OpenStack Installation Guide for Red Hat Enterprise Linux and CentOS

UPDATED: 2016-08-13 23:58

Abstract
Contents
Appendix
Glossary
Search in this guide

The OpenStack system consists of several key services that are separately installed. These services work together depending on your cloud needs. These services include Compute service, Identity service, Networking service, Image service, Block Storage service, Object Storage service, Telemetry service, Orchestration service, and Database service. You can install any of these projects separately and configure them stand-alone or as connected entities.

This guide will show you how to install OpenStack by using packages available on Red Hat Enterprise Linux 7 and its derivatives through the RDO repository.

Explanations of configuration options and sample configuration files are included.

This guide documents OpenStack Mitaka release.

- Conventions
  - Notices
  - Command prompts
- Overview
  - Example architecture
  - Networking

- Environment
  - Security
  - Host networking
  - Network Time Protocol (NTP)
  - OpenStack packages
  - SQL database
  - NoSQL database
  - Message queue
  - Memcached
- Identity service
  - Identity service overview
  - Install and configure
  - Create the service entity and API endpoints
  - Create a domain, projects, users, and roles
  - Verify operation
  - Create OpenStack client environment scripts
- Image service
  - Image service overview
  - Install and configure
  - Verify operation
- Compute service
  - Compute service overview
  - Install and configure controller node
  - Install and configure a compute node
  - Verify operation
- Networking service
  - Networking service overview
  - Networking (neutron) concepts
  - Install and configure controller node
  - Install and configure compute node
  - Verify operation
  - Next steps
- Dashboard
  - Install and configure
  - Verify operation
  - Next steps
- Block Storage service
  - Block Storage service overview
  - Install and configure controller node
  - Install and configure a storage node
  - Verify operation
  - Next steps
- Shared File Systems service
  - Shared File Systems service overview
  - Install and configure controller node
  - Install and configure a share node
  - Verify operation
  - Next steps
- Object Storage service
  - Object Storage service overview
  - Install and configure the controller node

    - Install and configure the storage nodes
    - Create and distribute initial rings
    - Finalize installation
    - Verify operation
    - Next steps
- Orchestration service
    - Orchestration service overview
    - Install and configure
    - Verify operation
    - Next steps
- Telemetry service
    - Telemetry service overview
    - Install and configure
    - Enable Image service meters
    - Enable Compute service meters
    - Enable Block Storage meters
    - Enable Object Storage meters
    - Alarming service
    - Verify operation
    - Next steps
- Database service
    - Database service overview
    - Install and configure
    - Verify operation
    - Next steps
- Launch an instance
    - Create virtual networks
    - Create m1.nano flavor
    - Generate a key pair
    - Add security group rules
    - Launch an instance
    - Block Storage
    - Orchestration
    - Shared File Systems
- Community support
- Glossary
- *Search Page*

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups

Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

Welcome to OpenStack Documentation

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.

What Are The Next Steps?

View The Docs | Get OpenStack | Get Training

Tim Bell
OpenStack Operator

Documentation for Mitaka (April 2016)

This is the latest release. Use the menu to select a prior release if needed.

More Releases & Languages

Release Notes

New features, upgrade and deprecation notes, known issues, and bug fixes

OpenStack Projects Release Notes

OpenStack Documentation Release Notes

Install Guides

Getting started with the most commonly used OpenStack services

Installation Guide for openSUSE Leap 42.1 and SUSE Linux Enterprise Server 12 SP1

Installation Guide for Red Hat Enterprise Linux 7 and CentOS 7

Installation Guide for Ubuntu 14.04 (LTS)

Operations And Administration Guides

Administrator Guide

Manage and troubleshoot an OpenStack cloud

High Availability Guide

Install and configure OpenStack for high availability

Operations Guide

Design, create, and administer a production OpenStack cloud

Security Guide

Guidelines and scenarios for creating more secure OpenStack clouds

Virtual Machine Image Guide

Obtain, create, and modify OpenStack-compatible virtual machine images

Architecture Design Guide

Guidelines for designing an OpenStack cloud

Networking Guide

Deploy and manage OpenStack Networking (neutron)

Configuration Guides

Configuration Reference

Installation and configuration options for OpenStack

API Guides

API Complete References

Comprehensive OpenStack API reference

API Guide

Introduction to using the OpenStack API

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

Project-Specific Guides

Services and Libraries

Documentation for OpenStack services and libraries

Language Bindings and Python Clients

Documentation for the OpenStack Python bindings and clients

Contributor Guides

OpenStack Documentation Contributor Guide

Documentation workflow and conventions

OpenStack Technical Committee Governance Documents

OpenStack Technical Committee reference documents and official resolutions

OpenStack Project specifications

Specifications for future project features

OpenStack Project Team Guide

Guide to the OpenStack project and community

OpenStack Developer and Community Infrastructure Documentation

Development and infrastructure documentation

OpenStack I18n Guide

Internationalization workflow and conventions

Join the OpenStack Foundation

How to join the OpenStack Foundation

Take the Deployment Survey

Influence the future of OpenStack

Training Guides And Training Labs

Upstream training

Resources for the OpenStack Upstream Training program

Training Labs

Tools for OpenStack Training. Automatic deployment of lean virtualized multi-node openstack cluster.

Documentation treated like code, powered by the community - interested?

How To Contribute

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing .

openstack SEARCH

Conventions

UPDATED: 2016-08-13 23:58

Contents

Notices
Command prompts

The OpenStack documentation uses several typesetting conventions.

Notices

Notices take these forms:

> **Note**
>
> A comment with additional information that explains a part of the text.

> **Important**
>
> Something you must be aware of before proceeding.

> **Tip**
>
> An extra but helpful piece of practical advice.

> **Caution**
>
> Helpful information that prevents the user from making mistakes.

> **Warning**
>
> Critical information about the risk of data loss or security issues.

Command prompts

```
$ command
```

Any user, including the `root` user, can run commands that are prefixed with the `$` prompt.

```
# command
```

The `root` user must run commands that are prefixed with the `#` prompt. You can also prefix these commands with the `sudo` command, if available, to run them.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
  - Notices
  - Command prompts
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

Overview

UPDATED: 2016-08-13 23:58

Contents

- Example architecture
  - Controller
  - Compute
  - Block Storage
  - Object Storage
- Networking
  - Networking Option 1: Provider networks
  - Networking Option 2: Self-service networks

The *OpenStack* project is an open source cloud computing platform that supports all types of cloud environments. The project aims for simple implementation, massive scalability, and a rich set of features. Cloud computing experts from around the world contribute to the project.

OpenStack provides an *Infrastructure-as-a-Service (IaaS)* solution through a variety of complemental services. Each service offers an *Application Programming Interface (API)* that facilitates this integration.

This guide covers step-by-step deployment of the following major OpenStack services using a functional example architecture suitable for new users of OpenStack with sufficient Linux experience:

OpenStack services

Service	Project name	Description
Dashboard	Horizon	Provides a web-based self-service portal to interact with underlying OpenStack services, such as launching an instance, assigning IP addresses and configuring access controls.

Compute	Nova	Manages the lifecycle of compute instances in an OpenStack environment. Responsibilities include spawning, scheduling and decommissioning of virtual machines on demand.
Networking	Neutron	Enables Network-Connectivity-as-a-Service for other OpenStack services, such as OpenStack Compute. Provides an API for users to define networks and the attachments into them. Has a pluggable architecture that supports many popular networking vendors and technologies.
Storage		
Object Storage	Swift	Stores and retrieves arbitrary unstructured data objects via a *RESTful*, HTTP based API. It is highly fault tolerant with its data replication and scale-out architecture. Its implementation is not like a file server with mountable directories. In this case, it writes objects and files to multiple drives, ensuring the data is replicated across a server cluster.
Block Storage	Cinder	Provides persistent block storage to running instances. Its pluggable driver architecture facilitates the creation and management of block storage devices.
Shared services		
Identity service	Keystone	Provides an authentication and authorization service for other OpenStack services. Provides a catalog of endpoints for all OpenStack services.
Image service	Glance	Stores and retrieves virtual machine disk images. OpenStack Compute makes use of this during instance provisioning.
Telemetry	Ceilometer	Monitors and meters the OpenStack cloud for billing, benchmarking, scalability, and statistical purposes.
Higher-level services		
Orchestration	Heat	Orchestrates multiple composite cloud applications by using either the native *HOT* template format or the AWS CloudFormation template format, through both an OpenStack-native REST API and a CloudFormation-compatible Query API.

After becoming familiar with basic installation, configuration, operation, and troubleshooting of these OpenStack services, you should consider the following steps toward deployment using a production architecture:

- Determine and implement the necessary core and optional services to meet performance and redundancy requirements.
- Increase security using methods such as firewalls, encryption, and service policies.
- Implement a deployment tool such as Ansible, Chef, Puppet, or Salt to automate deployment and management of the production environment.

Example architecture

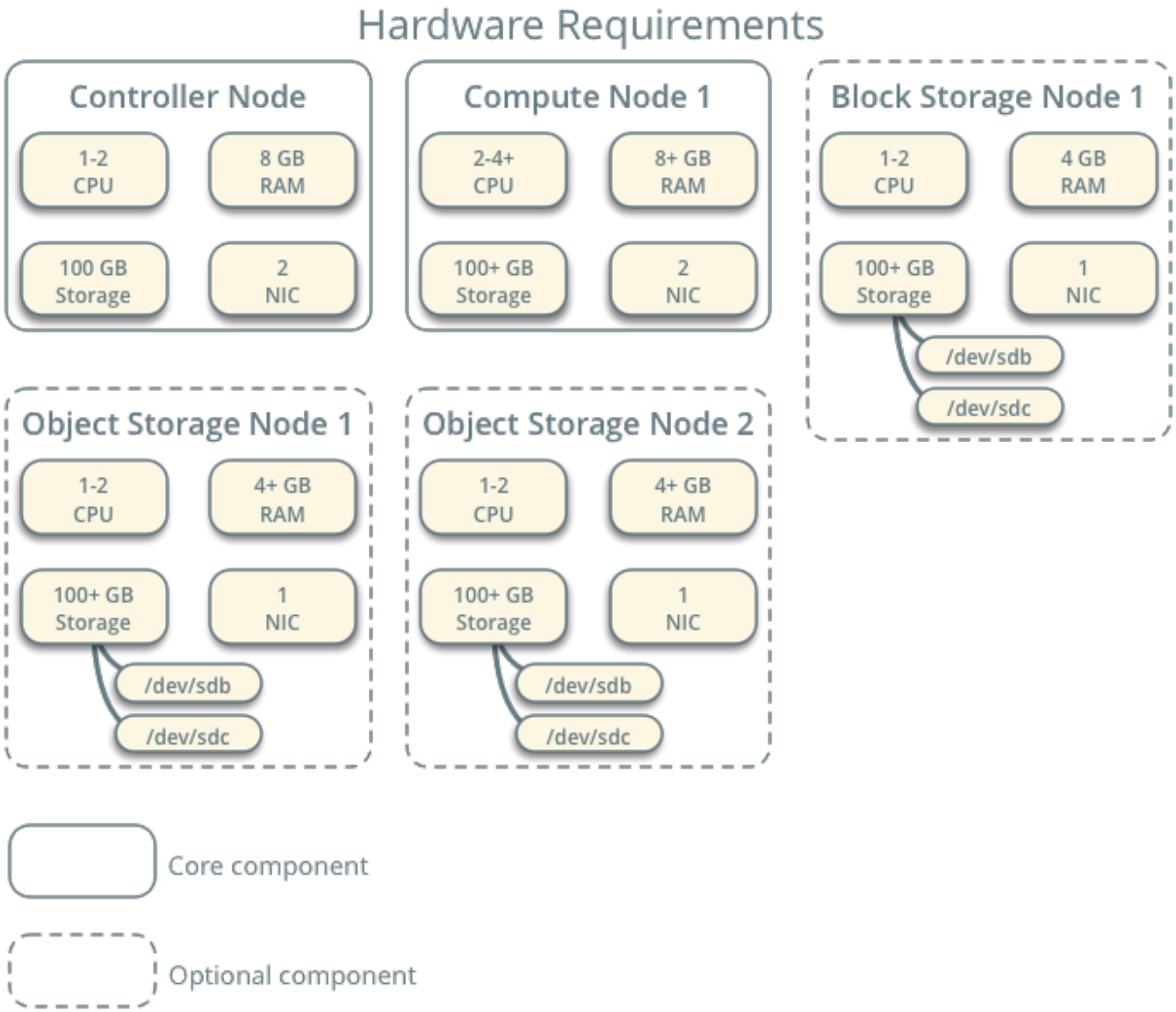
The example architecture requires at least two nodes (hosts) to launch a basic *virtual machine* or instance. Optional services such as Block Storage and Object Storage require additional nodes.

This example architecture differs from a minimal production architecture as follows:

- Networking agents reside on the controller node instead of one or more dedicated network nodes.
- Overlay (tunnel) traffic for self-service networks traverses the management network instead of a dedicated network.

For more information on production architectures, see the Architecture Design Guide, Operations Guide, and Networking

Guide.



Hardware requirements

Controller

The controller node runs the Identity service, Image service, management portions of Compute, management portion of Networking, various Networking agents, and the dashboard. It also includes supporting services such as an SQL database, *message queue*, and *NTP*.

Optionally, the controller node runs portions of Block Storage, Object Storage, Orchestration, and Telemetry services.

The controller node requires a minimum of two network interfaces.

Compute

The compute node runs the *hypervisor* portion of Compute that operates instances. By default, Compute uses the *KVM* hypervisor. The compute node also runs a Networking service agent that connects instances to virtual networks and provides firewalling services to instances via *security groups*.

You can deploy more than one compute node. Each node requires a minimum of two network interfaces.

Block Storage

The optional Block Storage node contains the disks that the Block Storage and Shared File System services provision for instances.

For simplicity, service traffic between compute nodes and this node uses the management network. Production environments should implement a separate storage network to increase performance and security.

You can deploy more than one block storage node. Each node requires a minimum of one network interface.

Object Storage

The optional Object Storage node contain the disks that the Object Storage service uses for storing accounts, containers, and objects.

For simplicity, service traffic between compute nodes and this node uses the management network. Production environments should implement a separate storage network to increase performance and security.

This service requires two nodes. Each node requires a minimum of one network interface. You can deploy more than two object storage nodes.

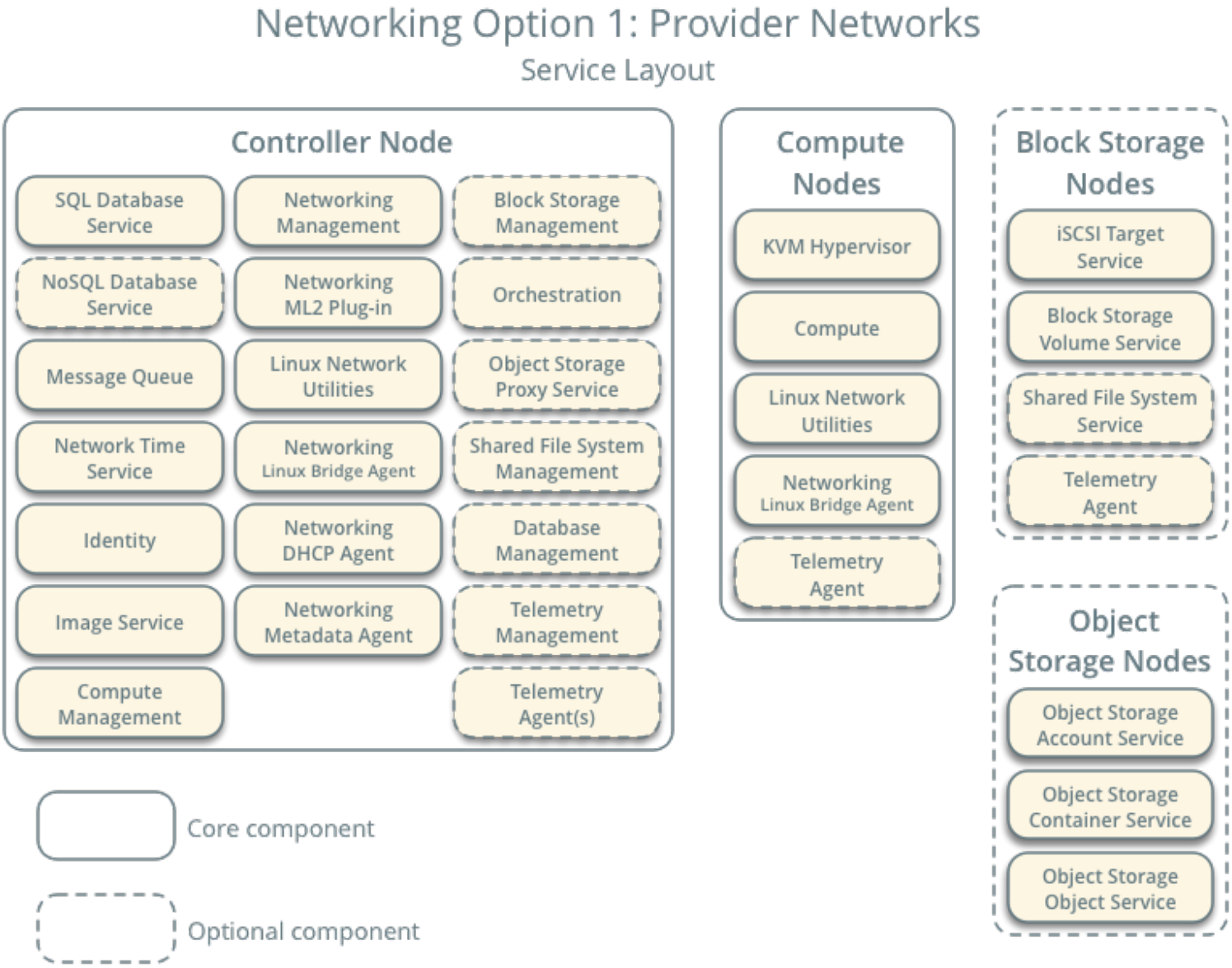
Networking

Choose one of the following virtual networking options.

Networking Option 1: Provider networks

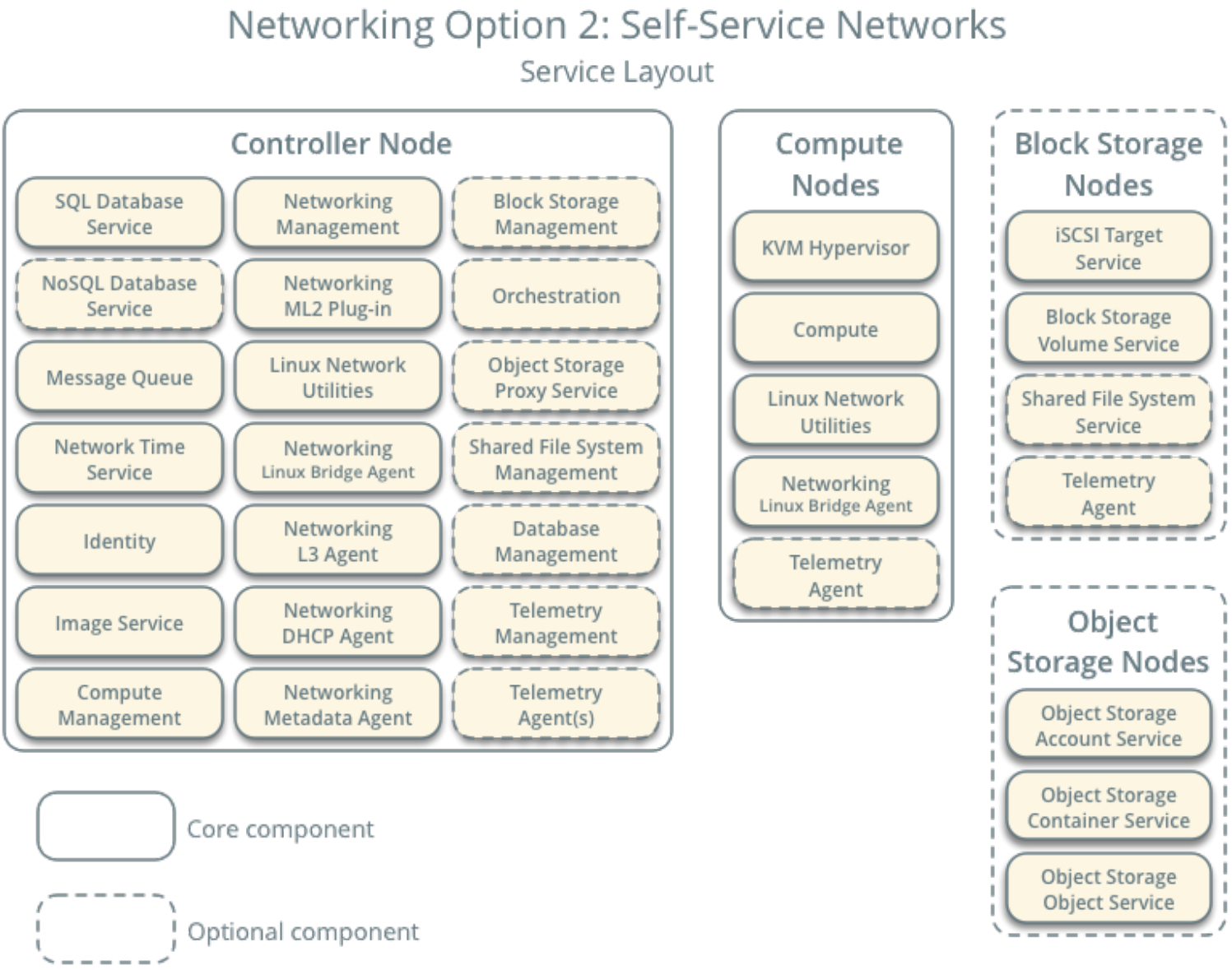
The provider networks option deploys the OpenStack Networking service in the simplest way possible with primarily layer-2 (bridging/switching) services and VLAN segmentation of networks. Essentially, it bridges virtual networks to physical networks and relies on physical network infrastructure for layer-3 (routing) services. Additionally, a *DHCP* service provides IP address information to instances.

> Note
>
> This option lacks support for self-service (private) networks, layer-3 (routing) services, and advanced services such as *LBaaS* and *FWaaS*. Consider the self-service networks option if you desire these features.



Networking Option 2: Self-service networks

The self-service networks option augments the provider networks option with layer-3 (routing) services that enable *self-service* networks using overlay segmentation methods such as *VXLAN*. Essentially, it routes virtual networks to physical networks using *NAT*. Additionally, this option provides the foundation for advanced services such as LBaaS and FWaaS.



UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
  - Example architecture
  - Networking
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

Environment

UPDATED: 2016-08-13 23:58

This section explains how to configure the controller and one compute node using the example architecture.

Although most environments include Identity, Image service, Compute, at least one networking service, and the dashboard, the Object Storage service can operate independently. If your use case only involves Object Storage, you can skip to *Object Storage service* after configuring the appropriate nodes for it. However, the dashboard requires at least the Image service, Compute, and Networking.

You must use an account with administrative privileges to configure each node. Either run the commands as the `root` user or configure the `sudo` utility.

For best performance, we recommend that your environment meets or exceeds the hardware requirements in *Hardware requirements*.

The following minimum requirements should support a proof-of-concept environment with core services and several *CirrOS* instances:

- Controller Node: 1 processor, 4 GB memory, and 5 GB storage
- Compute Node: 1 processor, 2 GB memory, and 10 GB storage

As the number of OpenStack services and virtual machines increase, so do the hardware requirements for the best performance. If performance degrades after enabling additional services or virtual machines, consider adding hardware resources to your environment.

To minimize clutter and provide more resources for OpenStack, we recommend a minimal installation of your Linux distribution. Also, you must install a 64-bit version of your distribution on each node.

A single disk partition on each node works for most basic installations. However, you should consider *Logical Volume Manager (LVM)* for installations with optional services such as Block Storage.

For first-time installation and testing purposes, many users elect to build each host as a *virtual machine (VM)*. The primary benefits of VMs include the following:

- One physical server can support multiple nodes, each with almost any number of network interfaces.
- Ability to take periodic “snap shots” throughout the installation process and “roll back” to a working configuration in the event of a problem.

However, VMs will reduce performance of your instances, particularly if your hypervisor and/or processor lacks support for hardware acceleration of nested VMs.

> **Note**
>
> If you choose to install on VMs, make sure your hypervisor provides a way to disable MAC address filtering on the provider network interface.

For more information about system requirements, see the OpenStack Operations Guide.

- Security
- Host networking
- Network Time Protocol (NTP)
- OpenStack packages
- SQL database
- NoSQL database
- Message queue
- Memcached

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
  - Security
  - Host networking
  - Network Time Protocol (NTP)
  - OpenStack packages
  - SQL database
  - NoSQL database
  - Message queue
  - Memcached
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups

Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Security

UPDATED: 2016-08-13 23:58

OpenStack services support various security methods including password, policy, and encryption. Additionally, supporting services including the database server and message broker support at least password security.

To ease the installation process, this guide only covers password security where applicable. You can create secure passwords manually, generate them using a tool such as pwgen, or by running the following command:

```
$ openssl rand -hex 10
```

For OpenStack services, this guide uses `SERVICE_PASS` to reference service account passwords and `SERVICE_DBPASS` to reference database passwords.

The following table provides a list of services that require passwords and their associated references in the guide:

Passwords

Password name	Description
Database password (no variable used)	Root password for the database
`ADMIN_PASS`	Password of user `admin`
`CEILOMETER_DBPASS`	Database password for the Telemetry service
`CEILOMETER_PASS`	Password of Telemetry service user `ceilometer`
`CINDER_DBPASS`	Database password for the Block Storage service
`CINDER_PASS`	Password of Block Storage service user `cinder`
`DASH_DBPASS`	Database password for the dashboard

DEMO_PASS	Password of user demo
GLANCE_DBPASS	Database password for Image service
GLANCE_PASS	Password of Image service user glance
HEAT_DBPASS	Database password for the Orchestration service
HEAT_DOMAIN_PASS	Password of Orchestration domain
HEAT_PASS	Password of Orchestration service user heat
KEYSTONE_DBPASS	Database password of Identity service
NEUTRON_DBPASS	Database password for the Networking service
NEUTRON_PASS	Password of Networking service user neutron
NOVA_DBPASS	Database password for Compute service
NOVA_PASS	Password of Compute service user nova
RABBIT_PASS	Password of user guest of RabbitMQ
SWIFT_PASS	Password of Object Storage service user swift

OpenStack and supporting services require administrative privileges during installation and operation. In some cases, services perform modifications to the host that can interfere with deployment automation tools such as Ansible, Chef, and Puppet. For example, some OpenStack services add a root wrapper to `sudo` that can interfere with security policies. See the Administrator Guide for more information.

