Red Hat OpenStack Red Hat OpenStack 1.0 (Essex) Preview Getting Started Guide

Getting Started with Red Hat OpenStack 1.0 (Essex) Preview

Red Hat Cloud Engineering

Red Hat OpenStack Red Hat OpenStack 1.0 (Essex) Preview Getting Started Guide

Getting Started with Red Hat OpenStack 1.0 (Essex) Preview

Red Hat Cloud Engineering

Keywords

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

Table of Contents

Preface 4
1. Document Conventions 4
1.1. Typographic Conventions 4
1.2. Pull-quote Conventions 5
1.3. Notes and Warnings 6

Chapter 1. Introduction 7
1.1. Repository Configuration 8
1.2. Getting Started 8

Chapter 2. Keystone (Identity) 9
2.1. Installation and Initial Configuration 9
2.2. Creating Users 10

Chapter 3. Glance (Images) 14
3.1. Adding an Image to Glance 16

Chapter 4. Nova (Compute) 17

Chapter 5. Horizon (Dashboard) 21
5.1. VNC Consoles 22

Chapter 6. Swift (Object Storage) 23
6.1. Creating the Swift Ring Files 23
6.2. Configuring the Swift Proxy 24
6.3. Configuring Keystone 25
6.4. Configuring Swift Storage Nodes 26
6.5. Testing Swift 29

Chapter 7. Exercises 30
7.1. Starting an Instance 30
7.2. Creating and Attaching a Volume 31
7.3. Creating a Snapshot 33
7.4. Creating Security Groups 35
7.5. Adding Floating IP Addresses 36
7.6. Controlling Instance State (Suspend, Resume, Reboot, Terminate) 37
7.7. Deleting Instances 37

Revision History 39

Preface

1. Document Conventions

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

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

1.1. Typographic Conventions

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

`Mono-spaced Bold`

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

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

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

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

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

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

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

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

Proportional Bold

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

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

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

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

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

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

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

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

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

> Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

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

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

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

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

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

import javax.naming.InitialContext;

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

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

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

1.3. Notes and Warnings

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

Note

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

Important

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

Warning

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

Chapter 1. Introduction

OpenStack is a collection of services that can be used to provide Infrastructure as a Service (IaaS). This guide aims to provide two things:

1. Instructions on how to install Red Hat OpenStack on Red Hat Enterprise Linux (RHEL).
2. Instructions on how to get started with OpenStack by configuring a single node environment.

For more in-depth documentation about the configuration and administration of more complex installations, please see http://docs.openstack.org.

OpenStack includes the following services:

Keystone (Identity)
A common identity service that provides authentication for other services.

Glance (Image)
A service that acts as a registry for virtual machine images.

Nova (Compute)
A service that manages virtual machines and the storage and networking associated with them.

Horizon (Dashboard)
A web-based interface for interacting with the rest of the services.

Swift (Object)
Object storage.

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

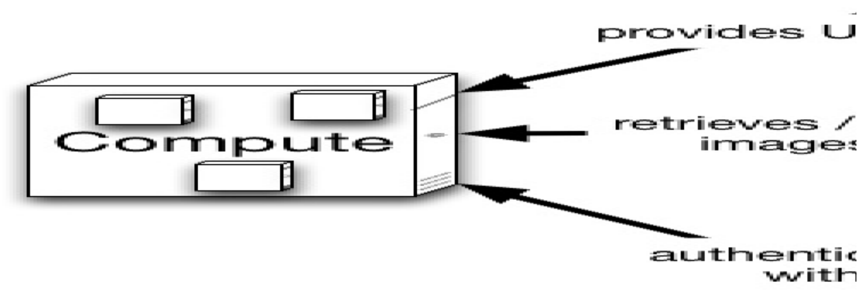

Figure 1.1. Relationships between OpenStack services

1.1. Repository Configuration

Before you can install the OpenStack packages, you must assign the appropriate subscriptions to the system. First register the system with Red Hat Network by running the **`subscription-manager register`** command. Supply your Red Hat Network username and password when prompted:

```
$ sudo subscription-manager register
```

To apply the appropriate subscriptions to the system, locate the identifiers for the relevant subscription pools. Use the **`subscription-manager list`** command to find these IDs. The system requires subscriptions for both OpenStack and Red Hat Enterprise Linux Server:

```
$ sudo subscription-manager list --available | grep -A8 "Red Hat Enterprise
Linux Server"
$ sudo subscription-manager list --available | grep -A8 "OpenStack"
```

Use the **`subscription-manager`** command to assign the OpenStack and Red Hat Enterprise Linux Server subscriptions to the system:

```
$ sudo subscription-manager subscribe --pool=POOLID
```

1.2. Getting Started

Before you continue with the installation and configuration of OpenStack components, you must install some other packages:

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

The **`openstack-utils`** package will provide a few important utilities that will be used in the rest of the instructions:

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

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

`openstack-status`
: Displays service status information.

Chapter 2. Keystone (Identity)

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

2.1. Installation and Initial Configuration

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

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

Keystone uses a MySQL database. Use the **`openstack-db`** utility to create and initialize the tables for Keystone. If MySQL has not yet been installed on this server, the script will manage that as well:

```
$ sudo openstack-db --init --service keystone
```

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

```
$ export SERVICE_TOKEN=$(openssl rand -hex 10)
$ export SERVICE_ENDPOINT=http://127.0.0.1:35357/v2.0
$ echo $SERVICE_TOKEN > /tmp/ks_admin_token
```

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

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

Now start the Keystone service:

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

Finally, verify that the Keystone service is running and that no errors are present in the Keystone log file:

