



**Hewlett Packard
Enterprise**

Fundamentals of OpenStack® Technology

Student guide (1 of 2)

H6C68S E.01

Use of this material to deliver training without prior written permission from HPE is prohibited.

© Copyright 2017 Hewlett Packard Enterprise Development LP

The information contained herein is subject to change without notice. The only warranties for HPE products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.

This is an HPE copyrighted work that may not be reproduced without the written permission of Hewlett Packard Enterprise. You may not use these materials to deliver training to any person outside of your organization without the written permission of HPE.

Microsoft®, Windows®, Windows NT® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

All other product names mentioned herein may be trademarks of their respective companies.

Export Compliance Agreement

Export Requirements. You may not export or re-export products subject to this agreement in violation of any applicable laws or regulations.

Without limiting the generality of the foregoing, products subject to this agreement may not be exported, re-exported, otherwise transferred to or within (or to a national or resident of) countries under U.S. economic embargo and/or sanction including the following countries: Cuba, Iran, North Korea, Sudan and Syria. This list is subject to change.

In addition, products subject to this agreement may not be exported, re-exported, or otherwise transferred to persons or entities listed on the U.S. Department of Commerce Denied Persons List; U.S. Department of Commerce Entity List (15 CFR 744, Supplement 4); U.S. Treasury Department Designated/Blocked Nationals exclusion list; or U.S. State Department Debarred Parties List; or to parties directly or indirectly involved in the development or production of nuclear, chemical, or biological weapons, missiles, rocket systems, or unmanned air vehicles as specified in the U.S. Export Administration Regulations (15 CFR 744); or to parties directly or indirectly involved in the financing, commission or support of terrorist activities.

By accepting this agreement you confirm that you are not located in (or a national or resident of) any country under U.S. embargo or sanction; not identified on any U.S. Department of Commerce Denied Persons List, Entity List, US State Department Debarred Parties List or Treasury Department Designated Nationals exclusion list; not directly or indirectly involved in the development or production of nuclear, chemical, biological weapons, missiles, rocket systems, or unmanned air vehicles as specified in the U.S. Export Administration Regulations (15 CFR 744), and not directly or indirectly involved in the financing, commission or support of terrorist activities.

Printed in US

Fundamentals of OpenStack® Technology

Student guide (1 of 2)

March 2017

Contents

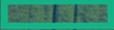
Module 1 – Course Overview.....	1-1
Course objectives.....	1-2
Course audience.....	1-3
Course prerequisites	1-4
Introductions.....	1-5
Course agenda—Day 1.....	1-6
Course agenda—Day 2.....	1-7
Course agenda—Day 3.....	1-8
Module 2 – Introduction to OpenStack.....	2-1
Learning objectives	2-2
OpenStack overview	2-3
What is Cloud Computing.....	2-4
Cloud service types	2-5
Cloud Service Type Examples	2-6
Cloud Operating System	2-7
What is OpenStack®?	2-8
OpenStack® benefits	2-10
OpenStack® organization	2-11
OpenStack release history	2-12
OpenStack Documentation	2-13
HPE and the cloud	2-14
HPE history in the open source community	2-15
HPE Helion-related Products	2-16
OpenStack architecture.....	2-17
OpenStack projects (as of Newton release).....	2-18
OpenStack Conceptual Architecture	2-20
OpenStack Inter-Service Communications	2-21
OpenStack high-level architecture	2-22
OpenStack Big Tent and Core Services.....	2-23
OpenStack Project Adoption, Maturity, Age	2-24
OpenStack Service APIs	2-26
Communication to and between OpenStack Projects	2-27
Message lifecycle	2-28
OpenStack Service API Versioning	2-29
REST API Microversion support.....	2-30
OpenStack API Reference Materials.....	2-31
OpenStack Installation and configuration	2-32
Basic OpenStack physical components	2-33
Wide range of configuration options.....	2-34
Primary Types of OpenStack Installations	2-35
OpenStack development environment (DevStack)	2-36
Scripted DevStack installation process	2-37

Example local.conf file for devstack installation	2-38
Installing an OpenStack® sandbox manually.....	2-39
Module summary.....	2-40
Learning Check	2-41
Module 3 – OpenStack Management Tools.....	3-1
Learning objectives	3-2
Overview of OpenStack Management Tools	3-3
REST Architecture.....	3-4
OpenStack RESTful API Access Methods	3-5
REST Client Browser-Based UI Plug-ins	3-6
Firefox RESTED plug-in	3-7
Using cURL for OpenStack API Requests	3-8
Example cURL Request and Endpoint Response.....	3-9
OpenStack CLI client.....	3-10
OpenStack CLI command with --debug	3-11
OpenStackClient	3-12
OpenStack Horizon (dashboard) user interface	3-13
Using Python to invoke an OpenStack API.....	3-16
Module summary.....	3-17
Learning check	3-18
Module 4 – Keystone – OpenStack Identity Service.....	4-1
Learning Objectives.....	4-2
OpenStack Keystone (identity) Overview.....	4-3
Keystone identity and service management.....	4-4
Keystone API version states as of the Newton release.....	4-5
Keystone Identity (Authentication/Authorization) Process.....	4-6
Keystone Identity-related objects	4-8
Keystone Identity-related object relationships.....	4-10
Hierachial multitenancy	4-11
Additional Keystone terminology	4-12
Keystone Support for Federated Identity	4-13
Keystone-specific reference resources	4-15
Identity service architecture.....	4-16
Internal components of the Identity service.....	4-17
Keystone backend services.....	4-18
Common Keystone management tasks	4-19
Managing projects with the CLI client.....	4-20
Managing users with the CLI client	4-21
Creating a role, project, user, and adding project user to role	4-22
Managing the service catalog with the CLI client	4-23
Keystone Identity in the Horizon UI	4-24
Setting user policies in OpenStack.....	4-25
Keystone configuration files	4-26
Keystone troubleshooting.....	4-27
Verifying the Identity service installation	4-28
Keystone troubleshooting.....	4-29
Module summary.....	4-30
Learning check	4-31

Module 5 –OpenStack Image Services.....	5-1
Learning Objectives.....	5-2
Glance overview.....	5-3
Glance Functionality.....	5-4
Example Glance Use Case	5-5
Glance API version states as of Newton release	5-6
Glance architecture	5-7
Glance-supported backend stores	5-8
Glance disk and container formats.....	5-9
Glance images	5-10
Pre-built images for OpenStack	5-11
Custom OpenStack images.....	5-12
Handling custom Glance images.....	5-13
Cloud-init	5-14
Public tools for image creation	5-15
Common Glance management tasks	5-16
Glance-related configuration files.....	5-17
Common Glance-related CLI commands	5-18
Using the CLI to load an image in Glance.....	5-19
Glance operations using Horizon	5-20
Troubleshooting Glance	5-21
Troubleshooting	5-22
Glance-related log files.....	5-23
Module summary.....	5-24
Learning check.....	5-25
Module 6 – Neutron – OpenStack Networking Service.....	6-1
Learning Objectives.....	6-2
Virtual Networking Concepts	6-3
Hosting device virtual machines and baremetal machines	6-4
Virtual L2 Switch	6-5
Virtual Local Area Network.....	6-6
Virtual Overlay Networks.....	6-7
Floating IP Address	6-9
Open Virtual Switch (OVS) Components – Compute Host	6-10
Open Virtual Switch (OVS) Components – Network Host configuration	6-13
Neutron overview	6-15
Neutron Functionality	6-16
Neutron API version states as of the Newton release.....	6-16
Neutron API Network Resources	6-18
Neutron Networking Terminology	6-19
Neutron Architecture	6-20
Neutron components	6-21
OpenStack Nodes with Neutron Components.....	6-26
OpenStack Node Networks	6-27
Network Topologies – Single Flat and Private Network	6-28
Neutron Topology – Using routers to separate Project Networks	6-29
Neutron Virtual Networks	6-30
Example Neutron GRE Overlay Network	6-32
Neutron DVR (<i>Distributed Virtual Router</i>)	6-34
Neutron L3 in high availability configuration.....	6-35

Common Neutron Management Tasks	6-36
Common CLI management tasks	6-37
Neutron in Horizon	6-39
Neutron in Horizon	6-40
Troubleshooting Neutron.....	6-41
Approach to network troubleshooting.....	6-42
Neutron log files	6-43
Module summary.....	6-44
Learning check	6-45
 Module 7 – Nova – OpenStack Compute Service	7-1
Learning Objectives.....	7-2
Nova overview.....	7-3
Nova Features and Functions	7-4
Nova Design Guidelines.....	7-5
Nova (Compute) API version states as of Newton release	7-6
API Microversioning	7-7
Common Nova terminology	7-8
Nova architecture	7-10
Nova System Architecture.....	7-11
Nova supported hypervisors.....	7-12
VM Instance creation process	7-14
Compartmentalizing OpenStack Deployments.....	7-16
Segregating a cloud implementation	7-17
Regions, Availability Zones, and Host Aggregates	7-19
Regional architecture	7-20
Cells – API Cell, Child Cells, Grandchild Cells.....	7-21
Common Nova Management Tasks.....	7-22
Nova configuration files	7-23
Gathering information required to create an instance	7-24
Creating an Instance	7-27
Verify Instance was created properly	7-28
Using Instance-Specific Data	7-29
Creating an instance using the Horizon UI.....	7-31
Viewing instance details in the Horizon UI	7-33
Managing Security Groups from the Horizon UI.....	7-34
Assigning a Floating IP Address to an Instance.....	7-35
Connecting to an instance	7-37
Managing Instances from the Horizon UI	7-38
Nova scheduler	7-39
Nova Scheduler overview.....	7-40
Nova-scheduler scheduling algorithms	7-41
Filter Scheduler - Filters and Weights	7-42
Nova-scheduler filters.....	7-43
Default nova-scheduler configuration.....	7-44
Nova-scheduler weights	7-45
Nova Maintenance	7-46
Planned compute node maintenance	7-46
Inspecting and recovering data from failed instances	7-48
Nova Troubleshooting	7-51
Verifying the Nova installation	7-52

Troubleshooting Nova – Common issues	7-53
Troubleshooting the Nova service.....	7-55
Module Summary	7-56
Learning check.....	7-57
Module 8 – OpenStack® Block Storage (Cinder)	8-1
Learning Objectives.....	8-2
Cinder overview	8-3
OpenStack storage overview	8-4
Comparison between the OpenStack storage technologies	8-5
OpenStack block storage (Cinder).....	8-6
Cinder API version states as of Newton release	8-7
Block Storage Use Cases	8-8
Cinder Storage Terms	8-9
OpenStack Block Storage (Cinder) Drivers.....	8-10
Cinder architecture	8-11
High-level Cinder architecture	8-12
Cinder Volume Creation Workflow	8-13
Common Cinder Management Tasks.....	8-15
Common Management Tasks	8-16
Viewing the status of cinder services	8-17
Creating volumes	8-18
Attach a volume to an existing instance	8-19
Cinder snapshots and backups	8-21
Creating consistent snapshots	8-22
Snapshot operations	8-23
Creating a bootable volume	8-24
Consistency Groups	8-25
Cinder types	8-26
Cinder Volumes in the Horizon UI	8-27
Troubleshooting Cinder	8-28
Block storage creation failures	8-30
Module summary.....	8-31
Learning Check	8-32

 **Hewlett Packard
Enterprise**

Fundamentals of OpenStack® Technology

Module 1—Course Overview

H6C68S E.01

© Copyright 2017 Hewlett Packard Enterprise Development LP

Course objectives

After completing this course, you should be able to:

- Describe the primary purpose and major features of OpenStack®
- Discuss how HPE is involved in the OpenStack® project
- List the primary components of the OpenStack® architecture and their function
- Describe what the core services of OpenStack® are and what is meant by the OpenStack® big tent
- Access the OpenStack® endpoint services from the CLI, a REST client, the cURL, and the dashboard UI
- Configure and use OpenStack® to deploy compute instances
- Use the primary features of the core OpenStack® services
- Discuss other OpenStack® projects that exist under the OpenStack® big tent



© Copyright 2017 Hewlett Packard Enterprise Development LP

2

Course objectives

After completing this course, you should be able to:

- Describe the primary purpose and major features of OpenStack®
- Discuss how HPE is involved in the OpenStack® project
- List the primary components of the OpenStack® architecture and their function
- Describe what the core services of OpenStack are and what is meant by the OpenStack® big tent
- Access the OpenStack® endpoint services from the CLI, a REST client, the cURL, and the dashboard UI
- Configure and use OpenStack® to deploy compute instances
- Use the primary features of the core OpenStack® services
- Discuss other OpenStack® projects that exist under the OpenStack® big tent

Course audience

– Primary audience

- Linux system administrators and cloud administrators interested in or responsible for maintaining a private cloud using the OpenStack® technology

– Secondary audience

- Architects
- Solution designers
- Technical consultants



© Copyright 2017 Hewlett Packard Enterprise Development LP

3

Course audience

This course was developed primarily for cloud administrators and Linux system administrators who are interested in or responsible for maintaining a private cloud using the OpenStack® technology. The information provided in this training could also be beneficial to architects, solution designers, and technical consultants.

Course prerequisites

- Mandatory experience:
 - Linux system management
 - Linux networking
 - Shell scripting
 - Linux terminal-based text editors such as vi, nano, or pico
 - Terminal emulators such as PuTTY and MobaXterm
- Optional experience:
 - QEMU and KVM virtualization
 - MySQL or other relational database



© Copyright 2017 Hewlett Packard Enterprise Development LP

4

Course prerequisite

To successfully complete the labs in this course, you should have prior experience in the following areas:::

- Linux system management
- Linux networking
- Shell scripting
- Linux terminal-based text editors such as vi, nano, or pico
- Terminal emulators such as PuTTY and MobaXterm

Experience in following technologies is not mandatory, but useful to better understand the topics of the course:

- QEMU and KVM virtualization
- MySQL or other relational database

Introductions

- Your name
- Your company name
- Location
- Years of experience (Linux, virtualization, and so on)
- Course expectations

NOTE: If you are experiencing problems with your audio connection, use the public chat room for your introduction.



© Copyright 2017 Hewlett Packard Enterprise Development LP

5

Introductions

Take a few minutes for introduction. This will help your instructor match the content of the training and adjust the delivery to better suit your needs. We would like to hear:

- Your name
- Your company name
- Location
- Years of experience (Linux, virtualization, and so on)
- Course expectations

Training agenda—Day 1

Lectures

- Module 1—Course Overview
- Module 2—Introduction to OpenStack
- Module 3—OpenStack Management Tools
- Module 4—Keystone (Identity) service
- Module 5—Glance (Image Repository) service

- Lab 1—OpenStack documentation and lab environment
- Lab 2—OpenStack management tools
- Lab 3—OpenStack keystone
- Lab 4—OpenStack glance

Labs



© Copyright 2017 Hewlett Packard Enterprise Development LP

6

Training agenda—Day 1

Here is a suggested agenda for the first day of the training.

Training agenda—Day 2

Lectures

- Module 6—Neutron (Networking) service
- Module 7—Nova (Compute) service
- Module 8—Cinder (Block Storage) service
- Module 9—Swift (Object Storage) service

Labs

- Lab 5—OpenStack neutron
- Lab 6—OpenStack nova
- Lab 7—OpenStack cinder
- Lab 8—OpenStack swift



© Copyright 2017 Hewlett Packard Enterprise Development LP

7

Training agenda—Day 2

Here is a suggested agenda for the second day of the training.

Training agenda—Day 3

Lectures

- Module 10—Heat (Orchestration) service
- Module 11—Ceilometer (Metering) service
- Module 12—Other OpenStack services
- Module 13—OpenStack deployment planning

Labs

- Lab 9—OpenStack heat
- Lab 10—OpenStack ceilometer
- Extra lab time



© Copyright 2017 Hewlett Packard Enterprise Development LP

8

Training agenda—Day 3

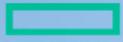
Here is a suggested agenda for the third day of the training.



Q&A



**Hewlett Packard
Enterprise**

 **Hewlett Packard
Enterprise**

Fundamentals of OpenStack® Technology

Module 2—Introduction to OpenStack®

H6C68S E.01



Learning objectives

After completing this module, you should be able to:

- Discuss the OpenStack® offering and the cloud service model it employs
- Highlight some significant events in the history of OpenStack®
- Identify the architectural components of OpenStack®
- Explain how to install and configure OpenStack®
- Use some of the tools for accessing and managing the OpenStack® services



© Copyright 2017 Hewlett Packard Enterprise Development LP

2

Learning objectives

After completing this module, you should be able to:

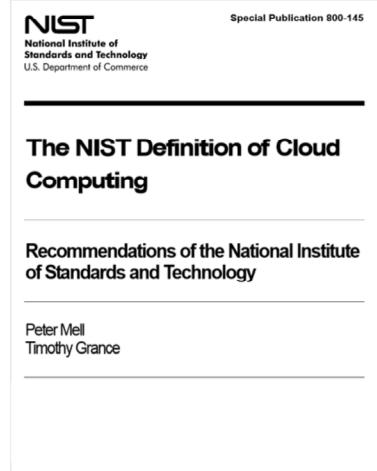
- Discuss the OpenStack® offering and the cloud service model it employs
- Highlight some significant events in the history of OpenStack®
- Identify the architectural components of OpenStack®
- Explain how to install and configure OpenStack®
- Use some of the tools for accessing and managing the OpenStack® services



OpenStack® overview

What is cloud computing?

- Essential characteristics of cloud computing—from the NIST definition:
 - On-demand self-service
 - Network access for computing capabilities based on standard protocols
 - Resource pooling for multiple concurrent users
 - Elasticity for rapid scaling and provisioning
 - Metered or measured service for network, storage, and CPU resources



© Copyright 2017 Hewlett Packard Enterprise Development LP

4

What is cloud computing?

The National Institute of Standards and Technology (NIST) provides the following definition of cloud computing: “Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.”

Five essential characteristics of cloud computing can be elaborated as follows:

- On-demand self-service: Users can provision servers and networks with little human intervention.
- Network access: Any computing capabilities are available over the network. Many different devices are allowed access through standardized mechanisms.
- Resource pooling: Multiple users can access clouds that serve other consumers according to demand.
- Elasticity: Provisioning is rapid and scales out or in based on need.
- Metered or measured service: Just like utilities that are paid for by the hour, clouds optimize resource use and control it for the level of service or the type of server, such as storage or processing.

Cloud types

Based on the service type

- Infrastructure as a Service (IaaS)
 - OpenStack®
 - HPE Helion OpenStack®, HPE Helion CloudSystem
- Platform as a Service (PaaS)
 - Microsoft Azure, Google App Engine
 - Stackato
- Software as a Service (SaaS)
 - Gmail, Twitter, Facebook

Based on location

- Private clouds
- Public clouds
- Hybrid (private + public) clouds



© Copyright 2017 Hewlett Packard Enterprise Development LP

5

Cloud types

Cloud computing is usually described in one of the two ways—either based on the cloud location or the service that the cloud is offering.

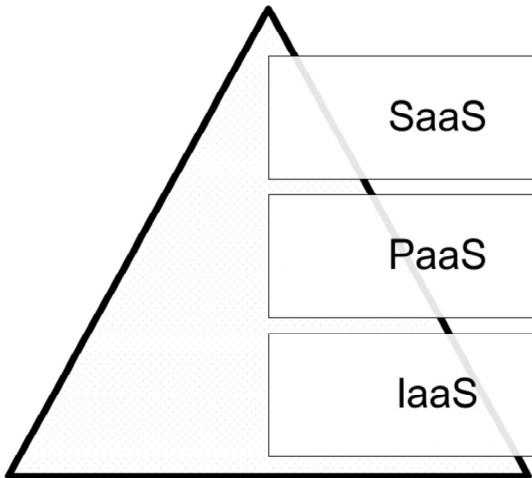
Based on a cloud location, a cloud can be classified as follows:

- **Private cloud**, which is operated by a single organization. In most cases, it is intended to be exclusively used by consumers associated with a single organization.
- **Public cloud**, which is generally provided by a cloud services company and accessible to the general public.
- **Hybrid cloud**, which is a combination of both a private and a public cloud. One of the primary use cases is where an organization primarily uses the private cloud, but has an SLA with a public cloud provider to use public cloud resources when private cloud resources become overloaded.

When it comes to the service that the cloud is offering, there are three different categories:

- **IaaS (Infrastructure as a Service)**: Provides all the compute systems, storage, and networking required to connect those devices together and to the consumer. OpenStack® is used to deploy the IaaS solutions. Examples include OpenStack®, HPE Helion Community Cloud, and HPE CloudSystem Matrix.
- **PaaS (Platform as a Service)**: Allows deploying applications through a programming language or tools supported by the cloud platform provider (for example, Eclipse/Java programming platform provided, where no downloads are required). Think of this as a Developer Platform as a Service. Examples include Microsoft Azure, Google App Engine, and Stackato.
- **SaaS (Software as a Service)**: Allows using software in a cloud environment (for example, web-based email). Software exists on and is accessed from the cloud. Google, Twitter, Facebook, and Flickr are examples of SaaS.

Cloud service type examples



- **Function:** Providing software services
- **Audience:** Application end users

- **Function:** Hardware and dev tools abstraction
- **Audience:** Application developers

- **Function:** Hardware platform abstraction
- **Audience:** System architects

Cloud service type examples

The above slide provides an example and audiences for three types of cloud models.

- Software as a Service (SaaS) is used by the application end users. It provides software services, such as Google Apps or Microsoft Office 365 online services. Services are deployed instantly and are available immediately. The service can be accessed once the account is created online. This type of the cloud does not provide the administrative interface or the API.
- Platform as a Service (PaaS) is used primarily by application developers. This type of the cloud provides the means of creating the cloud-native applications. Platform as a Service provides the selection of cloud frameworks (Python, PHP, Java, and so on) and databases (MySQL, PostgreSQL, MongoDB, and so on). At the same time, PaaS enables the application high availability, and provides the elasticity and application scalability on demand.
- Infrastructure as a Service (IaaS) provides the hardware platform abstraction. It is primarily used by the system architects and end users involved in designing the infrastructure based on traditional IT components, such as network, disks, servers, and so on. OpenStack® is an example of the Infrastructure a Service platform.

Cloud Operating System

- OpenStack® provides pools of compute, networking, and storage resources
- Abstraction of the “cloud operating system” that:
 - Is capable of scaling to thousands of computes
 - Provides a secure remote access to all resources through open APIs (Application Programming Interfaces)
 - Offers a web- and command line-based user interface for end users and administrators
- The end user server instances running in the cloud are available, over the private or public network



© Copyright 2017 Hewlett Packard Enterprise Development LP

7

Cloud Operating System

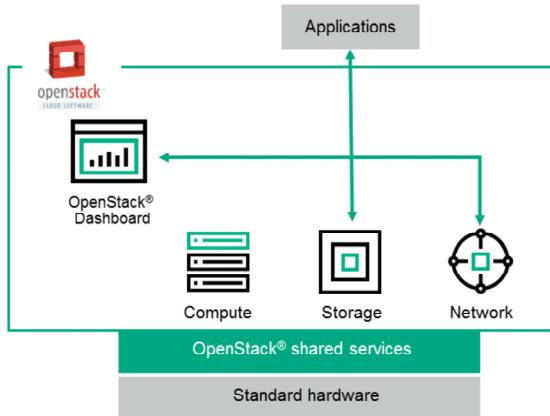
When you begin to observe data centers as pools of compute, networking, and storage resources, they start to resemble the basic design of a computer. Any computer, whether a tiny embedded device, a powerful server, or a smartphone in your pocket, needs an operating system.

What you need now is the equivalent of Linux for the cloud. It must support a variety of hardware used in data centers and expose the data centers' capabilities via simple, standard APIs. Only then will it become interesting for developers to build apps, services, and additional content that enhance the platform.

Examples of such add-ons for a cloud operating system include drivers for new storage hardware, application monitoring tools, energy efficiency monitoring, prebuilt virtual images, prebuilt infrastructure templates for readymade mobile app backends, and more.

What is OpenStack®?

- Open source IaaS (Infrastructure as a Service) implementation
- Free open source (Apache license) software governed by a non-profit foundation called the OpenStack® Foundation
- Ongoing, significant code contributions from many industry leaders, including HPE
- Pluggable and hardware-agnostic architecture
- It began as a joint project between NASA (Nebula platform) and Rackspace (Cloud Files platform)



What is OpenStack®?

OpenStack® is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface. OpenStack® is architected to provide flexibility as you design your cloud, with no proprietary hardware or software requirements and the ability to integrate with legacy systems and third-party technologies. It is designed to manage and automate pools of compute resources and can work with widely available virtualization technologies, as well as bare metal and high-performance computing (HPC) configurations.

Compute

Administrators often deploy OpenStack® Compute using one of multiple supported hypervisors in a virtualized environment. KVM and XenServer are popular choices for hypervisor technology and recommended for most use cases. Linux container technology such as LXC is also supported for scenarios when you want to minimize virtualization overhead and achieve greater efficiency and performance. In addition to different hypervisors, OpenStack® supports ARM and alternative hardware architectures.

Storage

In addition to traditional enterprise-class storage technology, many organizations now have a variety of storage needs with varying performance and price requirements. OpenStack® has support for both Object Storage and Block Storage, with many deployment options for each, depending on the use case. Object Storage is ideal for cost-effective, scale-out storage. It provides a fully distributed, API-accessible storage platform that can be integrated directly into applications or used for backup, archiving, and data retention. Block Storage allows block devices to be exposed and connected to compute instances for expanded storage, better performance, and integration with enterprise storage platforms.

Network

Today's datacenter networks contain more devices than ever before including servers, network equipment, storage systems, and security appliances, many of which are further divided into virtual machines and virtual networks. The number of IP addresses, routing configurations, and security rules can quickly grow into the millions. Traditional network management techniques fall short of providing a truly scalable, automated approach to managing these next-generation networks. At the same time, you expect more control and flexibility with quicker provisioning.

OpenStack® Networking is a pluggable, scalable, and API-driven system for managing networks and IP addresses. Like other aspects of the cloud operating system, you can use it to increase the value of existing datacenter assets. OpenStack® Networking ensures the network will not be the bottleneck or a limiting factor in a cloud deployment, and gives you real self service, even over your network configurations.

OpenStack® benefits

- **Automation for IT**
 - Self-service, programmatic access to infrastructure
- **Converged cloud deployment model**
 - Support for federated public and private deployments
- **Accelerated time to market and innovation**
 - Benefit from community contributions
 - Ecosystem of cloud solutions
 - Ease of integration and management
 - Open, web-based APIs
- **Portability of workloads**
 - Across public, private, and hybrid OpenStack®-based clouds
- **No vendor lock-in**
 - Not tied to a road map of a single vendor
 - Contributions from wide array of SMEs
 - Developed for a broad set of use cases
- **Open source**
 - OpenStack® is licensed under OSS Apache license
 - Open platform to develop value-added solutions



© Copyright 2017 Hewlett Packard Enterprise Development LP

10

OpenStack® benefits

Clouds can lead to improved server utilization over conventional server virtualization. With an OpenStack® cloud solution, you can move workloads across public, private, and hybrid-based clouds without any limitations.

All cloud platforms offer similar benefits, but there are advantages to OpenStack® because of the way this platform was created and developed.

The OpenStack® platform:

- Brings automation through self-service, programmatic access to infrastructure
- Supports federated public and private deployments
- Benefits from all of the community and vendor contributions:
 - It was developed with a broad set of use cases in mind
 - It brings transparency into governance, road maps, blueprints, and development

The OpenStack® software:

- Is developed under the open source software (OSS) Apache license, which allows developers and companies unrestricted access and freedom to build their own value-added solutions based on OpenStack®
- Is built from contributions from many subject matter experts (SMEs), which eliminates vendor lock-in

OpenStack® organization

– **The OpenStack® Foundation** is an independent body providing shared resources to help achieve the OpenStack® mission by protecting, empowering, and promoting OpenStack® software and the community around it, including users, developers, and the entire ecosystem

– **The Board:**

- Provides legal management of the Foundation and its financial resources
- Oversees Foundation operations
- Sets overall budget and goals for the Foundation staff
- Hires the Executive Director
- Individual Members elect 1/3 of the seats; Gold Members elect 1/3 of the seats; and Platinum Members appoint 1/3 of the seats

PTL
(project technical leads)

- Elected by a body of project contributors, responsible for the direction of each project

Technical Committee

- Elected members who set technical policies that cross projects and determine the incubation of new projects

User Committee

- A formal mechanism to provide input on user priorities and user feedback and to raise important issues

Legal Affairs Committee

- Advises on legal requirements and the overall intellectual property strategy

 Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

11

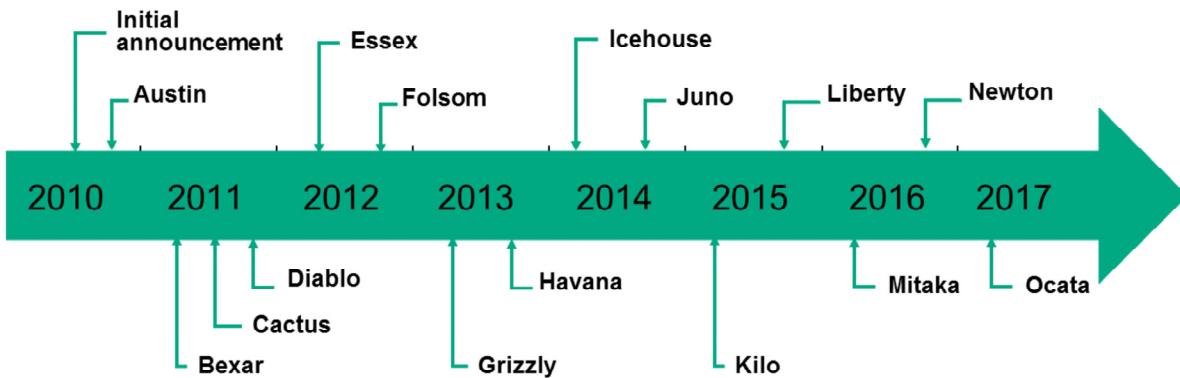
OpenStack® organization

The OpenStack® Foundation is an independent body providing shared resources to help achieve the OpenStack® mission by protecting, empowering, and promoting the OpenStack® software and the community around it, including users, developers, and the entire ecosystem. The foundation comprises several groups:

- **Project Technical Leads (PTLs)** are responsible for the direction of each project.
- The **Technical Committee** sets technical policies across projects and determines the incubation of new projects.
- The **User Committee** sets user priorities, analyzes user feedback, and raises important issues.
- The **Legal Affairs Committee** advises on legal requirements and the overall intellectual property strategy.

OpenStack® release history

- OpenStack® releases are every 6 months and are named alphabetically



NOTE: The OpenStack® contribution analysis is available at: <http://stackalytics.com/?release=newton&metric=commits>.

OpenStack® release history

The initial code for OpenStack® came from NASA's Nebula platform and Rackspace's Cloud Files that were being independently developed. When NASA and Rackspace became aware of each other's cloud platform development effort, they agreed to combine their efforts. The Nebula project of NASA became the initial code for Nova, the compute project of OpenStack®, and much of Rackspace's Cloud Files code evolved into what is now Swift, the object storage project of OpenStack®.

In mid-2010, the OpenStack® Foundation was established with more than 25 contributors. The Austin release of OpenStack® was released three months after OpenStack® was formed.

The OpenStack® community collaborates around a six-month, time-based release cycle with frequent development milestones. During the planning phase of each release, the community gathers for an OpenStack® Design Summit to facilitate developer working sessions and to assemble plans for future development.

The previous, current, and future releases are shown in the timeline on the above slide. OpenStack® releases are named alphabetically.

Before being released, every release passes two phases:

- **Development:** The code base is under development and is not recommended for production.
- **Stable or Security Fixes:** Stable branches are centralized efforts to maintain bug and vulnerability fixes for released OpenStack® project versions and ready-to-use branches.

As shown, the current release is Newton, and the next release of OpenStack® is Ocata.

The link on the above slide details the corporate contributions to OpenStack®.

OpenStack® documentation

- <http://docs.openstack.org>

The screenshot shows the OpenStack documentation homepage. It features three main sections: 'Installation Tutorials And Guides', 'API Guides', and 'User Guides'. The 'API Guides' section contains a link to 'OpenStack API Documentation'. The 'User Guides' section contains links to 'End User Guide (includes Python SDK)', 'Command-Line Interface Reference', and 'Open source software for application development'. Two callout boxes highlight specific content: 'API reference material (essential for the course labs)' points to the API documentation, and 'User Guides (essential for the course labs)' points to the End User Guide and Command-Line Interface Reference.

Installation Tutorials And Guides
Getting started with the most commonly used OpenStack services
[Installation Tutorials and Guides](#)

API Guides
API Guide
OpenStack API Documentation

User Guides
End User Guide (includes Python SDK)
Create and manage resources using the OpenStack dashboard, command-line client, and Python SDK

Command-Line Interface Reference
Comprehensive OpenStack command-line reference

Open source software for application development
Resources for application development on OpenStack clouds

API reference material (essential for the course labs)

User Guides (essential for the course labs)

Hewlett Packard Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

13

OpenStack® documentation

OpenStack® documentation is available at <http://docs.openstack.org>. The documentation website provides a link to a substantial amount of documentation for installing OpenStack®, including:

- OpenStack® installation and initial configuration information for some of the more common Linux distributions
- OpenStack® service-specific configuration documentation
- OpenStack® command line interface reference documentation



HPE and the cloud

HPE history in the open source community

Areas of leadership, contribution, and sponsorship

Platform enablement: Testing, performance, management, virtualization, and security



Open source policy and governance



Printing and imaging



Tesseract OCR
and
Google™

Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

15

HPE history in the open source community—Areas of leadership, contribution, and sponsorship

The slide above describes HPE history in the open source community. Looking back, HPE has a long history of bringing open source solutions to the enterprise. It began by making Linux enterprise-ready, adding reference architectures and support for open source components. With the OpenStack® project, HPE has taken another leadership position in terms of both governance and significant code contributions. HPE is continuing to build on this track record by developing other leading technology solutions for the enterprise.

HPE continues to be a leading contributor and sponsor within the open source community. The great work includes adding testing, management, and security for Linux on HPE platforms. Additionally, HPE innovators have open-sourced many printer drivers, allowing the community to connect to HPE world-class printers through a variety of systems. In the cloud arena, where HPE has the first cloud based on a common architecture in the industry, HPE is using OpenStack® technology.

HPE Helion-related products

– <http://hpe.com/helion>

Cloud Software	Cloud Infrastructure	Managed Cloud Services
⇒ HPE Helion Cloud Suite	⇒ HPE Helion CloudSystem	⇒ HPE Helion Managed Private Cloud
⇒ HPE Helion Eucalyptus	⇒ HPE ConvergedSystem for Cloud	⇒ HPE Helion Managed Virtual Private Cloud
⇒ HPE Helion OpenStack®	⇒ HPE Cloudline	⇒ HPE Helion Managed Cloud Applications
⇒ HPE Helion Stackato	⇒ HPE Helion Content Depot	⇒ HPE Helion Managed Cloud Broker
⇒ HPE Propel	⇒ HPE Helion Rack	



© Copyright 2017 Hewlett Packard Enterprise Development LP

16

HPE Helion-related products

HPE provides a rich portfolio of cloud-related software, infrastructure, and service-related products. HPE Helion OpenStack® is the HPE flavor of OpenStack® available under the Cloud Software category on the slide above. For more information on cloud solutions, visit <http://hpe.com/helion>.



OpenStack® architecture

OpenStack® projects (as of Newton release) (1 of 2)

Project	Description
Barbican (Key Manager)	Secure storage, provisioning, and management of secrets (e.g. passwords, encryption keys)
Ceilometer (Metering)	Enables metering of various aspects of OpenStack® deployments
Cinder (Block Storage)	Provides persistent block storage volumes to guest VMs
Designate (DNS)	Provides DNSaaS services for OpenStack®
Glance (Image Repository)	Provides a catalog and repository for virtual disk images
Heat (Orchestration)	Enables provisioning of infrastructure services
Ironic (Bare Metal Provisioning)	Provides ability to provision to bare metal systems
Keystone (Security)	Provides authentication and authorization for all of the OpenStack® services
Manila (Shared File System)	Provides coordinated access to shared or distributed file systems
Neutron (Networking)	Provides network connectivity to, from, and between the OpenStack® services
Nova (Compute)	Provides virtual servers on demand
Sahara (Data Processing)	Enables provisioning of a data-intensive application cluster (Hadoop or Spark)
Searchlight (Search)	Provides searches across OpenStack® services
Swift (Object Storage)	Provides object (file) storage that is API accessible and can be referenced by URL
Trove (Database)	Provides scalable cloud Database as a Service
Zaqar (Messaging Service)	A multi-tenant cloud messaging service between various components of a SaaS and mobile applications

Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

- Projects highlighted in bold are presented in this course as separate modules with associated labs
- Projects that are not bold are discussed in other OpenStack® project modules of this course and do not have associated labs

18

OpenStack® projects (as of Newton release) (1 of 2)

OpenStack® is built by the community. Various parts of OpenStack® code are arranged in projects.

An OpenStack® project follows the ideas of “4 opens”:

- Open Source
- Open Community
- Open Development
- Open Design

An OpenStack® project:

- Ensures basic interoperability with the rest of OpenStack®
- Has an active team of one or more contributors
- Meets any policies the TC requires all projects to meet

For more information on requirements for new OpenStack® project applications, refer to the following website:
<http://governance.openstack.org/reference/new-projects-requirements.html>.

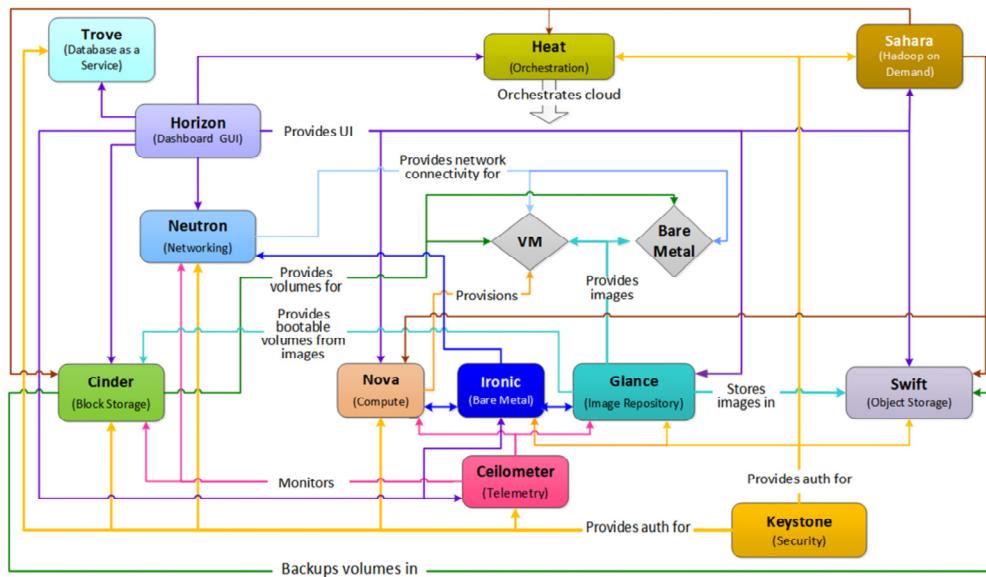
OpenStack® projects (as of Newton release) (2 of 2)

Project	Description
Aodh	Telemetry alarming service (previously part of Ceilometer)
CloudKitty	Rating-as-a-Service project
Congress	Provides policy as a service across any collection of cloud services in order to offer governance and compliance for dynamic infrastructures
Freezer	Cloud backup service
Magnum	Container orchestration engines (for example, Docker, Swarm, Kubernetes, and so on)
Mistral	The OpenStack® workflow service
Monasca	Multi-tenant, highly scalable, fault-tolerant monitoring-as-a-service solution that integrates with OpenStack®
Murano	Application catalog to OpenStack®
Panko	Event storage and REST API for Ceilometer
Searchlight	Advanced and scalable indexing and search across multi-tenant cloud resources
Senlin	Generic clustering service for an OpenStack® cloud
Solum	A project designed to make cloud services easier to consume and integrate into your application development process
Tacker	Virtual Network Functions Manager (VNFM) and a NFV (Network Function Virtualization) Orchestrator (NFVO)
Vitrage	A solution that analyzes the alarms and events to provide a root cause analysis service
Watcher	A flexible and scalable resource optimization service for multi-tenant OpenStack®-based clouds

OpenStack® projects (as of Newton release) (2 of 2)

The slide above lists the remaining projects available in Newton release of OpenStack®, in addition to the previous slide.

Conceptual architecture



Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

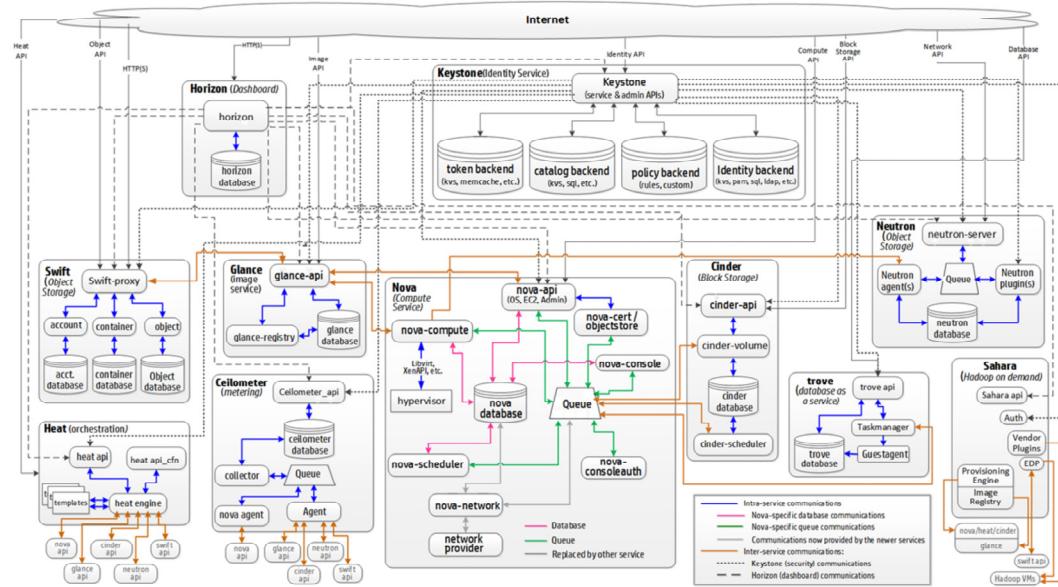
20

Conceptual architecture

The above conceptual architecture illustration shows how the primary core components of OpenStack® are interconnected:

- **Swift** provides object storage. It allows you to store or retrieve files (but not mount directories like a fileserver). Several companies provide commercial storage services based on Swift. These include KT, Rackspace (from which Swift originated) and Internap. Swift is also used internally at many large companies to store their data.
- **Glance** provides a catalog and repository for virtual disk images. These disk images are most commonly used in OpenStack® Compute. While this service is technically optional, any large cloud requires it.
- **Nova** provides virtual servers upon demand. Rackspace and HPE provide commercial compute services built on Nova. Nova is used internally at companies like Mercado Libre and NASA (where it originated).
- **Horizon** provides a modular web-based user interface for performing most cloud operations, such as launching an instance, assigning IP addresses, and setting access controls.
- **Keystone** provides authentication and authorization for all OpenStack® services. Keystone also provides a service catalog of services within a particular OpenStack® cloud.
- **Neutron** provides "network connectivity as a service" between interface devices managed by other OpenStack® services (most likely Nova). Neutron allows you to create your own networks and then attach interfaces to them. OpenStack® Network has a pluggable architecture to support many popular networking vendors and technologies.
- **Cinder** provides persistent block storage to guest VMs.
- **Ceilometer** provides metering of the various aspects of OpenStack® deployments.
- **Heat** is a template-based orchestration solution for provisioning the resources required to deploy a cloud.
- **Trove** is a database service.

OpenStack® inter-service communications



Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

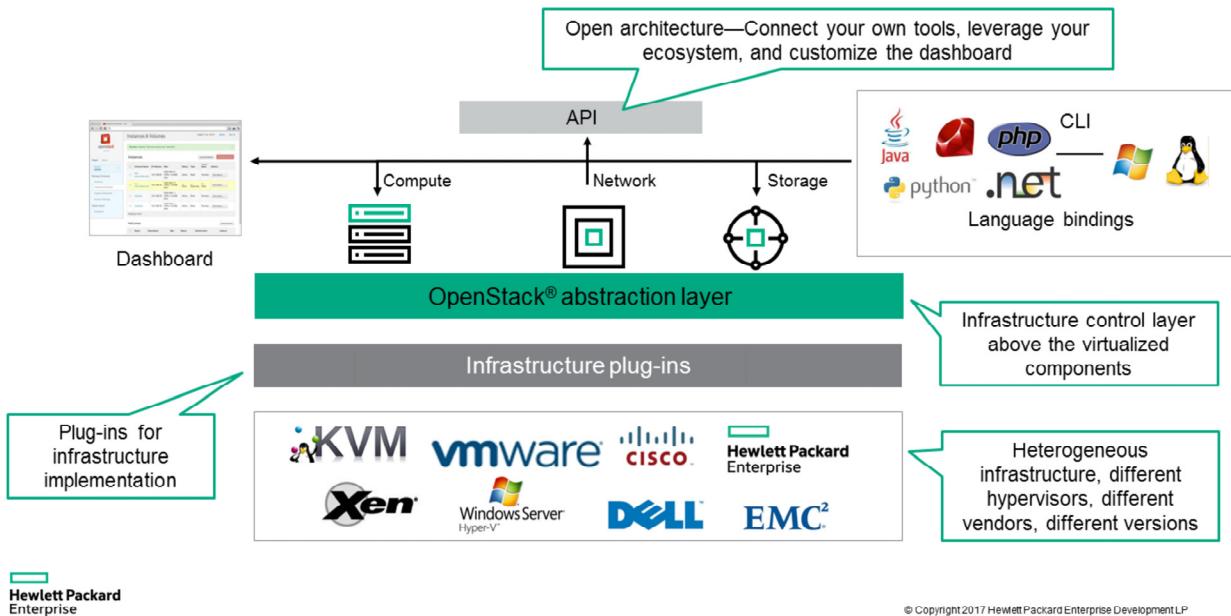
21

OpenStack® inter-service communications

The above logical diagram shows the major services of OpenStack® and how they are interconnected. The diagram also shows the inter-service communications between various components within each of the services. The functions of the service components will be discussed in subsequent modules of this course. Here are several points of interest about this diagram:

- End users can interact through a common web interface (Horizon) or directly to each service through their API.
- All services authenticate through a common source (facilitated through Keystone).
- Individual services interact with each other through their public APIs (except where the privileged administrator commands are necessary).

High-level architecture



22

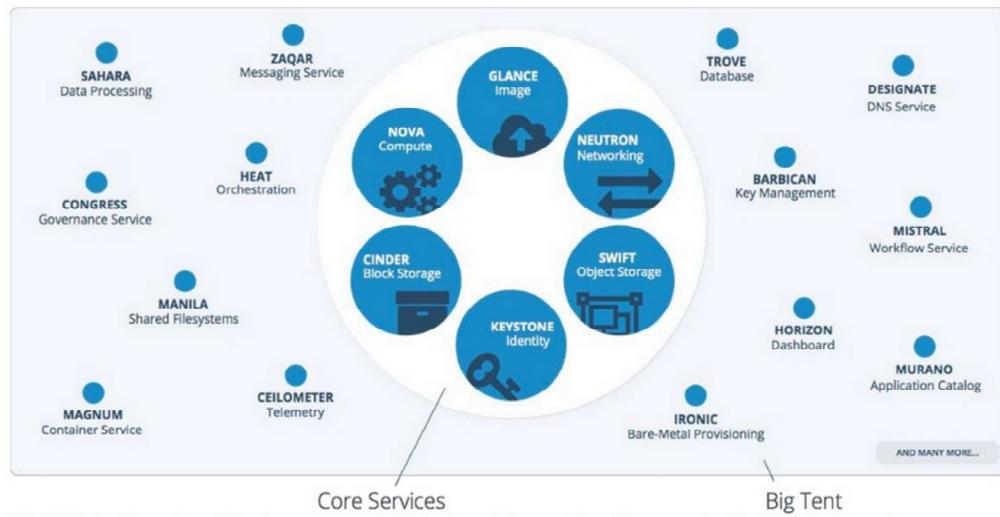
High-level architecture

A cloud operating system requires components such as drivers for new storage hardware, application monitoring tools, energy efficiency monitoring, prebuilt virtual images, prebuilt infrastructure templates for readymade mobile app backends, and more.