Also, the Networking service assumes default values for kernel network parameters and modifies firewall rules. To avoid most issues during your initial installation, we recommend using a stock deployment of a supported distribution on your hosts. However, if you choose to automate deployment of your hosts, review the configuration and policies applied to them before proceeding further.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
  - Security
  - Host networking
  - Network Time Protocol (NTP)
  - OpenStack packages
  - SQL database
  - NoSQL database
  - Message queue
  - Memcached
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack

SEARCH

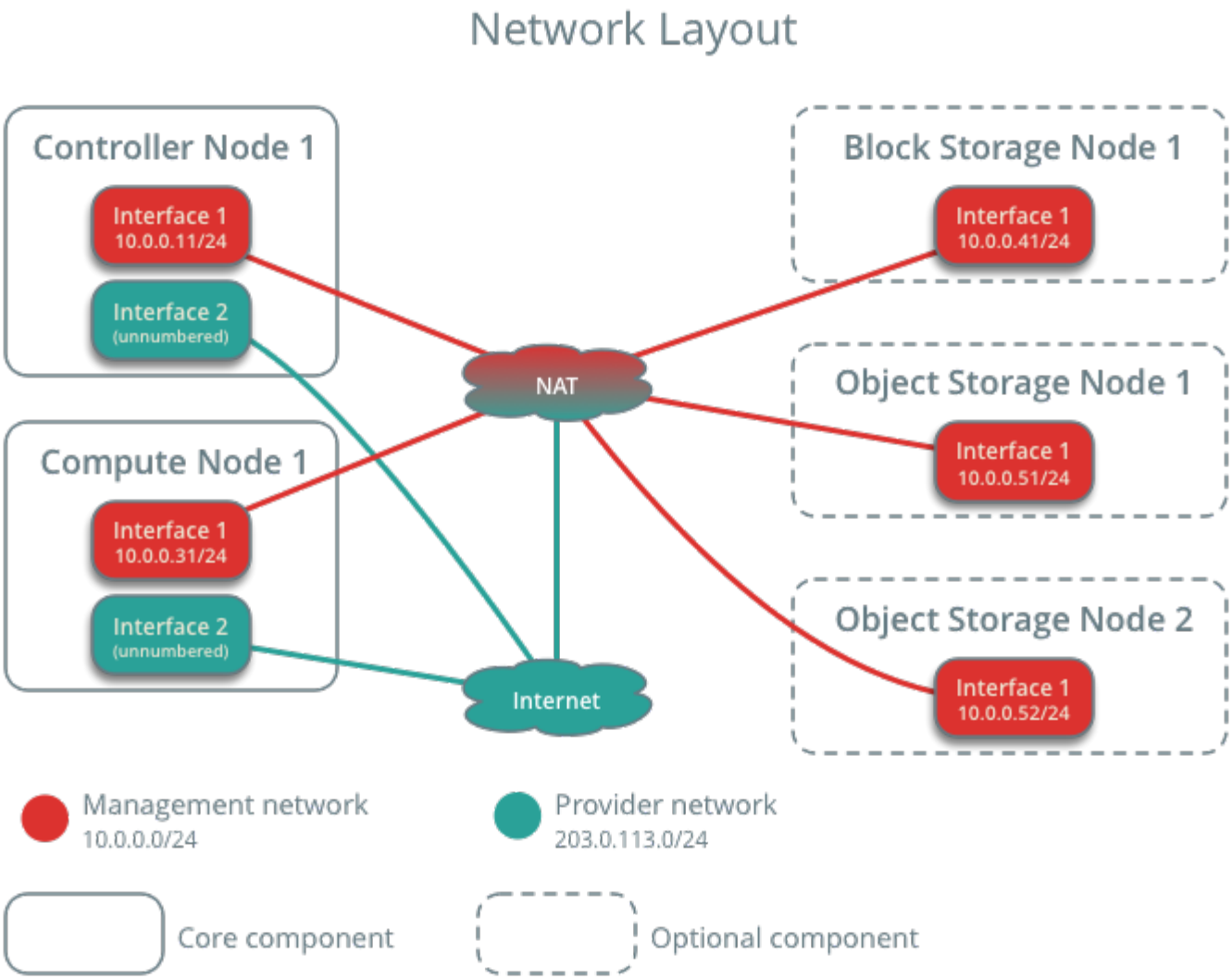
Host networking

UPDATED: 2016-08-13 23:58

After installing the operating system on each node for the architecture that you choose to deploy, you must configure the network interfaces. We recommend that you disable any automated network management tools and manually edit the appropriate configuration files for your distribution. For more information on how to configure networking on your distribution, see the documentation .

All nodes require Internet access for administrative purposes such as package installation, security updates, *DNS*, and *NTP*. In most cases, nodes should obtain internet access through the management network interface. To highlight the importance of network separation, the example architectures use private address space for the management network and assume that the physical network infrastructure provides Internet access via *NAT* or other method. The example architectures use routable IP address space for the provider (external) network and assume that the physical network infrastructure provides direct Internet access.

In the provider networks architecture, all instances attach directly to the provider network. In the self-service (private) networks architecture, instances can attach to a self-service or provider network. Self-service networks can reside entirely within OpenStack or provide some level of external network access using *NAT* through the provider network.



The example architectures assume use of the following networks:

- Management on 10.0.0.0/24 with gateway 10.0.0.1

  This network requires a gateway to provide Internet access to all nodes for administrative purposes such as package installation, security updates, *DNS*, and *NTP*.

- Provider on 203.0.113.0/24 with gateway 203.0.113.1

  This network requires a gateway to provide Internet access to instances in your OpenStack environment.

You can modify these ranges and gateways to work with your particular network infrastructure.

Network interface names vary by distribution. Traditionally, interfaces use “eth” followed by a sequential number. To cover all variations, this guide simply refers to the first interface as the interface with the lowest number and the second interface as the interface with the highest number.

Unless you intend to use the exact configuration provided in this example architecture, you must modify the networks in this procedure to match your environment. Also, each node must resolve the other nodes by name in addition to IP address. For example, the `controller` name must resolve to `10.0.0.11`, the IP address of the management interface on the controller node.

> Warning

Reconfiguring network interfaces will interrupt network connectivity. We recommend using a local terminal session for these procedures.

Note

Your distribution enables a restrictive *firewall* by default. During the installation process, certain steps will fail unless you alter or disable the firewall. For more information about securing your environment, refer to the OpenStack Security Guide.

- Controller node
- Compute node
- Block storage node (Optional)
- Object storage nodes (Optional)
- Verify connectivity

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

    - Network Time Protocol (NTP)
    - OpenStack packages
    - SQL database
    - NoSQL database
    - Message queue
    - Memcached
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Network Time Protocol (NTP)

UPDATED: 2016-08-13 23:58

You should install Chrony, an implementation of *NTP*, to properly synchronize services among nodes. We recommend that you configure the controller node to reference more accurate (lower stratum) servers and other nodes to reference the controller node.

- Controller node
- Other nodes
- Verify operation

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
  - Security
  - Host networking
  - Network Time Protocol (NTP)
  - OpenStack packages
  - SQL database
  - NoSQL database
  - Message queue
  - Memcached
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack

SEARCH

OpenStack packages

UPDATED: 2016-08-13 23:58

Contents

Prerequisites
Enable the OpenStack repository
Finalize the installation

Distributions release OpenStack packages as part of the distribution or using other methods because of differing release schedules. Perform these procedures on all nodes.

Warning

Your hosts must contain the latest versions of base installation packages available for your distribution before proceeding further.

Note

Disable or remove any automatic update services because they can impact your OpenStack environment.

Prerequisites

> Warning
>
> We recommend disabling EPEL when using RDO packages due to updates in EPEL breaking backwards compatibility. Or, preferably pin package versions using the `yum-versionlock` plugin.

> Note
>
> CentOS does not require the following steps.

1. On RHEL, register your system with Red Hat Subscription Management, using your Customer Portal user name and password:

```
# subscription-manager register --username="USERNAME" --password="PASSWORD"
```

2. Find entitlement pools containing the channels for your RHEL system:

```
# subscription-manager list --available
```

3. Use the pool identifiers found in the previous step to attach your RHEL entitlements:

```
# subscription-manager attach --pool="POOLID"
```

4. Enable required repositories:

```
# subscription-manager repos --enable=rhel-7-server-optional-rpms \
  --enable=rhel-7-server-extras-rpms --enable=rhel-7-server-rh-common-rpms
```

Enable the OpenStack repository

- On CentOS, the `extras` repository provides the RPM that enables the OpenStack repository. CentOS includes the `extras` repository by default, so you can simply install the package to enable the OpenStack repository.

```
# yum install centos-release-openstack-mitaka
```

On RHEL, download and install the RDO repository RPM to enable the OpenStack repository.

```
# yum install https://rdoproject.org/repos/rdo-release.rpm
```

Finalize the installation

1. Upgrade the packages on your host:

   ```
   # yum upgrade
   ```

   > **Note**
   >
   > If the upgrade process includes a new kernel, reboot your host to activate it.

2. Install the OpenStack client:

   ```
   # yum install python-openstackclient
   ```

3. RHEL and CentOS enable *SELinux* by default. Install the `openstack-selinux` package to automatically manage security policies for OpenStack services:

   ```
   # yum install openstack-selinux
   ```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
  - Security
  - Host networking
  - Network Time Protocol (NTP)
  - OpenStack packages
  - SQL database
  - NoSQL database
  - Message queue
  - Memcached
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

SQL database

UPDATED: 2016-08-13 23:58

Contents

Install and configure components
Finalize installation

Most OpenStack services use an SQL database to store information. The database typically runs on the controller node. The procedures in this guide use MariaDB or MySQL depending on the distribution. OpenStack services also support other SQL databases including PostgreSQL.

Install and configure components

1. Install the packages:

```
# yum install mariadb mariadb-server python2-PyMySQL
```

2. Create and edit the `/etc/my.cnf.d/openstack.cnf` file and complete the following actions:
   - In the `[mysqld]` section, set the `bind-address` key to the management IP address of the controller node to enable access by other nodes via the management network:

```
[mysqld]
...
bind-address = 10.0.0.11
```

```
```

- In the `[mysqld]` section, set the following keys to enable useful options and the UTF-8 character set:

```
[mysqld]
...
default-storage-engine = innodb
innodb_file_per_table
max_connections = 4096
collation-server = utf8_general_ci
character-set-server = utf8
```

Finalize installation

1. Start the database service and configure it to start when the system boots:

```
# systemctl enable mariadb.service
# systemctl start mariadb.service
```

2. Secure the database service by running the `mysql_secure_installation` script. In particular, choose a suitable password for the database `root` account.

```
# mysql_secure_installation
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
  - Security
  - Host networking
  - Network Time Protocol (NTP)
  - OpenStack packages
  - SQL database
  - NoSQL database
  - Message queue
  - Memcached
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups

Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

NoSQL database

UPDATED: 2016-08-13 23:58

Contents

Install and configure components
Finalize installation

The Telemetry service uses a NoSQL database to store information. The database typically runs on the controller node. The procedures in this guide use MongoDB.

> **Note**
>
> The installation of the NoSQL database server is only necessary when installing the Telemetry service as documented in *Telemetry service*.

Install and configure components

1. Install the MongoDB packages:

```
# yum install mongodb-server mongodb
```

2. Edit the `/etc/mongod.conf` file and complete the following actions:

   - Configure the `bind_ip` key to use the management interface IP address of the controller node.

     ```
     bind_ip = 10.0.0.11
     ```

   - By default, MongoDB creates several 1 GB journal files in the `/var/lib/mongodb/journal` directory. If you want to reduce the size of each journal file to 128 MB and limit total journal space consumption to 512 MB, assert the `smallfiles` key:

     ```
     smallfiles = true
     ```

     You can also disable journaling. For more information, see the MongoDB manual.

Finalize installation

- Start the MongoDB service and configure it to start when the system boots:

  ```
  # systemctl enable mongod.service
  # systemctl start mongod.service
  ```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
  - Security
  - Host networking
  - Network Time Protocol (NTP)
  - OpenStack packages
  - SQL database
  - NoSQL database
  - Message queue
  - Memcached
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

Message queue

UPDATED: 2016-08-13 23:58

Contents

Install and configure components

OpenStack uses a *message queue* to coordinate operations and status information among services. The message queue service typically runs on the controller node. OpenStack supports several message queue services including RabbitMQ, Qpid, and ZeroMQ. However, most distributions that package OpenStack support a particular message queue service. This guide implements the RabbitMQ message queue service because most distributions support it. If you prefer to implement a different message queue service, consult the documentation associated with it.

Install and configure components

1. Install the package:

```
# yum install rabbitmq-server
```

2. Start the message queue service and configure it to start when the system boots:

```
# systemctl enable rabbitmq-server.service
# systemctl start rabbitmq-server.service
```

3. Add the `openstack` user:

```
# rabbitmqctl add_user openstack RABBIT_PASS
Creating user "openstack" ...
...done.
```

Replace `RABBIT_PASS` with a suitable password.

4. Permit configuration, write, and read access for the `openstack` user:

```
# rabbitmqctl set_permissions openstack ".*" ".*" ".*"
Setting permissions for user "openstack" in vhost "/" ...
...done.
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
  - Security
  - Host networking
  - Network Time Protocol (NTP)
  - OpenStack packages
  - SQL database
  - NoSQL database
  - Message queue

    - Memcached
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Memcached

UPDATED: 2016-08-13 23:58

Contents

Install and configure components
Finalize installation

The Identity service authentication mechanism for services uses Memcached to cache tokens. The memcached service typically runs on the controller node. For production deployments, we recommend enabling a combination of firewalling, authentication, and encryption to secure it.

Install and configure components

1. Install the packages:

```
# yum install memcached python-memcached
```

Finalize installation

- Start the Memcached service and configure it to start when the system boots:

```
# systemctl enable memcached.service
# systemctl start memcached.service
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
  - Security
  - Host networking
  - Network Time Protocol (NTP)
  - OpenStack packages
  - SQL database
  - NoSQL database
  - Message queue
  - Memcached
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance

- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack

SEARCH

Identity service

UPDATED: 2016-08-13 23:58

- Identity service overview
- Install and configure
  - Prerequisites
  - Install and configure components
  - Configure the Apache HTTP server
  - Finalize the installation
- Create the service entity and API endpoints
  - Prerequisites
  - Create the service entity and API endpoints
- Create a domain, projects, users, and roles
- Verify operation
- Create OpenStack client environment scripts
  - Creating the scripts
  - Using the scripts

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all

OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
  - Identity service overview
  - Install and configure
  - Create the service entity and API endpoints
  - Create a domain, projects, users, and roles
  - Verify operation
  - Create OpenStack client environment scripts
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Identity service overview

UPDATED: 2016-08-13 23:58

The OpenStack *Identity service* provides a single point of integration for managing authentication, authorization, and service catalog services. Other OpenStack services use the Identity service as a common unified API. Additionally, services that provide information about users but that are not included in OpenStack (such as LDAP services) can be integrated into a pre-existing infrastructure.

In order to benefit from the Identity service, other OpenStack services need to collaborate with it. When an OpenStack service receives a request from a user, it checks with the Identity service whether the user is authorized to make the request.

The Identity service contains these components:

Server
: A centralized server provides authentication and authorization services using a RESTful interface.

Drivers
: Drivers or a service back end are integrated to the centralized server. They are used for accessing identity information in repositories external to OpenStack, and may already exist in the infrastructure where OpenStack is deployed (for example, SQL databases or LDAP servers).

Modules
: Middleware modules run in the address space of the OpenStack component that is using the Identity service. These modules intercept service requests, extract user credentials, and send them to the centralized server for authorization. The integration between the middleware modules and OpenStack components uses the Python Web Server Gateway Interface.

When installing OpenStack Identity service, you must register each service in your OpenStack installation. Identity service can then track which OpenStack services are installed, and where they are located on the network.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
  - Identity service overview
  - Install and configure
  - Create the service entity and API endpoints
  - Create a domain, projects, users, and roles
  - Verify operation
  - Create OpenStack client environment scripts
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Install and configure

UPDATED: 2016-08-13 23:58

Contents

Prerequisites
Install and configure components
Configure the Apache HTTP server
Finalize the installation

This section describes how to install and configure the OpenStack Identity service, code-named keystone, on the controller node. For performance, this configuration deploys Fernet tokens and the Apache HTTP server to handle requests.

Prerequisites

Before you configure the OpenStack Identity service, you must create a database and an administration token.

1. To create the database, complete the following actions:
   - Use the database access client to connect to the database server as the `root` user:

     ```
     $ mysql -u root -p
     ```

   - Create the `keystone` database:

```
CREATE DATABASE keystone;
```

- Grant proper access to the `keystone` database:

```
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' \
  IDENTIFIED BY 'KEYSTONE_DBPASS';
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' \
  IDENTIFIED BY 'KEYSTONE_DBPASS';
```

  Replace `KEYSTONE_DBPASS` with a suitable password.

- Exit the database access client.

2. Generate a random value to use as the administration token during initial configuration:

```
$ openssl rand -hex 10
```

Install and configure components

> **Note**
>
> Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (`...`) in the configuration snippets indicates potential default configuration options that you should retain.

> **Note**
>
> This guide uses the Apache HTTP server with `mod_wsgi` to serve Identity service requests on ports 5000 and 35357. By default, the keystone service still listens on these ports. Therefore, this guide manually disables the keystone service.

1. Run the following command to install the packages:

```
# yum install openstack-keystone httpd mod_wsgi
```

3. Edit the `/etc/keystone/keystone.conf` file and complete the following actions:

- In the `[DEFAULT]` section, define the value of the initial administration token:

```
```

```
[DEFAULT]
...
admin_token = ADMIN_TOKEN
```

Replace `ADMIN_TOKEN` with the random value that you generated in a previous step.

- In the `[database]` section, configure database access:

```
[database]
...
connection = mysql+pymysql://keystone:KEYSTONE_DBPASS@controller/keystone
```

Replace `KEYSTONE_DBPASS` with the password you chose for the database.

- In the `[token]` section, configure the Fernet token provider:

```
[token]
...
provider = fernet
```

4. Populate the Identity service database:

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

> **Note**
>
> Ignore any deprecation messages in this output.

5. Initialize Fernet keys:

```
# keystone-manage fernet_setup --keystone-user keystone --keystone-group keystone
```

Configure the Apache HTTP server

1. Edit the `/etc/httpd/conf/httpd.conf` file and configure the `ServerName` option to reference the controller node:

```
ServerName controller
```

2. Create the `/etc/httpd/conf.d/wsgi-keystone.conf` file with the following content:

```
Listen 5000
Listen 35357

<VirtualHost *:5000>
    WSGIDaemonProcess keystone-public processes=5 threads=1 user=keystone group=keystone di
splay-name=%{GROUP}
    WSGIProcessGroup keystone-public
    WSGIScriptAlias / /usr/bin/keystone-wsgi-public
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    ErrorLogFormat "%{cu}t %M"
    ErrorLog /var/log/httpd/keystone-error.log
    CustomLog /var/log/httpd/keystone-access.log combined

    <Directory /usr/bin>
        Require all granted
    </Directory>
</VirtualHost>

<VirtualHost *:35357>
    WSGIDaemonProcess keystone-admin processes=5 threads=1 user=keystone group=keystone dis
play-name=%{GROUP}
    WSGIProcessGroup keystone-admin
    WSGIScriptAlias / /usr/bin/keystone-wsgi-admin
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    ErrorLogFormat "%{cu}t %M"
    ErrorLog /var/log/httpd/keystone-error.log
    CustomLog /var/log/httpd/keystone-access.log combined

    <Directory /usr/bin>
        Require all granted
    </Directory>
</VirtualHost>
```

Finalize the installation

- Start the Apache HTTP service and configure it to start when the system boots:

```
# systemctl enable httpd.service
# systemctl start httpd.service
```


UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
  - Identity service overview
  - Install and configure
  - Create the service entity and API endpoints
  - Create a domain, projects, users, and roles
  - Verify operation
  - Create OpenStack client environment scripts
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Create the service entity and API endpoints

UPDATED: 2016-08-13 23:58

Contents

Prerequisites
Create the service entity and API endpoints

The Identity service provides a catalog of services and their locations. Each service that you add to your OpenStack environment requires a *service* entity and several *API endpoints* in the catalog.

Prerequisites

By default, the Identity service database contains no information to support conventional authentication and catalog services. You must use a temporary authentication token that you created in the section called *Install and configure* to initialize the service entity and API endpoint for the Identity service.

You must pass the value of the authentication token to the `openstack` command with the `--os-token` parameter or set the OS_TOKEN environment variable. Similarly, you must also pass the value of the Identity service URL to the `openstack` command with the `--os-url` parameter or set the OS_URL environment variable. This guide uses environment variables to reduce command length.

> Warning
>
> For security reasons, do not use the temporary authentication token for longer than necessary to initialize the Identity service.

1. Configure the authentication token:

```
$ export OS_TOKEN=ADMIN_TOKEN
```

   Replace `ADMIN_TOKEN` with the authentication token that you generated in the section called *Install and configure*. For example:

```
$ export OS_TOKEN=294a4c8a8a475f9b9836
```

2. Configure the endpoint URL:

```
$ export OS_URL=http://controller:35357/v3
```

3. Configure the Identity API version:

```
$ export OS_IDENTITY_API_VERSION=3
```

Create the service entity and API endpoints

1. The Identity service manages a catalog of services in your OpenStack environment. Services use this catalog to determine the other services available in your environment.

   Create the service entity for the Identity service:

```
$ openstack service create \
  --name keystone --description "OpenStack Identity" identity
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Identity               |
| enabled     | True                             |
| id          | 4ddaae90388b4ebc9d252ec2252d8d10 |
| name        | keystone                         |
| type        | identity                         |
+-------------+----------------------------------+
```

> **Note**
>
> OpenStack generates IDs dynamically, so you will see different values in the example command output.

2. The Identity service manages a catalog of API endpoints associated with the services in your OpenStack environment. Services use this catalog to determine how to communicate with other services in your environment.

   OpenStack uses three API endpoint variants for each service: admin, internal, and public. The admin API endpoint allows modifying users and tenants by default, while the public and internal APIs do not allow these operations. In a production environment, the variants might reside on separate networks that service different types of users for security reasons. For instance, the public API network might be visible from the Internet so customers can manage their clouds. The admin API network might be restricted to operators within the organization that manages cloud infrastructure. The internal API network might be restricted to the hosts that contain OpenStack services. Also, OpenStack supports multiple regions for scalability. For simplicity, this guide uses the management network for all endpoint variations and the default `RegionOne` region.

   Create the Identity service API endpoints:

```
$ openstack endpoint create --region RegionOne \
  identity public http://controller:5000/v3
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 30fff543e7dc4b7d9a0fb13791b78bf4 |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 8c8c0927262a45ad9066cfe70d46892c |
| service_name | keystone                         |
| service_type | identity                         |
| url          | http://controller:5000/v3        |
+--------------+----------------------------------+

$ openstack endpoint create --region RegionOne \
  identity internal http://controller:5000/v3
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 57cfa543e7dc4b712c0ab137911bc4fe |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 6f8de927262ac12f6066cfe70d99ac51 |
| service_name | keystone                         |
| service_type | identity                         |
| url          | http://controller:5000/v3        |
+--------------+----------------------------------+

$ openstack endpoint create --region RegionOne \
  identity admin http://controller:35357/v3
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
```

```
| enabled      | True                             |
| id           | 78c3dfa3e7dc44c98ab1b1379122ecb1 |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 34ab3d27262ac449cba6cfe704dbc11f |
| service_name | keystone                         |
| service_type | identity                         |
| url          | http://controller:35357/v3       |
+--------------+----------------------------------+
```

Note

Each service that you add to your OpenStack environment requires one or more service entities and three API endpoint variants in the Identity service.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
  - Identity service overview

- Install and configure
- Create the service entity and API endpoints
- Create a domain, projects, users, and roles
- Verify operation
- Create OpenStack client environment scripts

- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance

- Community support

- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines

Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Create a domain, projects, users, and roles

UPDATED: 2016-08-13 23:58

The Identity service provides authentication services for each OpenStack service. The authentication service uses a combination of *domains*, *projects* (tenants), *users*, and *roles*.

1. Create the `default` domain:

```
$ openstack domain create --description "Default Domain" default
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Default Domain                   |
| enabled     | True                             |
| id          | e0353a670a9e496da891347c589539e9 |
| name        | default                          |
+-------------+----------------------------------+
```

2. Create an administrative project, user, and role for administrative operations in your environment:
   - Create the `admin` project:

```
$ openstack project create --domain default \
  --description "Admin Project" admin
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Admin Project                    |
```

```
| domain_id   | e0353a670a9e496da891347c589539e9 |
| enabled     | True                             |
| id          | 343d245e850143a096806dfaefa9afdc |
| is_domain   | False                            |
| name        | admin                            |
| parent_id   | None                             |
+-------------+----------------------------------+
```

> **Note**
>
> OpenStack generates IDs dynamically, so you will see different values in the example command output.

- Create the `admin` user:

```
$ openstack user create --domain default \
  --password-prompt admin
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | e0353a670a9e496da891347c589539e9 |
| enabled   | True                             |
| id        | ac33776331494012 96f6c0d92d79dc16 |
| name      | admin                            |
+-----------+----------------------------------+
```

- Create the `admin` role:

```
$ openstack role create admin
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | cd2cb9a39e874ea69e5d4b896eb16128 |
| name      | admin                            |
+-----------+----------------------------------+
```

- Add the `admin` role to the `admin` project and user:

```
$ openstack role add --project admin --user admin admin
```

> **Note**
>
> This command provides no output.

> **Note**
>
> Any roles that you create must map to roles specified in the `policy.json` file in the configuration file directory of each OpenStack service. The default policy for most services grants administrative access to the `admin` role. For more information, see the Operations Guide - Managing Projects and Users.

3. This guide uses a service project that contains a unique user for each service that you add to your environment. Create the `service` project:

```
$ openstack project create --domain default \
  --description "Service Project" service
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Service Project                  |
| domain_id   | e0353a670a9e496da891347c589539e9 |
| enabled     | True                             |
| id          | 894cdfa366d34e9d835d3de01e752262 |
| is_domain   | False                            |
| name        | service                          |
| parent_id   | None                             |
+-------------+----------------------------------+
```

4. Regular (non-admin) tasks should use an unprivileged project and user. As an example, this guide creates the `demo` project and user.

   - Create the `demo` project:

```
$ openstack project create --domain default \
  --description "Demo Project" demo
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Demo Project                     |
| domain_id   | e0353a670a9e496da891347c589539e9 |
| enabled     | True                             |
| id          | ed0b60bf607743088218b0a533d5943f |
| is_domain   | False                            |
| name        | demo                             |
| parent_id   | None                             |
```

```
+-------------+----------------------------------+
```

> **Note**
>
> Do not repeat this step when creating additional users for this project.

- Create the `demo` user:

```
$ openstack user create --domain default \
  --password-prompt demo
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | e0353a670a9e496da891347c589539e9 |
| enabled   | True                             |
| id        | 58126687cbcc4888bfa9ab73a2256f27 |
| name      | demo                             |
+-----------+----------------------------------+
```

- Create the `user` role:

```
$ openstack role create user
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | 997ce8d05fc143ac97d83fdfb5998552 |
| name      | user                             |
+-----------+----------------------------------+
```

- Add the `user` role to the `demo` project and user:

```
$ openstack role add --project demo --user demo user
```

> **Note**
>
> This command provides no output.

Note

You can repeat this procedure to create additional projects and users.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
  - Identity service overview
  - Install and configure
  - Create the service entity and API endpoints
  - Create a domain, projects, users, and roles
  - Verify operation
  - Create OpenStack client environment scripts
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service

Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Verify operation

UPDATED: 2016-08-13 23:58

Verify operation of the Identity service before installing other services.

> **Note**
>
> Perform these commands on the controller node.

1. For security reasons, disable the temporary authentication token mechanism:

   Edit the `/etc/keystone/keystone-paste.ini` file and remove `admin_token_auth` from the `[pipeline:public_api]`, `[pipeline:admin_api]`, and `[pipeline:api_v3]` sections.

2. Unset the temporary `OS_TOKEN` and `OS_URL` environment variables:

   ```
   $ unset OS_TOKEN OS_URL
   ```

3. As the `admin` user, request an authentication token:

   ```
   $ openstack --os-auth-url http://controller:35357/v3 \
     --os-project-domain-name default --os-user-domain-name default \
     --os-project-name admin --os-username admin token issue
   Password:
   ```

```
+------------+--------------------------------------------------------------------+
| Field      | Value                                                              |
+------------+--------------------------------------------------------------------+
| expires    | 2016-02-12T20:14:07.056119Z                                        |
| id         | gAAAAABWvi7_B8kKQD9wdXac8MoZiQldmjEO643d-e_j-XXq9AmIegIbA7UHGPv |
|            | atnN21qtOMjCFWX7BReJEQnVOAj3nclRQgAYRsfSU_MrsuWb4EDtnjU7HEpoBb4 |
|            | o6ozsA_NmFWEpLeKy0uNn_WeKbAhYygrsmQGA49dclHVnz-OMVLiyM9ws       |
| project_id | 343d245e850143a096806dfaefa9afdc                                   |
| user_id    | ac3377633149401296f6c0d92d79dc16                                   |
+------------+--------------------------------------------------------------------+
```

> **Note**
>
> This command uses the password for the `admin` user.

4. As the `demo` user, request an authentication token:

```
$ openstack --os-auth-url http://controller:5000/v3 \
  --os-project-domain-name default --os-user-domain-name default \
  --os-project-name demo --os-username demo token issue
Password:
+------------+--------------------------------------------------------------------+
| Field      | Value                                                              |
+------------+--------------------------------------------------------------------+
| expires    | 2016-02-12T20:15:39.014479Z                                        |
| id         | gAAAAABWvi9bsh7vkiby5BpCCnc-JkbGhm9wH3fabS_cY7uabOubesi-Me6IGWW |
|            | yQqNegDDZ5jw7grI26vvgy1J5nCVwZ_zFRqPiz_qhbq29mgbQLglbkq6FQvzBRQ |
|            | JcOzq3uwhzNxszJWmzGC7rJE_H0A_a3UFhqv8M4zMRYSbS2YF0MyFmp_U       |
| project_id | ed0b60bf607743088218b0a533d5943f                                   |
| user_id    | 58126687cbcc4888bfa9ab73a2256f27                                   |
+------------+--------------------------------------------------------------------+
```

> **Note**
>
> This command uses the password for the `demo` user and API port 5000 which only allows regular (non-admin) access to the Identity service API.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
  - Identity service overview
  - Install and configure
  - Create the service entity and API endpoints
  - Create a domain, projects, users, and roles
  - Verify operation
  - Create OpenStack client environment scripts
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Create OpenStack client environment scripts

UPDATED: 2016-08-13 23:58

Contents

Creating the scripts
Using the scripts

The previous section used a combination of environment variables and command options to interact with the Identity service via the `openstack` client. To increase efficiency of client operations, OpenStack supports simple client environment scripts also known as OpenRC files. These scripts typically contain common options for all clients, but also support unique options. For more information, see the OpenStack End User Guide.

Creating the scripts

Create client environment scripts for the `admin` and `demo` projects and users. Future portions of this guide reference these scripts to load appropriate credentials for client operations.

1. Edit the `admin-openrc` file and add the following content:

```
export OS_PROJECT_DOMAIN_NAME=default
export OS_USER_DOMAIN_NAME=default
export OS_PROJECT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_AUTH_URL=http://controller:35357/v3
export OS_IDENTITY_API_VERSION=3
```

```
export OS_IMAGE_API_VERSION=2
```

Replace `ADMIN_PASS` with the password you chose for the `admin` user in the Identity service.

2. Edit the `demo-openrc` file and add the following content:

```
export OS_PROJECT_DOMAIN_NAME=default
export OS_USER_DOMAIN_NAME=default
export OS_PROJECT_NAME=demo
export OS_USERNAME=demo
export OS_PASSWORD=DEMO_PASS
export OS_AUTH_URL=http://controller:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

Replace `DEMO_PASS` with the password you chose for the `demo` user in the Identity service.

Using the scripts

To run clients as a specific project and user, you can simply load the associated client environment script prior to running them. For example:

1. Load the `admin-openrc` file to populate environment variables with the location of the Identity service and the `admin` project and user credentials:

```
$ . admin-openrc
```

2. Request an authentication token:

```
$ openstack token issue
+------------+----------------------------------------------------------------------+
| Field      | Value                                                                |
+------------+----------------------------------------------------------------------+
| expires    | 2016-02-12T20:44:35.659723Z                                          |
| id         | gAAAAABWvjYj-Zjfg8WXFaQnUd1DMYTBVrKw4h3fIagi5NoEmh21U72SrRv2trl |
|            | JWFYhLi2_uPR31Igf6A8mH2Rw9kv_bxNo1jbLNPLGzW_u5FC7InFqx0yYtTwa1e |
|            | eq2b0f6-18KZyQhs7F3teAta143kJEWuNEYET-y7u29y0be1_64KYkM7E       |
| project_id | 343d245e850143a096806dfaefa9afdc                                    |
| user_id    | ac3377633149401296f6c0d92d79dc16                                     |
+------------+----------------------------------------------------------------------+
```

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
  - Identity service overview
  - Install and configure
  - Create the service entity and API endpoints
  - Create a domain, projects, users, and roles
  - Verify operation
  - Create OpenStack client environment scripts
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack

SEARCH

Image service

UPDATED: 2016-08-13 23:58

- Image service overview
- Install and configure
  - Prerequisites
  - Install and configure components
  - Finalize installation
- Verify operation

The Image service (glance) enables users to discover, register, and retrieve virtual machine images. It offers a *REST* API that enables you to query virtual machine image metadata and retrieve an actual image. You can store virtual machine images made available through the Image service in a variety of locations, from simple file systems to object-storage systems like OpenStack Object Storage.

> **Important**
>
> For simplicity, this guide describes configuring the Image service to use the `file` back end, which uploads and stores in a directory on the controller node hosting the Image service. By default, this directory is `/var/lib/glance/images/`.
>
> Before you proceed, ensure that the controller node has at least several gigabytes of space available in this directory.
>
> For information on requirements for other back ends, see Configuration Reference.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
  - Image service overview
  - Install and configure
  - Verify operation
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Image service overview

UPDATED: 2016-08-13 23:58

The OpenStack Image service is central to Infrastructure-as-a-Service (IaaS) as shown in *Conceptual architecture*. It accepts API requests for disk or server images, and metadata definitions from end users or OpenStack Compute components. It also supports the storage of disk or server images on various repository types, including OpenStack Object Storage.

A number of periodic processes run on the OpenStack Image service to support caching. Replication services ensure consistency and availability through the cluster. Other periodic processes include auditors, updaters, and reapers.

The OpenStack Image service includes the following components:

glance-api
: Accepts Image API calls for image discovery, retrieval, and storage.

glance-registry
: Stores, processes, and retrieves metadata about images. Metadata includes items such as size and type.

> **Warning**
>
> The registry is a private internal service meant for use by OpenStack Image service. Do not expose this service to users.

Database
: Stores image metadata and you can choose your database depending on your preference. Most deployments use MySQL or SQLite.

Storage repository for image files
: Various repository types are supported including normal file systems, Object Storage, RADOS block devices, HTTP, and Amazon S3. Note that some repositories will only support read-only usage.

Metadata definition service
: A common API for vendors, admins, services, and users to meaningfully define their own custom metadata. This metadata can be used on different types of resources like images, artifacts, volumes, flavors, and aggregates. A definition includes the new property's key, description, constraints, and the resource types which it can be associated with.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
  - Image service overview
  - Install and configure
  - Verify operation
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service

- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Install and configure

UPDATED: 2016-08-13 23:58

Contents

Prerequisites
Install and configure components
Finalize installation

This section describes how to install and configure the Image service, code-named glance, on the controller node. For simplicity, this configuration stores images on the local file system.

Prerequisites

Before you install and configure the Image service, you must create a database, service credentials, and API endpoints.

1. To create the database, complete these steps:
   - Use the database access client to connect to the database server as the `root` user:

     ```
     $ mysql -u root -p
     ```

   - Create the `glance` database:

     ```
     CREATE DATABASE glance;
     ```

- Grant proper access to the `glance` database:

```
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \
  IDENTIFIED BY 'GLANCE_DBPASS';
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' \
  IDENTIFIED BY 'GLANCE_DBPASS';
```

  Replace `GLANCE_DBPASS` with a suitable password.

- Exit the database access client.

2. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

3. To create the service credentials, complete these steps:

   - Create the `glance` user:

```
$ openstack user create --domain default --password-prompt glance
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | e0353a670a9e496da891347c589539e9 |
| enabled   | True                             |
| id        | e38230eeff474607805b596c91fa15d9 |
| name      | glance                           |
+-----------+----------------------------------+
```

   - Add the `admin` role to the `glance` user and `service` project:

```
$ openstack role add --project service --user glance admin
```

> **Note**
>
> This command provides no output.

   - Create the `glance` service entity:

```
$ openstack service create --name glance \
  --description "OpenStack Image" image
```

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Image                  |
| enabled     | True                             |
| id          | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| name        | glance                           |
| type        | image                            |
+-------------+----------------------------------+
```

4. Create the Image service API endpoints:

```
$ openstack endpoint create --region RegionOne \
  image public http://controller:9292
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 340be3625e9b4239a6415d034e98aace |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+

$ openstack endpoint create --region RegionOne \
  image internal http://controller:9292
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | a6e4b153c2ae4c919eccfdbb7dceb5d2 |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+

$ openstack endpoint create --region RegionOne \
  image admin http://controller:9292
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 0c37ed58103f4300a84ff125a539032d |
| interface    | admin                            |
```

```
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+
```

Install and configure components

> **Note**
>
> Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (`...`) in the configuration snippets indicates potential default configuration options that you should retain.

1. Install the packages:

```
# yum install openstack-glance
```

2. Edit the `/etc/glance/glance-api.conf` file and complete the following actions:

   - In the `[database]` section, configure database access:

```
[database]
...
connection = mysql+pymysql://glance:GLANCE_DBPASS@controller/glance
```

     Replace `GLANCE_DBPASS` with the password you chose for the Image service database.

   - In the `[keystone_authtoken]` and `[paste_deploy]` sections, configure Identity service access:

```
[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = glance
password = GLANCE_PASS
```

```
[paste_deploy]
...
flavor = keystone
```

Replace `GLANCE_PASS` with the password you chose for the `glance` user in the Identity service.

> **Note**
>
> Comment out or remove any other options in the `[keystone_authtoken]` section.

- In the `[glance_store]` section, configure the local file system store and location of image files:

```
[glance_store]
...
stores = file,http
default_store = file
filesystem_store_datadir = /var/lib/glance/images/
```

3. Edit the `/etc/glance/glance-registry.conf` file and complete the following actions:
   - In the `[database]` section, configure database access:

```
[database]
...
connection = mysql+pymysql://glance:GLANCE_DBPASS@controller/glance
```

Replace `GLANCE_DBPASS` with the password you chose for the Image service database.

- In the `[keystone_authtoken]` and `[paste_deploy]` sections, configure Identity service access:

```
[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = glance
password = GLANCE_PASS

[paste_deploy]
...
flavor = keystone
```

Replace `GLANCE_PASS` with the password you chose for the `glance` user in the Identity service.

> **Note**
>
> Comment out or remove any other options in the `[keystone_authtoken]` section.

4. Populate the Image service database:

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

> **Note**
>
> Ignore any deprecation messages in this output.

Finalize installation

- Start the Image services and configure them to start when the system boots:

```
# systemctl enable openstack-glance-api.service \
  openstack-glance-registry.service
# systemctl start openstack-glance-api.service \
  openstack-glance-registry.service
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
  - Image service overview
  - Install and configure
  - Verify operation
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs

Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack

SEARCH

Verify operation

UPDATED: 2016-08-13 23:58

Verify operation of the Image service using CirrOS, a small Linux image that helps you test your OpenStack deployment.

For more information about how to download and build images, see OpenStack Virtual Machine Image Guide. For information about how to manage images, see the OpenStack End User Guide.

> **Note**
>
> Perform these commands on the controller node.

1. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

2. Download the source image:

```
$ wget http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img
```

> **Note**

> Install `wget` if your distribution does not include it.

3. Upload the image to the Image service using the *QCOW2* disk format, *bare* container format, and public visibility so all projects can access it:

```
$ openstack image create "cirros" \
  --file cirros-0.3.4-x86_64-disk.img \
  --disk-format qcow2 --container-format bare \
  --public
+------------------+------------------------------------------------------+
| Property         | Value                                                |
+------------------+------------------------------------------------------+
| checksum         | 133eae9fb1c98f45894a4e60d8736619                     |
| container_format | bare                                                 |
| created_at       | 2015-03-26T16:52:10Z                                 |
| disk_format      | qcow2                                                |
| file             | /v2/images/cc5c6982-4910-471e-b864-1098015901b5/file |
| id               | cc5c6982-4910-471e-b864-1098015901b5                 |
| min_disk         | 0                                                    |
| min_ram          | 0                                                    |
| name             | cirros                                               |
| owner            | ae7a98326b9c455588edd2656d723b9d                     |
| protected        | False                                                |
| schema           | /v2/schemas/image                                    |
| size             | 13200896                                             |
| status           | active                                               |
| tags             |                                                      |
| updated_at       | 2015-03-26T16:52:10Z                                 |
| virtual_size     | None                                                 |
| visibility       | public                                               |
+------------------+------------------------------------------------------+
```

For information about the `openstack image create` parameters, see Image service command-line client in the `OpenStack Command-Line Interface Reference`.

For information about disk and container formats for images, see Disk and container formats for images in the `OpenStack Virtual Machine Image Guide`.

> Note
>
> OpenStack generates IDs dynamically, so you will see different values in the example command output.

4. Confirm upload of the image and validate attributes:

```
$ openstack image list
```

```
+--------------------------------------+--------+--------+
| ID                                   | Name   | Status |
+--------------------------------------+--------+--------+
| 38047887-61a7-41ea-9b49-27987d5e8bb9 | cirros | active |
+--------------------------------------+--------+--------+
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
  - Image service overview
  - Install and configure
  - Verify operation
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance

- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack

SEARCH

Compute service

UPDATED: 2016-08-13 23:58

- Compute service overview
- Install and configure controller node
  - Prerequisites
  - Install and configure components
  - Finalize installation
- Install and configure a compute node
  - Install and configure components
  - Finalize installation
- Verify operation

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
  - Compute service overview
  - Install and configure controller node
  - Install and configure a compute node
  - Verify operation
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

Compute service overview

UPDATED: 2016-08-13 23:58

Use OpenStack Compute to host and manage cloud computing systems. OpenStack Compute is a major part of an Infrastructure-as-a-Service (*IaaS*) system. The main modules are implemented in Python.

OpenStack Compute interacts with OpenStack Identity for authentication; OpenStack Image service for disk and server images; and OpenStack dashboard for the user and administrative interface. Image access is limited by projects, and by users; quotas are limited per project (the number of instances, for example). OpenStack Compute can scale horizontally on standard hardware, and download images to launch instances.

OpenStack Compute consists of the following areas and their components:

`nova-api` service
: Accepts and responds to end user compute API calls. The service supports the OpenStack Compute API, the Amazon EC2 API, and a special Admin API for privileged users to perform administrative actions. It enforces some policies and initiates most orchestration activities, such as running an instance.

`nova-api-metadata` service
: Accepts metadata requests from instances. The `nova-api-metadata` service is generally used when you run in multi-host mode with `nova-network` installations. For details, see Metadata service in the OpenStack Administrator Guide.

`nova-compute` service
: A worker daemon that creates and terminates virtual machine instances through hypervisor APIs. For example:

  - XenAPI for XenServer/XCP
  - libvirt for KVM or QEMU
  - VMwareAPI for VMware

  Processing is fairly complex. Basically, the daemon accepts actions from the queue and performs a series of system commands such as launching a KVM instance and updating its state in the database.

`nova-scheduler` service
: Takes a virtual machine instance request from the queue and determines on which compute server host it runs.

`nova-conductor` module
: Mediates interactions between the `nova-compute` service and the database. It eliminates direct accesses to the cloud database made by the `nova-compute` service. The `nova-conductor` module scales horizontally. However, do not deploy it on nodes where the `nova-compute` service runs. For more information, see Configuration Reference Guide.

`nova-cert` module
: A server daemon that serves the Nova Cert service for X509 certificates. Used to generate certificates for `euca-bundle-image`. Only needed for the EC2 API.

`nova-network worker` daemon
: Similar to the `nova-compute` service, accepts networking tasks from the queue and manipulates the network. Performs tasks such as setting up bridging interfaces or changing IPtables rules.

`nova-consoleauth` daemon
: Authorizes tokens for users that console proxies provide. See `nova-novncproxy` and `nova-xvpvncproxy`. This service must be running for console proxies to work. You can run proxies of either type against a single nova-consoleauth service in a cluster configuration. For information, see About nova-consoleauth.

`nova-novncproxy` daemon
: Provides a proxy for accessing running instances through a VNC connection. Supports browser-based novnc clients.

`nova-spicehtml5proxy` daemon
: Provides a proxy for accessing running instances through a SPICE connection. Supports browser-based HTML5 client.

`nova-xvpvncproxy` daemon
: Provides a proxy for accessing running instances through a VNC connection. Supports an OpenStack-specific Java client.

`nova-cert` daemon
: x509 certificates.

`nova` client
: Enables users to submit commands as a tenant administrator or end user.

The queue
: A central hub for passing messages between daemons. Usually implemented with RabbitMQ, also can be implemented with another AMQP message queue, such as ZeroMQ.

SQL database
: Stores most build-time and run-time states for a cloud infrastructure, including:

  - Available instance types
  - Instances in use
  - Available networks
  - Projects

  Theoretically, OpenStack Compute can support any database that SQL-Alchemy supports. Common databases are SQLite3 for test and development work, MySQL, and PostgreSQL.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
  - Compute service overview
  - Install and configure controller node
  - Install and configure a compute node
  - Verify operation
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Install and configure controller node

UPDATED: 2016-08-13 23:58

Contents

Prerequisites
Install and configure components
Finalize installation

This section describes how to install and configure the Compute service, code-named nova, on the controller node.

Prerequisites

Before you install and configure the Compute service, you must create databases, service credentials, and API endpoints.

1. To create the databases, complete these steps:
   - Use the database access client to connect to the database server as the `root` user:

     ```
     $ mysql -u root -p
     ```

   - Create the `nova_api` and `nova` databases:

     ```
     CREATE DATABASE nova_api;
     CREATE DATABASE nova;
     ```

```
```

- Grant proper access to the databases:

```
GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';
```

  Replace `NOVA_DBPASS` with a suitable password.

- Exit the database access client.

2. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

3. To create the service credentials, complete these steps:

   - Create the `nova` user:

```
$ openstack user create --domain default \
  --password-prompt nova
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | e0353a670a9e496da891347c589539e9 |
| enabled   | True                             |
| id        | 8c46e4760902464b889293a74a0c90a8 |
| name      | nova                             |
+-----------+----------------------------------+
```

   - Add the `admin` role to the `nova` user:

```
$ openstack role add --project service --user nova admin
```

> **Note**
>
> This command provides no output.

- Create the `nova` service entity:

```
$ openstack service create --name nova \
  --description "OpenStack Compute" compute
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Compute                |
| enabled     | True                             |
| id          | 060d59eac51b4594815603d75a00aba2 |
| name        | nova                             |
| type        | compute                          |
+-------------+----------------------------------+
```

4. Create the Compute service API endpoints:

```
$ openstack endpoint create --region RegionOne \
  compute public http://controller:8774/v2.1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 3c1caa473bfe4390a11e7177894bcc7b        |
| interface    | public                                  |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | e702f6f497ed42e6a8ae3ba2e5871c78        |
| service_name | nova                                    |
| service_type | compute                                 |
| url          | http://controller:8774/v2.1/%(tenant_id)s |
+--------------+-----------------------------------------+

$ openstack endpoint create --region RegionOne \
  compute internal http://controller:8774/v2.1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | e3c918de680746a586eac1f2d9bc10ab        |
| interface    | internal                                |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | e702f6f497ed42e6a8ae3ba2e5871c78        |
| service_name | nova                                    |
| service_type | compute                                 |
| url          | http://controller:8774/v2.1/%(tenant_id)s |
+--------------+-----------------------------------------+

$ openstack endpoint create --region RegionOne \
  compute admin http://controller:8774/v2.1/%\(tenant_id\)s
```

```
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 38f7af91666a47cfb97b4dc790b94424        |
| interface    | admin                                   |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | e702f6f497ed42e6a8ae3ba2e5871c78        |
| service_name | nova                                    |
| service_type | compute                                 |
| url          | http://controller:8774/v2.1/%(tenant_id)s |
+--------------+-----------------------------------------+
```

Install and configure components

> **Note**
>
> Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (`...`) in the configuration snippets indicates potential default configuration options that you should retain.

1. Install the packages:

```
# yum install openstack-nova-api openstack-nova-conductor \
  openstack-nova-console openstack-nova-novncproxy \
  openstack-nova-scheduler
```

2. Edit the `/etc/nova/nova.conf` file and complete the following actions:
   - In the `[DEFAULT]` section, enable only the compute and metadata APIs:

```
[DEFAULT]
...
enabled_apis = osapi_compute,metadata
```

   - In the `[api_database]` and `[database]` sections, configure database access:

```
[api_database]
...
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova_api

[database]
...
```

```
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova
```

Replace `NOVA_DBPASS` with the password you chose for the Compute databases.

- In the `[DEFAULT]` and `[oslo_messaging_rabbit]` sections, configure `RabbitMQ` message queue access:

```
[DEFAULT]
...
rpc_backend = rabbit

[oslo_messaging_rabbit]
...
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS
```

Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.

- In the `[DEFAULT]` and `[keystone_authtoken]` sections, configure Identity service access:

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = nova
password = NOVA_PASS
```

Replace `NOVA_PASS` with the password you chose for the `nova` user in the Identity service.

> **Note**
>
> Comment out or remove any other options in the `[keystone_authtoken]` section.

- In the `[DEFAULT]` section, configure the `my_ip` option to use the management interface IP address of the controller node:

```
```

```
[DEFAULT]
...
my_ip = 10.0.0.11
```

- In the `[DEFAULT]` section, enable support for the Networking service:

```
[DEFAULT]
...
use_neutron = True
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

> **Note**
>
> By default, Compute uses an internal firewall driver. Since the Networking service includes a firewall driver, you must disable the Compute firewall driver by using the `nova.virt.firewall.NoopFirewallDriver` firewall driver.

- In the `[vnc]` section, configure the VNC proxy to use the management interface IP address of the controller node:

```
[vnc]
...
vncserver_listen = $my_ip
vncserver_proxyclient_address = $my_ip
```

- In the `[glance]` section, configure the location of the Image service API:

```
[glance]
...
api_servers = http://controller:9292
```

- In the `[oslo_concurrency]` section, configure the lock path:

```
[oslo_concurrency]
...
lock_path = /var/lib/nova/tmp
```

3. Populate the Compute databases:

```
# su -s /bin/sh -c "nova-manage api_db sync" nova
# su -s /bin/sh -c "nova-manage db sync" nova
```

OpenStack Documentation

Ignore any deprecation messages in this output.

Finalize installation

- Start the Compute services and configure them to start when the system boots:

```
# systemctl enable openstack-nova-api.service \
  openstack-nova-consoleauth.service openstack-nova-scheduler.service \
  openstack-nova-conductor.service openstack-nova-novncproxy.service
# systemctl start openstack-nova-api.service \
  openstack-nova-consoleauth.service openstack-nova-scheduler.service \
  openstack-nova-conductor.service openstack-nova-novncproxy.service
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment

- Identity service
- Image service
- Compute service
  - Compute service overview
  - Install and configure controller node
  - Install and configure a compute node
  - Verify operation
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines

Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Install and configure a compute node

UPDATED: 2016-08-13 23:58

Contents

Install and configure components
Finalize installation

This section describes how to install and configure the Compute service on a compute node. The service supports several *hypervisors* to deploy *instances* or *VMs*. For simplicity, this configuration uses the *QEMU* hypervisor with the *KVM* extension on compute nodes that support hardware acceleration for virtual machines. On legacy hardware, this configuration uses the generic QEMU hypervisor. You can follow these instructions with minor modifications to horizontally scale your environment with additional compute nodes.