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

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

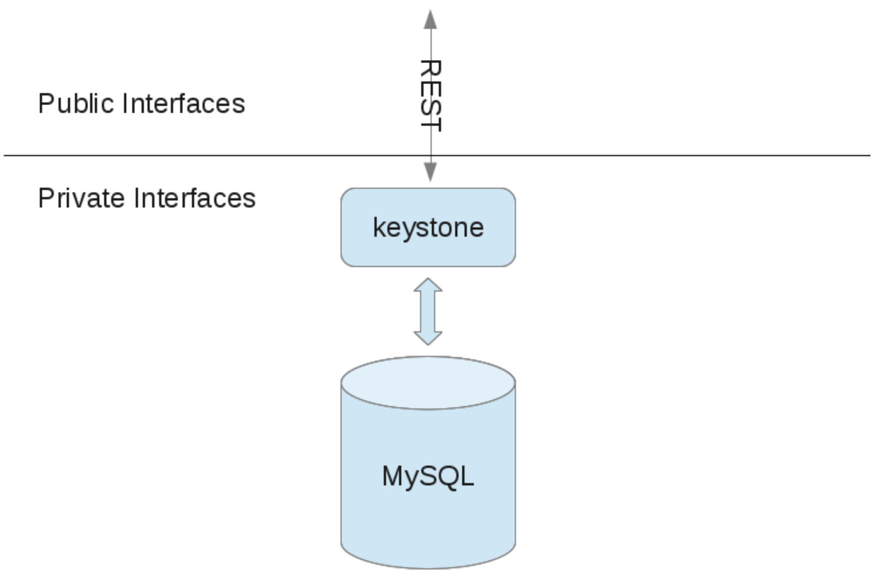

Figure 2.1. Keystone installed and configured

2.2. Creating Users

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

```
$ keystone user-create --name admin --pass secret
+----------+-----------------------------------+
| Property |              Value                |
+----------+-----------------------------------+
| email    | None                              |
| enabled  | True                              |
| id       | 94d659c3c9534095aba5f8475c87091a  |
| name     | admin                             |
| password | ...                               |
| tenantId | None                              |
+----------+-----------------------------------+
$ keystone role-create --name admin
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
| id       | 78035c5d3cd94e62812d6d37551ecd6a |
| name     | admin                            |
+----------+----------------------------------+
$ keystone tenant-create --name admin
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description | None                             |
| enabled     | True                             |
| id          | 6f8e3e36c4194b86b9a9b55d4b722af3 |
| name        | admin                            |
+-------------+----------------------------------+
```

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

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

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

```
export OS_USERNAME=admin
export OS_TENANT_NAME=admin
export OS_PASSWORD=secret
export OS_AUTH_URL=http://127.0.0.1:35357/v2.0/
export PS1="[\u@\h \W(keystone_admin)]\$ "
```

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

```
$ unset SERVICE_TOKEN
$ unset SERVICE_ENDPOINT
$ . ~/keystonerc_admin
$ keystone user-list
+----------------------------------+---------+-------+-------+
|                id                | enabled | email |  name |
+----------------------------------+---------+-------+-------+
| 94d659c3c9534095aba5f8475c87091a | True    | None  | admin |
+----------------------------------+---------+-------+-------+
```

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

```
$ keystone service-create --name=keystone --type=identity \
  --description="Keystone Identity Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description | Keystone Identity Service        |
| id          | a8bff1db381f4751bd8ac126464511ae |
| name        | keystone                         |
| type        | identity                         |
+-------------+----------------------------------+
$ keystone endpoint-create --region RegionOne \
  --service_id a8bff1db381f4751bd8ac126464511ae \
  --publicurl 'http://127.0.0.1:5000/v2.0' \
  --adminurl 'http://127.0.0.1:35357/v2.0' \
  --internalurl 'http://127.0.0.1:5000/v2.0'
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| adminurl    | http://127.0.0.1:35357/v2.0      |
| id          | 1295011fdc874a838f702518e95a0e13 |
| internalurl | http://127.0.0.1:5000/v2.0       |
| publicurl   | http://127.0.0.1:5000/v2.0       |
| region      | RegionOne                        |
| service_id  | a8bff1db381f4751bd8ac126464511ae |
+-------------+----------------------------------+
```

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

```
$ keystone user-create --name username --pass secret
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
| email    | None                             |
| enabled  | True                             |
| id       | 1d59c0bfef9b4ea9ab63e2a058e68ae0 |
| name     | username                         |
| password | ...                              |
| tenantId | None                             |
+----------+----------------------------------+
$ keystone role-create --name user
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
| id       | 8261ac4eabcc4da4b01610dbad6c038a |
| name     | user                             |
+----------+----------------------------------+
$ keystone tenant-create --name rhsummit
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description | None                             |
| enabled     | True                             |
| id          | 05816b0106994f95a83b913d4ff995eb |
| name        | rhsummit                         |
+-------------+----------------------------------+
```

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

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

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

```
export OS_USERNAME=username
export OS_TENANT_NAME=rhsummit
export OS_PASSWORD=secret
export OS_AUTH_URL=http://127.0.0.1:5000/v2.0/
export PS1="[\u@\h \W(keystone_username)]\$ "
```

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

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

Chapter 3. Glance (Images)

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

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

Glance makes use of MySQL to store metadata about images. Use **`openstack-db`** to initialize the database for use with Glance:

```
$ sudo openstack-db --init --service glance
```

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

```
$ sudo openstack-config --set /etc/glance/glance-api.conf paste_deploy flavor
keystone
$ sudo openstack-config --set /etc/glance/glance-api-paste.ini \
  filter:authtoken admin_token $(cat /tmp/ks_admin_token)
$ sudo openstack-config --set /etc/glance/glance-registry.conf \
  paste_deploy flavor keystone
$ sudo openstack-config --set /etc/glance/glance-registry-paste.ini \
  filter:authtoken admin_token $(cat /tmp/ks_admin_token)
```

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

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

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

```
$ . ~/keystonerc_admin
$ keystone service-create --name=glance --type=image --description="Glance
Image Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description | Glance Image Service             |
| id          | 1447f490e9784aa8890f9e39ddaa8d44 |
| name        | glance                           |
| type        | image                            |
+-------------+----------------------------------+
$ keystone endpoint-create --service_id 1447f490e9784aa8890f9e39ddaa8d44 \
  --publicurl http://127.0.0.1:9292/v1 \
  --adminurl http://127.0.0.1:9292/v1 \
  --internalurl http://127.0.0.1:9292/v1
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| adminurl    | http://127.0.0.1:9292/v1         |
| id          | 9693e1af560843f68e9e91f2b3664a19 |
| internalurl | http://127.0.0.1:9292/v1         |
| publicurl   | http://127.0.0.1:9292/v1         |
| region      | regionOne                        |
| service_id  | 1447f490e9784aa8890f9e39ddaa8d44 |
+-------------+----------------------------------+
```

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

```
$ . ~/keystonerc_username
$ glance index
```

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

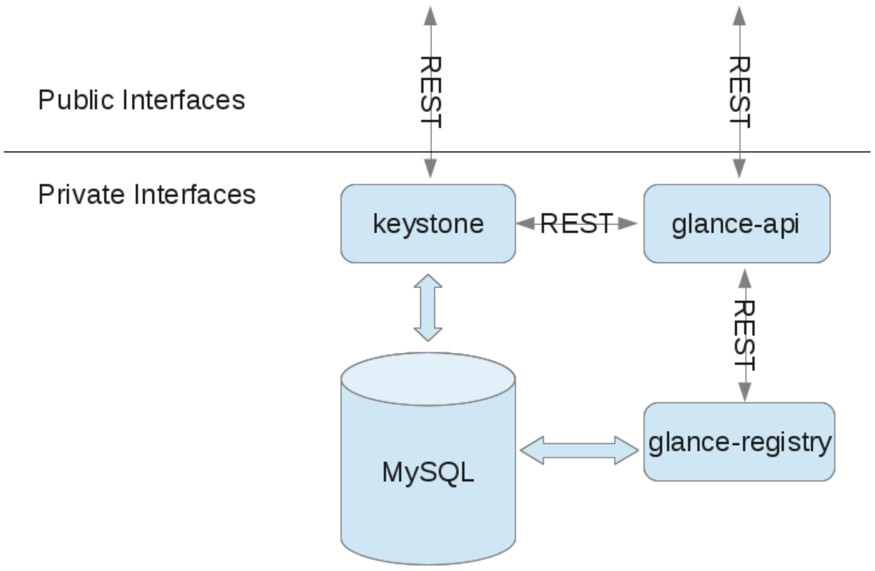

Figure 3.1. Keystone and Glance installed and configured

3.1. Adding an Image to Glance

Glance is set up and configured at this point. If you would like to follow the examples in Chapter 7, *Exercises* you have to register an image with Glance. If you would like to create your own image, refer to the Virtualization Host Configuration and Guest Installation Guide.