The compute, network, and storage pictures in the diagram represent the major components of OpenStack®. This diagram illustrates how storage, networking, and hypervisors from various vendors can be incorporated into OpenStack® through the use of the OpenStack® abstraction layer.

Many infrastructure plug-ins for that abstraction layer already exist for some of the more common services. The OpenStack® APIs enable you to easily develop plug-ins for the ever-growing number of cloud components that can be integrated with OpenStack®.

OpenStack® big tent and core services



Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

23

OpenStack® big tent and core services

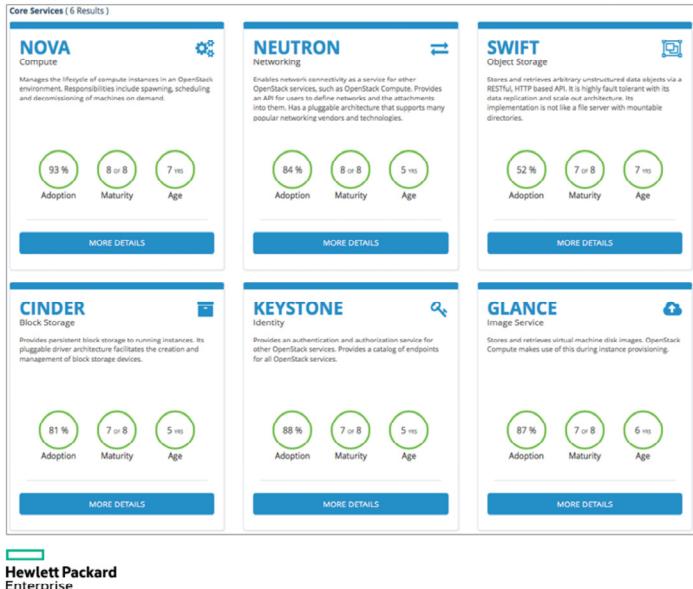
Changes toward a “big tent” model endeavor to:

- More clearly define what it means to be an OpenStack® project
- Be more broadly inclusive than projects that meet this definition, even those which compete with each other.
- Provide a more precise taxonomy for navigating the ecosystem of OpenStack® projects

The following tags are introduced to help operators navigate all the projects that make up OpenStack®:

- Technical Committee Managed
- Team Description
- Project Assertions
- Release Management
- Vulnerability Management (<http://governance.openstack.org/reference/tags/index.html>)

OpenStack® project adoption, maturity, and age (1 of 2)



– Most up-to-date project adoption, maturity, and age numbers are at:
<https://www.openstack.org/software/project-navigator/>

Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

24

OpenStack® project adoption, maturity and age (1 of 2)

The most up-to-date project adoption, maturity, and age numbers can be obtained from <https://www.openstack.org/software/project-navigator/>. The slide above shows the information for the core OpenStack® services.

Adoption is the percentage of production deployments running the project based on the latest biannual user survey results.

Maturity comes from looking at 8 distinct tags that indicate stability and sustainability. The current criteria includes whether or not the project has an install guide, whether it is supported by 7 or more SDKs, if the adoption percentage is greater than 75%, whether or not the team has achieved corporate diversity and whether or not there are stable branches.

Age is the number of years the project has been in development.

OpenStack® project adoption, maturity, and age (2 of 2)

Optional Services (13 Results)				
NAME	SERVICE	MATURITY	AGE	ADOPTION
Horizon	Dashboard	6 of 8	5 Yrs	86 %
Ceilometer	Telemetry	1 of 8	4 Yrs	59 %
Heat	Orchestration	5 of 8	4 Yrs	64 %
Trove	Database	3 of 8	3 Yrs	17 %
Sahara	Elastic Map Reduce	4 of 8	3 Yrs	11 %
Ironic	Bare-Metal Provisioning	5 of 8	3 Yrs	20 %
Zaqar	Messaging Service	3 of 8	3 Yrs	2 %
Manila	Shared Filesystems	5 of 8	3 Yrs	11 %
Designate	DNS Service	3 of 8	3 Yrs	17 %
Barbican	Key Management	1 of 8	3 Yrs	2 %
Magnum	Containers	1 of 8	2 Yrs	11 %
Murano	Application Catalog	2 of 8	2 Yrs	15 %
Congress	Governance	1 of 8	2 Yrs	1 %

 Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

25

OpenStack® project adoption, maturity, and age (2 of 2)

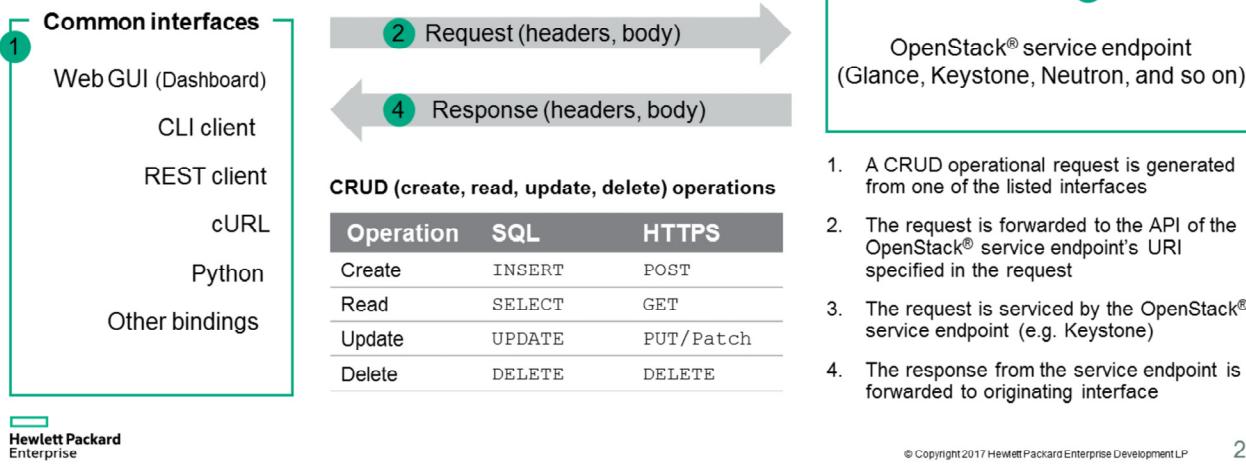
The slide above displays the adoption, maturity, and age information for the remaining optional projects.



OpenStack® service APIs

Communication to and between OpenStack® projects

- Every OpenStack® project is designed to have minimal dependency on any other OpenStack® project
- APIs (Application Programming Interfaces) are used for communications between the OpenStack® services and from a user/app to an OpenStack® service



Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

27

Communication to and between OpenStack® projects

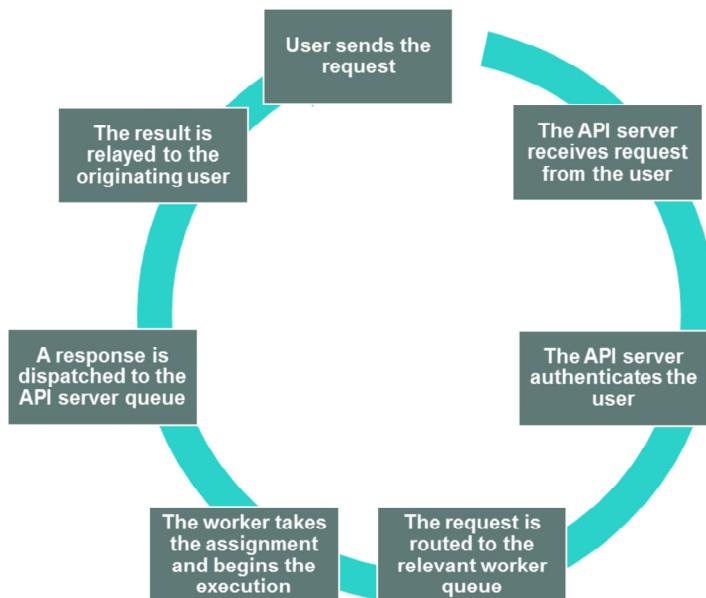
OpenStack® has several supported API interfaces through which requests can be made and responses received from service endpoints. The endpoints are OpenStack® services such as Glance, Keystone, and Neutron.

Those interfaces include:

- Web dashboard GUI, referred to as Horizon, an OpenStack® web-based management GUI
- The CLI client, which supports the use of a command line
- Representational State Transfer (REST) client
- cURL
- Python code
- Other applications, such as database workbenches

The OpenStack® API supports the standard `create`, `read`, `update`, and `delete` (CRUD) operations, which are known as `post`, `get`, `put`, and `delete` from an HTTPS perspective.

Message lifecycle



 Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

28

Message lifecycle

A typical message-passing event begins with the API server receiving a request from a user.

The API server then authenticates the user and ensures that the user is permitted to issue the subject command.

The availability of the objects implicated in the request is evaluated and, if available, the request is routed to the queuing engine for the relevant workers. Workers continually listen to the queue based on their role and occasionally their type or host name.

When an applicable work request arrives on the queue, the worker takes assignment and begins to execute the task.

Upon completion, a response is dispatched to the queue. It is received by the API server and then relayed to the originating user. Database entries are queried, added, or removed as necessary throughout the process.

OpenStack® service API versioning

- Features or functionality changes to an OpenStack® service that would necessitate a break in API compatibility require a new version
- An OpenStack® service API is in one of the following supported states:
 - **Current**: The most current version of the service
 - **Supported**: Supported but does not include the latest features and functions
 - **Deprecated**: No longer supported
- For example: The Keystone API v1.1 is **deprecated**, Keystone API v2.0 is **supported**, and Keystone API v3.0 is in a **current** state of support



© Copyright 2017 Hewlett Packard Enterprise Development LP

29

OpenStack® service API versioning

Each release of OpenStack® introduces a number of new features and functionality to existing OpenStack® services (for example, Keystone or Nova) and in some releases new services are introduced. Each OpenStack® service has a unique API that provides access to the features and functions of the service. Also, each OpenStack® service has its own unique API versioning scheme.

New features and functionality introduced in an OpenStack® service that do not break the API compatibility are introduced in the current version of the API as extensions. Features or functionality changes to an OpenStack® service that would necessitate a break in the API compatibility require a new version.

An OpenStack® service API is in one of the following supported states:

- **Current**: The most current version of the service
- **Supported**: Supported, but does not include the latest features and functions
- **Deprecated**: No longer supported

REST API microversion support

- The following types of changes are generally considered acceptable:
 - The change is the only way to fix a security bug
 - Fixing a bug so that a request which resulted in an error response before is now successful
 - Adding a new response header
 - Changing an error response code to be more accurate
- Specify the version of the API you want via a new HTTP header using the `X-OpenStack-Nova-API-Version` field

NOTE: API changes are governed by the API working group guideline. For more information, visit http://specs.openstack.org/openstack/api-wg/guidelines/evaluating_api_changes.html.



© Copyright 2017 Hewlett Packard Enterprise Development LP

30

REST API microversion support

REST API microversion support provides the ability for the OpenStack® REST API to:

- Introduce changes to the REST API, such as fixing bugs and adding new features
- Provide backward compatibility for users of the REST API
- Support changes that are **not** backward-incompatible
- Allow developers to modify the API in a backward-compatible way
- Signal the users of the API that the change is available and that there is no need to create a new API extension

Users of the REST API are able to decide if they want the REST API to behave in the new or old manner on per request basis. You can specify the version of the API you want via a new HTTP header using the following syntax:

`X-OpenStack-Nova-API-Version: <MicroVersion (e.g.) 2.114>`

API changes are governed by the API working group guideline. For more information, visit http://specs.openstack.org/openstack/api-wg/guidelines/evaluating_api_changes.html.

The screenshot shows the OpenStack API Documentation page. At the top, there's a navigation bar with links like SOFTWARE, USERS, COMMUNITY, MARKETPLACE, EVENTS, LEARN, and DOCS. Below the navigation bar, there's a search bar and a 'OpenStack APIs' button. A green arrow points from the 'OpenStack APIs' button to a dropdown menu on the right. The dropdown menu is titled 'OpenStack APIs' and contains sections for REFERENCES (Bare Metal, Clustering, Compute, Data Processing, DNS, Identity, Image, Messaging Service, Networking, Object Storage, Orchestration, Shared File Systems) and GUIDES (API Quick Start, Compute API Guide). At the bottom left, there's a Hewlett Packard Enterprise logo, and at the bottom right, it says '© Copyright 2017 Hewlett Packard Enterprise Development LP'.

OpenStack® API reference materials

The OpenStack® API reference materials web page is available at <http://developer.openstack.org/api-guide/quick-start/>. At the top of the navigation bar on the left-hand side of the documentation page, click the **OpenStack APIs** button to display the selection of available references and guides.

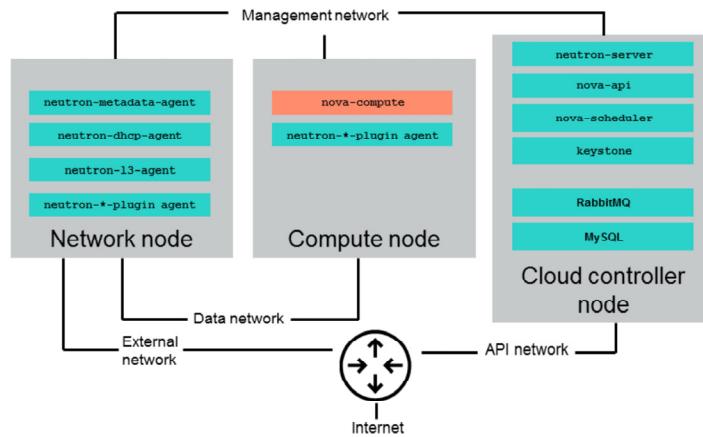


OpenStack® installation and configuration

Basic OpenStack® physical components

- This example three-node configuration shows the basic components of an OpenStack® installation:
 - Cloud Controller node
 - Network node
 - One or more compute nodes

NOTE: All of the basic components can exist on the same device or be horizontally distributed with multiple network, compute, and controller nodes, as shown on the following slide.



Basic OpenStack® physical components

OpenStack® is designed to be massively horizontally scalable, which allows all services to be widely distributed. According to the OpenStack® documentation, there are three primary functional components of OpenStack®:

- The **cloud controller** is a functional unit that provides the OpenStack® Nova, Keystone, Glance, Cinder, Swift, and dashboard services, as well as hosting of the databases and message queue services.
- The **network node** generally provides the virtual bridging, Dynamic Host Configuration Protocol (DHCP) server, virtual routing services, and plug-in agents for the associated network hardware.
- The **compute node** generally provides the hypervisor, the nova-compute to communicate with the cloud controller node, and the neutron plug-in agent to communicate with the network node.

Also, four types of functional networks are identified:

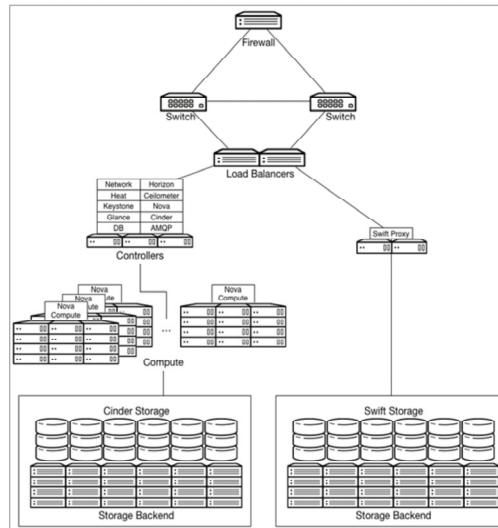
- Management network—Provides communications between and to the OpenStack® services
- API network—Supports the standard CRUD operations
- Data network—Provides the data stream between the network nodes and the compute nodes
- External network

All of these functional components can exist on a single server, such as the one used for the lab activities for this training, or they can be distributed throughout the cloud across multiple servers.

Wide range of configuration options

- Types of configurations:
 - General purpose
 - Compute-focused
 - Storage-focused
 - Network-focused
 - Multi-site
 - Hybrid
 - Massively scalable
 - Specialized cases

NOTE: The picture on the right shows a general purpose configuration diagram, which can be found at <http://docs.openstack.org/arch-design>.



Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

34

Wide range of configuration options

OpenStack® architecture allows for wide range of configuration options. The *OpenStack Architecture Design Guide* available at <http://docs.openstack.org/arch-design/> offers the following types of configurations:

- **General purpose**—As a baseline product, general purpose clouds do not provide optimized performance for any particular function.
- **Compute-focused**—Configuration that is primarily driven by the type of instance workloads you provision in the OpenStack® cloud.
- **Storage-focused**—Addresses a range of storage management-related considerations, as well as storage focused input-output performance, security, and scalability requirements.
- **Network-focused**—A configuration model that takes in consideration the content delivered over the network, management functions, service offerings and connected web portals, network high availability, and so on.
- **Multi-site**—A configuration model that provides the OpenStack® capable of running in a multi-region configuration.
- **Hybrid**—Configuration design providing the OpenStack® that uses more than one cloud. Hybrid clouds are designs that use, for example, both an OpenStack®-based private cloud and an OpenStack®-based public cloud, or an OpenStack® cloud and a non-OpenStack® cloud.
- **Massively scalable**—A massively scalable architecture is a cloud implementation that supports a very large deployment. An example is an infrastructure in which requests to service 500 or more instances at a time.
- **Specialized cases**—This includes specialized networking and software-defined networking configurations, Desktop-as-a-Service and OpenStack®-on-OpenStack® deployments, as well as the deployments that support specialized hardware.

Primary types of OpenStack® installations

- Manual—Generally through a Linux distribution-specific installation guide
- Scripted Installers—Used to create a core development environment
 - Example: devstack, which is the method used in the labs associated with this course
- Deployment frameworks (for example, Triple 0, Chef, Puppet, JuJu)
- Commercial distributions (for example, Canonical, Mirantis, Red Hat, HPE Helion)



© Copyright 2017 Hewlett Packard Enterprise Development LP

35

Primary types of OpenStack® installations

Triple 0 is an official OpenStack® program with the goal of allowing you to deploy a production cloud onto bare metal hardware using a subset of existing OpenStack® components.

Chef is a configuration management tool used to streamline the task of configuring and maintaining a company's servers. Chef can integrate with cloud-based platforms (e.g. OpenStack®, Rackspace, and Amazon EC2) to automatically provision and configure new machines. Chef contains solutions for both small and large scale systems, with features and pricing for the respective ranges.

Puppet is similar in function to Chef and sold by Puppet Labs.

JuJu is an open source, solution-driven orchestration tool from Ubuntu that helps you deploy, manage, and scale your environments on any cloud.

OpenStack® development environment (DevStack)

- What is DevStack?
 - A documented shell script used to quickly create an OpenStack® development environment
 - Can be easily modified to include specific OpenStack projects
- DevStack is **not**:
 - A general OpenStack® installer
 - To be used in a production environment

Resource	Link
Documentation	http://devstack.org
FAQ	http://devstack.org/faq.html
GitHub	https://github.com/openstack-dev/devstack
LaunchPad	https://launchpad.net/devstack

NOTE: Your lab environment for this course uses DevStack.



© Copyright 2017 Hewlett Packard Enterprise Development LP

36

OpenStack® development environment (DevStack)

DevStack is a script used to quickly create an OpenStack® development environment. It can also be used to demonstrate how to start and run OpenStack® services and to provide examples of how to use them from a command line or GUI. Some example exercises were fleshed out beyond simple examples, and they became useful as a quick “sanity check” for an OpenStack® installation.

DevStack is not and never has been intended to be a general OpenStack® installer. It has evolved to support a large number of configuration options, as well as alternative platforms and support services.

Scripted DevStack installation process

1. Select a Linux Distribution

NOTE: Only Ubuntu 14.04 (Trusty), Fedora 20, and CentOS/RHEL 6.5 are documented.

2. Install the OS

3. To install git, enter one of these commands:

```
$ sudo apt-get install git
$ sudo yum install git
```

4. Download DevStack:

```
$ sudo git clone
git://github.com/openstack-
dev/devstack.git
```

NOTE: The DevStack repo contains a script that installs OpenStack® and the templates for configuration files.

5. Start the DevStack installation:

```
$ cd devstack; ./stack.sh
```

NOTE: This process takes 30 to 60 minutes, depending on the bandwidth.



© Copyright 2017 Hewlett Packard Enterprise Development LP

37

Scripted DevStack installation process

A shell script can also be used to install and configure OpenStack®. One of the more common examples of this, and one that will be used in your lab environment, is DevStack, which uses a script to install a complete OpenStack® developer environment. To install OpenStack®, follow these steps:

1. Select a Linux distribution.

NOTE: Currently, documentation only exists for Linux Ubuntu 14.04 LTS, Fedora 20, and CentOS/RHEL 6.5.

2. Install the OS.

3. Install git on your OS. The installation command you enter depends on the Linux distribution, as shown in the example.

4. Download DevStack by entering the command shown in step 4 of the slide. In the command, git is a code repository for open source applications such as OpenStack®. The DevStack download will include the script and several configuration files, each of which can be modified for the DevStack implementation of OpenStack®.

5. Start the DevStack installation, which entails changing to the DevStack directory and then running the `./stack.sh` script.

Much of the installation process requires access to an Internet connection; therefore, your installation time is dependent on your network bandwidth.

Example local.conf file for devstack installation

```

1 [[local|localrc]]
2 HOST_IP=192.168.1.5
3 SERVICE_HOST=$HOST_IP
4
5 IP_VERSION=4
6 SERVICE_IP_VERSION=$IP_VERSION
7
8 ADMIN_PASSWORD=hadmin
9 DATABASE_PASSWORD=hadmin_password
10 SERVICE_TOKEN=hadmin_token
11 SERVICE_PASSWORD=hadmin_password
12 RABBIT_PASSWORD=hadmin_password
13 MYSQL_PASSWORD=hadmin_password
14
15 # ceilometer Settings
16 # enable_service ceilometer-compute ceilometer-central ceilometer-analyzer ceilometer-collector
17 enable_plugin ceilometer https://git.openstack.org/openstack/ceilometer.git
18 enable_plugin wodl https://git.openstack.org/openstack/wodl
19 enable_service aodh-evaluator,aodh-notifier,aodh-api
20 disable_service ceilometer-alarm-notifiers,ceilometer-alarm-evaluator
21
22 # Cinder Settings
23 # ENABLED_SERVICES+=,cinder,c-api,c-vol,c-sch,c-bak
24
25 # Glance Settings
26
27 # Images
28 # Use this image when creating test instances
29 IMAGE_URL=http://download.cirros-cloud.net/0.3.2/cirros-0.3.2-x86_64-disk.img
30 # Use this image when working with Orchestration (Heat)
31 IMAGE_URL="http://download.cloudcontroller.org/pub/fedora/linux/releases/23/Cloud/x86_64/Images/Fedora-Cloud-Base-23-20151030.x86_64.qcow2"
32
33 # Heat Settings
34 # ENABLED_SERVICES+=,heat,h-api,h-api-cfn,h-api-cw,h-eng
35
36 # Neutron settings
37 # ENABLED_SERVICES+=,q-svc,q-agt,q-dhcp,q-l3,q-metas,neutron,q-metering,q-lbaas
38 # Q_ML2_INI=ml2_neutron.ini
39 # Q_DVR_MODE=dvr_nat
40
41 # Nova Settings
42 # ENABLED_SERVICES+=,n-cauth,n-novnc
43 disable_service n-net
44
45 # Swift Settings
46 # ENABLED_SERVICES+=,s-proxy,s-object,s-container,s-account

```

Hewlett Packard
Enterprise

IP address of the DevStack VM

Passwords to be used by various
OpenStack® components

OpenStack® project
services to be installed

Images that can be
downloaded to the
DevStack install for use in
creating instances

- The local.conf file is used to provide the install-specific parameters to the stack.sh script as it is installing DevStack

- Information about each of the parameters that can be specified in the local.conf file are located at <http://docs.openstack.org/developer/devstack/configuration.html>

© Copyright 2017 Hewlett Packard Enterprise Development LP

38

Example local.conf file for devstack installation

The local.conf file is used to provide the install-specific parameters to the stack.sh script as it is installing DevStack.

Information about each of the parameters that can be specified in the local.conf file are located at <http://docs.openstack.org/developer/devstack/configuration.html>.

Installing an OpenStack® sandbox manually

1. Prepare the nodes (compute, controller, network, and so on):
 - a. Install the OS
 - b. Configure the networking
 - c. Install and configure a database such as MySQL
 - d. Queue the necessary services such as RabbitMQ
 - e. Set up any other networking (such as bridge utilities and IP forwarding)
2. Configure each endpoint service
 - a. Install the service components
 - b. Configure each component:
 - Create the component MySQL database
 - Edit the configuration files
 - Synchronize the database to create tables
 - c. Validate
 - Restart the service
 - Test the service

Keystone
Glance
Neutron
Nova
Cinder
Swift
Horizon
Swift
Heat
Ceilometer
Trove
Sahara

TIP: For more details, see the official *OpenStack Configuration Reference* document for the Newton release, which is available at <http://docs.openstack.org/newton/config-reference/>.

Installing an OpenStack® sandbox manually

OpenStack® is usually installed manually by following the documentation for different Linux distributions given on one of the previous slides. In general, the OpenStack® installation procedure has two main steps:

1. Prepare the compute, controller, and network nodes. Within this step, you have to install the OS, configure the networking, install and set up a database such as MySQL, queue the services such as RabbitMQ, and set up any other networking functionality, such as bridge utilities and IP forwarding.
2. To configure each of the endpoint services that will be used, such as Keystone, Glance, and Neutron, install each component, configure it, and then verify that it is operational.

Module summary

In this module:

- The OpenStack® offering and the OpenStack® cloud service model were discussed
- Some significant events in the history of OpenStack® were highlighted
- Architectural components of OpenStack® were identified
- OpenStack® installation and configuration were explained
- DevStack installation and usage were explained



© Copyright 2017 Hewlett Packard Enterprise Development LP

40

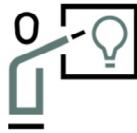
Module summary

In this module:

- The OpenStack® offering and the OpenStack® cloud service model were discussed
- Some significant events in the history of OpenStack® were highlighted
- Architectural components of OpenStack® were identified
- OpenStack® installation and configuration were explained
- DevStack installation and usage were explained

Learning check

Learning check



What type of infrastructure service does OpenStack® provide?

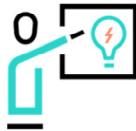
- A. IaaS
- B. PaaS
- C. SaaS
- D. MaaS

Learning check

What type of infrastructure service does OpenStack® provide?

- A. IaaS
- B. PaaS
- C. SaaS
- D. MaaS

Learning check answer



What type of infrastructure service does OpenStack® provide?

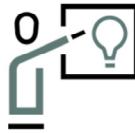
- A. IaaS
- B. PaaS
- C. SaaS
- D. MaaS

Learning check answer

What type of infrastructure service does OpenStack® provide?

- A. IaaS
- B. PaaS
- C. SaaS
- D. MaaS

Learning check



Draw a line between each OpenStack® project and its description.

Provides object (file) storage that is API accessible and URL referable

Provides network connectivity services between interface devices

Provides a web-based user interface for most of the OpenStack® services

Provides persistent block storage volumes to guest VMs

Keystone

Provides authentication and authorization for all of the OpenStack® services

Provides virtual servers on demand

Neutron

Horizon

Cinder

Glance

Nova

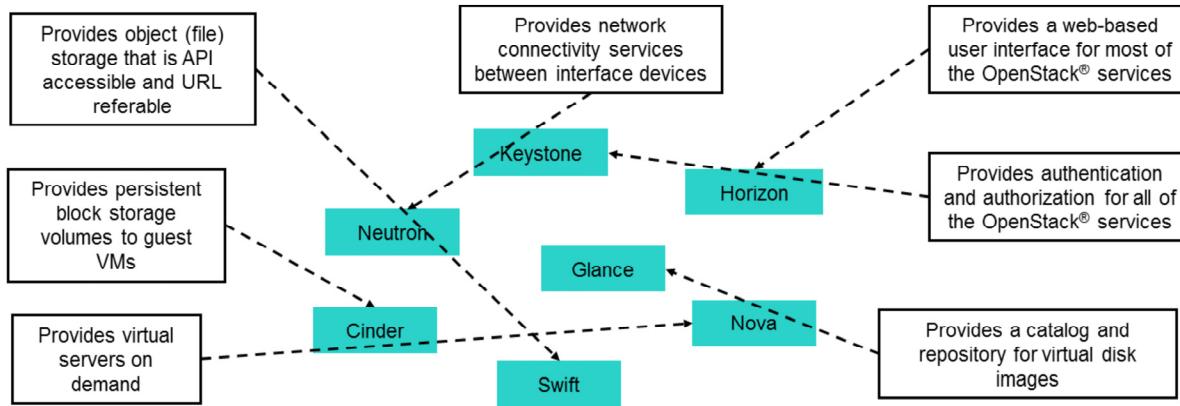
Provides a catalog and repository for virtual disk images

Swift

Learning check

Draw a line between each OpenStack® project and its description.

Learning check answer



Hewlett Packard
Enterprise

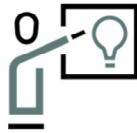
© Copyright 2017 Hewlett Packard Enterprise Development LP

45

Learning check answer

The answer is illustrated on the slide above.

Learning check



What type of OpenStack® nodes is hosting the VM instances?

- A. Network node
- B. Controller node
- C. Compute node
- D. Storage node

Learning check

What type of OpenStack® nodes is hosting the VM instances?

- A. Network node
- B. Controller node
- C. Compute node
- D. Storage node

Learning check answer



What type of OpenStack® nodes is hosting the VM instances?

- A. Network node
- B. Controller node
- C. Compute node**
- D. Storage node

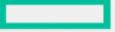
Learning check answer

What type of OpenStack® nodes is hosting the VM instances?

- A. Network node
- B. Controller node
- C. Compute node**
- D. Storage node



**Hewlett Packard
Enterprise**

 **Hewlett Packard
Enterprise**

Fundamentals of OpenStack® Technology

Module 3—OpenStack® Management Tools

H6C68S E.01

© Copyright 2017 Hewlett Packard Enterprise Development LP



Learning objectives

After completing this module, you should be able to:

- Describe the OpenStack® RESTful API (Application Programming Interface) access methods
- Use a REST Client browser plugin to manage OpenStack® services
- Use a cURL to manage OpenStack® services
- Use the OpenStack® CLI to manage OpenStack® services
- Use the Horizon (dashboard) UI to manage OpenStack® services
- Explain how Python can be used to manage OpenStack® services



© Copyright 2017 Hewlett Packard Enterprise Development LP

2

Learning objectives

After completing this module, you should be able to:

- Describe the OpenStack® RESTful API (Application Programming Interface) access methods
- Use a REST Client browser plugin to manage OpenStack® services
- Use a cURL to manage OpenStack® services
- Use the OpenStack® CLI to manage OpenStack® services
- Use the Horizon (dashboard) UI to manage OpenStack® service
- Explain how Python can be used to manage OpenStack® services

Overview of OpenStack® management tools

REST architecture

- An architectural method, **not** a standard
- APIs that use REST architecture are referred to as RESTful APIs
- RESTful APIs are easy to implement, consume, and scale up
- Used by OpenStack®, and many server management applications
- To use REST:
 - Client is targeting the assigned URI (Uniform Resource Identifier)
 - Client communicates using basic HTTP operations (GET, PUT, POST, DELETE, PATCH)
 - Data is transmitted in JSON (JavaScript Object Notation) form



© Copyright 2017 Hewlett Packard Enterprise Development LP

4

REST architecture

The communication architecture used by the OpenStack® API is called REST (Representational State Transfer). REST is referred to as an architectural method (not a standard) which is used to prescribe how web-based client/server systems should be designed. APIs that use REST architecture are referred to as RESTful APIs.

Many web services in a variety of industries use RESTful APIs because they are easy to implement, easy to consume, and offer scalability advantages over previous technologies. OpenStack, and many other server management applications from HPE and other companies use RESTful APIs; REST enables hardware and management software to "understand" each other.

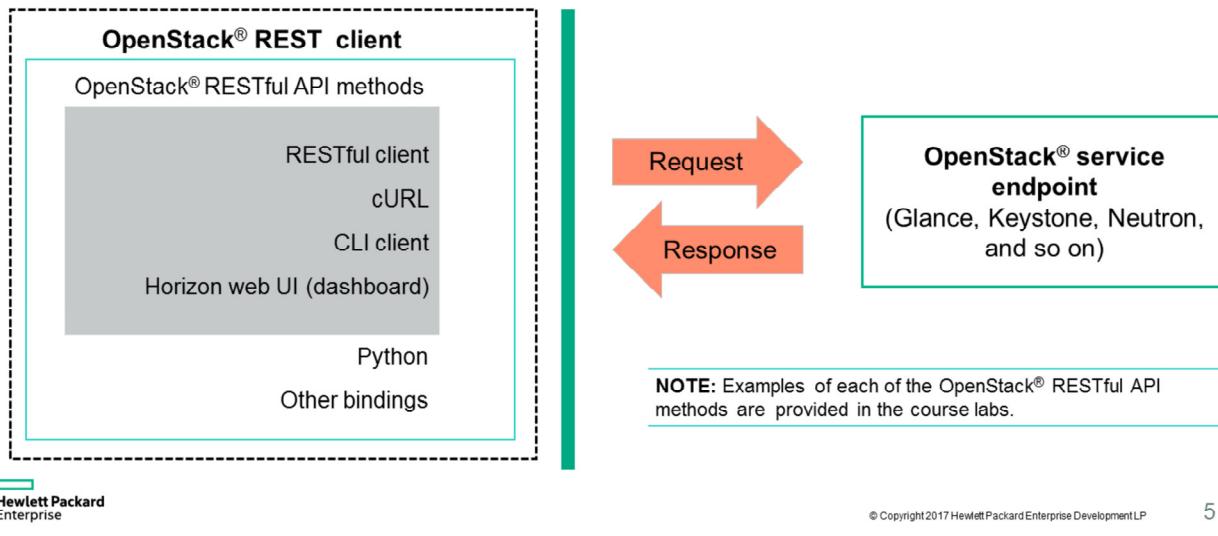
REST systems interface with external systems (e.g. OpenStack® Keystone, Nova, and so on) through their assigned URI (Uniform Resource Identifier). URLs that you are familiar with on the web are a subtype of URI. REST can use the same basic HTTP operations of GET, PUT, POST, DELETE, and similar, that web browsers use to retrieve web pages and send data to remote servers. Each HTTP operation either submits or returns a resource in JSON (JavaScript Object Notation) format; a human-readable alternative to XML.

The supported HTTP operations include:

- GET—Requests a representation of the specified resource. Requests using GET should only retrieve data and should have no other effect.
- POST—Requests that the server accept the entity enclosed in the request as a new subordinate of the web resource identified by the URI.
- PUT—Requests that the enclosed entity be stored under the supplied URI. If the URI refers to an already existing resource, it is modified; if the URI does not point to an existing resource, then the server can create the resource with that URI.
- DELETE—Deletes the specified resource.
- PATCH—Applies partial modifications to a resource.

OpenStack® RESTful API access methods

- Authentication (authorization) is required between all API requests and responses



OpenStack® RESTful API access methods

Each OpenStack® service communicates with external modules, applications, and users with standard RESTful APIs (Application Programming Interface). As illustrated in the diagram above, OpenStack® has a number of supported API interfaces through which requests can be made and through which responses from service endpoints (that is, the OpenStack® services, such as Glance, Keystone, and Neutron) can be received.

Those interfaces include:

- Horizon, an OpenStack web-based management GUI
- CLI client, which supports the use of a command line
- REST client
- cURL
- Python code
- Other applications, such as database workbenches

The OpenStack® API supports the standard CRUD operations (create, read, update, and delete) which are known as post, get, put, and delete from an HTTPS perspective.

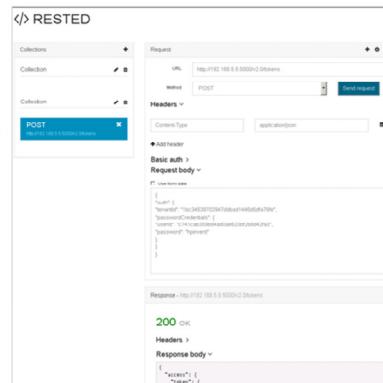
REST client browser-based UI plug-ins

- A REST client browser-based UI plugin is a REST API client that provides the ability to:
 - Specify each of the REST API parameters necessary to send a REST API request including:
 - The HTTP request method (e.g. GET, POST, PUT, and so on)
 - Content type (e.g. JSON, security tokens)
 - The URI of the server or service (e.g. OpenStack® service endpoint)
 - The body of the REST API request
 - Send the REST API request to the designated endpoint
 - Receive and display the endpoint's response to the RESTful API's request

NOTE: RESTED is used in the course labs to demonstrate how the OpenStack® API operates.

- Browser-based REST client plug-ins:

- Mozilla:
 - RESTED
 - REST client
- Chrome:
 - Rest client
 - Rest console
 - Advanced REST
 - Postman



REST client browser-based UI plug-ins

RESTED is a useful utility that allows you to construct and send custom HTTP requests to an OpenStack® endpoint service API. In the labs associated with this module, you will be using a RESTful client plug-in that has been installed on your system's Firefox browser. You will use the RESTED client to construct the OpenStack® API requests and examine their responses from the OpenStack® service endpoints.

The following REST constraint details were extracted and modified from the following document: Fielding, Roy T.; Taylor, Richard N. (2002-05), "Principled Design of the Modern Web Architecture" (PDF), ACM Transactions on Internet Technology (TOIT) (New York: Association for Computing Machinery) ISSN 1533-5399.

- Client-server: A uniform interface separates clients from servers.
- Separation of concern: For example, clients are not concerned with data storage, which remains internal to each server, so that the portability of client code is improved. Servers are not concerned with the user interface or user state, so that servers can be simpler and more scalable. Servers and clients can also be replaced and developed independently, as long as the interface between them is not altered.
- Stateless: The client-server communication is further constrained by no client context being stored on the server between requests. Each request from any client contains all of the information necessary to service the request, and session state is held in the client. Important to note is that the session state can be transferred by the server to another service such as a database to maintain a persistent state for a period of time and allow authentication.
- Cacheable: As on the World Wide Web, clients can cache responses. Responses must, therefore, implicitly or explicitly, define themselves as cacheable, or not, to prevent clients reusing stale or inappropriate data in response to further requests. Well-managed caching partially or completely eliminates some client-server interactions, further improving scalability and performance.
- Code on demand (optional): Servers can temporarily extend or customize the functionality of a client by the transfer of executable code. Examples of this might include compiled components such as Java applets and client-side scripts such as JavaScript. "Code on demand" is the only optional constraint of the REST architecture.

This is a screenshot of the RESTED plug-in for Firefox that is used in the course labs to demonstrate how the OpenStack® API operates.

Request body, as specified in the *Identity Service API Reference Guide*

OpenStack® service endpoint's URI

POST
GET
PUT
DELETE

Response info

200 OK

Headers >

Response body

```
{
  "auth": {
    "tenantId": "bc24e59702947ddbad144bd5dfs78fe",
    "passwordCredentials": {
      "username": "747ca2d59204a90ee623fb6d42fa3",
      "password": "747ca2d59204a90ee623fb6d42fa3"
    }
  }
}
```

Request body:

```
{
  "auth": {
    "tenantId": "bc24e59702947ddbad144bd5dfs78fe",
    "passwordCredentials": {
      "username": "747ca2d59204a90ee623fb6d42fa3",
      "password": "747ca2d59204a90ee623fb6d42fa3"
    }
  }
}
```

RESTED

Hewlett Packard Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

7

Firefox RESTED plug-in

The RESTED client used in the examples exists as a plug-in for Firefox. To access it, click the appropriate **RESTED icon**, as shown on the slide above.

The initial RESTED screen consists of a number of fields, as shown and annotated in the above screens. You will use this RESTED UI to generate a request to an OpenStack® service and then view the response received from the OpenStack® service in the following pages.

Using cURL for the OpenStack® API requests

- cURL (Client URL Request Library) is a command line tool for transferring data with URL syntax
- Provides another method for submitting requests to the OpenStack® service APIs

Mapping content from the REST client browser UI to cURL statement

```
curl -k -X 'POST' -i -H "Content-Type: application/json"
-d '{"auth": {"identity": {"methods": ["password"], "password": {"user": {"name": "admin", "domain": { "id": "default" }, "password": "hpinvent" }}}}' http://localhost:5000/v3/auth/tokens ; echo
```

Hewlett Packard Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

Using cURL for the OpenStack® API requests

cURL is a command line tool for transferring data with URL syntax. cURL provides a very useful and quick method for submitting requests to OpenStack® service APIs because all the parameters in the request are on a single command line. However, it is very format-sensitive, and that is why you need to construct a cURL statement, verify its operation, and then save it, simply replacing the parameters in the cURL statement as needed.

As indicated in the screenshot above, the parameters required for the cURL request can actually be obtained from the REST client, making sure that the URL, username, password, and tenant ID values are relevant to the current request.

Note: The token security values obtained for the tenant ID are only valid for up to 24 hours.

cURL command options include:

- **-k**—This option explicitly allows cURL to perform "insecure" SSL connections and transfers. All SSL connections are attempted to be made secure by using the CA certificate bundle installed by default. This makes all connections considered "insecure" fail unless **-k** is used.
- **-v**—Verbose
- **-X**—Specifies a custom request method to use when communicating with the HTTP server. The specified request will be used instead of the method otherwise used (which defaults to **GET**).
- **-d**—Sends the specified data in a **POST** request to the HTTP server, in the same way that a browser does when you have filled in an HTML form and pressed the **Submit** button. This will cause cURL to pass the data to the server using the content-type **application/x-www-form-urlencoded**.
- **-H**—Extra header to use when getting a web page. You can specify any number of extra headers. If you add a custom header that has the same name as one of the internal ones curl would use, you externally set header instead of using the internal one.

Example cURL request and endpoint response

```
student@libDevStack:~/devstack$ curl -k -X 'POST' -i -H "Content-Type: application/json"  
-d '{"auth": {"identity": {"methods": ["password"], "password": {"user": {"name":  
"admin", "domain": { "id": "default" }, "password": "hpinvent" }}}}}'  
http://localhost:5000/v3/auth/tokens ; echo  
HTTP/1.1 201 Created  
Date: Wed, 30 Dec 2015 14:52:20 GMT  
Server: Apache/2.4.7 (Ubuntu)  
X-Subject-Token: d964bfff72694d3697c21d84f40f7682  
Vary: X-Auth-Token  
x-openstack-request-id: req-8d7b0453-689b-472a-870b-d27a9cf5c6ed  
Content-Length: 297  
Content-Type: application/json  
  
{"token": {"methods": ["password"], "expires_at": "2015-12-30T15:52:20.607060Z",  
"extras": {}, "user": {"domain": {"id": "default", "name": "Default"}, "id":  
"4e6c6473124e43d38a9c63186ee70fde", "name": "admin"}, "audit_ids": ["L8rD28m7Tf--  
qfcZMIIQYQ"], "issued_at": "2015-12-30T14:52:20.607091Z"}}
```

cURL request

OpenStack service's
response to cURL request

Example cURL request and endpoint response

cURL is a useful tool for transferring data to or from a server (an OpenStack® service endpoint), using supported protocol services such as the HTTP protocol used by the OpenStack® APIs. In the course labs, you will have the opportunity to use cURL to invoke REST API calls, as shown in the screenshot.

OpenStack® CLI client

- OpenStack® commands run from a terminal session
- All OpenStack® CLI commands are documented in the *OpenStack Command-Line Interface Reference Guide* at

<http://docs.openstack.org/cli-reference/content/>

Example of using help to show various parameters for the openstack CLI command

```
student@libDevStack:~/devStack$ openstack help
usage: openstack [--version] [-v] [-log-file LOG_FILE] [-q] [-h] [--debug]
                  [--os-cloud <cloud-config-name>]
                  [--os-region-name <auth-region-name>]
                  [--os-verify <ca-bundle-file>] [--verify | --insecure]
                  [--os-default-domain <auth-domain>]
                  [--os-interface <interface>] [--timing]
                  [--os-compute-api-version <compute-api-version>]
                  [--os-network-api-version <network-api-version>]
                  [--os-image-api-version <image-api-version>]
                  [--os-volume-api-version <volume-api-version>]
                  [--os-identity-api-version <identity-api-version>]
                  [--os-project-id <auth-project-id>]
                  [--os-project-domain-id <auth-project-domain-id>]
                  [--os-protocol <auth-protocol>]
                  [--os-project-name <auth-project-name>]
                  [--os-trust-id <auth-trust-id>]
                  [--os-service-provider-endpoint <auth-service-provider-endpoint>]
                  [--os-domain-name <auth-domain-name>]
                  [--os-user-domain-id <auth-user-domain-id>]
                  [--os-identity-provider-url <auth-identity-provider-url>]
                  [--os-access-token-endpoint <auth-access-token-endpoint>]
                  [--os-domain-id <auth-domain-id>]
                  [--os-user-domain-name <auth-user-domain-name>]
                  [--os-scope <auth-scope>] [-o <user-id <auth-user-id>]
                  [--os-identity-provider <auth-identity-provider>]
```

NOTE: After DevStack is initialized, enter the following command to set up the necessary environment variables before submitting any CLI commands:

\$. openrc <user name> <project name>.

Commands:
action definition create Create new action.
action definition delete Delete action.
action definition list List all actions.
action definition show Show specific action.
action definition show definition Show action definition.
action definition update Update action.
action execution delete Delete action execution.
action execution list List all Action executions.

OpenStack® CLI client

Another common method for accessing and using OpenStack® is through the OpenStack® CLI client. Commands are executed from a terminal session command line.

OpenStack® has an extensive help system that can be referenced by entering an OpenStack® service name followed by the word `help`. This command displays the syntax for the available commands and any associated arguments. The screenshots above display examples of the output of the `openstack help` command.

OpenStack® CLI command with --debug

**Hewlett Packard
Enterprise**

© Copyright 2017 Hewlett Packard Enterprise Development LP

11

OpenStack® CLI command with –debug

The `--debug` statement after an OpenStack® CLI command shows what is really happening to process the CLI command, as shown in the example above.

OpenStack® client

– Single command that manages the Compute, Identity, Image, Object Storage and Block Storage APIs

– Supports both v2 and v3 API

– Consistent and predictable format

```
openstack [<global-options>] <object-1> <action> [<object-2>] [<command-arguments>]
```

– Command examples:

- openstack user list
- openstack domain create new-domain
- openstack volume create --size 1 test-volume

– For more information, visit <http://docs.openstack.org/developer/python-openstackclient/>



© Copyright 2017 Hewlett Packard Enterprise Development LP

12

OpenStack® client

The OpenStack® client (OSC) brings the command set for the OpenStack services of Nova, Keystone, Glance, Swift, and Cinder APIs together in a single shell with a uniform command structure. It is the current command line client for the core OpenStack® services that supports both v2 and v3 API.

