+-----+-----+
| description | Services Tenant |
| enabled | True |
| id | 7e193e36c4194b86b9a9b55d4b722af3 |
| name | services |
+-----+-----+
```



## Note

To obtain a list of all Identity service tenants and their IDs, execute:

```
# keystone tenant-list
```

[Report a bug](#)

## 5.10. Validating the Identity Service Installation

Follow the steps outlined in this procedure to verify that an identity service installation is functioning correctly. These steps must be performed while logged in to either the identity server or another system.

The logged in user must have access to `keystonerc_admin` and `keystonerc_user` files containing the environment variables required to authenticate as the administrator user and a regular user respectively.

1. Run the source command on the file containing the environment variables used to identify the identity service administrator.

```
# source ~/keystonerc_admin
```

2. Run the `keystone user-list` command to authenticate with the identity service and list the users defined in the system.



```
# keystone user-list
+-----+-----+-----+-----+
|          id          | name | enabled |          email          |
+-----+-----+-----+-----+
| 94d659c3c9534095aba5f8475c87091a | admin | True   |                          |
| b8275d7494dd4c9cb3f69967a11f9765 | USER | True   |                          |
+-----+-----+-----+-----+
```

The list of users defined in the system is displayed. If the list is not displayed then there is an issue with the installation.

- a. If the message returned indicates a permissions or authorization issue then check that the administrator user account, tenant, and role were created properly. Also ensure that the three objects are linked correctly.

```
Unable to communicate with identity service: {"error": {"message": "You are not authorized to perform the requested action: admin_required", "code": 403, "title": "Not Authorized"}}. (HTTP 403)
```

- b. If the message returned indicates a connectivity issue then verify that the openstack-keystone service is running and that iptables is configured to allow connections on ports 5000 and 35357.

```
Authorization Failed: [Errno 111] Connection refused
```

3. Run the source command on the file containing the environment variables used to identify the regular identity service user.

```
# source ~/keystonerc_user
```

4. Run the keystone user-list command to authenticate with the identity service and list the users defined in the system.

```
# keystone user-list
Unable to communicate with identity service: {"error": {"message": "You are not authorized to perform the requested action: admin_required", "code": 403, "title": "Not Authorized"}}. (HTTP 403)
```

An error message is displayed indicating that the user is Not Authorized to run the command. If the error message is not displayed but instead the user list appears then the regular user account was incorrectly attached to the admin role.

5. Run the keystone token-get command to verify that the regular user account is able to run commands that it is authorized to access.

```
# keystone token-get
+-----+-----+-----+-----+
| Property |          Value          |
+-----+-----+-----+-----+
| expires  | 2013-05-07T13:00:24Z   |
| id       | 5f6e089b24d94b198c877c58229f2067 |
| tenant_id | f7e8628768f2437587651ab959fbe239 |
| user_id  | 8109f0e3deaf46d5990674443dcf7db7 |
+-----+-----+-----+-----+
```

The identity service is installed and functioning correctly.

# Chapter 6. Installing the OpenStack Object Storage Service

## 6.1. Services that Make Up the Object Storage Service

The Object Storage Service is made up of 4 services that work together to manage the storage of data objects.

### Proxy Service

The proxy service uses the object ring to decide where to direct newly uploaded objects. It updates the relevant container database to reflect the presence of a new object. If a newly uploaded object goes to a new container, the proxy service updates the relevant account database to reflect the new container.

The proxy service also directs get requests to one of the nodes where a replica of the requested object is stored, either randomly, or based on response time from the node.

### Object Service

The object service is responsible for storing data objects in partitions on disk devices. Each partition is a directory. Each object is held in a subdirectory of its partition directory. A MD5 hash of the path to the object is used to identify the object itself.

### Container Service

The container service maintains databases of objects in containers. There is one database file for each container, and the database files are replicated across the cluster. Containers are defined when objects are put in them. Containers make finding objects faster by limiting object listings to specific container namespaces.

### Account Service

The account service maintains databases of all of the containers accessible by any given account. There is one database file for each account, and the database files are replicated across the cluster. Any account has access to a particular group of containers. An account maps to a tenant in the Identity Service.

## 6.2. Architecture of the Object Storage Service

The OpenStack Object Storage service is a modular group of services, including **openstack-swift-proxy**, **openstack-swift-object**, **openstack-swift-container**, and **openstack-swift-account**.

All of the services can be installed on each node. Alternatively, services can be run on dedicated nodes.

### Common Object Storage Service Deployment Configurations

#### All services on all nodes.

Simplest to set up.

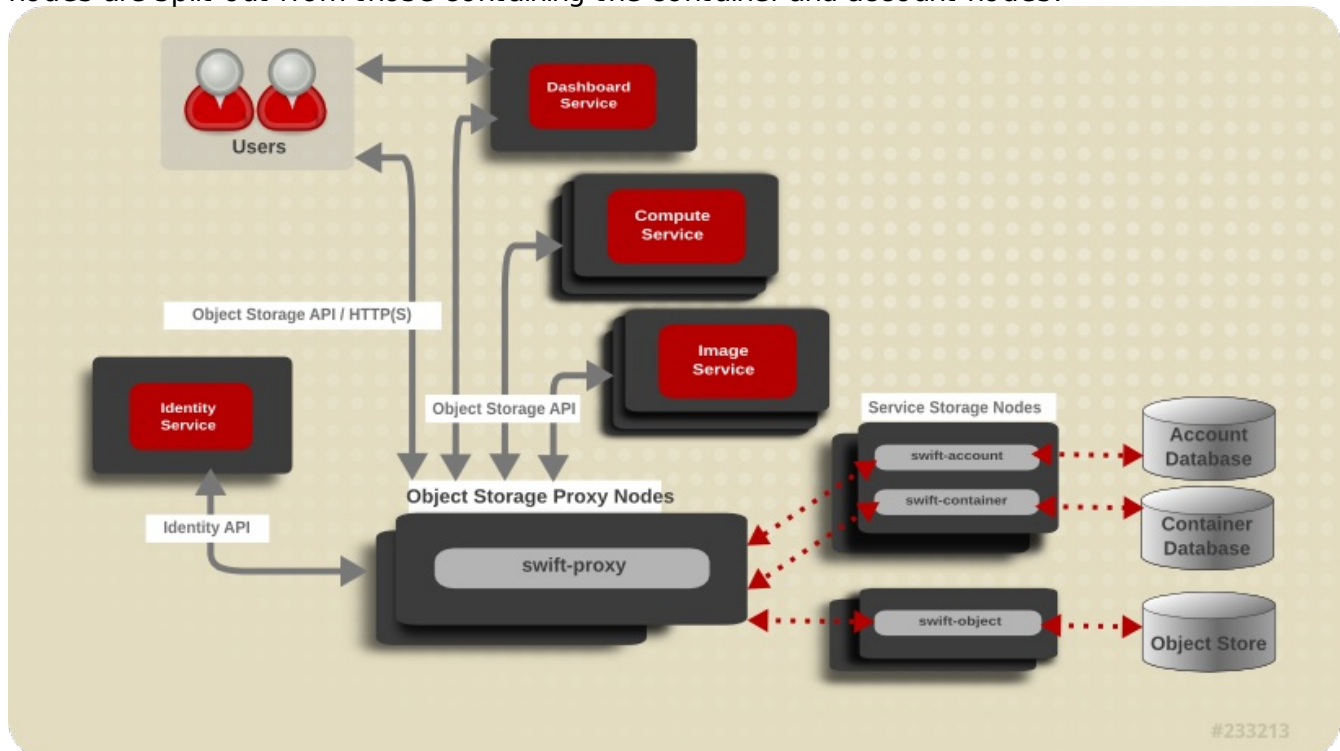
**Dedicated proxy nodes, all other services combined on other nodes.**

The proxy service is CPU and I/O intensive. The other services are disk and I/O intensive. This configuration allows you to optimize your hardware usage.

**Dedicated proxy nodes, dedicated object service nodes, container and account services combined on other nodes.**

The proxy service is CPU and I/O intensive. The container and account services are more disk and I/O intensive than the object service. This configuration allows you to optimize your hardware usage even more.

The following diagram provides an overview of the third option, where the proxy and object nodes are split out from those containing the container and account nodes:



**Figure 6.1. Service Architecture**

[Report a bug](#)

## 6.3. Object Storage Service Requirements

### Supported Filesystems

The Object Storage Service stores objects in filesystems. Currently, XFS and ext4 are supported. The ext4 filesystem is recommended.

Your filesystem must be mounted with `xattr` enabled. For example, this is from `/etc/fstab`:

```
/dev/sdb1 /srv/node/d1 ext4 acl,user_xattr 0 0
```

### Acceptable Mountpoints

The Object Storage service expects devices to be mounted at `/srv/node/`.

## 6.4. Installing the Object Storage Service Packages

The OpenStack Object Storage service is packaged in the following packages.

### Primary OpenStack Object Storage packages

#### ***openstack-swift-proxy***

Proxies requests for objects.

#### ***openstack-swift-object***

Stores data objects of up to 5GB.

#### ***openstack-swift-container***

Maintains a database that tracks all of the objects in each container.

#### ***openstack-swift-account***

Maintains a database that tracks all of the containers in each account.

### OpenStack Object Storage dependencies

#### ***openstack-swift***

Contains code common to the specific services.

#### ***openstack-swift-plugin-swift3***

The swift3 plugin for OpenStack Object Storage.

#### ***memcached***

Soft dependency of the proxy server, caches authenticated clients rather than making them reauthorize at every interaction.

#### ***openstack-utils***

Provides utilities for configuring Openstack.

### Procedure 6.1. Installing the Object Storage Service Packages

- » Install the required packages using the yum command as the root user:

```
# yum install -y openstack-swift-proxy \  
    openstack-swift-object \  
    openstack-swift-container \  
    openstack-swift-account \  
    openstack-utils \  
    memcached
```

The services that make up the OpenStack Object Storage service are installed and ready to be configured.

## 6.5. Configuring the Object Storage Service

### 6.5.1. Configuring the Identity Service to work with the Object Storage Service

#### Prerequisites:

- ▶ [Section 5.7, “Creating an Administrator Account”](#)

In this procedure, you will:

1. Create the `swift` user, who has the `admin` role in the `services` tenant.
2. Create the `swift` service entry and assign it an endpoint.

This procedure can be performed from your Identity Service server or on any machine where you've copied the `keystonerc_admin` file (which contains administrator credentials) and the `keystone` command-line utility is installed.

#### Procedure 6.2. Configuring the Identity Service to work with the Object Storage Service

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

```
$ source ~/keystonerc_admin
```

2. Create the `swift` user and set its password by replacing `PASSWORD` with your chosen password:

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

Note the create user's ID as it will be used in subsequent steps.

3. Get the ID of the `admin` role:

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

If no `admin` role exists, create one:

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

4. Get the ID of the `services` tenant:

```
$ keystone tenant-list | grep services
```

If no `services` tenant exists, create one:

```
$ keystone tenant-create --name services --description "Services Tenant"
```

This guide uses one tenant for all service users. For more information, refer to *Creating the Services Tenant*.

5. Add the `swift` user to the `services` tenant with the `admin` role:

```
$ keystone user-role-add --role-id ROLEID --tenant-id TENANTID --user-id USERID
```

Replace the user, role, and tenant IDs with those obtained in the previous steps.

6. Create the `swift` Object Storage service entry:

```
$ keystone service-create --name swift --type object-store \
  --description "Swift Storage Service"
```

Note the create service's ID as it will be used in the next step.

## 7. Create the swift endpoint entry:

```
$ keystone endpoint-create --service_id SERVICEID \
  --publicurl "http://IP:8080/v1/AUTH_\$(tenant_id)s" \
  --adminurl "http://IP:8080/v1" \
  --internalurl "http://IP:8080/v1/AUTH_\$(tenant_id)s"
```

Replace *SERVICEID* with the identifier returned by the `keystone service-create` command. Replace *IP* with the IP address of fully qualified domain name of the system hosting the Object Storage Proxy service.

You have configured the Identity Service to work with the Object Storage Service.

### 6.5.2. Configuring the Object Storage Service Storage Nodes

[Report a bug](#)

The Object Storage Service stores objects on the filesystem, usually on a number of connected physical storage devices. All of the devices which will be used for object storage must be formatted ext4 or XFS, and mounted under the `/srv/node/` directory. All of the services that will run on a given node must be enabled, and their ports opened.

While you can run the proxy service alongside the other services, the proxy service is not covered in this procedure.

#### Procedure 6.3. Configuring the Object Storage Service Storage Nodes

1. Format your devices using the ext4 or XFS filesystem. Make sure that xattrs are enabled.
2. Add your devices to the `/etc/fstab` file to ensure that they are mounted under `/srv/node/` at boot time.

Use the `blkid` command to find your device's unique ID, and mount the device using its unique ID.



#### Note

If using ext4, ensure that extended attributes are enabled by mounting the filesystem with the `user_xattr` option. (In XFS, extended attributes are enabled by default.)

3. Open the TCP ports used by each service running on each node. By default, the account service uses port 6002, the container service uses port 6001, and the object service uses port 6000.

```
# iptables -A INPUT -p tcp -m multiport --dports 6000,6001,6002,873 -j
ACCEPT
# service iptables save
# service iptables restart
```

The `-A` parameter appends the rule to the end of the iptables firewall. Make sure that the rule doesn't fall after a `reject-with icmp-host-prohibited` rule.

4. Change the owner of the contents of `/srv/node/` to `swift:swift` with the `chown` command.

```
# chown -R swift:swift /srv/node/
```

5. Set the SELinux context correctly for all directories under `/srv/node/` with the `restorcon` command.

```
# restorecon -R /srv
```

6. Use the `openstack-config` command to add a hash prefix and suffix to your `/etc/swift.conf`.

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

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

These details are required for finding and placing data on all of your nodes. Back `/etc/swift.conf` up.

7. Use the `openstack-config` command to set the IP address your storage services will listen on. Run these commands for every service on every node in your Object Storage cluster.

```
# openstack-config --set /etc/swift/object-server.conf DEFAULT bind_ip
node_ip_address
# openstack-config --set /etc/swift/account-server.conf DEFAULT bind_ip
node_ip_address
# openstack-config --set /etc/swift/container-server.conf DEFAULT bind_ip
node_ip_address
```

The `DEFAULT` argument specifies the `DEFAULT` section of the service configuration file. Replace `node_ip_address` with the IP address of the node you are configuring.

8. Copy `/etc/swift.conf` from the node you are currently configuring, to all of your Object Storage Service nodes.



### Important

The `/etc/swift.conf` file must be identical on all of your Object Storage Service nodes.

9. Start the services which will run on your node.

```
# service openstack-swift-account start
# service openstack-swift-container start
# service openstack-swift-object start
```

10. Use the `chkconfig` command to make sure the services automatically start at boot time.

```
# chkconfig openstack-swift-account on
# chkconfig openstack-swift-container on
# chkconfig openstack-swift-object on
```

All of the devices that your node will provide as object storage are formatted and mounted under `/srv/node/`. Any service running on the node has been enabled, and any ports used by services on the node have been opened.

### 6.5.3. Configuring the Object Storage Service Proxy Service

[Report a bug](#)

The Object Storage proxy service determines to which node gets and puts are directed.

Although you can run the account, container, and object services alongside the proxy service, only the proxy service is covered in the following procedure.

### Procedure 6.4. Configuring the Object Storage Service Proxy Service

1. Update the configuration file for the proxy server with the correct authentication details for the appropriate service user:

```
# openstack-config --set /etc/swift/proxy-server.conf \
  filter:authtoken auth_host IP
# 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
```

Where:

- ▶ *IP* - The IP address or host name of the Identity server.
  - ▶ *services* - The name of the tenant that was created for the use of the Object Storage service (previous examples set this to *services*).
  - ▶ *swift* - The name of the service user that was created for the Object Storage service (previous examples set this to *swift*).
  - ▶ *PASSWORD* - The password associated with the service user.
2. Start the memcached and openstack-swift-proxy services using the service command:

```
# service memcached start
# service openstack-swift-proxy start
```

3. Use the chown command to change the ownership of the keystone signing directory:

```
# chown swift:swift /tmp/keystone-signing-swift
```

4. Enable the memcached and openstack-swift-proxy services permanently using the chkconfig command:

```
# chkconfig memcached on
# chkconfig openstack-swift-proxy on
```

5. 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*.

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

```
# service iptables save
# service iptables restart
```

The Object Storage Service proxy service is now listening for HTTP put and get requests on port 8080, and directing them to the appropriate nodes.



[Report a bug](#)

### 6.5.4. Object Storage Service Rings

Rings determine where data is stored in a cluster of storage nodes. Ring files are generated using the **swift-ring-builder** tool. Three ring files are required, one each for the **object**, **container**, and **account** services.

Each storage device in a cluster is divided into partitions, with a recommended minimum of 100 partitions per device. Each partition is physically a directory on disk.

A configurable number of bits from the MD5 hash of the filesystem path to the partition directory, known as the *partition power*, is used as a partition index for the device. The *partition count* of a cluster that has 1000 devices, where each device has 100 partitions on it, is 100 000.

The partition count is used to calculate the partition power, where 2 to the partition power is the partition count. When the partition power is a fraction, it is rounded up. If the partition count is 100 000, the part power is 17 (16.610 rounded up).

Expressed mathematically:  $2^{\text{partition power}} = \text{partition count}$ .

Ring files are generated using 3 parameters: partition power, replica count, and the amount of time that must pass between partition reassignments.

A fourth parameter, zone, is used when adding devices to rings. Zones are a flexible abstraction, where each zone should be as separated from other zones as possible in your specific. You can use a zone to represent sites, cabinets, nodes, or even devices.

**Table 6.1. Parameters used when building ring files**

Ring File Parameter	Description
Partition power	$2^{\text{partition power}} = \text{partition count}$ . The partition is rounded up after calculation.
Replica count	The number of times that your data will be replicated in the cluster.
min_part_hours	Minimum number of hours before a partition can be moved. This parameter increases availability of data by not moving more than one copy of a given data item within that min_part_hours amount of time.

[Report a bug](#)

### 6.5.5. Building Object Storage Service Ring Files

Three ring files need to be created: one to track the objects stored by the Object Storage Service, one to track the containers that objects are placed in, and one to track which accounts can access which containers. The ring files are used to deduce where a particular piece of data is stored.

#### Procedure 6.5. Building Object Storage Service Ring Files

1. Use the `swift-ring-builder` command to build one ring for each service. Provide a builder file, a *partition power*, a *replica count*, and the *minimum hours between partition re-assignment*:

```
# swift-ring-builder /etc/swift/object.builder create part_power
replica_count min_part_hours
# swift-ring-builder /etc/swift/container.builder create part_power
replica_count min_part_hours
# swift-ring-builder /etc/swift/account.builder create part_power
replica_count min_part_hours
```

2. When the rings are created, add storage devices to each ring:

- a. Add devices to the accounts ring. Repeat for each device on each node in the cluster that you want added to the ring.

```
# swift-ring-builder /etc/swift/account.builder add zX-
127.0.0.1:6002/device_mountpoint partition_count
```

- ▶ Specify a zone with *zX*, where *X* is an integer (for example, *z1* for zone one).
- ▶ By default, all three services (account, container, and object) listen on the 127.0.0.1 address, and the above command matches this default. However, the service's machine IP address can also be used (for example, to handle distributed services). If you do use a real IP, remember to change the service's bind address to the same IP address or to '0.0.0.0' (configured in the `/etc/swift/service-server.conf` file).
- ▶ TCP port 6002 is the default port that the account server uses.
- ▶ The *device\_mountpoint* is the directory under `/srv/node/` that your device is mounted at.
- ▶ The recommended minimum number for *partition\_count* is 100, use the partition count you used to calculate your partition power.

- b. Add devices to the containers ring. Repeat for each device on each node in the cluster that you want added to the ring.

```
# swift-ring-builder /etc/swift/container.builder add zX-
127.0.0.1:6001/device_mountpoint partition_count
```

- ▶ TCP port 6001 is the default port that the container server uses.

- c. Add devices to the objects ring. Repeat for each device on each node in the cluster that you want added to the ring.

```
# swift-ring-builder /etc/swift/object.builder add zX-
127.0.0.1:6000/device_mountpoint partition_count
```

- ▶ TCP port 6000 is the default port that the object server uses.

3. Distribute the partitions across the devices in the ring using the `swift-ring-builder` command's `rebalance` argument.

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

4. 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
```

5. 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.



## Important

All mount points must be owned by root; all roots of mounted file systems must be owned by swift. Before running the following command, ensure that all devices are already mounted and owned by root.

```
# chown -R root:swift /etc/swift
```

6. Copy each ring builder file to each node in the cluster, storing them under /etc/swift/.

```
# scp /etc/swift/*.gz node_ip_address:/etc/swift
```

You have created rings for each of the services that require them. You have used the builder files to distribute partitions across the nodes in your cluster, and have copied the ring builder files themselves to each node in the cluster.

[Report a bug](#)

## 6.6. Validating the Object Storage Service Installation

### Summary

You've finished installing and configuring the Openstack Object Service. You want to make sure it is working.

1. On your proxy server node, use the `openstack-config` command to turn on debug level logging:

```
# openstack-config --set /etc/swift/proxy-server.conf DEFAULT log_level debug
```

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

Use the `swift list` to make sure you can connect to your proxy server:

```
$ swift list
Message from syslogd@thildred-swift-01 at Jun 14 02:46:00 ...
0135 proxy-server Server reports support for api versions: v3.0, v2.0
```

3. Use the `swift` command to upload some files to your Object Storage Service nodes

```
$ head -c 1024 /dev/urandom > data1.file ; swift upload c1 data1.file
$ head -c 1024 /dev/urandom > data2.file ; swift upload c1 data2.file
$ head -c 1024 /dev/urandom > data3.file ; swift upload c1 data3.file
```

4. Use the `swift` command to take a listing of the objects held in your Object Storage Service cluster.

```
$ swift list
$ swift list c1
data1.file
data2.file
data3.file
```

### Result

You have now uploaded 3 files into 1 container. If you look in the various storage devices you see more .data files, based on your replica count.

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

[Report a bug](#)

# Chapter 7. Installing the OpenStack Image Service

## 7.1. Image Service Requirements

To install the Image service, you must have access to:

- ▶ MySQL database server root credentials and IP address
- ▶ Identity service administrator credentials and endpoint URL

If using the OpenStack Object storage service as the storage backend, you will also need to know the service's endpoint public URL.



### Note

For software and hardware requirements, refer to the *Prerequisites* chapter.

### See Also:

- ▶ [Chapter 2, Prerequisites](#)

[Report a bug](#)

## 7.2. Installing the Image Service Packages

The OpenStack Image service requires the following packages:

### ***openstack-glance***

Provides the OpenStack Image service.

### ***openstack-utils***

Provides supporting utilities to assist with a number of tasks including the editing of configuration files.

### ***openstack-selinux***

Provides OpenStack specific SELinux policy modules.

To install all of the above packages, execute the following command while logged in as the root user.

```
# yum install -y openstack-glance openstack-utils openstack-selinux
```

The OpenStack Image service is now installed and ready to be configured.

[Report a bug](#)

## 7.3. Creating the Image Service Database

In this procedure, the database and database user that will be used by the Image service will be created. These steps must be performed while logged in to the database server as the root user (or as a user with suitable access: `create db, create user, grant permissions`).

1. Connect to the database service using the `mysql` command.

```
# mysql -u root -p
```

2. Create the glance database.

```
mysql> CREATE DATABASE glance;
```

3. Create a glance database user and grant it access to the glance database.

```
mysql> GRANT ALL ON glance.* TO 'glance'@'%' IDENTIFIED BY 'PASSWORD';  
mysql> GRANT ALL ON glance.* TO 'glance'@'localhost' IDENTIFIED BY  
'PASSWORD';
```

Replace `PASSWORD` with a secure password that will be used to authenticate with the database server as this user.

4. Flush the database privileges to ensure that they take effect immediately.

```
mysql> FLUSH PRIVILEGES;
```

5. Exit the `mysql` client.

```
mysql> quit
```

The Image Storage database has been created. The database will be populated during service configuration.

[Report a bug](#)

## 7.4. Configuring the Image Service

### 7.4.1. Configuration Overview

To configure the Image service, the following must be completed:

- ▶ Configure TLS/SSL.
- ▶ Configure the Identity service for Image service authentication (create database entries, set connection strings, and update configuration files).
- ▶ Configure the disk-image storage backend (this guide uses the Object Storage service).
- ▶ Configure the firewall for Image service access.
- ▶ Populate the Image service database.

### 7.4.2. Creating the Image Identity Records

[Report a bug](#)

The steps outlined in this procedure cover the creation of these identity records to support the Image service:

1. Create the glance, who has the admin role in the services tenant.
2. Create the glance service entry and assign it an endpoint.

These entries assist other OpenStack services attempting to locate and access the volume functionality provided by the Image service.

1. Authenticate as the administrator of the identity service by running the source command on the `keystonerc_admin` file containing the required credentials:

```
# source ~/keystonerc_admin
```

2. Create a user named glance for the Image service to use:

```
# keystone user-create --name glance --pass PASSWORD
+-----+-----+
| Property |           Value           |
+-----+-----+
| email    |                           |
| enabled  |             True          |
| id       | 8091eaf121b641bf84ce73c49269d2d1 |
| name     |             glance        |
| tenantId |                           |
+-----+-----+
```

Replace *PASSWORD* with a secure password that will be used by the image storage service when authenticating with the identity service. Take note of the returned user ID (used in subsequent steps).

3. Get the ID of the admin role:

```
# keystone role-get admin
```

If no admin role exists, create one:

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

4. Get the ID of the services tenant:

```
$ keystone tenant-list | grep services
```

If no services tenant exists, create one:

```
$ keystone tenant-create --name services --description "Services Tenant"
```

This guide uses one tenant for all service users. For more information, refer to *Creating the Services Tenant*.

5. Use the keystone `user-role-add` command to link the glance user and the admin role together within the context of the services tenant:

```
# keystone user-role-add --user-id USERID --role-id ROLEID --tenant-id TENANTID
```

6. Create the glance service entry:

```
# keystone service-create --name glance \
  --type image \
  --description "Glance Image Service"
+-----+-----+
| Property |           Value           |
+-----+-----+
| description | Glance Image Service |
| id         | 7461b83f96bd497d852fb1b85d7037be |
| name      |             glance        |
| type     |             image         |
+-----+-----+
```

Take note of the service's returned ID (used in the next step).

7. Create the glance endpoint entry:

```
# keystone endpoint-create --service-id SERVICEID \
  --publicurl "http://IP:9292" \
  --adminurl "http://IP:9292" \
  --internalurl "http://IP:9292"
```

Replace *SERVICEID* with the identifier returned by the keystone `service-create` command. Replace *IP* with the IP address or host name of the system hosting the Image

service.

All supporting identity service entries required by the Image service have been created.

### 7.4.3. Setting the Database Connection String

[Report a bug](#)

The database connection string used by the Image service is defined in the `/etc/glance/glance-api.conf` and `/etc/glance/glance-registry.conf` files. It must be updated to point to a valid database server before starting the service.

All commands in this procedure must be run while logged in as the root user on the server hosting the Image service.

1. Use the `openstack-config` command to set the value of the `sql_connection` configuration key in the `/etc/glance/glance.conf` file.

```
# openstack-config --set /etc/glance/glance-api.conf \
  DEFAULT sql_connection mysql://USER:PASS@IP/DB
```

Replace:

- ▶ *USER* with the database user name the Image service is to use, usually `glance`.
- ▶ *PASS* with the password of the chosen database user.
- ▶ *IP* with the IP address or host name of the database server.
- ▶ *DB* with the name of the database that has been created for use by the identity service, usually `glance`.

2. Use the `openstack-config` command to set the value of the `sql_connection` configuration key in the `/etc/glance/glance-registry.conf` file.

```
# openstack-config --set /etc/glance/glance-registry.conf \
  DEFAULT sql_connection mysql://USER:PASS@IP/DB
```

Replace the placeholder values *USER*, *PASS*, *IP*, and *DB* with the same values used in the previous step.

The database connection string has been set and will be used by the Image service.

### 7.4.4. Configuring the Use of the Identity Service

[Report a bug](#)

To update the Image configuration files for Identity usage, execute the following commands as the root user on each node hosting the Image service:

1. Configure the `glance-api` service:

```
# openstack-config --set /etc/glance/glance-api.conf \
  paste_deploy flavor keystone
# openstack-config --set /etc/glance/glance-api.conf \
  keystone_auth token auth_host IP
# openstack-config --set /etc/glance/glance-api.conf \
  keystone_auth token auth_port 35357
# openstack-config --set /etc/glance/glance-api.conf \
  keystone_auth token auth_protocol http
# openstack-config --set /etc/glance/glance-api.conf \
  keystone_auth token admin_tenant_name services
# openstack-config --set /etc/glance/glance-api.conf \
  keystone_auth token admin_user glance
# openstack-config --set /etc/glance/glance-api.conf \
  keystone_auth token admin_password PASSWORD
```

2. Configure the `glance-registry` service:



```
# openstack-config --set /etc/glance/glance-registry.conf \
paste_deploy flavor keystone
# openstack-config --set /etc/glance/glance-registry.conf \
keystone_auth token auth_host IP
# openstack-config --set /etc/glance/glance-registry.conf \
keystone_auth token auth_port 35357
# openstack-config --set /etc/glance/glance-registry.conf \
keystone_auth token auth_protocol http
# openstack-config --set /etc/glance/glance-registry.conf \
keystone_auth token admin_tenant_name services
# openstack-config --set /etc/glance/glance-registry.conf \
keystone_auth token admin_user glance
# openstack-config --set /etc/glance/glance-registry.conf \
keystone_auth token admin_password PASSWORD
```

where:

- ▶ *IP* - The IP address or host name of the Identity server.
- ▶ *services* - The name of the tenant that was created for the use of the Image service (previous examples set this to *services*).
- ▶ *glance* - The name of the service user that was created for the Image service (previous examples set this to *glance*).
- ▶ *PASSWORD* - The password associated with the service user.

### 7.4.5. Using the Object Storage Service for Image Storage

[Report a bug](#)

By default, the Image service uses the local file system (*file*) for its storage backend. However, either of the following storage backends can be used to store uploaded disk images:

- ▶ *file* - Local file system of the Image server (`/var/lib/glance/images/` directory)
- ▶ *swift* - OpenStack Object Storage service



#### Note

The configuration procedure below uses the `openstack-config` command. However, the `/etc/glance/glance-api.conf` file can also be manually updated. If manually updating the file:

1. Ensure that the `default_store` parameter is set to the correct backend (for example, `default_store=rbd`).
2. Update the parameters in that backend's section (for example, under 'RBD Store Options').

To change the configuration to use the Object Storage service, execute the following steps as the root user:

1. Set the `default_store` configuration key to `swift`:

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

2. Set the `swift_store_auth_address` configuration key to the public endpoint for the Identity service:

```
# openstack-config --set /etc/glance/glance-api.conf \
  DEFAULT swift_store_auth_address http://IP:5000/v2.0/
```

3. Add the container for storing images in the Object Storage Service:

```
# openstack-config --set /etc/glance/glance-api.conf \ DEFAULT
  swift_store_create_container_on_put True
```

4. 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 Object Storage, these values must be replaced with the services tenant and the swift user respectively.
  - ▶ If you did not follow the instructions in this guide to deploy Object Storage, these values must be replaced with the appropriate Object Storage tenant and user for your environment.

```
# openstack-config --set /etc/glance/glance-api.conf \
  DEFAULT swift_store_user services:swift
```

5. Set the `swift_store_key` configuration key to the password of the user to be used for authentication (that is, the password that was set for the swift user when deploying the Object Storage service).

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

## 7.4.6. Configuring the Firewall

[Report a bug](#)

Systems attempting to use the functionality provided by the image storage service access it over the network using port 9292.

To allow this, the Image service is configured to recognize the 9292 port, and the firewall on the system hosting the image storage service is altered to allow network traffic on the port. All steps in this procedure must be run while logged in to the server hosting the image storage service as the root user.

1. Open the `/etc/glance/glance-api.conf` file in a text editor, and remove any comment characters from in front of the following parameters:

```
bind_host = 0.0.0.0
bind_port = 9292
```

2. Open the `/etc/sysconfig/iptables` file in a text editor.
3. Add an INPUT rule allowing TCP traffic on port 9292 to the file. The new rule must appear before any INPUT rules that REJECT traffic.

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

4. Save the changes to the `/etc/sysconfig/iptables` file.
5. Restart the iptables service to ensure that the change takes effect.

```
# service iptables restart
```

The iptables firewall is now configured to allow incoming connections to the image storage service on port 9292.

## 7.4.7. Populating the Image Service Database

[Report a bug](#)

This procedure describes the steps required to populate the Image service database. These

steps must be performed while logged in to the system hosting the Image service. Log in as the root user initially. The database connection string must already be defined in the configuration of the service.

1. Use the `su` command to switch to the `glance` user.

```
# su glance -s /bin/sh
```

2. Run the `glance-manage db_sync` command to initialize and populate the database identified in `/etc/glance/glance-api.conf` and `/etc/glance/glance-registry.conf`.

```
# glance-manage db_sync
```

The Image service database has been initialized and populated.

[Report a bug](#)

## 7.5. Starting the Image API and Registry Services

Now that Glance has been configured, start the `glance-api` and `glance-registry` services as the root user:

```
# service openstack-glance-registry start
# service openstack-glance-api start
# chkconfig openstack-glance-registry on
# chkconfig openstack-glance-api on
```

[Report a bug](#)

## 7.6. Validating the Image Service Installation

### 7.6.1. Obtaining a Test Disk Image

A disk image can be downloaded from Red Hat, which can be used as a test in the import of images into the Image service (refer to [Uploading an Image](#)).

A new image is provided with each minor RHEL 6 release, and is available in the [Download](#) section of the RHEL 6 Server channel:

<https://rhn.redhat.com/rhn/software/channel/downloads/Download.do?cid=16952>

The `wget` command below uses an example URL.

```
# mkdir /tmp/images
# cd /tmp/images
# wget -c -O rhel-6-server-x86_64-disc1.iso "https://content-
web.rhn.redhat.com/rhn/isos/xxxx/rhel-6-server-x86_64-disc1.isoxxxxxxx"
```

### 7.6.2. Building a Custom Disk Image

[Report a bug](#)

Red Hat Enterprise Linux OpenStack Platform 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 the Image service and used to launch virtual machine instances.

- ▶ **Template Description Language (TDL) Files** - Oz accepts input in the form of XML-based TDL files, which describe the operating system being installed, the installation media's source, and any additional packages or customization changes that must be applied to the image.
- ▶ **virt-sysprep** - It is also recommended that the `virt-sysprep` command is run on Linux-based virtual machine images prior to uploading them to the Image service. The `virt-sysprep` command re-initializes a disk image in preparation for use in a virtual environment.

Default operations include the 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 7.1. 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>.

Although 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. The following example displays the syntax for a basic TDL file.

#### Example 7.1. TDL File

The template below can be used to create a Red Hat Enterprise Linux 6 disk image. In particular, note 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/user/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` command to build an image:

```
# oz-install -u -d3 TDL_FILE
```

Syntax:

- `-u` ensures any required customization changes to the image are applied after guest operating installation.
- `-d3` enables the display of errors, warnings, and informational messages.

- `TDL_FILE` provides the path to your TDL file.

By default, Oz stores 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 the Image service. 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 the Image service.

### 7.6.3. Uploading a Disk Image

[Report a bug](#)

To launch instances based on images stored in the Image service, you must first upload one or more images into the Image service.

To carry out this procedure, you must already have created or downloaded images suitable for use in the OpenStack environment (refer to the *Obtaining an Image* and *Building an Image* sections).



#### Important

It is recommended that the `virt-sysprep` command be run on all Linux-based virtual machine images prior to uploading them to the Image service. The `virt-sysprep` command re-initializes a disk image in preparation for use in a virtual environment. Default operations include the removal of SSH keys, removal of persistent MAC addresses, and removal of user accounts.

The `virt-sysprep` command is provided by the RHEL `libguestfs-tools` package. As the root user, execute:

```
# yum install -y libguestfs-tools
# virt-sysprep --add FILE
```

For information on enabling and disabling specific operations, refer to the command's manual page by executing:

```
# man virt-sysprep
```

To upload an image to the Image service:

1. Set the environment variables used for authenticating with the Identity service by loading them from the `keystonerc` file associated with your user (an administrative account is not required):

```
# source ~/keystonerc_UserName
```

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
```

Where:

- ▶ *NAME* = The name by which users will refer to the disk image.
- ▶ *IS\_PUBLIC* = Either true or false:
  - true - All users will be able to view and use the image.
  - false - Only administrators will be able to view and use the image.
- ▶ *DISK\_FORMAT* = The disk image's format. Valid values include: aki, ami, ari, iso, qcow2, raw, vdi, vhd, and vmdk.

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

#### **Example 7.2. Using `qemu-img info`**

In the following 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
```

- ▶ *CONTAINER\_FORMAT* = 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.
- ▶ *IMAGE* = The local path to the image file (for uploading).

For more information about the `glance image-create` syntax, execute:

```
# glance help image-create
```



### Note

If the image being uploaded is not locally accessible but is available using a remote URL, provide the URL using the `--location` parameter instead of using the `--file` parameter.

However, unless you also specify the `--copy-from` argument, the Image service will not copy the image into the object store. Instead, the image will be accessed remotely each time it is required.