For this example, assume that you have an image of RHEL 6.2 named **`rhel62_x86_64.qcow2`** that you would like to add to Glance. This is done using the **`glance add`** command. Once the image has been added, it will show up in the list of available images. You can also retrieve details about the image from Glance using the **`glance show`** command:

```
$ glance add name="RHEL 6.2" is_public=true disk_format=qcow2 \
  container_format=bare < /srv/rhsummit/images/rhel62_x86_64.qcow2
Uploading image 'RHEL 6.2'
==========================================[100%] 155.966310M/s, ETA  0h  0m  0s
Added new image with ID: 17a34b8e-c573-48d6-920c-b4b450172b41
$ glance index
ID                                   Name                           Disk Format
Container Format     Size
------------------------------------ ------------------------------ ------------
-------- -------------------- --------------
17a34b8e-c573-48d6-920c-b4b450172b41 RHEL 6.2                       qcow2
bare                 213581824
$ glance show 17a34b8e-c573-48d6-920c-b4b450172b41
URI: http://127.0.0.1:9292/v1/images/17a34b8e-c573-48d6-920c-b4b450172b41
Id: 17a34b8e-c573-48d6-920c-b4b450172b41
Public: Yes
Protected: No
Name: RHEL 6.2
Status: active
Size: 213581824
Disk format: qcow2
Container format: bare
Minimum Ram Required (MB): 0
Minimum Disk Required (GB): 0
Owner: 05816b0106994f95a83b913d4ff995eb
```

Chapter 4. Nova (Compute)

So far you have installed Keystone for identity management and Glance for storage of virtual machine images. Now you will install Nova, which is responsible for the life-cycle of virtual machines. The first step is to install the **openstack-nova** package:

```
$ sudo yum install openstack-nova
```

Nova makes use of MySQL just like Keystone and Glance. Use the **openstack-db** utility to initialize the database for Nova:

```
$ sudo openstack-db --init --service nova
```

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

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

You must also configure Nova to use the correct network interfaces:

```
$ sudo openstack-config --set /etc/nova/nova.conf DEFAULT flat_interface em1
$ sudo openstack-config --set /etc/nova/nova.conf DEFAULT public_interface
em1
```

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

Warning

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

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

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

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

One of the Nova services is **nova-volume**, which is responsible for managing persistent storage for virtual machines. There are multiple back ends for **nova-volume**. The one that is enabled by default utilizes an LVM volume group called **nova-volumes** to allocate storage for virtual machines on demand. For testing purposes, you can create a simple file-backed volume group.

Warning

This example setup for the nova-volumes group is only intended as a quick-and-easy way to try out persistent volumes in a testing environment. This method should not be used in a production environment.

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

You have completed the prerequisite steps for being able to start the Nova services. Run the following commands to start up all of the Nova services:

```
$ for srv in api cert network objectstore scheduler volume compute ; do \
   sudo service openstack-nova-$srv start ; \
  done
$ for srv in api cert network objectstore scheduler volume compute ; do \
   sudo chkconfig openstack-nova-$srv on ; \
  done
```

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

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

Now that the nova services are running, you may find it beneficial to monitor the nova logs during this lab, to do so you can open another terminal and tail the files in **/var/log/nova**:

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

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

```
$ . ~/keystonerc_admin
$ keystone service-create --name=nova --type=compute --description="Nova
Compute Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description | Nova Compute Service             |
| id          | 9f004f52a97e469b9983d5adefe9f6d0 |
| name        | nova                             |
| type        | compute                          |
+-------------+----------------------------------+
$ keystone endpoint-create --service_id 9f004f52a97e469b9983d5adefe9f6d0 \
  --publicurl "http://127.0.0.1:8774/v1.1/\$(tenant_id)s" \
  --adminurl "http://127.0.0.1:8774/v1.1/\$(tenant_id)s" \
  --internalurl "http://127.0.0.1:8774/v1.1/\$(tenant_id)s"
+-------------+------------------------------------------+
|   Property  |                  Value                   |
+-------------+------------------------------------------+
| adminurl    | http://127.0.0.1:8774/v1.1/$(tenant_id)s |
| id          | 247a56ec4fa94afca231aa0c304c7049         |
| internalurl | http://127.0.0.1:8774/v1.1/$(tenant_id)s |
| publicurl   | http://127.0.0.1:8774/v1.1/$(tenant_id)s |
| region      | regionOne                                |
| service_id  | 9f004f52a97e469b9983d5adefe9f6d0         |
+-------------+------------------------------------------+
```

You must also register the Nova volumes API with Keystone. Again, get the **`service_id`** used in the **`endpoint-create`** command from the output of the **`service-create`** command:

```
$ . ~/keystonerc_admin
$ keystone service-create --name=volume --type=volume --description="Nova
Volume Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description | Nova Volume Service              |
| id          | 9347a8dc271c44da9f9991b5c70a6361 |
| name        | volume                           |
| type        | volume                           |
+-------------+----------------------------------+
$ keystone endpoint-create --service_id 9347a8dc271c44da9f9991b5c70a6361 \
  --publicurl 'http://127.0.0.1:8776/v1/$(tenant_id)s' \
  --internalurl 'http://127.0.0.1:8776/v1/$(tenant_id)s' \
  --adminurl 'http://127.0.0.1:8776/v1/$(tenant_id)s'
+-------------+-----------------------------------------+
|   Property  |                  Value                  |
+-------------+-----------------------------------------+
| adminurl    | http://127.0.0.1:8776/v1/$(tenant_id)s |
| id          | b2a128758a274b09bb09337b99faa42a        |
| internalurl | http://127.0.0.1:8776/v1/$(tenant_id)s |
| publicurl   | http://127.0.0.1:8776/v1/$(tenant_id)s |
| region      | regionOne                               |
| service_id  | 9347a8dc271c44da9f9991b5c70a6361        |
+-------------+-----------------------------------------+
```

