**Red Hat OpenStack Administration**
**Student Workbook**
Red Hat OpenStack 4.0
Release en-1-20140207

# RED HAT OPENSTACK ADMINISTRATION

# Red Hat OpenStack 4.0 CL210
# Red Hat OpenStack Administration
# Edition 1

Author Forrest Taylor
Author Rudolf Kastl

Copyright © 2013 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are Copyright © 2013 Red Hat, Inc.

No part of this publication may be stored in a retrieval system, transmitted or reproduced in any way, including, but not limited to, photocopy, photograph, magnetic, electronic or other record, without the prior written permission of Red Hat, Inc.

This instructional program, including all material provided herein, is supplied without any guarantees from Red Hat, Inc. Red Hat, Inc. assumes no liability for damages or legal action arising from the use or misuse of contents or details contained herein.

If you believe Red Hat training materials are being used, copied, or otherwise improperly distributed please e-mail training@redhat.com or phone toll-free (USA) +1 (866) 626-2994 or +1 (919) 754-3700.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, Hibernate, Fedora, the Infinity Logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a registered trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

The OpenStack® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

# Document Conventions

## Notes and Warnings

### 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.

### Comparison

"Comparisons" look at similarities and differences between the technology or topic being discussed and similar technologies or topics in other operating systems or environments.


### References

"References" describe where to find external documentation relevant to a subject.

### 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.

# Introduction

## Welcome to class!

Thank you for attending this Red Hat training class. Please let us know if you have any special needs while at our training facility.

Please ask the instructor if you have any questions about the facility, such as operating hours of the facility and when you will have access to the classroom, locations of restrooms and break rooms, availability of telephones and network connectivity, and information about the local area.

As a courtesy to other students, please place your pager or cell phone's ringer on vibrate or mute, or turn off your devices during class. We ask that you only make calls during break periods.

If you have a personal emergency and are unable to attend or complete the class, please let us know. Thank you!

## About Red Hat Enterprise Linux

This course is taught using Red Hat Enterprise Linux, an enterprise-targeted Linux distribution focused on mature open source software designed specifically for organizations using Linux in production settings.

Red Hat Enterprise Linux is sold on a subscription basis, where the subscription gives you continues access to all supported versions of the operating system in binary and source form, not just the latest one, including all updates and bug fixes. Extensive support services are included which vary depending on the subscription level selected; see `http://www.redhat.com/subscriptions/` for details. Various Service Level Agreements are available that may provide up to 24x7 coverage with a guaranteed one hour response time for Severity 1 issues. Support will be available for up to seven years after a particular major release (ten years with the optional "Extended Update Support" Add-On).

Red Hat Enterprise Linux is released on a multi-year cycle between major releases. Minor updates to major releases are released periodically during the lifecycle of the product. Systems certified on one minor update of a major release continue to be certified for future minor updates of the major release. A core set of shared libraries have APIs and ABIs which will be preserved between major releases. Many other shared libraries are provided, which have APIs and ABIs which are guaranteed within a major release (for all minor updates) but which are not guaranteed to be stable across major releases.

Red Hat Enterprise Linux is based on code developed by the open source community, which is often first packaged through the Red Hat sponsored, freely-available Fedora distribution (`http://fedoraproject.org/`). Red Hat then adds performance enhancements, intensive testing, and certification on products produced by top independent software and hardware vendors. Red Hat Enterprise Linux provides a high degree of standardization through its support for four processor architectures (32-bit Intel x86-compatible, AMD64/Intel 64 (x86-64), IBM POWER, and IBM mainframe on System z). Furthermore, we support the 4000+ ISV certifications on Red Hat Enterprise Linux whether the RHEL operating system those applications are using

is running on "bare metal", in a virtual machine, as a software appliance, or in the cloud using technologies such as Amazon EC2.

Currently, the Red Hat Enterprise Linux product family includes:

- *Red Hat Enterprise Linux for Servers*: the datacenter platform for mission-critical servers running Red Hat Enterprise Linux. This product includes support for the largest x86-64 and x86-compatible servers and the highest levels of technical support, deployable on bare metal, as a guest on the major hypervisors, or in the cloud. Subscriptions are available with flexible guest entitlements of one, four, or unlimited guests per physical host. Pricing is based on the basis of the number of socket-pairs populated on the system motherboard, the number of guests supported, the level of support desired, and the length of subscription desired.

  *Red Hat Enterprise Linux for IBM POWER* and *Red Hat Enterprise Linux for IBM System z* are similar variants intended for those system architectures.

- *Red Hat Enterprise Linux Desktop*: built for the administrator and end-user, Red Hat Enterprise Linux Desktop provides an attractive and highly productive environment for knowledge workers on desktops and laptops. Client installations can be finely tailored and locked down for simplicity and security for any workstation task.

  The basic *Desktop* variant is designed for task workers who have a limited amount of administrative control over the system, who primarily use productivity applications like Firefox Evolution/Thunderbird, OpenOffice.org, and Planner/TaskJuggler. The more sophisticated *Workstation* variant is designed for advanced Linux users who need a stand-alone development environment, and who are expected to have local super-user privileges or selected super-user privileges.

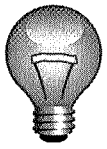
For more information please visit `http://www.redhat.com/`.

# Additional Red Hat Enterprise Linux Software

Two additional software update channels are provided with Red Hat Enterprise Linux beyond the core software packages shipped:

- *Supplementary*: the "Supplementary" channel provides selected closed source packages, built for Red Hat Enterprise Linux as a convenience to the customer. These include things like Adobe Flash or proprietary Java JVMs.

- *Optional*: the "Optional" channel provides selected open source packages, as a convenience only. They are generally included in another Red Hat Enterprise Linux variant as a fully-supported package, or are a build requirement for the distribution. These packages are only available through a Red Hat Network child channel.



## Important

*Supplementary* and *Optional* packages are provided with limited support, as a customer convenience only.

Red Hat also offers a portfolio of fully-supported *Add-Ons for Red Hat Enterprise Linux* which extend the features of your Red Hat Enterprise Linux subscription. These add-ons allow you to

add capabilities and tailor your computing environment to your particular needs. These Add-Ons include support for high availability application clustering, cluster file systems and very large file systems, enhanced system management with Red Hat Network, extended update support, and more.

### Note

Please visit `http://www.redhat.com/rhel/add-ons/` for more information about available *Add-Ons for Red Hat Enterprise Linux*.

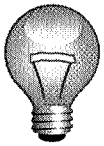
For information about other products which are provided by Red Hat, such as Red Hat Enterprise Virtualization, JBoss Enterprise Middleware, Red Hat Enterprise MRG, and various custom consulting and engineering services, `http://www.redhat.com/products/` also has useful information.

The Fedora Project also provides additional packages for Red Hat Enterprise Linux through *EPEL (Extra Packages for Enterprise Linux)*. EPEL is a volunteer-based community effort to create a repository of high-quality add-on packages which can be used with Red Hat Enterprise Linux and compatible derivatives. It accepts legally-unencumbered free and open source software which does not conflict with software in Red Hat Enterprise Linux or Red Hat add-on products. EPEL packages are built for a particular major release of Red Hat Enterprise Linux and will be updated by EPEL for the standard support lifetime of that major release.

Red Hat does not provide commercial support or service level agreements for EPEL packages. While not supported by Red Hat, EPEL provides a useful way to reduce support costs for unsupported packages which your enterprise wishes to use with Red Hat Enterprise Linux. EPEL allows you to distribute support work you would need to do by yourself across other organizations which share your desire to use this open source software with RHEL. The software packages themselves go through the same review process as Fedora packages, meaning that experienced Linux developers have examined the packages for issues. As EPEL does not replace or conflict with software packages shipped in RHEL, you can use EPEL with confidence that it will not cause problems with your normal software packages.

For developers who wish to see their open source software become part of Red Hat Enterprise Linux, often a first stage is to sponsor it in EPEL so that RHEL users have the opportunity to use it, and so experience is gained with managing the package for a Red Hat distribution.

Visit `http://fedoraproject.org/wiki/EPEL` for more information about EPEL.



### Important

*EPEL* is supported by the community-managed Fedora Project and not by Red Hat Support.

# Contacting Red Hat Technical Support

One of the benefits of your subscription to Red Hat Enterprise Linux is access to technical support through Red Hat's customer portal at `http://access.redhat.com/`. If you do not have a Red Hat account on the customer portal or are not able to log in, you can go to `https://`

access.redhat.com/support/faq/LoginAssistance.html or contact Customer Service for assistance.

You may be able to resolve your problem without formal technical support by searching Knowledgebase (https://access.redhat.com/kb/knowledgebase/). Otherwise, depending on your support level, Red Hat Support may be contacted through a web form or by phone. Phone numbers and business hours for different regions vary; see https://access.redhat.com/support/contact/technicalSupport.html for current information. Information about the support process is available at https://access.redhat.com/support/policy/support_process.html.

Some tips on preparing your bug report to most effectively engage Red Hat Support:

- *Define the problem.* Make certain that you can articulate the problem and its symptoms before you contact Red Hat. Be as specific as possible, and detail the steps you can use (if any) to reproduce the problem.
- *Gather background information.* What version of our software are you running? Are you using the latest update? What steps led to the failure? Can the problem be recreated and what steps are required? Have any recent changes been made that could have triggered the issue? Were messages or other diagnostic messages issued? What *exactly* were they (exact wording may be critical)?
- *Gather relevant diagnostic information.* Be ready to provide as much relevant information as possible; logs, core dumps, traces, the output of **sosreport**, etc. Technical Support can assist you in determining what is relevant.
- *Determine the Severity Level of your issue.* Red Hat uses a four-level scale to indicate the criticality of issues; criteria may be found at https://access.redhat.com/support/policy/GSS_severity.html.

## Warning

*Bugzilla is not a support tool!* For support issues affecting Red Hat Enterprise Linux, customers should file their bugs through the support channels discussed above in order to ensure that Red Hat is fully aware of your issue and can respond under the terms of your Service Level Agreement. Customers should *not* file bugs directly in the http://bugzilla.redhat.com/ web interface.

For Red Hat Enterprise Linux, Bugzilla is used by engineering to track issues and changes, and to communicate on a technical level with Engineering partners and other external parties. Anyone, even non-customers, can file issues against Bugzilla, and Red Hat does monitor them and review them for inclusion in errata.

However, Red Hat does not guarantee any SLA for bugs filed directly in Bugzilla (bypassing normal support channels). A review might happen immediately, or after a time span of any length. Issues coming through Support are always prioritized above issues of similar impact and severity filed against Bugzilla. Also, work arounds and hotfixes if possible and appropriate may be provided to customers by Support even before a permanent fix is issued through Red Hat Network.

Red Hat considers issues directly entered into Bugzilla important feedback, and it allows us to provide efficient interaction with the open source development community and as much

transparency as possible to customers as issues are processed. Nevertheless, for customers encountering production issues in Red Hat Enterprise Linux, Bugzilla is not the right channel.

# About this course

## Red Hat OpenStack Administration

The Red Hat OpenStack Administration course begins by explaining the OpenStack architecture and terms used throughout the course. The course shows how to install and configure OpenStack, including the message broker (Qpid), the identity service (Keystone), the object storage service (Swift), the image service (Glance), the block storage service (Cinder), the networking service (Neutron), the compute and controller services (Nova), the orchestration service (Heat) and the metering service (Ceilometer). The course finishes with a comprehensive review, implementing the services after a fresh installation of the operating system.

## Objectives

- Discuss the Red Hat OpenStack architecture
- Install Red Hat OpenStack
- Implement and secure the Qpid message broker
- Manage users, tenants and projects
- Implement the Swift object storage service
- Implement the Glance image service
- Implement the Cinder block storage service
- Implement the OpenStack networking service
- Implement the Nova compute and Nova controller services
- Implement an additional Nova compute node
- Deploy virtual machines
- Implement the Heat orchestration service
- Implement the Ceilometer metering service
- Discuss the future of Red Hat OpenStack

## Audience and prerequisites

- Linux system administrators and cloud administrators interested in, or responsible for, maintaining a private cloud. RHCSA certification or equivalent level of knowledge is highly recommended.

# Structure of the course

Red Hat training courses are interactive, hands-on, performance-based, real world classes meant to engage the mind and give students an opportunity to use real systems to develop real skills. We encourage students to participate in class and ask questions in order to get the most out of their training sessions.

This course is divided up into a number of *chapters* organized around a particular topic area. Each chapter is divided up into multiple *sections* which focus on a specific skill or task. The chapter will start with an introduction to the material, then move on to the first section.

In each section, there will be a *presentation* led by the instructor. During the presentation, it may be a good idea to take notes in the student workbook (this book), and the instructor may remind you to do so. The presentation is followed by a short activity or *assessment* to give students the opportunity to practice with the material or review procedures. After a review of the assessment, the instructor will move on to the next section. At the end of the chapter, there will normally be a hands-on lab exercise of some sort (a "*chapter test*") which will give you an opportunity to learn by doing and review your understanding of the chapter's content. Please feel free to ask questions in class, or ask the instructor for advice and help during the end-of-chapter exercise. We want the classroom environment to be a "low-risk" place where students feel comfortable asking questions and learning from things that work and things that do not at first.

# Orientation to the classroom network

Three subnets will be used in this course. The primary classroom network is 192.168.0.0/24, and belongs to hosts in the DNS domain "example.com". This network will be used for most classroom activities. The second classroom network is 192.168.32.0/24, which is used for the virtual machine non-routable IP addresses. desktop*X*.example.com has a 192.168.32.250 IP address to communicate with this network. The third classroom network is 172.24.*X*.0/24, which is used for the virtual machine floating IP addresses. The server*X*.example.com machine has a 172.24.*X*.250 IP address to route this network to 172.24.*X*.254 on instructor.example.com.

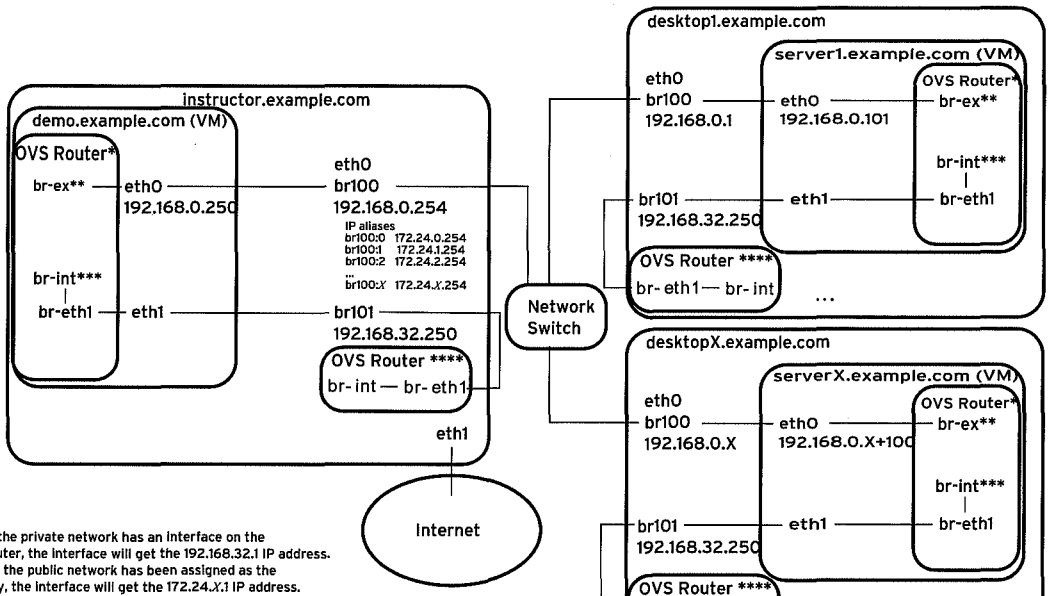
Students are each assigned a physical machine (desktop*X*.example.com on 192.168.0.*X*) which hosts a virtual machine for lab activities, server*X*.example.com on 192.168.0.*X+100*.

The instructor controls a number of machines which students may see as well. The machine instructor.example.com is the classroom utility server, providing default routing services, DHCP, DNS name service, one or more YUM repositories of software used by the class, and other network services. It is also connected to the classroom video projector to allow the instructor to display slides and demonstrations. It provides a virtual machine for the instructor, demo.example.com on 192.168.0.250, which the instructor will use for in-class demonstrations. instructor.example.com also has 172.24.*X*.254 IP addresses to provide routing for the 172.24.*X*.0/24 networks.

**Classroom machines**

| Machine name | IP addresses | Role |
|---|---|---|
| desktop*X*.example.com | 192.168.0.*X*, 192.168.32.250 | Physical student workstation |
| server*X*.example.com | 192.168.0.*X+100* | Main student virtual machine |
| instructor.example.com | 192.168.0.254, 172.24.*X*.254 | Physical instructor machine and utility server |

| Machine name | IP addresses | Role |
|---|---|---|
| demo.example.com | 192.168.0.250 | Instructor virtual demonstration machine |




* Once the private network has an interface on the OVS router, the interface will get the 192.168.32.1 IP address.
** Once the public network has been assigned as the gateway, the interface will get the 172.24.*X*.1 IP address.
*** The br-int bridge is the integration bridge for the instances, and acts as a patch panel.
**** As soon as nova-compute is moved to our physical machine, we have an OVS router there.

# Internationalization

## Language Support

Red Hat Enterprise Linux 6 officially supports twenty-two languages: English, Assamese, Bengali, Chinese (Simplified), Chinese (Traditional), French, German, Gujarati, Hindi, Italian, Japanese, Kannada, Korean, Malayalam, Marathi, Oriya, Portuguese (Brazilian), Punjabi, Russian, Spanish, Tamil, and Telugu. Support for Maithili, Nepalese, and Sinhala are provided as Technology Previews.

## System-wide Default Language

The operating system's default language is normally set to US English (en_US.UTF-8), but this can be changed during or after installation.

To use other languages, you may need to install additional package groups to provide the appropriate fonts, translations, dictionaries, and so forth. By convention, these package groups are always named ***language*-support**. These package groups can be selected during installation, or after installation with PackageKit (**System** > **Administration** > **Add/Remove Software**) or **yum**.

A system's default language can be changed with **system-config-language** (**System** > **Administration** > **Language**), which affects the **/etc/sysconfig/i18n** file.

## Per-user Language Selection

Users may prefer to use a different language for their own desktop environment or interactive shells than is set as the system default. This is indicated to the system through the **LANG** environment variable.

This may be set automatically for the GNOME desktop environment by selecting a language from the graphical login screen by clicking on the **Language** item at the bottom left corner of the graphical login screen immediately prior to login. The user will be prompted about whether the language selected should be used just for this one login session or as a default for the user from now on. The setting is saved in the user's **~/.dmrc** file by GDM.

If a user wants to make their shell environment use the same **LANG** setting as their graphical environment even when they login through a text console or over **ssh**, they can set code similar to the following in their **~/.bashrc** file. This code will set their preferred language if one is saved in **~/.dmrc** or will use the system default if one is not:

```
i=$(grep 'Language=' ${HOME}/.dmrc | sed 's/Language=//')
if [ "$i" != "" ]; then
    export LANG=$i
fi
```

Languages with non-ASCII characters may have problems displaying in some environments. Kanji characters, for example, may not display as expected on a virtual console. Individual commands can be made to use another language by setting **LANG** on the command-line:

```
[user@host ~]$ LANG=fr_FR.UTF-8 date
lun. oct. 24 10:37:53 CDT 2011
```

Subsequent commands will revert to using the system's default language for output. The **locale** command can be used to check the current value of **LANG** and other related environment variables.

# Input Methods

IBus (Intelligent Input Bus) can be used to input text in various languages under X if the appropriate language support packages are installed. You can enable IBus with the **im-chooser** command (**System** > **Preferences** > **Input Method**).

# Language Codes Reference

**Language Codes**

| Language | $LANG value | Language package group |
|---|---|---|
| English (US) | en_US.UTF-8 | *(default)* |
| Assamese | as_IN.UTF-8 | assamese-support |
| Bengali | bn_IN.UTF-8 | bengali-support |
| Chinese (Simplified) | zh_CN.UTF-8 | chinese-support |
| Chinese (Traditional) | zh_TW.UTF-8 | chinese-support |
| French | fr_FR.UTF-8 | french-support |
| German | de_DE.UTF-8 | german-support |
| Gujarati | gu_IN.UTF-8 | gujarati-support |
| Hindi | hi_IN.UTF-8 | hindi-support |
| Italian | it_IT.UTF-8 | italian-support |
| Japanese | ja_JP.UTF-8 | japanese-support |
| Kannada | kn_IN.UTF-8 | kannada-support |
| Korean | ko_KR.UTF-8 | korean-support |
| Malayalam | ml_IN.UTF-8 | malayalam-support |
| Marathi | mr_IN.UTF-8 | marathi-support |
| Oriya | or_IN.UTF-8 | oriya-support |
| Portuguese (Brazilian) | pt_BR.UTF-8 | brazilian-support |
| Punjabi | pa_IN.UTF-8 | punjabi-support |

| Language | $LANG value | Language package group |
|---|---|---|
| Russian | ru_RU.UTF-8 | russian-support |
| Spanish | es_ES.UTF-8 | spanish-support |
| Tamil | ta_IN.UTF-8 | tamil-support |
| Telugu | te_IN.UTF-8 | telugu-support |
| *Technology Previews* | | |
| Maithili | mai_IN.UTF-8 | maithili-support |
| Nepali | ne_NP.UTF-8 | nepali-support |
| Sinhala | si_LK.UTF-8 | sinhala-support |

# CHAPTER 1

# INTRODUCING RED HAT OPENSTACK ARCHITECTURE

## Introduction

| Chapter details | |
|---|---|
| **Chapter goal** | Understand the services and terms used in OpenStack. |
| **Chapter sections** | • Red Hat OpenStack architecture overview |
| **Hands-on activities** | • None |
| **Chapter test** | Explore the classroom environment |

# Red Hat OpenStack architecture overview

OpenStack includes the following services:

*Nova (compute)*: a service that manages networks of virtual machines running on nodes, providing virtual machines on demand. Nova is a distributed component and interacts with Keystone for authentication, Glance for images, and Horizon for web interface. Nova is designed to scale horizontally on standard hardware, downloading images to launch instances as required. Nova compute uses libvirtd, qemu, and kvm for the hypervisor.

*Glance (image)*: a service that acts as a registry for virtual machine images, allowing users to copy server images for immediate storage. These images can be used as templates when setting up new instances.

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

*Cinder (volume)*: a service that manages storage volumes for virtual machines. This is persistent block storage for the instances running in Nova. Snapshots can be taken for backing up data, either for restoring data or to be used to create new block storage volumes.

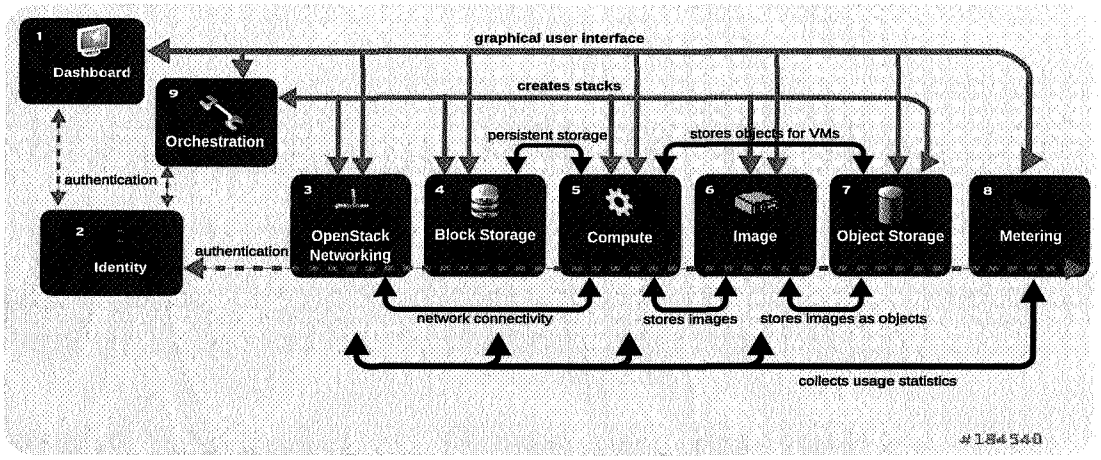
*Swift (object)*: a service providing object storage which allows users to store and retrieve files. Swift architecture is distributed to allow for horizontal scaling and to provide redundancy as failure-proofing. Data replication is managed by software, allowing greater scalability and redundancy than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

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

*Horizon (dashboard)*: a web-based interface for managing OpenStack services. It provides a graphical user interface for operations such as launching instances, managing networking, and setting access controls.

*Ceilometer (metering)*: a centralized source for metering and monitoring data. This information will provide a means to bill the users of OpenStack.

*Heat*: a service to orchestrate multiple composite cloud applications using the AWS CloudFormation template format, through both a REST API and a CloudFormation-compatible Query API. The software integrates other core components of OpenStack into a one-file template system. The templates allow creation of most OpenStack resource types (such as instances, floating IPs, volumes, security groups, users, etc.), as well as some more advanced functionality such as instance high availability, instance autoscaling, and nested stacks.




1. Horizon: web browser user interface for creating and managing instances.
2. Keystone: authentication and authorization framework.
3. OpenStack networking: network connectivity as a service.
4. Cinder: persistent block storage for runtime instances.
5. Nova: scheduler for networks of virtual machines running on nodes.
6. Glance: registry for virtual machine images.
7. Swift: file storage and retrieval.
8. Ceilometer: metering engine for collecting billable meters.
9. Heat: orchestration service for template-based virtual machine deployments.

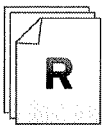
# OpenStack terminology

OpenStack uses the following terminology:

- *Cloud controller*: the coordinating manager. All machines in the OpenStack cloud communicate with the cloud controller using the Advanced Message Queuing Protocol (AMQP). Red Hat OpenStack uses the Qpid messaging daemon (**`qpidd`**) to provide AMQP.
- *Tenant*: also known as a project in Horizon. A group of items (users, images, instances, network(s), volumes, etc.).
- *Compute node*: a hypervisor; any machine running the Nova compute service. Most often, these *only* run the Nova compute service.
- *Volume*: a persistent disk presented to the instance. Volumes can be attached to a single instance. These volumes are persistent, and can be attached to (or detached from) running instances. The Cinder service is given an LVM volume group and volumes are created from this volume group (LVM logical volumes). A snapshot of the volume can be created, much as a snapshot is created of a logical volume.
- *Ephemeral disk*: a temporary disk used by an instance. When the instance is created, the ephemeral disk is created as a QCOW2 image in **`/var/lib/nova/instances/`**

**instance-00000000*X*/disk.local** on the compute node. When the instance is terminated, this disk is removed. The first ephemeral disk normally appears as **/dev/vdb**.

- *Server* or *instance*: a virtual machine.
- *Flavor*: the hardware associated with an instance. This includes RAM, CPUs, and disks.
- *Stack*: a group of instances built from a template. Template files are written in JSON. Stacks and the template files are used in the Heat orchestration service.
- *OpenStack networking*: a software-defined networking service. Includes many plug-ins (e.g., Open vSwitch, Cisco UCS/Nexus) and allows software-defined network (SDN) and quality of service (QoS). The OpenStack Networking API uses the following abstractions to describe network resources:
  - *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.




### References

Red Hat OpenStack Installation and Configuration Guide
- Section 1.2. Architecture
- Section 1.3. Service details

## Red Hat OpenStack architecture overview

- For each of the following statements, fill in the blanks:

  1. The _____ service provides virtualization using libvirtd, qemu, and kvm.
  2. The _____ service provides images that are used as templates to build instances.
  3. The _____ service provides networking capabilities using a pluggable architecture.
  4. The _____ service provides persistent volumes for instances.
  5. The _____ service provides object storage.
  6. The _____ service provides authentication and authorization.
  7. The _____ service provides a dashboard for managing OpenStack.

8. A _____ coordinates the Red Hat OpenStack cloud using the Qpid messaging service (AMQP).

9. _____ or _____ are the names used for a virtual machine in OpenStack.

# Chapter test

Performance Checklist

## Explore the classroom environment

**Lab overview:** Become oriented to the initial classroom environment.

**Success criteria:** Students will understand their system configurations.

***Before you begin...***
Login information for your Red Hat Enterprise Linux system(s):

- Username: **student** Password: **student**
- Username: **root** Password: **redhat**

**Lab outline:** The checklist defines a list of system information you need to look up or verify (host name, IP addresses, package repositories, etc.).

☐ 1. Identify the Red Hat Enterprise Linux physical machine

☐ 1.1. In the classroom, you should have one physical system, **desktop*X***, that is preinstalled with Red Hat Enterprise Linux 6.5.

In the virtual classroom, your **desktop*X*** is a virtual machine.

☐ 1.2. Log into the physical system **desktop*X*** using the username **student** with the password **student**. Open a terminal window with a shell prompt.

☐ 1.3. At the prompt of the **desktop*X*** system, run the **hostname** command to see what your machine's host name is. Write it down.

*Hostname:*_____

☐ 1.4. At the prompt of the **desktop*X*** system, run the **dig** command on your machine's host name to determine your expected IPv4 address. Write it down.

*IPv4 address:*_____

☐ 1.5. At the prompt of the **desktop*X*** system, run the **ip addr show** command to see what interface your **desktop*X*** machine's IPv4 address is attached to. Also, find the other interface that has a different private IP address. This private IP address will be used to connect to the private IP address range that the instances use. Write it down.

If you are in a virtual classroom, you will find that the other interface is simply a second UP interface with no IP assigned yet. Write down the name of that interface as the Private bridge name.

*Public bridge name:*_____

*Private bridge name:*_____

*Private IP address:*_____

☐ 2. Verify yum configuration on physical system
Your **desktop*X*** system may need to get software packages from the repositories on **instructor.example.com**. Review the yum repositories, and write down the names of the different repositories that are currently configured.

☐ 3. Apply updates
Become the **root** user on your **desktop*X*** system and update the system with the updates provided in class.

```
[student@desktopX ~]$ su -
Password: redhat
[root@desktopX ~]# yum update -y
```

☐ 4. Identify the Red Hat Enterprise Linux virtual machine

☐ 4.1. Log into your server*X* machine as **root** (with the password **redhat**).

If you are in a physical classroom, open a new terminal and become root (**su -**). Run the command **virt-viewer server*X***. This will open a window through which you can log into the server*X* virtual machine. Log into server*X* as **root** (with a password of **redhat**).

If you are working in the virtual training environment, ignore the preceding paragraph and use the classroom controls in your web browser to connect to server*X*. Log into server*X* as **root** (with the password **redhat**).

If you do not have a **server*X*** virtual machine, or have trouble accessing it, please notify your instructor.

☐ 4.2. At the prompt on your **server*X*** virtual machine, run the **hostname** command to see what your machine's host name is. Write it down.

*Hostname:*_____

☐ 4.3. At the prompt on your **server*X*** virtual machine, run the **dig** command on your machine's host name to determine your expected IPv4 address. Write it down.

*IPv4 address:*_____

☐ 4.4. At the prompt on your **server*X*** virtual machine, run the **ip addr show** command to see what interface your machine's IPv4 address is attached to. Write it down.

*Interface name:*_____

☐ 4.5. Notice that your **server*X*** virtual machine has two NICs in the output above. Write down the interface name of the second NIC.

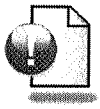
*Interface name:*_____

☐ 5. Verify yum configuration on virtual machine
Your **server*X*** system may need to get software packages from the repositories on **instructor.example.com**. Review the yum repositories, and write down the names of the different repositories that are currently configured on **server*X*.example.com**.

☐ 6. Apply updates
Update your **server*X*** system with the updates provided in class.

```
[root@serverX ~]# yum update -y
```

# Personal Notes



# Summary

Red Hat OpenStack architecture overview

- Understand OpenStack architecture.
- Understand OpenStack terminology.

# CHAPTER 2

# INSTALLING RED HAT OPENSTACK

## Introduction

| Chapter details | |
|---|---|
| **Chapter goal** | Install Red Hat OpenStack with the **packstack** utility and create an instance with the Horizon web front end. |
| **Chapter sections** | • Installing Red Hat OpenStack with packstack<br>• Using the Horizon web interface<br>• Deploying with Foreman |
| **Hands-on activities** | • Installing Red Hat OpenStack with packstack<br>• Creating a tenant in Horizon<br>• Creating a flavor in Horizon<br>• Creating a user in Horizon<br>• Launching an Instance in Horizon |
| **Chapter test** | Installing Red Hat OpenStack |

# Installing Red Hat OpenStack with packstack

Considerations to make before deploying Red Hat OpenStack:

## Hardware requirements

**Red Hat OpenStack cloud controller node hardware requirements**

| *Hardware* | *Requirement* |
|---|---|
| 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: | 2GB RAM |
| Disk space: | 50GB<br>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. 1TB of disk space is recommended for a realistic environment capable of hosting multiple instances of varying sizes. |
| Network: | 2x 1Gbps network interface card (NIC) |

**Red Hat OpenStack compute node hardware requirements**

| *Hardware* | *Requirement* |
|---|---|
| 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: | 2GB RAM minimum<br>For the compute node, 2GB RAM is the minimum necessary to deploy the m1.small instance on a node or three m1.tiny instances without memory swapping, so this is the minimum requirement for setting up a test environment. Add additional RAM to this requirement based on the amount of memory that you intend to make available to virtual machine instances. |
| Disk space: | 50GB minimum<br>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. 1TB of disk space is recommended for a realistic environment capable of hosting multiple instances of varying sizes. |
| Network: | 2x 1Gbps network interface card (NIC) |

## Software requirements

- To deploy Red Hat OpenStack, you need to have at least two machines with Red Hat Enterprise Linux 64-bit, version 6.5 or newer. One machine can act as a dedicated cloud controller node and the second machine can act as a Nova compute node. In the field, a minimum of two Nova compute nodes are recommended.

- Make sure your machines have their clocks synced via Network Time Protocol (NTP).

Workshop

# Installing Red Hat OpenStack with packstack

Follow along with the instructor as you perform the setup tasks required to install the Red Hat OpenStack software.

Red Hat OpenStack features a tool to help with the installation called **packstack**.

☐ 1. The *openstack-packstack* package includes the **packstack** utility to quickly deploy Red Hat OpenStack either interactively, or non-interactively by creating and using an answer file that can be tuned. Install the *openstack-packstack* package on **serverX**, using **yum**.

```
[root@serverX ~]# yum install -y openstack-packstack
```

☐ 2. Before we start with Red Hat OpenStack deployment via **packstack**, SSH keys are generated for easy access to the Nova compute nodes from the cloud controller node. We will not include a passphrase because the installation will require this passphrase hundreds of times during this process.

```
[root@serverX ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): Enter
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
...
```

☐ 3. Explore some of the options of the **packstack** command.

```
[root@serverX ~]# packstack -h | less
```

☐ 4. The recommended way to do an installation is non-interactive, because this way the installation settings are documented. An answer file with default settings can be generated with the **packstack** command.

```
[root@serverX ~]# packstack --gen-answer-file /root/answers.txt
```

☐ 5. Before we can start the actual installation, edit the **/root/answers.txt** file and ensure the following items are configured:

```
CONFIG_SSH_KEY=/root/.ssh/id_rsa.pub
CONFIG_NTP_SERVERS=192.168.0.254
CONFIG_HORIZON_SSL=y
```

If you are using the Red Hat Online Learning environment, do not set the NTP server. **packstack** will disable the NTP service and fail to contact the NTP server.

**Answer file settings for the cloud controller node**

| *Settings* | *Purpose* |
|---|---|
| CONFIG_SSH_KEY=/root/.ssh/id_rsa.pub | Configure the SSH pubkey to be deployed on every machine that is related to Red Hat OpenStack Cloud. |
| CONFIG_NTP_SERVERS=192.168.0.254 | Configure the NTP servers for time synchronization. |
| CONFIG_HORIZON_SSL=y | Enable use of SSL for Horizon. |

☐ 6. Now it is time to do the actual deployment of the Red Hat OpenStack Cloud controller using the answer file we just prepared:

```
[root@serverX ~]# packstack --answer-file /root/answers.txt
Welcome to Installer setup utility

Installing:
Clean Up...                                  [DONE]
Setting up ssh keys...root@192.168.0.X+100's password: redhat
...
```

☐ 7. Verify that the services are running:

```
[root@serverX ~]# for i in /etc/init.d/open* /etc/init.d/neutron* ; do $i status ;
 done
openstack-ceilometer-alarm-evaluator (pid  19318) is running...
openstack-ceilometer-alarm-notifier (pid  19244) is running...
openstack-ceilometer-api (pid  19285) is running...
openstack-ceilometer-central (pid  19207) is running...
openstack-ceilometer-collector (pid  19174) is running...
openstack-ceilometer-compute (pid  13062) is running...
openstack-cinder-api (pid  9502) is running...
openstack-cinder-backup is stopped
openstack-cinder-scheduler (pid  9665) is running...
openstack-cinder-volume (pid  9580) is running...
openstack-glance-api (pid  8625) is running...
openstack-glance-registry (pid  8591) is running...
openstack-glance-scrubber is stopped
keystone (pid  6883) is running...
openstack-nova-api (pid  10913) is running...
openstack-nova-cert (pid  13217) is running...
openstack-nova-compute (pid  13018) is running...
openstack-nova-conductor (pid  12979) is running...
openstack-nova-console is stopped
openstack-nova-consoleauth (pid  12898) is running...
openstack-nova-metadata-api is stopped
openstack-nova-novncproxy (pid  12863) is running...
openstack-nova-scheduler (pid  12940) is running...
openstack-nova-spicehtml5proxy is stopped
openstack-nova-xvpvncproxy is stopped
ovsdb-server is running with pid 14311
ovs-vswitchd is running with pid 14322
neutron-dhcp-agent (pid  14472) is running...
neutron-l3-agent (pid  14500) is running...
neutron-lbaas-agent is stopped
neutron-metadata-agent (pid  14632) is running...
neutron-openvswitch-agent (pid  14434) is running...
```

```
neutron (pid  14602) is running...
```


## References

Red Hat OpenStack Getting Started Guide

- Chapter 2. Product requirements
- Part II. Deploying OpenStack using packstack

# Using the Horizon web interface

## Logging into the Horizon web interface

The Horizon web interface allows for management of the entities for creating new projects with OpenStack. The web front end is accessible at `https://serverX.example.com/dashboard`. You can log in as **`admin`** with the password found in **`/root/keystonerc_admin`** as the **`OS_PASSWORD`** variable.

## Working with tenants

A tenant describes a project with an assigned number of OpenStack users and resources. This enables multiple projects to use a single cloud without interfering with each other in terms of permissions and resources.

A quota of resources are already set when a new tenant is created. The quota includes the amount of VCPUs, instances, RAM, and floating IPs that can be assigned to instances. Tenants can be added, modified, and deleted in Horizon with a few clicks.

**Workshop**

## Creating a tenant in Horizon

Follow along with the instructor as you create a tenant.

Create a new project with the following requirements:

**Project parameters**

| *Parameter* | *Value* |
|---|---|
| Name | project1 |
| Description | Project for project1 |
| Quota | 4 VCPUs, 4 instances, 4096 MB RAM, 2 floating IPs |

☐ 1. On desktop*X*.example.com, open **Firefox** and browse to `https://serverX.example.com/dashboard`. Add an exception for the self-signed certificate.

☐ 2. Find the admin password in the **`/root/keystonerc_admin`** file on server*X*.example.com (it will be after **`OS_PASSWORD=`**). Log in to the Horizon dashboard using **`admin`** as the username and the password found above.

☐ 3. To create the new tenant, go to the **Admin** tab in the left pane and click the **Projects** link.

☐ 4. Click the **Create Project** button.

☐ 5. Add the name and description as above. Go to the **Quota** tab and set the quotas as above.

☐ 6. Click the **Create Project** button.

# Manage flavors

When a new instance gets deployed, a flavor must be selected that outlines the virtual machine hardware specifications. The parameters include the number of VCPUs, RAM, and disk space used by the instance.