> **Note**
>
> This section assumes that you are following the instructions in this guide step-by-step to configure the first compute node. If you want to configure additional compute nodes, prepare them in a similar fashion to the first compute node in the *example architectures* section. Each additional compute node requires a unique IP address.

Install and configure components

> **Note**
>
> Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (`...`) in the configuration snippets indicates potential default configuration options that you should retain.

1. Install the packages:

```
# yum install openstack-nova-compute
```

2. Edit the `/etc/nova/nova.conf` file and complete the following actions:
   - In the `[DEFAULT]` and [oslo_messaging_rabbit] sections, configure `RabbitMQ` message queue access:

```
[DEFAULT]
...
rpc_backend = rabbit

[oslo_messaging_rabbit]
...
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS
```

     Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.
   - In the `[DEFAULT]` and `[keystone_authtoken]` sections, configure Identity service access:

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = nova
password = NOVA_PASS
```

     Replace `NOVA_PASS` with the password you chose for the `nova` user in the Identity service.

> **Note**
>
> Comment out or remove any other options in the `[keystone_authtoken]` section.

- In the `[DEFAULT]` section, configure the `my_ip` option:

```
[DEFAULT]
...
my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
```

Replace `MANAGEMENT_INTERFACE_IP_ADDRESS` with the IP address of the management network interface on your compute node, typically 10.0.0.31 for the first node in the *example architecture*.

- In the `[DEFAULT]` section, enable support for the Networking service:

```
[DEFAULT]
...
use_neutron = True
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

> **Note**
>
> By default, Compute uses an internal firewall service. Since Networking includes a firewall service, you must disable the Compute firewall service by using the `nova.virt.firewall.NoopFirewallDriver` firewall driver.

- In the `[vnc]` section, enable and configure remote console access:

```
[vnc]
...
enabled = True
vncserver_listen = 0.0.0.0
vncserver_proxyclient_address = $my_ip
novncproxy_base_url = http://controller:6080/vnc_auto.html
```

The server component listens on all IP addresses and the proxy component only listens on the management interface IP address of the compute node. The base URL indicates the location where you can use a web browser to access remote consoles of instances on this compute node.

> **Note**

> If the web browser to access remote consoles resides on a host that cannot resolve the `controller` hostname, you must replace `controller` with the management interface IP address of the controller node.

- In the `[glance]` section, configure the location of the Image service API:

```
[glance]
...
api_servers = http://controller:9292
```

- In the `[oslo_concurrency]` section, configure the lock path:

```
[oslo_concurrency]
...
lock_path = /var/lib/nova/tmp
```

Finalize installation

1. Determine whether your compute node supports hardware acceleration for virtual machines:

```
$ egrep -c '(vmx|svm)' /proc/cpuinfo
```

   If this command returns a value of `one or greater`, your compute node supports hardware acceleration which typically requires no additional configuration.

   If this command returns a value of `zero`, your compute node does not support hardware acceleration and you must configure `libvirt` to use QEMU instead of KVM.

   - Edit the `[libvirt]` section in the `/etc/nova/nova.conf` file as follows:

```
[libvirt]
...
virt_type = qemu
```

2. Start the Compute service including its dependencies and configure them to start automatically when the system boots:

```
# systemctl enable libvirtd.service openstack-nova-compute.service
# systemctl start libvirtd.service openstack-nova-compute.service
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions

OpenStack Documentation

- Image service
- Compute service
  - Compute service overview
  - Install and configure controller node
  - Install and configure a compute node
  - Verify operation
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Verify operation

UPDATED: 2016-08-13 23:58

Verify operation of the Compute service.

> **Note**
>
> Perform these commands on the controller node.

1. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

2. List service components to verify successful launch and registration of each process:

```
$ openstack compute service list
+----+--------------------+------------+----------+---------+-------+------------
---------------+
| Id | Binary             | Host       | Zone     | Status  | State | Updated At
|
+----+--------------------+------------+----------+---------+-------+------------
---------------+
|  1 | nova-consoleauth   | controller | internal | enabled | up    | 2016-02-09T23:11:15.
```

```
000000 |
|  2 | nova-scheduler     | controller | internal | enabled | up    | 2016-02-09T23:11:15.
000000 |
|  3 | nova-conductor     | controller | internal | enabled | up    | 2016-02-09T23:11:16.
000000 |
|  4 | nova-compute       | compute1   | nova     | enabled | up    | 2016-02-09T23:11:20.
000000 |
+----+--------------------+------------+----------+---------+-------+-------------
---------------+
```

> **Note**
>
> This output should indicate three service components enabled on the controller node and one service component enabled on the compute node.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service

- Compute service overview
- Install and configure controller node
- Install and configure a compute node
- Verify operation
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Networking service

UPDATED: 2016-08-13 23:58

- Networking service overview
- Networking (neutron) concepts
- Install and configure controller node
- Install and configure compute node
- Verify operation
- Next steps

This chapter explains how to install and configure the Networking service (neutron) using the *provider networks* or *self-service networks* option.

For more information about the Networking service including virtual networking components, layout, and traffic flows, see the Networking Guide.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
  - Networking service overview
  - Networking (neutron) concepts
  - Install and configure controller node
  - Install and configure compute node
  - Verify operation
  - Next steps
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects

OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Networking service overview

UPDATED: 2016-08-13 23:58

OpenStack Networking (neutron) allows you to create and attach interface devices managed by other OpenStack services to networks. Plug-ins can be implemented to accommodate different networking equipment and software, providing flexibility to OpenStack architecture and deployment.

It includes the following components:

neutron-server
: Accepts and routes API requests to the appropriate OpenStack Networking plug-in for action.

OpenStack Networking plug-ins and agents
: Plugs and unplugs ports, creates networks or subnets, and provides IP addressing. These plug-ins and agents differ depending on the vendor and technologies used in the particular cloud. OpenStack Networking ships with plug-ins and agents for Cisco virtual and physical switches, NEC OpenFlow products, Open vSwitch, Linux bridging, and the VMware NSX product.

  The common agents are L3 (layer 3), DHCP (dynamic host IP addressing), and a plug-in agent.

Messaging queue
: Used by most OpenStack Networking installations to route information between the neutron-server and various agents. Also acts as a database to store networking state for particular plug-ins.

OpenStack Networking mainly interacts with OpenStack Compute to provide networks and connectivity for its instances.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
  - Networking service overview
  - Networking (neutron) concepts
  - Install and configure controller node
  - Install and configure compute node
  - Verify operation
  - Next steps
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Networking (neutron) concepts

UPDATED: 2016-08-13 23:58

OpenStack Networking (neutron) manages all networking facets for the Virtual Networking Infrastructure (VNI) and the access layer aspects of the Physical Networking Infrastructure (PNI) in your OpenStack environment. OpenStack Networking enables tenants to create advanced virtual network topologies which may include services such as a *firewall*, a *load balancer*, and a *virtual private network (VPN)*.

Networking provides networks, subnets, and routers as object abstractions. Each abstraction has functionality that mimics its physical counterpart: networks contain subnets, and routers route traffic between different subnets and networks.

Any given Networking set up has at least one external network. Unlike the other networks, the external network is not merely a virtually defined network. Instead, it represents a view into a slice of the physical, external network accessible outside the OpenStack installation. IP addresses on the external network are accessible by anybody physically on the outside network.

In addition to external networks, any Networking set up has one or more internal networks. These software-defined networks connect directly to the VMs. Only the VMs on any given internal network, or those on subnets connected through interfaces to a similar router, can access VMs connected to that network directly.

For the outside network to access VMs, and vice versa, routers between the networks are needed. Each router has one gateway that is connected to an external network and one or more interfaces connected to internal networks. Like a physical router, subnets can access machines on other subnets that are connected to the same router, and machines can access the outside network through the gateway for the router.

Additionally, you can allocate IP addresses on external networks to ports on the internal network. Whenever something is connected to a subnet, that connection is called a port. You can associate external network IP addresses with ports to VMs. This way, entities on the outside network can access VMs.

Networking also supports *security groups*. Security groups enable administrators to define firewall rules in groups. A VM

can belong to one or more security groups, and Networking applies the rules in those security groups to block or unblock ports, port ranges, or traffic types for that VM.

Each plug-in that Networking uses has its own concepts. While not vital to operating the VNI and OpenStack environment, understanding these concepts can help you set up Networking. All Networking installations use a core plug-in and a security group plug-in (or just the No-Op security group plug-in). Additionally, Firewall-as-a-Service (FWaaS) and Load-Balancer-as-a-Service (LBaaS) plug-ins are available.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
  - Networking service overview
  - Networking (neutron) concepts
  - Install and configure controller node
  - Install and configure compute node
  - Verify operation
  - Next steps
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service

Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Install and configure controller node

UPDATED: 2016-08-13 23:58

Contents

- Prerequisites
- Configure networking options
- Configure the metadata agent
- Configure Compute to use Networking
- Finalize installation

Prerequisites

Before you configure the OpenStack Networking (neutron) service, you must create a database, service credentials, and API endpoints.

1. To create the database, complete these steps:
   - Use the database access client to connect to the database server as the `root` user:

     ```
     $ mysql -u root -p
     ```

   - Create the `neutron` database:

     ```
     CREATE DATABASE neutron;
     ```

```
```

- Grant proper access to the `neutron` database, replacing `NEUTRON_DBPASS` with a suitable password:

```
GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' \
  IDENTIFIED BY 'NEUTRON_DBPASS';
GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' \
  IDENTIFIED BY 'NEUTRON_DBPASS';
```

- Exit the database access client.

2. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

3. To create the service credentials, complete these steps:
   - Create the `neutron` user:

```
$ openstack user create --domain default --password-prompt neutron
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | e0353a670a9e496da891347c589539e9 |
| enabled   | True                             |
| id        | b20a6692f77b4258926881bf831eb683 |
| name      | neutron                          |
+-----------+----------------------------------+
```

   - Add the `admin` role to the `neutron` user:

```
$ openstack role add --project service --user neutron admin
```

> **Note**
>
> This command provides no output.

   - Create the `neutron` service entity:

```
$ openstack service create --name neutron \
  --description "OpenStack Networking" network
```

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Networking             |
| enabled     | True                             |
| id          | f71529314dab4a4d8eca427e701d209e |
| name        | neutron                          |
| type        | network                          |
+-------------+----------------------------------+
```

4. Create the Networking service API endpoints:

```
$ openstack endpoint create --region RegionOne \
  network public http://controller:9696
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 85d80a6d02fc4b7683f611d7fc1493a3 |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | f71529314dab4a4d8eca427e701d209e |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+

$ openstack endpoint create --region RegionOne \
  network internal http://controller:9696
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 09753b537ac74422a68d2d791cf3714f |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | f71529314dab4a4d8eca427e701d209e |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+

$ openstack endpoint create --region RegionOne \
  network admin http://controller:9696
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 1ee14289c9374dffb5db92a5c112fc4e |
| interface    | admin                            |
```

```
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | f71529314dab4a4d8eca427e701d209e |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+
```

Configure networking options

You can deploy the Networking service using one of two architectures represented by options 1 and 2.

Option 1 deploys the simplest possible architecture that only supports attaching instances to provider (external) networks. No self-service (private) networks, routers, or floating IP addresses. Only the `admin` or other privileged user can manage provider networks.

Option 2 augments option 1 with layer-3 services that support attaching instances to self-service networks. The `demo` or other unprivileged user can manage self-service networks including routers that provide connectivity between self-service and provider networks. Additionally, floating IP addresses provide connectivity to instances using self-service networks from external networks such as the Internet.

Self-service networks typically use overlay networks. Overlay network protocols such as VXLAN include additional headers that increase overhead and decrease space available for the payload or user data. Without knowledge of the virtual network infrastructure, instances attempt to send packets using the default Ethernet *maximum transmission unit (MTU)* of 1500 bytes. The Networking service automatically provides the correct MTU value to instances via DHCP. However, some cloud images do not use DHCP or ignore the DHCP MTU option and require configuration using metadata or a script.

> **Note**
>
> Option 2 also supports attaching instances to provider networks.

Choose one of the following networking options to configure services specific to it. Afterwards, return here and proceed to *Configure the metadata agent*.

- Networking Option 1: Provider networks
- Networking Option 2: Self-service networks

Configure the metadata agent

The *metadata agent* provides configuration information such as credentials to instances.

- Edit the `/etc/neutron/metadata_agent.ini` file and complete the following actions:
  - In the `[DEFAULT]` section, configure the metadata host and shared secret:

```
[DEFAULT]
...
nova_metadata_ip = controller
metadata_proxy_shared_secret = METADATA_SECRET
```

Replace `METADATA_SECRET` with a suitable secret for the metadata proxy.

Configure Compute to use Networking

- Edit the `/etc/nova/nova.conf` file and perform the following actions:
  - In the `[neutron]` section, configure access parameters, enable the metadata proxy, and configure the secret:

```
[neutron]
...
url = http://controller:9696
auth_url = http://controller:35357
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS

service_metadata_proxy = True
metadata_proxy_shared_secret = METADATA_SECRET
```

Replace `NEUTRON_PASS` with the password you chose for the `neutron` user in the Identity service.

Replace `METADATA_SECRET` with the secret you chose for the metadata proxy.

Finalize installation

1. The Networking service initialization scripts expect a symbolic link `/etc/neutron/plugin.ini` pointing to the ML2 plug-in configuration file, `/etc/neutron/plugins/ml2/ml2_conf.ini`. If this symbolic link does not exist, create it using the following command:

```
# ln -s /etc/neutron/plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
```

2. Populate the database:

```
# su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/neutron.conf \
  --config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron
```

> **Note**
>
> Database population occurs later for Networking because the script requires complete server and plug-in configuration files.

3. Restart the Compute API service:

```
# systemctl restart openstack-nova-api.service
```

4. Start the Networking services and configure them to start when the system boots.

   For both networking options:

```
# systemctl enable neutron-server.service \
  neutron-linuxbridge-agent.service neutron-dhcp-agent.service \
  neutron-metadata-agent.service
# systemctl start neutron-server.service \
  neutron-linuxbridge-agent.service neutron-dhcp-agent.service \
  neutron-metadata-agent.service
```

   For networking option 2, also enable and start the layer-3 service:

```
# systemctl enable neutron-l3-agent.service
# systemctl start neutron-l3-agent.service
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
  - Networking service overview
  - Networking (neutron) concepts
  - Install and configure controller node
  - Install and configure compute node
  - Verify operation
  - Next steps
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events

Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack

SEARCH

Install and configure compute node

UPDATED: 2016-08-13 23:58

Contents

- Install the components
- Configure the common component
- Configure networking options
- Configure Compute to use Networking
- Finalize installation

The compute node handles connectivity and *security groups* for instances.

Install the components

```
# yum install openstack-neutron-linuxbridge ebtables ipset
```

Configure the common component

The Networking common component configuration includes the authentication mechanism, message queue, and plug-in.

> Note

> Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (`...`) in the configuration snippets indicates potential default configuration options that you should retain.

- Edit the `/etc/neutron/neutron.conf` file and complete the following actions:
  - In the `[database]` section, comment out any `connection` options because compute nodes do not directly access the database.
  - In the `[DEFAULT]` and `[oslo_messaging_rabbit]` sections, configure RabbitMQ message queue access:

```
[DEFAULT]
...
rpc_backend = rabbit

[oslo_messaging_rabbit]
...
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS
```

    Replace `RABBIT_PASS` with the password you chose for the `openstack` account in RabbitMQ.

  - In the `[DEFAULT]` and `[keystone_authtoken]` sections, configure Identity service access:

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = NEUTRON_PASS
```

    Replace `NEUTRON_PASS` with the password you chose for the `neutron` user in the Identity service.

    > Note
    >
    > Comment out or remove any other options in the `[keystone_authtoken]` section.

- In the `[oslo_concurrency]` section, configure the lock path:

```
[oslo_concurrency]
...
lock_path = /var/lib/neutron/tmp
```

Configure networking options

Choose the same networking option that you chose for the controller node to configure services specific to it. Afterwards, return here and proceed to *Configure Compute to use Networking*.

- Networking Option 1: Provider networks
- Networking Option 2: Self-service networks

Configure Compute to use Networking

- Edit the `/etc/nova/nova.conf` file and complete the following actions:
  - In the `[neutron]` section, configure access parameters:

```
[neutron]
...
url = http://controller:9696
auth_url = http://controller:35357
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
```

Replace `NEUTRON_PASS` with the password you chose for the `neutron` user in the Identity service.

Finalize installation

1. Restart the Compute service:

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

2. Start the Linux bridge agent and configure it to start when the system boots:

```
# systemctl enable neutron-linuxbridge-agent.service
# systemctl start neutron-linuxbridge-agent.service
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
  - Networking service overview
  - Networking (neutron) concepts
  - Install and configure controller node
  - Install and configure compute node
  - Verify operation
  - Next steps
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service

- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Verify operation

UPDATED: 2016-08-13 23:58

> **Note**
>
> Perform these commands on the controller node.

1. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

2. List loaded extensions to verify successful launch of the `neutron-server` process:

```
$ neutron ext-list
+---------------------------+-----------------------------------------------+
| alias                     | name                                          |
+---------------------------+-----------------------------------------------+
| default-subnetpools       | Default Subnetpools                           |
| network-ip-availability   | Network IP Availability                       |
| network_availability_zone | Network Availability Zone                     |
| auto-allocated-topology   | Auto Allocated Topology Services              |
| ext-gw-mode               | Neutron L3 Configurable external gateway mode |
| binding                   | Port Binding                                  |
```

```
| agent                     | agent                                          |
| subnet_allocation         | Subnet Allocation                              |
| l3_agent_scheduler        | L3 Agent Scheduler                             |
| tag                       | Tag support                                    |
| external-net              | Neutron external network                       |
| net-mtu                   | Network MTU                                    |
| availability_zone         | Availability Zone                              |
| quotas                    | Quota management support                       |
| l3-ha                     | HA Router extension                            |
| flavors                   | Neutron Service Flavors                        |
| provider                  | Provider Network                               |
| multi-provider            | Multi Provider Network                         |
| address-scope             | Address scope                                  |
| extraroute                | Neutron Extra Route                            |
| timestamp_core            | Time Stamp Fields addition for core resources  |
| router                    | Neutron L3 Router                              |
| extra_dhcp_opt            | Neutron Extra DHCP opts                        |
| dns-integration           | DNS Integration                                |
| security-group            | security-group                                 |
| dhcp_agent_scheduler      | DHCP Agent Scheduler                           |
| router_availability_zone  | Router Availability Zone                       |
| rbac-policies             | RBAC Policies                                  |
| standard-attr-description | standard-attr-description                      |
| port-security             | Port Security                                  |
| allowed-address-pairs     | Allowed Address Pairs                          |
| dvr                       | Distributed Virtual Router                     |
+---------------------------+------------------------------------------------+
```

> **Note**
>
> Actual output may differ slightly from this example.

Use the verification section for the networking option that you chose to deploy.

- Networking Option 1: Provider networks
- Networking Option 2: Self-service networks

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
  - Networking service overview
  - Networking (neutron) concepts
  - Install and configure controller node
  - Install and configure compute node
  - Verify operation
  - Next steps
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects

OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

Next steps

UPDATED: 2016-08-13 23:58

Your OpenStack environment now includes the core components necessary to launch a basic instance. You can *Launch an instance* or add more OpenStack services to your environment.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
  - Networking service overview
  - Networking (neutron) concepts
  - Install and configure controller node
  - Install and configure compute node
  - Verify operation
  - Next steps
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Dashboard

UPDATED: 2016-08-13 23:58

- Install and configure
  - Install and configure components
  - Finalize installation
- Verify operation
- Next steps

The Dashboard (horizon) is a web interface that enables cloud administrators and users to manage various OpenStack resources and services.

This example deployment uses an Apache web server.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
  - Install and configure
  - Verify operation
  - Next steps
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Install and configure

UPDATED: 2016-08-13 23:58

Contents

Install and configure components
Finalize installation

This section describes how to install and configure the dashboard on the controller node.

The dashboard relies on functional core services including Identity, Image service, Compute, and either Networking (neutron) or legacy networking (nova-network). Environments with stand-alone services such as Object Storage cannot use the dashboard. For more information, see the developer documentation.

> Note
>
> This section assumes proper installation, configuration, and operation of the Identity service using the Apache HTTP server and Memcached service as described in the *Install and configure the Identity service* section.

Install and configure components

> Note

> Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (`...`) in the configuration snippets indicates potential default configuration options that you should retain.

1. Install the packages:

```
# yum install openstack-dashboard
```

2. Edit the `/etc/openstack-dashboard/local_settings` file and complete the following actions:
   - Configure the dashboard to use OpenStack services on the `controller` node:

```
OPENSTACK_HOST = "controller"
```

   - Allow all hosts to access the dashboard:

```
ALLOWED_HOSTS = ['*', ]
```

   - Configure the `memcached` session storage service:

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'

CACHES = {
    'default': {
         'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
         'LOCATION': 'controller:11211',
    }
}
```

> Note
>
> Comment out any other session storage configuration.

   - Enable the Identity API version 3:

```
OPENSTACK_KEYSTONE_URL = "http://%s:5000/v3" % OPENSTACK_HOST
```

   - Enable support for domains:

```
```

```
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
```

- Configure API versions:

```
OPENSTACK_API_VERSIONS = {
    "identity": 3,
    "image": 2,
    "volume": 2,
}
```

- Configure `default` as the default domain for users that you create via the dashboard:

```
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "default"
```

- Configure `user` as the default role for users that you create via the dashboard:

```
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
```

- If you chose networking option 1, disable support for layer-3 networking services:

```
OPENSTACK_NEUTRON_NETWORK = {
    ...
    'enable_router': False,
    'enable_quotas': False,
    'enable_distributed_router': False,
    'enable_ha_router': False,
    'enable_lb': False,
    'enable_firewall': False,
    'enable_vpn': False,
    'enable_fip_topology_check': False,
}
```

- Optionally, configure the time zone:

```
TIME_ZONE = "TIME_ZONE"
```

  Replace `TIME_ZONE` with an appropriate time zone identifier. For more information, see the list of time zones.

Finalize installation

- Restart the web server and session storage service:

```
```

```
# systemctl restart httpd.service memcached.service
```

> **Note**
>
> The `systemctl restart` command starts each service if not currently running.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
  - Install and configure
  - Verify operation
  - Next steps
- Block Storage service
- Shared File Systems service
- Object Storage service

- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Verify operation

UPDATED: 2016-08-13 23:58

Verify operation of the dashboard.

Access the dashboard using a web browser at `http://controller/dashboard`.

Authenticate using `admin` or `demo` user and `default` domain credentials.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
  - Install and configure
  - Verify operation
  - Next steps
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Next steps

UPDATED: 2016-08-13 23:58

Your OpenStack environment now includes the dashboard. You can *Launch an instance* or add more services to your environment.

After you install and configure the dashboard, you can complete the following tasks:

- Provide users with a public IP address, a username, and a password so they can access the dashboard through a web browser. In case of any SSL certificate connection problems, point the server IP address to a domain name, and give users access.
- Customize your dashboard. See section Customize the dashboard.
- Set up session storage. See Set up session storage for the dashboard.
- To use the VNC client with the dashboard, the browser must support HTML5 Canvas and HTML5 WebSockets.

  For details about browsers that support noVNC, see README and browser support.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
  - Install and configure
  - Verify operation
  - Next steps
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog

News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack

SEARCH

Block Storage service

UPDATED: 2016-08-13 23:58

- Block Storage service overview
- Install and configure controller node
  - Prerequisites
  - Install and configure components
  - Configure Compute to use Block Storage
  - Finalize installation
- Install and configure a storage node
  - Prerequisites
  - Install and configure components
  - Finalize installation
- Verify operation
- Next steps

The Block Storage service (cinder) provides block storage devices to guest instances. The method in which the storage is provisioned and consumed is determined by the Block Storage driver, or drivers in the case of a multi-backend configuration. There are a variety of drivers that are available: NAS/SAN, NFS, iSCSI, Ceph, and more.

The Block Storage API and scheduler services typically run on the controller nodes. Depending upon the drivers used, the volume service can run on controllers, compute nodes, or standalone storage nodes.

For more information, see the Configuration Reference.

Note

> This chapter omits the backup manager because it depends on the Object Storage service.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
  - Block Storage service overview
  - Install and configure controller node
  - Install and configure a storage node
  - Verify operation
  - Next steps
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance

- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Block Storage service overview

UPDATED: 2016-08-13 23:58

The OpenStack Block Storage service (cinder) adds persistent storage to a virtual machine. Block Storage provides an infrastructure for managing volumes, and interacts with OpenStack Compute to provide volumes for instances. The service also enables management of volume snapshots, and volume types.

The Block Storage service consists of the following components:

`cinder-api`
: Accepts API requests, and routes them to the `cinder-volume` for action.

`cinder-volume`
: Interacts directly with the Block Storage service, and processes such as the `cinder-scheduler`. It also interacts with these processes through a message queue. The `cinder-volume` service responds to read and write requests sent to the Block Storage service to maintain state. It can interact with a variety of storage providers through a driver architecture.

`cinder-scheduler daemon`
: Selects the optimal storage provider node on which to create the volume. A similar component to the `nova-scheduler`.

`cinder-backup daemon`
: The `cinder-backup` service provides backing up volumes of any type to a backup storage provider. Like the `cinder-volume` service, it can interact with a variety of storage providers through a driver architecture.