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

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

Run the following command to create a network that Nova can use to allocate IP addresses for instances:

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

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

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

Chapter 5. Horizon (Dashboard)

Installing Horizon, the OpenStack web Dashboard, is very easy to do. You install the **openstack-dashboard** package and then start **httpd**. There is no additional configuration needed. Once **httpd** is running, you should be able to log in to the dashboard using your OpenStack user's credentials at http://127.0.0.1/dashboard.

```
$ sudo yum install openstack-dashboard
$ sudo service httpd start
$ sudo chkconfig httpd on
```

If you have selinux in enforcing mode, then the selinux policies will prevent apache **httpd** from accessing the Keystone server. You have to relax the policies a bit by setting a selinux boolean:

```
$ sudo setsebool httpd_can_network_connect on
```

Load the dashboard in a web browser to confirm that it is running. The login page will look like this:

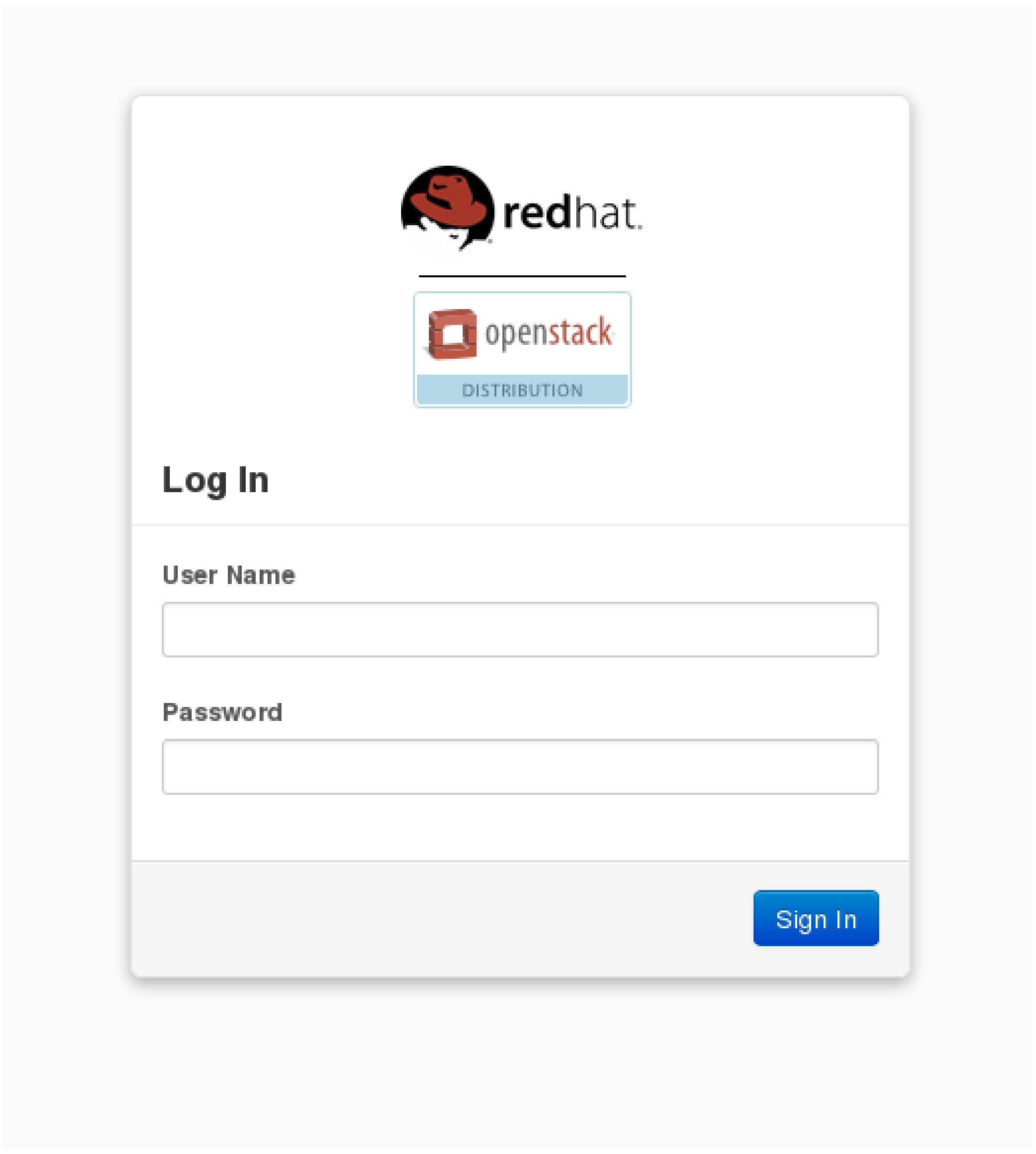

Figure 5.1. OpenStack login page

5.1. VNC Consoles

If you would like to enable VNC consoles to instances through Horizon, there is a bit more work to do. The following options should be set in **`/etc/nova/nova.conf`**. Replace **`NOVNCPROXY_FQDN`** with the address that dashboard users would use to be able to connect to the VNC proxy running on this machine:

```
novncproxy_host = 0.0.0.0
novncproxy_port = 6080
novncproxy_base_url=http://NOVNCPROXY_FQDN:6080/vnc_auto.html
vnc_enabled=true
vncserver_listen=127.0.0.1
vncserver_proxyclient_address=127.0.0.1
```

You must also start the **`novncproxy`** and **`consoleauth`** Nova services. The compute service must be restarted as well when VNC has been enabled:

```
$ sudo yum install openstack-nova-novncproxy
$ sudo service openstack-nova-novncproxy start
$ sudo service openstack-nova-consoleauth start
$ sudo chkconfig openstack-nova-novncproxy on
$ sudo chkconfig openstack-nova-consoleauth on
$ sudo service openstack-nova-compute restart
```

Chapter 6. Swift (Object Storage)

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

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

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

The first step is to install the Swift packages:

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

6.1. Creating the Swift Ring Files

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

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

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

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

Once the ring file is created you need to add the storage devices to each ring, starting with the accounts:

```
 $  sudo swift-ring-builder /etc/swift/account.builder add z1-
127.0.0.1:6002/z1d1 100
 $  sudo swift-ring-builder /etc/swift/account.builder add z1-
127.0.0.1:6002/z1d2 100
 $  sudo swift-ring-builder /etc/swift/account.builder add z2-
127.0.0.1:6002/z2d1 100
 $  sudo swift-ring-builder /etc/swift/account.builder add z2-
127.0.0.1:6002/z2d2 100
 $  sudo swift-ring-builder /etc/swift/account.builder add z3-
127.0.0.1:6002/z3d1 100
 $  sudo swift-ring-builder /etc/swift/account.builder add z3-
127.0.0.1:6002/z3d2 100
```

then the containers:

```
 $  sudo swift-ring-builder /etc/swift/container.builder add z1-
127.0.0.1:6001/z1d1 100
 $  sudo swift-ring-builder /etc/swift/container.builder add z1-
127.0.0.1:6001/z1d2 100
 $  sudo swift-ring-builder /etc/swift/container.builder add z2-
127.0.0.1:6001/z2d1 100
 $  sudo swift-ring-builder /etc/swift/container.builder add z2-
127.0.0.1:6001/z2d2 100
 $  sudo swift-ring-builder /etc/swift/container.builder add z3-
127.0.0.1:6001/z3d1 100
 $  sudo swift-ring-builder /etc/swift/container.builder add z3-
127.0.0.1:6001/z3d2 100
```

and then the objects:

```
 $  sudo swift-ring-builder /etc/swift/object.builder add z1-127.0.0.1:6000/z1d1
100
 $  sudo swift-ring-builder /etc/swift/object.builder add z1-127.0.0.1:6000/z1d2
100
 $  sudo swift-ring-builder /etc/swift/object.builder add z2-127.0.0.1:6000/z2d1
100
 $  sudo swift-ring-builder /etc/swift/object.builder add z2-127.0.0.1:6000/z2d2
100
 $  sudo swift-ring-builder /etc/swift/object.builder add z3-127.0.0.1:6000/z3d1
100
 $  sudo swift-ring-builder /etc/swift/object.builder add z3-127.0.0.1:6000/z3d2
100
```

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

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

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

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

should reveal:

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

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

```
 $  sudo touch /etc/swift/swift.conf
 $  sudo chmod 600 /etc/swift/swift.conf
 $  sudo openstack-config --set /etc/swift/swift.conf swift-hash \
swift_hash_path_suffix $(openssl rand -hex 10)
```

6.2. Configuring the Swift Proxy

Create a config file for the proxy and add contents as follows:

```
$ sudo tee /etc/swift/proxy-server.conf <<EOF
[DEFAULT]
bind_port = 8080
workers = 3
user = swift

[pipeline:main]
pipeline = healthcheck cache authtoken keystone proxy-server

[app:proxy-server]
use = egg:swift#proxy
allow_account_management = true
account_autocreate = true

[filter:cache]
use = egg:swift#memcache
memcache_servers = 127.0.0.1:11211

[filter:catch_errors]
use = egg:swift#catch_errors

[filter:healthcheck]
use = egg:swift#healthcheck

[filter:keystone]
paste.filter_factory = keystone.middleware.swift_auth:filter_factory
operator_roles = admin, SwiftOperator
is_admin = true
cache = swift.cache

[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
auth_host = 127.0.0.1
auth_port = 35357
auth_protocol = http
auth_uri = http://127.0.0.1:35357
# if its defined
admin_tenant_name = services
admin_user = swift
admin_password = secret
EOF
```

Followed by:

```
$ sudo chown -R swift:swift /etc/swift/
$ sudo service memcached start
$ sudo service openstack-swift-proxy start
```

6.3. Configuring Keystone

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

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

```
$ . ~/keystonerc_admin
```

Check if there is a services tenant:

```
$ keystone tenant-list
```

If not, create one:

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

Create a Swift user and set its password to 'secret':

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

Get the uuid of the admin role:

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

If no admin role exists, create one:

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

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

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

Create swift as an endpoint in Keystone:

```
$ keystone service-list
```

If the object-store service doesn't already exist, create one:

```
$ keystone service-create --name swift --type object-store \
    --description "Swift Storage Service"
$ keystone endpoint-create --service_id <serviceid> \
    --publicurl "http://127.0.0.1:8080/v1/AUTH_\$(tenant_id)s" \
    --adminurl "http://127.0.0.1:8080/v1/AUTH_\$(tenant_id)s" \
    --internalurl "http://127.0.0.1:8080/v1/AUTH_\$(tenant_id)s"
```

At this point you can start the Swift proxy:

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

6.4. Configuring Swift Storage Nodes

Three config files need to be created (one each for account, container and object):

```
$ sudo tee /etc/swift/account-server.conf <<EOF
[DEFAULT]
devices = /srv/node
bind_ip = 127.0.0.1
bind_port = 6002
mount_check = false
user = swift
log_facility = LOG_LOCAL2
workers = 3

[pipeline:main]
pipeline = account-server

[app:account-server]
use = egg:swift#account

[account-replicator]
concurrency = 1

[account-auditor]

[account-reaper]
concurrency = 1
EOF
```

```

$ sudo tee /etc/swift/container-server.conf <<EOF
[DEFAULT]
devices = /srv/node
bind_ip = 127.0.0.1
bind_port = 6001
mount_check = false
user = swift
log_facility = LOG_LOCAL2
workers = 3

[pipeline:main]
pipeline = container-server

[app:container-server]
use = egg:swift#container

[container-replicator]
concurrency = 1

[container-updater]
concurrency = 1

[container-auditor]

[container-sync]

EOF
```

```
$ sudo tee /etc/swift/object-server.conf <<EOF
[DEFAULT]
devices = /srv/node
bind_ip = 127.0.0.1
bind_port = 6000
mount_check = false
user = swift
log_facility = LOG_LOCAL2
workers = 3

[pipeline:main]
pipeline = object-server

[app:object-server]
use = egg:swift#object

[object-replicator]
concurrency = 1

[object-updater]
concurrency = 1

[object-auditor]
EOF
```

In this case we need a single block device to correspond with each device added to the Swift ring file. This Swift storage device should be a file system mounted at **`/srv/node/`**.

Note

You can use loopback devices for testing purposes, but if you are deploying Swift in a production environment, physical devices should be used.

Create and mount loopback devices:

```
$ for zone in 1 2 3 ; do
    for device in 1 2 ; do
   truncate /var/tmp/swift-device-z${zone}d${device} --size 5G
   LOOPDEVICE=$(losetup --show -f /var/tmp/swift-device-z${zone}d${device})
   mkfs.ext4 -I 1024 $LOOPDEVICE
   mkdir -p /srv/node/z${zone}d${device}
   mount -o noatime,nodiratime,nobarrier,user_xattr $LOOPDEVICE \
   /srv/node/z${zone}d${device}
      done
done
```

In a production environment you typically run **`chkconfig openstack-swift-whatever on`** and add the mounts to **`/etc/fstab`**, so that the Swift service starts automatically.

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

Start the account, container and object storage services:

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

6.5. Testing Swift

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

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

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

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

Chapter 7. Exercises

This section includes a number of exercises that use various features of the OpenStack installation that was set up using the instructions in earlier sections.

Note

Commands that begin with **`vm$`** as opposed to just **`$`** are commands that should be run inside a virtual machine.