**Example 7.3. Uploading an Image to the Image service**

In this example the qcow2 format image in the file named `rhel-64.qcow2` is uploaded to the Image service. It is created in the service 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

- To verify that your image was successfully uploaded, use the `glance image-list` command:

```
# glance image-list
```

ID	Name	Disk Format	Container Format	Size
0ce782c6-...	RHEL 6.4	qcow2	bare	213581824

To view detailed information about an uploaded image, execute the `glance image-show` command using the image's identifier:

```
# 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 the Image service. This disk image can now be used as the basis for launching virtual machine instances in your OpenStack environment.

**See Also:**

- ▶ [Section 7.6.1, “Obtaining a Test Disk Image”](#)
- ▶ [Section 7.6.2, “Building a Custom Disk Image”](#)



# Chapter 8. Installing OpenStack Block Storage

## 8.1. Block Storage Installation Overview

Block storage functionality is provided in OpenStack by three separate services collectively referred to as the block storage service or cinder. The three services are:

### The API service (`openstack-cinder-api`)

The API service provides a HTTP endpoint for block storage requests. When an incoming request is received the API verifies identity requirements are met and translates the request into a message denoting the required block storage actions. The message is then sent to the message broker for processing by the other block storage services.

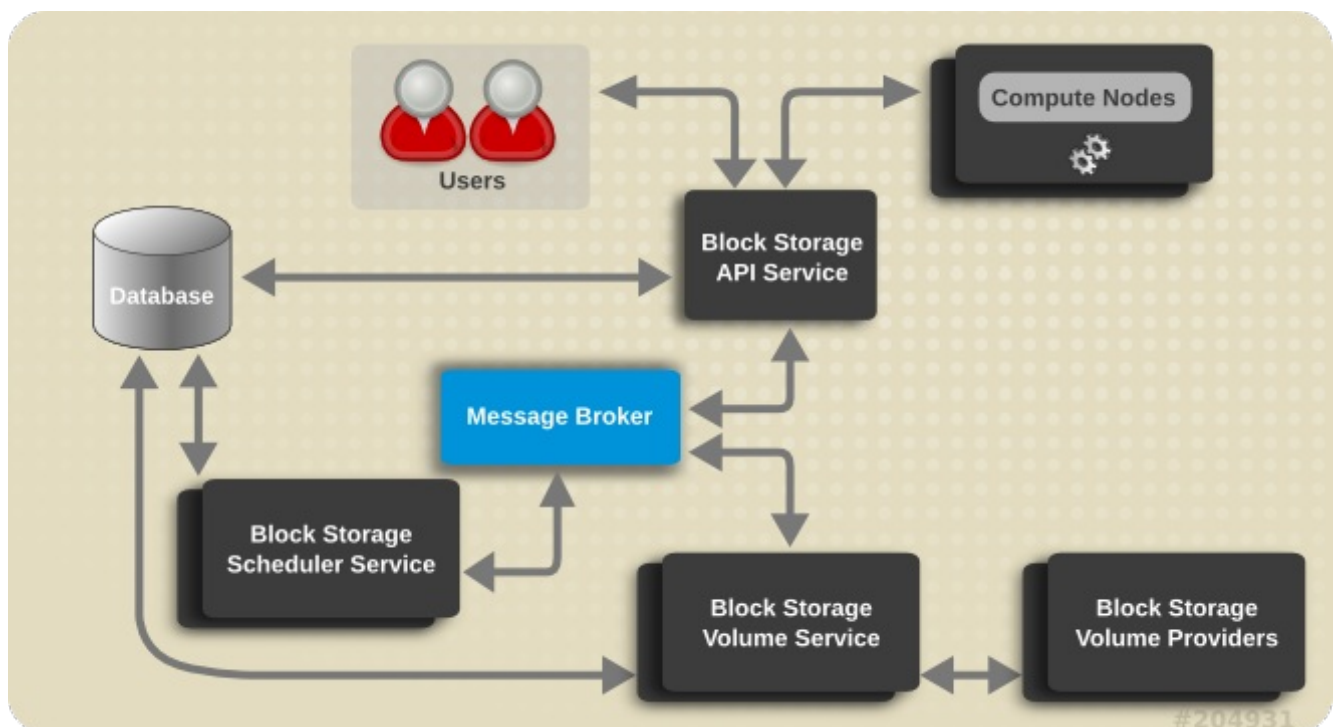
### The scheduler service (`openstack-cinder-scheduler`)

The scheduler service reads requests from the message queue and determines on which block storage host the request must be actioned. The scheduler then communicates with the volume service on the selected host to process the request.

### The volume service (`openstack-cinder-volume`)

The volume service manages the interaction with the block storage devices. As requests come in from the scheduler, the volume service creates, modifies, and removes volumes as required.

Although the three services can be co-located in a production environment, it is more common to deploy many instances of the volume service with one or more instances of the API and scheduler services managing them.



**Figure 8.1. Block Storage Architecture**

To aid in the creation of such deployments the installation of the block storage service has been broken down into:

### Preparing for Block Storage Installation

Steps that must be performed before installing any of the block storage services. These procedures include the creation of identity records, the database, and a database user.

### Common Block Storage Configuration

Steps that are common to all of the block storage services and as such must be performed on all block storage nodes in the environment. These procedures include configuring the services to refer to the correct database and message broker. Additionally they include the initialization and population of the database which must only be performed once but can be performed from any of the block storage systems.

### Volume Service Specific Configuration

Steps that are specific to systems that will be hosting the volume service and as such require direct access to block storage devices.

As well as these configuration tasks, steps to start the services and validate the resulting deployment are also provided.

#### See Also:

- ▶ [Section 8.2, “Block Storage Prerequisite Configuration”](#)
- ▶ [Section 8.3, “Common Block Storage Configuration”](#)
- ▶ [Section 8.4, “Volume Service Specific Configuration”](#)
- ▶ [Section 8.5, “Starting the Block Storage Services”](#)

[Report a bug](#)

## 8.2. Block Storage Prerequisite Configuration

### 8.2.1. Creating the Block Storage Database

In this procedure the database and database user that will be used by the block storage services will be created. These steps must be performed while logged in to the database server as the root user.

1. Connect to the database service using the `mysql` command.

```
# mysql -u root -p
```

2. Create the cinder database.

```
mysql> CREATE DATABASE cinder;
```

3. Create a cinder database user and grant it access to the cinder database.

```
mysql> GRANT ALL ON cinder.* TO 'cinder'@'%' IDENTIFIED BY 'PASSWORD';
```

```
mysql> GRANT ALL ON cinder.* TO 'cinder'@'localhost' IDENTIFIED BY 'PASSWORD';
```

Replace `PASSWORD` with a secure password that will be used to authenticate with the

database server as this user.

4. Flush the database privileges to ensure that they take effect immediately.

```
mysql> FLUSH PRIVILEGES;
```

5. Exit the mysql client command.

```
mysql> quit
```

The block storage database has been created. The database will be populated during service configuration.

## 8.2.2. Creating the Block Storage Identity Records

[Report a bug](#)

The steps outlined in this procedure cover the creation of identity records to support the Block Storage service:

1. Create the cinder user, who has the admin role in the services tenant.
2. Create the cinder service entry and assign it an endpoint.

These entries provided authentication for the block storage services and guide other OpenStack services attempting to locate and access the volume functionality it provides.

1. Authenticate as the administrator of the identity service by running the source command on the keystone\_admin file containing the required credentials.

```
# source ~/keystone_admin
```

2. Create a user named cinder for the block storage service to use.

```
# keystone user-create --name cinder --pass PASSWORD
+-----+-----+
| Property | Value |
+-----+-----+
| email | |
| enabled | True |
| id | e1765f70da1b4432b54ced060139b46a |
| name | cinder |
| tenantId | |
+-----+-----+
```

Replace *PASSWORD* with a secure password that will be used by the block storage service when authenticating with the identity service. Take note of the created user's returned ID as it will be used in subsequent steps.

3. Get the ID of the admin role:

```
# keystone role-get admin
```

If no admin role exists, create one:

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

4. Get the ID of the services tenant:

```
$ keystone tenant-list | grep services
```

If no services tenant exists, create one:

```
$ keystone tenant-create --name services --description "Services Tenant"
```

This guide uses one tenant for all service users. For more information, refer to *Creating*

the *Services Tenant*.

- Use the keystone `user-role-add` command to link the cinder user, admin role, and services tenant together:

```
# keystone user-role-add --user-id USERID --role-id ROLEID --tenant-id
TENANTID
```

Replace the user, role, and tenant IDs with those obtained in the previous steps.

- Create the cinder service entry:

```
# keystone service-create --name cinder \
    --type volume \
    --description "Cinder Volume Service"
+-----+-----+
| Property | Value |
+-----+-----+
| description | Cinder Volume Service |
| id | dfde7878671e484c9e581a3eb9b63e66 |
| name | cinder |
| type | volume |
+-----+-----+
```

Take note of the created service's returned ID as it will be used in the next step.

- Create the cinder endpoint entry.

```
# keystone endpoint-create --service-id SERVICEID \
    --publicurl "http://IP:8776/v1/\$(tenant_id)s" \
    --adminurl "http://IP:8776/v1/\$(tenant_id)s" \
    --internalurl "http://IP:8776/v1/\$(tenant_id)s"
```

Replace *SERVICEID* with the identifier returned by the keystone `service-create` command. Replace *IP* with the IP address or host name of the system that will be hosting the block storage service API (`openstack-cinder-api`).



### Important

If you intend to install and run multiple instances of the API service then you must repeat this step for the IP address or host name of each instance.

All supporting identity service entries required by the block storage services have been created.

[Report a bug](#)

## 8.3. Common Block Storage Configuration

### 8.3.1. Installing the Block Storage Service Packages

The OpenStack Block Storage service requires the following packages:

#### ***openstack-cinder***

Provides the block storage services and associated configuration files.

#### ***openstack-utils***

Provides supporting utilities to assist with a number of tasks including the editing of configuration files.

#### ***openstack-selinux***

Provides OpenStack specific SELinux policy modules.

To install all of the above packages, execute the following command while logged in as the root user:

```
# yum install -y openstack-cinder openstack-utils openstack-selinux
```

The Block Storage services are now installed and ready to be configured.

### 8.3.2. Configuring Authentication

[Report a bug](#)

The block storage service must be explicitly configured to use the identity service for authentication. Follow the steps listed in this procedure to configure this.

All steps listed in this procedure must be performed on each system hosting block storage services while logged in as the root user.

1. Set the authentication strategy (`auth_strategy`) configuration key to `keystone` using the `openstack-config` command.

```
# openstack-config --set /etc/cinder/cinder.conf \
  DEFAULT auth_strategy keystone
```

2. Set the authentication host (`auth_host`) configuration key to the IP address or host name of the identity server.

```
# openstack-config --set /etc/cinder/cinder.conf \
  keystone_auth token auth_host IP
```

Replace *IP* with the IP address or host name of the identity server.

3. Set the administration tenant name (`admin_tenant_name`) configuration key to the name of the tenant that was created for the use of the block storage service. In this guide, examples use *services*.

```
# openstack-config --set /etc/cinder/cinder.conf \
  keystone_auth token admin_tenant_name services
```

4. Set the administration user name (`admin_user`) configuration key to the name of the user that was created for the use of the block storage service. In this guide, examples use *cinder*.

```
# openstack-config --set /etc/cinder/cinder.conf \
  keystone_auth token admin_user cinder
```

5. Set the administration password (`admin_password`) configuration key to the password that is associated with the user specified in the previous step.

```
# openstack-config --set /etc/cinder/cinder.conf \
  keystone_auth token admin_password PASSWORD
```

The authentication keys used by the block storage services have been set and will be used when the services are started.

### 8.3.3. Setting the Message Broker

[Report a bug](#)

The block storage services must be explicitly configured with the type, location, and authentication details of the message broker.

All steps in this procedure must be run on each system that will be hosting block storage

services while logged in as the root user.

### 1. General Settings

Use the `openstack-config` utility to set the value of the `rpc_backend` configuration key to Qpid.

```
# openstack-config --set /etc/cinder/cinder.conf \  
DEFAULT rpc_backend cinder.openstack.common.rpc.impl_qpid
```

2. Use the `openstack-config` utility to set the value of the `qpid_hostname` configuration key to the host name of the Qpid server.

```
# openstack-config --set /etc/cinder/cinder.conf \  
DEFAULT qpid_hostname IP
```

Replace `IP` with the IP address or host name of the message broker.

### 3. Authentication Settings

If you have configured Qpid to authenticate incoming connections then you must provide the details of a valid Qpid user in the block storage configuration.

- a. Use the `openstack-config` utility to set the value of the `qpid_username` configuration key to the username of the Qpid user that the block storage services must use when communicating with the message broker.

```
# openstack-config --set /etc/cinder/cinder.conf \  
DEFAULT qpid_username USERNAME
```

Replace `USERNAME` with the required Qpid user name.

- b. Use the `openstack-config` utility to set the value of the `qpid_password` configuration key to the password of the Qpid user that the block storage services must use when communicating with the message broker.

```
# openstack-config --set /etc/cinder/cinder.conf \  
DEFAULT qpid_password PASSWORD
```

Replace `PASSWORD` with the password of the Qpid user.

### 4. Encryption Settings

If you configured Qpid to use SSL then you must inform the block storage services of this choice. Use `openstack-config` utility to set the value of the `qpid_protocol` configuration key to `ssl`.

```
# openstack-config --set /etc/cinder/cinder.conf \  
DEFAULT qpid_protocol ssl
```

The value of the `qpid_port` configuration key must be set to 5671 as Qpid listens on this different port when SSL is in use.

```
# openstack-config --set /etc/cinder/cinder.conf \  
DEFAULT qpid_port 5671
```



## Important

To communicate with a Qpid message broker that uses SSL the node must also have:

- ▶ The *nss* package installed.
- ▶ The certificate of the relevant certificate authority installed in the system NSS database (`/etc/pki/nssdb/`).

The `certtool` command is able to import certificates into the NSS database. See the `certtool` manual page for more information (`man certtool`).

The block storage services have been configured to use the message broker and any authentication schemes that it presents.

### 8.3.4. Setting the Database Connection String

[Report a bug](#)

The database connection string used by the block storage services (the value of the `sql_connection` configuration key) is defined in the `/etc/cinder/cinder.conf` file. The string must be updated to point to a valid database server before starting the service.

The following command must be executed as the root user on each system hosting block storage services:

```
# openstack-config --set /etc/cinder/cinder.conf \
  DEFAULT sql_connection mysql://USER:PASS@IP/DB
```

Replace:

- ▶ *USER* with the database user name the block storage services are to use, usually `cinder`.
- ▶ *PASS* with the password of the chosen database user.
- ▶ *IP* with the IP address or host name of the database server.
- ▶ *DB* with the name of the database that has been created for use by the block storage services, usually `cinder`.

The database connection string has been set and will be used by the block storage services.

### 8.3.5. Configuring the Firewall

[Report a bug](#)

Systems attempting to use the functionality provided by the block storage services access it over the network using ports 3260 and 8776.

To allow this the firewall on the system hosting the block storage service must be altered to allow network traffic on these ports. All steps in this procedure must be run on each system hosting block storage services while logged in as the root user.

1. Open the `/etc/sysconfig/iptables` file in a text editor.
2. Add an INPUT rule allowing TCP traffic on ports 3260 and 8776 to the file. The new rule must appear before any INPUT rules that REJECT traffic.

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

3. Save the changes to the `/etc/sysconfig/iptables` file.
4. Restart the `iptables` service to ensure that the change takes effect.

```
# service iptables restart
```

The `iptables` firewall is now configured to allow incoming connections to the block storage

service on ports 3260 and 8776.

### 8.3.6. Populating the Block Storage Database

[Report a bug](#)

This procedure describes the steps required to populate the block storage service database. These steps must be performed while logged in to a system hosting one of the block storage services. The database connection string must already be defined in the configuration of the service.



#### Important

This procedure only needs to be followed once to initialize and populate the database. You do not need to perform these steps again when adding additional systems hosting block storage services.

1. Use the `su` command to switch to the `cinder` user.

```
# su cinder -s /bin/sh
```

2. Run the `cinder-manage db sync` command to initialize and populate the database identified in `/etc/cinder/cinder.conf`.

```
$ cinder-manage db sync
```

The block storage service database has been initialized and populated.

[Report a bug](#)

## 8.4. Volume Service Specific Configuration

### 8.4.1. Block Storage Driver Support

The volume service (`openstack-cinder-volume`) requires access to suitable block storage. The service includes volume drivers for a number of block storage providers. Supported drivers for LVM, NFS, and Red Hat Storage are included.

### 8.4.2. Configuring for LVM Storage Backend

[Report a bug](#)

The volume service is able to make use of a volume group attached directly to the server on which the service runs. This volume group must be created exclusively for use by the block storage service and the configuration updated to point to the name of the volume group.

The following steps must be performed while logged into the system hosting the volume service as the root user:

1. Use the `pvcreate` command to create a physical volume.

```
# pvcreate DEVICE
Physical volume "DEVICE" successfully created
```

Replace `DEVICE` with the path to a valid, unused, device. For example:

```
# pvcreate /dev/sdX
```

2. Use the `vgcreate` command to create a volume group.

```
# vgcreate cinder-volumes DEVICE
Volume group "cinder-volumes" successfully created
```

Replace `DEVICE` with the path to the device used when creating the physical volume.



Optionally replace `cinder-volumes` with an alternative name for the new volume group.

3. Set the `volume_group` configuration key to the name of the newly created volume group.

```
# openstack-config --set /etc/cinder/cinder.conf \
DEFAULT volume_group cinder-volumes
```

The name provided must match the name of the volume group created in the previous step.

4. Ensure that the correct volume driver for accessing LVM storage is in use by setting the `volume_driver` configuration key to `cinder.volume.drivers.lvm.LVMISCSIDriver`.

```
# openstack-config --set /etc/cinder/cinder.conf \
DEFAULT volume_driver cinder.volume.drivers.lvm.LVMISCSIDriver
```

The volume service has been configured to use LVM storage.

### 8.4.3. Configuring for NFS Storage Backend

[Report a bug](#)

This procedure documents the steps involved in configuring the volume service to use NFS storage. The NFS shares to be used must already exist and be accessible from the server hosting the volume service.

All steps listed in this procedure must be performed while logged into the system hosting the volume service as the root user.



#### Important

To access instance volumes on NFS shares, SELinux requires the `virt_use_nfs` Boolean enabled on any client machine accessing the instance volumes. This includes all compute nodes. Run the following command on each client machine as the root user to enable the Boolean and make it persistent across reboots:

```
# setsebool -P virt_use_nfs on
```

1. Create a text file in the `/etc/cinder/` directory containing a list of the NFS shares that the volume service is to use for backing storage.

```
nfs1.example.com:/export
nfs2.example.com:/export
```

Each line must contain an NFS share in the format `HOST:/SHARE` where `HOST` is replaced by the IP address or host name of the NFS server and `SHARE` is replaced with the particular NFS share to be used.

2. Use the `chown` command to set the file to be owned by the root user and the `cinder` group.

```
# chown root:cinder FILE
```

Replace `FILE` with the path to the file containing the list of NFS shares.

3. Use the `chmod` command to set the file permissions such that it can be read by members of the `cinder` group.

```
# chmod 0640 FILE
```

Replace `FILE` with the path to the file containing the list of NFS shares.

4. Set the value of the `nfs_shares_config` configuration key to the path of the file

containing the list of NFS shares.

```
# openstack-config --set /etc/cinder/cinder.conf \
  DEFAULT nfs_shares_config FILE
```

Replace *FILE* with the path to the file containing the list of NFS shares.

- The `nfs_sparsed_volumes` configuration key determines whether volumes are created as sparse files and grown as needed or fully allocated up front. The default and recommended value is `true`, which ensures volumes are initially created as sparse files. Setting the `nfs_sparsed_volumes` configuration key to `false` will result in volumes being fully allocated at the time of creation. This leads to increased delays in volume creation.

```
# openstack-config --set /etc/cinder/cinder.conf \
  DEFAULT nfs_sparsed_volumes true
```

- Optionally, provide any additional NFS mount options required in your environment in the `nfs_mount_options` configuration key. If your NFS shares do not require any additional mount options or you are unsure then skip this step.

```
# openstack-config --set /etc/cinder/cinder.conf \
  DEFAULT nfs_mount_options OPTIONS
```

Replace *OPTIONS* with the mount options to be used when accessing NFS shares. See the manual page for NFS for more information on available mount options (`man nfs`).

- Ensure that the correct volume driver for accessing NFS storage is in use by setting the `volume_driver` configuration key to `cinder.volume.drivers.nfs.NfsDriver`.

```
# openstack-config --set /etc/cinder/cinder.conf \
  DEFAULT volume_driver cinder.volume.drivers.nfs.NfsDriver
```

The volume service has been configured to use NFS storage.

#### 8.4.4. Configuring for Red Hat Storage Backend

[Report a bug](#)

This procedure documents the steps involved in configuring the volume service to use Red Hat Storage. The Red Hat Storage shares to be used must already exist and be accessible from the server hosting the volume service.

All steps listed in this procedure must be performed while logged into the system hosting the volume service as the root user.



#### Important

Mounting of Red Hat Storage volumes requires utilities and libraries from the *glusterfs-fuse* package. For information on enabling the necessary Red Hat Network channels and installing the *glusterfs-fuse* package refer to the *Red Hat Storage Administration Guide*. This package must be installed on all systems that will access volumes backed by Red Hat Storage.



## Important

To access instance Red Hat Storage volume, SELinux requires that the `virt_use_fusefs` Boolean is enabled on any client machine accessing the instance volumes. This includes all compute nodes. Run the following command on each client machine as the root user to enable the Boolean and make it persistent across reboots:

```
# setsebool -P virt_use_fusefs on
```

1. Create a text file in the `/etc/cinder/` directory containing a list of the Red Hat Storage shares that the volume service is to use for backing storage.

```
HOST:/VOLUME
```

Each line must contain a Red Hat Storage share in the format `HOST:VOLUME` where `HOST` is replaced by the IP address or host name of the Red Hat Storage server and `VOLUME` is replaced with the name of a particular volume that exists on that host.

If required additional mount options must also be added in the same way that they would be provided to the mount command line tool:

```
HOST:/VOLUME -o OPTIONS
```

Replace `OPTIONS` with a comma separated list of mount options.

2. Use the `chown` command to set the file to be owned by the root user and the `cinder` group.

```
# chown root:cinder FILE
```

Replace `FILE` with the path to the file containing the list of Red Hat Storage shares.

3. Use the `chmod` command to set the file permissions such that it can be read by members of the `cinder` group.

```
# chmod 0640 FILE
```

Replace `FILE` with the path to the file containing the list of Red Hat Storage shares.

4. Set the value of the `glusterfs_shares_config` configuration key to the path of the file containing the list of Red Hat Storage shares.

```
# openstack-config --set /etc/cinder/cinder.conf \
  DEFAULT glusterfs_shares_config FILE
```

Replace `FILE` with the path to the file containing the list of Red Hat Storage shares.

5. The `glusterfs_sparsed_volumes` configuration key determines whether volumes are created as sparse files and grown as needed or fully allocated up front. The default and recommended value is `true`, which ensures volumes are initially created as sparse files.

Setting the `glusterfs_sparsed_volumes` configuration key to `false` will result in volumes being fully allocated at the time of creation. This leads to increased delays in volume creation.

```
# openstack-config --set /etc/cinder/cinder.conf \
  DEFAULT glusterfs_sparsed_volumes true
```

6. Ensure that the correct volume driver for accessing Red Hat Storage is in use by setting the `volume_driver` configuration key to `cinder.volume.drivers.glusterfs`.

```
# openstack-config --set /etc/cinder/cinder.conf \
DEFAULT volume_driver cinder.volume.drivers.glusterfs.GlusterfsDriver
```

The volume service has been configured to use Red Hat Storage.

### 8.4.5. Configuring for Multiple Storage Backends

[Report a bug](#)

In previous OpenStack releases each system hosting an instance of the volume service was only able to access a single storage backend or volume. In Red Hat Enterprise Linux OpenStack Platform 3 and later this restriction no longer applies and the volume service can optionally be configured to access multiple storage backends. The scheduler then ensures that a unique instance of the volume service is launched to cater for each backend. These instances are able to run in parallel on the same system.

Follow the steps outlined in this procedure on a system hosting the volume service while logged in as the root user. Each system that needs to be able to access multiple storage drivers or volumes must be configured in this way.

1. Open the `/etc/cinder/cinder.conf` configuration file in a text editor.
2. Add a configuration block for each storage driver or volume. This configuration block must have a unique name (avoid spaces or special characters) and contain values for at least these configuration keys:

`volume_group`

A volume group name. This is the name of the volume group that will be accessed by the driver.

`volume_driver`

A volume driver. This is the name of the driver that will be used when accessing the volume group.

`volume_backend_name`

A backend name. This is an administrator-defined name for the backend, which groups the drivers so that user requests for storage served from the given backend can be serviced by any driver in the group. It is not related to the name of the configuration group which must be unique.

Any additional driver specific configuration must also be included in the configuration block.

```
[NAME]
volume_group=GROUP
volume_driver=DRIVER
volume_backend_name=BACKEND
```

Replace *NAME* with a unique name for the backend and replace *GROUP* with the unique name of the applicable volume group. Replace *DRIVER* with the driver to use when accessing this storage backend, valid values include:

- ▶ `cinder.volume.drivers.lvm.LVMISCSIDriver` for LVM and iSCSI storage.
- ▶ `cinder.volume.drivers.nfs.NfsDriver` for NFS storage.
- ▶ `cinder.volume.drivers.glusterfs.GlusterfsDriver` for Red Hat Storage.

Finally replace *BACKEND* with a name for the storage backend.

3. Update the value of the `enabled_backends` configuration key in the `DEFAULT` configuration block. This configuration key must contain a comma separated list containing the names of the configuration blocks for each storage driver.

**Example 8.1. Multiple Backend Configuration**

In this example two logical volume groups, `cinder-volumes-1` and `cinder-volumes-2`, are grouped into the storage backend named LVM. An additional volume, backed by a list of NFS shares, is grouped into a storage backend named NFS.

```
[DEFAULT]
...
enabled_backends=cinder-volumes-1-driver,cinder-volumes-2-driver,cinder-
volumes-3-driver
...
[cinder-volumes-1-driver]
volume_group=cinder-volumes-1
volume_driver=cinder.volume.drivers.lvm.LVMISCSIDriver
volume_backend_name=LVM
[cinder-volumes-2-driver]
volume_group=cinder-volumes-2
volume_driver=cinder.volume.drivers.lvm.LVMISCSIDriver
volume_backend_name=LVM
[cinder-volumes-3-driver]
nfs_shares_config=/etc/cinder/shares.txt
volume_driver=cinder.volume.drivers.nfs.NfsDriver
volume_backend_name=NFS
```

**Important**

The default block storage scheduler driver in Red Hat Enterprise Linux OpenStack Platform 3 is the filter scheduler. If you have changed the value of the `scheduler_driver` configuration key on any of your block storage nodes then you must update the value to `cinder.scheduler.filter_scheduler.FilterScheduler` for the multiple storage backends feature to function correctly.

4. Save the changes to the `/etc/cinder/cinder.conf` file.

The volume service is now configured to use multiple storage backends when started. Note that if the `openstack-cinder-volume` service has already been started then you must restart it for the changes to take effect.

**Note**

To allow users to specify which backend their volumes are created on rather than relying on the scheduler a volume type must be defined in the database. Once the services are operational this can be done using these commands:

```
# source ~/keystonerc_admin
# cinder type-create TYPE
# cinder type-key TYPE set volume_backend_name=BACKEND
```

Replace `TYPE` with the name that users must provide to select this specific storage backend and replace a `BACKEND` with the relevant `volume_backend_name` as set in the `/etc/cinder/cinder.conf` configuration file.

**8.4.6. Configuring tgt**

[Report a bug](#)

The volume storage service makes use of the SCSI target daemon, `tgtd`, when mounting

storage. To support this the `tgt` service must be configured to read additional configuration files.

The steps listed in this procedure must be performed on each system hosting an instance of the volume service while logged in as the root user.

1. Open the `/etc/tgt/targets.conf` file.
2. Add this line to the file:

```
include /etc/cinder/volumes/*
```

3. Save the changes to the file.

When the `tgt` service is started it will be configured to support the volume service.

[Report a bug](#)

## 8.5. Starting the Block Storage Services

To bring up the block storage functionality at least one instance of each of the three services must be started:

- ▶ The API service (`openstack-cinder-api`).
- ▶ The scheduler service (`openstack-cinder-scheduler`).
- ▶ The volume service (`openstack-cinder-volume`).

The services do not need to be located on the same system, but must be configured to communicate using the same message broker and database instance. Once the services are running, the API will accept incoming volume requests, the scheduler will assign them as appropriate, and the volume service will action them.

### 1. Starting the API Service

Log in to each server that you intend to run the API on as the root user and start the API service.

- a. Use the `service` command to start the API service (`openstack-cinder-api`).

```
# service openstack-cinder-api start
```

- b. Use the `chkconfig` command to enable the API service permanently (`openstack-cinder-api`).

```
# chkconfig openstack-cinder-api on
```

### 2. Starting the Scheduler Service

Log in to each server that you intend to run the scheduler on as the root user and start the scheduler service.

- a. Use the `service` command to start the scheduler (`openstack-cinder-scheduler`).

```
# service openstack-cinder-scheduler start
```

- b. Use the `chkconfig` command to enable the scheduler permanently (`openstack-cinder-scheduler`).

```
# chkconfig openstack-cinder-scheduler on
```

### 3. Starting the Volume Service

Log in to each server that block storage has been attached to as the root user and start the volume service.

- a. Use the `service` command to start the volume service (`openstack-cinder-volume`).

```
# service openstack-cinder-volume start
```

- b. Use the service command to start the The SCSI target daemon (tgttd).

```
# service tgttd start
```

- c. Use the chkconfig command to enable the volume service permanently (openstack-cinder-volume).

```
# chkconfig openstack-cinder-volume on
```

- d. Use the chkconfig command to enable the SCSI target daemon permanently (tgttd).

```
# chkconfig tgttd on
```

The volume service is running and ready to begin allocating volumes as they are requested.

[Report a bug](#)

## 8.6. Validating the Block Storage Service Installation

This procedure lists steps for validating that the block storage installation is complete and ready for use.

### 1. Testing Locally

The steps outlined in this section of the procedure must be performed while logged in to the server hosting the block storage API service as the root user or a user with access to a keystone\_admin file containing the credentials of the OpenStack administrator. Transfer the keystone\_admin file to the system before proceeding.

- a. Run the source command on the keystone\_admin file to populate the environment variables used for identifying and authenticating the user.

```
# source ~/keystone_admin
```

- b. Run the cinder list command and verify that no errors are returned.

```
# cinder list
```

- c. Run the cinder create command to create a volume.

```
# cinder create SIZE
```

Replace *SIZE* with the size of the volume to create in Gigabytes (GB).

- d. Run the cinder delete command to remove the volume.

```
# cinder delete ID
```

Replace *ID* with the identifier returned when the volume was created.

### 2. Testing Remotely

The steps outlined in this section of the procedure must be performed while logged in to a system other than the server hosting the block storage API service. Transfer the keystone\_admin file to the system before proceeding.

- a. Install the *python-cinderclient* package using the yum command. You will need to authenticate as the root user for this step.

```
# yum install -y python-cinderclient
```

- b. Run the source command on the keystone\_admin file to populate the environment variables used for identifying and authenticating the user.

```
$ source ~/keystonerc_admin
```

- c. Run the `cinder list` command and verify that no errors are returned.

```
$ cinder list
```

- d. Run the `cinder create` command to create a volume.

```
$ cinder create SIZE
```

Replace *SIZE* with the size of the volume to create in Gigabytes (GB).

- e. Run the `cinder delete` command to remove the volume.

```
$ cinder delete ID
```

Replace *ID* with the identifier returned when the volume was created.

Basic validation of the block storage service installation has been performed.



# Chapter 9. Installing the OpenStack Networking Service

## 9.1. OpenStack Networking Installation Overview

### 9.1.1. 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 machine 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 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.

### 9.1.2. 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 the above capabilities to tenants directly, the OpenStack Networking API can be very useful for administrative purposes. The API

provides significantly more flexibility for the cloud administrator when customizing network offerings.

### 9.1.3. OpenStack Networking API Extensions

[Report a bug](#)

The OpenStack networking API allows plug-ins to provide extensions to enable additional networking functional not available in the core API itself.

#### Provider Networks

Provider networks allow the creation of virtual networks that map directly to networks in the physical data center. This allows the administrator to give tenants direct access to a public network such as the Internet or to integrate with existing VLANs in the physical networking environment that have a defined meaning or purpose.

When the provider extension is enabled OpenStack networking users with administrative privileges are able to see additional provider attributes on all virtual networks. In addition such users have the ability to specify provider attributes when creating new provider networks.

Both the Open vSwitch and Linux Bridge plug-ins support the provider networks extension.

#### Layer 3 (L3) Routing and Network Address Translation (NAT)

The L3 routing API extensions provides abstract L3 routers that API users are able to dynamically provision and configure. These routers are able to connect to one or more Layer 2 (L2) OpenStack networking controlled networks. Additionally the routers are able to provide a gateway that connects one or more private L2 networks to an common public or external network such as the Internet.

The L3 router provides basic NAT capabilities on gateway ports that connect the router to external networks. The router supports floating IP addresses which give a static mapping between a public IP address on the external network and the private IP address on one of the L2 networks attached to the router.

This allows the selective exposure of compute instances to systems on an external public network. Floating IP addresses are also able to be reallocated to different OpenStack networking ports as necessary.

#### Security Groups

Security groups and security group rules allow the specification of the specific type and direction of network traffic that is allowed to pass through a given network port. This provides an additional layer of security over and above any firewall rules that exist within a compute instance. The security group is a container object which can contain one or more security rules. A single security group can be shared by multiple compute instances.

When a port is created using OpenStack networking it is associated with a security group. If a specific security group was not specified then the port is associated with the default security group. By default this group will drop all inbound traffic and allow all outbound traffic. Additional security rules can be added to the default security group to modify its behaviour or new security groups can be created as necessary.

The Open vSwitch, Linux Bridge, Nicira NVP, NEC, and Ryu networking plug-ins currently support security groups.

### 9.1.4. OpenStack Networking Plug-ins

[Report a bug](#)

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 Enterprise Linux OpenStack Platform:

- ▶ 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-nec*)
- ▶ 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.

### 9.1.5. OpenStack Networking Agents

[Report a bug](#)

As well as the OpenStack Networking service and the installed plug-in a number of components combine to provide networking functionality in the OpenStack environment.

#### 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.

#### DHCP Agent

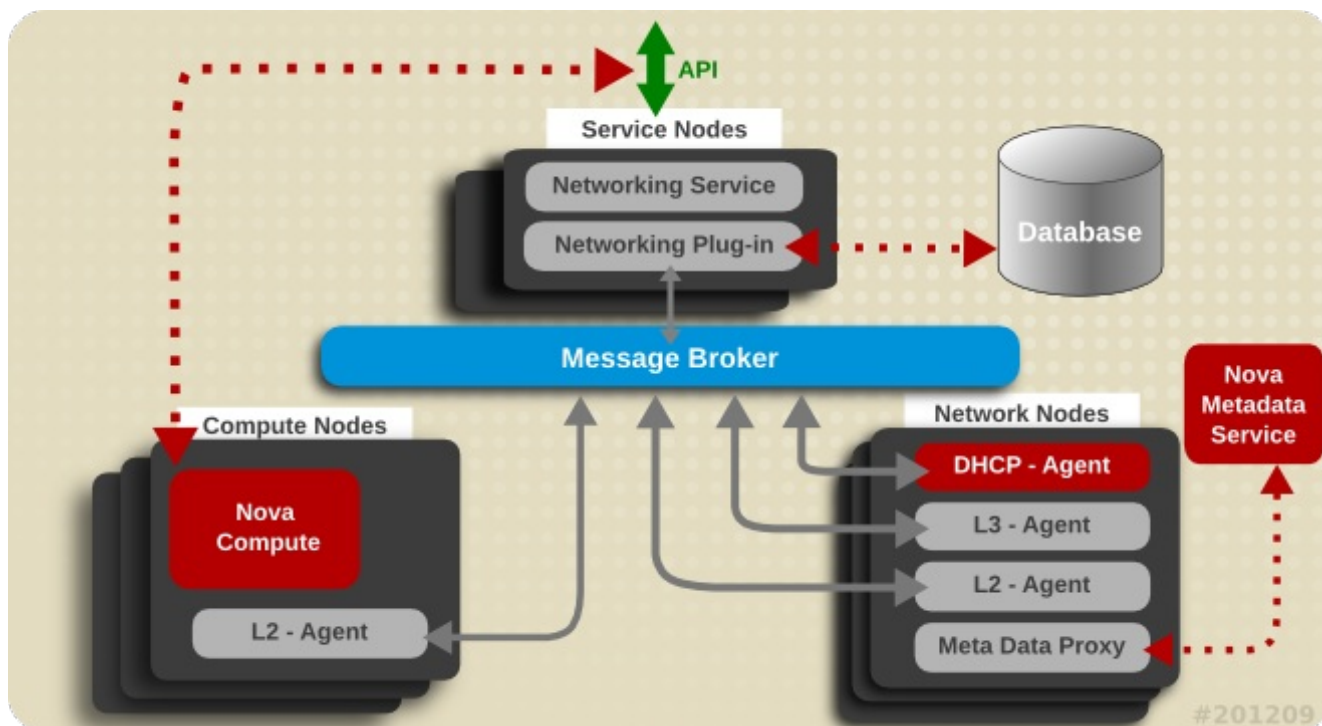
The OpenStack Networking DHCP agent is capable of allocating IP addresses to virtual machines running on the network. If the agent is enabled and running when a subnet is created then by default that subnet has DHCP enabled.

#### Plug-in Agent

Many of the OpenStack Networking plug-ins, including Open vSwitch and Linux Bridge, 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*.

### 9.1.6. Recommended Networking Deployment

OpenStack Networking provides an extreme amount of flexibility when deploying networking in support of a compute environment. As a result, the exact layout of a deployment will depend on a combination of expected workloads, expected scale, and available hardware.



**Figure 9.1. Deployment Architecture**

For demonstration purposes, this chapter concentrates on a networking deployment that consists of these types of nodes:

#### Service Node

The service node exposes the networking API to clients and handles incoming requests before forwarding them to a message queue to be actioned by the other nodes. The service node hosts both the networking service itself and the active networking plug-in.

In environments that use *controller nodes* to host the client-facing APIs and schedulers for all services, the controller node would also fulfil the role of service node as it is applied in this chapter.

#### Network Node

The network node handles the majority of the networking workload. It hosts the DHCP agent, the Layer 3 (L3) agent, the Layer 2 (L2) Agent, and the metadata proxy. In addition to plug-ins that require an agent, it runs an instance of the plug-in agent (as do all other systems that handle data packets in an environment where such plug-ins are in use). Both the Open vSwitch and Linux Bridge plug-ins include an agent.

#### Compute Node

The compute hosts the compute instances themselves. To connect compute instances to the networking services, compute nodes must also run the L2 agent. Like all other systems that handle data packets it must also run an instance of the plug-in agent.

This deployment type and division of responsibilities is made only as a suggestion. Other divisions are equally valid, in particular in some environments the network and service nodes may be combined.



## Warning

Environments that have been configured to use Compute 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 Compute networking. If you wish to proceed with such a conversion, 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
```



## Important

When running 2 or more active controller nodes, do not run `nova-consoleauth` on more than one node. Running more than one instance of `nova-consoleauth` causes a conflict between nodes with regard to token requests which may cause errors.

### See Also:

- ▶ [Section 9.2, “Networking Prerequisite Configuration”](#)
- ▶ [Section 9.3, “Common Networking Configuration”](#)
- ▶ [Section 9.4, “Configuring the Networking Service”](#)
- ▶ [Section 9.5, “Configuring the DHCP Agent”](#)
- ▶ [Section 9.6, “Configuring a Provider Network”](#)
- ▶ [Section 9.8, “Configuring the L3 Agent”](#)
- ▶ [Section 9.7, “Configuring the Plug-in Agent”](#)

[Report a bug](#)

## 9.2. Networking Prerequisite Configuration

### 9.2.1. Creating the OpenStack Networking Database

In this procedure the database and database user that will be used by the networking service will be created. These steps must be performed while logged in to the database server as the root user.

