Red Hat Enterprise Linux OpenStack Platform 4 Installation and Configuration Guide

Installing and Configuring OpenStack environments manually

25 Mar 2014

Don Domingo
Scott Radvan
Tim Hildred
Bruce Reeler
Joshua Wulf
Steve Gordon
Deepti Navale
Martin Lopes
Summer Long

Red Hat Enterprise Linux OpenStack Platform 4 Installation and Configuration Guide

Installing and Configuring OpenStack environments manually

25 Mar 2014

Bruce Reeler
Red Hat Engineering Content Services
breeler@redhat.com

Deepti Navale
Red Hat Engineering Content Services
dnavale@redhat.com

Don Domingo
Red Hat Engineering Content Services
ddomingo@redhat.com

Joshua Wulf
Red Hat Engineering Content Services
jwulf@redhat.com

Martin Lopes
Red Hat Engineering Content Services
mlopes@redhat.com

Scott Radvan
Red Hat Engineering Content Services
sradvan@redhat.com

Steve Gordon
Red Hat Engineering Content Services
sgordon@redhat.com

Summer Long
Red Hat Engineering Content Services
slong@redhat.com

Tim Hildred
Red Hat Engineering Content Services
thildred@redhat.com

Abstract

Installing and Configuring OpenStack environments with Red Hat Enterprise Linux OpenStack Platform 4 (Havana).

Table of Contents

Preface **8**
1. Document Conventions 8
1.1. Typographic Conventions 8
1.2. Pull-quote Conventions 9
1.3. Notes and Warnings 9
2. Getting Help and Giving Feedback 10

Part I. Introduction **11**

Chapter 1. Product Introduction **12**
1.1. Overview 12
1.2. Architecture 12
1.3. Service Details 13
1.3.1. Dashboard Service 13
1.3.2. Identity Service 14
1.3.3. OpenStack Networking Service 15
1.3.4. Block Storage Service 15
1.3.5. Compute Service 16
1.3.6. Image Service 17
1.3.7. Object Storage Service 18
1.3.8. Telemetry Service 19
1.3.9. Orchestration Service 19
1.4. Deployment Tools 20
1.4.1. The PackStack Deployment Utility 20
1.4.2. Foreman OpenStack Manager 20
1.5. Supported Virtual Machine Operating Systems 21
1.6. Additional Documentation 21

Chapter 2. Prerequisites **22**
2.1. Software Requirements 22
2.1.1. Operating System Requirements 22
2.1.2. Software Repository Configuration 22
2.1.2.1. Register to Red Hat Network 22
2.1.2.2. Red Hat Enterprise Linux Repository Configuration 23
2.1.2.3. Red Hat Enterprise Linux OpenStack Platform Repository Configuration 24
2.1.3. Reserved UIDs and GIDs 27
2.2. Hardware Requirements 27
2.2.1. Compute Node Requirements 27
2.2.2. Network Node Requirements 28
2.2.3. Block Storage Node Requirements 28

Part II. Deploying OpenStack with Foreman **30**

Chapter 3. Foreman Overview and Installation **31**
3.1. Introduction to Foreman OpenStack Manager 31
3.2. Foreman Modes 31
3.3. Foreman Host Groups 31
3.4. Configure the Firewall to Allow Foreman Traffic 32
3.5. Install Foreman Packages 33
3.6. Configure the Foreman Installer 33
3.7. Run the Foreman Installer 34

Chapter 4. Configuring Foreman **36**
4.1. Change the Password 36
4.2. Configure Installation Media 37
4.3. Edit Host Groups 38

4.3. Edit Host Groups 38
4.3.1. Controller (Nova Network) 39
4.3.2. Compute (Nova Network) 40
4.3.3. Controller (Neutron) 41
4.3.4. Neutron Networker 41
4.3.5. Compute (Neutron) 41

Chapter 5. Adding Hosts 42
5.1. Register Existing Hosts in the Foreman User Interface 42
5.2. Provision New Hosts 43
5.3. Assign Client Hosts to Host Groups 46

Chapter 6. Logging In 48

Part III. Installing OpenStack Manually 49

Chapter 7. Installing the Database Server 50
7.1. Install the MySQL Database Packages 50
7.2. Configure the Firewall to Allow Database Traffic 50
7.3. Start the Database Service 50
7.4. Set the Database Administrator Password 51

Chapter 8. Installing the Message Broker 52
8.1. Install the Message Broker Packages 52
8.2. Configuring the Message Broker 52
8.2.1. Basic Qpid Configuration 52
8.2.2. Simple Authentication and Security Layer - SASL 52
8.2.2.1. Simple Authentication and Security Layer (SASL) 52
8.2.2.2. SASL Mechanisms 52
8.2.2.3. SASL Mechanisms and Packages 53
8.2.2.4. Configure SASL Using a Local Password File 53
8.2.3. Configuring TLS/SSL 54
8.2.3.1. Encryption Using SSL 54
8.2.3.2. Enabling SSL on the Broker 54
8.2.3.3. Exporting an SSL Certificate for Clients 56
8.3. Firewall Configuration 56
8.4. Start the Messaging Server 57

Chapter 9. Installing the OpenStack Identity Service 58
9.1. Identity Service Requirements 58
9.2. Install the Identity Packages 58
9.3. Create the Identity Database 58
9.4. Configuring the Service 59
9.4.1. Set the Identity Service Administration Token 59
9.4.2. Configure the Identity Service Database Connection 60
9.4.3. Configuring the Public Key Infrastructure 60
9.4.3.1. Public Key Infrastructure Overview 60
9.4.3.2. Create the Public Key Infrastructure Files 61
9.4.3.3. Configure the Identity Service to Use Public Key Infrastructure Files 61
9.4.4. Integrate Identity Service with LDAP 61
9.4.5. Harden LDAP Integration 63
9.4.6. Configure the Firewall to Allow Identity Service Traffic 64
9.4.7. Populate the Identity Service Database 65
9.5. Start the Identity Service 65
9.6. Create the Identity Service Endpoint 65
9.7. Create an Administrator Account 66
9.8. Create a Regular User Account 68
9.9. Create the Services Tenant 69
9.10. Configuring the Identity Service to Use External Authentication 70
9.10.1. Configure HTTPD to use SSL 70

9.10.1. Configure HTTPD to use SSL 70
9.10.2. Configure the Identity Service to Run in HTTPD 71
9.10.3. Implement External Authentication for the Identity Service Using HTTPD 73
9.10.4. Additional Setup Instructions for Connecting Services 73
9.11. Validate the Identity Service Installation 73

Chapter 10. Installing the OpenStack Object Storage Service 75
10.1. Services that Make Up the Object Storage Service 75
10.2. Architecture of the Object Storage Service 75
10.3. Object Storage Service Requirements 76
10.4. Install the Object Storage Service Packages 76
10.5. Configuring the Object Storage Service 77
10.5.1. Create the Object Storage Service Identity Records 77
10.5.2. Configure the Object Storage Service Storage Nodes 78
10.5.3. Configure the Object Storage Service Proxy Service 80
10.5.4. Object Storage Service Rings 81
10.5.5. Build Object Storage Service Ring Files 82
10.6. Validate the Object Storage Service Installation 83

Chapter 11. Installing the OpenStack Image Service 85
11.1. Image Service Requirements 85
11.2. Install the Image Service Packages 85
11.3. Create the Image Service Database 85
11.4. Configuring the Image Service 86
11.4.1. Configuration Overview 86
11.4.2. Create the Image Identity Records 86
11.4.3. Configure the Image Service Database Connection 87
11.4.4. Configure Image Service Authentication 88
11.4.5. Using the Object Storage Service for Image Storage 89
11.4.6. Configure the Firewall to Allow Image Service Traffic 90
11.4.7. Configure the Image Service to Use SSL 90
11.4.8. Populate the Image Service Database 90
11.4.9. Enabling Image Loading via the Local File System 91
11.4.9.1. Configure File System Sharing Across Different Image and Compute Nodes 91
11.4.9.2. Configure the Image Service to Provide Images Through the Local File System 92
11.4.9.3. Configure the Compute Service to Use Local File System Metadata 93
11.5. Launch the Image API and Registry Services 94
11.6. Validating the Image Service Installation 94
11.6.1. Obtain a Test Disk Image 94
11.6.2. Build a Custom Virtual Machine Image 95
11.6.3. Upload a Disk Image 97

Chapter 12. Installing OpenStack Block Storage 100
12.1. Block Storage Installation Overview 100
12.2. Block Storage Prerequisite Configuration 101
12.2.1. Create the Block Storage Database 101
12.2.2. Create the Block Storage Service Identity Records 101
12.3. Common Block Storage Configuration 103
12.3.1. Install the Block Storage Service Packages 103
12.3.2. Configure Block Storage Service Authentication 103
12.3.3. Configure the Block Storage Service to Use SSL 104
12.3.4. Configure Message Broker Settings for the Block Storage Service 104
12.3.5. Configure the Block Storage Service Database Connection 106
12.3.6. Configure the Firewall to Allow Block Storage Service Traffic 106
12.3.7. Populate the Block Storage Database 107
12.4. Volume Service Specific Configuration 107
12.4.1. Block Storage Driver Support 107
12.4.2. Configure OpenStack Block Storage to use an LVM Storage Back End 108
12.4.3. Configuring for NFS Storage Backend 108

12.4.3. Configuring for NFS Storage Backend 108
12.4.4. Configuring for Red Hat Storage Backend 110
12.4.5. Configuring for Multiple Storage Backends 112
12.4.6. Configure tgtd 113
12.5. Launch the Block Storage Services 114
12.6. Validate the Block Storage Service Installation 115

Chapter 13. Installing the OpenStack Networking Service 117
13.1. OpenStack Networking Installation Overview 117
13.1.1. OpenStack Networking Architecture 117
13.1.2. OpenStack Networking API 117
13.1.3. OpenStack Networking API Extensions 117
13.1.4. OpenStack Networking Plug-ins 118
13.1.5. Open vSwitch overview 119
13.1.6. Modular Layer 2 (ML2) Overview 119
13.1.7. Configure the L2 Population mechanism driver 120
13.1.8. OpenStack Networking Agents 120
13.1.9. Comparing Tenant and Provider networks 120
13.1.10. Recommended Networking Deployment 121
13.1.11. Kernel Requirements 123
13.2. Networking Prerequisite Configuration 123
13.2.1. Create the OpenStack Networking Database 123
13.2.2. Configure OpenStack Networking Authentication 124
13.3. Common Networking Configuration 125
13.3.1. Disable Network Manager 125
13.3.2. Install the Networking Service Packages 127
13.3.3. Configure the Firewall to Allow OpenStack Networking Traffic 127
13.4. Configuring the Networking Service 128
13.4.1. Configure Networking Service Authentication 128
13.4.2. Configure the Message Broker Settings of the Networking Service 129
13.4.3. Set the Networking Service Plug-in 130
13.4.4. Configure Open vSwitch Tunnels 131
13.4.5. Configure the Networking Service Database Connection 133
13.4.6. Launch the Networking Service 133
13.5. Configure the DHCP Agent 133
13.6. Create an External Network 135
13.7. Configuring the Plug-in Agent 136
13.7.1. Configure the Open vSwitch Plug-in Agent 136
13.7.2. Configure the Linux Bridge Plug-in Agent 138
13.8. Configure the L3 Agent 139
13.9. Validate the OpenStack Networking Installation 141
13.10. Load Balancing as a Service (LBaaS) Overview 142
13.11. Configure Load Balancing as a Service (LBaaS) 143
13.12. Troubleshoot OpenStack Networking Issues 143

Chapter 14. Installing the OpenStack Compute Service 145
14.1. Compute Service Requirements 145
14.1.1. Check for Hardware Virtualization Support 145
14.2. Installing a Compute VNC Proxy 145
14.2.1. Install the Compute VNC Proxy Packages 145
14.2.2. Configure the Firewall to Allow Compute VNC Proxy Traffic 146
14.2.3. Configure the VNC Proxy Service 146
14.2.4. Access Instances with the Compute VNC Proxy 147
14.3. Installing a Compute Node 148
14.3.1. Create the Compute Service Database 148
14.3.2. Configure Compute Service Authentication 148
14.3.3. Install the Compute Service Packages 149
14.3.4. Configure the Compute Service to Use SSL 150
14.3.5. Configuring the Compute Service 150

14.3.5. Configuring the Compute Service 150
14.3.5.1. Configure Compute Service Authentication 150
14.3.5.2. Configure the Compute Service Database Connection 151
14.3.5.3. Configure Message Broker Settings for the Compute Service 152
14.3.5.4. Configure Resource Overcommitment 153
14.3.5.5. Reserve Host Resources 154
14.3.5.6. Configuring Compute Networking 154
14.3.5.6.1. Compute Networking Overview 154
14.3.5.6.2. Update the Compute Configuration 154
14.3.5.6.3. Configure the L2 Agent 155
14.3.5.6.4. Configure Virtual Interface Plugging 155
14.3.5.7. Configure the Firewall to Allow Compute Service Traffic 156
14.3.6. Populate the Compute Service Database 157
14.3.7. Launch the Compute Services 157

Chapter 15. Installing the OpenStack Telemetry Service 160
15.1. Overview of Telemetry Service Deployment 160
15.2. Install the Telemetry Service Packages 160
15.3. Create the Telemetry Identity Records 161
15.4. Configure Telemetry Service Authentication 163
15.5. Setting the Telemetry Service Database Connection URL 164
15.5.1. Configure and Launch the MongoDB Back-End 165
15.6. Configure the Compute Node 165
15.7. Configure Monitored Services 166
15.8. Launch the Telemetry API and Agents 167

Chapter 16. Installing the OpenStack Orchestration Service 168
16.1. Install the Orchestration Service Packages 168
16.2. Configuring the Orchestration Service 168
16.2.1. Configure the Orchestration Service Database Connection 168
16.2.2. Restrict the Bind Addresses of each Orchestration API Service 169
16.2.3. Create the Orchestration Service Identity Records 170
16.2.4. Configure Orchestration Service Authentication 171
16.2.5. Configure the Message Broker Settings of the Orchestration Service 172
16.3. Launch the Orchestration Service 173
16.4. Deploy a Stack Using Orchestration Templates 174
16.5. Integrate Telemetry and Orchestration Services 175

Chapter 17. Installing the Dashboard 177
17.1. Dashboard Service Requirements 177
17.2. Install the Dashboard Packages 177
17.3. Launch the Apache Web Service 178
17.4. Configuring the Dashboard 178
17.4.1. Configure Connections and Logging 178
17.4.2. Configure the Dashboard to Use HTTPS 179
17.4.3. Create a Member Role 180
17.4.4. Configure SELinux 180
17.4.5. Configure the Dashboard Firewall 181
17.4.6. Session Storage Options 181
17.4.6.1. Configure Local Memory Cache Session Storage 182
17.4.6.2. Configure Memcached Session Storage 182
17.4.6.3. Configure Database Session Storage 182
17.4.6.4. Configure Cached Database Session Storage 183
17.4.6.5. Configure Cookies Session Storage 183
17.5. Validate the Dashboard Installation 184

Part IV. Validating the Installation 186

Chapter 18. Touring the Cloud 187

18.1. OpenStack Dashboard - Admin Tab 187
18.2. OpenStack Dashboard - Project Tab 188
18.3. OpenStack Dashboard - Red Hat Tab 189

Chapter 19. Working with Instances 191
19.1. Uploading a Disk Image 191
19.2. Creating a Keypair 193
19.3. Creating a Network 193
19.4. Launching an Instance 194
19.5. Creating a Volume 197
19.6. Attaching a Volume to an Instance 198
19.7. Creating an Instance Snapshot 199
19.8. Controlling the State of an Instance (Pause, Suspend, Reboot) 199

Chapter 20. Updating the Environment 201
20.1. Defining a Floating IP-Address Pool 201
20.2. Creating a Router 202
20.3. Associating a Floating IP with the Instance 204
20.4. Adding a Rule to a Security Group 205

Part V. Monitoring the OpenStack Environment 208

Chapter 21. Monitoring OpenStack using Nagios 209
21.1. Installing Nagios 209
21.1.1. Installing the Nagios Service 209
21.1.2. Installing the NRPE Addon 209
21.2. Configuring Nagios 210
21.2.1. Setting up Nagios 210
21.2.2. Configuring HTTPD 210
21.2.3. Configuring OpenStack Services 212
21.2.4. Configure NRPE 213
21.2.5. Creating Host Definitions 213
21.2.6. Creating Service Definitions 214
21.2.7. Verifying the Configuration 215

Chapter 22. Installing and Configuring Remote Logging 216
22.1. Introduction to Remote Logging 216
22.2. Install rsyslog Server 216
22.3. Configure rsyslog on the Centralized Logging Server 216
22.4. Configure rsyslog on Individual Nodes 217
22.5. Start the rsyslog Server 217

Part VI. Managing OpenStack Environment Expansion 219

Chapter 23. Managing Compute Expansion 220
23.1. Defining Regions 220
23.2. Adding Compute Resources 220
23.3. Safely Remove Compute Resources 221
23.4. Using Config Drive 222
23.4.1. Config Drive Overview 222
23.4.2. Setting Up Config Drive 222
23.4.2.1. Enabling Config Drive 222
23.4.2.2. Config Drive Options 223
23.4.3. Access Config Drive 224
23.4.4. Data Formats 225
23.4.4.1. OpenStack Metadata Format 225
23.4.4.2. EC2 Metadata Format 226
23.4.4.3. User Data Format 226

Installation Checklist 227

Installation Checklist 227
A.1. Installation Prerequisites Checklists 227

Troubleshooting the OpenStack Environment 231
B.1. No Networks or Routers Tab Appears in the Dashboard 231
B.2. Dashboard Reports ERROR When Launching Instances 231
B.3. Compute Instance Log Shows no Output 231
B.4. Troubleshoot Identity Client (keystone) Connectivity Problems 231
B.5. Backup and Restore 232
B.5.1. Backup Overview 232
B.5.2. Database Backup 233
B.5.3. File System Backup 234
B.5.4. Recovery 235

Additional Procedures 237
C.1. Using Foreman to deploy Neutron 237
C.2. Testing that SSL certificates are set up correctly 238

Service Log Files 240
D.1. Block Storage Service Log Files 240
D.2. Compute Service Log Files 240
D.3. Dashboard Log Files 240
D.4. Identity Service Log Files 241
D.5. Image Service Log Files 241
D.6. Networking Service Log Files 241
D.7. Object Storage Service Log Files 241

Revision History 243

Preface

1. Document Conventions

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

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

1.1. Typographic Conventions

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

`Mono-spaced Bold`

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

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

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

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

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

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

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

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

Proportional Bold

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

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, select the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

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

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

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

> To connect to a remote machine using ssh, type **`ssh`** ***`username@domain.name`*** at a shell prompt. If the remote machine is **`example.com`** and your username on that machine is john, type **`ssh john@example.com`**.
>
> The **`mount -o remount`** ***`file-system`*** command remounts the named file system. For example, to remount the **`/home`** file system, the command is **`mount -o remount /home`**.
>
> To see the version of a currently installed package, use the **`rpm -q`** ***`package`*** command. It will return a result as follows: ***`package-version-release`***.

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

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

> Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

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

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

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

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

```
static int kvm_vm_ioctl_deassign_device(struct kvm *kvm,
                 struct kvm_assigned_pci_dev *assigned_dev)
{
         int r = 0;
         struct kvm_assigned_dev_kernel *match;

         mutex_lock(&kvm->lock);

         match = kvm_find_assigned_dev(&kvm->arch.assigned_dev_head,
                                       assigned_dev->assigned_dev_id);
         if (!match) {
                 printk(KERN_INFO "%s: device hasn't been assigned before, "
                   "so cannot be deassigned\n", __func__);
                 r = -EINVAL;
                 goto out;
         }

         kvm_deassign_device(kvm, match);

         kvm_free_assigned_device(kvm, match);

out:
         mutex_unlock(&kvm->lock);
         return r;
}
```

1.3. Notes and Warnings

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

Note

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

Important

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

Warning

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

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 Havana, 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.

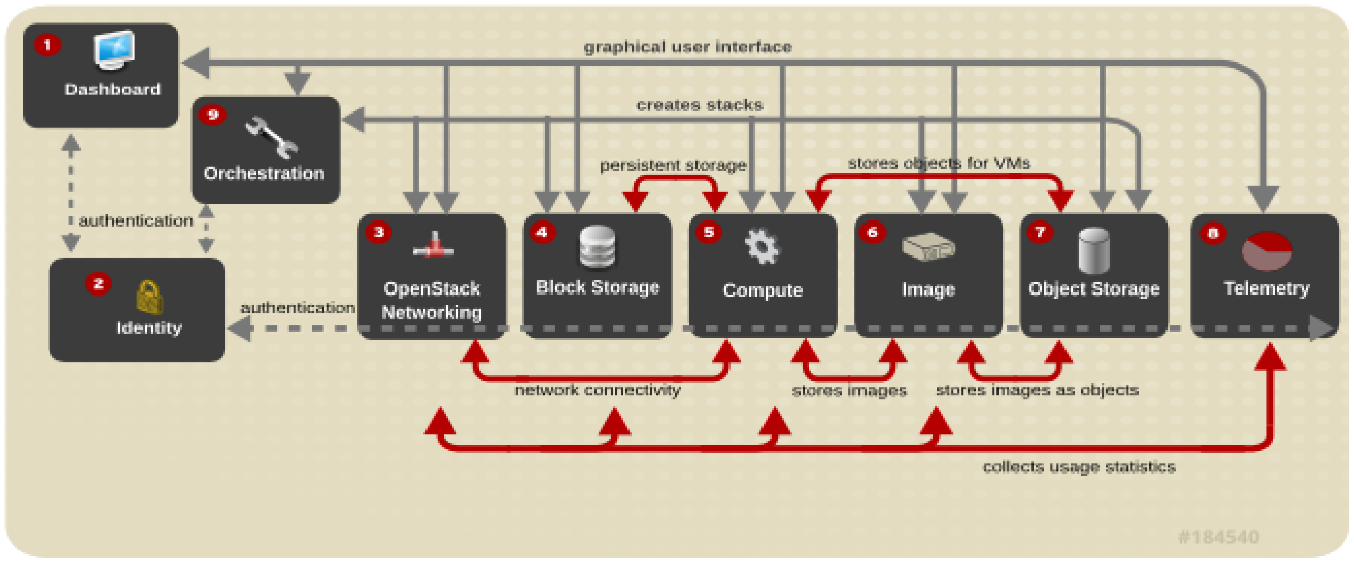


Figure 1.1. OpenStack Architecture

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	Codename	Description
❶	Dashboard	horizon	A web-based dashboard for managing OpenStack services.
❷	Identity	keystone	A centralized Identity service that provides authentication and authorization for other services, and manages users, tenants, and roles.
❸	OpenStack Networking	neutron	A networking service that provides connectivity between the interfaces of other OpenStack services.
❹	Block Storage	cinder	A service that manages persistent block storage volumes for virtual machines.
❺	Compute	nova	A service that launches and schedules networks of machines running on nodes.
❻	Image	glance	A registry service for virtual machine images.
❼	Object Storage	swift	A service providing object storage which allows users to store and retrieve files (arbitrary data).
❽	Telemetry	ceilometer	A service providing measurements of cloud resources.
❾	Orchestration	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.

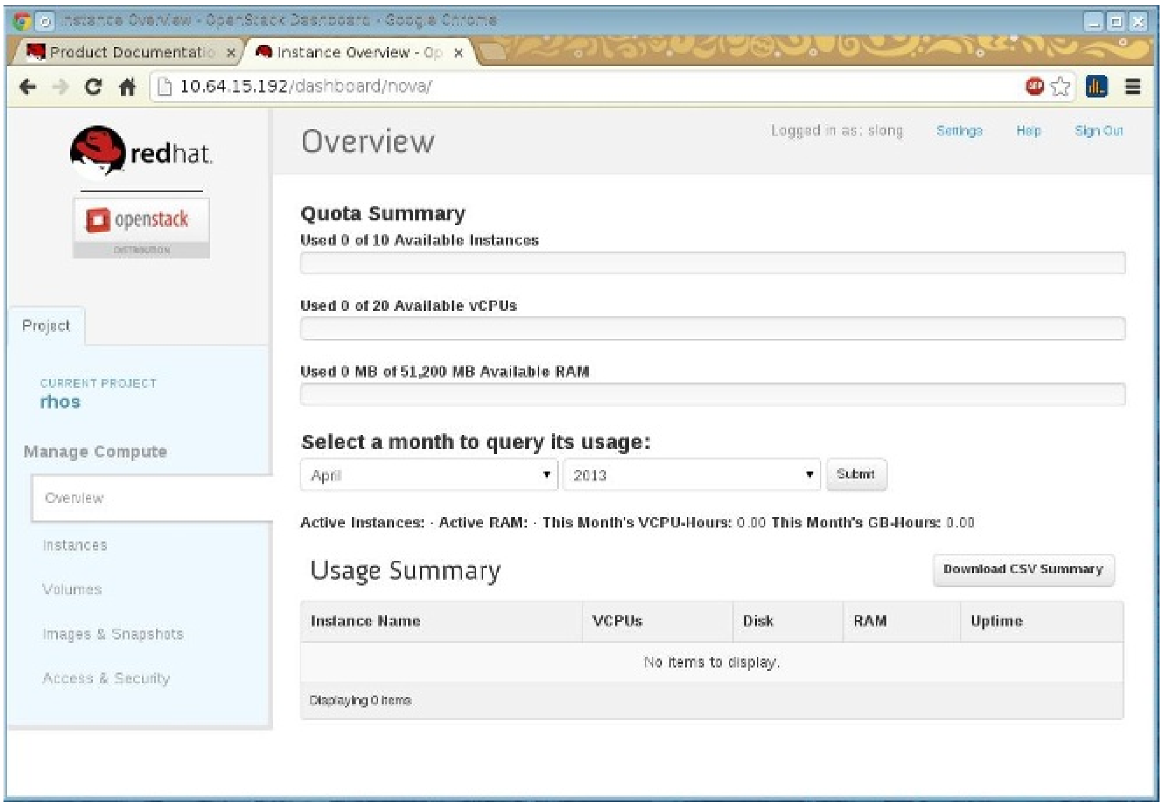
Report a bug

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:



Overview

Quota Summary

Used 0 of 10 Available Instances

Used 0 of 20 Available vCPUs

Used 0 MB of 51,200 MB Available RAM

Select a month to query its usage:

April 2013

Active Instances: - Active RAM: - This Month's VCPU-Hours: 0.00 This Month's GB-Hours: 0.00

Usage Summary

Instance Name	VCPUs	Disk	RAM	Uptime
No items to display.				

Displaying 0 items

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.

Report a bug

1.3.2. Identity Service

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. OpenStack services use the following endpoints:

- **`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 the 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.

Report a bug

1.3.3. OpenStack Networking Service

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
neutron-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 **`openvswitch`** and **`linuxbridge`** plugins utilize native Linux networking mechanisms, while other plugins interface with external devices or SDN controllers.
neutron-l3-agent	An agent providing L3/NAT forwarding.
neutron-*-agent	A plug-in agent that runs on each node to perform local networking configuration for the node's VMs and networking services.
neutron-dhcp-agent	An agent providing DHCP services to tenant networks.
Database	Provides persistent storage.

Report a bug

1.3.4. Block Storage Service

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

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-backup	Provides the ability to back up a Block Storage volume to an external storage repository.
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	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.
Cells	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. This means that: - A single API server can be used to control access to multiple Compute installations. - 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.

Concept	Description
Host Aggregates and Availability Zones	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). If the user is: » 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). » 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. Aggregates, or zones, can be used to: » Handle load balancing and instance distribution. » Provide some form of physical isolation and redundancy from other zones (such as by using a separate power supply or network equipment). » Identify a set of servers that have some common attribute. » Separate out different classes of hardware.

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 AMQP 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.
libvirtd	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.

Report a bug

1.3.6. Image Service

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.

Report a bug

1.3.7. Object Storage Service

The Object Storage service provides object storage in virtual containers, allowing 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, 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 through 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.

Component	Description
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	Stores account data.
Container database	Stores container data.
ext4 (recommended) or XFS file system	Used for object storage.
Housekeeping processes	Replication and auditors.

Report a bug

1.3.8. Telemetry Service

The Telemetry 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).

Telemetry includes a storage daemon that communicates with authenticated agents through 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. Telemetry 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 Telemetry messages, and sent back out on to the message bus using the appropriate topic. Telemetry messages are written to the data store without modification.
MongoDB 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.

Telemetry service dependencies

- Each **`nova-compute`** node must have a **`ceilometer-compute`** agent deployed and running.
- All nodes running the **`ceilometer-api`** service must have firewall rules granting appropriate access.
- The **`ceilometer-central-agent`** cannot currently be horizontally scaled, so only a single instance of this service should be running at any given moment.
- You can choose where to locate the additional Telemetry agents, as all intra-agent communication is either based on AMQP or REST calls to the **`ceilometer-api`** service; as is the case for the **`ceilometer-alarm-evaluator`** service.

Report a bug

1.3.9. Orchestration Service

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 using 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-cfntools	A package of helper scripts (for example, cfn-hup, which handles updates to metadata and executes custom hooks).

Note

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

Report a bug

1.4. Deployment Tools

1.4.1. The PackStack Deployment Utility

PackStack is a command-line utility that uses Puppet modules to enable rapid deployment of OpenStack on existing servers over an SSH connection. Deployment options are provided either interactively, using the command line, or non-interactively by means of a text file containing a set of preconfigured values for OpenStack parameters.

PackStack is suitable for deploying the following types of configurations:

- Single-node proof-of-concept installations, where all controller services and your virtual machines run on a single physical host. This is referred to as an all-in-one install.
- Proof-of-concept installations where there is a single controller node and multiple compute nodes. This is similar to the all-in-one install above, except you may use one or more additional hardware nodes for running virtual machines.

PackStack is not suitable as a production deployment tool, as it uses many assumptions in its configuration to make demos and proof-of-concept installations easier. For production deployments it is recommended that you use Foreman or configure individual services manually.

PackStack is not able to deploy services in a highly available (HA) or load balanced configuration, nor can it provide the flexibility needed to configure complex networking. These features are reserved for the production deployment tools, such as Foreman.

For more information about PackStack, see the *Red Hat Enterprise Linux OpenStack Platform Getting Started Guide*.

Report a bug

1.4.2. Foreman OpenStack Manager

Foreman OpenStack Manager is a deployment management tool. It provides a web user interface for managing the installation and configuration of remote systems. Deployment of changes is performed using Puppet. Additionally, Dynamic Host Configuration Protocol (DHCP), Domain Name System (DNS), Preboot Execution Environment (PXE), and Trivial File Transfer Protocol (TFTP) services can be provided. Controlling these services also enables provisioning of physical systems that do not yet have an operating system installed.

Report a bug

1.5. Supported Virtual Machine Operating Systems

Red Hat Enterprise Linux OpenStack Platform presently supports the virtualization of these guest operating systems:

- Red Hat Enterprise Linux 3 (32 bit and 64 bit)
- Red Hat Enterprise Linux 4 (32 bit and 64 bit)
- Red Hat Enterprise Linux 5 (32 bit and 64 bit)
- Red Hat Enterprise Linux 6 (32 bit and 64 bit)
- Windows XP Service Pack 3 and newer (32 bit only)
- Windows 7 (32 bit and 64 bit)
- Windows 8 (32 bit and 64 bit)
- Windows Server 2003 Service Pack 2 and newer (32 bit and 64 bit)
- Windows Server 2008 (32 bit and 64 bit)
- Windows Server 2008 R2 (64 bit only)

Report a bug

1.6. Additional Documentation

You can find additional documentation for the Red Hat Enterprise Linux OpenStack Platform in the customer portal:

https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform

The following documents are included in the documentation suite:

Table 1.12. Red Hat Enterprise Linux OpenStack Platform documentation suite

Guide	Description
Administration User Guide	HowTo procedures for administrating Red Hat Enterprise Linux OpenStack Platform environments.
Configuration Reference Guide	Configuration options and sample configuration files for each OpenStack component.
End User Guide	HowTo procedures for using Red Hat Enterprise Linux OpenStack Platform environments.
Getting Started Guide	Packstack deployment procedures for a Red Hat Enterprise Linux OpenStack Platform cloud, as well as brief HowTos for getting your cloud up and running.
Installation and Configuration Guide (this guide)	Deployment procedures for a Red Hat Enterprise Linux OpenStack Platform cloud; procedures for both a manual and Foreman installation are included. Also included are brief procedures for validating and monitoring the installation.
Release Notes	Information about the current release, including notes about technology previews, recommended practices, and known issues.

Report a bug

Chapter 2. Prerequisites

2.1. Software Requirements

2.1.1. Operating System Requirements

Red Hat Enterprise Linux OpenStack Platform 4 requires Red Hat Enterprise Linux 6.5 Server.

For further information on installing Red Hat Enterprise Linux 6.5 Server, refer to:

- 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”

For detailed information on installing Red Hat Enterprise Linux 6.5 Server, refer to the *Red Hat Enterprise Linux 6 Installation Guide* from the following link:

https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/

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”

Report a bug

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 documentation from the following link::

https://access.redhat.com/site/documentation/en-US/Red_Hat_Subscription_Management/

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.

Procedure 2.1. Registering a Red Hat Enterprise Linux system to Red Hat Network

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/products/enterprise-linux/openstack-platform/.

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.

Report a bug

2.1.2.2. Red Hat Enterprise Linux Repository Configuration

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.

Procedure 2.2. Registering a Red Hat Enterprise Linux system with Red Hat Network

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.

Report a bug

2.1.2.3. Red Hat Enterprise Linux OpenStack Platform Repository Configuration

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)

Note

Required and optional repository names for each version are listed in the *Red Hat Enterprise Linux OpenStack Platform Release Notes*. This document is available from the following page:

https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform

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

Procedure 2.3. Configuring a Red Hat Enterprise Linux system to receive OpenStack packages and updates from Red Hat Network

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:                02/14/2014
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:                02/14/2014
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. Use either the **`subscription-manager`** or **`yum-config-manager`** commands to enable or disable the appropriate software repositories (channels).

Note

Unless already installed, you can use the following to install yum-config-manager:

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

For example, to ensure that the repository for Red Hat Enterprise Linux OpenStack Platform 3 (Grizzly) has been disabled, run:

```
# yum-config-manager --disable 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/openstack/3/os
cache = 0
cachedir = /var/cache/yum/x86_64/6Server/rhel-server-ost-6-3-rpms
cost = 1000
enabled = False
...
```

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
```

4. 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 current list of enabled repositories will be displayed, along with the number of available packages. For example:

```
repo id                   repo name                                  status
rhel-6-server-rpms        Red Hat Enterprise Linux 6 Server (RPMs)    11,610+460
rhel-6-server-openstack-4.0-rpms   \
             Red Hat Enterprise Linux OpenStack Platform 4 (RPMs)  487+143
```

5. Use the **yum-config-manager** command to enable the Red Hat Enterprise Linux OpenStack Platform repository. Remember to use the repository name listed in the *Red Hat Enterprise Linux OpenStack Platform Release Notes*.

```
# yum-config-manager --enable REPO_NAME
```

6. 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
```

7. 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. Remember to use the repository name listed in the *Red Hat Enterprise Linux OpenStack Platform Release Notes*.

```
# yum-config-manager --enable REPO_NAME --setopt="REPO_NAME.priority=1"
```

For example:

```
# yum-config-manager --enable rhel-6-server-openstack-4.0-rpms \
        --setopt="rhel-6-server-openstack-4.0-rpms.priority=1"
Loaded plugins: product-id
==== repo: rhel-6-server-openstack-4.0-rpms ====
[rhel-6-server-openstack-4.0-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/4/os
cache = 0
cachedir = /var/cache/yum/x86_64/6Server/rhel-6-server-openstack-4.0-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

2.1.3. Reserved UIDs and GIDs

The reserved range for UIDs and GIDs in Red Hat Enterprise Linux is currently 0-500. Because your organization may need to change one or more of these, the following tables are provided for OpenStack and third-party components that it uses. If you are assigning UIDs and GIDS, it is best to start at a number higher than 1000 (higher than 5000 might be a good strategy).

Table 2.1. OpenStack Daemons

Component	Code	Reserved UID	Reserved GID
Identity	keystone	163	163
Block Storage	cinder	165	165
Compute	nova	162	162
Image	glance	161	161
Object Storage	swift	160	160
Telemetry	ceilometer	166	166
Orchestration	heat	187	187

Table 2.2. Third-party Components

Component	Reserved UID	Reserved GID
MongoDB	184	184
Memcached	497	496
MySQL	27	27
Nagios	496	495
Qpidd	498	499

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.

Report a bug

2.2.2. Network Node Requirements

Network nodes are responsible for hosting the services that provide 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.

Report a bug

2.2.3. Block Storage Node Requirements

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.

Report a bug

Part II. Deploying OpenStack with Foreman

Chapter 3. Foreman Overview and Installation

3.1. Introduction to Foreman OpenStack Manager

Foreman OpenStack Manager is a deployment management tool. It provides a web user interface for managing the installation and configuration of remote systems. Deployment of changes is performed using Puppet. Additionally, *Dynamic Host Configuration Protocol* (DHCP), *Domain Name System* (DNS), *Preboot Execution Environment* (PXE), and *Trivial File Transfer Protocol* (TFTP) services can be provided. Controlling these services also enables provisioning of physical systems that do not yet have an operating system installed.

Important

With the release of Red Hat Enterprise Linux OpenStack Platform 4, services deployed using Foreman OpenStack Manager can use either nova or neutron networking services.

This part documents the steps involved in installing, configuring and adding hosts to Foreman OpenStack Manager.

Completion of the steps detailed in this guide will result in the deployment of a Foreman OpenStack Manager server, an OpenStack controller node, and one or more OpenStack compute nodes.

Report a bug

3.2. Foreman Modes

Foreman can operate in different modes:

Provisioning Mode

In provisioning mode, Foreman will boot machines using PXE and apply all the puppet configuration in order to deploy the machine as a member of the defined host group. This mode can also import content from either RHN or Satellite.

Non-Provisioning Mode

Non-provisioning mode requires that you provision the machine and manually register it to the correct host group in Foreman, and then apply all the correct puppet configuration to the existing machine. Non-provisioning mode likewise can pull content from either RHN or Satellite depending on how the machine is initially provisioned.

Report a bug

3.3. Foreman Host Groups

Host Groups allow hosts with common configuration to be defined and grouped together. These host group definitions are created by default in Red Hat Enterprise Linux OpenStack Platform:

Controller (Nova Network)

This host group is intended for use on a single host that will act as a controller for the OpenStack deployment, using Nova Networking. Services that will be deployed to hosts added to this host group include:

- OpenStack Dashboard (Horizon).
- OpenStack Image service (Glance).
- OpenStack Identity service (Keystone).
- MySQL database server.
- Qpid message broker.

The OpenStack API and scheduling services, including those of the Compute service (Nova) and Block Storage service (Cinder), also run on the controller.

Compute (Nova Network)

This host group is intended for use on one or more hosts that will act as compute nodes for the OpenStack deployment. These are the systems that virtual machine instances will run on, while accessing the authentication, storage, and messaging infrastructure provided by the controller node. An instance of the Compute service (Nova) runs on each compute node.

Controller (Neutron)

This host group is intended for use on a single host that will act as a controller for the OpenStack deployment using Neutron Networking, however the Nova Compute service is still used. Services that will be deployed to hosts added to this host group include:

- OpenStack Dashboard (Horizon).
- OpenStack Image service (Glance).
- OpenStack Identity service (Keystone).
- MySQL database server.
- Qpid message broker.

The OpenStack API and scheduling services, including those of the Compute service (Nova) and Block Storage service (Cinder), also run on the controller.

Compute (Neutron)

This host group is the same as Compute (Nova Network) but Networking is handled by Neutron.

Neutron Networker

This host group is intended for use with Controller (Neutron) and Compute (Neutron). It provides various network services such as dhcp + L3 support.

LVM Block Storage

This host group is intended for use on one or more hosts that will act as block storage nodes for the OpenStack deployment. An instance of OpenStack Storage Service (Cinder) runs on each node, enabling block storage capacity to scale outwards.

HA MySQL

This host group provides a mechanism to set up a HA active/passive MySQL cluster which may be used as the database back end for the various service endpoints.

Report a bug

3.4. Configure the Firewall to Allow Foreman Traffic

Run the following **lokkit** commands as the **root** user to configure the firewall in preparation for the Foreman OpenStack Manager installation:

```
# lokkit --service http
# lokkit --service https
# lokkit --service dns
# lokkit --service tftp
# lokkit --port 8140:tcp
```

Note

The **lokkit** binary is provided by the *system-config-firewall-base* package in Red Hat Enterprise Linux.

Report a bug

3.5. Install Foreman Packages

The Foreman installer is included in the *openstack-foreman-installer* package. Additionally it is recommended that the *foreman-selinux* package is also installed, allowing SELinux to be run in enforcing mode on the Foreman server.

Important

The steps outlined here assume you already meet the hardware, software, and repository configuration requirements detailed in the Prerequisites chapter.

Procedure 3.1. Installing Foreman Packages

1. Log in to the system that will host the Foreman installation as the **root** user.
2. Install the *openstack-foreman-installer* and *foreman-selinux* packages.