7.1. Starting an Instance

When starting an instance using OpenStack, you must specify the ID for the flavor you want to use for the instance. A flavor is a resource allocation profile. For example, it specifies how many virtual CPUs and how much RAM your instance will get. To see a list of the available profiles, run the **`nova flavor-list`** command.

```
$ nova flavor-list
+----+-----------+-----------+------+-----------+------+-------+-------------+
| ID |    Name   | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor |
+----+-----------+-----------+------+-----------+------+-------+-------------+
| 1  | m1.tiny   | 512       | 0    | 0         |      | 1     | 1.0         |
| 2  | m1.small  | 2048      | 10   | 20        |      | 1     | 1.0         |
| 3  | m1.medium | 4096      | 10   | 40        |      | 2     | 1.0         |
| 4  | m1.large  | 8192      | 10   | 80        |      | 4     | 1.0         |
| 5  | m1.xlarge | 16384     | 10   | 160       |      | 8     | 1.0         |
+----+-----------+-----------+------+-----------+------+-------+-------------+
```

Get the ID of the image you would like to use for the instance using the **`nova image-list`** command. Create the instance using **`nova boot`**. If there is not enough RAM available to start an instance, Nova will not do so. Create an instance using flavor **`1`** or **`2`**. Once the instance has been created, it will show up in the output of **`nova list`**. You can also retrieve additional details about the specific instance using the **`nova show`** command.

```
$ nova image-list
+--------------------------------------+----------+--------+--------+
|                  ID                  |   Name   | Status | Server |
+--------------------------------------+----------+--------+--------+
| 17a34b8e-c573-48d6-920c-b4b450172b41 | RHEL 6.2 | ACTIVE |        |
+--------------------------------------+----------+--------+--------+
$ nova boot --flavor 2 --key_name oskey --image 17a34b8e-c573-48d6-920c-
b4b450172b41 rhel
+------------------------+--------------------------------------+
|        Property        |                Value                 |
+------------------------+--------------------------------------+
| OS-DCF:diskConfig      | MANUAL                               |
| OS-EXT-STS:power_state | 0                                    |
| OS-EXT-STS:task_state  | scheduling                           |
| OS-EXT-STS:vm_state    | building                             |
| accessIPv4             |                                      |
| accessIPv6             |                                      |
| adminPass              | QVAmyS5i5etE                         |
| config_drive           |                                      |
| created                | 2012-05-18T13:41:40Z                 |
| flavor                 | m1.small                             |
| hostId                 |                                      |
| id                     | 0e4011a4-3128-4674-ab16-dd1b7ecc126e |
| image                  | RHEL 6.2                             |
| key_name               | oskey                                |
| metadata               | {}                                   |
| name                   | rhel                                 |
| progress               | 0                                    |
| status                 | BUILD                                |
| tenant_id              | 05816b0106994f95a83b913d4ff995eb     |
| updated                | 2012-05-18T13:41:40Z                 |
| user_id                | 1d59c0bfef9b4ea9ab63e2a058e68ae0     |
+------------------------+--------------------------------------+
$ nova list
+--------------------------------------+------+--------+------------------+
|                  ID                  | Name | Status |     Networks     |
+--------------------------------------+------+--------+------------------+
| 0e4011a4-3128-4674-ab16-dd1b7ecc126e | rhel | BUILD  | demonet=10.0.0.2 |
+--------------------------------------+------+--------+------------------+
$ nova list
+--------------------------------------+------+--------+------------------+
|                  ID                  | Name | Status |     Networks     |
+--------------------------------------+------+--------+------------------+
| 0e4011a4-3128-4674-ab16-dd1b7ecc126e | rhel | ACTIVE | demonet=10.0.0.2 |
+--------------------------------------+------+--------+------------------+
$ nova show 0e4011a4-3128-4674-ab16-dd1b7ecc126e
```

Once enough time has passed so that the instance is fully booted and initialized, you can ssh into the instance. You can obtain the IP address of the instance from the output of **nova list**.

```
$ ssh -i oskey.priv root@10.0.0.2
```

7.2. Creating and Attaching a Volume

You must first create a volume before you can attach it to an instance. When a new volume is created, it will be created as a logical volume in the **nova-volumes** volume group created in Chapter 4, *Nova (Compute)*. Run the following commands to create a 1 GB volume and then attach it to an instance. Use the **nova list** command to get the ID of the instance you wish to attach the volume to.

```
$ nova volume-create --display_name "test volume" 1
$ nova volume-list
+----+-----------+--------------+------+-------------+-------------+
| ID |   Status  | Display Name | Size | Volume Type | Attached to |
+----+-----------+--------------+------+-------------+-------------+
| 1  | available | test volume  | 1    | None        |             |
+----+-----------+--------------+------+-------------+-------------+
$ nova volume-attach <instanceid> 1 /dev/vdc
```

You should now see the new device (**/dev/vdc** in this case) available from inside the virtual machine:

```
vm$ cat /proc/partitions
major minor  #blocks  name

 252        0   10485760 vda
 252        1        960 vda1
 252        2    9765625 vda2
 252       16   20971520 vdb
 252       32    1048576 vdc
```

Create a filesystem on the device and mount it in the virtual machine:

```
vm$ mkfs.ext4 /dev/vdc
vm$ mkdir -p /mnt/volume
vm$ mount /dev/vdc /mnt/volume
```

Write some data to the mounted volume:

```
vm$ echo "Red Hat Summit" > /mnt/volume/test.txt
```

Unmount the volume inside the virtual machine. From the host running Nova, detach the volume from the instance. The **volume-detach** command requires an instance ID and the volume ID you would like to detach from that instance:

```
vm$ umount /mnt/volume
```

```
$ nova volume-detach <instanceid> 1
```