1. Connect to the database service using the `mysql` command.

```
# mysql -u root -p
```

2. Create the database. If you intend to use the:
  - ▶ Open vSwitch plug-in, the recommended database name is `ovs_quantum`.
  - ▶ Linux Bridge plug-in, the recommended database name is `quantum_linux_bridge`.

This example uses the database name of `'ovs_quantum'`.

```
mysql> CREATE DATABASE ovs_quantum;
```

3. Create a quantum database user and grant it access to the ovs\_quantum database.

```
mysql> GRANT ALL ON ovs_quantum.* TO 'quantum'@'%' IDENTIFIED BY 'PASSWORD';
```

```
mysql> GRANT ALL ON ovs_quantum.* TO 'quantum'@'localhost' IDENTIFIED BY 'PASSWORD';
```

Replace *PASSWORD* with a secure password that will be used to authenticate with the database server as this user.

4. Flush the database privileges to ensure that they take effect immediately.

```
mysql> FLUSH PRIVILEGES;
```

5. Exit the mysql client command.

```
mysql> quit
```

The OpenStack Networking database 'ovs\_quantum' has been created. The database will be populated during service configuration.

## 9.2.2. Creating the OpenStack Networking Identity Records

[Report a bug](#)

The steps outlined in this procedure cover the creation of these identity records to support the Image service:

1. Create the quantum user, who has the admin role in the services tenant.
2. Create the quantum service entry and assign it an endpoint.

These entries will assist other OpenStack services attempting to locate and access the networking functionality provided by the OpenStack Networking service:

1. Authenticate as the administrator of the identity service by running the source command on the keystone\_admin file containing the required credentials:

```
# source ~/keystone_admin
```

2. Create a user named quantum for the OpenStack networking service to use:

```
# keystone user-create --name quantum --pass PASSWORD
+-----+-----+
| Property | Value |
+-----+-----+
| email | |
| enabled | True |
| id | 1df18bcd14404fa9ad954f9d5eb163bc |
| name | quantum |
| tenantId | |
+-----+-----+
```

Replace *PASSWORD* with a secure password that will be used by the OpenStack networking service when authenticating with the identity service. Take note of the created user's returned ID as it will be used in subsequent steps.

3. Get the ID of the admin role:

```
# keystone role-get admin
```

If no admin role exists, create one:

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

4. Get the ID of the services tenant:

```
$ keystone tenant-list | grep services
```

If no services tenant exists, create one:

```
$ keystone tenant-create --name services --description "Services Tenant"
```

This guide uses one tenant for all service users. For more information, refer to *Creating the Services Tenant*.

5. Use the keystone `user-role-add` command to link the quantum user, admin role, and services tenant together:

```
# keystone user-role-add --user-id USERID --role-id ROLEID --tenant-id TENANTID
```

Replace the user, role, and tenant IDs with those obtained in the previous steps.

6. Create the quantum service entry:

```
# keystone service-create --name quantum \
  --type network \
  --description "OpenStack Networking Service"
+-----+-----+
| Property | Value |
+-----+-----+
| description | OpenStack Networking Service |
| id | 134e815915f442f89c39d2769e278f9b |
| name | quantum |
| type | network |
+-----+-----+
```

Take note of the created service's returned ID as it will be used in the next step.

7. Create the network endpoint entry:

```
# keystone endpoint-create --service-id SERVICEID \
  --publicurl "http://IP:9696" /
  --adminurl "http://IP:9696" /
  --internalurl "http://IP:9696"
```

Replace `SERVICEID` with the ID returned by the keystone `service-create` command. Replace `IP` with the IP address or host name of the system that will be acting as the network node.

All supporting identity service entries required by the OpenStack networking service have been created.

[Report a bug](#)

## 9.3. Common Networking Configuration

### 9.3.1. Upgrading the Kernel

Red Hat Enterprise Linux OpenStack Platform includes a custom Red Hat Enterprise Linux kernel that supports the use of network namespaces. This kernel **must** be installed on nodes that will handle OpenStack networking traffic. Additionally the Open vSwitch plug-in will not work with kernels with versions lower than `2.6.32-343.el6.x86_64`.

Follow the steps listed in this procedure on each node in the environment that will handle

OpenStack networking traffic. You must log in to each node as the root user to complete the procedure.

1. Use the `uname` command to identify the kernel that is currently in use on the system.

```
# uname --kernel-release
```

- A. If the output includes the text `openstack` then the system already has a network namespaces enabled kernel.

```
2.6.32-358.6.2.openstack.el6.x86_64
```

No further action is required to install a network namespaces enabled kernel on this system.

- B. If the output does **not** include the text `openstack` then the system does not currently have a network namespaces enabled kernel and further action must be taken.

```
2.6.32-358.el6.x86_64
```

Further action is required to install a network namespaces enabled kernel on this system. Follow the remaining steps outlined in this procedure to perform this task.



### Note

Note that the release field may contain a higher value than `358`. As new kernel updates are released this value is increased.

2. Install the updated kernel with network namespaces support using the `yum` command.

```
# yum install "kernel-2.6.*.openstack.el6.x86_64"
```

The use of the wildcard character (`*`) ensures that the latest kernel release available will be installed.

3. Reboot the system to ensure that the new kernel is running before proceeding with OpenStack networking installation.

```
# reboot
```

4. Run the `uname` command again once the system has rebooted to confirm that the newly installed kernel is running.

```
# uname --kernel-release
2.6.32-358.6.2.openstack.el6.x86_64
```

The system is now running a kernel with network namespaces support, enabling advanced OpenStack networking configurations.

## 9.3.2. Disabling Network Manager

[Report a bug](#)

OpenStack networking currently does not work on systems that have the Network Manager (NetworkManager) service enabled. The Network Manager service is currently enabled by default on Red Hat Enterprise Linux installations where one of these package groups was selected during installation:

- ▶ Desktop
- ▶ Software Development Workstation

The Network Manager service is **not** currently enabled by default on Red Hat Enterprise Linux



installations where one of these package groups was selected during installation:

- ▶ Basic Server
- ▶ Database Server
- ▶ Web Server
- ▶ Identity Management Server
- ▶ Virtualization Host
- ▶ Minimal Install

Follow the steps listed in this procedure while logged in as the root user on each system in the environment that will handle network traffic. This includes the system that will host the OpenStack Networking service, all network nodes, and all compute nodes.

These steps ensure that the NetworkManager service is disabled and replaced by the standard network service for all interfaces that will be used by OpenStack Networking.

1. Verify Network Manager is currently enabled using the `chkconfig` command.

```
# chkconfig --list NetworkManager
```

The output displayed by the `chkconfig` command indicates whether or not the Network Manager service is enabled.

- A. The system displays an error if the Network Manager service is not currently installed:

```
error reading information on service NetworkManager: No such file or directory
```

If this error is displayed then no further action is required to disable the Network Manager service.

- B. The system displays a list of numerical run levels along with a value of `on` or `off` indicating whether the Network Manager service is enabled when the system is operating in the given run level.

```
NetworkManager 0:off 1:off 2:off 3:off 4:off 5:off 6:off
```

If the value displayed for all run levels is `off` then the Network Manager service is disabled and no further action is required. If the value displayed for any of the run levels is `on` then the Network Manager service is enabled and further action is required.

2. Ensure that the Network Manager service is stopped using the `service` command.

```
# service NetworkManager stop
```

3. Ensure that the Network Manager service is disabled using the `chkconfig` command.

```
# chkconfig NetworkManager off
```

4. Open each interface configuration file on the system in a text editor. Interface configuration files are found in the `/etc/sysconfig/network-scripts/` directory and have names of the form `ifcfg-X` where `X` is replaced by the name of the interface. Valid interface names include `eth0`, `p1p5`, and `em1`.

In each file ensure that the `NM_CONTROLLED` configuration key is set to `no` and the `ONBOOT` configuration key is set to `yes`.

```
NM_CONTROLLED=no
ONBOOT=yes
```

This action ensures that the standard network service will take control of the interfaces and automatically activate them on boot.

5. Ensure that the network service is started using the `service` command.

```
# service network start
```

6. Ensure that the network service is enabled using the `chkconfig` command.

```
# chkconfig network on
```

The Network Manager service has been disabled. The standard network service has been enabled and configured to control the required network interfaces.

### 9.3.3. Installing the Packages

[Report a bug](#)

The OpenStack Networking service requires the following packages:

#### ***openstack-quantum***

Provides the networking service and associated configuration files.

#### ***openstack-quantum-PLUGIN***

Provides a networking plug-in. Replace *PLUGIN* with one of the recommended plug-ins (openvswitch and linuxbridge).

#### ***openstack-utils***

Provides supporting utilities to assist with a number of tasks including the editing of configuration files.

#### ***openstack-selinux***

Provides OpenStack specific SELinux policy modules.

The packages must be installed on all systems that will handle network traffic. This includes the OpenStack Networking service node, all network nodes, and all Compute nodes.

To install all of the above packages, execute the following command while logged in as the root user:

```
# yum install -y openstack-quantum \
  openstack-quantum-PLUGIN \
  openstack-utils \
  openstack-selinux
```

Replace *PLUGIN* with `openvswitch` or `linuxbridge` (determines which plug-in is installed).

The networking services are installed and ready to be configured.

### 9.3.4. Configuring the Firewall

[Report a bug](#)

Systems attempting to use the functionality provided by the networking service access it over the network using port 9696.

To allow this the firewall on the service node must be altered to allow network traffic on this port. All steps in this procedure must be run while logged in to the server hosting the image storage service as the root user.

1. Open the `/etc/sysconfig/iptables` file in a text editor.
2. Add an INPUT rule allowing TCP traffic on port 9696 to the file. The new rule must appear before any INPUT rules that REJECT traffic.

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

3. Save the changes to the `/etc/sysconfig/iptables` file.
4. Restart the `iptables` service to ensure that the change takes effect.

```
# service iptables restart
```

The `iptables` firewall is now configured to allow incoming connections to the networking service on port 9696.

[Report a bug](#)

## 9.4. Configuring the Networking Service

### Prerequisites:

- ▶ [Section 9.3, “Common Networking Configuration”](#)

The configuration of the OpenStack Networking service is contained in the `/etc/quantum/quantum.conf` file.

Repeat these configuration steps on each node that will host an instance of the OpenStack Networking service while logged in as the root user.

#### 1. Setting the Identity Values

The Networking service must be explicitly configured to use the Identity service for authentication.

- a. Set the authentication strategy (`auth_strategy`) configuration key to `keystone` using the `openstack-config` command.

```
# openstack-config --set /etc/quantum/quantum.conf \
  DEFAULT auth_strategy keystone
```

- b. Set the authentication host (`auth_host` configuration key) to the IP address or host name of the Identity server.

```
# openstack-config --set /etc/quantum/quantum.conf \
  keystone_auth token auth_host IP
```

Replace *IP* with the IP address or host name of the Identity server.

- c. Set the administration tenant name (`admin_tenant_name`) configuration key to the name of the tenant that was created for the use of the Networking service. Examples in this guide use *services*.

```
# openstack-config --set /etc/quantum/quantum.conf \
  keystone_auth token admin_tenant_name services
```

- d. Set the administration user name (`admin_user`) configuration key to the name of the user that was created for the use of the networking services. Examples in this guide use *quantum*.

```
# openstack-config --set /etc/quantum/quantum.conf \
  keystone_auth token admin_user quantum
```

- e. Set the administration password (`admin_password`) configuration key to the password that is associated with the user specified in the previous step.

```
# openstack-config --set /etc/quantum/quantum.conf \
  keystone_auth token admin_password PASSWORD
```

The authentication keys used by the Networking service have been set and will be used

when the services are started.

## 2. Setting the Message Broker

The Networking service must be explicitly configured with the type, location, and authentication details of the message broker.

- a. Use the `openstack-config` utility to set the value of the `rpc_backend` configuration key to Qpid.

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

- b. Use the `openstack-config` utility to set the value of the `qpid_hostname` configuration key to the host name of the Qpid server.

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

Replace *IP* with the IP address or host name of the message broker.

- c. If you have configured Qpid to authenticate incoming connections, you must provide the details of a valid Qpid user in the networking configuration.
  - a. Use the `openstack-config` utility to set the value of the `qpid_username` configuration key to the username of the Qpid user that the Networking service must use when communicating with the message broker.

```
# openstack-config --set /etc/quantum/quantum.conf \
  DEFAULT qpid_username USERNAME
```

Replace *USERNAME* with the required Qpid user name.

- b. Use the `openstack-config` utility to set the value of the `qpid_password` configuration key to the password of the Qpid user that the Networking service must use when communicating with the message broker.

```
# openstack-config --set /etc/quantum/quantum.conf \
  DEFAULT qpid_password PASSWORD
```

Replace *PASSWORD* with the password of the Qpid user.

- d. If you configured Qpid to use SSL, you must inform the Networking service of this choice. Use `openstack-config` utility to set the value of the `qpid_protocol` configuration key to `ssl`.

```
# openstack-config --set /etc/quantum/quantum.conf \
  DEFAULT qpid_protocol ssl
```

The value of the `qpid_port` configuration key must be set to 5671 as Qpid listens on this different port when SSL is in use.

```
# openstack-config --set /etc/quantum/quantum.conf \
  DEFAULT qpid_port 5671
```



### Important

To communicate with a Qpid message broker that uses SSL the node must also have:

- The *nss* package installed.
- The certificate of the relevant certificate authority installed in the system NSS database (`/etc/pki/nssdb/`).

The `certtool` command is able to import certificates into the NSS database. See the `certtool` manual page for more information (`man certtool`).

The OpenStack Networking service has been configured to use the message broker and any authentication schemes that it presents.

### 3. Setting the Plug-in

Additional configuration settings must be applied to enable the desired plug-in.

#### A. Open vSwitch

- a. Create a symbolic link between the `/etc/quantum/plugin.ini` path referred to by the Networking service and the plug-in specific configuration file.

```
# ln -s /etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini \
  /etc/quantum/plugin.ini
```

- b. Update the value of the `tenant_network_type` configuration key in the `/etc/quantum/plugin.ini` file to refer to the type of network that must be used for tenant networks. Supported values are `flat`, `vlan`, and `local`.

The default is `local` but this is not recommended for real deployments.

```
# openstack-config --set /etc/quantum/plugin.ini \
  OVS tenant_network_type TYPE
```

Replace *TYPE* with the type chosen tenant network type.

- c. If `flat` or `vlan` networking was chosen, the value of the `network_vlan_ranges` configuration key must also be set. This configuration key maps physical networks to VLAN ranges.

Mappings are of the form *NAME:START:END* where *NAME* is replaced by the name of the physical network, *START* is replaced by the VLAN identifier that starts the range, and *END* is replaced by the replaced by the VLAN identifier that ends the range.

```
# openstack-config --set /etc/quantum/plugin.ini \
  OVS network_vlan_ranges NAME:START:END
```

Multiple ranges can be specified using a comma separated list, for example:

```
physnet1:1000:2999,physnet2:3000:3999
```

- d. Update the value of the `core_plugin` configuration key in the `/etc/quantum/quantum.conf` file to refer to the Open vSwitch plug-in.

```
# openstack-config --set /etc/quantum/quantum.conf \
  DEFAULT core_plugin \
  quantum.plugins.openvswitch.ovs_quantum_plugin.OVSQuantumPluginV2
```

#### B. Linux Bridge

- a. Create a symbolic link between the `/etc/quantum/plugin.ini` path referred to by the Networking service and the plug-in specific configuration file.

```
# ln -s /etc/quantum/plugins/linuxbridge/linuxbridge_conf.ini \
  /etc/quantum/plugin.ini
```

- b. Update the value of the `tenant_network_type` configuration key in the `/etc/quantum/plugin.ini` file to refer to the type of network that must be used for tenant networks. Supported values are `flat`, `vlan`, and `local`.

The default is `local` but this is not recommended for real deployments.

```
# openstack-config --set /etc/quantum/plugin.ini \
  VLAN tenant_network_type TYPE
```

Replace *TYPE* with the type chosen tenant network type.

- c. If flat or vlan networking was chosen, the value of the `network_vlan_ranges` configuration key must also be set. This configuration key maps physical networks to VLAN ranges.

Mappings are of the form `NAME:START:END` where `NAME` is replaced by the name of the physical network, `START` is replaced by the VLAN identifier that starts the range, and `END` is replaced by the replaced by the VLAN identifier that ends the range.

```
# openstack-config --set /etc/quantum/plugin.ini \
  LINUX_BRIDGE network_vlan_ranges NAME:START:END
```

Multiple ranges can be specified using a comma separated list, for example:

```
physnet1:1000:2999,physnet2:3000:3999
```

- d. Update the value of the `core_plugin` configuration key in the `/etc/quantum/quantum.conf` file to refer to the Linux Bridge plug-in.

```
# openstack-config --set /etc/quantum/quantum.conf \
  DEFAULT core_plugin \
  quantum.plugins.linuxbridge.lb_quantum_plugin.LinuxBridgePluginV2
```

#### 4. Setting the Database Connection String

The database connection string used by the networking service is defined in the `/etc/quantum/plugin.ini` file. It must be updated to point to a valid database server before starting the service.

- a. Use the `openstack-config` command to set the value of the connection configuration key.

```
# openstack-config --set /etc/quantum/plugin.ini \
  DATABASE sql_connection mysql://USER:PASS@IP/DB
```

Replace:

- ▶ `USER` with the database user name the networking service is to use, usually `quantum`.
- ▶ `PASS` with the password of the chosen database user.
- ▶ `IP` with the IP address or host name of the database server.
- ▶ `DB` with the name of the database that has been created for use by the networking service (`ovs_quantum` was used as the example in the previous *Creating the OpenStack Networking Database* section).

#### 5. Start the Networking Service

- a. Start the OpenStack Networking service using the service command.

```
# service quantum-server start
```

- b. Enable the Networking service permanently using the `chkconfig` command.

```
# chkconfig quantum-server on
```

The OpenStack Networking service is configured and running. Further action is however required to configure and run the various networking agents that are also fundamental to providing networking functionality.



## Important

By default, OpenStack Networking does not enforce Classless Inter-Domain Routing (CIDR) checking of IP addresses. This is to maintain backwards compatibility with previous releases. If you require such checks set the value of the `force_gateway_on_subnet` configuration key to `True` in the `/etc/quantum/quantum.conf` file.

[Report a bug](#)

## 9.5. Configuring the DHCP Agent

### Prerequisites:

- [Section 9.3, “Common Networking Configuration”](#)

Follow the steps listed in this procedure to configure the DHCP Agent. All steps listed in this procedure must be performed on the network node while logged in as the root user on the system hosting the DHCP agent.

### 1. Configuring Authentication

The DHCP agent must be explicitly configured to use the identity service for authentication.

- Set the authentication strategy (`auth_strategy`) configuration key to `keystone` using the `openstack-config` command.

```
# openstack-config --set /etc/quantum/dhcp_agent.ini \
  DEFAULT auth_strategy keystone
```

- Set the authentication host (`auth_host` configuration key) to the IP address or host name of the identity server.

```
# openstack-config --set /etc/quantum/dhcp_agent.ini \
  keystone_authtoken auth_host IP
```

Replace *IP* with the IP address or host name of the identity server.

- Set the administration tenant name (`admin_tenant_name`) configuration key to the name of the tenant that was created for the use of the networking services. Examples in this guide use *services*.

```
# openstack-config --set /etc/quantum/dhcp_agent.ini \
  keystone_authtoken admin_tenant_name services
```

- Set the administration user name (`admin_user`) configuration key to the name of the user that was created for the use of the networking services. Examples in this guide use *quantum*.

```
# openstack-config --set /etc/quantum/dhcp_agent.ini \
  keystone_authtoken admin_user quantum
```

- Set the administration password (`admin_password`) configuration key to the password that is associated with the user specified in the previous step.

```
# openstack-config --set /etc/quantum/dhcp_agent.ini \
  keystone_authtoken admin_password PASSWORD
```

### 2. Configuring the Interface Driver

Set the value of the `interface_driver` configuration key in the `/etc/quantum/dhcp_agent.ini` file based on the networking plug-in being used. Execute only the configuration step that applies to the plug-in used in your environment.

## A. Open vSwitch Interface Driver

```
# openstack-config --set /etc/quantum/dhcp_agent.ini \
  DEFAULT interface_driver
quantum.agent.linux.interface.OVSInterfaceDriver
```

## B. Linux Bridge Interface Driver

```
# openstack-config --set /etc/quantum/dhcp_agent.ini \
  DEFAULT interface_driver
quantum.agent.linux.interface.BridgeInterfaceDriver
```

## 3. Starting the DHCP Agent

- a. Use the service command to start the quantum-dhcp-agent service.

```
# service quantum-dhcp-agent start
```

- b. Use the chkconfig command to ensure that the quantum-dhcp-agent service will be started automatically in the future.

```
# chkconfig quantum-dhcp-agent on
```

The DHCP agent has been configured and started.

[Report a bug](#)

# 9.6. Configuring a Provider Network

## Prerequisites:

- ▶ [Section 9.3, “Common Networking Configuration”](#)

OpenStack networking provides two mechanisms for connecting the Layer 3 (L3) agent to an external network. The first, attaching it to an external bridge (br-ex) directly, is only supported when the Open vSwitch plug-in is in use. The second method, which is supported by both the Open vSwitch plug-in and the Linux Bridge plug-in, is to use an external provider network.

To use an external provider network it is first necessary to create one. Follow the steps outlined in this procedure while logged in to a system with the OpenStack networking client - provided by the *python-quantumclient* package installed. You must also have access to a *keystonerc\_admin* file containing the authentication details of the OpenStack administrative user.

Take note of the unique identifiers generated by the steps listed in this procedure. These identifiers will be required when configuring the L3 agent.

1. Use the source command to load the credentials of the administrative user.

```
$ source ~/keystonerc_admin
```

2. Use the net-create action of the quantum command line client to create a new provider network.

```
$ quantum net-create EXTERNAL_NAME \
  --router:external True \
  --provider:network_type TYPE \
  --provider:physical_network PHYSICAL_NAME \
  --provider:segmentation_id VLAN_TAG
```

Replace these strings with the appropriate values for your environment:

- ▶ Replace *EXTERNAL\_NAME* with a name for the new external network provider.
- ▶ Replace *PHYSICAL\_NAME* with a name for the physical network. This is not applicable if



you intend to use a local network type.

- Replace *TYPE* with the type of provider network you wish to use. Supported values are *flat* (for flat networks), *vlan* (for VLAN networks), and *local* (for local networks).
- Replace *VLAN\_TAG* with the VLAN tag that will be used to identify network traffic. The VLAN tag specified must have been defined by the network administrator.

If the *network\_type* was set to a value other than *vlan* then this parameter is not required.

Take note of the unique external network identifier returned, this will be required in subsequent steps.

3. Use the *subnet-create* action of the command line client to create a new subnet for the new external provider network.

```
$ quantum subnet-create --gateway GATEWAY \
  --allocation-pool start=IP_RANGE_START,end=IP_RANGE_END \
  --disable-dhcp EXTERNAL_NAME EXTERNAL_CIDR
```

Replace these strings with the appropriate values for your environment:

- Replace *GATEWAY* with the IP address or hostname of the system that is to act as the gateway for the new subnet.
- Replace *IP\_RANGE\_START* with the IP address that denotes the start of the range of IP addresses within the new subnet that floating IP addresses will be allocated from.
- Replace *IP\_RANGE\_END* with the IP address that denotes the end of the range of IP addresses within the new subnet that floating IP addresses will be allocated from.
- Replace *EXTERNAL\_NAME* with the name of the external network the subnet is to be associated with. This must match the name that was provided to the *net-create* action in the previous step.
- Replace *EXTERNAL\_CIDR* with the Classless Inter-Domain Routing (CIDR) representation of the block of IP addresses the subnet represents. An example would be *192.168.100.0/24*.

Take note of the unique subnet identifier returned, this will be required in subsequent steps.



### Important

The IP address used to replace the string *GATEWAY* **must** be within the block of IP addresses specified in place of the *EXTERNAL\_CIDR* string but outside of the block of IP addresses specified by the range started by *IP\_RANGE\_START* and ended by *IP\_RANGE\_END*.

The block of IP addresses specified by the range started by *IP\_RANGE\_START* and ended by *IP\_RANGE\_END* **must** also fall within the block of IP addresses specified by *EXTERNAL\_CIDR*.

4. Use the *router-create* action of the quantum command line client to create a new router.

```
$ quantum router-create NAME
```

Replace *NAME* with the name to give the new router. Take note of the unique router identifier returned, this will be required in subsequent steps.

5. Use the *router-gateway-set* action of the quantum command line client to link the newly created router to the external provider network.

```
$ quantum router-gateway-set ROUTER NETWORK
```

Replace *ROUTER* with the unique identifier of the router, replace *NETWORK* with the unique

identifier of the external provider network.

6. Use the `router-interface-add` action of the quantum command line client to link the newly created router to the subnet.

```
$ quantum router-interface-add ROUTER SUBNET
```

Replace `ROUTER` with the unique identifier of the router, replace `SUBNET` with the unique identifier of the subnet.

An external provider network has been created. Use the unique identifier of the router when configuring the L3 agent.

[Report a bug](#)

## 9.7. Configuring the Plug-in Agent

### 9.7.1. Configuring the Open vSwitch Plug-in Agent

#### Prerequisites:

- ▶ [Section 9.3, “Common Networking Configuration”](#)

The Open vSwitch plug-in has a corresponding agent. When the Open vSwitch plug-in is in use all nodes in the environment that handle data packets must have the agent installed and configured.

This includes all compute nodes and systems hosting the dedicated DHCP and L3 agents.

1. Confirm that the `openvswitch` package is installed. This is normally installed as a dependency of the `quantum-plugin-openvswitch` package.

```
# rpm -qa | grep openvswitch
openvswitch-1.10.0-1.el6.x86_64
openstack-quantum-openvswitch-2013.1-3.el6.noarc
```

2. Start the `openvswitch` service.

```
# service openvswitch start
```

3. Enable the `openvswitch` service permanently.

```
# chkconfig openvswitch on
```

4. 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.

```
# 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.

5. Ensure that the `br-int` device persists on reboot by creating a `/etc/sysconfig/network-scripts/ifcfg-br-int` file with these contents:

```
DEVICE=br-int
DEVICETYPE=ovs
TYPE=OVSBridge
ONBOOT=yes
BOOTPROTO=none
```

- Set the value of the `bridge_mappings` configuration key. This configuration key must contain a list of physical networks and the network bridges associated with them. The format for each entry in the comma separated list is:

```
PHYSNET:BRIDGE
```

Where *PHYSNET* is replaced with the name of a physical network, and *BRIDGE* is replaced by the name of the network bridge.

The physical network must have been defined in the `network_vlan_ranges` configuration variable on the OpenStack Networking server.

```
# openstack-config --set /etc/quantum/plugin.ini \
  OVS bridge_mappings MAPPINGS
```

Replace *MAPPINGS* with the physical network to bridge mappings.

- Use the service command to start the `quantum-openvswitch-agent` service.

```
# service quantum-openvswitch-agent start
```

- Use the `chkconfig` command to ensure that the `quantum-openvswitch-agent` service is started automatically in the future.

```
# chkconfig quantum-openvswitch-agent on
```

- Use the `chkconfig` command to ensure that the `quantum-ovs-cleanup` service is started automatically on boot. When started at boot time this service ensures that the OpenStack Networking agents maintain full control over the creation and management of tap devices.

```
# chkconfig quantum-ovs-cleanup on
```

The networking configuration has been updated to use the Open vSwitch plug-in.

## 9.7.2. Configuring the Linux Bridge Plug-in Agent

[Report a bug](#)

### Prerequisites:

- ▶ [Section 9.3, “Common Networking Configuration”](#)

The Linux Bridge plug-in has a corresponding agent. When the Linux Bridge plug-in is in use all nodes in the environment that handle data packets must have the agent installed and configured.

This includes all compute nodes and systems hosting the dedicated DHCP and L3 agents.

- Set the value of the `physical_interface_mappings` configuration key. This configuration key must contain a list of physical networks and the VLAN ranges associated with them that are available for allocation to tenant networks.

The format for each entry in the comma separated list is:

```
PHYSNET:VLAN_START:VLAN_END
```

Where *PHYSNET* is replaced with the name of a physical network, *VLAN\_START* is replaced by an identifier indicating the start of the VLAN range, and *VLAN\_END* is replaced by an

identifier indicating the end of the VLAN range.

The physical networks must have been defined in the `network_vlan_ranges` configuration variable on the OpenStack Networking server.

```
# openstack-config --set /etc/quantum/plugin.ini \
  LINUX_BRIDGE physical_interface_mappings MAPPINGS
```

Replace `MAPPINGS` with the physical network to VLAN range mappings.

2. Use the service command to start the `quantum-linuxbridge-agent` service.

```
# service quantum-linuxbridge-agent start
```

3. Use the `chkconfig` command to ensure that the `quantum-linuxbridge-agent` service is started automatically in the future.

```
# chkconfig quantum-linuxbridge-agent on
```

The networking configuration has been updated to use the Linux Bridge plug-in.

[Report a bug](#)

## 9.8. Configuring the L3 Agent

### Prerequisites:

- ▶ [Section 9.3, “Common Networking Configuration”](#)

Follow the steps listed in this procedure to configure the L3 agent. All steps listed in this procedure must be performed on the network node while logged in as the root user.

#### 1. Configuring Authentication

- a. Set the authentication strategy (`auth_strategy`) configuration key to `keystone` using the `openstack-config` command.

```
# openstack-config --set /etc/quantum/metadata_agent.ini \
  DEFAULT auth_strategy keystone
```

- b. Set the authentication host (`auth_host` configuration key to the IP address or host name of the identity server.

```
# openstack-config --set /etc/quantum/metadata_agent.ini \
  keystone_authtoken auth_host IP
```

Replace `IP` with the IP address or host name of the identity server.

- c. Set the administration tenant name (`admin_tenant_name`) configuration key to the name of the tenant that was created for the use of the networking services. Examples in this guide use `services`.

```
# openstack-config --set /etc/quantum/metadata_agent.ini \
  keystone_authtoken admin_tenant_name services
```

- d. Set the administration user name (`admin_user`) configuration key to the name of the user that was created for the use of the networking services. Examples in this guide use `quantum`.

```
# openstack-config --set /etc/quantum/metadata_agent.ini \
  keystone_authtoken admin_user quantum
```

- e. Set the administration password (`admin_password`) configuration key to the password that is associated with the user specified in the previous step.

```
# openstack-config --set /etc/quantum/metadata_agent.ini \
  keystone_auth_token admin_password PASSWORD
```

## 2. Configuring the Interface Driver

Set the value of the `interface_driver` configuration key in the `/etc/quantum/l3_agent.ini` file based on the networking plug-in being used. Execute only the configuration step that applies to the plug-in used in your environment.

### A. Open vSwitch Interface Driver

```
# openstack-config --set /etc/quantum/l3_agent.ini \
  DEFAULT interface_driver
  quantum.agent.linux.interface.OVSInterfaceDriver
```

### B. Linux Bridge Interface Driver

```
# openstack-config --set /etc/quantum/l3_agent.ini \
  DEFAULT interface_driver
  quantum.agent.linux.interface.BridgeInterfaceDriver
```

## 3. Configuring External Network Access

The L3 agent connects to external networks using either an external bridge or an external provider network. When using the Open vSwitch plug-in either approach is supported. When using the Linux Bridge plug-in only the use of an external provider network is supported. Choose the approach that is most appropriate for the environment.

### A. Using an External Bridge

To use an external bridge you must create and configure it. Finally the OpenStack networking configuration must be updated to use it. This must be done on each system hosting an instance of the L3 agent.

- a. Use the `ovs-vsctl` command to create the external bridge named `br-ex`.

```
# ovs-vsctl add-br br-ex
```

- b. Ensure that the `br-ex` device persists on reboot by creating a `/etc/sysconfig/network-scripts/ifcfg-br-ex` file with these contents:

```
DEVICE=br-ex
DEVICETYPE=ovs
TYPE=OVSBridge
ONBOOT=yes
BOOTPROTO=none
```

- c. Ensure that the value of the `external_network_bridge` configuration key in the `/etc/quantum/l3_agent.ini` file is `br-ex`. This ensures that the L3 agent will use the external bridge.

```
# openstack-config --set /etc/quantum/l3_agent.ini \
  DEFAULT external_network_bridge br-ex
```

### B. Using a Provider Network

To connect the L3 agent to external networks using a provider network you must first have created the provider network. You must also have created a subnet and router to associate with it. The unique identifier of the router will be required to complete these steps.

- a. Ensure that the value of the `external_network_bridge` configuration key in the `/etc/quantum/l3_agent.ini` file is blank. This ensures that the L3 agent does not attempt to use an external bridge.

```
# openstack-config --set /etc/quantum/l3_agent.ini \
  DEFAULT external_network_bridge ""
```

- b. Set the value of the `router_id` configuration key in the `/etc/quantum/l3_agent.ini` file to the identifier of the external router that must be used by the L3 agent when accessing the external provider network.

```
# openstack-config --set /etc/quantum/l3_agent.ini \
  DEFAULT router_id ROUTER
```

Replace `ROUTER` with the unique identifier of the router that has been defined for use when accessing the external provider network.

#### 4. Starting the L3 Agent

- a. Use the service command to start the `quantum-l3-agent` service.

```
# service quantum-l3-agent start
```

- b. Use the `chkconfig` command to ensure that the `quantum-l3-agent` service will be started automatically in the future.

```
# chkconfig quantum-l3-agent on
```

#### 5. Starting the Metadata Agent

The OpenStack networking metadata agent allows virtual machine instances to communicate with the compute metadata service. It runs on the same hosts as the Layer 3 (L3) agent.

- a. Use the service command to start the `quantum-metadata-agent` service.

```
# service quantum-metadata-agent start
```

- b. Use the `chkconfig` command to ensure that the `quantum-metadata-agent` service will be started automatically in the future.