```
# yum install -y openstack-foreman-installer foreman-selinux
```

The Foreman installer is now locally installed and ready to be configured and run.

Report a bug

3.6. Configure the Foreman Installer

Before running the Foreman installer, you must set a key environment variable that the installer uses during the installation process.

Procedure 3.2. Configuring the Foreman Installer

1. Log in to the system on which Foreman will be installed.
2. Export the following variable, replacing *address* with either the IP address or fully qualified domain name of the system on which Foreman will be installed:

```
# export FOREMAN_GATEWAY=address
```

3. Export the following variable, replacing *eth_name* with the name of the network interface that Foreman will use to define the provisioning network:

```
# export PROVISIONING_INTERFACE=eth_name
```

The Foreman installer is now configured and ready for use.

Important

Previously, the **`PRIVATE_CONTROLLER_IP`** variable in **`foreman_server.sh`** was used to populate parameters in the Foreman host group parameters user interface. This variable is no longer used, and you must manually update the following parameters:

- controller_priv_ip
- mysql_host
- qpid_host

These parameters are present in the following Host Groups:

- Controller (Neutron)
- Controller (Nova Network)
- Compute (Neutron)
- Compute (Nova Network)
- Neutron Networker
- LVM Block Storage

By default, the value of these parameters is set to **`172.16.0.1`**. You must replace this IP address by overriding each of the above variables in the above host groups with the IP address of the private network interface controller on the system you will use as the controller for your Red Hat Enterprise Linux OpenStack Platform deployment.

Important

Previously, the **`PUBLIC_CONTROLLER_IP`** variable in **`foreman_server.sh`** was used to populate a parameter in the Foreman host group parameters user interface. This variable is no longer used, and you must manually update the following parameter:

- controller_pub_ip

These parameter is present in the following Host Groups:

- Controller (Neutron)
- Controller (Nova Network)
- Compute (Neutron)
- Compute (Nova Network)

By default, the value of this parameter is set to **`172.16.1.1`**. You must replace this IP address by overriding the above variable in the above host groups with the IP address of the public network interface controller on the system you will use as the controller for your Red Hat Enterprise Linux OpenStack Platform deployment.

Report a bug

3.7. Run the Foreman Installer

The installer is a script provided by the *openstack-foreman-installer* package for deploying Foreman OpenStack Manager. It uses Puppet manifests to deploy Foreman OpenStack Manager on the local system.

Important

The Foreman installer **`foreman_server.sh`** expects the system to be configured with a fully qualified domain name (FQDN). The hostname can be verified with the command **`hostname -f`**. The expected output would be similar to this example FQDN: **`foreman.example.org`**

Otherwise, review the hostname settings in **`/etc/hosts`** and **`/etc/sysconfig/network`**. For further information on hostname configuration, see the *Red Hat Enterprise Linux Deployment Guide* from the following link:

https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/

Procedure 3.3. Running the Foreman Installer

1. Log in to the system that will host the Foreman installation as the **`root`** user.
2. Change to the **`/usr/share/openstack-foreman-installer/bin/`** directory. The installer *must* be run from this directory.

   ```
   # cd /usr/share/openstack-foreman-installer/bin/
   ```

3. Run the **`foreman_server.sh`** script.

   ```
   # sh foreman_server.sh
   ```

4. The installer deploys Foreman using Puppet manifests. This may take a significant amount of time.
5. A message is displayed once deployment of the Puppet manifests has been completed:

   ```
   Foreman is installed and almost ready for setting up your OpenStack
   First, you need to alter a few parameters in Foreman.
   Visit:
   https://FQDN/puppetclasses/quickstack::compute/edit
   https://FQDN/puppetclasses/quickstack::controller/edit
   Go to the Smart Class Parameters tab and work though each of the parameters
   in the left hand column

   Then copy /tmp/foreman_client.sh to your client nodes
   Run that script and visit the HOSTS tab in foreman. Pick CONTROLLER
   host group for your controller node and COMPUTE host group for the rest

   Once puppet runs on the machines, OpenStack is ready!
   ```

   In the actual output *FQDN* will have been replaced with the fully qualified domain name of the system to which Foreman was deployed.

Foreman has been installed successfully.

Report a bug

Chapter 4. Configuring Foreman

4.1. Change the Password

By default the Foreman installer creates an administration account with the user name **admin** and password **changeme**. It is highly recommended that users change this password immediately following installation.

Procedure 4.1. Changing the Password

1. Open a web browser either on the Foreman server itself or on a system with network access to the Foreman server.
2. Browse to **`https://`*`FQDN`*`/`**. Replace *FQDN* with the fully qualified domain name of your Foreman server.

   > **Example 4.1. Foreman URL**
   >
   > **`https://`*`foreman.example.com`*`/`**

3. The login screen is displayed. Type **admin** in the **Username** field and **changeme** in the **Password** field. Click the **Login** button to log in.

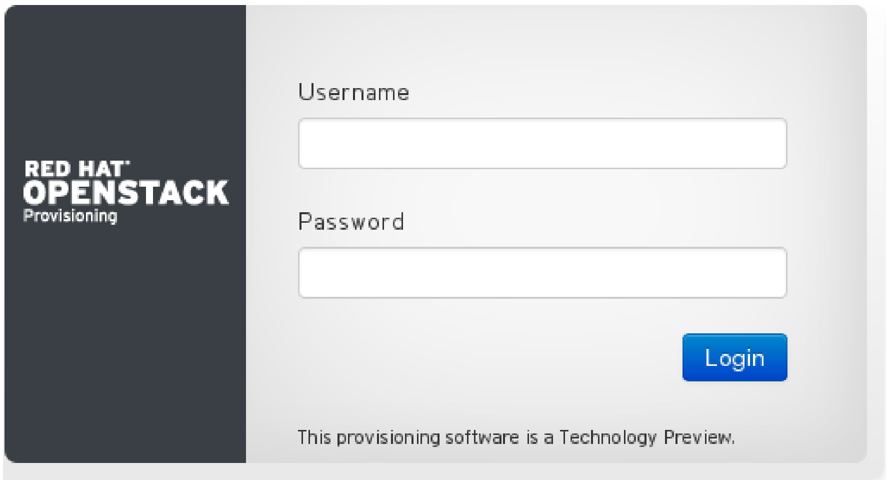


Figure 4.1. Foreman Login Screen

4. The **Overview** screen is displayed. Select the **Admin User** → **My account** option in the top right hand corner of the screen to access account settings.

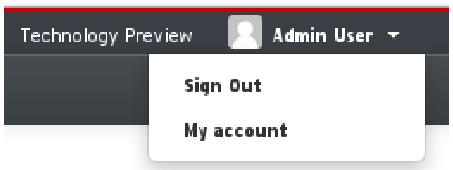


Figure 4.2. Accessing Account Settings

5. The **Edit User** screen is displayed. Enter a new password in the **Password** field.
6. Enter the new password again in the **Verified** field.
7. Click the **Submit** button to save the change.

The password of the Foreman **admin** user has been updated.

Report a bug

4.2. Configure Installation Media

To support provisioning of bare-metal hosts some additional configuration is required. In particular the Foreman installation media configuration must be updated to include a path to a local copy of the Red Hat Enterprise Linux installation media. This installation media will be used when provisioning bare-metal hosts.

The installation media must already exist and be accessible using HTTP on the Foreman network. For more information on preparing installation media for network installation see the *Red Hat Enterprise Linux Installation Guide* from the following link:

https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/

Procedure 4.2. Configuring Installation Media

1. Use a web browser on a system with network access to the Foreman server to open the Foreman web user interface.
2. Log in using the **`admin`** user and the password that was set in Section 4.1, “Change the Password”.
3. Click **More** → **Provisioning** → **Installation Media** in the top right hand corner of the page.

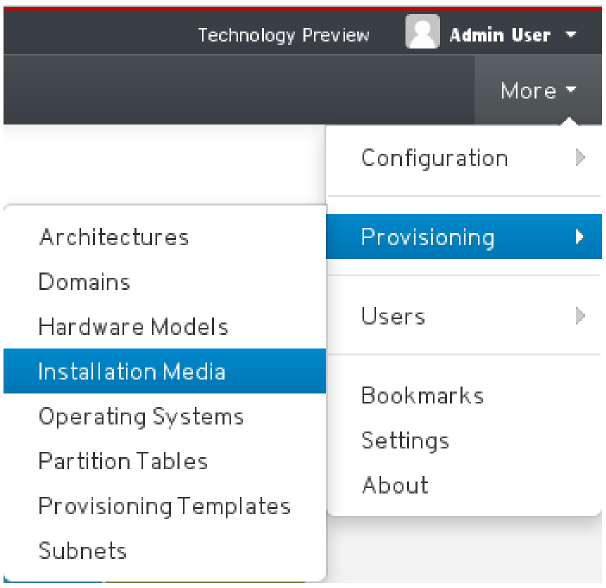
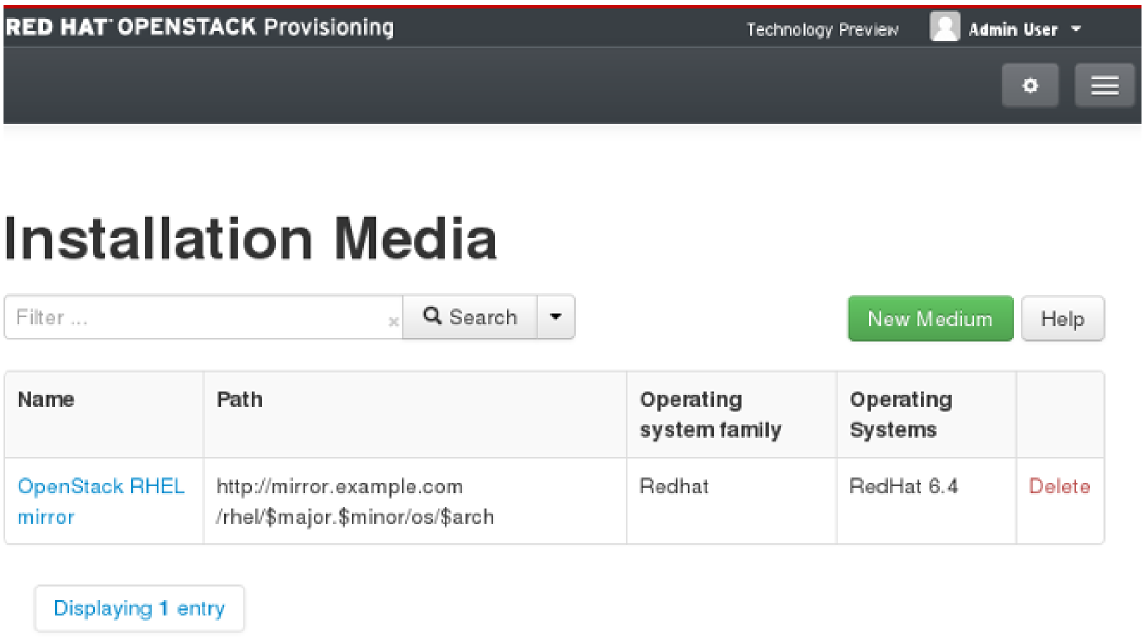


Figure 4.3. Provisioning Menu Items

4. The **`Installation Media`** page is displayed. An **`OpenStack RHEL mirror`** entry already exists by default but the associated path is only provided as an example and must be corrected.



Installation Media

Name	Path	Operating system family	Operating Systems	
OpenStack RHEL mirror	http://mirror.example.com/rhel/$major.$minor/os/$arch	Redhat	RedHat 6.4	Delete

Figure 4.4. `Installation Media` Page

5. Click the **`OpenStack RHEL mirror`** entry.

6. The **`Edit Medium`** page is displayed.

7. Update the **`Path`** field to contain the URL of a local installation mirror. These variables can be used in the URL and will be replaced automatically:

   `$arch`

   The system architecture, for example **`x86_64`**.

   `$version`

   The operating system version, for example **`6.5`**.

   `$major`

   The operating system major version, for example **`6`**.

   `$minor`

   The operating system minor version, for example **`4`**.

   Example 4.2. Path

   `http://download.example.com/rhel/$version/Server/$arch/os/`

8. Click **`Submit`** to save the updated **`Path`**.

The Foreman **`OpenStack RHEL mirror`** installation media configuration has been updated.

Report a bug

4.3. Edit Host Groups

Foreman allows users to override the parameters that will be used when adding new hosts to a host group. In particular the host groups contain parameters for manipulating the passwords that will be used for a variety of user and service accounts that will be created on hosts that Foreman manages.

Important

It is highly recommended that users edit the value of the **`admin_password`** configuration key in the OpenStack Controller host group. This configuration key determines the password to be used when logging into the OpenStack dashboard as the **`admin`** user.

Procedure 4.3. Editing Host Groups

1. Use a web browser on a system with network access to the Foreman server to open the Foreman web user interface.
2. Log in using the **`admin`** user and the associated password.
3. Click **More** → **Configuration** → **Host Groups**.
4. The **`Host Groups`** page is displayed. Click the name of the host group to edit.
5. The **`Edit`** page is displayed. Select the **`Parameters`** tab.
6. The list of parameters associated with the host group is displayed. Each parameter consists of a **`Name`** field and a **`Value`** field. It is possible to alter each of these fields. To create a new parameter, click **`+ Add Parameter`**.
7. Click **`Submit`** to save the updated parameter values.

The host group has been updated and the values of the selected parameters have been overridden.

Note

For more information on the available host group parameters see:

- Section 4.3.1, “Controller (Nova Network)”
- Section 4.3.2, “Compute (Nova Network)”
- Section 4.3.3, “Controller (Neutron)”
- Section 4.3.4, “Neutron Networker”
- Section 4.3.5, “Compute (Neutron)”

Report a bug

4.3.1. Controller (Nova Network)

The following is a list of parameters for the Controller (Nova Network) host group. For more details on each of the parameters in this host group, click the **`Additional info`** next to those parameters in the **`Parameters`** tab.

Table 4.1. Controller (Nova Network) Parameters

Parameter Name	Description
`admin_email`	The email address to associate with the OpenStack **`admin`** user when it is created using the Identity service. The default value is admin@*DOMAIN*.
`admin_password`	The password to associate with the **`admin`** Identity user when it is created. This is the password of the user that will be used to administer the cloud.
`cinder_backend_gluster`	Set to true if a **`gluster`** back end is used for **`cinder`**.
`cinder_backend_iscsi`	Set to true if using an LVM Block Storage node that runs the cinder-volume service and exports a local cinder-volume through tgtd.
`cinder_db_password`	The password to associate with the **`cinder`** database user, for use by the Block Storage service.

Parameter Name	Description
`cinder_gluster_servers`	An array of server IP addresses that represent the **`gluster`** storage nodes. Only applicable if **`cinder_backend_gluster`** is true.
`cinder_iscsi_iface`	The network interface used for the iscsi target IP address.
`cinder_user_password`	The password to associate with the **`cinder`** Identity service user, for use by the Block Storage service.
`glance_db_password`	The password to associate with the **`glance`** database user, for use by the Image Storage service.
`glance_user_password`	The password to associate with the **`glance`** Identity service user, for use by the Image Storage service.
`horizon_secret_key`	The unique secret key to be stored in the Dashboard configuration.
`keystone_admin_token`	The unique administrator token to be used by the Identity service. This token can be used by an administrator to access the Identity service when normal user authentication is not working or not yet configured.
`keystone_db_password`	The password to associate with the **`keystone`** database user, for use by the Identity service.
`mysql_root_password`	The password to associate with the **`root`** database user, for use when administering the database.
`nova_db_password`	The password to associate with the **nova** database user, for use by the Compute service.
`nova_user_password`	The password to associate with the **nova** Identity service user, for use by the Compute service.
`pacemaker_priv_floating_ip`	The private gateway IP for a Pacemaker cluster.
`pacemaker_pub_floating_ip`	The public gateway IP for a Pacemaker cluster.
`verbose`	Boolean value (true or false) indicating whether or not verbose logging information must be generated. The default value is **`true`**.

Note

If **`cinder_backend_gluster`** and **`cinder_backend_iscsi`** are both false, then the controller node (either nova or neutron) will run cinder-volume, locally backed by a cinder-volume group.

Report a bug

4.3.2. Compute (Nova Network)

The following is a list of parameters for the Compute (Nova Network) host group. For more details on each of the parameters in this host group, click the **`Additional info`** next to those parameters in the **`Parameters`** tab.

Table 4.2. Compute (Nova Network) Parameters

Parameter Name	Description
`fixed_network_range`	The range of IP addresses (in CIDR format; for example: **`192.168.100.0/24`**) to be used as fixed addresses.
`floating_network_range`	The range of IP addresses (in CIDR format; for example: **`192.168.100.0/24`**) to be used as floating IP addresses.
`nova_db_password`	The password associated with the **nova** database user. This password must match the value used for the same field in the controller host group.
`nova_user_password`	The password associated with the **nova** Identity service user. This password must match the value used for the same field in the controller host group.
`pacemaker_priv_floating_ip`	The private gateway IP address for a Pacemaker cluster.

Parameter Name	Description
`nova_network_private_iface`	The network interface to attach to the private OpenStack network. The default value is **`eth1`**.
`nova_network_public_iface`	The network interface to attach to the public OpenStack network. The default value is **`eth2`**.
`verbose`	Boolean value (true or false) indicating whether or not verbose logging information must be generated. The default value is **`true`**.

Report a bug

4.3.3. Controller (Neutron)

Table 4.3. Controller (Neutron) Parameters

Parameter Name	Description
`controller_priv_host`	The IP of the server on the management network.
`controller_pub_host`	The IP of the server on the public network.
`mysql_host`	The controller's **`controller_priv_host`** IP address. MySQL is needed by services on the management network.
`qpid_host`	The controller's **`controller_priv_host`** IP address. QPID is needed by services on the management network.
`tenant_network_type`	If the network type is GRE, all NICs on the private network must be addressed with IPv4.

Report a bug

4.3.4. Neutron Networker

Table 4.4. Neutron Networker Parameters

Parameter Name	Description
`configure_ovswitch`	Use Open vSwitch instead of Linux bridge.
`controller_priv_host`	The management IP for the controller.
`fixed_network_range`	IP range for tenant networks.
`floating_network_range`	IP range from which external IPs are sourced.
`mysql_host`	MySQL host's management IP.
`ovs_tunnel_iface`	Interface handling the private networks (tenants).
`qpid_host`	IP for QPID service on the management network.

Report a bug

4.3.5. Compute (Neutron)

Table 4.5. Compute (Neutron) Parameters

Parameter Name	Description
`controller_priv_host`	Management IP for the controller.
`controller_pub_host`	Public IP for the controller.
`fixed_network_range`	IP range for tenant networks.
`floating_network_range`	IP range from which external IPs are sourced.
`mysql_host`	MySQL host's management IP.
`ovs_tunnel_iface`	Interface handling the private networks (tenants).
`qpid_host`	IP for QPID service on the management network.
`tenant_network_type`	Network type in use (GRE or Vlan).

Report a bug

Chapter 5. Adding Hosts

To deploy OpenStack on hosts in a Foreman managed environment it is necessary to add the hosts to Foreman. Hosts are added by either adding the Puppet agent to existing servers or using the PXE boot provided by Foreman to provision new hosts.

Once the hosts have been added to Foreman they are assigned to one of the host groups defined in earlier procedures. The host group selected for each host determines which of the provided deployment templates will be applied to it. Once a host group is selected the associated templates are then applied to the host when the Puppet agent next connects to Foreman.

- To add existing Red Hat Enterprise Linux hosts to Foreman, see Section 5.1, “Register Existing Hosts in the Foreman User Interface”.
- To provision bare metal hosts and add them to Foreman, see Section 5.2, “Provision New Hosts”.

Once all required hosts are visible from the Foreman web user interface follow the steps outlined in Section 5.3, “Assign Client Hosts to Host Groups” to assign roles to the hosts and complete deployment.

Report a bug

5.1. Register Existing Hosts in the Foreman User Interface

The Foreman installer generates a script for registering new hosts. By default this script is saved to the **/tmp/foreman_client.sh** file on the Foreman server. The script must be copied to each new host to facilitate registration with the Foreman server. When run on a new host the script performs these actions:

- Installs the *augeas* and *ruby193-puppet* packages.
- Configures the Puppet agent to access the Foreman server.
- Starts the Puppet agent.

Once started the Puppet agent registers the host to Foreman, allowing the host to be managed from the Foreman user interface.

Important

For the host to be added successfully it *must* already be registered to receive packages from the Red Hat Enterprise Linux and Red Hat Enterprise Linux OpenStack Platform software repositories.

Procedure 5.1. Registering Existing Hosts in the Foreman User Interface

1. Log in to the Foreman server.
2. Copy the **/tmp/foreman_client.sh** file from the Foreman server to the new host:

   ```
   $ scp /tmp/foreman_client.sh USER@IP:DIR
   ```

   Replace *USER* with the user to use when logging in to the new host, replace *IP* with the IP address or fully qualified domain name of the new host, and replace *DIR* with the path to the directory in which the file must be stored on the remote machine. The directory must already exist.

   Example 5.1. Copying the Script

   ```
   $ scp /tmp/foreman_client.sh root@controller.example.com:~/
   ```

3. Log in to the new host as the **root** user.
4. Change into the directory to which the **foreman_client.sh** script was copied:

```
# cd DIR
```

Replace *DIR* with the path to the directory used when copying the file to the host.

5. Run the **foreman_client.sh** script:

```
# sh foreman_client.sh
```

6. When the script completes successfully the Puppet agent is started and this message is displayed:

```
Starting puppet agent:          [  OK  ]
```

7. Use a web browser on a system with network access to the Foreman server to open the Foreman web user interface.

8. Log in using the **admin** user and the password that was defined during the Foreman installation.

9. Click the **Hosts** tab.

10. Verify that the newly registered host is listed.

Note

Note that although the fully qualified domain name indicates the role that the new host will take (controller node) this is not actually determined until the host is assigned to the relevant host group.

Repeat this process for all existing hosts that will be included in the OpenStack deployment. See Section 5.3, “Assign Client Hosts to Host Groups” for information on adding the newly registered hosts to a host group and deploying OpenStack to them.

Note

You may encounter the following warning output when you run **foreman_client.sh**:

```
Warning: Unable to fetch my node definition, but the agent run will continue:
Warning: Error 400 on SERVER
```

This can be safely ignored, as it means this host was unknown to Foreman at the start of the Puppet run.

Report a bug

5.2. Provision New Hosts

Foreman provisions new physical hosts by using networking infrastructure it controls including DHCP, DNS, PXE, and TFTP services. New physical hosts are added to Foreman by providing the MAC address associated with the network interface of the system that is connected to the Foreman network and some other configuration details. The new host will be provisioned using this configuration when it next starts.

To provision a new host an activation key will be required. An activation key facilitates non-interactive registration of systems to Red Hat Network or an equivalent service provided by a Red Hat Network Satellite server. To learn how to create an activation key see https://access.redhat.com/site/solutions/2474. The activation key must be created with access to both the Red Hat Enterprise Linux and Red Hat OpenStack software repositories.

Procedure 5.2. Provisioning New Hosts

1. Use a web browser on a system with network access to the Foreman server to open the Foreman web user interface.

2. Log in using the **admin** user and associated password.

3. Click the **Hosts** tab.

4. Click the **New Host** button.

5. Enter the desired fully qualified domain name for the new host in the **Name** field.

6. Do not select a **Host Group** at this time.

7. Click the **Network** tab. The **Network** tab contains settings that define the networking configuration for the new host.

   a. Enter the MAC address of the network interface on the physical machine that is connected to the Foreman network in the **MAC address** field.

      Example 5.2. MAC Address

      In this example the **ip link** command is used to identify the MAC address of the **eth0** network interface.

      ```
      # ip link show eth0
      2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen
      1000
          link/ether 00:1a:4a:0f:18:bb brd ff:ff:ff:ff:ff:ff
      ```

      The MAC address of the **eth0** network device is *00:1a:4a:0f:18:bb*.

   b. Ensure that the value in the **Domain** field matches the domain that Foreman is expected to manage.

      Example 5.3. Domain

      example.com

   c. Ensure that a subnet is selected in the **Subnet** field. The **IP address** field will be automatically selected based on the subnet selection.

      Example 5.4. Subnet

      OpenStack (192.0.43.0/24)

8. Click the **Operating System** tab. The **Operating System** tab contains options for configuring the operating system installation for the host.

   Note

   OpenStack Block Storage hosts require an initial **cinder-volumes** LVM volume group, unless a Cinder driver for an external storage array is used. This LVM volume group must be backed by at least one physical volume. LVM layout can be customized using the **Custom partition table** field. For more information on Kickstart, see *Kickstart Installations* in the *Red Hat Enterprise Linux Installation Guide*. This guide is available from the following link:

   https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/

   a. Select **x86_64** in the **Architecture** field.

   b. Select **RedHat 6.5** in the **Operating System** field.

   c. Enter a password for the **root** user in the **Root password** field.

Warning

It is highly recommended that a secure **root** password is provided. Strong passwords contain a mix of uppercase, lowercase, numeric and punctuation characters. They are six or more characters long and do not contain dictionary words. Failure to enter a secure **root** password in this field will result in the default **root** password, **123123**, being used.

d. Click the **Resolve** button. Two templates will appear:

- **OpenStack PXE Template**
- **OpenStack Kickstart Template**

9. Click the **Parameters** tab. The **Parameters** tab contains settings that will be used to register the new host to Red Hat Network or a Red Hat Network Satellite server.

A. To configure the host to register to Red Hat Network:

a. Click the **Override** button next to the **satellite_type** configuration key. Enter **hosted** in the **Value** field.

b. Click the **Override** button next to the **satellite_host** configuration key. Enter **xmlrpc.rhn.redhat.com** in the associated **Value** field.

c. Click the **Override** button next to the **activation_key** configuration key. Enter the Red Hat Network activation key to be used when registering the host in the associated **Value** field.

B. To configure the host to register to a Red Hat Network Satellite server:

a. Ensure the **satellite_type** is set to the default value, **site**.

b. Click the **Override** button next to the **satellite_host** configuration key. Enter the IP address or fully qualified domain name of the Red Hat Network Satellite server in the associated **Value** field.

c. Click the **Override** button next to the **activation_key** configuration key. Enter the Red Hat Network activation key to be used when registering the host in the associated **Value** field.

Important

If the system is intended to be provisioned using packages from a different location instead of Red Hat Network or a Red Hat Network Satellite server then the **OpenStack Kickstart Template** must be edited. Access the template editor by clicking **More** → **Provisioning** → **Provisioning Templates**, selecting the **OpenStack Kickstart Template** entry, and removing this line from the template:

```
<%= snippets "redhat_register" %>
```

The line must be replaced with environment specific commands suitable for registering the system and adding equivalent software repositories.

10. Click the **Submit** button. Foreman will prepare to add the hosts the next time they boot.

11. Ensure network boot (PXE) is enabled on the new host. Instructions for confirming this will be in the manufacturer instructions for the specific system.

12. Restart the host. The host will retrieve an installation image from the Foreman PXE/TFTP server and begin the installation process.

13. The host will restart again once installation has completed. Once this occurs use the Foreman web user interface to verify that the new host appears by clicking the **Hosts** tab.

Repeat this process for all new hosts that will be added to the OpenStack deployment. See Section 5.3, “Assign Client Hosts to Host Groups” for information on adding the newly added hosts to a host group and deploying OpenStack to them.

Important

Once a host has been booted successfully over the network the Foreman server updates the PXE configuration directory (**`/var/lib/tftpboot/pxelinux.cfg/`**) to ensure that future boots are performed using local boot devices. This ensures that the host does not re-provision itself when restarted. To re-provision a system that has already been registered to Foreman either:

- Use the web user interface to delete and re-add the host; or
- Use the web user interface to view the details of the host and click the **`Build`** button.

In either case use the **`Resolve Templates`** button on the resultant page to ensure the correct PXE and Kickstart templates are used. Additionally use this command on the Foreman server while logged in as the **`root`** user to allow the host to register with a new certificate:

```
# scl enable ruby193 'puppet cert clean HOST'
```

Replace *HOST* with the fully qualified domain name of the host.

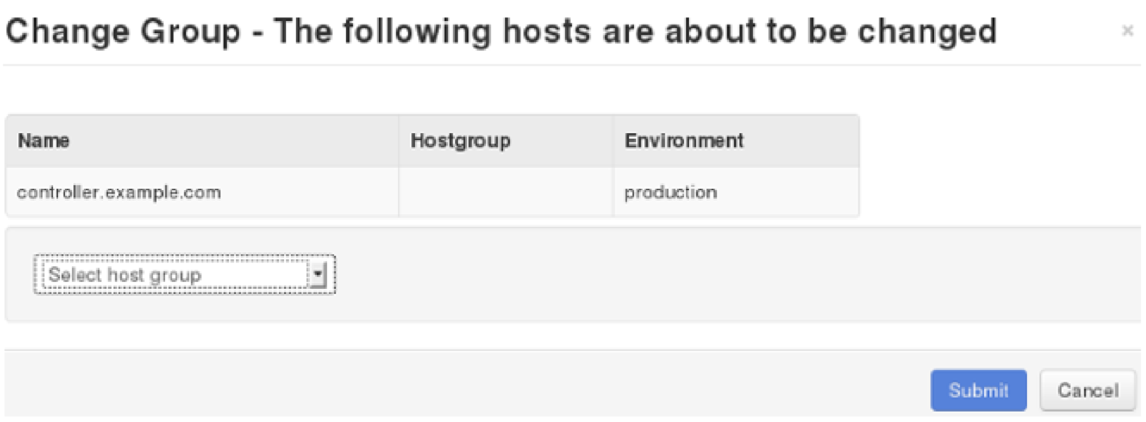
Report a bug

5.3. Assign Client Hosts to Host Groups

Once client hosts have been registered with Foreman they must be assigned to host groups to allow installation and configuration of OpenStack. Which host group a host is assigned to defines what role it performs in the environment and which packages will be deployed.

Procedure 5.3. Assigning Client Hosts to Host Groups

1. Use a web browser on a system with network access to the Foreman server to open the Foreman web user interface.
2. Log in using the **`admin`** user and associated password.
3. Click the **`Hosts`** tab.
4. Select the host from the list displayed in the table.
5. Click the **Select Action** → **Change Group** option.
6. The **`Change Group`** window is displayed.



Change Group - The following hosts are about to be changed

Name	Hostgroup	Environment
controller.example.com		production

Figure 5.1. The `Change Group` Window

7. Use the **`Select host group`** field to select a host group. The available options are:
   - **`Controller (Nova Network)`**

- **`Compute (Nova Network)`**
- **`Controller (Neutron)`**
- **`Compute (Neutron)`**
- **`Neutron Networker`**
- **`LVM Block Storage`**

Important

A controller node *must* be provisioned before attempting to provision a compute node. For instance, if you are using Nova Networking, you must provision Controller (Nova Network) before Compute (Nova Network). Puppet *must* have completed installation and configuration of the controller node before compute nodes are deployed.

8. Click the **`Submit`** button to save the host group selection.

The host group configuration of the host has been updated. The Puppet agent installed on the host will connect to Foreman every 30 minutes by default. When the Puppet agent next connects, the host group change will be detected and the associated configuration changes applied.

Once a controller has been provisioned using the **`OpenStack Controller`** host group, repeat the process to provision each compute node, selecting the **`OpenStack Nova Compute`** host group.

Note

To force the Puppet agent to connect to Foreman immediately, log in to the host as the **`root`** user and run this command:

```
# scl enable ruby193 "puppet agent --test"
```

Report a bug

Chapter 6. Logging In

Once at least one controller node and one compute node are successfully installed and configured, access the user interface with a web browser. Replace *HOSTNAME* with the host name or IP address of the server acting as the controller node:

- HTTPS

```
https://HOSTNAME/dashboard/
```

- HTTP

```
http://HOSTNAME/dashboard/
```

When prompted, log in using the credentials of the **admin** user. This is the password that was set for the **admin_password** configuration key of the controller node (Section 4.3.1, “Controller (Nova Network)”). To begin using the OpenStack deployment refer to *Using OpenStack With the Dashboard* in the *Red Hat Enterprise Linux OpenStack Platform Getting Started Guide*. This guide is available from the following link:

https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/

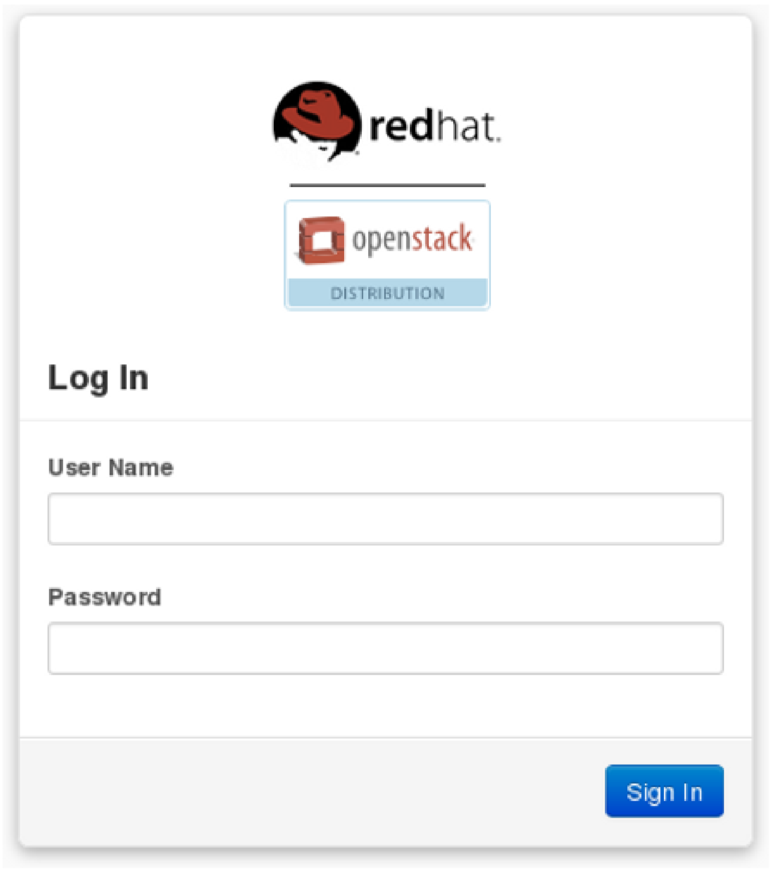


Figure 6.1. Dashboard Login Screen

Report a bug

Part III. Installing OpenStack Manually

Chapter 7. Installing the Database Server

7.1. Install the MySQL Database 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.

To install the required packages, log in as the **root** user and run:

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

The database server is installed and ready to be configured.

Report a bug

7.2. Configure the Firewall to Allow Database Traffic

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.

Procedure 7.1. Configuring the firewall to allow database traffic

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

7.3. Start 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.

Procedure 7.2. Starting the database service

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

7.4. Set 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.

Procedure 7.3. Setting a MySQL database administrator password

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 **`root`** password is now set. This password will be required when logging in to create databases and database users.

Report a bug

Chapter 8. Installing the Message Broker

8.1. Install the Message Broker Packages

This procedure installs 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 layer for AMQP traffic. Using SSL to support secure configuration of Qpid is optional, but if you do want to use SSL you must install this package.

To install the required packages, log in as the **`root`** user and run:

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

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

Report a bug

8.2. Configuring the Message Broker

8.2.1. Basic Qpid Configuration

Qpid configuration is contained in the **`/etc/qpidd.conf`** file. For small deployments, its default configuration can be used. However, for most OpenStack deployments, the **`max-connections`** value should be increased to a large number (for example, 65535).

Report a bug

8.2.2. Simple Authentication and Security Layer - SASL

8.2.2.1. Simple Authentication and Security Layer (SASL)

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.

Report a bug

8.2.2.2. SASL Mechanisms

The SASL authentication mechanisms allowed by the broker are controlled by the file **`/etc/sasl2/qpidd.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, use Security Services Library (SSL).

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.

Report a bug

8.2.2.3. SASL Mechanisms and Packages

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 8.1.

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

Report a bug

8.2.2.4. Configure SASL Using a Local Password File

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

Procedure 8.1. Configuring 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.sasldb** 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.sasldb -u QPID USERNAME
   ```

   Replace *USERNAME* with the name you wish to assign to a user (namely, the value of the user's **qpid_username** configuration key).