To verify that the data written to the volume has persisted, you can start up a new instance. Once the new instance is in the **ACTIVE** state, attach the volume to that instance, and then mount the volume in the instance:

```
$ nova boot --image <imageid> --flavor 2 --key_name oskey rhel2
+-------------------------+--------------------------------------+
|        Property         |                Value                 |
+-------------------------+--------------------------------------+
| OS-DCF:diskConfig       | MANUAL                               |
| OS-EXT-STS:power_state  | 0                                    |
| OS-EXT-STS:task_state   | scheduling                           |
| OS-EXT-STS:vm_state     | building                             |
| accessIPv4              |                                      |
| accessIPv6              |                                      |
| adminPass               | uPnzQhpdZZf9                         |
| config_drive            |                                      |
| created                 | 2012-05-18T13:45:56Z                 |
| flavor                  | m1.small                             |
| hostId                  |                                      |
| id                      | b8d5c952-f2fc-4556-83f2-57c79378d867 |
| image                   | RHEL 6.2                             |
| key_name                | oskey                                |
| metadata                | {}                                   |
| name                    | rhel2                                |
| progress                | 0                                    |
| status                  | BUILD                                |
| tenant_id               | 05816b0106994f95a83b913d4ff995eb     |
| updated                 | 2012-05-18T13:45:56Z                 |
| user_id                 | 1d59c0bfef9b4ea9ab63e2a058e68ae0     |
+-------------------------+--------------------------------------+
$ nova list
+--------------------------------------+---------+--------+------------------+
|                  ID                  |   Name  | Status |     Networks     |
+--------------------------------------+---------+--------+------------------+
| 0e4011a4-3128-4674-ab16-dd1b7ecc126e | rhel    | ACTIVE | demonet=10.0.0.2 |
| b8d5c952-f2fc-4556-83f2-57c79378d867 | rhel2   | BUILD  | demonet=10.0.0.3 |
+--------------------------------------+---------+--------+------------------+
$ nova list
+--------------------------------------+---------+--------+------------------+
|                  ID                  |   Name  | Status |     Networks     |
+--------------------------------------+---------+--------+------------------+
| 0e4011a4-3128-4674-ab16-dd1b7ecc126e | rhel    | ACTIVE | demonet=10.0.0.2 |
| b8d5c952-f2fc-4556-83f2-57c79378d867 | rhel2   | ACTIVE | demonet=10.0.0.3 |
+--------------------------------------+---------+--------+------------------+
$ nova volume-attach b8d5c952-f2fc-4556-83f2-57c79378d867 1 /dev/vdc
$ ssh -i oskey.priv root@10.0.0.3
```

```
vm2$ mkdir -p /mnt/volume
vm2$ mount /dev/vdc /mnt/volume
vm2$ cat /mnt/volume/test.txt
Red Hat Summit
vm2$ umount /mnt/volume
```

```
$ nova volume-detach b8d5c952-f2fc-4556-83f2-57c79378d867 1
```

7.3. Creating a Snapshot

It it possible to create a snapshot of a running instance. This may be done for backup purposes or for creating a base image to create other instances from after applying some customizations to it. As an example, you may want every instance to have a user called `projectuser`. Create that user in the virtual machine and then create a snapshot. That snapshot can be used as the base for new instances.

Start by applying some sort of customization to the virtual machine. These commands could be used to create a user and set its password:

```
vm$ useradd projectuser
vm$ passwd projectuser
```

Now create a snapshot of that running instance:

```
$ nova image-create <instanceid> "snapshot 1"
```

The snapshot is complete when its status in **`nova image-list`** changes from **`SAVING`** to **`ACTIVE`**.

```
$ nova image-list
+--------------------------------------+------------+--------+--------+
|                  ID                  |    Name    | Status | Server |
+--------------------------------------+------------+--------+--------+
| 17a34b8e-c573-48d6-920c-b4b450172b41 | RHEL 6.2   | ACTIVE |        |
| 84420f08-1e4b-499a-837a-5c6c1b9903d0 | snapshot 1 | SAVING | ...... |
+--------------------------------------+------------+--------+--------+
$ nova image-list
+--------------------------------------+------------+--------+--------+
|                  ID                  |    Name    | Status | Server |
+--------------------------------------+------------+--------+--------+
| 17a34b8e-c573-48d6-920c-b4b450172b41 | RHEL 6.2   | ACTIVE |        |
| 84420f08-1e4b-499a-837a-5c6c1b9903d0 | snapshot 1 | ACTIVE | ...... |
+--------------------------------------+------------+--------+--------+
```

Once the snapshot's status is active, you can start up a new instance using this image:

```
$ nova boot --image 84420f08-1e4b-499a-837a-5c6c1b9903d0 --flavor 2 --key_name
oskey \
  snapshot-instance
+------------------------+--------------------------------------+
|        Property        |                Value                 |
+------------------------+--------------------------------------+
| OS-DCF:diskConfig      | MANUAL                               |
| OS-EXT-STS:power_state | 0                                    |
| OS-EXT-STS:task_state  | scheduling                           |
| OS-EXT-STS:vm_state    | building                             |
| accessIPv4             |                                      |
| accessIPv6             |                                      |
| adminPass              | QASX3r8jKzVd                         |
| config_drive           |                                      |
| created                | 2012-05-18T13:49:07Z                 |
| flavor                 | m1.small                             |
| hostId                 |                                      |
| id                     | ac9e6a9f-58c3-47c3-9b4c-485aa421b8a8 |
| image                  | snapshot 1                           |
| key_name               | oskey                                |
| metadata               | {}                                   |
| name                   | snapshot-instance                    |
| progress               | 0                                    |
| status                 | BUILD                                |
| tenant_id              | 05816b0106994f95a83b913d4ff995eb     |
| updated                | 2012-05-18T13:49:08Z                 |
| user_id                | 1d59c0bfef9b4ea9ab63e2a058e68ae0     |
+------------------------+--------------------------------------+
$ nova list
+--------------------------------------+-------------------+--------+-----------
-------+
|                  ID                  |        Name       | Status |
Networks     |
+--------------------------------------+-------------------+--------+-----------
-------+
| 0e4011a4-3128-4674-ab16-dd1b7ecc126e | rhel              | ACTIVE |
demonet=10.0.0.2 |
| ac9e6a9f-58c3-47c3-9b4c-485aa421b8a8 | snapshot-instance | BUILD  |
demonet=10.0.0.4 |
| b8d5c952-f2fc-4556-83f2-57c79378d867 | rhel2             | ACTIVE |
demonet=10.0.0.3 |
+--------------------------------------+-------------------+--------+-----------
-------+
```

Finally, test that the new instance contains the expected customizations made earlier in this exercise. If you followed the example, the instance should have a user named **`projectuser`**.

```
$ ssh -i oskey.priv root@10.0.0.4
```

```
vm$ su projectuser
```

7.4. Creating Security Groups

Security groups are used to specify what IP traffic is allowed to reach an instance on its public IP address. In this exercise, you will add a security group rule that allows port 22 for ssh. This will be relevant in Section 7.5, “Adding Floating IP Addresses”, when you go through assigning floating IP addresses to running instances.

```
$ nova secgroup-list
+---------+-------------+
|   Name  | Description |
+---------+-------------+
| default | default     |
+---------+-------------+
$ nova secgroup-list-rules default
+-------------+-----------+---------+----------+--------------+
| IP Protocol | From Port | To Port | IP Range | Source Group |
+-------------+-----------+---------+----------+--------------+
+-------------+-----------+---------+----------+--------------+
$ nova secgroup-add-rule default tcp 22 22 172.31.0.224/28
+-------------+-----------+---------+-----------------+--------------+
| IP Protocol | From Port | To Port |     IP Range    | Source Group |
+-------------+-----------+---------+-----------------+--------------+
| tcp         | 22        | 22      | 172.31.0.224/28 |              |
+-------------+-----------+---------+-----------------+--------------+
```