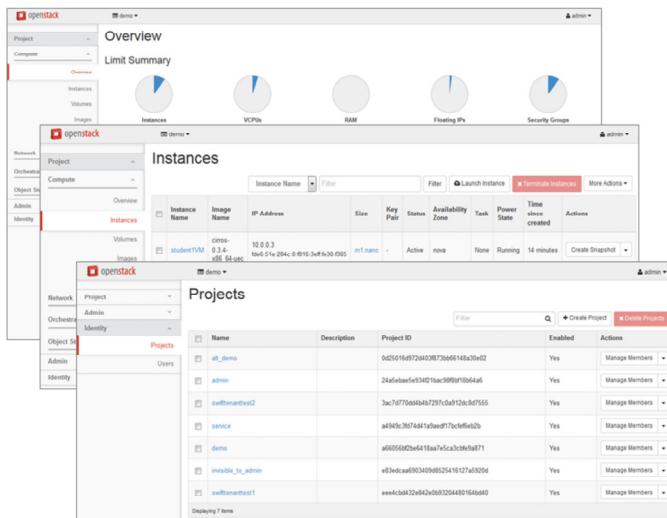
The OSC client follows the consistent and predictable format for all of its commands. A command takes the following form:

```
openstack [<global-options>] <object-1> <action> [<object-2>] [<command-arguments>]
```

Here are some examples of commands:

- openstack user list
- openstack domain create new-domain
- openstack volume create --size 1 test-volume

OpenStack® Horizon (dashboard) user interface



Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

13

– A web-based UI that provides the ability to manage OpenStack® services (for example, Nova, Keystone)

– Type the IP address of the device hosting the OpenStack® installation in a web browser to access it

– Accordion-style navigation of dashboards and panels

OpenStack® Horizon (dashboard) user interface

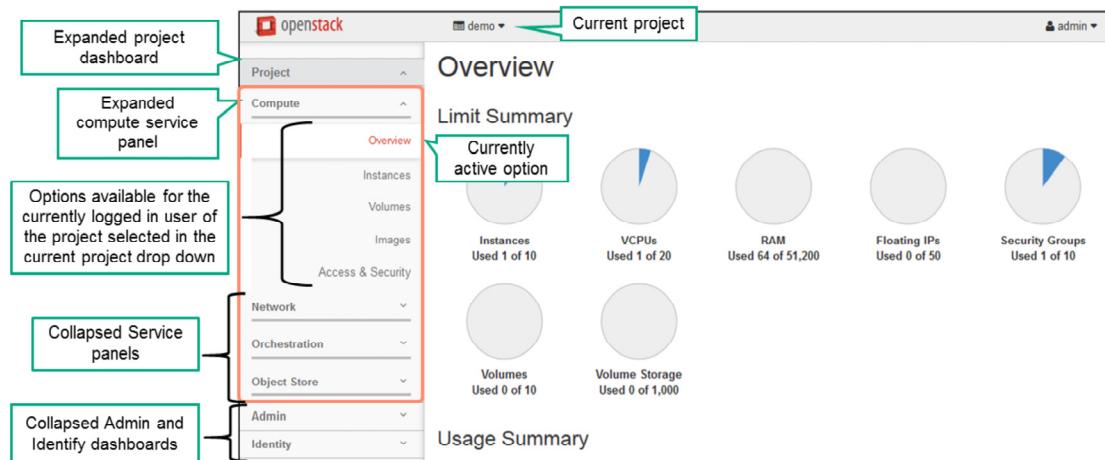
One of the easiest ways to work with OpenStack® is through Horizon—the web-based management GUI.

Horizon ships with three central dashboards, a “User Dashboard,” a “System Dashboard,” and a “Settings” dashboard. Together, they cover the core OpenStack® applications and deliver core support.

The Horizon application also ships with a set of API abstractions for the core OpenStack® projects to provide a consistent and stable set of reusable methods for developers. Owing to these abstractions, developers working in Horizon do not need to be intimately familiar with the APIs of each OpenStack® project.

Note: The labs associated with this module will provide ample opportunities for you to use the various methods for accessing OpenStack®.

OpenStack® Horizon user interface (1 of 2)



OpenStack® Horizon user interface (1 of 2)

The Horizon user interface is divided in three sections. Information is displayed in the main area for a selected project. The current project is visible on the top of the screen. The navigation bar at the left-hand side is divided in three main sections: Project, Admin, and Identity. Sections can be expanded to show the options for the currently logged in user or the project selected in the current project drop-down menu.

Note: Each section may contain several service panels, which can be expanded and reduced to display more options.

OpenStack® Horizon user interface (2 of 2)

Some workspace pages have tabs for accessing additional pages of info

Current project

Follow the link for more information

Edit Volume

Displays the user currently logged in, and when clicked provides access to user settings (e.g. language, time zone, and help)

Most table entries have one or more buttons for accessing a list of functions that can be performed on the associated item

Extend Volume
Launch as Instance
Manage Attachments
Create Snapshot
Change Volume Type
Upload to Image
Create Transfer
Delete Volume

Displaying 1 item

	Name	Description	Size	Status	Type	Attached To	Availability Zone	Bootable	Encrypted	Actions
<input type="checkbox"/>	student1VM_vola		10GB	Available	lvmdriver-1		nova	Yes	No	Edit Volume

Workspace for displaying information about the option selected in the navigation panel

Hewlett Packard Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

15

OpenStack Horizon user interface (2 of 2)

The main Horizon area in the center of the screen is where you interact with selected services. The workspace might have several tabs to address additional pages for complex services. The information is listed in tables. Most of the table entries have one or more buttons for accessing a list of functions that can be performed on the associated item. Values in the table appear as links, that can be followed to reach the pages providing the details about selected object.

Using Python to invoke an OpenStack® API

```
#!/usr/bin/env python
s = { "server": { "name": sname, "imageRef": sImageRef, "flavorRef": sFlavorRef, "metadata": sMetadata, "personality": sPersonality } }
sj = json.dumps(s)
params = sj
headers = { "X-Auth-Token":apitoken, "Content-type":"application/json" }
conn = httplib.HTTPConnection("localhost:8774")
conn.request("POST", "%s/servers" % apiurl[2], params, headers)
response = conn.getresponse()
data = response.read()
dd = json.loads(data4)
conn.close()
```

NOTE: The OpenStack® Python developer documentation is available at
<http://docs.openstack.org/developer/openstack-projects.html>.



© Copyright 2017 Hewlett Packard Enterprise Development LP

16

Using Python to invoke an OpenStack® API

Many users write automation scripts directly against the OpenStack® REST API or shell scripts that invoke the command line tools such as Keystone or Nova, but Python provides a better way to write OpenStack® automation scripts. All of the OpenStack® services have native Python APIs that expose the same feature set as the command line tools. Unfortunately, not much documentation is available to describe how to use these APIs.

If you are a Python programmer, the Python APIs are a simpler and more direct way to work with than the command line tools or the REST API.

Module summary

– In this module:

- We discussed the OpenStack® RESTful API (Application Programming Interface) access methods
- You were explained how to use:
 - A REST Client browser plugin to manage OpenStack® services
 - A cURL to manage OpenStack® services
 - The OpenStack® CLI to manage OpenStack® services
 - The Horizon (dashboard) UI to manage OpenStack® service
 - Python to manage OpenStack® services



© Copyright 2017 Hewlett Packard Enterprise Development LP

17

Module summary

In this module:

- We discussed the OpenStack® RESTful API (Application Programming Interface) access methods
- You were explained how to use:
 - A REST Client browser plugin to manage OpenStack® services
 - A cURL to manage OpenStack® services
 - The OpenStack® CLI to manage OpenStack® services
 - The Horizon (dashboard) UI to manage OpenStack® service
 - Python to manage OpenStack® services

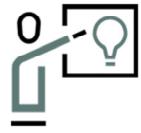
Learning check



© Copyright 2017 Hewlett Packard Enterprise Development LP

18

Learning check



After DevStack is initialized, what command should you enter to enable you (as admin) to use OpenStack® commands for the admin project?

- A. source openrc admin admin
- B. source localrc admin admin
- C. source adminrc admin admin
- D. source stackrc admin admin

Learning check

After DevStack is initialized, what command should you enter to enable you (as admin) to use OpenStack® commands for the admin project?

- A. source openrc admin admin
- B. source localrc admin admin
- C. source adminrc admin admin
- D. source stackrc admin admin

Learning check answer



After DevStack is initialized, what command should you enter to enable you (as admin) to use OpenStack® commands for the admin project?

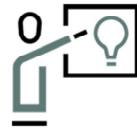
- A. `source openrc admin admin`
- B. `source localrc admin admin`
- C. `source adminrc admin admin`
- D. `source stackrc admin admin`

Learning check answer

After DevStack is initialized, what command should you enter to enable you (as admin) to use OpenStack® commands for the admin project?

- A. `source openrc admin admin`
- B. `source localrc admin admin`
- C. `source adminrc admin admin`
- D. `source stackrc admin admin`

Learning check



With what field in the REST client's form is the highlighted area in the cURL statement associated?

- A. URL
- B. Method
- C. Header
- D. Request body

```
curl -k -X 'POST' -i -H "Content-Type: application/json"
-d '{"auth": {"identity": {"methods": ["password"], "password": {"user": {"name": "admin", "domain": { "id": "default" }, "password": "hpinvent" }}}}}'
http://localhost:5000/v3/auth/tokens ; echo
```

Learning check

With what field in the REST client's form is the highlighted area in the cURL statement associated?

- A. URL
- B. Method
- C. Header
- D. Request body

Learning check answer



With what field in the REST client's form is the highlighted area in the cURL statement associated?

- A. URL
- B. Method
- C. Header
- D. Request body**

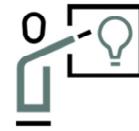
```
curl -k -X 'POST' -i -H "Content-Type: application/json"
-d '{"auth": {"identity": {"methods": ["password"], "password": {"user": {"name": "admin", "domain": { "id": "default" },
"password": "hpinvent" }}}}}'
http://localhost:5000/v3/auth/tokens ; echo
```

Learning check answer

With what field in the REST client's form is the highlighted area in the cURL statement associated?

- A. URL
- B. Method
- C. Header
- D. Request body**

Learning check



Which of the highlighted drop-down fields in the Horizon UI is used to select the project to be managed?

- A. A
- B. B
- C. C
- D. D

The screenshot shows the OpenStack Horizon interface. On the left, there's a sidebar with a 'Project' dropdown (labeled A) containing 'Compute', 'Instances', 'Volumes' (which is selected), 'Images', 'Access & Security', 'Network', 'Orchestration', 'Object Store', and 'Admin'. Below that is an 'Identity' dropdown (labeled D). At the top right, there's a user dropdown (labeled B) with 'demo' and 'admin' options, and an admin dropdown (labeled C) with 'admin'. The main content area is titled 'Volumes' and shows a table with one item: 'student1VM_volA' (Name), 10GB (Size), Available (Status), lvmdriver-1 (Type), nova (Attached To), nova (Availability Zone), Yes (Bootable), No (Encrypted). An 'Actions' column has a dropdown menu open with options: Extend Volume, Launch as Instance, Manage Attachments, Create Snapshot, Change Volume Type, Upload to Image, Create Transfer, and Delete Volume.

**Hewlett Packard
Enterprise**

© Copyright 2017 Hewlett Packard Enterprise Development LP

23

Learning check

Which of the highlighted drop-down fields in the Horizon UI is used to select the project to be managed?

- A. A
- B. B
- C. C
- D. D

Learning check answer



Which of the highlighted drop-down fields in the Horizon UI is used to select the project to be managed?

- A. A
- B. B
- C. C
- D. D

The screenshot shows the OpenStack Horizon interface. On the left, there's a sidebar with various service links like Compute, Instances, Volumes, Images, Network, Orchestration, Object Store, Admin, and Identity. The 'Volumes' link is selected. The main content area is titled 'Volumes' and shows a table with one item: 'student1VM_volA'. A context menu is open over this item, listing options such as Extend Volume, Launch as Instance, Manage Attachments, Create Snapshot, Change Volume Type, Upload to Image, Create Transfer, and Delete Volume.

**Hewlett Packard
Enterprise**

© Copyright 2017 Hewlett Packard Enterprise Development LP

24

Learning check answer

Which of the highlighted drop-down fields in the Horizon UI is used to select the project to be managed?

- A. A
- B. B
- C. C
- D. D



**Hewlett Packard
Enterprise**



Fundamentals of OpenStack® Technology

Module 4—Keystone Identity Service

H6C68S E.01



© Copyright 2017 Hewlett Packard Enterprise Development LP

Learning objectives

After completing this module, you should be able to:

- Describe the purpose of Keystone
- Describe the high-level architecture of Keystone
- Configure Keystone
- Complete common Keystone management tasks
- Troubleshoot some of the common Keystone-related issues



© Copyright 2017 Hewlett Packard Enterprise Development LP

2

Learning objectives

After completing this module, you should be able to:

- Describe the purpose of Keystone
- Describe the high-level architecture of Keystone
- Configure Keystone
- Complete common Keystone management tasks
- Troubleshoot some of the common Keystone-related issues



OpenStack® Keystone (identity) overview

Keystone identity and service management

Identity management

- Tracks users and their permissions
- Provides authentication and authorization for all the OpenStack® services
- Defines projects, users, and roles
- Manages domains and groups

```
$ openstack user list
+-----+-----+-----+-----+
| id | name | enabled | email |
+-----+-----+-----+-----+
| ab41e1204ba342060fda0fd6d4904da9 | admin | True | admin@example.com |
| 87b5c1674d05453fb1bd3e560d592ca1 | demo | True | demo@example.com |
| fbdce77b686a4df09b3d3cc32266580a | glance | True | |
...
...
```

Service catalog

- Catalog of available services with their API endpoints
- Maintains a user that corresponds to each service (such as Nova for the Compute service)
- Special service project, which is called “service”

`$ openstack catalog list`

Name	Type	Endpoints
nova	compute	RegionOne publicURL: http://192.168.5.5:8774/v2/1/bc34539702947ddbad1446d5dfa76fe internalURL: http://192.168.5.5:8774/v2/1/bc34539702947ddbad1446d5dfa76fe adminURL: http://192.168.5.5:8774/v2/1/bc34539702947ddbad1446d5dfa76fe
neutron	network	RegionOne publicURL: http://192.168.5.5:9696/ internalURL: http://192.168.5.5:9696/ adminURL: http://192.168.5.5:9696/
cinderv2	volumev2	RegionOne publicURL: http://192.168.5.5:8776/v2/1/bc34539702947ddbad1446d5dfa76fe internalURL: http://192.168.5.5:8776/v2/1/bc34539702947ddbad1446d5dfa76fe adminURL: http://192.168.5.5:8776/v2/1/bc34539702947ddbad1446d5dfa76fe
aodh	alarming	RegionOne publicURL: http://192.168.5.5:8842/ internalURL: http://192.168.5.5:8842/ adminURL: http://192.168.5.5:8842
glance	image	RegionOne publicURL: http://192.168.5.5:9292/ internalURL: http://192.168.5.5:9292



© Copyright 2017 Hewlett Packard Enterprise Development LP

4

Keystone identity and service management

The functions of Keystone can be broken down into two primary responsibilities, that of identity management and service catalog management.

Identity management:

- Tracks users and their permissions
- Provides authentication and authorization for all of the OpenStack® services
- Supports the creation and maintenance of OpenStack® users, projects, and roles

Keystone:

- Can serve as the common authentication system across the cloud
- Can be integrated with existing backend directory services like LDAP
- Supports multiple forms of authentication
- Supports standard username and password credentials and token-based authentication

In service catalog management, Keystone:

- Provides catalog of available services with the API endpoints
- Maintains a user that corresponds to each service
- Supports the use of a special service project called “service”

Keystone API version states as of the Newton release

- **Identity API v3 (current)**
- **Identity API v3 extensions (current)**
- Identity API v2 (deprecated)
- Identity admin API v2.0 (deprecated)
- Identity API v2.0 extensions (deprecated)



© Copyright 2017 Hewlett Packard Enterprise Development LP

5

Keystone API version states as of the Newton release

OpenStack® Newton release currently provides the v3.8 of Identity API and API extensions. Version 2 API is deprecated. For compatibility reasons, Newton provides the endpoints for old API to be handled through the new API engine. For more information, run `openstack endpoint list --long` in your lab environment.

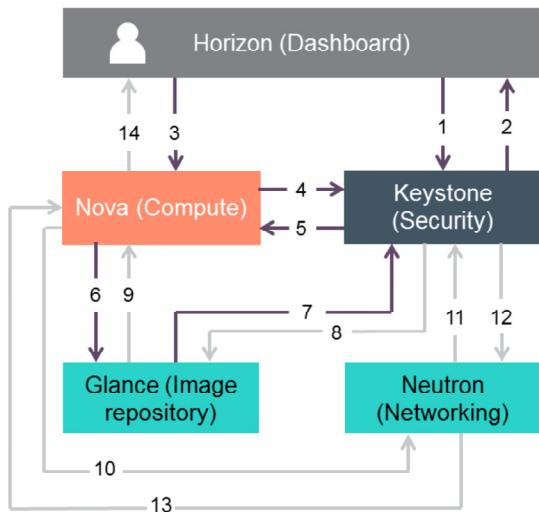
Here are some of the changes included in the Newton release:

- Deprecation of PKI and PKIZ token formats
- Caching of the tokens when issued
- Upgrading Keystone to a new version can now be undertaken as a rolling upgrade

Note: Tokens will be explained later in this module.

Keystone identity (authentication/authorization) process (1 of 2)

Example: Provisioning a VM



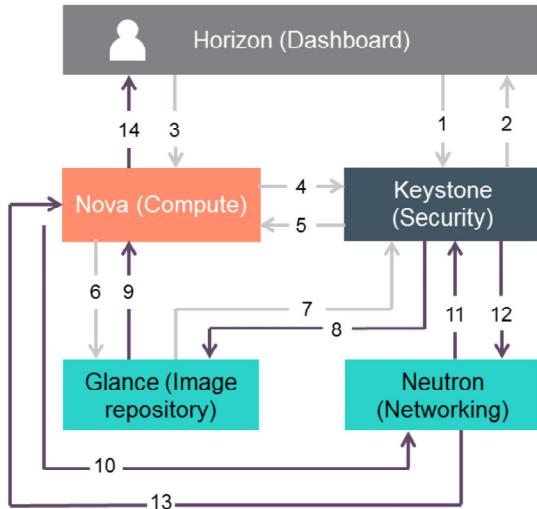
1. Horizon sends an HTTP authentication request to Keystone with user credentials
2. Keystone validates credentials and replies with a temporary token
3. Horizon sends a POST request with the token to Nova to start provisioning a VM
4. Nova sends the token to Keystone for validation
5. Keystone validates the token
6. Nova forwards a request for an image with the attached token
7. Glance sends the token to Keystone for validation

Keystone identity (authentication/authorization) process (1 of 2)

The above flowchart demonstrates how Keystone is used in the process of provisioning a VM.

Keystone identity (authentication/authorization) process (2 of 2)

Example: Provisioning a VM



- Keystone validates the token
- Glance provides image-related information to Nova
- Nova sends a request for networks to Neutron with the token
- Neutron sends the token to Keystone for validation
- Keystone validates the token
- Neutron provides network-related information to Nova
- Nova reports the status of the VM provisioning request

Keystone identity (authentication/authorization) process (2 of 2)

The above flowchart demonstrates how keystone is used in the process of provisioning a VM.

Keystone Identity-related objects (1 of 2)

– **A user:**

- Is someone or something that can gain access through Keystone
- Comes with credentials that can be checked (like passwords or API keys)

– **A group** is a collection of users

- Roles assigned to the group are automatically associated with all users of that group
- Applies to Keystone API 3.0 and more recent API keys

– **A project:**

- Is a collection of resources and quotas associated with users by roles
- Was formerly known as tenant and is still referred to as such in older APIs and command line tools
- Can have some machines in Nova, a number of images in Swift/Glance, a few networks in Neutron ,and a floating IP quota

– **A role** represents the number of privileges or rights a user has or actions they are allowed to perform

– Users can be added to any role:

- Within a specific project, where a user's access is limited to resources of the corresponding project
- Inter-project, where the user gains access implied by the role to the resources in multiple projects
- Example: A user who has an admin role can take admin actions (like view all projects)



© Copyright 2017 Hewlett Packard Enterprise Development LP

8

Keystone identity-related objects (1 of 2)

There are six types of Keystone objects that need to be defined at this point.

A user is someone or something that can gain access through Keystone. Users come with credentials that can be checked like passwords or API keys.

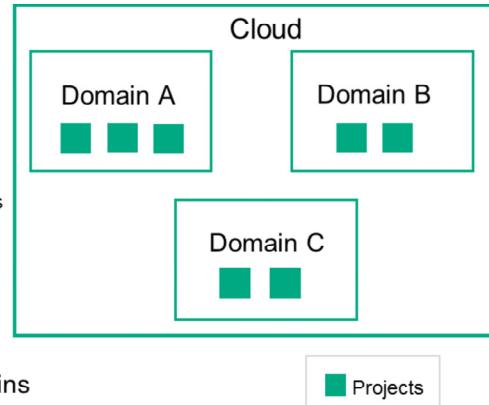
A group is a collection of users. Roles assigned to the group are automatically associated with all users of that group. This applies to Keystone API 3.0 and more recent API keys.

A project represents what is called the tenant in Nova originally, meaning something that aggregates the number of resources in each service. For example, a project can have some machines in Nova, a number of images in Swift or Glance, and a few networks in Neutron. Users are bound to a project when they are assigned a role on that project.

A role represents the number of privileges or rights a user has or actions they are allowed to perform. For example, users who have an admin role can take admin actions (for example, they can view all projects). Users can be added to any role either globally or in a project. In the first case, users gain access implied by the role to the resources in all projects; in the second case, the user access is limited to resources of the corresponding project. For example, users can be operators of all projects and admins of their own projects.

Keystone Identity-related objects (2 of 2)

- A domain:
 - Defines administrative boundaries for the management of Keystone entities
 - May represent an individual, a company, or operator-owned space
 - Allows a cloud to be organized into multiple independently managed units
 - A Cloud Provider supplies virtual private clouds to cloud customers as domains
 - A User may be given a domain administrator role, allowing them to create projects, users, and groups within a domain
 - Each domain may have its own identity backend (LDAP, and similar)
 - A “default” domain allows installations to effectively ignore domains and is applied when no domain is specified



Keystone identity-related objects (2 of 2)

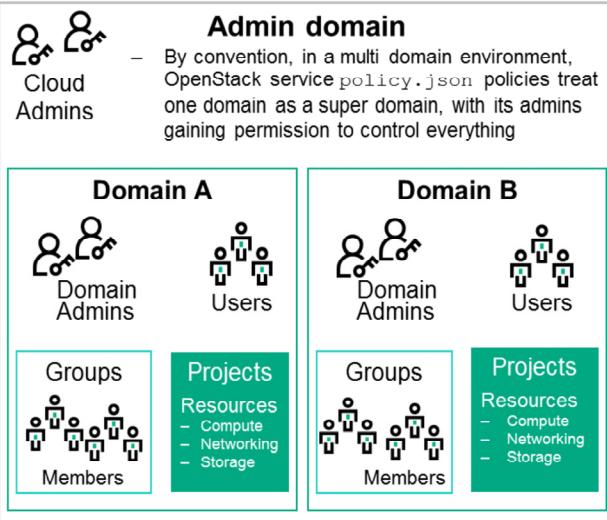
A domain defines administrative boundaries for the management of Keystone entities. Domains are transparent to most OpenStack® services, which only require token authentication and role information (not knowledge of the authentication/authorization structure). Services ask, “what roles do they have?”, not “how did they get them?”. Domains are supported by the v3 Identity API, but have an impact on the v2 API. The Keystone Identity API v3 is a superset of v2.0 and provides a much more consistent developer experience. API v2 may be deprecated in the near future.

Keystone has supported domains for the last few releases; however, it has been transparent to API v2 users. All projects created and accessed via the v2 API in recent Keystone implementations have been placed in the “default” domain. This causes Keystone to rationalize legacy use as a single domain cloud. “Non-default” domains created with Keystone API v3 will not be visible to systems using v2. This works in most cases because most services do not make domain-specific requests.

A region is a discrete OpenStack® environment with dedicated API endpoints that typically shares only Identity (Keystone) with other regions. These will be discussed in the Nova module.

Keystone Identity-related object relationships

- All projects, users, or groups are contained within a single domain
- Only users from a group's domain can be added to the group
- Users are given roles to associate with projects
 - Users are not contained within projects, but can be given roles on projects in another domain
 - Roles can be assigned to a user without a target project
- Most OpenStack services know nothing about domains
 - Services only care about tokens and roles
 - Horizon is the exception, it provides an auth interface
- A "Default" Domain allows installations to effectively ignore Domains and is applied when no domain is specified



Keystone Identity-related object relationships

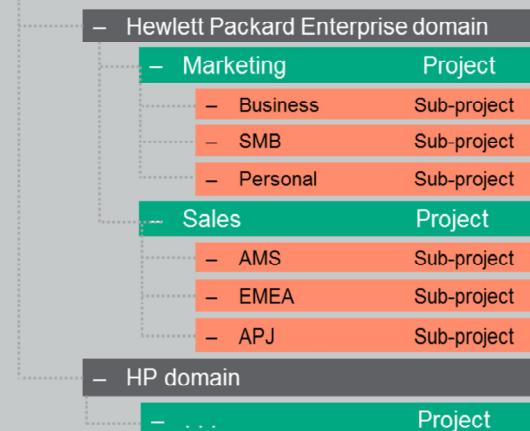
OpenStack® provides no particular support for Cloud Admins. A Cloud Admin is a super administrator capable of administering all domains. To create a Cloud Admin, you need to create a domain which acts as a “super domain,” and specify the domain ID as the `admin_domain_id` value in `policy.json` “cloud_admin” rules. The “default” domain is typically configured in DevStack as the “super admin domain”. Admins in the super domain are then, by policy, given full control of everything by all OpenStack® services.

Domains serve as containers for collections of projects that could be administered as a group. When discussing authorization, domains can simply be thought of as groups of projects. Domains allow you to model real-world use cases such as a single user having multiple environments, where each environment is isolated from the others using the existing OpenStack® (flat) model of multi-tenancy.

For more information on domains, refer to <https://www.mirantis.com/blog/manage-openstack-projects-using-domains-havana/>.

Hierachial multitenancy

Cloud Provider



- Hierarchical multitenancy, introduced in Kilo:
 - Projects can be nested under other projects by setting the `parent_id` attribute to an existing project when creating a new project
 - Parent-child hierarchy can be discovered through the Keystone v3 API
 - Role assignments can now be assigned to both users and groups on sub-trees in the project hierarchy

Hierarchical multitenancy

Hierarchical multitenancy is introduced in Kilo release of OpenStack®. Hierarchical multitenancy allows nesting projects. For example, administrators can use this feature to organize departmental divisions to work as “sub-projects”.

Hierarchical multitenancy requires the Keystone v3 API. Projects can be nested under other projects by setting the `parent_id` attribute to an existing project when creating a new project.

Hierarchical multitenancy enables a company to be represented through departmental divisions with authorizations applicable to a higher level in the hierarchy trickling down to the layers beneath it.

Additional Keystone terminology

Identity management	Service catalog
Role-based Access Control (RBAC): <ul style="list-style-type: none"> Customized API access by role 	Endpoint: <ul style="list-style-type: none"> A network-accessible address, usually described by URL, where a service may be accessed
User credentials: <ul style="list-style-type: none"> Data owned by a user to prove identity 	Service: <ul style="list-style-type: none"> An OpenStack® project, such as Compute (Nova), Object Storage (Swift), or Image Service (Glance)
Token: <ul style="list-style-type: none"> Arbitrary bit of text used to access resources Scope which describes the resources are accessible Valid for limited duration and revocable 	Service catalog: <ul style="list-style-type: none"> Endpoint registry used for service endpoint discovery
policy.json: <ul style="list-style-type: none"> Is the file located in each service's /etc/<service name> directory Sets role-based policies for using the service-specific functions 	

NOTE: OpenStack® Glossary is available at <http://docs.openstack.org/glossary/content/glossary.html>.



© Copyright 2017 Hewlett Packard Enterprise Development LP

12

Additional Keystone terminology

Role-based Access Control (RBAC) provides a predefined list of actions that the user can perform, such as start or stop VMs, reset passwords, and so on. RBAC is supported in both Identity and Compute and can be configured using the dashboard.

A **token** is an alpha-numeric string of text used to access the OpenStack® APIs and resources. Tokens can be scoped in order to describe the resources that are accessible. They are valid for limited duration, and can be revoked.

An **endpoint** is a network-accessible address, described by the URL, where a service might be accessed. Service catalog is the endpoint registry used for service endpoint discovery.

policy.json file is a configuration file located in the /etc/keystone/ folder that is used to set role-based policies for using the service-specific functions.

Keystone support for Federated Identity (1 of 2)

Federated Identity terms

- **Service Provider (SP)**—A system entity that provides services to principals or other system entities, in this case, OpenStack Identity is the Service Provider.
- **Identity Provider (IdP)** —A directory service (e.g. LDAP, Active Directory) which allows users to log in with a username and password
- **Mapping**—A set of rules used to map Federation protocol attributes to Identity API objects
- **Protocol**—Contains information that dictates which mapping rules to use for an incoming request made by an IdP
 - There are two protocols supported by OpenStack® Federated Identity:
 - **SAML (Security Assertion Markup Language)** provides IdP-supplied information about a user that indicates that a user has been authenticated
 - **OID (OpenID)** allows users to be authenticated by co-operating sites using a third party service



© Copyright 2017 Hewlett Packard Enterprise Development LP

13

Keystone support for Federated Identity (1 of 2)

Federated Identity is a mechanism used to establish trusts between Identity Providers and Service Providers (SP), in this case, between Identity Providers and the services provided by an OpenStack® Cloud.

Some important definitions include:

- **Service Provider (SP)**—A system entity that provides services to principals or other system entities; in this case, OpenStack® Identity is the Service Provider.
- **Identity Provider (IdP)** —A directory service (e.g. LDAP, Active Directory) which allows users to log in with a username and password.
- **Mapping**—A set of rules used to map Federation protocol attributes to Identity API objects.
- **Protocol**—Contains information that dictates which mapping rules to use for an incoming request made by an IdP. There are two protocols supported by OpenStack® Federated Identity:
 - SAML (Security Assertion Markup Language) provides IdP-supplied information about a user that indicates that a user has been authenticated.
 - OID (OpenID) allows users to be authenticated by co-operating sites using a third-party service.

Keystone support for Federated Identity (2 of 2)

Definition of Federated Identity

- Provides a way to securely use existing credentials to access cloud resources such as servers, volumes, and databases, across multiple endpoints provided in multiple authorized clouds using a single set of credentials
- Eliminates the need to provision additional identities or log in multiple times
- Removes a block to cloud brokering and multi-cloud workload management because there is no need to build additional authentication mechanisms to authenticate a user
- For more information, visit
http://docs.openstack.org/security-guide/content/section_identity-federated-keystone.html#



© Copyright 2017 Hewlett Packard Enterprise Development LP

14

Keystone support for Federated Identity (2 of 2)

Federated Identity Provider offers services that enable users in a corporate enterprise environment to use a single digital identity to access applications and services that they have access rights to, regardless of which security domain the application or service resides in.

Federated Identity provides a way to securely use existing credentials to access cloud resources such as servers, volumes, and databases, across multiple endpoints provided in multiple authorized clouds using a single set of credentials, without having to provision additional identities or log in multiple times. The credentials are maintained by the user's Identity Provider.

Federated Identity Eliminates the need to provision additional identities or log in multiple times.

Removes a block to cloud brokering and multi-cloud workload management because there is no need to build additional authentication mechanisms to authenticate user

Keystone-specific reference resources

- Keystone documentation
 - Keystone overview: <http://docs.openstack.org/developer/keystone/>
 - Keystone security: <http://docs.openstack.org/security-guide/identity.html>
 - Keystone concepts: http://docs.openstack.org/admin-guide-cloud/identity_management.html



© Copyright 2017 Hewlett Packard Enterprise Development LP

15

Keystone-specific reference resources

The following links contain the relevant information resources:

- Keystone overview: <http://docs.openstack.org/developer/keystone/>
- Keystone security: <http://docs.openstack.org/security-guide/identity.html>
- Keystone concepts: http://docs.openstack.org/admin-guide-cloud/identity_management.html

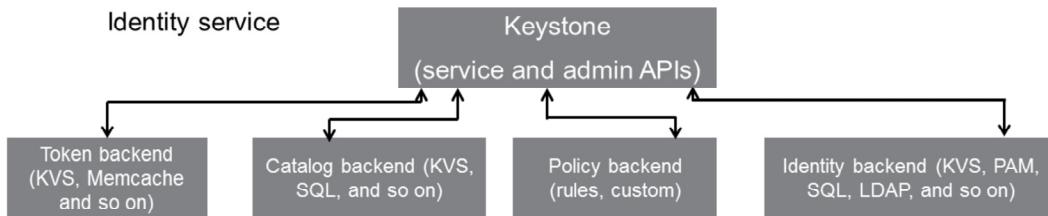
For more project information, refer to <https://wiki.openstack.org/wiki/Keystone>.



Identity service architecture

Internal components of the Identity service

- Keystone is organized as a group of internal services exposed to one or many endpoints
 - **Identity**: Provides authentication credential validation and data about users, projects, and roles as well as any associated metadata
 - **Token**: Validates and manages tokens used for authenticating requests after the user or project credentials have been verified
 - **Catalog**: Provides an endpoint registry used for the endpoint discovery
 - **Policy**: Provides a rule-based affinity authorization engine



Internal components of the Identity service

Keystone is organized as a group of internal services exposed to one or many endpoints. The primary components of Keystone, aside from the Keystone process itself, are the databases: the token backend, the catalog backend, the policy backend, and the identity backend.

Keystone backend services

- Each of the Keystone components (such as Identity or Token) can be configured to use a backend to allow Keystone to fit a variety of environments and needs
 - **KVS backend:** Meant to be further back-ended on anything that can support primary key lookups
 - **SQL backend:** Uses SQL Alchemy to store data persistently
 - **PAM backend:** Uses the current systems PAM service to authenticate, providing a one-to-one relationship between users and projects
 - **LDAP backend:** Stores users and projects in separate sub-trees
 - **Templated backend:** A simple template used to configure Keystone

NOTE: The backend for each service is defined in the Keystone configuration file located at `/etc/keystone/keystone.conf`.



© Copyright 2017 Hewlett Packard Enterprise Development LP

18

Keystone backend services

Each of the Keystone components (such as the Identity or Token) can be configured to use a backend to allow Keystone to fit a variety of environments and needs. The backend for each service is defined in the Keystone configuration file, located at `/etc/keystone/keystone.conf`.

The types of backends supported by Keystone are:

- **KVS backend:** Meant to be further back-ended on anything that can support primary key lookups
- **SQL backend:** Uses SQL Alchemy to store data persistently
- **PAM backend:** Uses the current systems PAM service to authenticate, providing a one-to-one relationship between users and projects
- **LDAP backend:** Stores users and projects in separate sub-trees
- **Templated backend:** A simple template used to configure Keystone



Common Keystone management tasks

Managing projects with the CLI client

```
student@libDevStack:~/devstack$ openstack project list
+-----+-----+
| ID      | Name   |
+-----+-----+
| 0d25016d972d403f873bb66148a30e02 | alt_demo
| 24a5ebae5e934f21bac98f8bf18b64a6 | admin
| 3ac7d770dd4b4b7297c0a912dc8d7555 | swifttenanttest2
| a4949c3fd74d41a9aedf17bcfef6eb2b | service
| a66056bf2be6418aa7e5ca3cbfe9a871 | demo
| e83edcaa6903409d3525416127a5920d | invisible_to_admin
| eee4cbd432e842e0092304499164bd40 | swifttenanttest1
+-----+-----+
```

– Common project-related CLI commands:

- openstack project create
- openstack project delete
- openstack project set
- openstack project show

```
student@libDevStack:~/devstack$ openstack project show 24a5ebae5e934f21bac98f8bf18b64a6
+-----+-----+
| Field    | Value  |
+-----+-----+
| description |          |
| enabled     | True   |
| id          | 24a5ebae5e934f21bac98f8bf18b64a6 |
| name        | admin   |
+-----+-----+
```

The ID or the name can be used when referring to an entity, but the ID is unique and the name is not

 Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

20

Managing projects with the CLI client

Several CLI commands are available for managing projects, a couple of which are shown in the screenshots.

The screenshots show a list of project IDs that you can obtain by first entering the `openstack project list` command, and then the `openstack project show` command with one of those IDs to display information about those specific projects.

The ID or the name can be used when referring to an entity, but the ID is unique and the name is not.

Managing users with the CLI client

ID	Name	Project	Email	Enabled
15579bfd93544858b106fd04e91ad19	cinder			True
20085f161e0b476289ee602807cd917	heat			True
3c30b2633cf01a937ca203ec224b7ea602	glance-swift		glance-swift@example.com	True
4e6c6473124e43d38a9c63186ee70fde	admin	swift	alt_demo@example.com	True
5305320a02341f9e7fb0404e5ac0e00			test3@example.com	True
6468d2816d02414fffdcc04b9183b50		swiftusertest3	test2@example.com	True
70828c8c50434ae03ad2ffcc14102a6cb		swiftusertest2		True
8fd7ac2b80034ab993527f74afbb18584	neutron			True
b252e01586f6494b0845f9811157049c	glance			True
b1f65032e012400cb2954d15efdb2f		swiftusertest1	teste@example.com	True
c647ff4fad9944329cb049020f0c400f				True
e4bf95b457ef470bbfc67d061a9f92f	nova			True
fd6331d7560948e2a5e9947ac9c0a531	demo		demo@example.com	True

Field	Value
enabled	True
id	4e6c6473124e43d38a9c63186ee70fde
name	admin
username	admin

The ID or the name can be used when referring to an entity, but the ID is unique and the name is not

- A user:

- Must belong to at least one project
- Can be restricted to performing specific actions

- Common user-related CLI commands:

- openstack user create
- openstack user delete
- openstack user list
- openstack user role list
- openstack user set
- openstack user show

Managing users with the CLI client

A number of CLI commands are available for managing users, a couple of which are shown in the screenshots.

When listing available users with the `openstack user list` command, notice that the user has five parameters associated with it:

- The ID that is automatically assigned to it when it is created
- The name of the user
- The project the user is associated to
- The user's email address
- The information on whether the user is currently enabled

The screenshots display a list of user IDs obtained through the `openstack user list` command, and the information about a specific user when the `openstack user show` command is used.

The ID or the name can be used when referring to an entity, but the ID is unique and the name is not.

Creating a role, project, user, and adding project user to role

```

1 student@libDevStack:~/devstack$ openstack role create testrole
+-----+
| Field | Value
+-----+
| id   | 5e8e4962004443ac8310cf737cce2a86
| name | testrole
+-----+
2 student@libDevStack:~/devstack$ openstack user create testuser
+-----+
| Field | Value
+-----+
| email | None
| enabled | True
| id   | 6d472da2bd6644c8a1201e63b09a3aa5
| name | testuser
| username | testuser
+-----+
3 student@libDevStack:~/devstack$ openstack project create testproject
+-----+
| Field | Value
+-----+
| description | None
| enabled | True
| id   | 9563ecfe4bcd492982a332414f57ecd6
| name | testproject
+-----+
4 student@libDevStack:~/devstack$ openstack role add --project 9563ecfe4bcd492982a332414f57ecd6
--user 6d472da2bd6644c8a1201e63b09a3aa5 5e8e4962004443ac8310cf737cce2a86
+-----+
| Field | Value
+-----+
| id   | 5e8e4962004443ac8310cf737cce2a86
| name | testrole
+-----+

```

Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

22

- A role:

- Represents a number of privileges or rights a user has or actions they are allowed to perform
- UserA for projectX might be able to complete the same OpenStack® actions as UserA for projectZ

- Common user-related CLI commands:

- openstack role add
- openstack role create
- openstack role delete
- openstack role list
- openstack role remove
- openstack role show

Creating a role, project, user, and adding project user to role

This slide illustrates the procedure for using the CLI to create a role, a project , and a user, as well as for adding a role to a project or a user

Managing the service catalog with the CLI client

ID	Name	Type	Description
6c1abc67e884722af4b479d3412e738	cinderv2	volumev2	Volume Service V2
171b0c2f4ad99aaad151907c028760	glance	image	
251d0c2f4ad99aaad151907c028760	cinder	volume	
202d3ec4b90a40fa974b6aef0fc2d2	s3	s3	
56f8bd7c8d44feee839ae15205e3ee2b	neutron	network	
59146747cf241108095de675a294291	heat	orchestration	
7b9c1a58b5d340e18f9fb1c2c06b1262	keystone	identity	Keystone Identity Service
807d4cebaefc436ea5fs2f5482c5f748	swift	object-store	Swift Service
859f1053fb9a4cc2e905f1f1937bcafe	pova	compute	Nova Compute Service
a7296d3f6a604e0182e0a0c08ead85e5	heat-cfn	cloudformation	Heat CloudFormation Service
dd40aa0df1efc4eb981b5440bc9fdb4d7	nova_legacy	compute_legacy	Nova Compute Service (Legacy 2.0)

Notice the `--long` parameter is used with each of the CLI commands on this page to display additional information

Common service-related CLI commands:

- `openstack service create`
- `openstack service delete`
- `openstack service list`
- `openstack service show`
- `openstack endpoint create`
- `openstack endpoint delete`
- `openstack endpoint list`
- `openstack service show`

Public URL of each OpenStack® service					Admin URL of each OpenStack® service		Internal URL of each OpenStack® service	
ID	Region	Service Name	Service Type	PublicURL	AdminURL	InternalURL	InternalURL	
f617a37bf3ca4bb00a08a37c783d7548	RegionOne	cinder	volume	http://192.168.0.13:8776/v1/\$tenant_ids	http://192.168.0.13:8776/v1/\$tenant_ids	http://192.168.0.13:8776/v1/\$tenant_ids	http://192.168.0.13:8776/v1/\$tenant_ids	
d0c100054db3199937a7640628	RegionOne	glance	image	http://192.168.0.13:9292/v1/\$tenant_ids	http://192.168.0.13:9292/v1/\$tenant_ids	http://192.168.0.13:9292/v1/\$tenant_ids	http://192.168.0.13:9292/v1/\$tenant_ids	
404bdbee3713443e2a3dc1677a30d628	RegionOne	cinderv2	volumev2	http://192.168.0.13:8776/v2/\$tenant_ids	http://192.168.0.13:8776/v2/\$tenant_ids	http://192.168.0.13:8776/v2/\$tenant_ids	http://192.168.0.13:8776/v2/\$tenant_ids	
8c-f6eb5457d42499ed9a37678e4d6	RegionOne	s3	s3	http://192.168.0.13:9696/v1	http://192.168.0.13:9696/v1	http://192.168.0.13:9696/v1	http://192.168.0.13:9696/v1	
ee8eb0192c374aedbb28ca6c230c90	RegionOne	neutron	network	http://192.168.0.13:9996/v1	http://192.168.0.13:9996/v1	http://192.168.0.13:9996/v1	http://192.168.0.13:9996/v1	
a5bf6ac28ff24eb69378a1047efc680	RegionOne	heat-cfn	cloudformation	http://192.168.0.13:8000/v1	http://192.168.0.13:8000/v1	http://192.168.0.13:8000/v1	http://192.168.0.13:8000/v1	
ec73a00954db44c6a272aa842fcff8c4	RegionOne	s3	s3	http://192.168.0.13:3333	http://192.168.0.13:3333	http://192.168.0.13:3333	http://192.168.0.13:3333	
c079eae31aa1435ea0207467b3969913	RegionOne	heat	orchestration	http://192.168.0.13:8004/v1/\$tenant_ids	http://192.168.0.13:8004/v1/\$tenant_ids	http://192.168.0.13:8004/v1/\$tenant_ids	http://192.168.0.13:8004/v1/\$tenant_ids	
c520784ad484981c53597fc9b987a7	RegionOne	keystone	identity	http://192.168.0.13:5000/v2.0	http://192.168.0.13:5000/v2.0	http://192.168.0.13:5000/v2.0	http://192.168.0.13:5000/v2.0	
7f0784ad484981c53597fc9b987a7	RegionOne	nova_legacy	compute_legacy	http://192.168.0.13:8000/v2/\$tenant_ids	http://192.168.0.13:8000/v2/\$tenant_ids	http://192.168.0.13:8000/v2/\$tenant_ids	http://192.168.0.13:8000/v2/\$tenant_ids	
4fc1870db4709b987413739eaf5b9	RegionOne	swift	object-store	http://192.168.0.13:8000/v1/AUTH_5\$tenant_ids	http://192.168.0.13:8000/v1/AUTH_5\$tenant_ids	http://192.168.0.13:8000/v1/AUTH_5\$tenant_ids	http://192.168.0.13:8000/v1/AUTH_5\$tenant_ids	
5d9dc25b2ec44ab1970e6dd08c5d0	RegionOne	ec2	ec2	http://192.168.0.13:8773	http://192.168.0.13:8773	http://192.168.0.13:8773	http://192.168.0.13:8773	

Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

23

Managing the service catalog with the CLI client

Any OpenStack® service (such as Nova, Glance, and so on) can use a project, a user, or a role to check whether the user has access to resources. But to access some services in the project, you have to know that the service exists and then find a way to access it.

To access a service, you have to know its endpoint. Keystone endpoint templates provide information about all existing endpoints of all existing services. One endpoint template provides a list of URLs to access an instance of a service:

- `publicurl` is intended to be accessible from the global world (like <http://compute.example.com>).
- `internalurl` can be used to access a service from a local network (like <http://compute.example.local>).
- `adminurl` is used in case admin access to a service is separated from the common access (like it is in Keystone).

Several service-related CLI commands are listed in the slide. Notice that this list includes endpoint commands.

The bottom screenshot shows the output of the `openstack endpoint list` command, which lists the available endpoints. You can locate the service endpoint URL of a specific service by locating the service ID in the service ID field in the table. In this example, the Keystone service URL is highlighted. This URL will be useful when using the REST client later in this module.

Keystone Identity in the Horizon UI

Projects

Name	Description	Project ID	Enabled	Actions
alt_demo		0d25016d972d4039f73bb66148a30e02	Yes	Manage Members, Edit Project, View Usage, Modify Quotas, Delete Project
admin		24a5ebae5e9342f1bac98fb0f10b64a6	Yes	
swifttenantest2		3ac7d770d64b4b7297c0a912dc8d7555	Yes	
testproject		9563aecf4abc4d992a35314487ac4d	Yes	
service		a1949c36f74d41a		
demo		a6605682be6418	Admin	
invisible_to_admin		e83edca6903405	Identity	
swifttenantest1		eee4cbd432e842		

Users

User Name	Email	User ID	Enabled	Actions
cinder		15579b6dc9354858b1069dc04e91ad19	Yes	Edit, Change Password, Disable User, Delete User
heat		2008bf161a0b476298ee6029807c4d917		
glance-swift	glance-swift@example.com	3c30b2633c61437ca263ec224b7ea662		
admin		4a6c4f473124a43d38a9c63186e70fde	Yes	Edit
swift		55663526ab254f9a7cb4b4e3acd0b	Yes	Edit
alt_demo	alt_demo@example.com	646bd2816e02414f9ffde604b9181b60	Yes	Edit
testuser		6d472da2bd6544c8a120fe63b09a3aa5	Yes	Edit
swiftuserest3	test3@example.com	70828c8c50434e03ad29c14102afcb	Yes	Edit

Hewlett Packard
Enterprise

Copyright 2017 Hewlett Packard Enterprise Development LP

24

Keystone Identity in the Horizon UI

Keystone Identity can be managed in the Horizon UI, using the Identity section of the navigation panel.

Setting user policies in OpenStack®

- The

`/etc/[SERVICE_CODENAME]/policy.json` file controls what users, in accordance with their role, are allowed to do for a given service

- With `[SERVICE_CODENAME]`= nova, keystone, glance, and so on
- Example: `/etc/nova/policy.json` specifies the access policies for the Compute service

```
stack@hpdevstack:~/devstack$ cat /etc/keystone/policy.json
{
    "admin_required": "role:admin or is_admin:1",
    "service_role": "role:service",
    "service_or_admin": "rule:admin_required or rule:service_role",
    "owner": "user_id:%(user_id)s",
    "admin_or_owner": "rule:admin_required or rule:owner",
    "default": "rule:admin_required",

    "identity:get_region": "",
    "identity:list_regions": "",
    "identity:create_region": "rule:admin_required",
    "identity:update_region": "rule:admin_required",
    "identity:delete_region": "rule:admin_required",

    "identity:get_service": "rule:admin_required",
    "identity:list_services": "rule:admin_required",
    "identity:create_service": "rule:admin_required",
    "identity:update_service": "rule:admin_required",
    "identity:delete_service": "rule:admin_required",

    "identity:get_endpoint": "rule:admin_required",
    "identity:list_endpoints": "rule:admin_required",
    "identity:create_endpoint": "rule:admin_required",
    "identity:update_endpoint": "rule:admin_required",
    "identity:delete_endpoint": "rule:admin_required",
}
```



© Copyright 2017 Hewlett Packard Enterprise Development LP

25

Setting user policies in OpenStack®

The `/etc/[SERVICE_CODENAME]/policy.json` file controls what users are allowed to do for a given service. For example, `/etc/nova/policy.json` specifies the access policy for the Compute service, `/etc/glance/policy.json` specifies the access policy for the Image service, and `/etc/keystone/policy.json` specifies the access policy for the Identity service.