2. Create new Qpid users for the Block Storage, Networking, and Compute services. These services will use their respective Qpid user accounts to connect to the Message Broker:

   ```
   # saslpasswd2 -f /var/lib/qpidd/qpidd.sasldb -u QPID cinder
   ```

   ```
   # saslpasswd2 -f /var/lib/qpidd/qpidd.sasldb -u QPID neutron
   ```

   ```
   # saslpasswd2 -f /var/lib/qpidd/qpidd.sasldb -u QPID nova
   ```

3. Existing user accounts can be listed by using the **-f** option:

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

Note

The user database at **/var/lib/qpidd/qpidd.sasldb** 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.

4. 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.

Report a bug

8.2.3. Configuring TLS/SSL

8.2.3.1. Encryption Using SSL

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

Encryption using SSL is not mandatory, for example encryption is not required for development environments. However, encryption is recommended in production environments.

Report a bug

8.2.3.2. Enabling SSL on the Broker

This section describes two types of certificates that can be generated: self-signed, or issued by a certification authority. Self-signed certificates may not be acceptable to all SSL clients, hence they are recommended for development but not for production. If you use 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.so** 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 as explained in the step *Start the broker* below.

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 is generally 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.

Procedure 8.2. Creating a database, adding a certificate and starting the broker

1. This procedure shows how to create a certificate database using **certutil**. Three steps are required, creating the database, adding the certificate, (either self-signed or from a certification authority), and starting the broker:

   Create the database: Use the following script to create the database

   ```
   CERT_DIR=/etc/pki/qpid
   CERT_PW_FILE=${CERT_DIR}/certpw
   mkdir ${CERT_DIR}
   echo your_password_here > ${CERT_PW_FILE}
   ```

```
chmod 700 ${CERT_DIR}
chmod 600 ${CERT_PW_FILE}
certutil -N -d ${CERT_DIR} -f ${CERT_PW_FILE}
chown -R ${QPID_USER} /etc/pki/qpid
```

Note that:

- **`/etc/pki/qpid`** is the recommended directory to store certificates and keys,
- you must replace "your_password_here" with your password,
- you must replace ${qpid_user} with the uid of the qpid process.

2. **`Add the certificate`**: Choose one of the two following options. Note that self-signed certificates may not be acceptable to all SSL clients, hence they are recommended for development but not for production:

   a. **`Self-signed certificate`** - run the following command to add a self-signed certificate:

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

   b. **`CA signed certificate`** - additional commands are required to generate a certificate signing request (CSR) for a CA, and then to install the resulting certificate.

      a. To generate a request to get a certificate signed, use:

      ```
      # certutil -R -d ${CERT_DIR} -s "CN=${HOSTNAME}" \ -a -f ${CERT_PW_FILE} >
      ${HOSTNAME}.csr
      ```

      Note that the subject (-s) may need to include additional values depending on the CA, for example ou=Engineering, o=Example Corp, c=US.

      b. Provide the resulting file to your CA for signing.

      You will get back a certificate, probably in PEM format.

      c. Run the following commands to add the CA-signed certificate:

      ```
      # certutil -A -d ${CERT_DIR} -n ${HOSTNAME} -f ${CERT_PW_FILE} \ -t u,u,u -a -i
      /path/to/server.crt
       # certutil -A -d ${CERT_DIR} -n "Your CA certificate" \ -f ${CERT_PW_FILE} -t
      CT,C,C -a -i /path/to/ca.crt
      ```

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

   You can set up the required parameters in the **`/etc/qpidd.conf`** file, similar to this example:

   ```
   ssl-cert-db=/etc/pki/qpid/
   ssl-cert-password-file=/etc/pki/qpid/certpw
   ssl-cert-name=your_hostname
   ```

   Note that you must replace "your_hostname" with the host name of the machine that qpid is running on.

A Python script that can be used for testing if certificates are set up correctly is included in an appendix. See Section C.2, “Testing that SSL certificates are set up correctly”.

The following is a summary of SSL options that can be used when starting the broker. These options can either be passed as command-line arguments when starting the server, or placed in the **`/etc/qpidd.conf`** file (without the initial --):

`--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 (as in, 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 (as in, 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.

Report a bug

8.2.3.3. Exporting an SSL Certificate for Clients

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

8.3. 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`**. You can configure the firewall by editing the **`iptables`** configuration file, namely **`/etc/sysconfig/iptables`**. To do so:

1. Open the **`/etc/sysconfig/iptables`** file in a text editor.

2. Add an **`INPUT`** rule allowing incoming connections on port **`5672`** to the file. The new rule must appear before any **`INPUT`**

rules that **REJECT** traffic.

```
-A INPUT -p tcp -m tcp --dport 5672  -j ACCEPT
```

3. Save the changes to the **/etc/sysconfig/iptables** file.

4. Restart the **iptables** service for the firewall changes to take effect.

```
# service iptables restart
```

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

```
# service iptables status
```

Report a bug

8.4. Start the Messaging Server

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

Procedure 8.3. Launching the qpidd service

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.

Report a bug

Chapter 9. Installing the OpenStack Identity Service

9.1. 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.

Report a bug

9.2. Install the Identity Packages

The packages that provide the components of the Identity service 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.

To install these packages, log in as the **root** user and run:

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

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

Report a bug

9.3. Create 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**).

Procedure 9.1. Creating the Identity Service database

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.

```
mysql> quit
```

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

Report a bug

9.4. Configuring the Service

9.4.1. Set the Identity Service 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 as the **root** user into the server that will host the Identity service.

Procedure 9.2. Setting the Identity Service administrative token

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.

Note

The Identity server's token database table grows unconditionally over time as new tokens are generated. To clear the token table, the administrator must run the **keystone-manage token_flush** command to flush the tokens. Flushing tokens simply deletes expired tokens, eliminating any means of traceability. It is recommended that this command be run approximately once per minute.

```
# keystone-manage token_flush
```

Report a bug

9.4.2. Configure the Identity Service Database Connection

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.

Procedure 9.3. Configuring the Identity Service SQL database connection

- 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.

Report a bug

9.4.3. Configuring the Public Key Infrastructure

9.4.3.1. 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. These keys 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**.

Report a bug

9.4.3.2. Create the Public Key Infrastructure Files

This section explains how to create and configure the PKI files to be used 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.

Procedure 9.4. Creating the PKI files to be used by the Identity service

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 files have been created and will be used when generating and signing tokens.

Report a bug

9.4.3.3. Configure the Identity Service to Use Public Key Infrastructure Files

After generating the PKI files for use by the Identity service, you will need to enable the Identity service to use them.

Set the values of the attributes in the **/etc/keystone/keystone.conf** file by using the following commands:

```
# openstack-config --set /etc/keystone/keystone.conf \ signing token_format PKI
# openstack-config --set /etc/keystone/keystone.conf \ signing certfile
/etc/keystone/ssl/certs/signing_cert.pem
# openstack-config --set /etc/keystone/keystone.conf \ signing keyfile
/etc/keystone/ssl/private/signing_key.pem
# openstack-config --set /etc/keystone/keystone.conf \ signing ca_certs
/etc/keystone/ssl/certs/ca.pem
# openstack-config --set /etc/keystone/keystone.conf \ signing key_size 1024
# openstack-config --set /etc/keystone/keystone.conf \ signing valid_days 3650
# openstack-config --set /etc/keystone/keystone.conf \ signing ca_password None
```

You can also update these values directly by editing the **/usr/share/keystone/keystone-dist.conf** file.

Report a bug

9.4.4. Integrate Identity Service with LDAP

The Identity service supports integration with an existing LDAP directory for authentication and authorization services.

Important

For the Identity service to access an LDAP backend, SELinux requires that the **`authlogin_nsswitch_use_ldap`** boolean setting be enabled on the Identity server. Run the following command on the Identity server as **`root`** to enable the boolean option and make it persistent across reboots:

```
# setsebool -P authlogin_nsswitch_use_ldap
```

Note

The Identity service supports integration with a single LDAP server.

Configuration options for the Identity service are set in: **`/etc/keystone/keystone.conf`**. These samples use the example Distinguished Name **`dc=example,dc=org`** and can be adjusted appropriately.

Procedure 9.5. Sample procedure for integrating Identity with LDAP

1. Enabling the LDAP driver:

```
[identity]
#driver = keystone.identity.backends.sql.Identity
driver = keystone.identity.backends.ldap.Identity
```

2. Defining the destination LDAP server:

```
[ldap]
url = ldap://localhost
user = dc=Manager,dc=example,dc=org
password = samplepassword
suffix = dc=example,dc=org
use_dumb_member = False
allow_subtree_delete = False
```

3. Create the Organizational Units in the LDAP directory, and define their corresponding location in **`keystone.conf`**:

```
[ldap]
user_tree_dn = ou=Users,dc=example,dc=org
user_objectclass = inetOrgPerson

tenant_tree_dn = ou=Groups,dc=example,dc=org
tenant_objectclass = groupOfNames

role_tree_dn = ou=Roles,dc=example,dc=org
role_objectclass = organizationalRole
```

Note

These schema attributes are extensible for compatibility with various schemas. For example, this entry would map to the **`person`** attribute of Active Directory:

```
user_objectclass = person
```

4. A read-only implementation is recommended for LDAP integration. These permissions are applied to object types in the **`keystone.conf`** file:

```
[ldap]
user_allow_create = False
```

```
user_allow_update = False
user_allow_delete = False

tenant_allow_create = False
tenant_allow_update = False
tenant_allow_delete = False

role_allow_create = False
role_allow_update = False
role_allow_delete = False
```

5. Restart the Identity service. Authentication and Authorization will be unavailable for the duration of the service restart.

```
# service keystone restart
```

Additional settings for LDAP integration

These options are configurable in **keystone.conf**

Filters

Filters are available to control the scope of data presented through LDAP.

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

LDAP Account Status

Account status values can be masked for compatibility with various directory services. Superfluous accounts can also be filtered using **user_filter**.

For example, masking account status for Active Directory:

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

Note

Information on integrating RDO Keystone with Red Hat IdM is available here: http://openstack.redhat.com/Keystone_integration_with_IDM

Report a bug

9.4.5. Harden LDAP Integration

The Identity service supports features to assist with hardening LDAP network traffic using LDAPS and TLS. Management of authentication and authorization can be split using the Assignments feature.

Assignment

The Assignment feature enables a combination of LDAP and SQL for Identity service authentication and authorization. Assignment enables administrators to manage project and role authorization within the Identity service's own SQL database, while still providing user authentication using an LDAP directory.

The Assignment driver is enabled in **keystone.conf** alongside the LDAP driver:

```
[identity]
driver = keystone.identity.backends.ldap.Identity

[assignment]
driver = keystone.assignment.backends.sql.Assignment
```

LDAP Traffic Encryption

Identity service supports encryption of network traffic using LDAPS and TLS.

Configuring TLS Encryption

Procedure 9.6. Hardening LDAP integration

1. Specify the path to the certificate authorities file or folder in **keystone.conf** :

   ```
   tls_cacertfile = /etc/keystone/ssl/certs/cacert.pem
   ```

   or:

   ```
   tls_cacertdir = /etc/keystone/ssl/certs/
   ```

   Note

   The **tls_cacertfile** option takes precedence over **tls_cacertdir.**

2. Define the certificate requirement behaviour. Valid options are **demand**, **never**, and **allow**.

   ```
   tls_req_cert = demand
   ```

Example TLS configuration for **keystone.conf**:

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

Report a bug

9.4.6. Configure the Firewall to Allow Identity Service Traffic

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.

Procedure 9.7. Configuring the firewall to allow Identity Service traffic

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**.

Report a bug

9.4.7. Populate the Identity Service Database

You can populate the Identity service database after you have successfully configured the Identity service database connection string (refer to Section 9.4.2, “Configure the Identity Service Database Connection”).

Procedure 9.8. Populating the Identity Service database

1. Log in to the system hosting the Identity service.
2. 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

9.5. Start 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.

Procedure 9.9. Launching the Identity Service

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

9.6. Create 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.

Procedure 9.10. Creating the Identity Service Endpoint

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             |
+-------------+----------------------------------+
```

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 keystone \
        --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 *IP* with the IP address or host name of the 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

9.7. Create 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.

Procedure 9.11. Creating an Administrator account

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:

```
# export 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.

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

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

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               |
   +-------------+----------------------------------+
   ```

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 admin --role admin --tenant admin
   ```

7. The newly created **admin** account will be used for future management of the Identity service. To facilitate authentication, create a **keystonerc_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

9.8. Create a Regular User Account

Executing the following procedure will result in the creation of an 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.

Procedure 9.12. Creating a regular user account

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.

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              |
   +----------+----------------------------------+
   ```

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.

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 USER --role Member --tenant TENANT
   ```

where:

- *USER* is the same user name specified earlier during user creation.
- *TENANT* is the same tenant name specified earlier during tenant creation.

6. To facilitate authentication, create a **keystonerc_user** file in a secure location (for example, 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)]\$ '
   ```

   where:

   - *PASSWORD* is the same password specified earlier during user creation.
   - *IP* is 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

9.9. Create 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.

Note

For more information about quotas, refer to the *View and manage quotas* section in the *Red Hat Enterprise Linux OpenStack Platform Administration User Guide*. This document is available from the following page:

https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform

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, then 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, then only one service tenant is required (but of course this is just one option; more can be created for administrative purposes).

The service setup examples in the *Installation and Configuration Guide* assume that all services are deployed on one node, therefore only one service tenant is required. All such 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 Section 9.7, “Create an Administrator Account” and Section 9.8, “Create a Regular User Account”.

To create the **services** tenant:

Procedure 9.13. Creatinig 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

9.10. Configuring the Identity Service to Use External Authentication

You can configure the OpenStack Identity service to use external authentication methods. Doing so can enhance the security of Identity authentication, especially when compared to simple username/password authentication provided by the Identity store database.

Currently, the only supported method for implementing external authentication for the Identity service is by running the Identity service in HTTPD. Doing so allows the **httpd** service to perform all authentication tasks required by the Identity service.

With this setup, user authentication occurs as follows:

1. The **httpd** service checks a user's credentials using an external authentication service (such as X509 or Kerberos).
2. After verifying the user's credentials, **httpd** sets the user's **REMOTE_USER** environment variable accordingly.
3. Whenever required, the Identity service checks this **REMOTE_USER** environment variable to determine whether the **httpd** service authenticated the user.

Note

When configuring the OpenStack Identity service to run in HTTPD, it is recommended that you configure HTTPD to use SSL.

Report a bug

9.10.1. Configure HTTPD to use SSL

When configuring the Identity service to use **httpd**, you are not required to configure **httpd** to use SSL. However, it is highly recommended that you do so for enhanced security.

Procedure 9.14. Configuring HTTPD to use SSL

1. As the **root** user, install the *mod_nss* package:

   ```
   # yum install -y mod_nss
   ```

2. Configure the **Listen** directive to use port 443. Replace the following line:

   ```
   Listen 8443
   ```

   With:

   ```
   Listen 443
   ```

3. Set the Virtual Host to use the same port. Replace the following line:

   ```
   <virtualhost _default_:8443="">
   ```

   With:

   ```
   <virtualhost _default_:443="">
   ```

4. Configure the firewall to allow SSL traffic to pass through.

   a. Open the **/etc/sysconfig/iptables** file in a text editor.

   b. Add an INPUT rule allowing TCP traffic on port **443** by adding this lines to the file.

      ```
      -A INPUT -m state --state NEW -m tcp -p tcp --dport 443 -j ACCEPT
      ```

      This new rule must appear before any INPUT rules that REJECT traffic.

   c. Save the changes to the **/etc/sysconfig/iptables** file.

   d. Restart the **iptables** service to ensure that the change takes effect.

      ```
      # service iptables restart
      ```

5. Restart **httpd**.

   ```
   # service httpd restart
   ```

Report a bug

9.10.2. Configure the Identity Service to Run in HTTPD

Prerequisites:

- Section 9.4.1, “Set the Identity Service Administration Token”
- Section 9.6, “Create the Identity Service Endpoint”

This section assumes that you have already set the admninistration token and created the Identity service endpoint. For more information, refer to the following sections:

- Section 9.4.1, “Set the Identity Service Administration Token”
- Section 9.6, “Create the Identity Service Endpoint”

Configuring the OpenStack Identity service to run in **httpd** involves the following major steps:

- Configuring the appropriate file links
- Setting the correct SELinux context (optional, but recommended)

- Configuring Keystone

Procedure 9.15. Configuring the appropriate file links

1. Create a symbolic link to **/usr/share/keystone/wsgi-keystone.conf** from the configuration directory of the **httpd** server:

   ```
   # ln -s /usr/share/keystone/wsgi-keystone.conf /etc/httpd/conf.d/
   ```

2. Create the directory **/var/www/cgi-bin/keystone/**:

   ```
   # mkdir /var/www/cgi-bin/keystone/
   ```

3. Inside **/var/www/cgi-bin/keystone/**, create links to **/usr/share/keystone/keystone.wsgi** from the files **main** and **admin**:

   ```
   # ln /usr/share/keystone/keystone.wsgi /var/www/cgi-bin/keystone/main
   # ln /usr/share/keystone/keystone.wsgi /var/www/cgi-bin/keystone/admin
   ```

If SELinux is enabled (optional, but recommended), you will also need to set the appropriate SELinux context for **/var/www/cgi-bin/**. To do so, run the following command as **root**:

```
# restorecon /var/www/cgi-bin
```

Once the file links and SELinux (when applicable) are configured, you can now configure the Identity service to run in **httpd**. Before doing so, you will need to ensure that the **httpd** service can access the **/etc/keystone/keystone.conf** file. If this is not the case, rectify this by changing the group name and file mode of **/etc/keystone/keystone.conf**:

```
# chmod 640 /etc/keystone/keystone.conf
# chown root:apache /etc/keystone/keystone.conf
```

Procedure 9.16. Configuring the Identity Service Default Settings

Note

You should already have an Administration Token configured at this point.

1. Set the port numbers on which the public service and admin service should listen:

   ```
   # openstack-config -set /etc/keystone/keystone.conf \
   DEFAULT public_port 443
   # openstack-config -set /etc/keystone/keystone.conf \
   DEFAULT admin_port 443
   ```

   Note

   The ports specified in this example assumes that SSL is enabled, and as such the **httpd** Virtual Host is configured to listen on port **443**.

2. Set the base endpoint URLs that the Identity service uses to advertise to clients:

   ```
   # keystone endpoint-create \
    --publicurl 'http://localhost:%(public_port)s/keystone/main'\
    --adminurl 'http://localhost:%(admin_port)s/keystone/admin'
   ```

Report a bug

9.10.3. Implement External Authentication for the Identity Service Using HTTPD

Once you configure the Identity service to use SSL (optional) and run in HTTPD (required), you can then configure **httpd** use external authentication. Doing so effectively configures HTTPD to perform all external authentication tasks for the Identity service.

At present, the only external authentication methods officially supported on the OpenStack Identity service (using **httpd**) are:

- X509
- Kerberos

To configure HTTPD (and, by extension, the Identity service) to use external authentication, use the HTTPD configuration file. In most Red Hat Enterprise Linux versions, this file is **/etc/httpd/conf/httpd.conf**. Refer to the corresponding documentation of your HTTPD service for more information.

By default, the Identity service is already configured to use external authentication as well. To verify this, run:

```
# openstack-config --get /etc/keystone/keystone.conf \
 auth methods
```

This command retrieves a comma-delimited list of enabled authentication methods. If **external** is included in the list, then external authentication is enabled.

Report a bug

9.10.4. Additional Setup Instructions for Connecting Services

Once HTTPD (and, by extension, the OpenStack Identity service) has been configured to use external authentication, you will need to configure how all OpenStack services connect to the Identity service. Each connecting service will need to use the correct admin_prefix, authentication port, and authentication protocol; these must be in the **[keystone_authtoken]** settings of each OpenStack service configuration file:

```
[keystone_authtoken]
auth_admin_prefix = /keystone/admin
auth_port = 443
auth_protocol = https
```

For example, to apply these settings to the Networking (**neutron**) service, run the following commands:

```
# openstack-config --set /etc/neutron/neutron.conf \
 keystone_authtoken auth_admin_prefix /keystone/admin
# openstack-config --set /etc/neutron/neutron.conf \
 keystone_authtoken auth_port 443
# openstack-config --set /etc/neutron/neutron.conf \
 keystone_authtoken auth_protocol https
```

Report a bug

9.11. Validate 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.

Procedure 9.17. Validating the Identity Service installation

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.

Report a bug

Chapter 10. Installing the OpenStack Object Storage Service

10.1. 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.

Report a bug

10.2. 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 services:

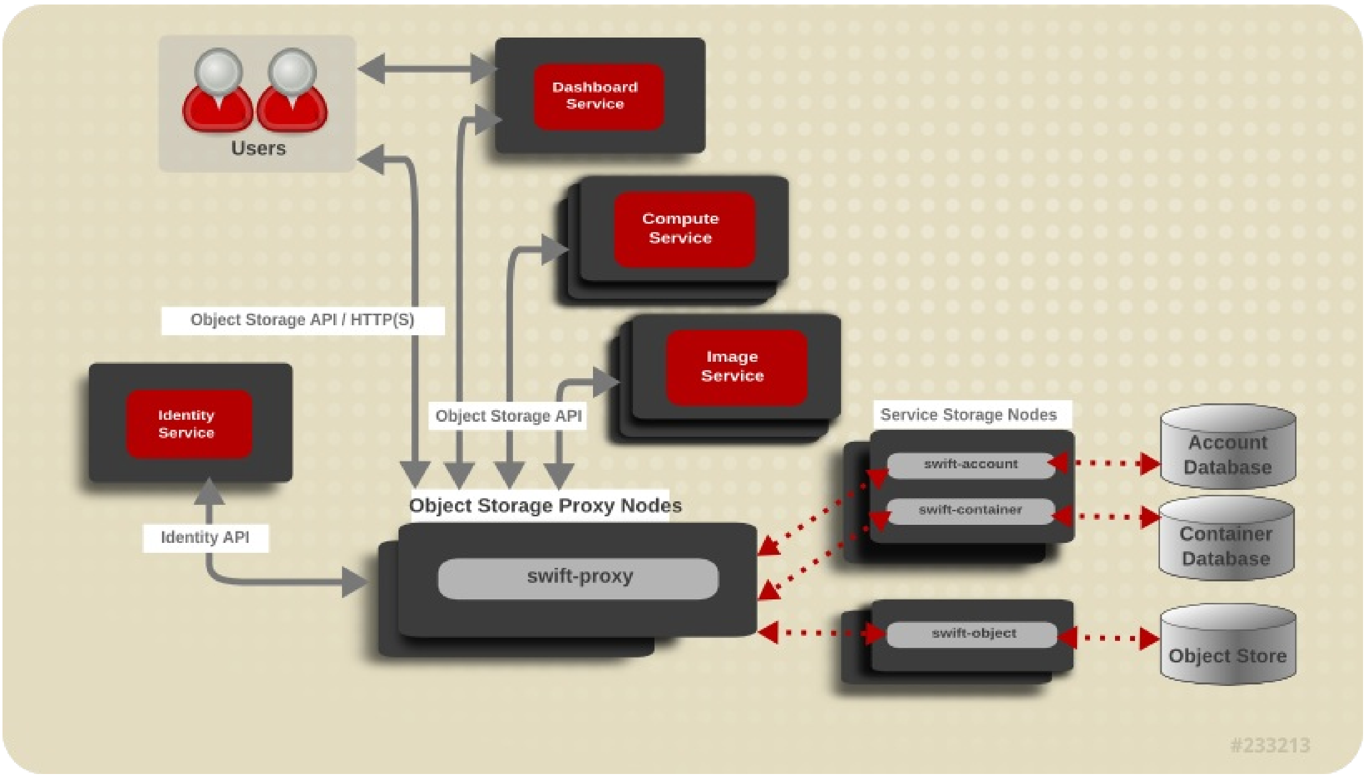


Figure 10.1. Service Architecture

Report a bug

10.3. 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/**.

Report a bug

10.4. Install 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 10.1. 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.

Report a bug

10.5. Configuring the Object Storage Service

10.5.1. Create the Object Storage Service Identity Records

Prerequisites:

- Section 9.7, “Create an Administrator Account”
- Section 9.9, “Create the Services Tenant”

This section assumes that you have already created an administrator account and **services** tenant. For more information, refer to:

- Section 9.7, “Create an Administrator Account”
- Section 9.9, “Create the Services Tenant”

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.

In order to proceed, you should have already performed the following (using the Identity service):

1. Created an Administrator role named **admin** (refer to Section 9.7, “Create an Administrator Account” for instructions)
2. Created the **services** tenant (refer to Section 9.9, “Create the Services Tenant” for instructions)

Note

The *Installation and Configuration Guide* uses one tenant for all service users. For more information, refer to Section 9.9, “Create the Services Tenant”.

You can perform this procedure 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 10.2. Configuring the Object Storage Service to authenticate through the Identity 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
```

3. Add the **swift** user to the **services** tenant with the **admin** role:

```
# keystone user-role-add --user swift --role admin --tenant services
```

4. Create the **swift** Object Storage service entry:

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

5. Create the **swift** endpoint entry:

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

   Replace *IP* with the IP address or 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.

Report a bug

10.5.2. Configure the Object Storage Service Storage Nodes

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 10.3. Configuring the Object Storage Service storage nodes

1. Format your devices using the **ext4** or **XFS** filesystem. Make sure that **xattr**s 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. Configure the firewall to 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.

   a. Open the **/etc/sysconfig/iptables** file in a text editor.

   b. Add an **INPUT** rule allowing TCP traffic on the ports used by the account, container, and object service. The new rule must appear before any **reject-with icmp-host-prohibited** rule.

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

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 (swift_hash_path_prefix) to your **/etc/swift.conf**:

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

7. Use the **openstack-config** command to add a hash suffix (swift_hash_path_suffix) to your **/etc/swift.conf**:

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

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

8. 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.

9. 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.

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

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

11. 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.

Report a bug

10.5.3. Configure the Object Storage Service Proxy Service

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 10.4. 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 Security Guide* from the following link:

https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/

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

10.5.4. 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 ^ partition power = 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 deployment. You can use a zone to represent sites, cabinets, nodes, or even devices.

Table 10.1. Parameters used when building ring files

Ring File Parameter	Description
Partition power	2 ^ partition power = partition count. The partition is rounded up after calculation.
Replica count	The number of times that your data will be replicated in the cluster.

Ring File Parameter	Description
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

10.5.5. Build 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 10.5. 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 devices to the account ring.

   ```
    # swift-ring-builder /etc/swift/account.builder add zX-SERVICE_IP:6002/dev_mountpt
   part_count
   ```

   Where:

   - *X* is the corresponding integer of a specified zone (for example, **z1** would correspond to Zone One).
   - *SERVICE_IP* is the IP on which the account, container, and object services should listen. This IP should match the **bind_ip** value set during the configuration of the Object Storage service Storage Nodes.
   - *dev_mountpt* is the **/srv/node** subdirectory under which your device is mounted.
   - *part_count* is the partition count you used to calculate your partition power.

   For example, if:

   - The account, container, and object services are configured to listen on **10.64.115.44** on Zone One,
   - Your device is mounted on **/srv/node/accounts**, and
   - You wish to set a partition count of 100.

   Then run:

   ```
    # swift-ring-builder /etc/swift/account.builder add z1-10.64.115.44:6002/accounts 100
   ```

   Note

   Repeat this step for each device (on each node in the cluster) you want added to the ring.

3. In a similar fashion, add devices to both containers and objects rings:

   ```
    # swift-ring-builder /etc/swift/container.builder add zX-SERVICE_IP:6001/dev_mountpt
   part_count
   ```

```
# swift-ring-builder /etc/swift/object.builder add zX-SERVICE_IP:6000/dev_mountpt
part_count
```

Replace the variables herein with the same ones used in the previous step.

Note

Repeat these commands for each device (on each node in the cluster) you want added to the ring.

4. 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
```

5. 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
```

6. 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
```

7. 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

10.6. Validate the Object Storage Service Installation

After installing and configuring the Block Storage service, perform the following steps to validate it:

Procedure 10.6. Validating the Object Storage Service installation

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 the admin user:

```
$ source ~/keystonerc_admin
```

3. 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 ...
    135 proxy-server Server reports support for api versions: v3.0, v2.0
   ```

4. 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
   ```

5. 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
   ```

You have now uploaded 3 files into 1 container. If you check the other storage devices, you will find more **.data** files, depending on your replica count.

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

Report a bug

Chapter 11. Installing the OpenStack Image Service

11.1. 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. This endpoint is configured as part of Section 10.5.1, “Create the Object Storage Service Identity Records”.

See Also:

- Chapter 2, *Prerequisites*

Report a bug

11.2. Install 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

11.3. Create 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**).

Procedure 11.1. Creating the Image Service database

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

Report a bug

11.4. Configuring the Image Service

11.4.1. 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.

Report a bug

11.4.2. Create the Image Identity Records

This section outlines the steps for creating and configuring Identity service records required by the Image service.

1. Create the **glance** user, 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.

In order to proceed, you should have already performed the following (through the Identity service):

1. Created an Administrator role named **admin** (refer to Section 9.7, “Create an Administrator Account” for instructions)
2. Created the **services** tenant (refer to Section 9.9, “Create the Services Tenant” for instructions)

Note

The *Installation and Configuration Guide* uses one tenant for all service users. For more information, refer to Section 9.9, “Create the Services Tenant”.

You can perform this procedure from your Identity service host 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 11.2. Configuring the Image Service to authenticate through the Identity Service

1. Authenticate as the administrator of the Identity service by running the **source** command on the **keystonerc_admin** file:

   ```
   # 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.

3. 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 glance --role admin --tenant services
   ```

4. 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              |
   +-------------+----------------------------------+
   ```

5. Create the **glance** endpoint entry:

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

   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.

Report a bug

11.4.3. Configure the Image Service Database Connection

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.

Procedure 11.3. Configuring the Image Service SQL database connection

1. Use the **openstack-config** command to set the value of the **sql_connection** configuration key in the **/etc/glance/glance-api.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 Image 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.

Report a bug

11.4.4. Configure Image Service Authentication

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

Procedure 11.4. Configuring the Image Service to authenticate through the Identity 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_authtoken auth_host IP
   # openstack-config --set /etc/glance/glance-api.conf \
      keystone_authtoken auth_port 35357
   # openstack-config --set /etc/glance/glance-api.conf \
      keystone_authtoken auth_protocol http
   # openstack-config --set /etc/glance/glance-api.conf \
      keystone_authtoken admin_tenant_name services
   # openstack-config --set /etc/glance/glance-api.conf \
      keystone_authtoken admin_user glance
   # openstack-config --set /etc/glance/glance-api.conf \
      keystone_authtoken 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_authtoken auth_host IP
   # openstack-config --set /etc/glance/glance-registry.conf \
      keystone_authtoken auth_port 35357
   # openstack-config --set /etc/glance/glance-registry.conf \
      keystone_authtoken auth_protocol http
   # openstack-config --set /etc/glance/glance-registry.conf \
      keystone_authtoken admin_tenant_name services
   ```

```
# openstack-config --set /etc/glance/glance-registry.conf \
   keystone_authtoken admin_user glance
# openstack-config --set /etc/glance/glance-registry.conf \
   keystone_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 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.

Report a bug

11.4.5. Using the Object Storage Service for Image Storage

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
   ```

Report a bug

11.4.6. Configure the Firewall to Allow Image Service Traffic

The Image Service should be accessible over the network through port **9292**.

To allow this, the Image service should be configured to recognize the 9292 port, and the firewall on the system hosting the image storage service should also 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.

Procedure 11.5. Configuring the firewall to allow Image Service traffic

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**.

Report a bug

11.4.7. Configure the Image Service to Use SSL

Use the following options in the **glance-api.conf** file to configure SSL.

Table 11.1. SSL options for Block Storage

Configuration Option	Description
cert_file	Path to certificate file to use when starting API server securely.
key_file	Path to private key file to use when starting API server securely.
ca_file	Path to CA certificate file to use to verify connecting clients.

Report a bug

11.4.8. Populate the Image Service Database

You can populate the Image Service database after you have successfully configured the Image Service database connection string (refer to Section 11.4.3, “Configure the Image Service Database Connection”).

Procedure 11.6. Populating the Image Service database

1. Log in to the system hosting the Image service.
2. Use the **su** command to switch to the **glance** user.

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

3. 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

11.4.9. Enabling Image Loading via the Local File System

By default, the Image service provides images to instances using the HTTP protocol. That is, image data is transmitted from the image store to the local disk of the Compute node using HTTP.

This process is typical for most deployments where the Image and Compute services are installed on different hosts.

Note

You can use direct image access even if Image and Compute services are not installed on the same hosts, but are sharing a shared file system. The only requirement in this case would be to have the file system mounted in the same location.

In deployments where both services are installed on the same host (and, consequently, share the same file system), it is more efficient to skip the HTTP steps altogether. Instead, you will need to configure both Image and Compute services to send and receive images using the local file system.

To do this:

Procedure 11.7. Configuring Image and Compute services to send/receive images through the local file system

Important

The Image file system metadata to be generated for this procedure will only apply to new images. Any existing images will not use this metadata.

1. Create a JSON document that exposes the Image file system metadata required by **openstack-nova-compute**.
2. Configure the Image service to use the JSON document.
3. Configure **openstack-nova-compute** to use the file system metadata provided by the Image service.

Note

If both Image and Compute services are hosted on different nodes, you can emulate local file system sharing through Gluster.

The following sections describe this in more detail.

Report a bug

11.4.9.1. Configure File System Sharing Across Different Image and Compute Nodes

If both the Image and Compute services are hosted on different nodes, you can still enable them to share images locally. To do so, you will have to use Gluster (Red Hat Storage shares).

With this configuration, both Image and Compute services will have to share the same Gluster volume. For this, the same volume must be mounted on their respective nodes. Doing so will allow both services to access the same volume locally, thereby allowing image loading through local file system.

This configuration requires that you:

1. Install and configure the packages required for Gluster on the nodes hosting Image and Compute services.
2. Create the GlusterFS volume to be shared by both Image and Compute services.
3. Mount the GlusterFS volume on the Image and Compute service nodes.

Note

For instructions on this procedure, refer to the latest version of the *Configuring Red Hat OpenStack with Red Hat Storage* guide available from:

https://access.redhat.com/site/documentation/en-US/Red_Hat_Storage/

Once you have configured the GlusterFS volume and mounted it on the Image service node, you will also have to configure the Compute service node to use the mounted Gluster volume. To do so:

Procedure 11.8. Configuring the Compute service node to use a mounted Gluster volume

1. Log in to the Compute service node.
2. From there, install the packages required for Gluster:

```
# yum install -y glusterfs glusterfs-fuse
```

3. Ensure that the drivers required to load the Gluster volume are enabled. To do so:
   a. Open the **/etc/nova/nova.conf** configuration file.
   b. Search for the **Libvirt** handlers for remote volumes (specifically, **libvirt_volume_drivers**). The value for this parameter should be a comma-delimited list of drivers for different types of volumes.
   c. Depending on your Compute service deployment, the **libvirt_volume_drivers** may already be enabled (as in, uncommented). If so, ensure that the Gluster volume driver (namely **glusterfs=nova.virt.libvirt.volume.LibvirtGlusterfsVolumeDriver**) is also listed.

      If the **libvirt_volume_drivers** parameter is disabled or is not listed, edit the file accordingly.
4. Configure the Compute service to use the mounted Gluster volume:

```
# openstack-config --set /etc/nova/nova-conf \
 DEFAULT glusterfs_mount_point_base GLUSTER_MOUNT
```

   Replace *GLUSTER_MOUNT* with the directory where the Gluster volume is mounted.
5. Restart the Compute service.

```
# service openstack-nova-compute restart
```

After completing these procedures, both Image and Compute services can now emulate accessing the same file system as if it were a local file system. You can then enable image loading through the local file system as normal.

Report a bug

11.4.9.2. Configure the Image Service to Provide Images Through the Local File System

In order to enable image loading through the local file system (as opposed to HTTP), the Image service needs to first expose its local file-system metadata to the **openstack-nova-compute** service. To do so:

Procedure 11.9. Configuring the Image service to expose local file system metadata to the Compute service

1. Determine the mount point of the file system used by the Image service:

   ```
   # df
   Filesystem     1K-blocks     Used Available Use% Mounted on
   /dev/sda3       51475068 10905752  37947876  23% /
   devtmpfs         2005504        0   2005504   0% /dev
   tmpfs            2013248      668   2012580   1% /dev/shm
   ```

   For example, if the Image service uses the **/dev/sda3** file system, then its corresponding mount point is **/**.

2. Create a unique ID for the mount point using:

   ```
   # uuidgen
   ad5517ae-533b-409f-b472-d82f91f41773
   ```

   Note the output of the **uuidgen**, as this will be used in the next step.

3. Create a file with the **.json** extension.

4. Open the file and add the following information:

   ```
   {
   "id": "UID",
   "mountpoint": "MOUNTPT"
   }
   ```

   Where:

   - *UID* is the unique ID created in the previous step.
   - *MOUNTPT* is the mount point of the Image service's file system (as determined in the first step).

5. Configure the Image service to use this JSON file:

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

   ```
   # openstack-config --set /etc/glance/glance-api.conf \
    DEFAULT filesystem_store_metadata_file JSON_PATH
   ```

   Replace *JSON_PATH* with the full path to the JSON file.

6. Restart the Image service (if already running).

   ```
   # service openstack-glance-registry restart
   # service openstack-glance-api restart
   ```

Important

The Image file-system metadata generated for this procedure only applies to new images. Any image that exists (that is, prior to this procedure) will not use this metadata.

Report a bug

11.4.9.3. Configure the Compute Service to Use Local File System Metadata

Prerequisites:

- Section 11.4.9.2, “Configure the Image Service to Provide Images Through the Local File System”
- Chapter 14, *Installing the OpenStack Compute Service*

After configuring the Image Service to expose local file system metadata (as part of Section 11.4.9.2, “Configure the Image Service to Provide Images Through the Local File System”, you can then configure the Compute service to use this metadata. Doing so allows **openstack-nova-compute** to load images from the local file system.

To do so:

Procedure 11.10. Configuring the Compute service to use file system metadata provided by the Image Service

1. Configure **openstack-nova-compute** to enable the use of direct URLs that have the **file://** scheme:

   ```
   # openstack-config --set /etc/nova/nova.conf \
    DEFAULT allowed_direct_url_schemes file
   ```

2. Create an entry for the Image service's file system:

   ```
   # openstack-config --set /etc/nova/nova.conf \
    image_file_url filesystems FSENTRY
   ```

   Replace *FSENTRY* with the name you wish to assign to the Image service's file system.

3. Open the **.json** file used by the Image service to expose its local file system metadata. The information in this file will be used in the next step.

4. Associate the entry created in the previous step to the file system metadata exposed by the Image service:

   ```
   # openstack-config --set /etc/nova/nova.conf \
    image_file_url:FSENTRY id UID
   ```

   ```
   # openstack-config --set /etc/nova/nova.conf \
    image_file_url:FSENTRY mountpoint MOUNTPT
   ```

   Where:

   - *UID* is the unique ID used by the Image service. In the **.json** file used by the Image service, the *UID* is the **"id":** value.
   - *MOUNTPT* is the mount point used by the Image service's file system. In the **.json** file used by the Image service, the *MOUNTPT* is the **"mountpoint":** value.