**Workshop**

## Creating a flavor in Horizon

Follow along with the instructor as you create a new flavor using the Horizon dashboard.

Create a new flavor with the following parameters:

**Flavor parameters**

| *Parameter* | *Value* |
|---|---|
| Name | m2.tiny |
| ID | auto |
| VCPUs | 1 |
| RAM MB | 1024 |
| Root Disk GB | 20 |
| Ephemeral Disk GB | 2 |
| Swap Disk MB | 512 |

☐ 1. To create the new flavor, go to the **Admin** tab in the left pane and click the **Flavors** link.

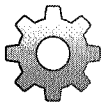
☐ 2. Click the **Create Flavor** button.

☐ 3. Enter the information as given previously.

☐ 4. Click the **Create Flavor** button.

# User management in Horizon

Users can be added, modified, and deleted with the Horizon web interface. OpenStack permission management is role-based. By default, there are two predefined roles: a member role that gets attached to a tenant, and an administrative role to enable users other than the admin to administrate the cloud.



**Workshop**

## Creating a user in Horizon

Follow along with the instructor as you create a user in the Horizon dashboard.

Create a new user with the following parameters:

**User parameters**

| *Parameter* | *Value* |
|---|---|
| Username | user1 |
| Email | root@desktop*X*.example.com |
| Password | redhat |
| Primary project | project1 |
| Role | Member |

☐ 1. Go to the **Admin** tab in the left pane and click the **Users** link.

☐ 2. Click the **Create User** button.

☐ 3. Enter the information as given previously.

☐ 4. Click the **Create User** button.

☐ 5. Log out as **admin** and log in as the newly-created **user1** user. Recall that the password is **redhat**.

# Launch an instance in Horizon

Horizon can be used to launch and manage instances. Before launching an instance, you will probably want to create and upload an image, configure a security group, create an SSH keypair, and allocate floating IP addresses. The following workshop will walk through the steps of preparing to launch an instance, and then creating an instance.

**Workshop**

## Launching an Instance in Horizon

Follow along with the instructor as you launch an instance using the Horizon dashboard.

Create an instance using the following parameters:

**Instance parameters**

| *Parameter* | *Value* |
|---|---|
| Image name | small |
| Image location | `http://instructor.example.com/pub/materials/small.img` |
| Image format | QCOW2 - QEMU Emulator |
| Image settings | No minimum disk, minimum 1024 MB RAM, public |
| Private network name | net1 |
| Private subnet information | • Subnet name: **`subnet1`**<br>• Network address: **`192.168.32.0/24`**<br>• IP version: IPv4<br>• Gateway IP: *Leave blank (not disabled)* |
| Public network name | net2 |
| Public subnet information | • Subnet name: **`subnet2`**<br>• Network address: **`172.24.X.0/24`**<br>• IP version: IPv4<br>• Gateway IP: **`172.24.X.254`** |
| Public subnet details | • Enable DHCP: deselected<br>• Allocation pools: **`172.24.X.1,172.24.X.100`** |
| Router name | router1 |
| External network | net2 |
| Router gateway | net2 |
| IP to allocate for the instance | 172.24.X.2 |
| Security group name | sec1 |
| Security group description | Web and SSH |

| *Parameter* | *Value* |
|---|---|
| Security group permissions | Allow SSH (TCP/22) and HTTPS (TCP/443) from CIDR 0.0.0.0/0, and HTTP (TCP/80) from the **sec1** source group |
| SSH keypair name | key1 |
| Instance image | small |
| Instance name | small |
| Instance flavor | m2.tiny |
| Instance keypair | key1 |
| Instance security group | sec1 |
| Instance floating IP address | 172.24.*X*.2 |
| Volume name | myvol1 |
| Volume description | myvol1 volume |
| Volume size | 2GB |
| Volume snapshot name | myvol1-snap1 |
| Volume snapshot description | myvol1-snap1 snapshot |

☐ 1. Ensure you are logged into the Horizon dashboard as **user1**.

☐ 2. The first thing to do to create a new machine is to import a new image. In the **Project** tab on the left pane, select the **Images & Snapshots** link. Click the **Create Image** button. Enter **small** for the name, enter `http://instructor.example.com/pub/materials/small.img` for the image location, and **QCOW2** for the image format. Leave the minimum disk blank, enter **1024** for the minimum RAM, and select the **Public** checkbox. Click the **Create Image** button.

## Note

Making the image "public" will allow users in other tenants to use this image.

☐ 3. Next, we configure networking. First, configure the private network (**net1**). In the **Project** tab on the left pane, select the **Networks** link. Click the **Create Network** button. Enter **net1** for the network name. In the **Subnet** tab, enter **subnet1** for the name, **192.168.32.0/24** for the network address, and **IPv4** for the IP version. Leave the other options as they are and click the **Create** button.

Next, configure the public network (**net2**). Click the **Create Network** button again. Enter the public network name as **net2**. Browse to the **Subnet** tab. Enter the public subnet name as **subnet2**, the network address as **172.24.*X*.0/24**, **IPv4** for the IP version and **172.24.*X*.254** as the gateway IP. Browse to the **Subnet Detail** tab. Deselect the **Enable DHCP** checkbox and enter **172.24.*X*.1,172.24.*X*.100** for the allocation pool. Click the **Create** button.

In the **Project** tab on the left pane, select the **Routers** link. Click the **Create Router** button. Enter the router name as **router1** and click the **Create Router** button.

☐ 4. Setting an external network can only be done as an administrator. Sign out as the **user1** user and sign in as **admin**. In the **Admin** tab in the left pane, click the **Networks** link. Click the **Edit Network** button in the public (**net2**) network row. Select the **External Network** checkbox and click the **Save Changes** button.

☐ 5. Sign out as the **admin** user and sign in as **user1**. In the **Project** tab in the left pane, click the **Routers** link. Click the **Set Gateway** button in the **router1** row. In the **External Network** menu, choose **net2**. Click the **Set Gateway** button.

☐ 6. Click the **router1** link. Click the **Add Interface** button. In the **Subnet** menu, select **net1: 192.168.32.0/24 (subnet1)**. Click the **Add Interface** button.

Verify that both networks are attached to the router. In the **Project** tab in the left pane, click the **Network Topology** link. **net1** and **net2** should both connect to the **router1** router and both network ranges should be displayed.

## Note

The interfaces attached to the router have been assigned the first address in the range (192.168.32.1 and 172.24.*X*.1). You should see the **192.168.32.1** IP address, but the **172.24.*X*.1** IP address will only be displayed for **admin**.

☐ 7. Next, allocate a floating IP address. In the **Project** tab in the left pane, click the **Access & Security** link. Choose the **Floating IPs** tab and click the **Allocate IP to Project** button. Use the *net2* pool and click the **Allocate IP** button to allocate the previously given floating IP address.

☐ 8. To determine the network access permissions, set up a security group for the instance. In the **Project** tab in the left pane, select the **Access & Security** link. Choose the **Security Groups** tab and click the **Create Security Group** button. Enter **sec1** for the name and **Web and SSH** for the description. Click the **Create Security Group** button.

Click the **Edit Rules** button for the **sec1** security group. Click the **Add Rule** button. Choose **SSH** in the **Rule** drop-down menu, and leave **Remote** and **CIDR** as default. Click the **Add** button. Click the **Add Rule** button again. Choose **HTTPS** in the **Rule** drop-down menu and click the **Add** button. Click the **Add Rule** button once more. Choose **HTTP** in the **Rule** drop-down menu. Choose **Security Group** in the **Remote** drop-down menu. Choose **sec1 (current)** as the **Security Group** and **IPv4** as the **Ether Type**. Click the **Add** button.

☐ 9. The next step is to create a SSH keypair for the instance. In the **Project** tab in the left pane, select the **Access & Security** link. Choose the **Keypairs** tab and click the **Create Keypair** button. Enter **key1** for the name and click the **Create Keypair** button. Save the **key1.pem** file to the default location, which should be in **/home/student/Downloads/key1.pem** on **desktop*X*.example.com**.

☐ 10. Bring up your terminal on **server*X*.example.com**. Before attaching **eth0** to the **br-ex** bridge, configure the **br-ex** network device configuration file.

```
[root@serverX ~]# cp /etc/sysconfig/network-scripts/ifcfg-eth0 /root/
[root@serverX ~]# cp /etc/sysconfig/network-scripts/ifcfg-eth0 /etc/sysconfig/
network-scripts/ifcfg-br-ex
```

If you are in a physical classroom, remove everything but the **DEVICE**, **HWADDR**, and **ONBOOT** settings from **/etc/sysconfig/network-scripts/ifcfg-eth0**, so that it looks like the following:

```
DEVICE=eth0
HWADDR=52:54:00:00:00:XX
ONBOOT=yes
```

Leave the MAC address as it is. Do not change it. The *XX* is the desktop*X* number in hexadecimal.

If you are in a virtual classroom, remove everything but the **DEVICE** and **ONBOOT** settings from **/etc/sysconfig/network-scripts/ifcfg-eth0**, so that it looks like the following:

```
DEVICE=eth0
ONBOOT=yes
```

In the **/etc/sysconfig/network-scripts/ifcfg-br-ex** file, remove the **HWADDR** line if present and change the device name to **br-ex**. Make sure the **/etc/sysconfig/network-scripts/ifcfg-br-ex** file contains the following:

```
DEVICE=br-ex
IPADDR=192.168.0.X+100
PREFIX=24
GATEWAY=192.168.0.254
DNS1=192.168.0.254
SEARCH1=example.com
ONBOOT=yes
```

☐ 11. Once you have verified the network files contain the correct information, add the **eth0** network device to the **br-ex** bridge and restart the network.

```
[root@serverX ~]# ovs-vsctl add-port br-ex eth0 ; service network restart
```

☐ 12. To create the instance, go back to the Horizon dashboard, and in the **Project** tab in the left pane, select the **Instances** link. Click the **Launch Instance** button. In the **Details** tab, enter **small** as the **Instance Name** and choose the **m2.tiny Flavor**. Choose **Boot from image** in the **Instance Boot Source** drop-down menu. Select **small** as the **Image Name**. In the **Access & Security** tab, enable the **key1** keypair and **sec1** security group, and deselect the **default** security group. In the **Networking** tab, click the **+** button next to the **net1** network. Click the **Launch** button.

☐ 13. Once the networking has been created for the instance, open the **More** drop-down menu (under **Actions**) and select **Associate Floating IP**. Choose the **172.24.*X*.2** floating IP address, and choose **small: 192.168.32.2** under **Port to be associated**, then click the **Associate** button.

☐ 14. In the **Project** tab in the left pane, select the **Instances** link. Right-click on the **small** instance link and choose **Open Link in New Tab** (or middle-click on the link). In the new

tab, choose the **Console** tab, then click the **Click here to show only console** link. Watch the virtual machine boot. This may take several minutes to boot.

☐ 15. For a simple networking verification of our setup, open a new terminal on desktop*X*.example.com and **ssh** to 172.24.*X*.2.

```
[student@desktopX ~]# chmod 600 /home/student/Downloads/key1.pem
[student@desktopX ~]# ssh -i /home/student/Downloads/key1.pem root@172.24.X.2
The authenticity of host '172.24.X.2 (172.24.X.2)' can't be established.
RSA key fingerprint is aa:bb:cc:dd:ee:ff:00:11:22:33:44:55:66:77:88:99.
Are you sure you want to continue connecting (yes/no)?  yes
Warning: Permanently added '172.24.X.2' (RSA) to the list of known hosts.
[root@host-192-168-32-2 ~]#
```

☐ 16. Now, create a volume. Back in the dashboard, click the **Volumes** link in the left pane and then click the **Create Volume** button. Enter **myvol1** as the volume name, **myvol1 volume** as the description, and **2** GB as the size. Click the **Create Volume** button.

☐ 17. Once the volume has been created, create a snapshot of the volume. While still in the **Volume** section in the left pane, click the **More** dropdown menu on the right side of the row that describes the new volume and select **Create Snapshot**. Enter **myvol1-snap1** as the snapshot name and **myvol1-snap1 snapshot** as the description. Click the **Create Volume Snapshot** button.

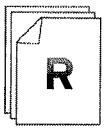
## Note

A snapshot of a volume can only be generated if it is *not* attached to a running instance.

☐ 18. Attach the volume to the running instance. Browse to the **Volumes** link in the left pane and click the **Edit Attachments** button in the **myvol1** row. Select the **small** instance we just created in the **Attach to Instance** dropdown menu. Click the **Attach Volume** button.

☐ 19. Back on the instance terminal, run **fdisk -l** to view the disks attached to the instance. You should have a 20 GB root disk (/dev/vda), a 2 GB ephemeral disk (/dev/vdb), a 512 MB swap disk (/dev/vdc), and a new 2 GB volume (/dev/vdd).

```
[root@host-192-168-32-2 ~]# fdisk -l
...
Disk /dev/vdd: 2147 MB, 2147483648 bytes
16 heads, 63 sectors/track, 4161 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
...
```




## References

Red Hat OpenStack Getting Started Guide

- Chapter 6. Using OpenStack with the dashboard

Red Hat OpenStack Installation and Configuration Guide

- Section 12.4. Launching an instance
- Chapter 17. Managing quotas

# Deploying with Foreman

## What is Foreman?

Foreman 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, Foreman is able to provide:

- Dynamic Host Configuration Protocol (DHCP)
- Domain Name System (DNS)
- Preboot Execution Environment (PXE)
- Trivial File Transfer Protocol (TFTP)

Controlling these services allows Foreman to provision even physical systems that do not yet have an operating system installed.

### Note

Foreman is supported as technical preview.

**Workshop**

## Installing Foreman

***Before you begin...***

Reset your **server*X*** virtual machine.

If you are in a physical classroom, log in as **root** on **desktop*X*.example.com** and reset your **server*X*** machine:

```
[student@desktopX ~]$ su -
Password: redhat
[root@desktopX ~]# lab-reset-vm
This will destroy the virtual machine and reset it to the last saved state.
Is this ok [y/N]: y
Waiting for things to settle...
Done.
```

If you are working in the virtual training environment, ignore the preceding paragraph and use the virtual machine controls in your web browser to reset your **server*X*** machine.

Follow along with the instructor as you perform the setup tasks required to install the Red Hat Foreman software.

☐ 1. The *openstack-foreman-installer* includes the installer for Foreman to quickly deploy Red Hat OpenStack. Install the *openstack-foreman-installer* package on **desktop*X***, using **yum**.

```
[root@desktopX ~]# yum install -y openstack-foreman-installer
```

☐ 2. Download the **foreman-params.env** file from **instructor.example.com**:

```
[root@desktopX ~]# wget http://instructor.example.com/pub/materials/foreman-
params.env
```

☐ 3. Edit the **foreman-params.env** file and change **X+100** to the actual value (in both places: **PRIVATE_CONTROLLER_IP** and **PUBLIC_CONTROLLER_IP**). E.g., if you were on desktop7, change **X+100** to **107**, or if you were on desktop17, change **X+100** to **117**.

☐ 4. Source the variables from the script into your shell environment.

```
[root@desktopX ~]# source foreman-params.env
```

☐ 5. Install Foreman with the provided script.

```
[root@desktopX ~]# cd /usr/share/openstack-foreman-installer/bin
[root@desktopX ~]# sh foreman_server.sh
#################### RED HAT OPENSTACK ####################
Thank you for using the Red Hat OpenStack Foreman Installer!
###########################################################
Press [Enter] to continue Enter
...
Notice: Finished catalog run in 239.15 seconds
...
You'll find Foreman at https://desktopX.example.com
The user name is 'admin' and default password is 'changeme'.
Please change the password at https://desktopX.example.com/users/1-admin/edit
...
```

☐ 6. Once the installation is done, log into the Foreman dashboard at: `https://desktopX.example.com` with the username **admin** and the password **changeme**.

☐ 7. For security reasons, it is advisable to change the default admin user password. To do that, select the **Admin User** dropdown menu and then the **My account** option in the top-right corner of the screen to access account settings. The **Edit User** screen is displayed. Enter redhat as the new password in the **Password** field. Enter redhat again in the password **Verified** field. Click the **Submit** button to save the change.

# Deploying Red Hat OpenStack with Foreman

## Foreman host groups

Foreman uses what is referred to as a host group definition to group hosts that share common configuration requirements together. Two host groups are provided with the version of Foreman included in the Red Hat Enterprise Linux OpenStack platform:

## OpenStack controller host group

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

- OpenStack dashboard (Horizon).
- OpenStack image storage 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), also run on the controller.

## OpenStack Nova compute host group

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.

**Workshop**

## Deploying OpenStack with Foreman

Follow along with the instructor as you perform the setup tasks required to configure the Red Hat Foreman software to deploy Red Hat OpenStack instances.

☐ 1. Before deploying Red Hat OpenStack on our machines with Foreman, we have to register the server*X*.example.com machine with Foreman. Open a terminal on **server*X*** (use **virt-viewer** or **ssh**) and run the following commands:

```
[root@serverX ~]# scp root@desktopX:/tmp/foreman_client.sh /root/
[root@serverX ~]# sh /root/foreman_client.sh
```

Ignore the warning about being unable to fetch the host definition. We will manually pull it down later.

☐ 2. Edit the host groups in the Foreman dashboard. Go to `https://desktopX.example.com/hostgroups` and click on the **Controller (Neutron)** link. Click on the **Parameters** tab (make sure Firefox is wide enough to see the **Action** column). Find the **admin_password** row and press the **override** button. At the bottom of the page,

highlight the current **admin_password** in the box and replace it with **redhat**. When done, press the **Submit** button.

☐ 3. We want to perform the same steps to change the **admin_password** to **redhat** for the compute host. We also want to change the ethernet interfaces from **eth0** to **br100**. Click on the **Compute (Neutron)** link and browse to the **Parameters** tab. Find the **admin_password** row and press the **override** button. Find the **private_interface** row and press the **override** button. Find the **public_interface** row and press the **override** button. At the bottom of the page, change the**admin_password** to **redhat** and change both **private_interface** and **public_interface** to **br100**. Press the **Submit** button.

☐ 4. Map the host groups to our machines. Browse to the **Hosts** tab at the top of the Foreman web interface. You should see server*X*.example.com and desktop*X*.example.com listed. Find the server*X*.example.com machine and press the **Edit** button in that row. Select the **Controller (Neutron)** host group from the drop-down menu and click the **Submit** button.

☐ 5. By default, if the Puppet service is running, it will check every 30 minutes if there are any changes scheduled in Foreman for the particular host. Since we do not want to wait that long, we can speed up the installation by forcing a recheck on server*X*.example.com. Open a terminal on **server*X*.example.com** and run the following command.

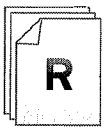
```
[root@serverX ~]# puppet agent -tv
```

☐ 6. Once the installation of the OpenStack controller is finished, log into the OpenStack Horizon dashboard with the web browser at `http://serverX.example.com/dashboard`. Use a username of **admin** and a password of **redhat**.

☐ 7. Go back to the Foreman dashboard and go to the **Hosts** tab. In the **desktop*X*.example.com** machine row, press the **Edit** button. Select **Compute (Neutron)** in the **Host Group** drop-down menu and press the **Submit** button. Open a **root** terminal on **desktop*X*.example.com** and speed up the deployment of Nova compute on the **desktop*X*.example.com** machine:

```
[root@desktopX ~]# puppet agent -tv
```

☐ 8. After the installation is finished, you should be able to see the added hypervisor in the OpenStack dashboard on `http://serverX.example.com/dashboard`. Login as **admin** (password: **redhat**). In the **Admin** tab in the left pane, click on the **Hypervisors** link. If everything has been done properly, you will find desktop*X*.example.com as the hypervisor.




## References

Red Hat OpenStack Deployment Guide;

- Section 2. Installing Foreman (technical preview)
- Section 3. Configuring Foreman (technical preview)

# Chapter Test

Case Study

## Installing Red Hat OpenStack

***Before you begin...***

Reset your **server*X*** virtual machine.

If you are in a physical classroom, log in as **root** on **desktop*X*.example.com** and reset your **server*X*** machine:

```
[root@desktopX ~]# lab-reset-vm
This will destroy the virtual machine and reset it to the last saved state.
Is this ok [y/N]: y
Waiting for things to settle...
Done.
```

If you are working in the virtual training environment, ignore the preceding paragraph and use the virtual machine controls in your web browser to reset your machine to a snapshot of your **server*X*** machine. Open the **state** dropdown menu and select the **Snapshots** tab. Select **Initial Snapshot** via the radio buttons and click the **Revert to selected snapshot** button. Press the **Power On** button to start **server*X*.example.com**.

You must disable the hosts in Foreman. Log in to the Foreman web interface (`https://desktopX.example.com/`) as the **admin** user with a password of **redhat**. Browse to the **Hosts** tab. In the **desktop*X*.example.com** row, open the drop-down menu and choose **Delete**. Press the **OK** button. Do the same for **server*X*.example.com**.

You must also disable the OpenStack services running on **desktop*X*.example.com** configured by Foreman.

```
[root@desktopX ~]# service openstack-ceilometer-compute stop
[root@desktopX ~]# chkconfig openstack-ceilometer-compute off
[root@desktopX ~]# service openstack-nova-compute stop
[root@desktopX ~]# chkconfig openstack-nova-compute off
[root@desktopX ~]# service neutron-openvswitch-agent stop
[root@desktopX ~]# chkconfig neutron-openvswitch-agent off
```

After you have reset your virtual machine, **ssh** to **server*X*.example.com**. Update the software. Log into **server*X*.example.com** and install the packages necessary for **packstack**. Configure Red Hat OpenStack on **server*X*.example.com** according to the following table.

**Instance parameters**

| *Category* | *Parameter/value* |
|---|---|
| Red Hat OpenStack information | • Configure SSH keys<br>• Use **192.168.0.254** for the NTP server (unless you are using the Red Hat Online Learning environment)<br>• Configure Horizon to use SSL<br>• Project name: **tenant1** |

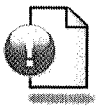
| *Category* | *Parameter/value* |
|---|---|
| | • User account name: **user1**<br>• User account email: **root@desktop*X*.example.com**<br>• User account password: **redhat** |
| Image information | • Image name: **small**<br>• Image location: `http://instructor.example.com/pub/materials/small.img`<br>• Image format: QCOW2<br>• Image settings: No minimum disk, no minimum RAM, public |
| Private network information | • Private network name: **private**<br>• Private subnet name: **privatesub**<br>• Private network range: **192.168.32.0/24**<br>• Private IP version: IPv4<br>• Private gateway IP: *Leave blank, but not disabled* |
| Public network information | • Public network name: **public**<br>• Public subnet name: **publicsub**<br>• Public network range: **172.24.*X*.0/24**<br>• Public IP version: IPv4<br>• Public gateway IP: **172.24.*X*.254**<br>• Public DHCP: disabled<br>• Public allocation pools: **172.24.*X*.1,172.24.*X*.100** |
| Router information | • Router name: **router1**<br>• Set the public network as an external network<br>• Assign the public network as the gateway for the router<br>• Add an interface for the private subnet to the router |
| Security group information | • Security group name: **secgrp**<br>• Security group description: **SSH and Web**<br>• Allow SSH from CIDR 0.0.0.0/0<br>• Allow HTTPS from CIDR 0.0.0.0/0 |

| *Category* | *Parameter/value* |
|---|---|
| | • Allow HTTP from this source group |
| Instance information | • Instance name: **small**<br>• Instance flavor: **m1.tiny**<br>• Instance Boot Source: **Boot from image.**<br>• Instance image: **small**<br>• Instance keypair: **key2**<br>• Instance security group: **secgrp**<br>• Instance floating IP address: **172.24.*X*.2** |
| Volume information | • Volume name: **myvol2**<br>• Volume description: **myvol2 volume**<br>• Volume size (GB): **2**<br>• Volume snapshot name: **myvol2-snap1**<br>• Volume snapshot description: **myvol2-snap1** |

If you need to troubleshoot your installation, you may want to disable debugging in Nova (**debug = False** in **/etc/nova/nova.conf**) and restart the **openstack-nova-*** services.

*How would you address the case study described above? Take notes on your process in the space below and then implement it.*

# Personal Notes



# Summary

Installing Red Hat OpenStack with packstack

In this section you learned how to:

- Install Red Hat OpenStack software.

Using the Horizon web interface

- In this section we will explore the Horizon web interface.

Deploying with Foreman

In this section you learned how to:

- Install Foreman.
- Install Red Hat OpenStack software with Foreman.
- Deploy Red Hat OpenStack instances with Foreman.

# CHAPTER 3

# IMPLEMENTING THE QPID MESSAGE BROKER

## Introduction

| Chapter details | |
|---|---|
| **Chapter goal** | Install and configure the Qpid message broker. Secure Qpid using authentication and encryption. |
| **Chapter sections** | • Installing and securing the Qpid message broker |
| **Hands-on activities** | • Installing and securing the Qpid message broker |

# Installing and securing the Qpid message broker

All Red Hat OpenStack services use the Qpid messaging system to communicate. There are two ways to secure Qpid communication. First, requiring a username and password (authentication) ensures that only machines with this username and password can communicate with the other OpenStack services. Second, using SSL to encrypt communication helps to prevent snooping and injection of rogue commands in the communication channels. In this chapter, you will learn how to secure Qpid using both of these methods.

When first installing or troubleshooting Qpid, you may want to install the *qpid-tools* package, which includes several Qpid testing tools.

When configuring Qpid to use SSL, you may have to change the ports that the services listen on. TCP/5672 is that standard port for Qpid, but SSL will use TCP/5671. You will likely have to change the **qpid_port** to **5671** for Glance, Cinder, etc.

Workshop

## Installing and securing the Qpid message broker

***Before you begin...***

Reset your **server*X*** virtual machine.

If you are in a physical classroom, log in as **root** on **desktop*X*.example.com** and reset your **server*X*** machine:

```
[root@desktopX ~]# lab-reset-vm
This will destroy the virtual machine and reset it to the last saved state.
Is this ok [y/N]: y
Waiting for things to settle...
Done.
```

If you are working in the virtual training environment, ignore the preceding paragraph and use the virtual machine controls in your web browser to reset your **server*X*** machine. Press the **Power Off** button. Once the machine has powered off, expand the state bar at the top of the page. Select the **Snapshots** tab. Select **Initial Snapshot** and press the **Revert to selected snapshot** button. Wait for a few seconds while the image is reset, then press the **Power On** button.

Perform the following steps on **server*X*.example.com** unless instructed otherwise.

☐ 1. Install updates and power off **server*X*.example.com**. On **desktop*X*.example.com**, save the state of the virtual machine. This will automatically start the **server*X*.example.com** machine.

```
[root@serverX ~]# yum update -y
```

If you are in a physical classroom, log in as **root** on **desktop*X*.example.com** and save the state of your **server*X*** machine:

```
[root@serverX ~]# poweroff
```

```
[root@desktopX ~]# lab-save-vm
This will save the current state of the virtual machine.
Is this ok [y/N]: y
```

If you are working in the virtual training environment, ignore the preceding paragraph and use the virtual machine controls in your web browser to create a snapshot of your **serverX** machine. First, shut down **serverX.example.com**. When **serverX.example.com** is shut down, open the **State** dropdown menu and select the **Snapshots** tab. Enter **Chapter 3** as the name in the **Create new snapshots...** box and press the **Create** button. Press the **Refresh** button to verify that the new snapshot was created. Press the **Power On** button to start **serverX.example.com**.

☐ 2. Once the machine has booted, log into **serverX.example.com** and install the packages necessary for Qpid:

```
[root@serverX ~]# yum install -y qpid-cpp-server qpid-cpp-server-ssl cyrus-sasl-
md5
```

☐ 3. Create a new SASL user and password for use with Qpid (**qpidauth:redhat**). Note that SASL uses the **QPID** realm by default.

```
[root@serverX ~]# saslpasswd2 -f /var/lib/qpidd/qpidd.sasldb -u QPID qpidauth
Password: redhat
Again (for verification): redhat
```

☐ 4. Verify the user was created.

```
[root@serverX ~]# sasldblistusers2 -f /var/lib/qpidd/qpidd.sasldb
qpidauth@QPID: userPassword
```

☐ 5. Provide authorization for the **qpidauth** user.

```
[root@serverX ~]# echo 'acl allow qpidauth@QPID all all' > /etc/qpid/qpidauth.acl
[root@serverX ~]# echo "QPIDD_OPTIONS='--acl-file /etc/qpid/qpidauth.acl'" >> /
etc/sysconfig/qpidd
[root@serverX ~]# chown qpidd /etc/qpid/qpidauth.acl
[root@serverX ~]# chmod 600 /etc/qpid/qpidauth.acl
```

☐ 6. Disable anonymous connections in **/etc/qpidd.conf** (remove **ANONYMOUS**). The **/etc/qpidd.conf** file should contain:

```
cluster-mechanism=DIGEST-MD5
auth=yes
```

☐ 7. Now that the username and password are configured, begin work on the SSL mechanisms. Create a new directory for Qpid certificates and protect the directory.

```
[root@serverX ~]# mkdir /etc/pki/tls/qpid
[root@serverX ~]# chmod 700 /etc/pki/tls/qpid/
```

```
[root@serverX ~]# chown qpidd /etc/pki/tls/qpid/
```

☐ 8. Create a password file for the certificate.

```
[root@serverX ~]# echo redhat > /etc/qpid/qpid.pass
[root@serverX ~]# chmod 600 /etc/qpid/qpid.pass
[root@serverX ~]# chown qpidd /etc/qpid/qpid.pass
```

☐ 9. Generate the certificate database. Ensure **$HOSTNAME** is properly set.

```
[root@serverX ~]# echo $HOSTNAME
serverX.example.com
[root@serverX ~]# certutil -N -d /etc/pki/tls/qpid/ -f /etc/qpid/qpid.pass
[root@serverX ~]# certutil -S -d /etc/pki/tls/qpid/ -n $HOSTNAME -s "CN=$HOSTNAME"
 -t "CT,," -x -f /etc/qpid/qpid.pass -z /usr/bin/certutil
Generating key.  This may take a few moments...
```

☐ 10. Make sure the certificate directory is readable by the **qpidd** user.

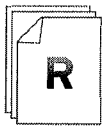
```
[root@serverX ~]# chown -R qpidd /etc/pki/tls/qpid/
```

☐ 11. Add the following to **/etc/qpidd.conf**:

```
ssl-cert-db=/etc/pki/tls/qpid/
ssl-cert-name=serverX.example.com
ssl-cert-password-file=/etc/qpid/qpid.pass
require-encryption=yes
```

☐ 12. Start the **qpidd** service, check for errors, and make it persistent:

```
[root@serverX ~]# service qpidd start
[root@serverX ~]# tail /var/log/messages
[root@serverX ~]# chkconfig qpidd on
```




## References

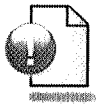
Red Hat OpenStack Installation and Configuration Guide

- Chapter 4. Installing the message broker

Apache Qpid AMQP messaging broker (implemented in C++) (`http://qpid.apache.org/releases/qpid-0.14/books/AMQP-Messaging-Broker-CPP-Book/html/`)

- Section 1.5. Security

# Personal Notes



# Summary

Installing and securing the Qpid message broker

- Install Qpid.
- Secure Qpid using authentication.
- Secure Qpid using SSL.

# CHAPTER 4

# IMPLEMENTING THE KEYSTONE IDENTITY SERVICE

## Introduction

| Chapter details | |
|---|---|
| **Chapter goal** | Install, configure, and use the Keystone identity service. |
| **Chapter sections** | • Deploying the Keystone identity service<br>• Managing users with the **keystone** command |
| **Hands-on activities** | • Deploying the Keystone identity service<br>• Creating the Keystone **admin** user |
| **Chapter test** | Adding a new user to Keystone |

# Deploying the Keystone identity service

## What is the Keystone identity service?

Keystone identity service is a project providing identity, token, catalog, and policy services for use with Red Hat OpenStack. Keystone provides token- and password-based authentication (authN) and high-level authorization (authZ), and a central directory of users mapped to the services they can access.

## Add services to the Keystone service catalog and register their end points

The **keystone service-create** command needs three options to register a service:

```
[root@serverX ~]# keystone service-create --name=SERVICENAME --type=SERVICETYPE --
description="DESCRIPTION OF SERVICE"
```

While **--name** and **--description** can be user-selected strings, the argument passed with the **--type** switch must be one of identity, compute, network, image, or object-store.

After a service is registered in the service catalog, the end point of the service can be defined:

```
[root@serverX ~]# keystone endpoint-create --service-id SERVICEID --publicurl 'URL' --
adminurl 'URL' --internalurl 'URL'
```

The **--service-id** can be obtained from the output of the **service-create** command shown previously or by getting the information from the Keystone service catalog with the command **keystone service-list**.

## Remove services and end points

Of course, it is also possible to remove service end points and services from the Keystone catalog.

To delete an end point, figure out its id with:

```
[root@serverX ~]# keystone endpoint-list
```

Next, delete the end point of choice with:

```
[root@serverX ~]# keystone endpoint-delete ENDPOINTID
```

Deleting a service is quite similar. First query the catalog to get a list of services and their service IDs:

```
[root@serverX ~]# keystone service-list
```

Then, remove the service with:

```
[root@serverX ~]# keystone service-delete SERVICEID
```

OpenStack configuration files use the INI format, where there can be separate sections. Each section uses à name enclosed in square brackets ([]). All OpenStack configuration files include a **DEFAULT** section; some OpenStack configuration files include other sections. The OpenStack configuration files can be managed with the **openstack-config** command. The example that follows explains the options and arguments used with **openstack-config**.

```
[root@serverX ~]# openstack-config --set❶ /etc/keystone/keystone.conf❷ DEFAULT❸
 admin_token❹ abcdef1234567890❺
```

❶ *Option*: The option can be one of **--set** or **--del** to set or delete a parameter/value pair.

❷ *Configuration file*: This is the location of the OpenStack configuration file.

❸ *Section*: This is the section name. The OpenStack configuration file will have parameter/value pairs beneath each section. In the OpenStack configuration file, the section name can be found within square brackets, e.g., **[DEFAULT]**.

❹ *Parameter*: This is the parameter to be set.

❺ *Value*: This is the value to be set.

If you ran the previous command, the **/etc/keystone/keystone.conf** file would include the following:

```
[DEFAULT]
admin_token = abcdef1234567890
...
```

Workshop

## Deploying the Keystone identity service

Follow along with the instructor as you perform the setup tasks required to install the Red Hat OpenStack software.

We are going to deploy the Keystone identity service without using the **packstack** command now. This will give us complete control over the installation and allow us to, for example, install it on a separate machine.

☐ 1. Perform the following steps on **serverX.example.com** unless otherwise specified.

Install the *openstack-keystone* package with **yum**. Also, install the *openstack-selinux* package that will provide the SELinux policy for Red Hat OpenStack.

```
[root@serverX ~]# yum install -y openstack-keystone openstack-selinux
```

☐ 2. Next, it is time to get the database back end installed. The **openstack-db** command will take care of installing and initializing MySQL for the Keystone service. Find this command in the *openstack-utils* package. Install and start the MySQL server and enter **redhat** for the root MySQL user.

```
[root@serverX ~]# yum install -y openstack-utils
[root@serverX ~]# openstack-db --init --service keystone
...
mysql-server is not installed.  Would you like to install it now? (y/n): y
...
```

```
Total download size: 10 M
Installed size: 29 M
Is this ok [y/N]: y
...
mysqld is not running.  Would you like to start it now? (y/n): y
...
Since this is a fresh installation of MySQL, please set a password for the 'root'
 mysql user.
Enter new password for 'root' mysql user: redhat
Enter new password again: redhat
Verified connectivity to MySQL.
Creating 'keystone' database.
Initializing the keystone database, please wait...
Complete!
```

☐ 3. Setup the PKI infrastructure for Keystone.

```
[root@serverX ~]# keystone-manage pki_setup --keystone-user keystone --keystone-
group keystone
```

☐ 4. To be able to administrate the Keystone identity service, specify the **SERVICE_TOKEN** and **SERVICE_ENDPOINT** environment variables. Save the value of the generated **SERVICE_TOKEN** to a file for later use.

```
[root@serverX ~]# export SERVICE_TOKEN=$(openssl rand -hex 10)
[root@serverX ~]# export SERVICE_ENDPOINT=http://serverX.example.com:35357/v2.0
[root@serverX ~]# echo $SERVICE_TOKEN > /root/ks_admin_token
[root@serverX ~]# cat /root/ks_admin_token
0123456789abcdef0123
```

☐ 5. The generated **SERVICE_TOKEN** must correspond to the **admin_token** setting in the **/etc/keystone/keystone.conf** file.

```
[root@serverX ~]# openstack-config --set /etc/keystone/keystone.conf DEFAULT
 admin_token $SERVICE_TOKEN
```

☐ 6. Start the **openstack-keystone** service and make sure the service is persistent.

```
[root@serverX ~]# service openstack-keystone start
[root@serverX ~]# chkconfig openstack-keystone on
```

☐ 7. To verify success, check if the **keystone-all** process is running.

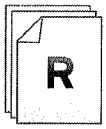
```
[root@serverX ~]# ps -ef | grep keystone-all
[root@serverX ~]# grep ERROR /var/log/keystone/keystone.log
```

☐ 8. Add Keystone as an end point in the registry of end points in Keystone, which is required for the Horizon web dashboard. Note that the ID returned from the **service-create** command is then used as a part of the **endpoint-create** command:

```
[root@serverX ~]# keystone service-create --name=keystone --type=identity --
description="Keystone Identity Service"
+-------------+----------------------------------+
```

```
|   Property  |              Value               |
+-------------+----------------------------------+
| description |    Keystone Identity Service     |
|      id     | dcba9876543210fedcba9876543210fe |
|     name    |             keystone             |
|     type    |             identity             |
+-------------+----------------------------------+
[root@serverX ~]# keystone endpoint-create --service-
id dcba9876543210fedcba9876543210fe --publicurl 'http://serverX.example.com:5000/
v2.0' --adminurl 'http://serverX.example.com:35357/v2.0' --internalurl 'http://
serverX.example.com:5000/v2.0'
+-------------+-----------------------------------+
|   Property  |               Value               |
+-------------+-----------------------------------+
|   adminurl  | http://serverX.example.com:35357/v2.0 |
|      id     |    dad1234567890dad1234567890dad123   |
| internalurl |  http://serverX.example.com:5000/v2.0 |
|  publicurl  |  http://serverX.example.com:5000/v2.0 |
|    region   |               regionOne           |
|  service_id |    dcba9876543210fedcba9876543210fe   |
+-------------+-----------------------------------+
```

Check the output carefully for mistakes. If needed, delete the end point (**keystone endpoint-delete *ID***), then recreate it.




## References

Red Hat OpenStack Installation and Configuration Guide
- Chapter 5. Installing the OpenStack identity service

The **`openstack-config`**(1) man page.

# Managing users with the keystone command

The **keystone** command can be used to create, delete, and modify users.

Before starting to use the command, it is important to source our environment variables to have administrative permissions:

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

Adding a new user with the username *USERNAME* and a password of *PASSWORD* is as simple as:

```
[root@serverX ~]# keystone user-create --name USERNAME --pass PASSWORD
```

To list existing users and their user IDs, use the command:

```
[root@serverX ~]# keystone user-list
```

To delete a user issue:

```
[root@serverX ~]# keystone user-delete USERID
```

## Manage tenants with the keystone command

Tenants can be created, listed, and deleted as well with the **keystone** command.

To create a tenant named *TENANTNAME*:

```
[root@serverX ~]# keystone tenant-create --name TENANTNAME
```

For listing all tenants, run:

```
[root@serverX ~]# keystone tenant-list
```

To delete a tenant issue:

```
[root@serverX ~]# keystone tenant-delete TENANTID
```

## Roles in Keystone

By default, there are two standard roles defined in Keystone:

- *admin* - a role with administrative privileges
- *member* - a role of a project member

Even though the definitions for the roles are present, they still have to be added manually to the Keystone catalog if Keystone is manually deployed.