```
# chkconfig quantum-metadata-agent on
```

The L3 agent has been configured and started.

[Report a bug](#)

## 9.9. Validating the OpenStack Networking Installation

To begin using OpenStack networking it is necessary to deploy networking components to compute nodes. Initial networks and routers must also be defined. It is however possible to perform basic sanity checking of the OpenStack networking deployment by following the steps outline in this procedure.

### 1. All Nodes

- a. Verify that the customized Red Hat Enterprise Linux kernel intended for use with Red Hat Enterprise Linux OpenStack Platform is running:

```
$ uname --kernel-release
2.6.32-358.6.2.openstack.el6.x86_64
```

If the kernel release value returned does not contain the string `openstack` then update the kernel and reboot the system.

- b. Ensure that the installed IP utilities support network namespaces:

```
$ ip netns
```

If an error indicating that the argument is not recognised or supported is returned then update the system using `yum`.

### 2. Service Nodes

- a. Ensure that the `quantum-server` service is running:

```
$ service quantum-server status
quantum-server (pid 3011) is running...
```

### 3. Network Nodes

- a. Ensure that the DHCP agent is running:

```
$ service quantum-dhcp-agent status
quantum-dhcp-agent (pid 3012) is running...
```

- b. Ensure that the L3 agent is running:

```
$ service quantum-l3-agent status
quantum-l3-agent (pid 3013) is running...
```

- c. Ensure that the plug-in agent, if applicable, is running:

```
$ service quantum-PLUGIN-agent status
quantum-PLUGIN-agent (pid 3014) is running...
```

Replace *PLUGIN* with the appropriate plug-in for the environment. Valid values include `openvswitch` and `linuxbridge`.

- d. Ensure that the metadata agent is running:

```
$ service quantum-metadata-agent status
quantum-metadata-agent (pid 3015) is running...
```

All required services on the service and network nodes are operational. Proceed to deploy some compute nodes, define networks, and define routers to begin using OpenStack networking.

# Chapter 10. Installing the OpenStack Compute Service

## 10.1. Compute Service Requirements

### 10.1.1. Checking for Hardware Virtualization Support

The OpenStack Compute Service requires hardware virtualization support and certain kernel modules. Follow this procedure to determine whether your system has hardware virtualization support and the correct kernel modules available.

All steps listed must be performed while logged into the system as the root user.

1. Use the `grep` command to check for the presence of the `svm` or `vmx` CPU extensions by inspecting the `/proc/cpuinfo` file generated by the kernel:

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

If any output is shown after running this command then the CPU is hardware virtualization capable and the functionality is enabled in the system BIOS.

2. Use the `lsmod` command to list the loaded kernel modules and verify that the `kvm` modules are loaded:

```
# lsmod | grep kvm
```

If the output includes `kvm_intel` or `kvm_amd` then the `kvm` hardware virtualization modules are loaded and your kernel meets the module requirements for the OpenStack Compute Service.

This completes the required checks to ensure hardware virtualization support is available and enabled, and that you have the correct kernel modules loaded.

If the checks indicated that either hardware virtualization support or the required kernel modules are not available or not enabled then you must take action to either find a system that does have the required hardware virtualization support and modules, or enable it on the system that failed the checks.

## 10.2. Installing a Compute VNC Proxy

### 10.2.1. Installing the Compute VNC Proxy Packages

The VNC Proxy is available to users of the Compute service. Two types of VNC proxy server packages are available. The `openstack-nova-novncproxy` package provides VNC support to instances through a web browser (using Websockets), while the `openstack-nova-console` package provides access to instances through a traditional VNC client (via the `openstack-nova-xvncproxy` service).

The console authentication service, also provided by the `openstack-nova-console` package, is used to authenticate the VNC connections. Typically the console authentication service and the proxy utilities are installed on the same host as the Compute API service.



The following steps must be performed while logged in as the root user.

- ▶ Install the VNC proxy utilities and the console authentication service:
  - A. Install the *openstack-nova-novncproxy* package using the yum command:

```
# yum install -y openstack-nova-novncproxy
```

- B. Install the *openstack-nova-console* package using the yum command:

```
# yum install -y openstack-nova-console
```

The VNC Proxy packages and the console authentication service are now installed and ready for configuration.

### 10.2.2. Configuring the Firewall

[Report a bug](#)

The node that hosts VNC access to instances must be configured to allow VNC traffic through its firewall. By default, the *openstack-nova-novncproxy* service listens on TCP port 6080 and the *openstack-nova-xvncproxy* service listens on TCP port 6081.

The following procedure allows traffic on TCP port 6080 to traverse through the firewall for use by the *openstack-nova-novncproxy* package:

The following steps must be performed while logged in as the root user.

1. Edit the */etc/sysconfig/iptables* file and add the following on a new line underneath the *-A INPUT -i lo -j ACCEPT* line and before any *-A INPUT -j REJECT* rules:

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 6080 -j ACCEPT
```

2. Save the file and exit the editor.

- ▶ Similarly, when using the *openstack-nova-xvncproxy* service, enable traffic on TCP port 6081 with the following on a new line in the same location:

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 6081 -j ACCEPT
```

Once the file has been edited with the new firewall rule or rules, run the following commands as the root user to apply the changes:

```
# service iptables restart
```

```
# iptables-save
```

The new firewall rules will be applied to the running system and take effect immediately. The rules will also remain available after rebooting the system. The firewall is now configured to allow VNC proxy traffic.

### 10.2.3. Controlling the VNC Proxy service

[Report a bug](#)

VNC access to instances is available over a web browser or with a traditional VNC client. The */etc/nova/nova.conf* file holds the following VNC options:

- ▶ *vnc\_enabled* - Default is true.
- ▶ *vncserver\_listen* - The IP address to which VNC services will bind.
- ▶ *vncserver\_proxyclient\_address* - The IP address of the compute host used by proxies to connect to instances.
- ▶ *novncproxy\_base\_url* - The browser address where clients connect to instance.
- ▶ *novncproxy\_port* - The port listening for browser VNC connections. Default is 6080.

- ▶ *xvpvncproxy\_port* - The port to bind for traditional VNC clients. Default is 6081.

As the root user, use the service command to start the console authentication service:

```
# service openstack-nova-consoleauth start
```

Use the chkconfig command to permanently enable the service:

```
# chkconfig openstack-nova-consoleauth on
```

As the root user, use the service command on the nova node to start the browser-based service:

```
# service openstack-nova-novncproxy start
```

Use the chkconfig command to permanently enable the service:

```
# chkconfig openstack-nova-novncproxy on
```

To control access to the VNC service that uses a traditional client (non browser-based), substitute *openstack-nova-xvpvncproxy* into the previous commands.

### 10.2.4. Accessing Instances with the Compute VNC Proxy

[Report a bug](#)

Browse to the *novncproxy\_base\_url* URL provided in the */etc/nova/nova.conf* file to access instance consoles.

The following image shows VNC access to a Fedora Linux instance with a web browser. It is provided only as an example, and settings such as IP addresses will be different in your environment.

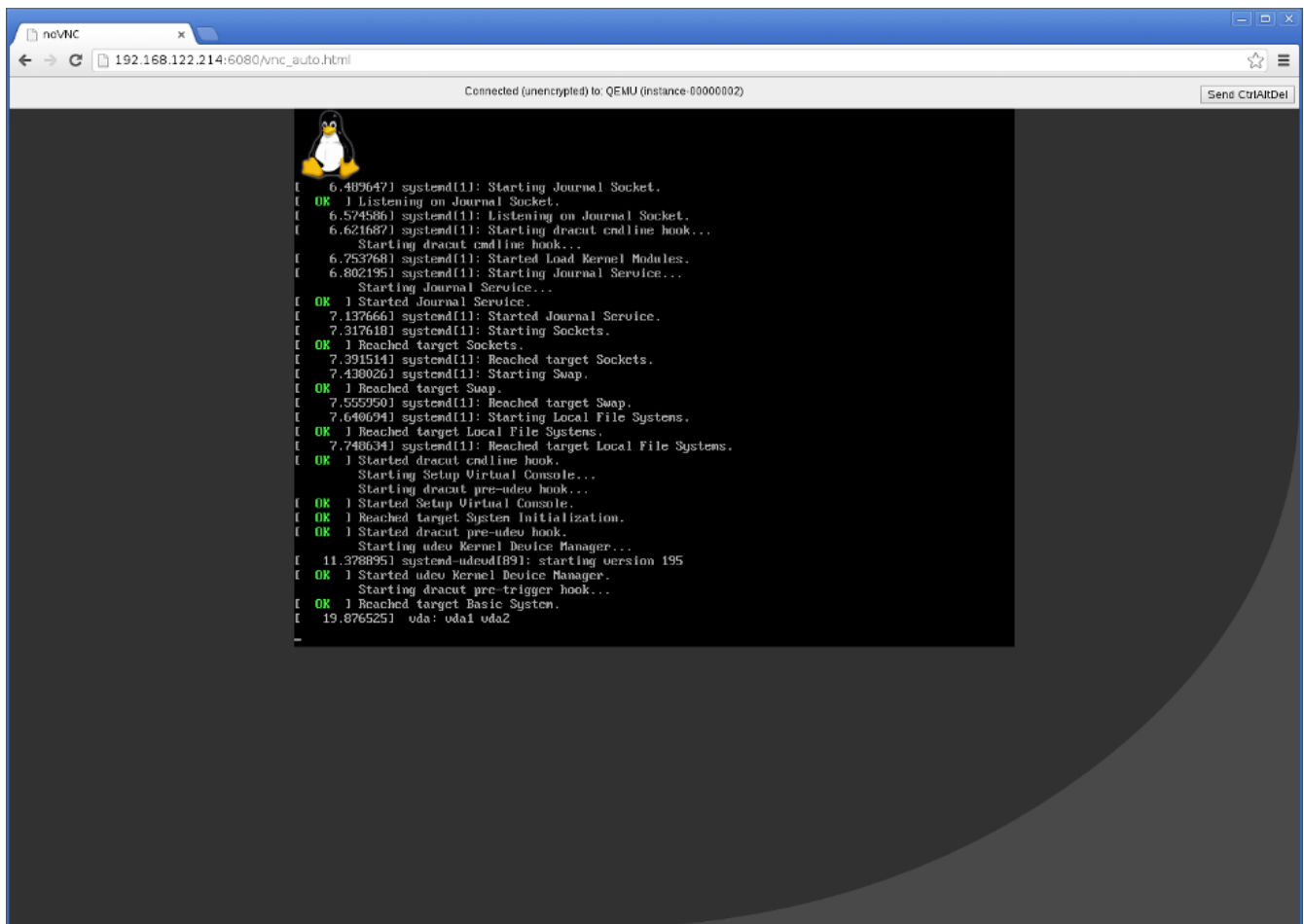


Figure 10.1. VNC instance access

[Report a bug](#)

## 10.3. Installing a Compute Node

### 10.3.1. Creating the Compute Service Database

The following procedure creates the database and database user used by the Compute service. These steps must be performed while logged in to the database server as the root user.

1. Connect to the database service using the `mysql` command.

```
# mysql -u root -p
```

2. Create the nova database.

```
mysql> CREATE DATABASE nova;
```

3. Create a nova database user and grant it access to the nova database.

```
mysql> GRANT ALL ON nova.* TO 'nova'@'%' IDENTIFIED BY 'PASSWORD';
```

```
mysql> GRANT ALL ON nova.* TO 'nova'@'localhost' IDENTIFIED BY 'PASSWORD';
```

Replace `PASSWORD` with a secure password that will be used to authenticate with the database server as this user.

4. Flush the database privileges to ensure that they take effect immediately.

```
mysql> FLUSH PRIVILEGES;
```

5. Exit the mysql client command.

```
mysql> quit
```

The Compute database has been created. The database will be populated during service configuration.

### 10.3.2. Creating the Compute Identity Records

[Report a bug](#)

The steps outlined in this procedure cover the creation of these identity records to support the Image service:

1. Create the compute user, who has the admin role in the services tenant.
2. Create the compute service entry and assign it an endpoint.

These entries will assist other OpenStack services attempting to locate and access the functionality provided by the Compute service:

1. Authenticate as the administrator of the Identity service by running the source command on the keystone\_erc\_admin file containing the required credentials:

```
# source ~/keystone_erc_admin
```

2. Create a user named compute for the OpenStack Compute service to use:

```
# keystone user-create --name compute --pass PASSWORD
+-----+-----+
| Property |          Value          |
+-----+-----+
| email    |                          |
| enabled  |          True           |
| id       | 96cd855e5bfe471ce4066794bbafb615 |
| name     |          compute       |
| tenantId |                          |
+-----+-----+
```

Replace *PASSWORD* with a secure password that will be used by the Compute service when authenticating against the Identity service. Take note of the created user's returned ID as it will be used in subsequent steps.

3. Get the ID of the admin role:

```
# keystone role-get admin
```

If no admin role exists, create one:

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

4. Get the ID of the services tenant:

```
$ keystone tenant-list | grep services
```

If no services tenant exists, create one:

```
$ keystone tenant-create --name services --description "Services Tenant"
```

This guide uses one tenant for all service users. For more information, refer to *Creating the Services Tenant*.

5. Use the keystone `user-role-add` command to link the compute user, admin role, and services tenant together:

```
# keystone user-role-add --user-id USERID --role-id ROLEID --tenant-id
TENANTID
```

Replace the user, role, and tenant IDs with those obtained in the previous steps.

6. Create the compute service entry:

```
# keystone service-create --name compute \
  --type compute \
  --description "OpenStack Compute Service"
+-----+-----+
| Property | Value |
+-----+-----+
| description | OpenStack Compute Service |
| id | 8dea97f5ee254b309c1792d2bd821e59 |
| name | compute |
| type | compute |
+-----+-----+
```

Take note of the created service's returned ID as it will be used in the next step.

7. Create the compute endpoint entry:

```
# keystone endpoint-create --service-id SERVICEID \
  --publicurl "http://IP:8774/v2/\$(tenant_id)s" \
  --adminurl "http://IP:8774/v2/\$(tenant_id)s" \
  --internalurl "http://IP:8774/v2/\$(tenant_id)s"
```

Replace:

- *SERVICEID* with the ID returned by the keystone `service-create` command.
- *IP* with the IP address or host name of the system that will be acting as the compute node.

All supporting identity service entries required by the OpenStack Compute service have been created.

### 10.3.3. Installing the Compute Service

[Report a bug](#)

The OpenStack Compute services are provided the following packages:

#### ***openstack-nova-api***

Provides the OpenStack Compute API service. At least one node in the environment must host an instance of the API service. This must be the node pointed to by the Identity service endpoint definition for the Compute service.

#### ***openstack-nova-compute***

Provides the OpenStack Compute service.

#### ***openstack-nova-conductor***

Provides the Compute conductor service. The conductor handles database requests made by Compute nodes, ensuring that individual Compute nodes do not require direct database access. At least one node in each environment must act as a Compute conductor.

#### ***openstack-nova-scheduler***

Provides the Compute scheduler service. The scheduler handles scheduling of requests made to the API across the available Compute resources. At least one node in each environment must act as a Compute scheduler.

**python-cinderclient**

Provides client utilities for accessing storage managed by the OpenStack Block Storage service. This package is not required if you do not intend to attach block storage volumes to your instances or you intend to manage such volumes using a service other than the OpenStack Block Storage service.

To install the above packages, execute the following command while logged in as the root user:

```
# yum install -y openstack-nova-api openstack-nova-compute \
  openstack-nova-conductor openstack-nova-scheduler \
  python-cinderclient
```

**Note**

In the command presented here all Compute service packages are installed on a single node. In a production deployment it is recommended that the API, conductor, and scheduler services are installed on a separate controller node or on separate nodes entirely. The Compute service itself must be installed on each node that is expected to host virtual machine instances.

The Compute service package is now installed.

**10.3.4. Configuring the Compute Service**

[Report a bug](#)

**10.3.4.1. Configuring Authentication**

The Compute service must be explicitly configured to use the Identity service for authentication. Follow the steps listed in this procedure to configure this.

All steps listed in this procedure must be performed on each system hosting Compute services while logged in as the root user.

1. Set the authentication strategy (`auth_strategy`) configuration key to `keystone` using the `openstack-config` command.

```
# openstack-config --set /etc/nova/nova.conf \
  DEFAULT auth_strategy keystone
```

2. Set the authentication host (`auth_host`) configuration key to the IP address or host name of the identity server.

```
# openstack-config --set /etc/nova/api-paste.ini \
  filter:authtoken auth_host IP
```

Replace *IP* with the IP address or host name of the identity server.

3. Set the administration tenant name (`admin_tenant_name`) configuration key to the name of the tenant that was created for the use of the Compute service. In this guide, examples use *services*.

```
# openstack-config --set /etc/nova/api-paste.ini \
  filter:authtoken admin_tenant_name services
```

4. Set the administration user name (`admin_user`) configuration key to the name of the user

that was created for the use of the Compute service. In this guide, examples use *nova*.

```
# openstack-config --set /etc/nova/api-paste.ini \
  filter:authtoken admin_user nova
```

5. Set the administration password (`admin_password`) configuration key to the password that is associated with the user specified in the previous step.

```
# openstack-config --set /etc/nova/api-paste.ini \
  filter:authtoken admin_password PASSWORD
```

The authentication keys used by the Compute services have been set and will be used when the services are started.

#### 10.3.4.2. Setting the Database Connection String

[Report a bug](#)

The database connection string used by the Compute service is defined in the `/etc/nova/nova.conf` file. It must be updated to point to a valid database server before starting the service.

In the "Grizzly" release of OpenStack the database connection string only needs to be set on nodes that will be hosting the conductor service (`openstack-nova-conductor`). Compute nodes communicate with the conductor using the messaging infrastructure, the conductor in turn orchestrates communication with the database. As a result individual compute nodes no longer require direct access to the database. This procedure only needs to be followed on nodes that will host the conductor service. There must be at least one instance of the conductor service in any compute environment.

All commands in this procedure must be run while logged in as the root user on the server hosting the Compute service.

- Use the `openstack-config` command to set the value of the `sql_connection` configuration key.

```
# openstack-config --set /etc/nova/nova.conf \
  DEFAULT sql_connection mysql://USER:PASS@IP/DB
```

Replace:

- *USER* with the database user name the Compute service is to use, usually `nova`.
- *PASS* with the password of the chosen database user.
- *IP* with the IP address or host name of the database server.
- *DB* with the name of the database that has been created for use by the compute, usually `nova`.

The database connection string has been set and will be used by the Compute service.

#### 10.3.4.3. Setting the Message Broker

[Report a bug](#)

The Compute services must be explicitly configured with the type, location, and authentication details of the message broker.

All steps in this procedure must be run on each system that will be running Compute services and executed as the root user:

##### 1. General Settings

Use the `openstack-config` utility to set the value of the `rpc_backend` configuration key to `Qpid`.

```
# openstack-config --set /etc/nova/nova.conf \
  DEFAULT rpc_backend nova.openstack.common.rpc.impl_qpid
```

## 2. Configuration Key

Use the `openstack-config` utility to set the value of the `qpid_hostname` configuration key to the host name of the Qpid server.

```
# openstack-config --set /etc/nova/nova.conf \
  DEFAULT qpid_hostname IP
```

Replace *IP* with the IP address or host name of the message broker.

## 3. Authentication Settings

If you have configured Qpid to authenticate incoming connections, you must provide the details of a valid Qpid user in the Compute configuration:

- a. Use the `openstack-config` utility to set the value of the `qpid_username` configuration key to the username of the Qpid user that the Compute services must use when communicating with the message broker.

```
# openstack-config --set /etc/nova/nova.conf \
  DEFAULT qpid_username USERNAME
```

Replace *USERNAME* with the required Qpid user name.

- b. Use the `openstack-config` utility to set the value of the `qpid_password` configuration key to the password of the Qpid user that the Compute services must use when communicating with the message broker.

```
# openstack-config --set /etc/nova/nova.conf \
  DEFAULT qpid_password PASSWORD
```

Replace *PASSWORD* with the password of the Qpid user.

## 4. Encryption Settings

If you configured Qpid to use SSL, you must inform the Compute services of this choice. Use `openstack-config` utility to set the value of the `qpid_protocol` configuration key to `ssl`.

```
# openstack-config --set /etc/nova/nova.conf \
  DEFAULT qpid_protocol ssl
```

The value of the `qpid_port` configuration key must be set to 5671 as Qpid listens on this different port when SSL is in use.

```
# openstack-config --set /etc/nova/nova.conf \
  DEFAULT qpid_port 5671
```



### Important

To communicate with a Qpid message broker that uses SSL the node must also have:

- The `nss` package installed.
- The certificate of the relevant certificate authority installed in the system NSS database (`/etc/pki/nssdb/`).

The `certtool` command is able to import certificates into the NSS database. See the `certtool` manual page for more information (`man certtool`).

The Compute services have been configured to use the message broker and any authentication schemes that it presents.

### 10.3.4.4. Configuring Resource Overcommitment

[Report a bug](#)

OpenStack supports overcommitting of CPU and memory resources on compute nodes.



Overcommitting is a technique of allocating more virtualized CPUs and/or memory than there are physical resources.



### Important

Overcommitting increases the amount of instances you are able to run, but reduces instance performance.

CPU and memory overcommit settings are represented as a ratio. OpenStack uses the following ratios by default:

- ▶ Default CPU overcommit ratio - 16
- ▶ Default memory overcommit ratio - 1.5

These default settings have the following implications:

- ▶ The default CPU overcommit ratio of 16 means that up to 16 virtual cores can be assigned to a node for each physical core.
- ▶ The default memory overcommit ratio of 1.5 means that instances can be assigned to a physical node if the total instance memory usage is less than 1.5 times the amount of physical memory available.

Use the `cpu_allocation_ratio` and `ram_allocation_ratio` directives in `/etc/nova/nova.conf` to change these default settings.

#### 10.3.4.5. Reserving Host Resources

[Report a bug](#)

You can reserve host memory and disk resources as always available to OpenStack. To prevent a given amount of memory and disk resources from being considered as available to be allocated for usage by virtual machines, edit the following directives in `/etc/nova/nova.conf`:

- ▶ `reserved_host_memory_mb` - Defaults to 512MB.
- ▶ `reserved_host_disk_mb` - Defaults to 0MB.

#### 10.3.4.6. Configuring Compute Networking

[Report a bug](#)

##### 10.3.4.6.1. Compute Networking Overview

Unlike Nova-only deployments, when OpenStack networking is in use, the `nova-network` service **must** not run. Instead all network related decisions are delegated to the OpenStack networking Service.

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 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`.

### 10.3.4.6.2. Updating the Compute Configuration

Each time a Compute instance is provisioned or deprovisioned, the service communicates with OpenStack Networking via its API. To facilitate this connection, it is necessary to configure each compute node with the connection and authentication details outlined in this procedure.

These steps must be performed on each Compute node while logged in as the root user.

1. Modify the `network_api_class` configuration key to indicate that the OpenStack Networking service is in use.

```
# openstack-config --set /etc/nova/nova.conf \  
DEFAULT network_api_class nova.network.quantumv2.api.API
```

2. Set the value of the `quantum_url` configuration key to point to the endpoint of the networking API.

```
# openstack-config --set /etc/nova/nova.conf \  
DEFAULT quantum_url http://IP:9696/
```

Replace *IP* with the IP address or host name of the server hosting the API of the OpenStack Networking service.

3. Set the value of the `quantum_admin_tenant_name` configuration key to the name of the tenant used by the OpenStack Networking service. Examples in this guide use *services*.

```
# openstack-config --set /etc/nova/nova.conf \  
DEFAULT quantum_admin_tenant_name services
```

4. Set the value of the `quantum_admin_username` configuration key to the name of the administrative user for the OpenStack Networking service. Examples in this guide use *quantum*.

```
# openstack-config --set /etc/nova/nova.conf \  
DEFAULT quantum_admin_username quantum
```

5. Set the value of the `quantum_admin_password` configuration key to the password associated with the administrative user for the networking service.

```
# openstack-config --set /etc/nova/nova.conf \  
DEFAULT quantum_admin_password PASSWORD
```

6. Set the value of the `quantum_admin_auth_url` configuration key to the URL associated with the identity service endpoint.

```
# openstack-config --set /etc/nova/nova.conf \  
DEFAULT quantum_admin_auth_url http://IP:35357/v3
```

Replace *IP* with the IP address or host name of the Identity service endpoint.

7. Set the value of the `security_group_api` configuration key to `quantum`.

```
# openstack-config --set /etc/nova/nova.conf \  
DEFAULT security_group_api quantum
```

This enables the use of OpenStack Networking security groups.

8. Set the value of the `firewall_driver` configuration key to `nova.virt.firewall.NoopFirewallDriver`.

```
# openstack-config --set /etc/nova/nova.conf \  
DEFAULT firewall_driver nova.virt.firewall.NoopFirewallDriver
```

This **must** be done when OpenStack Networking security groups are in use.

The configuration has been updated and the Compute service will use OpenStack Networking when it is next started.

#### 10.3.4.6.3. Configuring the L2 Agent

[Report a bug](#)

Each compute node must run an instance of the Layer 2 (L2) agent appropriate to the networking plug-in that is in use.

- ▶ [Section 9.7.1, “Configuring the Open vSwitch Plug-in Agent”](#)
- ▶ [Section 9.7.2, “Configuring the Linux Bridge Plug-in Agent”](#)

#### 10.3.4.6.4. Configuring Virtual Interface Plugging

[Report a bug](#)

When nova -compute creates an instance, it must 'plug' each of the vNIC associated with the instance into a OpenStack networking controlled virtual switch. It must also inform the virtual switch of the OpenStack networking port identifier associated with each vNIC.

In previous releases this was done by specifying a driver specific value for the `libvirt_vif_driver` field in the `/etc/nova/nova.conf` configuration file. In Red Hat Enterprise Linux OpenStack Platform 3 a generic virtual interface driver, `nova.virt.libvirt.vif.LibvirtGenericVIFDriver`, is provided. This driver relies on OpenStack networking being able to return the type of virtual interface binding required. These plug-ins support this operation:

- ▶ Linux Bridge
- ▶ Open vSwitch
- ▶ NEC
- ▶ BigSwitch
- ▶ CloudBase Hyper-V
- ▶ brocade

To use the generic driver use the `openstack-config` command to set the value of the `libvirt_vif_driver` configuration key appropriately:

```
# openstack-config --set /etc/nova/nova.conf \
  DEFAULT libvirt_vif_driver \
  nova.virt.libvirt.vif.LibvirtGenericVIFDriver
```



### Important

If using Open vSwitch with security groups enabled then use the Open vSwitch specific driver, `nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver`, instead of the generic driver.



## Important

When using Linux Bridge, you need to add the following to the `/etc/libvirt/qemu.conf` file to ensure that the virtual machine launches properly:

```

user = "root"
group = "root"
cgroup_device_acl = [
    "/dev/null", "/dev/full", "/dev/zero",
    "/dev/random", "/dev/urandom",
    "/dev/ptmx", "/dev/kvm", "/dev/kqemu",
    "/dev/rtc", "/dev/hpet", "/dev/net/tun",
]

```

### 10.3.4.7. Configuring the Firewall

[Report a bug](#)

Systems attempting to use functionality provided by the Compute service do so over the network. Connections to the compute APIs are received on ports 8773, 8774, and 8775.

Connections to virtual machine consoles, whether direct or via the proxy, are received on ports in the range 5900 to 5999.

To allow this the firewall on the service node must be altered to allow network traffic on these ports. All steps in this procedure must be run while logged in to the server hosting the Compute service as the root user. Repeat the process for each compute node.

1. Open the `/etc/sysconfig/iptables` file in a text editor.
2. Add an INPUT rule allowing TCP traffic on ports in the ranges 5900 to 5999 and 8773 to 8775 by adding these lines to the file.

```

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

```

The new rule must appear before any INPUT rules that REJECT traffic.

3. Save the changes to the `/etc/sysconfig/iptables` file.
4. Restart the iptables service to ensure that the change takes effect.

```
# service iptables restart
```

The iptables firewall is now configured to allow incoming connections to the Compute services.

### 10.3.5. Populating the Compute Service Database

[Report a bug](#)

This procedure describes the steps required to populate the Compute service database. These steps must be performed while logged in to a system hosting an instance of the conductor service. The database connection string must already be defined in the configuration of the service.



## Important

This procedure only needs to be followed once to initialize and populate the database. You do not need to perform these steps again when adding additional systems hosting Compute services.

1. Use the `su` command to switch to the nova user.

```
# su nova -s /bin/sh
```

2. Run the `nova-manage db sync` command to initialize and populate the database identified in `/etc/nova/nova.conf`.

```
$ nova-manage db sync
```

The Compute service database has been initialized and populated.

### 10.3.6. Starting the Compute Services

[Report a bug](#)

Editor initialized empty topic content

#### 1. Starting the Message Bus Service

Libvirt requires that the messagebus service be enabled and running.

- a. Use the service command to start the messagebus service.

```
# service messagebus start
```

- b. Use the `chkconfig` command to enable the messagebus service permanently.

```
# chkconfig messagebus on
```

#### 2. Starting the Libvirtd Service

The Compute service requires that the libvirtd service be enabled and running.

- a. Use the service command to start the libvirtd service.

```
# service libvirtd start
```

- b. Use the `chkconfig` command to enable the libvirtd service permanently.

```
# chkconfig libvirtd on
```

#### 3. Starting the API Service

Start the API service on each system that will be hosting an instance of it. Note that each API instance should either have its own endpoint defined in the identity service database or be pointed to by a load balancer that is acting as the endpoint.

- a. Use the service command to start the `openstack-nova-api` service.

```
# service openstack-nova-api start
```

- b. Use the `chkconfig` command to enable the `openstack-nova-api` service permanently.

```
# chkconfig openstack-nova-api on
```

#### 4. Starting the Scheduler

Start the scheduler on each system that will be hosting an instance of it.

- a. Use the service command to start the `openstack-nova-scheduler` service.

```
# service openstack-nova-scheduler start
```

- b. Use the `chkconfig` command to enable the `openstack-nova-scheduler` service permanently.

```
# chkconfig openstack-nova-scheduler on
```

## 5. Starting the Conductor

The conductor is intended to minimize or eliminate the need for Compute nodes to access the database directly. Compute nodes instead communicate with the conductor via a message broker and the conductor handles database access.

Start the conductor on each system that is intended to host an instance of it. Note that it is recommended that this service is not run on each and every Compute node as this eliminates the security benefits of restricting direct database access from the Compute nodes.

- a. Use the service command to start the `openstack-nova-conductor` service.

```
# service openstack-nova-conductor start
```

- b. Use the `chkconfig` command to enable the `openstack-nova-conductor` service permanently.

```
# chkconfig openstack-nova-conductor on
```

## 6. Starting the Compute Service

Start the Compute service on every system that is intended to host virtual machine instances.

- a. Use the service command to start the `openstack-nova-compute` service.

```
# service openstack-nova-compute start
```

- b. Use the `chkconfig` command to enable the `openstack-nova-compute` service permanently.

```
# chkconfig openstack-nova-compute on
```

## 7. Starting Optional Services

Depending on environment configuration you may also need to start these services:

`openstack-nova-cert`

The X509 certificate service, required if you intend to use the EC2 API to the Compute service.

`openstack-nova-network`

The Nova networking service. Note that you **must** not start this service if you have installed and configured, or intend to install and configure, OpenStack networking.

`openstack-nova-objectstore`

The Nova object storage service. It is recommended that the OpenStack Object Storage service (Swift) is used for new deployments.

The Compute services have been started and are ready to accept virtual machine instance requests.

# Chapter 11. Installing the Dashboard

## 11.1. Dashboard Service Requirements

- ▶ The system hosting the Dashboard service must have:
  - The following already installed: `httpd`, `mod_wsgi`, and `mod_ssl` (for security purposes).
  - A connection to the Identity service, as well as to the other OpenStack API services (OpenStack Compute, Block Storage, Object Storage, Image, and Networking services).
- ▶ The installer must know the URL of the Identity service endpoint.



### Note

To install `mod_wsgi`, `httpd`, and `mod_ssl`, execute as root:

```
# yum install -y mod_wsgi httpd mod_ssl
```

## 11.2. Installing the Dashboard Packages

The steps in this procedure install the packages required by the OpenStack Dashboard service.



### Note

The Dashboard service uses a configurable backend session store. This installation uses `memcached` as the session store. However, other options do exist. For more details, refer to *Session Storage Options*.

The only required package is:

**`openstack-dashboard`**

Provides the OpenStack Dashboard service.

If using `memcached`, the following must also be installed:

**`memcached`**

Memory-object caching system, which speeds up dynamic web applications by alleviating database load.

**`python-memcached`**

Python interface to the `memcached` daemon.

This installation must be performed while logged in as the root user.

1. Install the `memcached` object caching system:

```
# yum install -y memcached python-memcached
```

2. Install the Dashboard package:

```
# yum install -y openstack-dashboard
```

The OpenStack Dashboard service is installed and ready to be configured.

[Report a bug](#)

## 11.3. Starting the Apache Web Service

Because the Dashboard is a Django (Python) web application, it is hosted by the httpd service. To start the service, execute the following commands as the root user:

1. To start the service, execute on the command line:

```
# service httpd start
```

2. To ensure that the httpd service starts automatically in the future, execute:

```
# chkconfig httpd on
```

3. You can confirm that httpd is running by executing:

```
# service --status-all | grep httpd
```

[Report a bug](#)

## 11.4. Configuring the Dashboard

### 11.4.1. Configuring Connections and Logging

Before users connect to the dashboard for the first time, the following must be configured in the `/etc/openstack-dashboard/local_settings` file (refer to the sample file in the Appendix):

1. Cache Backend - As the root user, update the CACHES settings with the memcached values:

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
CACHES = {
    'default': {
        'BACKEND' : 'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION' : 'memcacheURL:port',
    }
}
```

Where:

- ▶ `memcacheURL` is the host on which memcache was installed
  - ▶ `port` is the value from the `PORT` parameter in the `/etc/sysconfig/memcached` file.
2. Dashboard Host - Specify the host URL for your OpenStack Identity service endpoint. For example:

```
OPENSTACK_HOST="127.0.0.1"
```

3. Time Zone - To change the dashboard's timezone, update the following (the time zone can also be changed using the dashboard GUI):

```
TIME_ZONE="UTC"
```

4. To ensure the configuration changes take effect, restart the Apache web server.



**Note**

The HORIZON\_CONFIG dictionary contains all the settings for the Dashboard. Whether or not a service is in the Dashboard depends on the Service Catalog configuration in the Identity service. For a full listing, refer to <http://docs.openstack.org/developer/horizon/topics/settings.html> (*Horizon Settings and Configuration*).

**11.4.2. Configuring Secured Deployment (HTTPS)**[Report a bug](#)

Although the default installation uses a non-encrypted channel (HTTP), it is possible to enable SSL support for the OpenStack Dashboard. Use the following steps for enable HTTPS (switch out the example domain 'openstack.example.com' for that of your current setup):

1. Edit the `/etc/openstack-dashboard/local_settings` file, and uncomment the following parameters:

```
SECURE_PROXY_SSL_HEADER = ('HTTP_X_FORWARDED_PROTOCOL', 'https')
CSRF_COOKIE_SECURE = True
SESSION_COOKIE_SECURE = True
```

The latter two settings instruct the browser to only send dashboard cookies over HTTPS connections, ensuring that sessions will not work over HTTP.

2. Edit the `/etc/httpd/conf/httpd.conf` file, and add the following line:

```
NameVirtualHost *:443
```

3. Edit the `/etc/httpd/conf.d/openstack-dashboard.conf` file, and substitute the 'Before' section for 'After':

Before:

```
WSGIScriptAlias /dashboard /usr/share/openstack-
dashboard/openstack_dashboard/wsgi/django.wsgi
Alias /static /usr/share/openstack-dashboard/static/

<Directory /usr/share/openstack-dashboard/openstack_dashboard/wsgi>
  <IfModule mod_deflate.c>
    SetOutputFilter DEFLATE
  <IfModule mod_headers.c>
    # Make sure proxies don't deliver the wrong content
    Header append Vary User-Agent env=!dont-vary
  </IfModule>
</IfModule>

  Order allow,deny
  Allow from all
</Directory>
```

After:

```

<VirtualHost *:80>
  ServerName openstack.example.com
  RedirectPermanent / https://openstack.example.com
</VirtualHost>

<VirtualHost *:443>
  ServerName openstack.example.com
  SSLEngine On
  SSLCertificateFile /etc/httpd/SSL/openstack.example.com.crt
  SSLCACertificateFile /etc/httpd/SSL/openstack.example.com.crt
  SSLCertificateKeyFile /etc/httpd/SSL/openstack.example.com.key
  SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-shutdown
  WSGIScriptAlias / /usr/share/openstack-
dashboard/openstack_dashboard/wsgi/django.wsgi
  WSGIDaemonProcess horizon user=www-data group=www-data processes=3
  threads=10
  Alias /static /usr/share/openstack-dashboard/static/
  <Directory /usr/share/openstack-dashboard/openstack_dashboard/wsgi>
    Order allow,deny
    Allow from all
  </Directory>
</VirtualHost>

```

In the 'After' configuration, Apache listens on port 443 and redirects all non-secured requests to the HTTPS protocol. In the secured section, the private key, the public key, and the certificate are defined for usage.

4. As the root user, restart Apache and memcached:

```
# service httpd restart
# service memcached restart
```

If the HTTP version of the dashboard is used now via the browser, the user should be redirected to the HTTPS version of the page.

### 11.4.3. Creating a Member Role

[Report a bug](#)

The Dashboard service requires a Identity role named the 'Member' role. You must create this role in the Identity service prior to using the dashboard.

1. Log in to the system on which your `keystonerc_admin` file resides and authenticate as the Identity administrator:

```
# source ~/keystonerc_admin
```

2. 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 the Dashboard service to use a role other than the Member role, change the value of the `OPENSTACK_KEYSTONE_DEFAULT_ROLE` configuration key, which is stored in: `/etc/openstack-dashboard/local_settings`  
The `httpd` service must be restarted for the change to take effect.

### 11.4.4. Configuring SELinux

[Report a bug](#)

SELinux is a security feature of Red Hat Enterprise Linux, which provides access control. Possible status values are Enforcing, Permissive, and Disabled. If SELinux is configured in 'Enforcing' mode, you must modify the SELinux policy to allow connections from the `httpd` service to the Identity server. This is also recommended if SELinux is configured in 'Permissive' mode.

1. Use the `getenforce` command to check the status of SELinux on the system:

```
# getenforce
```

2. If the resulting value is 'Enforcing' or 'Permissive', use the `setsebool` command as the root user to allow `httpd-Identity` service connections:

```
# setsebool -P httpd_can_network_connect on
```



## Note

You can also view the status of SELinux using:

```
# sestatus
SELinux status:                enabled
SELinuxfs mount:              /selinux
Current mode:                  permissive
Mode from config file:         enforcing
Policy version:                24
Policy from config file:       targeted
```

For more information, refer to the *Security-Enhanced Linux User Guide for Red Hat Enterprise Linux*.

### 11.4.5. Configuring the Dashboard Firewall

[Report a bug](#)

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.



## Note

To protect authentication credentials and other data, it is highly recommended that you only enable HTTPS connections.

Execute the following as the root user:

1. Edit the `/etc/sysconfig/iptables` configuration file:
  - Allow incoming connections using just HTTPS by adding this firewall rule to the file:

```
-A INPUT -p tcp --dport 443 -j ACCEPT
```

- ▶ Allow incoming connections using both HTTP and HTTPS by adding this firewall rule to the file:

```
-A INPUT -p tcp -m multiport --dports 80,443 -j ACCEPT
```

- Restart the iptables service for the changes to take effect.

```
# service iptables restart
```



## Important

These rules allow communication from all remote hosts to the system running the Dashboard service 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*.

### 11.4.6. Session Storage Options

[Report a bug](#)

#### 11.4.6.1. Configuring Local Memory Cache Session Storage

Local memory storage is the quickest and easiest session backend to set up, because it has no external dependencies. However, it does have two significant drawbacks:

- ▶ No shared storage across processes or workers.
- ▶ No persistence after a process terminates.

The local memory backend is enabled as the default for the Dashboard service solely because it has no dependencies. However, it is not recommended for production use, or even for serious development work.

To use local memory for storage, include the following in the `/etc/openstack-dashboard/local_settings` file:

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.locmem.LocMemCache'
    }
}
```

#### 11.4.6.2. Configuring Memcached Session Storage

[Report a bug](#)

External caching using an application such as memcached offers persistence and shared storage, and can be very useful for small-scale deployment and/or development. The Dashboard installation process in this guide recommends the use of memcached for external caching (for configuration details, refer to *Configuring Connections and Logging*).

However, for distributed and high-availability scenarios, memcached has inherent problems which are beyond the scope of this documentation. Memcached is an extremely fast and efficient cache backend for cases where it fits the deployment need, but it's not appropriate for all scenarios.

#### 11.4.6.3. Configuring Database Session Storage

[Report a bug](#)

Database-backed sessions are scalable (using an appropriate database strategy), persistent, and can be made high-concurrency and highly-available. The downside to this approach is that database-backed sessions are one of the slower session storages, and incur a high overhead

under heavy usage. Proper configuration of your database deployment can also be a substantial undertaking and is far beyond the scope of this documentation.

To enable database session storage, follow the below steps as the root user to initialize the database and configure it for use:

1. Start the MySQL command-line client, by executing:

```
# mysql -u root -p
```

2. Specify the MySQL root user's password when prompted.
3. Create the dash database:

```
mysql> CREATE DATABASE dash;
```

4. Create a MySQL user for the newly-created dash database who has full control of the database.

```
mysql> GRANT ALL ON dash.* TO 'dash'@'%' IDENTIFIED BY 'PASSWORD';
```

```
mysql> GRANT ALL ON dash.* TO 'dash'@'localhost' IDENTIFIED BY 'PASSWORD';
```

Replace *PASSWORD* with a secure password for the new database user to authenticate with.

5. Enter quit at the `mysql>` prompt to exit the MySQL client.
6. In the `/etc/openstack-dashboard/local_settings` file, change the following options to refer to the new MySQL database:

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cached_db'
DATABASES = {
    'default': {
        # Database configuration here
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'dash',
        'USER': 'dash',
        'PASSWORD': 'PASSWORD',
        'HOST': 'HOST',
        'default-character-set': 'utf8'
    }
}
```

Replace *PASSWORD* with the password of the dash database user and replace *HOST* with the IP address or fully qualified domain name of the database server.

7. Populate the new database by executing:

```
# cd /usr/share/openstack-dashboard
# python manage.py syncdb
```

Note: You will be asked to create an admin account; this is not required.

As a result, the following should be displayed:

```
Installing custom SQL ...
Installing indexes ...
DEBUG:django.db.backends:(0.008) CREATE INDEX `django_session_c25c2c28` ON
`django_session` (`expire_date`);; args=()
No fixtures found.
```

8. Restart Apache to pick up the default site and symbolic link settings:

```
# service httpd restart
```

- Restart the `openstack-nova-api` service to ensure the API server can connect to the Dashboard and to avoid an error displayed in the Dashboard.

```
# service openstack-nova-api restart
```

#### 11.4.6.4. Configuring Cached Database Session Storage

[Report a bug](#)

To mitigate the performance issues of database queries, Django's `cached_db` session backend can be used, which utilizes both the database and caching infrastructure to perform write-through caching and efficient retrieval.