Report a bug

11.5. Launch 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

11.6. Validating the Image Service Installation

11.6.1. Obtain 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 (see Section 11.6.3, “Upload a Disk Image”).

A new image is provided with each minor Red Hat Enterprise Linux 6 release, and is available in the download section of the Red Hat Enterprise Linux 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.isoxxxxxxxxx"
```

Report a bug

11.6.2. Build a Custom Virtual Machine Image

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 11.11. 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.

   Although Oz supports the use of network-based installation media, in this procedure a Red Hat Enterprise Linux 6 DVD 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 11.1. 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>rhel65_x86_64</name>
 <description>Red Hat 6.5 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-6.5-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 / console=ttyS0,115200n8 console=tty0 /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.

Note

The Red Hat Enterprise Linux OpenStack Platform 4 release included the **Disk Image Builder** as a Technology Preview. This tool provides automation to build RAM disks and disk images for deploying instances through OpenStack. While users and operators can manually script or put together RAM disks and disk images, automation makes customization and testing easier.

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

Report a bug

11.6.3. Upload a Disk Image

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. For more information, refer to:

- Section 11.6.1, "Obtain a Test Disk Image"
- Section 11.6.2, "Build a Custom Virtual Machine Image"

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 11.2. 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 **`./RHEL65.img`**.

```
# qemu-img info ./RHEL65.img
image: ./RHEL65.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 11.3. Uploading an Image to the Image service

In this example the **`qcow2`** format image in the file named **`RHEL65.img`** is uploaded to the Image service. It is created in the service as a publicly accessible image named **`RHEL 6.5`**.

```
# glance image-create --name "RHEL 6.5" --is-public true --disk-format qcow2 \
         --container-format bare \
         --file RHEL65.img
+------------------+--------------------------------------+
| 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 6.5                             |
| owner            | b1414433c021436f97e9e1e4c214a710     |
| protected        | False                                |
| size             | 25165824                             |
| status           | active                               |
| updated_at       | 2013-01-25T14:45:50                  |
+------------------+--------------------------------------+
```

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

```
 # glance image-list
+--------------+----------+------------+-----------------+----------+--------+
| ID             | Name     | Disk Format | Container Format |Size       | Status |
+--------------+----------+-------------+------------------+----------+--------+
| 0ce782c6-... | RHEL 6.5 | qcow2       | bare             |213581824 | active |
+--------------+----------+-------------+------------------+----------+--------+
```

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 6.5                             |
| 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 11.6.1, “Obtain a Test Disk Image”
- Section 11.6.2, “Build a Custom Virtual Machine Image”

Report a bug

Chapter 12. Installing OpenStack Block Storage

12.1. 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.

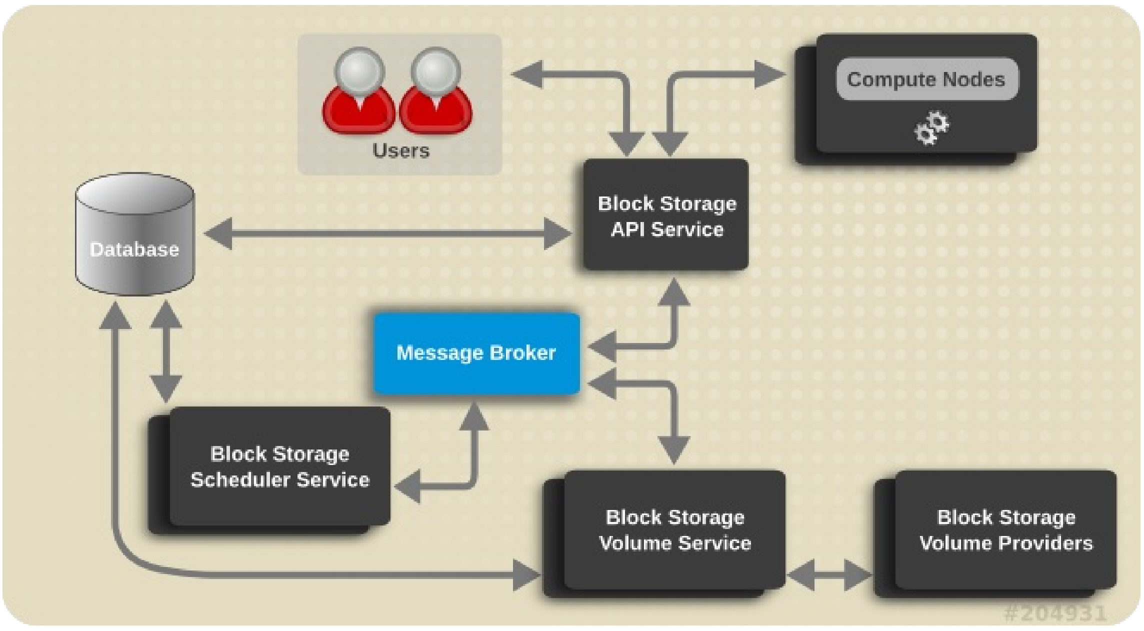

Users
Compute Nodes
Database
Block Storage API Service
Message Broker
Block Storage Scheduler Service
Block Storage Volume Service
Block Storage Volume Providers
#204931

Figure 12.1. Block Storage Architecture

Deploying the Block Storage service is made up of three major phases:

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.

See Also:

- Section 12.2, “Block Storage Prerequisite Configuration”
- Section 12.3, “Common Block Storage Configuration”
- Section 12.4, “Volume Service Specific Configuration”
- Section 12.5, “Launch the Block Storage Services”

Report a bug

12.2. Block Storage Prerequisite Configuration

12.2.1. Create 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.

Procedure 12.1. Creating the database to be used by Block Storage Services

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.

   ```
   mysql> quit
   ```

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

Report a bug

12.2.2. Create the Block Storage Service Identity Records

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.

In order to proceed, you should have already performed the following (through the Identity service):

1. Created an Administrator role named **admin** (refer to Section 9.7, "Create an Administrator Account" for instructions)
2. Created the **services** tenant (refer to Section 9.9, "Create the Services Tenant" for instructions)

Note

The *Installation and Configuration Guide* uses one tenant for all service users. For more information, refer to Section 9.9, "Create the Services Tenant".

You can perform the following procedure from your Identity service host 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 12.2. Creating Identityrecords for the Block Storage 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 **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.

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

```
# keystone user-role-add --user cinder --role admin --tenant services
```

4. 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              |
+-------------+----------------------------------+
```

5. Create the **cinder** endpoint entry.

```
# keystone endpoint-create \
       --service cinder \
       --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 *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

12.3. Common Block Storage Configuration

12.3.1. Install 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.

Report a bug

12.3.2. Configure Block Storage Service Authentication

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.

Procedure 12.3. Configuring the Block Storage Service to authenticate through the Identity Service

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_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 Block Storage service. In this guide, examples use *services*.

```
# openstack-config --set /etc/cinder/cinder.conf \
  keystone_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 Block Storage service. In this guide, examples use *cinder*.

```
# openstack-config --set /etc/cinder/cinder.conf \
  keystone_authtoken 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_authtoken admin_password PASSWORD
```

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

Report a bug

12.3.3. Configure the Block Storage Service to Use SSL

Use the following options in the **cinder.conf** file to configure SSL.

Table 12.1. SSL options for Block Storage

Configuration Option	Description
backlog	Number of backlog requests to configure the socket with.
tcp_keepidle	Sets the value of TCP_KEEPIDLE in seconds for each server socket.
ssl_ca_file	CA certificate file to use to verify connecting clients.
ssl_cert_file	Certificate file to use when starting the server securely.
ssl_key_file	Private key file to use when starting the server securely.

Report a bug

12.3.4. Configure Message Broker Settings for the Block Storage Service

Prerequisites:

- Section 8.2, “Configuring the Message Broker”

This section assumes that you have already configured a Qpid Message Broker. For more information, refer to:

- Section 8.1, “Install the Message Broker Packages”
- Section 8.2.1, “Basic Qpid Configuration”
- Section 8.2.2.1, “Simple Authentication and Security Layer (SASL)”

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.

Procedure 12.4. Configuring the Message Broker settings of the Block Storage Service

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 cinder
      ```

   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.

   Note

   The Block Storage service Qpid user account is configured as part of Section 8.2.2.4, "Configure SASL Using a Local Password File".

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**).

5. **Exchange Setting**

   Set the value of the **control_exchange** configuration key to **cinder**.

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

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

See Also:

- Section 8.2.2.4, “Configure SASL Using a Local Password File”

Report a bug

12.3.5. Configure the Block Storage Service Database Connection

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.

Report a bug

12.3.6. Configure the Firewall to Allow Block Storage Service Traffic

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.

Procedure 12.5. Configuring the firewall to allow Block Storage Service traffic

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**.

Report a bug

12.3.7. Populate the Block Storage Database

You can populate the Block Storage database after you have successfully configured the Block Storage service database connection string (refer to Section 12.3.5, "Configure the Block Storage Service Database Connection").

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.

Procedure 12.6. Populating the Block Storage Service database

1. Log in to the system hosting one of the Block Storage services.
2. Use the **su** command to switch to the **cinder** user.

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

3. 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

12.4. Volume Service Specific Configuration

12.4.1. Block Storage Driver Support

The volume service (**openstack-cinder-volume**) requires access to suitable block storage. As such, Red Hat Enterprise Linux OpenStack Platform provides volume drivers for several supported block storage types. These types include:

- LVM/iSCSI
- ThinLVM
- NFS
- NetAPP NFS
- Red Hat Storage (Gluster)

For more detailed information on configuring volume drivers for the Block Storage service, refer to the *Volume Drivers* section of the *Red Hat Enterprise Linux OpenStack Platform 4 Configuration Reference Guide*. This document is available from the following link:

https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform

Report a bug

12.4.2. Configure OpenStack Block Storage to use an LVM Storage Back End

The **openstack-cinder-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 **openstack-cinder-volume** service as the **root** user:

Procedure 12.7. Configuring openstack-cinder-volume to use LVM storage as a back end

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 **openstack-cinder-volume** service has been configured to use LVM storage.

Report a bug

12.4.3. Configuring for NFS Storage Backend

This section presents the steps involved in configuring the **openstack-cinder-volume** service to use NFS storage. The NFS shares to be used must:

- already exist
- be accessible from the **openstack-cinder-volume** host

1. Log in as **root** to the system hosting the **openstack-cinder-volume** service.

2. Create a text file named **nfsshares** in **/etc/cinder/**.

3. Add an entry to **/etc/cinder/nfsshares** for each NFS share that the **openstack-cinder-volume** should use for backend storage. Each entry should be a separate line, and should use the following format:

   ```
   HOST:SHARE
   ```

   Where:

   - *HOST* is the IP address or host name of the NFS server.
   - *SHARE* is the absolute path to an existing and accessible NFS share.

4. Set **/etc/cinder/nfsshares** to be owned by the **root** user and the **cinder** group:

   ```
   # chown root:cinder /etc/cinder/nfsshares
   ```

5. Set **/etc/cinder/nfsshares** to be readable by members of the **cinder** group:

   ```
   # chmod 0640 /etc/cinder/nfsshares
   ```

6. In **/etc/cinder/cinder.conf**, set the **nfs_shares_config** configuration key to **/etc/cinder/nfsshares**:

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

7. Optionally, provide any additional NFS mount options required in your environment in the **nfs_mount_options** configuration key of **/etc/cinder/cinder.conf**. If your NFS shares do not require any additional mount options (or if you are unsure), 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**).

8. Set the **volume_driver** configuration key in **/etc/cinder/cinder.conf** to the correct volume driver, namely **cinder.volume.drivers.nfs.NfsDriver**:

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

The **openstack-cinder-volume** service is now configured to use NFS storage.

Note

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 **nfs_sparsed_volumes** to **false** will result in volumes being fully allocated at the time of creation. This leads to increased delays in volume creation. To set this:

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

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.

To enable this Boolean on a client machine, run the following command as the **root** user:

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

Report a bug

12.4.4. Configuring for Red Hat Storage Backend

This section documents the procedures involved in configuring the volume service (**openstack-cinder-volume**) 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.

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.

The Red Hat Storage server must also be configured accordingly in order to allow the volume service to use Gluster shares:

Procedure 12.8. Configuring Red Hat Storage for the OpenStack volume service

1. Log in as **root** to the Red Hat Storage server.

2. Set each Gluster volume to use the same UID and GID as the **cinder** user:

   ```
   # gluster volume set VOL_NAME storage.owner-uid 165
   # gluster volume set VOL_NAME storage.owner-gid 165
   ```

   Replace *VOL_NAME* with the Gluster volume name.

   Note

   These commands set the volume storage GID and UID to **165**, which is the default GID and UID of the **cinder** user.

3. Next, you will need to allow each Gluster volume to accept **libgfapi** connections. To do this, set each Gluster volume to allow insecure ports:

   ```
   # gluster volume set VOL_NAME server.allow-insecure on
   ```

4. Enable client connections from unprivileged ports. To do this, add the following line to **/etc/glusterfs/glusterd.vol**:

   ```
   option rpc-auth-allow-insecure on
   ```

5. Restart the **glusterd** service:

   ```
   # service glusterd restart
   ```

Once the Red Hat Storage service is configured as needed, perform the following procedure:

Procedure 12.9. Configuring openstack-cinder-volume to use Red Hat Storage

1. Log in as **root** to the system hosting the volume service.

2. Create a text file in the **/etc/cinder/** directory containing a list of the Red Hat Storage shares that the volume service should use for backing storage. Each line in the list should appear as:

   ```
   HOST:/VOL_NAME
   ```

   Where:

   - *HOST* is the IP address or host name of the Red Hat Storage server.
   - *VOL_NAME* is the name of a volume on that Red Hat Storage server.

   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:/VOL_NAME -o OPTIONS
   ```

   Replace *OPTIONS* with a comma-separated list of mount options.

3. 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.

4. 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.

5. 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.

6. 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
   ```

7. 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.

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
```

Report a bug

12.4.5. Configuring for Multiple Storage Backends

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 12.1. 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.

Report a bug

12.4.6. Configure tgtd

The volume storage service makes use of the SCSI target daemon, **`tgtd`**, when mounting storage. To support this the **`tgtd`** 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.

Procedure 12.10. Configuring tgtd

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 **tgtd** service is started it will be configured to support the volume service.

Report a bug

12.5. Launch 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.

Procedure 12.11. Launching Block Storage Services

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 (**tgtd**).

      ```
      # service tgtd 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 (**tgtd**).

```
# chkconfig tgtd on
```

The volume service is running and ready to begin allocating volumes as they are requested.

Report a bug

12.6. Validate the Block Storage Service Installation

This procedure lists steps for validating that the block storage installation is complete and ready for use.

Procedure 12.12. Validating the Block Storage Service installation

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 **keystonerc_admin** file containing the credentials of the OpenStack administrator. Transfer the **keystonerc_admin** file to the system before proceeding.

   a. Run the **source** command on the **keystonerc_admin** file to populate the environment variables used for identifying and authenticating the user.

   ```
   # source ~/keystonerc_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 **keystonerc_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 **keystonerc_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.

Report a bug

Chapter 13. Installing the OpenStack Networking Service

13.1. OpenStack Networking Installation Overview

13.1.1. 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.

Report a bug

13.1.2. 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.

Report a bug

13.1.3. OpenStack Networking API Extensions

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.

Note

Unlike Compute security groups, OpenStack Networking security groups are applied on a per port basis rather than on a per instance basis.

Report a bug

13.1.4. OpenStack Networking Plug-ins

The original OpenStack Compute network implementation assumed a basic networking model where all network isolation was performed through the use of Linux VLANs and iptables firewalls. 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 iptables firewalls, 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-neutron-openvswitch*)
- Linux Bridge (*openstack-neutron-linuxbridge*)

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

- Cisco (*openstack-neutron-cisco*)
- NEC OpenFlow (*openstack-neutron-nec*)
- Nicira (*openstack-neutron-nicira*)
- Ryu (*openstack-neutron-ryu*)

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

Report a bug

13.1.5. Open vSwitch overview

Open vSwitch is a *software-defined networking* (SDN) virtual switch designed to supercede the heritage Linux software bridge. It provides switching services to virtualized networks with support for industry standard NetFlow, OpenFlow, and sFlow. Open vSwitch is also able to integrate with physical switches due to its support for the layer 2 features STP, LACP, and 802.1Q VLAN tagging.

Open vSwitch tunneling is supported with Open vSwitch version 1.11.0-1.el6 or later. Refer to the table below for specific kernel requirements:

Feature	Kernel Requirement
GRE Tunneling	2.6.32-358.118.1.openstack.el6 or later
VXLAN Tunneling	2.6.32-358.123.4.openstack.el6 or later

Report a bug

13.1.6. Modular Layer 2 (ML2) Overview

ML2 is the new Networking core plug-in introduced in OpenStack's Havana release. Superseding the previous model of singular plug-ins, ML2's modular design enables the concurrent operation of mixed network technologies. The monolithic Open vSwitch and linuxbridge plug-ins have been deprecated, and their functionality has been implemented as a ML2 mechanism.

The requirement behind ML2

Previously, Networking deployments were only able to use the plug-in that had been selected at implementation time. For example, a deployment running the Open vSwitch plug-in was only able to use Open vSwitch exclusively; it wasn't possible to simultaneously run another plug-in such as linuxbridge. This was found to be a limitation in environments with heterogeneous requirements.

ML2 network types

Multiple network segment types can be operated concurrently. In addition, these network segments can interconnect using ML2's support for multi-segmented networks. Ports are automatically bound to the segment with connectivity; it is not necessary to bind them to a specific segment. Depending on the mechanism driver, ML2 supports the following network segment types:

- flat
- GRE
- local
- VLAN
- VXLAN

The various Type drivers are enabled in the ML2 section of the **`ml2_conf.ini`** file:

```
[ml2]
type_drivers = local,flat,vlan,gre,vxlan
```

ML2 Mechanisms

Plug-ins have been reimplemented as mechanisms with a common code base. This approach enables code reuse and eliminates much of the complexity around code maintenance and testing. Supported mechanism drivers currently include:

- Arista
- Cisco
- Nexus
- Hyper-V Agent

- L2 Population
- Linuxbridge Agent
- Open vSwitch Agent
- Tail-f NCS

The various mechanism drivers are enabled in the ML2 section of the **`ml2_conf.ini`** file:

```
[ml2]
mechanism_drivers = openvswitch,linuxbridge,l2population
```

Report a bug

13.1.7. Configure the L2 Population mechanism driver

The L2 Population driver enables broadcast, multicast, and unicast traffic to scale out on large overlay networks. By default, Open vSwitch GRE and VXLAN replicate broadcasts to every agent, including those that donât host the destination network. This design requires the acceptance of significant network and processing overhead. The alternative design introduced by the L2 Population driver implements a partial mesh for ARP resolution and MAC learning traffic. This traffic is sent only to the necessary agent by encapsulating it as a targeted unicast.

Enable the L2 population driver by adding it to the list of mechanism drivers. You also need to have at least one tunneling driver enabled; either GRE, VXLAN, or both. Add the appropriate configuration options to the **`ml2_conf.ini`** file:

```
[ml2]
type_drivers = local,flat,vlan,gre,vxlan
mechanism_drivers = openvswitch,linuxbridge,l2population
```

Report a bug

13.1.8. OpenStack Networking Agents

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-neutron* 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.

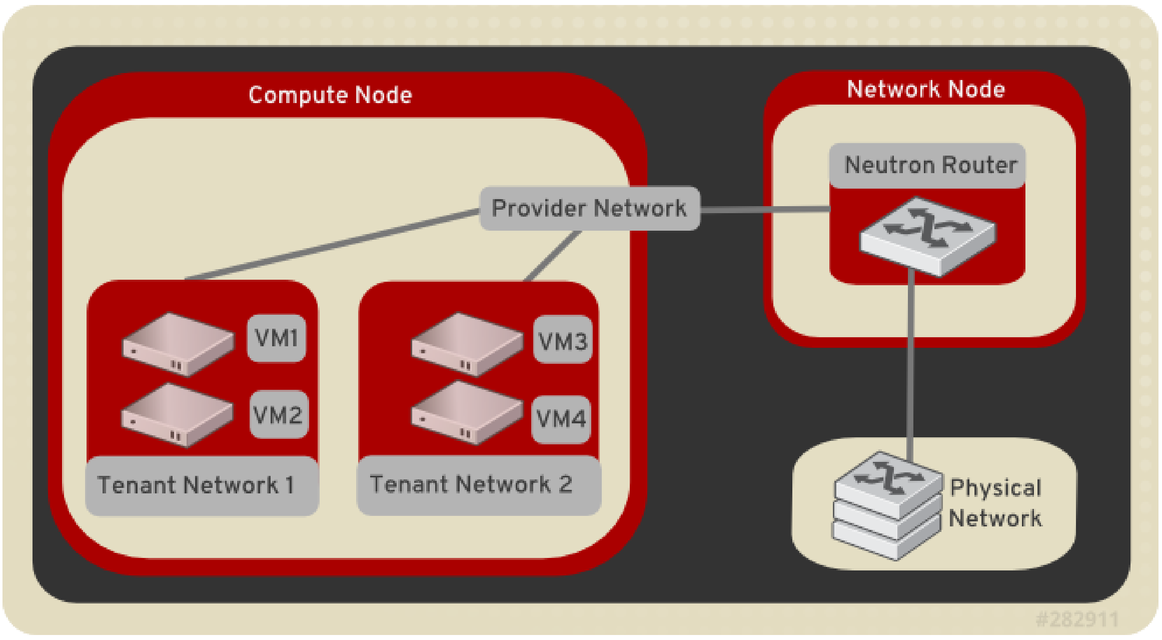
Plug-in Agent

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 **`neutron-dhcp-agent`** and **`neutron-l3-agent`**.

Report a bug

13.1.9. Comparing Tenant and Provider networks

The following diagram provides an overview of the tenant and provider network types, and illustrates how they interact within the overall Networking topology:



Tenant networks

Tenant networks are created by users for connectivity within projects; they are fully isolated by default and are not shared with other projects. Available tenant network types include VLAN (802.1Q tagged), GRE (Unique ID) and VXLAN (Unique VNI).

Flat

All instances reside on the same network, which can also be shared with the hosts. No VLAN tagging or other network segregation takes place.

Local

Instances reside on the local Compute host and are effectively isolated from any external networks.

VLAN

Networking allows users to create multiple provider or tenant networks using VLAN IDs that correspond to VLANs present in the external physical network. This allows instances to communicate with instances across the environment. They can also communicate with dedicated servers, firewalls, load balancers and other networking infrastructure on the same layer 2 VLAN.

VXLAN/GRE

VXLAN and GRE use network overlays to support private communication between instances. A Networking router is required to enable traffic to traverse outside of the GRE or VXLAN tenant network. A router is also required to connect directly-connected tenant networks with external networks, including the Internet; the router provides the ability to connect to instances directly from an external network using floating IP addresses.

Provider networks

Provider networks are networks created by the OpenStack administrator that map directly to an existing physical network in the data center. Useful network types in this category are flat (untagged) and VLAN (802.1Q tagged). It is possible to allow provider networks to be shared among tenants as part of the network creation process.

Report a bug

13.1.10. 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.

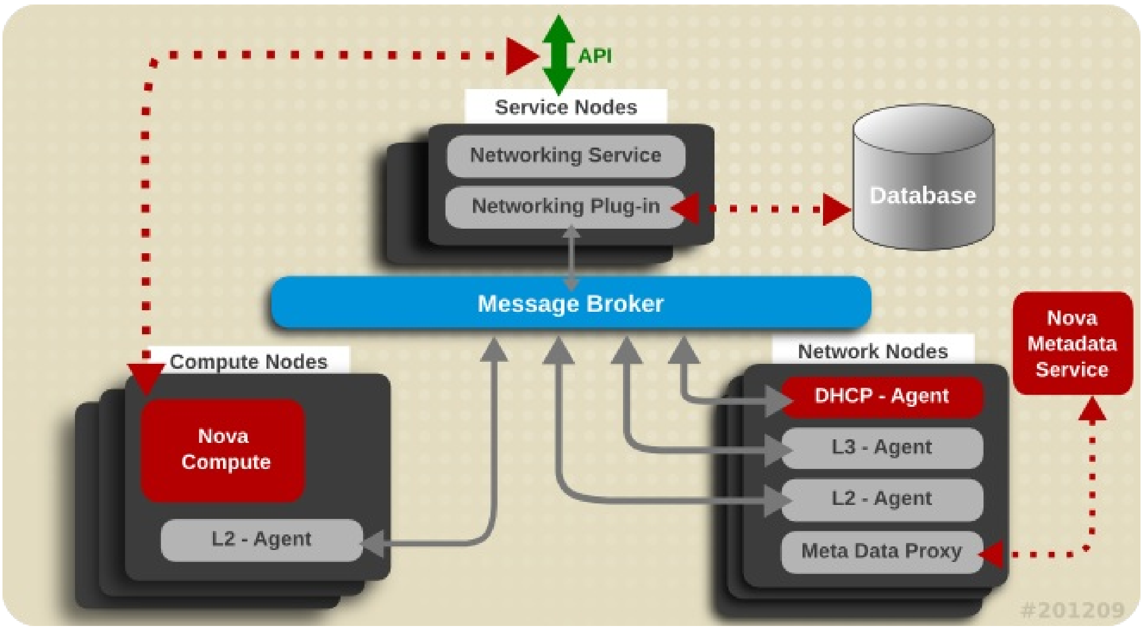


Figure 13.1. Deployment Architecture

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, stop the **openstack-nova-network** service on each Compute node first; to do so, run:

```
# 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 13.2, “Networking Prerequisite Configuration”
- Section 13.3, “Common Networking Configuration”
- Section 13.4, “Configuring the Networking Service”
- Section 13.5, “Configure the DHCP Agent”
- Section 13.6, “Create an External Network”
- Section 13.8, “Configure the L3 Agent”
- Section 13.7, “Configuring the Plug-in Agent”

Report a bug

13.1.11. Kernel Requirements

Nodes that handle OpenStack networking traffic require a Red Hat Enterprise Linux kernel that supports the use of network namespaces. In addition, the Open vSwitch plug-in requires kernel version **2.6.32-431.el6.x86_64** or later.

The default kernel included in Red Hat Enterprise Linux 6.5 fulfills both requirements.

Previous Red Hat Enterprise Linux OpenStack Platform releases required manually installing and booting to a custom Red Hat Enterprise Linux kernel that supported network namespaces and other OpenStack features. As of this release, doing so is no longer required. All OpenStack Networking requirements are now built into the default Red Hat Enterprise Linux 6.5 kernel.

Report a bug

13.2. Networking Prerequisite Configuration

13.2.1. Create 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.

Procedure 13.1. Creating the OpenStack Networking database

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_neutron**.
   - Linux Bridge plug-in, the recommended database name is **neutron_linux_bridge**.

   This example uses the database name of '**ovs_neutron**'.

   ```
   mysql> CREATE DATABASE ovs_neutron;
   ```

3. Create a **neutron** database user and grant it access to the **ovs_neutron** database.

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

   ```
   mysql> GRANT ALL ON ovs_neutron.* TO 'neutron'@'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 OpenStack Networking database 'ovs_neutron' has been created. The database will be populated during service configuration.

Report a bug

13.2.2. Configure OpenStack Networking Authentication

This section outlines the steps for creating and configuring Identity service records required by the Networking service.

1. Create the **neutron** user, who has the **admin** role in the **services** tenant.
2. Create the **neutron** 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. In order to proceed, you should have already performed the following (through the Identity service):

1. Created an Administrator role named **admin** (refer to Section 9.7, “Create an Administrator Account” for instructions)
2. Created the **services** tenant (refer to Section 9.9, “Create the Services Tenant” for instructions)

Note

The *Installation and Configuration Guide* uses one tenant for all service users. For more information, refer to Section 9.9, “Create the Services Tenant”.

You can perform the following procedure from your Identity service host 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 13.2. Configuring OpenStack Networking to authenticate through the Identity 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 **neutron** for the OpenStack Networking service to use:

   ```
   # keystone user-create --name neutron --pass PASSWORD
   +----------+----------------------------------+
   | Property |              Value               |
   +----------+----------------------------------+
   |  email   |                                  |
   | enabled  |               True               |
   |    id    | 1df18bcd14404fa9ad954f9d5eb163bc |
   |   name   |             neutron              |
   | tenantId |                                  |
   +----------+----------------------------------+
   ```

   Replace *PASSWORD* with a secure password that will be used by the OpenStack Networking service when authenticating with the Identity service.

3. Use the **keystone user-role-add** command to link the **neutron** user, **admin** role, and **services** tenant together:

   ```
   # keystone user-role-add --user neutron --role admin --tenant services
   ```

4. Create the **neutron** service entry:

   ```
   # keystone service-create --name neutron \
           --type network \
           --description "OpenStack Networking Service"
   +-------------+----------------------------------+
   |   Property  |              Value               |
   +-------------+----------------------------------+
   | description |   OpenStack Networking Service   |
   |      id     | 134e815915f442f89c39d2769e278f9b |
   |     name    |             neutron              |
   |     type    |             network              |
   +-------------+----------------------------------+
   ```

5. Create the **network** endpoint entry:

   ```
   # keystone endpoint-create --service-id \
           --service neutron \
           --publicurl "http://IP:9696" \
           --adminurl "http://IP:9696" \
           --internalurl "http://IP:9696"
   ```

   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

13.3. Common Networking Configuration

13.3.1. Disable Network Manager

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.

Procedure 13.3. Disabling the Network Manager service

1. Verify Network Manager is currently enabled using the **chkconfig** command.

   ```
   # chkconfig --list NetworkManager
   ```

   The output displayed by the **chkconfig** command inicates 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**. Add these keys manually if they do not already exist in each file.

```
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.

Report a bug

13.3.2. Install the Networking Service Packages

The OpenStack Networking service requires the following packages:

openstack-neutron

Provides the networking service and associated configuration files.

openstack-neutron-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-neutron \
  openstack-neutron-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.

Report a bug

13.3.3. Configure the Firewall to Allow OpenStack Networking Traffic

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.

Procedure 13.4. Configuring the firewall to allow OpenStack Networking traffic

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

13.4. Configuring the Networking Service

13.4.1. Configure Networking Service Authentication

Prerequisites:

- Section 13.3, “Common Networking Configuration”

The Networking service must be explicitly configured to use the Identity service for authentication. To accomplish this, perform the following procedure on the network node while logged in as **root** on the DHCP agent's host:

Procedure 13.5. Configuring the Networking Service to authenticate through the Identity Service

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

   ```
   # openstack-config --set /etc/neutron/neutron.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/neutron/neutron.conf \
      keystone_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 Networking service. Examples in this guide use *services*.

   ```
   # openstack-config --set /etc/neutron/neutron.conf \
      keystone_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 networking services. Examples in this guide use *neutron*.

   ```
   # openstack-config --set /etc/neutron/neutron.conf \
      keystone_authtoken admin_user neutron
   ```

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/neutron/neutron.conf \
      keystone_authtoken admin_password PASSWORD
   ```

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

Report a bug

13.4.2. Configure the Message Broker Settings of the Networking Service

Prerequisites:

- Section 8.2, "Configuring the Message Broker"

This section assumes that you have already configured a Qpid Message Broker. For more information, refer to:

- Section 8.1, "Install the Message Broker Packages"
- Section 8.2.1, "Basic Qpid Configuration"
- Section 8.2.2.1, "Simple Authentication and Security Layer (SASL)"

The Networking service must be explicitly configured with the type, location, and authentication details of the message broker. These details are configured in the **/etc/neutron/neutron.conf** file. Perform the following procedure on the network node while logged in as **root** on the DHCP agent's host:

Procedure 13.6. Configuring the Message Broker settings of the Networking Service

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

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

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

3. 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/neutron/neutron.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/neutron/neutron.conf \
     DEFAULT qpid_password PASSWORD
   ```

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

   Note

   The Networking service Qpid user account is configured as part of Section 8.2.2.4, "Configure SASL Using a Local Password File".

4. 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/neutron/neutron.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/neutron/neutron.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.

See Also:

- Section 8.2.2.4, “Configure SASL Using a Local Password File”

Report a bug

13.4.3. Set the Networking Service Plug-in

Additional configuration settings must be applied to enable the desired plug-in. If you are using the Open vSwitch plug-in, perform the following procedure:

Procedure 13.7. Enabling the Open vSwitch plug-in

1. Create a symbolic link between the **/etc/neutron/plugin.ini** path referred to by the Networking service and the plug-in specific configuration file.

   ```
   # ln -s /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini \
            /etc/neutron/plugin.ini
   ```

2. Update the value of the **tenant_network_type** configuration key in the **/etc/neutron/plugin.ini** file to refer to the type of network that must be used for tenant networks. Supported values are **flat**, **gre**, **local**, **vlan**, and **vxlan**.

   The default is **local** but this is not recommended for real deployments.

   Open vSwitch Tunneling allows virtual machines across multiple hosts to share a single layer 2 network. GRE and VXLAN tunnels are supported for encapsulating traffic between Open vSwitch endpoints on each host. Ensure that MTUs are an appropriate size from end-to-end, including those on the virtual machines.

   ```
   # openstack-config --set /etc/neutron/plugin.ini \
      OVS tenant_network_type TYPE
   ```

   Replace *TYPE* with the type chosen tenant network type.

3. 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/neutron/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
```

4. Update the value of the **core_plugin** configuration key in the **/etc/neutron/neutron.conf** file to refer to the Open vSwitch plug-in.

```
# openstack-config --set /etc/neutron/neutron.conf \
   DEFAULT core_plugin \
   neutron.plugins.openvswitch.ovs_neutron_plugin.OVSNeutronPluginV2
```

If you are using a Linux Bridge plug-in, perform the following procedure instead:

Procedure 13.8. Enabling the Linux Bridge plug-in

1. Create a symbolic link between the **/etc/neutron/plugin.ini** path referred to by the Networking service and the plug-in specific configuration file.

```
# ln -s /etc/neutron/plugins/linuxbridge/linuxbridge_conf.ini \
         /etc/neutron/plugin.ini
```

2. Update the value of the **tenant_network_type** configuration key in the **/etc/neutron/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/neutron/plugin.ini \
   VLAN tenant_network_type TYPE
```

   Replace *TYPE* with the type chosen tenant network type.

3. 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/neutron/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
```

4. Update the value of the **core_plugin** configuration key in the **/etc/neutron/neutron.conf** file to refer to the Linux Bridge plug-in.

```
# openstack-config --set /etc/neutron/neutron.conf \
   DEFAULT core_plugin \
   neutron.plugins.linuxbridge.lb_neutron_plugin.LinuxBridgePluginV2
```

Report a bug

13.4.4. Configure Open vSwitch Tunnels

Tunneling with Open vSwitch allows you to encapsulate network traffic between physical Networking hosts. This enables VLANs to span multiple physical hosts and consequently permits instances to communicate as if they're on the same layer 2 network. Open vSwitch supports tunneling with the VXLAN and GRE encapsulation protocols.

GRE configuration example

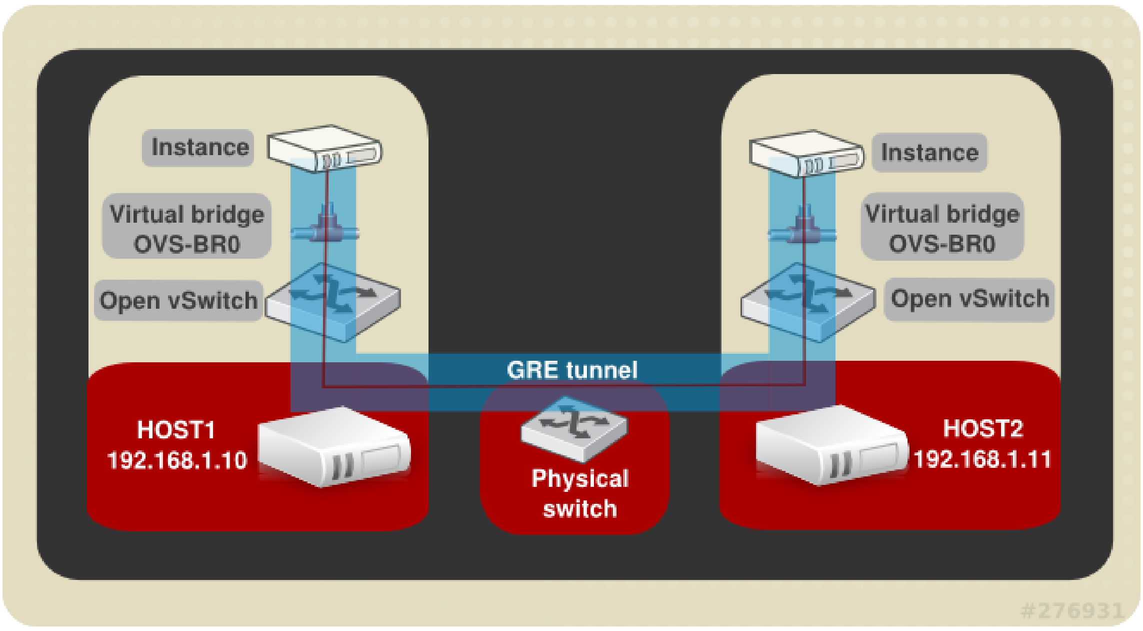


Figure 13.2. Example GRE Tunnel

This diagram represents two instances running on two separate hosts connected by a GRE tunnel. Also illustrated are the required supporting physical and virtual components. This example steps through the creation of a GRE or VXLAN tunnel between two OVS running on separate Networking hosts.

Procedure 13.9. Tunnel configuration steps

1. Each host participating in the network will required a virtual bridge named **`br-int`**; created using the **`ovs-ctl`** command on each host:

   ```
   ovs-vsctl add-br OVS-BR0
   ```

2. With each bridge created, you are now able to create the tunnel, and create links between the two OVS-BR0 virtual bridges created above.

   Run the ovs-vsctl command on HOST1 to create the tunnel and link it to the bridge on HOST2

   GRE tunnel command:

   ```
   ovs-vsctl add-port ovsbr0 gre1 -- set Interface gre1 type=gre
   options:remote_ip=192.168.1.11
   ```

   For a VXLAN tunnel:

   ```
   ovs-vsctl add-port ovsbr0 vxlan1 -- set Interface vxlan1 type=vxlan
   options:remote_ip=192.168.1.11
   ```

3. Run the ovs-vsctl command on HOST1 to create the tunnel and link it to the bridge on HOST2

   GRE tunnel command:

   ```
   ovs-vsctl add-port ovsbr0 gre1 -- set Interface gre1 type=gre
   options:remote_ip=192.168.1.10
   ```

VXLAN tunnel command:

```
ovs-vsctl add-port ovsbr0 vxlan1 -- set Interface vxlan1 type=vxlan
options:remote_ip=192.168.1.10
```

Successful completion of these steps results in the two instances sharing a layer-2 network between separate hosts.

Report a bug

13.4.5. Configure the Networking Service Database Connection

The database connection string used by the networking service is defined in the **/etc/neutron/plugin.ini** file. It must be updated to point to a valid database server before starting the service.

Procedure 13.10. Configuring the OpenStack Networking SQL database connection

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

```
# openstack-config --set /etc/neutron/plugin.ini \
   DATABASE sql_connection mysql://USER:PASS@IP/DB
```

  Replace:

  - *USER* with the database user name the networking service is to use, usually **neutron**.
  - *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_neutron** was used as the example in the previous *Creating the OpenStack Networking Database* section).

Report a bug

13.4.6. Launch the Networking Service

Once the required settings are configured, you can now launch the Networking service (**neutron**) using the **service** command:

```
# service neutron-server start
```

Enable the Networking service permanently using the **chkconfig** command.