For example, to add the role *member* to the Keystone catalog, use:

```
[root@serverX ~]# keystone role-create --name Member
```

## Associate a user from a specific tenant with a role

Of course, we also have to be able to add one or more roles to a user.

To accomplish this, it is necessary to have the *USERID*, the *TENANTID*, and the *ROLEID* we want to attach the user to, then connect them with:

```
[root@serverX ~]# keystone user-role-add --user-id USERID --role-id ROLEID --tenant-
id TENANTID
```

## Is this all I can do with the keystone command line?

There are several additional commands for various tasks. To explore more of the **keystone** command line, take a look at:

```
[root@serverX ~]# keystone help
    endpoint-create      Create a new endpoint associated with a service
    endpoint-delete      Delete a service endpoint
    endpoint-get
    endpoint-list        List configured service endpoints
    role-create          Create new role
    role-delete          Delete role
    role-get             Display role details
    role-list            List all roles
    service-create       Add service to Service Catalog
    service-delete       Delete service from Service Catalog
    service-get          Display service from Service Catalog
    service-list         List all services in Service Catalog
    tenant-create        Create new tenant
    tenant-delete        Delete tenant
    tenant-get           Display tenant details
    tenant-list          List all tenants
    tenant-update        Update tenant name, description, enabled status
    token-get
    user-create          Create new user
    user-delete          Delete user
    user-get             Display user details.
    user-list            List users
    user-password-update Update user password
    user-role-add        Add role to user
    user-role-list       List roles granted to a user
    user-role-remove     Remove role from user
    user-update          Update user's name, email, and enabled status
```

For more specific help on a particular command-line option, try, for example:

```
[root@serverX ~]# keystone help user-role-list
```

Workshop

## Creating the Keystone admin user

Follow along with the instructor as you perform the setup tasks required to install the Red Hat OpenStack software.

To finish the setup of the Keystone environment, create an **admin** user. The **admin** user of the **admin** tenant has to be associated with an **admin** role. A **keystonerc_admin** script makes authentication as the **admin** user easy.

☐ 1. Create the **admin** user with a corresponding password.

```
[root@serverX ~]# keystone user-create --name admin --pass redhat
+----------+--------------------------------------+
| Property |              Value                   |
+----------+--------------------------------------+
|  email   |                                      |
| enabled  |              True                    |
|    id    |  34567890abcdef1234567890abcdef12    |
|   name   |              admin                   |
| tenantId |                                      |
+----------+--------------------------------------+
```

☐ 2. Create an **admin** role as well.

```
[root@serverX ~]# keystone role-create --name admin
+----------+---------------------------------+
| Property |              Value              |
+----------+---------------------------------+
|    id    | fad9876543210fad9876543210fad987 |
|   name   |              admin              |
+----------+---------------------------------+
```

☐ 3. Create the **admin** tenant as well.

```
[root@serverX ~]# keystone tenant-create --name admin
+-------------+---------------------------------+
|   Property  |              Value              |
+-------------+---------------------------------+
| description |                                 |
|   enabled   |               True              |
|      id     | 4567890abcdef1234567890abcdef123 |
|     name    |              admin              |
+-------------+---------------------------------+
```

☐ 4. Add the user from the **admin** tenant to the **admin** role.

```
[root@serverX ~]# keystone user-role-add --user admin --role admin --tenant admin
```

☐ 5. Create the **keystonerc_admin** script.

```
[root@serverX ~]# cat >> /root/keystonerc_admin << EOF
> export OS_USERNAME=admin
> export OS_TENANT_NAME=admin
> export OS_PASSWORD=redhat
> export OS_AUTH_URL=http://serverX.example.com:35357/v2.0/
> export PS1='[\u@\h \W(keystone_admin)]\\$ '
> EOF
```

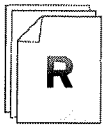
☐ 6. Test the **keystonerc_admin** file by running the command to list users. Only an administrator can perform this action. Start by unsetting the token and end point created earlier so as to verify the **keystonerc_admin** file.

```
[root@serverX ~]# unset SERVICE_TOKEN
[root@serverX ~]# unset SERVICE_ENDPOINT
[root@serverX ~]# source /root/keystonerc_admin
[root@serverX ~(keystone_admin)]# keystone user-list
+----------------------------------+-------+---------+-------+
|                id                |  name | enabled | email |
+----------------------------------+-------+---------+-------+
| 34567890abcdef1234567890abcdef12 | admin |   True  |       |
+----------------------------------+-------+---------+-------+
```

## Note

If you need to troubleshoot Keystone because the **/root/keystonerc_admin** file did not work you must export the two variables:

```
[root@serverX ~]# export SERVICE_TOKEN=$(cat /root/ks_admin_token)
[root@serverX ~]# export SERVICE_ENDPOINT=http://
serverX.example.com:35357/v2.0
```




## References

Red Hat OpenStack Installation and Configuration Guide

- Section 5.7. Creating an administrator account
- Section 5.8. Creating a regular user account
- Section 5.10. Validating the identity service installation

# Chapter test

Case Study

## Adding a new user to Keystone

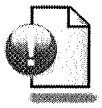
Create a new user with the **keystone** command according to the following specifications:

- The username is **myuser** with a password of **redhat**.
- The user is part of the **myopenstack** tenant.
- The user is attached to the **Member** role.

For easier testing, create a **keystonerc_myuser** file in **root**'s home directory. Verify the user exists and the **keystonerc_myuser** works by getting a token (**keystone token-get**).

*How would you address the case study described above? Take notes on your process in the space below and then implement it.*

# Personal Notes



# Summary

Deploying the Keystone identity service

- Deploy the Keystone identity service manually.

# CHAPTER 5

# IMPLEMENTING THE SWIFT OBJECT STORAGE SERVICE

## Introduction

| Chapter details | |
|---|---|
| **Chapter goal** | Install, configure, and use the Swift object storage service. |
| **Chapter sections** | • Installing the Swift object storage service<br>• Deploying a Swift storage node<br>• Configuring Swift object storage service rings<br>• Deploying the Swift object storage proxy service<br>• Validating Swift object storage |
| **Hands-on activities** | • Installing the Swift object storage service<br>• Deploying a Swift storage node<br>• Configuring Swift object storage service rings<br>• Deploying the Swift object storage proxy service<br>• Validating Swift object storage |
| **Chapter test** | None |

# Installing the Swift object storage service

## What is the Swift object storage service?

The object storage service provides object storage in virtual containers, which allows users to store and retrieve files. The service's distributed architecture supports horizontal scaling; redundancy as failure-proofing is provided through software-based data replication. Because it supports asynchronous eventual consistency replication, it is well-suited to multiple data-center deployment.

Object storage uses the concept of:

- *Storage replicas*: used to maintain the state of objects in the case of outage. A minimum of three replicas is recommended.
- *Storage zones*: 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*: essentially a group of zones sharing a location. Regions can be, for example, groups of servers or server farms, usually located in the same geographical area. Regions have a separate API end point per object storage service installation, which allows for a discrete separation of services.

## Architecture of the object storage service

The OpenStack object storage service is a modular service with the following components:

- *openstack-swift-proxy*: 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 also 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. It exposes the public API, and is responsible for handling requests and routing them accordingly. Objects are streamed through the proxy server to the user (not spooled). Objects can also be served out via HTTP.
- *openstack-swift-object*: The object service is responsible for storing data objects in partitions on disk devices. Each partition is a directory, and 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. The service stores, retrieves, and deletes objects.
- *openstack-swift-container*: The container service maintains databases of objects in containers. There is one database file for each container, and they 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. The container service is responsible for listings of containers using the account database.
- *openstack-swift-account*: The account service maintains databases of all of the containers accessible by any given account. There is one database file for each account, and they 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. The account service handles listings of objects (what objects are in a specific container) using the container database.

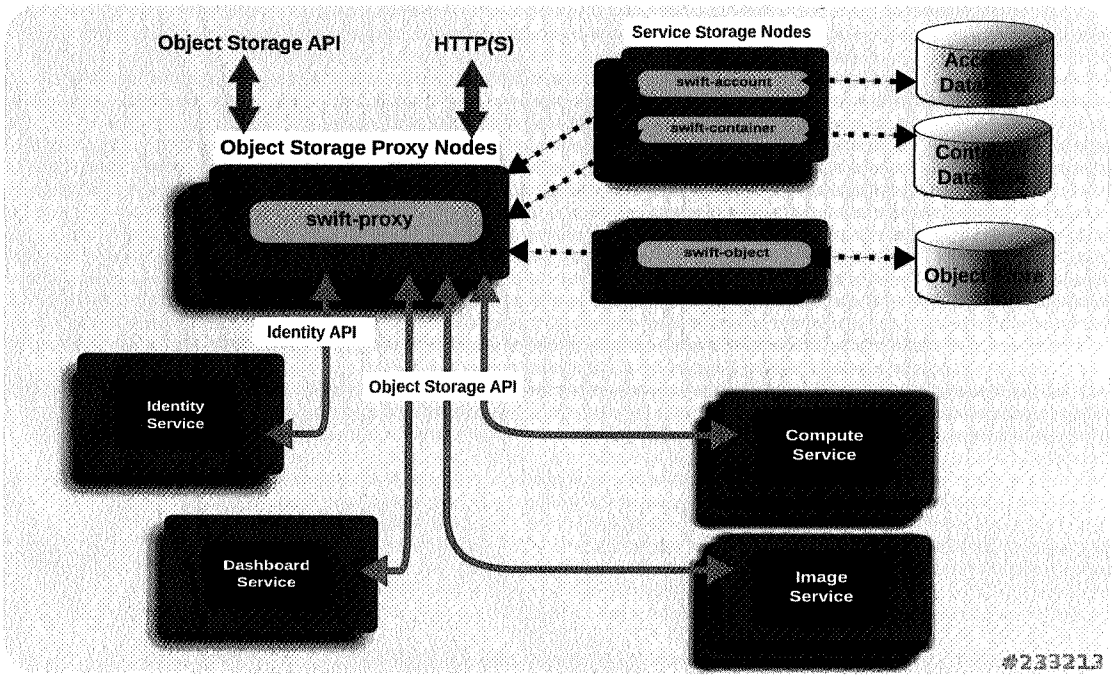
All of the services can be installed on each node or alternatively on dedicated machines. In addition, the following components are in place for proper operation:

- *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.
- *Object storage*: with either EXT4 (recommended) or XFS file system. The mount point is expected to be **`/srv/node`**.
- *Housekeeping processes*: for example replication and auditors.

# Object storage service deployment configurations

- *All services run on all nodes.*: Simplest setup.
- *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 shows the proxy and object nodes split out from the nodes containing the container and account nodes:



Workshop

# Installing the Swift object storage service

Follow along with the instructor as you perform the setup tasks required to set up Keystone for Swift.

We are going to prepare the Keystone identity service to be used with the Swift object storage service.

☐ 1. On **server*X*.example.com**, install the necessary components for the Swift object storage service.

```
[root@serverX ~]# yum install -y openstack-swift-proxy openstack-swift-object
 openstack-swift-container openstack-swift-account memcached
```

☐ 2. Make sure that the Keystone environment variables with the authentication information are loaded.

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

☐ 3. Create a Swift user with the password redhat.

```
[root@serverX ~(keystone_admin)]# keystone user-create --name swift --pass redhat
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |                                  |
| enabled  |               True               |
|    id    | 90abcdef1234567890abcdef12345678 |
|   name   |              swift               |
| tenantId |                                  |
+----------+----------------------------------+
```

☐ 4. Make sure the **admin** role exists before proceeding.

```
[root@serverX ~(keystone_admin)]# keystone role-list | grep admin
| fad9876543210fad9876543210fad987 |   admin   |
```

If there is no **admin** role, create one.

```
[root@serverX ~(keystone_admin)]# keystone role-create --name admin
```

☐ 5. Make sure the **services** tenant exists before proceeding.

```
[root@serverX ~(keystone_admin)]# keystone tenant-list | grep services
```

If there is no **services** tenant, create one.

```
[root@serverX ~(keystone_admin)]# keystone tenant-create --name services
+-------------+----------------------------------+
|   Property  |              Value               |
```

```
+-------------+----------------------------------+
| description |                                  |
|   enabled   |               True               |
|      id     | 890abcdef1234567890abcdef1234567 |
|     name    |             services             |
+-------------+----------------------------------+
```

☐ 6. Add the Swift user to the **services** tenant with the **admin** role.

```
[root@serverX ~(keystone_admin)]# keystone user-role-add --role admin --tenant
 services --user swift
```

☐ 7. Check if the object store service already exists in Keystone.

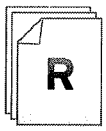
```
[root@serverX ~(keystone_admin)]# keystone service-list
```

If it does not exist, create it.

```
[root@serverX ~(keystone_admin)]# keystone service-create --name swift --type
 object-store --description "Swift Storage Service"
+-------------+----------------------------------+
|  Property   |              Value               |
+-------------+----------------------------------+
| description |      Swift Storage Service       |
|      id     | 325f9876543210efdbca9876543210ef |
|     name    |              swift               |
|     type    |          object-store            |
+-------------+----------------------------------+
```

☐ 8. Create the end points for the Swift object storage service.

```
[root@serverX ~(keystone_admin)]# keystone endpoint-create --service-
id 325f9876543210efdbca9876543210ef --publicurl "http://serverX.example.com:8080/
v1/AUTH_%(tenant_id)s" --adminurl "http://serverX.example.com:8080/v1/
AUTH_%(tenant_id)s" --internalurl "http://serverX.example.com:8080/v1/AUTH_
%(tenant_id)s"
+-------------+----------------------------------------------------------+
|   Property  |                          Value                           |
+-------------+----------------------------------------------------------+
|   adminurl  | http://serverX.example.com:8080/v1/AUTH_%(tenant_id)s    |
|      id     |           487ee7a9a501444682e525b95a945312               |
| internalurl | http://serverX.example.com:8080/v1/AUTH_%(tenant_id)s    |
|  publicurl  | http://serverX.example.com:8080/v1/AUTH_%(tenant_id)s    |
|   region    |                        regionOne                         |
| service-id  |            325f9876543210efdbca9876543210ef              |
+-------------+----------------------------------------------------------+
```




## References

Red Hat OpenStack Installation and Configuration Guide

- Chapter 6. Installing the OpenStack object storage service

# Deploying a Swift storage node

The object storage service stores objects on the file system, usually on a number of connected physical storage devices. All of the devices which will be used for object storage must be formatted with either ext4 or XFS, and mounted under the **/srv/node/** directory.

Any dedicated storage node needs to have the following packages installed: *openstack-swift-object*, *openstack-swift-container*, and *openstack-swift-account*.

**Workshop**

## Deploying a Swift storage node

Follow along with the instructor as you perform the setup tasks required to install a Swift storage node.

We are going to prepare and deploy a Swift storage node.

☐ 1. The **server*X*.example.com** virtual machine has some extra storage disks. If you are in a physical classroom, use the **lab-create-single-partition** script to create a single partition on **/dev/vdb** and a single partition on **/dev/vdc**.

```
[root@serverX ~]# lab-create-single-partition /dev/vdb
/dev/vdb: block special

Are you sure you want to continue?
This will destroy the partition table and all data on /dev/vdb. (y/N)  y
...
[root@serverX ~]# lab-create-single-partition /dev/vdc
/dev/vdc: block special

Are you sure you want to continue?
This will destroy the partition table and all data on /dev/vdc. (y/N)  y
...
```

If you are attending virtual training, ignore the preceding paragraph and use the **lab-create-single-partition** script to create a single partition on **/dev/sdb** and a single partition on **/dev/sdc**.

```
[root@serverX ~]# lab-create-single-partition /dev/sdb
/dev/sdb: block special

Are you sure you want to continue?
This will destroy the partition table and all data on /dev/sdb. (y/N)  y
...
[root@serverX ~]# lab-create-single-partition /dev/sdc
/dev/sdc: block special

Are you sure you want to continue?
This will destroy the partition table and all data on /dev/sdc. (y/N)  y
...
```

☐ 2. The node needs its local storage disks formatted with either xfs or ext4 (recommended).

If you are in a physical classroom, create an ext4 file system on **/dev/vdb1** and **/dev/vdc1**.

```
[root@serverX ~]# mkfs.ext4 /dev/vdb1
[root@serverX ~]# mkfs.ext4 /dev/vdc1
```

If you are attending virtual training, ignore the preceding paragraph and create an ext4 file system on **/dev/sdb1** and **/dev/sdc1**.

```
[root@serverX ~]# mkfs.ext4 /dev/sdb1
[root@serverX ~]# mkfs.ext4 /dev/sdc1
```

☐ 3. Create the mount points and mount the devices persistently to the appropriate zone directories.

If you are in a physical classroom, the first machine with its exported disk vdb1 acts as zone 1 (z1). The second disk will act as zone 2 (z2) and become a replica of zone 1 (z1).

```
[root@serverX ~]# mkdir -p /srv/node/z{1,2}d1
[root@serverX ~]# cp /etc/fstab /etc/fstab.orig
[root@serverX ~]# echo "/dev/vdb1 /srv/node/z1d1 ext4 acl,user_xattr 0 0" >> /etc/
fstab
[root@serverX ~]# echo "/dev/vdc1 /srv/node/z2d1 ext4 acl,user_xattr 0 0" >> /etc/
fstab
```

If you are attending virtual training, ignore the preceding paragraph. The first machine with its exported disk sdb1 acts as zone 1 (z1). The second disk will act as zone 2 (z2) and become a replica of zone 1 (z1).

```
[root@serverX ~]# mkdir -p /srv/node/z{1,2}d1
[root@serverX ~]# cp /etc/fstab /etc/fstab.orig
[root@serverX ~]# echo "/dev/sdb1 /srv/node/z1d1 ext4 acl,user_xattr 0 0" >> /etc/
fstab
[root@serverX ~]# echo "/dev/sdc1 /srv/node/z2d1 ext4 acl,user_xattr 0 0" >> /etc/
fstab
```

While using different zones for different disks is legitimate, one would use at least three separate storage nodes to act as three different zones in a real production setup, rather than having different disks.

☐ 4. Mount the new Swift storage disks.

```
[root@serverX ~]# mount -a
```

☐ 5. Change the ownership of the contents of **/srv/node** to **swift:swift**.

```
[root@serverX ~]# chown -R swift:swift /srv/node/
```

☐ 6. Restore the SELinux context of **/srv**.

```
[root@serverX ~]# restorecon -R /srv
```

☐ 7. Make backups of the files that will be changed.

```
[root@serverX ~]# cp /etc/swift/swift.conf /etc/swift/swift.conf.orig
[root@serverX ~]# cp /etc/swift/account-server.conf /etc/swift/account-
server.conf.orig
[root@serverX ~]# cp /etc/swift/container-server.conf /etc/swift/container-
server.conf.orig
[root@serverX ~]# cp /etc/swift/object-server.conf /etc/swift/object-
server.conf.orig
```

☐ 8. Use the **openstack-config** command to add a hash prefix and suffix to **/etc/swift/swift.conf**. These details are required for finding and placing data on all of the nodes. Back up **/etc/swift/swift.conf** after setting it.

```
[root@serverX ~]# openstack-config --set /etc/swift/swift.conf swift-hash
 swift_hash_path_prefix $(openssl rand -hex 10)
[root@serverX ~]# openstack-config --set /etc/swift/swift.conf swift-hash
 swift_hash_path_suffix $(openssl rand -hex 10)
```

☐ 9. The account container and object swift services need to bind to the same IP used for mapping the rings later on. Localhost only works for a single storage node configuration.

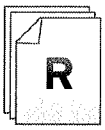
```
[root@serverX ~]# openstack-config --set /etc/swift/account-server.conf DEFAULT
 bind_ip 192.168.0.X+100
[root@serverX ~]# openstack-config --set /etc/swift/container-server.conf DEFAULT
 bind_ip 192.168.0.X+100
[root@serverX ~]# openstack-config --set /etc/swift/object-server.conf DEFAULT
 bind_ip 192.168.0.X+100
```

☐ 10. Start up the services and make them persistent.

```
[root@serverX ~]# service openstack-swift-account start
[root@serverX ~]# service openstack-swift-container start
[root@serverX ~]# service openstack-swift-object start
[root@serverX ~]# tail /var/log/messages
[root@serverX ~]# chkconfig openstack-swift-account on
[root@serverX ~]# chkconfig openstack-swift-container on
[root@serverX ~]# chkconfig openstack-swift-object on
```

## Note

If you configure multiple storage nodes, copy **`/etc/swift/swift.conf`** from the first node configured to all of your object storage service nodes.




## References

Red Hat OpenStack Installation and Configuration Guide

- Section 6.5.2. Configuring the object storage service storage nodes

# Configuring Swift 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:

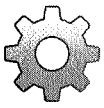
- Object
- Container
- Account services

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 file system 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 with 1000 devices with 100 partitions on each device 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 partition power is 17 (16.610 rounded up).

Expressed mathematically: 2 ^ partition power = partition count.

Ring files are generated using three parameters:
- *Partition power:* The value is calculated as shown previously and rounded up after calculation.
- *Replica count:* This represents the number of times the data gets replicated in the cluster.
- *min_part_hours:* This is the minimum number of hours before a partition can be moved. It ensures availability by not moving more than one copy of a given data item within the min_part_hours time period.

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



**Workshop**

## Configuring Swift object storage service rings

Follow along with the instructor as you perform the setup tasks required to configure the Swift service rings.

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.

☐ 1. Make sure that the Keystone environment variables with the authentication information are loaded in the terminal on **server*X*.example.com**.

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

☐ 2. Use the **swift-ring-builder** command to build one ring for each service.

```
[root@serverX ~(keystone_admin)]# swift-ring-builder /etc/swift/account.builder
 create 12 2 1
[root@serverX ~(keystone_admin)]# swift-ring-builder /etc/swift/container.builder
 create 12 2 1
[root@serverX ~(keystone_admin)]# swift-ring-builder /etc/swift/object.builder
 create 12 2 1
```

☐ 3. Add the devices to the account service.

```
[root@serverX ~(keystone_admin)]# for i in 1 2; do
> swift-ring-builder /etc/swift/account.builder add z${i}-192.168.0.X+100:6002/z
${i}d1 100
> done
```

☐ 4. Add the devices to the container service.

## Note

The file name and the port values change with each for loop.

```
[root@serverX ~(keystone_admin)]# for i in 1 2; do
> swift-ring-builder /etc/swift/container.builder add z${i}-192.168.0.X+100:6001/
z${i}d1 100
> done
```

☐ 5. Add the devices to the object service.

```
[root@serverX ~(keystone_admin)]# for i in 1 2; do
> swift-ring-builder /etc/swift/object.builder add z${i}-192.168.0.X+100:6000/z
${i}d1 100
> done
```

☐ 6. After successfully adding the devices, rebalance the rings.

```
[root@serverX ~(keystone_admin)]# swift-ring-builder /etc/swift/account.builder
 rebalance
[root@serverX ~]# swift-ring-builder /etc/swift/container.builder rebalance
[root@serverX ~]# swift-ring-builder /etc/swift/object.builder rebalance
```

☐ 7. Verify the ring files have been successfully created.

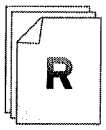
```
[root@serverX ~(keystone_admin)]# ls /etc/swift/*gz
/etc/swift/account.ring.gz  /etc/swift/container.ring.gz
/etc/swift/object.ring.gz
```

☐ 8. Make sure all files in the **/etc/swift** directory are owned by **root:swift**.

```
[root@serverX ~(keystone_admin)]# chown -R root:swift /etc/swift
```

## Note

The content of the **/etc/swift** directory needs to be copied to each node on the cluster into the **/etc/swift** directory.




## References

Red Hat OpenStack Installation and Configuration Guide

- Section 6.5.5. Building object storage service ring files

# Deploying the Swift object storage proxy service

The object storage proxy service determines to which node gets and puts are directed. While it can be installed alongside the account, container, and object services, it will usually end up on a separate system in production deployments.

**Workshop**

## Deploying the Swift object storage proxy service

Follow along with the instructor as you perform the configuration of the Swift object storage proxy service.

Configure the Swift object storage proxy service.

☐ 1. On **server*X*.example.com**, start by making a backup of the proxy configuration file:

```
[root@serverX ~]# cp /etc/swift/proxy-server.conf /etc/swift/proxy-
server.conf.orig
```

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

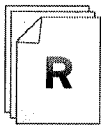
```
[root@serverX ~]# openstack-config --set /etc/swift/proxy-server.conf
 filter:authtoken admin_tenant_name services
[root@serverX ~]# openstack-config --set /etc/swift/proxy-server.conf
 filter:authtoken auth_host 192.168.0.X+100
[root@serverX ~]# openstack-config --set /etc/swift/proxy-server.conf
 filter:authtoken admin_user swift
[root@serverX ~]# openstack-config --set /etc/swift/proxy-server.conf
 filter:authtoken admin_password redhat
```

☐ 3. Enable the memcached and openstack-swift-proxy services permanently.

```
[root@serverX ~]# service memcached start
[root@serverX ~]# service openstack-swift-proxy start
[root@serverX ~]# tail /var/log/messages
[root@serverX ~]# chkconfig memcached on
[root@serverX ~]# chkconfig openstack-swift-proxy on
```

## Note

To provide a redundant setup, one would install multiple Swift proxies on different machines and use a load balancer or round-robin DNS.




## References

Red Hat OpenStack Installation and Configuration Guide

- Section 6.5.3. Configuring the object storage service proxy service

# Validating Swift object storage

After successfully installing the Swift object storage components, validate that it is working properly.

Workshop

## Validating Swift object storage

Follow along with the instructor as you perform the tests for Swift storage.

Validate the Swift storage setup.

☐ 1. Validate the Swift object storage service functionality by uploading three files into two containers.

```
[root@serverX ~]# source /root/keystonerc_admin
[root@serverX ~(keystone_admin)]# swift list
[root@serverX ~(keystone_admin)]# head -c 1024 /dev/urandom > data.file ; swift
 upload c1 data.file
[root@serverX ~(keystone_admin)]# head -c 1024 /dev/urandom > data2.file ; swift
 upload c1 data2.file
[root@serverX ~(keystone_admin)]# head -c 1024 /dev/urandom > data3.file ; swift
 upload c2 data3.file
```

☐ 2. View the list of containers.

```
[root@serverX ~(keystone_admin)]# swift list
c1
c2
```

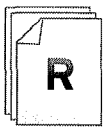
☐ 3. View the contents of the containers.

```
[root@serverX ~(keystone_admin)]# swift list c1
data.file
data2.file
[root@serverX ~(keystone_admin)]# swift list c2
data3.file
```

☐ 4. If you look in the various storage devices, you should see six **.data** files because each file has two copies:

```
[root@serverX ~(keystone_admin)]# find /srv/node/ -type f -name "*data"
/srv/node/z1d1/objects/3527/a06/
dc7bf1d3af32afad862dc4e51fb5ea06/1374699203.80346.data
/srv/node/z1d1/
objects/1470/462/5bea91be4b86637fdad8d69e12353462/1374699232.00766.data
/srv/node/z1d1/objects/3252/737/
cb49ae28aefe6d56256e6533c8509737/1374699168.49401.data
/srv/node/z2d1/objects/3527/a06/
dc7bf1d3af32afad862dc4e51fb5ea06/1374699203.80346.data
/srv/node/z2d1/objects/3252/737/
cb49ae28aefe6d56256e6533c8509737/1374699168.49401.data
```

```
/srv/node/z2d1/
objects/1470/462/5bea91be4b86637fdad8d69e12353462/1374699232.00766.data
```

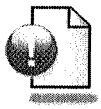



## References

Red Hat OpenStack Installation and Configuration Guide

- Section 6.6. Validating the object storage service installation

# Personal Notes



# Summary

Installing the Swift object storage service
- Deploy the Swift object storage service manually.

Deploying a Swift storage node
- In this section we will install the Swift storage node.

Configuring Swift object storage service rings
- In this section we will create the Swift object storage service rings.

Deploying the Swift object storage proxy service
- In this section we will install the Swift storage proxy service.

Validating Swift object storage
- In this section we will validate the Swift object storage installation.

# CHAPTER 6

# IMPLEMENTING THE GLANCE IMAGE SERVICE

## Introduction

| Chapter details | |
|---|---|
| **Chapter goal** | Install and use the Glance image service. |
| **Chapter sections** | • Deploying the Glance image service<br>• Using the **glance** command to upload a system image |
| **Hands-on activities** | • Deploying the Glance image service<br>• Using Glance to upload a system image |
| **Chapter test** | None |

# Deploying the Glance image service

The Glance image server requires Keystone to be in place for identity management and authorization. It uses a MySQL database to store the metadata information for the images. Glance supports a variety of disk formats, such as:

- raw
- vhd
- vmdk
- vdi
- iso
- qcow2
- aki, ari, and ami

And a variety of container formats:

- bare
- ovf
- aki, ari, and ami

To install Glance manually, start by installing the package and getting an initial configuration file.

```
[root@serverX ~]# yum install -y openstack-glance
[root@serverX ~]# cp /usr/share/glance/glance-registry-dist.conf /etc/glance/glance-
registry.conf
```

Initialize the database. Use a strong password.

```
[root@serverX ~]# openstack-db --init --service glance --password redhat
```

Update the Glance configuration to use Keystone as the identity service:

```
[root@serverX ~]# openstack-config --set /etc/glance/glance-api.conf paste_deploy flavor
 keystone
[root@serverX ~]# openstack-config --set /etc/glance/glance-api.conf keystone_authtoken
 admin_tenant_name admin
[root@serverX ~]# openstack-config --set /etc/glance/glance-api.conf keystone_authtoken
 admin_user admin
[root@serverX ~]# openstack-config --set /etc/glance/glance-api.conf keystone_authtoken
 admin_password PASSWORD
[root@serverX ~]# openstack-config --set /etc/glance/glance-registry.conf paste_deploy
 flavor keystone
[root@serverX ~]# openstack-config --set /etc/glance/glance-registry.conf
 keystone_authtoken admin_tenant_name admin
[root@serverX ~]# openstack-config --set /etc/glance/glance-registry.conf
 keystone_authtoken admin_user admin
[root@serverX ~]# openstack-config --set /etc/glance/glance-registry.conf
 keystone_authtoken admin_password PASSWORD
```

Start and enable the services.

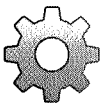
```
[root@serverX ~]# service openstack-glance-registry start
[root@serverX ~]# chkconfig openstack-glance-registry on
[root@serverX ~]# service openstack-glance-api start
[root@serverX ~]# chkconfig openstack-glance-api on
```

Finally, add the service to the Keystone catalog.

```
[root@serverX ~]# source ~/keystonerc_admin
[root@serverX ~]# keystone service-create --name glance --type image --description
 "Glance Image Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |       Glance Image Service       |
|      id     | 91b485c9d88d44299d0407f894bdfb0f |
|     name    |              glance              |
|     type    |              image               |
+-------------+----------------------------------+
[root@serverX ~]# keystone endpoint-create --service-id 1447f490e9784aa8890f9e39ddaa8d44
 --publicurl http://serverX.example.com:9292 --adminurl http://serverX.example.com:9292
 --internalurl http://serverX.example.com:9292
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  | http://serverX.example.com:9292  |
|      id     | 65ffbdac69ee427db4b777691a8cb4e8 |
| internalurl | http://serverX.example.com:9292  |
|  publicurl  | http://serverX.example.com:9292  |
|    region   |            regionOne             |
|  service_id | 91b485c9d88d44299d0407f894bdfb0f |
+-------------+----------------------------------+
```

If Glance was already configured on another machine, and you want to add another machine to provide redundant service, the configuration is a bit simpler. Start by installing the *openstack-glance* as shown previously. Copy the **/etc/glance/glance-api.conf** and **/etc/glance/glance-registry.conf** files from the previously installed Glance server to the new Glance server. Start and enable the services.

Once the services have started, either create new end points for the new Glance server or place a load balancer in front of the two Glance servers to balance the load. The load balancer can either be hardware (such as F5) or software (such as HAproxy). If you are using a load balancer, use a single set of end points for the Glance service using the front-end IP address of the load balancer.



Workshop

## Deploying the Glance image service

Follow along with the instructor as you perform the setup tasks required to install the Red Hat OpenStack software.

We are going to deploy the Glance image service without using the **packstack** command now. This will give us complete control over the installation, allowing us to, for example, install it on a separate machine.

Perform the following steps on **serverX.example.com** unless instructed otherwise.

☐ 1. Install the appropriate RPM package via yum. On **serverX.example.com**, install the *openstack-glance* package:

```
[root@serverX ~]# yum install -y openstack-glance
```

☐ 2. Back up files that you will be changing.

```
[root@serverX ~]# cp /etc/glance/glance-registry.conf /etc/glance/glance-
registry.conf.orig
[root@serverX ~]# cp /etc/glance/glance-api.conf /etc/glance/glance-api.conf.orig
```

☐ 3. Copy the **/usr/share/glance/glance-registry-dist.conf** file to **/etc/glance/glance-registry.conf** to configure some basic settings.

```
[root@serverX ~]# cp /usr/share/glance/glance-registry-dist.conf /etc/glance/
glance-registry.conf
```

☐ 4. To be able to authenticate with administrative privileges, source the **keystonerc_admin** file.

```
[root@serverX ~]# source /root/keystonerc_admin
[root@serverX ~(keystone_admin)]#
```

☐ 5. Initialize the database for use with Glance with a password of **redhat**. For a production deployment, pick a more difficult password.

```
[root@serverX ~(keystone_admin)]# openstack-db --init --service glance --password
 redhat --rootpw redhat
```

☐ 6. Create the **glance** user, then link the **glance** user and the **admin** role within the **services** tenant.

```
[root@serverX ~(keystone_admin)]# keystone user-create --name glance --pass redhat
[root@serverX ~(keystone_admin)]# keystone user-role-add --user glance --role
 admin --tenant services
```

☐ 7. Update the Glance configuration to use Keystone as the identity service.

```
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/glance/glance-
api.conf paste_deploy flavor keystone
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/glance/glance-
api.conf keystone_authtoken admin_tenant_name services
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/glance/glance-
api.conf keystone_authtoken admin_user glance
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/glance/glance-
api.conf keystone_authtoken admin_password redhat
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/glance/glance-
api.conf DEFAULT qpid_username qpidauth
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/glance/glance-
api.conf DEFAULT qpid_password redhat
```

```
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/glance/glance-
api.conf DEFAULT qpid_protocol ssl
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/glance/glance-
api.conf DEFAULT qpid_port 5671

[root@serverX ~(keystone_admin)]# openstack-config --set /etc/glance/glance-
registry.conf paste_deploy flavor keystone
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/glance/glance-
registry.conf keystone_authtoken admin_tenant_name services
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/glance/glance-
registry.conf keystone_authtoken admin_user glance
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/glance/glance-
registry.conf keystone_authtoken admin_password redhat
```

☐ 8. Start and enable the services. Check for any errors.

```
[root@serverX ~(keystone_admin)]# service openstack-glance-registry start
[root@serverX ~(keystone_admin)]# chkconfig openstack-glance-registry on
[root@serverX ~(keystone_admin)]# service openstack-glance-api start
[root@serverX ~(keystone_admin)]# chkconfig openstack-glance-api on
[root@serverX ~(keystone_admin)]# egrep 'ERROR|CRITICAL' /var/log/glance/*
```

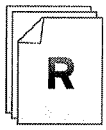
## Note

You will likely see some collie/sheepdog errors because you have not configured Sheepdog storage. You can safely ignore these errors.

☐ 9. Add the service and end points to the Keystone catalog.

```
[root@serverX ~(keystone_admin)]# keystone service-create --name glance --type
 image --description "OpenStack Image Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |     OpenStack Image Service      |
|      id     | abcdef1234567890abcdef1234567890 |
|     name    |              glance              |
|     type    |              image               |
+-------------+----------------------------------+

[root@serverX ~(keystone_admin)]# keystone endpoint-create --service-
id abcdef1234567890abcdef1234567890 --publicurl http://serverX.example.com:9292
 --adminurl http://serverX.example.com:9292 --internalurl http://
serverX.example.com:9292
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  | http://serverX.example.com:9292  |
|      id     | 6543210fedcba9876543210fedcba987 |
| internalurl | http://serverX.example.com:9292  |
|  publicurl  | http://serverX.example.com:9292  |
|    region   |            regionOne             |
|  service_id | abcdef1234567890abcdef1234567890 |
+-------------+----------------------------------+
```




## References

Red Hat OpenStack Installation and Configuration Guide

- Chapter 7. Installing the OpenStack image service

`http://www.f5.com/glossary/load-balancer/`

`http://haproxy.1wt.eu/`

# Using the glance command to upload a system image

The **glance** command can be used to manage images. Before adding a Linux image, it is important to prepare the image properly with **virt-sysprep** before uploading it to Glance. The command will remove SSH keys, persistent MAC addresses, and user accounts. By default, the format of the virtual machine disk is autodetected.

```
[root@serverX ~]# yum install -y libguestfs-tools
[root@serverX ~]# virt-sysprep --add IMAGEFILE
```

The following is an example command using **glance** to upload an image.

```
[root@serverX ~]# glance image-create --name "NAME" --is-public IS_PUBLIC --disk-format
 DISK_FORMAT --container-format CONTAINER_FORMAT --file IMAGEFILE
```

## Note

If unsure what image format has been used with a given system image, try to use the **qemu-img info** *IMAGENAME* command to identify it.

For local images, the **--file** switch is used. If we have the system image on a remote location, **--location** can be used to provide the URL directly without prior downloading the system image. The **--copy-from** option is like the **--location** option, but it will also populate the image in the Glance cache.

The **--is-public** switch is a Boolean switch, so it can be set to either true or false. If it is set to true, every user in Keystone can use that image inside their tenants. If set to false, only the uploading user is able to use it.

**Workshop**

## Using Glance to upload a system image

Follow along with the instructor as you perform the setup tasks required to install the Red Hat OpenStack software.

Add a system image to Glance using the command-line interface.

☐ 1. On **serverX.example.com**, apply the Keystone **admin** credentials.

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

☐ 2. In this case, the system image is already prepared, so upload it to Glance.

```
[root@serverX ~(keystone_admin)]# glance image-create --name test --is-
public True --disk-format qcow2 --container-format bare --copy-from http://
instructor.example.com/pub/materials/small.img
+------------------+--------------------------------------+
```

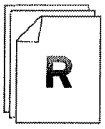
```
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | None                                 |
| container_format | bare                                 |
| created_at       | 2013-04-18T23:58:09                  |
| deleted          | False                                |
| deleted_at       | None                                 |
| disk_format      | qcow2                                |
| id               | fedcba09-8765-4321-fedc-ba0987654321 |
| is_public        | True                                 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | test                                 |
| owner            | 34567890abcdef1234567890abcdef12     |
| protected        | False                                |
| size             | 87363584                             |
| status           | queued                               |
| updated_at       | 2013-04-18T23:58:09                  |
+------------------+--------------------------------------+
```

☐ 3. Look at the list of Glance images available for later use:

```
[root@serverX ~(keystone_admin)]# glance image-list
+--------------------------------------+-------+-------------+------------------
+----------+--------+
| ID                                   | Name  | Disk Format | Container Format |
 Size     | Status |
+--------------------------------------+-------+-------------+------------------
+----------+--------+
| fedcba09-8765-4321-fedc-ba0987654321 | test  | qcow2       | bare             |
 87363584 | active |
+--------------------------------------+-------+-------------+------------------
+----------+--------+
```

☐ 4. To see more verbose details on a particular image, run the following command using a valid name or ID from the **`glance image-list`** output.

```
[root@serverX ~(keystone_admin)]# glance image-show test
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 210fedcba9876543210fedcba9876543     |
| container_format | bare                                 |
| created_at       | 2013-04-18T23:58:09                  |
| deleted          | False                                |
| disk_format      | qcow2                                |
| id               | fedcba09-8765-4321-fedc-ba0987654321 |
| is_public        | True                                 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | test                                 |
| owner            | 34567890abcdef1234567890abcdef12     |
| protected        | False                                |
| size             | 87363584                             |
| status           | active                               |
| updated_at       | 2013-04-18T23:58:09                  |
+------------------+--------------------------------------+
```

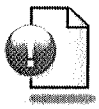



## References

Red Hat OpenStack Getting Started Guide

- Section 7.1. Uploading an image

# Personal Notes



# Summary

Deploying the Glance image service

- Deploy the Glance image service.