Enable this hybrid setting by configuring both your database and cache as discussed above and then using:

```
SESSION_ENGINE = "django.contrib.sessions.backends.cached_db"
```

#### 11.4.6.5. Configuring Cookies Session Storage

[Report a bug](#)

The `cookies-session` backend avoids server load and scaling problems because it stores session data in a cookie, which is stored by the user's browser. The backend uses a cryptographic signing technique, together with the `SECRET_KEY`, to ensure session data is not tampered with during transport (this is not the same as encryption, session data is still readable by an attacker).

- ▶ Advantages:
  - Does not require additional dependencies or infrastructure overhead.
  - Scales indefinitely as long as the quantity of session data being stored fits into a normal cookie.
- ▶ Disadvantages:
  - Places session data into storage on the user's machine and transports it over the wire.
  - Limits the quantity of session data which can be stored.

### Note

For a thorough discussion of the security implications of this session backend, please read the Django documentation on cookie-based sessions: <https://docs.djangoproject.com/en/dev/topics/http/sessions/#s-using-cookie-based-sessions>

To enable cookie-session storage:

- In the `/etc/openstack-dashboard/local_settings` file, set:

```
SESSION_ENGINE = "django.contrib.sessions.backends.signed_cookies"
```

- Add a randomly-generated `SECRET_KEY` to the project by executing on the command line:

```
$ django-admin.py startproject
```

### Note

The `SECRET_KEY` is a text string, which can be specified manually or automatically generated (as in this procedure). You will just need to ensure that the key is unique (that is, does not match any other password on the machine).

## 11.5. Validating the Dashboard Installation

Once the Dashboard has been successfully installed and configured, you can access the user interface with your web browser. Replace `HOSTNAME` with the host name or IP address of the server on which you installed the Dashboard service:

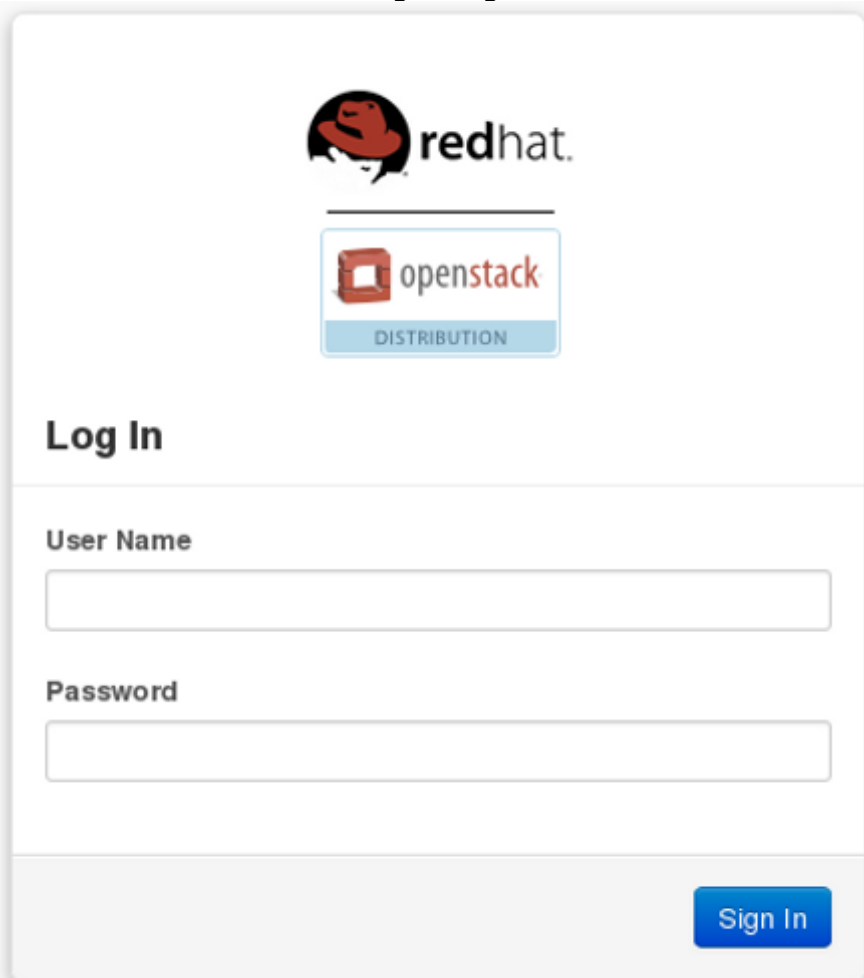
► **HTTPS**

```
https://HOSTNAME/dashboard/
```

► **HTTP**

```
http://HOSTNAME/dashboard/
```

When prompted, log in using the credentials of your OpenStack user. For information on creating an OpenStack user, refer to *Creating a Regular User Account*.



The screenshot shows a web browser window displaying the login page for the Red Hat OpenStack Distribution. At the top, there is the Red Hat logo (a red fedora hat) and the text 'redhat.'. Below that is the OpenStack logo (a red square with a white 'O') and the text 'openstack' and 'DISTRIBUTION'. The main heading is 'Log In'. There are two input fields: 'User Name' and 'Password'. A blue button labeled 'Sign In' is positioned at the bottom right of the form.

**Figure 11.1. Dashboard Login Screen**

**See Also:**

- [Section 5.8, “Creating a Regular User Account”](#)

## **Part III. Validating the Installation**



# Chapter 12. Working with Instances

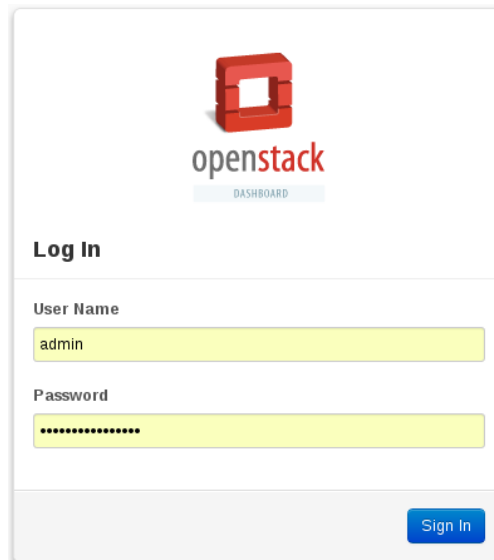
## 12.1. Uploading a Disk Image

You can upload an image by either using the command-line glance tool (refer to *Adding an Image to the Image service*), or by using the dashboard. This procedure uses the dashboard. This means that you must have first:

- ▶ Installed the dashboard (refer to *Installing the Dashboard*).
- ▶ Have an available image for use (refer to *Obtaining a Test Disk Image*).

To upload an image using the dashboard:

1. Log in to the dashboard as a user that has the Member role.



The screenshot shows the OpenStack Dashboard login interface. At the top center is the OpenStack logo, which consists of a red 3D cube with a square cutout in the center, and the text 'openstack' below it. Underneath the logo is a small blue button labeled 'DASHBOARD'. Below this is a section titled 'Log In'. Under the title, there are two input fields. The first is labeled 'User Name' and contains the text 'admin'. The second is labeled 'Password' and contains a series of dots representing a masked password. At the bottom right of the form is a blue button labeled 'Sign In'.

**Figure 12.1. User login**

2. Click **Images & Snapshots** in the **Manage Compute** menu.
3. Click the **Create Image** button. The **Create An Image** dialog is displayed.

**Figure 12.2. Create An Image Dialog**

4. Configure the settings that define your instance on the **Details** tab.
  - a. Enter a name for the image.
  - b. Use the location of your image file in the **Image Location** field.
  - c. Select the correct type from the drop-down menu in the **Format** field (for example, QCOW2).
  - d. Leave the **Minimum Disk (GB)** and **Minimum RAM (MB)** fields empty.
  - e. Select the **Public** box.
  - f. Click the **Create Image** button.

You have successfully uploaded an image.

**See Also:**

► [Chapter 11, Installing the Dashboard](#)

[Report a bug](#)

## 12.2. Creating a Keypair

When a Compute instance is launched, a keypair must be specified, which allows the secure logging in of users into the instance. This section details the steps to create a keypair using the dashboard; this means you must have first installed the dashboard (refer to *Installing the Dashboard*).

On the host running the Compute service:

1. Log in to the dashboard.
2. Click **Access & Security** in the **Manage Compute** menu.
3. On the **Keypairs** tab, click the **Create Keypair** button. The **Create Keypair** dialog is displayed.

**Create Keypair**

Keypair Name

**Description:**  
Keypairs are ssh credentials which are injected into images when they are launched. Creating a new key pair registers the public key and downloads the private key (a .pem file).  
Protect and use the key as you would any normal ssh private key.

Cancel Create Keypair

**Figure 12.3. Create Keypair**

4. Specify a name in the **Keypair Name** field, and click the **Create Keypair** button.

This creates the keypair, which can be used when launching an instance.



### Note

- ▶ When a keypair is created, a keypair file is automatically downloaded through the browser. You can optionally load this file into ssh, for command-line ssh connections, by executing:

```
# ssh-add ~/.ssh/NAME.pem
```

- ▶ To delete an existing keypair, click the keypair's **Delete Keypair** button on the **Keypairs** tab.

### See Also:

- ▶ [Chapter 11, Installing the Dashboard](#)

[Report a bug](#)

## 12.3. Creating a Network

To create a network from the dashboard, you must have first:

- ▶ Installed the dashboard (refer to *Installing the Dashboard*).
- ▶ Installed OpenStack Networking Services (refer to *Installing the OpenStack Networking Service*).

To create a network using the dashboard:

1. Log in to the dashboard.
2. Click **Networks** in the **Manage Network** menu.
3. Click the **Create Network** button. The **Create Network** dialog is displayed.

**Figure 12.4. Create Network: Network Tab**

4. By default, the dialog opens to the **Network** tab. You have the option of specifying a network name.
5. To define the network's subnet, click on the **Subnet** and **Subnet Detail** tabs. Click into each field for field tips.



### Note

You do not have to initially specify a subnet (although this will result in any attached instance having the status of 'error'). If you do not define a specific subnet, clear the **Create Subnet** check box.

6. Click the **Create** button.

You have now created a new network.

### See Also:

- ▶ [Chapter 11, Installing the Dashboard](#)
- ▶ [Chapter 9, Installing the OpenStack Networking Service](#)

[Report a bug](#)

## 12.4. Launching an Instance

To launch an instance from the dashboard you must have first:

- ▶ Uploaded an image to use as the basis for your instances (refer to *Uploading a Disk Image*).
- ▶ Created a network (refer to *Creating a Network*).

To launch an instance using the dashboard:

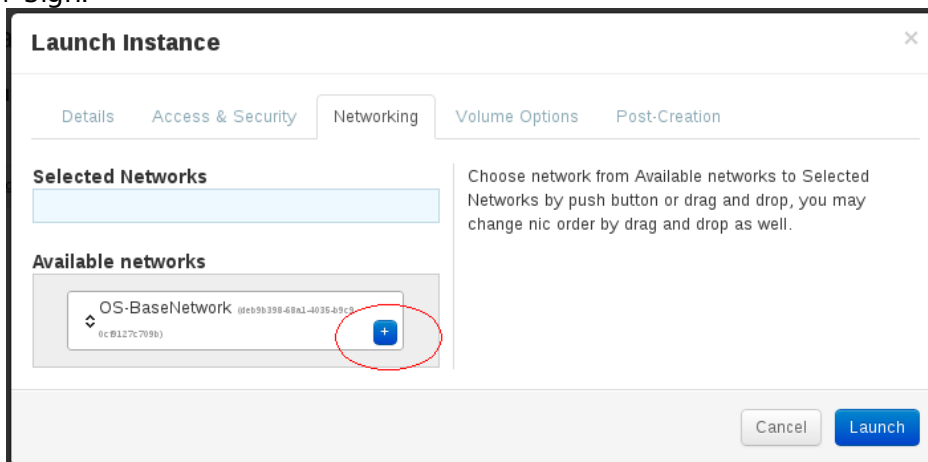
1. Log in to the dashboard.
2. Click **Instances** in the **Manage Compute** menu.
3. Click the **Launch Instance** button. The **Launch Instance** dialog is displayed.

**Figure 12.5. Launch Instance: Details Tab**

4. By default, the dialog opens to 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 determines the compute resources available to your instance. After a flavor is selected, their resources are displayed in the **Flavor Details** pane for preview.
  - e. Enter an **Instance Count**. This determines how many instances to launch using the selected options.
5. Click the **Access & Security** tab and configure the security settings for your instance:

**Figure 12.6. Launch Instance: Access and Security Tab**

- a. Either select an existing keypair from the **Keypair** drop down box, or 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 the **Networking** tab and select the network for the instance by clicking on the network's **+** sign:



**Figure 12.7. Launch Instance: Networking Tab**

7. Click the **Launch** button.

You have just created a Compute instance.



## Note

To launch the instance console from the Dashboard:

1. On the **Instances** tab, click the name of your instance. The **Instance Detail** page is displayed.
2. Click the **Console** tab on the resultant page.

An instance of the VNC console is run within the browser.

## See Also:

- ▶ [Section 12.1, "Uploading a Disk Image"](#)
- ▶ [Section 12.2, "Creating a Keypair"](#)
- ▶ [Section 12.3, "Creating a Network"](#)

[Report a bug](#)

## 12.5. Creating a Volume

Compute instances support the attachment and detachment of block storage volumes. This procedure details the steps involved in creating a logical volume using the dashboard.

To create a volume from the dashboard, you must have first:

- ▶ Installed the dashboard (refer to *Installing the Dashboard*).
- ▶ Installed the Block Storage service (refer to *Installing OpenStack Block Storage*).

To create a volume using the dashboard:

1. Log in to the dashboard.
2. Click **Volumes** in the **Manage Compute** menu.
3. Click the **Create Volume** button. The **Create Volume** dialog is displayed:

**Figure 12.8. Create Volume Dialog**

4. To configure the 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

In this guide, LVM storage is configured as the cinder-volumes volume group (refer to *Configuring for LVM Storage Backend*). There must be enough free disk space in the cinder-volumes volume group for your new volume to be allocated.

5. Click the **Create Volume** button to create the new volume.

You have successfully created a Block Storage volume using the dashboard.

### See Also:

- ▶ [Chapter 11, Installing the Dashboard](#)
- ▶ [Chapter 8, Installing OpenStack Block Storage](#)

[Report a bug](#)

## 12.6. Attaching a Volume to an Instance

This procedure details the steps involved in attaching a Block Storage volume to an existing Compute instance using the dashboard.

To create a volume from the dashboard, you must have first:

- ▶ Launched an instance (refer to *Launching an Instance*).
- ▶ Created a volume (refer to *Creating a Volume*).

To attach a volume to an instance using the dashboard:

1. Log in to the dashboard as a user.
2. Click **Volumes** in the **Manage Compute** menu.
3. 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.

**Figure 12.9. Manage Volume Attachments**

4. Select the instance for the volume in the **Attach to Instance** field.
5. Specify the device name in the **Device Name** field (for example, '/dev/vdc').
6. Click the **Attach Volume** button.

You have successfully attached a Block Storage volume to an instance using the dashboard. The volume will appear as a physical hard disk drive to the guest operating system.

#### See Also:

- ▶ [Section 12.4, “Launching an Instance”](#)
- ▶ [Section 12.5, “Creating a Volume”](#)

[Report a bug](#)

## 12.7. Creating an Instance Snapshot

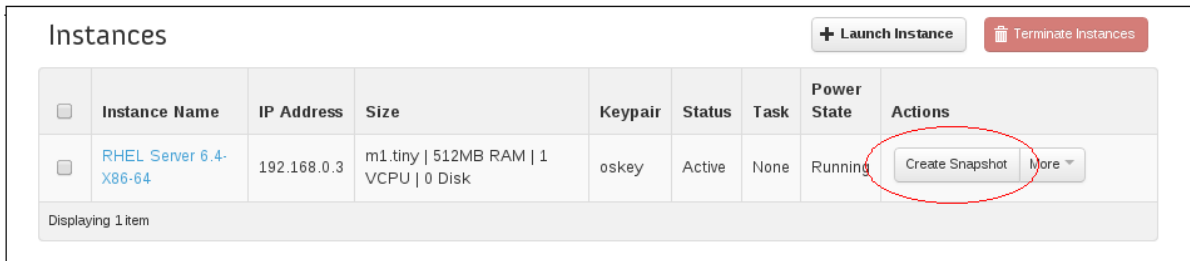
This procedure details the steps involved in creating a snapshot based on a running instance using the dashboard. This may be done for backup purposes or for creating a base image to create other instances after applying customization.

To create a snapshot, a running instance must be available (refer to *Launching an Instance*).

To create a snapshot using the dashboard:

1. Log in to the dashboard.
2. Click **Instances** in the **Manage Compute** menu.
3. Click the **Create Snapshot** button on the row associated with the instance of which you want to take a snapshot.





**Figure 12.10. Creating a Snapshot**

The **Create Snapshot** dialog is displayed.

4. Enter a descriptive name for your snapshot in the **Snapshot Name** field.
5. Click the **Create Snapshot** button to create the snapshot.

Your new snapshot will appear in the **Image Snapshots** table in the **Images & Snapshots** screen.

You have successfully created a snapshot of your instance; the snapshot can be used to restore an instance state or as a basis for spawning new instances.

**See Also:**

- ▶ [Section 12.4, “Launching an Instance”](#)

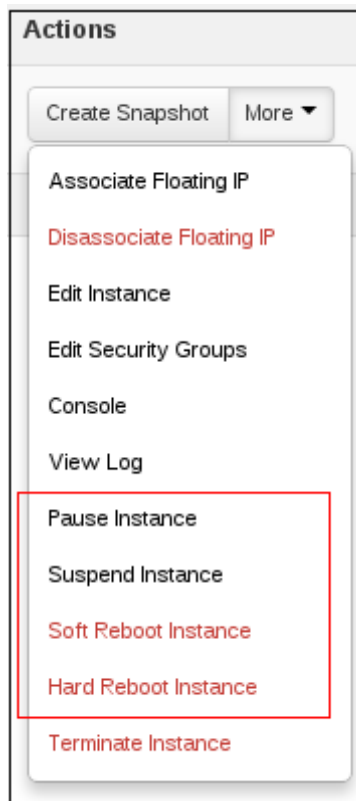
[Report a bug](#)

## 12.8. Controlling the State of an Instance (Pause, Suspend, Reboot)

To change the state of an instance using the dashboard, you must have first launched an instance (refer to *Launching an Image*).

To change the state of an instance:

1. Log in to the dashboard.
2. Click **Instances** in the **Manage Compute** menu.
3. Select the instance for which you want to change the state and click on the **More** dropdown button. The dropdown list is displayed.



**Figure 12.11. Instance Menu**

4. Click one of the options to change the instance state.

You have successfully changed the state of the instance.

**See Also:**

- ▶ [Section 12.4, “Launching an Instance”](#)

# Chapter 13. Updating the Environment

## 13.1. Defining a Floating IP-Address Pool

By default, each virtual instance is automatically assigned a private IP address in the network to which it is assigned. You may optionally assign public IP addresses to instances.

OpenStack uses the term "floating IP" to refer to an IP address that can be dynamically added to a running virtual instance. In OpenStack Networking, a floating IP pool is represented as an external network and a floating IP is allocated from a subnet associated with the external network.

For this procedure, you must first have installed OpenStack Networking (refer to *Installing the OpenStack Networking Service*).

To define a pool of floating IP addresses:

1. Create an external network for the pool:

```
# quantum net-create networkName --router:external=True
```

### Example 13.1. Defining an External Network

```
# quantum net-create ext-net --router:external=True
Created a new network:
+-----+-----+
| Field                | Value                                |
+-----+-----+
| admin_state_up      | True                                  |
| id                   | 3a53e3be-bd0e-4c05-880d-2b11aa618aff |
| name                 | ext-net                              |
| provider:network_type | local                                 |
| provider:physical_network |                                       |
| provider:segmentation_id |                                       |
| router:external      | True                                  |
| shared               | False                                 |
| status                | ACTIVE                               |
| subnets              |                                       |
| tenant_id            | 6b406408dff14a2ebf6d2cd7418510b2    |
+-----+-----+
```

2. Create the pool of floating IP addresses:

```
$ quantum subnet-create --allocation-pool start=IPStart,end=IPStart --
gateway GatewayIP --disable-dhcp networkName CIDR
```

**Example 13.2. Defining a Pool of Floating IP Addresses**

```
$ quantum subnet-create --allocation-pool
start=10.38.15.128,end=10.38.15.159 --gateway 10.38.15.254 --disable-dhcp
ext-net 10.38.15.0/24
Created a new subnet:
+-----+
| Field          | Value                                                                 |
+-----+-----+
| allocation_pools | {"start": "10.38.15.128", "end": "10.38.15.159"} |
| cidr            | 10.38.15.0/24                                                       |
| dns_nameservers |                                                                 |
| enable_dhcp     | False                                                                |
| gateway_ip      | 10.38.15.254                                                         |
| host_routes     |                                                                 |
| id              | 6a15f954-935c-490f-a1ab-c2a1c1b1529d                               |
| ip_version      | 4                                                                    |
| name            |                                                                 |
| network_id      | 4ad5e73b-c575-4e32-b581-f9207a22eb09                               |
| tenant_id       | e5be83dc0a474eeb92ad2cda4a5b94d5                                   |
+-----+-----+
```

You have successfully created a pool of floating IP addresses.

**See Also:**

- ▶ [Chapter 9, Installing the OpenStack Networking Service](#)

[Report a bug](#)

## 13.2. Creating a Router

This section details the steps to create a router using the dashboard, which connects an internal network to an external one. You must first have:

- ▶ Installed the dashboard (refer to *Installing the Dashboard*).
- ▶ Created an external network (refer to *Defining a Floating IP-Address Pool*).
- ▶ Created an internal network (refer to *Creating a Network*).

To create a router:

1. Log in to the dashboard.
2. Click **Routers** in the **Manage Network** menu.
3. Click on the **Create Router** button. The **Create Router** window is displayed:

**Figure 13.1. Create Router**

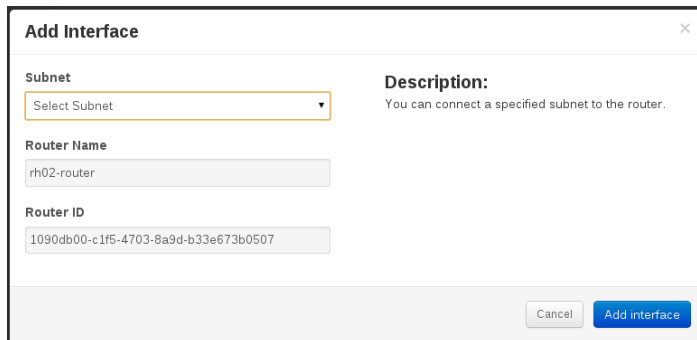
4. Specify the router's name and click the **Create router** button. The new router is now displayed in the router list.
5. Click the new router's **Set Gateway** button.
6. Specify the network to which the router will connect in the **External Network** field, and click the **Set Gateway** button.

7. To connect a private network to the newly created router:
  - a. Click on the router name in the router list:



**Figure 13.2. Select the router**

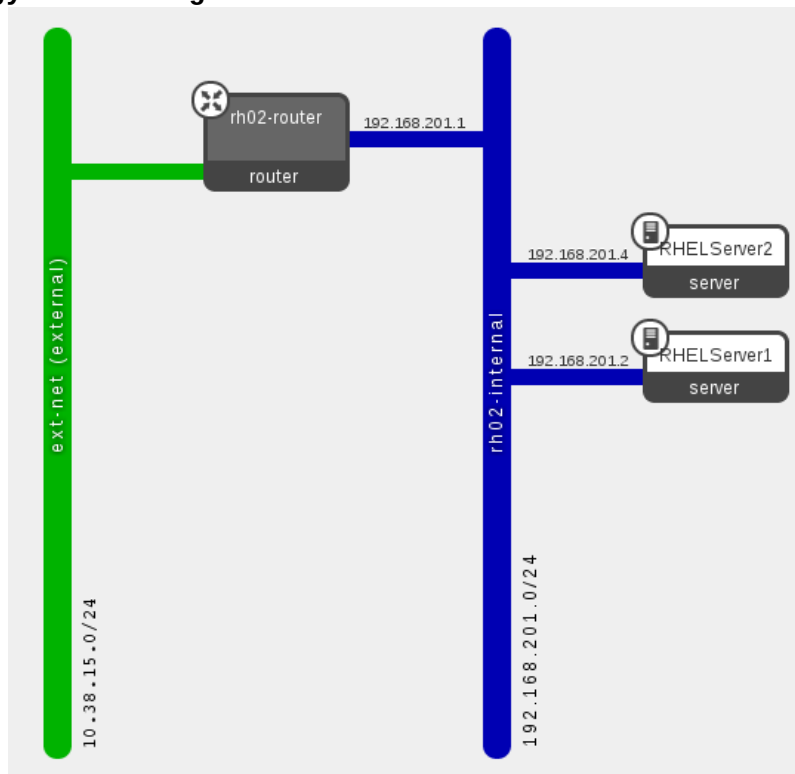
- b. Click the **Add Interface** button. The **Add Interface** window is displayed:



**Figure 13.3. Add Interface**

- c. Specify the new subnet to which the router should be attached in the **Subnet** field, and click **Add Interface**.

You have successfully created the router; you can view the new topology by clicking on **Network Topology** in the **Manage Network** menu.



**Figure 13.4. Network Topology**

**See Also:**

- ▶ [Chapter 11, \*Installing the Dashboard\*](#)
- ▶ [Section 13.1, “Defining a Floating IP-Address Pool”](#)
- ▶ [Section 12.3, “Creating a Network”](#)

[Report a bug](#)

## 13.3. Associating a Floating IP with the Instance

When an instance is created in OpenStack, it is automatically assigned a fixed IP address in the network to which the instance is assigned. This IP address is permanently associated with the instance until the instance is terminated.

However, a floating IP address can also be attached to an instance (in addition to their fixed IP address). 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. 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 instance.

To associate a floating IP address, you must have first:

- ▶ Created a pool of floating IP addresses (refer to *Defining a Floating IP-Address Pool*).
- ▶ Launched an instance (refer to *Launching an Instance*).

To associate an instance with a floating IP address:

1. Log in to the dashboard as a user that has the Member role.
2. Click **Access & Security** in the **Manage Compute** 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:

**Figure 13.5. Manage Floating IP Associations**

7. The **IP Address** field is automatically set to the selected floating IP address.

Select the instance (with which to associate the floating IP address) from the **Port to be associated** 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.

### See Also:

- ▶ [Section 13.1, “Defining a Floating IP-Address Pool”](#)
- ▶ [Section 12.4, “Launching an Instance”](#)

[Report a bug](#)

## 13.4. Adding a Rule to a Security Group

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. You must first have:

- ▶ Installed the dashboard (refer to *Installing the Dashboard*).
- ▶ Installed OpenStack Networking (refer to *Installing the OpenStack Networking Service*).



### Note

In the default configuration, the 'default' security group accepts all connections from the 'default' source; all instances with the 'default' group can talk to each other on any port.

To add a new rule using the dashboard:

1. Log into the dashboard.
2. Click **Access & Security** in the **Manage Compute** 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. Click the **Add Rule** button. The **Add Rule** window is displayed.
5. Configure the rule:
  - a. Select the protocol to which the rule must apply from the **IP Protocol** list.
  - b. Define the port or ports to which the rule will apply using the **Open** field:
    - ▶ Port - Define a specific port in the **Port** field
    - ▶ Port Range - Define the port range using the **From Port** and **To Port** fields.
  - c. Define the IP address from which connections should be accepted on the defined port using the **Source** field:
    - ▶ CIDR - Enter a specific IP address in the **CIDR** field using the Classless Inter-Domain Routing (CIDR) notation. A value of 0.0.0.0/0 allows connections from all IP addresses.
    - ▶ Security Group - Select an existing security group from the **Source Group** drop-down list. This allows connections from any instances from the specified security group.
6. 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.

**See Also:**

- ▶ [Chapter 11, \*Installing the Dashboard\*](#)
- ▶ [Chapter 9, \*Installing the OpenStack Networking Service\*](#)



# Part IV. Monitoring the OpenStack Environment

[Report a bug](#)

# Chapter 14. Monitoring OpenStack using Nagios

## 14.1. Installing Nagios

### 14.1.1. Installing the Nagios Service

The Nagios monitoring system can be used to provide monitoring and alerts for the OpenStack network and infrastructure. The following installation procedure installs:

***nagios***

Nagios program that monitors hosts and services on the network, and which can send email or page alerts when a problem arises and when a problem is resolved.

***nagios-devel***

Includes files which can be used by Nagios-related applications.

***nagios-plugins\****

Nagios plugins for Nagios-related applications (including ping and nrpe).

***gd and gd-devel***

gd Graphics Library, for dynamically creating images, and the gd development libraries for gd.

***php***

HTML-embedded scripting language, used by Nagios for the web interface.

***gcc, glibc, and glibc-common***

GNU compiler collection, together with standard programming libraries and binaries (including locale support).

***openssl***

OpenSSL toolkit, which provides support for secure communication between machines.

Install the required packages as the root user, using the yum command:

```
# yum install nagios nagios-devel nagios-plugins* gd gd-devel php gcc glibc  
glibc-common openssl
```

**Note**

If any of the packages is not immediately available (for example, `gd-devel` or `gcc`), you might have to enable the optional Red Hat channel using `subscription-manager`:

```
# subscription-manager repos --enable rhel-6-server-optional-rpms
```

**14.1.2. Installing the NRPE Addon**[Report a bug](#)

NRPE (Nagios Remote Plugin Executor) plugins are compiled executables or scripts that are used to check the status of a host's service, and report back to the Nagios service. If the OpenStack cloud is distributed across machines, the NRPE addon can be used to run access plugin information on those remote machines.

NRPE and the Nagios plugins must be installed on each remote machine to be monitored. On the remote machine, and as the root user, execute the following:

```
# yum install -y nrpe nagios-plugins* openssl
```

After the installation, you can view all available plugins in the `/usr/lib64/nagios/plugins` directory (depending on the machine, they may be in `/usr/lib/nagios/plugins`).

**Note**

SSH can also be used to access remote Nagios plugins. However, this can result in too high a CPU load on both the Nagios host and remote machine, and is not recommended.

[Report a bug](#)**14.2. Configuring Nagios****14.2.1. Setting up Nagios**

Nagios is composed of a server, plugins that report object/host information from both local and remote machines back to the server, a web interface, and configuration that ties all of it together.

At a minimum, the following must be done:

1. Check web-interface user name and password, and check basic configuration.
2. Add OpenStack monitoring to the local server.
3. If the OpenStack cloud includes distributed hosts:
  - a. Install and configure NRPE on each remote machine (that has services to be monitored).
  - b. Tell Nagios which hosts are being monitored.
  - c. Tell Nagios which services are being monitored for each host.

**Table 14.1. Nagios Configuration Files**

File Name	Description
/etc/nagios/nagios.cfg	Main Nagios configuration file.
/etc/nagios/cgi.conf	CGI configuration file.
/etc/httpd/conf.d/nagios.conf	Nagios configuration for httpd.
/etc/nagios/passwd	Password file for Nagios users.
/usr/local/nagios/etc/ResourceName.cfg	Contains user-specific settings.
/etc/nagios/objects/ObjectsDirectory/ObjectsFile.cfg	Object definition files that are used to store information about items such as services or contact groups.
/etc/nagios/nrpe.cfg	NRPE configuration file.

### 14.2.2. Configuring HTTPD

[Report a bug](#)

By default, when Nagios is installed, the default httpd user and password is: nagiosadmin / nagiosadmin. This value can be viewed in the /etc/nagios/cgi.conf file.

To configure HTTPD for Nagios, execute the following as the root user:

1. To change the default password for the user nagiosadmin, execute:

```
# htpasswd -c /etc/nagios/passwd nagiosadmin
```



#### Note

To create a new user, use the following command with the new user's name:

```
# htpasswd /etc/nagios/passwd newUser
```

2. Update the nagiosadmin email address in /etc/nagios/objects/contacts.cfg

```
define contact{
    contact_name    nagiosadmin          ; Short name of user
    [...snip...]
    email           yourName@example.com ; <<*****CHANGE THIS*****>>
}
```

3. Verify that the basic configuration is working:

```
# nagios -v /etc/nagios/nagios.cfg
```

If errors occur, check the parameters set in /etc/nagios/nagios.cfg

4. Ensure that Nagios is started automatically when the system boots.

```
# chkconfig --add nagios
# chkconfig nagios on
```

5. Start up Nagios and restart httpd:

```
# service httpd restart
# service nagios start
```

6. Check your Nagios access by using the following URL in your browser, and using the nagiosadmin user and the password that was set in Step 1:

```
http://nagiosHostURL/nagios
```

**Figure 14.1. Nagios Login**

## Note

If the Nagios URL cannot be accessed, ensure your firewall rules has been set up correctly (refer to *Configuring the Dashboard Firewall*).

### 14.2.3. Configuring OpenStack Services

[Report a bug](#)

By default, on the Nagios server, the `/etc/nagios/objects/localhost.cfg` file is used to define services for basic local statistics (for example, swap usage or the number of current users). You can always comment these services out if they are no longer needed by prefacing each line with a '#' character. This same file can be used to add new OpenStack monitoring services.

## Note

Additional service files can be used, but they must be specified as a `cfg_file` parameter in the `/etc/nagios/nagios.cfg` file.

To add an OpenStack service into the list of monitored services, execute the following as the root user:

1. Write a short script for the item to be monitored (for example, whether a service is running), and place it in the `/usr/lib64/nagios/plugins` directory.  
For example, the following script checks the number of Compute instances, and is stored in a file named `nova-list`:

```
#!/bin/env bash
export OS_USERNAME=userName
export OS_TENANT_NAME=tenantName
export OS_PASSWORD=password
export OS_AUTH_URL=http://identityURL:35357/v2.0/

data=$(nova list 2>&1)
rv=$?

if [ "$rv" != "0" ] ; then
    echo $data
    exit $rv
fi

echo "$data" | grep -v -e '-----' -e '| Status |' -e '^$' | wc -l
```

2. Ensure the script is executable:

```
# chmod u+x nova-list
```

3. In the `/etc/nagios/objects/commands.cfg` file, specify a command section for each new script:

```
define command {
    command_line          /usr/lib64/nagios/plugins/nova-list
    command_name          nova-list
}
```

4. In the `/etc/nagios/objects/localhost.cfg` file, define a service for each new item, using the defined command. For example:

```
define service {
    check_command      nova-list
    host_name          localURL
    name               nova-list
    normal_check_interval 5
    service_description Number of nova vm instances
    use                 generic-service
}
```

5. Restart nagios using:

```
# service nagios restart
```

## 14.2.4. Configuring NRPE

[Report a bug](#)

To set up monitoring on each remote machine, execute the following as the root user:

1. In the `/etc/nagios/nrpe.cfg` file, add the central Nagios server IP address in the `allowed_hosts` line:

```
allowed_hosts=127.0.0.1, NagiosServerIP
```

2. In the `/etc/nagios/nrpe.cfg` file, add any commands to be used to monitor the OpenStack services. For example:

```
command[keystone]=/usr/lib64/nagios/plugins/check_procs -c 1: -w 3: -C
keystone-all
```

Each defined command can then be specified in the `services.cfg` file on the Nagios monitoring server (refer to *Creating Service Definitions*).



## Note

Any complicated monitoring can be placed into a script, and then referred to in the command definition. For an OpenStack script example, refer to *Configuring OpenStack Services*.

- Open up the machine's NRPE port:

```
# iptables -I INPUT -p tcp --dport 5666 -j ACCEPT
# iptables-save > /etc/sysconfig/iptables
```

- Start the NRPE service:

```
# service nrpe start
```

### 14.2.5. Creating Host Definitions

[Report a bug](#)

If additional machines are being used in the cloud, in addition to the host on which Nagios is installed, they must be made known to Nagios by configuring them in an objects file.

Execute the following as the root user:

- In the `/etc/nagios/objects` directory, create a `hosts.cfg` file.
- In the file, specify a host section for each machine on which an OpenStack service is running and should be monitored:

```
define host{
    use linux-server
    host_name remoteHostName
    alias remoteHostAlias
    address remoteAddress
}
```

where:

- ▶ `host_name` = Name of the remote machine to be monitored (typically listed in the local `/etc/hosts` file). This name is used to reference the host in service and host group definitions.
- ▶ `alias` = Name used to easily identify the host (typically the same as the `host_name`).
- ▶ `address` = Host address (typically its IP address, although a FQDN can be used instead, just make sure that DNS services are available).

For example:

```
define host{
    host_name      Server-ABC
    alias  OS-ImageServices
    address 192.168.1.254
}
```

- In the `/etc/nagios/nagios.cfg` file, under the `OBJECT CONFIGURATION FILES` section, specify the following line:

```
cfg_file=/etc/nagios/objects/hosts.cfg
```

### 14.2.6. Creating Service Definitions

[Report a bug](#)

To monitor remote services, you must create a new file, `/etc/nagios/objects/services.cfg` (as the root user):

1. In the `/etc/nagios/objects/commands.cfg` file, specify the following to handle the use of the `check_nrpe` plugin with remote scripts or plugins:

```
define command{
    command_name    check_nrpe
    command_line    $USER1$/check_nrpe -H $HOSTADDRESS$ -c $ARG1$
}
```

2. In the `/etc/nagios/objects` directory, create the `services.cfg` file.
3. In the file, specify the following service sections for each remote OpenStack host to be monitored:

```
##Basic remote checks#####
##Remember that remoteHostName is defined in the hosts.cfg file.

define service{
    use generic-service
    host_name remoteHostName
    service_description PING
    check_command check_ping!100.0,20%!500.0,60%
}

define service{
    use generic-service
    host_name remoteHostName
    service_description Load Average
    check_command check_nrpe!check_load
}

##OpenStack Service Checks#####
define service{
    use generic-service
    host_name remoteHostName
    service_description Identity Service
    check_command check_nrpe!keystone
}
```

The above sections ensure that a server heartbeat, load check, and the OpenStack Identity service status are reported back to the Nagios server. All OpenStack services can be reported, just ensure that a matching command is specified in the remote server's `nrpe.cfg` file.

4. In the `/etc/nagios/nagios.cfg` file, under the `OBJECT CONFIGURATION FILES` section, specify the following line:

```
cfg_file=/etc/nagios/objects/services.cfg
```

### 14.2.7. Verifying the Configuration

[Report a bug](#)

Execute the following as the root user:

1. Verify that the updated configuration is working:

```
# nagios -v /etc/nagios/nagios.cfg
```

If errors occur, check the parameters set in `/etc/nagios/nagios.cfg`, `/etc/nagios/services.cfg`, and `/etc/nagios/hosts.cfg`.

2. Restart Nagios:

```
# service nagios restart
```



3. Log into the Nagios dashboard again by using the following URL in your browser, and using the nagiosadmin user and the password that was set in Step 1:

`http://nagiosHostURL/nagios`

# Chapter 15. Installing and Configuring Remote Logging

## 15.1. Introduction to Remote Logging

All systems generate and update log files recording their actions and any problems they encounter. In a distributed or cloud computing environment that contains many systems collecting these log files in a central location simplifies debugging.