```
# chkconfig neutron-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/neutron/neutron.conf** file.

Report a bug

13.5. Configure the DHCP Agent

Prerequisites:

- Section 13.3, “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.

Procedure 13.11. Configuring the DHCP Agent

1. **Configuring Authentication**

   The DHCP agent 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/neutron/dhcp_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/neutron/dhcp_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/neutron/dhcp_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 *neutron*.

   ```
   # openstack-config --set /etc/neutron/dhcp_agent.ini \
      keystone_authtoken admin_user neutron
   ```

   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/neutron/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/neutron/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/neutron/dhcp_agent.ini \
      DEFAULT interface_driver neutron.agent.linux.interface.OVSInterfaceDriver
   ```

   B. **Linux Bridge Interface Driver**

   ```
   # openstack-config --set /etc/neutron/dhcp_agent.ini \
      DEFAULT interface_driver neutron.agent.linux.interface.BridgeInterfaceDriver
   ```

3. **Starting the DHCP Agent**

   a. Use the **service** command to start the **neutron-dhcp-agent** service.

   ```
   # service neutron-dhcp-agent start
   ```

   b. Use the **chkconfig** command to ensure that the **neutron-dhcp-agent** service will be started automatically in the future.

```
# chkconfig neutron-dhcp-agent on
```

The DHCP agent has been configured and started.

Report a bug

13.6. Create an External Network

Prerequisites:

- Section 13.3, “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-neutronclient* 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.

Procedure 13.12. Creating and configuring an external network

1. Use the **source** command to load the credentials of the administrative user.

   ```
   $ source ~/keystonerc_admin
   ```

2. Use the **net-create** action of the **neutron** command line client to create a new provider network.

   ```
   $ neutron net-create EXTERNAL_NAME \
      --router:external True \
      --provider:network_type TYPE \
      --provider:physical_network PHYSNET \
      --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 *PHYSNET* with a name for the physical network. This is not applicable if you intend to use a local network type. *PHYSNET* must match one of the values defined under **bridge_mappings** in the **/etc/neutron/plugin.ini** file.
   - 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.

```
$ neutron 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 specifed 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 **`neutron`** command line client to create a new router.

```
$ neutron 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 **`neutron`** command line client to link the newly created router to the external provider network.

```
$ neutron 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 **`neutron`** command line client to link the newly created router to each private network subnet.

```
$ neutron router-interface-add ROUTER SUBNET
```

Replace *ROUTER* with the unique identifier of the router, replace *SUBNET* with the unique identifier of a private network subnet. Perform this step for each existing private network subnet to which you wish to link the router.

An external provider network has been created. Use the unique identifier of the router when configuring the L3 agent.

Report a bug

13.7. Configuring the Plug-in Agent

13.7.1. Configure the Open vSwitch Plug-in Agent

Prerequisites:

- Section 13.3, “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.

Note

Open vSwitch support for TCP segmentation offload (TSO) and Generic Segmentation Offload (GSO) to VXLAN and GRE is enabled by default.

Procedure 13.13. Configuring the Open vSwitch Plug-in Agent

1. Confirm that the *openvswitch* package is installed. This is normally installed as a dependency of the *neutron-plugin-openvswitch* package.

   ```
   # rpm -qa | grep openvswitch
   openvswitch-1.10.0-1.el6.x86_64
   openstack-neutron-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
   ```

6. 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/neutron/plugin.ini \
   OVS bridge_mappings MAPPINGS
```

Replace *MAPPINGS* with the physical network to bridge mappings.

7. Use the **service** command to start the **neutron-openvswitch-agent** service.

```
#  service neutron-openvswitch-agent start
```

8. Use the **chkconfig** command to ensure that the **neutron-openvswitch-agent** service is started automatically in the future.

```
# chkconfig neutron-openvswitch-agent on
```

9. Use the **chkconfig** command to ensure that the **neutron-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 neutron-ovs-cleanup on
```

The networking configuration has been updated to use the Open vSwitch plug-in.

Report a bug

13.7.2. Configure the Linux Bridge Plug-in Agent

Prerequisites:

- Section 13.3, “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.

Procedure 13.14. Configuring the Linux Bridge plug-in agent

1. 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/neutron/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 **neutron-linuxbridge-agent** service.

```
# service neutron-linuxbridge-agent start
```

3. Use the **chkconfig** command to ensure that the **neutron-linuxbridge-agent** service is started automatically in the future.

   ```
   # chkconfig neutron-linuxbridge-agent on
   ```

The networking configuration has been updated to use the Linux Bridge plug-in.

Report a bug

13.8. Configure the L3 Agent

Prerequisites:

- Section 13.3, “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.

Procedure 13.15. Configuring the L3 Agent

1. **Configuring Authentication**

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

      ```
      # openstack-config --set /etc/neutron/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/neutron/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/neutron/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 *neutron*.

      ```
      # openstack-config --set /etc/neutron/metadata_agent.ini \
         keystone_authtoken admin_user neutron
      ```

   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/neutron/metadata_agent.ini \
         keystone_authtoken admin_password PASSWORD
      ```

2. **Configuring the Interface Driver**

   Set the value of the **interface_driver** configuration key in the **/etc/neutron/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/neutron/l3_agent.ini \
   DEFAULT interface_driver neutron.agent.linux.interface.OVSInterfaceDriver
```

B. **Linux Bridge Interface Driver**

```
# openstack-config --set /etc/neutron/l3_agent.ini \
   DEFAULT interface_driver neutron.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-ctl** 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/neutron/l3_agent.ini** file is **br-ex**. This ensures that the L3 agent will use the external bridge.

   ```
   # openstack-config --set /etc/neutron/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.

   The value of the **external_network_bridge** configuration key in the **/etc/neutron/l3_agent.ini** file must be blank. This ensures that the L3 agent does not attempt to use an external bridge.

   ```
   # openstack-config --set /etc/neutron/l3_agent.ini \
      DEFAULT external_network_bridge ""
   ```

4. **Starting the L3 Agent**

   a. Use the **service** command to start the **neutron-l3-agent** service.

   ```
   # service neutron-l3-agent start
   ```

   b. Use the **chkconfig** command to ensure that the **neutron-l3-agent** service will be started automatically in the future.

   ```
   # chkconfig neutron-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 **neutron-metadata-agent** service.

```
# service neutron-metadata-agent start
```

b. Use the **chkconfig** command to ensure that the **neutron-metadata-agent** service will be started automatically in the future.

```
# chkconfig neutron-metadata-agent on
```

The L3 agent has been configured and started.

6. **Enable leastrouter scheduling**

The leastrouter scheduler enumerates L3 Agent router assignment, and consequently schedules the router to the L3 Agent with the fewest routers. This differs from the ChanceScheduler behaviour, which randomly selects from the candidate pool of L3 Agents. Enable the leastrouter scheduler by changing the **router_scheduler_driver** option in the **neutron.conf** file:

```
router_scheduler_driver=neutron.scheduler.l3_agent_scheduler.LeastRoutersScheduler
```

The router is scheduled once connected to a network. Unschedule the router using the **neutron** command:

```
# neutron l3-agent-router-remove [l3 node] [router]
```

Assign the router with using the **neutron** command:

```
#  neutron l3-agent-router-add [l3 node] [router]
```

Report a bug

13.9. Validate 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 santiy checking of the OpenStack Networking deployment by following the steps outline in this procedure.

Procedure 13.16. Validating the OpenStack Networking installation

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 **neutron-server** service is running:

```
$ service neutron-server status
neutron-server (pid  3011) is running...
```

3. **Network Nodes**

   a. Ensure that the DHCP agent is running:

   ```
   $ service neutron-dhcp-agent status
   neutron-dhcp-agent (pid  3012) is running...
   ```

   b. Ensure that the L3 agent is running:

   ```
   $ service neutron-l3-agent status
   neutron-l3-agent (pid  3013) is running...
   ```

   c. Ensure that the plug-in agent, if applicable, is running:

   ```
   $ service neutron-PLUGIN-agent status
   neutron-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 neutron-metadata-agent status
   neutron-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.

Report a bug

13.10. Load Balancing as a Service (LBaaS) Overview

Load Balancing as a Service (LBaaS) allows Networking to distribute incoming requests evenly between designated instances. This ensures workload is shared predictably among instances, and allows more effective use of system resources. Incoming requests are distributed using these load balancing methods:

- **Round Robin** - Rotates requests evenly between multiple instances
- **Source IP** - Requests from a unique source IP address are consistently directed to the same instance
- **Least Connections** - Allocates requests to the instance with the least number of active connections

Monitors

LBaaS provides availability monitoring with **ping**, **TCP**, **HTTP get**, and **HTTPS get**. *Monitors* are implemented to determine whether pool members are available to handle requests.

Management

LBaaS is managed using a variety of toolsets. The *REST API* is available for programmatic administration and scripting. Administrative management is performed using the CLI commands and Dashboard's graphical user interface.

Connection Limits

Ingress traffic can be shaped with *Connection Limits*. This feature allows administrators to control workloads, and can also assist with mitigating DoS attacks.

Session Persistence

LBaaS supports session persistence by ensuring incoming requests are routed to the same instance within a pool of multiple instances. LBaaS supports routing decisions based on cookies and source IP address data.

Note

Third parties may provide drivers for their own load balancer products.

Report a bug

13.11. Configure Load Balancing as a Service (LBaaS)

This section outlines the steps for configuring Load Balancing as a Service (LBaas) with the Open vSwitch and Linux Bridge plug-ins. In addition, Load Balancing management options are enabled in Dashboard.

1. Enable the HAProxy plug-in using the **service_provider** parameter in the **/usr/share/neutron/neutron-dist.conf** file:

   ```
   service_provider =
   LOADBALANCER:Haproxy:neutron.services.loadbalancer.drivers.haproxy.plugin_driver.HaproxyOn
   HostPluginDriver:default
   ```

2. Enable the load balancer plugin using **service_plugin** in the **/etc/neutron/neutron.conf** file:

   ```
   service_plugins = neutron.services.loadbalancer.plugin.LoadBalancerPlugin
   ```

3. Enable the HAProxy load balancer in the **/etc/neutron/lbaas_agent.ini** file:

   ```
   device_driver =
   neutron.services.loadbalancer.drivers.haproxy.namespace_driver.HaproxyNSDriver
   ```

4. Enable the required driver in the**/etc/neutron/lbaas_agent.ini** file:

   Select the Open vSwitch driver for LBaaS:

   ```
   interface_driver = neutron.agent.linux.
   interface.OVSInterfaceDriver
   ```

   or select the Linux Bridge driver for LBaaS:

   ```
   interface_driver = neutron.agent.linux.interface.BridgeInterfaceDriver
   ```

   Apply the new settings by restarting the **neutron-server** and **neutron-lbaas-agent** services.

5. Enable Load Balancing in the **Project** section of the Dashboard user interface:

   Toggle the **enable_lb** feature to **True** in the **/etc/openstack-dashboard/local_settings** file:

   ```
   OPENSTACK_NEUTRON_NETWORK = {
   'enable_lb': True,
   ```

   Apply the new settings by restarting the **httpd** service. The Load Balancer management options are now available in Dashboard's **Project** view.

Report a bug

13.12. Troubleshoot OpenStack Networking Issues

This section discusses the different commands you can use and procedures you can follow to troubleshoot the OpenStack Networking service issues.

Debugging Networking Device

- Use the **ip a** command to display all the physical and virtual devices.

- Use the **`ovs-vsctl show`** command to diplay the interfaces and bridges in a virtual switch.
- Use the **`ovs-dpctl show`** command to show datapaths on the switch.

Tracking Networking Packets

- Use the **`tcpdump`** command to see where packets are not getting through.

  ```
  # tcpdump -n -i INTERFACE -e -w FILENAME
  ```

  Replace *INTERFACE* with the name of the network interface to see where the packets are not getting through. The interface name can be the name of the bridge or host Ethernet device.

  The **`-e`** flag ensures that the link-level header is dumped (in which the vlan tag will appear).

  The **`-w`** flag is optional. You can use it only if you want to write the output to a file. If not, the output is written to the standard output (**`stdout`**).

  For more information about **`tcpdump`**, refer to its manual page by running **`man tcpdump`**.

- Use the **`iptables`** command to check the iptables rules.

Debugging Network Namespaces

- Use the **`ip netns list`** command to list all known network namespaces.
- Use the **`ip netns exec`** command to show routing tables inside specific namespaces.

  ```
  # ip netns exec NAMESPACE_ID bash
  # route -n
  ```

  Start the **`ip netns exec`** command in a bash shell so that subsequent commands can be invoked without the **`ip netns exec`** command.

Report a bug

Chapter 14. Installing the OpenStack Compute Service

14.1. Compute Service Requirements

14.1.1. Check 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.

Procedure 14.1. Verifying hardware virtualization support

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.

Report a bug

14.2. Installing a Compute VNC Proxy

14.2.1. Install 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 (through the **openstack-nova-xvpvncproxy** 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.

Procedure 14.2. Installing the Compute VNC proxy packages

- 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.

Report a bug

14.2.2. Configure the Firewall to Allow Compute VNC Proxy Traffic

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-xvpvncproxy** 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.

Procedure 14.3. Configuring the firewall to allow Compute VNC proxy traffic

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-xvpvncproxy** 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.

Report a bug

14.2.3. Configure the VNC Proxy Service

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
```

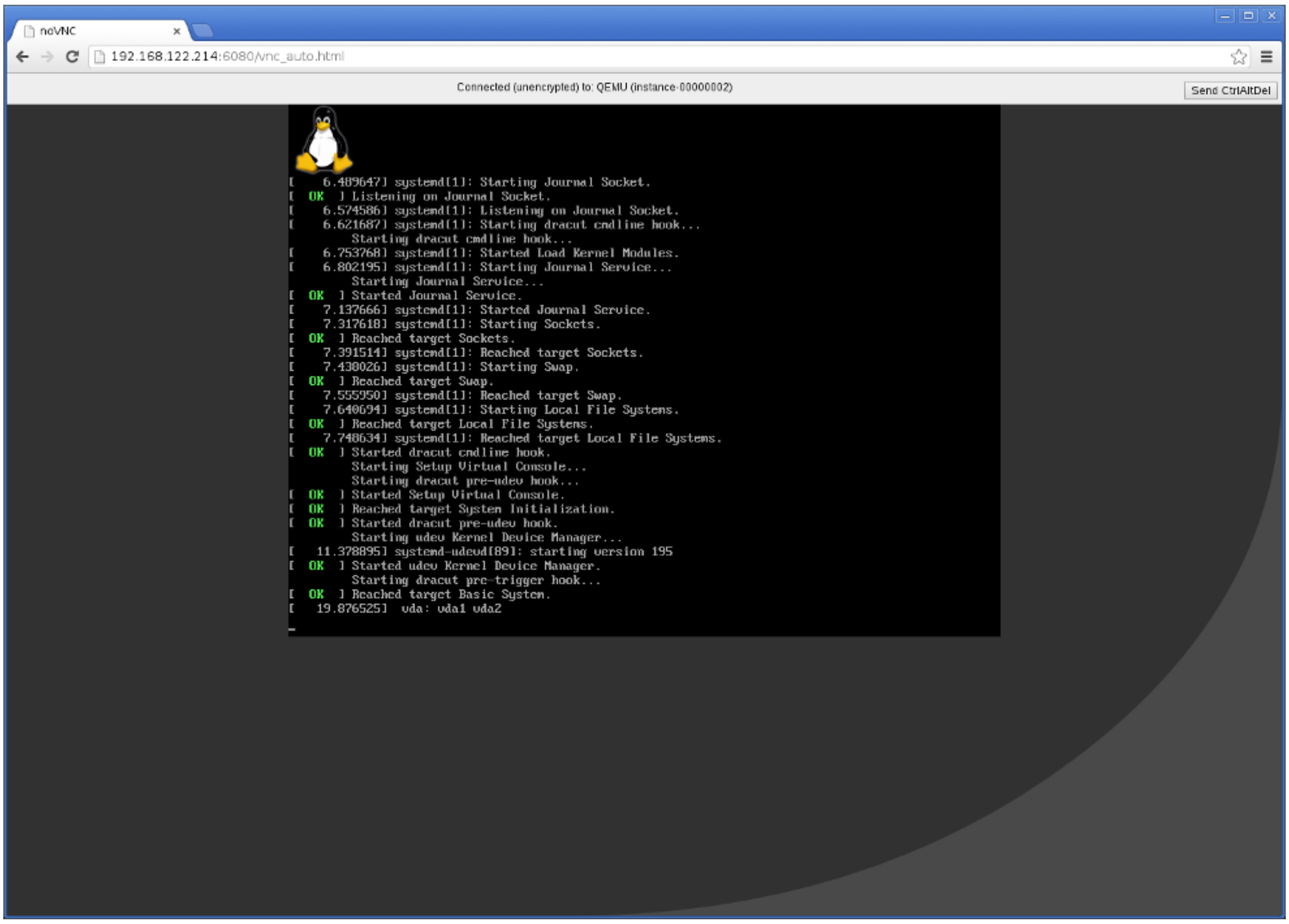
To control access to the VNC service that uses a traditional client (non browser-based), substitute *openstack-nova-xvpvncproxy* into the previous commands.

Report a bug

14.2.4. Access Instances with the Compute VNC Proxy

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.



Connected (unencrypted) to: QEMU (instance-00000002)

```
[    6.489647] systemd[1]: Starting Journal Socket.
[  OK  ] Listening on Journal Socket.
[    6.574586] systemd[1]: Listening on Journal Socket.
[    6.621687] systemd[1]: Starting dracut cmdline hook...
         Starting dracut cmdline hook...
[    6.753768] systemd[1]: Started Load Kernel Modules.
[    6.802195] systemd[1]: Starting Journal Service...
         Starting Journal Service...
[  OK  ] Started Journal Service.
[    7.137666] systemd[1]: Started Journal Service.
[    7.317618] systemd[1]: Starting Sockets.
[  OK  ] Reached target Sockets.
[    7.391514] systemd[1]: Reached target Sockets.
[    7.438026] systemd[1]: Starting Swap.
[  OK  ] Reached target Swap.
[    7.555950] systemd[1]: Reached target Swap.
[    7.640694] systemd[1]: Starting Local File Systems.
[  OK  ] Reached target Local File Systems.
[    7.748634] systemd[1]: Reached target Local File Systems.
[  OK  ] Started dracut cmdline hook.
         Starting Setup Virtual Console...
         Starting dracut pre-udev hook...
[  OK  ] Started Setup Virtual Console.
[  OK  ] Reached target System Initialization.
[  OK  ] Started dracut pre-udev hook.
         Starting udev Kernel Device Manager...
[   11.378895] systemd-udevd[89]: starting version 195
[  OK  ] Started udev Kernel Device Manager.
         Starting dracut pre-trigger hook...
[  OK  ] Reached target Basic System.
[   19.876525]  vda: vda1 vda2
_
```

Figure 14.1. VNC instance access

Report a bug

14.3. Installing a Compute Node

14.3.1. Create 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.

Procedure 14.4. Creating the Compute Service database

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.

   ```
   mysql> quit
   ```

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

Report a bug

14.3.2. Configure Compute Service Authentication

This section outlines the steps for creating and configuring Identity service records required by the Compute 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. In order to proceed, you should have already performed the following (through the Identity service):

1. Created an Administrator role named **admin** (refer to Section 9.7, “Create an Administrator Account” for instructions)
2. Created the **services** tenant (refer to Section 9.9, “Create the Services Tenant” for instructions)

Note

The *Installation and Configuration Guide* uses one tenant for all service users. For more information, refer to Section 9.9, “Create the Services Tenant”.

You can perform the following procedure from your Identity service host 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 14.5. Configuring the Compute Service to authenticate through the Identity 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 **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.

3. Use the **keystone user-role-add** command to link the **compute** user, **admin** role, and **services** tenant together:

   ```
   # keystone user-role-add --user compute --role admin --tenant services
   ```

4. 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              |
   +-------------+----------------------------------+
   ```

5. Create the **compute** endpoint entry:

   ```
   # keystone endpoint-create \
           --service compute
           --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 *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.

Report a bug

14.3.3. Install the Compute Service Packages

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 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.

Report a bug

14.3.4. Configure the Compute Service to Use SSL

Use the following options in the **nova.conf** file to configure SSL.

Table 14.1. SSL options for Compute

Configuration Option	Description
enabled_ssl_apis	A list of APIs with enabled SSL.
ssl_ca_file	CA certificate file to use to verify connecting clients.
ssl_cert_file	SSL certificate of API server.
ssl_key_file	SSL private key of API server.
tcp_keepidle	Sets the value of TCP_KEEPIDLE in seconds for each server socket. Defaults to 600.

Report a bug

14.3.5. Configuring the Compute Service

14.3.5.1. Configure Compute Service 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.

Procedure 14.6. Configuring the Compute Service to authenticate through the Identity Service

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.

Report a bug

14.3.5.2. Configure the Compute Service Database Connection

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.

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 do not 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.

Procedure 14.7. Configuring the Compute Service SQL database connection

- 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.

Report a bug

14.3.5.3. Configure Message Broker Settings for the Compute Service

Prerequisites:

- Section 8.2, “Configuring the Message Broker”

This section assumes that you have already configured a Qpid Message Broker. For more information, refer to:

- Section 8.1, “Install the Message Broker Packages”
- Section 8.2.1, “Basic Qpid Configuration”
- Section 8.2.2.1, “Simple Authentication and Security Layer (SASL)”

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:

Procedure 14.8. Configuring the Message Broker settings of the Compute Service

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.

Note

The Compute service Qpid user is configured as part of Section 8.2.2.4, "Configure SASL Using a Local Password File".

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.

See Also:

- Section 8.2.2.4, "Configure SASL Using a Local Password File"

Report a bug

14.3.5.4. Configure Resource Overcommitment

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.

Report a bug

14.3.5.5. Reserve Host Resources

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.

Report a bug

14.3.5.6. Configuring Compute Networking

14.3.5.6.1. 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 these nodes to run OpenStack Network. Problems can arise from inadvertently running the **nova-network** process while using OpenStack Networking service; for example, a previously running **nova-network** could push down stale **iptables** rules.

Report a bug

14.3.5.6.2. Update the Compute Configuration

Each time a Compute instance is provisioned or deprovisioned, the service communicates with OpenStack Networking through 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.

Procedure 14.9. Updating the connection and authentication settings of Compute nodes

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.neutronv2.api.API
   ```

2. Set the value of the **neutron_url** configuration key to point to the endpoint of the networking API.

   ```
   # openstack-config --set /etc/nova/nova.conf \
      DEFAULT neutron_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 **neutron_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 neutron_admin_tenant_name services
```

4. Set the value of the **neutron_admin_username** configuration key to the name of the administrative user for the OpenStack Networking service. Examples in this guide use *neutron*.

```
# openstack-config --set /etc/nova/nova.conf \
   DEFAULT neutron_admin_username neutron
```

5. Set the value of the **neutron_admin_password** configuration key to the password associated with the administrative user for the networking service.

```
# openstack-config --set /etc/nova/nova.conf \
   DEFAULT neutron_admin_password PASSWORD
```

6. Set the value of the **neutron_admin_auth_url** configuration key to the URL associated with the Identity service endpoint.

```
# openstack-config --set /etc/nova/nova.conf \
  DEFAULT neutron_admin_auth_url http://IP:35357/v2.0
```

   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 **neutron**.

```
# openstack-config --set /etc/nova/nova.conf \
  DEFAULT security_group_api neutron
```

   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.

Report a bug

14.3.5.6.3. Configure the L2 Agent

Each compute node must run an instance of the Layer 2 (L2) agent appropriate to the networking plug-in that is in use.

- Section 13.7.1, “Configure the Open vSwitch Plug-in Agent”
- Section 13.7.2, “Configure the Linux Bridge Plug-in Agent”

Report a bug

14.3.5.6.4. Configure Virtual Interface Plugging

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",
]
```

Report a bug

14.3.5.7. Configure the Firewall to Allow Compute Service Traffic

Connections to virtual machine consoles, whether direct or through the proxy, are received on ports **5900** to **5999**.

To allow this, the firewall on the service node must be configured to allow network traffic on these ports. Log in as the **root** user to the server hosting the Compute service and perform the following procedure:

Procedure 14.10. Configuring the firewall to allow Compute Service traffic

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** by adding the following line to the file.

   ```
   -A INPUT -p tcp -m multiport --dports 5900:5999 -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. Repeat this process for each compute node.

Report a bug

14.3.6. Populate the Compute Service Database

You can populate the Compute Service database after you have successfully configured the Compute Service database connection string (refer to Section 14.3.5.2, “Configure the Compute Service Database Connection”).

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.

Procedure 14.11. Populating the Compute Service database

1. Log in to a system hosting an instance of the **openstack-nova-conductor** service.
2. Use the **su** command to switch to the **nova** user.

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

3. 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.

Report a bug

14.3.7. Launch the Compute Services

Procedure 14.12. Launching Compute services

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 through 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.

Note

If you intend to use the EC2 API to the Compte service, you need to set the options in the **`nova.conf`** configuration file. For more information, see *Configuring the EC2 API* section in the *Red Hat Enterprise Linux OpenStack Platform Configuration Reference Guide*. This document is available from the following link:

https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform

`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.

Report a bug

Chapter 15. Installing the OpenStack Telemetry Service

15.1. Overview of Telemetry Service Deployment

The OpenStack Telemetry service is composed of an API server, three *openstack-ceilometer* agents, and two alarm services. The API Server (provided by the *openstack-ceilometer-api*) runs on one or more central management servers to provide access to the back-end repository.

Note

At present, **`mongod`** is the only back-end repository supported by the Telemetry service.

The three Telemetry agents (and their respective packages) are:

- Central agent (provided by *openstack-ceilometer-central*): runs on a central management server to poll public REST APIs for utilization statistics about resources that are not visible (either through notifications or from the hypervisor layer)
- Collector (provided by *openstack-ceilometer-collector*): runs on one or more central management servers to receive notifications on resource usage. The Collector also parses resource usage statistics and saves them as datapoints in the Telemetry store.
- Compute agent (provided by *openstack-ceilometer-compute*): runs on each Compute service node to poll for instance utilization statistics. You must install and configure the Compute service first before installing the *openstack-ceilometer-compute* on any node.

The two alarm services (and their respective packages) that comprise the rest of the Telemetry service are:

- Evaluator (provided by *ceilometer-alarm-evaluator*): triggers state transitions on alarms.
- Notifier (provided by *ceilometer-alarm-notifier*): executes required actions when alarms are triggered.

You can deploy the API Server, Central agent, data store service, and Collector on different hosts. In addition, each Compute node must also have a Compute agent installed; this agent gathers detailed usage metrics on instances running on the Compute node.

Each of these components must have the following settings configured:

- Authentication, as in Identity service tokens and Telemetry secret
- Database connection URL, for connecting to the Telemetry store

The authentication settings (for example, Identity credentials) and database connection string for these components are all configured in **`/etc/ceilometer/ceilometer.conf`**. As such, components deployed on the same host will share the same settings.

Conversely, if Telemetry components are deployed on multiple hosts then you will have to replicate any authentication changes to these hosts. To do so, you can copy the **`/etc/ceilometer/ceilometer.conf`** file across hosts after applying the new settings.

Once the Telemetry service (as in, all of its components, wherever each is hosted) is deployed and configured, you will need to configure each monitored service (Image, Networking, Object Storage, Block Storage, and each Compute node) to submit data to the Telemetry service. The related settings are configured in each service's configuration file.

See Also:

- Section 1.3.8, “Telemetry Service”

Report a bug

15.2. Install the Telemetry Service Packages

The OpenStack Telemetry service requires the following packages:

mongodb-server

Provides the MongoDB database server. The Telemetry service uses **mongodb** as its back-end data repository.

openstack-ceilometer-api

Provides the **`ceilometer`** API Server.

openstack-ceilometer-central

Provides the Central **`ceilometer`** agent.

openstack-ceilometer-collector

Provides the **`ceilometer`** Collector agent.

openstack-ceilometer-common

Provides components common to all **`ceilometer`** services.

openstack-ceilometer-compute

Provides the **`ceilometer`** agent that should run on each Compute node.

openstack-ceilometer-alarm

Provides the **`ceilometer`** alarm notification and evaluation services.

python-ceilometer

Provides the **`ceilometer`** python library.

python-ceilometerclient

Provides the **`ceilometer`** command-line tool and a Python API (specifically, the **`ceilometerclient`** module).

To install these required packages on the same host, run the following command as the **`root`** user:

```
# yum install -y mongodb-server openstack-ceilometer-* python-ceilometer python-
ceilometerclient
```

You can deploy the API Server, Central agent, data store service, and Collector on different hosts. In this case, log in to a host and install the corresponding package of the component you wish to deploy there:

```
# yum install -y PACKAGE
```

Replace *PACKAGE* with the corresponding package of the component you wish to install on the host. All nodes hosting services that you wish to monitor with the Telemetry services must also have the *python-ceilometer* and *python-ceilometerclient* packages installed (see Section 15.7, “Configure Monitored Services” for more information).

Report a bug

15.3. Create the Telemetry Identity Records

Prerequisites:

- Section 23.1, “Defining Regions”
- Section 9.9, “Create the Services Tenant”
- Section 9.7, “Create an Administrator Account”

In this section, you will:

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

In order to proceed, you should have already created the **`services`** tenant (refer to Section 9.9, “Create the Services Tenant” for instructions).

1. Created an Administrator role named **admin** (refer to Section 9.7, "Create an Administrator Account" for instructions)
2. Created the **services** tenant (refer to Section 9.9, "Create the Services Tenant" for instructions)

Note

The *Installation and Configuration Guide* uses one tenant for all service users. For more information, refer to Section 9.9, "Create the Services Tenant".

You can perform this procedure from your Identity service host or on any machine where you've copied the **keystonerc_admin** file (which contains administrator credentials) and the **keystone** command-line utility is installed. For more information about the **keystonerc_admin** file, refer to Section 9.7, "Create an Administrator Account".

Procedure 15.1. Creating Identity records for the Telemetry Service

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

```
# source ~/keystonerc_admin
```

2. Create a **ceilometer** user using the following command:

```
# keystone user-create --name=ceilometer \
--pass=SERVICE_PASSWORD \
--email=CEILOMETER_EMAIL
```

   Where:

   - *SERVICE_PASSWORD* is the password the Telemetry service should use when authenticating with the Identity service.
   - *CEILOMETER_EMAIL* is the email address used by the Telemetry service.

3. Create a **ResellerAdmin** role:

```
# keystone role-create --name=ResellerAdmin
```

4. Establish the relationship between the Telemetry service, the **services** tenant, and **ResellerAdmin** role:

```
# keystone user-role-add --user ceilometer
    --role ResellerAdmin
    --tenant services
```

5. Establish the relationship between the Telemetry service, the **services** tenant, and **admin** role:

```
# keystone user-role-add --user ceilometer
    --role admin
    --tenant services
```

6. Create the **ceilometer** service entry:

```
# keystone service-create --name=ceilometer \
  --type=metering \
  --description="OpenStack Telemetry Service"
+------------+--------------------------------+
|   Property  |              Value              |
+------------+--------------------------------+
| description |     OpenStack Telemetry Service   |
| id          | a511aea8bc1264641f4dff1db38751br |
| name        |            ceilometer           |
| type        |            metering             |
+------------+--------------------------------+
```

7. Create the **ceilometer** endpoint entry:

   ```
   # keystone endpoint-create \
    --service ceilometer \
    --publicurl "IP:8777"
    --adminurl "IP:8777"
    --internalurl "IP:8777"
   ```

   Replace *IP* with the IP address or host name of the system hosting the Telemetry service.

   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. See Section 23.1, “Defining Regions” for more information.

Report a bug

15.4. Configure Telemetry Service Authentication

After creating and configuring the required Telemetry service users and roles (namely, Identity records), configure the Telemetry API service (**openstack-ceilometer-api**) to authenticate with the Identity service. Doing so involves setting the required Identity credentials in the **/etc/ceilometer/ceilometer.conf** configuration file.

To configure the required Identity credentials for the Telemetry API service, perform the following procedure.

Procedure 15.2. Configuring the Telemetry Service to authenticate through the Identity Service

1. Set the Telemetry service's authentication host (**auth_host**) configuration key:

   ```
   # openstack-config --set /etc/ceilometer/ceilometer.conf \
     keystone_authtoken auth_host IP
   ```

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

2. Set the Telemetry service's authentication port (**auth_port**) configuration key:

   ```
   # openstack-config --set /etc/ceilometer/ceilometer.conf \
     keystone_authtoken auth_port PORT
   ```

   Replace *PORT* with the authentication port used by the Identity server.

3. Set the Telemetry service to use the **http** protocol for authenticating:

   ```
   # openstack-config --set /etc/ceilometer/ceilometer.conf \
     keystone_authtoken auth_protocol PORT
   ```

4. Set the Telemetry service to authenticate as the correct tenant:

   ```
   # openstack-config --set /etc/ceilometer/ceilometer.conf \
     keystone_authtoken admin_tenant_name services
   ```

   Where *services* is the name of the tenant created for the use of the Telemetry service. Examples in this guide use *services*.

5. Set the Telemetry service to authenticate using the **ceilometer** administration user account:

   ```
   # openstack-config --set /etc/ceilometer/ceilometer.conf \
     keystone_authtoken admin_user ceilometer
   ```

6. Set the Telemetry service to use the correct **ceilometer** administration user account password:

   ```
   # openstack-config --set /etc/ceilometer/ceilometer.conf \
     keystone_authtoken admin_password SERVICE_PASSWORD
   ```

   Where *SERVICE_PASSWORD* is the password set during the creation of the **ceilometer** user.

7. Set the Telemetry secret. This is a string used to help secure communication between all components of the Telemetry service across multiple hosts (for example, between the Collector agent and a Compute node agent). To set the Telemetry secret:

   ```
   # openstack-config --set /etc/ceilometer/ceilometer.conf \
   publisher_rpc metering_secret SECRET
   ```

   Replace *SECRET* with the string that all Telemetry service components should use to sign and verify messages that are sent or received over AMQP.

After configuring the credentials of the Telemetry API service, configure the service endpoints to be used by the Central agent, Compute agents, and Alarm Evaluator. To do so, run the following commands on each host where these components are deployed:

```
# openstack-config --set /etc/ceilometer/ceilometer.conf \
   DEFAULT os_auth_url http://IP:35357/v2.0
# openstack-config --set /etc/ceilometer/ceilometer.conf \
   DEFAULT os_username ceilometer
# openstack-config --set /etc/ceilometer/ceilometer.conf \
   DEFAULT os_tenant_name services
# openstack-config --set /etc/ceilometer/ceilometer.conf \
   DEFAULT os_password SERVICE_PASSWORD
```

Where:

- *IP* is the IP address or host name of the Identity server.
- *SERVICE_PASSWORD* is the password set during the creation of the **ceilometer** user.

Report a bug

15.5. Setting the Telemetry Service Database Connection URL

The database connection URL used by the Telemetry service is defined in the **/etc/ceilometer/ceilometer.conf** file. This URL must be set to point to a valid database server before launching the Telemetry API server (**openstack-ceilometer-api**) and Collector agent (**openstack-ceilometer-collector**). The Telemetry service uses this database server as its back-end data repository; by default, the Telemetry service uses MongoDB.

To set the database connection string, run the following command as the **root** user on the system hosting the **openstack-ceilometer-api** and **openstack-ceilometer-collector** services:

```
# openstack-config --set /etc/ceilometer/ceilometer.conf \
  database connection DB_TYPE://USER:PASS@IP:PORT/ceilometer
```

Where:

- *DB_TYPE* is the database type of the database server (in this case, **mongodb**).
- *USER* and *PASS* are the username and password required by the Telemetry service to log on to the database server. Supply these credentials only when required by the database server (for example, when the database server is hosted on another system or node).
- *IP:PORT* is the IP address/hostname and port of the system hosting the database server.

If MongoDB is hosted locally on the same host, the required database connection string would be:

```
mongodb://localhost:27017/ceilometer
```

As in:

```
# openstack-config --set /etc/ceilometer/ceilometer.conf \
 database connection mongodb://localhost:27017/ceilometer
```

Report a bug

15.5.1. Configure and Launch the MongoDB Back-End

The Telemetry service uses MongoDB as its back-end data repository. As such, you will need to start the **mongod** service.

Before doing so, you may want to configure **mongod** to run with the **--smallfiles** parameter. This parameter onfigures MongoDB to use a smaller default data file and journal size. With this, MongoDB will limit the size of each data file, creating and writing to a new one when it reaches 512MB.

You can configure **mongod** to run with this parameter using **/etc/sysconfig/mongod**. To do so, specify **--smallfiles** in the **OPTIONS** section of **/etc/sysconfig/mongod**, as in:

```
OPTIONS="--smallfiles /etc/mongodb.conf"
```

MongoDB will use all parameters specified in the **OPTIONS** section when **mongod** launches. To start the MongoDB service:

```
# service mongod start
```

Report a bug

15.6. Configure the Compute Node

Prerequisites:

- Section 15.4, “Configure Telemetry Service Authentication”
- Section 14.3, “Installing a Compute Node”

The Telemetry service monitors each node by collecting usage data from the Compute agent (**openstack-ceilometer-compute**) installed on that node. You can configure a node's Compute agent by replicating the **/etc/ceilometer/ceilometer.conf** from another host (as in, a host whose Telemetry components have already been configured).