`Messaging queue`
: Routes information between the Block Storage processes.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
  - Block Storage service overview
  - Install and configure controller node
  - Install and configure a storage node
  - Verify operation
  - Next steps
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Install and configure controller node

UPDATED: 2016-08-13 23:58

Contents

- Prerequisites
- Install and configure components
- Configure Compute to use Block Storage
- Finalize installation

This section describes how to install and configure the Block Storage service, code-named cinder, on the controller node. This service requires at least one additional storage node that provides volumes to instances.

Prerequisites

Before you install and configure the Block Storage service, you must create a database, service credentials, and API endpoints.

1. To create the database, complete these steps:
   - Use the database access client to connect to the database server as the `root` user:

     ```
     $ mysql -u root -p
     ```

   - Create the `cinder` database:

```
CREATE DATABASE cinder;
```

- Grant proper access to the `cinder` database:

```
GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' \
  IDENTIFIED BY 'CINDER_DBPASS';
GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' \
  IDENTIFIED BY 'CINDER_DBPASS';
```

  Replace `CINDER_DBPASS` with a suitable password.

- Exit the database access client.

2. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

3. To create the service credentials, complete these steps:

   - Create a `cinder` user:

```
$ openstack user create --domain default --password-prompt cinder
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | e0353a670a9e496da891347c589539e9 |
| enabled   | True                             |
| id        | bb279f8ffc444637af38811a5e1f0562 |
| name      | cinder                           |
+-----------+----------------------------------+
```

   - Add the `admin` role to the `cinder` user:

```
$ openstack role add --project service --user cinder admin
```

> **Note**
>
> This command provides no output.

   - Create the `cinder` and `cinderv2` service entities:

```
$ openstack service create --name cinder \
  --description "OpenStack Block Storage" volume
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Block Storage          |
| enabled     | True                             |
| id          | ab3bbbef780845a1a283490d281e7fda |
| name        | cinder                           |
| type        | volume                           |
+-------------+----------------------------------+
```

```
$ openstack service create --name cinderv2 \
  --description "OpenStack Block Storage" volumev2
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Block Storage          |
| enabled     | True                             |
| id          | eb9fd245bdbc414695952e93f29fe3ac |
| name        | cinderv2                         |
| type        | volumev2                         |
+-------------+----------------------------------+
```

> **Note**
>
> The Block Storage services require two service entities.

4. Create the Block Storage service API endpoints:

```
$ openstack endpoint create --region RegionOne \
  volume public http://controller:8776/v1/%\(tenant_id\)s
  +--------------+-----------------------------------------+
  | Field        | Value                                   |
  +--------------+-----------------------------------------+
  | enabled      | True                                    |
  | id           | 03fa2c90153546c295bf30ca86b1344b        |
  | interface    | public                                  |
  | region       | RegionOne                               |
  | region_id    | RegionOne                               |
  | service_id   | ab3bbbef780845a1a283490d281e7fda        |
  | service_name | cinder                                  |
  | service_type | volume                                  |
  | url          | http://controller:8776/v1/%(tenant_id)s |
  +--------------+-----------------------------------------+
```

```
$ openstack endpoint create --region RegionOne \
  volume internal http://controller:8776/v1/%\(tenant_id\)s
  +--------------+-----------------------------------------+
  | Field        | Value                                   |
  +--------------+-----------------------------------------+
  | enabled      | True                                    |
  | id           | 94f684395d1b41068c70e4ecb11364b2        |
  | interface    | internal                                |
  | region       | RegionOne                               |
  | region_id    | RegionOne                               |
  | service_id   | ab3bbbef780845a1a283490d281e7fda        |
  | service_name | cinder                                  |
  | service_type | volume                                  |
  | url          | http://controller:8776/v1/%(tenant_id)s |
  +--------------+-----------------------------------------+

$ openstack endpoint create --region RegionOne \
  volume admin http://controller:8776/v1/%\(tenant_id\)s
  +--------------+-----------------------------------------+
  | Field        | Value                                   |
  +--------------+-----------------------------------------+
  | enabled      | True                                    |
  | id           | 4511c28a0f9840c78bacb25f10f62c98        |
  | interface    | admin                                   |
  | region       | RegionOne                               |
  | region_id    | RegionOne                               |
  | service_id   | ab3bbbef780845a1a283490d281e7fda        |
  | service_name | cinder                                  |
  | service_type | volume                                  |
  | url          | http://controller:8776/v1/%(tenant_id)s |
  +--------------+-----------------------------------------+
```

```
$ openstack endpoint create --region RegionOne \
  volumev2 public http://controller:8776/v2/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 513e73819e14460fb904163f41ef3759        |
| interface    | public                                  |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | eb9fd245bdbc414695952e93f29fe3ac        |
| service_name | cinderv2                                |
| service_type | volumev2                                |
| url          | http://controller:8776/v2/%(tenant_id)s |
+--------------+-----------------------------------------+

$ openstack endpoint create --region RegionOne \
  volumev2 internal http://controller:8776/v2/%\(tenant_id\)s
+--------------+-----------------------------------------+
```

```
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | 6436a8a23d014cfdb69c586eff146a32         |
| interface    | internal                                 |
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | eb9fd245bdbc414695952e93f29fe3ac         |
| service_name | cinderv2                                 |
| service_type | volumev2                                 |
| url          | http://controller:8776/v2/%(tenant_id)s  |
+--------------+------------------------------------------+

$ openstack endpoint create --region RegionOne \
  volumev2 admin http://controller:8776/v2/%\(tenant_id\)s
+--------------+------------------------------------------+
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | e652cf84dd334f359ae9b045a2c91d96         |
| interface    | admin                                    |
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | eb9fd245bdbc414695952e93f29fe3ac         |
| service_name | cinderv2                                 |
| service_type | volumev2                                 |
| url          | http://controller:8776/v2/%(tenant_id)s  |
+--------------+------------------------------------------+
```

> **Note**
>
> The Block Storage services require endpoints for each service entity.

Install and configure components

1. Install the packages:

```
# yum install openstack-cinder
```

2. Edit the `/etc/cinder/cinder.conf` file and complete the following actions:
   - In the `[database]` section, configure database access:

```
[database]
...
connection = mysql+pymysql://cinder:CINDER_DBPASS@controller/cinder
```

```
```

Replace `CINDER_DBPASS` with the password you chose for the Block Storage database.

- In the `[DEFAULT]` and `[oslo_messaging_rabbit]` sections, configure `RabbitMQ` message queue access:

```
[DEFAULT]
...
rpc_backend = rabbit

[oslo_messaging_rabbit]
...
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS
```

Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.

- In the `[DEFAULT]` and `[keystone_authtoken]` sections, configure Identity service access:

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = cinder
password = CINDER_PASS
```

Replace `CINDER_PASS` with the password you chose for the `cinder` user in the Identity service.

> **Note**
>
> Comment out or remove any other options in the `[keystone_authtoken]` section.

- In the `[DEFAULT]` section, configure the `my_ip` option to use the management interface IP address of the controller node:

```
[DEFAULT]
```

```
...
my_ip = 10.0.0.11
```

- In the `[oslo_concurrency]` section, configure the lock path:

```
[oslo_concurrency]
...
lock_path = /var/lib/cinder/tmp
```

3. Populate the Block Storage database:

```
# su -s /bin/sh -c "cinder-manage db sync" cinder
```

> **Note**
>
> Ignore any deprecation messages in this output.

Configure Compute to use Block Storage

- Edit the `/etc/nova/nova.conf` file and add the following to it:

```
[cinder]
os_region_name = RegionOne
```

Finalize installation

1. Restart the Compute API service:

```
# systemctl restart openstack-nova-api.service
```

2. Start the Block Storage services and configure them to start when the system boots:

```
# systemctl enable openstack-cinder-api.service openstack-cinder-scheduler.service
# systemctl start openstack-cinder-api.service openstack-cinder-scheduler.service
```

OpenStack Documentation

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
  - Block Storage service overview
  - Install and configure controller node
  - Install and configure a storage node
  - Verify operation
  - Next steps
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Install and configure a storage node

UPDATED: 2016-08-13 23:58

Contents

Prerequisites
Install and configure components
Finalize installation

This section describes how to install and configure storage nodes for the Block Storage service. For simplicity, this configuration references one storage node with an empty local block storage device. The instructions use `/dev/sdb`, but you can substitute a different value for your particular node.

The service provisions logical volumes on this device using the *LVM* driver and provides them to instances via *iSCSI* transport. You can follow these instructions with minor modifications to horizontally scale your environment with additional storage nodes.

Prerequisites

Before you install and configure the Block Storage service on the storage node, you must prepare the storage device.

> **Note**
>
> Perform these steps on the storage node.

1. Install the supporting utility packages:
   - Install the LVM packages:

     ```
     # yum install lvm2
     ```

   - Start the LVM metadata service and configure it to start when the system boots:

     ```
     # systemctl enable lvm2-lvmetad.service
     # systemctl start lvm2-lvmetad.service
     ```

> **Note**
>
> Some distributions include LVM by default.

2. Create the LVM physical volume `/dev/sdb`:

   ```
   # pvcreate /dev/sdb
   Physical volume "/dev/sdb" successfully created
   ```

3. Create the LVM volume group `cinder-volumes`:

   ```
   # vgcreate cinder-volumes /dev/sdb
   Volume group "cinder-volumes" successfully created
   ```

   The Block Storage service creates logical volumes in this volume group.

4. Only instances can access Block Storage volumes. However, the underlying operating system manages the devices associated with the volumes. By default, the LVM volume scanning tool scans the `/dev` directory for block storage devices that contain volumes. If projects use LVM on their volumes, the scanning tool detects these volumes and attempts to cache them which can cause a variety of problems with both the underlying operating system and project volumes. You must reconfigure LVM to scan only the devices that contain the `cinder-volume` volume group. Edit the `/etc/lvm/lvm.conf` file and complete the following actions:
   - In the `devices` section, add a filter that accepts the `/dev/sdb` device and rejects all other devices:

     ```
     devices {
     ...
     filter = [ "a/sdb/", "r/.*/"]
     ```

Each item in the filter array begins with `a` for `accept` or `r` for `reject` and includes a regular expression for the device name. The array must end with `r/.*/` to reject any remaining devices. You can use the `vgs -vvvv` command to test filters.

> **Warning**
>
> If your storage nodes use LVM on the operating system disk, you must also add the associated device to the filter. For example, if the `/dev/sda` device contains the operating system:
>
> ```
> filter = [ "a/sda/", "a/sdb/", "r/.*/"]
> ```
>
> Similarly, if your compute nodes use LVM on the operating system disk, you must also modify the filter in the `/etc/lvm/lvm.conf` file on those nodes to include only the operating system disk. For example, if the `/dev/sda` device contains the operating system:
>
> ```
> filter = [ "a/sda/", "r/.*/"]
> ```

Install and configure components

1. Install the packages:

   ```
   # yum install openstack-cinder targetcli
   ```

2. Edit the `/etc/cinder/cinder.conf` file and complete the following actions:

   - In the `[database]` section, configure database access:

     ```
     [database]
     ...
     connection = mysql+pymysql://cinder:CINDER_DBPASS@controller/cinder
     ```

     Replace `CINDER_DBPASS` with the password you chose for the Block Storage database.

   - In the `[DEFAULT]` and `[oslo_messaging_rabbit]` sections, configure `RabbitMQ` message queue access:

     ```
     [DEFAULT]
     ...
     rpc_backend = rabbit

     [oslo_messaging_rabbit]
     ```

```
...
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS
```

Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.

- In the `[DEFAULT]` and `[keystone_authtoken]` sections, configure Identity service access:

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = cinder
password = CINDER_PASS
```

Replace `CINDER_PASS` with the password you chose for the `cinder` user in the Identity service.

> **Note**
>
> Comment out or remove any other options in the `[keystone_authtoken]` section.

- In the `[DEFAULT]` section, configure the `my_ip` option:

```
[DEFAULT]
...
my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
```

Replace `MANAGEMENT_INTERFACE_IP_ADDRESS` with the IP address of the management network interface on your storage node, typically 10.0.0.41 for the first node in the *example architecture*.

- In the `[lvm]` section, configure the LVM back end with the LVM driver, `cinder-volumes` volume group, iSCSI protocol, and appropriate iSCSI service:

```
[lvm]
...
volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
```

```
volume_group = cinder-volumes
iscsi_protocol = iscsi
iscsi_helper = lioadm
```

- In the `[DEFAULT]` section, enable the LVM back end:

```
[DEFAULT]
...
enabled_backends = lvm
```

> **Note**
>
> Back-end names are arbitrary. As an example, this guide uses the name of the driver as the name of the back end.

- In the `[DEFAULT]` section, configure the location of the Image service API:

```
[DEFAULT]
...
glance_api_servers = http://controller:9292
```

- In the `[oslo_concurrency]` section, configure the lock path:

```
[oslo_concurrency]
...
lock_path = /var/lib/cinder/tmp
```

Finalize installation

- Start the Block Storage volume service including its dependencies and configure them to start when the system boots:

```
# systemctl enable openstack-cinder-volume.service target.service
# systemctl start openstack-cinder-volume.service target.service
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
  - Block Storage service overview
  - Install and configure controller node
  - Install and configure a storage node
  - Verify operation
  - Next steps
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack

SEARCH

Verify operation

UPDATED: 2016-08-13 23:58

Verify operation of the Block Storage service.

> **Note**
>
> Perform these commands on the controller node.

1. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

2. List service components to verify successful launch of each process:

```
$ cinder service-list
+------------------+------------+------+---------+-------+----------------------------
----+-----------------+
|      Binary      |    Host    | Zone |  Status | State |         Updated_at         | Dis
abled Reason |
+------------------+------------+------+---------+-------+----------------------------
----+-----------------+
```

```
| cinder-scheduler | controller | nova | enabled |   up   | 2014-10-18T01:30:54.000000 |
None      |
| cinder-volume    | block1@lvm | nova | enabled |   up   | 2014-10-18T01:30:57.000000 |
None      |
+------------------+------------+------+---------+-------+----------------------------
----+-----------------+
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
  - Block Storage service overview
  - Install and configure controller node
  - Install and configure a storage node
  - Verify operation
  - Next steps
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service

- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Next steps

UPDATED: 2016-08-13 23:58

Your OpenStack environment now includes Block Storage. You can *launch an instance* or add more services to your environment in the following chapters.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
  - Block Storage service overview
  - Install and configure controller node
  - Install and configure a storage node
  - Verify operation
  - Next steps
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies

Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

Shared File Systems service

UPDATED: 2016-08-13 23:58

- Shared File Systems service overview
- Install and configure controller node
  - Prerequisites
  - Install and configure components
  - Finalize installation
- Install and configure a share node
  - Install and configure components
  - Configure share server management support options
    - Shared File Systems Option 1: No driver support for share servers management
      - Prerequisites
      - Configure components
    - Shared File Systems Option 2: Driver support for share servers management
      - Prerequisites
      - Configure components
  - Finalize installation
- Verify operation
- Next steps

The Shared File Systems service (manila) provides coordinated access to shared or distributed file systems. The method in which the share is provisioned and consumed is determined by the Shared File Systems driver, or drivers in the case of a multi-driver configuration. There are a variety of drivers that support NFS, CIFS, HDFS and/or protocols as well.

The Shared File Systems API and scheduler services typically run on the controller nodes. Depending upon the drivers used, the share service can run on controllers, compute nodes, or storage nodes.

For more information, see the Configuration Reference.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
  - Shared File Systems service overview
  - Install and configure controller node
  - Install and configure a share node
  - Verify operation
  - Next steps
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Shared File Systems service overview

UPDATED: 2016-08-13 23:58

The OpenStack Shared File Systems service (manila) provides file storage to a virtual machine. The Shared File Systems service provides an infrastructure for managing and provisioning of file shares. The service also enables management of share types as well as share snapshots if a driver supports them.

The Shared File Systems service consists of the following components:

manila-api
: A WSGI app that authenticates and routes requests throughout the Shared File Systems service. It supports the OpenStack APIs.

manila-data
: A standalone service whose purpose is to receive requests, process data operations such as copying, share migration or backup, and send back a response after an operation has been completed.

manila-scheduler
: Schedules and routes requests to the appropriate share service. The scheduler uses configurable filters and weighers to route requests. The Filter Scheduler is the default and enables filters on things like Capacity, Availability Zone, Share Types, and Capabilities as well as custom filters.

manila-share
: Manages back-end devices that provide shared file systems. A manila-share process can run in one of two modes, with or without handling of share servers. Share servers export file shares via share networks. When share servers are not used, the networking requirements are handled outside of Manila.

Messaging queue
: Routes information between the Shared File Systems processes.

For more information, see Configuration Reference Guide.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
  - Shared File Systems service overview
  - Install and configure controller node
  - Install and configure a share node
  - Verify operation
  - Next steps
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Install and configure controller node

UPDATED: 2016-08-13 23:58

Contents

Prerequisites
Install and configure components
Finalize installation

This section describes how to install and configure the Shared File Systems service, code-named manila, on the controller node. This service requires at least one additional share node that manages file storage drivers.

Prerequisites

Before you install and configure the Share File System service, you must create a database, service credentials, and API endpoints.

1. To create the database, complete these steps:
   - Use the database access client to connect to the database server as the `root` user:

     ```
     $ mysql -u root -p
     ```

   - Create the `manila` database:

     ```
     ```

```
CREATE DATABASE manila;
```

- Grant proper access to the `manila` database:

```
GRANT ALL PRIVILEGES ON manila.* TO 'manila'@'localhost' \
  IDENTIFIED BY 'MANILA_DBPASS';
GRANT ALL PRIVILEGES ON manila.* TO 'manila'@'%' \
  IDENTIFIED BY 'MANILA_DBPASS';
```

  Replace `MANILA_DBPASS` with a suitable password.

- Exit the database access client.

2. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

3. To create the service credentials, complete these steps:

   - Create a `manila` user:

```
$ openstack user create --domain default --password-prompt manila
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | e0353a670a9e496da891347c589539e9 |
| enabled   | True                             |
| id        | 83a3990fc2144100ba0e2e23886d8acc |
| name      | manila                           |
+-----------+----------------------------------+
```

   - Add the `admin` role to the `manila` user:

```
$ openstack role add --project service --user manila admin
```

> **Note**
>
> This command provides no output.

   - Create the `manila` and `manilav2` service entities:

```
```

```
$ openstack service create --name manila \
  --description "OpenStack Shared File Systems" share
  +-------------+----------------------------------+
  | Field       | Value                            |
  +-------------+----------------------------------+
  | description | OpenStack Shared File Systems    |
  | enabled     | True                             |
  | id          | 82378b5a16b340aa9cc790cdd46a03ba |
  | name        | manila                           |
  | type        | share                            |
  +-------------+----------------------------------+
```

```
$ openstack service create --name manilav2 \
  --description "OpenStack Shared File Systems" sharev2
  +-------------+----------------------------------+
  | Field       | Value                            |
  +-------------+----------------------------------+
  | description | OpenStack Shared File Systems    |
  | enabled     | True                             |
  | id          | 30d92a97a81a4e5d8fd97a32bafd7b88 |
  | name        | manilav2                         |
  | type        | sharev2                          |
  +-------------+----------------------------------+
```

> **Note**
>
> The Share File System services require two service entities.

4. Create the Shared File Systems service API endpoints:

```
$ openstack endpoint create --region RegionOne \
  share public http://controller:8786/v1/%\(tenant_id\)s
  +--------------+-----------------------------------------+
  | Field        | Value                                   |
  +--------------+-----------------------------------------+
  | enabled      | True                                    |
  | id           | 0bd2bbf8d28b433aaea56a254c69f69d        |
  | interface    | public                                  |
  | region       | RegionOne                               |
  | region_id    | RegionOne                               |
  | service_id   | 82378b5a16b340aa9cc790cdd46a03ba        |
  | service_name | manila                                  |
  | service_type | share                                   |
  | url          | http://controller:8786/v1/%(tenant_id)s |
  +--------------+-----------------------------------------+
```

```
$ openstack endpoint create --region RegionOne \
  share internal http://controller:8786/v1/%\(tenant_id\)s
  +--------------+-----------------------------------------+
  | Field        | Value                                   |
  +--------------+-----------------------------------------+
  | enabled      | True                                    |
  | id           | a2859b5732cc48b5b083dd36dafb6fd9        |
  | interface    | internal                                |
  | region       | RegionOne                               |
  | region_id    | RegionOne                               |
  | service_id   | 82378b5a16b340aa9cc790cdd46a03ba        |
  | service_name | manila                                  |
  | service_type | share                                   |
  | url          | http://controller:8786/v1/%(tenant_id)s |
  +--------------+-----------------------------------------+

$ openstack endpoint create --region RegionOne \
  share admin http://controller:8786/v1/%\(tenant_id\)s
  +--------------+-----------------------------------------+
  | Field        | Value                                   |
  +--------------+-----------------------------------------+
  | enabled      | True                                    |
  | id           | f7f46df93a374cc49c0121bef41da03c        |
  | interface    | admin                                   |
  | region       | RegionOne                               |
  | region_id    | RegionOne                               |
  | service_id   | 82378b5a16b340aa9cc790cdd46a03ba        |
  | service_name | manila                                  |
  | service_type | share                                   |
  | url          | http://controller:8786/v1/%(tenant_id)s |
  +--------------+-----------------------------------------+
```

```
$ openstack endpoint create --region RegionOne \
  sharev2 public http://controller:8786/v2/%\(tenant_id\)s
  +--------------+-----------------------------------------+
  | Field        | Value                                   |
  +--------------+-----------------------------------------+
  | enabled      | True                                    |
  | id           | d63cc0d358da4ea680178657291eddc1        |
  | interface    | public                                  |
  | region       | RegionOne                               |
  | region_id    | RegionOne                               |
  | service_id   | 30d92a97a81a4e5d8fd97a32bafd7b88        |
  | service_name | manilav2                                |
  | service_type | sharev2                                 |
  | url          | http://controller:8786/v2/%(tenant_id)s |
  +--------------+-----------------------------------------+

$ openstack endpoint create --region RegionOne \
  sharev2 internal http://controller:8786/v2/%\(tenant_id\)s
  +--------------+-----------------------------------------+
  | Field        | Value                                   |
```

```
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | afc86e5f50804008add349dba605da54        |
| interface    | internal                                |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | 30d92a97a81a4e5d8fd97a32bafd7b88        |
| service_name | manilav2                                |
| service_type | sharev2                                 |
| url          | http://controller:8786/v2/%(tenant_id)s |
+--------------+-----------------------------------------+

$ openstack endpoint create --region RegionOne \
  sharev2 admin http://controller:8786/v2/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | e814a0cec40546e98cf0c25a82498483        |
| interface    | admin                                   |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | 30d92a97a81a4e5d8fd97a32bafd7b88        |
| service_name | manilav2                                |
| service_type | sharev2                                 |
| url          | http://controller:8786/v2/%(tenant_id)s |
+--------------+-----------------------------------------+
```

> **Note**
>
> The Share File System services require endpoints for each service entity.

Install and configure components

1. Install the packages:

```
# yum install openstack-manila python-manilaclient
```

2. Edit the `/etc/manila/manila.conf` file and complete the following actions:
   - In the `[database]` section, configure database access:

```
[database]
...
connection = mysql+pymysql://manila:MANILA_DBPASS@controller/manila
```

Replace `MANILA_DBPASS` with the password you chose for the Share File System database.

- In the `[DEFAULT]` and `[oslo_messaging_rabbit]` sections, configure `RabbitMQ` message queue access:

```
[DEFAULT]
...
rpc_backend = rabbit

[oslo_messaging_rabbit]
...
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS
```

Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.

- In the `[DEFAULT]` section, set the following config values:

```
[DEFAULT]
...
default_share_type = default_share_type
rootwrap_config = /etc/manila/rootwrap.conf
```

- In the `[DEFAULT]` and `[keystone_authtoken]` sections, configure Identity service access:

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
memcached_servers = controller:11211
auth_uri = http://controller:5000
auth_url = http://controller:35357
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = manila
password = MANILA_PASS
```

Replace `MANILA_PASS` with the password you chose for the `manila` user in the Identity service.

- In the `[DEFAULT]` section, configure the `my_ip` option to use the management interface IP address of the controller node:

```
[DEFAULT]
```

```
...
my_ip = 10.0.0.11
```

- In the `[oslo_concurrency]` section, configure the lock path:

```
[oslo_concurrency]
...
lock_path = /var/lib/manila/tmp
```

3. Populate the Share File System database:

```
# su -s /bin/sh -c "manila-manage db sync" manila
```

> **Note**
>
> Ignore any deprecation messages in this output.

Finalize installation

- Start the Share File System services and configure them to start when the system boots:

```
# systemctl enable openstack-manila-api.service openstack-manila-scheduler.service
# systemctl start openstack-manila-api.service openstack-manila-scheduler.service
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
  - Shared File Systems service overview
  - Install and configure controller node
  - Install and configure a share node
  - Verify operation
  - Next steps
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events

Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

Install and configure a share node

UPDATED: 2016-08-13 23:58

Contents

Install and configure components
Configure share server management support options
Finalize installation

This section describes how to install and configure a share node for the Shared File Systems service.

Install and configure components

> **Note**
>
> Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (`...`) in the configuration snippets indicates potential default configuration options that you should retain.

1. Install the packages:

```
# yum install openstack-manila-share python2-PyMySQL
```

2. Edit the `/etc/manila/manila.conf` file and complete the following actions:

   - In the `[database]` section, configure database access:

     ```
     [database]
     ...
     connection = mysql://manila:MANILA_DBPASS@controller/manila
     ```

     Replace `MANILA_DBPASS` with the password you chose for the Share File System database.

   - In the `[DEFAULT]` and `[oslo_messaging_rabbit]` sections, configure `RabbitMQ` message queue access:

     ```
     [DEFAULT]
     ...
     rpc_backend = rabbit

     [oslo_messaging_rabbit]
     ...
     rabbit_host = controller
     rabbit_userid = openstack
     rabbit_password = RABBIT_PASS
     ```

     Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.

   - In the `[DEFAULT]` section, set the following config values:

     ```
     [DEFAULT]
     ...
     default_share_type = default_share_type
     rootwrap_config = /etc/manila/rootwrap.conf
     ```

   - In the `[DEFAULT]` and `[keystone_authtoken]` sections, configure Identity service access:

     ```
     [DEFAULT]
     ...
     auth_strategy = keystone

     [keystone_authtoken]
     ...
     memcached_servers = controller:11211
     auth_uri = http://controller:5000
     auth_url = http://controller:35357
     auth_type = password
     project_domain_name = default
     user_domain_name = default
     project_name = service
     username = manila
     ```

```
password = MANILA_PASS
```

Replace `MANILA_PASS` with the password you chose for the `manila` user in the Identity service.