The default `policy.json` files in the Compute, Identity, and Image service recognize only the administrator role; all operations that do not require the administrator role are accessible by default by any user that has any role in a project.

If you want to restrict users from performing operations in, for example, the Compute service, you need to create a role in the Identity service and then modify `/etc/nova/policy.json` so that this role is required for Compute operations.

For example, this line in `/etc/nova/policy.json` specifies that there are no restrictions on which users can create volumes; if the user has any role in a project, they will be able to create volumes in that project:

`"volume:create": []`,

To restrict the creation of volumes to users who have a compute-user role in a particular project, you need to add `"role:compute-user"`, for example:

`"volume:create": ["role:compute-user"]`,

The policy engine reads entries from the `policy.json` file. The actual location of this file might vary from distribution to distribution. Entries can be updated while the system is running, and no service restart is required; that is, every time the policy file is updated, the policies are automatically reloaded. You can update policies by editing the policy file.

Keystone configuration files

- After Keystone is installed, it is configured by using the CLI client or by modifying two configuration files and then restarting the Keystone service

File location	Explanation
/etc/keystone/keystone.conf	The primary configuration file of Keystone, which includes general and driver-specific configuration parameters
/etc/keystone/keystone-paste.ini	Configuration entries (Web Server Gateway Interface pipeline definitions) for Python WSGI-based applications



© Copyright 2017 Hewlett Packard Enterprise Development LP

26

Keystone configuration files

After Keystone is installed, it is configured by using the CLI client or by modifying configuration files and then restarting the Keystone service.



Keystone troubleshooting

Verifying the Identity service installation

- Verify that the authentication is behaving as expected:
 - Use an established username and password to generate an authentication token

```
$ keystone --os-username=admin --os-password=hpinvent \--os-auth-url=http://localhost:35357/v2.0 token-get
+-----+
| Property |          Value          |
+-----+
| expires  | 2014-12-04T23:01:27Z |
| id       | e7ff38b91b0943c4b675b0b2be15a194 |
| tenant_id | 7beb159b0a1d44f7b4e45c44ed2e9460 |
| user_id  | 01344065016e4cacbafe515feb3033b7 |
+-----+
```

- Receive a token in response, paired with your user ID

NOTE: The successful execution of a `keystone` command verifies that the Identity service is running on the expected endpoint and that your user account is established with the expected credentials.

Verifying the Identity service installation

To verify if the authentication is behaving as expected, enter the `keystone token-get` command with the established username and password to generate an authentication token. Successful execution of the `keystone` command will confirm that the Identity service is running on the expected endpoint, and that your user account is established with the expected credentials.

Keystone troubleshooting

A Keystone issue	Recommended steps
Check the service status	<ul style="list-style-type: none">– Check whether your Keystone service is running– Check the configuration file for syntax errors
Change the logging to debug for Keystone	<ul style="list-style-type: none">– Open the <code>/etc/keystone/keystone.conf</code> file and set <code>debug=TRUE</code>– Restart the Keystone service and monitor the log file output
Unable to communicate with the Identity service	<ul style="list-style-type: none">– Check if the Keystone service is working– Check if the <code>admin_token</code> configuration in the Keystone configuration file is configured correctly and restart the service if necessary
A service returns “Unauthorized Error” when trying to use Keystone	<ul style="list-style-type: none">– A token that you supplied does not have authorization to the resource– Check if the token has expired, and use the debug option running the affected service to monitor for error messages (for example, <code>nova --debug</code>)

Keystone troubleshooting

Keystone is an essential OpenStack® service. A problem with Keystone functionality immediately reflects on all services. Some of the troubleshooting steps to repair some common Keystone issues are provided in the table on the slide above.

Module summary

In this module:

- The purpose of Keystone was described
- The high-level architecture of Keystone was described
- Keystone was configured
- The completion of common Keystone management tasks was explained
- Troubleshooting of some common Keystone-related issues was addressed



© Copyright 2017 Hewlett Packard Enterprise Development LP

30

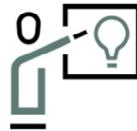
Module summary

In this module:

- The purpose of Keystone was described
- The high-level architecture of Keystone was described
- Keystone was configured
- The completion of common Keystone management tasks was explained
- Troubleshooting of some common Keystone-related issues was addressed

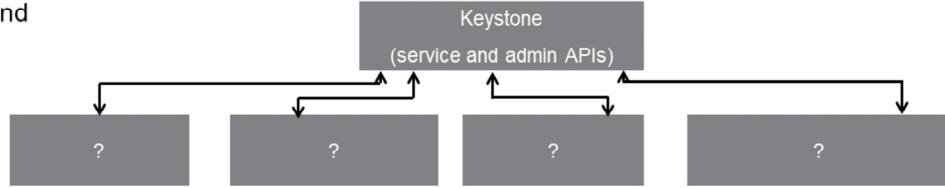
Learning check

Learning check sample



What are the four Keystone service components? (Select all that apply.)

- A. Keystone backend
- B. Token backend
- C. Database backend
- D. Service catalog backend
- E. Extensions backend
- F. Policy backend
- G. Identity backend
- H. Nova backend



Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

32

Learning check

What are the four Keystone service components? (Select all that apply.)

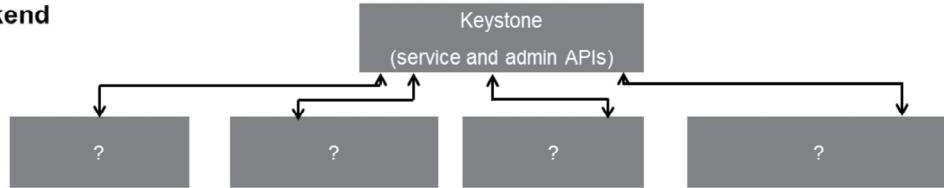
- A. Keystone backend
- B. Token backend
- C. Database backend
- D. Service catalog backend
- E. Extensions backend
- F. Policy backend
- G. Identity backend
- H. Nova backend

Learning check answer



What are the four Keystone service components? (Select all that apply.)

- A. Keystone backend
- B. Token backend**
- C. Database backend
- D. Service catalog backend**
- E. Extensions backend
- F. Policy backend**
- G. Identity backend**
- H. Nova backend



© Copyright 2017 Hewlett Packard Enterprise Development LP

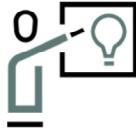
33

Learning check answer

What are the four Keystone service components? (Select all that apply.)

- A. Keystone backend
- B. Token backend**
- C. Database backend
- D. Service catalog backend**
- E. Extensions backend
- F. Policy backend**
- G. Identity backend**
- H. Nova backend

Learning check



Which user management components are used by Keystone? (Select all that apply.)

- A. Users
- B. Roles
- C. Projects
- D. Services
- E. Networks
- F. Databases

Learning check

Which user management components are used by Keystone? (Select all that apply.)

- A. Users
- B. Roles
- C. Projects
- D. Services
- E. Networks
- F. Databases

Learning check answer



Which user management components are used by Keystone? (Select all that apply.)

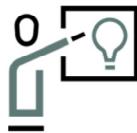
- A. Users
- B. Roles
- C. Projects
- D. Services
- E. Networks
- F. Databases

Learning check answer

Which user management components are used by Keystone? (Select all that apply.)

- A. Users
- B. Roles
- C. Projects
- D. Services
- E. Networks
- F. Databases

Learning check



With a service, what mechanism can you use to specify the actions a user is allowed to perform?

- A. Access restrictions
- B. Projects
- C. Policies
- D. Endpoints

Learning check

With a service, what mechanism can you use to specify the actions a user is allowed to perform?

- A. Access restrictions
- B. Projects
- C. Policies
- D. Endpoints

Learning check answer



With a service, what mechanism can you use to specify the actions a user is allowed to perform?

- A. Access restrictions
- B. Projects
- C. Policies**
- D. Endpoints

Learning check answer

With a service, what mechanism can you use to specify the actions a user is allowed to perform?

- A. Access restrictions
- B. Projects
- C. Policies**
- D. Endpoints



**Hewlett Packard
Enterprise**



**Hewlett Packard
Enterprise**

Fundamentals of OpenStack® Technology

Module 5—OpenStack® Image Services (Glance)

H6C68S E.01

© Copyright 2017 Hewlett Packard Enterprise Development LP

Learning objectives

After completing this module you should be able to:

- Describe the function of Glance
- Identify the Glance configuration files
- Locate the pre-built Glance images
- Explain how to use the CLI and Horizon UI management tools for Glance
- Explain common troubleshooting steps for the Glance service



© Copyright 2017 Hewlett Packard Enterprise Development LP

2

Learning objectives

After completing this module, you should be able to:

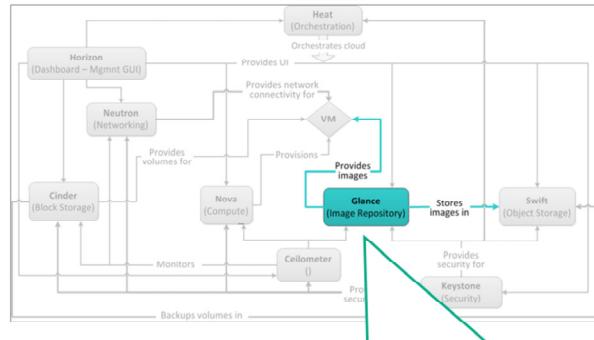
- Describe the function of Glance
- Identify the Glance configuration files
- Locate the pre-built Glance images
- Explain how to use the CLI and Horizon UI management tools for Glance
- Explain common troubleshooting steps for the Glance service



Glance overview

Glance functionality

- Allows updating and discovering data assets that are used with other services
- Currently includes images and metadata definitions
- Has a RESTful API that allows querying of VM image metadata as well as retrieval of the actual image
- VM images can be stored in a variety of locations from simple file systems to object-storage systems like Swift
- You can use stored images as templates to get new instances up and running quickly and consistently
- Back up a system by creating a copy (snapshot) of that system as it exists in that point in time



Glance is not used to store images, but it can be used to:

- Stream images between OpenStack and systems outside of OpenStack
- Cache images

Glance functionality

Glance is an OpenStack® project with the primary function of providing discovery registration and delivery services for disk and server images.

Glance enables you to choose available images or create your own from existing servers. Also, you can store images that can be used as templates to get new instances up and running quickly and consistently.

Glance supports the storage and cataloging of snapshots, enabling virtual machines (VMs) to be backed up quickly.

Communication is enabled through its restful supported API.

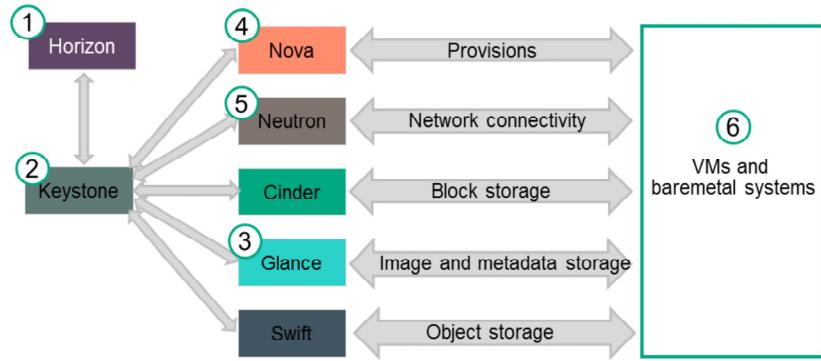
Glance has the ability to copy (or snapshot) a server image and then to store it promptly. Stored images can be also used as templates to store and catalog unlimited backup.

The concept of generalizing Glance into a central artifact management system has been considered, the rationale being that the OpenStack® platform requires the ability to manage and track a number of different types of artifacts across its various projects. These artifacts are associated with the operation of the cloud, not the operation of user functionality. For example, in addition to VM images, the Glance metadata system could track Heat templates. Glance would provide a single consistent location for such artifact lookups while supporting an array of storage backends, each appropriate for a particular range of artifact types.

For more information, refer to <https://blueprints.launchpad.net/glance/+spec/artifact-repository-api>.

Example Glance use case

Creating an instance



1. Horizon is accessed
2. Keystone verifies the authentication and authorization of the user and user/project approved options are made available in the Horizon UI
3. An image is selected from the table of available Glance images in the Horizon UI (e.g. `cirros-0.3.4-x86_64-uec`)
4. Compute (Nova) parameters are specified (e.g. number of VCPUs, disk space, RAM, and so on)
5. Network is selected
6. Instance is launched

Example Glance use case—Creating an instance

The slide above describes the process of an instance creation from Glance perspective. The steps are straightforward with exception of the second step that involves authorization and authentication.

Authentication primarily deals with user identity. There is a large number of systems that handle this “checkpoint” level of identity and access management, and help to reduce the number of credentials you need to provide, often through single sign-on (SSO).

Authorization, on the other hand, refers to what a user or a system is allowed to access. Authorization can manage service-to-service as well as user-to-service permissioning. For example, an authorization platform can determine if a user is a developer, and then grant the user the permission to push source code to a Git repository, but prohibit them from directly changing the software deployed into the production environment.

Glance API version states as of Newton release

- Image API v2 (**current**)
- As of Newton release:
 - VHDX is added to the list of supported disk formats
 - The Image (Glance) version 1 API has been deprecated
 - The S3 (Amazon) store driver has been removed



© Copyright 2017 Hewlett Packard Enterprise Development LP

6

Glance API version states as of Newton release

The API status reflects the state of the endpoint on the service:

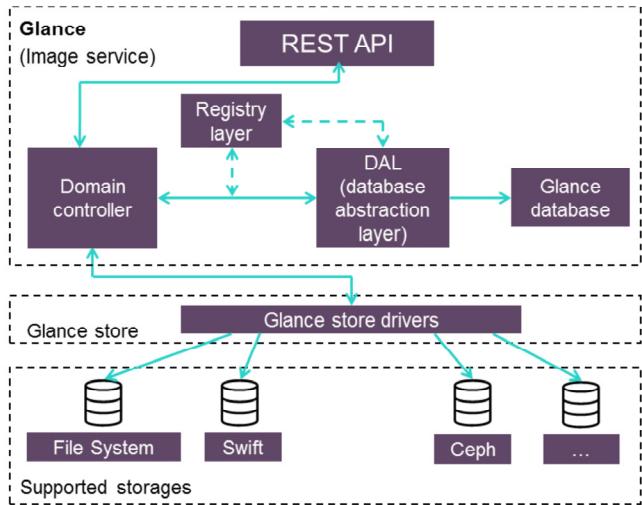
- *Current* indicates a stable version that is up-to-date, recent, and might receive future versions. This endpoint should be prioritized over all others.
- *Supported* is a stable version that is available on the server. However, it is not likely the most recent available and might not be updated or might be deprecated at some time in the future.
- *Deprecated* is a stable version that is still available but is being deprecated and might be removed in the future.

As of Newton release, the following changes have been implemented to Glance service:

- VHDX is added to the list of supported disk formats
- The Images (Glance) version 1 API has been deprecated
- The S3 (Amazon) store driver has been removed

Glance architecture

- Glance comprises following primary components:
 - Glance API
 - Domain Controller
 - Glance registry
 - Pluggable image storage system:
 - Glance store
 - Glance storage drivers
 - Metadata Database
 - DAL (Database Abstraction Layer)
 - Glance database



Hewlett Packard Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

7

Glance architecture

The Glance architecture comprises four primary components:

- Glance API, which allows users to query the VM image metadata and retrieve actual images from the HTTP requests. The API routes requests from clients to registries of image metadata and to its back-end stores.
- Glance storage adapter, which provides an interface between the Glance and vendor-specific devices for performing image saving, retrieval, deletion, and verification operations.
- Glance database, for storage of image-related information such as location of the images, image properties, and image membership.
- Glance registry, which stores and retrieves metadata about images

Glance-supported backend stores

- Types of Glance-supported storage backends used to store disk images, include:
 - **Swift**—The highly-available object storage project in OpenStack®
 - **Cinder**—The block-storage project in OpenStack®
 - **Filesystem**—Writes image files to the Glance node local filesystem
 - **HTTP**—VM images that are available through HTTP somewhere on the Internet (this storage is read only)
 - **GridFS**—MongoDB images
 - **Ceph RBD (RADOS [Reliable Autonomic Distribute Object Store] Block Device)**—Part of the Ceph distributed storage system
 - **Sheepdog**—A distributed storage system for QEMU hypervisor and RESTful services that provides highly available block level storage volumes that can be attached to QEMU-based virtual machines
 - **VMware ESX**—Datastore as a storage backend



© Copyright 2017 Hewlett Packard Enterprise Development LP



Glance-supported backend stores

VM images made available through Glance can be stored in a variety of locations. The OpenStack® Image service stores virtual machine images in a file system backend on the Glance server by default. This simple backend writes image files to the local file system. A different backend (such as a Cinder volume or Swift) can be used as an image store.

Types of Glance-supported storage backends used to store disk images include:

- **Swift**—The highly available object storage project in OpenStack®
- **Cinder**—The block-storage project in OpenStack®
- **Filesystem**—Writes image files to the Glance node local filesystem
- **HTTP**—VM images that are available through HTTP somewhere on the Internet (this store is read only)
- **GridFS**—MongoDB images
- **Ceph RBD (RADOS [Reliable Autonomic Distribute Object Store] Block Device)**—Part of the Ceph distributed storage system
- **Sheepdog**—A distributed storage system for QEMU hypervisor and RESTful services that provides highly available block level storage volumes that can be attached to QEMU-based virtual machines
- **VMware ESX**—Datastore as a storage backend

Glance disk and container formats

- Glance-supported image disk formats:
 - **AKI** (Amazon Kernel Image)
 - **ARI** (Amazon Ramdisk Image)
 - **AMI** (Amazon Machine Image)
 - **ISO**: An archive format for the data contents of an optical disk (for example, CDROM)
 - **Raw**: An unstructured disk format; if you have a file without an extension, it is possibly a raw format
 - **QCOW2** (QEMU/KVM Copy-On-Write)
 - **VDI** (VirtualBox)
 - **VHD** (Hyper-V, VMware, Xen, VirtualBox)
 - **VMDK** (VMware)

- Glance-supported container formats:
 - **AKI**: Indicates that what is stored in Glance is an Amazon kernel image
 - **AMI**: Indicates that what is stored in Glance is an Amazon machine image
 - **ARI**: Indicates that what is stored in Glance is an Amazon ramdisk image
 - **Bare**: Indicates that there is no container or metadata envelope for the image
 - **Docker**: Indicates what is stored in Glance is a Docker tar archive of the container filesystem
 - **OVA**: Indicates that what is stored in Glance is an OVA tar archive file
 - **OVF**: This is the OVF container format

NOTE: The container format string that is specified when storing an image to Glance is not currently used by Glance or other OpenStack® components, so if you are unsure what to use, it is safe to specify **bare** as the container format.

Glance disk and container formats

When adding an image to Glance, you must specify the disk format and the container format of the VM image. Disk and container formats are configurable on a per-deployment basis.

The disk format of a VM image is the format of the underlying disk image. Virtual appliance vendors have different formats for laying out the information contained in a VM disk image.

The container format refers to whether the VM image is in a file format that also contains metadata about the actual virtual machine.

Raw and QCOW2 are the most common types of Glance-supported images.

Note: The container format string is not currently used by Glance or other OpenStack® components, so if you are unsure of what to use, it is safe to specify **bare** as the container format.

Glance images

Pre-built images for OpenStack®

- **CentOS**—Official CentOS images
- **Red Hat**—Official Red Hat Enterprise Linux images
- **Debian**—Official Debian images
- **CirrOS**—Test images
- **Fedora**—Fedora-based images
- **Microsoft**—OpenStack Windows Server 2012 Standard Evaluation image
- **Ubuntu**—Official Ubuntu images
- **SUSE**—openSUSE and SLES images
- **Other**—Images for other Linux distributions, Rackspace Cloud Builders (multiple distribution) images

NOTE: The *OpenStack Virtual Machine Image Guide* is available at:
http://docs.openstack.org/image-guide/content/ch_obtaining_images.html.



© Copyright 2017 Hewlett Packard Enterprise Development LP

11

Pre-built images for OpenStack®

While Nova and Glance can work with most VM images, it is useful to create images that are cloud-aware and capable of hosting metadata, receiving injected files, and automatically running configuration scripts on boot. The simplest way to obtain a virtual machine image that works with OpenStack® is to download one that someone else has already created. A VM image is a single file that contains a virtual disk that has a bootable operating system installed on it.

VM images come in different formats, depending on the intended hypervisor usage. For example, if your deployment uses Quick Emulator (QEMU) or KVM, the recommended format to use is QCOW2. For Windows Server 2012 Standard Evaluation, Hyper-V-, KVM-, and XenServer/XCP-based images are available.

To find more information about supported images, see the *OpenStack Virtual Machine Image Guide* available at: http://docs.openstack.org/image-guide/content/ch_obtaining_images.html.

Custom OpenStack® images

- Custom VM Images can be created and configured for a range of systems
- The *OpenStack Image Guide* describes the process for:
 - CentOS
 - Ubuntu, Debian
 - Fedora, Red Hat
 - Microsoft Windows
 - FreeBSD
 - openSUSE and SLES
- Documentation:
 - Creating an image: http://docs.openstack.org/image-guide/content/ch_openstack_images.html
 - Meeting the image requirements: <http://docs.openstack.org/image-guide/content/>



© Copyright 2017 Hewlett Packard Enterprise Development LP

12

Custom OpenStack® images

In production environment, you will most probably have to make your own Glance images. Custom VM images can be created and configured for a range of systems. Documentation is provided (as listed on the slide above), for following operating systems:

- CentOS
- Ubuntu, Debian
- Fedora, Red Hat
- Microsoft Windows
- FreeBSD
- openSUSE and SLES

Handling custom Glance images

- You can build your own custom Glance images using the steps provided at <http://docs.openstack.org/image-guide/create-images-manually.html>
- To modify the image before uploading it to the OpenStack® Image service, use some of the available tools described at <http://docs.openstack.org/image-guide/modify-images.html>
- To convert between image formats, use the tools described at <http://docs.openstack.org/image-guide/convert-images.html>

NOTE: Glance supports attaching the metadata information to the images. Metadata can be used by the Nova scheduler to determine which VM can be used to host a specific image. More information about handling the image metadata is available at http://docs.openstack.org/cli-reference/content/glanceclient_commands.html.



© Copyright 2017 Hewlett Packard Enterprise Development LP

13

Handling custom Glance images

Besides creating the image from scratch, it is possible to modify an existing image. After you have downloaded or created the custom Glance image, you can perform modifications or convert it between available VM-supported file formats. Some of the most important tools and steps to create your own custom Glance images are provided in the *OpenStack Virtual Machine Image Guide* at <http://docs.openstack.org/image-guide/create-images-manually.html>.

To learn how to modify existing images, see the document at <http://docs.openstack.org/image-guide/modify-images.html>.

If you want to convert the format of the image provided, you can use the `qemu-img` tool. More information and detailed instructions can be found at <http://docs.openstack.org/image-guide/convert-images.html>.

Cloud-init

Customize Linux-based VM images at the time of the first boot

- Handles early initialization of a cloud instance to:
 - Set a default locale
 - Set the host name
 - Generate SSH private keys
 - Add SSH keys to a user's `.ssh/authorized_keys` so they can log in
 - Set up ephemeral mount points
 - Set instance-specific parameters by means of a user-data supplied by a user at the instance launch
- Documentation:
 - Cloud-init reference: <http://cloudinit.readthedocs.org/en/latest/>
 - User-data boot scripts and cloud-init:
<http://docs.openstack.org/user-guide/content/user-data-boot-scripts-and-cloud-init.html>



© Copyright 2017 Hewlett Packard Enterprise Development LP

14

Cloud-init—Customize Linux-based VM images at the time of the first boot

Some of the OpenStack® images provide the cloud-init service, which is activated the first time an image is booted. This service is used for modifying the image based on the input provided by the cloud administrator.

Linux-based systems use the init script to initialize the environment immediately after a boot. This script checks the status of the file systems, mounts the file systems, creates the necessary special-device files, and so on.

The image of the Linux operating system (OS) running in the cloud environment differs from the one used on physical systems, because the OS is assembled each time a server is started, using the image shared between different instances. To handle this OpenStack®-specific task, you must use a special type of init script called cloud-init.

The initialization process is slightly different for each Linux distribution, so cloud-init scripts are tailored for the different Linux distributions such as Ubuntu (<https://help.ubuntu.com/community/CloudInit>) or Red Hat (<http://cloudinit.readthedocs.org/en/latest/>).

Public tools for image creation

- **Oz**—A Python application that interacts with KVM to step through the process of installing a VM
- **VMBuilder**—A command-line tool that can be used to create VM images for different hypervisors
- **BoxGrinder**:
 - Is a tool for creating VM images, which it calls appliances
 - Creates Fedora, Red Hat Enterprise Linux, or CentOS images
 - Is supported on Fedora only
- **VeeWee**—Often used to build Vagrant boxes, but it can also be used to build KVM images
- **Imagefactory**—Automates building, converting, and uploading images and uses Oz as its backend
- **SUSE Studio**:
 - Is a web application that supports the creation of physical, virtual, or cloud-based applications
 - Includes support for building images for OpenStack®-based clouds using SUSE Linux Enterprise and openSUSE as distributions



© Copyright 2017 Hewlett Packard Enterprise Development LP

15

Public tools for image creation

Several tools are designed to automate image creation:

- **Oz** is a command-line tool that automates the process of creating a VM image file. Oz is a Python application that interacts with KVM to step through the process of installing a VM. It uses a predefined set of kick-start (Red Hat-based systems) and preseed files (Debian-based systems) for operating systems that it supports. Also, it can be used to create Microsoft Windows images.
- **VMBuilder** is a command-line tool that creates VM images for different hypervisors. The version of VMBuilder that ships with Ubuntu can only create Ubuntu VM guests. The version of VMBuilder that ships with Debian can create Ubuntu and Debian VM guests. The *Ubuntu Server Guide* has documentation on how to use VMBuilder to create an Ubuntu image.
- **BoxGrinder** is another tool for creating VM images, which it calls appliances. BoxGrinder can create Fedora, Red Hat Enterprise Linux, or CentOS images. BoxGrinder is currently supported only on Fedora.
- **VeeWee** is often used to build Vagrant boxes, but it can also be used to build KVM images.
- **Imagefactory** is a newer tool that is designed to automate the building, converting, and uploading of images to different cloud providers. It uses Oz as its backend and includes support for OpenStack®-based clouds.
- **SUSE Studio** is a web application for building and testing software applications in a web browser. It supports the creation of physical, virtual, or cloud-based applications. It includes support for building images for OpenStack®-based clouds using SUSE Linux Enterprise and openSUSE as distributions.

Common Glance management tasks

Glance-related configuration files

File name	Description
glance-api.conf	Main Glance configuration file
glance-api-paste.ini	Image Service API middleware pipeline configuration
glance-cache.conf	Image cache configuration
glance-registry.conf	Metadata storage interface configuration
glance-registry-paste.ini	Registry middleware pipeline configuration
policy.json	Role-based security policies for Glance

NOTE: Common configuration options in Glance are provided at <http://docs.openstack.org/developer/glance/configuring.html> and <http://docs.openstack.org/newton/config-reference/image.html>.



© Copyright 2017 Hewlett Packard Enterprise Development LP

17

Glance-related configuration files

Glance has a number of options that you can use to configure the Glance API server, the Glance Registry server, and the various storage backends that Glance can use to store images. Most of the configuration is done via configuration files, with the Glance API server and Glance Registry server using separate configuration files.

When starting up a Glance server, you can specify the configuration file to use. If you do not specify a configuration file, Glance will look in the following directories for a configuration file, in the following order:

- ~/.glance
- ~/
- /etc/glance
- /etc

The Glance API server configuration file should be named `glance-api.conf`. Similarly, the Glance Registry server configuration file should be named `glance-registry.conf`. If you installed Glance via your operating system's package management system, it is likely that you will have sample configuration files installed in `/etc/glance`.

The PasteDeploy configuration (controlling the deployment of the WSGI application for each component) can be found by default in `<component>-paste.ini` alongside the main configuration file, `<component>.conf`. For example, `glance-api-paste.ini` corresponds to `glance-api.conf`.

Common Glance-related CLI commands

```
student@ubuntu14libdevStack:~/devstack$ openstack image list
+---+-----+-----+-----+
| ID | Name | Status |
+---+-----+-----+-----+
| f826796b-73c1-45d3-929f-43e759a89004 | cirros-0.3.4-x86_64-uec | active |
| a044078d-5bcf-4df8-8172-27346d419181 | cirros-0.3.4-x86_64-uec-ramdisk | active |
| 5d3b18a2-9e-4f38-ad18-86faa2ed4af3 | cirros-0.3.4-x86_64-uec-kernel | active |
| 8c52dde7-745e-4e06-a328-fc092e56905c | cirros-a-Cloud-Base-23-20151030.x86_64 | active |
| fa1c0ace-fee3-44e7-9375-0991a868e83 | cirros-0.3.2-x86_64-disk | active |
+---+-----+-----+-----+
student@ubuntu14libdevStack:~/devstack$ openstack image show 8c52dde7-745e-4e06-a328-fc092e56905c
+-----+-----+
| Field | Value |
+-----+-----+
| checksum | 3bb62e2e1909c89ff72ba4d5f5c0005d5 |
| container_format | raw |
| created_at | 2016-01-08T01:51:41Z |
| disk_format | qcow2 |
| file | /v2/images/8c52dde7-745e-4e06-a328-fc092e56905c/file |
| id | 8c52dde7-745e-4e06-a328-fc092e56905c |
| min_disk | 0 |
| min_ram | 0 |
| name | cirros-a-Cloud-Base-23-20151030.x86_64 |
| owner | 5cab5c4beaf44d7aab14c706d5c187 |
| protected | False |
| schema | /v2/schemas/image |
| size | 234363392 |
| status | active |
| tags | |
| updated_at | 2016-01-08T01:51:45Z |
| virtual_size | None |
| visibility | public |
+-----+-----+
```

CLI command	Description
openstack image add project	Associate an image to a project
openstack image create/delete	Create/upload or delete an image
openstack image list	Metadata storage interface configuration
openstack image remove project	Disassociate an image from a project
openstack image save	Save image locally
openstack image set	Modify image properties
openstack image show	Describe the attributes of a selected image
openstack image update	Update a specific image

NOTE: For help on a specific Glance command, enter `openstack image COMMAND --help`.

Common Glance-related CLI commands

As with most of your OpenStack® services, you can use the `openstack help` command to view the usage, positional arguments, and optional arguments that are available for the `openstack` CLI commands related to Glance.

Using the CLI to load an image in Glance

- Obtain an image:

```
$ wget http://download.cirros-cloud.net/0.3.2/cirros-0.3.2-i386-disk.img
```

- Upload a test image:

```
$ openstack image create --name myFirstImage --is-public true \  
--container-format bare --disk-format qcow2 \  
< cirros-0.3.1-x86_64-disk.img
```

- Verify that the image is listed:

```
$ openstack image list
```

- Display the image information:

```
$ openstack image show 86b76b4a-de6a-44c2-ab40-1785da79010e
```



© Copyright 2017 Hewlett Packard Enterprise Development LP

19

Using the CLI to load an image in Glance

You can use the `openstack image create` command to add a new VM image in Glance, and use the `openstack image update` to modify the properties of an image that has been updated.

The `image-create` command takes several optional arguments, but you should specify:

- A name for your image by using the `--name` flag
- The disk format by using `--diskformat`
- The container format by using `--container-format`

You can pass in the file by using the standard input or by using the `file` command. After creating the image, you can use `openstack image show` with the Universal Unique Identifier (UUID) of the available image to display the image details.

Glance operations using Horizon

The screenshot shows the OpenStack Horizon interface for managing images. The left sidebar navigation includes 'Project', 'Compute' (selected), 'Overview', 'Instances', 'Volumes', 'Images' (selected), 'Access & Security', 'Network', 'Orchestration', 'Object Store', 'Admin', and 'Identity'. The main content area is titled 'Images' and displays a table of available images. One image, 'cirros-0.3.4-x86_64-uec', is selected and its details are shown in a modal window. The modal includes sections for 'Information' (with fields like Name, ID, Owner, Status, Public, Protected, Checksum, Created, Updated) and 'Specs' (with fields like Size, Container Format, Disk Format). A callout box highlights the 'Actions available for the associated image' button in the 'Actions' column of the main table.

20

Glance operations using Horizon

Most of the Glance operations can also be performed using the horizon UI. As shown in the screenshot, clicking the **Images** option in the navigation pane displays a table of available images. From this table you can:

- Click an image name link to view details about the selected image
- View the status of the image, whether it is public, private, or protected, and its disk format

The Actions column of the table enables you to update some of the image information, including its name, description, architecture, disk format, and the information on whether it is publicly available and whether it is protected.

The Horizon UI also enables you to create an image and delete existing images.

Troubleshooting Glance

Troubleshooting

– Checking Glance:

- Glance does not have an integrated tool to check

```
ps -ef | grep glance
```

```
netstat -ant | grep 9292.*LISTEN
```

– Check some of the common issues and their solutions

Common issues	Solutions
The Keystone integration is incorrect	<ul style="list-style-type: none"> – Check <code>/etc/glance/glance-api.conf</code> in the <code>[keystone_auth_token]</code> section
The OS_* environment variables are not set correctly	<ul style="list-style-type: none"> – Check the environment variables – For DevStack, look at <code>/opt/stack/devstack/openrc</code>
The MySQL database parameters are set incorrectly	<ul style="list-style-type: none"> – Check <code>/etc/glance/glance-api.conf</code> in the <code>[DEFAULT]</code> section – Look for the <code>sql_connection</code> parameter

Troubleshooting

Glance does not have a tool to check, so you have to use system commands. To check if the Glance service is running and listening on the 9292 default port, enter the following commands:

```
ps -ef | grep glance
```

```
netstat -ant | grep 9292.*LISTEN
```

If you cannot find a Glance process that is running properly, check the Glance configuration files. Some of the common issues and their solutions are listed in the table on the slide.

You can also check the log files located in the `/var/log/glance` folder.

Glance-related log files

File name	Description
/var/log/glance/api.log	API-related logs
/var/log/glance/registry.log	Registry-related logs



© Copyright 2017 Hewlett Packard Enterprise Development LP

23

Glance-related log files

Glance-related log files are located in the `/var/log/glance` folder. These files can be searched for image service-related error messages.

Module summary

- In this module:
 - The function of Glance was described
 - The Glance configuration files were identified
 - The pre-built Glance images were located
 - The usage of the CLI and Horizon UI management tools for Glance was explained
 - Common troubleshooting steps for the Glance service were explained



© Copyright 2017 Hewlett Packard Enterprise Development LP

24

Module summary

In this module:

- The function of Glance was described
- The Glance configuration files were identified
- The pre-built Glance images were located
- The usage of the CLI and Horizon UI management tools for Glance was explained
- Common troubleshooting steps for the Glance service were explained

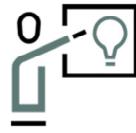
Learning check



© Copyright 2017 Hewlett Packard Enterprise Development LP

25

Learning check



What is the function of the Glance registry?

- A. Accepts image API calls for image discovery, retrieval, and storage
- B. Stores, processes, and retrieves metadata for images
- C. Stores image metadata and supports many backends, including MySQL, SQLite, and MongoDB
- D. Supports normal file systems, RADOS block devices, Amazon S3, and HTTP for image storage

Learning check

What is the function of the Glance registry?

- A. Accepts image API calls for image discovery, retrieval, and storage
- B. Stores, processes, and retrieves metadata for images
- C. Stores image metadata and supports many backends, including MySQL, SQLite, and MongoDB
- D. Supports normal file systems, RADOS block devices, Amazon S3, and HTTP for image storage

Learning check answer



What is the function of the Glance registry?

- A. Accepts image API calls for image discovery, retrieval, and storage
- B. Stores, processes, and retrieves metadata for images**
- C. Stores image metadata and supports many backends, including MySQL, SQLite, and MongoDB
- D. Supports normal file systems, RADOS block devices, Amazon S3, and HTTP for image storage

Learning check answer

What is the function of the Glance registry?

- A. Accepts image API calls for image discovery, retrieval, and storage
- B. Stores, processes, and retrieves metadata for images**
- C. Stores image metadata and supports many backends, including MySQL, SQLite, and MongoDB
- D. Supports normal file systems, RADOS block devices, Amazon S3, and HTTP for image storage

Learning check



Which statement about Glance is true?

- A. OpenStack® contains a set of default images to use with Glance.
- B. An image must be uploaded to be used by Glance.
- C. Only images preconfigured by a vendor can be used with Glance.
- D. Glance automatically patches images.

Learning check

Which statement about Glance is true?

- A. OpenStack® contains a set of default images to use with Glance.
- B. An image must be uploaded to be used by Glance.
- C. Only images preconfigured by a vendor can be used with Glance.
- D. Glance automatically patches images.

Learning check answer



Which statement about Glance is true?

- A. OpenStack® contains a set of default images to use with Glance.
- B. An image must be uploaded to be used by Glance.**
- C. Only images preconfigured by a vendor can be used with Glance.
- D. Glance automatically patches images.

Learning check answer

Which statement about Glance is true?

- A. OpenStack® contains a set of default images to use with Glance.
- B. An image must be uploaded to be used by Glance.**
- C. Only images preconfigured by a vendor can be used with Glance.
- D. Glance automatically patches images.



**Hewlett Packard
Enterprise**

Hewlett Packard
Enterprise

Fundamentals of OpenStack® Technology

Module 6—OpenStack® Networking Service (Neutron)

H6C68S E.01

© Copyright 2017 Hewlett Packard Enterprise Development LP

Learning objectives

After completing this module, you should be able to:

- Discuss the primary components of a virtual network
- Describe the purpose and components of OpenStack® Neutron
- Identify common configuration tasks for Neutron
- Explain basic Neutron networking configuration examples
- Create OpenStack® networks, subnets, and routers



© Copyright 2017 Hewlett Packard Enterprise Development LP

2

Learning objectives

After completing this module, you should be able to:

- Discuss the primary components of a virtual network
- Describe the purpose and components of OpenStack® Neutron
- Identify common configuration tasks for Neutron
- Explain basic Neutron networking configuration examples
- Create OpenStack® networks, subnets, and routers



Virtual networking concepts

Hosting device virtual machines and baremetal machines

- OpenStack® uses virtual networking to interconnect various components (for example, compute, storage node)
- Networking functions are performed by software on compute systems instead of dedicated network devices
- OpenStack® does not reinvent networking; it simply provides the ability to use the existing open and vendor-specific virtual networking technology

Compute node A
VM hosting device (e.g. DL360 system) that has a hypervisor OS installed (e.g. VMware, KVM, or Hyper-V)

Compute node A
Bare-metal machine (e.g. DL360 system) with 1 OS installed at most



There can be multiple VMs on a hosting device and each VM is a separate instance of an OS

Hosting device virtual machines and bare metal machines

This section of the module provides some basic operations and terminology of virtual networking in general, much of which can be used directly within the OpenStack® networking solution.

OpenStack® uses virtual networking to connect various components (for example, compute and storage node).

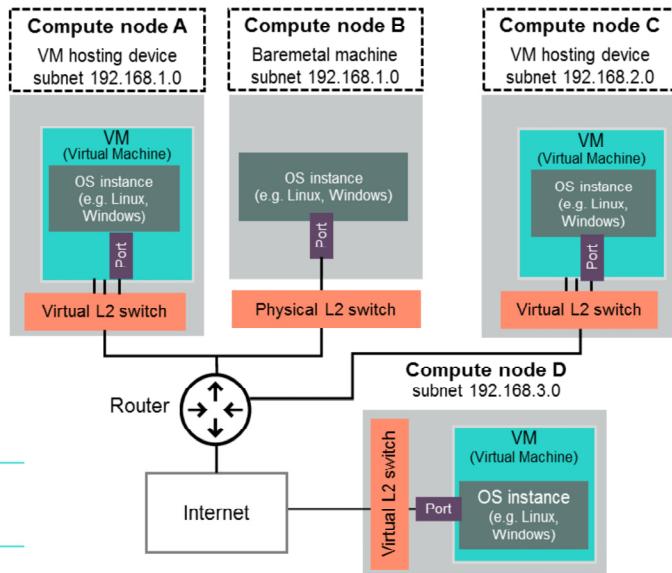
Virtual networking works similarly to standard networking, except that the networking functions are performed by software on compute systems instead of dedicated network devices.

OpenStack® does not reinvent networking; it simply provides the ability to use the existing open and vendor-specific virtual networking technology

Virtual Layer 2 switch

- Virtual Layer 2 (L2) switch:
 - Learns which VM or baremetal OS port MAC addresses are connected to each of its ports
 - Provides communication between VM nodes and/or baremetal nodes that exist on the same subnet
 - Forwards received packets through the port assigned to the MAC address of the packets destination address
 - Can be used to modify packets received from or going to an attached node in support of VLANs or tunnel transport

NOTE: In this example, a router is used to route packets between VMs and/or baremetal machines that exist on different subnets.



Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

5

Virtual Layer 2 switch

Another key component of virtualized networking is a virtual Layer 2 (L2) switch, which is configured from the VM host device's OS.

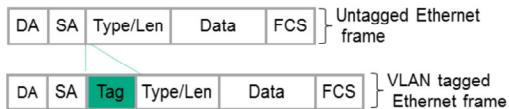
Virtual L2 switch:

- Operates much like its physical counterpart and learns which VM MAC addresses are connected to each of its ports, which enables it to forward packets received on one of its ports to the port associated with the destination MAC.
- Is used to interconnect all VMs on the same subnet, whether the VMs existing on the same VM host device, other VM host device, or through a physical switch to baremetal machines, as shown with compute nodes A and B in the screenshot.
- Can be used to modify packets received from or going to an attached node in support of VLANs or tunnel transport.

A virtual or physical router can be used to route packets between the subnets as shown between compute nodes on subnet 192.168.1.0 and the compute nodes on subnets 192.168.2.0, and 192.168.3.0 in the screenshot.

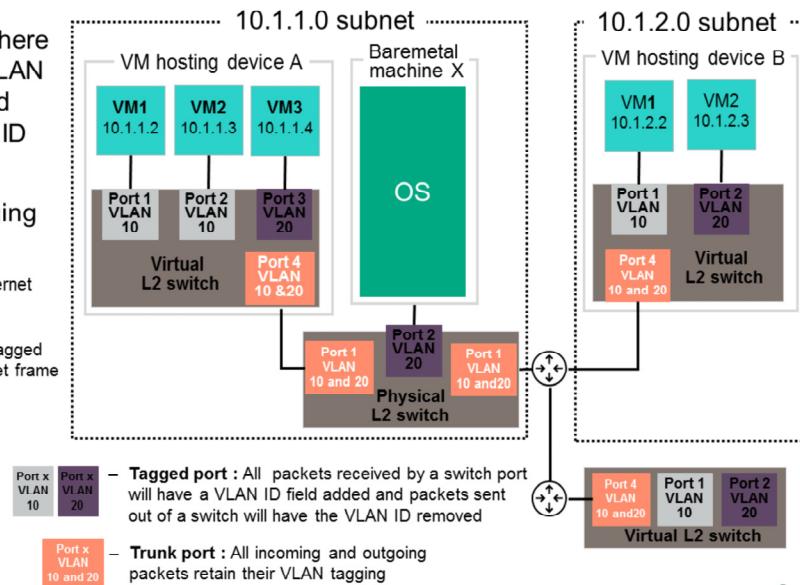
Virtual Local Area Network

- VLANs are used to define a domain where only packets belonging to a specific VLAN can communicate with other networked devices belonging to that same VLAN ID
- A VLAN field can be inserted into a Ethernet packet to identify it as belonging to a specific VLAN



- Up to 4096 VLANs are supported on network
- Very useful in isolating Client A data packets from Client B's data

Hewlett Packard
Enterprise



© Copyright 2017 Hewlett Packard Enterprise Development LP

6

Virtual Local Area Networks

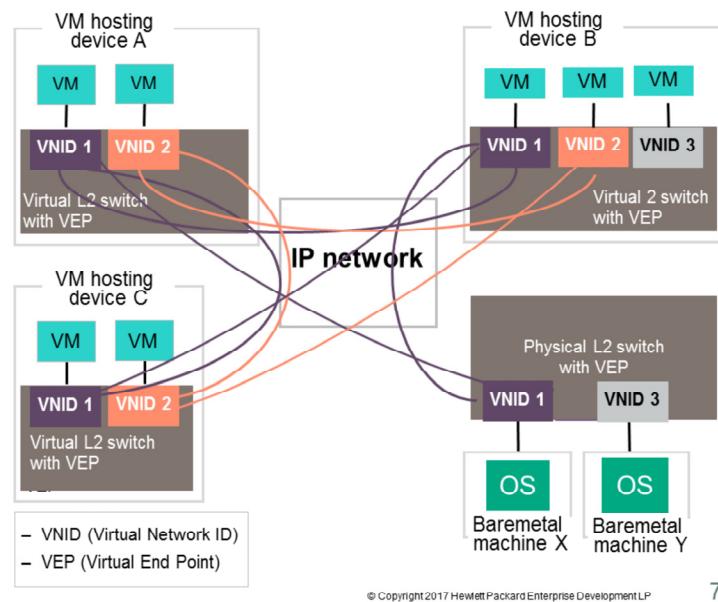
Virtual Local Area Networks (VLANs) are used to define a Layer 2 domain where only packets belonging to a specific VLAN can communicate with other networked devices belonging to that same VLAN ID. A VLAN field can be inserted into a Ethernet packet to identify it as belonging to a specific VLAN as shown in the diagram. Up to 4096 VLANs are supported on a network. VLANs are very useful in isolating Client A data packets from Client B's data.

In the diagram on the above slide:

- All traffic in VLAN ID 10 is isolated from traffic in VLAN ID 20, so VM3 in hosting device A can only communicate with the baremetal machine, VM2 in hosting device B.
- A customer assigned to VLAN 10 would only have access to networked devices that belong to VLAN 10.
- VMs on the same VM hosting device can belong to different VLANs, so VMs on the same hosting device can be isolated from one another using VLANs.

Virtual overlay networks

- Virtual overlay network:
 - Is a type of network virtualization that uses tunneling protocols to extend isolated network segments between servers for multi-tenant data center networks
 - Encapsulates one packet inside of another packet which is then forwarded to an endpoint where it is de-encapsulated
 - Virtual overlay networks supported by OpenStack® Neutron:
 - GRE (Generic Routing Encapsulation)
 - VxLAN (Virtual Extensible LAN)



Hewlett Packard
Enterprise

7

© Copyright 2017 Hewlett Packard Enterprise Development LP

Virtual overlay networks

Virtual overlay networks are another way to isolate data traffic between specific network nodes. Virtual overlay networks:

- Are a type of network virtualization that uses tunneling protocols to extend isolated network segments between servers for multi-tenant data center networks.
- Encapsulate a data packet inside of another packet which is then forwarded to an endpoint where it is de-encapsulated.