# CHAPTER 7

# IMPLEMENTING THE CINDER BLOCK STORAGE SERVICE

## Introduction

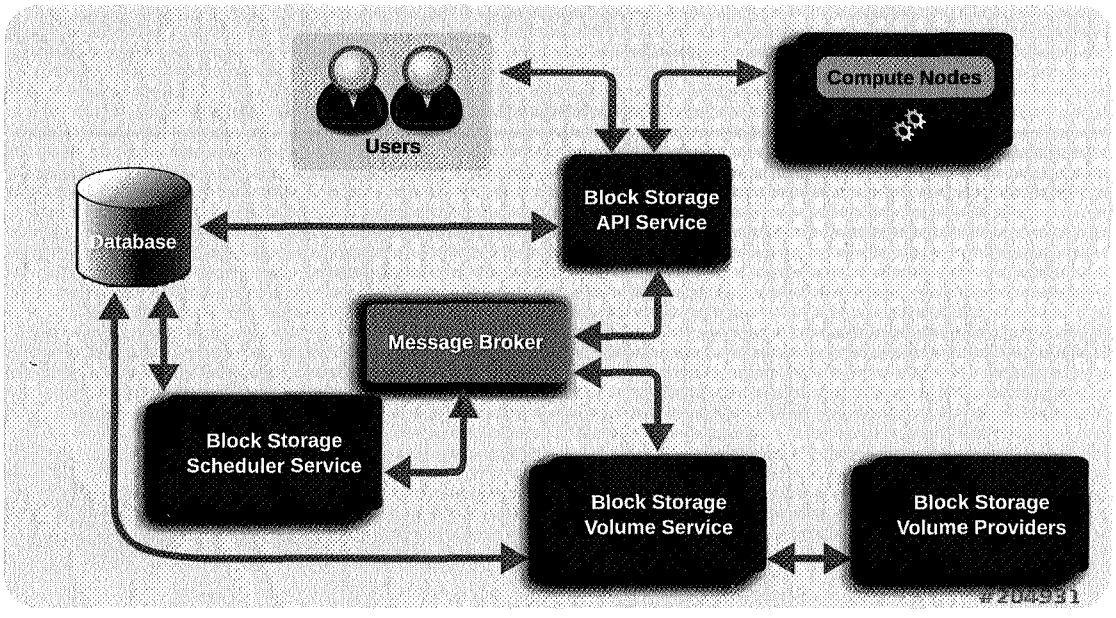
| Chapter details | |
|---|---|
| **Chapter goal** | Add an additional Cinder service to Red Hat OpenStack. |
| **Chapter sections** | • Installing the Cinder block storage service and managing volumes<br>• Adding a Red Hat storage volume to the Cinder block storage service |
| **Hands-on activities** | • Installing the Cinder block storage service and managing volumes<br>• Adding a Red Hat storage volume to Cinder |
| **Chapter test** | None |

# Installing the Cinder block storage service and managing volumes

Cinder is responsible for volume storage of virtual machines' data. These volumes can persistently store data and be attached to any instance. If a volume is attached to more than one instance at a time, make sure that the file system is read-only, or that the file system is cluster-aware.

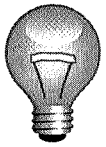
Block storage functionality is provided in OpenStack by three separate services, collectively referred to as the block storage service or Cinder. The three services are:

- The API service (**`openstack-cinder-api`**): The API service provides an HTTP end point 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 performed. 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.




To install the Cinder service, start by installing the *openstack-cinder* package. If the Cinder service has already been configured elsewhere, simply copy the **`/etc/cinder/cinder.conf`** and **`/etc/tgt/targets.conf`** files to the new machine and edit the **`my_ip`** and **`iscsi_ip_address`** options. Otherwise, you must configure **`/etc/cinder/cinder.conf`** with the Qpid, Keystone, Nova, and Glance settings for the Red Hat OpenStack deployment. The **`volume_group`** option in the **`/etc/cinder/cinder.conf`** file determines the name

of the volume group to use for Cinder. The default name is **cinder-volumes**, so edit the **volume_group** if the name of the volume group differs.



### Important

The **packstack** command will generate a loopback-mounted file for use with Cinder if it does not find a volume group named **cinder-volumes**. Using a loopback-mounted file for the Cinder volume service may have significant performance impact.

The classroom has been set up such that **server*X*.example.com** has a 5GB volume group named **cinder-volumes** and **desktop*X*.example.com** has a 20GB volume group named **cinder-volumes**. We will use these volume groups for Cinder.

When you use two or more Cinder services in Red Hat OpenStack, the cloud controller will use round-robin scheduling to determine the Cinder service to use.

Once the Cinder service is running, use the **cinder** command to manage the Cinder volumes. As with the other services, Cinder requires Keystone authentication to work properly, so source the credentials before working with the **cinder** command.

Demonstration

## Installing the Cinder block storage service and managing volumes

This demonstration will show you how to install and configure the Cinder service. We will use the Cinder block storage service to add a volume to an instance in a later chapter.

1. On **demo.example.com**, run the **lab-catchup-keystone** command to install the user, role, and tenant used in this demonstration.

```
[root@demo ~]# lab-catchup-keystone
```

2. Install the needed packages on **demo.example.com**.

```
[root@demo ~]# yum install -y openstack-cinder
```

3. Copy the **/usr/share/cinder/cinder-dist.conf** file to **/etc/cinder/cinder.conf** to set some default values.

```
[root@demo ~]# cp /etc/cinder/cinder.conf /etc/cinder/cinder.conf.orig
[root@demo ~]# cp /usr/share/cinder/cinder-dist.conf /etc/cinder/cinder.conf
```

4. To be able to authenticate with administrative privileges, source the **keystonerc_admin** file.

```
[root@demo ~]# source /root/keystonerc_admin
[root@demo ~(keystone_admin)]#
```

5. Initialize the database for use with Cinder with a password of **redhat**. For a production deployment, be sure to pick a more difficult password.

```
[root@demo ~(keystone_admin)]# openstack-db --init --service cinder --password redhat
 --rootpw redhat
```

6. Create the **cinder** user, then link the **cinder** user and the **admin** role within the **services** tenant.

```
[root@demo ~(keystone_admin)]# keystone user-create --name cinder --pass redhat
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |                                  |
| enabled  |               True               |
|    id    | 90abcdef1234567890abcdef12345678 |
|   name   |              cinder              |
| tenantId |                                  |
+----------+----------------------------------+
[root@demo ~(keystone_admin)]# keystone user-role-add --user cinder --role admin --
tenant services
```

7. Add the service to the Keystone catalog.

```
[root@demo ~(keystone_admin)]# keystone service-create --name=cinder --type=volume --
description="OpenStack Block Storage Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description | OpenStack Block Storage Service  |
|      id     | 9876543210fedcba9876543210fedcba |
|     name    |              cinder              |
|     type    |              volume              |
+-------------+----------------------------------+
```

8. Create the end points for the service.

```
[root@demo ~(keystone_admin)]# keystone endpoint-create --service-
id 9876543210fedcba9876543210fedcba --publicurl 'http://demo.example.com:8776/
v1/%(tenant_id)s' --adminurl 'http://demo.example.com:8776/v1/%(tenant_id)s' --
internalurl 'http://demo.example.com:8776/v1/%(tenant_id)s'
+-------------+-----------------------------------------+
|   Property  |                  Value                  |
+-------------+-----------------------------------------+
|   adminurl  | http://demo.example.com:8776/v1/%(tenant_id)s |
|      id     |       3210fedcba9876543210fedcba987654        |
| internalurl | http://demo.example.com:8776/v1/%(tenant_id)s |
|  publicurl  | http://demo.example.com:8776/v1/%(tenant_id)s |
|    region   |                regionOne                |
|  service_id |       9876543210fedcba9876543210fedcba        |
+-------------+-----------------------------------------+
```

9. Update the Cinder configuration to use Keystone as an identity service.

```
[root@demo ~(keystone_admin)]# openstack-config --set /etc/cinder/cinder.conf
 keystone_authtoken admin_tenant_name services
```

```
[root@demo ~(keystone_admin)]# openstack-config --set /etc/cinder/cinder.conf
 keystone_authtoken admin_user cinder
[root@demo ~(keystone_admin)]# openstack-config --set /etc/cinder/cinder.conf
 keystone_authtoken admin_password redhat
[root@demo ~(keystone_admin)]# openstack-config --set /etc/cinder/cinder.conf DEFAULT
 qpid_username qpidauth
[root@demo ~(keystone_admin)]# openstack-config --set /etc/cinder/cinder.conf DEFAULT
 qpid_password redhat
[root@demo ~(keystone_admin)]# openstack-config --set /etc/cinder/cinder.conf DEFAULT
 qpid_protocol ssl
[root@demo ~(keystone_admin)]# openstack-config --set /etc/cinder/cinder.conf DEFAULT
 qpid_port 5671
```

10. Start and enable the services. Check for any errors.

```
[root@demo ~(keystone_admin)]# service openstack-cinder-scheduler start
[root@demo ~(keystone_admin)]# service openstack-cinder-api start
[root@demo ~(keystone_admin)]# service openstack-cinder-volume start

[root@demo ~(keystone_admin)]# tail /var/log/cinder/*

[root@demo ~(keystone_admin)]# chkconfig openstack-cinder-scheduler on
[root@demo ~(keystone_admin)]# chkconfig openstack-cinder-api on
[root@demo ~(keystone_admin)]# chkconfig openstack-cinder-volume on
```

11. Edit the **/etc/tgt/targets.conf** file to include **include /etc/cinder/volumes/*** in order to configure iSCSI to include Cinder volumes.

```
echo 'include /etc/cinder/volumes/*' >> /etc/tgt/targets.conf
```

12. Start and enable the **tgtd** service.

```
[root@demo ~(keystone_admin)]# service tgtd start
[root@demo ~(keystone_admin)]# tail /var/log/messages
[root@demo ~(keystone_admin)]# chkconfig tgtd on
```

13. Check the status of all the OpenStack services. Enable any services marked "(disabled on boot)".

```
[root@demo ~(keystone_admin)]# openstack-status
== Glance services ==
openstack-glance-api:         active
openstack-glance-registry:    active
== Keystone service ==
openstack-keystone:           active
== Swift services ==
openstack-swift-proxy:        active
openstack-swift-account:      active
openstack-swift-container:    active
openstack-swift-object:       active
== Cinder services ==
openstack-cinder-api:         active
openstack-cinder-scheduler:   active
openstack-cinder-volume:      active
== Support services ==
mysqld:                       active
tgtd:                         active
```

```
qpidd:                       active
memcached:                   active
...
```

14. Create a new 2GB volume named **vol1** using the **myuser** credentials.

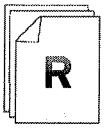
```
[root@demo ~(keystone_admin)]# source /root/keystonerc_myuser
[root@demo ~(keystone_myuser)]# cinder create --display-name vol1 2
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|     attachments     |                  []                  |
|  availability_zone  |                 nova                 |
|       bootable      |                false                 |
|      created_at     |      2013-04-09T14:22:54.228567      |
| display_description |                 None                 |
|     display_name    |                 vol1                 |
|          id         | cdef1234-5678-90ab-cdef-1234567890ab |
|       metadata      |                  {}                  |
|         size        |                  2                   |
|     snapshot_id     |                 None                 |
|     source_volid    |                 None                 |
|        status       |               creating               |
|     volume_type     |                 None                 |
+---------------------+--------------------------------------+
[root@demo ~(keystone_myuser)]# cinder list
+--------------------------------------+-----------+--------------+------
+-------------+----------+-------------+
|                  ID                  |   Status  | Display Name | Size | Volume
 Type | Bootable | Attached to |
+--------------------------------------+-----------+--------------+------
+-------------+----------+-------------+
| cdef1234-5678-90ab-cdef-1234567890ab |   available    |     vol1     |  2   |
 None    |  false   |             |
+--------------------------------------+-----------+--------------+------
+-------------+----------+-------------+
```

15. Use the normal LVM commands to view the volume group and logical volume information.

```
[root@demo ~(keystone_myuser)]# vgs
  VG             #PV #LV #SN Attr   VSize   VFree
  cinder-volumes   1   1   0 wz--n-  4.97g  2.97g
  ...
  [root@demo ~(keystone_myuser)]# lvs
  LV                                          VG             Attr      LSize  ...
  volume-cdef1234-5678-90ab-cdef-1234567890ab cinder-volumes -wi-ao---  2.00g
  ...
```

16. Delete the **vol1** volume.

```
[root@serverX ~(keystone_myuser)]# cinder delete vol1
```




## References

Red Hat OpenStack Installation and Configuration Guide
- Chapter 8. Installing OpenStack block storage

Red Hat OpenStack Getting Started Guide
- Section 7.3. Creating a volume

The **cinder** man page.

Performance Checklist

# Installing the Cinder block storage service and managing volumes

Install and configure the Cinder service. We will use the Cinder block storage service to add a volume to an instance in a later chapter.

☐ 1. Install the needed packages on **serverX.example.com**.

```
[root@serverX ~]# yum install -y openstack-cinder
```

☐ 2. Copy the **/usr/share/cinder/cinder-dist.conf** file to **/etc/cinder/cinder.conf** to set some default values.

```
[root@serverX ~]# cp /etc/cinder/cinder.conf /etc/cinder/cinder.conf.orig
[root@serverX ~]# cp /usr/share/cinder/cinder-dist.conf /etc/cinder/cinder.conf
```

☐ 3. In order to authenticate with administrative privileges, source the **keystonerc_admin** file.

```
[root@serverX ~]# source ~/keystonerc_admin
[root@serverX ~(keystone_admin)]#
```

☐ 4. Initialize the database for use with Cinder with a password of **redhat**. For a production deployment, be sure to pick a more difficult password.

```
[root@serverX ~(keystone_admin)]# openstack-db --init --service cinder --password
 redhat --rootpw redhat
```

☐ 5. Create the **cinder** user, then link the **cinder** user and the **admin** role within the **services** tenant.

```
[root@serverX ~(keystone_admin)]# keystone user-create --name cinder --pass redhat
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |                                  |
| enabled  |               True               |
|    id    | 90abcdef1234567890abcdef12345678 |
|   name   |              cinder              |
| tenantId |                                  |
+----------+----------------------------------+
[root@serverX ~(keystone_admin)]# keystone user-role-add --user cinder --role
 admin --tenant services
```

☐ 6. Add the service to the Keystone catalog.

```
[root@serverX ~(keystone_admin)]# keystone service-create --name cinder --type
 volume --description "OpenStack Block Storage Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
```

```
| description | OpenStack Block Storage Service  |
|    id       | 9876543210fedcba9876543210fedcba |
|    name     |              cinder              |
|    type     |              volume              |
+-------------+----------------------------------+
```

☐ 7. Create the end points for the service.

```
[root@serverX ~(keystone_admin)]# keystone endpoint-create --service-
id 9876543210fedcba9876543210fedcba --publicurl 'http://serverX.example.com:8776/
v1/%(tenant_id)s' --adminurl 'http://serverX.example.com:8776/v1/%(tenant_id)s' --
internalurl 'http://serverX.example.com:8776/v1/%(tenant_id)s'
+-------------+-----------------------------------------------+
|   Property  |                     Value                     |
+-------------+-----------------------------------------------+
|   adminurl  | http://serverX.example.com:8776/v1/%(tenant_id)s |
|      id     |         3210fedcba9876543210fedcba987654      |
| internalurl | http://serverX.example.com:8776/v1/%(tenant_id)s |
|  publicurl  | http://serverX.example.com:8776/v1/%(tenant_id)s |
|    region   |                   regionOne                   |
|  service_id |         9876543210fedcba9876543210fedcba      |
+-------------+-----------------------------------------------+
```

☐ 8. Update the Cinder configuration to use Keystone as an identity service.

```
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/cinder/cinder.conf
 keystone_authtoken admin_tenant_name services
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/cinder/cinder.conf
 keystone_authtoken admin_user cinder
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/cinder/cinder.conf
 keystone_authtoken admin_password redhat
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/cinder/cinder.conf
 DEFAULT verbose true
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/cinder/cinder.conf
 DEFAULT qpid_username qpidauth
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/cinder/cinder.conf
 DEFAULT qpid_password redhat
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/cinder/cinder.conf
 DEFAULT qpid_protocol ssl
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/cinder/cinder.conf
 DEFAULT qpid_port 5671
```

☐ 9. Start and enable the services. Check for any errors.

```
[root@serverX ~(keystone_admin)]# service openstack-cinder-scheduler start
[root@serverX ~(keystone_admin)]# service openstack-cinder-api start
[root@serverX ~(keystone_admin)]# service openstack-cinder-volume start

[root@serverX ~(keystone_admin)]# chkconfig openstack-cinder-scheduler on
[root@serverX ~(keystone_admin)]# chkconfig openstack-cinder-api on
[root@serverX ~(keystone_admin)]# chkconfig openstack-cinder-volume on

[root@serverX ~(keystone_admin)]# tail /var/log/cinder/*
```

☐ 10. Edit the **/etc/tgt/targets.conf** file to include **include /etc/cinder/volumes/*** in order to configure iSCSI to include Cinder volumes.

```
[root@serverX ~(keystone_admin)]# echo 'include /etc/cinder/volumes/*' >> /etc/
tgt/targets.conf
```

☐ 11. Start and enable the **tgtd** service.

```
[root@serverX ~(keystone_admin)]# service tgtd start
[root@serverX ~(keystone_admin)]# chkconfig tgtd on
[root@serverX ~(keystone_admin)]# tail /var/log/messages
```

☐ 12. Check the status of all the OpenStack services. Enable any services marked "(disabled on boot)".

```
[root@serverX ~(keystone_admin)]# openstack-status
== Glance services ==
openstack-glance-api:         active
openstack-glance-registry:    active
== Keystone service ==
openstack-keystone:           active
== Swift services ==
openstack-swift-proxy:        active
openstack-swift-account:      active
openstack-swift-container:    active
openstack-swift-object:       active
== Cinder services ==
openstack-cinder-api:         active
openstack-cinder-scheduler:   active
openstack-cinder-volume:      active
== Support services ==
mysqld:                       active
tgtd:                         active
qpidd:                        active
memcached:                    active
...
```

☐ 13. Create a new 2GB volume named **vol1** using the **myuser** credentials.

```
[root@serverX ~(keystone_admin)]# source /root/keystonerc_myuser
[root@serverX ~(keystone_myuser)]# cinder create --display-name vol1 2
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|     attachments     |                  []                  |
|  availability_zone  |                 nova                 |
|       bootable      |                false                 |
|      created_at     |      2013-04-09T14:22:54.228567      |
| display_description |                 None                 |
|     display_name    |                 vol1                 |
|          id         | cdef1234-5678-90ab-cdef-1234567890ab |
|       metadata      |                  {}                  |
|         size        |                  2                   |
|     snapshot_id     |                 None                 |
|     source_volid    |                 None                 |
|        status       |               creating               |
|     volume_type     |                 None                 |
+---------------------+--------------------------------------+
[root@serverX ~(keystone_myuser)]# cinder list
```

```
+--------------------------------------+-----------+--------------+------
+-------------+----------+-------------+
|                  ID                  |   Status  | Display Name | Size | Volume
 Type | Bootable | Attached to |
+--------------------------------------+-----------+--------------+------
+-------------+----------+-------------+
| cdef1234-5678-90ab-cdef-1234567890ab |   available    |     vol1     |  2   |
  None    |  false   |             |
+--------------------------------------+-----------+--------------+------
+-------------+----------+-------------+
```

☐ 14. Use the normal LVM commands to view the volume group and logical volume information.

```
[root@serverX ~(keystone_myuser)]# vgs
  VG             #PV #LV #SN Attr   VSize   VFree
  cinder-volumes   1   1   0 wz--n-  4.97g  2.97g
  vol0             1   2   0 wz--n- 29.97g    0
  [root@serverX ~(keystone_myuser)]# lvs
  LV                                          VG             Attr      LSize  ...
  volume-cdef1234-5678-90ab-cdef-1234567890ab cinder-volumes -wi-ao---  2.00g
  root                                        vol0           -wi-ao---  4.00g
  var                                         vol0           -wi-ao--- 25.97g
```

☐ 15. Clean up and delete the created volume.

```
[root@serverX ~(keystone_myuser)]# cinder delete vol1
```

# Adding a Red Hat storage volume to the Cinder block storage service

## Adding a Red Hat storage volume to the Cinder block storage service

The Cinder service features a driver for GlusterFS, which enables us to use a Red Hat storage volume as a block storage back end to Cinder. Since the Grizzly release of Red Hat OpenStack, the Cinder service supports multiple back ends, even with a single block storage node. For enabling that feature, we have to make a few adjustments to the **/etc/cinder/cinder.conf** configuration file.

Before configuring Cinder to use our GlusterFS volumes, install the *glusterfs-fuse* driver package on every Cinder host.

```
[root@serverX ~]# yum -y install glusterfs-fuse
```

In the following example, we enable two back ends in the DEFAULT section and add two new sections to the **/etc/cinder/cinder.conf** configuration file:

```
[DEFAULT]
(...)
enabled_backends=lvm1,glusterfs1
(...)

[lvm1]
volume_group=cinder-volumes
volume_driver=cinder.volume.drivers.lvm.LVMISCSIDriver
volume_backend_name=LVM_iSCSI

[glusterfs1]
volume_driver = cinder.volume.drivers.glusterfs.GlusterfsDriver
glusterfs_shares_config = /etc/cinder/shares.conf
volume_backend_name=GlusterFS
```

The above settings will enable LVM as it worked before by using our **cinder-volumes** LVM group, and in addition enable the **glusterfs** volumes specified in the **/etc/cinder/shares.conf** configuration file and specify **/var/lib/cinder/glusterfs** as the base directory where the **glusterfs** volumes are supposed to be found.

### Note

When there are enabled back end sections in the config file, they override the settings from the DEFAULT section in the **/etc/cinder/cinder.conf** configuration file.

The **/etc/cinder/shares.conf** file has to be created manually. It is a plain text file listing HOST:VOLUME on every line in the file:

```
[root@serverX ~]# cat /etc/cinder/shares.conf
desktopX.example.com:volumeX
```

```
serverX.example.com:volumeY
```

Since we have multiple back ends, we need to create volume types so we can specify to Cinder where it creates the volume. Start by sourcing the **/root/keystonerc_admin** file, creating the **lvm** type.

```
[root@serverX ~]# source /root/keystonerc_admin
[root@serverX ~(keystone_admin)]# cinder type-create lvm
[root@serverX ~(keystone_admin)]# cinder type-key lvm set volume_backend_name=LVM_iSCSI
```

Now, create the type for the Red Hat storage back end.

```
[root@serverX ~(keystone_admin)]# cinder type-create glusterfs
```

```
[root@serverX ~(keystone_admin)]# cinder type-key glusterfs set
 volume_backend_name=GlusterFS
```

## Note

It is also possible to enable NFS volumes by adding a section to enable the NFS driver. A potential NFS section has to be added to enabled_backends.

```
[nfs1]
nfs_shares_config=/etc/cinder/nfsshares.conf
volume_driver=cinder.volume.drivers.nfs.NfsDriver
volume_backend_name=NFS
```

Creating an NFS back end type is done with:

```
[root@serverX ~(keystone_admin)]# cinder type-create nfs
```

```
[root@serverX ~(keystone_admin)]# cinder type-key nfs set volume_backend_name=NFS
```

Restart the volume and schedule Cinder services.

```
[root@serverX ~]# for svc in scheduler volume; do service openstack-cinder-${svc}
 restart; done
```

To ensure everything worked, take a look at the Cinder volume log file.

```
[root@serverX ~]# tail -f /var/log/cinder/volume.log
```

Create a new Cinder volume and specify with **--volume-type** where we want to have it:

```
[root@serverX ~(keystone_user1)]# cinder create --display-name glustertest --volume-type
 glusterfs 1
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|     attachments     |                  []                  |
```

```
|  availability_zone  |                nova                |
|       bootable      |               false                |
|      created_at     |      2013-07-24T18:40:31.816005    |
| display_description |                None                |
|     display_name    |           glustertest              |
|          id         | 2f3db8fa-b731-4238-834a-caf5c8f63a7f |
|       metadata      |                {}                  |
|         size        |                1                   |
|     snapshot_id     |                None                |
|     source_volid    |                None                |
|        status       |              creating              |
|     volume_type     |             glusterfs              |
+---------------------+------------------------------------+
```

Verify that the volume was created properly and its status is set to available.

```
[root@serverX ~(keystone_user1)]# cinder list
+--------------------------------------+-----------+--------------+------+-------------
+----------+-------------+
|                  ID                  |   Status  | Display Name | Size | Volume Type |
 Bootable | Attached to |
+--------------------------------------+-----------+--------------+------+-------------
+----------+-------------+
| 2f3db8fa-b731-4238-834a-caf5c8f63a7f | available |     vol1     |  2   |  glusterfs  |
 false    |             |
+--------------------------------------+-----------+--------------+------+-------------
+----------+-------------+
```

Workshop

## Adding a Red Hat storage volume to Cinder

Follow along with the instructor as you perform the setup tasks required to add a Red Hat GlusterFS volume to the LVM volume that is already in use by Cinder.

☐ 1. On **serverX.example.com**, install the **glusterfs-fuse** driver.

```
[root@serverX ~]# yum install -y glusterfs-fuse
```

☐ 2. Make a backup of the config file so you can easily revert later.

```
[root@serverX ~]# cp /etc/cinder/cinder.conf /etc/cinder/cinder.conf.orig2
```

☐ 3. Adjust the config file to use the glusterfs back end and the lvm back end for Cinder.

```
[root@serverX ~]# openstack-config --set /etc/cinder/cinder.conf DEFAULT
 enabled_backends glusterfs,lvm
[root@serverX ~]# openstack-config --set /etc/cinder/cinder.conf lvm volume_group
 cinder-volumes
[root@serverX ~]# openstack-config --set /etc/cinder/cinder.conf lvm volume_driver
 cinder.volume.drivers.lvm.LVMISCSIDriver
[root@serverX ~]# openstack-config --set /etc/cinder/cinder.conf lvm
 volume_backend_name LVM
[root@serverX ~]# openstack-config --set /etc/cinder/cinder.conf glusterfs
 volume_driver cinder.volume.drivers.glusterfs.GlusterfsDriver
[root@serverX ~]# openstack-config --set /etc/cinder/cinder.conf glusterfs
 glusterfs_shares_config /etc/cinder/shares.conf
```

```
[root@serverX ~]# openstack-config --set /etc/cinder/cinder.conf glusterfs
 glusterfs_sparsed_volumes false
[root@serverX ~]# openstack-config --set /etc/cinder/cinder.conf glusterfs
 volume_backend_name RHS
```

☐ 4. Create the **/etc/cinder/shares.conf** file. Be advised that the *X* in volume*X* corresponds to your station number.

```
[root@serverX ~]# echo "rhs.example.com:volumeX" >> /etc/cinder/shares.conf
```

☐ 5. Restart the scheduler and volume Cinder services.

```
[root@serverX ~]# for svc in scheduler volume; do service openstack-cinder-${svc}
 restart; done
```

☐ 6. To ensure everything worked, take a look at the Cinder volume log file and view the **df** output.

```
[root@serverX ~]# tail /var/log/cinder/volume.log
[root@serverX ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/vol0-root
                       4128448   1376256   2542480  36% /
tmpfs                  1961440         0   1961440   0% /dev/shm
/dev/vda1               253871     55625    185139  24% /boot
/dev/mapper/vol0-var  26802428    360248  25080672   2% /var
rhs.example.com:volumeX
                       1300480     33152   1267328   3% /var/lib/cinder/
mnt/a8cd791a832c35664cb2e25ca9024293
```

☐ 7. To select where the volumes are created, create the Cinder volume types, starting with LVM.

```
[root@serverX ~]# source /root/keystonerc_admin
[root@serverX ~(keystone_admin)]# cinder type-create lvm
+--------------------------------------+------+
|                  ID                  | Name |
+--------------------------------------+------+
| 5ab65a3c-6ba1-4f8d-acda-81585864ac02 | lvm  |
+--------------------------------------+------+
[root@serverX ~(keystone_admin)]# cinder type-key 5ab65a3c-6ba1-4f8d-
acda-81585864ac02 set volume_backend_name=LVM
```

☐ 8. Create the type for the Red Hat storage back end.

```
[root@serverX ~(keystone_admin)]# cinder type-create glusterfs
+--------------------------------------+-----------+
|                  ID                  |    Name   |
+--------------------------------------+-----------+
| 0456b4c7-4c0f-4f17-bb8f-24a6659df5a0 | glusterfs |
+--------------------------------------+-----------+
[root@serverX ~(keystone_admin)]# cinder type-key 0456b4c7-4c0f-4f17-
bb8f-24a6659df5a0 set volume_backend_name=RHS
```

☐ 9. Verify that the types have been created correctly.

```
[root@serverX ~(keystone_admin)]# cinder type-list
+--------------------------------------+-----------+
|                  ID                  |    Name   |
+--------------------------------------+-----------+
| 0456b4c7-4c0f-4f17-bb8f-24a6659df5a0 | glusterfs |
| 5ab65a3c-6ba1-4f8d-acda-81585864ac02 |    lvm    |
+--------------------------------------+-----------+
```

☐ 10. Create a 1GB volume on our **lvm** back end. Continue to use the **admin** credentials for testing purposes.

```
[root@serverX ~(keystone_admin)]# cinder create --volume-type lvm --display-name
 vol2 1
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|     attachments     |                  []                  |
|  availability_zone  |                 nova                 |
|       bootable      |                false                 |
|      created_at     |      2013-07-24T18:39:57.844774      |
| display_description |                 None                 |
|     display_name    |                 vol2                 |
|          id         | a494f6e4-2c1e-46dc-b376-686993bb1696 |
|       metadata      |                  {}                  |
|         size        |                  1                   |
|     snapshot_id     |                 None                 |
|     source_volid    |                 None                 |
|        status       |                creating              |
|     volume_type     |                 lvm                  |
+---------------------+--------------------------------------+
```

☐ 11. Verify if the volume has been properly created.

```
[root@serverX ~(keystone_admin)]# cinder list
+--------------------------------------+-----------+--------------+------
+-------------+----------+-------------+
|                  ID                  |   Status  | Display Name | Size | Volume
 Type | Bootable | Attached to |
+--------------------------------------+-----------+--------------+------
+-------------+----------+-------------+
| a494f6e4-2c1e-46dc-b376-686993bb1696 | available |     vol2     |  1   |     lvm
     |  false   |             |
+--------------------------------------+-----------+--------------+------
+-------------+----------+-------------+
```

☐ 12. Create a volume on our **glusterfs** back end.

```
[root@serverX ~(keystone_admin)]# cinder create --volume-type glusterfs --display-
name vol3 1
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|     attachments     |                  []                  |
|  availability_zone  |                 nova                 |
|       bootable      |                false                 |
|      created_at     |      2013-07-24T18:40:20.547010      |
```

```
| display_description |                  None                  |
|   display_name      |                  vol3                  |
|          id         | ec14b0e4-dff7-4bd2-a9e8-67e10787faa1 |
|       metadata      |                   {}                   |
|         size        |                   1                    |
|     snapshot_id     |                  None                  |
|     source_volid    |                  None                  |
|        status       |                creating                |
|     volume_type     |               glusterfs                |
+---------------------+----------------------------------------+
```

☐ 13. Verify that the volume has been properly created. Wait until both volumes show a status of **available**.

```
[root@serverX ~(keystone_admin)]# cinder list
+--------------------------------------+-----------+--------------+------
+-------------+----------+-------------+
|                  ID                  |   Status  | Display Name | Size | Volume
 Type | Bootable | Attached to |
+--------------------------------------+-----------+--------------+------
+-------------+----------+-------------+
| a494f6e4-2c1e-46dc-b376-686993bb1696 | available |     vol2     |  1   |     lvm
     |  false   |             |
| ec14b0e4-dff7-4bd2-a9e8-67e10787faa1 | available |     vol3     |  1   |
 glusterfs  |  false   |             |
+--------------------------------------+-----------+--------------+------
+-------------+----------+-------------+
```

☐ 14. Create another volume on our **glusterfs** back end. This time, it should fail because we do not have another 1GB of free space on it.

```
[root@serverX ~(keystone_admin)]# cinder create --volume-type glusterfs --display-
name vol4 1
+---------------------+----------------------------------------+
|       Property      |                 Value                  |
+---------------------+----------------------------------------+
|     attachments     |                   []                   |
|  availability_zone  |                 nova                   |
|       bootable      |                false                   |
|      created_at     |      2013-07-24T18:40:31.816005        |
| display_description |                  None                  |
|     display_name    |                  vol4                  |
|          id         | b83db8fa-6f31-4238-89ba-c9f5c8f63a7f |
|       metadata      |                   {}                   |
|         size        |                   1                    |
|     snapshot_id     |                  None                  |
|     source_volid    |                  None                  |
|        status       |                creating                |
|     volume_type     |               glusterfs                |
+---------------------+----------------------------------------+
```

☐ 15. The error for **vol4** is expected since the **glusterfs** volume lacks enough free space.

```
[root@serverX ~(keystone_admin)]# cinder list
+--------------------------------------+-----------+--------------+------
+-------------+----------+-------------+
|                  ID                  |   Status  | Display Name | Size | Volume
 Type | Bootable | Attached to |
```

```
+--------------------------------------+-----------+--------------+------
+-------------+----------+-------------+
| a494f6e4-2c1e-46dc-b376-686993bb1696 | available |     vol2     |  1   |     lvm
     |  false   |              |
| b83db8fa-6f31-4238-89ba-c9f5c8f63a7f |   error   |     vol4     |  1   |
 glusterfs  |  false   |              |
| ec14b0e4-dff7-4bd2-a9e8-67e10787faa1 | available |     vol3     |  1   |
 glusterfs  |  false   |              |
+--------------------------------------+-----------+--------------+------
+-------------+----------+-------------+
```

☐ 16. Create another volume on the **lvm** back end, since it has enough free space.

```
[root@serverX ~(keystone_admin)]# cinder create --volume-type lvm --display-name
 vol5 1
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|     attachments     |                  []                  |
|  availability_zone  |                 nova                 |
|       bootable      |                false                 |
|      created_at     |      2013-07-24T18:40:43.741850      |
| display_description |                 None                 |
|     display_name    |                 vol5                 |
|          id         | 77d129b6-4c0b-4b14-828c-98ec8dd6be95 |
|       metadata      |                  {}                  |
|         size        |                  1                   |
|     snapshot_id     |                 None                 |
|     source_volid    |                 None                 |
|        status       |               creating               |
|     volume_type     |                 lvm                  |
+---------------------+--------------------------------------+
```

☐ 17. Verify that it succeeded.

```
[root@serverX ~(keystone_admin)]# cinder list
+--------------------------------------+-----------+--------------+------
+-------------+----------+-------------+
|                  ID                  |   Status  | Display Name | Size | Volume
 Type | Bootable | Attached to |
+--------------------------------------+-----------+--------------+------
+-------------+----------+-------------+
| 77d129b6-4c0b-4b14-828c-98ec8dd6be95 | available |     vol5     |  1   |     lvm
     |  false   |              |
| a494f6e4-2c1e-46dc-b376-686993bb1696 | available |     vol2     |  1   |     lvm
     |  false   |              |
| b83db8fa-6f31-4238-89ba-c9f5c8f63a7f |   error   |     vol4     |  1   |
 glusterfs  |  false   |              |
| ec14b0e4-dff7-4bd2-a9e8-67e10787faa1 | available |     vol3     |  1   |
 glusterfs  |  false   |              |
+--------------------------------------+-----------+--------------+------
+-------------+----------+-------------+
```

☐ 18. Remove all the newly created volumes with volume types **lvm** and **glusterfs**.

```
[root@serverX ~(keystone_admin)]# cinder delete vol2
[root@serverX ~(keystone_admin)]# cinder delete vol3
[root@serverX ~(keystone_admin)]# cinder delete vol4
```

```
[root@serverX ~(keystone_admin)]# cinder delete vol5
```

## Note

This may take some time to complete because Cinder will fill the volume with zeros when the volume is deleted.

Check that all of the volumes were successfully deleted.

```
[root@serverX ~(keystone_admin)]# cinder list
```

☐ 19. Check for the created types and remove them.

```
[root@serverX ~(keystone_admin)]# cinder type-list
+--------------------------------------+-----------+
|                  ID                  |    Name   |
+--------------------------------------+-----------+
| 0456b4c7-4c0f-4f17-bb8f-24a6659df5a0 | glusterfs |
| 5ab65a3c-6ba1-4f8d-acda-81585864ac02 |    lvm    |
+--------------------------------------+-----------+
```

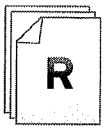
```
[root@serverX ~(keystone_admin)]# cinder type-delete 0456b4c7-4c0f-4f17-
bb8f-24a6659df5a0
[root@serverX ~(keystone_admin)]# cinder type-delete 5ab65a3c-6ba1-4f8d-
acda-81585864ac02
```

☐ 20. Revert Cinder to the original state.

```
[root@serverX ~(keystone_admin)]# cp /etc/cinder/cinder.conf.orig2 /etc/cinder/
cinder.conf
[root@serverX ~(keystone_admin)]# chown cinder:cinder /etc/cinder/cinder.conf
[root@serverX ~(keystone_admin)]# chmod 600 /etc/cinder/cinder.conf
[root@serverX ~(keystone_admin)]# restorecon -v /etc/cinder/cinder.conf
```

☐ 21. Restart the affected Cinder services.

```
[root@serverX ~(keystone_admin)]# for svc in scheduler volume; do service
 openstack-cinder-${svc} restart; done
```




## References

Red Hat OpenStack Installation and Configuration Guide

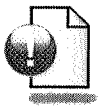
- Section 8.4.4. Configuring for Red Hat storage back end

GlusterFS Cinder

- `http://www.gluster.org/community/documentation/index.php/GlusterFS_Cinder`

The **`cinder`** man page.

# Personal Notes



# Summary

Installing the Cinder block storage service and managing volumes

- Add an additional Cinder service to Red Hat OpenStack.
- Manage a Cinder volume.

Adding a Red Hat storage volume to the Cinder block storage service

- In this section we will explore the possibility of adding Red Hat storage volumes as a storage back end to Cinder.

# CHAPTER 8

# IMPLEMENTING THE OPENSTACK NETWORKING SERVICE

## Introduction

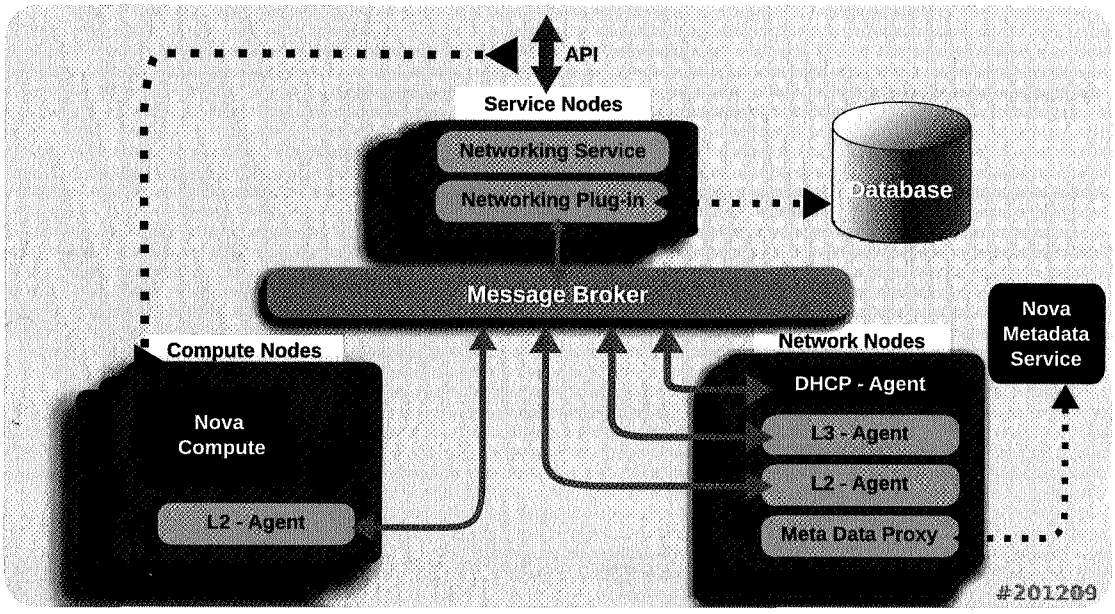
| Chapter details | |
|---|---|
| **Chapter goal** | Configure OpenStack networking |
| **Chapter sections** | • Installing OpenStack networking<br>• Configuring OpenStack networking |
| **Hands-on activities** | • Installing OpenStack networking<br>• Configuring OpenStack networking |
| **Chapter test** | None |

# Installing OpenStack networking

OpenStack networking is a virtual network service that aims to provide a rich interface for defining network connectivity and addressing in the OpenStack environment. It does this while providing plug-ins that give administrators the flexibility they require to leverage different networking technologies and strategies. In Red Hat OpenStack 3.0 (Grizzly), the OpenStack networking service is the default networking option. The Nova networking service is offered as an alternative, but will be deprecated in a future release.