- In the `[DEFAULT]` section, configure the `my_ip` option:

```
[DEFAULT]
...
my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
```

Replace `MANAGEMENT_INTERFACE_IP_ADDRESS` with the IP address of the management network interface on your share node, typically 10.0.0.41 for the first node in the *example architecture*.

- In the `[oslo_concurrency]` section, configure the lock path:

```
[oslo_concurrency]
...
lock_path = /var/lib/manila/tmp
```

Configure share server management support options

The share node can support two modes, with and without the handling of share servers. The mode depends on driver support.

Option 1 deploys the service without driver support for share management. In this mode, the service does not do anything related to networking. The operator must ensure network connectivity between instances and the NFS server. This option uses LVM driver that requires LVM and NFS packages as well as an additional disk for the `manila-share` LVM volume group.

Option 2 deploys the service with driver support for share management. In this mode, the service requires Compute (nova), Networking (neutron) and Block storage (cinder) services for managing share servers. The information used for creating share servers is configured as share networks. This option uses the generic driver with the handling of share servers capacity and requires attaching the `selfservice` network to a router.

> **Warning**
>
> A bug prevents using both driver options on the same share node. For more information, see LVM Driver section at the Configuration Reference.

Choose one of the following options to configure the share driver. Afterwards, return here and proceed to *Finalize installation*.

- Shared File Systems Option 1: No driver support for share servers management
- Shared File Systems Option 2: Driver support for share servers management

Finalize installation

- Start the Share File Systems service including its dependencies and configure them to start when the system boots:

```
# systemctl enable openstack-manila-share.service
# systemctl start openstack-manila-share.service
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
  - Shared File Systems service overview
  - Install and configure controller node
  - Install and configure a share node
  - Verify operation

- Next steps
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Verify operation

UPDATED: 2016-08-13 23:58

Verify operation of the Shared File Systems service.

> **Note**
>
> Perform these commands on the controller node.

1. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

2. List service components to verify successful launch of each process:

   For deployments using option 1:

```
$ manila service-list
+----+------------------+-------------+------+---------+-------+-----------------
----------+
| Id | Binary           | Host        | Zone | Status  | State | Updated_at
|
```

```
+----+------------------+-------------+------+---------+-------+-----------------
----------+
| 1  | manila-scheduler | controller  | nova | enabled | up    | 2016-03-30T20:17:28.00000
0 |
| 2  | manila-share     | storage@lvm | nova | enabled | up    | 2016-03-30T20:17:29.00000
0 |
+----+------------------+-------------+------+---------+-------+-----------------
----------+
```

For deployments using option 2:

```
$ manila service-list
+----+------------------+-----------------+------+---------+-------+-------------
--------------+
| Id | Binary           | Host            | Zone | Status  | State | Updated_at
|
+----+------------------+-----------------+------+---------+-------+-------------
--------------+
| 1  | manila-scheduler | controller      | nova | enabled | up    | 2016-03-30T20:17:28.0
00000 |
| 2  | manila-share     | storage@generic | nova | enabled | up    | 2016-03-30T20:17:29.0
00000 |
+----+------------------+-----------------+------+---------+-------+-------------
--------------+
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
  - Shared File Systems service overview
  - Install and configure controller node
  - Install and configure a share node
  - Verify operation
  - Next steps
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started

API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack

SEARCH

Next steps

UPDATED: 2016-08-13 23:58

Your OpenStack environment now includes the Shared File Systems service. You can *launch an instance* or add more services to your environment in the following chapters.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
  - Shared File Systems service overview
  - Install and configure controller node
  - Install and configure a share node
  - Verify operation
  - Next steps
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies

Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Object Storage service

UPDATED: 2016-08-13 23:58

- Object Storage service overview
- Install and configure the controller node
- Install and configure the storage nodes
- Create and distribute initial rings
- Finalize installation
- Verify operation
- Next steps

The Object Storage services (swift) work together to provide object storage and retrieval through a *REST API*.

Your environment must at least include the Identity service (keystone) prior to deploying Object Storage.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
  - Object Storage service overview
  - Install and configure the controller node
  - Install and configure the storage nodes
  - Create and distribute initial rings
  - Finalize installation
  - Verify operation
  - Next steps
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects

OpenStack Security

Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Object Storage service overview

UPDATED: 2016-08-13 23:58

The OpenStack Object Storage is a multi-tenant object storage system. It is highly scalable and can manage large amounts of unstructured data at low cost through a RESTful HTTP API.

It includes the following components:

Proxy servers (swift-proxy-server)
: Accepts OpenStack Object Storage API and raw HTTP requests to upload files, modify metadata, and create containers. It also serves file or container listings to web browsers. To improve performance, the proxy server can use an optional cache that is usually deployed with memcache.

Account servers (swift-account-server)
: Manages accounts defined with Object Storage.

Container servers (swift-container-server)
: Manages the mapping of containers or folders, within Object Storage.

Object servers (swift-object-server)
: Manages actual objects, such as files, on the storage nodes.

Various periodic processes
: Performs housekeeping tasks on the large data store. The replication services ensure consistency and availability through the cluster. Other periodic processes include auditors, updaters, and reapers.

WSGI middleware
: Handles authentication and is usually OpenStack Identity.

swift client
: Enables users to submit commands to the REST API through a command-line client authorized as either a admin user, reseller user, or swift user.

swift-init
: Script that initializes the building of the ring file, takes daemon names as parameter and offers commands. Documented in http://docs.openstack.org/developer/swift/admin_guide.html#managing-services.

swift-recon
: A cli tool used to retrieve various metrics and telemetry information about a cluster that has been collected by the swift-recon middleware.

swift-ring-builder
: Storage ring build and rebalance utility. Documented in http://docs.openstack.org/developer/swift/admin_guide.html#managing-the-rings.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
  - Object Storage service overview
  - Install and configure the controller node
  - Install and configure the storage nodes
  - Create and distribute initial rings
  - Finalize installation
  - Verify operation

Next steps
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Install and configure the controller node

UPDATED: 2016-08-13 23:58

Contents

Prerequisites
Install and configure components

This section describes how to install and configure the proxy service that handles requests for the account, container, and object services operating on the storage nodes. For simplicity, this guide installs and configures the proxy service on the controller node. However, you can run the proxy service on any node with network connectivity to the storage nodes. Additionally, you can install and configure the proxy service on multiple nodes to increase performance and redundancy. For more information, see the Deployment Guide.

Prerequisites

The proxy service relies on an authentication and authorization mechanism such as the Identity service. However, unlike other services, it also offers an internal mechanism that allows it to operate without any other OpenStack services. However, for simplicity, this guide references the Identity service in *Identity service*. Before you configure the Object Storage service, you must create service credentials and an API endpoint.

> **Note**
>
> The Object Storage service does not use an SQL database on the controller node. Instead, it uses distributed

> SQLite databases on each storage node.

1. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

2. To create the Identity service credentials, complete these steps:
   - Create the `swift` user:

```
$ openstack user create --domain default --password-prompt swift
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | e0353a670a9e496da891347c589539e9 |
| enabled   | True                             |
| id        | d535e5cbd2b74ac7bfb97db9cced3ed6 |
| name      | swift                            |
+-----------+----------------------------------+
```

   - Add the `admin` role to the `swift` user:

```
$ openstack role add --project service --user swift admin
```

> **Note**
>
> This command provides no output.

   - Create the `swift` service entity:

```
$ openstack service create --name swift \
  --description "OpenStack Object Storage" object-store
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Object Storage         |
| enabled     | True                             |
| id          | 75ef509da2c340499d454ae96a2c5c34 |
| name        | swift                            |
| type        | object-store                     |
```

```
   +-------------+----------------------------------+
```

3. Create the Object Storage service API endpoints:

```
$ openstack endpoint create --region RegionOne \
  object-store public http://controller:8080/v1/AUTH_%\(tenant_id\)s
+--------------+------------------------------------------------+
| Field        | Value                                          |
+--------------+------------------------------------------------+
| enabled      | True                                           |
| id           | 12bfd36f26694c97813f665707114e0d               |
| interface    | public                                         |
| region       | RegionOne                                      |
| region_id    | RegionOne                                      |
| service_id   | 75ef509da2c340499d454ae96a2c5c34               |
| service_name | swift                                          |
| service_type | object-store                                   |
| url          | http://controller:8080/v1/AUTH_%(tenant_id)s |
+--------------+------------------------------------------------+

$ openstack endpoint create --region RegionOne \
  object-store internal http://controller:8080/v1/AUTH_%\(tenant_id\)s
+--------------+------------------------------------------------+
| Field        | Value                                          |
+--------------+------------------------------------------------+
| enabled      | True                                           |
| id           | 7a36bee6733a4b5590d74d3080ee6789               |
| interface    | internal                                       |
| region       | RegionOne                                      |
| region_id    | RegionOne                                      |
| service_id   | 75ef509da2c340499d454ae96a2c5c34               |
| service_name | swift                                          |
| service_type | object-store                                   |
| url          | http://controller:8080/v1/AUTH_%(tenant_id)s |
+--------------+------------------------------------------------+

$ openstack endpoint create --region RegionOne \
  object-store admin http://controller:8080/v1
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | ebb72cd6851d4defabc0b9d71cdca69b |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 75ef509da2c340499d454ae96a2c5c34 |
| service_name | swift                            |
| service_type | object-store                     |
| url          | http://controller:8080/v1        |
+--------------+----------------------------------+
```

Install and configure components

> **Note**
>
> Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (`...`) in the configuration snippets indicates potential default configuration options that you should retain.

1. Install the packages:

```
# yum install openstack-swift-proxy python-swiftclient \
  python-keystoneclient python-keystonemiddleware \
  memcached
```

> **Note**
>
> Complete OpenStack environments already include some of these packages.

2. Obtain the proxy service configuration file from the Object Storage source repository:

```
# curl -o /etc/swift/proxy-server.conf https://git.openstack.org/cgit/openstack/swift/plain
/etc/proxy-server.conf-sample?h=stable/mitaka
```

3. Edit the `/etc/swift/proxy-server.conf` file and complete the following actions:
   - In the `[DEFAULT]` section, configure the bind port, user, and configuration directory:

```
[DEFAULT]
...
bind_port = 8080
user = swift
swift_dir = /etc/swift
```

   - In the `[pipeline:main]` section, remove the `tempurl` and `tempauth` modules and add the `authtoken` and `keystoneauth` modules:

```
[pipeline:main]
pipeline = catch_errors gatekeeper healthcheck proxy-logging cache container_sync bul
k ratelimit authtoken keystoneauth container-quotas account-quotas slo dlo versioned_
```

```
writes proxy-logging proxy-server
```

> **Note**
>
> Do not change the order of the modules.

> **Note**
>
> For more information on other modules that enable additional features, see the Deployment Guide.

- In the `[app:proxy-server]` section, enable automatic account creation:

```
[app:proxy-server]
use = egg:swift#proxy
...
account_autocreate = True
```

- In the `[filter:keystoneauth]` section, configure the operator roles:

```
[filter:keystoneauth]
use = egg:swift#keystoneauth
...
operator_roles = admin,user
```

- In the `[filter:authtoken]` section, configure Identity service access:

```
[filter:authtoken]
paste.filter_factory = keystonemiddleware.auth_token:filter_factory
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = swift
password = SWIFT_PASS
delay_auth_decision = True
```

Replace `SWIFT_PASS` with the password you chose for the `swift` user in the Identity service.

> **Note**
>
> Comment out or remove any other options in the `[filter:authtoken]` section.

- In the `[filter:cache]` section, configure the `memcached` location:

```
[filter:cache]
use = egg:swift#memcache
...
memcache_servers = controller:11211
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard

- Block Storage service
- Shared File Systems service
- Object Storage service
  - Object Storage service overview
  - Install and configure the controller node
  - Install and configure the storage nodes
  - Create and distribute initial rings
  - Finalize installation
  - Verify operation
  - Next steps
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy

OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

Install and configure the storage nodes

UPDATED: 2016-08-13 23:58

Contents

Prerequisites
Install and configure components

This section describes how to install and configure storage nodes that operate the account, container, and object services. For simplicity, this configuration references two storage nodes, each containing two empty local block storage devices. The instructions use `/dev/sdb` and `/dev/sdc`, but you can substitute different values for your particular nodes.

Although Object Storage supports any file system with *extended attributes (xattr)*, testing and benchmarking indicate the best performance and reliability on *XFS*. For more information on horizontally scaling your environment, see the Deployment Guide.

Prerequisites

Before you install and configure the Object Storage service on the storage nodes, you must prepare the storage devices.

> **Note**
>
> Perform these steps on each storage node.

1. Install the supporting utility packages:

```
# yum install xfsprogs rsync
```

2. Format the `/dev/sdb` and `/dev/sdc` devices as XFS:

```
# mkfs.xfs /dev/sdb
# mkfs.xfs /dev/sdc
```

3. Create the mount point directory structure:

```
# mkdir -p /srv/node/sdb
# mkdir -p /srv/node/sdc
```

4. Edit the `/etc/fstab` file and add the following to it:

```
/dev/sdb /srv/node/sdb xfs noatime,nodiratime,nobarrier,logbufs=8 0 2
/dev/sdc /srv/node/sdc xfs noatime,nodiratime,nobarrier,logbufs=8 0 2
```

5. Mount the devices:

```
# mount /srv/node/sdb
# mount /srv/node/sdc
```

6. Create or edit the `/etc/rsyncd.conf` file to contain the following:

```
uid = swift
gid = swift
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
address = MANAGEMENT_INTERFACE_IP_ADDRESS

[account]
max connections = 2
path = /srv/node/
read only = False
lock file = /var/lock/account.lock

[container]
max connections = 2
path = /srv/node/
read only = False
lock file = /var/lock/container.lock
```

```
[object]
max connections = 2
path = /srv/node/
read only = False
lock file = /var/lock/object.lock
```

Replace `MANAGEMENT_INTERFACE_IP_ADDRESS` with the IP address of the management network on the storage node.

> **Note**
>
> The `rsync` service requires no authentication, so consider running it on a private network in production environments.

7. Start the `rsyncd` service and configure it to start when the system boots:

```
# systemctl enable rsyncd.service
# systemctl start rsyncd.service
```

Install and configure components

> **Note**
>
> Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (`...`) in the configuration snippets indicates potential default configuration options that you should retain.

> **Note**
>
> Perform these steps on each storage node.

1. Install the packages:

```
# yum install openstack-swift-account openstack-swift-container \
  openstack-swift-object
```

2. Obtain the accounting, container, and object service configuration files from the Object Storage source repository:

```
# curl -o /etc/swift/account-server.conf https://git.openstack.org/cgit/openstack/swift/pla
in/etc/account-server.conf-sample?h=stable/mitaka
# curl -o /etc/swift/container-server.conf https://git.openstack.org/cgit/openstack/swift/p
lain/etc/container-server.conf-sample?h=stable/mitaka
# curl -o /etc/swift/object-server.conf https://git.openstack.org/cgit/openstack/swift/plai
n/etc/object-server.conf-sample?h=stable/mitaka
```

3. Edit the `/etc/swift/account-server.conf` file and complete the following actions:

   - In the `[DEFAULT]` section, configure the bind IP address, bind port, user, configuration directory, and mount point directory:

     ```
     [DEFAULT]
     ...
     bind_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
     bind_port = 6002
     user = swift
     swift_dir = /etc/swift
     devices = /srv/node
     mount_check = True
     ```

     Replace `MANAGEMENT_INTERFACE_IP_ADDRESS` with the IP address of the management network on the storage node.

   - In the `[pipeline:main]` section, enable the appropriate modules:

     ```
     [pipeline:main]
     pipeline = healthcheck recon account-server
     ```

     > **Note**
     >
     > For more information on other modules that enable additional features, see the Deployment Guide.

   - In the `[filter:recon]` section, configure the recon (meters) cache directory:

     ```
     [filter:recon]
     use = egg:swift#recon
     ...
     recon_cache_path = /var/cache/swift
     ```

4. Edit the `/etc/swift/container-server.conf` file and complete the following actions:

   - In the `[DEFAULT]` section, configure the bind IP address, bind port, user, configuration directory, and mount point directory:

```
[DEFAULT]
...
bind_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
bind_port = 6001
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = True
```

Replace `MANAGEMENT_INTERFACE_IP_ADDRESS` with the IP address of the management network on the storage node.

- In the `[pipeline:main]` section, enable the appropriate modules:

```
[pipeline:main]
pipeline = healthcheck recon container-server
```

> **Note**
>
> For more information on other modules that enable additional features, see the Deployment Guide.

- In the `[filter:recon]` section, configure the recon (meters) cache directory:

```
[filter:recon]
use = egg:swift#recon
...
recon_cache_path = /var/cache/swift
```

5. Edit the `/etc/swift/object-server.conf` file and complete the following actions:

   - In the `[DEFAULT]` section, configure the bind IP address, bind port, user, configuration directory, and mount point directory:

```
[DEFAULT]
...
bind_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
bind_port = 6000
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = True
```

   Replace `MANAGEMENT_INTERFACE_IP_ADDRESS` with the IP address of the management network on the

storage node.

- In the `[pipeline:main]` section, enable the appropriate modules:

```
[pipeline:main]
pipeline = healthcheck recon object-server
```

> **Note**
>
> For more information on other modules that enable additional features, see the Deployment Guide.

- In the `[filter:recon]` section, configure the recon (meters) cache and lock directories:

```
[filter:recon]
use = egg:swift#recon
...
recon_cache_path = /var/cache/swift
recon_lock_path = /var/lock
```

6. Ensure proper ownership of the mount point directory structure:

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

7. Create the `recon` directory and ensure proper ownership of it:

```
# mkdir -p /var/cache/swift
# chown -R root:swift /var/cache/swift
# chmod -R 775 /var/cache/swift
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
  - Object Storage service overview
  - Install and configure the controller node
  - Install and configure the storage nodes
  - Create and distribute initial rings
  - Finalize installation
  - Verify operation
  - Next steps
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects

OpenStack Security

Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack
SEARCH

Create and distribute initial rings

UPDATED: 2016-08-13 23:58

Contents

Create account ring
Create container ring
Create object ring
Distribute ring configuration files

Before starting the Object Storage services, you must create the initial account, container, and object rings. The ring builder creates configuration files that each node uses to determine and deploy the storage architecture. For simplicity, this guide uses one region and two zones with 2^10 (1024) maximum partitions, 3 replicas of each object, and 1 hour minimum time between moving a partition more than once. For Object Storage, a partition indicates a directory on a storage device rather than a conventional partition table. For more information, see the Deployment Guide.

> **Note**
>
> Perform these steps on the controller node.

Create account ring

The account server uses the account ring to maintain lists of containers.

1. Change to the `/etc/swift` directory.
2. Create the base `account.builder` file:

```
# swift-ring-builder account.builder create 10 3 1
```

> **Note**
>
> This command provides no output.

3. Add each storage node to the ring:

```
# swift-ring-builder account.builder \
  add --region 1 --zone 1 --ip STORAGE_NODE_MANAGEMENT_INTERFACE_IP_ADDRESS --port 6002 \
  --device DEVICE_NAME --weight DEVICE_WEIGHT
```

Replace `STORAGE_NODE_MANAGEMENT_INTERFACE_IP_ADDRESS` with the IP address of the management network on the storage node. Replace `DEVICE_NAME` with a storage device name on the same storage node. For example, using the first storage node in *Install and configure the storage nodes* with the `/dev/sdb` storage device and weight of 100:

```
# swift-ring-builder account.builder add \
  --region 1 --zone 1 --ip 10.0.0.51 --port 6002 --device sdb --weight 100
```

Repeat this command for each storage device on each storage node. In the example architecture, use the command in four variations:

```
# swift-ring-builder account.builder add \
  --region 1 --zone 1 --ip 10.0.0.51 --port 6002 --device sdb --weight 100
Device d0r1z1-10.0.0.51:6002R10.0.0.51:6002/sdb_"" with 100.0 weight got id 0
# swift-ring-builder account.builder add \
  --region 1 --zone 1 --ip 10.0.0.51 --port 6002 --device sdc --weight 100
Device d1r1z2-10.0.0.51:6002R10.0.0.51:6002/sdc_"" with 100.0 weight got id 1
# swift-ring-builder account.builder add \
  --region 1 --zone 2 --ip 10.0.0.52 --port 6002 --device sdb --weight 100
Device d2r1z3-10.0.0.52:6002R10.0.0.52:6002/sdb_"" with 100.0 weight got id 2
# swift-ring-builder account.builder add \
  --region 1 --zone 2 --ip 10.0.0.52 --port 6002 --device sdc --weight 100
Device d3r1z4-10.0.0.52:6002R10.0.0.52:6002/sdc_"" with 100.0 weight got id 3
```

4. Verify the ring contents:

```
# swift-ring-builder account.builder
```

```
account.builder, build version 4
1024 partitions, 3.000000 replicas, 1 regions, 2 zones, 4 devices, 100.00 balance, 0.00 dis
persion
The minimum number of hours before a partition can be reassigned is 1
The overload factor is 0.00% (0.000000)
Devices:    id  region  zone      ip address  port  replication ip  replication port      n
ame weight partitions balance meta
             0       1     1       10.0.0.51  6002       10.0.0.51             6002      sd
b  100.00          0 -100.00
             1       1     1       10.0.0.51  6002       10.0.0.51             6002      sd
c  100.00          0 -100.00
             2       1     2       10.0.0.52  6002       10.0.0.52             6002      sd
b  100.00          0 -100.00
             3       1     2       10.0.0.52  6002       10.0.0.52             6002      sd
c  100.00          0 -100.00
```

5. Rebalance the ring:

```
# swift-ring-builder account.builder rebalance
Reassigned 1024 (100.00%) partitions. Balance is now 0.00.  Dispersion is now 0.00
```

Create container ring

The container server uses the container ring to maintain lists of objects. However, it does not track object locations.

1. Change to the `/etc/swift` directory.
2. Create the base `container.builder` file:

```
# swift-ring-builder container.builder create 10 3 1
```

> **Note**
>
> This command provides no output.

3. Add each storage node to the ring:

```
# swift-ring-builder container.builder \
  add --region 1 --zone 1 --ip STORAGE_NODE_MANAGEMENT_INTERFACE_IP_ADDRESS --port 6001 \
  --device DEVICE_NAME --weight DEVICE_WEIGHT
```

Replace `STORAGE_NODE_MANAGEMENT_INTERFACE_IP_ADDRESS` with the IP address of the management network on the storage node. Replace `DEVICE_NAME` with a storage device name on the same storage node.

For example, using the first storage node in *Install and configure the storage nodes* with the `/dev/sdb` storage device and weight of 100:

```
# swift-ring-builder container.builder add \
  --region 1 --zone 1 --ip 10.0.0.51 --port 6001 --device sdb --weight 100
```

Repeat this command for each storage device on each storage node. In the example architecture, use the command in four variations:

```
# swift-ring-builder container.builder add \
  --region 1 --zone 1 --ip 10.0.0.51 --port 6001 --device sdb --weight 100
Device d0r1z1-10.0.0.51:6001R10.0.0.51:6001/sdb_"" with 100.0 weight got id 0
# swift-ring-builder container.builder add \
  --region 1 --zone 1 --ip 10.0.0.51 --port 6001 --device sdc --weight 100
Device d1r1z2-10.0.0.51:6001R10.0.0.51:6001/sdc_"" with 100.0 weight got id 1
# swift-ring-builder container.builder add \
  --region 1 --zone 2 --ip 10.0.0.52 --port 6001 --device sdb --weight 100
Device d2r1z3-10.0.0.52:6001R10.0.0.52:6001/sdb_"" with 100.0 weight got id 2
# swift-ring-builder container.builder add \
  --region 1 --zone 2 --ip 10.0.0.52 --port 6001 --device sdc --weight 100
Device d3r1z4-10.0.0.52:6001R10.0.0.52:6001/sdc_"" with 100.0 weight got id 3
```

4. Verify the ring contents:

```
# swift-ring-builder container.builder
container.builder, build version 4
1024 partitions, 3.000000 replicas, 1 regions, 2 zones, 4 devices, 100.00 balance, 0.00 dis
persion
The minimum number of hours before a partition can be reassigned is 1
The overload factor is 0.00% (0.000000)
Devices:    id  region  zone      ip address  port  replication ip  replication port      n
ame weight partitions balance meta
             0       1     1       10.0.0.51  6001       10.0.0.51             6001      sd
b  100.00          0 -100.00
             1       1     1       10.0.0.51  6001       10.0.0.51             6001      sd
c  100.00          0 -100.00
             2       1     2       10.0.0.52  6001       10.0.0.52             6001      sd
b  100.00          0 -100.00
             3       1     2       10.0.0.52  6001       10.0.0.52             6001      sd
c  100.00          0 -100.00
```

5. Rebalance the ring:

```
# swift-ring-builder container.builder rebalance
Reassigned 1024 (100.00%) partitions. Balance is now 0.00.  Dispersion is now 0.00
```

Create object ring

The object server uses the object ring to maintain lists of object locations on local devices.

1. Change to the `/etc/swift` directory.
2. Create the base `object.builder` file:

```
# swift-ring-builder object.builder create 10 3 1
```

> **Note**
>
> This command provides no output.