Virtual overlay networks supported by OpenStack® Neutron are:

- GRE (Generic Routing Encapsulation)
- VxLAN (Virtual Extensible LAN)

The benefits of virtual overlay networks include:

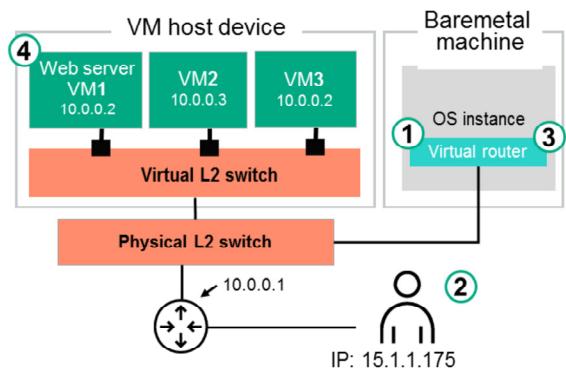
- **Traffic isolation for multi-tenancy:** Cloud service providers can use traffic isolation features to securely offer services to multiple customers; large enterprises can isolate business unit traffic or specific traffic types (for example, R&D network) from production traffic.
- **Ease of VM provisioning:** IT managers have the freedom to migrate VMs to new locations without worrying about attributes (or limitations) of the physical network. All the higher level elements of the network, including policy, security and VLANs, migrate with the VMs.
- **Scalability:** IT managers can scale their data center networks beyond the 4096 VLAN limit.
- **Physical network independence:** Overlay networks enable you to migrate to low-cost data center switches.
- **Simplified connection to OpenStack®**

In the diagram on the slide, there are three virtual hosts and two baremetal machines connected via an IP network. Each VM host contains a Virtual End Point (VEP), which is a virtual switch capable of acting as the encapsulation or de-encapsulation point for the virtual networks (VNIDs). Bare-metal machines are connected to the virtual overlay via a physical switch that supports the VEP functionality. Each host has two or more VNIDs operating on it and each workload assigned to a given VNID can communicate with other workloads in the same VNID, while maintaining separation from workloads in other VNIDs on the same host or other hosts. Depending on the encapsulation and configuration method chosen, hosts that do not contain a given VNID will either never see packets destined for that VNID, or will see the packets and drop them at ingress. This ensures the separation of tenant traffic.

With a physical switch capable of acting as the tunnel end-point, you can add both physical servers and appliances (firewalls, load balancers, and so on) to the overlay. This model is key to a cohesive deployment in mixed workload environments common in today's data centers.

Floating IP address

- Each VM or baremetal machine instance, has a private, fixed IP address and can also have public, or floating IP address
 - **Private IP addresses** are used for communication between instances
 - **Public IP addresses** are used for communication with networks outside the cloud, including the Internet
- A range of public IP addresses are reserved by the administrator and placed in a Floating IP address pool
- A Floating IP can be associated with an instance's private IP address
- An instance with an associated Floating IP address can be accessed from the public network



1. Administrator assigns an address from its floating IP address pool to the web server, which is 15.1.1.200 in this example
2. Person at 15.1.1.175 has been provided the web server address of 15.1.1.200 and inputs it into their web browser
3. The request is communicated to the virtual router which uses a NAT table (iptable) to determine the private IP address associated with 15.1.1.200
4. The virtual router sends the request to the web server

© Copyright 2017 Hewlett Packard Enterprise Development LP

9

Floating IP address

Most virtual networks support two types of IP addresses to be assigned: private and public.

Private IP addresses are assigned automatically by an internal DHCP service on instance boot. Instances in the same broadcast domain communicate using the private IP addresses by using the L2 networking service.

A private IP address is used for communication with an external network, but it cannot be used for inbound connections. This setup is similar to the setup of a computer behind NAT or firewall in the physical world; the computer has the private IP and can communicate with external networks, but outside clients cannot initiate a connection with the computer behind the firewall.

To make an instance visible from the data center network, the administrator has to assign a Floating IP address from the predefined pool of Floating IPs. When assigned, the public IP address remains connected to the instance for as long as the instance exists. When you terminate the instance, the Floating IP address is released and returned to the pool of available IP addresses.

The delivery of packets to the interface with an assigned floating address is the responsibility of the virtual router.

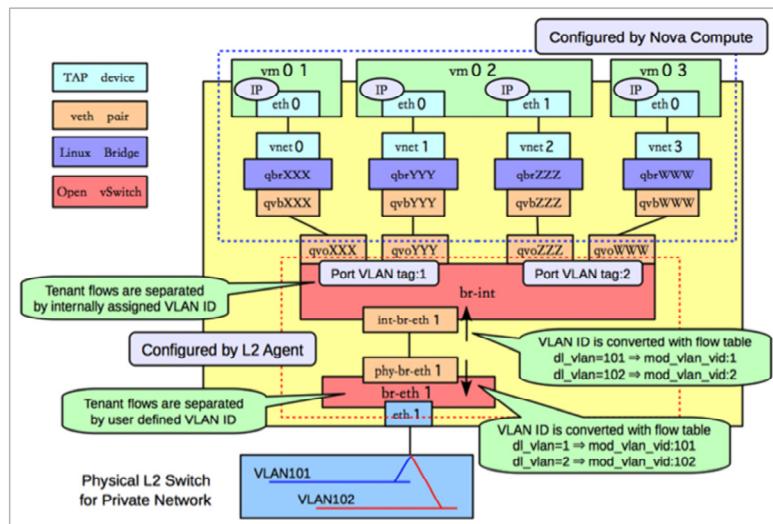
Note: The above example demonstrates a bare-metal machine, but the virtual router can exist on a VM host or a network node as well.

Open Virtual Switch (OVS) components

Compute host

- Virtual networking devices
 - TAP devices
 - veth pairs
 - Linux bridges
 - Open vSwitch bridges

NOTE: The diagram shows the compute host configuration with the two networks and several instances using them.



Open Virtual Switch (OVS) components—Compute host

Open vSwitch consists of bridges and ports. Ports represent connections to other things, such as physical interfaces and patch cables. Packets from any given port on a bridge are shared with all other ports on that bridge. Bridges can be connected through Open vSwitch virtual patch cables or through Linux virtual Ethernet cables (veth). Additionally, bridges look like network interfaces to Linux, so they can be assigned IP addresses.

Open vSwitch uses several main bridges. The integration bridge, called “br-int,” connects directly to the VMs and associated services. The external bridge, called “br-ex,” connects to the external network. The VLAN configuration of the Open vSwitch plug-in uses bridges associated with each physical network.

In addition to defining bridges, Open vSwitch uses OpenFlow, which enables you to define networking flow rules. These rules are used in certain configurations to transfer packets between VLANs.

The most common security group driver used with Open vSwitch is the Hybrid iptables/Open vSwitch plug-in. It uses a combination for iptables and OpenFlow rules. iptables is a tool used for creating firewalls and setting up NATs on Linux. It uses a complex rule system and “chains” of rules to allow for the complex rules required by the Neutron security groups.

In the example, four types of virtual networking devices are shown: test access point (TAP) devices, veth pairs, Linux bridges, and Open vSwitch bridges. For an Ethernet frame to travel from eth0 of virtual machine vm01, to the physical network, it must pass through nine devices in the host: TAP vnet0 → Linux bridge qbrXXX → veth pair (qvBXXX, qvoXXX) → Open vSwitch bridge br-int → veth pair (int-br-eth1, phy-br-eth1) → physical network interface card eth1.

Hypervisors such as KVM and Xen implement a virtual network interface card (typically called a VIF or vNIC) through a **TAP device** such as vnet0. An Ethernet frame sent to a TAP device is received by the guest operating system.

A veth pair is a pair of directly connected virtual network interfaces. An Ethernet frame sent to one end of a veth pair is received by the other end of a veth pair. OpenStack® networking uses veth pairs as virtual patch cables to make connections between virtual bridges.

A Linux bridge behaves like a hub where you can connect multiple (physical or virtual) network interface devices to a Linux bridge. Any Ethernet frame that comes in from one interface attached to the bridge is transmitted to all other devices.

An Open vSwitch bridge behaves like a virtual switch. The network interface devices connect to ports on the Open vSwitch bridge, and these ports can be configured similarly to the ports on a physical switch, including VLAN configurations.

The br-int Open vSwitch bridge is **the integration bridge**. All guests running on the compute host connect to this bridge. OpenStack® Networking implements isolation across these guests by configuring the br-int ports.

The **br-eth1 bridge** provides connectivity to the physical NIC, eth1. It connects to the integration bridge through a veth pair: int-br-eth1 and phy-br-eth1.

In the **VLAN translation** example, net01 and net02 have VLAN IDs of 1 and 2, respectively. However, the physical network in the example only supports VLAN IDs in the range of 101 through 110. The Open vSwitch agent configures flow rules on br-int and br-eth1 to enable the VLAN translation. When br-eth1 receives a frame marked with VLAN ID 1 on the port associated with phy-br-eth1, it modifies the VLAN ID in the frame to 101. Similarly, when br-int receives a frame marked with VLAN ID 101 on the port associated with int-br-eth1, it modifies the VLAN ID in the frame to 1.

Security groups: iptables and Linux bridges

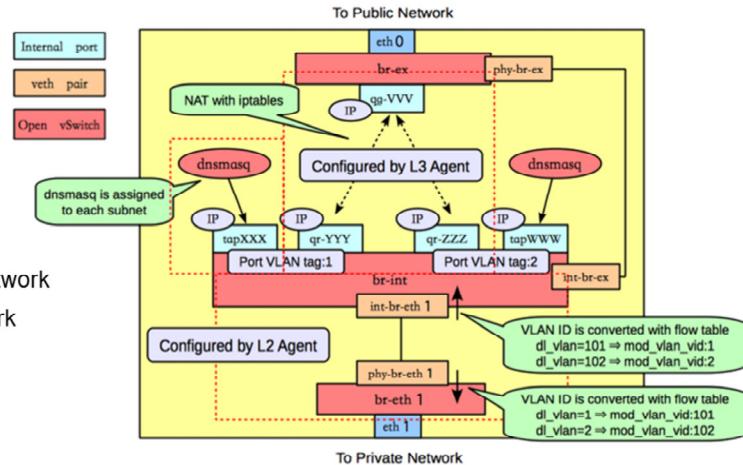
Ideally, the TAP device vnet0 would be connected directly to the integration bridge—br-int. Unfortunately, this is not possible because of how OpenStack security groups are currently implemented. OpenStack uses iptables rules on the TAP devices, such as vnet0 to implement security groups, and Open vSwitch is not compatible with the iptables rules that are applied directly on TAP devices that are connected to an Open vSwitch port.

The OpenStack Networking service uses an extra Linux bridge and a veth pair as a workaround for this issue. Instead of connecting vnet0 to an Open vSwitch bridge, it is connected to a Linux bridge, qbrXXX. This bridge is connected to the integration bridge, br-int, by means of the veth pair qvbXXX and qvoXXX.

Open Virtual Switch (OVS) components

Network host configuration

- The network host runs these services:
 - neutron-openvswitch-plugin-agent
 - neutron-dhcp-agent
 - neutron-I3-agent
 - neutron-metadata-agent
- Network connections
 - eth0 is connected to the external, public network
 - eth1 is connected to the private data network



Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

13

Open Virtual Switch (OVS) components—Network host configuration

As on the compute host, there is an Open vSwitch integration bridge (**br-int**) and an Open vSwitch bridge connected to the data network (**br-eth1**), and the two are connected by a veth pair. The **neutron-openvswitch-plugin-agent** configures the ports on both switches to do VLAN translation.

An additional Open vSwitch bridge, **br-ex**, connects to the physical interface that is connected to the external network. In this example, that physical interface is **eth0**. Although the integration bridge and the external bridge are connected by a veth pair (**int-br-ex**, **phy-br-ex**), this example uses Layer 3 connectivity to route packets from the internal networks to the public network; no packets traverse that veth pair in this example.

Open vSwitch internal ports

The network host uses Open vSwitch internal ports. Internal ports enable you to assign one or more IP addresses to an Open vSwitch bridge. In the example on the picture above, the **br-int** bridge had four internal ports: **tapXXX**, **qr-YYY**, **qr-ZZZ**, and **tapWWW**. Each internal port had a separate IP address associated with it. An internal port, **qg-VVV**, was on the **br-ex** bridge.

DHCP agent

By default, the OpenStack® Networking DHCP agent uses a program called **dnsmasq** to provide DHCP services to guests. The OpenStack® Networking service must create an internal port for each network that requires DHCP services and attach a **dnsmasq** process to that port. In the previous example, the interface **tapXXX** was on **net01_subnet01**, and the interface **tapWWW** was on **net02_subnet01**.

L3 agent (routing)

The OpenStack® Networking L3 agent implements routing through the use of Open vSwitch internal ports and relies on the network host to route the packets across the interfaces. In this example:

- Interface qr-YYY, which is on subnet net01_subnet01, has an IP address of 192.168.101.1/24.
- Interface qr-ZZZ, which is on subnet net02_subnet01, has an IP address of 192.168.102.1/24.
- Interface qg-VVV, which has an IP address of 10.64.201.254/24.

Because all of these interfaces are visible to the network host operating system, they will route the packets appropriately across the interfaces, as long as an administrator has enabled IP forwarding.

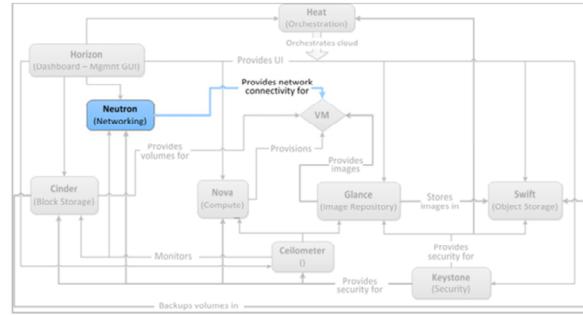
The L3 agent uses iptables to implement Floating IP addresses to do the network address translation.



Neutron overview

Neutron functionality

- Neutron:
 - Is an enterprise-class, networking-as-a-service OpenStack® component
 - Provides an API to define network components to be used by Nova compute instances
 - Uses SDN (Software Defined Networking)
 - Abstracts the physical network implementations through the use of industry standard and vendor-specific network resource plug-ins
 - Enables projects to have multiple private networks, and allows them to choose their own IP addressing scheme and routing topology
 - Supports virtual network services including services such as firewalls, load balancers, and VPNs (Virtual Private Networks)



Neutron functionality

Neutron is the OpenStack® service that is used to interconnect the OpenStack® components within an Infrastructure as a Service (IaaS) environment, as well as the IaaS environment to the public network.

Neutron:

- Provides an API to define networks, subnets, and network connectivity used by other OpenStack services, in particular Nova compute instances
- Uses SDN (Software Defined Networking), a fundamental shift in how data center networks are defined, provisioned, and consumed
- Abstracts the physical network implementations through the use of industry standard and vendor-specific network resource plug-ins
- Enables projects to have multiple private networks, and allows them to choose their own IP addressing scheme and routing topology
- Supports virtual network services, including services such as firewalls, load balancers, and VPNs (Virtual Private Networks)

Neutron API version states as of the Newton release

- Networking API v2 (**current**)
- Changes in the Newton release:
 - DHCP and L3 agent scheduling is availability zone-aware
 - L3 agent is used by default for external networks
 - Pluggable IPAM (IP address management) implementation is used by default
 - Support for VLAN-aware VMs to use a single trunk parent port to connect to multiple Neutron logical networks
 - Improved network performance



© Copyright 2017 Hewlett Packard Enterprise Development LP

17

Neutron API version states as of the Newton release

Currently, Neutron supports the API v2 in Newton release. Some of the changes in this release include:

- Availability zone-aware DHCP and L3 agent scheduling
- The default value for `external_network_bridge` in the L3 agent
- Pluggable IPAM (IP address management) implementation is used by default
- Support for VLAN-aware VMs to use a single trunk parent port to connect to multiple Neutron logical networks
- Improved network performance

Neutron API 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 their associated configurations
 - Serves as a address pool from which OpenStack® can assign IP addresses to VMs
 - Must be associated with a network
- **Port**:
 - A connection point for attaching a single device, such as the NIC of a virtual server, to a virtual network
 - Describes the associated network configuration, such as the MAC and IP addresses to be used on that port
 - On creation of a port, an IP address is assigned from the subnet IP address pool
 - OpenStack® can also define the MAC address of a port
- **Router**—Used to interconnect two or more subnets



© Copyright 2017 Hewlett Packard Enterprise Development LP

18

Neutron API network resources

The OpenStack® Networking API has a virtual network, a subnet, and port abstractions to describe network resources:

- **A network**—An isolated L2 segment, analogous to VLAN in the physical networking world.
- **A subnet**—A block of v4 or v6 IP addresses and associated configuration state.
- **A 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. Compute instances are attached to ports.
- **A router**—Used to interconnect two or more subnets

Cloud administrators can expose additional API capabilities through API extensions.

Neutron networking terminology

- **Compute node**—An OpenStack® server that runs user workloads in VMs, baremetal machine, or containers
- **Layer2 Interconnect**—A Bridge/Layer 2 switch used to provide Layer 1 networking connections between end points, possibly filtering at Layer2
- **Network node**—An OpenStack® infrastructure server that connects project networks to infrastructure networks
- **OSI Layer 2**—The layer just above physical connections (Layer 1) that manages traffic between servers and provides a logical separation of traffic
- **Project (formerly tenant)**—The owner of compute instances and their networks that provides logical isolation (many project models require network traffic from one project to be invisible to others)
- **Provider network**—A term used to refer to the cloud-provided external network tenants use to reach the outside
- **Project network**—A term used to refer to networks created and internally used by a project
- **VLAN (Virtual Local Area Network)**—Switch-enforced isolation zones created by adding 1 of 4096 tags in the network traffic (also known as *tagged traffic*)
- **VXLAN (Virtual Extensible LAN)**—A network virtualization technology that uses VLAN-like encapsulation of Layer 2 Ethernet frames within Layer 4 UDP packets (UDP port 4789 IANA-assigned destination number)



© Copyright 2017 Hewlett Packard Enterprise Development LP

19

Neutron networking terminology

Neutron uses the following terminology for networking components:

- **Compute node**—An OpenStack® server that runs user workloads in VMs, bare metal machine, or containers
- **Layer2 interconnect**—Bridge/Layer 2 switch used to provide layer 1 networking connections between end points, possibly filtering at layer2
- **Network node**—An OpenStack® infrastructure server that connects project networks to infrastructure networks
- **OSI Layer 2**—The layer just above physical connections (Layer 1) that manages traffic between servers and provides a logical separation of traffic
- **Project (formerly tenant)**—The owner of compute instances and their networks providing logical isolation (many project models require network traffic from one project to be invisible to others)
- **Provider network**—A term used to refer to the cloud-provided external network that tenants use to reach the outside
- **Project network**—A term used to refer to networks created and internally used by a project
- **VLAN (Virtual Local Area Network)**—Switch-enforced isolation zones created by adding 1 of 4096 tags in the network traffic (also known as *tagged traffic*)
- **VXLAN (Virtual Extensible LAN)**—A network virtualization technology that uses VLAN-like encapsulation of layer 2 Ethernet frames within layer 4 UDP packets (UDP port 4789 IANA-assigned destination number)



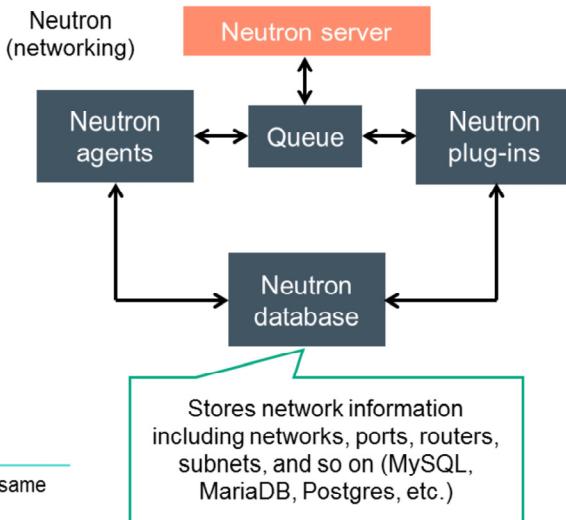
Neutron architecture

Neutron components

Neutron server

- **Neutron server** is a Python daemon that:
 - Exposes the OpenStack® Networking API
 - Passes user requests to the configured OpenStack® Networking plug-in for additional processing
- **Neutron plug-ins:**
 - Are pluggable back-end implementations of the OpenStack® Networking API
 - Use a variety of technologies to implement the logical API requests
 - Typically require access to a database for persistent storage, similarly to other OpenStack® services

NOTE: The Neutron server can be deployed on the same server as the controller host; however, it is entirely standalone and can be deployed on its own server as well.



Neutron components—Neutron server

Like other OpenStack® services, Neutron Networking provides cloud administrators with significant flexibility in deciding which individual services should run on which physical devices. In a test environment, all service daemons can be run on a single physical host for evaluation purposes. In a highly scaled, high availability environment, each service can be hosted on a different physical server, and in some cases be replicated across multiple servers to provide redundancy.

The Neutron server API receives user requests and passes them to the configured Networking plug-in for additional processing. Typically, the plug-in requires access to a database for persistent storage (also similar to other OpenStack® services).

If your deployment uses a controller host to run centralized compute components, you can deploy the Networking server on that same host. However, Networking is a stand-alone service, like the other OpenStack® services such as Compute, Image, Identity, or Dashboard, so you can deploy the Networking service on its own host as well.

Neutron components

Neutron plug-ins

- Provide the flexibility to support a wide range of network vendor products and OS-specific functions

Example OpenStack® Neutron plug-ins

- Modular Layer 2
- Arista L3 Routing
- Big Switch Controller
- Brocade
- Cisco UCS/Nexus
- Embrane Heleos
- HPE internal plug-in
- IBM SDN-VE

- MidoNet
- NEC OpenFlow
- Nuage
- OneConvergence NVSD
- PLUMgrid
- Ryu OpenFlow Controller
- SR-IOV
- VMware NSX

- Provide advanced network capabilities, such as:
 - Automatic scaling to cloud proportions and automatic configurations
 - Traffic isolation through systems such as Linux VLANs and IP tables
 - Load balancing, NAT, firewalling, dynamic configuration, and so on
 - Open support for a variety of technologies at various network layers

NOTE: There are many more Neutron plug-ins. The full list is available at
https://wiki.openstack.org/wiki/Neutron_Plugins_and_Drivers#Plugin_and_Driver_Processes

Neutron components—Neutron plug-ins

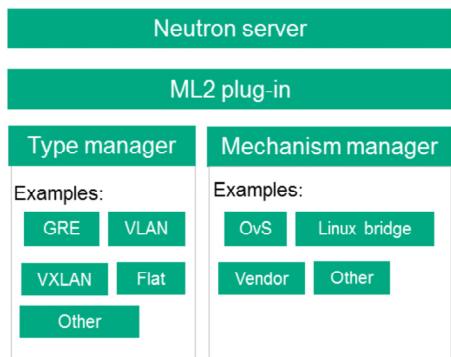
Neutron lets you use a set of different backends called "plug-ins" that work with a growing variety of networking technologies. Some OpenStack® Networking plugins can use basic Linux VLANs and iptables, while others can use more advanced technologies, such as L2-in-L3 tunneling or OpenFlow, to provide similar benefits.

These plugins may be distributed as part of the main Neutron release, or separately.

Plug-ins can have different properties for hardware requirements, features, performance, scale, or operator tools. Because Networking supports a large number of plug-ins, the cloud administrator can weigh options to decide on the right networking technology for the deployment.

Neutron components

Neutron ML2 plug-in



NOTE: More information on ML2 plug-in is available at <https://wiki.openstack.org/wiki/Neutron/ML2>.

- The Open vSwitch and Linux bridge plug-ins are deprecated and replaced with the ML2 (Modular Layer 2) plug-in
- Existing L2 agents work with the ML2 plugin and continue to work with the deprecated Open vSwitch and Linux bridge plug-ins
- The ML2 plug-in:
 - Supports multiple Layer 2 technologies concurrently
 - Organizes Layer 2 functionality into:
 - Type drivers used to support multiple networking technologies
 - Mechanism drivers used to facilitate the access to the networking configuration
- One type driver may support several mechanism drivers
- Long term goal is to transition all vendor-specific plug-ins to the ML2 type and mechanism drivers

Neutron components—Neutron ML2 plug-in

The Open vSwitch and Linux bridge plug-ins have been deprecated and replaced with the ML2 (Modular Layer 2) plug-in. However, existing L2 agents work with the ML2 plug-in and continue to work with the deprecated Open vSwitch and Linux bridge plug-ins.

The ML2 framework simplifies the addition of support for new L2 technologies and reduces the effort that is required to add and maintain them, as compared to monolithic L2 plug-ins.

The ML2 plugin:

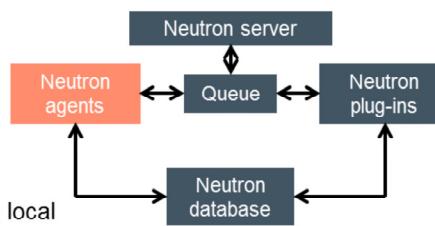
- Supports multiple layer 2 technologies concurrently
- Organizes layer 2 functionality into:
 - **Type drivers** used to support multiple networking technologies
 - **Mechanism drivers** used to facilitate the access to the networking configuration

One type driver can support several mechanism drivers. In most cases, networking vendors need to implement only the mechanism driver for their technology. A long-term goal is to transition all vendor-specific plug-ins to the ML2 type and mechanism drivers.

Neutron components

Neutron agents

- Interact with the main Neutron-server process through:
 - RPC (e.g. RabbitMQ or Qpid)
 - Standard OpenStack® Networking API
- May or may not be required, depending on deployment:
 - Plug-in agent (`neutron-*Agent`) runs on each hypervisor to perform local vSwitch configuration (plug-ins do not require an agent)
 - DHCP agent (`neutron-dhcp-agent`) provides DHCP services to tenant networks and is the same across all plug-ins
 - L3 agent (`neutron-l3-agent`) provides L3/NAT forwarding to provide external network access for VMs on project networks, and is the same across all plug-ins
 - L2 plug-in agent (`neutron-*Agent`, e.g. `neutron-openvswitch-agent`) runs on each compute node to perform local network configuration
 - Metering and others



Neutron components—Neutron agents

Neutron agents interact with the main Neutron-server process through:

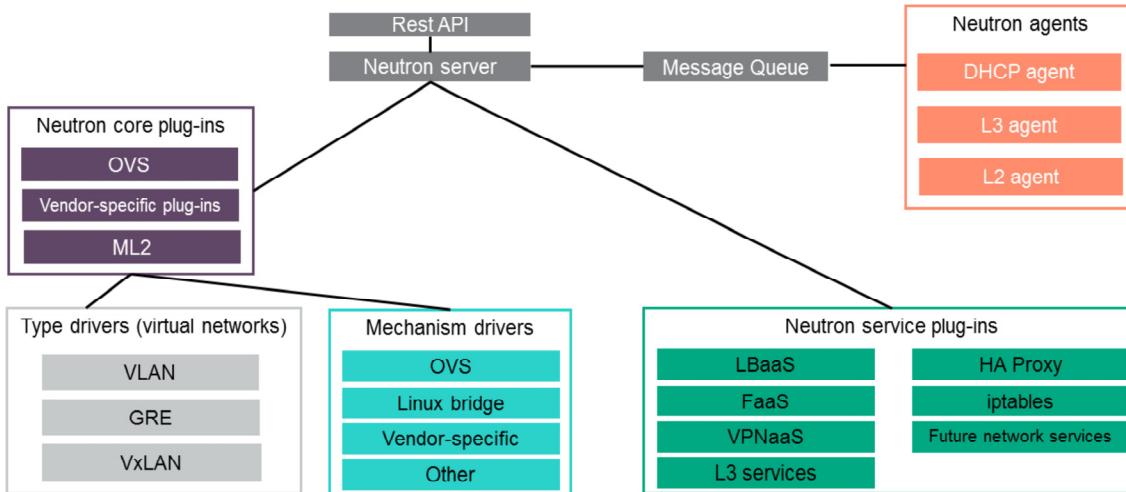
- RPC (e.g. RabbitMQ or Qpid)
- Standard OpenStack® Networking API

Depending on deployment, Neutron agents may or may not be required. The following Neutron agents may or may not be used:

- Plug-in agent (`neutron-*Agent`), which runs on each hypervisor to perform local vSwitch configuration (plug-ins do not require an agent).
- DHCP agent (`neutron-dhcp-agent`), which provides DHCP services to tenant networks and is the same across all plugins.
- L3 agent (`neutron-l3-agent`), which provides L3/NAT forwarding to provide external network access for VMs on project networks, and is the same across all plug-ins.
- L2 plug-in agent (`neutron-*Agent`, for example, `neutron-openvswitch-agent`) which runs on each compute node to perform local network configuration
- Metering
- Others

Neutron components

A composite view of Neutron components



Neutron components—A composite view of Neutron components

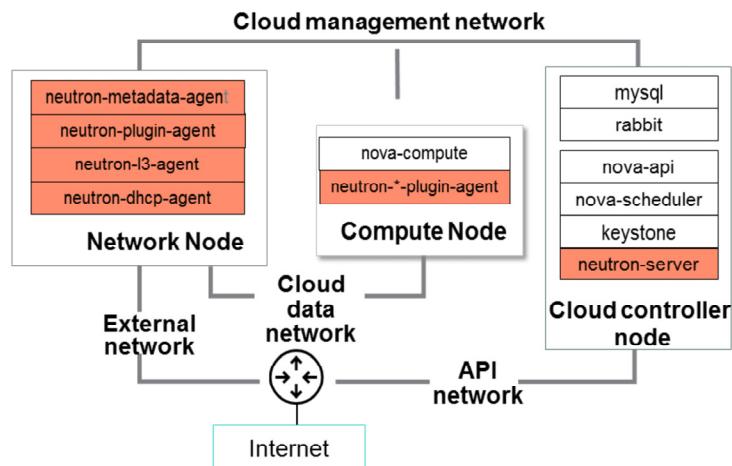
This diagram on the slide above is designed to help you consolidate what you have learned from the previous 4 slides.

The Neutron server can communicate with a wide range of flexible network devices and topologies through the use of plug-ins:

- Neutron core plug-ins provide the ability to support a number of LAN types (for example, GRE, VxLAN) and mechanisms (for example, OvS, vendor-specific)
- Neutron service plug-ins provide support for network services (for example, LBaaS, FWaaS)
- Network agents support network functions (for example, DHCP, L3)

OpenStack® nodes with Neutron components

- Cloud controller node runs the Neutron server, as well as other OpenStack® API components
- Network node includes:
 - neutron-*-plugin-agent
 - neutron-dhcp-agent—Allocates IP addresses to the VMs on the network
 - neutron-l3-agent—L3 forwarding and NAT, enabling the VMs to access and be accessed from the public network
 - neutron-metadata-agent—Allows Compute API metadata to be reachable by VMs on tenant
- Compute node runs nova-compute node and the neutron plugin-agent



Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

26

OpenStack® nodes with Neutron components

The diagram on the slide above shows a basic multi-node deployment. The cloud controller node is the main OpenStack® management node, the network node provides and manages network services within the cloud, and the compute nodes that host the deployed VMs have a network agent running on them.

The cloud controller node runs the Neutron server, as well as other OpenStack® API components.

The network node runs the plug-in agent, the DHCP agent, and the L3 agent. It has access to the public network (not all plug-ins have an agent on the compute node). The DHCP agent allocates IP addresses to the VMs on the network. The L3 agent performs NAT and enables the VMs to access the public network. The node must have three network interfaces:

- The first interface is used to communicate with the controller node and computer nodes through the management network.
- The second interface is used for the VM traffic on the data network.
- The third interface is used to connect to the external gateway on the network.

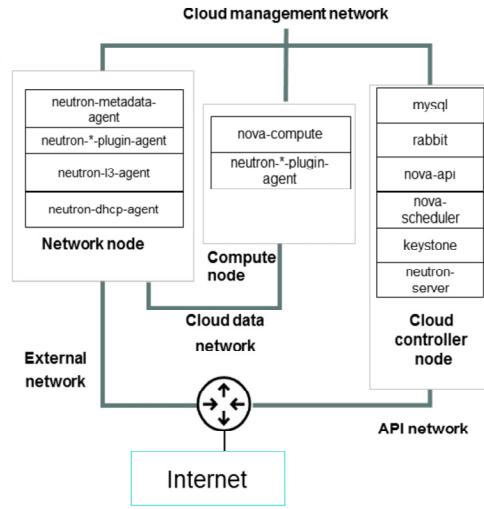
In Open vSwitch, the interface is bridged to the Open vSwitch bridge "br-ex" interface. Although Open vSwitch, which uses the ML2 mechanism driver, is the most popular implementation, many others are supported.

The compute node runs Nova compute and the Neutron plug-in agent. It does not have access to the public network. Also, compute node must have at least two network interfaces:

- The first interface is used to communicate with the controller node through the management network.
- The second interface is used for the VM traffic through the data network; the VM is able to receive its IP address from the DHCP agent on this network.

OpenStack® node networks

- Cloud management network
 - Used for internal communication between OpenStack® components
 - The IP addresses on this network should be reachable only within the data center
- Cloud data network
 - Used for VM data communication within the cloud deployment
 - The IP addressing requirements of this network depend on the OpenStack® Networking plug-in in use
- External network
 - Provides VMs with Internet access in some deployment scenarios
 - The IP addresses on this network should be reachable by anyone on the Internet
- API access network
 - Exposes all OpenStack® APIs, including the OpenStack® Networking API, to tenants
 - The IP addresses on this network should be reachable by anyone on the Internet



Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

27

OpenStack® node networks

A standard OpenStack® Networking setup has at least four distinct physical data center networks (there can be others).

Cloud management network is used for internal communication between OpenStack® components. The IP addresses on this network should be reachable only within the data center.

Cloud data network is used for VM data communication within the cloud deployment. The IP addressing requirements of this network depend on the OpenStack® Networking plug-in in use.

External network provides VMs with Internet access in some deployment scenarios. The IP addresses on this network should be reachable by anyone on the Internet.

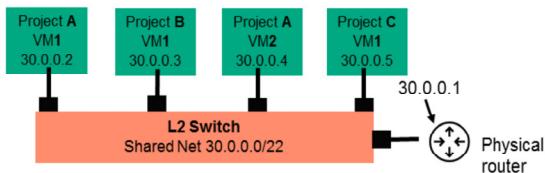
API access network exposes all OpenStack® APIs (including the OpenStack® Networking API) to tenants. The IP addresses on this network should be reachable by anyone on the Internet.

Network topologies

Single flat and private network

Single flat network

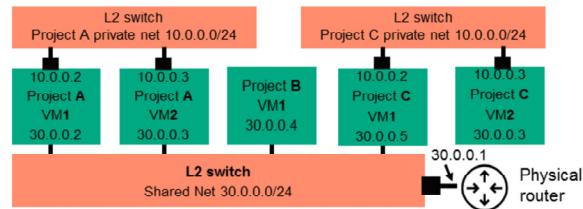
- Simplest network topology
- Single flat network shared by and visible to all projects
- All VMs/baremetal machines are on the same subnet
- A single NIC per VM with a fixed IP address
- The router connected to the L2 switch port can be used to connect all of the devices on that subnet to an external network



Hewlett Packard
Enterprise

Single flat network with private networks

- Projects can share a flat network and have private, non-shared networks
- VMs and baremetal machine need two ports (NICs) to connect to both the private and flat networks
- The router connected to the L2 switch port can be used to connect all of the devices on that subnet to an external network, so the project C VM2 would not have access to the external network



© Copyright 2017 Hewlett Packard Enterprise Development LP

28

Network topologies—Single flat and private network

Flat or flat DHCP network

All VMs are placed on a single network, using the same subnet and bridge, created earlier by the administrator. All VMs share the same network, hence the visual phrasing—"flat."

The Flat and Flat DHCP network types are very much alike except for the allocation of IP addresses to VMs is done through DHCP, thus the name "flat DHCP network."

There is no network separation between the VMs of different projects, meaning their VMs can be inter-connected, unless limited by other means. In the single flat network example, project A's VM1 can communicate with project B's VM1 and project C's VM1, which is not very secure.

Mixed flat and private networks

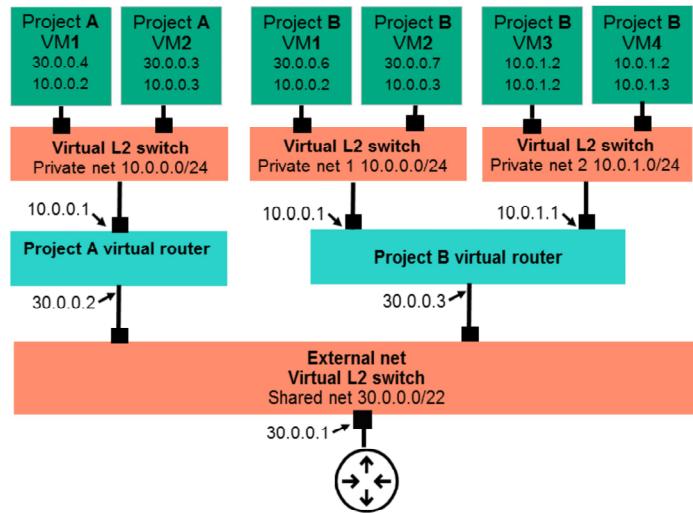
This use case is an extension of the flat network use cases, in which projects optionally have access to private, per-project networks. In addition to seeing one or more shared networks via the OpenStack® Networking API, project administrators can create additional networks that are only visible to users of that project. When creating VMs, those VMs can have NICs on any of the shared networks and any of the private networks belonging to the project.

In the example for this type of network, project A's VM1 can communicate with project B's VM1 and project C's VM1, but it cannot communicate with project C's VM2.

Neutron topology

Using routers to separate project networks

- Project-dedicated routers:
 - Provide a significant amount of network design flexibility
 - Enable project-defined multi-tiered applications, with each tier being a separate network behind the router
 - Support use of overlapping project subnets without conflicts because of the access to the external networks
- If a project is provided access to the Neutron API, it can create its own networks by means of the Neutron API



 Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

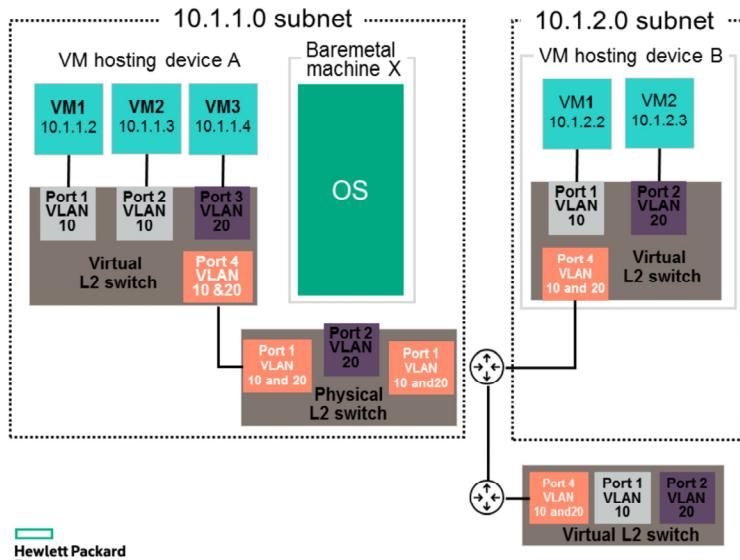
29

Neutron topology—Using routers to separate project networks

A more advanced router scenario is one in which each project gets at least one router and potentially has access to the OpenStack Networking API to create additional routers. The projects can create their own networks, potentially uplinking those networks to a router. This model enables project-defined multi-tiered applications, with each tier being a separate network behind the router. With multiple routers, project subnets can overlap without conflicting because access to the external networks happens via SNAT or Floating IP addresses. Each router uplink and Floating IP address is allocated from the external network subnet.

Neutron Virtual Networks

VLANs



- Operate at OSI Layer 2:

- Add a 4-byte VLAN tag to the Ethernet header with a value that ranges between 1 and 4094 (the 0 tag is special, and the 4095 tag, made of all ones, is equivalent to an untagged packet)
- VLAN-supported NICs, switches, and routers; similarly to OVS (Open vSwitch), it can interpret the VLAN tags
- Packets tagged for one VLAN are only shared with other devices configured to be on that VLAN, despite the fact that all of the devices are on the same physical network

Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

30

Neutron Virtual Networks—VLANs

As discussed in the virtual networking concepts section of this module, VLANs (Virtual LAN) operate at OSI Layer 2:

- Add a 4-byte VLAN tag to the Ethernet header with a value that ranges between 1 and 4094 (the 0 tag is special, and the 4095 tag, made of all ones, is equivalent to an untagged packet).
- VLAN-supported NICs, switches, and routers; similarly to OVS (Open vSwitch), it can interpret the VLAN tags.
- Packets tagged for one VLAN will only be shared with other devices configured to be on that VLAN, despite the fact that all of the devices are on the same physical network.

Neutron Virtual Networks (2 of 2)

Supported types of overlay networks

- GRE (Generic Routing Encapsulation) operates at OSI Layer 3:
 - Wraps IP packets with different routing information
 - When the new packet reaches its destination, it is unwrapped and the underlying packet is routed
- VxLANs (Virtual Extensible LAN) operates at OSI Layer 3:
 - An encapsulation protocol for running an overlay network on an existing Layer 3 infrastructure
 - Used to extend the number of virtual networks from the 4096 available for VLANs to a 24-bit segment ID which provides up to 16 million network IDs
 - As with VLANs, the VXLAN segment ID in each frame differentiates individual logical networks, so millions of isolated Layer 2 VXLAN networks can co-exist on a common Layer 3 infrastructure
 - As with VLANs, only virtual machines within the same logical network can communicate with each other



© Copyright 2017 Hewlett Packard Enterprise Development LP

31

Neutron Virtual Networks—Supported types of overlay networks

As discussed in the virtual networking concepts section of the module, overlay networks are another way to isolate data traffic between specific network nodes. They use tunneling protocols to extend isolated network segments between servers for multi-tenant data center networks.

As mentioned, Neutron supports two types of overlay networks:

- GRE (Generic Routing Encapsulation)
- VxLANs (Virtual Extensible LAN)

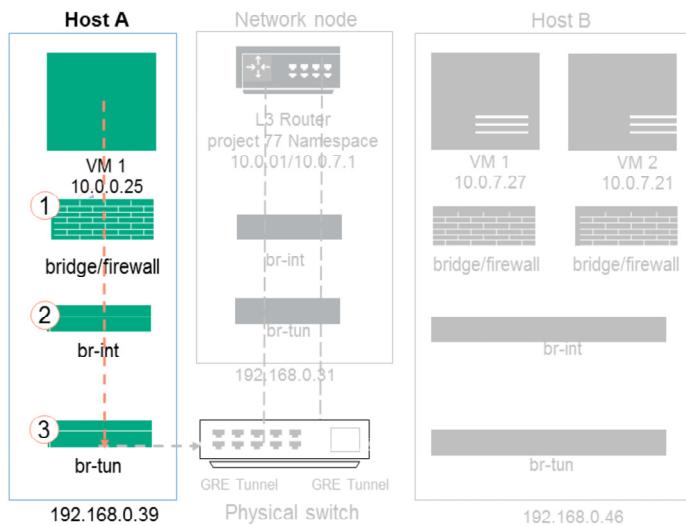
GRE network operates at OSI Layer 3. It wraps IP packets with different routing information. When the new packet reaches its destination, it is unwrapped, and the underlying packet is routed.

VxLAN network operates at OSI Layer 3. There is an encapsulation protocol for running an overlay network on an existing Layer 3 infrastructure. This network is used to extend the number of virtual networks from the 4096 available for VLANs to a 24-bit segment ID which provides up to 16 million network IDs. As with VLANs, the VXLAN segment ID in each frame differentiates individual logical networks, so millions of isolated Layer 2 VXLAN networks can co-exist on a common Layer 3 infrastructure. As with VLANs, only virtual machines within the same logical network can communicate with each other.

Example Neutron GRE overlay network (1 of 2)

- As depicted, VM 1 of host A sends traffic to VM1 of host B on 10.0.7.0/24 via router 10.0.0.1

- Traffic from VM 1 (10.0.0.25) first passes through the instance firewall (implemented in Neutron virtually)
- Traffic then passes through the compute host integration bridge (br-int), which is a virtual switch
- Traffic passes to br-tun, another virtual switch, where it is encapsulated for GRE tunneling on the physical network (192.168.0.0/24)



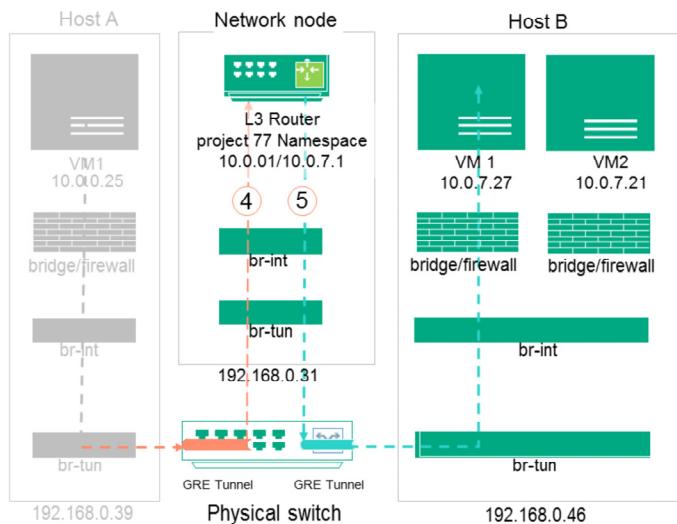
Example Neutron GRE overlay network (1 of 2)

This example flow demonstrates the flexibility that Neutron network nodes and the ML2 tunnel type bring to OpenStack Networking. Because firewalls, networks, and routers are all virtual, a single physical data network can manage the traffic from a dynamic set of cloud projects. This configuration also makes it possible to enable Cloud users to create their own isolated networks without causing networking problems for others.

Neutron networks are always associated with a project. By using network namespaces, Neutron can allow projects A and B to both create a network using subnet 10.0.0.0/24.

Example Neutron GRE overlay network (2 of 2)

4. The traffic reaches the network node where it is de-encapsulated at the br-tun, and passed to the integration bridge (br-int), after which it arrives at the router
5. The router forwards the traffic to 10.0.7.27 whereupon it flows through the 10.0.7.0/24 GRE tunnel to instance B



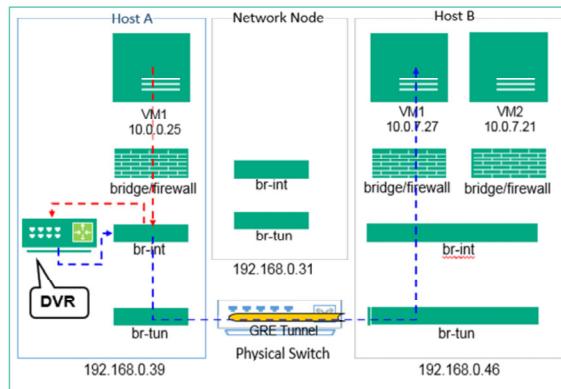
Example Neutron GRE overlay network (2 of 2)

Virtual routers can route traffic between any two Neutron networks.

Note: In the model above, any two instances on the same network can communicate directly through their network's GRE tunnel without passing traffic through the network node. However, in this model, traffic relying on routing must pass through the network node's virtual router. This creates a potential bottleneck and should the router (host) fail, all routing will stop, isolating the attached networks.

Neutron DVR (Distributed Virtual Router)

- The DVR adds a wealth of critical new functionality to Neutron that combines to eliminate bottlenecks and single points of failure and make Neutron networking extremely flexible:
 - Removes the potential for the network node to become a routing bottleneck
 - Supports east-west (inter instance) traffic directly between compute nodes
 - Supports north-south (one-to-one NAT connections to the outside [SNAT without port mapping]) directly from the compute node if a Floating IP has been assigned
 - If not, traffic must flow through the network node to make use of the NAT gateway IP (SNAT with port mapping)
 - Supports north-south (one-to-one NAT for inbound connections [DNAT with no port mapping]) directly to the compute node with a Floating IP



This model illustrates the previous example with the addition of the DVR

NOTE: For introduction to DVR, go to <https://wiki.openstack.org/wiki/Neutron/DVR/HowTo>.