You can configure rich network topologies by creating and configuring networks and subnets, and then instructing other OpenStack services, such as 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.

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




**Workshop**

## Installing OpenStack networking

Follow along with your instructor as you complete this workshop together.

OpenStack networking (Neutron née Quantum) was made a core project in the Folsom release. In this lab, you will set up OpenStack networking.

☐ 1. On **serverX**, create a service entry for OpenStack networking in Keystone.

```
[root@serverX ~]# source /root/keystonerc_admin
[root@serverX ~(keystone_admin)]# keystone service-create --name neutron --type
 network --description 'OpenStack Networking Service'
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |   OpenStack Networking Service   |
|      id     | dcba9876543210fedcba9876543210fe |
|     name    |              neutron             |
|     type    |              network             |
+-------------+----------------------------------+
```

☐ 2. Create an OpenStack networking end point in **keystone** using the ID from the previous output.

```
[root@serverX ~(keystone_admin)]# keystone endpoint-create --service-
id dcba9876543210fedcba9876543210fe --publicurl http://serverX.example.com:9696
 --adminurl http://serverX.example.com:9696 --internalurl http://
serverX.example.com:9696
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  | http://serverX.example.com:9696  |
|      id     | dad1234567890dad1234567890dad123 |
| internalurl | http://serverX.example.com:9696  |
|  publicurl  | http://serverX.example.com:9696  |
|    region   |            regionOne             |
|  service_id | dcba9876543210fedcba9876543210fe |
+-------------+----------------------------------+
[root@serverX ~(keystone_admin)]# keystone catalog
...
Service: network
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminURL  |  http://serverX.example.com:9696 |
|      id     | dad1234567890dad1234567890dad123 |
| internalURL |  http://serverX.example.com:9696 |
|  publicURL  |  http://serverX.example.com:9696 |
|    region   |            regionOne             |
+-------------+----------------------------------+
...
```

☐ 3. Create an OpenStack networking service user named **neutron** using the password **redhat**.

```
[root@serverX ~(keystone_admin)]# keystone user-create --name neutron --pass
 redhat
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |                                  |
| enabled  |               True               |
|    id    | cab1234567890cab1234567890cab123 |
|   name   |             neutron              |
| tenantId |                                  |
+----------+----------------------------------+
```

☐ 4. Link the **neutron** user and the **admin** role within the **services** tenant.

```
[root@serverX ~(keystone_admin)]# keystone user-role-add --user neutron --role
 admin --tenant services
```

☐ 5. Verify the user role was added.

```
[root@serverX ~(keystone_admin)]# keystone user-role-list
+----------------------------------+-------+----------------------------------
+----------------------------------+
|                id                |  name |             user_id              |
        tenant_id             |
+----------------------------------+-------+----------------------------------
+----------------------------------+
| fad9876543210fad9876543210fad987 | admin | 34567890abcdef1234567890abcdef12
 | 4567890abcdef1234567890abcdef123 |
+----------------------------------+-------+----------------------------------
+----------------------------------+
```

The previous output does not look correct because it displayed the information for the **admin** user identity, which was sourced at the beginning of this demonstration. The following command specifies the user, password, and tenant on the command line.

```
[root@serverX ~(keystone_admin)]# keystone --os-username neutron --os-password
 redhat --os-tenant-name services user-role-list
+----------------------------------+-------+----------------------------------
+----------------------------------+
|                id                |  name |             user_id              |
        tenant_id             |
+----------------------------------+-------+----------------------------------
+----------------------------------+
| fad9876543210fad9876543210fad987 | admin | cab1234567890cab1234567890cab123
 | bad9876543210bad9876543210bad987 |
+----------------------------------+-------+----------------------------------
+----------------------------------+
```

Alternately, you could use the **--user** and **--tenant** options to the **user-role-list** subcommand.

☐ 6. Install the OpenStack networking service and the Open vSwitch plug-in on **serverX**.

```
[root@serverX ~(keystone_admin)]# yum -y install openstack-neutron openstack-
neutron-openvswitch
```

☐ 7. OpenStack networking must connect to a service that provides AMQP. Ensure the **qpidd** service is running on **serverX**.

```
[root@serverX ~(keystone_admin)]# service qpidd status
 qpidd (pid  27406) is running...
```

☐ 8. Update the OpenStack networking service to use the Keystone information created previously.

```
[root@serverX ~(keystone_admin)]# cp /etc/neutron/neutron.conf /etc/neutron/
neutron.conf.orig
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/neutron/neutron.conf
 DEFAULT rpc_backend quantum.openstack.common.rpc.impl_qpid
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/neutron/neutron.conf
 DEFAULT qpid_hostname 192.168.0.X+100
```

☐ 9. Configure OpenStack networking to use Qpid and Keystone using the values configured previously.

```
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/neutron/neutron.conf
 DEFAULT qpid_username qpidauth
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/neutron/neutron.conf
 DEFAULT qpid_password redhat
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/neutron/neutron.conf
 DEFAULT qpid_protocol ssl
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/neutron/neutron.conf
 DEFAULT qpid_port 5671
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/neutron/neutron.conf
 keystone_authtoken admin_tenant_name services
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/neutron/neutron.conf
 keystone_authtoken admin_user neutron
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/neutron/neutron.conf
 keystone_authtoken admin_password redhat
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/neutron/neutron.conf
 agent root_helper "sudo neutron-rootwrap /etc/neutron/rootwrap.conf"
```

☐ 10. Create a **/root/keystonerc_neutron** Keystone file for the **neutron** user with the following information:

```
export OS_USERNAME=neutron
export OS_TENANT_NAME=services
export OS_PASSWORD=redhat
export OS_AUTH_URL=http://192.168.0.X+100:35357/v2.0/
export PS1='[\u@\h \W(keystone_neutron)]\$ '
```

☐ 11. Source the **/root/keystonerc_neutron** file. It is important to use these credentials throughout this lab because the setup scripts will use the username, password, and tenant exported in the shell.

```
[root@serverX ~(keystone_admin)]# source /root/keystonerc_neutron
[root@serverX ~(keystone_neutron)]#
```

☐ 12. The OpenStack networking setup scripts expect the host name of the local machine to be a resolvable network address. Verify that the **serverX** host name is resolvable.

```
[root@serverX ~(keystone_neutron)]# hostname
serverX.example.com
[root@serverX ~(keystone_neutron)]# host serverX.example.com
serverX.example.com has address 192.168.0.X+100
```

☐ 13. The OpenStack networking setup scripts will make several changes to the **/etc/nova/nova.conf** file. However, we will not cover Nova until *Chapter 9: Implementing the Nova*

*compute and Nova controller services*. Install the *openstack-nova-common* package, which provides the **/etc/nova/nova.conf** file, and we will discuss this configuration file later.

```
[root@serverX ~(keystone_neutron)]# yum install -y openstack-nova-common
```

☐ 14. Run the OpenStack networking setup script using the **openvswitch** plug-in.

## Note

The MySQL password should be **redhat**, as configured in *Chapter 4: Implementing the Keystone identity service*.

```
[root@serverX ~(keystone_neutron)]# neutron-server-setup --yes --rootpw redhat --
plugin openvswitch
Neutron plugin: openvswitch
Plugin: openvswitch => Database: ovs_neutron
Verified connectivity to MySQL.
Configuration updates complete!
[root@serverX ~(keystone_neutron)]# neutron-db-manage --config-file /usr/share/
neutron/neutron-dist.conf --config-file /etc/neutron/neutron.conf --config-file /
etc/neutron/plugin.ini stamp head
```

☐ 15. Start and enable the **neutron-server** service after checking for errors.

```
[root@serverX ~(keystone_neutron)]# service neutron-server start
[root@serverX ~(keystone_neutron)]# egrep 'ERROR|CRITICAL' /var/log/neutron/
server.log
[root@serverX ~(keystone_neutron)]# chkconfig neutron-server on
[root@serverX ~(keystone_neutron)]# openstack-status
...
== Neutron services ==
neutron-server:                 active
neutron-dhcp-agent:             inactive (disabled on boot)
neutron-l3-agent:               inactive (disabled on boot)
neutron-linuxbridge-agent:      dead (disabled on boot)
neutron-openvswitch-agent:      inactive (disabled on boot)
openvswitch:                    dead
...
```

☐ 16. Configure Open vSwitch and start the necessary services.

```
[root@serverX ~(keystone_neutron)]# neutron-node-setup --plugin openvswitch --
qhost 192.168.0.X+100
Neutron plugin: openvswitch
Would you like to update the nova configuration files? (y/n):
y
Configuration updates complete!
[root@serverX ~(keystone_neutron)]# service openvswitch start
[root@serverX ~(keystone_neutron)]# egrep 'ERROR|CRITICAL' /var/log/openvswitch/*
[root@serverX ~(keystone_neutron)]# chkconfig openvswitch on
```

☐ 17. Create an Open vSwitch bridge named **br-int**. This is the integration bridge that will be used as a patch panel to assign interface(s) to an instance.

```
[root@serverX ~(keystone_neutron)]# ovs-vsctl add-br br-int
[root@serverX ~(keystone_neutron)]# ovs-vsctl show
12345678-90ab-cdef-fedc-ba0987654321
    Bridge br-int
        Port br-int
            Interface br-int
                type: internal
    ovs_version: "1.9.0"
```

☐ 18. Configure the Open vSwitch plug-in to use **br-int** as the integration bridge. Start and enable the **neutron-openvswitch-agent** service.

```
[root@serverX ~(keystone_neutron)]# cp /etc/neutron/plugins/
openvswitch/ovs_neutron_plugin.ini /etc/neutron/plugins/openvswitch/
ovs_neutron_plugin.ini.orig
[root@serverX ~(keystone_neutron)]# openstack-config --set /etc/neutron/plugins/
openvswitch/ovs_neutron_plugin.ini OVS integration_bridge br-int
[root@serverX ~(keystone_neutron)]# service neutron-openvswitch-agent start
[root@serverX ~(keystone_neutron)]# egrep 'ERROR|CRITICAL' /var/log/neutron/
openvswitch-agent.log
[root@serverX ~(keystone_neutron)]# chkconfig neutron-openvswitch-agent on
```

☐ 19. Enable the **neutron-ovs-cleanup** service. When started at boot time, this service ensures that the OpenStack networking agents maintain full control over the creation and management of tap devices.

```
[root@serverX ~(keystone_neutron)]# chkconfig neutron-ovs-cleanup on
```

☐ 20. Configure and enable the OpenStack networking DHCP agent (**neutron-dhcp-agent**) on **serverX**.

```
[root@serverX ~(keystone_neutron)]# neutron-dhcp-setup --plugin openvswitch --
qhost 192.168.0.X+100
Neutron plugin: openvswitch
Configuration updates complete!
[root@serverX ~(keystone_neutron)]# service neutron-dhcp-agent start
[root@serverX ~(keystone_neutron)]# egrep 'ERROR|CRITICAL' /var/log/neutron/dhcp-
agent.log
[root@serverX ~(keystone_neutron)]# chkconfig neutron-dhcp-agent on
```

☐ 21. Create the **br-ex** bridge that will be used for external network traffic in Open vSwitch.

```
[root@serverX ~(keystone_neutron)]# ovs-vsctl add-br br-ex
```

☐ 22. Before attaching **eth0** to the **br-ex** bridge, configure the **br-ex** network device configuration file.

```
[root@serverX ~(keystone_neutron)]# cp /etc/sysconfig/network-scripts/ifcfg-eth0 /
root/
```

```
[root@serverX ~(keystone_neutron)]# cp /etc/sysconfig/network-scripts/ifcfg-eth0 /
etc/sysconfig/network-scripts/ifcfg-br-ex
```

If you are in a physical classroom, remove everything but the **DEVICE**, **HWADDR**, and **ONBOOT** settings from the **/etc/sysconfig/network-scripts/ifcfg-eth0** file, so that it looks like the following:

```
DEVICE=eth0
HWADDR=52:54:00:00:00:XX
ONBOOT=yes
```

If you are in a virtual classroom, remove everything but the **DEVICE** and **ONBOOT** settings from the **/etc/sysconfig/network-scripts/ifcfg-eth0** file, so that it looks like the following:

```
DEVICE=eth0
ONBOOT=yes
```

In the **/etc/sysconfig/network-scripts/ifcfg-br-ex** file, change the device name to **br-ex** and remove the **HWADDR** line if present. Make sure the **/etc/sysconfig/network-scripts/ifcfg-br-ex** file contains the following:

```
DEVICE=br-ex
IPADDR=192.168.0.X+100
PREFIX=24
GATEWAY=192.168.0.254
DNS1=192.168.0.254
SEARCH1=example.com
ONBOOT=yes
```

☐ 23. Once you have verified the network files contain the correct information, add the **eth0** network device to the **br-ex** bridge and restart the network.

```
[root@serverX ~(keystone_neutron)]# ovs-vsctl add-port br-ex eth0 ; service
 network restart
[root@serverX ~(keystone_neutron)]# ovs-vsctl show
12345678-90ab-cdef-fedc-ba0987654321
    Bridge br-ex
        Port "eth0"
            Interface "eth0"
        Port br-ex
            Interface br-ex
                type: internal
    Bridge br-int
        Port br-int
            Interface br-int
                type: internal
    ovs_version: "1.9.0"
```

☐ 24. Run the **neutron-l3-setup** script to configure the OpenStack networking L3 agent (**neutron-l3-agent**).

```
[root@serverX ~(keystone_neutron)]# neutron-l3-setup --plugin openvswitch --qhost
 192.168.0.X+100
Neutron plugin: openvswitch
```

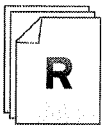
```
Configuration updates complete!
```

☐ 25. Start and enable the **neutron-l3-agent** service.

```
[root@serverX ~(keystone_neutron)]# service neutron-l3-agent start
[root@serverX ~(keystone_neutron)]# egrep 'ERROR|CRITICAL' /var/log/neutron/l3-
agent.log
[root@serverX ~(keystone_neutron)]# chkconfig neutron-l3-agent on
```

☐ 26. The OpenStack networking services are now running, so verify the status of the services:

```
[root@serverX ~(keystone_Neutron)]# openstack-status
...
== neutron services ==
neutron-server:                 active
neutron-dhcp-agent:             active
neutron-l3-agent:               active
neutron-linuxbridge-agent:      dead (disabled on boot)
neutron-openvswitch-agent:      active
openvswitch:                    active
...
```




## References

Red Hat OpenStack Installation and Configuration Guide

- Chapter 9. Installing the OpenStack networking service

# Configuring OpenStack networking

In order for the instances to use the network, you must configure OpenStack networking. First, start by creating a router, then create a network and subnet. If the subnet is a private network (only accessible to other instances), add an interface to the router (**neutron router-interface-add...**). One network can be an external network that can pass traffic out from the router. Set the gateway for the router to this external network. The external network can be used to allocate and assign floating IP addresses for the instances.

The *kernel* provided by the Red Hat OpenStack repository includes network namespaces. Network namespaces cover network access with a layer of abstraction, by virtualizing access to the network resources such as ports and interfaces. The **ifconfig** and **ip** commands will not show IP addresses within these namespaces. The *iproute* command has also been updated to allow queries to these namespaces. To access a namespace, use the **ip netns** command using the UUID of the router. The following example shows the steps to find the router UUID, list the IP address in the namespace, and ping an IP address in the namespace:

```
[root@demo ~]# source keystonerc_user1
[root@demo ~(keystone_user1]# neutron router-list
+--------------------------------------+---------
+-------------------------------------------------------+
| id                                   | name    | external_gateway_info
                |
+--------------------------------------+---------
+-------------------------------------------------------+
| 567890ab-cdef-1234-567890abcdef1234 | router1 | {"network_id": "98765432-10fe-
dcba-9876-543210fedcba"} |
+--------------------------------------+---------
+-------------------------------------------------------+
[root@demo ~(keystone_user1]# ip netns exec qrouter-567890ab-cdef-1234-567890abcdef1234
 ip a
...
[root@demo ~(keystone_user1]# ip netns exec qrouter-567890ab-cdef-1234-567890abcdef1234
 ping 172.24.X.1
...
```

The router ID can also be found in the **/var/run/netns/** directory (**qrouter-*567890ab-cdef-1234-567890abcdef1234***).

Workshop

## Configuring OpenStack networking

Now that you have installed OpenStack networking, you need to configure the networks, subnets, and routers needed to pass network traffic to and from the instances. This workshop will follow the steps needed to configure OpenStack networking for use with your instances.

☐ 1. Configure the network settings. Create the private network solely for the **myopenstack** tenant; start by sourcing the **/root/keystonerc_myuser** file and creating a router named **router1**.

```
[root@serverX ~(keystone_neutron)]# source /root/keystonerc_myuser
[root@serverX ~(keystone_myuser)]# neutron router-create router1
Created a new router:
+-----------------------+--------------------------------------+
| Field                 | Value                                |
```

```
+-----------------------+--------------------------------------+
| admin_state_up        | True                                 |
| external_gateway_info |                                      |
| id                    | 567890ab-cdef-1234-567890abcdef1234  |
| name                  | router1                              |
| status                | ACTIVE                               |
| tenant_id             | 890abcdef1234567890abcdef1234567     |
+-----------------------+--------------------------------------+
```

☐ 2. Create a new network named **private**.

```
[root@serverX ~(keystone_myuser)]# neutron net-create private
Created a new network:
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 0abcdef1-2345-6789-0abc-def123456789 |
| name                      | private                              |
| router:external           | False                                |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | 890abcdef1234567890abcdef1234567     |
+---------------------------+--------------------------------------+
```

☐ 3. Create a new subnet named **subpriv** in **private** using the **192.168.32.0/24** range.

```
[root@serverX ~(keystone_myuser)]# neutron subnet-create --name subpriv private
 192.168.32.0/24
Created a new subnet:
+-------------------+------------------------------------------------------+
| Field             | Value                                                |
+-------------------+------------------------------------------------------+
| allocation_pools  | {"start": "192.168.32.2", "end": "192.168.32.254"} |
| cidr              | 192.168.32.0/24                                      |
| dns_nameservers   |                                                      |
| enable_dhcp       | True                                                 |
| gateway_ip        | 192.168.32.1                                         |
| host_routes       |                                                      |
| id                | bcdef123-4567-890a-bcde-f1234567890a                 |
| ip_version        | 4                                                    |
| name              | subpriv                                              |
| network_id        | 0abcdef1-2345-6789-0abc-def123456789                 |
| tenant_id         | 890abcdef1234567890abcdef1234567                     |
+-------------------+------------------------------------------------------+
```

☐ 4. Add an interface to the router for the **192.168.32.0/24** subnet created previously.

```
[root@serverX ~(keystone_myuser)]# neutron router-interface-add router1 subpriv
Added interface to router router1
```

☐ 5. View the interface information.

```
[root@serverX ~(keystone_myuser)]# neutron port-list
+--------------------------------------+------+-------------------
+-------------------------------------------------------------------------------------
+
```

```
| id                                   | name | mac_address       | fixed_ips
                                                                    |
+--------------------------------------+------+-------------------
+-------------------------------------------------------------------------------
+
| f1234567-890a-bcde-f123-4567890abcde |      | fa:16:3e:83:46:d9 | {"subnet_id":
 "bcdef123-4567-890a-bcde-f1234567890a", "ip_address": "192.168.32.1"} |
+--------------------------------------+------+-------------------
+-------------------------------------------------------------------------------
+
```

☐ 6. Create a public network for floating IP addresses using the **172.24.*X*.0/24** range in the **services** tenant. This must be done as the **admin** role, so make sure you source the **keystonerc_neutron** file.

```
[root@serverX ~(keystone_myuser)]# source /root/keystonerc_admin
[root@serverX ~(keystone_admin)]# neutron net-create --tenant-id services public
 --router:external=True
Created a new network:
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | ef123456-7890-abcd-ef12-34567890abcd |
| name                      | public                               |
| provider:network_type     | local                                |
| provider:physical_network |                                      |
| provider:segmentation_id  |                                      |
| router:external           | True                                 |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | services                             |
+---------------------------+--------------------------------------+
```

☐ 7. Create a subnet named **subpub** in the **public** network using the **172.24.*X*.0/24**. Allow IP addresses in the 172.24.*X*.1-172.24.*X*.100 range to be allocated. Set the gateway to **172.24.*X*.254**.

```
[root@serverX ~(keystone_admin)]# neutron subnet-create --tenant-id services
 --allocation-pool start=172.24.X.1,end=172.24.X.100 --gateway 172.24.X.254 --
disable-dhcp --name subpub public 172.24.X.0/24
Created a new subnet:
+-------------------+--------------------------------------------------+
| Field             | Value                                            |
+-------------------+--------------------------------------------------+
| allocation_pools  | {"start": "172.24.X.1", "end": "172.24.X.100"}   |
| cidr              | 172.24.X.0/24                                    |
| dns_nameservers   |                                                  |
| enable_dhcp       | False                                            |
| gateway_ip        | 172.24.X.254                                     |
| host_routes       |                                                  |
| id                | 00000000-1111-2222-aaaa-bbbbbbbbbbbb             |
| ip_version        | 4                                                |
| name              | subpub                                           |
| network_id        | ef123456-7890-abcd-ef12-34567890abcd             |
| tenant_id         | services                                         |
+-------------------+--------------------------------------------------+
```

☐ 8. Set the **public** network as the gateway for the **router1** router.

```
[root@serverX ~(keystone_admin)]# neutron router-gateway-set router1 public
Set gateway for router router1
```

Notice that the router was assigned the 172.24.*X*.1 IP address:

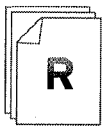
```
[root@serverX ~(keystone_admin)]# neutron port-list
+--------------------------------------+------+-------------------
+-------------------------------------------------------------------------------------
+
| id                                   | name | mac_address       | fixed_ips
                                                                                      |
+--------------------------------------+------+-------------------
+-------------------------------------------------------------------------------------
+
| f1234567-890a-bcde-f123-4567890abcde |      | fa:16:3e:83:46:d9 | {"subnet_id":
 "bcdef123-4567-890a-bcde-f1234567890a", "ip_address": "192.168.32.1"} |
| aaaaaaa-bbbb-cccc-dddd-eeeeeeffffff |      | fa:16:3e:e2:40:66 | {"subnet_id":
 "00000000-1111-2222-aaaa-bbbbbbbbbbbb", "ip_address": "172.24.X.1"}   |
+--------------------------------------+------+-------------------
+-------------------------------------------------------------------------------------
+
```

☐ 9. Now that the external network has been configured, we can use the **myuser** authentication. Source the **/root/keystonerc_myuser** file and create a floating IP address.

```
[root@serverX ~(keystone_admin)]# source /root/keystonerc_myuser
[root@serverX ~(keystone_myuser)]# neutron floatingip-create public
Created a new floatingip:
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| fixed_ip_address    |                                      |
| floating_ip_address | 172.24.X.2                           |
| floating_network_id | ef123456-7890-abcd-ef12-34567890abcd |
| id                  | bedbed00-bedb-ed00-bedb-ed00bedbed00 |
| port_id             |                                      |
| router_id           |                                      |
| tenant_id           | 890abcdef1234567890abcdef1234567     |
+---------------------+--------------------------------------+
```

☐ 10. View the floating IP address.

```
[root@serverX ~(keystone_myuser)]# neutron floatingip-list
+--------------------------------------+------------------+---------------------
+---------+
| id                                   | fixed_ip_address | floating_ip_address |
 port_id |
+--------------------------------------+------------------+---------------------
+---------+
| bedbed00-bedb-ed00-bedb-ed00bedbed00 |                  | 172.24.X.2          |
         |
+--------------------------------------+------------------+---------------------
+---------+
```




## References

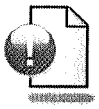
Red Hat OpenStack Installation and Configuration Guide
- Section 9.9. Validating the OpenStack networking installation

Red Hat OpenStack Getting Started Guide
- Section 7.8. Working with OpenStack networking

Common L3 workflow
- `http://docs.openstack.org/grizzly/openstack-network/admin/content/l3_workflow.html`

The **ip**(8) man page.

# Personal Notes



# Summary

Installing OpenStack networking

- Configure OpenStack networking.

Configuring OpenStack networking

- Configure OpenStack networking.

# CHAPTER 9

# IMPLEMENTING THE NOVA COMPUTE AND NOVA CONTROLLER SERVICES

## Introduction

| Chapter details | |
|---|---|
| **Chapter goal** | Configure Nova compute and Nova network. |
| **Chapter sections** | • Installing Nova compute and Nova controller<br>• Deploying instances using the command line |
| **Hands-on activities** | • Installing Nova compute and Nova controller<br>• Deploying instances using the command line |
| **Chapter test** | None |

# Installing Nova compute and Nova controller

The Nova controller node is the node that runs the **nova-scheduler** and coordinates the activities of OpenStack. The Nova compute node runs the virtualization software to launch and manage instances for OpenStack.

Use the **openstack-db** command to configure the database for the Nova service. Create a user for Nova and add this user to the **services** tenant (as all services should be). Create the end points and configure the **/etc/nova/nova.conf** file with the proper settings for your environment. Finally, start and enable the Nova services. Check the status of any of the Red Hat OpenStack services by running the **openstack-status** command.

Workshop

## Installing Nova compute and Nova controller

☐ 1. On **serverX.example.com**, install the packages necessary for Nova compute and controller.

```
[root@serverX ~]# yum install -y openstack-nova openstack-nova-novncproxy
```

☐ 2. Source the **/root/keystonerc_admin** file to prepare to manage Keystone.

```
[root@serverX ~]# source /root/keystonerc_admin
[root@serverX ~(keystone_admin)]#
```

☐ 3. Fix the permissions on the log file and initialize the database.

```
[root@serverX ~(keystone_admin)]# chown nova:nova /var/log/nova/nova-manage.log
[root@serverX ~(keystone_admin)]# openstack-db --init --service nova --password
 redhat --rootpw redhat
```

☐ 4. Create the **nova** user, then link the **nova** user and the **admin** role within the **services** tenant.

```
[root@serverX ~(keystone_admin)]# keystone user-create --name nova --pass redhat
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |                                  |
| enabled  |               True               |
|    id    | 90abcdef1234567890abcdef12345678 |
|   name   |               nova               |
| tenantId |                                  |
+----------+----------------------------------+
[root@serverX ~(keystone_admin)]# keystone user-role-add --user nova --role admin
 --tenant services
```

☐ 5. Add the service to the Keystone catalog.

```
[root@serverX ~(keystone_admin)]# keystone service-create --name nova --type
 compute --description "OpenStack Compute Service"
```

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |    OpenStack Compute Service     |
|      id     | 9876543210fedcba9876543210fedcba |
|     name    |               nova               |
|     type    |             compute              |
+-------------+----------------------------------+
```

☐ 6. Create the end points for the compute service.

```
[root@serverX ~(keystone_admin)]# keystone endpoint-create --service-
id 9876543210fedcba9876543210fedcba --publicurl 'http://serverX.example.com:8774/
v2/%(tenant_id)s' --adminurl 'http://serverX.example.com:8774/v2/%(tenant_id)s' --
internalurl 'http://serverX.example.com:8774/v2/%(tenant_id)s'
+-------------+-----------------------------------------------+
|   Property  |                     Value                     |
+-------------+-----------------------------------------------+
|   adminurl  | http://serverX.example.com:8774/v2/%(tenant_id)s |
|      id     |         3210fedcba9876543210fedcba987654      |
| internalurl | http://serverX.example.com:8774/v2/%(tenant_id)s |
|  publicurl  | http://serverX.example.com:8774/v2/%(tenant_id)s |
|    region   |                   regionOne                   |
|  service_id |         9876543210fedcba9876543210fedcba      |
+-------------+-----------------------------------------------+
```

☐ 7. Before you begin, back up the configuration files that you will change.

```
[root@serverX ~(keystone_admin)]# cp /etc/nova/api-paste.ini /etc/nova/api-
paste.ini.orig
[root@serverX ~(keystone_admin)]# cp /etc/nova/nova.conf /etc/nova/nova.conf.orig
```

☐ 8. Update the Nova configuration to use the Keystone information.

```
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/nova/api-paste.ini
 filter:authtoken admin_tenant_name services
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/nova/api-paste.ini
 filter:authtoken admin_user nova
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/nova/api-paste.ini
 filter:authtoken admin_password redhat
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/nova/api-paste.ini
 filter:authtoken auth_host 192.168.0.X+100
```

☐ 9. Update the Nova configuration with the Qpid authentication information.

```
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/nova/nova.conf
 DEFAULT qpid_username qpidauth
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/nova/nova.conf
 DEFAULT qpid_password redhat
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/nova/nova.conf
 DEFAULT qpid_protocol ssl
```

☐ 10. Update the Nova configuration to set the VNC server and LibVirt parameters.

```
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/nova/nova.conf
 DEFAULT novncproxy_base_url http://192.168.0.X+100:6080/vnc_auto.html
```

```
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/nova/nova.conf
 DEFAULT vncserver_listen 192.168.0.X+100
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/nova/nova.conf
 DEFAULT vncserver_proxyclient_address 192.168.0.X+100

[root@serverX ~(keystone_admin)]# openstack-config --set /etc/nova/nova.conf
 DEFAULT libvirt_vif_driver nova.virt.libvirt.vif.LibvirtGenericVIFDriver
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/nova/nova.conf
 DEFAULT auth_strategy keystone
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/nova/nova.conf
 DEFAULT libvirt_type qemu
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/nova/nova.conf
 DEFAULT libvirt_cpu_mode none
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/nova/nova.conf
 DEFAULT verbose true
```

☐ 11. Start and enable the services. Check for any errors.

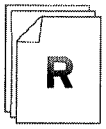
```
[root@serverX ~(keystone_admin)]# service libvirtd start
[root@serverX ~(keystone_admin)]# service openstack-nova-scheduler start
[root@serverX ~(keystone_admin)]# service openstack-nova-api start
[root@serverX ~(keystone_admin)]# service openstack-nova-compute start
[root@serverX ~(keystone_admin)]# service openstack-nova-conductor start
[root@serverX ~(keystone_admin)]# service openstack-nova-consoleauth start
[root@serverX ~(keystone_admin)]# service openstack-nova-novncproxy start

[root@serverX ~(keystone_admin)]# chkconfig openstack-nova-scheduler on
[root@serverX ~(keystone_admin)]# chkconfig openstack-nova-api on
[root@serverX ~(keystone_admin)]# chkconfig openstack-nova-compute on
[root@serverX ~(keystone_admin)]# chkconfig openstack-nova-conductor on
[root@serverX ~(keystone_admin)]# chkconfig openstack-nova-consoleauth on
[root@serverX ~(keystone_admin)]# chkconfig openstack-nova-novncproxy on

[root@serverX ~(keystone_admin)]# tail /var/log/nova/*
```

☐ 12. Check the status of all the OpenStack services.

```
[root@serverX (keystone_admin)~]# openstack-status
== Nova services ==
openstack-nova-api:           active
openstack-nova-cert:          inactive (disabled on boot)
openstack-nova-compute:       active
openstack-nova-network:       inactive (disabled on boot)
openstack-nova-scheduler:     active
openstack-nova-volume:        dead (disabled on boot)
openstack-nova-conductor:     active
...
== Support services ==
mysqld:                       active
libvirtd:                     active
...
```




## References

Red Hat OpenStack Installation and Configuration Guide

- Chapter 10. Installing the OpenStack compute service

# Deploying instances using the command line

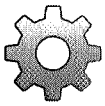
The following list of commands can be used to view IDs and other information needed for deploying instances.

- **nova flavor-list**: Used to view the "hardware" settings for the instance, such as RAM, VCPUs, etc.
- **nova image-list**: Used to view the images used for launching an instance.
- **nova network-list**: Used to view the networks available to the instances.

Once you have gathered the IDs, use the **nova boot** command to launch an instance. **nova list** is used to view the list of running instances.

## Note

The Keystone credentials used when running the **nova boot** command are important. These credentials provide the tenant (project) that will be used to launch the instance. **nova list** will only show running instances in the tenant provided with the credentials, unless you also use the **--all-tenants** option (admin only).



Workshop

## Deploying instances using the command line

Perform this workshop together with your instructor.

Perform the following steps on **serverX.example.com** unless instructed otherwise.

☐ 1. Source the **/root/keystonerc_myuser** file to prepare to launch an instance.

```
[root@serverX ~]# source /root/keystonerc_myuser
[root@serverX ~(keystone_myuser)]#
```

☐ 2. Copy **/etc/issue** to **/tmp/issue** and add a line: **Installed using nova command-line**.

```
[root@serverX ~(keystone_myuser)]# cp /etc/issue /tmp/
[root@serverX ~(keystone_myuser)]# echo "Installed using nova command-line" >> /
tmp/issue
```

☐ 3. Create a new SSH key pair named **key1** and save the private key file to **/root/key1.pem**.

```
[root@serverX ~(keystone_myuser)]# nova keypair-add key1 > /root/key1.pem
```

☐ 4. Create a new security group named **mysecgroup** and allow TCP/22 for 0.0.0.0/0.

```
[root@serverX ~(keystone_myuser)]# nova secgroup-create mysecgroup "SSH"
+------------+-------------+
```

```
| Name       | Description |
+------------+-------------+
| mysecgroup | SSH         |
+------------+-------------+
[root@serverX ~(keystone_myuser)]# nova secgroup-add-rule mysecgroup tcp 22 22
 0.0.0.0/0
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| tcp         | 22        | 22      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
```

☐ 5. Find the name of the image uploaded earlier in class (it should be **test**).

```
[root@serverX ~(keystone_myuser)]# nova image-list
+--------------------------------------+-------+--------+--------+
| ID                                   | Name  | Status | Server |
+--------------------------------------+-------+--------+--------+
| 67890abc-def1-2345-6789-0abcdef12345 | test  | ACTIVE |        |
+--------------------------------------+-------+--------+--------+
```

☐ 6. Display the list of flavors.

```
[root@serverX ~(keystone_myuser)]$ nova flavor-list
+----+----------+-----------+------+-----------+------+-------+-------------
+-----------+-------------+
| ID | Name     | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor |
 Is_Public | extra_specs |
+----+----------+-----------+------+-----------+------+-------+-------------
+-----------+-------------+
| 1  | m1.tiny   | 512       | 0    | 0         |      | 1     | 1.0         |
 True      | {}          |
| 2  | m1.small  | 2048      | 20   | 0         |      | 1     | 1.0         |
 True      | {}          |
| 3  | m1.medium | 4096      | 40   | 0         |      | 2     | 1.0         |
 True      | {}          |
| 4  | m1.large  | 8192      | 80   | 0         |      | 4     | 1.0         |
 True      | {}          |
| 5  | m1.xlarge | 16384     | 160  | 0         |      | 8     | 1.0         |
 True      | {}          |
+----+----------+-----------+------+-----------+------+-------+-------------
+-----------+-------------+
```

☐ 7. Find the network ID of the **private** network.

```
[root@serverX ~(keystone_myuser)]# neutron net-list
+--------------------------------------+---------
+------------------------------------------------------+
| id                                   | name    | subnets
                    |
+--------------------------------------+---------
+------------------------------------------------------+
| ef123456-7890-abcd-ef12-34567890abcd | public  | 00000000-1111-2222-aaaa-
bbbbbbbbbbbb                         |
| 0abcdef1-2345-6789-0abc-def123456789 | private | bcdef123-4567-890a-bcde-
f1234567890a 192.168.32.0/24 |
+--------------------------------------+---------
+------------------------------------------------------+
```

☐ 8. Use the **m1.tiny** flavor, **test** image, **key1** key, **mysecgroup** security group, **private** network ID, and **/tmp/issue** file to create a new instance named **test**.

```
[root@serverX ~(keystone_myuser)]# nova boot --flavor m1.tiny --image test --
key-name key1 --security-groups mysecgroup --nic net-id=0abcdef1-2345-6789-0abc-
def123456789 --file /etc/issue=/tmp/issue test
+----------------------------+--------------------------------------+
| Property                   | Value                                |
+----------------------------+--------------------------------------+
| status                     | BUILD                                |
| updated                    | 2014-01-01T00:12:34Z                 |
| OS-EXT-STS:task_state      | scheduling                           |
| key_name                   | key1                                 |
| image                      | test                                 |
| hostId                     |                                      |
| OS-EXT-STS:vm_state        | building                             |
| flavor                     | m1.tiny                              |
| id                         | 7890abcd-ef12-3456-7890-abcdef123456 |
| security_groups            | [{u'name': u'mysecgroup'}]           |
| user_id                    | 90abcdef1234567890abcdef12345678     |
| name                       | test                                 |
| adminPass                  | aA1bB2yY9zZ0                         |
| tenant_id                  | 890abcdef1234567890abcdef1234567     |
| created                    | 2013-04-01T20:28:21Z                 |
| OS-DCF:diskConfig          | MANUAL                               |
| metadata                   | {}                                   |
| accessIPv4                 |                                      |
| accessIPv6                 |                                      |
| progress                   | 0                                    |
| OS-EXT-STS:power_state     | 0                                    |
| OS-EXT-AZ:availability_zone | nova                                |
| config_drive               |                                      |
+----------------------------+--------------------------------------+
```

☐ 9. Use **nova list** to view the status of the instance. It will start in the **BUILD** status and move to the **ACTIVE** status as follows.

```
[root@serverX ~(keystone_myuser)]# nova list
+--------------------------------------+-------+--------+----------+
| ID                                   | Name  | Status | Networks |
+--------------------------------------+-------+--------+----------+
| 7890abcd-ef12-3456-7890-abcdef123456 | test  | BUILD  |          |
+--------------------------------------+-------+--------+----------+

[root@serverX ~(keystone_myuser)]# nova list
+--------------------------------------+-------+--------+---------------------+
| ID                                   | Name  | Status | Networks            |
+--------------------------------------+-------+--------+---------------------+
| 7890abcd-ef12-3456-7890-abcdef123456 | test  | ACTIVE | private=192.168.32.2 |
+--------------------------------------+-------+--------+---------------------+
```

☐ 10. While waiting for the instance to boot, install the Horizon dashboard. Begin by installing the packages needed.

```
[root@serverX ~(keystone_myuser)]# yum install -y mod_wsgi httpd mod_ssl
 openstack-dashboard memcached python-memcached
```

☐ 11. Fix a broken permission:

```
[root@serverX ~(keystone_myuser)]# chown apache /var/lib/openstack-
dashboard/.secret_key_store
```

☐ 12. In **/etc/openstack-dashboard/local_settings**, find the **OPENSTACK_HOST** line and change the IP address as listed in the following section. Create a new line and add the **CACHE_BACKEND** value as follows.