The `rsyslog` service provides facilities both for running a centralized logging server and for configuring individual systems to send their log files to the centralized logging server. This is referred to as configuring the systems for "remote logging".

[Report a bug](#)

## 15.2. Installing `rsyslog` Server

The `rsyslog` package must be installed on the system that you intend to use as a centralized logging server and all systems that will be configured to send logs to it.

- ▶ While logged in as the root user install the `rsyslog` package. Using the `yum` command.

```
# yum install rsyslog
```

The `rsyslog` package is installed and ready to be configured.

[Report a bug](#)

## 15.3. Configuring `rsyslog` on the Centralized Logging Server

The steps in this procedure must be followed on the system that you intend to use as your centralized logging sever. All steps in this procedure must be run while logged in as the root user.

1. Configure SELinux to allow **`rsyslog`** traffic.

```
# semanage -a -t syslogd_port_t -p udp 514
```

2. Configure the `iptables` firewall to allow **`rsyslog`** traffic.
  - a. Open the `/etc/sysconfig/iptables` file in a text editor.
  - b. Add an INPUT rule allowing UDP traffic on port 514 to the file. The new rule must appear before any INPUT rules that REJECT traffic.

```
-A INPUT -m state --state NEW -m udp -p udp --dport 514 -j ACCEPT
```

- c. Save the changes to the `/etc/sysconfig/iptables` file.
- d. Restart the `iptables` service for the firewall changes to take effect.

```
# service iptables restart
```

3. Open the `/etc/rsyslog.conf` file in a text editor.
  - a. Add this line to the file, defining the location logs will be saved to:

```
$template TmplAuth, "/var/log/%HOSTNAME%/%PROGRAMNAME%.log"
authpriv.*    ?TmplAuth
*.info,mail.none,authpriv.none,cron.none    ?TmplMsg
```

- b. Remove the comment character (#) from the beginning of these lines in the file:

```
#$ModLoad imudp
#$UDPServerRun 514
```

Save the changes to the `/etc/rsyslog.conf` file.

Your centralized log server is now configured to receive and store log files from the other systems in your environment.

[Report a bug](#)

## 15.4. Configuring rsyslog on the Individual Nodes

Apply the steps listed in this procedure to each of your systems to configure them to send logs to a centralized log server. All steps listed in this procedure must be performed while logged in as the root user.

- Edit the `/etc/rsyslog.conf`, and specify the address of your centralized log server by adding the following:

```
*.* @YOURSERVERADDRESS:YOURSERVERPORT
```

Replace `YOURSERVERADDRESS` with the address of the centralized logging server. Replace `YOURSERVERPORT` with the port on which the `rsyslog` service is listening. For example:

```
*.* @192.168.20.254:514
```

Or:

```
*.* @log-server.company.com:514
```

The single `@` specifies the UDP protocol for transmission. Use a double `@@` to specify the TCP protocol for transmission.



### Important

The use of the wildcard `*` character in these example configurations indicates to `rsyslog` that log entries from all log facilities and of all log priorities must be sent to the remote `rsyslog` server.

For information on applying more precise filtering of log files refer to the manual page for the `rsyslog` configuration file, `rsyslog.conf`. Access the manual page by running the command `man rsyslog.conf`.

Once the `rsyslog` service is started or restarted the system will send all log messages to the centralized logging server.

[Report a bug](#)

## 15.5. Starting rsyslog Server

The `rsyslog` service must be running on both the centralized logging server and the systems attempting to log to it.

The steps in this procedure must be performed while logged in as the root user.

1. Use the service command to start the `rsyslog` service.

```
# service rsyslog start
```

2. Use the `chkconfig` command to ensure the `rsyslog` service starts automatically in future.

```
# chkconfig rsyslog on
```

The `rsyslog` service has been started. The service will start sending or receiving log messages based on its local configuration.

[Report a bug](#)

# Part V. Managing OpenStack Environment Expansion

# Chapter 16. Managing Compute Expansion

## 16.1. Defining Regions

Each service cataloged in the Identity service is identified by its region, which typically represents a geographical location, and its endpoint. In a cloud with multiple Compute deployments, regions allow for the discrete separation of services, and are a robust way to share some infrastructure between Compute installations, while allowing for a high degree of failure tolerance.

Administrators determine which services are shared between regions and which services are used only with a specific region. By default when an endpoint is defined and no region is specified it is created in the region named `regionOne`.

To begin using separate regions specify the `--region` argument when adding service endpoints.

```
$ keystone endpoint-create --region REGION \
  --service-id SERVICEID \
  --publicurl PUBLICURL \
  --adminurl ADMINURL \
  --internalurl INTERNALURL
```

Replace `REGION` with the name of the region that the endpoint belongs to. When sharing an endpoint between regions create an endpoint entry containing the same URLs for each applicable region. For information on setting the URLs for each service refer to the identity service configuration information of the service in question.

### Example 16.1. Endpoints within Discrete Regions

In this example the APAC and EMEA regions share an identity server (`identity.example.com`) endpoint while providing region specific compute API endpoints.

```
$ keystone endpoint-list
+-----+-----+-----+
| id      | region | publicurl                                     |
+-----+-----+-----+
| 0d8b... | APAC   | http://identity.example.com:5000/v3         |
| 769f... | EMEA   | http://identity.example.com:5000/v3         |
| 516c... | APAC   | http://nova-apac.example.com:8774/v2/$(tenant_id)s |
| cf7e... | EMEA   | http://nova-emea.example.com:8774/v2/$(tenant_id)s |
+-----+-----+-----+
```

[Report a bug](#)

## 16.2. Adding Compute Resources

Adding Compute resources is the most common way to expand the OpenStack environment. This is done by adding additional nodes that host, at a minimum, an instance of the Compute (`openstack-nova-compute`) service. Once the Compute service is configured and running it communicates with other nodes in the environment, including Compute API endpoints and Compute conductors, via the message broker.

To add additional compute nodes repeat the steps performed when installing the original compute node:

- ▶ [Section 10.3.3, “Installing the Compute Service”](#)
- ▶ [Section 10.3.4.3, “Setting the Message Broker”](#)
- ▶ [Section 10.3.4.4, “Configuring Resource Overcommitment”](#)
- ▶ [Section 10.3.4.5, “Reserving Host Resources”](#)
- ▶ [Section 10.3.4.6.2, “Updating the Compute Configuration”](#)
- ▶ [Section 10.3.4.6.3, “Configuring the L2 Agent”](#)
- ▶ [Section 10.3.4.6.4, “Configuring Virtual Interface Plugging”](#)
- ▶ [Section 10.3.4.7, “Configuring the Firewall”](#)
- ▶ [Section 10.3.6, “Starting the Compute Services”](#)

If you wish to run an instance of the Compute conductor service (`openstack-nova-conductor`) then you must also ensure that the service is configured to access the compute database, refer to [Section 10.3.4.2, “Setting the Database Connection String”](#) for more information.

Once the additional instance of the Compute service has been started the node will begin accepting requests to launch virtual machine instances. Use the `nova service-list` while authenticated as the OpenStack administrator (using a `keystonerc` file) to confirm the status of the new node.

[Report a bug](#)

## 16.3. Safely Removing Compute Resources

Removal of a compute node must be done carefully to prevent negative consequences for virtual machine instances running on that node. There must be capacity available on other compute nodes within the environment to run these virtual machine instances while the node being removed is out of service.

The steps listed in this procedure must be executed by a user who has access to the credentials of an OpenStack administrative user and a system with the Nova command line client (`python-novaclient`) installed.

1. Use the `source` command to load the administrative credentials from the `keystonerc_admin` file.

```
$ source ~/keystonerc_admin
```

2. Use the `nova service-list` command to identify the compute node to be removed.

```
$ nova service-list
+-----+-----+-----+-----+-----+
| Binary          | Host      | Zone      | Status  | State   |
+-----+-----+-----+-----+-----+
| nova-cert       | node0001  | internal  | enabled | up      |
| nova-compute    | node0001  | nova      | enabled | up      |
| nova-conductor  | node0001  | internal  | enabled | up      |
| nova-consoleauth | node0001  | internal  | enabled | up      |
| nova-network    | node0001  | internal  | enabled | up      |
| nova-scheduler  | node0001  | internal  | enabled | up      |
| ...             | ...       | ...       | ...     | ...     |
+-----+-----+-----+-----+-----+
```

3. Use the `nova service-disable` command to disable the `nova-compute` service on the node. This prevents new instances from being scheduled to run on the host.

```
$ nova service-disable HOST nova-compute
+-----+-----+-----+
| Host      | Binary      | Status    |
+-----+-----+-----+
| node0001  | nova-compute | disabled  |
+-----+-----+-----+
```

Replace *HOST* with the name of the node to disable as indicated in the output of the `nova service-list` command in the previous step.

- Use the `nova service-list` command to verify that the relevant instance of the `nova-compute` service is now disabled.

```
$ nova service-list
+-----+-----+-----+-----+-----+
| Binary      | Host      | Zone      | Status    | State |
+-----+-----+-----+-----+-----+
| nova-cert   | node0001  | internal  | enabled   | up    |
| nova-compute | node0001  | nova      | disabled | up    |
| nova-conductor | node0001  | internal  | enabled   | up    |
| nova-consoleauth | node0001  | internal  | enabled   | up    |
| nova-network | node0001  | internal  | enabled   | up    |
| nova-scheduler | node0001  | internal  | enabled   | up    |
| ...         | ...       | ...       | ...       | ...   |
+-----+-----+-----+-----+-----+
```

- Use the `nova migrate` command to migrate running instances to other compute nodes.

```
$ nova migrate HOST
```

Replace *HOST* with the name of the host being removed as indicated by the `nova service-list` command in the previous steps.

Once the migration of virtual machine instances has been completed the node can safely be shutdown or otherwise operated on without negatively impacting running virtual machine instances.

[Report a bug](#)

## 16.4. Using Config Drive

### 16.4.1. Config Drive Overview

OpenStack Compute can be configured to write metadata to a special configuration drive, config drive, that is attached to the instance when it boots. The instance can retrieve information from the config drive that would normally be available through the metadata service (for example instance ID, host name, or user data).

Config drive is typically used to pass networking configuration (for example, IP address, netmask, or gateway) when DHCP is not being used to assign IP addresses to instances. The instance's IP configuration can be transmitted using the config drive, which is mounted and accessed before the instance's network settings have been configured.

The config drive can be used by any guest operating system that is capable of mounting an ISO 9660 or VFAT file system (that is, all modern operating systems).

In addition, if an image:

- Has been built with the `cloud-init` package, it can automatically access metadata passed via config drive.
- Does not have the `cloud-init` package installed, it must be customized to run a script that mounts the config drive on boot, reads the data from the drive, and takes appropriate action such as adding the public key to an account.



## 16.4.2. Setting Up Config Drive

### 16.4.2.1. Enabling Config Drive

To enable the config drive, use one of the following:

- ▶ Use the `--config-drive=true` parameter when calling `nova boot`.

The following complex example enables the config drive as well as passing user data, two files, and two key/value metadata pairs, all of which are accessible from the config drive.

```
$ nova boot --config-drive=true --image my-image-name \
--key-name mykey --flavor 1 --user-data ./my-user-data.txt myinstance \
--file /etc/network/interfaces=/home/myuser/instance-interfaces \
--file known_hosts=/home/myuser/.ssh/known_hosts --meta role=webservers \
--meta essential=false
```

- ▶ Configure the Compute service to automatically create a config drive when booting by setting the following option in `/etc/nova/nova.conf`:

```
force_config_drive=true
```



#### Note

If a user uses the `--config-drive=true` flag with the `nova boot` command, an administrator cannot disable the config drive.



#### Warning

If using the default settings for config drive, the `genisoimage` program must be installed on each Compute host before attempting to use config drive (or the instance will not boot properly).

### 16.4.2.2. Config Drive Options

By default, the config drive is configured as an ISO 9660 filesystem. To explicitly specify the:

- ▶ ISO 9660 format, add the following line to `/etc/nova/nova.conf`:

```
config_drive_format=iso9660
```

- ▶ VFAT format, add the following line to `/etc/nova/nova.conf`:

```
config_drive_format=vfat
```



#### Note

For legacy reasons, the config drive can be configured to use VFAT format instead of ISO 9660. However, it is unlikely that you would require VFAT format, since ISO 9660 is widely supported across operating systems. If you use the VFAT format, the config drive will be 64 MBs.

The following table includes all `nova boot` options for config drive:

**Table 16.1. Description of configuration options for config drive**

Configuration option=Default value	(Type) Description
config_drive_cdrom=False	(BoolOpt) Whether Compute attaches the Config Drive image as a cdrom drive instead of a disk drive.
config_drive_format=iso9660	(StrOpt) Config drive format (valid options: iso9660 or vfat).
config_drive_inject_password=False	(BoolOpt) Sets the administrative password in the config drive image.
config_drive_skip_versions=1.0 2007-01-19 2007-03-01 2007-08-29 2007-10-10 2007-12-15 2008-02-01 2008-09-01	(StrOpt) List of metadata versions to skip placing into the config drive.
config_drive_tmpdir=None	(StrOpt) Where to put temporary files associated with config drive creation.
force_config_drive=False	(StrOpt) Whether Compute automatically creates config drive on startup.
mkisofs_cmd=genisoimage	(StrOpt) Name and optional path of the tool used for ISO image creation. Ensure that the specified tool is installed on each Compute host before attempting to use config drive (or the instance will not boot properly).

[Report a bug](#)

### 16.4.3. Accessing Config Drive

You must mount the drive depending on the guest operating system type. By default, the config drive has the config-2 volume label.

#### Procedure 16.1. Mount the Drive by Label

If the guest OS supports accessing disk by label, you should be able to mount the config drive as the /dev/disk/by-label/config-2 device. For example:

1. Create the directory to use for access:

```
# mkdir -p /mnt/config
```

2. Mount the device:

```
# mount /dev/disk/by-label/config-2 /mnt/config
```

#### Procedure 16.2. Mount the Drive using Disk Identification

If the guest OS does not use udev, then the /dev/disk/by-label directory will not be present.

1. Use the blkid command to identify the block device that corresponds to the config drive. For example, when booting the cirros image with the ml.tiny flavor, the device will be /dev/vdb:

```
# blkid -t LABEL="config-2" -odevice /dev/vdb
```

2. After you have identified the disk, the device can then be mounted:
  - a. Create the directory to use for access:

```
# mkdir -p /mnt/config
```

- b. Mount the device:

```
# mount /dev/vdb /mnt/config
```



## Note

- ▶ If using a Windows guest, the config drive is automatically displayed as the next available drive letter (for example, 'D:').
- ▶ When accessing the config drive, do not rely on the presence of the EC2 metadata (files under the `ec2` directory) to be present in the config drive (this content may be removed in a future release).
- ▶ When creating images that access config drive data, if there are multiple directories under the `openstack` directory, always select the highest API version by date that your consumer supports. For example, if your guest image can support versions 2012-03-05, 2012-08-05, 2013-04-13, it is best to try 2013-04-13 first and only revert to an earlier version if 2013-04-13 is absent.

### Procedure 16.3. View the Mounted Config Drive

1. Move to the newly mounted drive's files. For example:

```
$ cd /mnt/config
```

2. The files in the resulting config drive vary depending on the arguments that were passed to `nova boot`. Based on the example in *Enabling Config Drive*, the contents of the config drive would be:

```
$ ls
ec2/2013-04-13/meta-data.json ec2/2013-04-13/user-data ec2/latest/meta-
data.json ec2/latest/user-data openstack/2013-08-10/meta_data.json
openstack/2013-08-10/user_data openstack/content openstack/content/0000
openstack/content/0001 openstack/latest/meta_data.json
openstack/latest/user_data
```

## 16.4.4. Data Formats

[Report a bug](#)

### 16.4.4.1. OpenStack Metadata Format

The following listing provides an example of the `meta_data.json` file (`openstack/2012-08-10/meta_data.json`, `openstack/latest/meta_data.json`, these two files are identical), formatted to improve readability:

```

{
  "availability_zone": "nova",
  "files": [
    {
      "content_path": "/content/0000",
      "path": "/etc/network/interfaces"
    },
    {
      "content_path": "/content/0001",
      "path": "known_hosts"
    }
  ],
  "hostname": "test.novalocal",
  "launch_index": 0,
  "name": "test",
  "meta": {
    "role": "webservers"
    "essential": "false"
  },
  "public_keys": {
    "mykey": "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQgQBqUfVvCSez0/Wfpd8dLLgZXV9GtXQ7hnMN+Z00WQUyebVEHey
1CXuin0uY1cAJMhUq8j98SiW+cU0sU4J3x5l2+xi1bodDm1BtFWVeLIOQINpfV1n8fKjHB+ynPpe1F6tMD
vrFGU1Js44t30BrujMXBe8Rq44cCk6wqyjATA3rQ== Generated by Nova\n"
  },
  "uuid": "83679162-1378-4288-a2d4-70e13ec132aa"
}

```



## Note

If the option `--file /etc/network/interfaces=/home/myuser/instance-interfaces` is used with the `nova boot` command, the contents of this file are contained in the file `openstack/content/0000` file on the config drive, and the path is specified as `/etc/network/interfaces` in the `meta_data.json` file.

### 16.4.4.2. EC2 Metadata Format

[Report a bug](#)

Here is an example of the contents of `ec2/2009-04-04/meta-data.json`, `latest/meta-data.json`, formatted to improve readability (the two files are identical) :

```

{
  "ami-id": "ami-00000001",
  "ami-launch-index": 0,
  "ami-manifest-path": "FIXME",
  "block-device-mapping": {
    "ami": "sda1",
    "ephemeral0": "sda2",
    "root": "/dev/sda1",
    "swap": "sda3"
  },
  "hostname": "test.novalocal",
  "instance-action": "none",
  "instance-id": "i-00000001",
  "instance-type": "m1.tiny",
  "kernel-id": "aki-00000002",
  "local-hostname": "test.novalocal",
  "local-ipv4": null,
  "placement": {
    "availability-zone": "nova"
  },
  "public-hostname": "test.novalocal",
  "public-ipv4": "",
  "public-keys": {
    "0": {
      "openssh-key": "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQgQBqUfVvCSez0/Wfpd8dLLgZXV9GtXQ7hnMN+Z00WQUyebVEHey
1CXuin0uY1cAJMhUq8j98SiW+cU0sU4J3x5l2+xi1bodDm1BtFWVeLIOQINpfV1n8fKjHB+ynPpe1F6tMD
vrFGUlJs44t30BrujMXBe8Rq44cCk6wqyjATA3rQ== Generated by Nova\n"
    }
  },
  "ramdisk-id": "ari-00000003",
  "reservation-id": "r-7lfps8wj",
  "security-groups": [
    "default"
  ]
}

```

### 16.4.4.3. User Data Format

[Report a bug](#)

The following files will only be present if the `--user-data` flag was passed to `nova boot`, and will contain the contents of the user data file passed as the argument.

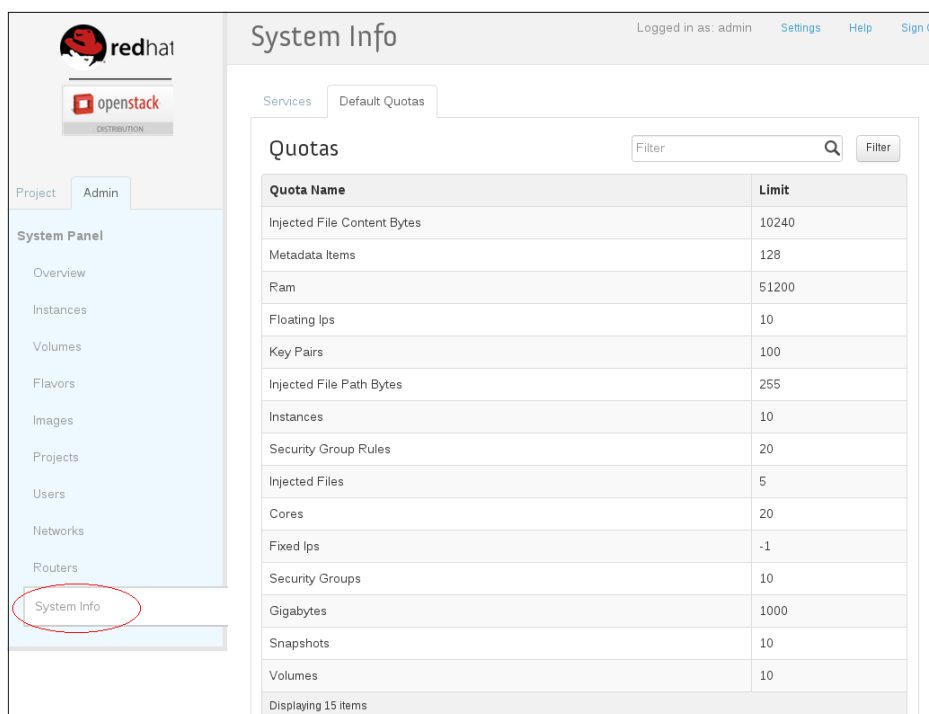
- ▶ `openstack/2013-08-10/user_data`
- ▶ `openstack/latest/user_data`
- ▶ `ec2/2013-04-13/user-data`
- ▶ `ec2/latest/user-data`

# Chapter 17. Managing Quotas

## 17.1. Viewing and Updating Quotas in the Dashboard

OpenStack provides a number of quotas (or operational limits), which are all enforced at the tenant level (rather than by user). For example, the number of gigabytes allowed per tenant can be controlled so that cloud resources are optimized. Typically, the default values are changed because a tenant requires more than 10 volumes, or more than 1TB on a node.

As an administrative user in the Dashboard, you can view (but not edit) the default quotas for a new project using the **Admin > System Info** option in the navigation sidebar, and clicking on the **Default Quotas** tab.



The screenshot shows the OpenStack Dashboard interface. On the left is a navigation sidebar with the 'System Info' option circled in red. The main content area is titled 'System Info' and has the 'Default Quotas' tab selected. Below the tab is a table of quotas with the following data:

Quota Name	Limit
Injected File Content Bytes	10240
Metadata Items	128
Ram	51200
Floating Ips	10
Key Pairs	100
Injected File Path Bytes	255
Instances	10
Security Group Rules	20
Injected Files	5
Cores	20
Fixed Ips	-1
Security Groups	10
Gigabytes	1000
Snapshots	10
Volumes	10

At the bottom of the table, it says 'Displaying 15 items'.

**Figure 17.1. Quota Defaults**

### Note

Not all possible project quotas are displayed in the dashboard. To obtain and update the complete list for a service, use its command-line tools. Refer to:

- ▶ *Updating Compute Quotas using the Command Line*
- ▶ *Updating Block Storage Quotas using the Command Line*

To update quotas for an individual tenant (project) in the dashboard:

1. Select the **Admin > Projects** option in the navigation sidebar.

- Click the project's **More** button, and then select **Modify Quotas**. The **Edit Project** window is displayed.
- Edit quota values on the **Quota** tab, and click the **Save** button. The table below provides parameter descriptions (listed in order of appearance).

**Table 17.1. Compute Quota Descriptions**

Quota	Description	Service
Metadata Items	Number of metadata items allowed per instance.	Compute
VCPUs	Number of instance cores allowed per tenant.	Compute
Instances	Number of instances allowed per tenant.	Compute
Injected Files	Number of injected files allowed per tenant.	Compute
Injected File Content Bytes	Number of content bytes allowed per injected file.	Compute
Volumes	Number of volumes allowed per tenant.	Block Storage
Gigabytes	Number of volume gigabytes allowed per tenant.	Block Storage
RAM (MB)	Megabytes of ram allowed per instance.	Compute
Floating IPs	Number of floating IP addresses allowed per tenant.	Compute
Fixed IPs	Number of fixed IP addresses allowed per tenant. This number should equal at least the number of allowed instances.	Compute
Security Groups	Number of security groups allowed per tenant.	Compute
Security Group Rules	Number of rules per security group.	Compute

[Report a bug](#)

## 17.2. Updating Compute Service Quotas on the Command Line

To update any of the Compute service quotas on the command line:

- Because Compute quotas are managed per tenant, use the following to obtain a tenant list:

```
# keystone tenant-list
+-----+-----+-----+
|          id          |  name  | enabled |
+-----+-----+-----+
| a981642d22c94e159a4a6540f70f9f8d | admin  |   True  |
| 934b662357674c7b9f5e4ec6ded4d0e7 | redhat01 |   True  |
| 7bc1dbfd7d284ec4a856ea1eb82dca80 | redhat02 |   True  |
| 9c554aaef7804ba49e1b21cbd97d218a | services |   True  |
+-----+-----+-----+
```

- To update a particular Compute quota for the tenant, use:

```
# nova-manage project quota tenantName --key quotaName --value quotaValue
```

For example:

```
# nova-manage project quota redhat01 --key floating_ips --value 20
metadata_items: 128
injected_file_content_bytes: 10240
ram: 51200
floating_ips: 20
security_group_rules: 20
instances: 10
key_pairs: 100
injected_files: 5
cores: 20
fixed_ips: unlimited
injected_file_path_bytes: 255
security_groups: 10
```

### 3. Restart the Compute service:

```
# service openstack-nova-api restart
```



## Note

To update the Compute quota defaults for a new project, update the parameters in the `/etc/nova/nova.conf` file.

**Table 17.2. Compute Quota Descriptions**

Quota	Description	Parameter
Injected File Content Bytes	Number of bytes allowed per injected file.	<code>injected_file_content_bytes</code>
Metadata Items	Number of metadata items allowed per instance	<code>metadata_items</code>
Ram	Megabytes of instance ram allowed per tenant.	<code>ram</code>
Floating Ips	Number of floating IP addresses allowed per tenant.	<code>floating_ips</code>
Key Pairs	Number of key pairs allowed per user.	<code>key_pairs</code>
Injected File Path Bytes	Number of bytes allowed per injected file path.	<code>injected_file_path_bytes</code>
Instances	Number of instances allowed per tenant.	<code>instances</code>
Security Group Rules	Number of rules per security group.	<code>security_group_rules</code>
Injected Files	Number of allowed injected files.	<code>injected_files</code>
Cores	Number of instance cores allowed per tenant.	<code>cores</code>
Fixed Ips	Number of fixed IP addresses allowed per tenant. This number should equal at least the number of allowed instances.	<code>fixed_ips</code>
Security Groups	Number of security groups per tenant.	<code>security_group_s</code>

[Report a bug](#)

## 17.3. Updating Block Storage Service Quotas on the Command Line

To update Block Storage service quotas:

1. Because Block Storage quotas are managed per tenant, use the following to obtain a tenant list:



```
# keystone tenant-list
+-----+-----+-----+
|          id          |  name  | enabled |
+-----+-----+-----+
| a981642d22c94e159a4a6540f70f9f8d | admin  |   True  |
| 934b662357674c7b9f5e4ec6ded4d0e7 | redhat01 |   True  |
| 7bc1dbfd7d284ec4a856ea1eb82dca80 | redhat02 |   True  |
| 9c554aaef7804ba49e1b21cbd97d218a | services |   True  |
+-----+-----+-----+
```

2. To view Block Storage quotas for a tenant, use:

```
# cinder quota-show tenantName
```

For example:

```
# cinder quota-show redhat01
+-----+-----+
| Property | Value |
+-----+-----+
| gigabytes | 1000 |
| snapshots | 10 |
| volumes   | 10 |
+-----+-----+
```

3. To update a particular quota value, use:

```
# cinder quota-update tenantName --quotaKey=NewValue
```

For example:

```
# cinder quota-update redhat01 --volumes=15
# cinder quota-show redhat01
+-----+-----+
| Property | Value |
+-----+-----+
| gigabytes | 1000 |
| snapshots | 10 |
| volumes   | 15 |
+-----+-----+
```



## Note

To update the Block Storage quota defaults for a new project, update the parameters in the `/etc/cinder/cinder.conf` file.

**Table 17.3. Block Storage Quota Descriptions**

Quota Parameter	Description
gigabytes	Number of volume gigabytes allowed per tenant.
snapshots	Number of Block Storage snapshots allowed per tenant.
volumes	Number of Block Storage volumes allowed per tenant.

# Installation Checklist

## A.1. Installation Prerequisites Checklists

The following tables describe prerequisites for successfully installing a Red Hat Enterprise Linux OpenStack Platform cloud. Checklist items are the minimum that should be known or verified before the installation is started.

The *Value/Verified* column can be used to provide the appropriate value or a 'check' that the item has been verified.



### Note

If installing single components after the initial Red Hat Enterprise Linux OpenStack Platform installation, ensure that you have:

- ▶ root access to the host machine (to install components, as well other administrative tasks such as updating the firewall).
- ▶ Administrative access to the Identity service.
- ▶ Administrative access to the database (ability to add both databases and users).

**Table A.1. OpenStack Installation-General**

Item	Description	Value/Verified
Hardware Requirements	Requirements in section 3.2 <i>Hardware Requirements</i> must be verified.	Yes   No
Operating System	Red Hat Enterprise Linux 6.4 Server	Yes   No
Red Hat Subscription	You must have a subscription to: <ul style="list-style-type: none"> <li>» Receive package updates from Red Hat Network or an equivalent source such as a Red Hat Network Satellite server.</li> <li>» Receive software updates for both Red Hat Enterprise Linux 6.4 Server and Red Hat Enterprise Linux OpenStack Platform</li> </ul>	Yes   No
Administrative access on all installation machines	Almost all procedures in this guide must be performed as the root user, so the installer must have root access.	Yes   No
Red Hat Subscriber Name/Password	You must know the Red Hat subscriber name and password.	<ul style="list-style-type: none"> <li>» Name:</li> <li>» Password:</li> </ul>
Machine addresses	You must know the host IP address of the machine or machines on which any OpenStack components and supporting software will be installed.	Provide host addresses for the following: <ul style="list-style-type: none"> <li>» Identity service</li> <li>» OpenStack Networking service</li> <li>» Block Storage service</li> <li>» Compute service</li> <li>» Image service</li> <li>» Object Storage service</li> <li>» Dashboard service</li> <li>» MySQL server (default database for this guide)</li> <li>» Qpid message broker</li> </ul>

**Table A.2. OpenStack Identity Service**

Item	Description	Value
Host Access	The system hosting the Identity service must have: <ul style="list-style-type: none"> <li>▶ Access to Red Hat Network or equivalent service.</li> <li>▶ Network interface addressable by all OpenStack hosts.</li> <li>▶ Network access to the database server.</li> <li>▶ If using LDAP, network access to the directory server .</li> </ul>	Verify whether the system has: <ul style="list-style-type: none"> <li>▶ Yes   No</li> <li>▶ Yes   No</li> <li>▶ Yes   No</li> <li>▶ Yes   No</li> </ul>
SSL Certificates	If using external SSL certificates, you must know where the database and certificates are located, and have access to them.	Yes   No
LDAP Information	If using LDAP, you must have administrative access to configure a new directory server schema.	Yes   No
Connections	The system hosting the Identity service must have a connection to all other OpenStack services.	Yes   No

**Table A.3. OpenStack Object Storage Service**

Item	Description	Value
File System	Red Hat currently supports the XFS and ext4 file systems for object storage; one of these must be available.	<ul style="list-style-type: none"> <li>▶ XFS</li> <li>▶ ext4</li> </ul>
Mount Point	The /srv/node mount point must be available.	Yes   No
Connections	For the cloud installed in this guide, the system hosting the Object Storage service will need a connection to the Identity service.	Yes   No

**Table A.4. OpenStack Image Service**

Item	Description	Value
Backend Storage	The Image service supports a number of storage backends. You must decide on one of the following: <ul style="list-style-type: none"> <li>▶ file (local directory)</li> <li>▶ OpenStack Object Storage service</li> </ul>	Storage:
Connections	The system hosting the Image service must have a connection to the Identity, Dashboard , and Compute services, as well as to the Object Storage service if using OpenStack Object Storage as its backend.	Yes   No

**Table A.5. OpenStack Block Storage Service**

Item	Description	Value
Backend Storage	The Block Storage service supports a number of storage backends. You must decide on one of the following: <ul style="list-style-type: none"> <li>‣ LVM</li> <li>‣ NFS</li> <li>‣ Red Hat Storage</li> </ul>	Storage:
Connections	The system hosting the Block Storage service must have a connection to the Identity, Dashboard, and Compute services.	Yes   No

**Table A.6. OpenStack Networking Service**

Item	Description	Value
Plugin agents	In addition to the standard OpenStack Networking components, a wide choice of plugin agents are also available that implement various networking mechanisms. You'll need to decide which of these apply to your network and install them.	Circle appropriate: <ul style="list-style-type: none"> <li>‣ Open vSwitch</li> <li>‣ Cisco UCS/Nexus</li> <li>‣ Linux Bridge</li> <li>‣ Nicira Network Virtualization Platform (NCP)</li> <li>‣ Ryu OpenFlow Controller</li> <li>‣ NEC OpenFlow</li> <li>‣ Big Switch Controller Plugin</li> <li>‣ Cloudbase Hyper-V</li> <li>‣ MidoNet</li> <li>‣ Brocade Quantum Plugin</li> <li>‣ PLUMgrid</li> </ul>
Connections	The system hosting the OpenStack Networking service must have a connection to the Identity, Dashboard, and Compute services.	Yes   No

**Table A.7. OpenStack Compute Service**

Item	Description	Value
Hardware virtualization support	The OpenStack Compute service requires hardware virtualization support. Note: a procedure is included in this Guide to verify this (refer to <i>Checking for Hardware Virtualization Support</i> ).	Yes   No
VNC client	The Compute service supports the Virtual Network Computing (VNC) console to instances through a web browser. You must decide whether this will be provided to your users.	Yes   No
Resources: CPU and Memory	OpenStack supports overcommitting of CPU and memory resources on Compute nodes (refer to <i>Configuring Resource Overcommitment</i> ). <ul style="list-style-type: none"> <li>▶ The default CPU overcommit ratio of 16 means that up to 16 virtual cores can be assigned to a node for each physical core.</li> <li>▶ The default memory overcommit ratio of 1.5 means that instances can be assigned to a physical node if the total instance memory usage is less than 1.5 times the amount of physical memory available.</li> </ul>	Decide: <ul style="list-style-type: none"> <li>▶ CPU setting:</li> <li>▶ Memory setting:</li> </ul>
Resources:Host	You can reserve resources for the host, to prevent a given amount of memory and disk resources from being automatically assigned to other resources on the host (refer to <i>Reserving Host Resources</i> ).	Decide: <ul style="list-style-type: none"> <li>▶ Host Disk (default 0MB):</li> <li>▶ Host Memory (default 512MB):</li> </ul>
libvirt Version	You will need to know the version of your libvirt for the configuration of Virtual Interface Plugging (refer to <i>Configuring Virtual Interface Plugging</i> ).	Version:
Connections	The system hosting the Compute service must have a connection to all other OpenStack services.	Yes   No

**Table A.8. OpenStack Dashboard Service**

Item	Description	Value
Host software	The system hosting the Dashboard service must have the following already installed: <ul style="list-style-type: none"> <li>▶ httpd</li> <li>▶ mod_wsgi</li> <li>▶ mod_ssl</li> </ul>	Yes   No
Connections	The system hosting the Dashboard service must have a connection to all other OpenStack services.	Yes   No

# Troubleshooting the OpenStack Environment

## B.1. No Networks or Routers Tab Appears in the Dashboard

The **Networks** and **Routers** tabs only appear in the dashboard when the environment is configured to use OpenStack Networking. In particular note that by default the PackStack utility currently deploys Nova Networking and as such in environments deployed in this manner the tab will not be visible.

If OpenStack Networking is deployed in the environment but the tabs still do not appear ensure that the service endpoints are defined correctly in the identify service, that the firewall is allowing access to the endpoints, and that the services are running.

[Report a bug](#)

## B.2. Dashboard Reports ERROR When Launching Instances

When using the dashboard to launch instances if the operation fails, a generic ERROR message is displayed. Determining the actual cause of the failure requires the use of the command line tools.

Use the `nova list` to locate the unique identifier of the instance. Then use this identifier as an argument to the `nova show` command. One of the items returned will be the error condition. The most common value is `NoValidHost`.

This error indicates that no valid host was found with enough available resources to host the instance. To work around this issue, consider choosing a smaller instance size or increasing the overcommit allowances for your environment.



### Note

To host a given instance, the compute node must have not only available CPU and RAM resources but also enough disk space for the ephemeral storage associated with the instance.

[Report a bug](#)

## B.3. Compute Instance Log Shows no Output

The log output from running instances is visible both from the **Log** tab on the dashboard and in the output of the `nova console-log` command. Sometimes however the log of a running instance will either appear to be completely empty or contain a single errant character, often a `?` character.

This occurs when the Compute service is trying to retrieve the log output of the guest via a serial console but the guest itself is not sending output to the console.

To avoid this Linux guests must be configured to append the parameters `console=tty0 console=ttyS0,115200n8` to the list of kernel arguments specified in the boot loader.

[Report a bug](#)

## B.4. Identity Client (keystone) Reports "Unable to communicate with identity service"

When the identity client (keystone) is unable to contact the identity service it returns an error:

```
$ keystone ACTION
Unable to communicate with identity service: [Errno 113] No route to host. (HTTP
400)
```

Replace *ACTION* with any valid identity service client action such as `user-list` or `service-list`. When the service is unreachable any identity client command that requires it will fail.

To debug the issue check for these common causes:

### Service Status

On the system hosting the identity service check the service status:

```
$ service openstack-keystone status
keystone (pid 2847) is running...
```

If the service is not running then log in as the root user and start it.

```
# service openstack-keystone start
```

### Firewall Rules

On the system hosting the identity service check that the firewall allows TCP traffic on ports 5000 and 35357. This command must be run while logged in as the root user.

```
# iptables --list -n | grep -P "(5000|35357)"
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 5000,35357
```

If no rule is found then add one to the `/etc/sysconfig/iptables` file:

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

Restart the `iptables` service for the change to take effect.

```
# service iptables restart
```

### Service Endpoints

On the system hosting the identity service check that the endpoints are defined correctly.

- ▶ Obtain the administration token:

```
$ grep admin_token /etc/keystone/keystone.conf
admin_token = 0292d404a88c4f269383ff28a3839ab4
```

- ▶ Determine the correct administration endpoint for the identity service:

```
http://IP:35357/VERSION
```

Replace *IP* with the IP address or host name of the system hosting the identity service. Replace *VERSION* with the API version (v2.0, or v3) that is in use.

- ▶ Unset any pre-defined identity service related environment variables:

```
$ unset OS_USERNAME OS_TENANT_NAME OS_PASSWORD OS_AUTH_URL
```

- ▶ Use the administration token and endpoint to authenticate with the identity service.