3. Add each storage node to the ring:

```
# swift-ring-builder object.builder \
  add --region 1 --zone 1 --ip STORAGE_NODE_MANAGEMENT_INTERFACE_IP_ADDRESS --port 6000 \
  --device DEVICE_NAME --weight DEVICE_WEIGHT
```

Replace `STORAGE_NODE_MANAGEMENT_INTERFACE_IP_ADDRESS` with the IP address of the management network on the storage node. Replace `DEVICE_NAME` with a storage device name on the same storage node. For example, using the first storage node in *Install and configure the storage nodes* with the `/dev/sdb` storage device and weight of 100:

```
# swift-ring-builder object.builder add \
  --region 1 --zone 1 --ip 10.0.0.51 --port 6000 --device sdb --weight 100
```

Repeat this command for each storage device on each storage node. In the example architecture, use the command in four variations:

```
# swift-ring-builder object.builder add \
  --region 1 --zone 1 --ip 10.0.0.51 --port 6000 --device sdb --weight 100
Device d0r1z1-10.0.0.51:6000R10.0.0.51:6000/sdb_"" with 100.0 weight got id 0
# swift-ring-builder object.builder add \
  --region 1 --zone 1 --ip 10.0.0.51 --port 6000 --device sdc --weight 100
Device d1r1z2-10.0.0.51:6000R10.0.0.51:6000/sdc_"" with 100.0 weight got id 1
# swift-ring-builder object.builder add \
  --region 1 --zone 2 --ip 10.0.0.52 --port 6000 --device sdb --weight 100
Device d2r1z3-10.0.0.52:6000R10.0.0.52:6000/sdb_"" with 100.0 weight got id 2
# swift-ring-builder object.builder add \
  --region 1 --zone 2 --ip 10.0.0.52 --port 6000 --device sdc --weight 100
Device d3r1z4-10.0.0.52:6000R10.0.0.52:6000/sdc_"" with 100.0 weight got id 3
```

4. Verify the ring contents:

```
# swift-ring-builder object.builder
object.builder, build version 4
1024 partitions, 3.000000 replicas, 1 regions, 2 zones, 4 devices, 100.00 balance, 0.00 dis
persion
The minimum number of hours before a partition can be reassigned is 1
The overload factor is 0.00% (0.000000)
Devices:    id  region  zone      ip address  port  replication ip  replication port      n
ame weight partitions balance meta
             0       1     1       10.0.0.51  6000       10.0.0.51              6000      sd
b  100.00          0 -100.00
             1       1     1       10.0.0.51  6000       10.0.0.51              6000      sd
c  100.00          0 -100.00
             2       1     2       10.0.0.52  6000       10.0.0.52              6000      sd
b  100.00          0 -100.00
             3       1     2       10.0.0.52  6000       10.0.0.52              6000      sd
c  100.00          0 -100.00
```

5. Rebalance the ring:

```
# swift-ring-builder object.builder rebalance
Reassigned 1024 (100.00%) partitions. Balance is now 0.00.  Dispersion is now 0.00
```

Distribute ring configuration files

- Copy the `account.ring.gz`, `container.ring.gz`, and `object.ring.gz` files to the `/etc/swift` directory on each storage node and any additional nodes running the proxy service.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
  - Object Storage service overview
  - Install and configure the controller node
  - Install and configure the storage nodes
  - Create and distribute initial rings
  - Finalize installation
  - Verify operation
  - Next steps
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Finalize installation

UPDATED: 2016-08-13 23:58

> **Note**
>
> Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (`...`) in the configuration snippets indicates potential default configuration options that you should retain.

1. Obtain the `/etc/swift/swift.conf` file from the Object Storage source repository:

```
# curl -o /etc/swift/swift.conf \
  https://git.openstack.org/cgit/openstack/swift/plain/etc/swift.conf-sample?h=stable/mitak
a
```

2. Edit the `/etc/swift/swift.conf` file and complete the following actions:
   - In the `[swift-hash]` section, configure the hash path prefix and suffix for your environment.

```
[swift-hash]
...
swift_hash_path_suffix = HASH_PATH_SUFFIX
swift_hash_path_prefix = HASH_PATH_PREFIX
```

```
```

Replace HASH_PATH_PREFIX and HASH_PATH_SUFFIX with unique values.

> **Warning**
>
> Keep these values secret and do not change or lose them.

- In the `[storage-policy:0]` section, configure the default storage policy:

```
[storage-policy:0]
...
name = Policy-0
default = yes
```

3. Copy the `swift.conf` file to the `/etc/swift` directory on each storage node and any additional nodes running the proxy service.
4. On all nodes, ensure proper ownership of the configuration directory:

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

5. On the controller node and any other nodes running the proxy service, start the Object Storage proxy service including its dependencies and configure them to start when the system boots:

```
# systemctl enable openstack-swift-proxy.service memcached.service
# systemctl start openstack-swift-proxy.service memcached.service
```

6. On the storage nodes, start the Object Storage services and configure them to start when the system boots:

```
# systemctl enable openstack-swift-account.service openstack-swift-account-auditor.service
\
  openstack-swift-account-reaper.service openstack-swift-account-replicator.service
# systemctl start openstack-swift-account.service openstack-swift-account-auditor.service
\
  openstack-swift-account-reaper.service openstack-swift-account-replicator.service
# systemctl enable openstack-swift-container.service \
  openstack-swift-container-auditor.service openstack-swift-container-replicator.service \
  openstack-swift-container-updater.service
# systemctl start openstack-swift-container.service \
  openstack-swift-container-auditor.service openstack-swift-container-replicator.service \
  openstack-swift-container-updater.service
# systemctl enable openstack-swift-object.service openstack-swift-object-auditor.service \
  openstack-swift-object-replicator.service openstack-swift-object-updater.service
# systemctl start openstack-swift-object.service openstack-swift-object-auditor.service \
```

```
openstack-swift-object-replicator.service openstack-swift-object-updater.service
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
  - Object Storage service overview
  - Install and configure the controller node
  - Install and configure the storage nodes
  - Create and distribute initial rings
  - Finalize installation
  - Verify operation
  - Next steps
- Orchestration service
- Telemetry service
- Database service
- Launch an instance

- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack

SEARCH

Verify operation

UPDATED: 2016-08-13 23:58

Verify operation of the Object Storage service.

> **Note**
>
> Perform these steps on the controller node.

> **Warning**
>
> If one or more of these steps do not work, check the `/var/log/audit/audit.log` file for SELinux messages indicating denial of actions for the `swift` processes. If present, change the security context of the `/srv/node` directory to the lowest security level (s0) for the `swift_data_t` type, `object_r` role and the `system_u` user:
>
> ```
> # chcon -R system_u:object_r:swift_data_t:s0 /srv/node
> ```

1. Source the `demo` credentials:

```
$ . demo-openrc
```

2. Show the service status:

```
$ swift stat
                        Account: AUTH_ed0b60bf607743088218b0a533d5943f
                     Containers: 0
                        Objects: 0
                          Bytes: 0
Containers in policy "policy-0": 0
   Objects in policy "policy-0": 0
     Bytes in policy "policy-0": 0
    X-Account-Project-Domain-Id: default
                   X-Timestamp: 1444143887.71539
                    X-Trans-Id: tx1396aeaf17254e94beb34-0056143bde
                  Content-Type: text/plain; charset=utf-8
                 Accept-Ranges: bytes
```

3. Create `container1` container:

```
$ openstack container create container1
+---------------------------------------+------------+------------------------------
--------+
| account                               | container  | x-trans-id                       |
+---------------------------------------+------------+------------------------------
--------+
| AUTH_ed0b60bf607743088218b0a533d5943f | container1 | tx8c4034dc306c44dd8cd68-0056f00a4a |
+---------------------------------------+------------+------------------------------
--------+
```

4. Upload a test file to the `container1` container:

```
$ openstack object create container1 FILE
+--------+------------+----------------------------------+
| object | container  | etag                             |
+--------+------------+----------------------------------+
| FILE   | container1 | ee1eca47dc88f4879d8a229cc70a07c6 |
+--------+------------+----------------------------------+
```

   Replace `FILE` with the name of a local file to upload to the `container1` container.

5. List files in the `container1` container:

```
$ openstack object list container1
+------+
| Name |
+------+
```

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

6. Download a test file from the `container1` container:

```
$ openstack object save container1 FILE
```

Replace `FILE` with the name of the file uploaded to the `container1` container.

> **Note**
>
> This command provides no output.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service

Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
  - Object Storage service overview
  - Install and configure the controller node
  - Install and configure the storage nodes
  - Create and distribute initial rings
  - Finalize installation
  - Verify operation
  - Next steps
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines

Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack

SEARCH

Next steps

UPDATED: 2016-08-13 23:58

Your OpenStack environment now includes Object Storage. You can *Launch an instance* or add more services to your environment.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
  - Object Storage service overview
  - Install and configure the controller node
  - Install and configure the storage nodes
  - Create and distribute initial rings
  - Finalize installation
  - Verify operation
  - Next steps
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack

SEARCH

Orchestration service

UPDATED: 2016-08-13 23:58

- Orchestration service overview
- Install and configure
  - Prerequisites
  - Install and configure components
  - Finalize installation
- Verify operation
- Next steps

The Orchestration service (heat) uses a Heat Orchestration Template (HOT) to create and manage cloud resources.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
  - Orchestration service overview
  - Install and configure
  - Verify operation
  - Next steps
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

Orchestration service overview

UPDATED: 2016-08-13 23:58

The Orchestration service provides a template-based orchestration for describing a cloud application by running OpenStack API calls to generate running cloud applications. The software integrates other core components of OpenStack into a one-file template system. The templates allow you to create most OpenStack resource types such as instances, floating IPs, volumes, security groups, and users. It also provides advanced functionality such as instance high availability, instance auto-scaling, and nested stacks. This enables OpenStack core projects to receive a larger user base.

The service enables deployers to integrate with the Orchestration service directly or through custom plug-ins.

The Orchestration service consists of the following components:

`heat` command-line client
: A CLI that communicates with the `heat-api` to run *AWS* CloudFormation APIs. End developers can directly use the Orchestration REST API.

`heat-api` component
: An OpenStack-native REST API that processes API requests by sending them to the `heat-engine` over *Remote Procedure Call (RPC)*.

`heat-api-cfn` component
: An AWS Query API that is compatible with AWS CloudFormation. It processes API requests by sending them to the `heat-engine` over RPC.

`heat-engine`
: Orchestrates the launching of templates and provides events back to the API consumer.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
  - Orchestration service overview
  - Install and configure
  - Verify operation
  - Next steps
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

Install and configure

UPDATED: 2016-08-13 23:58

Contents

- Prerequisites
- Install and configure components
- Finalize installation

This section describes how to install and configure the Orchestration service, code-named heat, on the controller node.

Prerequisites

Before you install and configure Orchestration, you must create a database, service credentials, and API endpoints. Orchestration also requires additional information in the Identity service.

1. To create the database, complete these steps:
   - Use the database access client to connect to the database server as the `root` user:

     ```
     $ mysql -u root -p
     ```

   - Create the `heat` database:

     ```
     CREATE DATABASE heat;
     ```

```
```

- Grant proper access to the `heat` database:

```
GRANT ALL PRIVILEGES ON heat.* TO 'heat'@'localhost' \
  IDENTIFIED BY 'HEAT_DBPASS';
GRANT ALL PRIVILEGES ON heat.* TO 'heat'@'%' \
  IDENTIFIED BY 'HEAT_DBPASS';
```

Replace `HEAT_DBPASS` with a suitable password.

- Exit the database access client.

2. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

3. To create the service credentials, complete these steps:

- Create the `heat` user:

```
$ openstack user create --domain default --password-prompt heat
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | e0353a670a9e496da891347c589539e9 |
| enabled   | True                             |
| id        | ca2e175b851943349be29a328cc5e360 |
| name      | heat                             |
+-----------+----------------------------------+
```

- Add the `admin` role to the `heat` user:

```
$ openstack role add --project service --user heat admin
```

> **Note**
>
> This command provides no output.

- Create the `heat` and `heat-cfn` service entities:

```
$ openstack service create --name heat \
```

```
  --description "Orchestration" orchestration
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Orchestration                    |
| enabled     | True                             |
| id          | 727841c6f5df4773baa4e8a5ae7d72eb |
| name        | heat                             |
| type        | orchestration                    |
+-------------+----------------------------------+

$ openstack service create --name heat-cfn \
  --description "Orchestration"  cloudformation
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Orchestration                    |
| enabled     | True                             |
| id          | c42cede91a4e47c3b10c8aedc8d890c6 |
| name        | heat-cfn                         |
| type        | cloudformation                   |
+-------------+----------------------------------+
```

4. Create the Orchestration service API endpoints:

```
$ openstack endpoint create --region RegionOne \
  orchestration public http://controller:8004/v1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 3f4dab34624e4be7b000265f25049609        |
| interface    | public                                  |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | 727841c6f5df4773baa4e8a5ae7d72eb        |
| service_name | heat                                    |
| service_type | orchestration                           |
| url          | http://controller:8004/v1/%(tenant_id)s |
+--------------+-----------------------------------------+

$ openstack endpoint create --region RegionOne \
  orchestration internal http://controller:8004/v1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 9489f78e958e45cc85570fec7e836d98        |
| interface    | internal                                |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | 727841c6f5df4773baa4e8a5ae7d72eb        |
```

```
| service_name | heat                                      |
| service_type | orchestration                             |
| url          | http://controller:8004/v1/%(tenant_id)s |
+--------------+-------------------------------------------+

$ openstack endpoint create --region RegionOne \
  orchestration admin http://controller:8004/v1/%\(tenant_id\)s
+--------------+-------------------------------------------+
| Field        | Value                                     |
+--------------+-------------------------------------------+
| enabled      | True                                      |
| id           | 76091559514b40c6b7b38dde790efe99          |
| interface    | admin                                     |
| region       | RegionOne                                 |
| region_id    | RegionOne                                 |
| service_id   | 727841c6f5df4773baa4e8a5ae7d72eb          |
| service_name | heat                                      |
| service_type | orchestration                             |
| url          | http://controller:8004/v1/%(tenant_id)s |
+--------------+-------------------------------------------+
```

```
$ openstack endpoint create --region RegionOne \
  cloudformation public http://controller:8000/v1
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | b3ea082e019c4024842bf0a80555052c |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | c42cede91a4e47c3b10c8aedc8d890c6 |
| service_name | heat-cfn                         |
| service_type | cloudformation                   |
| url          | http://controller:8000/v1        |
+--------------+----------------------------------+

$ openstack endpoint create --region RegionOne \
  cloudformation internal http://controller:8000/v1
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 169df4368cdc435b8b115a9cb084044e |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | c42cede91a4e47c3b10c8aedc8d890c6 |
| service_name | heat-cfn                         |
| service_type | cloudformation                   |
| url          | http://controller:8000/v1        |
+--------------+----------------------------------+
```

```
$ openstack endpoint create --region RegionOne \
  cloudformation admin http://controller:8000/v1
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 3d3edcd61eb343c1bbd629aa041ff88b |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | c42cede91a4e47c3b10c8aedc8d890c6 |
| service_name | heat-cfn                         |
| service_type | cloudformation                   |
| url          | http://controller:8000/v1        |
+--------------+----------------------------------+
```

5. Orchestration requires additional information in the Identity service to manage stacks. To add this information, complete these steps:

   - Create the `heat` domain that contains projects and users for stacks:

     ```
     $ openstack domain create --description "Stack projects and users" heat
     +-------------+----------------------------------+
     | Field       | Value                            |
     +-------------+----------------------------------+
     | description | Stack projects and users         |
     | enabled     | True                             |
     | id          | 0f4d1bd326f2454dacc72157ba328a47 |
     | name        | heat                             |
     +-------------+----------------------------------+
     ```

   - Create the `heat_domain_admin` user to manage projects and users in the `heat` domain:

     ```
     $ openstack user create --domain heat --password-prompt heat_domain_admin
     User Password:
     Repeat User Password:
     +-----------+----------------------------------+
     | Field     | Value                            |
     +-----------+----------------------------------+
     | domain_id | 0f4d1bd326f2454dacc72157ba328a47 |
     | enabled   | True                             |
     | id        | b7bd1abfbcf64478b47a0f13cd4d970a |
     | name      | heat_domain_admin                |
     +-----------+----------------------------------+
     ```

   - Add the `admin` role to the `heat_domain_admin` user in the `heat` domain to enable administrative stack management privileges by the `heat_domain_admin` user:

     ```
     ```

```
$ openstack role add --domain heat --user-domain heat --user heat_domain_admin admin
```

> **Note**
>
> This command provides no output.

- Create the `heat_stack_owner` role:

```
$ openstack role create heat_stack_owner
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | 15e34f0c4fed4e68b3246275883c8630 |
| name      | heat_stack_owner                 |
+-----------+----------------------------------+
```

- Add the `heat_stack_owner` role to the `demo` project and user to enable stack management by the `demo` user:

```
$ openstack role add --project demo --user demo heat_stack_owner
```

> **Note**
>
> This command provides no output.

> **Note**
>
> You must add the `heat_stack_owner` role to each user that manages stacks.

- Create the `heat_stack_user` role:

```
$ openstack role create heat_stack_user
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
```

```
| id        | 88849d41a55d4d1d91e4f11bffd8fc5c |
| name      | heat_stack_user                  |
+-----------+----------------------------------+
```

> **Note**
>
> The Orchestration service automatically assigns the `heat_stack_user` role to users that it creates during stack deployment. By default, this role restricts *API* operations. To avoid conflicts, do not add this role to users with the `heat_stack_owner` role.

Install and configure components

> **Note**
>
> Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (`...`) in the configuration snippets indicates potential default configuration options that you should retain.

1. Install the packages:

```
# yum install openstack-heat-api openstack-heat-api-cfn \
  openstack-heat-engine
```

2. Edit the `/etc/heat/heat.conf` file and complete the following actions:

   - In the `[database]` section, configure database access:

```
[database]
...
connection = mysql+pymysql://heat:HEAT_DBPASS@controller/heat
```

     Replace `HEAT_DBPASS` with the password you chose for the Orchestration database.

   - In the `[DEFAULT]` and `[oslo_messaging_rabbit]` sections, configure `RabbitMQ` message queue access:

```
[DEFAULT]
...
rpc_backend = rabbit
```

```
[oslo_messaging_rabbit]
...
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS
```

Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.

- In the `[keystone_authtoken]`, `[trustee]`, `[clients_keystone]`, and `[ec2authtoken]` sections, configure Identity service access:

```
[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = heat
password = HEAT_PASS

[trustee]
...
auth_plugin = password
auth_url = http://controller:35357
username = heat
password = HEAT_PASS
user_domain_name = default

[clients_keystone]
...
auth_uri = http://controller:35357

[ec2authtoken]
...
auth_uri = http://controller:5000
```

Replace `HEAT_PASS` with the password you chose for the `heat` user in the Identity service.

- In the `[DEFAULT]` section, configure the metadata and wait condition URLs:

```
[DEFAULT]
...
heat_metadata_server_url = http://controller:8000
heat_waitcondition_server_url = http://controller:8000/v1/waitcondition
```

- In the `[DEFAULT]` section, configure the stack domain and administrative credentials:

```
```

```
[DEFAULT]
...
stack_domain_admin = heat_domain_admin
stack_domain_admin_password = HEAT_DOMAIN_PASS
stack_user_domain_name = heat
```

Replace `HEAT_DOMAIN_PASS` with the password you chose for the `heat_domain_admin` user in the Identity service.

3. Populate the Orchestration database:

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

> **Note**
>
> Ignore any deprecation messages in this output.

Finalize installation

- Start the Orchestration services and configure them to start when the system boots:

```
# systemctl enable openstack-heat-api.service \
  openstack-heat-api-cfn.service openstack-heat-engine.service
# systemctl start openstack-heat-api.service \
  openstack-heat-api-cfn.service openstack-heat-engine.service
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
  - Orchestration service overview
  - Install and configure
  - Verify operation
  - Next steps
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs

Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack

SEARCH

Verify operation

UPDATED: 2016-08-13 23:58

Verify operation of the Orchestration service.

> **Note**
>
> Perform these commands on the controller node.

1. Source the `admin` tenant credentials:

```
$ . admin-openrc
```

2. List service components to verify successful launch and registration of each process:

```
$ openstack orchestration service list
+------------+-------------+--------------------------------------+------------+--
------+----------------------------+---------+
| hostname   | binary      | engine_id                            | host       | topic  | u
pdated_at                  | status |
+------------+-------------+--------------------------------------+------------+--
------+----------------------------+---------+
```

```
| controller | heat-engine | 3e85d1ab-a543-41aa-aa97-378c381fb958 | controller | engine |
2015-10-13T14:16:06.000000 | up     |
| controller | heat-engine | 45dbdcf6-5660-4d5f-973a-c4fc819da678 | controller | engine |
2015-10-13T14:16:06.000000 | up     |
| controller | heat-engine | 51162b63-ecb8-4c6c-98c6-993af899c4f7 | controller | engine |
2015-10-13T14:16:06.000000 | up     |
| controller | heat-engine | 8d7edc6d-77a6-460d-bd2a-984d76954646 | controller | engine |
2015-10-13T14:16:06.000000 | up     |
+------------+-------------+--------------------------------------+------------+--
------+-----------------------------+--------+
```

> **Note**
>
> This output should indicate four `heat-engine` components on the controller node.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service

- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
  - Orchestration service overview
  - Install and configure
  - Verify operation
  - Next steps
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Next steps

UPDATED: 2016-08-13 23:58

Your OpenStack environment now includes Orchestration. You can *Launch an instance* or add more services to your environment in the following chapters.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
  - Orchestration service overview
  - Install and configure
  - Verify operation
  - Next steps
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

Telemetry service

UPDATED: 2016-08-13 23:58

- Telemetry service overview
  - Telemetry Data Collection service
  - Telemetry Alarming service
- Install and configure
  - Prerequisites
  - Install and configure components
  - Finalize installation
- Enable Image service meters
  - Configure the Image service to use Telemetry
  - Finalize installation
- Enable Compute service meters
  - Install and configure components
  - Configure Compute to use Telemetry
  - Finalize installation
- Enable Block Storage meters
  - Configure Cinder to use Telemetry
  - Finalize installation
- Enable Object Storage meters
  - Prerequisites
  - Install components
  - Configure Object Storage to use Telemetry
  - Finalize installation
- Alarming service
  - Prerequisites

- Install and configure components
- Finalize installation
- Verify operation
- Next steps

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
  - Telemetry service overview
  - Install and configure
  - Enable Image service meters
  - Enable Compute service meters
  - Enable Block Storage meters
  - Enable Object Storage meters
  - Alarming service

    - Verify operation
    - Next steps
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

Telemetry service overview

UPDATED: 2016-08-13 23:58

Contents

Telemetry Data Collection service
Telemetry Alarming service

Telemetry Data Collection service

The Telemetry Data Collection services provide the following functions:

- Efficiently polls metering data related to OpenStack services.
- Collects event and metering data by monitoring notifications sent from services.
- Publishes collected data to various targets including data stores and message queues.

The Telemetry service consists of the following components:

A compute agent (`ceilometer-agent-compute`)
: Runs on each compute node and polls for resource utilization statistics. There may be other types of agents in the future, but for now our focus is creating the compute agent.

A central agent (`ceilometer-agent-central`)
: Runs on a central management server to poll for resource utilization statistics for resources not tied to instances or compute nodes. Multiple agents can be started to scale service horizontally.

A notification agent (`ceilometer-agent-notification`)
: Runs on a central management server(s) and consumes messages from the message queue(s) to build event and metering data.

A collector (`ceilometer-collector`)
: Runs on central management server(s) and dispatches collected telemetry data to a data store or external consumer without modification.

An API server (`ceilometer-api`)
: Runs on one or more central management servers to provide data access from the data store.

Telemetry Alarming service

The Telemetry Alarming services trigger alarms when the collected metering or event data break the defined rules.

The Telemetry Alarming service consists of the following components:

An API server (`aodh-api`)
: Runs on one or more central management servers to provide access to the alarm information stored in the data store.

An alarm evaluator (`aodh-evaluator`)
: Runs on one or more central management servers to determine when alarms fire due to the associated statistic trend crossing a threshold over a sliding time window.

A notification listener (`aodh-listener`)
: Runs on a central management server and determines when to fire alarms. The alarms are generated based on defined rules against events, which are captured by the Telemetry Data Collection service's notification agents.

An alarm notifier (`aodh-notifier`)
: Runs on one or more central management servers to allow alarms to be set based on the threshold evaluation for a collection of samples.

These services communicate by using the OpenStack messaging bus. Only the collector and API server have access to the data store.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
  - Telemetry service overview
  - Install and configure
  - Enable Image service meters
  - Enable Compute service meters
  - Enable Block Storage meters
  - Enable Object Storage meters
  - Alarming service
  - Verify operation
  - Next steps
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies

Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Install and configure

UPDATED: 2016-08-13 23:58

Contents

Prerequisites
Install and configure components
Finalize installation

This section describes how to install and configure the Telemetry service, code-named ceilometer, on the controller node. The Telemetry service collects measurements from most OpenStack services and optionally triggers alarms.

Prerequisites

Before you install and configure the Telemetry service, you must create a database, service credentials, and API endpoints. However, unlike other services, the Telemetry service uses a NoSQL database. See *NoSQL database* to install and configure MongoDB before proceeding further.

1. Create the `ceilometer` database:

```
# mongo --host controller --eval '
  db = db.getSiblingDB("ceilometer");
  db.createUser({user: "ceilometer",
  pwd: "CEILOMETER_DBPASS",
  roles: [ "readWrite", "dbAdmin" ]})'

MongoDB shell version: 2.6.x
```

```
connecting to: controller:27017/test
Successfully added user: { "user" : "ceilometer", "roles" : [ "readWrite", "dbAdmin" ] }
```

Replace `CEILOMETER_DBPASS` with a suitable password.

2. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

3. To create the service credentials, complete these steps:
   - Create the `ceilometer` user:

```
$ openstack user create --domain default --password-prompt ceilometer
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | e0353a670a9e496da891347c589539e9 |
| enabled   | True                             |
| id        | c859c96f57bd4989a8ea1a0b1d8ff7cd |
| name      | ceilometer                       |
+-----------+----------------------------------+
```

   - Add the `admin` role to the `ceilometer` user.

```
$ openstack role add --project service --user ceilometer admin
```

> **Note**
>
> This command provides no output.