```
OPENSTACK_HOST = "192.168.0.X+100"
CACHE_BACKEND = "memcached://127.0.0.1:11211/"
```

☐ 13. The Horizon dashboard requires a Keystone role named **Member**. Source the admin credentials and verify you have this role.

```
[root@serverX ~(keystone_myuser)]# source /root/keystonerc_admin
[root@serverX ~(keystone_admin)]# keystone role-list
+----------------------------------+----------+
|                id                |   name   |
+----------------------------------+----------+
| 234567890abcdef1234567890abcdef1 |  Member  |
| 684fd48df02d4643ae7cb257bddb41cb | _member_ |
| fad9876543210fad9876543210fad987 |  admin   |
+----------------------------------+----------+
```

If the role is not present, create it:

```
[root@serverX ~(keystone_admin)]# keystone role-create --name Member
```

☐ 14. Modify the SELinux policy to allow connections from the web server to the Keystone identity server.

```
[root@serverX ~(keystone_admin)]# setsebool -P httpd_can_network_connect on
```

☐ 15. The dashboard is a Django (Python) web application, so start and enable the web server.

```
[root@serverX ~(keystone_admin)]# service httpd start
[root@serverX ~(keystone_admin)]# chkconfig httpd on
```

☐ 16. Verify that the Horizon dashboard is working properly. Use **Firefox** to browse to `http://serverX.example.com/dashboard`. Log in as **myuser** with a password of **redhat**.

☐ 17. Connect to the console of the new instance. In the **Project** pane to the left, click the **Instances** link. Click the **test** instance link. Browse to the **Console** tab to view the console of the instance.

☐ 18. You should be able to view the result of the new **/etc/issue** file. Log into the instance as **root** (password: **redhat**).

☐ 19. Back on **serverX.example.com**, source the **/root/keystonerc_myuser** file to use the **myuser** credentials.

```
[root@serverX ~(keystone_admin)]# source /root/keystonerc_myuser
[root@serverX ~(keystone_myuser)]#
```

☐ 20. Assign a floating IP address to the instance. First, find the IP address of the instance.

```
[root@serverX ~(keystone_myuser)]# nova list
+--------------------------------------+-------+--------+---------------------+
| ID                                   | Name  | Status | Networks            |
+--------------------------------------+-------+--------+---------------------+
| 7890abcd-ef12-3456-7890-abcdef123456 | test  | ACTIVE | private=192.168.32.2 |
+--------------------------------------+-------+--------+---------------------+
```

Next, find the port ID associated with the IP address of the instance.

```
[root@serverX ~(keystone_myuser)]# neutron port-list
+--------------------------------------+------+-------------------
+-----------------------------------------------------------------------------------------
+
| id                                   | name | mac_address       | fixed_ips
                                                                                        |
+--------------------------------------+------+-------------------
+-----------------------------------------------------------------------------------------
+
| f1234567-890a-bcde-f123-4567890abcde |      | fa:16:3e:20:0e:8b | {"subnet_id":
 "bcdef123-4567-890a-bcde-f1234567890a", "ip_address": "192.168.32.1"} |
| aaaaaaa-bbbb-cccc-dddd-eeeeeeffffff |      | fa:16:3e:3e:5a:39 | {"subnet_id":
 "bcdef123-4567-890a-bcde-f1234567890a", "ip_address": "192.168.32.2"} |
+--------------------------------------+------+-------------------
+-----------------------------------------------------------------------------------------
+
```

Find the floating IP address ID.

```
[root@serverX ~(keystone_myuser)]# neutron floatingip-list
+--------------------------------------+------------------+---------------------
+---------+
| id                                   | fixed_ip_address | floating_ip_address |
 port_id |
+--------------------------------------+------------------+---------------------
+---------+
| bedbed00-bedb-ed00-bedb-ed00bedbed00 |                  | 172.24.X.2          |
         |
+--------------------------------------+------------------+---------------------
+---------+
```

Finally, associate the floating IP address with the port.

```
[root@serverX ~(keystone_myuser)]# neutron floatingip-associate bedbed00-bedb-
ed00-bedb-ed00bedbed00 aaaaaaa-bbbb-cccc-dddd-eeeeeeffffff
Associated floatingip bedbed00-bedb-ed00-bedb-ed00bedbed00
```

☐ 21. **ssh** to the instance using the new floating IP address and the **key1.pem** SSH key.

```
[root@serverX ~(keystone_myuser)]# chmod 600 /root/key1.pem
```

```
[root@serverX ~(keystone_myuser)]# ssh -i /root/key1.pem root@172.24.X.2
```

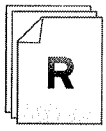
Leave this **ssh** session open, as we will run some commands here later.

☐ 22. In another terminal on **serverX**, attach the volume to the instance and verify that the instance can manage the volume. Source the **/root/keystonerc_myuser** file if necessary.

```
[root@desktopX ~]# ssh root@serverX
[root@serverX ~]# source /root/keystonerc_myuser
[root@serverX ~]# cinder create --display-name vol1 2
[root@serverX ~(keystone_myuser)]# cinder list
+--------------------------------------+-----------+--------------+------
+-------------+----------+-------------+
|                  ID                  |   Status  | Display Name | Size | Volume
 Type | Bootable | Attached to |
+--------------------------------------+-----------+--------------+------
+-------------+----------+-------------+
| cdef1234-5678-90ab-cdef-1234567890ab | available |     vol1     |  2   |
 None    |  false   |             |
+--------------------------------------+-----------+--------------+------
+-------------+----------+-------------+
[root@serverX ~(keystone_myuser)]# nova volume-attach test cdef1234-5678-90ab-
cdef-1234567890ab auto
+----------+--------------------------------------+
| Property | Value                                |
+----------+--------------------------------------+
| device   | /dev/vdb                             |
| serverId | 7890abcd-ef12-3456-7890-abcdef123456 |
| id       | cdef1234-5678-90ab-cdef-1234567890ab |
| volumeId | cdef1234-5678-90ab-cdef-1234567890ab |
+----------+--------------------------------------+
[root@serverX ~(keystone_myuser)]# cinder list
+--------------------------------------+--------+--------------+------
+-------------+----------+--------------------------------------+
|                  ID                  | Status | Display Name | Size | Volume
 Type | Bootable |             Attached to              |
+--------------------------------------+--------+--------------+------
+-------------+----------+--------------------------------------+
| cdef1234-5678-90ab-cdef-1234567890ab | in-use |     vol1     |  2   |     None
  |  false   | 7890abcd-ef12-3456-7890-abcdef123456 |
+--------------------------------------+--------+--------------+------
+-------------+----------+--------------------------------------+
```

Back in the **ssh** session on the instance, verify that the volume was attached.

```
[root@192-168-32-2 ~]# fdisk -cul /dev/vdb
Disk /dev/vdb: 2147 MB, 2147483648 bytes
16 heads, 63 sectors/track, 4161 cylinders, total 4194304 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
```




## References

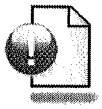
Red Hat OpenStack Installation and Configuration Guide

- Chapter 11. Installing the dashboard
- Chapter 12. Working with instances
- Chapter 13. Updating the environment

Red Hat OpenStack Getting Started Guide

- Chapter 7. Using OpenStack with the command-line interface

# Personal Notes



# Summary

Installing Nova compute and Nova controller

- Add a Nova compute node to an existing OpenStack cloud.
- Remove a Nova compute node from an existing OpenStack cloud.

Deploying instances using the command line

- Deploy an instance using **nova**.

# CHAPTER 10

# IMPLEMENTING AN ADDITIONAL NOVA COMPUTE NODE

## Introduction

| Chapter details | |
|---|---|
| **Chapter goal** | Provide redundancy for compute, image, and block storage services. |
| **Chapter sections** | • Preparing the Nova compute node<br>• Managing Nova compute nodes<br>• Configuring networking on the Nova compute node and launching an instance |
| **Hands-on activities** | • Rebuilding Red Hat OpenStack all-in-one<br>• Configuring OpenStack networking on the Nova controller node<br>• Managing Nova compute nodes<br>• Configuring OpenStack networking on the Nova compute node<br>• Preparing and launching an instance |
| **Chapter test** | None |

# Preparing the Nova compute node

In this workshop, you will reinstall Red Hat OpenStack using packstack, then configure OpenStack networking. Once you have configured Red Hat OpenStack, you will add an additional compute node. This section is a review meant to prepare your system so that you can add an additional compute node to Red Hat OpenStack.

Workshop

## Rebuilding Red Hat OpenStack all-in-one

Follow along with the instructor as you perform the setup tasks required to install Red Hat OpenStack.

☐ 1. Reset your **server*X*** virtual machine to the last saved state.

If you are in a physical classroom, log in as **root** on **desktop*X*.example.com** and reset your **server*X*** machine:

```
[root@desktopX ~]# lab-reset-vm
This will destroy the virtual machine and reset it to the last saved state.
Is this ok [y/N]: y
Waiting for things to settle...
Done.
```

If you are working in the virtual training environment, ignore the preceding paragraph and use the virtual machine controls in your web browser to reset your machine to a snapshot of your **server*X*** machine. Open the **state** dropdown menu and select the **Snapshots** tab. Select the **Chapter 3** snapshot via the radio buttons and click the **Revert to selected snapshot** button. Press the **Power On** button to start **server*X*.example.com**.

☐ 2. Once **server*X*** is available, **ssh** to **server*X*** as **root**.

```
[root@desktopX ~]# ssh root@serverX
Password: redhat
[root@serverX ~]#
```

☐ 3. On **server*X***, install the software for **packstack**.

```
[root@serverX ~]# yum install -y openstack-packstack
```

☐ 4. Create an SSH key.

```
[root@serverX ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): Enter
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
...
```

☐ 5. Generate an answer file.

```
[root@serverX ~]# packstack --gen-answer-file /root/answers.txt
```

☐ 6. Edit the answer file to *disable* Ceilometer (we will install it manually in a later chapter), include **192.168.0.254** as the NTP server, and allow Horizon to use secure SSL connections.

```
CONFIG_CEILOMETER_INSTALL=n
CONFIG_NTP_SERVERS=192.168.0.254
CONFIG_HORIZON_SSL=y
```

If you are using the Red Hat Online Learning environment, do not set the CONFIG_NTP_SERVERS variable.

☐ 7. Use **packstack** to install Red Hat OpenStack.

```
[root@serverX ~]# packstack --answer-file /root/answers.txt
Welcome to Installer setup utility

Installing:
Clean Up...                                  [DONE]
Setting up ssh keys...root@192.168.0.X+100's password: redhat
...
```

☐ 8. Once the installation has completed, verify that there are no errors. Power off **serverX.example.com** and save the state of the virtual machine.

If you are in a physical classroom, log in as **root** on **desktopX.example.com** and save the state of your **serverX** machine:

```
[root@serverX ~]# poweroff
```

```
[root@desktopX ~]# virsh list
 Id    Name                           State
----------------------------------------------------
 1     serverX                        running

...

[root@desktopX ~]# virsh list
 Id    Name                           State
----------------------------------------------------

[root@desktopX ~]# lab-save-vm
This will save the current state of the virtual machine.
Is this ok [y/N]: y
```

If you are using the Red Hat Online Learning environment, you may see other virtual machines besides the **serverX** virtual machine.

If you are working in the virtual training environment, ignore the preceding paragraph and use the virtual machine controls in your web browser to create a snapshot of your **serverX** machine. First, shut down **serverX.example.com**. When

**server*X*.example.com** is shut down, open the **state** dropdown menu and select the **Snapshots** tab. Enter **Chapter 10** as the name in the **Create new snapshots...** box and click the **Create** button. Click the **Refresh** button to verify that the new snapshot was created. Press the **Power On** button to start **server*X*.example.com**.

The Open vSwitch plug-in for the OpenStack networking service was configured by **packstack** using the **local** tenant network type, which is only useful for the "all-in-one" installation. If more than one node is configured (as we will do later), you must configure Open vSwitch with some other method. VLAN and GRE are the two methods supported by Red Hat as of this writing.

Workshop

## Configuring OpenStack networking on the Nova controller node

☐ 1. Log in as **root** on **server*X*.example.com**.

```
[root@desktopX ~]# ssh serverX.example.com
[root@serverX ~]#
```

☐ 2. As **root** on **server*X*.example.com**, configure VLAN for Open vSwitch using VLANs 1-100 on the bridge **br-eth1** using a physical network named **physnet1**.

```
[root@serverX ~]# cp /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini /etc/
neutron/plugins/openvswitch/ovs_neutron_plugin.ini.orig
[root@serverX ~]# openstack-config --set /etc/neutron/plugins/openvswitch/
ovs_neutron_plugin.ini OVS tenant_network_type vlan
[root@serverX ~]# openstack-config --set /etc/neutron/plugins/openvswitch/
ovs_neutron_plugin.ini OVS network_vlan_ranges physnet1:1:100
[root@serverX ~]# openstack-config --set /etc/neutron/plugins/openvswitch/
ovs_neutron_plugin.ini OVS bridge_mappings physnet1:br-eth1
```

☐ 3. Create the **br-eth1** bridge in Open vSwitch and attach the **eth1** network device to the bridge.

```
[root@serverX ~]# ovs-vsctl add-br br-eth1
[root@serverX ~]# ovs-vsctl add-port br-eth1 eth1
```

☐ 4. Restart the **neutron-openvswitch-agent** service and check for errors.

```
[root@serverX ~]# service neutron-openvswitch-agent restart
[root@serverX ~]# egrep 'ERROR|CRITICAL' /var/log/neutron/openvswitch-agent.log
```

☐ 5. Before attaching **eth0** to the **br-ex** bridge, configure the **br-ex** network device configuration file.

```
[root@serverX ~]# cp /etc/sysconfig/network-scripts/ifcfg-eth0 /root/
[root@serverX ~]# cp /etc/sysconfig/network-scripts/ifcfg-eth0 /etc/sysconfig/
network-scripts/ifcfg-br-ex
```

If you are in a physical classroom, remove everything but the **DEVICE**, **HWADDR**, and **ONBOOT** settings from **/etc/sysconfig/network-scripts/ifcfg-eth0** so that it looks like the following:

```
DEVICE=eth0
HWADDR=52:54:00:00:00:XX
```

```
ONBOOT=yes
```

If you are in a virtual classroom, remove everything but the **DEVICE** and **ONBOOT** settings from **/etc/sysconfig/network-scripts/ifcfg-eth0** so that it looks like the following:

```
DEVICE=eth0
ONBOOT=yes
```

In the **/etc/sysconfig/network-scripts/ifcfg-br-ex** file, remove the **HWADDR** line and change the device name to **br-ex**. Make sure the **/etc/sysconfig/network-scripts/ifcfg-br-ex** file contains the following:

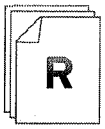
```
DEVICE=br-ex
IPADDR=192.168.0.X+100
PREFIX=24
GATEWAY=192.168.0.254
DNS1=192.168.0.254
SEARCH1=example.com
ONBOOT=yes
```

☐ 6. Once you have verified the network files contain the correct information, add the **eth0** network device to the **br-ex** bridge and restart the network.

```
[root@serverX ~]# ovs-vsctl add-port br-ex eth0 ; service network restart
```

☐ 7. Once the VLAN settings are configured and the network devices are attached to the bridges, restart the OpenStack networking services.

```
[root@serverX ~]# for i in /etc/init.d/neutron* ; do $i condrestart ; done
```




## References

Red Hat OpenStack Getting Started Guide

- Section 7.8. Working with OpenStack networking

Red Hat OpenStack Installation and Configuration Guide

- Section 9.4. Configuring the networking service

# Managing Nova compute nodes

Adding extra compute nodes is one of the first ways to expand in an OpenStack cloud. Nova is the compute service used on a hypervisor to manage instances.

## Install a Nova compute node

To install a Nova compute node, subscribe the system to the Red Hat OpenStack channel and install the *openstack-nova-compute* package. Copy the **/etc/nova/nova.conf** file from the cloud controller or another Nova compute node to the new Nova compute node. Edit the **/etc/nova/nova.conf** file on the new Nova compute node and change at least the following:

```
my_ip=192.168.0.X
vncserver_listen=$my_ip
vncserver_proxyclient_address=$my_ip
```

On the new Nova compute node, start and enable the **openstack-nova-compute** service. Check for errors in **/var/log/nova/compute.log** and verify the status using **openstack-status**.

Copy the **keystonerc_admin** file to the new Nova compute node and source the file. Run **nova-manage host list** and **nova-manage service list** to verify the status of the new Nova compute node.

To disable a machine running as a Nova compute node, run the **nova-manage service disable** command. The following shows an example removing the server*X*.example.com host:

```
[root@serverX ~(keystone_admin)]# nova-manage service disable --host serverX.example.com
 --service nova-compute
```

Demonstration

## Adding a Nova compute node

1. Disable debugging in Nova so as not to have too much output.

```
[root@demo ~]# openstack-config --set /etc/nova/nova.conf DEFAULT debug False
[root@demo ~]# for i in /etc/init.d/openstack-nova-* ; do $i condrestart ; done
```

2. Ensure the Red Hat OpenStack cloud is up and running with **demo** serving as a Nova compute node.

```
[root@demo ~]# nova-manage host list
host                    zone
demo.example.com        internal
[root@demo ~(keystone_admin)]# nova-manage service list
Binary           Host                  Zone             Status     State Updated_At
nova-consoleauth demo.example.com      internal         enabled    :-)   2014-01-01
 00:01:46
nova-scheduler   demo.example.com      internal         enabled    :-)   2014-01-01
 00:01:43
nova-compute     demo.example.com      nova             enabled    :-)
 2014-01-01 00:01:46
```

```
nova-conductor   demo.example.com        internal         enabled    :-)   2014-01-01
 00:01:43
nova-cert        demo.example.com        internal         enabled    :-)   2014-01-01
 00:01:46
```

3. On **instructor.example.com**, install the packages necessary for a Nova compute node.

```
[instructor@instructor ~]# sudo yum install -y openstack-nova-compute
```

4. Copy the **/etc/nova/nova.conf** file from **demo.example.com** to **instructor.example.com**.

```
[root@demo ~]# scp /etc/nova/nova.conf root@instructor.example.com:/etc/nova/
```

5. On **instructor**, change the following in the **/etc/nova/nova.conf** file:

```
my_ip=192.168.0.254
libvirt_type=kvm
vncserver_listen=$my_ip
vncserver_proxyclient_address=$my_ip
```

If you are in a virtual classroom, leave the **libvirt_type** as **qemu**. Do not change it to **kvm**.

6. Start and enable the **openstack-nova-compute** service once you have checked for errors.

```
[instructor@instructor ~]$ sudo service openstack-nova-compute start
[instructor@instructor ~]$ sudo grep ERROR /var/log/nova/compute.log
[instructor@instructor ~]$ sudo chkconfig openstack-nova-compute on
```

7. Verify the hosts and services.

```
[instructor@instructor ~]# nova-manage host list
host                    zone
demo.example.com        internal
instructor.example.com  nova
[instructor@instructor ~]$ nova-manage service list
Binary           Host                    Zone             Status     State Updated_At
nova-consoleauth demo.example.com        internal         enabled    :-)   2014-01-01
 00:01:46
nova-scheduler   demo.example.com        internal         enabled    :-)   2014-01-01
 00:01:43
nova-compute     demo.example.com        nova             enabled    :-)   2014-01-01
 00:01:46
nova-network     demo.example.com        internal         enabled    :-)   2014-01-01
 00:01:43
nova-cert        demo.example.com        internal         enabled    :-)   2014-01-01
 00:01:46
nova-compute     instructor.example.com  nova             enabled    :-)   2014-01-01
 00:01:46
```

Demonstration

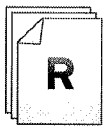
# Removing a Nova compute node

1. From **instructor.example.com**, disable the Nova compute service on **demo.example.com**.

```
[root@instructor ~]# nova-manage service disable --host demo.example.com --service
 nova-compute
```

2. Verify that **demo.example.com** is no longer functioning as a Nova compute node.

```
[root@instructor ~]# nova-manage service list
Binary           Host                  Zone             Status     State Updated_At
nova-consoleauth demo.example.com      internal         enabled    :-)   2014-01-01
 00:01:46
nova-scheduler   demo.example.com      internal         enabled    :-)   2014-01-01
 00:01:43
nova-compute     demo.example.com      nova             disabled   :-)   2014-01-01
 00:01:46
nova-network     demo.example.com      internal         enabled    :-)   2014-01-01
 00:01:43
nova-cert        demo.example.com      internal         enabled    :-)   2014-01-01
 00:01:46
nova-compute     instructor.example.com nova            enabled    :-)   2014-01-01
 00:01:46
```



## References

Red Hat OpenStack Installation and Configuration Guide

- Section 10.3. Installing a compute node
- Chapter 16. Managing compute expansion

Performance Checklist

# Managing Nova compute nodes

***Before you begin...***

Ensure **serverX.example.com** is configured with a Red Hat OpenStack all-in-one installation.

**Lab Outline:** In this lab, you will configure your physical server (**desktopX**) as a Nova compute node and disable Nova compute on the virtual server (**serverX**).

☐ 1. Disable debugging in Nova so as not to have too much output.

```
[root@serverX ~]# openstack-config --set /etc/nova/nova.conf DEFAULT debug False
[root@serverX ~]# for i in /etc/init.d/openstack-nova-* ; do $i condrestart ; done
```

☐ 2. Ensure that your Red Hat OpenStack cloud is running on **serverX.example.com**.

```
[root@serverX ~]# source /root/keystonerc_admin
[root@serverX ~(keystone_admin)]# nova-manage host list
host                    zone
serverX.example.com     nova
[root@serverX ~(keystone_admin)]# nova-manage service list
Binary           Host                   Zone             Status     State
 Updated_At
nova-consoleauth serverX.example.com    internal         enabled    :-)
 2014-01-01 00:01:46
nova-scheduler   serverX.example.com    internal         enabled    :-)
 2014-01-01 00:01:43
nova-compute     serverX.example.com    nova             enabled    :-)
 2014-01-01 00:01:46
nova-network     serverX.example.com    internal         enabled    :-)
 2014-01-01 00:01:43
nova-cert        serverX.example.com    internal         enabled    :-)
 2014-01-01 00:01:46
```

☐ 3. On **desktopX.example.com**, install the packages necessary for a Nova compute node.

```
[root@desktopX ~]# yum install -y openstack-nova-compute
```

☐ 4. Make a backup of the **/etc/nova/nova.conf** file, then copy the **/etc/nova/nova.conf** file from **serverX.example.com** to **desktopX.example.com**.

```
[root@desktopX ~]# cp /etc/nova/nova.conf /etc/nova/nova.conf.orig
[root@desktopX ~]# scp serverX:/etc/nova/nova.conf /etc/nova/
```

☐ 5. On **desktopX**, change the following in the **/etc/nova/nova.conf** file:

```
my_ip=192.168.0.X
libvirt_type=kvm
vncserver_listen=$my_ip
vncserver_proxyclient_address=$my_ip
```

If you are in a virtual classroom, leave the **libvirt_type** as **qemu**. Do not change it to **kvm**.

☐ 6. Start and enable the **openstack-nova-compute** service once you have checked for errors.

```
[root@desktopX ~]# service openstack-nova-compute start
[root@desktopX ~]# grep ERROR /var/log/nova/compute.log
[root@desktopX ~]# chkconfig openstack-nova-compute on
```

## Note

You will likely see an error about having a virtual machine running on the compute node that is not in the database. That VM is the **serverX.example.com** virtual machine, so this is to be expected.

☐ 7. Verify the hosts and services.

```
[root@desktopX ~]# nova-manage host list
host                    zone
serverX.example.com     internal
desktopX.example.com    nova
[root@desktopX ~]# nova-manage service list
Binary           Host                    Zone             Status     State
 Updated_At
nova-consoleauth serverX.example.com     internal         enabled    :-)
 2014-01-01 00:01:46
nova-scheduler   serverX.example.com     internal         enabled    :-)
 2014-01-01 00:01:43
nova-compute     serverX.example.com     nova             enabled    :-)
 2014-01-01 00:01:46
nova-network     serverX.example.com     internal         enabled    :-)
 2014-01-01 00:01:43
nova-cert        serverX.example.com     internal         enabled    :-)
 2014-01-01 00:01:46
nova-compute     desktopX.example.com    nova             enabled    :-)
 2014-01-01 00:01:46
```

☐ 8. Once **desktopX.example.com** is running as a Nova compute node, disable the Nova compute service on **serverX.example.com**.

```
[root@desktopX ~]# nova-manage service disable --host serverX.example.com --
service nova-compute
```

☐ 9. Verify that serverX.example.com is no longer functioning as a Nova compute node.

```
[root@desktopX ~]# nova-manage service list
Binary           Host                    Zone             Status     State
 Updated_At
nova-consoleauth serverX.example.com     internal         enabled    :-)
 2014-01-01 00:01:46
nova-scheduler   serverX.example.com     internal         enabled    :-)
 2014-01-01 00:01:43
nova-compute     serverX.example.com     nova             disabled   :-)
 2014-01-01 00:01:46
nova-network     serverX.example.com     internal         enabled    :-)
 2014-01-01 00:01:43
```

```
nova-cert        serverX.example.com     internal         enabled    :-)
 2014-01-01 00:01:46
nova-compute     desktopX.example.com    nova             enabled    :-)
 2014-01-01 00:01:46
```

# Configuring networking on the Nova compute node and launching an instance

Your final step in adding an additional compute node is to configure the network on the compute node, then launch an instance.

Workshop

## Configuring OpenStack networking on the Nova compute node

☐ 1. On **desktop*X*.example.com**, install the *openstack-neutron-openvswitch* package on **desktop*X*.example.com**.

```
[root@desktopX ~]# yum install -y openstack-neutron-openvswitch
```

☐ 2. Copy the OpenStack networking files from **server*X*.example.com** to **desktop*X*.example.com**.

```
[root@desktopX ~]# cp /etc/neutron/neutron.conf /etc/neutron/neutron.conf.orig
[root@desktopX ~]# cp /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini /
etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini.orig
[root@desktopX ~]# scp serverX:/etc/neutron/neutron.conf /etc/neutron/
[root@desktopX ~]# scp serverX:/etc/neutron/plugins/openvswitch/
ovs_neutron_plugin.ini /etc/neutron/plugins/openvswitch/
```

☐ 3. Start and enable the **openvswitch** service and check for errors.

```
[root@desktopX ~]# service openvswitch start
[root@desktopX ~]# tail /var/log/openvswitch/ovs*
[root@desktopX ~]# chkconfig openvswitch on
```

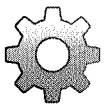
☐ 4. Create the bridges in Open vSwitch and add the **br101** network device to the **br-eth1** bridge.

```
[root@desktopX ~]# ovs-vsctl add-br br-int
[root@desktopX ~]# ovs-vsctl add-br br-eth1
[root@desktopX ~]# ovs-vsctl add-port br-eth1 br101
```

If you are in a virtual classroom, use **eth1** instead of **br101** in the previous command.

☐ 5. Start and enable the **neutron-openvswitch-agent** service and check for errors. Enable the **neutron-openvswitch-agent** service.

```
[root@desktopX ~]# service neutron-openvswitch-agent start
[root@desktopX ~]# tail /var/log/neutron/openvswitch-agent.log
[root@desktopX ~]# chkconfig neutron-openvswitch-agent on
[root@desktopX ~]# chkconfig neutron-ovs-cleanup on
```



Workshop

# Preparing and launching an instance

Once networking is configured, launch an instance using the skills taught in earlier chapters.

☐ 1. On **serverX.example.com**, source the **admin keystonerc_admin** file.

```
[root@serverX ~]# source /root/keystonerc_admin
[root@serverX ~(keystone_admin)]#
```

☐ 2. Create a user name of **user1** with a password of **redhat**.

```
[root@serverX ~(keystone_admin)]# keystone user-create --name user1 --pass redhat
```

☐ 3. Create a tenant named **myproject**.

```
[root@serverX ~(keystone_admin)]# keystone tenant-create --name myproject
```

☐ 4. Add the **user1** user to the **Member** role in the **myproject** tenant.

```
[root@serverX ~(keystone_admin)]# keystone user-role-add --user user1 --role
 Member --tenant myproject
```

☐ 5. Create a keystone file for the **user1** user named **/root/keystonerc_user1**, similar to the **/root/keystonerc_admin** file. It should include the following:

```
export OS_USERNAME=user1
export OS_TENANT_NAME=myproject
export OS_PASSWORD=redhat
export OS_AUTH_URL=http://192.168.0.X+100:35357/v2.0/
export PS1='[\u@\h \W(keystone_user1)]\$ '
```

☐ 6. Source the **user1** keystonerc file.

```
[root@serverX ~(keystone_admin)]# source /root/keystonerc_user1
[root@serverX ~(keystone_user1)]#
```

☐ 7. Upload the **web** image into the image service.

```
[root@serverX ~(keystone_user1)]# glance image-create --name web --is-
public True --disk-format qcow2 --container-format bare --copy-from http://
instructor.example.com/pub/materials/web.img
  ... output omitted ...
[root@serverX ~(keystone_user1)]# glance image-list
+--------------------------------------+------+-------------+-----------------
+----------+--------+
| ID                                   | Name | Disk Format | Container Format |
 Size      | Status |
+--------------------------------------+------+-------------+-----------------
+----------+--------+
```

```
| dcba0987-6543-21fe-dcba-0987654321fe | web  | qcow2       | bare             |
 214103040 | active |
+--------------------------------------+------+-------------+-----------------
+----------+--------+
```

☐ 8. Create a network named **net1**.

```
[root@serverX ~(keystone_user1)]# neutron net-create net1
  ... output omitted ...
```

☐ 9. Create a subnet named **subnet1** in the **net1** network using the **192.168.32.0/24** range.

```
[root@serverX ~(keystone_user1)]# neutron subnet-create --name subnet1 net1
 192.168.32.0/24
 ... output omitted ...
```

☐ 10. Create a router name of **router1**.

```
[root@serverX ~(keystone_user1)]# neutron router-create router1
 ... output omitted ...
```

☐ 11. Add an interface for **subnet1** to the **router1** router.

```
[root@serverX ~(keystone_user1)]# neutron router-interface-add router1 subnet1
Added interface to router router1
```

☐ 12. Using the **admin** credentials, create a network named **net2** with an external router in the **services** tenant.

```
[root@serverX ~(keystone_user1)]# source /root/keystonerc_admin
[root@serverX ~(keystone_admin)]# neutron net-create --tenant-id services net2 --
router:external=True
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 0abcdef1-2345-6789-0abc-def123456789 |
| name                      | net2                                 |
| provider:network_type     | vlan                                 |
| provider:physical_network | physnet1                             |
| provider:segmentation_id  | 2                                    |
| router:external           | True                                 |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | services                             |
+---------------------------+--------------------------------------+
```

Ensure the network was created with a **network_type** of **vlan**, a **physical_network** of **physnet1**, and a **segmentation_id** in the VLAN range configured previously. Also, ensure that the **router:external** shows **True**.

☐ 13. Still using the **admin** credentials, create a subnet named **subnet2** within the **net2** network. Include this subnet in the **services** tenant. Use the network range of **172.24.*X*.0/24** and a gateway of **172.24.*X*.254**. Disable DHCP, as these IP addresses will be assigned as floating IP addresses.

```
[root@serverX ~(keystone_admin)]# neutron subnet-create --tenant-id services --
gateway 172.24.X.254 --disable-dhcp --name subnet2 net2 172.24.X.0/24
Created a new subnet:
...
```

☐ 14. Still using the **admin** credentials, set the gateway for the router to the **net2** network. This will add an interface for the **net2** network.

```
[root@serverX ~(keystone_admin)]# neutron router-gateway-set router1 net2
Set gateway for router router1
```

☐ 15. Source the **user1** keystonerc file.

```
[root@serverX ~(keystone_admin)]# source /root/keystonerc_user1
[root@serverX ~(keystone_user1)]#
```

☐ 16. Create a keypair and save the private key to **/root/key1.pem**. Change the permission of the file to **0600**:

```
[root@serverX ~(keystone_user1)]# nova keypair-add key1 > /root/key1.pem
[root@serverX ~(keystone_user1)]# chmod 0600 /root/key1.pem
```

☐ 17. Create a new security group named **sec1**. Allow TCP/22 and TCP/443 from **0.0.0.0/0**, and allow TCP/80 from the security group.

```
[root@serverX ~(keystone_user1)]# nova secgroup-create sec1 "SSH and Web"
+------+-------------+
| Name | Description |
+------+-------------+
| sec1 | SSH and Web |
+------+-------------+
[root@serverX ~(keystone_user1)]# nova secgroup-add-rule sec1 tcp 22 22 0.0.0.0/0
 ... output omitted ...
[root@serverX ~(keystone_user1)]# nova secgroup-add-rule sec1 tcp 443 443
 0.0.0.0/0
 ... output omitted ...
[root@serverX ~(keystone_user1)]# nova secgroup-add-group-rule sec1 sec1 tcp 80 80
 ... output omitted ...
[root@serverX ~(keystone_user1)]# nova secgroup-list-rules sec1
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| tcp         | 22        | 22      | 0.0.0.0/0 |              |
| tcp         | 80        | 80      |           | sec1         |
| tcp         | 443       | 443     | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
```

☐ 18. Create a script in **/root/userdata** to be executed on the instance. The **/root/userdata** should contain the following:

```
#!/bin/bash
echo Hello >> /root/test
```

## Note

In order to execute a script such as the one previously using **nova boot --user-data...**, the image must have the *cloud-init* package installed and run at boot time. The **web** image has been prepared with the *cloud-init* package included.

☐ 19. Launch an instance named **testweb** using the **m1.small** flavor, the **web** image, the **key1** keypair, and the **sec1** security group, and pass the **/root/userdata** file as user data.

```
[root@serverX ~(keystone_user1)]# nova boot --flavor m1.small --image web --key-
name key1 --security-groups sec1 --user-data /root/userdata --poll testweb
```

☐ 20. Allocate and associate a floating IP address to the instance, and verify that it uses the correct key and allows access to the ports in the security group.

```
[root@serverX ~(keystone_user1)]# nova floating-ip-create net2
+------------+-------------+----------+------+
| Ip         | Instance Id | Fixed Ip | Pool |
+------------+-------------+----------+------+
| 172.24.X.2 | None        | None     | net2 |
+------------+-------------+----------+------+
[root@serverX ~(keystone_user1)]# nova add-floating-ip testweb 172.24.X.2
[root@serverX ~(keystone_user1)]# nova floating-ip-list
+------------+--------------------------------------+-------------+------+
| Ip         | Instance Id                          | Fixed Ip    | Pool |
+------------+--------------------------------------+-------------+------+
| 172.24.X.2 | 7890abcd-ef12-3456-7890-abcdef123456 | 192.168.32.2 | net2 |
+------------+--------------------------------------+-------------+------+
[root@serverX ~(keystone_user1)]# ssh -i key1.pem 172.24.X.2
[root@testweb ~]# cat /root/test
Hello
[root@testweb ~]# exit
[root@serverX ~(keystone_user1)]# curl http://172.24.X.2
My web page
[root@serverX ~(keystone_user1)]# curl -k https://172.24.X.2
My web page
```

☐ 21. Once you have carefully verified your work, remove the instance.

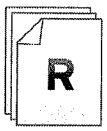
```
[root@serverX ~(keystone_user1)]# nova delete testweb
```

☐ 22. If you are in a physical classroom, log in as **root** on **desktopX.example.com** and save the state of your **serverX** machine:

```
[root@serverX ~(keystone_user1)]# poweroff
```

```
[root@desktopX ~]# lab-save-vm
This will save the current state of the virtual machine.
Is this ok [y/N]: y
```

If you are working in the virtual training environment, ignore the preceding paragraph and use the virtual machine controls in your web browser to create a snapshot of your **serverX** machine. First, shut down **serverX.example.com**. When **serverX.example.com** is shut down, open the **state** dropdown menu and select the **Snapshots** tab. Enter **Chapter 10-2** as the name in the **Create new snapshots...** box and click the **Create** button. Click the **Refresh** button to verify that the new snapshot was created. Press the **Power On** button to start **serverX.example.com**.




## References

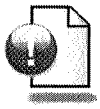
Red Hat OpenStack Getting Started Guide

- Chapter 7. Using OpenStack with the command line interface

Red Hat OpenStack Installation and Configuration Guide

- Chapter 12. Working with instances
- Chapter 13. Updating the environment

# Personal Notes



# Summary

Preparing the Nova compute node

- Install Red Hat OpenStack all-in-one.
- Manually configure OpenStack networking.

Managing Nova compute nodes

- Add a Nova compute node to an existing OpenStack cloud.
- Remove a Nova compute node from an existing OpenStack cloud.

Configuring networking on the Nova compute node and launching an instance

- Configure networking on the compute node.
- Launch an instance using the command line.

# CHAPTER 11

# IMPLEMENTING THE HEAT ORCHESTRATION SERVICE

## Introduction

| Chapter details | |
|---|---|
| **Chapter goal** | Install and configure the Heat orchestration service, and launch instances using preconfigured templates. |
| **Chapter sections** | • Implementing the Heat orchestration service |
| **Hands-on activities** | • Implementing the Heat orchestration service |

# Implementing the Heat 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 via a single modular template, with additional orchestration capabilities such as auto-scaling and basic high availability.

The orchestration service is composed of the following:

- **`heat-cfn`**, a CLI tool that communicates with **`heat-api`** to execute AWS CloudFormation APIs.
- **`heat-api`**, which is an OpenStack-native REST API that processes API requests by sending them to **`heat-engine`** over RPC.
- **`heat-api-cfn`**, which provides an AWS-Query API that is compatible with AWS CloudFormation and processes API requests by sending them to **`heat-engine`** over RPC.
- **`heat-engine`**, which orchestrates the launching of templates and provide events back to the API consumer.
- **`heat-api-cloudwatch`**, which provides monitoring (metrics collection) for the orchestration service.
- *heat-cfntools* and *cloud-init* packages. *heat-cfntools* is a package of helper scripts (for example, **`cfn-hup`**, which handles updates to metadata and executes custom hooks). The *cloud-init* package is used to manage user data passed to an instance. The *heat-cfntools* package is only installed in the images used for instances managed by the Heat orchestration service. The *cloud-init* package should be installed in any images used for instances getting user data.

## Note

The Heat orchestration service is supported as a technical preview in the Red Hat OpenStack 3.0 (Grizzly) release.

Workshop

## Implementing the Heat orchestration service

Follow along with your instructor as you complete this workshop together.

This workshop will guide you through the process of installing the Heat orchestration service. You will launch a stack using preconfigured templates found on `http://instructor.example.com/pub/materials/`.

☐ 1. Once **`serverX.example.com`** has been saved and booted, restart the **`openstack-nova-compute`** service on **`desktopX.example.com`**.

```
[root@desktopX ~]# service openstack-nova-compute restart
```

☐ 2. On **serverX.example.com**, install the packages necessary for the Heat orchestration service.

```
[root@serverX ~]# yum install -y openstack-heat-*
```

☐ 3. Find the MySQL root password.

```
[root@serverX ~]# grep MYSQL_PW /root/answers.txt
CONFIG_MYSQL_PW=234567890abcdef1
```

☐ 4. Export the MySQL password.

```
[root@serverX ~]# export CONFIG_MYSQL_PW=234567890abcdef1
```

☐ 5. Configure the Heat orchestration service database.

```
[root@serverX ~]# openstack-db --init --service heat --password redhat --rootpw
 $CONFIG_MYSQL_PW
[root@serverX ~]# grep sql_connection /etc/heat/heat.conf
sql_connection = mysql://heat:redhat@localhost/heat
```

☐ 6. Source the **/root/keystonerc_admin** file.

```
[root@serverX ~]# source /root/keystonerc_admin
[root@serverX ~(keystone_admin)]#
```

☐ 7. Create the **heat** user in Keystone, then link the **heat** user and the **admin** role within the **services** tenant.