Confirm that the identity service endpoint is correct:

```
$ keystone --os-token=TOKEN \  
           --os-endpoint=ENDPOINT \  
           endpoint-list
```

Verify that the listed `publicurl`, `internalurl`, and `adminurl` for the identity service are correct. In particular ensure that the IP addresses and port numbers listed within each endpoint are correct and reachable over the network.

If these values are incorrect then refer to [Section 5.6, “Creating the Identity Service Endpoint”](#) for information on adding the correct endpoint. Once the correct endpoints have been added remove any incorrect endpoints using the `endpoint-delete` action of the `keystone` command.

```
$ keystone --os-token=TOKEN \  
           --os-endpoint=ENDPOINT \  
           endpoint-delete ID
```

Replace `TOKEN` and `ENDPOINT` with the values identified previously. Replace `ID` with the identity of the endpoint to remove as listed by the `endpoint-list` action.

# Service Log Files

## C.1. Block Storage Service Log Files

The log files of the block storage service are stored in the `/var/log/cinder/` directory of the host on which each service runs.

**Table C.1. Log Files**

File name	Description
<code>api.log</code>	The log of the API service ( <code>openstack-cinder-api</code> ).
<code>cinder-manage.log</code>	The log of the management interface ( <code>cinder-manage</code> ).
<code>scheduler.log</code>	The log of the scheduler service ( <code>openstack-cinder-scheduler</code> ).
<code>volume.log</code>	The log of the volume service ( <code>openstack-cinder-volume</code> ).

[Report a bug](#)

## C.2. Compute Service Log Files

The log files of the Compute service are stored in the `/var/log/nova/` directory of the host on which each service runs.

**Table C.2. Log Files**

File name	Description
<code>api.log</code>	The log of the API service ( <code>openstack-nova-api</code> ).
<code>cert.log</code>	The log of the X509 certificate service ( <code>openstack-nova-cert</code> ). This service is only required by the EC2 API to the Compute service.
<code>compute.log</code>	The log file of the Compute service itself ( <code>openstack-nova-compute</code> ).
<code>conductor.log</code>	The log file of the conductor ( <code>openstack-nova-conductor</code> ) that provides database query support to the Compute service.
<code>consoleauth.log</code>	The log file of the console authentication service ( <code>openstack-nova-consoleauth</code> ).
<code>network.log</code>	The log of the network service ( <code>openstack-nova-network</code> ). Note that this service only runs in environments that are <i>not</i> configured to use OpenStack networking.
<code>nova-manage.log</code>	The log of the management interface ( <code>nova-manage</code> ).
<code>scheduler.log</code>	The log of the scheduler service ( <code>openstack-nova-scheduler</code> ).

[Report a bug](#)

### C.3. Dashboard Log Files

The dashboard is served to users using the Apache web server (httpd). As a result, the log files for the dashboard are stored in the `/var/log/httpd` directory.

**Table C.3. Log Files**

File name	Description
<code>access_log</code>	The access log details all attempts to access the web server.
<code>error_log</code>	The error log details all unsuccessful attempts to access the web server and the reason for each failure.

[Report a bug](#)

### C.4. Identity Service Log Files

The log file of the identity service is stored in the `/var/log/keystone/` directory of the host on which it runs.

**Table C.4. Log File**

File name	Description
<code>keystone.log</code>	The log of the identity service (openstack-keystone).

[Report a bug](#)

### C.5. Image Service Log Files

The log files of the Image service are stored in the `/var/log/glance/` directory of the host on which each service runs.

**Table C.5. Log Files**

File name	Description
<code>api.log</code>	The log of the API service (openstack-glance-api).
<code>registry.log</code>	The log of the image registry service (openstack-glance-registry).

[Report a bug](#)

### C.6. Networking Service Log Files

The log files of the networking service are stored in the `/var/log/quantum/` directory of the host on which each service runs.

**Table C.6. Log Files**

File name	Description
dhcp-agent.log	The log for the DHCP agent (quantum-dhcp-agent).
l3-agent.log	The log for the L3 agent (quantum-l3-agent).
lbaas-agent.log	The log for the Load Balancer as a Service (LBaaS) agent (quantum-lbass-agent).
linuxbridge-agent.log	The log for the Linux Bridge agent (quantum-linuxbridge-agent).
metadata-agent.log	The log for the metadata agent (quantum-metadata-agent).
openvswitch-agent.log	The log for the Open vSwitch agent (quantum-openvswitch-agent).
server.log	The log for the OpenStack networking service itself (quantum-server).

[Report a bug](#)

## C.7. Object Storage Service Log Files

The log file of the object storage service is stored in the `/var/log/` directory of the host on which each service runs.

**Table C.7. Log File**

File name	Description
swift-startup.log	The log for the object storage proxy service.

[Report a bug](#)

# Example Configuration Files

## D.1. Dashboard Service Configuration Files

The Dashboard service stores its configuration settings in the `/etc/openstack-dashboard/local_settings` file. This file must be modified after installation.

```
import os

from django.utils.translation import ugettext_lazy as _

from openstack_dashboard import exceptions

DEBUG = True
TEMPLATE_DEBUG = DEBUG

# Required for Django 1.5.
# If horizon is running in production (DEBUG is False), set this
# with the list of host/domain names that the application can serve.
# For more information see:
# https://docs.djangoproject.com/en/dev/ref/settings/#allowed-hosts
#ALLOWED_HOSTS = ['horizon.example.com', ]

# Set SSL proxy settings:
# For Django 1.4+ pass this header from the proxy after terminating the SSL,
# and don't forget to strip it from the client's request.
# For more information see:
# https://docs.djangoproject.com/en/1.4/ref/settings/#secure-proxy-ssl-header
# SECURE_PROXY_SSL_HEADER = ('HTTP_X_FORWARDED_PROTOCOL', 'https')

# If Horizon is being served through SSL, then uncomment the following two
# settings to better secure the cookies from security exploits
#CSRF_COOKIE_SECURE = True
#SESSION_COOKIE_SECURE = True

# Overrides for OpenStack API versions. Use this setting to force the
# OpenStack dashboard to use a specific API version for a given service API.
# NOTE: The version should be formatted as it appears in the URL for the
# service API. For example, The identity service APIs have inconsistent
# use of the decimal point, so valid options would be "2.0" or "3".
# OPENSTACK_API_VERSIONS = {
#     "identity": 3
# }

# Set this to True if running on multi-domain model. When this is enabled, it
# will require user to enter the Domain name in addition to username for login.
# OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = False

# Overrides the default domain used when running on single-domain model
# with Keystone V3. All entities will be created in the default domain.
# OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = 'Default'

# Default OpenStack Dashboard configuration.
HORIZON_CONFIG = {
    'dashboards': ('project', 'admin', 'settings',),
    'default_dashboard': 'project',
    'user_home': 'openstack_dashboard.views.get_user_home',
    'ajax_queue_limit': 10,
    'auto_fade_alerts': {
        'delay': 3000,
        'fade_duration': 1500,
        'types': ['alert-success', 'alert-info']
    },
    'help_url': "http://docs.openstack.org",
    'exceptions': {'recoverable': exceptions.RECOVERABLE,
                  'not_found': exceptions.NOT_FOUND,
                  'unauthorized': exceptions.UNAUTHORIZED},
}

# Specify a regular expression to validate user passwords.
# HORIZON_CONFIG["password_validator"] = {
```

```

#     "regex": '.*',
#     "help_text": _("Your password does not meet the requirements.")
# }

# Disable simplified floating IP address management for deployments with
# multiple floating IP pools or complex network requirements.
# HORIZON_CONFIG["simple_ip_management"] = False

# Turn off browser autocompletion for the login form if so desired.
# HORIZON_CONFIG["password_autocomplete"] = "off"

LOCAL_PATH = os.path.dirname(os.path.abspath(__file__))

# Set custom secret key:
# You can either set it to a specific value or you can let horizon generate a
# default secret key that is unique on this machine, e.i. regardless of the
# amount of Python WSGI workers (if used behind Apache+mod_wsgi): However, there
# may be situations where you would want to set this explicitly, e.g. when
# multiple dashboard instances are distributed on different machines (usually
# behind a load-balancer). Either you have to make sure that a session gets all
# requests routed to the same dashboard instance or you set the same SECRET_KEY
# for all of them.
from horizon.utils import secret_key
SECRET_KEY = secret_key.generate_or_read_from_file(os.path.join(LOCAL_PATH,
'.secret_key_store'))

# We recommend you use memcached for development; otherwise after every reload
# of the django development server, you will have to login again. To use
# memcached set CACHES to something like
# CACHES = {
#     'default': {
#         'BACKEND' : 'django.core.cache.backends.memcached.MemcachedCache',
#         'LOCATION' : '127.0.0.1:11211',
#     }
# }

CACHES = {
    'default': {
        'BACKEND' : 'django.core.cache.backends.locmem.LocMemCache'
    }
}

# Send email to the console by default
EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'
# Or send them to /dev/null
#EMAIL_BACKEND = 'django.core.mail.backends.dummy.EmailBackend'

# Configure these for your outgoing email host
# EMAIL_HOST = 'smtp.my-company.com'
# EMAIL_PORT = 25
# EMAIL_HOST_USER = 'djangomail'
# EMAIL_HOST_PASSWORD = 'top-secret!'

# For multiple regions uncomment this configuration, and add (endpoint, title).
# AVAILABLE_REGIONS = [
#     ('http://cluster1.example.com:5000/v2.0', 'cluster1'),
#     ('http://cluster2.example.com:5000/v2.0', 'cluster2'),
# ]

OPENSTACK_HOST = "127.0.0.1"
OPENSTACK_KEYSTONE_URL = "http://%s:5000/v2.0" % OPENSTACK_HOST
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "Member"

# Disable SSL certificate checks (useful for self-signed certificates):

```

```

# OPENSTACK_SSL_NO_VERIFY = True

# The OPENSTACK_KEYSTONE_BACKEND settings can be used to identify the
# capabilities of the auth backend for Keystone.
# If Keystone has been configured to use LDAP as the auth backend then set
# can_edit_user to False and name to 'ldap'.
#
# TODO(tres): Remove these once Keystone has an API to identify auth backend.
OPENSTACK_KEYSTONE_BACKEND = {
    'name': 'native',
    'can_edit_user': True,
    'can_edit_project': True
}

OPENSTACK_HYPERVISOR_FEATURES = {
    'can_set_mount_point': True,

    # NOTE: as of Grizzly this is not yet supported in Nova so enabling this
    # setting will not do anything useful
    'can_encrypt_volumes': False
}

# The OPENSTACK_QUANTUM_NETWORK settings can be used to enable optional
# services provided by quantum. Currently only the load balancer service
# is available.
OPENSTACK_QUANTUM_NETWORK = {
    'enable_lb': False
}

# OPENSTACK_ENDPOINT_TYPE specifies the endpoint type to use for the endpoints
# in the Keystone service catalog. Use this setting when Horizon is running
# external to the OpenStack environment. The default is 'internalURL'.
#OPENSTACK_ENDPOINT_TYPE = "publicURL"

# The number of objects (Swift containers/objects or images) to display
# on a single page before providing a paging element (a "more" link)
# to paginate results.
API_RESULT_LIMIT = 1000
API_RESULT_PAGE_SIZE = 20

# The timezone of the server. This should correspond with the timezone
# of your entire OpenStack installation, and hopefully be in UTC.
TIME_ZONE = "UTC"

LOGGING = {
    'version': 1,
    # When set to True this will disable all logging except
    # for loggers specified in this configuration dictionary. Note that
    # if nothing is specified here and disable_existing_loggers is True,
    # django.db.backends will still log unless it is disabled explicitly.
    'disable_existing_loggers': False,
    'handlers': {
        'null': {
            'level': 'DEBUG',
            'class': 'django.utils.log.NullHandler',
        },
        'console': {
            # Set the level to "DEBUG" for verbose output logging.
            'level': 'INFO',
            'class': 'logging.StreamHandler',
        },
    },
    'loggers': {
        # Logging from django.db.backends is VERY verbose, send to null

```



```
# by default.
'django.db.backends': {
    'handlers': ['null'],
    'propagate': False,
},
'requests': {
    'handlers': ['null'],
    'propagate': False,
},
'horizon': {
    'handlers': ['console'],
    'propagate': False,
},
'openstack_dashboard': {
    'handlers': ['console'],
    'propagate': False,
},
'novaclient': {
    'handlers': ['console'],
    'propagate': False,
},
'keystoneclient': {
    'handlers': ['console'],
    'propagate': False,
},
'glanceclient': {
    'handlers': ['console'],
    'propagate': False,
},
'nose.plugins.manager': {
    'handlers': ['console'],
    'propagate': False,
}
}
```

[Report a bug](#)

## D.2. Block Storage Service Configuration Files

### D.2.1. api-paste.ini

The block storage API service stores its configuration settings in the `/etc/cinder/api-paste.ini` file.

```
#####
# OpenStack #
#####

[composite:osapi_volume]
use = call:cinder.api:root_app_factory
/: apiversions
/v1: openstack_volume_api_v1
/v2: openstack_volume_api_v2

[composite:openstack_volume_api_v1]
use = call:cinder.api.middleware.auth:pipeline_factory
noauth = faultwrap sizelimit noauth apiv1
keystone = faultwrap sizelimit authtoken keystonecontext apiv1
keystone_nolimit = faultwrap sizelimit authtoken keystonecontext apiv1

[composite:openstack_volume_api_v2]
use = call:cinder.api.middleware.auth:pipeline_factory
noauth = faultwrap sizelimit noauth apiv2
keystone = faultwrap sizelimit authtoken keystonecontext apiv2
keystone_nolimit = faultwrap sizelimit authtoken keystonecontext apiv2

[filter:faultwrap]
paste.filter_factory = cinder.api.middleware.fault:FaultWrapper.factory

[filter:noauth]
paste.filter_factory = cinder.api.middleware.auth:NoAuthMiddleware.factory

[filter:sizelimit]
paste.filter_factory =
cinder.api.middleware.sizelimit:RequestBodySizeLimiter.factory

[app:apiv1]
paste.app_factory = cinder.api.v1.router:APIRouter.factory

[app:apiv2]
paste.app_factory = cinder.api.v2.router:APIRouter.factory

[pipeline:apiversions]
pipeline = faultwrap osvolumeverSIONapp

[app:osvolumeverSIONapp]
paste.app_factory = cinder.api.versions:Versions.factory

#####
# Shared #
#####

[filter:keystonecontext]
paste.filter_factory = cinder.api.middleware.auth:CinderKeystoneContext.factory

[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
service_protocol = http
service_host = 127.0.0.1
service_port = 5000
signing_dir = /var/lib/cinder
```

[Report a bug](#)

## D.2.2. cinder.conf

The majority of block storage service configuration is performed from the `/etc/cinder/cinder.conf` file.

```
[DEFAULT]
logdir = /var/log/cinder
state_path = /var/lib/cinder
lock_path = /var/lib/cinder/tmp
volumes_dir = /etc/cinder/volumes
iscsi_helper = tgtadm
sql_connection = mysql://cinder:f1ec062ca512429d@10.38.15.166/cinder
rpc_backend = cinder.openstack.common.rpc.impl_qpid
rootwrap_config = /etc/cinder/rootwrap.conf
qpid_hostname=10.38.15.166
api_paste_config=/etc/cinder/api-paste.ini
rabbit_port=5672
rabbit_password=
iscsi_ip_address=10.38.15.166
volume_group=cinder-volumes
rabbit_userid=nova
verbose=False
rabbit_virtual_host=/
glance_host=10.38.15.166
rabbit_host=127.0.0.1
auth_strategy=keystone

[keystone_authtoken]
admin_tenant_name = services
admin_user = cinder
admin_password = 5a9d3e08713f4b2c
auth_host = 10.38.15.166
auth_port = 35357
auth_protocol = http
signing_dirname = /tmp/keystone-signing-cinder
service_host=10.38.15.166
service_protocol=http
service_port=5000
```

[Report a bug](#)

### D.2.3. policy.json



#### Error

Topic 16849 failed validation and is not included in this build.

### D.2.4. rootwrap.conf

The `/etc/nova/rootwrap.conf` file defines configuration values used by the `rootwrap` script that is used by the Compute service when it needs to escalate its privileges to those of the root user.

```
# Configuration for cinder-rootwrap
# This file should be owned by (and only-writeable by) the root user

[DEFAULT]
# List of directories to load filter definitions from (separated by ',').
# These directories MUST all be only writeable by root !
filters_path=/etc/cinder/rootwrap.d,/usr/share/cinder/rootwrap

# List of directories to search executables in, in case filters do not
# explicitly specify a full path (separated by ',')
# If not specified, defaults to system PATH environment variable.
# These directories MUST all be only writeable by root !
exec_dirs=/sbin,/usr/sbin,/bin,/usr/bin

# Enable logging to syslog
# Default value is False
use_syslog=False

# Which syslog facility to use.
# Valid values include auth, authpriv, syslog, user0, user1...
# Default value is 'syslog'
syslog_log_facility=syslog

# Which messages to log.
# INFO means log all usage
# ERROR means only log unsuccessful attempts
syslog_log_level=ERROR
```

[Report a bug](#)

## D.3. Compute Service Configuration Files

### D.3.1. api-paste.ini

The compute API service stores its configuration settings in the `/etc/nova/api-paste.ini` file.

```
#####
# Metadata #
#####
[composite:metadata]
use = egg:Paste#urlmap
/: meta

[pipeline:meta]
pipeline = ec2faultwrap logrequest metaapp

[app:metaapp]
paste.app_factory = nova.api.metadata.handler:MetadataRequestHandler.factory

#####
# EC2 #
#####

[composite:ec2]
use = egg:Paste#urlmap
/services/Cloud: ec2cloud

[composite:ec2cloud]
use = call:nova.api.auth:pipeline_factory
noauth = ec2faultwrap logrequest ec2noauth cloudrequest validator ec2executor
keystone = ec2faultwrap logrequest ec2keystoneauth cloudrequest validator
ec2executor

[filter:ec2faultwrap]
paste.filter_factory = nova.api.ec2:FaultWrapper.factory

[filter:logrequest]
paste.filter_factory = nova.api.ec2:RequestLogging.factory

[filter:ec2lockout]
paste.filter_factory = nova.api.ec2:Lockout.factory

[filter:ec2keystoneauth]
paste.filter_factory = nova.api.ec2:EC2KeystoneAuth.factory

[filter:ec2noauth]
paste.filter_factory = nova.api.ec2:NoAuth.factory

[filter:cloudrequest]
controller = nova.api.ec2.cloud.CloudController
paste.filter_factory = nova.api.ec2:Requestify.factory

[filter:authorizer]
paste.filter_factory = nova.api.ec2:Authorizer.factory

[filter:validator]
paste.filter_factory = nova.api.ec2:Validator.factory

[app:ec2executor]
paste.app_factory = nova.api.ec2:Executor.factory

#####
# Openstack #
#####

[composite:osapi_compute]
use = call:nova.api.openstack.urlmap:urlmap_factory
/: oscomputeversions
/v1.1: openstack_compute_api_v2
/v2: openstack_compute_api_v2
```

```

[composite:openstack_compute_api_v2]
use = call:nova.api.auth:pipeline_factory
noauth = faultwrap sizelimit noauth ratelimit osapi_compute_app_v2
keystone = faultwrap sizelimit authtoken keystonecontext ratelimit
osapi_compute_app_v2
keystone_nolimit = faultwrap sizelimit authtoken keystonecontext
osapi_compute_app_v2

[filter:faultwrap]
paste.filter_factory = nova.api.openstack:FaultWrapper.factory

[filter:noauth]
paste.filter_factory = nova.api.openstack.auth:NoAuthMiddleware.factory

[filter:ratelimit]
paste.filter_factory =
nova.api.openstack.compute.limits:RateLimitingMiddleware.factory

[filter:sizelimit]
paste.filter_factory = nova.api.sizelimit:RequestBodySizeLimiter.factory

[app:osapi_compute_app_v2]
paste.app_factory = nova.api.openstack.compute:APIRouter.factory

[pipeline:oscomputeversions]
pipeline = faultwrap oscomputeversionapp

[app:oscomputeversionapp]
paste.app_factory = nova.api.openstack.compute.versions:Versions.factory

#####
# Shared #
#####

[filter:keystonecontext]
paste.filter_factory = nova.api.auth:NovaKeystoneContext.factory

[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
# signing_dir is configurable, but the default behavior of the authtoken
# middleware should be sufficient. It will create a temporary directory
# in the home directory for the user the nova process is running as.
#signing_dir = /var/lib/nova/keystone-signing
# Workaround for https://bugs.launchpad.net/nova/+bug/1154809
auth_version = v2.0

```

[Report a bug](#)

### D.3.2. nova.conf

The majority of Compute service configuration is performed from the `/etc/nova/nova.conf` file. Note that in this example configuration file Nova networking is in use.

```
[DEFAULT]
logdir = /var/log/nova
state_path = /var/lib/nova
lock_path = /var/lib/nova/tmp
volumes_dir = /etc/nova/volumes
dhcpbridge = /usr/bin/nova-dhcpbridge
dhcpbridge_flagfile = /etc/nova/nova.conf
force_dhcp_release = false
injected_network_template = /usr/share/nova/interfaces.template
libvirt_nonblocking = True
libvirt_inject_partition = -1
network_manager = nova.network.manager.FlatDHCPManager
iscsi_helper = tgtadm
sql_connection = mysql://nova:463f1c1aeea04c98@10.38.15.166/nova
compute_driver = libvirt.LibvirtDriver
firewall_driver = nova.virt.libvirt.firewall.IptablesFirewallDriver
rpc_backend = nova.openstack.common.rpc.impl_qpid
rootwrap_config = /etc/nova/rootwrap.conf
glance_api_servers=10.38.15.166:9292
osapi_compute_listen=0.0.0.0
image_service=nova.image.glance.GlanceImageService
api_paste_config=/etc/nova/api-paste.ini
metadata_listen=0.0.0.0
ec2_listen=0.0.0.0
qpid_hostname=10.38.15.166
service_down_time=60
auth_strategy=keystone
volume_api_class=nova.volume.cinder.API
enabled_apis=ec2,osapi_compute,metadata
rabbit_host=localhost
metadata_host=10.38.15.166
osapi_volume_listen=0.0.0.0
verbose=false
novncproxy_port=6080
flat_interface=lo
auto_assign_floating_ip=False
floating_range=10.3.4.0/22
scheduler_default_filters=RetryFilter,AvailabilityZoneFilter,RamFilter,ComputeFilter,ComputeCapabilitiesFilter,ImagePropertiesFilter,CoreFilter
vncserver_listen=10.38.15.166
novncproxy_base_url=http://10.38.15.166:6080/vnc_auto.html
network_host=10.38.15.166
fixed_range=192.168.32.0/22
flat_network_bridge=br100
cpu_allocation_ratio=16.0
ram_allocation_ratio=1.5
public_interface=eth1
connection_type=libvirt
vnc_enabled=true
novncproxy_host=0.0.0.0
vncserver_proxyclient_address=10.38.15.166
dhcp_domain=novalocal
flat_injected=false
libvirt_type=qemu
libvirt_cpu_mode=none
default_floating_pool=nova

[keystone_authtoken]
admin_tenant_name = services
admin_user = nova
admin_password = ea2f8908e4d74d87
auth_host = 10.38.15.166
auth_port = 35357
auth_protocol = http
```

```
signing_dir = /tmp/keystone-signing-nova
```

[Report a bug](#)

### D.3.3. policy.json



#### Error

Topic 16847 failed validation and is not included in this build.

### D.3.4. rootwrap.conf

The `/etc/nova/rootwrap.conf` file defines configuration values used by the `rootwrap` script that is used by the Compute service when it needs to escalate its privileges to those of the root user.

```
# Configuration for nova-rootwrap
# This file should be owned by (and only-writeable by) the root user

[DEFAULT]
# List of directories to load filter definitions from (separated by ',').
# These directories MUST all be only writeable by root !
filters_path=/etc/nova/rootwrap.d,/usr/share/nova/rootwrap

# List of directories to search executables in, in case filters do not
# explicitly specify a full path (separated by ',')
# If not specified, defaults to system PATH environment variable.
# These directories MUST all be only writeable by root !
exec_dirs=/sbin,/usr/sbin,/bin,/usr/bin

# Enable logging to syslog
# Default value is False
use_syslog=False

# Which syslog facility to use.
# Valid values include auth, authpriv, syslog, user0, user1...
# Default value is 'syslog'
syslog_log_facility=syslog

# Which messages to log.
# INFO means log all usage
# ERROR means only log unsuccessful attempts
syslog_log_level=ERROR
```

[Report a bug](#)

## D.4. Identity Service Configuration Files

### D.4.1. keystone.conf

The central configuration file for the identity service is found in:  
`/etc/keystone/keystone.conf`.



```
[DEFAULT]
log_file = /var/log/keystone/keystone.log
admin_token = e6bdb221b674074ff76e
# A "shared secret" between keystone and other openstack services
# admin_token = ADMIN

# The IP address of the network interface to listen on
# bind_host = 0.0.0.0

# The port number which the public service listens on
# public_port = 5000

# The port number which the public admin listens on
# admin_port = 35357

# The base endpoint URLs for keystone that are advertised to clients
# (NOTE: this does NOT affect how keystone listens for connections)
# public_endpoint = http://localhost:%(public_port)d/
# admin_endpoint = http://localhost:%(admin_port)d/

# The port number which the OpenStack Compute service listens on
# compute_port = 8774

# Path to your policy definition containing identity actions
# policy_file = policy.json

# Rule to check if no matching policy definition is found
# FIXME(dolph): This should really be defined as [policy] default_rule
# policy_default_rule = admin_required

# Role for migrating membership relationships
# During a SQL upgrade, the following values will be used to create a new role
# that will replace records in the user_tenant_membership table with explicit
# role grants. After migration, the member_role_id will be used in the API
# add_user_to_project, and member_role_name will be ignored.
# member_role_id = 9fe2ff9ee4384b1894a90878d3e92bab
# member_role_name = _member_

# === Logging Options ===
# Print debugging output
# (includes plaintext request logging, potentially including passwords)
debug = True

# Print more verbose output
verbose = True

# Name of log file to output to. If not set, logging will go to stdout.
# log_file = keystone.log

# The directory to keep log files in (will be prepended to --logfile)
# log_dir = /var/log/keystone
# Use syslog for logging.
# use_syslog = False

# syslog facility to receive log lines
# syslog_log_facility = LOG_USER

# If this option is specified, the logging configuration file specified is
# used and overrides any other logging options specified. Please see the
# Python logging module documentation for details on logging configuration
# files.
# log_config = logging.conf

# A logging.Formatter log message format string which may use any of the
```

```
# available logging.LogRecord attributes.
# log_format = %(asctime)s %(levelname)8s [% (name)s] %(message)s

# Format string for %(asctime)s in log records.
# log_date_format = %Y-%m-%d %H:%M:%S

# onready allows you to send a notification when the process is ready to serve
# For example, to have it notify using systemd, one could set shell command:
# onready = systemd-notify --ready
# or a module with notify() method:
# onready = keystone.common.systemd

[sql]
connection = mysql://keystone:Redhat123@sgordon-cinder-
api.usersys.redhat.com/keystone
# The SQLAlchemy connection string used to connect to the database
# connection = sqlite:///keystone.db

# the timeout before idle sql connections are reaped
# idle_timeout = 200

[identity]
driver = keystone.identity.backends.sql.Identity
# driver = keystone.identity.backends.sql.Identity

# This references the domain to use for all Identity API v2 requests (which are
# not aware of domains). A domain with this ID will be created for you by
# keystone-manage db_sync in migration 008. The domain referenced by this ID
# cannot be deleted on the v3 API, to prevent accidentally breaking the v2 API.
# There is nothing special about this domain, other than the fact that it must
# exist to order to maintain support for your v2 clients.
# default_domain_id = default

[trust]
# driver = keystone.trust.backends.sql.Trust

# delegation and impersonation features can be optionally disabled
# enabled = True

[catalog]
template_file = /etc/keystone/default_catalog.templates
driver = keystone.catalog.backends.sql.Catalog
# dynamic, sql-based backend (supports API/CLI-based management commands)
# driver = keystone.catalog.backends.sql.Catalog

# static, file-based backend (does *NOT* support any management commands)
# driver = keystone.catalog.backends.templated.TemplatedCatalog

# template_file = default_catalog.templates

[token]
driver = keystone.token.backends.sql.Token
# driver = keystone.token.backends.kvs.Token

# Amount of time a token should remain valid (in seconds)
# expiration = 86400

[policy]
# driver = keystone.policy.backends.sql.Policy

[ec2]
driver = keystone.contrib.ec2.backends.sql.Ec2
# driver = keystone.contrib.ec2.backends.kvs.Ec2
```

```
[ssl]
#enable = True
#certfile = /etc/keystone/ssl/certs/keystone.pem
#keyfile = /etc/keystone/ssl/private/keystonekey.pem
#ca_certs = /etc/keystone/ssl/certs/ca.pem
#cert_required = True

[signing]
#token_format = PKI
#certfile = /etc/keystone/ssl/certs/signing_cert.pem
#keyfile = /etc/keystone/ssl/private/signing_key.pem
#ca_certs = /etc/keystone/ssl/certs/ca.pem
#key_size = 1024
#valid_days = 3650
#ca_password = None

[ldap]
# url = ldap://localhost
# user = dc=Manager,dc=example,dc=com
# password = None
# suffix = cn=example,cn=com
# use_dumb_member = False
# allow_subtree_delete = False
# dumb_member = cn=dumb,dc=example,dc=com

# Maximum results per page; a value of zero ('0') disables paging (default)
# page_size = 0

# The LDAP dereferencing option for queries. This can be either 'never',
# 'searching', 'always', 'finding' or 'default'. The 'default' option falls
# back to using default dereferencing configured by your ldap.conf.
# alias_dereferencing = default

# The LDAP scope for queries, this can be either 'one'
# (onelevel/singlelevel) or 'sub' (subtree/wholeSubtree)
# query_scope = one

# user_tree_dn = ou=Users,dc=example,dc=com
# user_filter =
# user_objectclass = inetOrgPerson
# user_domain_id_attribute = businessCategory
# user_id_attribute = cn
# user_name_attribute = sn
# user_mail_attribute = email
# user_pass_attribute = userPassword
# user_enabled_attribute = enabled
# user_enabled_mask = 0
# user_enabled_default = True
# user_attribute_ignore = tenant_id,tenants
# user_allow_create = True
# user_allow_update = True
# user_allow_delete = True
# user_enabled_emulation = False
# user_enabled_emulation_dn =

# tenant_tree_dn = ou=Groups,dc=example,dc=com
# tenant_filter =
# tenant_objectclass = groupOfNames
# tenant_domain_id_attribute = businessCategory
# tenant_id_attribute = cn
# tenant_member_attribute = member
# tenant_name_attribute = ou
# tenant_desc_attribute = desc
# tenant_enabled_attribute = enabled
```

```
# tenant_attribute_ignore =
# tenant_allow_create = True
# tenant_allow_update = True

# tenant_allow_delete = True
# tenant_enabled_emulation = False
# tenant_enabled_emulation_dn =

# role_tree_dn = ou=Roles,dc=example,dc=com
# role_filter =
# role_objectclass = organizationalRole
# role_id_attribute = cn
# role_name_attribute = ou
# role_member_attribute = roleOccupant
# role_attribute_ignore =
# role_allow_create = True
# role_allow_update = True
# role_allow_delete = True

# group_tree_dn =
# group_filter =
# group_objectclass = groupOfNames
# group_id_attribute = cn
# group_name_attribute = ou
# group_member_attribute = member
# group_desc_attribute = desc
# group_attribute_ignore =
# group_allow_create = True
# group_allow_update = True
# group_allow_delete = True

[auth]
methods = password,token
password = keystone.auth.plugins.password.Password
token = keystone.auth.plugins.token.Token

[filter:debug]
paste.filter_factory = keystone.common.wsgi:Debug.factory

[filter:token_auth]
paste.filter_factory = keystone.middleware:TokenAuthMiddleware.factory

[filter:admin_token_auth]
paste.filter_factory = keystone.middleware:AdminTokenAuthMiddleware.factory

[filter:xml_body]
paste.filter_factory = keystone.middleware:XmlBodyMiddleware.factory

[filter:json_body]
paste.filter_factory = keystone.middleware:JsonBodyMiddleware.factory

[filter:user_crud_extension]
paste.filter_factory = keystone.contrib.user_crud:CrudExtension.factory

[filter:crud_extension]
paste.filter_factory = keystone.contrib.admin_crud:CrudExtension.factory

[filter:ec2_extension]
paste.filter_factory = keystone.contrib.ec2:Ec2Extension.factory

[filter:s3_extension]
paste.filter_factory = keystone.contrib.s3:S3Extension.factory

[filter:url_normalize]
```

```
paste.filter_factory = keystone.middleware:NormalizingFilter.factory

[filter:sizelimit]
paste.filter_factory = keystone.middleware:RequestBodySizeLimiter.factory

[filter:stats_monitoring]
paste.filter_factory = keystone.contrib.stats:StatsMiddleware.factory

[filter:stats_reporting]
paste.filter_factory = keystone.contrib.stats:StatsExtension.factory

[filter:access_log]
paste.filter_factory = keystone.contrib.access:AccessLogMiddleware.factory

[app:public_service]
paste.app_factory = keystone.service:public_app_factory

[app:service_v3]
paste.app_factory = keystone.service:v3_app_factory

[app:admin_service]
paste.app_factory = keystone.service:admin_app_factory

[pipeline:public_api]
pipeline = access_log sizelimit stats_monitoring url_normalize token_auth
admin_token_auth xml_body json_body debug ec2_extension user_crud_extension
public_service

[pipeline:admin_api]
pipeline = access_log sizelimit stats_monitoring url_normalize token_auth
admin_token_auth xml_body json_body debug stats_reporting ec2_extension
s3_extension crud_extension admin_service

[pipeline:api_v3]
pipeline = access_log sizelimit stats_monitoring url_normalize token_auth
admin_token_auth xml_body json_body debug stats_reporting ec2_extension
s3_extension service_v3

[app:public_version_service]
paste.app_factory = keystone.service:public_version_app_factory

[app:admin_version_service]
paste.app_factory = keystone.service:admin_version_app_factory

[pipeline:public_version_api]
pipeline = access_log sizelimit stats_monitoring url_normalize xml_body
public_version_service

[pipeline:admin_version_api]
pipeline = access_log sizelimit stats_monitoring url_normalize xml_body
admin_version_service

[composite:main]
use = egg:Paste#urlmap
/v2.0 = public_api
/v3 = api_v3
/ = public_version_api

[composite:admin]
use = egg:Paste#urlmap
/v2.0 = admin_api
/v3 = api_v3
/ = admin_version_api
```

## D.5. Image Service Configuration Files

### D.5.1. glance-registry.conf

The Image service's registry, which stores the metadata about images, is found in `/etc/glance/glance-registry.conf`. This file must be modified after installation.

```
[DEFAULT]
# Show more verbose log output (sets INFO log level output)
verbose = True

# Show debugging output in logs (sets DEBUG log level output)
debug = False

# Address to bind the registry server
bind_host = 0.0.0.0

# Port the bind the registry server to
bind_port = 9191

# Log to this file. Make sure you do not set the same log
# file for both the API and registry servers!
log_file = /var/log/glance/registry.log

# Backlog requests when creating socket
backlog = 4096

# TCP_KEEPIDLE value in seconds when creating socket.
# Not supported on OS X.
#tcp_keepidle = 600

# SQLAlchemy connection string for the reference implementation
# registry server. Any valid SQLAlchemy connection string is fine.
# See: http://www.sqlalchemy.org/docs/05/reference/sqlalchemy/connections.html#sqlalchemy.create\_engine
sql_connection = mysql://glance:YOUR_GLANCEDB_PASSWORD@192.168.206.130/glance

# Period in seconds after which SQLAlchemy should reestablish its connection
# to the database.
#
# MySQL uses a default `wait_timeout` of 8 hours, after which it will drop
# idle connections. This can result in 'MySQL Gone Away' exceptions. If you
# notice this, you can lower this value to ensure that SQLAlchemy reconnects
# before MySQL can drop the connection.
sql_idle_timeout = 3600

# Limit the api to return `param_limit_max` items in a call to a container. If
# a larger `limit` query param is provided, it will be reduced to this value.
api_limit_max = 1000

# If a `limit` query param is not provided in an api request, it will
# default to `limit_param_default`
limit_param_default = 25

# Role used to identify an authenticated user as administrator
#admin_role = admin

# ===== Syslog Options =====
# Send logs to syslog (/dev/log) instead of to file specified
# by `log_file`
use_syslog = False
# Facility to use. If unset defaults to LOG_USER.
#syslog_log_facility = LOG_LOCAL1

# ===== SSL Options =====
# Certificate file to use when starting registry server securely
#cert_file = /path/to/certfile

# Private key file to use when starting registry server securely
#key_file = /path/to/keyfile
```

```
# CA certificate file to use to verify connecting clients
#ca_file = /path/to/cafile

[keystone_authtoken]
auth_host = 127.0.0.1
auth_port = 35357
auth_protocol = http
admin_tenant_name = services
admin_user = glance
admin_password = secret

[paste_deploy]
# Name of the paste configuration file that defines the available pipelines
config_file = /etc/glance/glance-registry-paste.ini

# Partial name of a pipeline in your paste configuration file with the
# service name removed. For example, if your paste section name is
# [pipeline:glance-api-keystone], you would configure the flavor below
# as 'keystone'.
flavor=keystone
```

## D.5.2. glance-registry-paste.ini

[Report a bug](#)

The Image service's API middleware pipeline is found in: `/etc/glance/glance-registry-paste.ini`

This file must be modified after installation.

```
# Use this pipeline for no auth - DEFAULT
# [pipeline:glance-registry]
# pipeline = unauthenticated-context registryapp

# Use this pipeline for keystone auth
[pipeline:glance-registry-keystone]
pipeline = authtoken context registryapp

[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
admin_tenant_name = services
admin_user = glance
admin_password = secret

[app:registryapp]
paste.app_factory = glance.registry.api.v1:API.factory

[filter:context]
paste.filter_factory = glance.api.middleware.context:ContextMiddleware.factory

[filter:unauthenticated-context]
paste.filter_factory =
glance.api.middleware.context:UnauthenticatedContextMiddleware.factory

[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
```

## D.5.3. glance-api.conf

[Report a bug](#)

The configuration file for the Image API is found in: `/etc/glance/glance-api.conf`