Neutron DVR

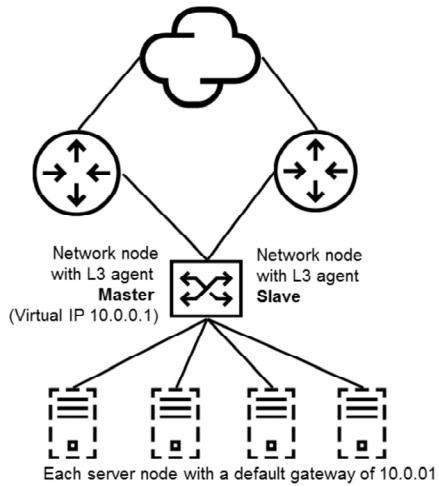
When added to a compute node, the Neutron DVR (Distributed Virtual Router) significantly reduces the amount of network traffic between the compute node and the controller node, because even though the network node is still in charge of north/south SNAT (source NAT) traffic, all of the north/south DNAT(destination NAT) and east/west L3 forwarding is done locally by the DVR on the compute node.

Not only is the traffic between the compute node and network traffic reduced, but this architecture offers some interesting FT (fault tolerance) and HA (high available) characteristics, because east/west and north/south DNAT traffic no longer flows through a central network node, thus providing better load distribution.

Furthermore, there is no longer a single point of failure as far as networking goes; a failure of the networking functions localized to the compute host will make the compute instances unreachable in the same way as if the compute host were to go down, bringing the compute instances down with it.

As far as north/south SNAT is concerned, this will still need to flow through a central network node. Multiple network nodes can still be run. However, a failure of a network node brings down all the routers associated with it, causing instances running on multiple compute nodes to lose external connectivity at once.

Neutron L3 in high availability configuration



- **Issue:** Each network node is a single point of failure (SPOF) for all instances requiring routing services
- **Resolution:** VRRP (Virtual Router Redundancy Protocol)
 - Each server node has a default gateway that is the VIP (Virtual IP) address of the master router
 - At any time that the standby router no longer receives hello packets from the master router, an election process ensues and the new master assumes the VIP address for its LAN-facing interface

NOTE: For more information on high availability using VRRP see <http://docs.openstack.org/newton/networking-guide/config-dvr-ha-snat.html>.

Neutron L3 in high availability configuration

As of Mitaka release, Neutron supports the DVR and SNAT high availability (HA)—a quick failover of the SNAT service to a backup DVR or SNAT router on an L3-agent running on a different node. SNAT high availability is implemented through the Keepalived service that uses VRRP to provide the failover.

During regular operation, the master router periodically transmits heartbeat packets over a hidden project network that connects all HA routers for a particular project.

If the DVR or SNAT backup router stops receiving these packets, it assumes failure of the master DVR or SNAT router and promotes itself to the master router by configuring IP addresses on the interfaces in the SNAT namespace. In environments with more than one backup router, the VRRP rules are followed to select a new master router.



Common Neutron management tasks

Common CLI management tasks (1 of 2)

Creating a network

```
student@ubuntu14libDevStack:~/devstack$ neutron net-create net1
Created a new network:
```

Field	Value
admin_state_up	True
id	753ae528-a473-4d7b-aa01-7c28714eb25f
mtu	0
name	net1
port_security_enabled	True
provider:network_type	vxlan
provider:physical_network	1092
provider:segmentation_id	False
router:external	False
shared	ACTIVE
status	5cab56c4beaf44d7a8ab14c766d5c187
subnets	
tenant_id	

If a project (tenant) is not specified when the network is created, the current project environment value will be used

Creating a subnet

```
student@ubuntu14libDevStack:~/devstack$ neutron subnet-create net1 192.168.2.0/24 --name subnet1
Created a new subnet:
```

Field	Value
allocation_pools	{"start": "192.168.2.2", "end": "192.168.2.254"}
cidr	192.168.2.0/24
dns_nameservers	True
enable_dhcp	192.168.2.1
gateway_ip	
host_routes	
id	60fa4bb6-c295-4683-a76d-8c35c8c84d83
ip_version	4
ipv6_address_mode	
ipv6_ra_mode	
name	subnet1
network_id	753ae528-a473-4d7b-aa01-7c28714eb25f
subnetpool_id	
tenant_id	5cab56c4beaf44d7a8ab14c766d5c187

When a subnet is created, DHCP is enabled and a pool of IP addresses are assigned to it by default

Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

37

Common CLI management tasks (1 of 2)

Use the following commands to perform common management tasks:

- To create a network, enter \$ `neutron net-create net1`
- To create a subnet, enter \$ `neutron subnet-create net1 192.168.2.0/24 --name subnet1`

Common CLI management tasks (2 of 2)

Creating a port

```
student@ubuntu14libDevStack:~/devstack$ neutron port-create net1
Created a new port:
+-----+-----+
| Field | Value |
+-----+-----+
| admin_state_up | True
| allowed_address_pairs |
| binding:host_id |
| binding:profile |
| binding:vif_details |
| binding:vif_type | unbound
| binding:vnic_type | normal
| device_id |
| device_owner |
| dns_assignment |
| dns_name |
| fixed_ips |
| id | b62e7c28-0ee5-46d7-be67-03f6c9699bb0
| mac_address | fa:16:3e:ed:ae:ed
| name |
| network_id | 753aae528-a473-4d7b-aa01-7c28714eb25f
| port_security_enabled | True
| security_groups | 393e4e3d-6553-409a-ab15-dd9624658a74
| status | DOWN
| tenant_id | 5cab56c4beaf44d7a8ab14c766d5c187
+-----+
```

In this example, Neutron selects an IP address from the pool; however, you can also specify an IP address at the CLI

Common CLI management tasks (2 of 2)

Use the following commands to perform common management tasks:

- To create a port and let Neutron select the IP address, enter: `$ neutron port-create net1`
- To create a port and specify an IP address, enter:

```
$ neutron port-create net1 --fixed-ip ip_address=192.168.2.40
```

Neutron in Horizon

Network Topology

The screenshot shows the OpenStack Horizon interface with the 'Network' tab selected. On the left, there's a sidebar with 'Network Topology' selected under 'Network'. The main canvas displays a network topology with a 'private' cloud icon and a 'public' network icon connected by a line, with a 'router1' icon in between. A callout box points to the 'router1' icon with the text: 'Click any of the icons to see details and, in some cases, action buttons, as shown for the router'. To the right, a detailed view of the 'router1' component is shown in a modal dialog. The dialog shows the router's ID (ac124940-e31c-4278-b436-3c8a5e0f448f), its status (ACTIVE), and its interfaces. The 'Interfaces' section lists three entries:

ID	MAC Address	IP Address	Status	Action
3fa2353a-59f...	fdde:7e04:b81b::1	172.24.4.2	ACTIVE	Add Interface
58557563-d7...	172.24.4.2	2001:db8:1	ACTIVE	Delete Interface
643be29c-08...	10.0.0.1		ACTIVE	Delete Interface

router1

ID: ac124940-e31c-4278-b436-3c8a5e0f448f
Status: ACTIVE

Interfaces

ID	MAC Address	IP Address	Status	Action
3fa2353a-59f...	fdde:7e04:b81b::1	172.24.4.2	ACTIVE	Add Interface
58557563-d7...	172.24.4.2	2001:db8:1	ACTIVE	Delete Interface
643be29c-08...	10.0.0.1		ACTIVE	Delete Interface

[View Router Details](#)

© Copyright 2017 Hewlett Packard Enterprise Development LP

39

Neutron in Horizon—Network Topology

OpenStack® Horizon provides the interface for Neutron configuration. This interface is convenient while working with complex network topologies. The Network Topology panel provides the interactive view of network components. Upon selecting the component, Horizon displays the dialog showing the details about the selected component.

Neutron in Horizon

Networks and Routers

The image contains two side-by-side screenshots of the OpenStack Horizon web interface.

Left Screenshot (Networks View):

- Header:** openstack admin admin
- Left Sidebar:** Project, Compute, Network (selected), Network Topology, Networks, Routers, Orchestration, Object Store, Admin, Identity.
- Table Headers:** Name, Subnets Associated, Shared, Status, Admin State, Actions.
- Data:**

Name	Subnets Associated	Shared	Status	Admin State	Actions
public	public-subnet 172.24.4.0/24 Ipv6-public-subnet 2001:db8::/64	No	Active	UP	Edit Network
net1	subnet1 192.168.2.0/24	No	Active	UP	Edit Network Add Subnet Delete Network

Right Screenshot (Routers View):

- Header:** openstack demo admin
- Left Sidebar:** Project, Compute, Network (selected), Network Topology, Networks, Routers, Orchestration, Object Store, Admin, Identity.
- Table Headers:** Name, Status, External Network, Admin State, Actions.
- Data:**

Name	Status	External Network	Admin State	Actions
router1	Active	public	UP	Clear Gateway

Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

40

Neutron in Horizon—Networks and Routers

Besides the topology view, Horizon can be used to see the details of network components. The screenshot above shows the view of Neutron networks and routers. The information is displayed in the form of a table. The right-hand side of the table provides the action button, which is context-sensitive. The Name column in the table provides the links to more details about the object in the column.



Troubleshooting Neutron

Approach to network troubleshooting

- To troubleshoot network issues:
 - Use OpenStack® commands and networking tools to isolate the cause of the network error
 - Network errors can have a severe impact on OpenStack® functioning, and a single error might manifest differently with several cloud services
 - Concentrate on looking for errors at the Linux bridge or the Open vSwitch configuration



© Copyright 2017 Hewlett Packard Enterprise Development LP

42

Approach to network troubleshooting

Network troubleshooting can be difficult because a single network issue can cause a problem at several points in the cloud. The issue could be caused by problems at the physical interface level or the switch level, or it could be caused by damaged cables. The problems with the software configuration will manifest in a way that is similar to the problems at the physical level.

Your goal should be to use available tools to quickly isolate the cause of the network error, looking at the Linux bridge or the Open vSwitch configuration. This section defines some of the available tools that can help to diagnose network problems.

Neutron log files

- Neutron log file: /var/log/neutron
- Messaging log file: /var/log/rabbitmq
- Plug-in log file: /var/log/openvswitch

NOTE: To learn more about network troubleshooting, go to
<http://docs.openstack.org/ops-guide/ops-network-troubleshooting.html>.



© Copyright 2017 Hewlett Packard Enterprise Development LP

43

Neutron log files

To start debugging Neutron problems, check the log files first. The process of debugging the network services is complex and out of the scope of this training. For more information, check the Network Troubleshooting chapter of the *OpenStack Operations Guide* that is available at http://docs.openstack.org/openstackops/content/network_troubleshooting.html.

Module summary

In this module:

- The primary components of a virtual network were discussed
- The purpose and components of OpenStack® Neutron were described
- Common configuration tasks for Neutron were identified
- Basic Neutron networking configuration examples were explained
- The process of creating OpenStack® networks, subnets, and routers was described



© Copyright 2017 Hewlett Packard Enterprise Development LP

44

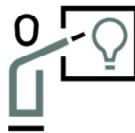
Module summary

In this module:

- The primary components of a virtual network were discussed
- The purpose and components of OpenStack® Neutron were described
- Common configuration tasks for Neutron were identified
- Basic Neutron networking configuration examples were explained
- The process of creating OpenStack® networks, subnets, and routers was described

Learning check

Learning check



What network managers are available in OpenStack®? (Select three)

- A. FlatManager
- B. FlatDHCPManager
- C. MultiManager
- D. FlatDNSManager
- E. VLAN Manager
- F. FlexManager

Learning check

What network managers are available in OpenStack®? (Select three)

- A. FlatManager
- B. FlatDHCPManager
- C. MultiManager
- D. FlatDNSManager
- E. VLAN Manager
- F. FlexManager

Learning check answer



What network managers are available in OpenStack®? (Select three)

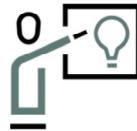
- A. FlatManager**
- B. FlatDHCPManager**
- C. MultiManager
- D. FlatDNSManager
- E. VLAN Manager**
- F. FlexManager

Learning check answer

What network managers are available in OpenStack®? (Select three)

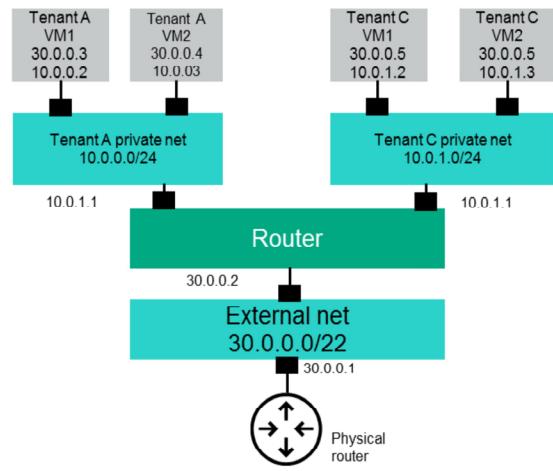
- A. FlatManager**
- B. FlatDHCPManager**
- C. MultiManager
- D. FlatDNSManager
- E. VLAN Manager**
- F. FlexManager

Learning check



What use case does this diagram represent?

- A. Multiple flat networks
- B. Mixed flat and private networks
- C. Per-tenant routers with private networks
- D. Provider router with private networks



© Copyright 2017 Hewlett Packard Enterprise Development LP

48

Learning check

What use case does this diagram represent?

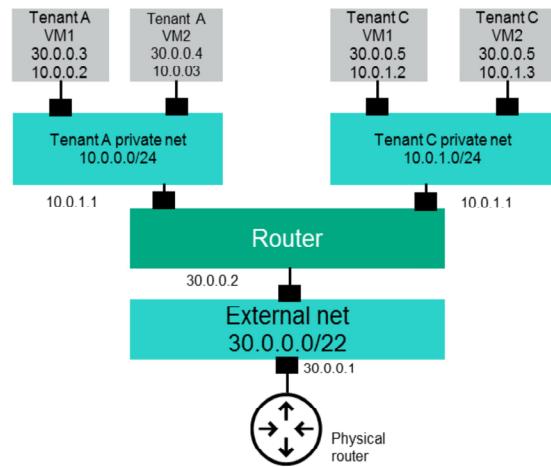
- A. Multiple flat networks
- B. Mixed flat and private networks
- C. Per-tenant routers with private networks
- D. Provider router with private networks

Learning check answer



What use case does this diagram represent?

- A. Multiple flat networks
- B. Mixed flat and private networks
- C. Per-tenant routers with private networks
- D. Provider router with private networks**



Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

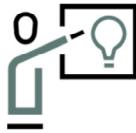
49

Learning check answer

What use case does this diagram represent?

- A. Multiple flat networks
- B. Mixed flat and private networks
- C. Per-tenant routers with private networks
- D. Provider router with private networks**

Learning check



What is the role of the Neutron L3 agent?

- A. Provide L3/NAT forwarding to provide external network access for VMs on tenant networks
- B. Provide DHCP services for VMs on tenant networks
- C. Create and configure all L3 networks in OpenStack®
- D. Manage the DHCP and plug-in agents

Learning check

What is the role of the Neutron L3 agent?

- A. Provide L3/NAT forwarding to provide external network access for VMs on tenant networks
- B. Provide DHCP services for VMs on tenant networks
- C. Create and configure all L3 networks in OpenStack®
- D. Manage the DHCP and plug-in agents

Learning check answer



What is the role of the Neutron L3 agent?

- A. **Provide L3/NAT forwarding to provide external network access for VMs on tenant networks**
- B. Provide DHCP services for VMs on tenant networks
- C. Create and configure all L3 networks in OpenStack®
- D. Manage the DHCP and plug-in agents

Learning check answer

What is the role of the Neutron L3 agent?

- A. **Provide L3/NAT forwarding to provide external network access for VMs on tenant networks**
- B. Provide DHCP services for VMs on tenant networks
- C. Create and configure all L3 networks in OpenStack®
- D. Manage the DHCP and plug-in agents



**Hewlett Packard
Enterprise**

Hewlett Packard
Enterprise

Fundamentals of OpenStack® Technology

Module 7—OpenStack® Compute Service (Nova)

H6C68S E.01

© Copyright 2017 Hewlett Packard Enterprise Development LP

Learning objectives

After completing this module, you should be able to:

- Discuss the function and features of Nova and related terminology
- Identify the major Nova OpenStack® Compute architectural components
- Discuss the types of compartmentalization of Nova instances
- Complete some common Nova management tasks
- Explain the operation of the Nova scheduler
- Describe the basic troubleshooting steps for instances and the Nova service



© Copyright 2017 Hewlett Packard Enterprise Development LP

2

Learning objectives

After completing this module, you should be able to:

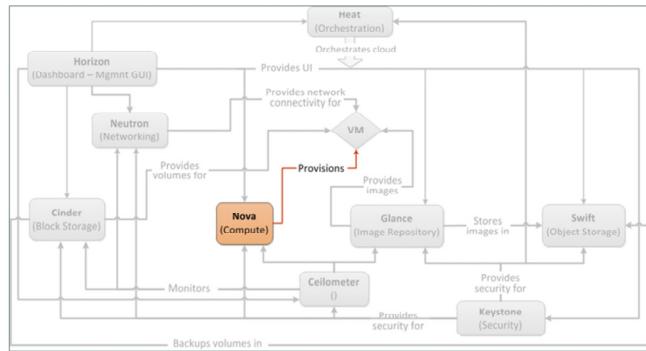
- Discuss the function and features of Nova and related terminology
- Identify the major Nova OpenStack® Compute architectural components
- Discuss the types of compartmentalization of Nova instances
- Complete some common Nova management tasks
- Explain the operation of the Nova scheduler
- Describe the basic troubleshooting steps for instances and the Nova service



Nova overview

Nova features and functions

- The cloud computing fabric controller for OpenStack®
- Implements services and associated libraries to provide massively scalable, on-demand, self-service access to compute resources, including baremetal, virtual machines, and containers
- Interacts with other OpenStack® services to provide the resources necessary for the provisioning and management of OpenStack® compute instances



Note: Baremetal (Ironic project) and Container (Magnum project) provisioning are discussed in the Ironic and Magnum sections of Module 12 of this course—*Other OpenStack® Projects*.

Nova features and functions

The Nova service is the heart of OpenStack®. It controls the Infrastructure as a Service (IaaS) cloud computing platform by:

- Providing a standard Representational State Transfer (REST) interface from which VMs can be managed
- Supporting multiple types of hypervisors
- Supporting bare-metal instances
- Starting and managing instances
- Supporting single and multi-tenancy

This course does not focus on the Nova functionality that has been ported out to separate services, such as Keystone, Neutron, and Glance. Some of that functionality still exists in Nova, but to reduce confusion and recognize the direction where OpenStack® is headed, this course discusses that functionality within the stand-alone services rather than how it can be used within Nova.

Nova design guidelines

- **Component-based architecture**—Quickly add new behaviors
- **High availability**—Scale for large workloads
- **Fault tolerance**—Isolate processes to avoid cascading failures
- **Recoverability**—Easily diagnose, debug, and rectify failures
- **Open standards**—Be a reference implementation for a community-driven API
- **API compatibility**—Provide API compatibility with popular systems like Amazon EC2, as well as the ability to integrate with legacy systems and third-party technologies



© Copyright 2017 Hewlett Packard Enterprise Development LP

5

Nova design guidelines

Nova is written with the following design guidelines in mind:

- **Component-based architecture**—Quickly add new behaviors
- **High availability**—Scale for large workloads
- **Fault tolerance**—Isolate processes to avoid cascading failures
- **Recoverability**—Easily diagnose, debug, and rectify failures
- **Open standards**—Be a reference implementation for a community-driven API
- **API compatibility**—Provide API compatibility with popular systems like Amazon EC2

Nova (Compute) API version states as of Newton release

- Compute API v2.38 (**current**)
- Changes in Newton:
 - Cells V2 supports booting instances for one Cell v2 only
 - Nova uses Glance v2 API for getting image resources
 - Hyper-V RemoteFX feature enhances the visual experience in RDP connections
 - Multitenant networking for the Ironic compute driver is supported



© Copyright 2017 Hewlett Packard Enterprise Development LP

6

Nova (Compute) API version states as of Newton release

Newton OpenStack® release supports the API v2.38. Some of the changes in this release include the following:

- Cells V2 supports booting instances for one cell v2 only.
- Nova uses Glance v2 API for getting image resources.
- Hyper-V RemoteFX feature enhances the visual experience in RDP connections.
- Multitenant networking for the Ironic compute driver is supported.

API micro-versioning

- API versioning is used to bring new features, without breaking the compatibility
- Nova provides support for API v2.1:
 - **Major versions**, which have dedicated URLs
 - **Micro-versions**, which can be requested through the use of:
 - The `X-OpenStack-Nova-API-Version` header, until micro-version 2.27
 - The `OpenStack-API-Version` header, since micro-version 2.27



© Copyright 2017 Hewlett Packard Enterprise Development LP

7

API micro-versioning

In order to bring new features to you over time, the Nova API supports versioning. There are two kinds of versions in Nova:

- “Major versions,” which have dedicated URLs
- “Micro-versions,” which can be requested through the use of the `X-OpenStack-Nova-API-Version` header

Note: As of microversion 2.27, the `OpenStack-API-Version` header can also be used.

Common Nova terminology (1 of 2)

Term	Description
Baremetal server	<ul style="list-style-type: none"> – A physical server; from the OpenStack® Nova perspective, it is the provisioning of physical hardware as opposed to virtual machines – See https://wiki.openstack.org/wiki/Ironic
Container	<ul style="list-style-type: none"> – Wraps up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries—anything you can install on a server – This enables the software to run in the same way, regardless of the environment it is running in – See: https://wiki.openstack.org/wiki/Magnum
Flavor	<ul style="list-style-type: none"> – An available hardware configuration for a server – Each flavor has a unique combination of disk space, memory capacity, and priority for CPU time
Image	<ul style="list-style-type: none"> – A collection of files used to create or rebuild a server – Operators provide a number of prebuilt OS images by default
Oslo Messaging	<ul style="list-style-type: none"> – A messaging API (Application Programming Interface) that supports RPC (Remote Procedure Calls) and notifications using the Nova Queue
Reboot	<ul style="list-style-type: none"> – Performs either a soft or a hard reboot of a server – A soft reboot allows for a graceful shutdown of all processes
Rebuild	<ul style="list-style-type: none"> – Removes all data on the server and replaces it with the specified image – The server ID and IP addresses remain the same

 Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

8

Common Nova terminology (1 of 2)

This table lists some common terms used in Nova. It is important to keep in mind that these are definitions from the Nova perspective. Some of the definitions mean something else in other technologies (for example, “baremetal server” can be defined differently outside Nova) and in other OpenStack® services (for example, “container” means an object storage unit in OpenStack® Swift).

Common Nova terminology (2 of 2)

Term	Description
Resize	Convert an existing server to a different flavor
Server (instance)	A VM or a baremetal system onto which an OS can be installed
Virtual server	Alternative term for a VM or the guest of a VM host



© Copyright 2017 Hewlett Packard Enterprise Development LP

9

Common Nova terminology (2 of 2)

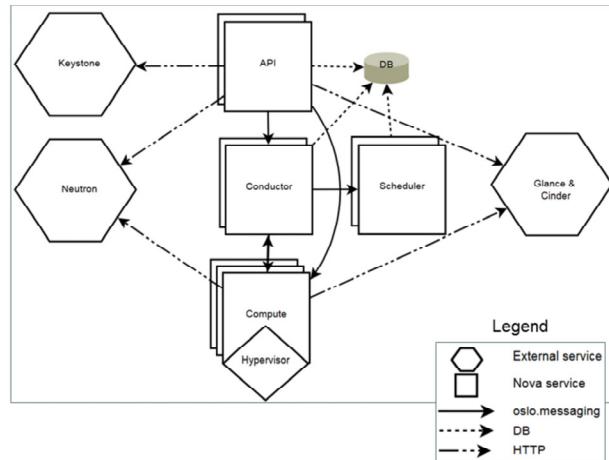
To resize a Nova instance means to convert the existing instance to a different flavor. Therefore, resizing affects RAM, CPU, and disk resources.



Nova architecture

Nova system architecture

- Built on a shared-nothing, messaging-based architecture
- **DB** is a SQL database for data storage
- **API:**
 - Receives HTTP requests, converts commands and communicates with other components via the oslo.messaging queue or HTTP
 - Supports the OpenStack® Compute API, Amazon EC2 API, and a special Admin API
- **Scheduler** decides which host gets each instance
- **Compute** manages communication with hypervisor and virtual machines
- **Conductor** handles requests that need coordination (build and resize), acts as a database proxy, or handles object conversions



Note: The above diagram is taken from <http://docs.openstack.org/developer/nova/architecture.html>.

Nova system architecture

Nova comprises multiple server processes, each performing different functions. The user-facing interface is a REST API, while internally, Nova components communicate via a Remote Procedure Call (RPC) message passing mechanism.

The API servers process the REST requests, which typically involve database reads and writes, optionally sending RPC messages to other Nova services and generating responses to the REST calls. RPC messaging is done via the oslo.messaging library, an abstraction on top of message queues. Most of major Nova components can be run on multiple servers and have a manager that is listening for RPC messages. The one major exception is **nova-compute**, where a single process runs on the hypervisor it is managing (except when using the VMware or Ironic drivers). The manager also has periodic tasks.

Nova also uses a central database that is (logically) shared among all components. However, to aid upgrade, the database is accessed through an object layer that ensures that an upgraded control plane can still communicate with a nova-compute running the previous release. To make this possible, nova-compute proxies database requests over RPC to a central manager called "**nova-conductor**".

Note: There are several other Nova components that are not discussed here because they are unrelated to the content presented in this course. In other words, their inclusion would only add a level of complexity that is not necessary.

Supported Nova hypervisors (1 of 2)

- Support of each of the hypervisors is not equal; they are not tested in the same amount
- Not all hypervisors support the same features, so OpenStack® groups them into groups (A and B)
- The official support matrix is maintained at <http://docs.openstack.org/developer/nova/support-matrix.html>

NOTE: Nova interoperates with the OpenStack® Ironic project to provide support for provisioning of baremetal servers.

Hypervisor	Description
Group A—Fully supported	
KVM	Kernel-based Virtual Machine
QEMU	Quick emulator

Hypervisor	Description
Group B—Functional testing provided by an external system that does not gate commits, but informs patch authors and reviewers of results in the code review system	
Hyper-V	Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2
LXC via libvirt	
Virtuozzo	
VMware	ESX or ESXi 4.1 update 1 and later
XenServer	Xen, Citrix XenServer, and Xen Cloud Platform XCP
Xen via libvirt	



© Copyright 2017 Hewlett Packard Enterprise Development LP

12

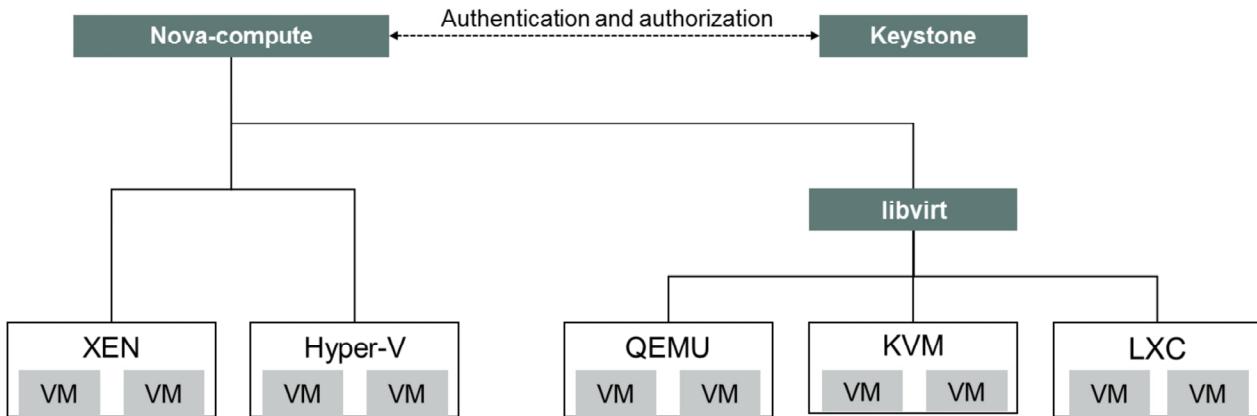
Supported Nova hypervisors (1 of 2)

Nova currently supports the following hypervisors:

- **KVM:** The virtual disk formats that are supported are inherited from the QEMU because it uses a modified QEMU program to launch the VM. The supported formats include raw images, the QCOW2, and VMware.
- **Linux Containers (LXC):** Through libvirt, Linux Containers are used to run Linux-based VMs.
- **QEMU:** This is generally only used for development purposes.
- **User Mode Linux (UML):** This is generally only used for development purposes.
- **VMware vSphere 4.1 update 1 and newer:** This runs VMware-based Linux and Microsoft Windows images through a connection with a vCenter server or directly with an ESXi host.
- **Xen:** XenServer and Xen Cloud Platform (XCP) are used to run Linux or Windows VMs. You must install the nova-compute service in a paravirtualized VM.
- **IBM PowerVM:** Server virtualization with PowerVM is used to run AIX, IBM i, and Linux environments on IBM POWER technology.
- **Microsoft Hyper-V:** Server virtualization with Hyper-V is used to run Windows, Linux, and FreeBSD VMs. It runs nova-compute natively on the Windows virtualization platform.
- **Ironic:** This bare metal option is not a hypervisor in the traditional sense. It is a driver that provisions physical hardware by means of pluggable sub-drivers, such as PXE for image deployment and Intelligent Platform Management Interface (IPMI) for power management.

Red Hat also has a hypervisor and a controller.

Supported Nova hypervisors (2 of 2)



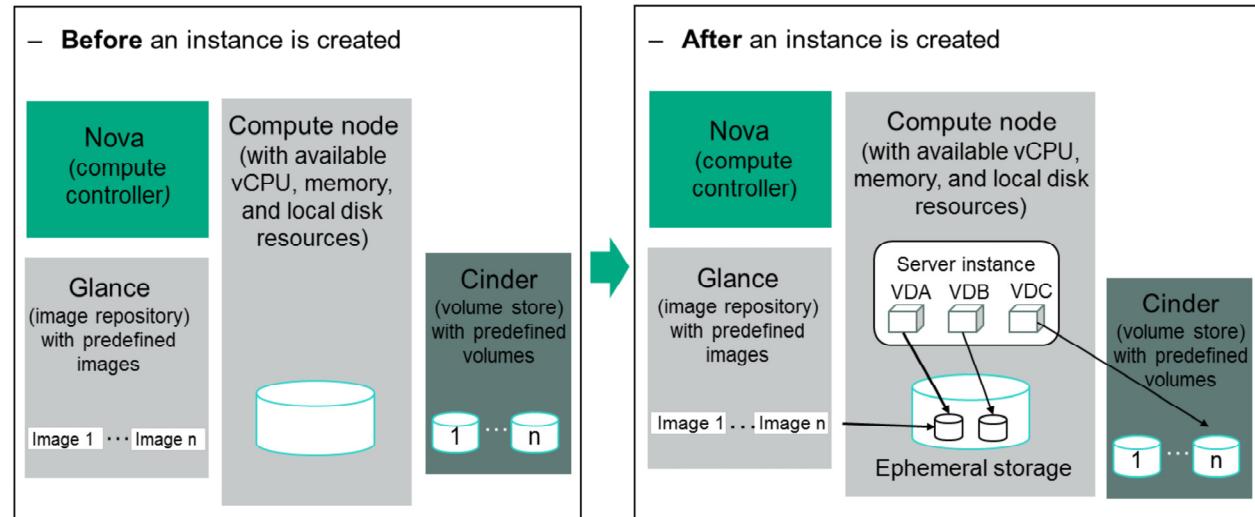
Supported Nova hypervisors (2 of 2)

Nova-compute supports the hypervisors shown on the slide and those supported by libvirt. Nova-compute does not include any virtualization software; instead, it uses drivers that interact with underlying virtualization mechanisms that run on the host operating system.

Currently, most OpenStack® development is done with the KVM and XEN hypervisors. This means that you are more likely to find community support for issues with these hypervisors. All features that are currently supported in KVM are also supported in QEMU.

For a detailed list of features and support across the hypervisors, go to
<http://docs.openstack.org/developer/nova/support-matrix.html>.

VM instance creation process (1 of 2)



VM instance creation process (1 of 2)

In this example, before an instance is created:

- The image must exist in Glance
- The volume, if required, must be available to Cinder
- The instance flavor must be specified
- The nova-scheduler must be able to locate a compute node that can accommodate the specified flavor

In the instance creation process:

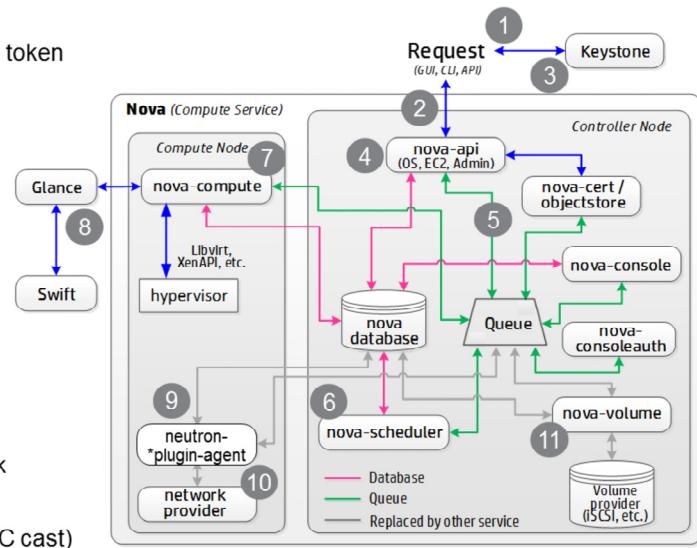
- An image is copied from Glance to the local storage of the selected compute node, on VDA in this example. If required, a new empty disk image is created to present as the second disk (VDB).
- If block storage is required, the compute node attaches to the requested cinder-volume using iSCSI or Fibre Channel and then maps this to the third disk (VDC) as requested.
- The vCPU and memory resources are provisioned and the instance is booted from the VDA node.

After an instance is created:

- The compute node OS is operational on VDA.
- The VDB node is available for storage.
- The VDC node is connected by means of an iSCSI or Fibre Channel connection to a Cinder volume.

VM instance creation process (2 of 2)

1. Request Keystone to generate and return an auth token
2. Send an API request to nova-api with the token
3. Nova-api validates the API token
4. Nova-api processes the API request
5. Nova-api makes an RPC call to the scheduler
6. A host is selected based on the scheduling policies (nova-scheduler)
7. Start provisioning the VM on the compute node (RPC cast to nova-compute)
8. Request an OS image from Glance
 - a. Get the image URI from Glance
 - b. The image can be downloaded from Swift
9. Nova-api makes an RPC call to allocate a network
10. Create the network and associate it with the VM
11. Attach a block volume appropriately (using an RPC cast)



Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

15

VM instance creation process (2 of 2)

The VM instance creation process is shown on the slide above.



Compartmentalizing OpenStack® deployments

Segregating a cloud implementation

	Separate Nova deployments	Divided single deployment		
	Cells 2 (as of Liberty, not fully functional)	Regions	Availability zones	Host aggregates
Use when you need	<ul style="list-style-type: none"> - A single API endpoint for compute, or you need a second level of scheduling 	<ul style="list-style-type: none"> - Discrete regions with separate API endpoints and no coordination between regions 	<ul style="list-style-type: none"> - Logical separation within your Nova deployment for physical isolation or redundancy 	<ul style="list-style-type: none"> - Scheduling a group of hosts with common features
Example	<ul style="list-style-type: none"> - A cloud with multiple sites where you can schedule VMs "anywhere" or on a particular site 	<ul style="list-style-type: none"> - A cloud with multiple sites, where you schedule VMs to a particular site and you want a shared infrastructure 	<ul style="list-style-type: none"> - A single site cloud with equipment fed by separate power supplies 	<ul style="list-style-type: none"> - Scheduling to hosts with trusted hardware support
Overhead	<ul style="list-style-type: none"> - Each cell has a full Nova installation except nova-api 	<ul style="list-style-type: none"> - A different API endpoint for every region - Each region has a full Nova installation 	<ul style="list-style-type: none"> - Configuration changes to nova.conf 	<ul style="list-style-type: none"> - Configuration changes to nova.conf
Shared services	<ul style="list-style-type: none"> - Keystone - Nova-api 	<ul style="list-style-type: none"> - Keystone 	<ul style="list-style-type: none"> - Keystone - All Nova services 	<ul style="list-style-type: none"> - Keystone - All Nova services

Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

17

Segregating a cloud implementation

OpenStack® Compute cells are designed to allow running the cloud in a distributed fashion without having to use more complicated technologies, or being invasive to existing Nova installations. Hosts in a cloud are partitioned into groups called "cells." Cells are configured in a tree. The top-level cell ("API cell") has a host that runs the nova-api service, but no nova-compute services. Each child cell runs all of the other typical nova-* services found in a regular installation, except for the nova-api service. Each cell has its own message queue and database service, and also runs nova-cells, which manages the communication between the API cell and child cells.

This allows for a single API server being used to control access to multiple cloud installations. Introducing a second level of scheduling (the cell selection), in addition to the regular nova-scheduler selection of hosts, provides greater flexibility to control where virtual machines are run.

Regions have a separate API endpoint per installation, allowing for a more discrete separation. If you want to run instances across sites, you have to explicitly select a region. However, the additional complexity of running a new service is not required.

The OpenStack® Dashboard (Horizon) currently uses only a single region, so one dashboard service should be run per region. Regions are a robust way to share infrastructure between OpenStack® Compute installations and they allow for a high degree of failure tolerance.

Both availability zones and host aggregates partition a single Nova deployment. Although it seems that host aggregates and availability zones can be configured in a similar fashion, they differ in their intended use.

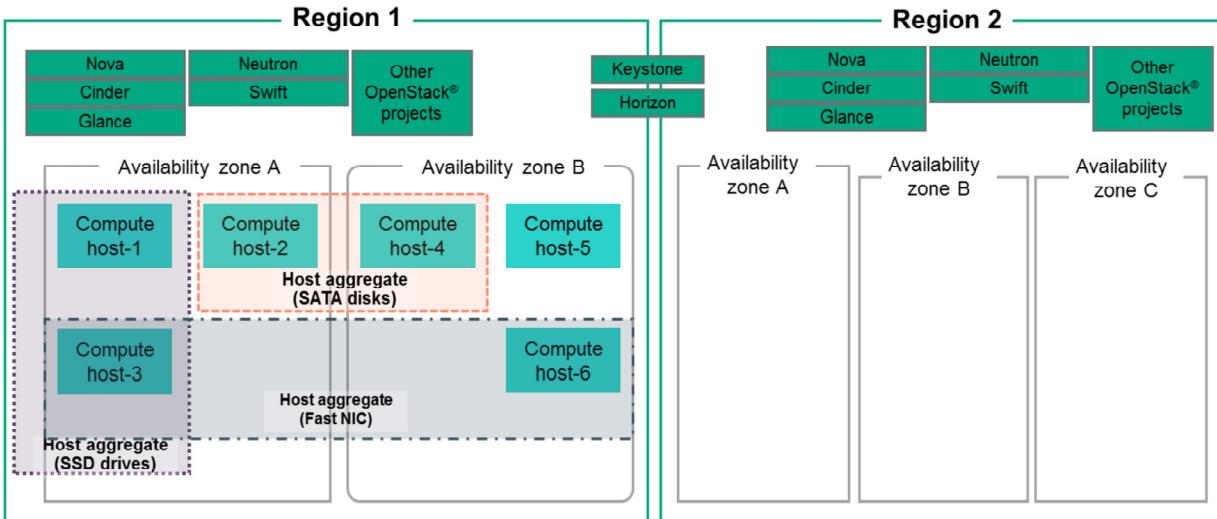
Availability zones (AZ) allow the partition of OpenStack® Compute deployments into logical groups for load balancing and instance distribution. They allow you to arrange sets of either OpenStack® Compute or OpenStack® Block Storage hosts into logical groups. Each availability zone should have a local compute or block storage host. Availability zones are commonly used to identify a sets of servers that have some common attribute. For instance, if some of the racks in your data center are on a separate power source, you can put servers in those racks in their own availability zone. Availability zones can also be helpful for separating out different classes of hardware. This is especially helpful with OpenStack® Block Storage where you can have storage servers with different types of hard drives. When provisioning resources, you can specify what availability zone you would like your instance or volume to come from. This allows cloud consumers to ensure that their application resources are spread across multiple disparate machines to achieve high availability in the event of hardware failure.

Host aggregates are used to provide some form of physical isolation and redundancy from other availability zones (such as by using separate power supply or network equipment). Host aggregates can be regarded as a mechanism to further partition an availability zone, that is, into multiple groups of hosts that share common resources like storage and network, or have a special property such as trusted computing hardware.

A common use of host aggregates is to provide information for use with the nova-scheduler—for example, limiting specific flavors or images to a subset of hosts.

You can find a detailed OpenStack® Summit presentation at <https://www.openstack.org/summit/openstack-summit-atlanta-2014/session-videos/presentation/divide-and-conquer-resource-segregation-in-the-openstack-cloud>.

Regions, availability zones, and host aggregates



Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

19

Regions, availability zones, and host aggregates

The slide above provides an example of the hierarchy of regions, availability zones, and host aggregates.

Points of interest:

- Region 1 and region 2 are sharing the Keystone and Horizon services, with the remaining OpenStack® services dedicated to the region they belong to.
- Any number of availability zones can exist in a region.
- The compute hosts in an availability zone can belong to the same host aggregate in a different availability zone.

Regional architecture

- Regional architecture:
 - Is designed with N+1 power redundancy made up of three independent power systems
 - Contains a minimum of 3 physically separate availability zones
- Each availability zone:
 - Is fed by two independent power feeds from separate substations
 - Has a minimum of two network drops at separate ends of the facility
 - Contains redundant power to each rack and diverse cabling to eliminate any single points of failure



Regional architecture

An availability zone is a cluster of compute nodes (servers) that hosts the HPE Cloud virtual machines (instances). HPE Cloud's AZs are totally isolated fault domains. The AZs are in separate physical zones within the switch facility which are further separated by fireproof walls.

In addition, HPE Cloud has redundant network cores which span AZ1 and AZ2 and are connected by redundant uplinks to the internet. HPE Cloud can lose an entire AZ and the remaining AZ continues to function regularly. The service could also lose an uplink to the internet and all AZs would continue to function regularly.

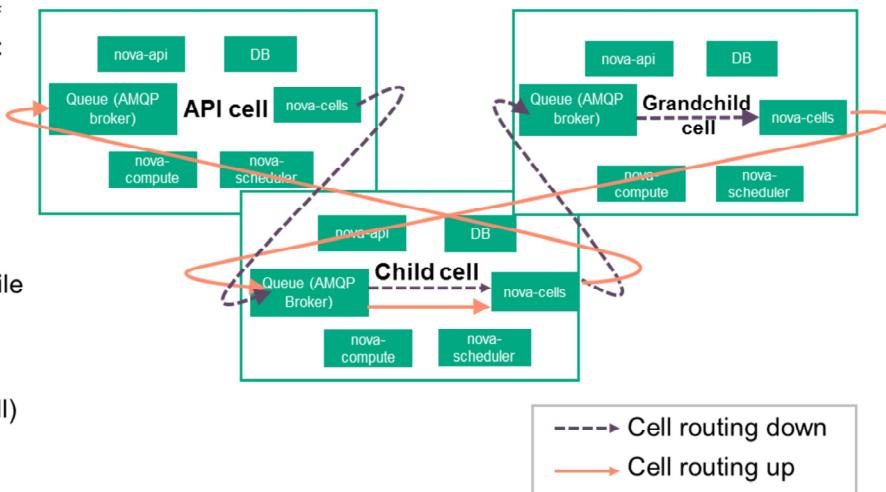
Currently, in the US West region there are three availability zones, AZ1, AZ2, and AZ3, which are all on the west coast of the United States. The US East region also has three availability zones, but because of the different architecture, they are not separated in the Management Console like US West is.

Most of the above aspects apply to Object Storage as well. Object Storage currently has both US West and US East regions ready for use.

Cells

API cell, child cells, and grandchild cells

- Nova cells create a hierarchy of Nova installs, each with its own:
 - Database
 - Message queue backend
 - Scheduler
 - Compute nodes
- This allows the scope of any one Nova cell to be smaller, while still being able to manage the whole environment through a single Nova API, which runs at a parent cell (that is, the API cell)



Cells—API cell, child cells, and grandchild cells

The cell functionality enables you to scale an OpenStack® Compute cloud in a more distributed fashion without having to use complicated technologies like database and message queue clustering. It supports very large deployments.

OpenStack® cells essentially provide the means to create logical fences around OpenStack® resources (compute and storage) in the same manner Swift and Keystone do via regions. Cells are an imaginary line that collects resources together. When coupled with policies that are specific to each region, cells create a cloud environment that can treat some resources differently than others.

Cells are configured as a tree. The top-level cell should have a host that runs a nova-api service, but no nova-compute services. Each child cell should run all of the typical nova-* services in a regular Compute cloud, except for nova-api. You can think of cells as a normal Compute deployment in that each cell has its own database server and message queue broker.

The nova-cells service handles communication between cells and selects cells for new instances. This service is required for every cell. Communication between cells is pluggable, and currently the only option is communication through RPC.

Cells scheduling is separate from host scheduling. Nova-cells first selects a cell. Once a cell is selected and the new build request reaches its nova-cells service, it is sent over to the host scheduler in that cell and the build proceeds, as it would have without cells.



Common Nova management tasks

Nova configuration files

- Nova is configured after Keystone is installed and configured

File name	Description
/etc/nova/nova.conf	The primary configuration file of Nova, which includes general and driver-specific configuration parameters
/etc/nova/nova-paste.ini	PasteDeploy configuration entries (Web Server Gateway Interface pipeline definitions) for Python WSGI-based applications
/var/log/nova	A directory containing Nova log files

NOTE: Linux distribution configuration files are available on the OpenStack® documentation page (<http://docs.openstack.org>) under the “Install OpenStack” section. The general Nova configuration reference guide for the Liberty release is available at <http://docs.openstack.org/newton/config-reference/compute.html>.



© Copyright 2017 Hewlett Packard Enterprise Development LP

23

Nova configuration files

The main Nova configuration file is located at /etc/nova/nova.conf. The nova.conf configuration file is an INI file format, and it provides settings for:

- Drivers
- Networking
- Service endpoints
- Message Q
- VNC
- Notification
- Instance usage auditing
- Logging
- Volume management
- Scheduler

You can use a particular configuration option file by using the option (nova.conf) parameter when you run one of the nova-* services. This parameter inserts configuration option definitions from the specified configuration file name, which might be useful for debugging or performance tuning.