You will also have to configure the Compute node itself to enable notifications. This is set in the Compute agent's configuration file, namely **/etc/nova/nova.conf**:

Procedure 15.3. Enabling notifications on a Compute node

1. Install *python-ceilometer* and *python-ceilometerclient* on the node:

   ```
   # yum install python-ceilometer python-ceilometerclient
   ```

2. Enable auditing on the node:

   ```
   # openstack-config --set /etc/nova/nova.conf \
   DEFAULT instance_usage_audit True
   ```

3. Configure the audit frequency:

   ```
   # openstack-config --set /etc/nova/nova.conf \
   DEFAULT instance_usage_audit_period hour
   ```

4. Configure what type of state changes should trigger a notification:

   ```
   # openstack-config --set /etc/nova/nova.conf \
   DEFAULT notify_on_state_change vm_and_task_state
   ```

5. Set the node to use the correct notification drivers:

```
# openstack-config --set /etc/nova/nova.conf \
DEFAULT notification_driver nova.openstack.common.notifier.rpc_notifier
```

```
# openstack-config --set /etc/nova/nova.conf \
DEFAULT notification_driver ceilometer.compute.nova_notifier
```

Once the Compute node is configured for Telemetry, start/restart the Compute agent:

```
# service openstack-ceilometer-compute restart
```

Configure the agent to launch automatically at boot:

```
# chkconfig openstack-ceilometer-compute on
```

Finally, restart the **openstack-nova-compute** service to apply all changes to **/etc/nova/nova.conf**:

```
# service openstack-nova-compute restart
```

Report a bug

15.7. Configure Monitored Services

The Telemetry service can also monitor the Image, Networking, Object Storage, and Block Storage service. To enable this, you will need to configure each service to submit samples to the Collector services. The following commands demonstrate how to do this for each service. Before configuring any of these services, install the *python-ceilometer* and *python-ceilometerclient* packages on the node hosting the service first:

```
# yum install python-ceilometer python-ceilometerclient
```

Note

Restart a service after configuring it to be monitored by the Telemetry service. Doing so will allow the configuration to take effect.

Image service (glance)

```
# openstack-config --set /etc/glance/glance-api.conf \
 DEFAULT notifier_strategy NOTIFYMETHOD
```

Replace *NOTIFYMETHOD* with a notification queue: **rabbit** (to use a **rabbitmq** queue) or **qpid** (to use a **qpid** message queue).

Block Storage service (cinder)

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

Object Storage service (swift)

The Telemetry service collects samples from the Object Storage service (**swift**) through the **ResellerAdmin** role. The steps to configure this should already have been performed when configuring the required Identity records for Telemetry.

In addition, you will also need to configure the Object Storage service to process traffic from **ceilometer**. To do so:

Procedure 15.4. Configuring the Object Storage service to process traffic from the Telemetry Service

1. Add the following line to **/etc/swift/proxy-server.conf**:

   ```
   [filter:ceilometer]
   use = egg:ceilometer#swift
   ```

2. Add **ceilometer** to the **pipeline** directive of the same file:

   ```
   [pipeline:main]
   pipeline = healthcheck cache authtoken keystoneauth proxy-server ceilometer
   ```

Networking service (neutron)

```
# openstack-config --set /etc/neutron/neutron.conf \
 DEFAULT notification_driver neutron.openstack.common.notifier.rpc_notifier
```

Report a bug

15.8. Launch the Telemetry API and Agents

Once the authentication settings and database connection strings have been configured for each component on each host, you can now launch the Telemetry service. To do so, launch the corresponding service of each component:

```
# service SERVICENAME start
```

Then, configure each service to launch automatically at boot:

```
# chkconfig SERVICENAME on
```

Replace *SERVICENAME* with the corresponding name of each Telemetry component service:

Table 15.1. Telemetry Component Service Names

Component	Service Name
openstack-ceilometer-compute	Compute agent (runs on each Compute node)
openstack-ceilometer-central	Central **ceilometer** agent
openstack-ceilometer-collector	Collector agent
openstack-ceilometer-api	API Server
openstack-ceilometer-alarm-evaluator	Evaluator (triggers alarm state transitions)
openstack-ceilometer-alarm-notifier	Notifier (executes alarm actions)

Report a bug

Chapter 16. Installing the OpenStack Orchestration Service

16.1. Install the Orchestration Service Packages

The OpenStack Orchestration service requires the following packages:

openstack-heat-api

Provides the OpenStack-native ReST API to the **heat** Engine.

openstack-heat-api-cfn

Provides the AWS CloudFormation-compatible API to the **heat** Engine.

openstack-heat-common

Provides components common to all **heat** services.

openstack-heat-engine

Provides the OpenStack API for launching templates and submitting events back to the API.

openstack-heat-api-cloudwatch

Provides the AWS CloudWatch-compatible API to the **heat** Engine.

heat-cfntools

Provides the tools required on **heat**-provisioned cloud instances.

python-heatclient

Provides a Python API and command-line script, both of which make up a client for the OpenStack **heat** API.

openstack-utils

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

To install these required packages, run the following command as the **root** user:

```
# yum install -y openstack-heat-* python-heatclient openstack-utils
```

Report a bug

16.2. Configuring the Orchestration Service

To configure the Orchestration service, you will need to:

- Configure a database for the Orchestration service.
- Bind each Orchestration API service to a corresponding IP address.
- Create and configure the Orchestration service Identity records.
- Configure how Orchestration services authenticate with the Identity service.

The following sections describe each procedure in detail.

Report a bug

16.2.1. Configure the Orchestration Service Database Connection

The OpenStack Orchestration service requires a database in order to work. The database connection settings used by this service are configured in **/etc/heat/heat.conf**.

After using the MySQL **root** account to automatically create the required databases, the Orchestration service will use its own MySQL account (specifically, **heat**) to use those databases.

Note

In order to connect to the MySQL database, create a database, and configure those database, you will need the password of the MySQL database **root** account.

The following procedure illustrates this in more detail:

Procedure 16.1. Configuring the Orchestration Service SQL database connection

1. Connect to the database service as **root** using the **mysql** utility.

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

2. After logging in, create the **heat** database:

```
mysql> CREATE DATABASE heat;
```

3. Create a database user named **heat** and grant that user access to the newly-created **heat** database:

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

   Replace *HEATPW* with a secure password that the **heat** user should use to authenticate to the database server.

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

```
mysql> FLUSH PRIVILEGES;
```

5. Exit the **mysql** utility:

```
mysql> quit
```

6. Update the database connection string used by **heat** with the *HEATPW* used earlier:

```
# openstack-config --set /etc/heat/heat.conf \
 DEFAULT sql_connection mysql://heat:HEATPW@MYSQLHOST/heat
```

   Where:

   - *HEATPW* is the password set earlier for the **heat** account.
   - *MYSQLHOST* is the IP address or hostname of the MySQL database server (**localhost** if it is on the same host).

7. As the **heat** user, sync the database using the **heat-manage** utility:

```
# runuser -s /bin/sh heat -c "heat-manage db_sync"
```

Report a bug

16.2.2. Restrict the Bind Addresses of each Orchestration API Service

After configuring the database, set the **bind_host** setting of each Orchestration API service. This setting controls which IP address a service should use for incoming connections.

The **/etc/heat/heat.conf** configuration file contains a section for each Orchestration API service. the **bind_host** should be set in each one, as in:

```
# openstack-config --set /etc/heat/heat.conf
 heat_api bind_host IP
```

```
# openstack-config --set /etc/heat/heat.conf
 heat_api_cfn bind_host IP
```

```
# openstack-config --set /etc/heat/heat.conf
 heat_api_cloudwatch bind_host IP
```

Replace each *IP* with the address that the corresponding API should use.

Report a bug

16.2.3. Create the Orchestration Service Identity Records

In this procedure, you will:

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

In order to proceed, you should have already performed the following (through the Identity service):

1. Created an Administrator role named **admin** (refer to Section 9.7, “Create an Administrator Account” for instructions)
2. Created the **services** tenant (refer to Section 9.9, “Create the Services Tenant” for instructions)

Note

The *Installation and Configuration Guide* uses one tenant for all service users. For more information, refer to Section 9.9, “Create the Services Tenant”.

You can perform this procedure 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 16.2. Creating Identity records for the Orchestration Service

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

   ```
   # source ~/keystonerc_admin
   ```

2. Create a **heat** user in **keystone**:

   ```
   # keystone user-create \
      --name=heat \
      --pass=SERVICE_PASSWORD \
   ```

   Replace *SERVICE_PASSWORD* with the password the Orchestration service should use when authenticating with the Identity service.

3. Establish the relationship between the Orchestration service, the **services** tenant, and **admin** role:

   ```
   # keystone user-role-add --user heat \
      --role admin \
      --tenant services \
   ```

4. Create Identity service entries for Orchestration and Cloud Formation:

   ```
   # keystone service-create --name heat \
      --type orchestration
   ```

```
# keystone service-create --name heat-cfn \
   --type cloudformation
```

Once the **heat** and **heat-cfn** services are created, note their respective IDs. These will be used later.

5. Create service endpoint entries for both **heat** and **heat-cfn** services:

```
# keystone endpoint-create \
   --service heat-cfn \
   --publicurl "HEAT_CFN_IP:8000/v1" \
   --adminurl "HEAT_CFN_IP:8000/v1" \
   --internalurl "HEAT_CFN_IP:8000/v1"
```

```
# keystone endpoint-create \
   --service heat \
   --publicurl "HEAT_IP:8004/v1/%(tenant_id)s" \
   --adminurl "HEAT_IP:8004/v1/%(tenant_id)s" \
   --internalurl "HEAT_IP:8004/v1/%(tenant_id)s"
```

Where:

- *HEAT_CFN_IP* is the IP or host name of the system hosting the **heat-cfn** service.
- *HEAT_IP* is the IP or host name of the system hosting the **heat** service.

Important

Include the **http://** prefix for *HEAT_CFN_IP* and *HEAT_IP* values.

Report a bug

16.2.4. Configure Orchestration Service Authentication

After creating and configuring the required Orchestration service users and roles (namely, Identity records), configure the Orchestration service to authenticate with the Identity service. Doing so involves setting the required Identity tokens for the **heat-api**, **heat-api-cfn**, and **heat-api-cloudwatch** services. These services use the token settings configured in **/etc/heat/heat.conf**.

Procedure 16.3. Configuring the Orchestration Service to authenticate through the Identity Service

1. Set the Orchestration services to authenticate as the correct tenant:

```
# openstack-config --set /etc/heat/heat.conf \
 keystone_authtoken admin_tenant_name services
```

Where *services* is the name of the tenant created for the use of the Orchestration service. Examples in this guide use *services*.

2. Set the Orchestration services to authenticate using the **heat** administration user account:

```
# openstack-config --set /etc/heat/heat.conf \
 keystone_authtoken admin_user heat
```

3. Set the Orchestration services to use the correct **heat** administration user account password:

```
# openstack-config --set /etc/heat/heat.conf \
 keystone_authtoken admin_password SERVICE_PASSWORD
```

Where *SERVICE_PASSWORD* is the password set during the creation of the **heat** user.

4. Set the correct Identity service host that the Orchestration services should use:

```
# openstack-config --set /etc/heat/heat.conf \
 keystone_authtoken service_host KEYSTONE_HOST
```

```
# openstack-config --set /etc/heat/heat.conf \
 keystone_authtoken auth_host KEYSTONE_HOST
```

```
# openstack-config --set /etc/heat/heat.conf \
 keystone_authtoken auth_uri http://KEYSTONE_HOST:35357/v2.0 \
```

```
# openstack-config --set /etc/heat/heat.conf \
 keystone_authtoken keystone_ec2_uri http://KEYSTONE_HOST:35357/v2.0
```

Where *KEYSTONE_HOST* is the hostname of the Identity service. If the Identity service is hosted on the same system, use **127.0.0.1**.

5. Configure the **heat-api-cfn** and **heat-api-cloudwatch** service hostnames to which VMs should connect.

```
# openstack-config --set /etc/heat/heat.conf \
 DEFAULT heat_metadata_server_url HEAT_CFN_HOST:8000
```

```
# openstack-config --set /etc/heat/heat.conf \
 DEFAULT heat_waitcondition_server_url HEAT_CFN_HOST:8000/v1/waitcondition
```

```
# openstack-config --set /etc/heat/heat.conf \
 DEFAULT heat_watch_server_url HEAT_CLOUDWATCH_HOST:8003
```

Where:

- *HEAT_CFN_HOST* is the IP or hostname of the system hosting the **heat-api-cfn** service.
- *HEAT_CLOUDWATCH_HOST* is the IP or hostname of the system hosting the **heat-api-cloudwatch** service.

Important

Even if all services are hosted on the same system, *do not* use **127.0.0.1** for either service hostname. This IP address would refer to the local host of each VM, and would therefore prevent the VM from reaching the actual service.

6. Application templates use wait conditions and signaling for orchestration. You will need to define the Identity role for users that will receive progress data. By default, this role is **heat_stack_user**.

```
# openstack-config --set /etc/heat/heat.conf \
 DEFAULT heat_stack_user_role heat_stack_user
```

Report a bug

16.2.5. Configure the Message Broker Settings of the Orchestration Service

Prerequisites:

- Section 8.2, “Configuring the Message Broker”

The Orchestration services must be explicitly configured with the type, location, and authentication details of the Message Broker. To do so, perform the following procedure as **root**:

Procedure 16.4. Configuring the Message Broker settings of the Orchestration Service

1. Set the value of the **rpc_backend** configuration key to Qpid.

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

2. Set the value of the **qpid_hostname** configuration key to the host name of the Qpid server.

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

   Replace *IP* with the IP address or host name of the Message Broker.

3. If you have configured Qpid to authenticate incoming connections then you must provide the details of a valid Qpid user in **/etc/heat/heat.conf**.

   a. Set the value of the **qpid_username** configuration key to the username of the Qpid user that the Orchestration service must use when communicating with the Message Broker.

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

      Replace *USERNAME* with the required Qpid user name.

   b. Set the value of the **qpid_password** configuration key to the password of the Qpid user that the Orchestration service must use when communicating with the Message Broker.

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

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

4. If you configured Qpid to use SSL then you must also configure the Orchestration service accordingly. Use **openstack-config** utility to set the value of the **qpid_protocol** configuration key to **ssl**.

   ```
   # openstack-config --set /etc/heat/heat.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/heat/heat.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**).

See Also:

- Section 8.2.2.4, "Configure SASL Using a Local Password File"

Report a bug

16.3. Launch the Orchestration Service

Once the required settings are configured, you can now launch the services that consist of the Orchestration service:

- **openstack-heat-api**
- **openstack-heat-api-cfn**
- **openstack-heat-api-cloudwatch**
- **openstack-heat-engine**

To do so for each service:

```
# service SERVICENAME start
```

Then, configure each service to launch automatically at boot:

```
# chkconfig SERVICENAME on
```

Replace *SERVICENAME* with the corresponding name of each Orchestration service.

Report a bug

16.4. Deploy a Stack Using Orchestration Templates

The Orchestration engine uses templates (defined as **.template** files) to launch instances, IPs, volumes, or other types of stacks. The **heat** utility is a command-line interface that allows you to create, configure, and launch stacks.

Note

The *openstack-heat-templates* package provides sample templates that you can use to test core Orchestration features. It also contains template-related scripts and conversion tools. To install this package, run the following command as **root**:

```
# yum install -y openstack-heat-templates
```

Some Orchestration templates launch instances that require access to **openstack-heat-api-cfn**. Such instances will need to be able to communicate with the **openstack-heat-api-cloudwatch** and **openstack-heat-api-cfn** services. The IPs and ports used by these services are the values set in the **/etc/heat/heat.conf** as **heat_metadata_server_url** and **heat_watch_server_url** (refer to Section 16.2.4, "Configure Orchestration Service Authentication" for details).

To allow access to these services, you may need to configure your firewall accordingly. Specifically, you will need to open the ports used by the **openstack-heat-api-cloudwatch** (8003) and **openstack-heat-api-cfn** (8000). To do so, perform the following tasks as **root**:

Procedure 16.5. Configuring the firewall for Orchestration Services traffic

1. Open the **/etc/sysconfig/iptables** file in a text editor.
2. Configure the firewall to allow TCP traffic on ports **8003** and **8000**. To do so, add the following **INPUT** rules to **/etc/sysconfig/iptables**:

   ```
   -A INPUT -i BR -p tcp --dport 8003 -j ACCEPT
   ```

   ```
   -A INPUT -i BR -p tcp --dport 8000 -j ACCEPT
   ```

   Replace *BR* with the interface of the bridge used by your instances (as in, the instances launched from Orchestration templates).

> **Note**
>
> Do not include the **`-i BR`** parameter in the **`INPUT`** rules if:
> - you are not using **`nova-network`**, or
> - the Orchestration service and **`nova-compute`** are not hosted on the same server.

3. Save the changes to the **`/etc/sysconfig/iptables`** file.

4. Restart the **`iptables`** service for the firewall changes to take effect.

```
# service iptables restart
```

To use **`heat`** to launch an application:

```
# heat stack-create STACKNAME \
 --template-file=PATH_TEMPLATE \
 --parameters="PARAMETERS"
```

Where:

- *STACKNAME* is the name you wish to assign to the stack. This name will appear when you run the **`heat stack-list`** command.
- *PATH_TEMPLATE* is the path to your **`.template`** file.
- *PARAMETERS* is a semicolon-delimeted list of stack creation parameters you wish to use. Supported parameters are defined in the template file itself.

For example, to launch a stack named **`myapplication`** using the template file **`/home/user/myapplication.template`** with specific database login parameters:

```
# heat stack-create myapplication \
 --template-file=/home/user/myapplication.template \
 --parameters="DBUsername=root;DBPassword=PASSWORD"
```

Report a bug

16.5. Integrate Telemetry and Orchestration Services

The Orchestration service can also use the Telemetry service (and its alarms) in monitoring the resource usage of stacks created using the **`heat stack-create`** command. To enable this, the Orchestration service must be installed and configured accordingly (see Section 15.1, “Overview of Telemetry Service Deployment” for more information) .

> **Note**
>
> For more information on how to use the Orchestration service, refer to the *Red Hat Enterprise Linux OpenStack Platform End User Guide* from the following link:
>
> https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/

Telemetry service alarms are used by the *autoscaling* feature. This feature allows the Orchestration service to automatically create stacks when the usage of a specific resource reaches a certain level. To allow Orchestration to use Telemetry alarms, the following line must be uncommented in/added to the **`resource_registry`** section of **`/etc/heat/environment.d/default.yaml`**:

```
"AWS::CloudWatch::Alarm":  "file:///etc/heat/templates/AWS_CloudWatch_Alarm.yaml"
```

See Also:

- Section 15.1, “Overview of Telemetry Service Deployment”

- Section 1.6, “Additional Documentation”

Report a bug

Chapter 17. Installing the Dashboard

17.1. 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
```

Report a bug

17.2. Install 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

17.3. Launch 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

17.4. Configuring the Dashboard

17.4.1. Configure 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 (sample files are available in the *Configuration Reference Guide* at https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform):

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*).

Note

It is recommended that you use **`django-secure`** module to ensure that most of the best practices and modern browser protection mechanisms are enabled. For more infomation http://django-secure.readthedocs.org/en/latest/ (*django-secure*).

Report a bug

17.4.2. Configure the Dashboard to Use HTTPS

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 donât 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=apache group=apache processes=3 threads=10
       RedirectPermanent /dashboard https://openstack.example.com
       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. The **<VirtualHost *:443>** section defines the required optionsfor this protocol, including private key, public key, and certificates.

4. As the **root** user, restart Apache and memcached:

```
# service httpd restart
# service memcached restart
```

When using the HTTP version of the dashboard (through the browser), the user should be redirected to the HTTPs version of the page.

Report a bug

17.4.3. Create a Member Role

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.

Report a bug

17.4.4. Configure SELinux

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 *Red Hat Enterprise Linux SELinux Users and Administrators Guide* at the following link:

https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/

Report a bug

17.4.5. Configure the Dashboard Firewall

To allow users to connect to the dashboard, you must configure the system firewall to allow connections. The httpd service, and the dashboard, support both HTTP and HTTPS connections.

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
     ```

2. 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 Security Guide* at the following link:

https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/

Report a bug

17.4.6. Session Storage Options

17.4.6.1. Configure 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'
     }
}
```

Report a bug

17.4.6.2. Configure Memcached Session Storage

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.

Report a bug

17.4.6.3. Configure Database Session Storage

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
```

9. 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
```

Report a bug

17.4.6.4. Configure Cached Database Session Storage

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"
```

Report a bug

17.4.6.5. Configure Cookies Session Storage

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:

1. In the **/etc/openstack-dashboard/local_settings** file, set:

   ```
   SESSION_ENGINE = "django.contrib.sessions.backends.signed_cookies"
   ```

2. 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).

Report a bug

17.5. Validate 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 Section 9.8, "Create a Regular User Account".

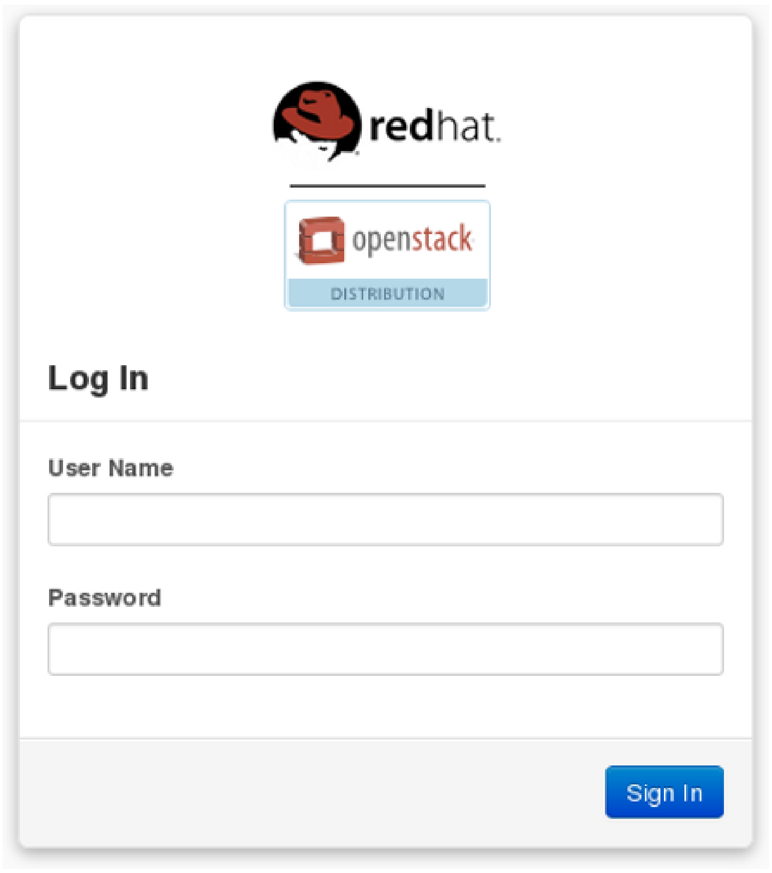


Figure 17.1. Dashboard Login Screen

See Also:

- Section 9.8, “Create a Regular User Account”

Report a bug

Part IV. Validating the Installation

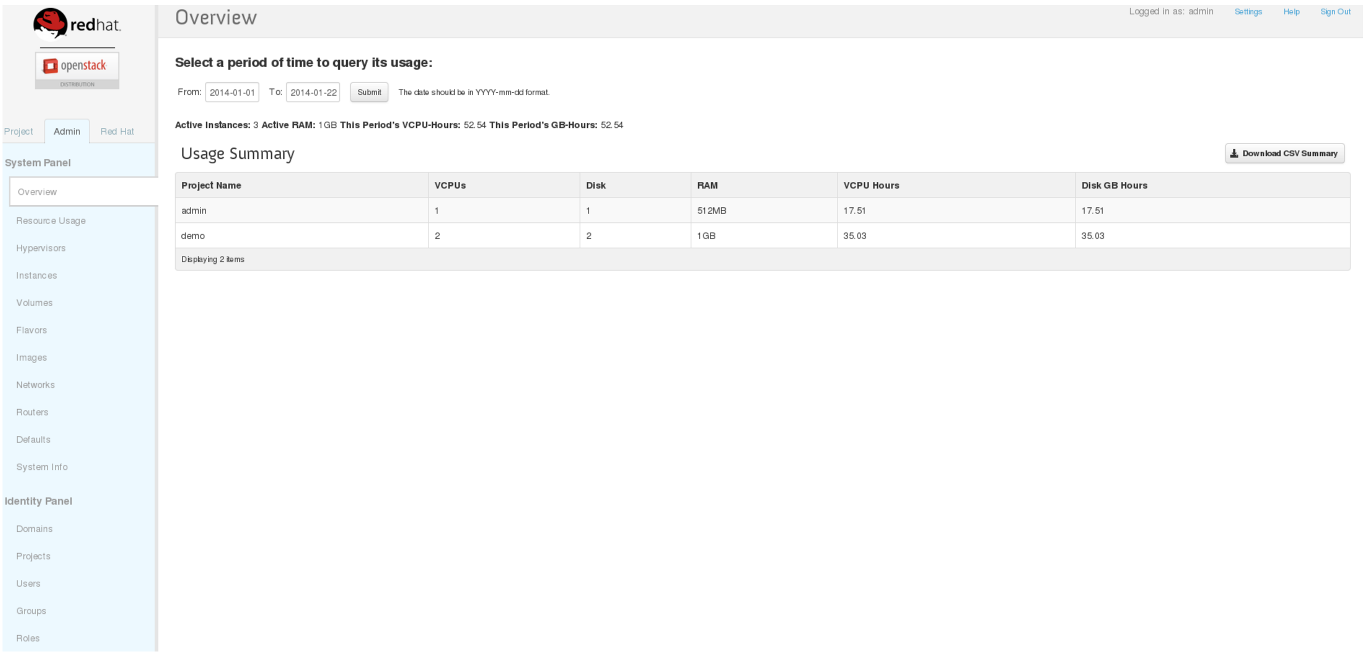
Chapter 18. Touring the Cloud

18.1. OpenStack Dashboard - Admin Tab

The **`Admin`** tab provides an interface where the administrators can view usage and manage instances, volumes, flavors, images, projects, users, services, and quotas.

Note

The **`Admin`** tab is only displayed in the main window if you have logged in as an administrator.



Overview

Select a period of time to query its usage:

From: 2014-01-01 To: 2014-01-22 The date should be in YYYY-mm-dd format.

Active Instances: 3 **Active RAM:** 1GB **This Period's VCPU-Hours:** 52.54 **This Period's GB-Hours:** 52.54

Usage Summary

Project Name	VCPUs	Disk	RAM	VCPU Hours	Disk GB Hours
admin	1	1	512MB	17.51	17.51
demo	2	2	1GB	35.03	35.03

Displaying 2 items

Figure 18.1. Dashboard - Admin Tab

You can access the following options available in the **`Admin`** tab.

Table 18.1. System Panel

Parameter Name	Description
`Overview`	View basic reports.
`Resource Usage`	This tab displays various statistics in a graphical format. You can choose different options under the **`Metric`**, **`Group by`**, **`Value`**, and **`Period`** parameters. For example, you can choose **`cpu`**, **`Project`**, **`Avg.`**, and **`Last week`** options to see the average CPU usage of your OpenStack cloud project during the previous week.
`Hypervisors`	View the hypervisor summary.
`Instances`	View, pause, resume, suspend, migrate, soft or hard reboot, and delete running instances that belong to users of some, but not all, projects. Also, view the log for an instance or access an instance using the console.
`Volumes`	View, create, edit, and delete volumes, and volume types.
`Flavors`	View, create, edit, view extra specs for, and delete flavors. Flavors are the virtual hardware templates in OpenStack.
`Images`	View, create, edit properties for, and delete custom images.
`Networks`	View, create, edit properties for, and delete networks.
`Routers`	View, create, edit properties for, and delete routers.

Parameter Name	Description
`Defaults`	View default quota values. Quotas are hard-coded in OpenStack Compute and define the maximum allowable size and number of resources.
`System Info`	Contains the following tabs: » **`Services`** tab- View a list of the services. » **`Compute Services`** tab- View a list of all Compute services. » **`Availability Zones`** tab- View the availability zones. » **`Host Aggregates`** tab- View host aggregates. » **`Network Agents`** tab- View the network agents.

Table 18.2. Identity Panel

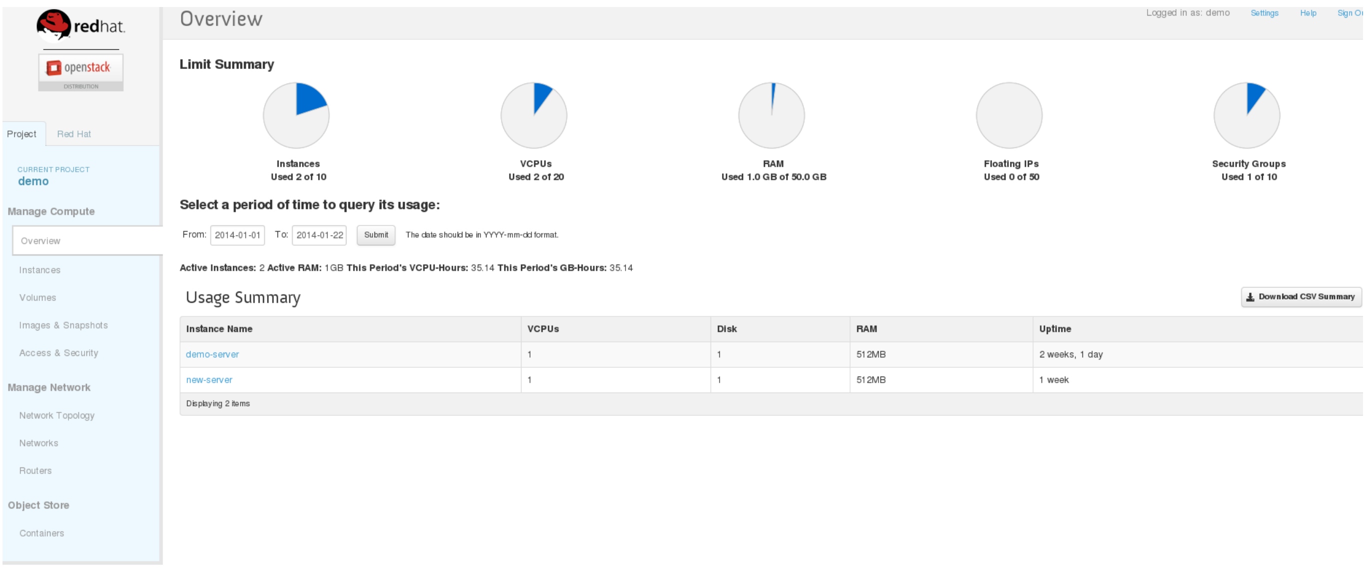
Parameter Name	Description
`Domains`	View domains.
`Projects`	View, create, assign users to, remove users from, and delete projects.
`Users`	View, create, enable, disable, and delete users.
`Groups`	View, create, enable, disable, and delete groups.
`Roles`	View, create, enable, disable, and delete roles.

Report a bug

18.2. OpenStack Dashboard - Project Tab

The Project tab provides an interface for viewing and managing the resources of a project. Select a project from the **`CURRENT PROJECT`** drop-down list on the left side to view and manage resources in that project.

The **`Project`** tab displays the details of the selected project.



Limit Summary

Instances
Used 2 of 10

VCPUs
Used 2 of 20

RAM
Used 1.0 GB of 50.0 GB

Floating IPs
Used 0 of 50

Security Groups
Used 1 of 10

Select a period of time to query its usage:

From: 2014-01-01 To: 2014-01-22 The date should be in YYYY-mm-dd format.

Active Instances: 2 **Active RAM:** 1GB **This Period's VCPU-Hours:** 35.14 **This Period's GB-Hours:** 35.14

Usage Summary

Instance Name	VCPUs	Disk	RAM	Uptime
demo-server	1	1	512MB	2 weeks, 1 day
new-server	1	1	512MB	1 week

Displaying 2 items

Figure 18.2. Dashboard - Project Tab

Access the following tabs:

Table 18.3. Manage Compute

Parameter Name	Description
`Instances`	View, launch, create a snapshot from, stop, pause, or reboot instances, or connect to them through the console.
`Volumes`	View, create, edit, and delete volumes.

Parameter Name	Description
`Images & Snapshots`	View images, instance snapshots, and volume snapshots created project users, and any images that are publicly available. Create, edit, and delete images, and launch instances from images and snapshots.
`Access & Security`	Use these tabs to complete these tasks: » `Security Groups` tab- View, create, edit, and delete security groups, and security group rules. » `Keypairs` tab- View, create, edit, and import keypairs, and delete keypairs. » `Floating IPs` tab- Allocate an IP address to or release it from a project. » `API Access` tab- View API endpoints.

Table 18.4. Manage Network

Parameter Name	Description
`Network Topology`	View the interactive topology of the network.
`Networks`	Create and manage public and private networks.
`Routers`	Create and manage subnets.

Table 18.5. Object Store

Parameter Name	Description
`Containers`	Create and manage storage containers. A container is a storage compartment for your data and provides a way for you to organize your data. It is similar to the concept as a Linux file directory but cannot be nested.

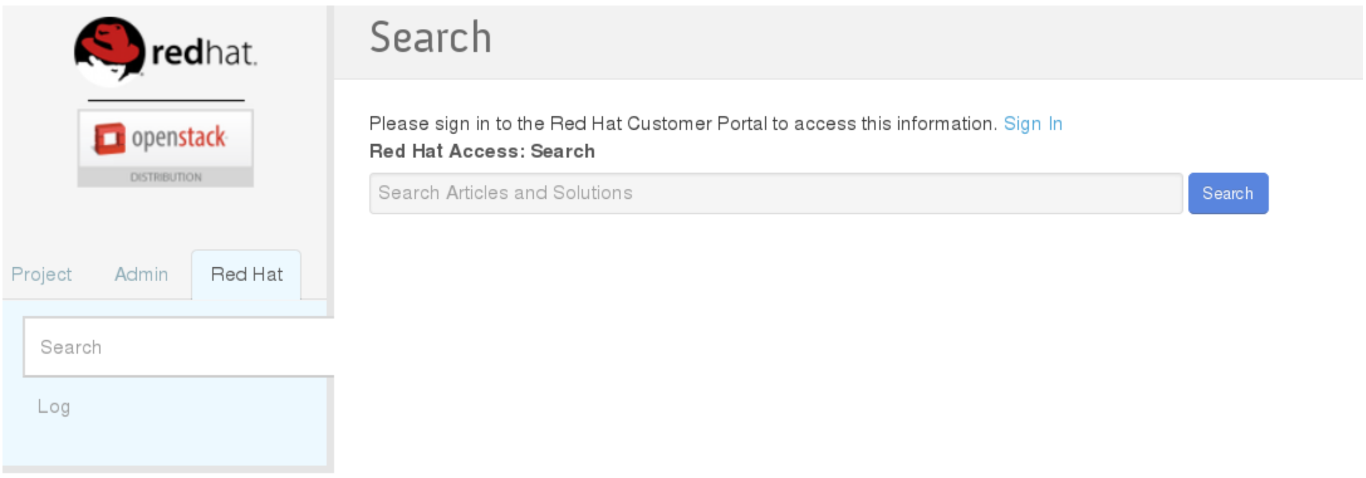
Table 18.6. Orchestration

Parameter Name	Description
`Stacks`	Orchestrate multiple composite cloud applications using templates, through both an OpenStack-native REST API and a CloudFormation-compatible Query API.

Report a bug

18.3. OpenStack Dashboard - Red Hat Tab

Red Hat tab gives you the option to sign in to the Red Hat Customer Portal and access information present there. This tab also contains a **Log** option that you can use to check the console log file for your instances.


Search

Please sign in to the Red Hat Customer Portal to access this information. Sign In

Red Hat Access: Search

Figure 18.3. Red Hat Tab

Report a bug

Chapter 19. Working with Instances

19.1. Uploading a Disk Image

You can upload an image by either using the command-line **`glance`** tool, or by using the dashboard. This procedure uses the dashboard. This means that you must have first:

- Installed the dashboard (refer to Section 1.3.1, “Dashboard Service”).
- Have an available image for use (refer to Section 11.6.1, “Obtain 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.

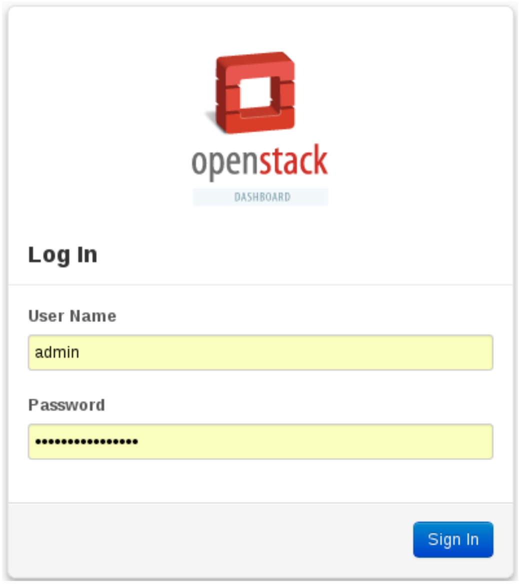
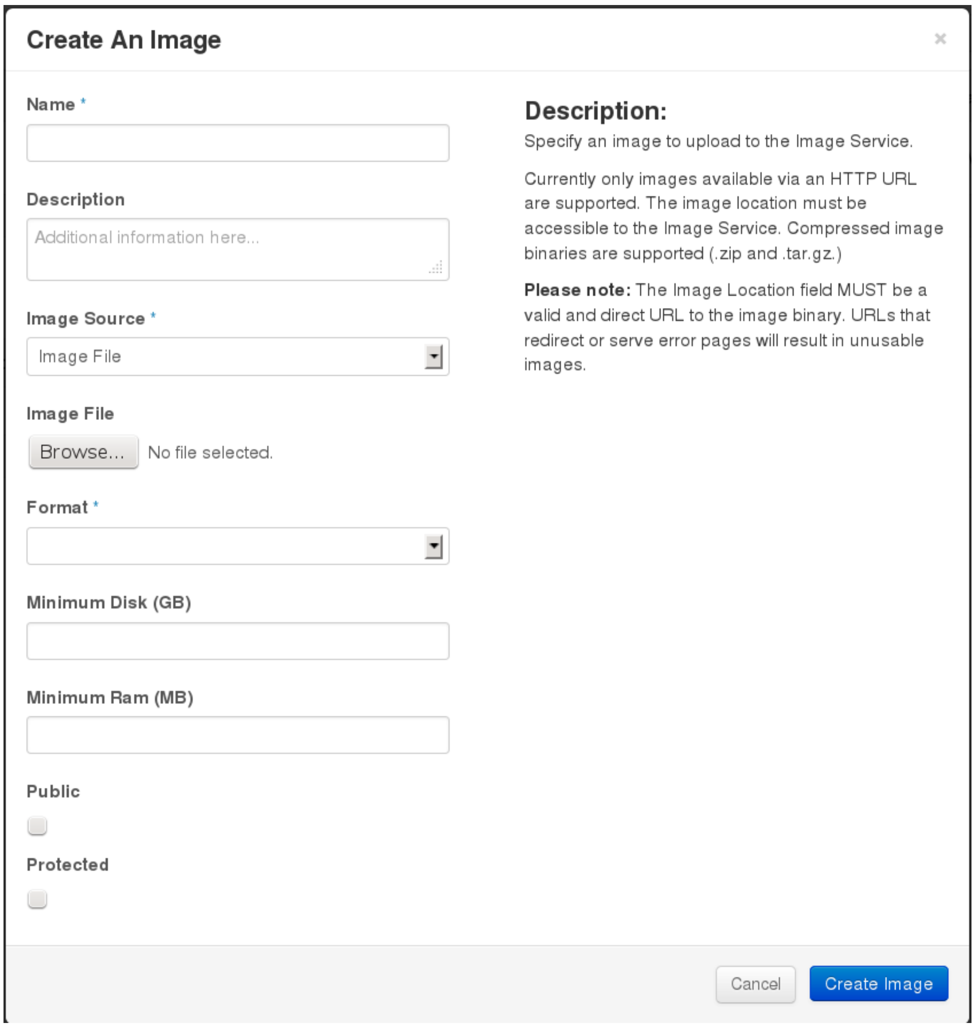


Figure 19.1. User login

2. Click **`Images & Snapshots`** in the **Manage Compute** menu.
3. Click the **`Create Image`** button. The **`Create An Image`** dialog is displayed.