```
[root@serverX ~(keystone_admin)]# keystone user-create --name heat --pass redhat
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |                                  |
| enabled  |               True               |
|    id    | cab1234567890cab1234567890cab123 |
|   name   |               heat               |
| tenantId |                                  |
+----------+----------------------------------+
[root@serverX ~(keystone_admin)]# keystone user-role-add --user heat --role admin
 --tenant services
```

☐ 8. Create the **heat** service in Keystone.

```
[root@serverX ~(keystone_admin)]# keystone service-create --name heat --type
 orchestration --description "Heat Orchestration Service"
+-------------+----------------------------------+
|   Property  |              Value               |
```

```
+-------------+---------------------------------+
| description |    Heat Orchestration Service   |
|      id     | dcba9876543210fedcba9876543210fe |
|     name    |               heat              |
|     type    |          orchestration          |
+-------------+---------------------------------+
```

☐ 9. Use the **heat** service ID to create the **heat** end points in Keystone.

```
[root@serverX ~(keystone_admin)]# keystone endpoint-create --region RegionOne
 --service-id dcba9876543210fedcba9876543210fe --publicurl "http://192.168.0.X
+100:8004/v1/%(tenant_id)s" --adminurl "http://192.168.0.X+100:8004/v1/
%(tenant_id)s" --internalurl "http://192.168.0.X+100:8004/v1/%(tenant_id)s"
+-------------+-----------------------------------------+
|   Property  |                  Value                  |
+-------------+-----------------------------------------+
|   adminurl  | http://192.168.0.X+100:8004/v1/%(tenant_id)s |
|      id     |       dad1234567890dad1234567890dad123       |
| internalurl | http://192.168.0.X+100:8004/v1/%(tenant_id)s |
|  publicurl  | http://192.168.0.X+100:8004/v1/%(tenant_id)s |
|    region   |                RegionOne                |
|  service_id |       dcba9876543210fedcba9876543210fe       |
+-------------+-----------------------------------------+
```

## Note

The **keystone** command uses a default region name of **regionOne**, but **packstack** uses a region of **RegionOne**. Because these are different regions, we must specify a region of **RegionOne** using **keystone** after a **packstack** installation. Otherwise, connections to Heat would not find the service or end points because they would not be in the proper region.

☐ 10. Create the **heat-cfn** service and end points in Keystone.

```
[root@serverX ~(keystone_admin)]# keystone service-create --name heat-cfn --type
 cloudformation --description "Heat Cloudformation Service"
+-------------+---------------------------------+
|   Property  |              Value              |
+-------------+---------------------------------+
| description |   Heat Cloudformation Service   |
|      id     | 9876543210fedcba9876543210fedcba |
|     name    |            heat-cfn             |
|     type    |         cloudformation          |
+-------------+---------------------------------+
[root@serverX ~(keystone_admin)]# keystone endpoint-create --region RegionOne
 --service-id 9876543210fedcba9876543210fedcba --publicurl http://192.168.0.X
+100:8000/v1 --adminurl http://192.168.0.X+100:8000/v1 --internalurl
 http://192.168.0.X+100:8000/v1
+-------------+-----------------------------------+
|   Property  |              Value                |
+-------------+-----------------------------------+
|   adminurl  | http://192.168.0.X+100:8000/v1 |
|      id     |   3210fedcba9876543210fedcba987654   |
| internalurl | http://192.168.0.X+100:8000/v1 |
|  publicurl  | http://192.168.0.X+100:8000/v1 |
|    region   |              RegionOne            |
```

```
| service_id |   9876543210fedcba9876543210fedcba     |
+------------+----------------------------------------+
```

☐ 11. Generate an encryption key and update the Heat configuration file: **/etc/heat/heat.conf**

```
[root@serverX ~(keystone_admin)]# export ENCKEY=$(openssl rand -hex 16)
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/heat/heat.conf
 DEFAULT auth_encryption_key ${ENCKEY}
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/heat/heat.conf
 DEFAULT sql_connection mysql://heat:redhat@localhost/heat
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/heat/heat.conf
 keystone_authtoken admin_tenant_name services
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/heat/heat.conf
 keystone_authtoken admin_user heat
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/heat/heat.conf
 keystone_authtoken admin_password redhat
```

☐ 12. Start and enable the services:

```
[root@serverX ~(keystone_admin)]# service openstack-heat-api start
[root@serverX ~(keystone_admin)]# service openstack-heat-api-cfn start
[root@serverX ~(keystone_admin)]# service openstack-heat-api-cloudwatch start
[root@serverX ~(keystone_admin)]# service openstack-heat-engine start

[root@serverX ~(keystone_admin)]# chkconfig openstack-heat-api on
[root@serverX ~(keystone_admin)]# chkconfig openstack-heat-api-cfn on
[root@serverX ~(keystone_admin)]# chkconfig openstack-heat-api-cloudwatch on
[root@serverX ~(keystone_admin)]# chkconfig openstack-heat-engine on

[root@serverX ~(keystone_admin)]# tail /var/log/heat/*
```

☐ 13. Add the default floating IP pool to the **/etc/nova/nova.conf** file and restart the Nova services.

```
[root@serverX ~(keystone_admin)]# nova floating-ip-pool-list
+--------+
|  name  |
+--------+
|  net2  |
+--------+
[root@serverX ~(keystone_admin)]# openstack-config --set /etc/nova/nova.conf
 DEFAULT default_floating_pool net2
[root@serverX ~(keystone_admin)]# for i in /etc/init.d/openstack-nova-* ; do $i
 condrestart ; done
```

☐ 14. Create a new key pair with the **user1** credentials.

```
[root@serverX ~(keystone_admin)]# source /root/keystonerc_user1
[root@serverX ~(keystone_user1)]# nova keypair-add multi-key > /root/multi-key.pem
[root@serverX ~(keystone_user1)]# chmod 600 /root/multi-key.pem
```

☐ 15. Launch the instances using the template file found at `http://instructor.example.com/pub/materials/web.template`.

```
[root@serverX ~(keystone_user1)]# heat stack-create multi --template-
url http://instructor.example.com/pub/materials/web.template --
parameters="DBPassword=redhat;KeyName=multi-key"
```

☐ 16. Follow the progress by running **heat stack-list** until the **stack_status** shows **CREATE_COMPLETE**. You may also want to watch the instances boot using **virt-viewer** or the Horizon dashboard.

```
[root@serverX ~(keystone_user1)]# heat stack-list
+--------------------------------------+------------+-----------------
+----------------------+
| id                                   | stack_name | stack_status    |
 creation_time        |
+--------------------------------------+------------+-----------------
+----------------------+
| 7f78863c-e6ea-4d13-b31c-afe2e9197cf8 | multi      | CREATE_COMPLETE |
 2015-01-01T00:00:27Z |
+--------------------------------------+------------+-----------------
+----------------------+
```

```
[root@desktopX ~]# virsh list
 Id    Name                           State
----------------------------------------------------
 1     server1                        running
 2     instance-00000001              running
 3     instance-00000002              running

[root@desktopX ~]# virt-viewer instance-00000001 &
[root@desktopX ~]# virt-viewer instance-00000002 &
```

☐ 17. View the information about the orchestration events.

```
[root@serverX ~(keystone_user1)]# heat stack-show multi | less
[root@serverX ~(keystone_user1)]# heat event-list multi | less
```

☐ 18. If everything worked properly, the web server should be running using the database for account information. Connect to the website and enter the account information.

```
[root@serverX ~(keystone_user1)]# nova list
+--------------------------------------
+-----------------------------------------------------+--------
+------------------------------+
| ID                                   | Name
              | Status | Networks                        |
+--------------------------------------
+-----------------------------------------------------+--------
+------------------------------+
| 7890abcd-ef12-3456-7890-abcdef123456 | mu-late-36td25jio3rq-MySqlDatabaseServer-
cdflwy3i5kfd | ACTIVE | int=192.168.32.2, 172.24.X.4 |
| def12345-6789-0abc-def1-234567890abc | multi-WebServer-4oob2dy546vo
             | ACTIVE | int=192.168.32.4, 172.24.X.3 |
+--------------------------------------
+-----------------------------------------------------+--------
+------------------------------+
```

On **desktop*X*.example.com**, connect to the website. Enter **student** for the username and **student** for the password.

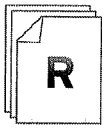
```
[root@desktopX ~]# firefox http://172.24.X.3
```

☐ 19. On **server*X*.example.com**, verify that the SSH key works to connect to each instance.

```
[root@serverX ~(keystone_user1)]# ssh -i /root/multi-key.pem 172.24.X.4
[root@mu-late-36td25jio3rq-mysqldatabasederver-cdflwy3i5kfd ~]# service mysqld
 status
mysqld (pid 1234) is running...
[root@mu-late-36td25jio3rq-mysqldatabaseserver-cdflwy3i5kfd ~]# exit
[root@serverX ~(keystone_user1)]# ssh -i /root/multi-key.pem 172.24.X.3
[root@multi-webserver-4oob2dy546vo ~]# service httpd status
httpd (pid 2345) is running...
[root@multi-webserver-4oob2dy546vo ~]# exit
```

☐ 20. Clean up by removing the stack and any other running instances.

```
[root@serverX ~(keystone_user1)]# heat stack-delete multi
```




## References

Red Hat OpenStack Installation and Configuration Guide

- Section 1.3.9. Orchestration (technical preview)

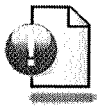
Deploy Heat and launch your first application

- `http://openstack.redhat.com/Deploy_Heat_and_launch_your_first_Application`

Heat AWS to OpenStack resource mapping

- `https://wiki.openstack.org/wiki/Heat/AWS-to-OpenStack-resource-mapping-in-templates`

# Personal Notes



# Summary

Implementing the Heat orchestration service

- Use **heat-cfn** to launch and manage instances.

# CHAPTER 12

# IMPLEMENTING THE CEILOMETER METERING SERVICE

## Introduction

| Chapter details | |
|---|---|
| **Chapter goal** | Use Ceilometer for gathering metrics as a base for billing projects and users. |
| **Chapter sections** | • Deploying the Ceilometer metering service<br>• Metering with the Ceilometer metering service |
| **Hands-on activities** | • Installing the Ceilometer metering service<br>• Configuring the Ceilometer metering service<br>• Metering with the Ceilometer metering service |
| **Chapter test** | Gathering meter information with Ceilometer |

# Deploying the Ceilometer metering service

## What is Ceilometer?

Ceilometer is a back end that provides an API and a command-line client to gather metrics to be used for customer billing, system monitoring, or alerts.

The metering service is composed of the following:

- *ceilometer-agent-compute:* an agent that runs on each compute node, and polls for resource utilization statistics.
- *ceilometer-agent-central:* an agent that runs on a central management server to poll for utilization statistics about resources not tied to instances or compute nodes.
- *ceilometer-collector:* an agent that runs on one or more central management servers to monitor the message queues. Notification messages are processed and turned into metering messages, and sent back out onto the message bus using the appropriate topic. Metering messages are written to the data store without modification.
- *Mongo database:* for storing 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.

## The difference between monitoring and metering

Monitoring is generally used to check for functionality on the overall system and to figure out if the hardware for the overall installation and usage needs to be scaled up. With monitoring, we also do not care that much if we have lost some samples in between. Metering is required for information gathering on usage as a base for billing users or projects, depending on resource usage.

### Note

The Ceilometer metering service is supported as a technical preview in the Red Hat OpenStack 3.0 (Grizzly) release.

**Workshop**

## Installing the Ceilometer metering service

Follow along with the instructor as you perform the setup tasks required to install the Ceilometer software.

We are going to deploy the Ceilometer metering service now.

☐ 1. Reset your **server*X*** virtual machine to the last saved state.

If you are in a physical classroom, log in as **root** on **desktop*X*.example.com** and reset your **server*X*** machine:

```
[root@desktopX ~]# lab-reset-vm
```

```
This will destroy the virtual machine and reset it to the original
state.
Is this ok [y/N]: y
Waiting for things to settle...
Done.
```

If you are working in the virtual training environment, ignore the preceding paragraph and use the virtual machine controls in your web browser to reset your machine to a snapshot of your **serverX** machine. Open the **state** dropdown menu and select the **Snapshots** tab. Select the **Chapter 10-2** snapshot via the radio buttons and press the **Revert to selected snapshot** button. Press the **Power On** button to start **serverX.example.com**.

☐ 2. The first step is to install the Ceilometer packages on **serverX**.

```
[root@serverX ~]# yum install -y *ceilometer* mongodb-server
```

☐ 3. Prepare the **mongodb-server** for use with Ceilometer. The **--smallfiles** option enforces a smaller default file size with **mongodb**. Add that option to the **/etc/sysconfig/mongod** file in the **OPTIONS** variable. The file should look like the following:

```
OPTIONS="--smallfiles --quiet -f /etc/mongodb.conf"
```

☐ 4. Start the service and make it persistent.

```
[root@serverX ~]# service mongod start
[root@serverX ~]# chkconfig mongod on
```

☐ 5. Source **keystonerc_admin** for the credentials.

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

☐ 6. Create the **ceilometer** user in Keystone.

```
[root@serverX ~(keystone_admin)]# keystone user-create --name ceilometer --pass
 redhat
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |                                  |
| enabled  |               True               |
|    id    | cab1234567890cab1234567890cab123 |
|   name   |            ceilometer            |
| tenantId |                                  |
+----------+----------------------------------+
```

☐ 7. Create a reseller administrator.

```
[root@serverX ~]# keystone role-create --name ResellerAdmin
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|    id    | 684fd48df02d4643ae7cb257bddb41cb |
```

```
|   name   |          ResellerAdmin          |
+----------+---------------------------------+
```

☐ 8. Add the **ceilometer** user from the **services** tenant to the **admin** role and the **ResellerAdmin** role.

```
[root@serverX ~]# keystone role-list
+----------------------------------+---------------+
|                id                |      name     |
+----------------------------------+---------------+
| 234567890abcdef1234567890abcdef1 |     Member    |
| 684fd48df02d4643ae7cb257bddb41cb | ResellerAdmin |
| 9fe2ff9ee4384b1894a90878d3e92bab |    _member_   |
| fad9876543210fad9876543210fad987 |     admin     |
+----------------------------------+---------------+
[root@serverX ~]# keystone user-role-add --tenant services --user ceilometer --
role ResellerAdmin
[root@serverX ~]# keystone user-role-add --tenant services --user ceilometer --
role admin
```

☐ 9.

```
[root@serverX ~]# openstack-config --set /etc/ceilometer/ceilometer.conf
 keystone_authtoken auth_host 192.168.0.X+100
[root@serverX ~]# openstack-config --set /etc/ceilometer/ceilometer.conf
 keystone_authtoken auth_port 35357
[root@serverX ~]# openstack-config --set /etc/ceilometer/ceilometer.conf
 keystone_authtoken auth_protocol http
[root@serverX ~]# openstack-config --set /etc/ceilometer/ceilometer.conf
 keystone_authtoken admin_tenant_name services
[root@serverX ~]# openstack-config --set /etc/ceilometer/ceilometer.conf
 keystone_authtoken admin_user ceilometer
[root@serverX ~]# openstack-config --set /etc/ceilometer/ceilometer.conf
 keystone_authtoken admin_password redhat
```

☐ 10.

```
[root@serverX ~]# openstack-config --set /etc/ceilometer/ceilometer.conf DEFAULT
 os_auth_url http://serverX.example.com:35357/v2.0
[root@serverX ~]# openstack-config --set /etc/ceilometer/ceilometer.conf DEFAULT
 os_tenant_name services
[root@serverX ~]# openstack-config --set /etc/ceilometer/ceilometer.conf DEFAULT
 os_username ceilometer
[root@serverX ~]# openstack-config --set /etc/ceilometer/ceilometer.conf DEFAULT
 os_password redhat
```

☐ 11. Start the Ceilometer services and make them persistent.

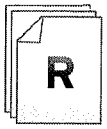
```
[root@serverX ~]# service openstack-ceilometer-compute start
[root@serverX ~]# service openstack-ceilometer-central start
[root@serverX ~]# service openstack-ceilometer-collector start
[root@serverX ~]# service openstack-ceilometer-api start
[root@serverX ~]# grep ERROR /var/log/ceilometer/*
[root@serverX ~]# chkconfig openstack-ceilometer-compute on
[root@serverX ~]# chkconfig openstack-ceilometer-central on
[root@serverX ~]# chkconfig openstack-ceilometer-collector on
[root@serverX ~]# chkconfig openstack-ceilometer-api on
```

☐ 12. Add the Ceilometer service to the service catalog and verify it has been added properly by listing all services in the catalog.

```
[root@serverX ~]# keystone service-create --name ceilometer --type metering --
description "Ceilometer Metering Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |        Ceilometer Service        |
|      id     | d2a52ad40c0a43f68368439b05f7ab3f |
|     name    |            ceilometer            |
|     type    |             metering             |
+-------------+----------------------------------+
```

☐ 13. Create the service end point for Ceilometer.

```
[root@serverX ~]# keystone endpoint-create \
 --region RegionOne \
 --service-id d2a52ad40c0a43f68368439b05f7ab3f \
 --publicurl "http://serverX.example.com:8777/" \
 --adminurl "http://serverX.example.com:8777/" \
 --internalurl "http://serverX.example.com:8777/"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  |  http://serverX.example.com:8777 |
|      id     | 487ee7a9a501444682e525b95a945312 |
| internalurl |  http://serverX.example.com:8777 |
|  publicurl  |  http://serverX.example.com:8777 |
|    region   |            RegionOne             |
|  service_id | d2a52ad40c0a43f68368439b05f7ab3f |
+-------------+----------------------------------+
```



## References

Red Hat OpenStack Installation and Configuration Guide
- Section 1.3.8. Metering (technical preview)

Ceilometer developer documentation
- `http://docs.openstack.org/developer/ceilometer/`

# Which OpenStack components have meters implemented?

Currently compute (Nova), network (Neutron), image (Glance), and volume (Cinder) have meters implemented. Once metering is turned on for a particular component, **`ceilometer meter-list`** shows all available meters per configured components that have actual samples recorded with Ceilometer.

# What type of meters are used?

There are three types of meters defined in Ceilometer:

- Cumulative: increasing over time (instance hours)
- Gauge: discrete items (floating IPs, image uploads) and fluctuating values (disk I/O)
- Delta: changing over time (bandwidth)

**Workshop**

# Configuring the Ceilometer metering service

Follow along with the instructor as you start configuring the OpenStack components for metering.

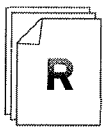
We are going to set up the OpenStack components for use with Ceilometer.

☐ 1. To set up the Nova compute service for metering, change the **DEFAULT** section of the **`/etc/nova/nova.conf`** file. Restart the service for the changes to take effect.

```
[root@serverX ~]# openstack-config --set /etc/nova/nova.conf DEFAULT
 instance_usage_audit True
[root@serverX ~]# openstack-config --set /etc/nova/nova.conf DEFAULT
 notification_driver ceilometer.compute.nova_notifier
[root@serverX ~]# service openstack-nova-compute restart
```

☐ 2. To set up Glance for metering with Ceilometer, make it use qpid as the message system for notifications.

```
[root@serverX ~]# openstack-config --set /etc/glance/glance-api.conf DEFAULT
 notifier_strategy qpid
[root@serverX ~]# service openstack-glance-api restart
```




## References

Red Hat OpenStack Installation and Configuration Guide

- Section 1.3.8. Metering (technical preview)

Ceilometer developer documentation

- `http://docs.openstack.org/developer/ceilometer/`

# Metering with the Ceilometer metering service

## The Ceilometer command-line client

With the **ceilometer** command from the *python-ceilometerclient* RPM package, the Ceilometer database can be queried for specific information.

With the **ceilometer meter-list**, we can query the available meters that already have entries in the Ceilometer database.

```
[root@serverX ~]# ceilometer meter-list
+--------------+-------+-------+-------------+---------+-------------+
| Name         | Type  | Unit  | Resource ID | User ID | Project ID  |
+--------------+-------+-------+-------------+---------+-------------+
| image        | gauge | image | ed611f(...) |         | 750aa2(...) |
| image.size   | gauge | B     | ed611f(...) |         | 750aa2(...) |
| image.update | delta | event | ed611f(...) |         | 750aa2(...) |
| image.upload | delta | event | ed611f(...) |         | 750aa2(...) |
+--------------+-------+-------+-------------+---------+-------------+
```

### Note

The **ceilometer meter-list** command will only provide output if there is some data already collected. In this case, for example, there needs to be images uploaded into OpenStack before any meters are shown.

Now we can, for example, display the recorded samples for the number of images in OpenStack with the following command:

```
[root@serverX ~]# ceilometer sample-list -m image
+-------------+-------+-------+--------+-------+----------------------------+
| Resource ID | Name  | Type  | Volume | Unit  | Timestamp                  |
+-------------+-------+-------+--------+-------+----------------------------+
| ed611f(...) | image | gauge | 1.0    | image | 2013-06-30T09:50:06.933000 |
| ed611f(...) | image | gauge | 1.0    | image | 2013-06-30T09:50:11.855000 |
| ed611f(...) | image | gauge | 1.0    | image | 2013-06-30T09:50:13.011000 |
+-------------+-------+-------+--------+-------+----------------------------+
```

Another interesting feature of the **ceilometer** client is to be able to display statistics:

```
[root@serverX ~]# ceilometer statistics -m image
+--------+---------------------+---------------------+-------+-----+-----+
| Period | Period Start        | Period End          | Count | Min | Max |
+--------+---------------------+---------------------+-------+-----+-----+
|        | 2013-06-30T09:50:06 | 2013-06-30T09:50:06 | 1     | 1.0 | 1.0 |
+--------+---------------------+---------------------+-------+-----+-----+
-----+-----+----------+---------------------+---------------------+
 Sum | Avg | Duration | Duration Start      | Duration End        |
-----+-----+----------+---------------------+---------------------+
 1.0 | 1.0 |          | 2013-06-30T09:50:06 | 2013-06-30T09:50:06 |
-----+-----+----------+---------------------+---------------------+
```

Workshop

# Metering with the Ceilometer metering service

Follow along with the instructor as you perform the setup tasks required to configure metering Glance with the Ceilometer software.

We are going to set up Ceilometer for metering Glance.

☐ 1. Source the **keystonerc_admin** file to authenticate.

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

☐ 2. Upload a new image with Glance, either with the command line or the Horizon web interface:

```
[root@serverX ~(keystone_admin)]# glance image-create --name ceilometertest --
is-public True --disk-format qcow2 --container-format bare --copy-from http://
instructor.example.com/pub/materials/small.img
```

☐ 3. Check for existing meter records with the **ceilometer meter-list** command.

```
[root@serverX ~]# ceilometer meter-list
+--------------+-------+-------+-------------+---------+-----------.-+
| Name         | Type  | Unit  | Resource ID | User ID | Project ID  |
+--------------+-------+-------+-------------+---------+-------------+
| image        | gauge | image | ed611f(...) |         | 750aa2(...) |
| image.size   | gauge | B     | ed611f(...) |         | 750aa2(...) |
| image.update | delta | event | ed611f(...) |         | 750aa2(...) |
| image.upload | delta | event | ed611f(...) |         | 750aa2(...) |
+--------------+-------+-------+-------------+---------+-------------+
```

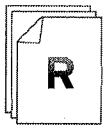
☐ 4. Take a look at the recorded samples with the **ceilometer sample-list -m image** command.

```
[root@serverX ~]# ceilometer sample-list -m image
+-------------+-------+-------+--------+-------+----------------------------+
| Resource ID | Name  | Type  | Volume | Unit  | Timestamp                  |
+-------------+-------+-------+--------+-------+----------------------------+
| ed611f(...) | image | gauge | 1.0    | image | 2013-06-30T09:50:06.933000 |
| ed611f(...) | image | gauge | 1.0    | image | 2013-06-30T09:50:11.855000 |
| ed611f(...) | image | gauge | 1.0    | image | 2013-06-30T09:50:13.011000 |
+-------------+-------+-------+--------+-------+----------------------------+
```

☐ 5. To view compiled statistics, you can use the **ceilometer statistics -m image** command.

```
[root@serverX ~]# ceilometer statistics -m image
+--------+---------------------+---------------------+-------+-----+-----+-----
+-----+
| Period | Period Start        | Period End          | Count | Min | Max | Sum |
 Avg |
+--------+---------------------+---------------------+-------+-----+-----+-----
+-----+
|        | 2013-06-30T09:50:06 | 2013-06-30T09:50:06 | 1     | 1.0 | 1.0 | 1.0 |
 1.0 |
```

```
+--------+--------------------+--------------------+-------+-----+-----+-----
+-----+
----------+--------------------+--------------------+
 Duration | Duration Start     | Duration End       |
----------+--------------------+--------------------+
          | 2013-06-30T09:50:06 | 2013-06-30T09:50:06 |
----------+--------------------+--------------------+
```




## References

Red Hat OpenStack Installation and Configuration Guide

- Section 1.3.8. Metering (technical preview)

Ceilometer developer documentation

- `http://docs.openstack.org/developer/ceilometer/`

# Chapter test

Case Study

## Gathering meter information with Ceilometer

**Lab overview:** In this lab you will set up an instance in Horizon and gather certain metrics around it.

**Success criteria:** Successfully set up an instance within Horizon and gather certain metrics around it.

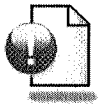
***Before you begin...***

Create a new user called **meteruser** with the password **redhat**. Create a new tenant called **meterproject**. Add the **meteruser** to the **meterproject** project and the **Member** role. As user **meteruser**, create an instance with the name **meterinstance** using the **ceilometertest** image.

**Lab outline:** Explore various meters from the output of **ceilometer meter-list** by looking at statistics and samples of the *instance*, *memory*, and *vcpu* meters.

*How would you address the case study described above? Take notes on your process in the space below and then implement it.*

# Personal Notes



# Summary

Deploying the Ceilometer metering service

- Use Ceilometer to gather metrics as a base for billing.

Metering with the Ceilometer metering service

- In this section we will do some metering with Ceilometer.

# CHAPTER 13

# THE FUTURE DIRECTION OF RED HAT OPENSTACK

## Introduction

| Chapter details | |
|---|---|
| **Chapter goal** | Learn about the future direction of Red Hat OpenStack. |
| **Chapter sections** | • The future of OpenStack |
| **Hands-on activities** | None |
| **Chapter test** | None |

# The future of OpenStack

Four projects will be incubated into Icehouse:

- *Trove*: a Database as a Service (DBaaS) for OpenStack to provide scalable and reliable cloud database capabilities as a service, provisioning functionality for both relational and non-relational database engines.
- *Ironic*: a bare-metal provisioning driver for OpenStack.
- *Marconi*: a queuing and notification service to enable development of complex web applications on top of OpenStack.
- *Savannah*: a component to provide simple means for provisioning a Hadoop cluster on top of OpenStack.

Icehouse is the "I" release of OpenStack and will be released in spring 2014.

RDO is the OpenStack community project that aims to provide OpenStack for Fedora and Red Hat Enterprise Linux. RDO will track OpenStack upstream more closely than Red Hat OpenStack, and is similar to Fedora in the Fedora/Red Hat Enterprise Linux life cycle. RDO can be found at `http://openstack.redhat.com`, and includes documentation and forums.


## References

Projects incubated in the Icehouse release:

- Database as a service (Trove) - `https://wiki.openstack.org/wiki/Trove`
- Bare-metal deployment (Ironic) - `https://wiki.openstack.org/wiki/Ironic`
- OpenStack on OpenStack (Marconi) - `https://wiki.openstack.org/wiki/Marconi`
- OpenStack on OpenStack (Savannah) - `https://wiki.openstack.org/wiki/Savannah`

Release notes for Icehouse:

- `https://wiki.openstack.org/wiki/ReleaseNotes/Icehouse`

Icehouse release schedule:

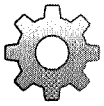
- `https://wiki.openstack.org/wiki/Icehouse_Release_Schedule`

Overview about Icehouse core components:

- `https://wiki.openstack.org/wiki/Programs`

RDO

- `http://openstack.redhat.com`



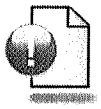
Fill-in-the-Blank Quiz

## OpenStack code names and projects

For each of the following statements, fill in the blanks:

1. The _____ is the **H** release of OpenStack and was released in fall 2013.
2. The _____ is the **I** release of OpenStack and will be released in spring 2014.
3. The _____ is the Database as a Service (DBaaS) project that will be incubated into the **I** release.
4. The _____ is the bare-metal driver that will be incubated into the **I** release.
5. The _____ is the queuing and notification service that will be incubated into the **I** release.
6. The _____ is a project to simplify provisioning of a Hadoop cluster that will be incubated into the **I** release.

# Personal Notes



# Summary

The future of OpenStack

- Understand the future of OpenStack.

# CHAPTER 14

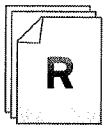
# COMPREHENSIVE REVIEW

## Introduction

| Chapter details | |
|---|---|
| **Chapter goal** | |
| **Chapter sections** | |
| **Hands-on activities** | |
| **Chapter test** | |

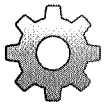
# Comprehensive review

Review the content of the previous chapters, then start the following case study.




## References

Red Hat OpenStack Getting Started Guide



Case Study

# Comprehensive review

**Lab overview:** Review Red Hat OpenStack installation and configuration.

**Success criteria:** Red Hat OpenStack cloud running according to specifications.

***Before you begin...***
Save any work on desktop*X* and server*X* that you want to keep because you will reinstall these machines. Reboot desktop*X* and choose the "Reinstall Desktop *X*" option in grub. Enter a password of **redhat** to begin the installation.

If you are working in the virtual training environment, ignore the preceding paragraph and use the virtual machine controls in your web browser to reset your machine to a snapshot of your **desktop*X*** and **server*X*** machine. On the **desktop*X*** tab in your browser, open the **state** dropdown menu and select the **Snapshots** tab. Select the **Initial Snapshot** snapshot via the radio buttons and press the **Revert to selected snapshot** button. Press the **Power On** button to start **desktop*X*.example.com**. Repeat the process for **server*X*.example.com**, again choosing the **Initial Snapshot** snapshot.

If you are using the Red Hat Online Learning environment, run the **lab-clean-desktop** command to clean desktop*X*.example.com and reset server*X*.example.com.

**Lab outline:** The itemized lists define the requirements necessary to install and configure Red Hat OpenStack. server*X* will provide most services, but desktop*X* will provide the Cinder service and will be the Nova compute node. You will launch two instances in Red Hat OpenStack.

**Install Red Hat OpenStack as follows:**

*Configure **server*X*.example.com** with the following settings:*

- Generate SSH keys to ease installation.
- Configure NTP using **192.168.0.254** as the NTP server for all machines.
- Create a private network named **int** and a subnet named **subint** using the **192.168.32.0/24** network. Set the gateway for this network to **192.168.32.1**.
- Create a public network named **ext** and a subnet named **subext** using the **172.24.*X*.0/24** network. The gateway for this network is **172.24.*X*.254**. This network must be configured for the floating IP addresses.
- Use a VLAN range of **1000-2999** and configure **br-eth1** as the OVS bridge for communication. **desktop*X*.example.com** will use the **br101** network device to communicate on **br-eth1**. **server*X*.example.com** will use the **eth1** network device to communicate on **br-eth1**.
- Configure Horizon to accept SSL connections.

*Configure **desktop*X*.example.com** with the following settings:*

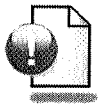
- Enable the Cinder service using the pre-existing **cinder-volumes** volume group.
- Enable the Nova compute service on **desktop*X***.

**Configure Red Hat OpenStack as follows:**

- Create a project named **project1** using a description of **Project for project1**. Set the quota for this project to four VCPUs, 4096MB RAM and two floating IP addresses.
- Create a new flavor named **m2.tiny** which includes one VPCU, 1024MB RAM, a 20GB root disk, a 2GB ephemeral disk and a 512MB swap disk.
- Create a new user named **user1** with an email address of *root@desktopX.example.com*. Set the password to **redhat** and include this user as a member of the **project1** project.
- Create a new user named **adm1** with an email address of *root@desktopX.example.com*. Set the password to **redhat** and include this user as an admin of the **project1** project.
- In the **project1** project, create a new image named **small** using the QCOW2 image located at `http://instructor.example.com/pub/materials/small.img`. Set no minimum disk, minimum 512MB RAM, and make the image public.
- In the **project1** project, create a new image named **web** using the QCOW2 image located at `http://instructor.example.com/pub/materials/web.img`. Set no minimum disk, minimum 1024MB RAM, and make the image public.
- In the **project1** project, allocate two floating IP addresses.
- In the **project1** project, create a new security group named **sec1** with a description of **Web and SSH**. Allow **TCP/22** and **TCP/443** from **CIDR 0.0.0.0/0**, and **TCP/80** from the **sec1** source group.
- In the **project1** project, create an SSH key pair named **key1**. Save the private key to **/home/student/Downloads/key1.pem** on **desktop*X*.example.com**.
- In the **project1** project, launch a new instance named **small** using the **small** image and the **m1.tiny** flavor. Include the **key1** key pair and the **sec1** security group. Associate the **172.24.*X*.2** floating IP address.
- In the **project1** project, launch a new instance named **web** using the **web** image and the new **m2.tiny** flavor. Include the **key1** key pair and the **sec1** security group. Associate the **172.24.*X*.3** floating IP address.
- In the **project1** project, create a new 10GB volume named **vol1** and attach this volume to the **web** instance.

*How would you address the case study described above? Take notes on your process in the space below and then implement it.*

# Personal Notes



# Summary

Comprehensive review

- Review Red Hat OpenStack configuration and administration.

# APPENDIX A
# SOLUTIONS

# Chapter 1: Introducing Red Hat OpenStack architecture

## Red Hat OpenStack architecture overview

- For each of the following statements, fill in the blanks:

1. The Nova compute service provides virtualization using libvirtd, qemu, and kvm.
2. The Glance service provides images that are used as templates to build instances.
3. The OpenStack networking service provides networking capabilities using a pluggable architecture.
4. The Cinder service provides persistent volumes for instances.
5. The Swift service provides object storage.
6. The Keystone service provides authentication and authorization.
7. The Horizon service provides a dashboard for managing OpenStack.
8. A cloud controller coordinates the Red Hat OpenStack cloud using the Qpid messaging service (AMQP).
9. Server or instance are the names used for a virtual machine in OpenStack.

Performance Checklist

## Explore the classroom environment

**Lab overview:** Become oriented to the initial classroom environment.

**Success criteria:** Students will understand their system configurations.

***Before you begin...***
Login information for your Red Hat Enterprise Linux system(s):

- Username: **student** Password: **student**
- Username: **root** Password: **redhat**

**Lab outline:** The checklist defines a list of system information you need to look up or verify (host name, IP addresses, package repositories, etc.).

☐ 1. Identify the Red Hat Enterprise Linux physical machine

☐ 1.1. In the classroom, you should have one physical system, **desktop*X***, that is preinstalled with Red Hat Enterprise Linux 6.5.

In the virtual classroom, your **desktop*X*** is a virtual machine.

☐ 1.2. Log into the physical system **desktop*X*** using the username **student** with the password **student**. Open a terminal window with a shell prompt.

☐ 1.3. At the prompt of the **desktop*X*** system, run the **hostname** command to see what your machine's host name is. Write it down.

*Hostname:* desktop*X*.example.com

```
[student@desktopX ~]$ hostname
desktopX.example.com
```

where *X* is your desktop number.

☐ 1.4. At the prompt of the **desktop*X*** system, run the **dig** command on your machine's host name to determine your expected IPv4 address. Write it down.

*IPv4 address:* 192.168.0.*X*

```
[student@desktopX ~]$ dig desktopX.example.com
...
;; ANSWER SECTION:
desktopX.example.com.    86400   IN      A    192.168.0.X
...
```

The IPv4 address is **192.168.0.*X*** (where *X* is your desktop number).

☐ 1.5. At the prompt of the **desktop*X*** system, run the **ip addr show** command to see what interface your **desktop*X*** machine's IPv4 address is attached to. Also, find the other interface that has a different private IP address. This private IP address will be used to connect to the private IP address range that the instances use. Write it down.

If you are in a virtual classroom, you will find that the other interface is simply a second UP interface with no IP assigned yet. Write down the name of that interface as the Private bridge name.

*Public bridge name:* br100

*Private bridge name:* br101 (or eth1 in VT)

*Private IP address:* 192.168.32.250 (or unassigned in VT)

```
[student@desktopX ~]$ ip addr show
...
3: br100: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
 UNKNOWN
    link/ether 00:11:22:33:aa:bb brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.X/24 brd 192.168.0.255 scope global br100
...
6: br101: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
 UNKNOWN
    link/ether 52:54:00:e0:ce:c3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.32.250/24 brd 192.168.32.255 scope global br101
 ...
```

The IPv4 addresses are **192.168.0.*X*** (where *X* is your desktop number) and **192.168.32.250**. Note that on this system, a physical bridge device was defined, named **br100**, in support of virtual machines directly connecting to the physical network (eth0). A second bridge named **br101** was defined and is not tied to any physical NIC, but is used to communicate with the virtual machine. In the virtual environment, you will find an interface named **eth1** with no IP assigned rather than the second bridge named **br101**.

☐ 2. Verify yum configuration on physical system
Your **desktop*X*** system may need to get software packages from the repositories on **instructor.example.com**. Review the yum repositories, and write down the names of the different repositories that are currently configured.

```
[student@desktopX ~]$ yum repolist
Loaded plugins: product-id, refresh-packagekit, security, subscription-manager
[Errno 13] Permission denied: '/etc/pki/entitlement'
repo id              repo name                                          status
OpenStack            OpenStack Repository                                 515
base                 Red Hat Enterprise Linux 6Server - x86_64          3,690
updates              Red Hat Enterprise Linux 6Server - x86_64 Errata       5
repolist: 4,210
```

☐ 3. Apply updates
Become the **root** user on your **desktop*X*** system and update the system with the updates provided in class.

```
[student@desktopX ~]$ su -
Password: redhat
[root@desktopX ~]# yum update -y
```

☐ 4. Identify the Red Hat Enterprise Linux virtual machine

☐ 4.1. Log into your server*X* machine as **root** (with the password **redhat**).

If you are in a physical classroom, open a new terminal and become root (**su -**). Run the command **virt-viewer server*X***. This will open a window through which you can log into the server*X* virtual machine. Log into server*X* as **root** (with a password of **redhat**).

If you are working in the virtual training environment, ignore the preceding paragraph and use the classroom controls in your web browser to connect to server*X*. Log into server*X* as **root** (with the password **redhat**).

If you do not have a **server*X*** virtual machine, or have trouble accessing it, please notify your instructor.

☐ 4.2. At the prompt on your **server*X*** virtual machine, run the **hostname** command to see what your machine's host name is. Write it down.

*Hostname:* server*X*.example.com

```
[root@serverX ~]# hostname
```

```
serverX.example.com
```

where *X* is your desktop number.

☐ 4.3. At the prompt on your **serverX** virtual machine, run the **dig** command on your machine's host name to determine your expected IPv4 address. Write it down.

*IPv4 address:* 192.168.0.*X+100*

```
[root@serverX ~]# dig serverX.example.com
...
;; ANSWER SECTION:
serverX.example.com.    86400   IN      A    192.168.0.X+100
...
```

The IPv4 address is **192.168.0.*X+100*** (where *X* is your desktop number).

☐ 4.4. At the prompt on your **serverX** virtual machine, run the **ip addr show** command to see what interface your machine's IPv4 address is attached to. Write it down.

*Interface name:* eth0

```
[root@serverX ~]# ip addr show
...
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen
 1000
    link/ether 52:54:00:00:00:XX brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.X+100/24 brd 192.168.0.255 scope global eth0
...
```

The IPv4 address is **192.168.0.*X+100*** (where *X* is your desktop number) on eth0.

☐ 4.5. Notice that your **serverX** virtual machine has two NICs in the output above. Write down the interface name of the second NIC.

*Interface name:* eth1

```
[root@serverX ~]# ip addr show
...
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen
 1000
...
```

☐ 5. Verify yum configuration on virtual machine

Your **serverX** system may need to get software packages from the repositories on **instructor.example.com**. Review the yum repositories, and write down the names of the different repositories that are currently configured on **serverX.example.com**.