Gathering information required to create an instance

Available images

- The following commands list the available images and details about a specific image

The screenshot shows a terminal window with two command outputs. The first output is the result of the command `openstack image list`, which lists several images with their IDs, names, disk formats, container formats, sizes, statuses, visibilities, protected status, owners, and tags. The second output is the result of the command `openstack image show 506e6b8e-a55d-4c78-9fa2-4fb0bb057f74`, which provides detailed information about a specific image, including its fields like checksum, container_format, created_at, disk_format, file_, id, min_disk, min_ram, name, owner, protected, schema, size, status, tags, updated_at, virtual_size, and visibility.

```

student@ubuntu14libDevStack:~/devstack$ openstack image list --long
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | Name | Disk Format | Container Format | Size | Status | Visibility | Protected | Owner | Tags |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| b392785a-7c72-47f8-8dd6-592806dc1cb | cirros-0.3.4-x86_64-uec-randisk | ari | ari | 3749163 | active | public | False | b9b43bae1bc9435d9dcfcf393654c9ddc4 |
| 37c66c39-8518-419e-02b1-6a2a0bc145bb | cirros-0.3.4-x86_64-uec | ari | ari | 25165324 | active | public | False | b9b43bae1bc9435d9dcfcf393654c9ddc4 |
| 67878c6a-febc-486c-8b71-3e5de98f5f2 | cirros-0.3.4-x86_64-uec-kernel | aki | aki | 4979632 | active | public | False | b9b43bae1bc9435d9dcfcf393654c9ddc4 |
| d8857hra-8dd3-4ad5-80a4-272a740a5f0f | Fedora-Cloud-Base-23-20151030.x86_64 | qcow2 | bare | 234363392 | active | public | False | b9b43bae1bc9435d9dcfcf393654c9ddc4 |
| 506e6b8e-a55d-4c78-9fa2-4fb0bb057f74 | cirros-0.3.2-x86_64-disk | qcow2 | bare | 13167616 | active | public | False | b9b43bae1bc9435d9dcfcf393654c9ddc4 |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
student@ubuntu14libDevStack:~/devstack$ openstack image show 506e6b8e-a55d-4c78-9fa2-4fb0bb057f74
+-----+-----+
| Field | Value |
+-----+-----+
| checksum | 64d7c1cd2b6f60c92c14662941cb7913 |
| container_format | bare |
| created_at | 2016-01-18T17:40:55Z |
| disk_format | qcow2 |
| file_ | /v2/images/506e6b8e-a55d-4c78-9fa2-4fb0bb057f74/file |
| id | 506e6b8e-a55d-4c78-9fa2-4fb0bb057f74 |
| min_disk | 0 |
| min_ram | 0 |
| name | cirros-0.3.2-x86_64-disk |
| owner | b9b43bae1bc9435d9dcfcf393654c9ddc4 |
| protected | False |
| schema | /v2/schemas/image |
| size | 13167616 |
| status | active |
| tags | |
| updated_at | 2016-01-18T17:40:57Z |
| virtual_size | None |
| visibility | public |
+-----+-----+

```

Using the `--long` parameter on the command line shows additional useful information

You will use this image when you create your example instance

Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

24

Gathering information required to create an instance—Available images

Nova is using images that are handled by Glance to build new instances. To create an instance, it is necessary to specify the instance. OpenStack® supports instance IDs or instance name parameters. If you want to use instance names instead of IDs, make sure that names are unique.

Gathering information required to create an instance

Available flavors

- The following commands list the available flavors and details about a specific flavor

The diagram illustrates the relationship between two OpenStack commands: `openstack flavor list` and `openstack flavor show <flavor id>`.

openstack flavor list

```
student@ubuntu14libDevStack:~/devstack$ openstack flavor list
+---+---+---+---+---+---+
| ID | Name | RAM | Disk | Ephemeral | VCPUs | Is Public |
+---+---+---+---+---+---+
| 1 | m1.tiny | 512 | 1 | 0 | 1 | True
| 2 | m1.small | 2048 | 20 | 0 | 1 | True
| 3 | m1.medium | 4096 | 40 | 0 | 2 | True
| 4 | m1.large | 8192 | 80 | 0 | 4 | True
| 42 | m1.nano | 64 | 0 | 0 | 1 | True
| 451 | m1.heat | 512 | 0 | 0 | 1 | True
| 5 | m1.xlarge | 16384 | 160 | 0 | 8 | True
| 84 | m1.micro | 128 | 0 | 0 | 1 | True
+---+---+---+---+---+---+
```

openstack flavor show <flavor id>

```
student@ubuntu14libDevStack:~/devstack$ openstack flavor show 1
+-----+-----+
| Field | Value |
+-----+-----+
| OS-FLV-DISABLED:disabled | False |
| OS-FLV-EXT-DATA:ephemeral | 0 |
| disk | 1 |
| id | 1 |
| name | m1.tiny |
| os-flavor-access:is_public | True |
| properties | |
| ram | 512 |
| rxtx_factor | 1.0 |
| swap | |
| vcpus | 1 |
+-----+-----+
```

A callout box points from the output of `openstack flavor show 1` to a note: "When this flavor is specified for the instance, that instance will have these hardware attributes (e.g. 1GB disk, 512KB RAM)".

Gathering information required to create an instance—Available flavors

Another parameter required for instance creation is the instance flavor. Flavors are an available hardware configuration for a server, with each flavor having a unique combination of disk space, memory capacity, and priority for CPU time. Administrator can modify the flavor to configure the minimum required amount of RAM and disk space. Flavors also support the metadata configuration, which can help the Nova filter find a compute host suitable to run the instance based on metadata information. Some of the examples include the metadata that specifies the CPU configuration, network speed, and so on.

Gathering information required to create an instance

Network

- If there are more than two networks available, you will have to specify to which one the instance will be directly attached

```
openstack network list
```

ID	Name	Subnets
acb6fc86-c5a6-46dc-8108-0cbc5bf43139	public	249db604-55c8-4909-9a4c-490a3ccf7571, e479248d-a461-4dd3-89e2-99fe6bf22216
342892c4-6883-4176-8794-2c51876e71d1	private	2476e7bd-e40e-44e8-b173-3d197be365b0, 7b1bc7d9-8a3c-45f6-8da3-06d1b179027a

```
openstack network show <network id>
```

Field	Value
id	342892c4-6883-4176-8794-2c51876e71d1
mtu	0
name	private
port_security_enabled	True
project_id	6189927cd823412aa05c894190f0:638
provider:network_type	vxlan
provider:physical_network	None
provider:segmentation_id	1066
router_type	Internal
shared	False
state	UP
status	ACTIVE
subnets	2476e7bd-e40e-44e8-b173-3d197be365b0, 7b1bc7d9-8a3c-45f6-8da3-06d1b179027a

You will use this one

Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

26

Gathering information required to create an instance—Network

Every instance requires at least one private (project) network to connect to. Networks are handled by Neutron and can be listed using the `openstack network list` command. Associated subnet should have DHCP enabled so that the new instance can be assigned with the IP address automatically.

Creating an instance

- To create new instance, use the `openstack server create` command

```
openstack server create \
--flavor <flavor ID> \
--key-name <key> \
--image <image ID> \
--nic net-id=<subnet-id> <instance name>
```



© Copyright 2017 Hewlett Packard Enterprise Development LP

27

Gathering information required to create an instance—Network

To create the new instance, run the `openstack server create` command, providing the previously collected information. It is generally recommended to provide IDs to the openstack commands. You can use the resource names (network, key, flavor, image...) as long as their names are unique over the stack.

Verify that the instance was created properly

```
openstack server list
student@ubuntu14libDevStack:~/devstack$ openstack server list
+-----+-----+-----+
| ID   | Name  | Status | Networks
+-----+-----+-----+
| aaa7c49c-51ba-437a-8db5-f5f97dc4ea06 | testinstance | ACTIVE | private=10.0.0.8, fd5c:2366:79cd:0:f816:3eff:fece:a13a |
+-----+-----+-----+
```

– Verify that the status is now active and that the parameters are correct

```
openstack server show [server id]
student@ubuntu14libDevStack:~/devstack$ openstack server show aaa7c49c-51ba-437a-8db5-f5f97dc4ea06
+-----+-----+
| Field          | Value
+-----+-----+
| OS-DCF:diskConfig | MANUAL
| OS-EXT-AZ:availability_zone | nova
| OS-EXT-SRV-ATTR:host | ubuntu14libDevStack
| OS-EXT-SRV-ATTR:hypervisor_hostname | ubuntu14libDevStack
| OS-EXT-SRV-ATTR:instance_name | instance-00000012
| OS-EXT-STS:power_state | 1
| OS-EXT-STS:task_state | None
| OS-EXT-STS:vm_state | active
| OS-EXT-SRV-ATTR:last_launched_at | 2016-01-19T15:54:24.000000
| OS-SRV-USG:terminated_at | None
| accessIPv4 | private=10.0.0.8, fd5c:2366:79cd:0:f816:3eff:fece:a13a
| accessIPv6 | None
| config_drive | True
| created | 2016-01-19T15:54:18Z
| flavor | m1.tiny (1)
| hostId | 172.16.1.15:49a539a469b51c599787801f8fc050c60146c43e66f46
| id | aaa7c49c-51ba-437a-8db5-f5f97dc4ea06
| image | cirros-0.3.2-x86_64-disk (500e0b8e-a55d-4c70-9fa2-4fb0bb057f74)
| key_name | None
| metadata | None
| os-extended-volumes:volumes_attached | []
| project_id | b9b438ae1c9435d9dcf393654c9ddc4
| properties | [{"u'name': u'default"}]
| status | ACTIVE
| updated | 2016-01-19T15:54:24Z
| user_id | 4c47a9f864ab46c9933f0045f146695e
+-----+-----+
```

Hewlett Packard
Enterprise

28

Verify that the instance was created properly

The `openstack server list` command enables you to verify that the instance was properly created, that it has a status of ACTIVE, and a Power State of Running. To display all running instances, enter the `openstack server list` command.

The `openstack server show` command displays details about the instance.

Using instance-specific data

SSH key pair

- Key pairs provide an SSH login to a running instance, so by including a key pair value when creating an instance you will be able to login to the instance from a secure SSH session

The screenshot shows a terminal window with two command-line outputs. The top output is the result of the command `openstack keypair list`, which lists a key pair named "testkey". A callout box points to this entry with the text "An SSH key pair was created earlier". The bottom output is the result of the command `openstack server create --image cirros-0.3.2-x86_64-disk --flavor m1.tiny --key-name testkey --nic net-id=private testinstance`. A callout box points to the `--key-name` option with the text "The instance created with the --key-name variable on the command line". Another callout box points to the "key" field in the instance details with the text "The key has been added to the instance".

```

student@ubuntu14l01:~/devstack$ openstack keypair list
+----+-----+
| Name | Fingerprint |
+----+-----+
| testkey | 2333cd5e-f05e-48b5-957d-abafff2558825 |
+----+-----+
student@ubuntu14l01:~/devstack$ openstack server create --image cirros-0.3.2-x86_64-disk --flavor m1.tiny --key-name testkey --nic net-id=private testinstance
+----+-----+
| Field | Value |
+----+-----+
| OS-DCF:diskConfig | MANUAL |
| OS-EXT-AZ:availability_zone | None |
| OS-EXT-SRV-ATTR:host | None |
| OS-EXT-SRV-ATTR:hypervisor_hostname | instance-00000013 |
| OS-EXT-SRV-ATTR:instance_name | 0 |
| OS-EXT-STS:power_state | scheduling |
| OS-EXT-STS:task_state | building |
| OS-SRV-USG:launched_at | None |
| OS-SRV-USG:terminated_at | None |
| accessIPv4 | ebWohz25W57Pf |
| addresses | 2016-01-19T21:38:42Z |
| adminPass | m1.tiny (1) |
| created | 2333cd5e-f05e-48b5-957d-abafff2558825 |
| flavor | cirros-0.3.2-x86_64-disk |
| hostId | 2-4fb0bb057f74 |
| id | testkey |
| image | testinstance |
| key_name | [] |
| os-extended-volumes:volumes_attached | n |
+----+-----+

```

Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

29

Using instance-specific data—SSH key pair

Many OpenStack® images available for download provide the SSH key-based authentication as only means of accessing the instance over the SSH. When you boot a virtual image for the first time, the private key of your key pair is added to `authorized_keys` file of the login account. Some images have built-in accounts created with associated passwords. However, because images are often shared by many users, it is not advisable to put passwords in the images. Nova supports injecting SSH keys into instances before they are booted. This enables you to securely log in to the instances that you create.

Nova generates a public and a private key pair, and sends the private key to the user. The public key is stored so that it can be injected into instances.

To create a key pair from the CLI of a Linux system, enter the following commands:

- `$ ssh-keygen`
- `$ nova keypair-add --pub_key id_rsa.pub testkey`

Using instance-specific data

Inserting --user-data and --file into an instance

- User Data is a special key in the metadata service which holds a file that the cloud-aware applications within the guest instance can access

```
openstack server create --image <image> --flavor <flavor> --user-data <file name> <server name>
student@ubuntu14libDevStack:~/devstack$ openstack server create --image cirros-0.3.2-x86_64-disk --flavor 1 --nic net-id=private --user-data testdatafile testinstance
| Field | Value |
| OS-DCF:diskConfig | MANUAL |
| OS-EXT-AZ:availability_zone | None |
| OS-EXT-SRV-ATTR:host | None |
| OS-EXT-SRV-ATTR:hyperervisor_hostname | None |
| OS-EXT-SRV-ATTR:instance_name | instance-00000014 |
| OS-EXT-STS:power_state | 0 |
| OS-EXT-STS:task_state | building |
| OS-EXT-STS:vm_state | None |
| OS-SRV-USG:launched_at | None |
| OS-SRV-USG:terminated_at | None |
| accessIPv4 | None |
| accessIPv6 | None |
| addresses | None |
| config_drive | Rj5nnkk66x5D |
```

The instance created with the `--user-data` variable on the command line

```
openstack server create --image <image> --flavor <flavor> --file <dest-filename>=<source-filename> <server name>
student@ubuntu14libDevStack:~/devstack$ openstack server create --image cirros-0.3.2-x86_64-disk --flavor 1 --nic net-id=private --file /root/newdata=testdatafile testinstance
| Field | Value |
| OS-DCF:diskConfig | MANUAL |
| OS-EXT-AZ:availability_zone | None |
| OS-EXT-SRV-ATTR:host | None |
| OS-EXT-SRV-ATTR:hyperervisor_hostname | None |
| OS-EXT-SRV-ATTR:instance_name | instance-00000015 |
| OS-EXT-STS:power_state | 0 |
| OS-EXT-STS:task_state | building |
| OS-EXT-STS:vm_state | None |
| OS-SRV-USG:launched_at | None |
```

Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

30

Using instance-specific data—Inserting --user-data and --file into an instance

User data is a special key in the metadata service which holds a file that cloud-aware applications within the guest instance can access. For example, the cloudinit system is an open source package from Ubuntu that handles early initialization of a cloud instance that makes use of this user data.

Arbitrary local files can also be placed into the instance file system at the creation time using the `--file <dst-path>=<src-path>` option. You can store up to 5 files.

Creating an instance in the Horizon UI (1 of 2)

The screenshot shows the OpenStack Horizon UI for the Nova service. The left sidebar has a 'Compute' section with 'Overview', 'Instances', 'Volumes', and 'Images' (which is highlighted with a green circle labeled '1'). Below that are 'Access & Security', 'Network', 'Orchestration', and 'Object Store'. The main content area is titled 'Images' and shows a table with four items:

	Name	Type	Status	Visibility	Protected	Action
<input type="checkbox"/>	> cirros-0.3.4-x86_64-uec	Image	Active	Public	No	<button>Launch</button>
<input type="checkbox"/>	> cirros-0.3.4-x86_64-uec-kernel	Image	Active	Public	No	<button>Launch</button>
<input type="checkbox"/>	> cirros-0.3.4-x86_64-uec-ramdisk	Image	Active	Public	No	<button>Launch</button>
<input type="checkbox"/>	> Fedora-Cloud-Base-25-1.3-x86_64	Image	Active	Public	No	<button>Launch</button>

At the bottom, it says 'Displaying 4 items'.

Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

31

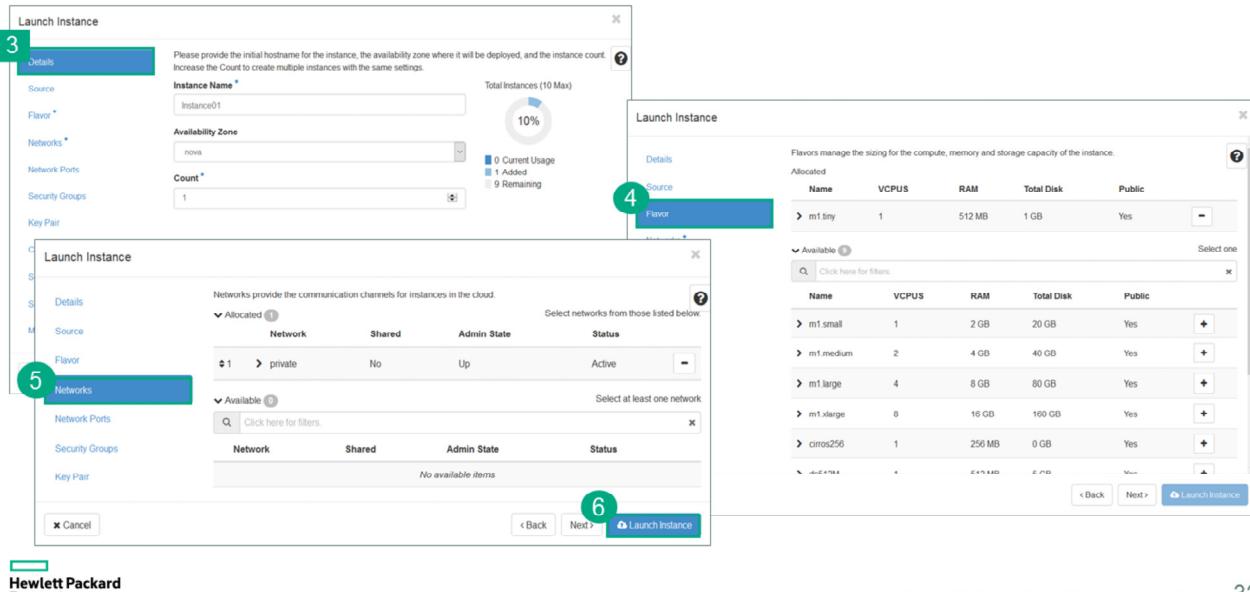
Creating an instance in the Horizon UI (1 of 2)

To create a new instance using the GUI:

1. On the left side of the Horizon UI screen, select **Project** → **Compute** → **Images**.
2. From the list, select the image from which you want to create an instance and click **Launch**.

(continued on the next page)

Creating an instance in the Horizon UI (2 of 2)



© Copyright 2017 Hewlett Packard Enterprise Development LP

32

Creating an instance in the Horizon UI (2 of 2)

(continued from the previous page)

3. Click on the **Details** tab. Complete the selection by providing the name for the instance. In addition to the different options, the Horizon UI provides you with a visual representation of project limits, including the number of created instances, the number of used vCPUs, and the total amount of RAM memory.
4. Click the **Flavor** tab and select one of available flavor for the new instance.
5. Click the **Network** tab and select the key pair to be used. Click the **Networking** tab and select the network on which the instance will be created
6. Click **Launch**.

The list of parameters on the left-hand side of the Launch instance dialog has the asterisk symbol for mandatory pages. Accessing the Access and Security page is not mandatory, since the instance can be brought up without an SSH key, but without the key, you will not be able to SSH to the instance.

Viewing instance details in the Horizon UI

From the Horizon UI:

1. Select your project
2. Click the **Instances** tab
3. Click the instance name

The Instances screen shows a table with columns: Instance Name, Image Name, and IP Address. A row for 'testinstance' is selected, highlighted with a green circle containing the number 3.

The Instance Details screen for 'testserver' displays the following information:

Instance Overview	
Name	testserver
ID	f3d280d6-0c21-472d-9f09-11410d1d0d25
Status	Active
Availability Zone	nova
Created	Jan. 20, 2016, 2:06 a.m.
Time Since Created	15 hours, 4 minutes
Host	ubuntu14libDevStack
Specs	
Flavor	m1.tiny
Flavor ID	1
RAM	512MB
VCPUs	1 VCPU
Disk	1GB
IP Addresses	
Private	10.0.0.13, 172.24.4.3

A callout box points to the 'Log', 'Console', and 'Action Log' tabs at the top of the details screen, with the text: "In addition to details about the instance, these links enable you to view the installation log, the instance console, and the actions taken".

Hewlett Packard Enterprise Development LP © Copyright 2017 Hewlett Packard Enterprise Development LP

33

Viewing instance details in the Horizon UI

Although details about instances can be gathered using the CLI, the Horizon UI provides a comfortable way to explore the instance details. To view the instance details, perform the following steps:

1. Log in to Horizon using the appropriate user name and password.
2. Select your project from the top of the Horizon UI.
3. On the left side of the screen, select **Project → Instances**.
4. Click the instance name to open the Instance Details screen.

On the Instance Details screen, you can access the console of the selected instance and view the Action log that contains a list of all actions pertaining to the selected instance.

Managing security groups from the Horizon UI

- A security group is a set of rules that define which traffic is allowed to the instance assigned to that security group
 - The default security group does not allow traffic to an instance

The screenshot shows the OpenStack Compute Access & Security interface. A green box highlights the 'Access & Security' tab. Step 1: The 'Compute' tab is selected. Step 2: The 'Security Groups' tab is selected. Step 3: A new security group named 'Default security group' is listed. Step 4: The 'Manage Rules' button is highlighted. Step 5: A new rule is being added for 'Custom TCP Rule'. Step 6: The 'Add Rule' button is highlighted.

SSH (port 22) traffic can now access the instance

Manage Security Group Rules: default (a41870aa-29f9-45c8-9c4f-2774eacceea5)

Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Actions
Egress	IPv4	Any	Any	0.0.0.0/0	-	<input type="button" value="Delete Rule"/> <input type="button" value="Edit Rule"/>
Egress	IPv6	Any	Any	/0	-	<input type="button" value="Delete Rule"/> <input type="button" value="Edit Rule"/>
Ingress	IPv6	Any	Any	-	default	<input type="button" value="Delete Rule"/> <input type="button" value="Edit Rule"/>
Ingress	IPv4	Any	Any	-	default	<input type="button" value="Delete Rule"/> <input type="button" value="Edit Rule"/>
Ingress	IPv4	ICMP	Any	0.0.0.0/0	-	<input type="button" value="Delete Rule"/> <input type="button" value="Edit Rule"/>
Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-	<input type="button" value="Delete Rule"/> <input type="button" value="Edit Rule"/>

Manage Security Group Rules: default (a41870aa-29f9-45c8-9c4f-2774eacceea5)

Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Actions
Egress	IPv4	Any	Any	0.0.0.0/0	-	<input type="button" value="Delete Rule"/> <input type="button" value="Edit Rule"/>
Egress	IPv6	Any	Any	/0	-	<input type="button" value="Delete Rule"/> <input type="button" value="Edit Rule"/>
Ingress	IPv6	Any	Any	-	default	<input type="button" value="Delete Rule"/> <input type="button" value="Edit Rule"/>
Ingress	IPv4	Any	Any	-	default	<input type="button" value="Delete Rule"/> <input type="button" value="Edit Rule"/>
Ingress	IPv4	ICMP	Any	0.0.0.0/0	-	<input type="button" value="Delete Rule"/> <input type="button" value="Edit Rule"/>
Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-	<input type="button" value="Delete Rule"/> <input type="button" value="Edit Rule"/>

Add Rule

Custom TCP Rule

Description:
Rules define what traffic is allowed to instances assigned to the security group. A security group rule consists of three main parts: direction, port range, and remote IP prefix.

Direction: Ingress

Open Port: Port 22

Remote: CIDR 0.0.0.0/0

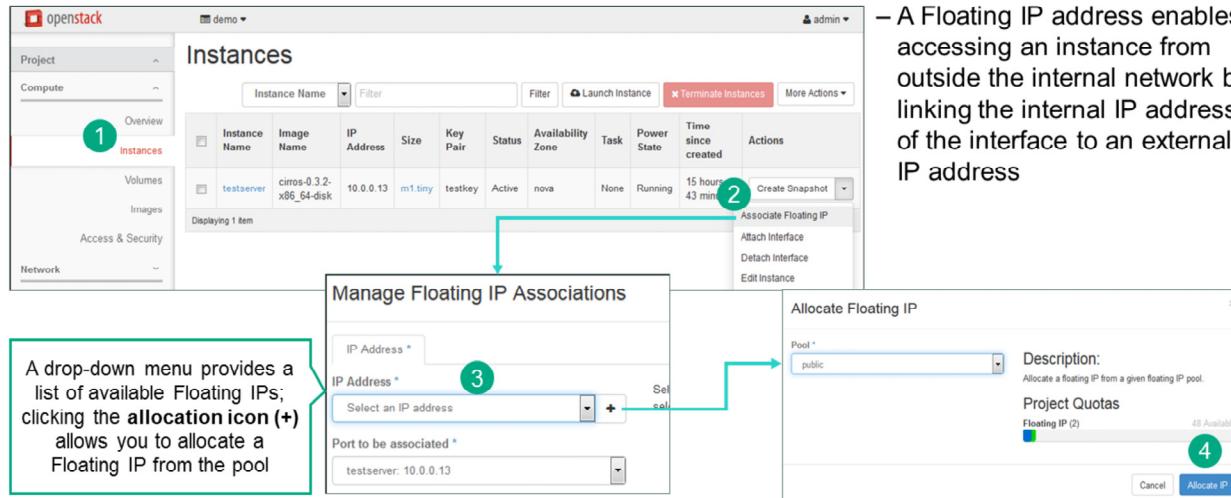
Buttons:

Managing security groups from the Horizon UI

To add rules to an existing security group using the Horizon UI, perform the following steps:

1. Click the **Project** → **Compute** → **Access & Security** link.
 2. On the Access & Security page, select the **Security Groups** tab.
 3. Select the **Manage Rules** button associated with the security group that is modified.
 4. On the Manage Security Group Rules screen, click the **Add Rule** button.
 5. At the Add Rule page, fill in the required fields.
 6. Click **Add** to save the rule.

Assigning a Floating IP address to an instance (1 of 2)



Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

35

Assigning a Floating IP address to an instance (1 of 2)

To allocate a Floating IP from the Floating IP pool and assign it to an instance using the Horizon UI, follow the steps listed on this and the next page:

1. Click **Project** → **Compute** → **Instances**.
2. At the end of the row associated with the instance to which you are assigning the Floating IP address, click the drop-down list arrow.
3. On the Manage Floating IP Associations screen, click the **allocation icon (+)** to display the Allocate Floating IP screen.
4. Select the pool from which the IP will be selected and then click **Allocate IP**.

(continued on the next page)

Assigning a Floating IP address to an instance (2 of 2)

The screenshot shows two windows. On the left is the 'Manage Floating IP Associations' dialog, which has fields for 'IP Address' (172.24.4.16) and 'Port to be associated' (testserver: 10.0.0.13). A callout box labeled 5 points to the IP address field with the instruction: 'Verify that this is the local IP address of the instance to which you are assigning the Floating IP address'. A green arrow points from this box to the IP address field. A callout box labeled 6 points to the 'Associate' button. On the right is the 'Instances' list page, which shows one item: testserver (cirros-0.3.2-x86_64-disk), IP Address 10.0.0.13, Floating IPs 172.24.4.16, m1.tiny key pair, testkey key pair, and Active status. A callout box labeled 7 points to the 'Floating IPs' column value 172.24.4.16 with the instruction: 'Verify the Floating IP is associated with the instance'. A red box highlights this value.

Instance Name	Image Name	IP Address	Size	Key Pair	Status
testserver	cirros-0.3.2-x86_64-disk	10.0.0.13 Floating IPs: 172.24.4.16	m1.tiny	testkey	Active

36

Assigning a Floating IP address to an instance (2 of 2)

(continued from the previous page)

5. Verify that the allocated IP appears in the IP address field and that the correct port of the instance to be associated to the Floating IP is selected.
6. Click **Associate**.
7. Verify that the Floating IP has been assigned to the instance.

Connecting to an instance

- You can connect to an instance from the command line or through the Horizon dashboard

The screenshot shows the OpenStack Horizon dashboard with the 'Instances' tab selected. On the left sidebar, 'Compute' is expanded, showing 'Instances'. In the main area, there is one instance listed:

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
testserver	cirros-0.3.2-x86_64-disk	10.0.0.13 Floating IPs: 172.24.4.3		m1.tiny testkey	Active	nova		None Running	15 hours, 12 minutes	Create Snapshot Disassociate Floating IP Attach Interface Detach Interface Edit Instance Edit Security Groups Console

A context menu is open over the 'testserver' instance, with a green arrow pointing to the 'Console' option. The menu options are: Disassociate Floating IP, Attach Interface, Detach Interface, Edit Instance, Edit Security Groups, and Console.

Hewlett Packard Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

37

Connecting to an instance

Connecting to your instance through the dashboard requires that the novnc service be running on the compute node. Access through the command line is not straightforward because the instance is running on a private network that is hidden behind the router. The best way to access the server is to assign a public IP address to it, which can be done easily in the dashboard.

To access the console in the dashboard UI:

1. Log in to Horizon.
2. In the left side of the screen, select **Project** → **Compute** → **Instances**.
3. Find the instance you want to connect to and click the down arrow next to the Create Snapshot option.
4. From the drop-down menu, select **Console**.

Managing instances from the Horizon UI

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions	
testserver	cimc-0.3.2-x86_64-disk	10.0.0.13	Floating IP	m1.tiny	testkey	Active	nova	None	Running	18 hours, 9 minutes	Create Snapshot

- **Snapshot an Instance:** Captures an image of a running instance without the need to pause that instance
 - The state of file system is captured, but not that of the memory, so it may be necessary to quiesce the instance
- **Pause Instance:** Stores the state of the VM in RAM; a paused instance continues to run in a frozen state
- **Suspend Instance:** Initiates a hypervisor-level suspend operation
- **Shelve Instance:** Shuts down the instance and stores it together with associated data and resources (a snapshot is taken, if not volume-backed); the instance memory is lost
- **Lock Instance:** Prevents a regular (non-admin) user from executing actions on that instance

Managing instances from the Horizon UI

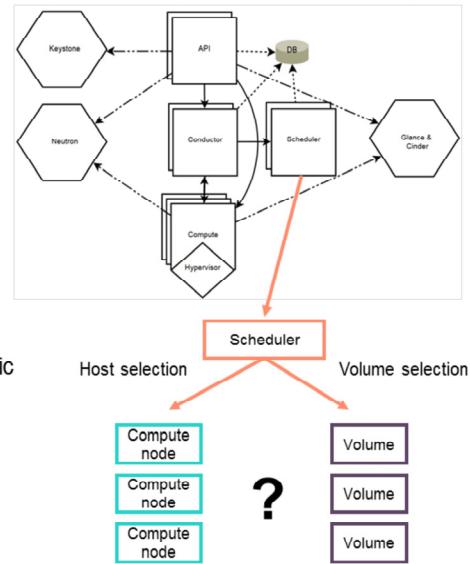
The actions that can be taken on the selected instance in the Horizon UI are displayed on the slide above.



Nova scheduler

Nova scheduler overview

- Nova-scheduler is a daemon which determines how to dispatch compute requests
 - Determines which host (hypervisor) a VM should launch on
 - Is configurable through a variety of options
- A scheme of configurable filters and weights enable Nova to automatically choose the best host or volume
- Example nova-scheduler use cases:
 - Associate a VM to particular host, that is, forward the request to a specific hypervisor
 - Provision VMs of a particular tenant to isolated hypervisors (this is applicable when dedicated hypervisor hosts are preferred)
 - Provision all VMs to different hypervisor hosts
 - Provision VMs to the hypervisor hosts which can support the request



Nova scheduler overview

Nova uses the nova-scheduler service to determine how to dispatch compute and volume requests. For example, the nova-scheduler service determines on which host a VM should launch. The term “host” in the context of filters means a physical node that has a nova-compute service running on it. The scheduler is configurable through a variety of options.

As shown in the graphic, the nova-scheduler interacts with other components through a queue and central database repository. All compute nodes periodically publish their status, available resources, and hardware capabilities to the nova-scheduler through the queue. The nova-scheduler then collects this data and uses it to make decisions when a request comes in. For scheduling, the queue is the essential communication hub.

Nova-scheduler scheduling algorithms

Type of algorithm	Behavior
Chance	Selects a random host from those available
Filter (not used for volume requests)	Selects a host that best meets the specified filter criteria
Multi	Calls multiple sub-schedulers

- Compute node selection:
 - By default, Compute is configured to use the Filter Scheduler
 - By default, the Filter Scheduler selects the host with the maximum amount of RAM available
- The volume scheduler is configured by default to use the Chance Scheduler, which randomly selects a host that has the nova-volume service running



© Copyright 2017 Hewlett Packard Enterprise Development LP

41

Nova-scheduler scheduling algorithms

Compute is configured by default in `/etc/nova/nova.conf` to use the Filter Scheduler, which allows the admin to specify different scheduling behavior for compute requests versus volume requests.

The compute scheduler is configured by default as a Filter Scheduler. In the default configuration, this scheduler will only consider hosts that are in the requested availability zone (`AvailabilityZoneFilter`), that have sufficient RAM available (`RamFilter`), and that are actually capable of servicing the request (`ComputeFilter`).

From the resulting filtered list of eligible hosts, the scheduler assigns a cost to each host based on the amount of free RAM (`nova.scheduler.least_cost.compute_fill_first_cost_fn`), multiplies each cost value by -1 (`compute_fill_first_cost_fn_weight`), and selects the host with the minimum cost. This is equivalent to selecting the host with the maximum amount of RAM available.

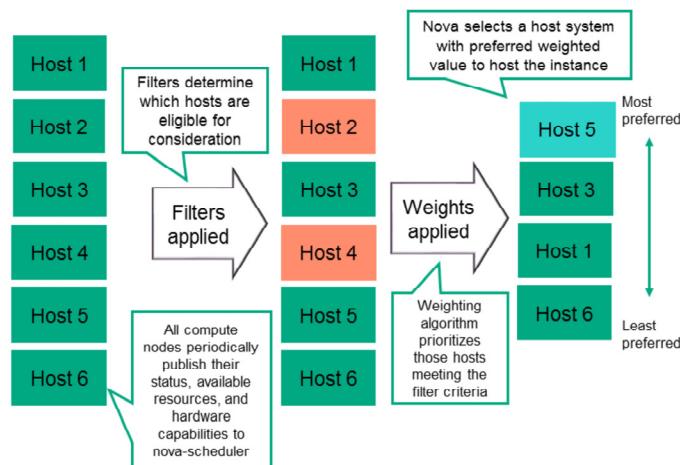
The volume scheduler is configured by default to use the Chance Scheduler, which randomly selects a host that has the nova-volume service running:

- `scheduler_driver=nova.scheduler.multi.MultiScheduler`
- `volume_scheduler_driver=nova.scheduler.chance.ChanceScheduler`
- `compute_scheduler_driver=nova.scheduler.filter_scheduler.FilterScheduler`
- `scheduler_available_filters=nova.scheduler.filters.standard_filters`
- `scheduler_default_filters=AvailabilityZoneFilter,RamFilter,ComputeFilter`
- `least_cost_functions=nova.scheduler.least_cost.compute_fill_first_cost_fn`
- `compute_fill_first_cost_fn_weight=-1.0`

Filter scheduler

Filters and weights

- **Filters** determine which hosts are eligible when dispatching a compute instance
 - Filters are binary—A host is accepted or rejected
 - Filters are written in Python and are pluggable
 - Some filters include: RamFilter, ComputeFilter, ComputeCapabilitiesFilter, AvailabilityZoneFilter, MetricsFilter
- **Weighers** are weighting algorithms that decide the order in which hosts meeting the filtration will be considered
 - Weighers are Python modules and are pluggable:
 - **RAMWeigher**—Weighs hosts by their available RAM
 - **IoOpsWeigher**—Weighs hosts by their io Ops number
 - **MetricsWeigher**—Allows individual metrics to be given a weight



NOTE: For more information, go to
<http://docs.openstack.org/newton/config-reference/compute/schedulers.html> or
https://01.org/sites/default/files/utilization_based_scheduling_in_openstack_compute_nova.pdf

Filter scheduler—Filters and weights

After filters are applied, the scheduler applies one or more cost functions to each remaining host to compute scores.

Each cost score is multiplied by a weighting constant specified in `nova.conf`. All weights are normalized before being summed up; the host with the largest weight is given the highest priority.

Scheduler and weight normalization was added in the Icehouse release (<https://review.openstack.org/#c/27160/>). In previous releases, the Compute and Cells scheduler used raw weights (that is, the weighers returned any value, and that was the value used by the weighing process).

The weights for a host are now always within the range of 0.0 to 1.0.

Nova-scheduler filters

- These are some of the filter options available (OpenStack® Liberty includes over 30 configurable filters)

Filter option	Description
AffinityFilter	Same host or different host
AvailabilityZoneFilter	Choose least cost inside the selected availability zone
Core	Choose least CPU core utilization
RamFilter	Return hosts with sufficient RAM
ComputeFilter	Returns hosts where requested <code>instance_type</code> matches capabilities
JSON	Allows simple JSON based grammar to configure custom scheduling <code>\$ nova boot --image 827d564a-e636-4fc4-a376-d36f7ebe1747 -flavor 1 \--hint query='[{">=", "\$free_ram_mb", 1024}]' server1</code>

NOTE: For more information on filters, go to <http://docs.openstack.org/newton/config-reference/compute/schedulers.html>.



© Copyright 2017 Hewlett Packard Enterprise Development LP

43

Nova-scheduler filters

Typically, an OpenStack® installation has a large number of compute nodes. Affinity scheduling enables you to launch an instance that is near (or not near) other specified instances. This provides more control over where a class of VMs is running in the cloud. This is accomplished using server instance groups.

Default nova-scheduler configuration

– Compute is configured in the `/etc/nova/nova.conf` file with the following default scheduler options:

- `scheduler_driver=nova.scheduler.multi.MultiScheduler`
- `scheduler_driver_task_period = 60`
- `scheduler_driver = nova.scheduler.filter_scheduler.FilterScheduler`
- `scheduler_available_filters = nova.scheduler.filters.all_filters`
- `scheduler_default_filters = RetryFilter, AvailabilityZoneFilter, RamFilter, ComputeFilter, ComputeCapabilitiesFilter, ImagePropertiesFilter, ServerGroupAntiAffinityFilter, ServerGroupAffinityFilter`

Uses multiple filter options

Filter options used by the default scheduling algorithm

NOTE: For more information go to <http://docs.openstack.org/newton/config-reference/compute/schedulers.html>.



© Copyright 2017 Hewlett Packard Enterprise Development LP

44

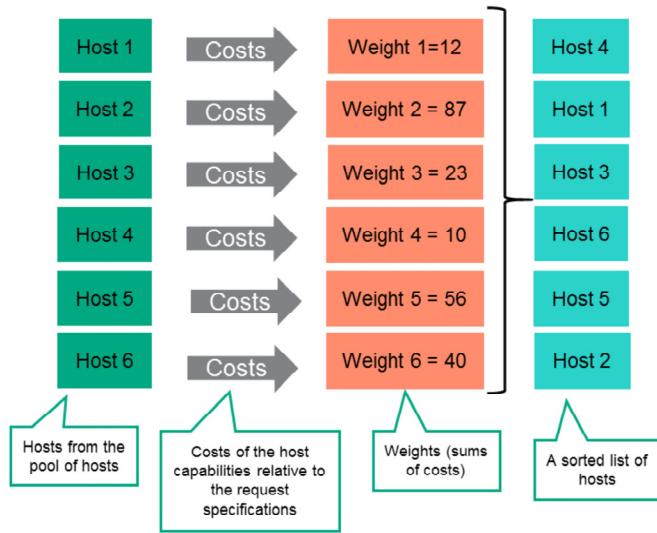
Default nova-scheduler configuration

By default, nova-scheduler considers hosts that meet all of the following criteria:

- Are in the requested availability zone (AvailabilityZoneFilter)
- Have sufficient RAM available (RamFilter)
- Are capable of servicing the request (ComputeFilter)

Nova-scheduler weights

- After filters identify the host list, the Filter Scheduler applies one or more weights (that is, cost functions) to each remaining host to compute the cost scores
- Each weigher has an associated multiplier, allowing multiple weigher costs to be combined proportionately or disproportionately
- Each weigher cost score is multiplied by the weigher's weighting constant specified in `nova.conf` and summed up by host
- The net host weight is used to determine the best host to schedule the instance to



Nova-scheduler weights

Weight normalization was added in the Icehouse release (<https://review.openstack.org/#/c/27160/>). In previous releases, the Compute and Cells scheduler used raw weights (that is, the weighers returned any value, and that was the value used by the weighing process). The weights for a host are now always within the range of 0.0 to 1.0.



Nova maintenance

Planned compute node maintenance

- These commands can be used to move instances off the compute node to be maintained:

- List all of the instances on the affected node:

```
# nova list --host <compute_host> --all-tenants
```

- If you are using shared storage, migrate all of the instances to the other compute node:

```
# nova server-migrate --live-migration <vm_uuid> <backup_compute_host>
```

- If you are not using shared storage, use the following syntax to migrate the instances:

```
# nova live-migration --block-migrate <vm_uuid> <backup_compute_host>
```

- Stop or start a compute instance:

```
# stop nova-compute or # start nova-compute
```

- Check if Nova is connected to the AMQP service after a reboot:

```
# grep AMQP /var/log/nova/nova-compute
```

- Reboot instances that are missing the libvirt XML file (/etc/libvirt/qemu/instance-xxxxxxxx.xml):

```
# nova reboot --hard <vm_uuid>
```

- Reboot instances:

```
# nova reboot <vm_uuid>
```

NOTE: These examples demonstrate the usage of the Nova CLI commands instead of the OSC OpenStack® CLI commands.


© Copyright 2017 Hewlett Packard Enterprise Development LP
47

Planned compute node maintenance

If, for any reason, you need to perform the maintenance on a compute node, follow this procedure:

1. Migrate all instances from the node you want to maintain.
2. After all of the instances have been migrated, stop the nova-compute service on the node.
3. Perform the necessary maintenance on the node.
4. Start the compute node and the nova-service.
5. Take care of instances that are hosted on the compute node that was under maintenance and could not be moved from that node when the maintenance was being performed.

The following commands can be used to maintain the instances on the compute node:

- List all of the instances on the affected node:

```
# nova list --host <compute_host> --all-tenants
```

- If you are using shared storage, migrate all of the instances to the other compute node:

```
# nova live-migration <vm_uuid> <backup_compute_host>
```

- If you are not using shared storage, enter the following syntax to migrate the instances:

```
# nova live-migration --block-migrate <vm_uuid> <backup_compute_host>
```

- Stop or start a compute instance:

```
# stop nova-compute or # start nova-compute
```

- Check if Nova is connected to the AMQP service after a reboot:

```
# grep AMQP /var/log/nova/nova-compute
```

- Reboot instances that are missing the libvirt XML file (/etc/libvirt/qemu/instance-xxxxxxxx.xml):

```
# nova reboot --hard <vm_uuid>
```

- Reboot instances:

```
# nova reboot <vm_uuid>
```

Inspecting and recovering data from failed instances (1 of 3)

1. List the available instances and suspend the instance you want to work with

```
$ virsh list
Id   Name           State
-----
5    instance-00000003 running
7    instance-00000005 running
8    instance-00000006 running

$ virsh suspend 8
Domain 8 suspended
```

2. Connect the `qemu-nbd` device to the disk

```
$ cd /opt/stack/data/nova/instances/441daf0b-766b-4c6a-ba43-44e9c5017cb6/
$ ls -lh
total 1.9M
-rw-rw--- 1 libvirt-qemu kvm      20K Dec  7 01:49 console.log
-rw-r--r-- 1 libvirt-qemu kvm     1.5M Dec  7 01:50 disk
-rw-r--r-- 1 libvirt-qemu kvm    410K Dec  6 12:53 disk.config
-rw-r--r-- 1 stack      libvирtd 176 Dec  6 12:53 disk.info
-rw-r--r-- 1 stack      libvирtd 2.6K Dec  6 12:53 libvirt.xml
$ sudo modprobe nbd
$ sudo qemu-nbd -c /dev/nbd0 $(pwd)/disk
```

Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

48

Inspecting and recovering data from failed instances (1 of 3)

In case of a kernel panic or a boot failure, instances might be running but become inaccessible through SSH and do not respond to any command. This could indicate a file system corruption on the VM. If you need to recover files or repair the instance file system, use the `qemu-nbd` command to mount the disk.

To access the disk used by the instance on the compute node (`/var/lib/nova/instances/instance-xxxxxx/disk`), follow these steps:

1. Enter the `virsh` command to suspend the instance and take note of the internal ID.
2. Connect the `qemu-nbd` device to the disk. If the kernel running on the node does not have a `/dev/nbd*` device, you must inject the appropriate driver.

Inspecting and recovering data from failed instances (2 of 3)

3. Mount the `qemu-nbd` device

```
$ sudo mount /dev/nbd0p1 /mnt/
$ ls -lh /mnt/
total 37K
drwxrwxr-x  2 root root 3.0K Mar 17  2014 bin
drwxr-xr-x  3 root root 1.0K Mar 17  2014 boot
drwxr-xr-x  3 root root 1.0K Mar 17  2014 dev
drwxrwxr-x 13 root root 1.0K Dec  6 12:53 etc
drwxrwxr-x  4 root root 1.0K Mar 17  2014 home
-rw-rxr-x  1 root root 2.6K Mar 17  2014 init
lrwxrwxrwx  1 root root   32 Mar 17  2014 initrd.img -> boot/initrd.img-3.2.0-60-virtual
drwxrwxr-x  4 root root 1.0K Mar 17  2014 lib
lrwxrwxrwx  1 root root   11 Mar 17  2014 linuxrc -> bin/busybox
drwx----- 2 root root 12K Mar 17  2014 lost+found
```

TIP: If VDA is the disk, and VDA1 is the root partition, `qemu-nbd` exports the device as `/dev/nbd0` and `/dev/nbd0p1`. To examine the secondary or ephemeral disk, enter `/dev/nbd1`, `/dev/nbd2`, and so on.

Inspecting and recovering data from failed instances (2 of 3)

The `qemu-nbd` command tries to export the different partitions of the instance disk as separate devices. The first (root) disk is mapped as `/dev/nbd0`, and the first partition on that disk as `/dev/nbd0p1`. Similarly, the secondary disk is mapped as `/dev/nbd1`, and the first partition on the secondary disk as `/dev/nbd1p1`.

Inspecting and recovering data from failed instances (3 of 3)

4. After you have completed the repair, unmount the mount point and release the `qemu-nbd` device
5. Use `virsh` to resume the instance

```
$ sudo umount /mnt
$ sudo qemu-nbd -d /dev/nbd0
/dev/nbd0 disconnected
$ virsh list
  Id   Name           State
  --  --
  5   instance-00000003    running
  7   instance-00000005    running
  8   instance-00000006    paused

$ virsh resume 8
Domain 8 resumed
```

Inspecting and recovering data from failed instances (3 of 3)

After you have completed the repair, unmount the mount point and release the `qemu-nbd` device. Then, use the `virsh` command to resume the instance.



Nova troubleshooting

Verifying the Nova installation

- Common causes of a disabled status:
 - Services run on separate hosts and there are time sync errors
 - Check the NTP synchronization
 - The service is not running
 - Start the service

```
$ openstack compute service list
+---+-----+-----+-----+-----+
| ID | Binary | Host | Zone | Status | State | Updated At |
+---+-----+-----+-----+-----+
| 5 | nova-conductor | stack | internal | enabled | up | 2017-01-30T21:31:38.000000 |
| 7 | nova-scheduler | stack | internal | enabled | up | 2017-01-30T21:31:40.000000 |
| 8 | nova-consoleauth | stack | internal | enabled | up | 2017-01-30T21:31:40.000000 |
| 9 | nova-compute | stack | nova | enabled | up | 2017-01-30T21:31:38.000000 |
+---+-----+-----+-----+-----+
```

Verifying the Nova installation

To verify if Nova is functioning, use a command line and query the service list or available images:

```
# openstack compute service list
# openstack image list
```

If the service is not responding, enter the following command to check if it started:

```
# service nova status
```

If necessary, enter this command to bring up the service:

```
# service nova start
```

A common reason of why Nova does not start might be found in time sync errors, if you have multiple hosts running the compute service. It is recommended that you configure the Network Time Protocol (NTP) service on all participating nodes.

Troubleshooting Nova—Common issues (1 of 2)

- Service endpoints and associated users were not created properly
 - The Keystone integration is incorrect
 - The Glance endpoint is not set up properly
- The hypervisor is not set up correctly—Use `virsh` tools to ensure that hypervisor is set up correctly using bridge networking
- MySQL database parameters are incorrect
- IP addresses that are provided by the network admin (public and private) are incorrect
- DNSmasq is not running
- The Nova CLI is not responding—The API is probably down
- An instance gets stuck in a scheduling state—The compute is probably down
- An instance gets stuck in a networking state—Check if the network is down



© Copyright 2017 Hewlett Packard Enterprise Development LP

53

Troubleshooting Nova—Common issues (1 of 2)

Some common issues and resolutions thereof include:

- Service endpoints and associated users were not created properly—The Keystone integration is incorrect or the Glance endpoint is not set up properly.
- The hypervisor is not set up correctly—Use `virsh` tools to ensure that the hypervisor is set up correctly using bridge networking.
- MySQL database parameters are incorrect.
- IP addresses that are provided by the network admin (public and private) are incorrect.
- DNSmasq is not running.
- The Nova CLI is not responding—The API is probably down.
- An instance gets stuck in a scheduling state—The compute is probably down.
- An instance gets stuck in a networking state—Check if the network is down.