Description:

Specify an image to upload to the Image Service.

Currently only images available via an HTTP URL are supported. The image location must be accessible to the Image Service. Compressed image binaries are supported (.zip and .tar.gz.)

Please note: The Image Location field MUST be a valid and direct URL to the image binary. URLs that redirect or serve error pages will result in unusable images.

Figure 19.2. Create An Image Dialog

4. Configure the settings that define your image.

   a. Enter a name for the image.

   b. Enter a brief description about the image.

   c. Choose the image source from the dropdown list. Your choices are **`Image Location`** and **`Image File`**.

   d. Based on your selection, there is an **`Image Location`** or **`Image File`** field. You can include the location URL or browse to the image file on your file system and include it.

   e. Select the correct type from the drop-down menu in the **`Format`** field (for example, **`QCOW2`**).

   f. Leave the **`Minimum Disk (GB)`** and **`Minimum RAM (MB)`** fields empty.

   g. Select the **`Public`** box, if you want to let other users access the image.

   h. Select the **`Protected`** box, if you want to keep the image protected from being accidentally deleted.

   i. Click the **`Create Image`** button.

You have successfully uploaded an image.

See Also:

- Chapter 17, *Installing the Dashboard*

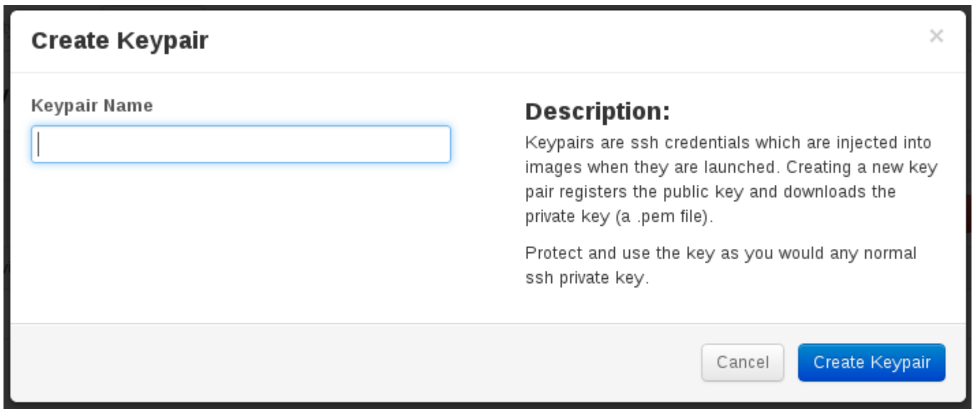
Report a bug

19.2. 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 Section 1.3.1, “Dashboard Service”).

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.

Figure 19.3. 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 17, Installing the Dashboard*

Report a bug

19.3. Creating a Network

To create a network from the dashboard, you must have first:

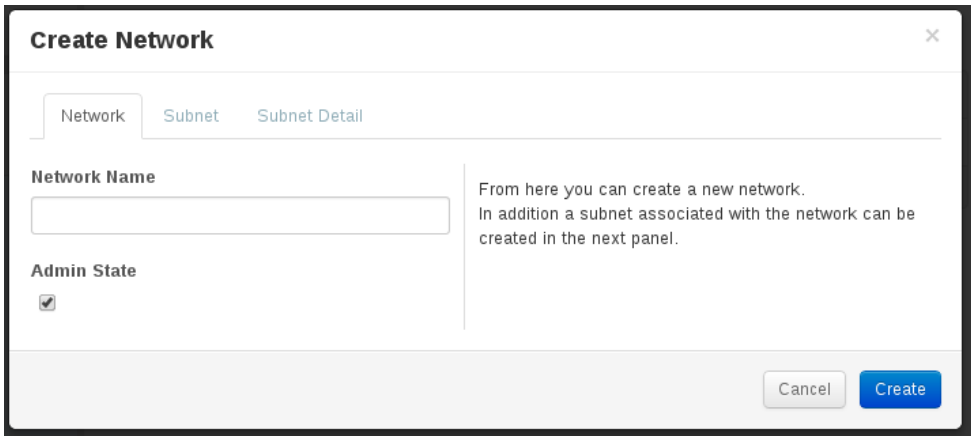
- Installed the dashboard (refer to Section 1.3.1, “Dashboard Service”).
- Installed OpenStack Networking (refer to Section 1.3.3, “OpenStack Networking Service”).

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.



From here you can create a new network.
In addition a subnet associated with the network can be created in the next panel.

Figure 19.4. 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 17, *Installing the Dashboard*
- Chapter 13, *Installing the OpenStack Networking Service*

Report a bug

19.4. Launching an Instance

To launch an instance from the dashboard you must have already:

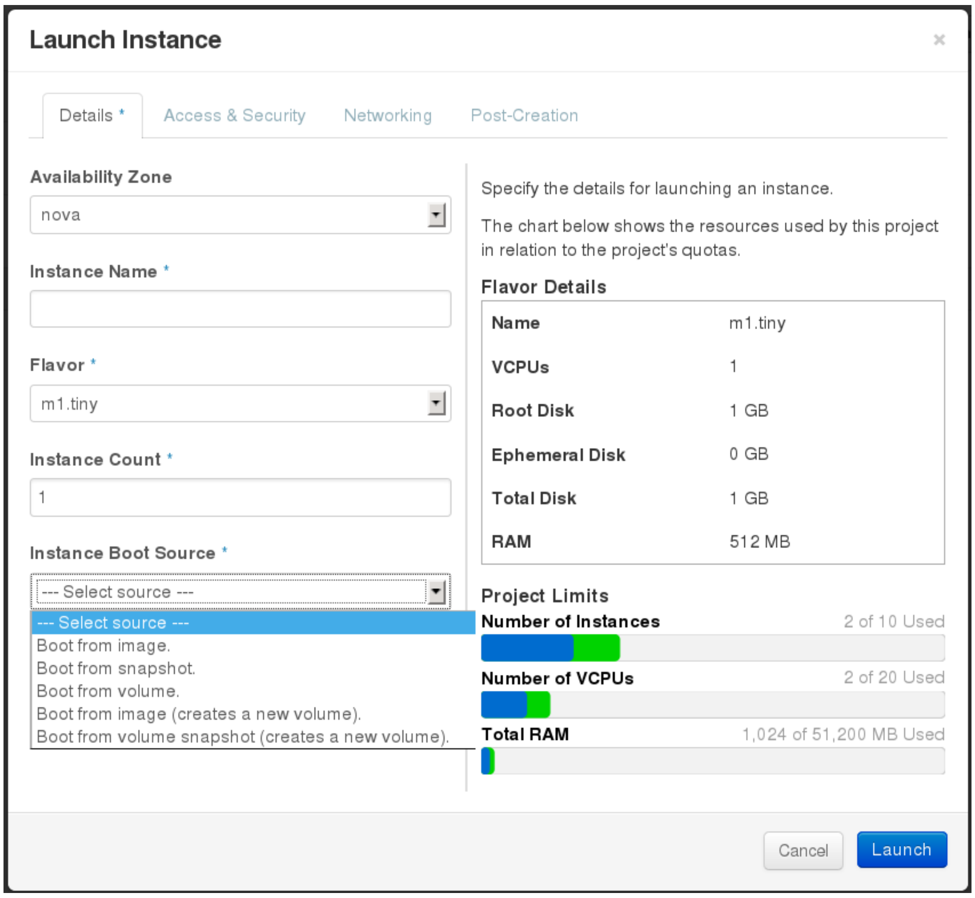
- Uploaded an image to use as the basis for your instances (refer to Section 19.1, “Uploading a Disk Image”).
- Created a keypair (refer to Section 19.2, “Creating a Keypair”)
- Created a network (refer to Section 19.3, “Creating a Network”).

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.



Availability Zone: nova

Instance Name *

Flavor *: m1.tiny

Instance Count *: 1

Instance Boot Source *: --- Select source ---

- --- Select source ---
- Boot from image.
- Boot from snapshot.
- Boot from volume.
- Boot from image (creates a new volume).
- Boot from volume snapshot (creates a new volume).

Specify the details for launching an instance.

The chart below shows the resources used by this project in relation to the project's quotas.

Flavor Details

Name	m1.tiny
VCPUs	1
Root Disk	1 GB
Ephemeral Disk	0 GB
Total Disk	1 GB
RAM	512 MB

Project Limits

Number of Instances — 2 of 10 Used

Number of VCPUs — 2 of 20 Used

Total RAM — 1,024 of 51,200 MB Used

Figure 19.5. Launch Instance: Details Tab

4. By default, the dialog opens to the **Details** tab:

   a. Select an **Availability Zone** for your instance from the dropdown list. The default option is **nova**.

   b. Enter an **Instance Name** to identify your instance.

   c. 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.

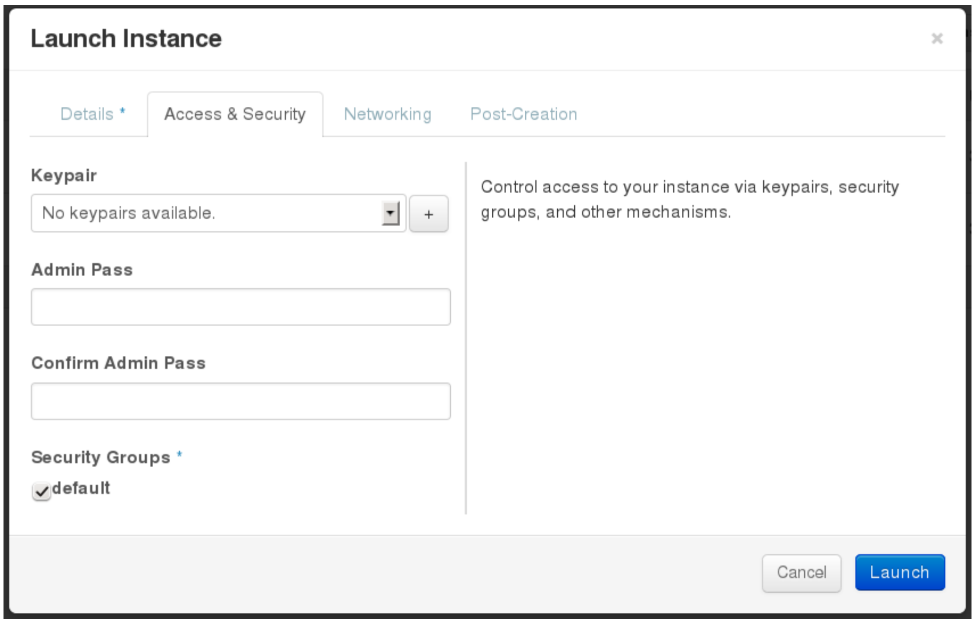
   d. Enter an **Instance Count**. This determines how many instances to launch using the selected options.

   e. Select a boot option from the **Instance Boot Source** dropdown list.

      You have the following boot options:

      - **Boot from image** - If you choose this option, a new field for **Image Name** is displayed. You can select the image from the dropdown list.
      - **Boot from snapshot** - If you choose this option, a new field for **Instance Snapshot** is displayed. You can select the snapshot from the dropdown list.
      - **Boot from volume** - If you choose this option, a new field for **Volume** is displayed. You can select the volume from the dropdown list.
      - **Boot from image (creates a new volume)** - With this option, you can boot from an image and create a volume by choosing **Device Size** and **Device Name** for your volume.
      - **Boot from volume snapshot (creates a new volume)** - Using this option, you can boot from a volume snapshot and create a new volume by choosing **Volume Snapshot** from a dropdown list and adding a **Device Name** for your volume.

5. Click the **Access & Security** tab and configure the security settings for your instance:


Launch Instance

Details * | Access & Security | Networking | Post-Creation

Keypair
No keypairs available.

Admin Pass

Confirm Admin Pass

Security Groups *
✓ default

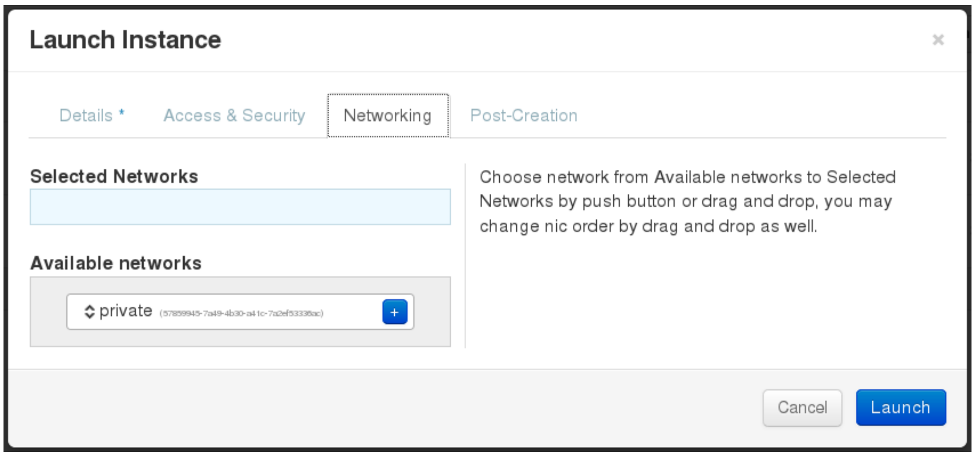
Control access to your instance via keypairs, security groups, and other mechanisms.

Figure 19.6. 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:


Launch Instance

Details * | Access & Security | Networking | Post-Creation

Selected Networks

Available networks
private (57859945-7a49-4b30-a41c-7a2ef53336ac)

Choose network from Available networks to Selected Networks by push button or drag and drop, you may change nic order by drag and drop as well.

Figure 19.7. 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 19.1, “Uploading a Disk Image”
- Section 19.2, “Creating a Keypair”
- Section 19.3, “Creating a Network”

Report a bug

19.5. Creating a Volume

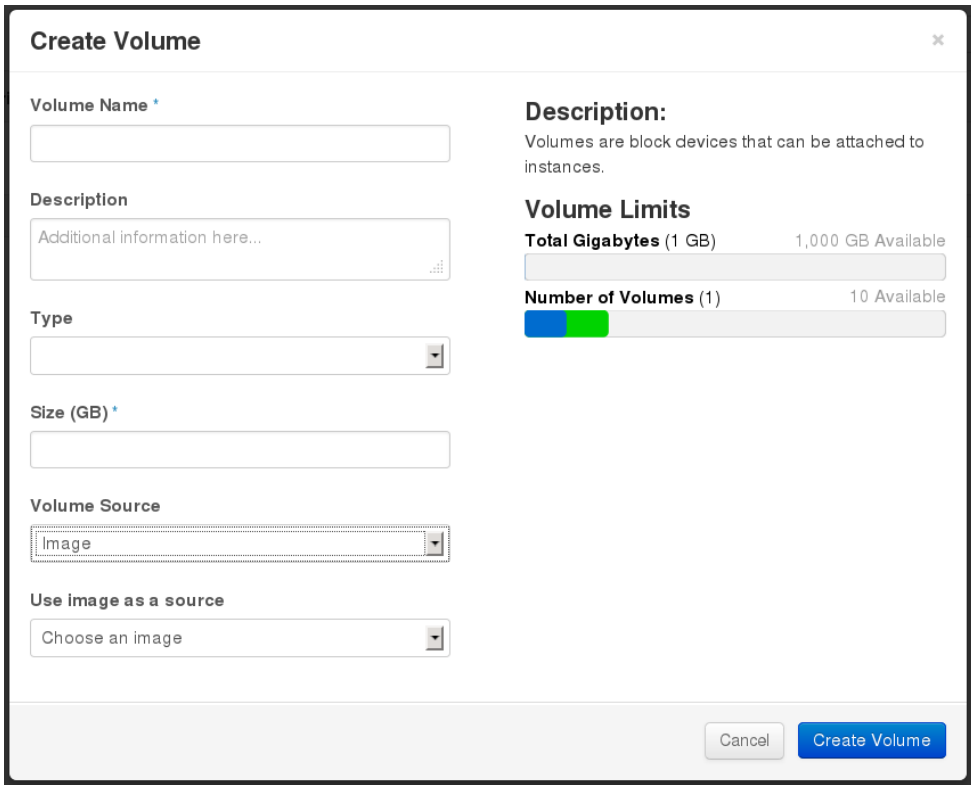
Compute instances support the attachment and detachment of block storage volumes. This procedure details the steps involved in creating a logical volume using the dashboard.

To create a volume from the dashboard, you must have first:

- Installed the dashboard (refer to Section 1.3.1, “Dashboard Service”).
- Installed the Block Storage service (refer to Section 1.3.4, “Block Storage Service”).

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:



Description:
Volumes are block devices that can be attached to instances.

Volume Limits
Total Gigabytes (1 GB) 1,000 GB Available
Number of Volumes (1) 10 Available

Figure 19.8. 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).
   d. Choose a source for your volume from the **Volume Source** dropdown list. You can choose to load an empty volume or from an image.

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 17, *Installing the Dashboard*
- Chapter 12, *Installing OpenStack Block Storage*

Report a bug

19.6. Attaching a Volume to an Instance

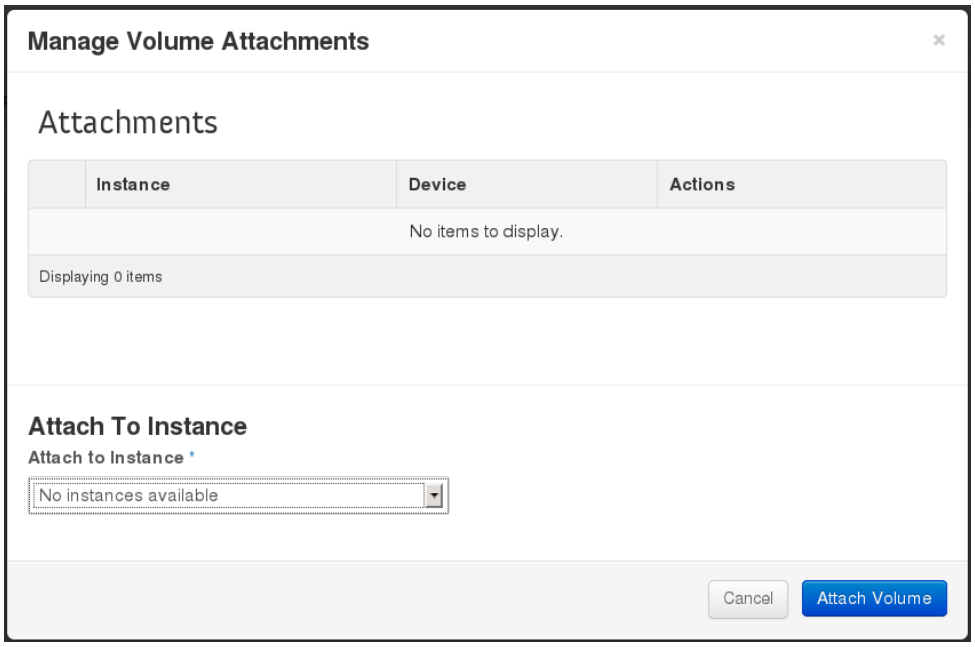
This procedure details the steps involved in attaching a Block Storage volume to an existing Compute instance using the dashboard.

To create a volume from the dashboard, you must have first:

- Launched an instance (refer to Section 19.4, “Launching an Instance”).
- Created a volume (refer to Section 19.5, “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.


Manage Volume Attachments

Attachments

	Instance	Device	Actions
	No items to display.		

Displaying 0 items

Attach To Instance

Attach to Instance *

No instances available

Figure 19.9. Manage Volume Attachments

4. Select the instance for the volume in the **`Attach to Instance`** field.

5. 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 19.4, “Launching an Instance”
- Section 19.5, “Creating a Volume”

Report a bug

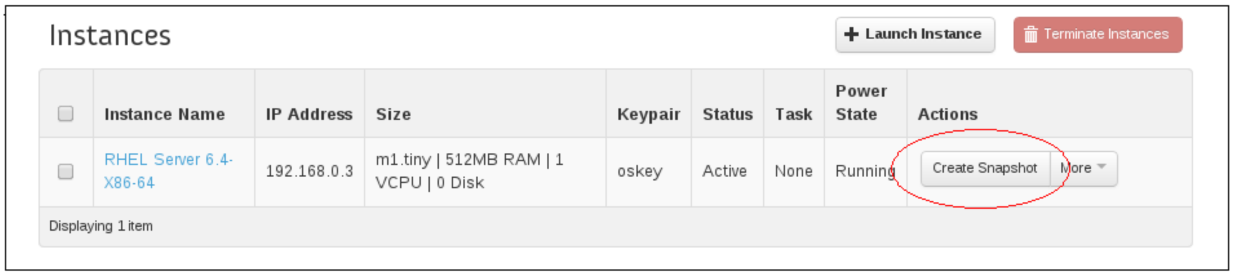
19.7. Creating an Instance Snapshot

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 Section 19.4, “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.



☐	Instance Name	IP Address	Size	Keypair	Status	Task	Power State	Actions
☐	RHEL Server 6.4-X86-64	192.168.0.3	m1.tiny | 512MB RAM | 1 VCPU | 0 Disk	oskey	Active	None	Running	Create Snapshot More

Displaying 1 item

Figure 19.10. 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`** 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 19.4, “Launching an Instance”

Report a bug

19.8. Controlling the State of an Instance (Pause, Suspend, Reboot)

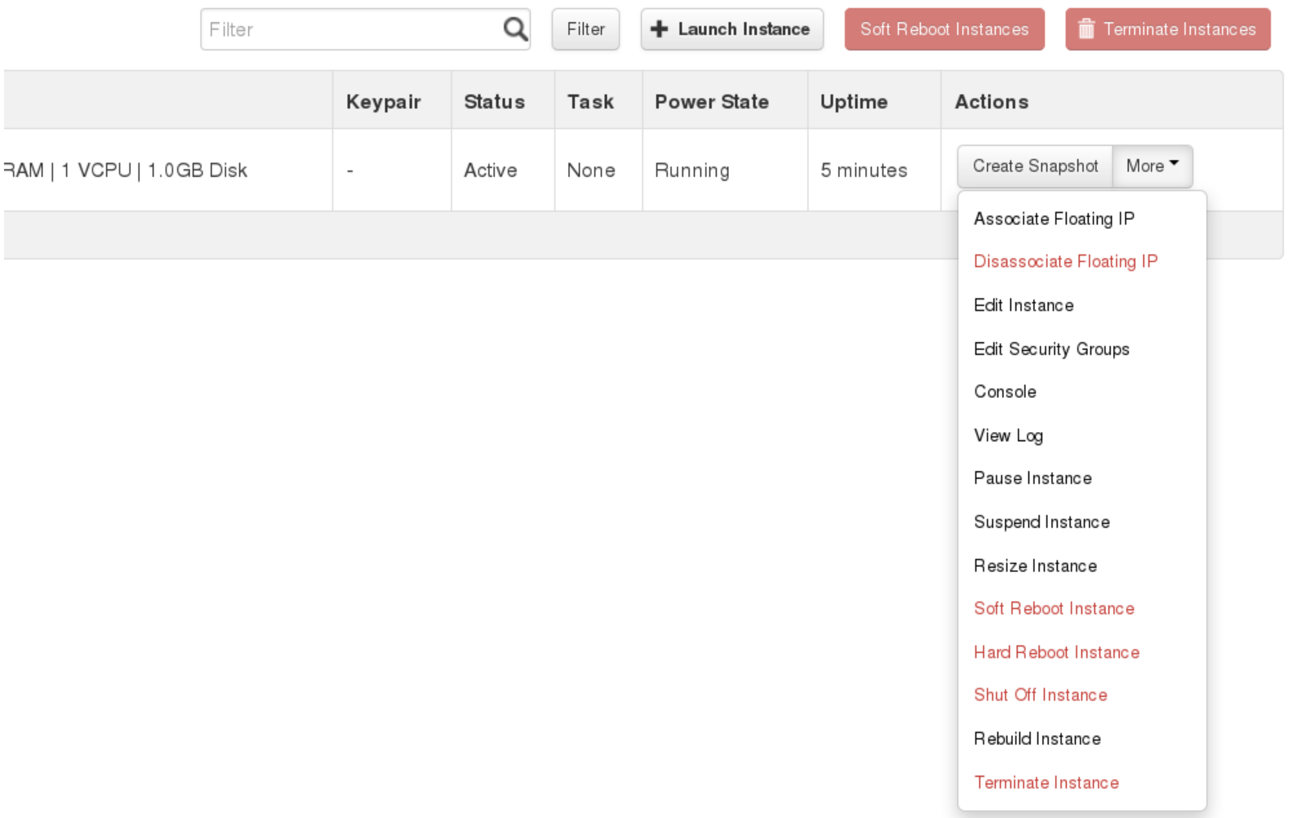
To change the state of an instance using the dashboard, you must have first launched an instance (refer to Section 19.4, “Launching an Instance”).

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.



	Keypair	Status	Task	Power State	Uptime	Actions
RAM | 1 VCPU | 1.0GB Disk	-	Active	None	Running	5 minutes	Create Snapshot More

Figure 19.11. 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 19.4, “Launching an Instance”

Report a bug

Chapter 20. Updating the Environment

20.1. 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:

   ```
   # neutron net-create networkName --router:external=True
   ```

   Example 20.1. Defining an External Network

   ```
   # neutron 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:

   ```
   $ neutron subnet-create  --allocation-pool start=IPStart,end=IPStart --gateway GatewayIP
   --disable-dhcp networkName CIDR
   ```

Example 20.2. Defining a Pool of Floating IP Addresses

```
$ neutron 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 13, *Installing the OpenStack Networking Service*

Report a bug

20.2. 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 Section 1.3.1, “Dashboard Service”).
- Created an external network (refer to Section 20.1, “Defining a Floating IP-Address Pool”).
- Created an internal network (refer to Section 19.3, “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:

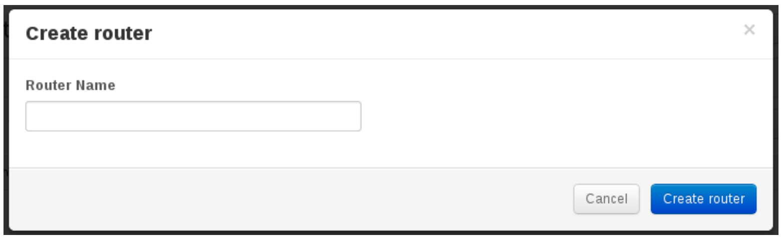
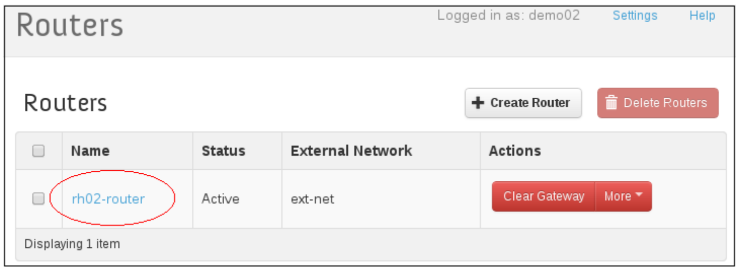


Figure 20.1. Create Router

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:

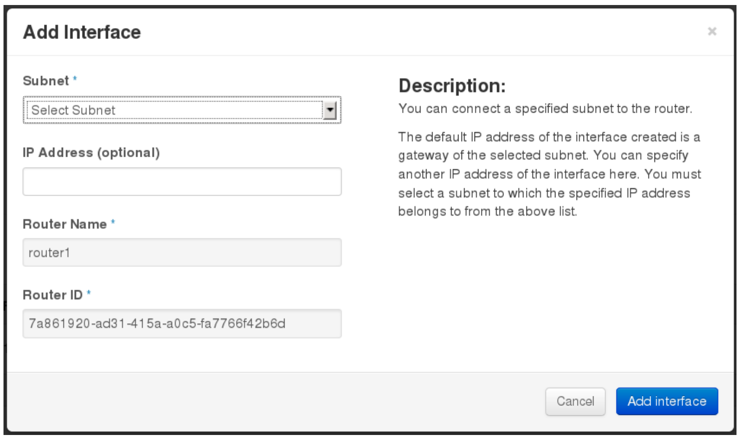


	Name	Status	External Network	Actions
	rh02-router	Active	ext-net	Clear Gateway More

Displaying 1 item

Figure 20.2. Select the router

   b. Click the **`Add Interface`** button. The **`Add Interface`** window is displayed:



Subnet *
Select Subnet

IP Address (optional)

Router Name *
router1

Router ID *
7a861920-ad31-415a-a0c5-fa7766f42b6d

Description:
You can connect a specified subnet to the router.

The default IP address of the interface created is a gateway of the selected subnet. You can specify another IP address of the interface here. You must select a subnet to which the specified IP address belongs to from the above list.

Figure 20.3. Add Interface

   c. Specify the new subnet to which the router should be attached in the **`Subnet`** field, an IP address in the **`IP Address (Optional)`** 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.

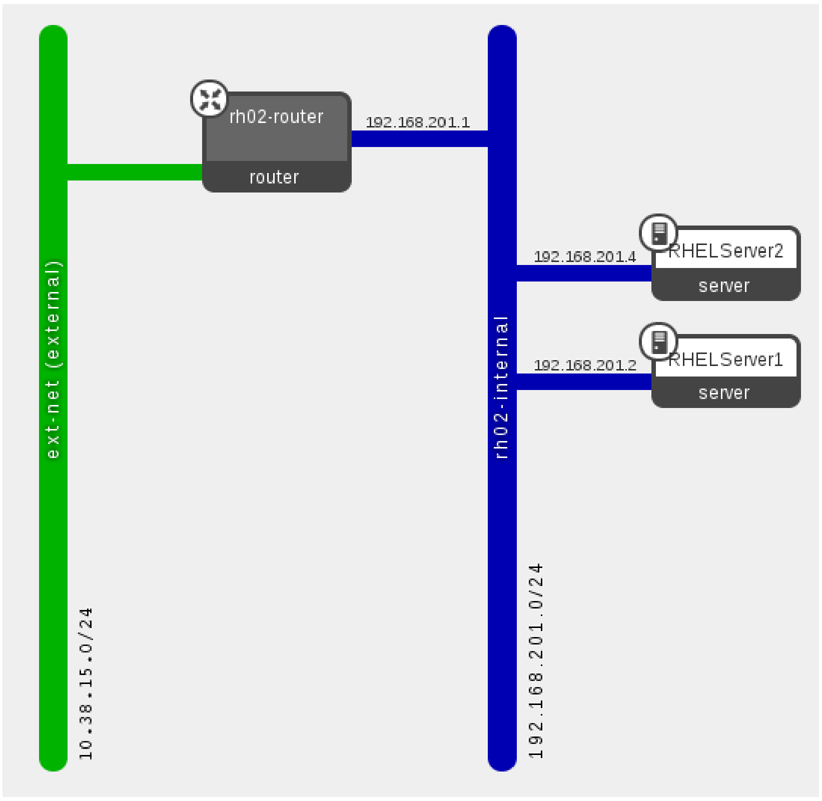


Figure 20.4. Network Topology

See Also:

- Chapter 17, *Installing the Dashboard*
- Section 20.1, "Defining a Floating IP-Address Pool"
- Section 19.3, "Creating a Network"

Report a bug

20.3. 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 Section 20.1, "Defining a Floating IP-Address Pool").
- Launched an instance (refer to Section 19.4, "Launching an Instance").

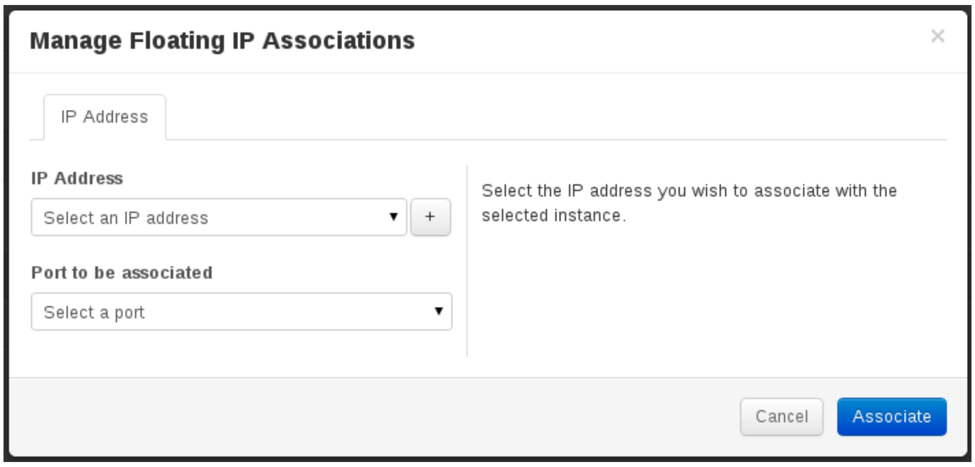
To associate an instance with a floating IP address:

1. Log in to the dashboard as a user that has the **`Member`** role.
2. Click **`Access & Security`** in the **Manage Compute** menu.
3. In the **`Floating IPs`** tab, 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** button to assign the IP address to a specific instance.

   The **Manage Floating IP Associations** window is displayed:



Select the IP address you wish to associate with the selected instance.

Figure 20.5. 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 20.1, “Defining a Floating IP-Address Pool”
- Section 19.4, “Launching an Instance”

Report a bug

20.4. 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:

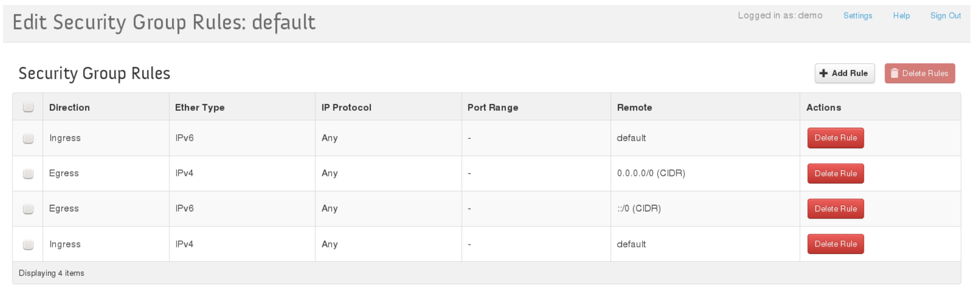
- Installed the dashboard (refer to Section 1.3.1, “Dashboard Service”).
- Installed OpenStack Networking (refer to Section 1.3.3, “OpenStack Networking Service”).

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`** tab, click the **`Edit Rules`** button on the row for the **`default`** security group. The **`Edit Security Group Rules: default`** window is displayed.



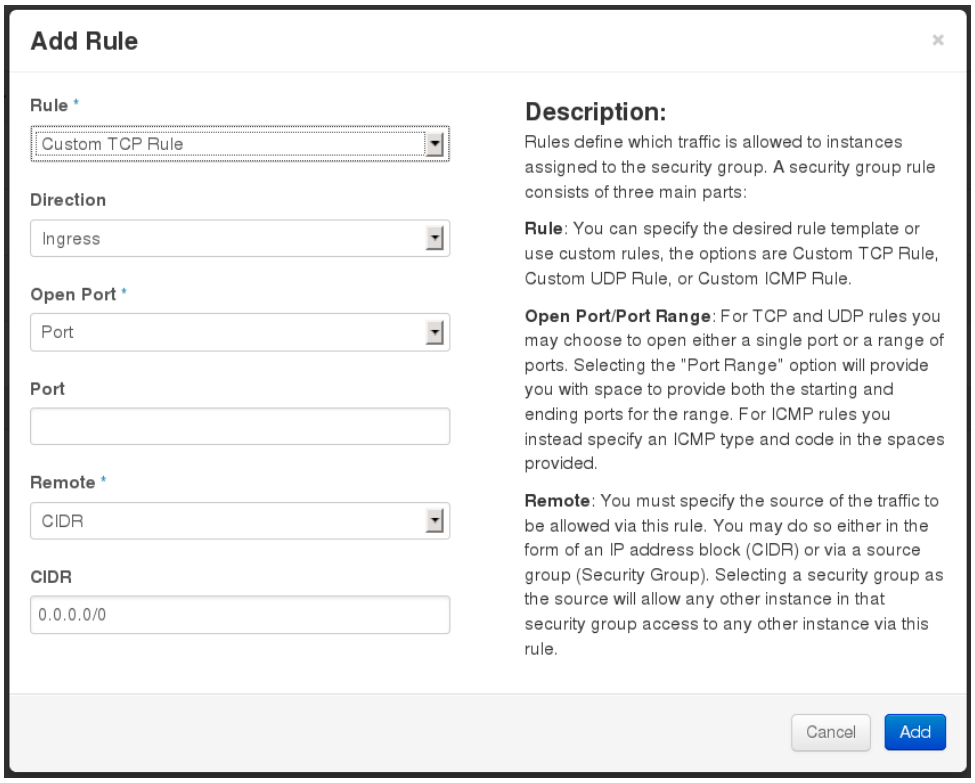
	Direction	Ether Type	IP Protocol	Port Range	Remote	Actions
	Ingress	IPv6	Any	-	default	Delete Rule
	Egress	IPv4	Any	-	0.0.0.0/0 (CIDR)	Delete Rule
	Egress	IPv6	Any	-	::/0 (CIDR)	Delete Rule
	Ingress	IPv4	Any	-	default	Delete Rule

Displaying 4 items

Figure 20.6. Edit Security Group Rules: default

These are the default rules. You can delete them using the **`Delete Rule`** button.

4. To add new rules, click the **`Add Rule`** button. The **`Add Rule`** window is displayed.



Rule *
Custom TCP Rule

Direction
Ingress

Open Port *
Port

Port

Remote *
CIDR

CIDR
0.0.0.0/0

Description:
Rules define which traffic is allowed to instances assigned to the security group. A security group rule consists of three main parts:

Rule: You can specify the desired rule template or use custom rules, the options are Custom TCP Rule, Custom UDP Rule, or Custom ICMP Rule.