7.5. Adding Floating IP Addresses

Floating IP addresses are typically used for assigning public IP addresses to instances. First you create a pool of addresses using **nova-manage**. Then you can use the **nova** command to reserve an address and apply it to an instance.

```
$ sudo nova-manage floating create 172.31.0.224/28
$ nova floating-ip-create
+--------------+-------------+----------+------+
|      Ip      | Instance Id | Fixed Ip | Pool |
+--------------+-------------+----------+------+
| 172.31.0.225 | None        | None     | nova |
+--------------+-------------+----------+------+
$ nova floating-ip-list
+--------------+-------------+----------+------+
|      Ip      | Instance Id | Fixed Ip | Pool |
+--------------+-------------+----------+------+
| 172.31.0.225 | None        | None     | nova |
+--------------+-------------+----------+------+
$ nova list
+--------------------------------------+------------------+--------+-----------
-------+
|                  ID                  |       Name       | Status |
Networks     |
+--------------------------------------+------------------+--------+-----------
-------+
| 0e4011a4-3128-4674-ab16-dd1b7ecc126e | rel              | ACTIVE |
demonet=10.0.0.2 |
| ac9e6a9f-58c3-47c3-9b4c-485aa421b8a8 | snapshot-instance | ACTIVE |
demonet=10.0.0.4 |
| b8d5c952-f2fc-4556-83f2-57c79378d867 | rhel2            | ACTIVE |
demonet=10.0.0.3 |
+--------------------------------------+------------------+--------+-----------
-------+
$ nova add-floating-ip 0e4011a4-3128-4674-ab16-dd1b7ecc126e 172.31.0.225
```

The previous step may take some time to apply. Watch the output of **nova list** for the floating IP address to show up in the **Networks** column of the output. After a short time you should be able to ssh to the instance using its floating IP address:

```
$ ssh -i oskey.priv root@172.31.0.225
```

Once you no longer need the floating IP address for an instance, you can remove it. Once the IP address has been removed, it should no longer show up in the output of **nova list**.

```
$ nova remove-floating-ip 0e4011a4-3128-4674-ab16-dd1b7ecc126e 172.31.0.225
```

7.6. Controlling Instance State (Suspend, Resume, Reboot, Terminate)

Up to this point you have only booted up instances. There are some other commands that can be used to adjust instance state. You can suspend, resume, reboot, and terminate an instance. The following commands show some examples of doing a suspend, resume, and reboot. Terminating instances is covered in Section 7.7, “Deleting Instances”.

```
$ nova list
+--------------------------------------+-------------------+--------+-----------
-------+
|                  ID                  |        Name       | Status |
Networks     |
+--------------------------------------+-------------------+--------+-----------
-------+
| 0e4011a4-3128-4674-ab16-dd1b7ecc126e | rhel              | ACTIVE |
demonet=10.0.0.2 |
| ac9e6a9f-58c3-47c3-9b4c-485aa421b8a8 | snapshot-instance | ACTIVE |
demonet=10.0.0.4 |
| b8d5c952-f2fc-4556-83f2-57c79378d867 | rhel2             | ACTIVE |
demonet=10.0.0.3 |
+--------------------------------------+-------------------+--------+-----------
-------+
$ nova suspend ac9e6a9f-58c3-47c3-9b4c-485aa421b8a8
$ ping 10.0.0.4   # should not get a response
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
Ctrl+c
--- 10.0.0.4 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2879ms
$ nova resume ac9e6a9f-58c3-47c3-9b4c-485aa421b8a8
$ ping 10.0.0.4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=1685 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=685 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.451 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.394 ms
Ctrl+c
--- 10.0.0.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3607ms
$ nova reboot ac9e6a9f-58c3-47c3-9b4c-485aa421b8a8
$ ssh -i oskey.priv root@10.0.0.4
Last login: Fri May 18 09:50:38 2012 from 10.0.0.1
vm$ uptime
 09:59:09 up 0 min,  1 user,  load average: 0.15, 0.03, 0.01
```

7.7. Deleting Instances

A running instance can be deleted using **nova delete**. The following example shows how to delete all running instances:

```
$ nova list
+--------------------------------------+-------------------+--------+-----------
-------+
|                  ID                  |        Name       | Status |
Networks     |
+--------------------------------------+-------------------+--------+-----------
-------+
| 0e4011a4-3128-4674-ab16-dd1b7ecc126e | rhel              | ACTIVE |
demonet=10.0.0.2 |
| ac9e6a9f-58c3-47c3-9b4c-485aa421b8a8 | snapshot-instance | ACTIVE |
demonet=10.0.0.4 |
| b8d5c952-f2fc-4556-83f2-57c79378d867 | rhel2             | ACTIVE |
demonet=10.0.0.3 |
+--------------------------------------+-------------------+--------+-----------
-------+
$ nova delete 0e4011a4-3128-4674-ab16-dd1b7ecc126e
$ nova delete ac9e6a9f-58c3-47c3-9b4c-485aa421b8a8
$ nova delete b8d5c952-f2fc-4556-83f2-57c79378d867
$ nova list
+----+------+--------+----------+
| ID | Name | Status | Networks |
+----+------+--------+----------+
+----+------+--------+----------+
```

Revision History

Revision 3.1-8	**Tue Jan 22 2013**	**Stephen Gordon**

Updated web_version_label.

Revision 3.1-6	**Wed Nov 7 2012**	**Bruce Reeler**

Minor edits to Swift commands and merged sections as per BZ87095.

Revision 3.1-5	**Fri Nov 2 2012**	**Bruce Reeler**

Added Swift installation:BZ870905.
Minor edits through entire document.

Revision 3.1-4	**Thu Oct 4 2012**	**Bruce Reeler**

Removed unnecessary punctuation in copyright:BZ860883.
Corrected %-symbols to $ in instructions for registering Nova volumes API with Keystone:BZ861265.

Revision 3.1-3	**Tue Oct 2 2012**	**Bruce Reeler**

Updated login screen capture:BZ856022.
Added novncpproxy installation step:BZ856671.

Revision 3.1-2	**Wed Sep 26 2012**	**Bruce Reeler**

Fixed terminology:BZ851716.

Revision 3.1-1	**Tue Sep 25 2012**	**Bruce Reeler**

Fixed typo:BZ856655.

Revision 3.1-0	**Fri Sep 7 2012**	**Michael Hideo**

Edits applied from engineering.

Revision 3.0-0	**Mon Aug 13 2012**	**Michael Hideo**

Edits applied from engineering.

Revision 2.0-0	**Fri Aug 10 2012**	**Michael Hideo**

Second version of the guide. Built for stage.

Revision 1.0-0	**Thu Jun 28 2012**	**Russell Bryant**

First version of the guide