Troubleshooting Nova—Common issues (2 of 2)

- An instance has an error in the spawning state—Check if libvirt is down
- You cannot use SSH with a public IP or private IP
 - Ensure that the addresses are legitimate and have routes
 - Enter `iptables --list`
- The console log of instances provides details with regard to networking
- Ensure that the stats for queues are legitimate: `rabbitmqctl list_queues` and `rabbitmqctl list_consumers`
 - `openstack compute service list` shows the up state for all services
 - `ps -ef | grep keystone`
 - `ps -ef | grep glance`
 - `openstack servers show --diagnostics <instance ID>`



© Copyright 2017 Hewlett Packard Enterprise Development LP

54

Troubleshooting—Common issues (2 of 2)

- An instance has an error in the spawning state—Check if libvirt is down.
- You cannot use SSH with a public IP or private IP:
 - Ensure that the addresses are legitimate and have routes.
 - Enter `iptables -list`.
- The console log of instances provides details with regard to networking.
- Ensure that the stats for queues are legitimate: `rabbitmqctl list_queues ; rabbitmqctl list_consumers`
 - `nova-manage service list` shows ☺ for all processes.
 - `ps -ef | grep keystone`
 - `ps -ef | grep glance`
 - `openstack servers show --diagnostics <instance ID>`

Troubleshooting the Nova service

- Instance and image issues
 - An instance gets stuck on a scheduled status
 - The scheduler is probably down
 - Run `openstack server list` to see if it has a state of up
 - An image is incorrect—Check the original image configuration
- Log files
 - Nova is located at `/var/log/nova` (multiple log files)
 - Instance logs are located at `/var/lib/nova/instances`



© Copyright 2017 Hewlett Packard Enterprise Development LP

55

Troubleshooting the Nova service

These are some potential instance and image issues and resolutions thereof:

- An instance gets stuck on a scheduled status:
 - The scheduler is probably down.
 - Run the `nova list` command to see if it works.
- An image is incorrect—Check the original image configuration.

In case of any inconsistency, check the information in the following Nova log files:

- Nova is located at `/var/log/nova` (multiple log files).
- Instance logs are located at `/var/lib/nova/instances`.

Module summary

In this module:

- The function and features of Nova and related terminology were discussed
- The major Nova OpenStack® Compute architectural components were identified
- The types of compartmentalization of Nova instances were discussed
- Some common Nova management tasks were completed
- The operation of the Nova scheduler was explained
- Basic troubleshooting steps for instances and the Nova service were described



© Copyright 2017 Hewlett Packard Enterprise Development LP

56

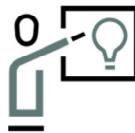
Module summary

In this module:

- The function and features of Nova and related terminology were discussed
- The major Nova OpenStack® Compute architectural components were identified
- The types of compartmentalization of Nova instances were discussed
- Some common Nova management tasks were completed
- The operation of the Nova scheduler was explained
- Basic troubleshooting steps for instances and the Nova service were described

Learning check

Learning check



OpenStack® Compute supports both VM provisioning and baremetal provisioning.

- A. True
- B. False

Learning check

OpenStack® Compute supports both VM provisioning and baremetal provisioning.

- A. True
- B. False

Learning check answer



OpenStack® Compute supports both VM provisioning and baremetal provisioning.

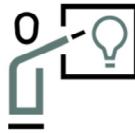
- A. True (see Ironic)
- B. False

Learning check answer

OpenStack® Compute supports both VM provisioning and baremetal provisioning.

- A. True (see Ironic)
- B. False

Learning check



Which command produces the following output?

ID	Binary	Host	Zone	Status	State	Updated At
1	nova-conductor	ubuntu14libDevStack	internal	enabled	up	2016-01-20T23:45:51.000000
2	nova-cert	ubuntu14libDevStack	internal	enabled	up	2016-01-20T23:45:50.000000
3	nova-scheduler	ubuntu14libDevStack	internal	enabled	up	2016-01-20T23:45:50.000000
4	nova-consoleauth	ubuntu14libDevStack	internal	enabled	up	2016-01-20T23:45:50.000000
5	nova-compute	ubuntu14libDevStack	nova	enabled	up	2016-01-20T23:45:52.000000

- A. openstack compute service check
- B. openstack compute status
- C. openstack compute status check
- D. openstack compute service list

Learning check

Which command produces the output given on the screenshot above?

- A. openstack compute service check
- B. openstack compute status
- C. openstack compute status check
- D. openstack compute service list

Learning check answer



Which command produces the following output?

ID	Binary	Host	Zone	Status	State	Updated At
1	nova-conductor	ubuntu14libDevStack	internal	enabled	up	2016-01-20T23:45:51.000000
2	nova-cert	ubuntu14libDevStack	internal	enabled	up	2016-01-20T23:45:50.000000
3	nova-scheduler	ubuntu14libDevStack	internal	enabled	up	2016-01-20T23:45:50.000000
4	nova-consoleauth	ubuntu14libDevStack	internal	enabled	up	2016-01-20T23:45:50.000000
5	nova-compute	ubuntu14libDevStack	nova	enabled	up	2016-01-20T23:45:52.000000

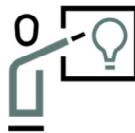
- A. openstack compute service check
- B. openstack compute status
- C. openstack compute status check
- D. **openstack compute service list**

Learning check answer

Which command produces the output given on the screenshot above?

- A. openstack compute service check
- B. openstack compute status
- C. openstack compute status check
- D. **openstack compute service list**

Learning check



What does the default security group do?

- A. Blocks all ingress packets
- B. Allows ingress packets on port 22 only
- C. Allows all ingress packets
- D. Blocks ICMP packets only

Learning check

What does the default security group do?

- A. Blocks all ingress packets
- B. Allows ingress packets on port 22 only
- C. Allows all ingress packets
- D. Blocks ICMP packets only

Learning check answer



What does the default security group do?

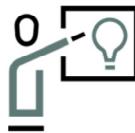
- A. Blocks all ingress packets**
- B. Allows ingress packets on port 22 only
- C. Allows all ingress packets
- D. Blocks ICMP packets only

Learning check answer

What does the default security group do?

- A. Blocks all ingress packets**
- B. Allows ingress packets on port 22 only
- C. Allows all ingress packets
- D. Blocks ICMP packets only

Learning check



What is the default scheduler for Nova VM instances?

- A. Filter Scheduler
- B. Chance Scheduler
- C. Multi Scheduler

Learning check

What is the default scheduler for Nova VM instances?

- A. Filter Scheduler
- B. Chance Scheduler
- C. Multi Scheduler

Learning check answer



What is the default scheduler for Nova VM instances?

- A. Filter Scheduler
- B. Chance Scheduler
- C. Multi Scheduler

Learning check answer

What is the default scheduler for Nova VM instances?

- A. Filter Scheduler
- B. Chance Scheduler
- C. Multi Scheduler



**Hewlett Packard
Enterprise**



Learning objectives

After completing this module, you should be able to:

- Describe the various types of OpenStack® storage
- Discuss the function and primary features of Cinder
- Provide some Cinder use cases
- Explain the primary components of the Cinder architecture
- Locate the latest list of Cinder drivers
- Describe the process of generating a Cinder volume
- Complete the common management tasks associated with Cinder
- Describe how to troubleshoot common Cinder issues



© Copyright 2017 Hewlett Packard Enterprise Development LP

2

Learning objectives

After completing this module, you should be able to:

- Describe the various types of OpenStack® storage
- Discuss function and primary features of Cinder
- Provide some Cinder use cases
- Explain the primary components of the Cinder architecture
- Locate the latest list of Cinder drivers
- Describe the process of generating a Cinder volume
- Complete the common management tasks associated with Cinder
- Describe how to troubleshoot common Cinder issues



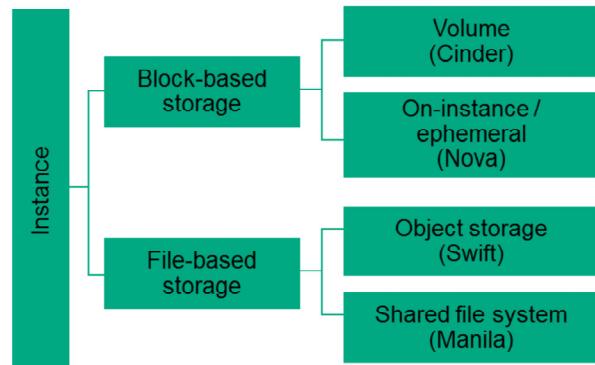
Cinder overview

OpenStack® storage overview

Storage technologies supported by OpenStack®

- On-instance / ephemeral storage:
 - Ephemeral—All of the stored information lasts as long as the image lasts, and after the image is restarted, that information is lost
 - Size of this disk is based on the image flavor
- Volume block storage:
 - Associated with the VM and is mounted every time the instance is booted
 - Storage space is controlled by means of a Cinder-supported protocol such as iSCSI or Fibre Channel
 - Persistent storage; it “survives” a reboot of the VM

NOTE: Object storage is discussed in the Swift module of this course, while shared file system is discussed in the Manila section of other OpenStack® modules of this course.



OpenStack® storage overview—Storage technologies supported by OpenStack®

When working with OpenStack®, it is important to understand the distinction between ephemeral storage and persistent storage.

If only the OpenStack® Compute service (Nova) is deployed, users do not have access to any form of persistent storage by default. The disks associated with VMs are "ephemeral," in that they effectively disappear when a virtual machine is terminated. The ephemeral storage is located at the compute node's local disk.

Persistent storage means that the storage resource outlives any other resource and is always available, regardless of the state of a running instance. A storage volume can be freely attached and detached from an instance without affecting the volume contents.

OpenStack® supports two types of persistent storage: object storage and block storage.

- **Block storage** (sometimes referred to as volume storage) provides access to block-storage devices. You can interact with block storage by attaching volumes to your running VM instances. These volumes are persistent; they can be detached from one instance and re-attached to another, and data remains intact. Block storage is implemented in OpenStack® by the OpenStack® Block Storage project (Cinder).
- **Object storage** allows you to store binary objects in an object store and access them by using a REST API. This capability is provided by the OpenStack® Swift project.

Manila is an OpenStack® service used to present the management of file shares (for example, NFS and CIFS) as a core service to OpenStack®.

Comparison between the OpenStack® storage technologies

- Comparison between the OpenStack® storage technologies

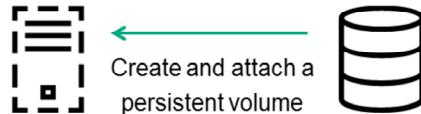
Ephemeral (Nova)	Volumes block storage (Cinder)	Object storage (Swift)	Shared filesystem (Manilla)
Block-based	Block-based	File-based	File-based (forked from Cinder)
Compute node local disk	Storage backend	Cluster of commodity systems	Storage backend
OS, scratch, swap space	Persistent volumes for VMs	VM images and data	Shared distributed filesystem
Persists until VM is terminated	Persists until deleted	Persists until deleted	Persists until deleted
Access associated with VM	Access associated with VM	Available from anywhere	Available from associated VMs or via API
Implemented as filesystem	Mounted via Cinder controlled protocol (iSCSI, FC, etc.)	REST API	Mounted via Manilla controlled protocol (iSCSI, FC, etc.)
Size settings based on flavors	Sizings based on need	Easily scalable for future growth	Sizing based on need
Example: 10GB first disk, 30GB/core second disk	Example: 1TB extra hard drive	Example: 10s of TBs of dataset storage	Example 3GB corporate document directory tree

Comparison between the OpenStack® storage technologies

The table above provides comparison between ephemeral, block, object and filesystem types of storage, along with some usage examples.

OpenStack® block storage (Cinder)

- Cinder is the block storage component of OpenStack®
 - Provides persistent volumes for instances
 - Supports storage protocols, such as iSCSI and FC
 - Does not provide a shared storage solution like NFS
- State of these volumes are independent of instances
 - By default, the primary OS volume does not come from Cinder, but rather from ephemeral storage
 - You can create a Cinder boot volume
- Supports the use of multiple storage providers within a zone
 - May be useful for Information Lifecycle Management Strategy
 - Volumes can be backed up to Swift and different types of storage (for example, Ceph Object Storage, IBM Tivoli Storage Manager)



OpenStack® block storage (Cinder)

Cinder is the block storage component of OpenStack® that provides persistent volumes for instances. It supports storage protocols, such as iSCSI and FC, but it does not provide a shared storage solution like NFS.

The state of persistent volumes is independent of instances; by default, the primary OS volume does not come from Cinder, but from ephemeral storage.

Cinder supports the use of multiple storage providers within a zone. This functionality can be helpful with the Information Lifecycle Management Strategy. Also, volumes can be backed up to Swift, or other storage types (Ceph Object Storage, IBM Tivoli, Storage Manager and so on).

Cinder API version states as of Newton release

- Block Storage API v3 (**current**)
- Block Storage API v2 (**supported**)
- In Newton release, Cinder provides a lot of device-specific improvements
 - Full list is available at <http://docs.openstack.org/releasenotes/cinder/newton.html>



© Copyright 2017 Hewlett Packard Enterprise Development LP

7

Cinder API version states as of Newton release

The OpenStack® Newton release supports the API v2, and currently provides v3 API. Besides an upgrade in API version, Cinder provides a large list of device-specific improvements and new drivers. To see the full list, visit <http://docs.openstack.org/releasenotes/cinder/newton.html>.

Block storage use cases

- Re-attach data to the new instance
- Keep data off the instance, so that it can be immediately available if the instance fails
- Keep data off the compute node
- Use the dedicated block storage device for large capacity
- Use the block device to boot from SAN



© Copyright 2017 Hewlett Packard Enterprise Development LP

8

Block storage use cases

Here are some block storage use cases in OpenStack®:

- If you need to terminate and re-launch an instance, you can keep any “non-disposable” pieces of data on Cinder volumes and re-attach them to the new instance.
- If an instance misbehaves or “crashes” unexpectedly, you can launch a new instance and attach Cinder volumes to that new instance with data intact.
- If a compute node crashes, you can launch new instances on the surviving compute nodes and attach Cinder volumes to those new instances with data intact.
- Using a dedicated storage node or storage subsystem to host Cinder volumes, capacity that is greater than what is available via the direct-attached storage in the compute nodes can be provided.

Note: This is also true if NFS is used with shared storage.

- A Cinder volume can be used as the boot disk for a Cloud instance; in that scenario, an ephemeral disk is not required.

Cinder storage terms

Term	Definition/Example
Backup	<ul style="list-style-type: none"> – Full copy of a volume stored in an external service (e.g. Object Storage or Ceph) – Can be restored to the same volume that the backup was originally taken from, or to a new volume
Snapshot	<ul style="list-style-type: none"> – A point-in-time copy of the data contained in a volume
Storage providers	<ul style="list-style-type: none"> – Offer volumes to other services (i.e. OpenStack® Compute) via network protocols like iSCSI or NFS – Examples: LVM (Logical Volume Manager), HPE 3PAR
Volume	<ul style="list-style-type: none"> – Block storage devices that you attach to instances to enable persistent storage – Can be attached to a running instance – Can be detached from an instance and attached to a different instance
Volume type	<ul style="list-style-type: none"> – An administrator-defined type of a block storage volume (e.g. two LVM backends, one with SSDs and the other with HDDs) – Allows Cinder operators to expose multiple backend storage providers to tenants. Tenants can then choose the volume type (or backend) that they prefer. – Administrator-defined extra_specs associated with your volume types, for example: Volume type = SATA, with extra_specs (RPM = 10000, RAID level = 5)

Cinder storage terms

The table above lists some important Cinder storage terms.

OpenStack® Block Storage (Cinder) drivers

Driver	Create Volume	Delete Volume	Attach Volume	Detach Volume	Extend Volume	Create Snapshot	Delete Snapshot	List Snapshots	Create Volume from Snapshot	Create Volume from Image	Create Volume from Volume (Clone)	Create Image from Volume	Volume Migration (host assisted)	QoS	Volume Replication	Consistency Group	Supported Protocols
CloudBeast	Kilo	Kilo	Kilo	Kilo	Kilo	Kilo	Kilo	Kilo	Kilo	Kilo	Kilo	Kilo	Kilo				iSCSI (Kilo), NFS (Mitaka)
Cohu Data	Mitaka	Mitaka	Mitaka	Mitaka	Mitaka	Mitaka	Mitaka	Mitaka	Mitaka	Mitaka	Mitaka	Mitaka	Mitaka				AoE (Grizzly)
Coraid	Grizzly	Grizzly	Grizzly	Grizzly	Grizzly	Grizzly	Grizzly	Grizzly	Grizzly	Grizzly	Grizzly	Havana	Havana	Havana			
Datera	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno			iSCSI (Juno)
Dell EqualLogic	Havana	Havana	Havana	Havana	Havana	Iochouse	Havana	Havana	Havana	Havana	Havana	Havana	Havana	Havana			iSCSI (Havana)
Dell Storage Center	Kilo	Kilo	Kilo	Kilo	Kilo	Kilo	Kilo	Kilo	Kilo	Kilo	Kilo	Kilo	Kilo	Kilo			iSCSI (Kilo), FC (Kilo)
EMC vMAX	Grizzly	Grizzly	Grizzly	Grizzly	Grizzly	Juno	Grizzly	Grizzly	Grizzly	Grizzly	Grizzly	Grizzly	Grizzly	Grizzly			iSCSI (Grizzly), FC (Iochouse)
EMC VNX Direct	Iochouse	Iochouse	Iochouse	Iochouse	Iochouse	Iochouse	Iochouse	Iochouse	Iochouse	Iochouse	Iochouse	Iochouse	Iochouse	Iochouse			iSCSI (Iochouse), FC (Juno)
EMC XtremIO	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno			Liberty iSCSI (Juno), FC (Juno)
EMC ScaleIO	Liberty	Liberty	Liberty	Liberty	Liberty	Liberty	Liberty	Liberty	Liberty	Liberty	Liberty	Liberty	Liberty	Liberty			ScaleIO (Liberty)
Fujitsu ETERNUS	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno			iSCSI (Juno), FC (Juno)
GlusterFS	Grizzly	Grizzly	Grizzly	Grizzly	Grizzly	Havana	Havana	Havana	Havana	Havana	Havana	Havana	Havana	Havana			GlusterFS (Grizzly)
Hitachi (HDS)	Havana	Havana	Havana	Havana	Havana	Havana	Havana	Havana	Havana	Havana	Havana	Havana	Havana	Havana			iSCSI (Havana), FC (Juno)
Hitachi (HUS-VM)	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno			FC (Juno)
Hitachi (VSP_VSP_G1000)	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno	Juno			FC (Juno), FCOE (Juno)

NOTE: To find the full list of supported devices, see the Cinder Supported Matrix at:
<https://wiki.openstack.org/wiki/CinderSupportMatrix>



© Copyright 2017 Hewlett Packard Enterprise Development LP

10

OpenStack® Block Storage (Cinder) drivers

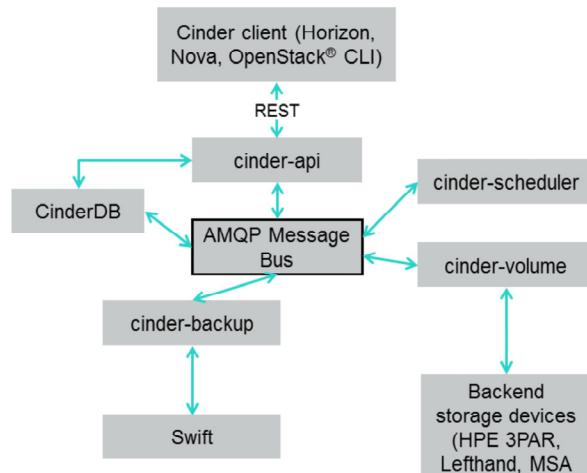
Cinder supports a large number of devices at different levels of functionality. The detailed list of supported devices is maintained in the Cinder Supported Matrix, which is available at <https://wiki.openstack.org/wiki/CinderSupportMatrix>.



Cinder architecture

High-level Cinder architecture

- **cinder-api** authenticates and routes requests for Cinder services throughout the various Cinder components
 - **cinder-scheduler** is the decision maker that decides which cinder-volume instance will be the request worker
 - **cinder-volume** represents workers with a volume driver loaded; each worker invokes methods in their specific driver to serve the requests from the cinder-api or cinder-scheduler
 - **cinder-backup** provides means to back up a Cinder volume to various backup targets
 - **CinderDB** is a database (MySQL, MariaDB, Postgres, etc.) housing Cinder block volume state, and metadata



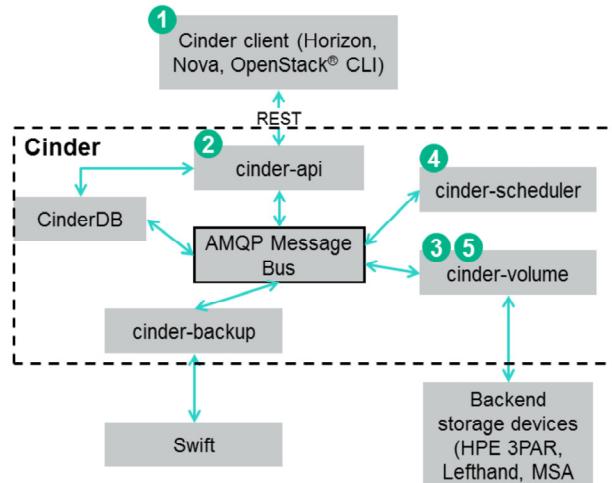
High-level Cinder architecture

Cinder works with volume abstractions. You specify features you want for the volume you create and Cinder determines the proper storage array in which to create the volume. A raw volume in the selected backend is returned.

- **cinder-api**: The cinder-api service is a Python WSGI (Web Server Gateway Interface) application that accepts and authenticates requests and routes them throughout the Block Storage system. It supports the OpenStack®-specific Cinder API only. The Nova EC2 interface provides an EC2 compatible emulation layer on top of the cinder-api.
 - **cinder-scheduler**: The cinder-scheduler is responsible for scheduling and routing requests to the appropriate cinder-volume service. This can be simple round-robin scheduling to the running volume services, or it can be more sophisticated through the use of the Filter Scheduler. The Filter Scheduler, similar in function to the Nova Filter Scheduler, is the default and enables filtering on features such as capacity, availability zone, volume types and capabilities, as well as custom filters.
 - **cinder-volume**: The cinder-volume service is responsible for managing Block Storage devices, specifically the backend devices themselves.
 - **cinder-backup**: The cinder-backup service provides means to back up a Cinder volume to the location supported by the backend driver (for example, OpenStack® Swift Object Storage, Ceph Object Storage, IBM Tivoli Storage Manager).
 - **Message Queue**: Cinder uses messaging to route information between the Block Storage service processes.
 - **CinderDB**: A database (MySQL, MariaDB, Postgres, and similar) housing Cinder block volume state and metadata.

Cinder volume creation workflow (1 of 2)

1. The client issues a request to create a volume through invoking REST API
2. The cinder-api process validates the request, and the user credentials, and puts a message onto AMQP queue for processing
3. The cinder-volume process takes the message off of the queue, sends the message to cinder-scheduler to determine into which backend to provision the volume
4. The cinder-scheduler process takes the message off of the queue, generates a candidate list based on current state and requested volume criteria (for example, size, availability zone)
5. The cinder-volume process obtains the cinder-scheduler response from the queue; it iterates through the candidate list until it is successful



Cinder volume creation workflow (1 of 2)

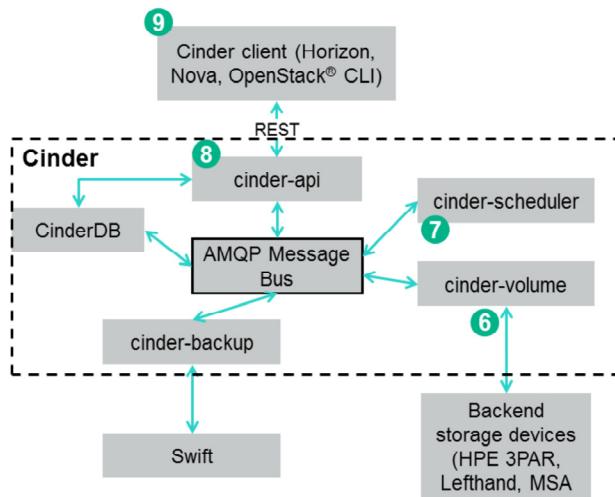
The slide above represents the Cinder volume creation workflow:

1. The client issues a request to create a volume through invoking REST API (the client can use the `python-cinderclient` CLI utility).
2. The cinder-api process validates the request and the user credentials; once validated, the message is put onto the AMQP queue for processing.
3. The cinder-volume process takes the message off of the queue and sends it to cinder-scheduler to determine which backend to provision the volume.
4. The cinder-scheduler process takes the message off of the queue and generates a candidate list based on current state and requested volume criteria (size, availability zone, volume type [including extra specs]).
5. The cinder-volume process reads the response message from cinder-scheduler from the queue; it iterates through the candidate list by invoking backend driver methods until it is successful.

(continued on the next page)

Cinder volume creation workflow (2 of 2)

6. Cinder driver creates the requested volume through interactions with storage subsystem (dependent on configuration and protocol)
7. The cinder-volume process collects volume metadata and connection information and posts a response message to the AMQP queue
8. The cinder-api process reads the response message from the queue and responds to the client
9. The client receives information including the status of creation request, volume UUID, and so on



Cinder volume creation workflow (2 of 2)

(continued from the previous page)

6. Cinder driver creates the requested volume through interactions with storage subsystem (dependent on configuration and protocol).
7. The cinder-volume process collects volume metadata and connection information and posts a response message to the AMQP queue.
8. The cinder-api process reads the response message from the queue and responds to the client.
9. The client receives information including the status of creation request, volume UUID, and so on.



Common Cinder management tasks

Common management tasks

– Managing volumes

- Create
- Delete
- List, show
- Metadata
- Detach, attach
- Extend
- Migrate
- Modify the type of an existing volume
- Automated FC Zoning during volume attaching/detaching

– Managing types

- Create volume types

– Snapshots and backups

- Create a snapshot
- Back up a volume
- Create a volume from a snapshot
- Add metadata to backup
- Import/export backups
- Add and delete quotas

– Service management

- Perform cinder service-list
- Disable, enable services



© Copyright 2017 Hewlett Packard Enterprise Development LP

16

Common management tasks

The slide above shows the common management tasks for Cinder service in four categories:

- Managing volumes
- Managing volume types
- Working with snapshots and backups
- Performing the cinder service management

Viewing the status of Cinder services

List the running Cinder services and view their status

```
$ cinder service-list
+-----+-----+-----+-----+-----+
| Binary | Host | Zone | Status | State | Updated_at |
+-----+-----+-----+-----+-----+
| cinder-scheduler | hpdevstack | nova | enabled | up | 2013-10-21T19:01:19.000000 |
| cinder-volume | hpdevstack@3parfc | nova | enabled | up | 2013-10-21T19:01:14.000000 |
| cinder-volume | hpdevstack@3pariscsi | nova | enabled | up | 2013-10-21T19:01:16.000000 |
| cinder-volume | hpdevstack@lefthand | nova | enabled | up | 2013-10-21T19:01:22.000000 |
+-----+-----+-----+-----+-----+
```

NOTE: Same result can be obtained if you enter the `openstack volume service list` command.

Viewing the status of Cinder services

To verify the status of the Cinder service, issue the `cinder service-list` command. You can obtain the same result if you run the `openstack volume service list` command.

Creating volumes

Create a volume

```
openstack volume create <name> --size <size>
```

Field	Value
attachments	[]
availability_zone	nova
bootable	false
consistencygroup_id	None
created_at	2016-01-24T13:55:45.362030
description	None
encrypted	False
id	9f173e22-9458-4f76-b674-eee5f8dbbf74
migration_status	None
multiattach	False
name	myvolA
properties	disabled
provisioning_status	disabled
size	1
snapshot_id	None
source_volid	None
status	creating
type	lvmdriver-1
user_id	4c47a0f864a846c9933f0045f146695e

Create a boot volume from an image

```
openstack volume create --size <in GB> --description <description of vol>
--image <image_id> <name of vol>
```

Create an instance using a boot volume

```
openstack server create --volume <VOLUME_ID> --flavor 1 myserv2
```

Creating volumes

The `openstack volume create` command can be used to create a new Cinder volume using the CLI. The example above shows the creation of the standard volume and the bootable volume. The bootable volume is a copy of the Glance image provisioned on the block storage. In addition, the slide above shows an example of using an existing volume to boot the new instance.

Attach a volume to an existing instance (1 of 2)

List existing instances

```
openstack server list
```

List existing Cinder storage

```
openstack volume list
```

Attach a volume to an instance

```
openstack server add volume <server id> <volume id>
```

ID	Display Name	Status	Size	Attached to
f02bdb3c-3e6a-46a0-a050-e42f5de2625d	vol4labs	in-use	1	Attached to instance4labs on /dev/vdb

Attach a volume to an existing instance (1 of 2)

The commands on the slide above can be used to obtain the information about available servers and volumes. The third command is used to attach a volume to the existing instance.

Attach a volume to an existing instance (2 of 2)

View volume attachment

```
openstack volume list
```

```
student@ubuntu14libDevStack:~/devstack$ openstack volume list
+-----+-----+-----+-----+
| ID   | Display Name | Status | Size | Attached to |
+-----+-----+-----+-----+
| 113090de-62ec-4d77-be76-f058e340e0f8 | myvol1 | in-use | 1 | Attached to testinstance on /dev/vdb |
+-----+-----+-----+-----+
```

```
openstack server show <server id>
```

```
student@ubuntu14libDevStack:~/devstack$ openstack server show 14b21f0b-e787-46fc-9b68-e288957a8055
+-----+-----+
| Field          | Value      |
+-----+-----+
| OS-DCF:diskConfig | AUTO      |
| OS-EXT-AZ:availability_zone | nova     |
| OS-EXT-KEY:ATTR:host | ubuntu14libDevStack |
| OS-EXT-SRV-ATTR:hostname | ubuntu14libDevStack |
+-----+-----+
```

```
key_name           testkey
name              testinstance
os-extended-volumes:volumes_attached [({u'id': u'113090de-62ec-4d77-be76-f058e340e0f8'})]
project_id        b9b438ae1bc94359dcf393654c9ddc4
properties
security_groups
status            ACTIVE
updated           2016-01-20T02:18:18Z
user_id           4c47a9f0864a846c9933f0045f146695e
+-----+-----+
```

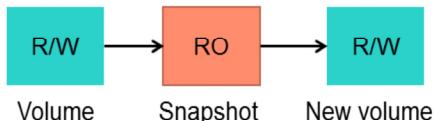
Attach a volume to an existing instance (2 of 2)

After attaching the volume, you can enter the first command shown on the slide to display the status of your volumes, including the device path for any volumes attached to an instance. The second command provides an example output of the `server show` command that also provides the ID of the attached volume.

Cinder snapshots and backups

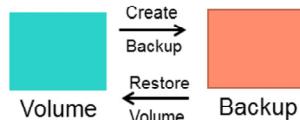
Snapshot

- A point-in-time copy of the data contained in a volume
- A read-only copy of a volume
- Can be created from a volume that is currently in use or in an available state
- Can be used to create another read-write volume from existing snapshot



Backup

- A full copy of a volume stored in an external service
- The service can be configured
- Volumes can be backed up to any supported backup system (e.g. Swift, Ceph Object Store, IBM Tivoli Storage Manager, NFS)
- You can restore to the original volume or to a new volume
- Volumes must be in an unattached and available state



Cinder snapshots and backups

Similarly to storage array snapshots, Cinder creates **snapshots** as point-in-time copies of the data contained in a volume. Cinder exposes an API to the Nova component, and provides the results of the volume creation request through the return states. The following return states are provided, depending on the API call and the result of the execution:

- **CREATING**—The snapshot is being created.
- **AVAILABLE**—The snapshot is ready to be used.
- **DELETING**—The snapshot is being deleted.
- **ERROR**—An error occurred with the snapshot.
- **ERROR_DELETING**—There was an error in deleting the snapshot.

A **backup** is a full copy of a volume stored in an external service. To make a backup of the volume, the volume must be in an unattached and available state. When a backup is created, it can be restored to the original volume or to a new volume.

The Cinder command line interface provides the tools for creating a volume backup. You can restore a volume from a backup as long as the backup's associated database information (or backup metadata) is intact in the block storage database. A volume backup can only be restored on the same Block Storage service, unless you have also backed up the volume metadata. This is because restoring a volume from a backup requires metadata used by the Block Storage service. A backup can be restored from the external service to either the same volume that the backup was originally taken from, or to a new volume.

It is recommended that you back up your block storage database regularly in order to ensure the safety of the volume metadata. Alternatively, you can export and save the metadata of the selected volume backups. This is particularly useful if you only need a small subset of volumes to survive a catastrophic database failure. Having a volume backup and its backup metadata also provides volume portability. Specifically, backing up a volume and exporting its metadata allow you to restore the volume on a completely different block storage database, or even on a different cloud service. To do so, first import the backup metadata to the block storage database and then restore the backup.

Creating consistent snapshots

- When creating snapshots, ensure that:
 - Running programs have written their contents to disk
 - Use the `sync` command before taking a snapshot
 - The file system does not have any “dirty” buffers, where programs have issued the command to write to disk, but the OS has not yet done the write
 - Use the `fsfreeze` command
- To create consistent snapshots in a multi-project environment, use the following sequence:
 - If necessary, install the `util-linux` package using a command such as:


```
# apt-get install util-linux
```
 - Create snapshots while the file system is in a frozen state:


```
# fsfreeze -f /mnt && nova image-create instance_name snapshot_name && fsfreeze -u /mnt
```



© Copyright 2017 Hewlett Packard Enterprise Development LP

22

Creating consistent snapshots

When creating a snapshot, Cinder captures the state of the file system, but it does not capture the state of the memory. Before taking snapshots, ensure that:

- Running programs have written their contents to disk—Use the `sync` command
- The file system does not have any “dirty” buffers, where programs have issued the command to write to disk, but the OS has not yet done the write—Use the `fsfreeze` command

`fsfreeze` is part of the `util-linux` package, and if necessary, it can be installed by using a command such as:

```
# apt-get install util-linux
```

After it has been installed, `fsfreeze` is ready to use.

Important: Keep in mind that when you freeze the file system, all writes are halted until the file system is unfrozen. This action might completely freeze your access to the affected system, especially if you freeze the root file system. It is strongly recommended to execute an `image-creation` command in a single command line:

```
# fsfreeze -f /mnt && nova image-create inst_name snaps_name && fsfreeze -u /mnt
```

Snapshot operations

Creating a snapshot

```
cinder snapshot-create <volume id> --display_name <name of snapshot>
OSC: openstack snapshot create <volume id> --name <name of snapshot>
student@ubuntu1libDevStack:~/devstack$ cinder snapshot-create 113090de-62ec-4d77-be76-f058e340e0f8 --display_name "snapshotA"
+---+-----+
| Property | Value |
+---+-----+
| created_at | 2016-01-24T18:03:06.567700 |
| description | None |
| id | 60dd78bd-c0f7-409a-9072-7e92a5dcdf2a |
| metadata | {} |
| name | snapshotA |
| size | 1 |
| status | creating |
| volume_id | 113090de-62ec-4d77-be76-f058e340e0f8 |
+---+-----+
```

List snapshots

```
cinder snapshot-list
OSC: openstack snapshot list
student@ubuntu1libDevStack:~/devstack$ cinder snapshot-list
+---+-----+-----+-----+-----+
| ID | Volume ID | Status | Name | Size |
+---+-----+-----+-----+-----+
| 60dd78bd-c0f7-409a-9072-7e92a5dcdf2a | 113090de-62ec-4d77-be76-f058e340e0f8 | available | snapshotA | 1 |
+---+-----+-----+-----+-----+
```

Snapshot operations

The commands used to create and list the available snapshots are displayed on the slide above.

Creating a bootable volume

List images

```
openstack image list
```

ID	Name	Status
6f9c1bef-760f-4622-8374-6c13fb7cd7	snappedImageInstance4Labs	active
4f788883-c18c-42b3-886b-fa9cf3cd5d40	image4labs	active
460067e7-baa1-47c7-b3c2-d5677a14e80e	cirros-0.3.4-x86_64-uec	active
e98cac2b-f6e5-465c-bfb4-73ce7af88335	cirros-0.3.4-x86_64-uec-ramdisk	active
9ed3b7bd-64c0-40ca-a999-67caf8a79d9	cirros-0.3.4-x86_64-uec-kernel	active
d97ee41a-7716-4fc7-9a85-7b558acb99f2	Fedora-Cloud-Base-23-20151030.x86_64	active
f6fb65ba-df56-4bb4-bfa1-4da0cde6fa3d	cirros-0.3.2-x86_64-disk	active

Create a bootable volume from an image stored in Glance by using Cinder

```
openstack volume create --image <image id> --size <GB> <volume name>
$ openstack volume create --image 4f788883-c18c-42b3-886b-fa9cf3cd5d40 --size 1 bootvol4labs
```

Field	Value
attachments	[]
availability_zone	nova
bootable	false
consistencygroup_id	None
created_at	2016-03-15T20:18:59.666312
description	None
encrypted	False
id	08f8629a-dbbe-40d7-905f-933702b2c2fe
migration_status	None
is_snapshot	False
name	bootvol4labs
properties	
replication_status	disabled
size	1
snapshot_id	None
source_volid	None
status	creating
type	lvmdriver-1
user_id	2009b34c35ad4d12842f96d0c192fdf4

Display volumes

```
$ openstack volume list --long
```

ID	Display Name	Status	Size	Type	Bootable	Att.
08f8629a-dbbe-40d7-905f-933702b2c2fe	bootvol4labs	available	1	lvmdriver-1	true	

Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

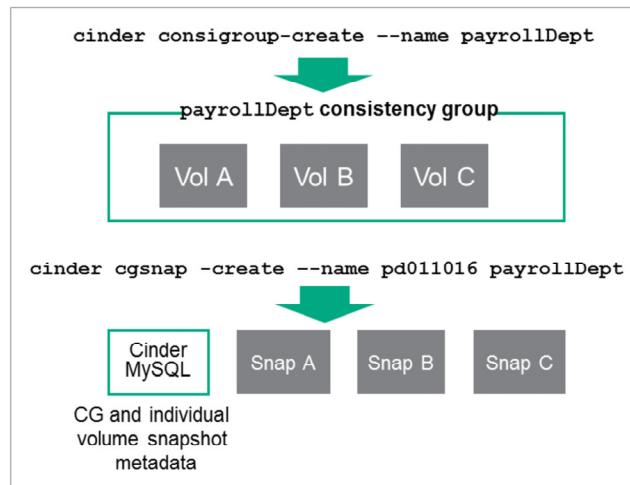
24

Creating a bootable volume

The commands displayed on the slide above can be used to create a bootable volume.

Consistency groups

- A consistency group (CG):
 - Is a named set of cinder volumes
 - Allows a set of volumes to be snapshotted or backed up as a unit
 - Is important to data protection and disaster recovery
 - Allows adding and deleting
 - Can be cloned
- When CG snapshot is taken, the sender creates a CG snapshot record in the Cinder database alongside individual volume snapshot records
- Requests are also made to the appropriate volume nodes to capture snapshots of the underlying volumes
- Backups work in a similar fashion but require volumes to be in the “available” state



Consistency groups

A consistency group (CG):

- Is a named set of Cinder volumes
- Allows a set of volumes to be snapshotted or backed up as a unit
- Is important to data protection and disaster recovery
- Allows adding and deleting volumes
- Can be cloned

When a **consistency group snapshot** is taken, the sender creates a CG snapshot record in the Cinder database alongside individual volume snapshot records. Requests are also made to the appropriate volume nodes to capture snapshots of the underlying volumes.

Backups work in a similar fashion, but require volumes to be in the “available” state.

Cinder types

Creating a custom classification system

- openstack volume type create Tier1
- openstack volume type create Tier2
- openstack volume type list

Field	Value
description	None
id	4fe0afa8-bb1e-429f-beab-d03561edd43f
is_public	True
name	Tier1
os-volume-type-access:is_public	True

Use the new type to create a volume

```
openstack volume create --type Tier1 5
```

Create custom extra_specs and relate them to types

```
openstack volume type set --property hp3par:cpg=CPG3 --property hp3par:provisioning=full \
--property qos:maxIOPS=2000 --property qos:maxBWS=100 volume_backend_name=3pariscsi Tier1
```

Display extra_specs

```
openstack volume type show Tier1
```

Cinder types

Cinder provides the functionality to create a custom Cinder type. In the following example, the `cinder type-create` command is used to build a custom classification system and perform the volume creation using the new `volume type`:

- `openstack volume type create Tier1`
- `openstack volume type create Tier2`
- `openstack volume type list`
- `openstack volume create --type Tier1 5`

Cinder allows for the creation of extra specifications. The following example shows how to create custom `extra_specs` and relate them to a type:

- `openstack volume type set --property hp3par:cpg=CPG3 \
--property hp3par:provisioning=full \
--property qos:maxIOPS=2000 --property qos:maxBWS=100 \
volume_backend_name=3pariscsi Tier1`
- `$ openstack volume type show Tier1`

Cinder volumes in the Horizon UI

The screenshot shows the OpenStack Horizon UI for Cinder volumes. The main interface displays a list of volumes, with one volume named "myvol1" selected. A context menu is open for this volume, listing options like "Manage Attachments", "Create Snapshot", "Change Volume Type", and "Upload to Image". Another context menu is shown for a volume snapshot named "snapshotA". A callout box at the top right explains that transferring allows volumes to be exchanged between projects. A callout box on the right side highlights the "Attached volume options" in the context menu.

Name	Description	Size	Status	Type	Attached To	Availability Zone	Bootable	Encrypted	Actions
myvol1	-	1GB	In-use	lvmdriver.1	Attached to testinstance on /dev/vdb	nova	No	No	<button>Edit Volume</button>

Name	Description	Size	Status	Volume Name	Actions
snapshotA	-	1GB	Available	myvol1	<button>Create Volume</button> <button>Edit Snapshot</button> <button>Delete Volume Snapshot</button>

Hewlett Packard
Enterprise

© Copyright 2017 Hewlett Packard Enterprise Development LP

27

Cinder volumes in the Horizon UI

In addition to volume creation, the Horizon UI allows transferring the volumes. Transferring is used to exchange volumes between projects.



Troubleshooting Cinder

Troubleshooting Cinder

- Log files of Cinder are located at `/var/log/cinder`
 - `cinder-api.log`—Keeps the API access log
 - `cinder-scheduler.log`—Keeps information on how the scheduler allocated volumes by host
 - `cinder-volume.log`—Keeps information on how Cinder manages the attaching and detaching of volumes
- Potential issues to be checked:
 - Service endpoints and the associated users were not created properly
 - The Keystone integration is incorrect
 - iSCSI is not created and attached properly



© Copyright 2017 Hewlett Packard Enterprise Development LP

29

Troubleshooting Cinder

As with all other OpenStack® projects, the general troubleshooting rules are applicable.

The cinder log files are located at `/var/log/cinder`:

- `cinder-api.log`—Keeps the API access log
- `cinder-scheduler.log`—Keeps information on how the scheduler allocated volumes by host
- `cinder-volume.log`—Keeps information on how Cinder manages the attaching and detaching of volumes

Potential issues to be checked:

- The service endpoints and associated users were not created properly
- The Keystone integration is incorrect
- iSCSI is not created and attached properly

Block storage creation failures

- If a volume immediately goes to an error state after its creation:
 - Grep the Cinder log files for the UUID of the volume
 - The log files are located at `/var/log/cinder/*.log`
 - First check the log files on the cloud controller, and then check the storage node where the volume creation was attempted
 - The command might look like:
`# grep 903b85d0-bacc-4855-a261-10843fc2d65b /var/log/cinder/*.log`



© Copyright 2017 Hewlett Packard Enterprise Development LP

30

Block storage creation failures

If you try to create a volume and the volume immediately goes into an error state, the best way to troubleshoot the problem is to grep the Cinder log files for the Universal Unique Identifier (UUID) of the volume. First check the log files on the cloud controller, and then check the storage node where you attempted to create the volume:

```
# grep 903b85d0-bacc-4855-a261-10843fc2d65b /var/log/cinder/*.log
```

Module summary

- In this module:
 - Various types of OpenStack® storage were described
 - The function and primary features of Cinder were discussed
 - Some Cinder use cases were provided
 - The primary components of the Cinder architecture were explained
 - The latest list of Cinder drivers was located
 - The process of generating a Cinder volume was described
 - Common management tasks associated with Cinder were demonstrated
 - The ways to troubleshoot common Cinder issues were described



© Copyright 2017 Hewlett Packard Enterprise Development LP

31

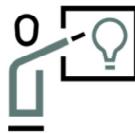
Module summary

In this module:

- Various types of OpenStack® storage were described
- The function and primary features of Cinder were discussed
- Some Cinder use cases were provided
- The primary components of the Cinder architecture were explained
- The latest list of Cinder drivers was located
- The process of generating a Cinder volume was described
- Common management tasks associated with Cinder were demonstrated
- The ways to troubleshoot common Cinder issues were described

Learning check

Learning check



What is the cinder-scheduler responsible for?

- A. Routing requests to the appropriate object service
- B. Scheduling storage replication
- C. Routing requests to the appropriate volume service
- D. Scheduling boot volume requests



© Copyright 2017 Hewlett Packard Enterprise Development LP

33

Learning check

What is the cinder-scheduler responsible for?

- A. Routing requests to the appropriate object service
- B. Scheduling storage replication
- C. Routing requests to the appropriate volume service
- D. Scheduling boot volume requests

Learning check answer



What is the cinder-scheduler responsible for?

- A. Routing requests to the appropriate object service
- B. Scheduling storage replication
- C. Routing requests to the appropriate volume service**
- D. Scheduling boot volume requests



© Copyright 2017 Hewlett Packard Enterprise Development LP

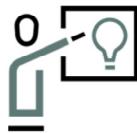
34

Learning check answer

What is the cinder-scheduler responsible for?

- A. Routing requests to the appropriate object service
- B. Scheduling storage replication
- C. Routing requests to the appropriate volume service**
- D. Scheduling boot volume requests

Learning check



What OpenStack® storage type is not persistent?

- A. Block storage
- B. Ephemeral storage
- C. Object storage
- D. File storage

Learning check

What OpenStack® storage type is not persistent?

- A. Block storage
- B. Ephemeral storage
- C. Object storage
- D. File storage

Learning check answer



What OpenStack® storage type is not persistent?

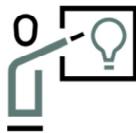
- A. Block storage
- B. Ephemeral storage**
- C. Object storage
- D. File storage

Learning check answer

What OpenStack® storage type is not persistent?

- A. Block storage
- B. Ephemeral storage**
- C. Object storage
- D. File storage

Learning check



What is a snapshot?

- A. A full copy of a volume stored on an external service
- B. A copy of an instance's runtime state
- C. A copy of an object stored in a Swift container
- D. A point-in-time copy of data contained in a volume

Learning check

What is a snapshot?

- A. A full copy of a volume stored on an external service
- B. A copy of an instance's runtime state
- C. A copy of an object stored in a Swift container
- D. A point-in-time copy of data contained in a volume

Learning check answer



What is a snapshot?

- A. A full copy of a volume stored on an external service
- B. A copy of an instance's runtime state
- C. A copy of an object stored in a Swift container
- D. A point-in-time copy of data contained in a volume**

Learning check answer

What is a snapshot?

- A. A full copy of a volume stored on an external service
- B. A copy of an instance's runtime state
- C. A copy of an object stored in a Swift container
- D. A point-in-time copy of data contained in a volume**



**Hewlett Packard
Enterprise**