Open Port/Port Range: For TCP and UDP rules you may choose to open either a single port or a range of ports. Selecting the "Port Range" option will provide you with space to provide both the starting and ending ports for the range. For ICMP rules you instead specify an ICMP type and code in the spaces provided.

Remote: You must specify the source of the traffic to be allowed via this rule. You may do so either in the form of an IP address block (CIDR) or via a source group (Security Group). Selecting a security group as the source will allow any other instance in that security group access to any other instance via this rule.

Figure 20.7. Add Rule

5. Configure the rule:
   a. Select the desired rule template or use custom rules from the **`Rule`** dropdown list.
   b. Select the **`Direction`** from the dropdown list.
   c. Define the port or ports to which the rule will apply using the **`Open Port`** field:
      - **`Port`** - Define a specific port in the **`Port`** field.

- **`Port Range`** - Define the port range using the **`From Port`** and **`To Port`** fields.

d. Specify the source of the traffic to be allowed via this rule using the **`Remote`** field.

   You can choose to do so either in the form of an IP address block (CIDR) or via a source group (Security Group).

   - **`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 the **`Security Group`** and **`Ether Type`** fields.

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 17, *Installing the Dashboard*
- Chapter 13, *Installing the OpenStack Networking Service*

Report a bug

Part V. Monitoring the OpenStack Environment

Chapter 21. Monitoring OpenStack using Nagios

21.1. Installing Nagios

21.1.1. 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

Graphics Library, for dynamically creating images

gd-devel

Development libraries for Graphics Library (*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 are 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
```

Report a bug

21.1.2. Installing the NRPE Addon

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 NPRE 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

21.2. Configuring Nagios

21.2.1. 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 21.1. Nagios Configuration Files

File Name	Description
`/etc/nagios/nagios.cfg`	Main Nagios configuration file.
`/etc/nagios/cgi.cfg`	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/ObjectsDir/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.

Report a bug

21.2.2. Configuring HTTPD

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.cfg`** 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 newUserName
   ```

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.

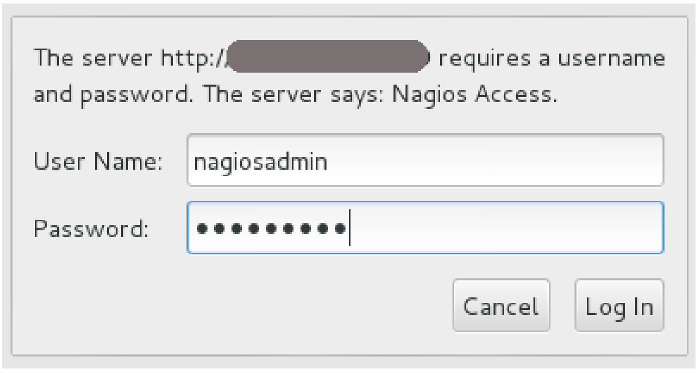
   ```
   # chkconfig --add nagios
   # chkconfig nagios on
   ```

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
   ```



The server http:/[redacted] requires a username and password. The server says: Nagios Access.

User Name: nagiosadmin

Password: ●●●●●●●●●●

Figure 21.1. 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*).

Report a bug

21.2.3. Configuring OpenStack Services

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
```

Report a bug

21.2.4. Configure NRPE

To set up monitoring on each remote machine, execute the following as the **root** user:

Procedure 21.1. Configuring monitoring on a remote machine

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*.

3. Configure the **iptables** firewall to allow **nrpe** traffic.

   a. Open the **/etc/sysconfig/iptables** file in a text editor.

   b. Add an **INPUT** rule allowing UDP traffic on port 5666 to this file. The new rule must appear before any **INPUT** rules that **REJECT** traffic.

   ```
   -A INPUT -p tcp --dport 5666 -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
   ```

4. Start the NRPE service:

```
# service nrpe start
```

Report a bug

21.2.5. Creating Host Definitions

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:

1. In the **/etc/nagios/objects** directory, create a **hosts.cfg** file.

2. 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
}
```

3. In the **/etc/nagios/nagios.cfg** file, under the **OBJECT CONFIGURATION FILES** section, specify the following line:

```
cfg_file=/etc/nagios/objects/hosts.cfg
```

Report a bug

21.2.6. Creating Service Definitions

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
```

Report a bug

21.2.7. Verifying the Configuration

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
```

Report a bug

Chapter 22. Installing and Configuring Remote Logging

22.1. 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

22.2. Install 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. To do so, log in as the **root** user and install the *rsyslog* package:

```
# yum install rsyslog
```

The *rsyslog* package is installed and ready to be configured.

Report a bug

22.3. Configure 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.

Procedure 22.1. Configuring rsyslog on the centralized logging server

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

22.4. Configure rsyslog on 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.

Procedure 22.2. Configuring a node to send logs to a centralized log server

- 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

22.5. Start the 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.

Procedure 22.3. Launching the rsyslog service

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 VI. Managing OpenStack Environment Expansion

Chapter 23. Managing Compute Expansion

23.1. 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 23.1. 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

23.2. 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) through the Message Broker.

To add additional compute nodes repeat the steps performed when installing the original compute node:

- Section 14.3.3, “Install the Compute Service Packages”
- Section 14.3.5.3, “Configure Message Broker Settings for the Compute Service”
- Section 14.3.5.4, “Configure Resource Overcommitment”
- Section 14.3.5.5, “Reserve Host Resources”
- Section 14.3.5.6.2, “Update the Compute Configuration”
- Section 14.3.5.6.3, “Configure the L2 Agent”
- Section 14.3.5.6.4, “Configure Virtual Interface Plugging”

- Section 14.3.5.7, “Configure the Firewall to Allow Compute Service Traffic”
- Section 14.3.7, “Launch 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 14.3.5.2, “Configure the Compute Service Database Connection” 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

23.3. Safely Remove 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.

Procedure 23.1. Safely removing Compute resources

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.

4. 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    |
| ...                | ...      | ...      | ...      | ...   |
+-------------------+----------+----------+----------+-------+
```

5. 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.

Note

For further information on migration, refer to the *Red Hat Enterprise Linux OpenStack Platform Configuration Reference Guide* from the following link:

https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform

Report a bug

23.4. Using Config Drive

23.4.1. 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 through 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.

Report a bug

23.4.2. Setting Up Config Drive

23.4.2.1. 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).

Report a bug

23.4.2.2. 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 23.1. 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_tempdir=None	(StrOpt) Where to put temporary files associated with config drive creation.

Configuration option=Default value	(Type) Description
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

23.4.3. Access 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 23.2. 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 23.3. 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 m1.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 expect the EC2 metadata (as in, files under the **ec2** directory) to be present. 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, and 2013-04-13, but only a **2012-08-05** directory is present, then select 2012-08-05.

Procedure 23.4. 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
```

Report a bug

23.4.4. Data Formats

23.4.4.1. 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
AAAAB3NzaC1yc2EAAAADAQABAAAAgQDBqUfVvCSez0/Wfpd8dLLgZXV9GtXQ7hnMN+Z0OWQUyebVEHey1CXuin0uY1cAJMh
Uq8j98SiW+cU0sU4J3x5l2+xi1bodDm1BtFWVeLIOQINpfV1n8fKjHB+ynPpe1F6tMDvrFGUlJs44t30BrujMXBe8Rq44cC
k6wqyjATA3rQ== 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.

Report a bug

23.4.4.2. EC2 Metadata Format

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
AAAAB3NzaC1yc2EAAAADAQABAAAAgQDBqUfVvCSez0/Wfpd8dLLgZXV9GtXQ7hnMN+Z0OWQUyebVEHey1CXuin0uY1cAJMh
Uq8j98SiW+cU0sU4J3x5l2+xi1bodDm1BtFWVeLIOQINpfV1n8fKjHB+ynPpe1F6tMDvrFGUlJs44t30BrujMXBe8Rq44cC
k6wqyjATA3rQ== Generated by Nova\n"
        }
    },
    "ramdisk-id": "ari-00000003",
    "reservation-id": "r-7lfps8wj",
    "security-groups": [
        "default"
    ]
}
```

Report a bug

23.4.4.3. User Data Format

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`**

Report a bug

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 2.2.1, “Compute Node Requirements”, Section 2.2.2, “Network Node Requirements”, and Section 2.2.3, “Block Storage Node Requirements” must be verified.	Yes | No
Operating System	Red Hat Enterprise Linux 6.5 Server	Yes | No
Red Hat Subscription	You must have a subscription to: » Receive package updates from Red Hat Network or an equivalent source such as a Red Hat Network Satellite server. » Receive software updates for both Red Hat Enterprise Linux 6.5 Server and Red Hat Enterprise Linux OpenStack Platform	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.	» Name: » Password:
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: » Identity service » OpenStack Networking service » Block Storage service » Compute service » Image service » Object Storage service » Dashboard service » MySQL server (default database for this guide) » Qpid message broker

Table A.2. OpenStack Identity Service

Item	Description	Value
Host Access	The system hosting the Identity service must have: » Access to Red Hat Network or equivalent service. » Network interface addressable by all OpenStack hosts. » Network access to the database server. » If using LDAP, network access to the directory server .	Verify whether the system has: » Yes | No » Yes | No » Yes | No » Yes | No
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.	» XFS » ext4
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: » file (local directory) » OpenStack Object Storage service	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: » LVM » NFS » Red Hat Storage	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

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: » Open vSwitch » Cisco UCS/Nexus » Linux Bridge » Nicira Network Virtualization Platform (NCP) » Ryu OpenFlow Controller » NEC OpenFlow » Big Switch Controller Plugin » Cloudbase Hyper-V » MidoNet » Brocade Neutron Plugin » PLUMgrid
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 Section 14.1.1, “Check for Hardware Virtualization Support”).	Yes | No
VNC client	The Compute service supports the Virtual Network Computing (VNC) console access 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 Section 14.3.5.4, “Configure Resource Overcommitment”). » 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.	Decide: » CPU setting: » Memory setting:

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 Section 14.3.5.5, “Reserve Host Resources”).	Decide:
		» Host Disk (default 0MB):
		» Host Memory (default 512MB):

libvirt Version	You will need to know the version of your libvirt for the configuration of Virtual Interface Plugging (refer to Section 14.3.5.6.4, “Configure Virtual Interface Plugging”).	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

Item	Description	Value
Host software	The system hosting the Dashboard service must have the following already installed: » httpd » mod_wsgi » mod_ssl	Yes | No
Connections	The system hosting the Dashboard service must have a connection to all other OpenStack services.	Yes | No

Report a bug

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 Identity 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

You can view the log output of running instances from either the **Log** tab of the dashboard or the output of **nova console-log**. In some cases, the log output of a running Linux instance will be empty or only display a single character (for example, the **?** character).

This occurs when the Compute service attempts to retrieve the log output of the instance via a serial console while the instance itself is not configured to send output to the console. To rectify this, append the following parameters to the kernel arguments specified in the instance's boot loader (for example, GRUB):

```
console=tty0 console=ttyS0,115200n8
```

Upon rebooting, the instance will be configured to send output to the Compute service.

Report a bug

B.4. Troubleshoot Identity Client (keystone) Connectivity Problems

When the Identity client (**keystone**) is unable to contact the Identity service it returns an error:

```
Unable to communicate with identity service: [Errno 113] No route to host. (HTTP 400)
```

To debug the issue check for these common causes:

Identity service is down

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 is not configured properly

The firewall might not be configured to allow TCP traffic on ports **5000** and **35357**. If so, refer to Section 9.4.6, “Configure the Firewall to Allow Identity Service Traffic” for instructions on how to correct this.

Service Endpoints not defined correctly

On the system hosting the Identity service check that the endpoints are defined correctly.

Procedure B.1. Verifying Identity Service endpoints

1. Obtain the administration token:

   ```
   $ grep admin_token /etc/keystone/keystone.conf
   admin_token = 0292d404a88c4f269383ff28a3839ab4
   ```

2. 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.

3. Unset any pre-defined Identity service related environment variables:

   ```
   $ unset OS_USERNAME OS_TENANT_NAME OS_PASSWORD OS_AUTH_URL
   ```

4. 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 9.6, “Create 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.

Report a bug

B.5. Backup and Restore

B.5.1. Backup Overview

Both a backup and a recovery policy should be formulated to minimize data loss and system downtime.

When determining your backup strategy, you will need to answer the following questions:

- How quickly will you need to recover from data loss? If you cannot have data loss at all, you should focus on High Availability as a deployment strategy, in addition to using backups.
- How many backups should you keep?
- Should your backups be kept off-site?
- How often should backups be tested?
- What will be backed up? Critical data in OpenStack is backed up per component (or service).

The following sections describe database and file-system backups for components, as well as information on recovering backups.

Report a bug

B.5.2. Database Backup

The installation process in this guide uses the MySQL server, which hosts the databases for the separate OpenStack components. With all these databases in one place, it is easy to create a database backup.

Note

If you are using a different database server, refer to your server's notes for more information. For example, if you are using MariaDB and MyISAM tables, consider using **`mysqlhotcopy`** to improve backup and recovery times.

Procedure B.2. Backing up all databases

1. Log in as the database root user.
2. Export database information into a backup file:

```
# mysqldump --opt --all-databases > fileName
```

For example:

```
# mysqldump --opt --all-databases > openstack.sql
```

Procedure B.3. Backing up a single database

1. Log in as the database root user.
2. Export a single database into a backup file:

```
# mysqldump --opt databaseName > fileName
```

For example:

```
# mysqldump --opt nova > nova.sql
```

Note

You can automate the backup process by creating a cron job that runs the following script once per day:

```
#!/bin/bash
backup_dir="/var/lib/backups/mysql"
filename="${backup_dir}/mysql-`hostname`-`eval date +%Y%m%d`.sql.gz"

# Dump the entire MySQL database
/usr/bin/mysqldump --opt --all-databases | gzip > $filename

# Delete backups older than 7 days
find $backup_dir -ctime +7 -type f -delete
```

Report a bug

B.5.3. File System Backup

Each OpenStack component has its own directory and files under **`/etc`**, **`/var/lib`**, and **`/var/log`**. Files in these directories should be regularly backed up:

/etc/component

Files in this directory (for example, **`/etc/glance`**) should be regularly backed up. This will be the case for all nodes on which the component is housed. In particular:

- **`/etc/nova`** will need to be backed up on both the cloud controller and all Compute nodes.
- **`/etc/swift`** is very important, because it contains Object Storage configuration files, ring files, and ring builder files. Losing these files will render the data on your cluster inaccessible. A best practice is to copy the builder files to all storage nodes along with the ring files. Try to spread multiple backup copies throughout your storage cluster.

/var/lib/component

It is good practice to back up this directory (for example, **`/var/lib/keystone`**) for all components. In particular:

- **`/var/lib/glance/images`** - If you are using a file-based backend for the Image service, this directory stores its images and should be backed up regularly. Using both of the following should ensure image availability:
  - Run the directory on a RAID array; if a disk fails, the directory is available.
  - Use a tool such as **rsync** to replicate the images to another server:

    ```
     #  rsync -az --progress /var/lib/glance/images backup-server: \
    /var/lib/glance/images/
    ```

- **`/var/lib/nova`** - All files in this directory should be backed up. The only exception is the **`/var/lib/nova/instances`** subdirectory on Compute nodes. This subdirectory contains the KVM images of running instances. You would only want to back up this directory if you need to maintain backup copies of all instances. Under most circumstances, you do not need to do this, but this can vary from cloud to cloud and your service levels. Also, be aware that making a backup of a live KVM instance can cause that instance to not boot properly if it is ever restored from a backup.

/var/log/component

Log files should be regularly backed up, unless you have all logs going to a central server. It is highly recommended to use a central logging server.

Note

Various tools can be used to backup files. Red Hat Enterprise Linux supports the following:

- **tar**

  For example, to create an archive file called **var-lib-cinder-backup.tar** in **/mnt/backup/**, run:

  ```
  # tar cf /mnt/backup/var-lib-cinder-backup.tar /var/lib/cinder
  ```

  The resulting archive file contains the contents of the **/var/lib/cinder** directory, and will be nearly as large as the data being backed up. Depending on the type of data being backed up, compressing the archive file can result in significant size reductions. The archive file can be compressed by adding a single option to the previous command:

  ```
  # tar czf /mnt/backup/var-lib-cinder-backup.tar.gz /var/lib/cinder
  ```

  The resulting **var-lib-cinder-backup.tar.gz** archive file is now **gzip** compressed.

- **cpio**

  Unlike **tar**, **cpio** reads the names of the files it processes through standard input. For example:

  ```
  # find /var/lib/cinder/ | cpio -o > /mnt/backup/var-lib-cinder-backup.cpio
  ```

  This command creates a **cpio** archive file (containing everything in **/var/lib/cinder/**) called **var-lib-cinder-backup.cpio** and places it in the **/mnt/backup/** directory.

Report a bug

B.5.4. Recovery

Recovery involves the simple procedure of stopping services, restoring files and databases for the component, then restarting the services. All procedures must be carried out as the root user.

Procedure B.4. Recover database backups and files

1. Ensure all services are stopped for the component. For example, for the Image service:

   ```
   # service openstack-glance-registry stop
   # service openstack-glance-api stop
   ```

2. Stop MySQL:

   ```
   # stop mysql
   ```

3. Import the previously backed-up database. For example, to import an Image service back up:

   ```
   # mysql glance < glance.sql
   ```

4. Copy backup files over to the necessary directories. For example, for Image files:

   ```
   # mv /etc/glance{,.orig}
   # cp -a /path/to/backup/glance /etc/
   ```

5. Restart the component's services. For example, to restart the Image service you will have to restart the SQL database, Image registry, and Image API.

```
# start mysql
# service openstack-glance-registry start
# service openstack-glance-api start
```

Report a bug

Additional Procedures

C.1. Using Foreman to deploy Neutron

This example procedure demonstrates how to deploy an OVS/GRE Neutron environment with Foreman. This procedure comprises three key steps: provisioning a node to act as the controller, provisioning a node to act as a dedicated host for Neutron networking, and deploying a Compute node.

Procedure C.1. Using Foreman to deploy Neutron

1. **Deploy the Controller**

   a. Open a web browser and browse to the Foreman webpage. Log in and select the **`Hosts`** tab.

   b. Select the Controller host.

   c. In the **`Host`** sub-tab, select the **`Edit`** button.

   d. In the **`Host Group`** pull-down, select **`Controller (Neutron)`**.

   e. Select the **`Submit`** button.

   f. Ensure the following parameters are set to these values. Other parameters should be configured as normal.

      - enable_tunneling: 'True'
      - ovs_tunnel_types: vxlan
      - ovs_vlan_ranges: physext
      - tenant_network_type: gre
      - tunnel_id_ranges: 1:1000

   g. Deploy the host group with the following command:

   ```
   # puppet agent --test --debug --verbose
   ```

2. **Deploy the Networker**

   a. Open a web browser and browse to the Foreman webpage. Log in and select the **`Hosts`** tab.

   b. Select the *Neutron* host.

   c. In the **`Host`** sub-tab, select the **`Edit`** button.

   d. In the **`Host Group`** pull-down, select **`Neutron Networker`**.

   e. Select the **`Submit`** button.

   f. Ensure the following parameters are set to these values. Other parameters should be configured as normal.

      - enable_tunneling: 'True'
      - external_network_bridge: br-ex
      - ovs_bridge_mappings: physext:br-ex
      - ovs_bridge_uplinks: br-ex:eth0
      - ovs_tunnel_types: vxlan
      - ovs_vlan_ranges: physext
      - tenant_network_type: gre

- tunnel_id_ranges: 1:1000

g. Deploy the host group with the following command:

```
# puppet agent --test --debug --verbose
```

3. **Deploy the Compute node**

a. Open a web browser and browse to the Foreman webpage. Log in and select the **Hosts** tab.

b. Select the *Compute* host.

c. In the **Host** sub-tab, select the **Edit** button.

d. In the **Host Group** pull-down, select **Compute (Neutron)**.

e. Select the **Submit** button.

f. Ensure the following parameters are set to these values. Other parameters should be configured as normal.

- enable_tunneling: 'True'
- ovs_bridge_mappings: physnet:br-ex
- ovs_bridge_uplinks: br-ex:eth0
- ovs_tunnel_types: vxlan
- ovs_vlan_ranges: physnet
- tenant_network_type: gre
- tunnel_id_ranges: 1:1000

g. Deploy the host group with the following command:

```
# puppet agent --test --debug --verbose
```

Report a bug

C.2. Testing that SSL certificates are set up correctly

To test that you have set up SSL certificates correctly, you can write a python script as set out below. Save this script as, say, testcerts.py

```
import sys
from qpid.messaging import *

broker = sys.argv[1]

address = "amq.topic"

connection = Connection(broker)
connection.transport = 'ssl'
connection.port = 5671

try:
  connection.open()
  session = connection.session()

  sender = session.sender(address)
  receiver = session.receiver(address)

  sender.send(Message("Hello world!"));

  message = receiver.fetch()
  print message.content
  session.acknowledge()
```

```
except MessagingError,m:
  print m

connection.close()
```

To run this script, use the command:

```
$ python testcerts.py ${HOSTNAME}
```

Substitute your host's name for ${HOSTNAME}.

Report a bug

Service Log Files

D.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 D.1. Log Files

File name	Description
api.log	The log of the API service (**openstack-cinder-api**).
cinder-manage.log	The log of the management interface (**cinder-manage**).
scheduler.log	The log of the scheduler service (**openstack-cinder-scheduler**).
volume.log	The log of the volume service (**openstack-cinder-volume**).

Report a bug

D.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 D.2. Log Files

File name	Description
api.log	The log of the API service (**openstack-nova-api**).
cert.log	The log of the X509 certificate service (**openstack-nova-cert**). This service is only required by the EC2 API to the Compute service.
compute.log	The log file of the Compute service itself (**openstack-nova-compute**).
conductor.log	The log file of the conductor (**openstack-nova-conductor**) that provides database query support to the Compute service.
consoleauth.log	The log file of the console authentication service (**openstack-nova-consoleauth**).
network.log	The log of the network service (**openstack-nova-network**). Note that this service only runs in environments that are *not* configured to use OpenStack Networking.
nova-manage.log	The log of the management interface (**nova-manage**).
scheduler.log	The log of the scheduler service (**openstack-nova-scheduler**).

Report a bug

D.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 D.3. Log Files

File name	Description
access_log	The access log details all attempts to access the web server.

File name	Description
error_log	The error log details all unsuccessful attempts to access the web server and the reason for each failure.

Report a bug

D.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 D.4. Log File

File name	Description
keystone.log	The log of the Identity service (**openstack-keystone**).

Report a bug

D.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 D.5. Log Files

File name	Description
api.log	The log of the API service (**openstack-glance-api**).
registry.log	The log of the image registry service (**openstack-glance-registry**).

Report a bug

D.6. Networking Service Log Files

The log files of the networking service are stored in the **/var/log/neutron/** directory of the host on which each service runs.

Table D.6. Log Files

File name	Description
dhcp-agent.log	The log for the DHCP agent (**neutron-dhcp-agent**).
l3-agent.log	The log for the L3 agent (**neutron-l3-agent**).
lbaas-agent.log	The log for the Load Balancer as a Service (LBaaS) agent (**neutron-lbaas-agent**).
linuxbridge-agent.log	The log for the Linux Bridge agent (**neutron-linuxbridge-agent**).
metadata-agent.log	The log for the metadata agent (**neutron-metadata-agent**).
openvswitch-agent.log	The log for the Open vSwitch agent (**neutron-openvswitch-agent**).
server.log	The log for the OpenStack Networking service itself (**neutron-server**).

Report a bug

D.7. Object Storage Service Log Files

The log file of the Object Storage service is stored in the **/var/log/** directory of the service's host.

Table D.7. Log File

File name	Description
`swift-startup.log`	The log for the object storage proxy service.

Report a bug

Revision History

Revision 4-80	Fri Oct 03 2014	Tahlia Richardson

BZ#1093354 - Corrected the title and content of the Compute (Nova) chapter to Compute (Nova Network).
BZ#1098569, BZ#1098610 - Updated the values of parameters in Using Foreman to Deploy Neutron.

Revision 4-79	Fri Sep 26 2014	Zac Dover

BZ#1107833 - zdover - Updated the procedure for configuring the Foreman installer.
BZ#1053897 - zdover - Updated the procedure for configuring the Foreman installer.

Revision 4-78	Thu Sep 25 2014	Zac Dover

zdover - build for the RHCI

Revision 4-73	Mon Sep 15 2014	Andrew Dahms

BZ#1107833 - adahms - Updated the procedure for configuring the Foreman installer.
BZ#1093362 - adahms - Removed the step on configuring the provisioning variable for the Foreman installer.
BZ#1092708 - adahms - Updated the formatting of the commands used to set firewall ports for Foreman.
BZ#1092687 - adahms - Updated the name of parameters in the section on disabling Network Manager.
BZ#1075020 - adahms - Added further detail on the default values for Foreman parameters.
BZ#1051361 - adahms - Updated the formatting of commands in the section on deploying Neutron using Foreman.

Revision 4-77	Tue Mar 25 2014	Don Domingo

Final Version for Red Hat Enterprise Linux OpenStack Platform Maintenance Release 4.0.3.

Revision 4-72	Mon Mar 24 2014	Martin Lopes

BZ#988923 - mlopes - Added note regarding loadbalancing drivers.
BZ#988892 - mlopes - Added Modular Layer 2 (ML2) overview.
BZ#996762 - mlopes - Applied consistent PHYSNET naming to Provider Networking.
BZ#1040208 - mlopes - Added comparison of the Tenant and Provider network types.

Revision 4-71	Fri Mar 14 2014	Bruce Reeler

BZ#969726 - breeler - Rewrote section 8.2.3. Configuring TLS/SSL. Added appendix containing Python script to test SSL certificate setup.

Revision 4-70	Thu Mar 13 2014	Bruce Reeler

BZ#1074343 - breeler - Rewrote PackStack Deployment Utility description.

Revision 4-69	Tue Mar 04 2014	Don Domingo

Final Version for Red Hat Enterprise Linux OpenStack Platform Maintenance Release 4.0.2.

Revision 4-66	Tue Feb 25 2014	Scott Radvan

BZ#960334 - sradvan - Added tcp_keepidle default for Nova SSL parameters.
BZ#1067000 - ddomingo - Corrected path of **`keystone`** configuration file in a procedure.

Revision 4-65	Wed Feb 19 2014	Bruce Reeler

BZ#990584 - breeler - Updated the note in '9.4.1. Setting the Administration Token' that token_flush cmd needs to be run every minute.

Revision 4-64	Wed Feb 19 2014	Deepti Navale

BZ#1037283 - dnavale - Tech review updates to section 'Enabling Image Loading via the Local File System'.

Revision 4-63	Wed Feb 19 2014	Summer Long

BZ#988837 - slong - Added 'Basic Qpid Configuration' section.

Revision 4-62	Tue Feb 18 2014	Don Domingo

BZ#974897 - ddomingo - fixed consistency in userinput tagging.
BZ#1059126 - ddomingo - simplified Identity public endpoint setup instructions by removing all steps requiring hash ID retrieval and usage.
BZ#973508 - ddomingo - fixed alignment consistency in all figures.

Revision 4-60 **Wed Feb 12 2014** **Scott Radvan**

BZ#1038346 - sradvan - Add HA MySQL host group description.
BZ#1032303 - dnavale - Included a note advising the use of 'django-secure' with dashboard.
BZ#1038864 - sradvan - Foreman provisioning modes.
BZ#969906 - dnavale - QE review comments updated.

Revision 4-56 **Mon Feb 03 2014** **Deepti Navale**

BZ#1059977 - dnavale - Updated 'Creating a Router', ' Controlling State of an Instance', and 'Attaching a Volume to an Instance' sections with new screenshots. Updated procedure for the 'Associating a Floating IP with an Instance' section. Updated the procedure and screenshot for 'Creating a Volume', 'Launch an Instance', 'Adding a Rule to a Security Group', and 'Uploading a Disk Image' sections.
BZ#990581 - slong - Updated with review comments, and small edits.

Revision 4-56 **Wed Jan 29 2014** **Scott Radvan**

BZ#1053897 - sradvan - Further Foreman script changes.

Revision 4-55 **Wed Jan 29 2014** **Scott Radvan**

BZ#1057058 - sradvan - Explain that Nova or Neutron are now available for networking.
BZ#1053899 - sradvan - Several Foreman parameter and script file changes.

Revision 4-54 **Fri Jan 24 2014** **Deepti Navale**

BZ#969906 - dnavale - Included new chapter 'Touring the Cloud' describing the dashboard landing page.
BZ#1044078 - ddomingo - Clarified that all RHOS4 networking requirements are now built into RHEL6.5 default kernel

Revision 4-53 **Wed Jan 22 2014** **Summer Long**

Final Version for Red Hat Enterprise Linux OpenStack Platform Maintenance Release 4.0.1.

Revision 4-49 **Tue Jan 21 2014** **Don Domingo**

BZ#987711 - Edited documentation on configuring external networks.
BZ#1043352 - Added information on integrating Telemetry alarms with Orchestration.
BZ#1048634 - Updated minimum kernel requirements for network namespaces.

Revision 4-48 **Mon Jan 20 2014** **Scott Radvan**

BZ#1053897 - Foreman parameter rename.

Revision 4-47 **Mon Jan 20 2014** **Scott Radvan**

BZ#978002 - Add list of supported guest operating systems to Introduction.
BZ#1051361 - Slice Foreman-Neutron Appendix into subsections to improve appearance.

Revision 4-46 **Fri Jan 17 2014** **Scott Radvan**

BZ#1049635 - Remove Foreman Technology Preview warning from Foreman installation procedure.

Revision 4-45 **Fri Jan 17 2014** **Deepti Navale**

BZ#978652 - Updated parameters to the kernel arguments to ensure console output is visible via nova, the dashboard, etc.

Revision 4-44 **Fri Jan 17 2014** **Scott Radvan**

BZ#1051361 - Add Foreman-Neutron Appendix

Revision 4-11 **Fri Jan 10 2014** **Martin Lopes**

BZ#1033204 - Added note describing Foreman FQDN requirements.
BZ#1045248 - Added step to enable RHELOSP repo.

Revision 4-10 **Wed Dec 18 2013** **Summer Long**

Final Version for Red Hat Enterprise Linux OpenStack Platform 4.0.

Revision 4-9.1 **Wed Dec 18 2013** **Scott Radvan**

BZ#1033152 - sradvan - Refine the definition of the openstack-cinder-backup service.

Revision 4-9 **Tue Dec 17 2013** **Don Domingo, Summer Long**

BZ#973201 - ddomingo - Added connectivity setup details for Orchestration service.
BZ#995508 - slong - Reworked legal notice.

Revision 4-8.1 **Mon Dec 16 2013** **Summer Long**

BZ#1038372 - slong - Repository section reworked to refer to the Release Notes for specific channel info. Also moved Software Collection note over to Release Notes.

Revision 4-6 **Mon Dec 09 2013** **Don Domingo, Martin Lopes**

BZ#990581 - ddomingo - Added new instructions on enabling direct access of images between Glance and Nova across different nodes.
BZ#988955 - mlopes - Updated with Configuration Reference Guide note.

Revision 4-5 **Thu Dec 05 2013** **Summer Long**

BZ#1011760 - New Backup and Recovery section

Revision 4-4 **Thu Dec 05 2013** **Scott Radvan**

BZ#1024577 - Include details on enabling RHSCL repository.
BZ#960666 - Add glance.conf SSL config options.
BZ#960334 - Add nova.conf SSL config options.

Revision 4-1 **Wed Dec 04 2013** **Don Domingo**

BZ#1009641 - Added content and cross-references to clarify significance of QPID username for message broker configuration.
BZ#1014365 - Revised instructions for building Object Storage Service ring files for clarity (as suggested).

Revision 3-51 **Wed Dec 04 2013** **Summer Long**

BZ#995508 - Updated legal notice.

Revision 3-50 **Tue Dec 03 2013** **Deepti Navale, Don Domingo**

BZ#988799 - dnavale- Included a link to 'Configuring EC2 API' information in the Configuration Reference Guide.
BZ#988927: - ddomingo - added instructions for enabling image loading via local file system, including glance and nova-compute topics

Revision 3-49 **Tue Dec 03 2013** **Deepti Navale, Martin Lopes**

BZ#989733 - Included information about HAProxy configuration and overview. New topic 'Configuring Load-Balancing OpenStack API Services', listing steps for configuring load balancing on all OpenStack tenants.
BZ#1002779 - mlopes - Added Open vSwitch Tunneling to 'Configuring the Open vSwitch Plug-in Agent'
BZ#990555 - mlopes - Updated note to include Open vSwitch TSO/GSO support for GRE

Revision 3-48 **Mon Dec 2 2013** **Scott Radvan**

BZ#960324 - sradvan - Add cinder.conf SSL options.

Revision 3-47 **Thu Nov 28 2013** **Deepti Navale, Don Domingo, Martin Lopes**

BZ#993962 - dnavale - Included 'Troubleshooting OpenStack Networking Issues' section listing the commands and procedu to debug network issues.
BZ#973201 - ddomingo - Added content for Heat Installation and Configuration.
BZ#1010307 - mlopes - Added OpenStack Block Storage host group to Foreman.
BZ#998048 - dnavale - Included new section 'Enabling the Identity Service to Use Public Key Infrastructure Files'.

Revision 3-46 **Tue Nov 26 2013** **Deepti Navale, Don Domingo, Martin Lopes**

BZ#889370 - dnavale - Updated '9.4.1. Setting the Administration Token' section with a note about the new command to manage tokens by flushing the expired ones.
BZ#995906 - dnavale - Updated nova-manage with nova in Managing Quotas and added function for listing default quotas.
BZ#1030129 - dnavale - Updated a note in 'Building a Custom Virtual Machine Image' section about the Disk Image Builder tool.
BZ#973191 - ddomingo - Initial topic creation
BZ#1009634 - mlopes - Updated LDAP example distinguished names.
BZ#990557 - mlopes - Updated '13.7.1. Configuring the Open vSwitch Plug-in Agent' section with a note for Open vSwitch TCP segmentation offload (TSO).

Revision 3-45	**Fri Nov 22 2013**	**Martin Lopes**

BZ#1033381 - Updated path to Foreman installer.

Revision 3-44	**Thu Nov 21 2013**	**Don Domingo, Martin Lopes, Summer Long**

BZ#985040 - mlopes - Revised LDAP integration. Added LDAP hardening.
BZ#988937 - mlopes - Added Identity Service Assignments.
BZ#988935 - BZ#988934 - ddomingo - Added instructions for configuring keystone external auth (X509 or Kerberos) via HTTPD (not tech reviewed yet).
BZ#974236 - slong - Updated with Configuration Reference Guide note.

Revision 3-43	**Fri Nov 15 2013**	**Summer Long**

BZ#974236 - Removed example configuration files from Appendix (files are now housed in the Configuration Reference Guide).

Revision 3-42	**Fri Nov 15 2013**	**Don Domingo**

BZ#976598: - Revised IPTables instructions to file editing for consistency (also minor edits).
BZ#1020071 - Updated package name with Havana version.

Revision 3-41	**Tue Nov 12 2013**	**Martin Lopes**

BZ#973845 - Removed duplicate language for RHN requirement.

Revision 3-40	**Tue Nov 12 2013**	**Summer Long**

BZ#1025133-Updated basic text, configuration files throughout for quantum -> neutron code changes and Havana updates. Removed 'Grizzly' references.
BZ#988039-Removed 'tech preview' note for cells, and removed Note about tech previews.

Revision 3-39	**Fri Oct 25 2013**	**Scott Radvan**

Test publishing of new Beta version.

Revision 3-38	**Mon Sep 16 2013**	**Scott Radvan**

BZ#1004575 - Only show configuring v2.0 of the endpoint API.

Revision 3-37	**Fri Sep 06 2013**	**Scott Radvan**

BZ#997589 - Update procedure formatting.

Revision 3-36	**Fri Sep 06 2013**	**Scott Radvan**

BZ#997589 - Specify correct package for VNC service.

Revision 3-35	**Tue Sep 03 2013**	**Scott Radvan**

BZ#975268 - Move component listings into tables.

Revision 3-34	**Mon Sep 02 2013**	**Scott Radvan**

BZ#984897 - Fix incorrect /etc file path.

Revision 3-33	**Thu Aug 08 2013**	**Stephen Gordon**

Updated brand.

Revision 3-32	**Tue Aug 06 2013**	**Stephen Gordon**

BZ#988471 - Updated explanation of RHS SELinux boolean.
BZ#988151 - Added brief explanation of NFS component of multi-backend example.

Revision 3-31 **Tue Aug 06 2013** **Stephen Gordon**

BZ#988471 - Add setting of virt_use_fusefs SELinux boolean to RHS backend configuration.
BZ#984898 - Added missing "on" argument to setsebool command.
BZ#989832 - Removed extra apostrophe in certutil example.
BZ#980547 - Replace /etc/nova/nova/conf with /etc/nova/nova.conf in Updating the Compute Configuration section.
BZ#988150 - Updated Block Storage multi-backend configuration content to provide correct Red Hat Storage driver.
BZ#988151 - Updated Block Storage multi-backend configuration to provide a valid NFS example.
BZ#991099 - Updated database connection string instructions in "Configuring the Networking Service" to refer to the correc
file.
BZ#928038 - Correct description of bridge_mappings configuration key.
BZ#981430 - Start openstack-nova-consoleauth service.

Revision 3-30 **Wed Jul 17 2013** **Stephen Gordon**

BZ#981430 - Added instructions on the use of the Compute console authentication service.
BZ#980559 - Added API, conductor, and scheduler services to compute installation instructions.
BZ#980902 - Added missing authentication configuration for the Compute service.
BZ#980860 - Added instructions for starting Libvirtd service before the Compute (Nova) service.
BZ#928038 - Added **`bridge_mappings`** and **`physical_interface_mappings`** configuration keys to networking documentation.

Revision 3-29 **Mon Jul 08 2013** **Stephen Gordon**

BZ#980565 - Added missing database grant for 'dash' database user.
BZ#975419 - Fixed Nova API restart command to refer to the correct service.

Revision 3-26 **Mon Jul 01 2013** **Stephen Gordon**

BZ#979150 - Updated entitlement information.
BZ#978657 - Fixed Identity service v3 endpoint definition.
BZ#979004 - Corrected Compute Identity records.

Revision 3-24 **Mon Jun 24 2013** **Stephen Gordon, Summer Long**

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.

Revision 3-23 **Thu Jun 20 2013** **Stephen Gordon**

BZ#973845 - Removed duplication in "Register to Red Hat Network" instructions.
BZ#877820 - Updated spacing of qemu settings.

Revision 3-22 **Tue Jun 18 2013** **Stephen Gordon, Summer Long**

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.

Revision 3-21 **Thu Jun 13 2013** **Steve Gordon**

Red Hat OpenStack 3.0 Preview update.

Revision 3-13 **Wed May 29 2013** **Steve Gordon**

Red Hat OpenStack 3.0 Preview.