```
[root@serverX ~]# yum repolist
...
repo id              repo name                                        status
OpenStack            OpenStack Repository                               515
```

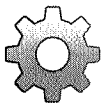
```
base                  Red Hat Enterprise Linux 6Server - x86_64           3,690
updates               Red Hat Enterprise Linux 6Server - x86_64 Errata        5
repolist: 4,210
```

☐ 6. Apply updates

Update your **server*X*** system with the updates provided in class.

```
[root@serverX ~]# yum update -y
```

# Chapter 2: Installing Red Hat OpenStack



Workshop

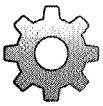
## Installing Red Hat OpenStack with packstack

*The solutions for this lab are included in the main text.*

Workshop

## Creating a tenant in Horizon

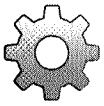
*The solutions for this lab are included in the main text.*



Workshop

## Creating a flavor in Horizon

*The solutions for this lab are included in the main text.*



Workshop

## Creating a user in Horizon

*The solutions for this lab are included in the main text.*

Workshop

## Launching an Instance in Horizon

*The solutions for this lab are included in the main text.*

Workshop

## Installing Foreman

*The solutions for this lab are included in the main text.*

Workshop

# Deploying OpenStack with Foreman

*The solutions for this lab are included in the main text.*

Case Study

# Installing Red Hat OpenStack

***Before you begin...***

Reset your **server*X*** virtual machine.

If you are in a physical classroom, log in as **root** on **desktop*X*.example.com** and reset your **server*X*** machine:

```
[root@desktopX ~]# lab-reset-vm
This will destroy the virtual machine and reset it to the last saved state.
Is this ok [y/N]: y
Waiting for things to settle...
Done.
```

If you are working in the virtual training environment, ignore the preceding paragraph and use the virtual machine controls in your web browser to reset your machine to a snapshot of your **server*X*** machine. Open the **state** dropdown menu and select the **Snapshots** tab. Select **Initial Snapshot** via the radio buttons and click the **Revert to selected snapshot** button. Press the **Power On** button to start **server*X*.example.com**.

You must disable the hosts in Foreman. Log in to the Foreman web interface (`https://desktopX.example.com/`) as the **admin** user with a password of **redhat**. Browse to the **Hosts** tab. In the **desktop*X*.example.com** row, open the drop-down menu and choose **Delete**. Press the **OK** button. Do the same for **server*X*.example.com**.

You must also disable the OpenStack services running on **desktop*X*.example.com** configured by Foreman.

```
[root@desktopX ~]# service openstack-ceilometer-compute stop
[root@desktopX ~]# chkconfig openstack-ceilometer-compute off
[root@desktopX ~]# service openstack-nova-compute stop
[root@desktopX ~]# chkconfig openstack-nova-compute off
[root@desktopX ~]# service neutron-openvswitch-agent stop
[root@desktopX ~]# chkconfig neutron-openvswitch-agent off
```

After you have reset your virtual machine, **ssh** to **server*X*.example.com**. Update the software. Log into **server*X*.example.com** and install the packages necessary for **packstack**. Configure Red Hat OpenStack on **server*X*.example.com** according to the following table.

**Instance parameters**

| *Category* | *Parameter/value* |
|---|---|
| Red Hat OpenStack information | • Configure SSH keys<br>• Use **192.168.0.254** for the NTP server (unless you are using the Red Hat Online Learning environment) |

| *Category* | *Parameter/value* |
|---|---|
| | • Configure Horizon to use SSL<br>• Project name: **tenant1**<br>• User account name: **user1**<br>• User account email: **root@desktop*X*.example.com**<br>• User account password: **redhat** |
| Image information | • Image name: **small**<br>• Image location: `http://instructor.example.com/pub/materials/small.img`<br>• Image format: QCOW2<br>• Image settings: No minimum disk, no minimum RAM, public |
| Private network information | • Private network name: **private**<br>• Private subnet name: **privatesub**<br>• Private network range: **192.168.32.0/24**<br>• Private IP version: IPv4<br>• Private gateway IP: *Leave blank, but not disabled* |
| Public network information | • Public network name: **public**<br>• Public subnet name: **publicsub**<br>• Public network range: **172.24.*X*.0/24**<br>• Public IP version: IPv4<br>• Public gateway IP: **172.24.*X*.254**<br>• Public DHCP: disabled<br>• Public allocation pools: **172.24.*X*.1,172.24.*X*.100** |
| Router information | • Router name: **router1**<br>• Set the public network as an external network<br>• Assign the public network as the gateway for the router<br>• Add an interface for the private subnet to the router |
| Security group information | • Security group name: **secgrp**<br>• Security group description: **SSH and Web** |

| *Category* | *Parameter/value* |
|---|---|
| | • Allow SSH from CIDR 0.0.0.0/0<br>• Allow HTTPS from CIDR 0.0.0.0/0<br>• Allow HTTP from this source group |
| Instance information | • Instance name: **small**<br>• Instance flavor: **m1.tiny**<br>• Instance Boot Source: **Boot from image.**<br>• Instance image: **small**<br>• Instance keypair: **key2**<br>• Instance security group: **secgrp**<br>• Instance floating IP address: **172.24.*X*.2** |
| Volume information | • Volume name: **myvol2**<br>• Volume description: **myvol2 volume**<br>• Volume size (GB): **2**<br>• Volume snapshot name: **myvol2-snap1**<br>• Volume snapshot description: **myvol2-snap1** |

If you need to troubleshoot your installation, you may want to disable debugging in Nova (**debug = False** in **/etc/nova/nova.conf**) and restart the **openstack-nova-*** services.

1. On **desktop*X*.example.com**, reset your **server*X*** virtual machine.

   If you are in a physical classroom login as **root** on **desktop*X*.example.com** and reset your **server*X*** machine:

   ```
   [root@desktopX ~]# lab-reset-vm
   ```

   If you are working in the virtual training environment, ignore the preceding paragraph and use the virtual machine controls in your web browser to reset your machine to a snapshot of your **server*X*** machine. Open the **state** drop-down menu and select the **Snapshots** tab. Select the **Initial Snapshot** snapshot via the radio buttons and press the **Revert to selected snapshot** button. Press the **Power On** button to start **server*X*.example.com**.

2. **ssh** to **server*X*.example.com**. Update the software:

   ```
   [student@desktopX ~]# ssh root@serverX.example.com
   [root@serverX ~]# yum update -y
   ```

3. Install the *openstack-packstack* package.

   ```
   [root@serverX ~]# yum install -y openstack-packstack
   ```

4. Generate SSH keys as **root** on server*X*.

```
[root@serverX ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): Enter
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
...
```

5. Generate an answer file with **packstack**.

```
[root@serverX ~]# packstack --gen-answer-file /root/answers.txt
```

6. Edit the **/root/answers.txt** and change the following items:

```
CONFIG_SSH_KEY=/root/.ssh/id_rsa.pub
CONFIG_NTP_SERVERS=192.168.0.254
CONFIG_HORIZON_SSL=y
```

If you are using the Red Hat Online Learning environment, do not set the NTP server.

7. Configure Red Hat OpenStack using the answer file.

```
[root@serverX ~]# packstack --answer-file /root/answers.txt
Welcome to Installer setup utility

Installing:
Clean Up...                                  [DONE]
Setting up ssh keys...root@192.168.0.X+100's password: redhat
...
```

8. Login to the Horizon Dashboard.

Once the installation has successfully completed, point your web browser to `https://serverX.example.com/dashboard`. You can login as **admin** with the password found in **/root/keystonerc_admin** as the **OS_PASSWORD** variable.

9. Go to the **Admin** tab in the left pane and click on the **Projects** link. Press the **Create Project** button and enter the name of the project as above. Press the **Create Project** button.

10. Go to the **Admin** tab in the left pane and click on the **Users** link. Press the **Create User** button and enter the information as above. Press the **Create User** button.

Sign out from the **admin** account and login to the **user1** account using the password above.

11. In the **Project** tab on the left pane, select the **Images & Snapshots** link. Click on the **Create Image** button. Enter the information above and press the **Create Image** button.

12. In the **Project** tab on the left pane, select the **Networks** link. Press the **Create Network** button. Enter the private network name as above. Click on the **Subnet** tab. Enter the private subnet information as above. Press the **Create** button.

13. Press the **Create Network** button again. Enter the public network name as above. Browse to the **Subnet** tab. Enter the public subnet information as above. Browse to the **Subnet Detail** tab. Enter the public subnet detail information as above. Press the **Create** button.

14. In the **Project** tab on the left pane select the **Routers** link. Press the **Create Router** button. Enter the router name as above and press the **Create** button.

15. Sign out as the **user1** user and sign in as **admin**. In the **Admin** tab in the left pane, click on the **Networks** link. Click on the **Edit Network** button in the public (**public**) network row. Select the **External Network** checkbox and press the **Save Changes** button.

16. Sign out as the **admin** user and sign in as **user1**. In the **Project** tab in the left pane, click on the **Routers** link. Press the **Set Gateway** button in the **router1** row. In the **External Network** menu, choose **public**. Press the **Set Gateway** button.

17. Click on the **router1** link. Press the **Add Interface** button. In the **Subnet** menu, select **private: 192.168.32.0/24 (privatesub)**. Press the **Add Interface** button.

    Verify that both networks are attached to the router. In the **Project** tab in the left pane, click on the **Network Topology** link. **private** and **public** should both connect to the **router1** router.

18. Allocate a floating IP address. In the **Project** tab in the left pane, click the **Access & Security** link. Choose the **Floating IPs** tab and click on the **Allocate IP to Project** button. Use the *public* pool and click on the **Allocate IP** button to allocate the above given floating IP address.

19. In the **Project** tab in the left pane, select the **Access & Security** link. Choose the **Security Group** tab and click on the **Create Security Group** button. Enter the name and description as above. Click on the **Create Security Group** button.

    Click the **Edit Rules** button for the security group. Click the **Add Rule** button. Choose **SSH** in the **Rule** drop-down menu, and leave **Remote** and **CIDR** as default. Click the **Add** button. Click the **Add Rule** button again. Choose **HTTPS** in the **Rule** drop-down menu and click the **Add** button. Click the **Add Rule** button once more. Choose **HTTP** in the **Rule** drop-down menu. Choose **Security Group** in the **Remote** drop-down menu. Choose **secgrp** as the **Security Group** and **IPv4** as the **Ether Type**. Press the **Add** button.

20. In the **Project** tab in the left pane, select the **Access & Security** link. Choose the **Keypairs** tab and press the **Create Keypair** button. Enter the name as above and press the **Create Keypair** button. Save the file to the default location, which should be in **/home/student/Downloads/** on **desktopX.example.com**.

21. Open a **root** terminal on **serverX.example.com**, and configure the **br-ex** network device configuration file.

```
[root@serverX ~]# cp /etc/sysconfig/network-scripts/ifcfg-eth0 /root/
[root@serverX ~]# cp /etc/sysconfig/network-scripts/ifcfg-eth0 /etc/sysconfig/
network-scripts/ifcfg-br-ex
```

    If you are in a physical classroom remove everything but the **DEVICE**, **HWADDR** and **ONBOOT** settings from **/etc/sysconfig/network-scripts/ifcfg-eth0** so that it looks like the following:

```
DEVICE=eth0
HWADDR=52:54:00:00:00:XX
ONBOOT=yes
```

If you are in a virtual classroom remove everything but the **DEVICE** and **ONBOOT** settings from **/etc/sysconfig/network-scripts/ifcfg-eth0** so that it looks like the following:

```
DEVICE=eth0
ONBOOT=yes
```

In the **/etc/sysconfig/network-scripts/ifcfg-br-ex** file, remove the **HWADDR** line if present and change the device name to **br-ex**. Make sure the **/etc/sysconfig/network-scripts/ifcfg-br-ex** file contains the following:

```
DEVICE=br-ex
IPADDR=192.168.0.X+100
PREFIX=24
GATEWAY=192.168.0.254
DNS1=192.168.0.254
SEARCH1=example.com
ONBOOT=yes
```

22. Once you have verified the network files contain the correct information, add the **eth0** network device to the **br-ex** bridge and restart the network. Make sure you perform the following on a single line (as shown) so you do not lose access to the machine.

```
[root@serverX ~]# ovs-vsctl add-port br-ex eth0 ; service network restart
```

23. Back in the Horizon web interface, browse to the **Project** tab in the left pane and select the **Instances** link. Press the **Launch Instance** button. In the **Details** tab, enter the image, name and flavor as above. In the **Instance Boot Source** menu, choose **Boot from image.** In the **Imiage Name** menu, choose the **small** image. Browse to the **Access & Security** tab. Enable the keypair and security group listed above, and deselect the **default** security group. In the **Networking** tab, press the **+** button next to the **private** network. Press the **Launch** button.

    Once the networking has been created for the instance, open the **Actions (More)** drop-down menu and select **Associate Floating IP**. Choose the **172.24.X.2** floating IP address, and choose the **small** instance, then press the **Associate** button.

24. Wait until the instance has finished booting. For a simple networking verification of our setup try to **ssh** to 172.24.X.2 from desktopX.example.com.

```
[student@desktopX ~]# chmod 600 /home/student/Downloads/key2.pem
[student@desktopX ~]# rm -f ~/.ssh/known_hosts
[student@desktopX ~]# ssh -i /home/student/Downloads/key2.pem root@172.24.X.2
The authenticity of host '172.24.X.2 (172.24.X.2)' can't be established.
RSA key fingerprint is aa:bb:cc:dd:ee:ff:00:11:22:33:44:55:66:77:88:99.
Are you sure you want to continue connecting (yes/no)?  yes
Warning: Permanently added '172.24.X.2' (RSA) to the list of known hosts.
[root@host-192-168-32-2 ~]#
```

25. Now we want to create the volume **myvol2**!

    Click on the **Volumes** link in the left pane and then click on the **Create Volume** button. Enter the Volume name, size and description as given above and click the **Create Volume** button.

26. Now we create a snapshot named **myvol2-snap1** of the volume!

    While still in the **Volume** section on the left pane, click on the **More** drop down menu on the right side of the row that describes the new volume and select **Create Snapshot.** Enter the snapshot and description as given in the table above and click the **Create Snapshot** button.

27. Now we attach the volume **myvol2** to the running instance!

    Be sure you are still in the **Volumes** section and click on the **Edit Attachments** button. Select the instance we just created with the **Attach to Instance** drop down menu, then press the **Attach Volume** button.

# Chapter 3: Implementing the Qpid message broker

Workshop

## Installing and securing the Qpid message broker

*The solutions for this lab are included in the main text.*

# Chapter 4: Implementing the Keystone identity service

Workshop

## Deploying the Keystone identity service

*The solutions for this lab are included in the main text.*

Workshop

## Creating the Keystone admin user

*The solutions for this lab are included in the main text.*

Case Study

## Adding a new user to Keystone

Create a new user with the **keystone** command according to the following specifications:

- The username is **myuser** with a password of **redhat**.
- The user is part of the **myopenstack** tenant.
- The user is attached to the **Member** role.

For easier testing, create a **keystonerc_myuser** file in **root**'s home directory. Verify the user exists and the **keystonerc_myuser** works by getting a token (**keystone token-get**).

1. Source the **admin** Keystone information if you have not done so already.

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

2. Create the **myuser** user in Keystone.

```
[root@serverX ~(keystone_admin)]# keystone user-create --name myuser --pass redhat
+----------+------------------------------------------+
| Property |                  Value                   |
+----------+------------------------------------------+
|  email   |                                          |
| enabled  |                   True                   |
|    id    |    90abcdef1234567890abcdef12345678      |
|   name   |                  myuser                  |
| password |  $6$rounds=40000$hbSKzWx2djNyw83i$q4U...  |
| tenantId |                                          |
+----------+------------------------------------------+
```

3. Add the **Member** role.

```
[root@serverX ~(keystone_admin)]# keystone role-create --name Member
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|    id    | 234567890abcdef1234567890abcdef1 |
|   name   |              Member              |
+----------+----------------------------------+
```

4. Add the **myopenstack** tenant.

```
[root@serverX ~(keystone_admin)]# keystone tenant-create --name myopenstack
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |                                  |
|   enabled   |               True               |
|      id     | 890abcdef1234567890abcdef1234567 |
|     name    |           myopenstack            |
+-------------+----------------------------------+
```

5. Associate the user from the **myopenstack** tenant with the **Member** role.

```
[root@serverX ~(keystone_admin)]# keystone user-role-add --user myuser --role Member
 --tenant myopenstack
[root@serverX ~(keystone_admin)]# keystone user-role-list --user myuser --tenant
 myopenstack
+----------------------------------+--------+----------------------------------
+----------------------------------+
|                id                |  name  |             user_id              |
       tenant_id             |
+----------------------------------+--------+----------------------------------
+----------------------------------+
| 234567890abcdef1234567890abcdef1 | Member | 90abcdef1234567890abcdef12345678
 | 890abcdef1234567890abcdef1234567 |
+----------------------------------+--------+----------------------------------
+----------------------------------+
```

6. Create a corresponding **/root/keystonerc_myuser** file with the following information:

```
export OS_USERNAME=myuser
export OS_TENANT_NAME=myopenstack
export OS_PASSWORD=redhat
export OS_AUTH_URL=http://serverX.example.com:5000/v2.0/
export PS1='[\u@\h \W(keystone_myuser)]\$ '
```

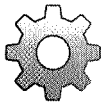
7. Verify the user exists.

   Since listing users as a non-admin user will not work, get a token to test if the user exists and our **keystonerc_myuser** file is correct.

```
[root@serverX ~(keystone_admin)]# source /root/keystonerc_myuser
[root@serverX ~(keystone_myuser)]# keystone token-get --wrap 60
+-----------+----------------------------------+
|  Property |              Value               |
+-----------+----------------------------------+
```

```
| expires   |        2013-03-19T13:29:37Z      |
| id        | 04b23f424b56440fb39378b844a754fe |
...
| tenant_id | 890abcdef1234567890abcdef1234567 |
| user_id   | 90abcdef1234567890abcdef12345678 |
+-----------+----------------------------------+
```

# Chapter 5: Implementing the Swift object storage service



Workshop

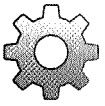
## Installing the Swift object storage service

*The solutions for this lab are included in the main text.*

Workshop

## Deploying a Swift storage node

*The solutions for this lab are included in the main text.*



Workshop

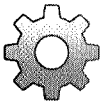
## Configuring Swift object storage service rings

*The solutions for this lab are included in the main text.*

Workshop

## Deploying the Swift object storage proxy service

*The solutions for this lab are included in the main text.*



Workshop

## Validating Swift object storage

*The solutions for this lab are included in the main text.*

# Chapter 6: Implementing the Glance image service

**Workshop**

## Deploying the Glance image service

*The solutions for this lab are included in the main text.*

**Workshop**

## Using Glance to upload a system image

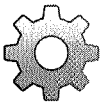
*The solutions for this lab are included in the main text.*

# Chapter 7: Implementing the Cinder Block Storage Service

Performance Checklist

## Installing the Cinder block storage service and managing volumes

*The solutions for this lab are included in the main text.*



Workshop

## Adding a Red Hat storage volume to Cinder

*The solutions for this lab are included in the main text.*

# Chapter 8: Implementing the OpenStack networking service

**Workshop**

## Installing OpenStack networking

*The solutions for this lab are included in the main text.*

**Workshop**

## Configuring OpenStack networking

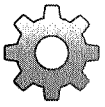
*The solutions for this lab are included in the main text.*

# Chapter 9: Implementing the Nova compute and Nova controller services

Workshop

## Installing Nova compute and Nova controller

*The solutions for this lab are included in the main text.*



Workshop

## Deploying instances using the command line

*The solutions for this lab are included in the main text.*

# Chapter 10: Implementing an additional Nova compute node

Workshop

## Rebuilding Red Hat OpenStack all-in-one

*The solutions for this lab are included in the main text.*

Workshop

## Configuring OpenStack networking on the Nova controller node

*The solutions for this lab are included in the main text.*

Performance Checklist

## Managing Nova compute nodes

*The solutions for this lab are included in the main text.*

Workshop

## Configuring OpenStack networking on the Nova compute node

*The solutions for this lab are included in the main text.*

Workshop

## Preparing and launching an instance

*The solutions for this lab are included in the main text.*

# Chapter 11: Implementing the Heat orchestration service

Workshop

## Implementing the Heat orchestration service

*The solutions for this lab are included in the main text.*

# Chapter 12: Implementing the Ceilometer metering service

Workshop

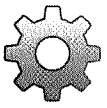
## Installing the Ceilometer metering service

*The solutions for this lab are included in the main text.*

Workshop

## Configuring the Ceilometer metering service

*The solutions for this lab are included in the main text.*



Workshop

## Metering with the Ceilometer metering service

*The solutions for this lab are included in the main text.*

Case Study

## Gathering meter information with Ceilometer

**Lab overview:** In this lab you will set up an instance in Horizon and gather certain metrics around it.

**Success criteria:** Successfully set up an instance within Horizon and gather certain metrics around it.

***Before you begin...***
Create a new user called **meteruser** with the password **redhat**. Create a new tenant called **meterproject**. Add the **meteruser** to the **meterproject** project and the **Member** role. As user **meteruser**, create an instance with the name **meterinstance** using the **ceilometertest** image.

**Lab outline:** Explore various meters from the output of **ceilometer meter-list** by looking at statistics and samples of the *instance*, *memory*, and *vcpu* meters.

1. Source the **keystonerc_admin** file.

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

2. Create a new user called **meteruser** with the password **redhat** either on the command-line or in the Horizon dashboard.

```
[root@serverX ~]# keystone user-create --name meteruser --pass redhat
```

3. Create a new tenant called **meterproject**.

```
[root@serverX ~]# keystone tenant-create --name meterproject
```

4. Add the **meteruser** to the *meterproject* project and the *Member* role.

```
[root@serverX ~]# keystone user-role-add --user-id meteruser --role-id Member --
tenant-id meterproject
```

5. Login to the Horizon dashboard as user *meteruser* and create an instance with the name *meterinstance* using the **m1.tiny** flavor and the **ceilometertest** image.

6. Explore various meters from the output of **ceilometer meter-list** by looking at statistics and samples of the *instance*, *memory* and *vcpu* meters.

```
[root@serverX ~]# ceilometer meter-list
+--------------------+-------+----------+--------------+---------+-------------+
| Name               | Type  | Unit     | Resource ID  | User ID | Project ID  |
+--------------------+-------+----------+--------------+---------+-------------+
| disk.ephemeral.size | gauge | GB       | d81f911(...) | fd(...) | e6a50a(...) |
| disk.root.size     | gauge | GB       | d81f911(...) | fd(...) | e6a50a(...) |
| image              | gauge | image    | ed611f8(...) |         | 750aa2(...) |
| image.download     | delta | B        | ed611f8(...) |         | 750aa2(...) |
| image.serve        | delta | B        | ed611f8(...) |         | 750aa2(...) |
| image.size         | gauge | B        | ed611f8(...) |         | 750aa2(...) |
| image.update       | delta | event    | ed611f8(...) |         | 750aa2(...) |
| image.upload       | delta | event    | ed611f8(...) |         | 750aa2(...) |
| instance           | gauge | instance | d81f911(...) | fd(...) | e6a50a(...) |
| instance:m1.small  | gauge | instance | d81f911(...) | fd(...) | e6a50a(...) |
| memory             | gauge | MB       | d81f911(...) | fd(...) | e6a50a(...) |
| vcpus              | gauge | vcpu     | d81f911(...) | fd(...) | e6a50a(...) |
+--------------------+-------+----------+--------------+---------+-------------+
[root@serverX ~]# ceilometer sample-list -m instance
...output ommitted...
[root@serverX ~]# ceilometer sample-list -m memory
...output ommitted...
[root@serverX ~]# ceilometer sample-list -m vcpu
...output ommitted...
[root@serverX ~]# ceilometer statistics -m instance
...output ommitted...
[root@serverX ~]# ceilometer statistics -m memory
...output ommitted...
[root@serverX ~]# ceilometer statistics -m vcpu
...output ommitted...
```

# Chapter 13: The future direction of Red Hat OpenStack

Fill-in-the-Blank Quiz

## OpenStack code names and projects

For each of the following statements, fill in the blanks:

1. The Havana is the **H** release of OpenStack and was released in fall 2013.
2. The Icehouse is the **I** release of OpenStack and will be released in spring 2014.
3. The Trove is the Database as a Service (DBaaS) project that will be incubated into the **I** release.
4. The Ironic is the bare-metal driver that will be incubated into the **I** release.
5. The Marconi is the queuing and notification service that will be incubated into the **I** release.
6. The Savannah is a project to simplify provisioning of a Hadoop cluster that will be incubated into the **I** release.

# Chapter 14: Comprehensive review

Case Study

## Comprehensive review

**Lab overview:** Review Red Hat OpenStack installation and configuration.

**Success criteria:** Red Hat OpenStack cloud running according to specifications.

***Before you begin...***

Save any work on desktop*X* and server*X* that you want to keep because you will reinstall these machines. Reboot desktop*X* and choose the "Reinstall Desktop *X*" option in grub. Enter a password of **redhat** to begin the installation.

If you are working in the virtual training environment, ignore the preceding paragraph and use the virtual machine controls in your web browser to reset your machine to a snapshot of your **desktop*X*** and **server*X*** machine. On the **desktop*X*** tab in your browser, open the **state** dropdown menu and select the **Snapshots** tab. Select the **Initial Snapshot** snapshot via the radio buttons and press the **Revert to selected snapshot** button. Press the **Power On** button to start **desktop*X*.example.com**. Repeat the process for **server*X*.example.com**, again choosing the **Initial Snapshot** snapshot.

If you are using the Red Hat Online Learning environment, run the **lab-clean-desktop** command to clean desktop*X*.example.com and reset server*X*.example.com.

**Lab outline:** The itemized lists define the requirements necessary to install and configure Red Hat OpenStack. server*X* will provide most services, but desktop*X* will provide the Cinder service and will be the Nova compute node. You will launch two instances in Red Hat OpenStack.

**Install Red Hat OpenStack as follows:**

*Configure **serverX.example.com** with the following settings:*

- Generate SSH keys to ease installation.
- Configure NTP using **192.168.0.254** as the NTP server for all machines.
- Create a private network named **int** and a subnet named **subint** using the **192.168.32.0/24** network. Set the gateway for this network to **192.168.32.1**.
- Create a public network named **ext** and a subnet named **subext** using the **172.24.*X*.0/24** network. The gateway for this network is **172.24.*X*.254**. This network must be configured for the floating IP addresses.
- Use a VLAN range of **1000-2999** and configure **br-eth1** as the OVS bridge for communication. **desktop*X*.example.com** will use the **br101** network device to communicate on **br-eth1**. **server*X*.example.com** will use the **eth1** network device to communicate on **br-eth1**.
- Configure Horizon to accept SSL connections.

*Configure **desktopX.example.com** with the following settings:*

- Enable the Cinder service using the pre-existing **cinder-volumes** volume group.
- Enable the Nova compute service on **desktop*X***.

**Configure Red Hat OpenStack as follows:**

- Create a project named **project1** using a description of **Project for project1**. Set the quota for this project to four VCPUs, 4096MB RAM and two floating IP addresses.
- Create a new flavor named **m2.tiny** which includes one VPCU, 1024MB RAM, a 20GB root disk, a 2GB ephemeral disk and a 512MB swap disk.
- Create a new user named **user1** with an email address of *root@desktopX.example.com*. Set the password to **redhat** and include this user as a member of the **project1** project.
- Create a new user named **adm1** with an email address of *root@desktopX.example.com*. Set the password to **redhat** and include this user as an admin of the **project1** project.
- In the **project1** project, create a new image named **small** using the QCOW2 image located at `http://instructor.example.com/pub/materials/small.img`. Set no minimum disk, minimum 512MB RAM, and make the image public.
- In the **project1** project, create a new image named **web** using the QCOW2 image located at `http://instructor.example.com/pub/materials/web.img`. Set no minimum disk, minimum 1024MB RAM, and make the image public.
- In the **project1** project, allocate two floating IP addresses.
- In the **project1** project, create a new security group named **sec1** with a description of **Web and SSH**. Allow **TCP/22** and **TCP/443** from **CIDR 0.0.0.0/0**, and **TCP/80** from the **sec1** source group.
- In the **project1** project, create an SSH key pair named **key1**. Save the private key to **/home/student/Downloads/key1.pem** on **desktop*X*.example.com**.
- In the **project1** project, launch a new instance named **small** using the **small** image and the **m1.tiny** flavor. Include the **key1** key pair and the **sec1** security group. Associate the **172.24.*X*.2** floating IP address.
- In the **project1** project, launch a new instance named **web** using the **web** image and the new **m2.tiny** flavor. Include the **key1** key pair and the **sec1** security group. Associate the **172.24.*X*.3** floating IP address.
- In the **project1** project, create a new 10GB volume named **vol1** and attach this volume to the **web** instance.

1. Save any work on desktop*X* and server*X* that you want to keep because you will reinstall these machines. Reboot desktop*X* and choose the "Reinstall Desktop *X*" option in grub. Enter a password of **redhat** to begin the installation.
2. Once the installation is finished, log in to desktop*X* as **student** with a password of **student**. Open a terminal and become **root** (with a password of **redhat**).

```
[student@desktopX ~]$ su -
Password: redhat
```

```
[root@desktopX ~]#
```

3. Install updates on **desktop*X*.example.com**.

```
[root@desktopX ~]# yum update -y
```

4. Use **virt-manager**, **virt-viewer server*X*** or **ssh root@server*X*** to login as **root** on server*X*.example.com (password: **redhat**).

   On **server*X*.example.com**, install all updates.

```
[root@serverX ~]# yum update -y
```

5. Install the *openstack-packstack* package.

```
[root@serverX ~]# yum install -y openstack-packstack
```

6. Generate SSH keys.

```
[root@serverX ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/root/.ssh/id_rsa): Enter
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/root/.ssh/id_rsa.
Your public key has been saved in /home/root/.ssh/id_rsa.pub.
...
```

7. Generate an answer file with **packstack**.

```
[root@serverX ~]# packstack --gen-answer-file /root/answers.txt
```

8. Edit the **/root/answers.txt** and ensure the following items are configured:

```
CONFIG_SSH_KEY=/root/.ssh/id_rsa.pub
CONFIG_NTP_SERVERS=192.168.0.254
CONFIG_CINDER_HOST=192.168.0.X
CONFIG_NOVA_COMPUTE_HOSTS=192.168.0.X
CONFIG_NEUTRON_OVS_TENANT_NETWORK_TYPE=vlan
CONFIG_NEUTRON_OVS_VLAN_RANGES=physnet1:1000:2999
CONFIG_NEUTRON_OVS_BRIDGE_MAPPINGS=physnet1:br-eth1
CONFIG_HORIZON_SSL=y
```

9. Configure Red Hat OpenStack using the answer file.

```
[root@serverX ~]# packstack --answer-file /root/answers.txt
Welcome to Installer setup utility

Installing:
Clean Up...                                  [DONE]
Setting up ssh keys...root@192.168.0.X+100's password: redhat
root@192.168.0.X's password: redhat
```

```
...
```

10. We need to tie the bridges together on desktop*X* and server*X*.

```
[root@server X ~]# ovs-vsctl add-port br-eth1 eth1
```

```
[root@desktop X ~]# ovs-vsctl add-port br-eth1 br101
```

11. On server*X* we need to fix up the interface configuration files. First we should make a backup copy of our original ifcfg file.

```
[root@server X ~]# cp /etc/sysconfig/network-scripts/ifcfg-eth0 /root/
```

And then we use **ifcfg-eth0** as a template for our new bridge by copying it to **ifcfg-br-ex**.

```
[root@server X ~]# cp /etc/sysconfig/network-scripts/ifcfg-eth0 /etc/sysconfig/
network-scripts/ifcfg-br-ex
```

12. If you are in a physical classroom remove all entries but DEVICE, HWADDR and ONBOOT from the **/etc/sysconfig/network-scripts/ifcfg-eth0** file.

```
[root@server X ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
HWADDR=52:54:00:00:00:XX
ONBOOT=yes
```

If you are in a virtual classroom remove all entries but DEVICE and ONBOOT from the **/etc/sysconfig/network-scripts/ifcfg-eth0** file.

```
[root@server X ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
ONBOOT=yes
```

13. And the bridge device configuration file **/etc/sysconfig/network-scripts/ifcfg-br-ex** has to be changed too.

```
[root@server X ~]# /etc/sysconfig/network-scripts/ifcfg-br-ex
DEVICE=br-ex
IPADDR=192.168.0.X+100
PREFIX=24
GATEWAY=instructor
DNS1=instructor
SEARCH1=example.com
ONBOOT=yes
```

14. On **serverX**, add the **eth0** device to the **br-ex** OVS bridge and restart the network.

```
[root@server X ~]# ovs-vsctl add-port br-ex eth0 ; service network restart
```

15. On desktop*X*, open the Red Hat OpenStack dashboard by browsing to `https://server*X*.example.com/dashboard`. Login as **admin** using the password found in **/root/keystonerc_admin** on server*X* (the OS_PASSWORD variable).

```
[root@server X ~]# source /root/keystonerc_admin
[root@server X ~(keystone_admin)]$ echo $OS_PASSWORD
abcdef1234567890

[root@desktop X ~]# firefox https://server X.example.com/dashboard
```

16. Create new project (tenant) named **project1**. Use a description of **Project for project1**. Set the quota to 4 VCPUs, 4 instances, 4096 MB RAM and 2 floating IPs.

    In the **Admin** tab in the left pane, click on the **Projects** link. Click on the **Create Project** button. Add the name and description as above. Go to the **Quota** tab and set the quotas as above. Click the **Finish** button.

17. Create a new flavor named **m2.tiny** which includes 1 VPCU, 1024 MB RAM, 20 GB root disk, 2 GB ephemeral disk and 512 MB swap disk.

    In the **Admin** tab in the left pane, click on the **Flavors** link. Click on the **Create Flavor** button. Enter the information as above.

18. Create a new user named **user1**. Use an email address of *root@desktopX.example.com* and a password of **redhat**. Include this user in the **project1** project with a **Member** role.

    In the **Admin** tab in the left pane, click on the **Users** link. Click on the **Create User** button. Add the information as above. Click the **Create User** button.

19. Create a new user named **adm1**. Use an email address of *root@desktopX.example.com* and a password of **redhat**. Include this user in the **project1** project with an **admin** role.

    In the **Admin** tab in the left pane, click on the **Users** link. Click on the **Create User** button. Add the information as above. Click the **Create User** button.

20. Logout as **admin** and login as **user1**.

21. Add a new image named **small** using the QCOW2 image located at `http://instructor.example.com/pub/materials/small.img`. Set no minimum disk, minimum 512 MB RAM and make the image public.

    In the **Project** tab in the left pane, select the **Images & Snapshots** link. Click on the **Create Image** button. Enter the information above and click the **Create Image** button.

22. Add a new image named **web** using the QCOW2 image located at `http://instructor.example.com/pub/materials/web.img`. Set no minimum disk, minimum 1024 MB RAM and make the image public.

    In the **Project** tab in the left pane, select the **Images & Snapshots** link. Click on the **Create Image** button. Enter the information above and click the **Create Image** button.

23. Next, we configure networking. In the **Project** tab on the left pane, select the **Networks** link. Press the **Create Network** button. Enter the private network name **int**. Click on the **Subnet** tab. Enter the private subnet information as above. Press the **Create** button.

Press the **Create Network** button again. Enter the public network name **ext**. Browse to the **Subnet** tab. Enter the public subnet information as above. Browse to the **Subnet Detail** tab. Deselect the **Enable DHCP** checkbox and leave the rest of the fields blank. Press the **Create** button.

In the **Project** tab on the left pane select the **Routers** link. Press the **Create Router** button. Enter **router1** for the router name and press the **Create** button.

24. Sign out as the **user1** user and sign in as **admin**. In the **Admin** tab in the left pane, click on the **Networks** link. Click on the **Edit Network** button in the public (**ext**) network row. Select the **External Network** checkbox and press the **Save Changes** button.

25. Sign out as the **admin** user and sign in as **user1**. In the **Project** tab in the left pane, click on the **Routers** link. Press the **Set Gateway** button in the **router1** row. In the **External Network** menu, choose **ext**. Press the **Set Gateway** button.

26. Click on the **router1** link. Press the **Add Interface** button. In the **Subnet** menu, select **int: 192.168.32.0/24 (subint)**. Press the **Add Interface** button.

27. Allocate two floating IP addresses.

    In the **Project** tab in the left pane, select the **Access & Security** link and choose the **Floating IPs** tab. Click on the **Allocate IP To Project** button. Use the **ext** pool and click on the **Allocate IP** button. Repeat the process so you have two floating IP addresses (172.24.*X*.2 and 172.24.*X*.3).

28. Create a new security group named **sec1** with a description of **Web and SSH**. Allow TCP/22 and TCP/443 from CIDR 0.0.0.0/0, and TCP/80 from the **sec1** source group.

    In the **Project** tab in the left pane, select the **Access & Security** link. Click on the **Create Security Group** button. Enter the name and description as above. Click on the **Edit Rules** button for the **sec1** security group. Press the **Add Rule** button. Choose TCP as the protocol, enter 22 as the Port, and leave the Source and CIDR as above. Press the **Add** button. Press the **Add Rule** button again. Enter 443 in the Port box and press the **Add** button. Press the **Add Rule** button one more time. Enter 80 in the Port box. Choose **Security Group** in the **Source** drop-down menu. Choose **sec1 (current)** from the **Security Group** drop-down menu. Press the **Add** button.

29. Create an SSH keypair for the virtual machines named **key1**.

    In the **Project** tab in the left pane, select the **Access & Security** link and choose the **Keypairs** tab. Press the **Create Keypair** button. Enter the name as above and press the **Create Keypair** button. Save the **key1.pem** file to the default location, which should be in **/home/student/Downloads/key1.pem** on **desktop*X*.example.com**.

30. Launch a new instance named **small** using the **small** image and the **m1.tiny** flavor. Use the **key1** keypair and include the **sec1** security group. Associate the **172.24.*X*.2** floating IP address.

    In the **Project** tab in the left pane, select the **Instances** link. Press the **Launch Instance** button. In the **Details** tab, enter the image and name as above. In the **Access & Security** tab, enable the keypair and security group listed above. In the **Networking** tab, press the **+** button next to the **int** network. Press the **Launch** button.

Once the instance has been created, open the **Actions** drop-down menu and select **Associate Floating IP**. Choose the **172.24.*X*.2** floating IP address, and choose the **small** instance, then press the **Associate** button.

31. Launch a new instance named **web** using the web image and the **m2.tiny** flavor. Use the **key1** keypair and include the **sec1** security group. Associate the 172.24.*X*.3 floating IP address.

    In the **Project** tab in the left pane, select the **Instances** link. Press the **Launch Instance** button. In the **Details** tab, enter the image, name and flavor as above. In the **Access & Security** tab, enable the keypair and security group listed above. In the **Networking** tab, press the **+** button next to the **int** network. Press the **Launch** button.

    Once the instance has been created, open the **Actions** drop-down menu and select **Associate Floating IP**. Choose the 172.24.*X*.3 floating IP address, and choose the **web** instance, then press the **Associate** button.

32. Once both instances are available, verify the network services.

```
[student@desktopX ~]$ firefox http://172.24.X.3
[student@desktopX ~]$ firefox https://172.24.X.3
[student@desktopX ~]$ chmod 600 /home/student/Downloads/key1.pem
[student@desktopX ~]$ ssh -i /home/student/Downloads/key1.pem root@172.24.X.2
[student@desktopX ~]$ ssh -i /home/student/Downloads/key1.pem root@172.24.X.3
```

33. Create a 10 GB volume with a name of **vol1**. Attach this volume to the **web** instance.

    In the **Project** tab in the left pane, select the **Volumes** link. Press the **Create Volume** button. Enter the name and size as above and press the **Create Volume** button.

    Press the **Edit Attachments** button for the **vol1** volume. Choose the **web** instance and press the **Attach Volume** button.

# Personal Notes