This file must be modified after installation.



```
[DEFAULT]
# Show more verbose log output (sets INFO log level output)
verbose = True

# Show debugging output in logs (sets DEBUG log level output)
debug = False

# Which backend scheme should the Image service use by default? Known schemes are
determined
# by the known_stores option below.
# Default: 'file'
default_store = file

# List of which store classes and store class locations are
# currently known to glance at startup.
#known_stores = glance.store.filesystem.Store,
# glance.store.http.Store,
# glance.store.rbd.Store,
# glance.store.s3.Store,
# glance.store.swift.Store,

# Maximum image size (in bytes) that may be uploaded through the
# Glance API server. Defaults to 1 TB.
# WARNING: this value should only be increased after careful consideration
# and must be set to a value under 8 EB (9223372036854775808).
#image_size_cap = 1099511627776

# Address to bind the API server
bind_host = 0.0.0.0

# Port to bind the API server to
bind_port = 9292

# Log to this file. Make sure you do not set the same log
# file for both the API and registry servers!
log_file = /var/log/glance/api.log

# Backlog requests when creating socket
backlog = 4096

# TCP_KEEPIDLE value in seconds when creating socket.
# Not supported on OS X.
#tcp_keepidle = 600

# SQLAlchemy connection string for the reference implementation
# registry server. Any valid SQLAlchemy connection string is fine.
# See: http://www.sqlalchemy.org/docs/05/reference/sqlalchemy/connections.html#sqlalchemy.create\_engine

# sql_connection = sqlite:///glance.sqlite
# sql_connection = mysql://
glance:YOUR_GLANCEDB_PASSWORD@192.168.206.130/glance

# Period in seconds after which SQLAlchemy should reestablish its connection
# to the database.
#
# MySQL uses a default `wait_timeout` of 8 hours, after which it will drop
# idle connections. This can result in 'MySQL Gone Away' exceptions. If you
# notice this, you can lower this value to ensure that SQLAlchemy reconnects
# before MySQL can drop the connection.
sql_idle_timeout = 3600

# Number of Glance API worker processes to start.
# On machines with more than one CPU increasing this value
```

```
# may improve performance (especially if using SSL with
# compression turned on). It is typically recommended to set
# this value to the number of CPUs present on your machine.
workers = 1

# Role used to identify an authenticated user as administrator
#admin_role = admin

# Allow unauthenticated users to access the API with read-only
# privileges. This only applies when using ContextMiddleware.
#allow_anonymous_access = False

# Allow access to version 1 of glance api
#enable_v1_api = True

# Allow access to version 2 of glance api
#enable_v2_api = True

# ===== Syslog Options =====
# Send logs to syslog (/dev/log) instead of to file specified
# by `log_file`
use_syslog = False

# Facility to use. If unset defaults to LOG_USER.
#syslog_log_facility = LOG_LOCAL0

# ===== SSL Options =====
# Certificate file to use when starting API server securely
#cert_file = /path/to/certfile

# Private key file to use when starting API server securely
#key_file = /path/to/keyfile

# CA certificate file to use to verify connecting clients
#ca_file = /path/to/cafile

# ===== Security Options =====
# AES key for encrypting store 'location' metadata, including
# -- if used -- Swift or S3 credentials
# Should be set to a random string of length 16, 24 or 32 bytes
#metadata_encryption_key = <16, 24 or 32 char registry metadata key>

# ===== Registry Options =====
# Address to find the registry server
registry_host = 0.0.0.0

# Port the registry server is listening on
registry_port = 9191

# What protocol to use when connecting to the registry server?
# Set to https for secure HTTP communication
registry_client_protocol = http

# The path to the key file to use in SSL connections to the
# registry server, if any. Alternately, you may set the
# GLANCE_CLIENT_KEY_FILE environ variable to a filepath of the key file
#registry_client_key_file = /path/to/key/file

# The path to the cert file to use in SSL connections to the
# registry server, if any. Alternately, you may set the
# GLANCE_CLIENT_CERT_FILE environ variable to a filepath of the cert file
#registry_client_cert_file = /path/to/cert/file

# The path to the certifying authority cert file to use in SSL connections
```

```
# to the registry server, if any. Alternately, you may set the
# GLANCE_CLIENT_CA_FILE environ variable to a filepath of the CA cert file
#registry_client_ca_file = /path/to/ca/file

# ===== Notification System Options =====
# Notifications can be sent when images are create, updated or deleted.
# There are three methods of sending notifications, logging (via the
# log_file directive), rabbit (via a rabbitmq queue), qpid (via a Qpid
# message queue), or noop (no notifications sent, the default)
notifier_strategy = noop

# Configuration options if sending notifications via rabbitmq (these are
# the defaults)
rabbit_host = localhost
rabbit_port = 5672
rabbit_use_ssl = false
rabbit_userid = guest
rabbit_password = guest
rabbit_virtual_host = /
rabbit_notification_exchange = glance
rabbit_notification_topic = glance_notifications
rabbit_durable_queues = False

# Configuration options if sending notifications via Qpid (these are
# the defaults)
qpid_notification_exchange = glance
qpid_notification_topic = glance_notifications
qpid_host = localhost
qpid_port = 5672
qpid_username =
qpid_password =
qpid_sasl_mechanisms =
qpid_reconnect_timeout = 0
qpid_reconnect_limit = 0
qpid_reconnect_interval_min = 0
qpid_reconnect_interval_max = 0
qpid_reconnect_interval = 0
qpid_heartbeat = 5

# Set to 'ssl' to enable SSL
qpid_protocol = tcp
qpid_tcp_nodelay = True

# ===== Filesystem Store Options =====
# Directory that the Filesystem backend store
# writes image data to
filesystem_store_datadir = /var/lib/glance/images/

# ===== Swift Store Options =====
# Version of the authentication service to use
# Valid versions are '2' for keystone and '1' for swauth and rackspace
swift_store_auth_version = 2

# Address where the Swift authentication service lives
# Valid schemes are 'http://' and 'https://'
# If no scheme specified, default to 'https://'
# For swauth, use something like '127.0.0.1:8080/v1.0/'
swift_store_auth_address = 127.0.0.1:5000/v2.0/

# User to authenticate against the Swift authentication service
# If you use Swift authentication service, set it to 'account':'user'
# where 'account' is a Swift storage account and 'user'
# is a user in that account
swift_store_user = jdoe:jdoe
```

```
# Auth key for the user authenticating against the
# Swift authentication service
swift_store_key = a86850deb2742ec3cb41518e26aa2d89

# Container within the account that the account should use
# for storing images in Swift
swift_store_container = glance

# Do we create the container if it does not exist?
swift_store_create_container_on_put = False

# What size, in MB, should Glance start chunking image files
# and do a large object manifest in Swift? By default, this is
# the maximum object size in Swift, which is 5GB
swift_store_large_object_size = 5120

# When doing a large object manifest, what size, in MB, should
# Glance write chunks to Swift? This amount of data is written
# to a temporary disk buffer during the process of chunking
# the image file, and the default is 200MB
swift_store_large_object_chunk_size = 200

# Whether to use ServiceNET to communicate with the Swift storage servers.
# (If you aren't RACKSPACE, leave this False!)
#
# To use ServiceNET for authentication, prefix hostname of
# `swift_store_auth_address` with 'snet-'.
# Ex. https://example.com/v1.0/ -> https://snet-example.com/v1.0/
swift_enable_snet = False

# If set to True enables multi-tenant storage mode which causes Glance images
# to be stored in tenant specific Swift accounts.
#swift_store_multi_tenant = False

# A list of tenants that will be granted read/write access on all Swift
# containers created by Glance in multi-tenant mode.
#swift_store_admin_tenants = []

# The region of the swift endpoint to be used for single tenant. This setting
# is only necessary if the tenant has multiple swift endpoints.
#swift_store_region =

# ===== S3 Store Options =====
# Address where the S3 authentication service lives
# Valid schemes are 'http://' and 'https://'
# If no scheme specified, default to 'http://'
s3_store_host = 127.0.0.1:8080/v1.0/

# User to authenticate against the S3 authentication service
s3_store_access_key = <20-char AWS access key>

# Auth key for the user authenticating against the
# S3 authentication service
s3_store_secret_key = <40-char AWS secret key>

# Container within the account that the account should use
# for storing images in S3. Note that S3 has a flat namespace,
# so you need a unique bucket name for your glance images. An
# easy way to do this is append your AWS access key to "glance".
# S3 buckets in AWS *must* be lowercased, so remember to lowercase
# your AWS access key if you use it in your bucket name below!
s3_store_bucket = <lowercased 20-char aws access key>glance
```

```

# Do we create the bucket if it does not exist?
s3_store_create_bucket_on_put = False

# When sending images to S3, the data will first be written to a
# temporary buffer on disk. By default the platform's temporary directory
# will be used. If required, an alternative directory can be specified here.
#s3_store_object_buffer_dir = /path/to/dir

# When forming a bucket url, boto will either set the bucket name as the
# subdomain or as the first token of the path. Amazon's S3 service will
# accept it as the subdomain, but Swift's S3 middleware requires it be
# in the path. Set this to 'path' or 'subdomain' - defaults to 'subdomain'.
#s3_store_bucket_url_format = subdomain

# ===== RBD Store Options =====
# Ceph configuration file path
# If using cephx authentication, this file should
# include a reference to the right keyring
# in a client.<USER> section
rbd_store_ceph_conf = /etc/ceph/ceph.conf

# RADOS user to authenticate as (only applicable if using cephx)
rbd_store_user = glance

# RADOS pool in which images are stored
rbd_store_pool = images

# Images will be chunked into objects of this size (in megabytes).
# For best performance, this should be a power of two
rbd_store_chunk_size = 8

# ===== Delayed Delete Options =====
# Turn on/off delayed delete
delayed_delete = False

# Delayed delete time in seconds
scrub_time = 43200

# Directory that the scrubber will use to remind itself of what to delete
# Make sure this is also set in glance-scrubber.conf
scrubber_datadir = /var/lib/glance/scrubber

# ===== Image Cache Options =====
# Base directory that the Image Cache uses
image_cache_dir = /var/lib/glance/image-cache/
[keystone_authtoken]
    auth_host = 127.0.0.1
auth_port = 35357
auth_protocol = http
admin_tenant_name = services
admin_user = glance
admin_password = secret

[paste_deploy]
# Name of the paste configuration file that defines the available pipelines
config_file = /etc/glance/glance-api-paste.ini

# Partial name of a pipeline in your paste configuration file with the
# service name removed. For example, if your paste section name is
# [pipeline:glance-api-keystone], you would configure the flavor below
# as 'keystone'.
flavor=keystone

```

### **D.5.4. glance-api-paste.ini**

[Report a bug](#)

The Image service's API middleware pipeline is found in: `/etc/glance/glance-api-paste.ini`

You should not need to modify this file.

```
# Use this pipeline for no auth or image caching - DEFAULT
# [pipeline:glance-api]
# pipeline = versionnegotiation unauthenticated-context rootapp

# Use this pipeline for image caching and no auth
# [pipeline:glance-api-caching]
# pipeline = versionnegotiation unauthenticated-context cache rootapp

# Use this pipeline for caching w/ management interface but no auth
# [pipeline:glance-api-cachemanagement]
# pipeline = versionnegotiation unauthenticated-context cache cachemanage
rootapp

# Use this pipeline for keystone auth
[pipeline:glance-api-keystone]
pipeline = versionnegotiation authtoken context rootapp

# Use this pipeline for keystone auth with image caching
# [pipeline:glance-api-keystone+caching]
# pipeline = versionnegotiation authtoken context cache rootapp

# Use this pipeline for keystone auth with caching and cache management
# [pipeline:glance-api-keystone+cachemanagement]
# pipeline = versionnegotiation authtoken context cache cachemanage rootapp

[composite:rootapp]
paste.composite_factory = glance.api:root_app_factory
/: apiversions
/v1: apiv1app
/v2: apiv2app

[app:apiversions]
paste.app_factory = glance.api.versions:create_resource

[app:apiv1app]
paste.app_factory = glance.api.v1.router:API.factory

[app:apiv2app]
paste.app_factory = glance.api.v2.router:API.factory

[filter:versionnegotiation]
paste.filter_factory =
glance.api.middleware.version_negotiation:VersionNegotiationFilter.factory

[filter:cache]
paste.filter_factory = glance.api.middleware.cache:CacheFilter.factory

[filter:cachemanage]
paste.filter_factory =
glance.api.middleware.cache_manage:CacheManageFilter.factory

[filter:context]
paste.filter_factory = glance.api.middleware.context:ContextMiddleware.factory

[filter:unauthenticated-context]
paste.filter_factory =
glance.api.middleware.context:UnauthenticatedContextMiddleware.factory

[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
delay_auth_decision = true
```

## D.5.5. glance-scrubber.conf

The scrubber is a utility for the Image service that cleans up images that have been deleted. Its configuration file is found in: `/etc/glance/glance-scrubber.conf`

```
[DEFAULT]
# Show more verbose log output (sets INFO log level output)
verbose = True

# Show debugging output in logs (sets DEBUG log level output)
debug = False

# Log to this file. Make sure you do not set the same log
# file for both the API and registry servers!
log_file = /var/log/glance/scrubber.log

# Send logs to syslog (/dev/log) instead of to file specified by `log_file`
use_syslog = False

# Delayed delete time in seconds
scrub_time = 43200

# Should we run our own loop or rely on cron/scheduler to run us
daemon = False

# Loop time between checking the registry for new items to schedule for delete
wakeup_time = 300

[app:glance-scrubber]
paste.app_factory = glance.store.scrubber:app_factory
```

## D.6. Networking Service Configuration Files:

### D.6.1. api-paste.ini

```
[composite:quantum]
use = egg:Paste#urlmap
/: quantumversions
/v2.0: quantumapi_v2_0

[composite:quantumapi_v2_0]
use = call:quantum.auth.pipeline_factory
noauth = extensions quantumapiapp_v2_0
keystone = authtoken keystonecontext extensions quantumapiapp_v2_0

[filter:keystonecontext]
paste.filter_factory = quantum.auth:QuantumKeystoneContext.factory

[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory

[filter:extensions]
paste.filter_factory =
quantum.api.extensions:plugin_aware_extension_middleware_factory

[app:quantumversions]
paste.app_factory = quantum.api.versions:Versions.factory

[app:quantumapiapp_v2_0]
paste.app_factory = quantum.api.v2.router:APIRouter.factory
```



[Report a bug](#)

## D.6.2. dhcp\_agent.ini

```
[DEFAULT]
# Show debugging output in log (sets DEBUG log level output)
# debug = true

# The DHCP agent will resync its state with Quantum to recover from any
# transient notification or rpc errors. The interval is number of
# seconds between attempts.
# resync_interval = 5

# The DHCP requires that an interface driver be set. Choose the one that best
# matches you plugin.

# OVS based plugins(OVS, Ryu, NEC, NVP, BigSwitch/Floodlight)
interface_driver = quantum.agent.linux.interface.OVSInterfaceDriver
# OVS based plugins(Ryu, NEC, NVP, BigSwitch/Floodlight) that use OVS
# as OpenFlow switch and check port status
#ovs_use_veth = True
# LinuxBridge
#interface_driver = quantum.agent.linux.interface.BridgeInterfaceDriver

# The agent can use other DHCP drivers. Dnsmasq is the simplest and requires
# no additional setup of the DHCP server.
dhcp_driver = quantum.agent.linux.dhcp.Dnsmasq

# Allow overlapping IP (Must have kernel build with CONFIG_NET_NS=y and
# iproute2 package that supports namespaces).
# use_namespaces = True

# The DHCP server can assist with providing metadata support on isolated
# networks. Setting this value to True will cause the DHCP server to append
# specific host routes to the DHCP request. The metadata service will only
# be activated when the subnet gateway_ip is None. The guest instance must
# be configured to request host routes via DHCP (Option 121).
# enable_isolated_metadata = False

# Allows for serving metadata requests coming from a dedicated metadata
# access network whose cidr is 169.254.169.254/16 (or larger prefix), and
# is connected to a Quantum router from which the VMs send metadata
# request. In this case DHCP Option 121 will not be injected in VMs, as
# they will be able to reach 169.254.169.254 through a router.
# This option requires enable_isolated_metadata = True
# enable_metadata_network = False
```

[Report a bug](#)

## D.6.3. l3\_agent.ini

```
[DEFAULT]
# Show debugging output in log (sets DEBUG log level output)
# debug = True

# L3 requires that an interface driver be set. Choose the one that best
# matches your plugin.

# OVS based plugins (OVS, Ryu, NEC) that supports L3 agent
interface_driver = quantum.agent.linux.interface.OVSInterfaceDriver
# OVS based plugins(Ryu, NEC) that use OVS
# as OpenFlow switch and check port status
#ovs_use_veth = True
# LinuxBridge
#interface_driver = quantum.agent.linux.interface.BridgeInterfaceDriver

# Allow overlapping IP (Must have kernel build with CONFIG_NET_NS=y and
# iproute2 package that supports namespaces).
# use_namespaces = True

# If use_namespaces is set as False then the agent can only configure one router.

# This is done by setting the specific router_id.
# router_id =

# Each L3 agent can be associated with at most one external network. This
# value should be set to the UUID of that external network. If empty,
# the agent will enforce that only a single external networks exists and
# use that external network id
# gateway_external_network_id =

# Indicates that this L3 agent should also handle routers that do not have
# an external network gateway configured. This option should be True only
# for a single agent in a Quantum deployment, and may be False for all agents
# if all routers must have an external network gateway
# handle_internal_only_routers = True

# Name of bridge used for external network traffic. This should be set to
# empty value for the linux bridge
# external_network_bridge = br-ex

# TCP Port used by Quantum metadata server
# metadata_port = 9697

# Send this many gratuitous ARPs for HA setup. Set it below or equal to 0
# to disable this feature.
# send_arp_for_ha = 3

# seconds between re-sync routers' data if needed
# periodic_interval = 40

# seconds to start to sync routers' data after
# starting agent
# periodic_fuzzy_delay = 5

# enable_metadata_proxy, which is true by default, can be set to False
# if the Nova metadata server is not available
# enable_metadata_proxy = True
```

[Report a bug](#)

## D.6.4. Ibaas\_agent.ini

```
[DEFAULT]
# Show debugging output in log (sets DEBUG log level output)
# debug = true

# The LBaaS agent will resync its state with Quantum to recover from any
# transient notification or rpc errors. The interval is number of
# seconds between attempts.
# periodic_interval = 10

# OVS based plugins(OVS, Ryu, NEC, NVP, BigSwitch/Floodlight)
interface_driver = quantum.agent.linux.interface.OVSInterfaceDriver
# OVS based plugins(Ryu, NEC, NVP, BigSwitch/Floodlight) that use OVS
# as OpenFlow switch and check port status
# ovs_use_veth = True
# LinuxBridge
# interface_driver = quantum.agent.linux.interface.BridgeInterfaceDriver

# The agent requires a driver to manage the loadbalancer. HAProxy is the
# opensource version.
device_driver =
quantum.plugins.services.agent_loadbalancer.drivers.haproxy.namespace_driver.HaproxyNSDriver

# Allow overlapping IP (Must have kernel build with CONFIG_NET_NS=y and
# iproute2 package that supports namespaces).
# use_namespaces = True

# The user group
# user_group = nogroup
```

[Report a bug](#)

### D.6.5. metadata\_agent.ini

```
[DEFAULT]
# Show debugging output in log (sets DEBUG log level output)
# debug = True

# The Quantum user information for accessing the Quantum API.
auth_url = http://localhost:35357/v2.0
auth_region = RegionOne
admin_tenant_name = %SERVICE_TENANT_NAME%
admin_user = %SERVICE_USER%
admin_password = %SERVICE_PASSWORD%

# Network service endpoint type to pull from the keystone catalog
# endpoint_type = adminURL

# IP address used by Nova metadata server
# nova_metadata_ip = 127.0.0.1

# TCP Port used by Nova metadata server
# nova_metadata_port = 8775

# When proxying metadata requests, Quantum signs the Instance-ID header with a
# shared secret to prevent spoofing. You may select any string for a secret,
# but it must match here and in the configuration used by the Nova Metadata
# Server. NOTE: Nova uses a different key: quantum_metadata_proxy_shared_secret
# metadata_proxy_shared_secret =
```

[Report a bug](#)

### D.6.6. policy.json



## Error

Topic 18131 failed validation and is not included in this build.

### **D.6.7. quantum.conf**

```
[DEFAULT]
# Default log level is INFO
# verbose and debug has the same result.
# One of them will set DEBUG log level output
# debug = False
# verbose = False

# Where to store Quantum state files. This directory must be writable by the
# user executing the agent.
# state_path = /var/lib/quantum

# Where to store lock files
lock_path = $state_path/lock

# log_format = %(asctime)s %(levelname)8s [% (name)s] %(message)s
# log_date_format = %Y-%m-%d %H:%M:%S

# use_syslog -> syslog
# log_file and log_dir -> log_dir/log_file
# (not log_file) and log_dir -> log_dir/{binary_name}.log
# use_stderr -> stderr
# (not user_stderr) and (not log_file) -> stdout
# publish_errors -> notification system

# use_syslog = False
# syslog_log_facility = LOG_USER

# use_stderr = True
# log_file =
# log_dir =

# publish_errors = False

# Address to bind the API server
bind_host = 0.0.0.0

# Port the bind the API server to
bind_port = 9696

# Path to the extensions. Note that this can be a colon-separated list of
# paths. For example:
# api_extensions_path = extensions:/path/to/more/extensions:/even/more/extensions
# The __path__ of quantum.extensions is appended to this, so if your
# extensions are in there you don't need to specify them here
# api_extensions_path =

# Quantum plugin provider module
# core_plugin =

# Advanced service modules
# service_plugins =

# Paste configuration file
api_paste_config = api-paste.ini

# The strategy to be used for auth.
# Supported values are 'keystone'(default), 'noauth'.
# auth_strategy = keystone

# Base MAC address. The first 3 octets will remain unchanged. If the
# 4h octet is not 00, it will also used. The others will be
# randomly generated.
# 3 octet
# base_mac = fa:16:3e:00:00:00
```

```
# 4 octet
# base_mac = fa:16:3e:4f:00:00

# Maximum amount of retries to generate a unique MAC address
# mac_generation_retries = 16

# DHCP Lease duration (in seconds)
# dhcp_lease_duration = 120

# Allow sending resource operation notification to DHCP agent
# dhcp_agent_notification = True

# Enable or disable bulk create/update/delete operations
# allow_bulk = True
# Enable or disable pagination
# allow_pagination = False
# Enable or disable sorting
# allow_sorting = False
# Enable or disable overlapping IPs for subnets
# Attention: the following parameter MUST be set to False if Quantum is
# being used in conjunction with nova security groups
# allow_overlapping_ips = False
# Ensure that configured gateway is on subnet
# force_gateway_on_subnet = False

# RPC configuration options. Defined in rpc __init__
# The messaging module to use, defaults to kombu.
# rpc_backend = quantum.openstack.common.rpc.impl_kombu
# Size of RPC thread pool
# rpc_thread_pool_size = 64,
# Size of RPC connection pool
# rpc_conn_pool_size = 30
# Seconds to wait for a response from call or multicall
# rpc_response_timeout = 60
# Seconds to wait before a cast expires (TTL). Only supported by impl_zmq.
# rpc_cast_timeout = 30
# Modules of exceptions that are permitted to be recreated
# upon receiving exception data from an rpc call.
# allowed_rpc_exception_modules = quantum.openstack.common.exception,
nova.exception
# AMQP exchange to connect to if using RabbitMQ or QPID
control_exchange = quantum

# If passed, use a fake RabbitMQ provider
# fake_rabbit = False

# Configuration options if sending notifications via kombu rpc (these are
# the defaults)
# SSL version to use (valid only if SSL enabled)
# kombu_ssl_version =
# SSL key file (valid only if SSL enabled)
# kombu_ssl_keyfile =
# SSL cert file (valid only if SSL enabled)
# kombu_ssl_certfile =
# SSL certification authority file (valid only if SSL enabled)'
# kombu_ssl_ca_certs =
# IP address of the RabbitMQ installation
# rabbit_host = localhost
# Password of the RabbitMQ server
# rabbit_password = guest
# Port where RabbitMQ server is running/listening
# rabbit_port = 5672
# RabbitMQ single or HA cluster (host:port pairs i.e: host1:5672, host2:5672)
```

```
# rabbit_hosts is defaulted to '$rabbit_host:$rabbit_port'
# rabbit_hosts = localhost:5672
# User ID used for RabbitMQ connections
# rabbit_userid = guest
# Location of a virtual RabbitMQ installation.
# rabbit_virtual_host = /
# Maximum retries with trying to connect to RabbitMQ
# (the default of 0 implies an infinite retry count)
# rabbit_max_retries = 0
# RabbitMQ connection retry interval
# rabbit_retry_interval = 1
# Use HA queues in RabbitMQ (x-ha-policy: all). You need to
# wipe RabbitMQ database when changing this option. (boolean value)
# rabbit_ha_queues = false

# QPID
# rpc_backend=quantum.openstack.common.rpc.impl_qpid
# Qpid broker hostname
# qpid_hostname = localhost
# Qpid broker port
# qpid_port = 5672
# Qpid single or HA cluster (host:port pairs i.e: host1:5672, host2:5672)
# qpid_hosts is defaulted to '$qpid_hostname:$qpid_port'
# qpid_hosts = localhost:5672
# Username for qpid connection
# qpid_username = ''
# Password for qpid connection
# qpid_password = ''
# Space separated list of SASL mechanisms to use for auth
# qpid_sasl_mechanisms = ''
# Seconds between connection keepalive heartbeats
# qpid_heartbeat = 60
# Transport to use, either 'tcp' or 'ssl'
# qpid_protocol = tcp
# Disable Nagle algorithm
# qpid_tcp_nodelay = True

# ZMQ
# rpc_backend=quantum.openstack.common.rpc.impl_zmq
# ZeroMQ bind address. Should be a wildcard (*), an ethernet interface, or IP.
# The "host" option should point or resolve to this address.
# rpc_zmq_bind_address = *

# ===== Notification System Options =====

# Notifications can be sent when network/subnet/port are create, updated or
# deleted.
# There are three methods of sending notifications: logging (via the
# log_file directive), rpc (via a message queue) and
# noop (no notifications sent, the default)

# Notification_driver can be defined multiple times
# Do nothing driver
# notification_driver = quantum.openstack.common.notifier.no_op_notifier
# Logging driver
# notification_driver = quantum.openstack.common.notifier.log_notifier
# RPC driver. DHCP agents needs it.
notification_driver = quantum.openstack.common.notifier.rpc_notifier

# default_notification_level is used to form actual topic name(s) or to set
# logging level
default_notification_level = INFO

# default_publisher_id is a part of the notification payload
```

```
# host = myhost.com
# default_publisher_id = $host

# Defined in rpc_notifier, can be comma separated values.
# The actual topic names will be %s.%(default_notification_level)s
notification_topics = notifications

# Default maximum number of items returned in a single response,
# value == infinite and value < 0 means no max limit, and value must
# greater than 0. If the number of items requested is greater than
# pagination_max_limit, server will just return pagination_max_limit
# of number of items.
# pagination_max_limit = -1

# Maximum number of DNS nameservers per subnet
# max_dns_nameservers = 5

# Maximum number of host routes per subnet
# max_subnet_host_routes = 20

# Maximum number of fixed ips per port
# max_fixed_ips_per_port = 5

# ===== items for agent management extension =====
# Seconds to regard the agent as down.
# agent_down_time = 5
# ===== end of items for agent management extension =====

# ===== items for agent scheduler extension =====
# Driver to use for scheduling network to DHCP agent
# network_scheduler_driver = quantum.scheduler.dhcp_agent_scheduler.ChanceScheduler
# Driver to use for scheduling router to a default L3 agent
# router_scheduler_driver = quantum.scheduler.l3_agent_scheduler.ChanceScheduler

# Allow auto scheduling networks to DHCP agent. It will schedule non-hosted
# networks to first DHCP agent which sends get_active_networks message to
# quantum server
# network_auto_schedule = True

# Allow auto scheduling routers to L3 agent. It will schedule non-hosted
# routers to first L3 agent which sends sync_routers message to quantum server
# router_auto_schedule = True

# Number of DHCP agents scheduled to host a network. This enables redundant
# DHCP agents for configured networks.
# dhcp_agents_per_network = 1

# ===== end of items for agent scheduler extension =====

# ===== WSGI parameters related to the API server =====
# Sets the value of TCP_KEEPIDLE in seconds to use for each server socket when
# starting API server. Not supported on OS X.
#tcp_keepidle = 600

# Number of seconds to keep retrying to listen
#retry_until_window = 30

# Number of backlog requests to configure the socket with.
#backlog = 4096

# Enable SSL on the API server
#use_ssl = False

# Certificate file to use when starting API server securely
```



```

#ssl_cert_file = /path/to/certfile

# Private key file to use when starting API server securely
#ssl_key_file = /path/to/keyfile

# CA certificate file to use when starting API server securely to
# verify connecting clients. This is an optional parameter only required if
# API clients need to authenticate to the API server using SSL certificates
# signed by a trusted CA
#ssl_ca_file = /path/to/cafile
# ===== end of WSGI parameters related to the API server =====

[QUOTAS]
# resource name(s) that are supported in quota features
# quota_items = network,subnet,port

# default number of resource allowed per tenant, minus for unlimited
# default_quota = -1

# number of networks allowed per tenant, and minus means unlimited
# quota_network = 10

# number of subnets allowed per tenant, and minus means unlimited
# quota_subnet = 10

# number of ports allowed per tenant, and minus means unlimited
# quota_port = 50

# number of security groups allowed per tenant, and minus means unlimited
# quota_security_group = 10

# number of security group rules allowed per tenant, and minus means unlimited
# quota_security_group_rule = 100

# default driver to use for quota checks
# quota_driver = quantum.quota.ConfDriver

[DEFAULT_SERVICETYPE]
# Description of the default service type (optional)
# description = "default service type"
# Enter a service definition line for each advanced service provided
# by the default service type.
# Each service definition should be in the following format:
# <service>:<plugin>[:driver]

[AGENT]
# Use "sudo quantum-rootwrap /etc/quantum/rootwrap.conf" to use the real
# root filter facility.
# Change to "sudo" to skip the filtering and just run the comand directly
# root_helper = sudo

# ===== items for agent management extension =====
# seconds between nodes reporting state to server, should be less than
# agent_down_time
# report_interval = 4

# ===== end of items for agent management extension =====

[keystone_authtoken]
auth_host = 127.0.0.1
auth_port = 35357
auth_protocol = http
admin_tenant_name = %SERVICE_TENANT_NAME%
admin_user = %SERVICE_USER%

```

```

admin_password = %SERVICE_PASSWORD%
signing_dir = /var/lib/quantum/keystone-signing

[LBAAS]
#
=====
=====
# driver_fqn is the fully qualified name of the lbaas driver that will be loaded
by the lbass plugin
#
=====
=====
#driver_fqn =
quantum.plugins.services.agent_loadbalancer.drivers.noop.noop_driver.NoopLbaasDrive
r

```

[Report a bug](#)

## D.6.8. rootwrap.conf

```

[DEFAULT]
# List of directories to load filter definitions from (separated by ',').
# These directories MUST all be only writeable by root !
filters_path=/etc/quantum/rootwrap.d,/usr/share/quantum/rootwrap

# List of directories to search executables in, in case filters do not
# explicitly specify a full path (separated by ',')
# If not specified, defaults to system PATH environment variable.
# These directories MUST all be only writeable by root !
exec_dirs=/sbin,/usr/sbin,/bin,/usr/bin

[XENAPI]
# XenAPI configuration is only required by the L2 agent if it is to
# target a XenServer/XCP compute host's dom0.
xenapi_connection_url=<None>
xenapi_connection_username=root
xenapi_connection_password=<None>

```

[Report a bug](#)

## D.7. Object Storage Service Configuration Files

### D.7.1. account-server.conf

This is an example of an account service configuration file.

```

[DEFAULT]
bind_ip = 1.2.3.4
bind_port = 6002
workers = 2

[pipeline:main]
pipeline = account-server

[app:account-server]
use = egg:swift#account

[account-replicator]

[account-auditor]

[account-reaper]

```

[Report a bug](#)

### D.7.2. container-server.conf

This is an example configuration file for a container service.

```
[DEFAULT]
bind_ip = 1.2.3.4
bind_port = 6001
workers = 2

[pipeline:main]
pipeline = container-server

[app:container-server]
use = egg:swift#container

[container-replicator]

[container-updater]

[container-auditor]

[container-sync]
```

[Report a bug](#)

### D.7.3. object-server.conf

This is an example of an object service configuration file.

```
[DEFAULT]
bind_ip = 1.2.3.4
bind_port = 6001
workers = 2

[pipeline:main]
pipeline = container-server

[app:container-server]
use = egg:swift#container

[container-replicator]

[container-updater]

[container-auditor]

[container-sync]
```

[Report a bug](#)

### D.7.4. proxy-server.conf

This is an example of a proxy service configuration file.

```
[DEFAULT]
bind_port = 8080
workers = 8
user = swift
log_level = debug

[pipeline:main]
pipeline = healthcheck cache authtoken keystone proxy-server

[app:proxy-server]
use = egg:swift#proxy
allow_account_management = true
account_autocreate = true

[filter:cache]
use = egg:swift#memcache
memcache_servers = 127.0.0.1:11211

[filter:catch_errors]
use = egg:swift#catch_errors

[filter:healthcheck]
use = egg:swift#healthcheck

[filter:keystone]
use = egg:swift#keystoneauth
operator_roles = admin, swift
is_admin = true
cache = swift.cache

[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
admin_tenant_name = services
admin_user = swift
admin_password = Redhat123
auth_host = sgordon-quantum.usersys.redhat.com
auth_port = 35357
auth_protocol = http
signing_dir = /tmp/keystone-signing-swift
```

[Report a bug](#)

# Revision History

<b>Revision 3-40</b>	<b>Mon Sep 16 2013</b>	<b>Scott Radvan</b>
<a href="#">BZ#1004575</a> - Only show configuring v2.0 of the endpoint API.		
<b>Revision 3-37</b>	<b>Fri Sep 06 2013</b>	<b>Scott Radvan</b>
<a href="#">BZ#997589</a> - Update procedure formatting.		
<b>Revision 3-36</b>	<b>Fri Sep 06 2013</b>	<b>Scott Radvan</b>
<a href="#">BZ#997589</a> - Specify correct package for VNC service.		
<b>Revision 3-35</b>	<b>Tue Sep 03 2013</b>	<b>Scott Radvan</b>
<a href="#">BZ#975268</a> - Move component listings into tables.		
<b>Revision 3-34</b>	<b>Mon Sep 02 2013</b>	<b>Scott Radvan</b>
<a href="#">BZ#984897</a> - Fix incorrect /etc file path.		
<b>Revision 3-33</b>	<b>Thu Aug 08 2013</b>	<b>Stephen Gordon</b>
Updated brand.		
<b>Revision 3-32</b>	<b>Tue Aug 06 2013</b>	<b>Stephen Gordon</b>
<a href="#">BZ#988471</a> - Updated explanation of RHS SELinux boolean.		
<a href="#">BZ#988151</a> - Added brief explanation of NFS component of multi-backend example.		
<b>Revision 3-31</b>	<b>Tue Aug 06 2013</b>	<b>Stephen Gordon</b>
<a href="#">BZ#988471</a> - Add setting of virt_use_fusefs SELinux boolean to RHS backend configuration.		
<a href="#">BZ#984898</a> - Added missing "on" argument to setsebool command.		
<a href="#">BZ#989832</a> - Removed extra apostrophe in certutil example.		
<a href="#">BZ#980547</a> - Replace /etc/nova/nova/conf with /etc/nova/nova.conf in Updating the Compute Configuration section.		
<a href="#">BZ#988150</a> - Updated Block Storage multi-backend configuration content to provider correct Red Hat Storage driver.		
<a href="#">BZ#988151</a> - Updated Block Storage multi-backend configuration to provide a valid NFS example.		
<a href="#">BZ#991099</a> - Updated database connection string instructions in "Configuring the Network Service" to refer to the correct file.		
<a href="#">BZ#928038</a> - Correct description of bridge_mappings configuration key.		
<a href="#">BZ#981430</a> - Start openstack-nova-consoleauth service.		
<b>Revision 3-30</b>	<b>Wed Jul 17 2013</b>	<b>Stephen Gordon</b>
<a href="#">BZ#981430</a> - Added instructions on the use of the Compute console authentication service.		
<a href="#">BZ#980559</a> - Added API, conductor, and scheduler services to compute installation instructions.		
<a href="#">BZ#980902</a> - Added missing authentication configuration for the Compute service.		
<a href="#">BZ#980860</a> - Added instructions for starting Libvirt service before the Compute (Nova) service.		
<a href="#">BZ#928038</a> - Added bridge_mappings and physical_interface_mappings configuration keys to networking documentation.		
<b>Revision 3-29</b>	<b>Mon Jul 08 2013</b>	<b>Stephen Gordon</b>
<a href="#">BZ#980565</a> - Added missing database grant for 'dash' database user.		
<a href="#">BZ#975419</a> - Fixed Nova API restart command to refer to the correct service.		
<b>Revision 3-26</b>	<b>Mon Jul 01 2013</b>	<b>Stephen Gordon</b>

BZ#[979150](#) - Updated entitlement information.  
BZ#[978657](#) - Fixed Identity service v3 endpoint definition.  
BZ#[979004](#) - Corrected Compute Identity records.

<b>Revision 3-24</b>	<b>Mon Jun 24 2013</b>	<b>Stephen Gordon, Summer Long</b>
----------------------	------------------------	------------------------------------

BZ#[976842](#) - Corrected endpoint definitions for Object Storage service.  
BZ#[975419](#) - Corrected restart command for Compute API service.  
BZ#[965900](#) - Finalized installation checklist.  
BZ#[973845](#) - Removed duplicated content in prerequisites information.  
BZ#[877820](#) - Updated example `/etc/libvirt/qemu.conf` configuration file.

<b>Revision 3-23</b>	<b>Thu Jun 20 2013</b>	<b>Stephen Gordon</b>
----------------------	------------------------	-----------------------

[BZ#973845](#) - Removed duplication in "Register to Red Hat Network" instructions.  
[BZ#877820](#) - Updated spacing of qemu settings.

<b>Revision 3-22</b>	<b>Tue Jun 18 2013</b>	<b>Stephen Gordon, Summer Long</b>
----------------------	------------------------	------------------------------------

BZ#[974430](#) - Updated to use the correct credentials when creating a new user.  
BZ#[974434](#) - Use root user account instead of sudo when configuring Object Storage.  
BZ#[967965](#) - Added steps for creating an external provider network.  
BZ#[928376](#) - Added Red Hat Storage backend for Block Storage Service.  
BZ#[959515](#) - Added new topic covering installation prerequisites.

<b>Revision 3-21</b>	<b>Thu Jun 13 2013</b>	<b>Steve Gordon</b>
----------------------	------------------------	---------------------

Red Hat OpenStack 3.0 Preview update.

<b>Revision 3-13</b>	<b>Wed May 29 2013</b>	<b>Steve Gordon</b>
----------------------	------------------------	---------------------

Red Hat OpenStack 3.0 Preview.