   - Create the `ceilometer` service entity:

```
$ openstack service create --name ceilometer \
  --description "Telemetry" metering
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Telemetry                        |
| enabled     | True                             |
| id          | 5fb7fd1bb2954fddb378d4031c28c0e4 |
| name        | ceilometer                       |
```

```
| type        | metering                         |
+-------------+----------------------------------+
```

4. Create the Telemetry service API endpoints:

```
$ openstack endpoint create --region RegionOne \
  metering public http://controller:8777
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | b808b67b848d443e9eaaa5e5d796970c |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 5fb7fd1bb2954fddb378d4031c28c0e4 |
| service_name | ceilometer                       |
| service_type | metering                         |
| url          | http://controller:8777           |
+--------------+----------------------------------+

$ openstack endpoint create --region RegionOne \
  metering internal http://controller:8777
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | c7009b1c2ee54b71b771fa3d0ae4f948 |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 5fb7fd1bb2954fddb378d4031c28c0e4 |
| service_name | ceilometer                       |
| service_type | metering                         |
| url          | http://controller:8777           |
+--------------+----------------------------------+

$ openstack endpoint create --region RegionOne \
  metering admin http://controller:8777
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | b2c00566d0604551b5fe1540c699db3d |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 5fb7fd1bb2954fddb378d4031c28c0e4 |
| service_name | ceilometer                       |
| service_type | metering                         |
| url          | http://controller:8777           |
+--------------+----------------------------------+
```

Install and configure components

1. Install the packages:

```
# yum install openstack-ceilometer-api \
  openstack-ceilometer-collector openstack-ceilometer-notification \
  openstack-ceilometer-central python-ceilometerclient
```

2. Edit the `/etc/ceilometer/ceilometer.conf` file and complete the following actions:

   - In the `[database]` section, configure database access:

```
[database]
...
connection = mongodb://ceilometer:CEILOMETER_DBPASS@controller:27017/ceilometer
```

     Replace `CEILOMETER_DBPASS` with the password you chose for the Telemetry service database. You must escape special characters such as ':', '/', '+', and '@' in the connection string in accordance with RFC2396.

   - In the `[DEFAULT]` and `[oslo_messaging_rabbit]` sections, configure `RabbitMQ` message queue access:

```
[DEFAULT]
...
rpc_backend = rabbit

[oslo_messaging_rabbit]
...
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS
```

     Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.

   - In the `[DEFAULT]` and `[keystone_authtoken]` sections, configure Identity service access:

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
```

```
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = ceilometer
password = CEILOMETER_PASS
```

Replace `CEILOMETER_PASS` with the password you chose for the `ceilometer` user in the Identity service.

- In the `[service_credentials]` section, configure service credentials:

```
[service_credentials]
...
auth_type = password
auth_url = http://controller:5000/v3
project_domain_name = default
user_domain_name = default
project_name = service
username = ceilometer
password = CEILOMETER_PASS
interface = internalURL
region_name = RegionOne
```

Replace `CEILOMETER_PASS` with the password you chose for the `ceilometer` user in the Identity service.

Finalize installation

- Start the Telemetry services and configure them to start when the system boots:

```
# systemctl enable openstack-ceilometer-api.service \
  openstack-ceilometer-notification.service \
  openstack-ceilometer-central.service \
  openstack-ceilometer-collector.service
# systemctl start openstack-ceilometer-api.service \
  openstack-ceilometer-notification.service \
  openstack-ceilometer-central.service \
  openstack-ceilometer-collector.service
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
  - Telemetry service overview
  - Install and configure
  - Enable Image service meters
  - Enable Compute service meters
  - Enable Block Storage meters
  - Enable Object Storage meters
  - Alarming service
  - Verify operation
  - Next steps
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Enable Image service meters

UPDATED: 2016-08-13 23:58

Contents

Configure the Image service to use Telemetry
Finalize installation

Telemetry uses notifications to collect Image service meters. Perform these steps on the controller node.

Configure the Image service to use Telemetry

- Edit the `/etc/glance/glance-api.conf` and `/etc/glance/glance-registry.conf` files and complete the following actions:
  - In the `[DEFAULT]`, `[oslo_messaging_notifications]`, and `[oslo_messaging_rabbit]` sections, configure notifications and RabbitMQ message broker access:

```
[DEFAULT]
...
rpc_backend = rabbit

[oslo_messaging_notifications]
...
driver = messagingv2

[oslo_messaging_rabbit]
```

```
...
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS
```

Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.

Finalize installation

- Restart the Image service:

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

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service

- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
  - Telemetry service overview
  - Install and configure
  - Enable Image service meters
  - Enable Compute service meters
  - Enable Block Storage meters
  - Enable Object Storage meters
  - Alarming service
  - Verify operation
  - Next steps
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy

Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Enable Compute service meters

UPDATED: 2016-08-13 23:58

Contents

Install and configure components
Configure Compute to use Telemetry
Finalize installation

Telemetry uses a combination of notifications and an agent to collect Compute meters. Perform these steps on each compute node.

Install and configure components

1. Install the packages:

```
# yum install openstack-ceilometer-compute python-ceilometerclient python-pecan
```

2. Edit the `/etc/ceilometer/ceilometer.conf` file and complete the following actions:
   - In the `[DEFAULT]` and `[oslo_messaging_rabbit]` sections, configure `RabbitMQ` message queue access:

```
[DEFAULT]
...
rpc_backend = rabbit
```

```
[oslo_messaging_rabbit]
...
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS
```

Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.

- In the `[DEFAULT]` and `[keystone_authtoken]` sections, configure Identity service access:

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = ceilometer
password = CEILOMETER_PASS
```

Replace `CEILOMETER_PASS` with the password you chose for the Telemetry service database.

- In the `[service_credentials]` section, configure service credentials:

```
[service_credentials]
...
auth_url = http://controller:5000/v2.0
username = ceilometer
project_name = service
password = CEILOMETER_PASS
interface = internalURL
region_name = RegionOne
```

Replace `CEILOMETER_PASS` with the password you chose for the `ceilometer` user in the Identity service.

Configure Compute to use Telemetry

- Edit the `/etc/nova/nova.conf` file and configure notifications in the `[DEFAULT]` section:

```
[DEFAULT]
```

```
...
instance_usage_audit = True
instance_usage_audit_period = hour
notify_on_state_change = vm_and_task_state
notification_driver = messagingv2
```

Finalize installation

1. Start the agent and configure it to start when the system boots:

```
# systemctl enable openstack-ceilometer-compute.service
# systemctl start openstack-ceilometer-compute.service
```

2. Restart the Compute service:

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

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service

- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
  - Telemetry service overview
  - Install and configure
  - Enable Image service meters
  - Enable Compute service meters
  - Enable Block Storage meters
  - Enable Object Storage meters
  - Alarming service
  - Verify operation
  - Next steps
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation

Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Enable Block Storage meters

UPDATED: 2016-08-13 23:58

Contents

Configure Cinder to use Telemetry
Finalize installation

Telemetry uses notifications to collect Block Storage service meters. Perform these steps on the controller and Block Storage nodes.

> **Note**
>
> Your environment must include the Block Storage service.

Configure Cinder to use Telemetry

Edit the `/etc/cinder/cinder.conf` file and complete the following actions:

- In the `[oslo_messaging_notifications]` section, configure notifications:

```
[oslo_messaging_notifications]
...
```

```
driver = messagingv2
```

Finalize installation

1. Restart the Block Storage services on the controller node:

   ```
   # systemctl restart openstack-cinder-api.service openstack-cinder-scheduler.service
   ```

2. Restart the Block Storage services on the storage nodes:

   ```
   # systemctl restart openstack-cinder-volume.service
   ```

3. Use the `cinder-volume-usage-audit` command on Block Storage nodes to retrieve meters on demand. For more information, see the Administrator Guide.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service

Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
  - Telemetry service overview
  - Install and configure
  - Enable Image service meters
  - Enable Compute service meters
  - Enable Block Storage meters
  - Enable Object Storage meters
  - Alarming service
  - Verify operation
  - Next steps
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Enable Object Storage meters

UPDATED: 2016-08-13 23:58

Contents

- Prerequisites
- Install components
- Configure Object Storage to use Telemetry
- Finalize installation

Telemetry uses a combination of polling and notifications to collect Object Storage meters.

> **Note**
>
> Your environment must include the Object Storage service.

Prerequisites

The Telemetry service requires access to the Object Storage service using the `ResellerAdmin` role. Perform these steps on the controller node.

1. Source the `admin` credentials to gain access to admin-only CLI commands.

```
$ . admin-openrc
```

2. Create the `ResellerAdmin` role:

```
$ openstack role create ResellerAdmin
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | 462fa46c13fd4798a95a3bfbe27b5e54 |
| name      | ResellerAdmin                    |
+-----------+----------------------------------+
```

3. Add the `ResellerAdmin` role to the `ceilometer` user:

```
$ openstack role add --project service --user ceilometer ResellerAdmin
```

> **Note**
>
> This command provides no output.

Install components

- Install the packages:

```
# yum install python-ceilometermiddleware
```

Configure Object Storage to use Telemetry

Perform these steps on the controller and any other nodes that run the Object Storage proxy service.

- Edit the `/etc/swift/proxy-server.conf` file and complete the following actions:
  - In the `[filter:keystoneauth]` section, add the `ResellerAdmin` role:

```
[filter:keystoneauth]
...
operator_roles = admin, user, ResellerAdmin
```

  - In the `[pipeline:main]` section, add `ceilometer`:

```
[pipeline:main]
pipeline = ceilometer catch_errors gatekeeper healthcheck proxy-logging cache contain
er_sync bulk ratelimit authtoken keystoneauth container-quotas account-quotas slo dlo
versioned_writes proxy-logging proxy-server
```

- In the `[filter:ceilometer]` section, configure notifications:

```
[filter:ceilometer]
paste.filter_factory = ceilometermiddleware.swift:filter_factory
...
control_exchange = swift
url = rabbit://openstack:RABBIT_PASS@controller:5672/
driver = messagingv2
topic = notifications
log_level = WARN
```

Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.

Finalize installation

- Restart the Object Storage proxy service:

```
# systemctl restart openstack-swift-proxy.service
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
  - Telemetry service overview
  - Install and configure
  - Enable Image service meters
  - Enable Compute service meters
  - Enable Block Storage meters
  - Enable Object Storage meters
  - Alarming service
  - Verify operation
  - Next steps
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

Alarming service

UPDATED: 2016-08-13 23:58

Contents

Prerequisites
Install and configure components
Finalize installation

This section describes how to install and configure the Telemetry Alarming service, code-named aodh.

Prerequisites

Before you install and configure the Alarming service, you must create a database, service credentials, and API endpoints.

1. To create the database, complete these steps:
   - Use the database access client to connect to the database server as the `root` user:

     ```
     $ mysql -u root -p
     ```

   - Create the `aodh` database:

     ```
     CREATE DATABASE aodh;
     ```

```
```

- Grant proper access to the `aodh` database:

```
GRANT ALL PRIVILEGES ON aodh.* TO 'aodh'@'localhost' \
  IDENTIFIED BY 'AODH_DBPASS';
GRANT ALL PRIVILEGES ON aodh.* TO 'aodh'@'%' \
  IDENTIFIED BY 'AODH_DBPASS';
```

Replace `AODH_DBPASS` with a suitable password.

- Exit the database access client.

2. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

3. To create the service credentials, complete these steps:

- Create the `aodh` user:

```
$ openstack user create --domain default \
  --password-prompt aodh
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | e0353a670a9e496da891347c589539e9 |
| enabled   | True                             |
| id        | b7657c9ea07a4556aef5d34cf70713a3 |
| name      | aodh                             |
+-----------+----------------------------------+
```

- Add the `admin` role to the `aodh` user:

```
$ openstack role add --project service --user aodh admin
```

> **Note**
>
> This command provides no output.

- Create the `aodh` service entity:

```
```

```
$ openstack service create --name aodh \
  --description "Telemetry" alarming
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Telemetry                        |
| enabled     | True                             |
| id          | 3405453b14da441ebb258edfeba96d83 |
| name        | aodh                             |
| type        | alarming                         |
+-------------+----------------------------------+
```

4. Create the Alarming service API endpoints:

```
$ openstack endpoint create --region RegionOne \
  alarming public http://controller:8042
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 340be3625e9b4239a6415d034e98aace |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| service_name | aodh                             |
| service_type | alarming                         |
| url          | http://controller:8042           |
+--------------+----------------------------------+

$ openstack endpoint create --region RegionOne \
  alarming internal http://controller:8042
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 340be3625e9b4239a6415d034e98aace |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| service_name | aodh                             |
| service_type | alarming                         |
| url          | http://controller:8042           |
+--------------+----------------------------------+

$ openstack endpoint create --region RegionOne \
  alarming admin http://controller:8042
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
```

```
| id           | 340be3625e9b4239a6415d034e98aace |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| service_name | aodh                             |
| service_type | alarming                         |
| url          | http://controller:8042           |
+--------------+----------------------------------+
```

Install and configure components

> **Note**
>
> Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (...) in the configuration snippets indicates potential default configuration options that you should retain.

1. Install the packages:

```
# yum install openstack-aodh-api \
  openstack-aodh-evaluator openstack-aodh-notifier \
  openstack-aodh-listener openstack-aodh-expirer \
  python-ceilometerclient
```

2. Edit the `/etc/aodh/aodh.conf` file and complete the following actions:

   - In the `[database]` section, configure database access:

```
[database]
...
connection = mysql+pymysql://aodh:AODH_DBPASS@controller/aodh
```

     Replace `AODH_DBPASS` with the password you chose for the Telemetry Alarming module database. You must escape special characters such as ':', '/', '+', and '@' in the connection string in accordance with RFC2396.

   - In the `[DEFAULT]` and `[oslo_messaging_rabbit]` sections, configure `RabbitMQ` message queue access:

```
[DEFAULT]
...
rpc_backend = rabbit
```

```
[oslo_messaging_rabbit]
...
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS
```

Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.

- In the `[DEFAULT]` and `[keystone_authtoken]` sections, configure Identity service access:

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = aodh
password = AODH_PASS
```

Replace `AODH_PASS` with the password you chose for the `aodh` user in the Identity service.

- In the `[service_credentials]` section, configure service credentials:

```
[service_credentials]
...
auth_type = password
auth_url = http://controller:5000/v3
project_domain_name = default
user_domain_name = default
project_name = service
username = aodh
password = AODH_PASS
interface = internalURL
region_name = RegionOne
```

Replace `AODH_PASS` with the password you chose for the `aodh` user in the Identity service.

Finalize installation

- Start the Alarming services and configure them to start when the system boots:

```
```

```
# systemctl enable openstack-aodh-api.service \
  openstack-aodh-evaluator.service \
  openstack-aodh-notifier.service \
  openstack-aodh-listener.service
# systemctl start openstack-aodh-api.service \
  openstack-aodh-evaluator.service \
  openstack-aodh-notifier.service \
  openstack-aodh-listener.service
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
  - Telemetry service overview
  - Install and configure

    - Enable Image service meters
    - Enable Compute service meters
    - Enable Block Storage meters
    - Enable Object Storage meters
    - Alarming service
    - Verify operation
    - Next steps
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

Verify operation

UPDATED: 2016-08-13 23:58

Verify operation of the Telemetry service. These steps only include the Image service meters to reduce clutter. Environments with ceilometer integration for additional services contain more meters.

> **Note**
>
> Perform these steps on the controller node.

1. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

2. List available meters:

```
$ ceilometer meter-list
+--------------+-------+-------+----------------------------------------+---------+-
-----------+
| Name         | Type  | Unit  | Resource ID                            | User ID | Project I
D |
+--------------+-------+-------+----------------------------------------+---------+-
-----------+
```

```
| image        | gauge | image | acafc7c0-40aa-4026-9673-b879898e1fc2 | None    | cf12a15.
.. |
| image.size   | gauge | B     | acafc7c0-40aa-4026-9673-b879898e1fc2 | None    | cf12a15.
.. |
+--------------+-------+-------+--------------------------------------+---------+-
-----------+
```

3. Download the CirrOS image from the Image service:

```
$ IMAGE_ID=$(glance image-list | grep 'cirros' | awk '{ print $2 }')
$ glance image-download $IMAGE_ID > /tmp/cirros.img
```

4. List available meters again to validate detection of the image download:

```
$ ceilometer meter-list
+----------------+-------+-------+--------------------------------------+---------
+------------+
| Name           | Type  | Unit  | Resource ID                          | User ID | Project
ID |
+----------------+-------+-------+--------------------------------------+---------
+------------+
| image          | gauge | image | acafc7c0-40aa-4026-9673-b879898e1fc2 | None    | cf12a1
5... |
| image.download | delta | B     | acafc7c0-40aa-4026-9673-b879898e1fc2 | None    | cf12a1
5... |
| image.serve    | delta | B     | acafc7c0-40aa-4026-9673-b879898e1fc2 | None    | cf12a1
5... |
| image.size     | gauge | B     | acafc7c0-40aa-4026-9673-b879898e1fc2 | None    | cf12a1
5... |
+----------------+-------+-------+--------------------------------------+---------
+------------+
```

5. Retrieve usage statistics from the `image.download` meter:

```
$ ceilometer statistics -m image.download -p 60
+--------+---------------------+---------------------+------------+------------+--
----------+------------+-------+----------+----------------------------+----------
------------------+
| Period | Period Start        | Period End          | Max        | Min        | Avg
| Sum        | Count | Duration | Duration Start             | Duration End               |
+--------+---------------------+---------------------+------------+------------+--
----------+------------+-------+----------+----------------------------+----------
------------------+
| 60     | 2015-04-21T12:21:45 | 2015-04-21T12:22:45 | 13200896.0 | 13200896.0 | 13200896.
0 | 13200896.0 | 1     | 0.0      | 2015-04-21T12:22:12.983000 | 2015-04-21T12:22:12.98300
0 |
+--------+---------------------+---------------------+------------+------------+--
----------+------------+-------+----------+----------------------------+----------
------------------+
```

6. Remove the previously downloaded image file `/tmp/cirros.img`:

```
$ rm /tmp/cirros.img
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
  - Telemetry service overview
  - Install and configure
  - Enable Image service meters
  - Enable Compute service meters
  - Enable Block Storage meters

- Enable Object Storage meters
- Alarming service
- Verify operation
- Next steps

- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Next steps

UPDATED: 2016-08-13 23:58

Your OpenStack environment now includes Telemetry. You can *Launch an instance* or add more services to your environment.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
  - Telemetry service overview
  - Install and configure
  - Enable Image service meters
  - Enable Compute service meters
  - Enable Block Storage meters
  - Enable Object Storage meters
  - Alarming service
  - Verify operation
  - Next steps
- Database service
- Launch an instance
- Community support
- Glossary

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack

SEARCH

Database service

UPDATED: 2016-08-13 23:58

- Database service overview
- Install and configure
  - Prerequisites
  - Install and configure components
  - Finalize installation
- Verify operation
- Next steps

The Database service (trove) provides cloud provisioning functionality for database engines.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
  - Database service overview
  - Install and configure
  - Verify operation
  - Next steps
- Launch an instance
- Community support
- Glossary

Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Database service overview

UPDATED: 2016-08-13 23:58

The Database service provides scalable and reliable cloud provisioning functionality for both relational and non-relational database engines. Users can quickly and easily use database features without the burden of handling complex administrative tasks. Cloud users and database administrators can provision and manage multiple database instances as needed.

The Database service provides resource isolation at high performance levels, and automates complex administrative tasks such as deployment, configuration, patching, backups, restores, and monitoring.

Process flow example

This example is a high-level process flow for using Database services:

1. The OpenStack Administrator configures the basic infrastructure using the following steps:
    a. Install the Database service.
    b. Create an image for each type of database. For example, one for MySQL and one for MongoDB.
    c. Use the `trove-manage` command to import images and offer them to tenants.
2. The OpenStack end user deploys the Database service using the following steps:
    a. Create a Database service instance using the `trove create` command.
    b. Use the `trove list` command to get the ID of the instance, followed by the `trove show` command to get the IP address of it.
    c. Access the Database service instance using typical database access commands. For example, with MySQL:

```
$ mysql -u myuser -p -h TROVE_IP_ADDRESS mydb
```

Components

The Database service includes the following components:

`python-troveclient` command-line client
: A CLI that communicates with the `trove-api` component.

`trove-api` component
: Provides an OpenStack-native RESTful API that supports JSON to provision and manage Trove instances.

`trove-conductor` service
: Runs on the host, and receives messages from guest instances that want to update information on the host.

`trove-taskmanager` service
: Instruments the complex system flows that support provisioning instances, managing the lifecycle of instances, and performing operations on instances.

`trove-guestagent` service
: Runs within the guest instance. Manages and performs operations on the database itself.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service

- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
  - Database service overview
  - Install and configure
  - Verify operation
  - Next steps
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

Install and configure

UPDATED: 2016-08-13 23:58

Contents

Prerequisites
Install and configure components
Finalize installation

This section describes how to install and configure the Database service, code-named trove, on the controller node.

This section assumes that you already have a working OpenStack environment with at least the following components installed: Compute, Image Service, Identity.

- If you want to do backup and restore, you also need Object Storage.
- If you want to provision datastores on block-storage volumes, you also need Block Storage.

Prerequisites

Before you install and configure the Database service, you must create a database, service credentials, and API endpoints.

1. To create the database, complete these steps:
   - Use the database access client to connect to the database server as the `root` user:

```
$ mysql -u root -p
```

```
```

- Create the `trove` database:

```
CREATE DATABASE trove;
```

- Grant proper access to the `trove` database:

```
GRANT ALL PRIVILEGES ON trove.* TO 'trove'@'localhost' \
  IDENTIFIED BY 'TROVE_DBPASS';
GRANT ALL PRIVILEGES ON trove.* TO 'trove'@'%' \
  IDENTIFIED BY 'TROVE_DBPASS';
```

  Replace `TROVE_DBPASS` with a suitable password.

- Exit the database access client.

2. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ source admin-openrc.sh
```

3. To create the service credentials, complete these steps:

   - Create the `trove` user:

```
$ openstack user create --domain default --password-prompt trove
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | default                          |
| enabled   | True                             |
| id        | ca2e175b851943349be29a328cc5e360 |
| name      | trove                            |
+-----------+----------------------------------+
```

   - Add the `admin` role to the `trove` user:

```
$ openstack role add --project service --user trove admin
```

> **Note**
>
> This command provides no output.

- Create the `trove` service entity:

```
$ openstack service create --name trove \
  --description "Database" database
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Database                         |
| enabled     | True                             |
| id          | 727841c6f5df4773baa4e8a5ae7d72eb |
| name        | trove                            |
| type        | database                         |
+-------------+----------------------------------+
```

4. Create the Database service API endpoints:

```
$ openstack endpoint create --region RegionOne \
  database public http://controller:8779/v1.0/%\(tenant_id\)s
+--------------+------------------------------------------+
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | 3f4dab34624e4be7b000265f25049609         |
| interface    | public                                   |
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | 727841c6f5df4773baa4e8a5ae7d72eb         |
| service_name | trove                                    |
| service_type | database                                 |
| url          | http://controller:8779/v1.0/%\(tenant_id\)s |
+--------------+------------------------------------------+

$ openstack endpoint create --region RegionOne \
  database internal http://controller:8779/v1.0/%\(tenant_id\)s
+--------------+------------------------------------------+
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | 9489f78e958e45cc85570fec7e836d98         |
| interface    | internal                                 |
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | 727841c6f5df4773baa4e8a5ae7d72eb         |
| service_name | trove                                    |
| service_type | database                                 |
| url          | http://controller:8779/v1.0/%\(tenant_id\)s |
+--------------+------------------------------------------+

$ openstack endpoint create --region RegionOne \
  database admin http://controller:8779/v1.0/%\(tenant_id\)s
```

```
+--------------+-------------------------------------------+
| Field        | Value                                     |
+--------------+-------------------------------------------+
| enabled      | True                                      |
| id           | 76091559514b40c6b7b38dde790efe99          |
| interface    | admin                                     |
| region       | RegionOne                                 |
| region_id    | RegionOne                                 |
| service_id   | 727841c6f5df4773baa4e8a5ae7d72eb          |
| service_name | trove                                     |
| service_type | database                                  |
| url          | http://controller:8779/v1.0/%\(tenant_id\)s |
+--------------+-------------------------------------------+
```

Install and configure components

> **Note**
>
> Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (`...`) in the configuration snippets indicates potential default configuration options that you should retain.

1. Install the packages:

```
# yum install openstack-trove python-troveclient
```

2. In the `/etc/trove` directory, edit the `trove.conf`, `trove-taskmanager.conf` and `trove-conductor.conf` files and complete the following steps:

   - Provide appropriate values for the following settings:

```
[DEFAULT]
log_dir = /var/log/trove
trove_auth_url = http://controller:5000/v2.0
nova_compute_url = http://controller:8774/v2
cinder_url = http://controller:8776/v1
swift_url = http://controller:8080/v1/AUTH_
notifier_queue_hostname = controller
...
[database]
connection = mysql://trove:TROVE_DBPASS@controller/trove
```

   - Configure the Database service to use the `RabbitMQ` message broker by setting the following options in each file:

```
[DEFAULT]
...
rpc_backend = rabbit

[oslo_messaging_rabbit]
...
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS
```

3. Verify that the `api-paste.ini` file is present in `/etc/trove`.

   If the file is not present, you can get it from this location.

4. Edit the `trove.conf` file so it includes appropriate values for the settings shown below:

```
[DEFAULT]
auth_strategy = keystone
...
# Config option for showing the IP address that nova doles out
add_addresses = True
network_label_regex = ^NETWORK_LABEL$
...
api_paste_config = /etc/trove/api-paste.ini
...
[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = trove
password = TROVE_PASS
```

5. Edit the `trove-taskmanager.conf` file so it includes the required settings to connect to the OpenStack Compute service as shown below:

```
[DEFAULT]
...
# Configuration options for talking to nova via the novaclient.
# These options are for an admin user in your keystone config.
# It proxy's the token received from the user to send to nova
# via this admin users creds,
# basically acting like the client via that proxy token.
nova_proxy_admin_user = admin
nova_proxy_admin_pass = ADMIN_PASS
nova_proxy_admin_tenant_name = service
taskmanager_manager = trove.taskmanager.manager.Manager
```

6. Edit the `/etc/trove/trove-guestagent.conf` file so that future trove guests can connect to your OpenStack environment:

```
rabbit_host = controller
rabbit_password = RABBIT_PASS
nova_proxy_admin_user = admin
nova_proxy_admin_pass = ADMIN_PASS
nova_proxy_admin_tenant_name = service
trove_auth_url = http://controller:35357/v2.0
```

7. Populate the trove database you created earlier in this procedure:

```
# su -s /bin/sh -c "trove-manage db_sync" trove
  ...
  2016-04-06 22:00:17.771 10706 INFO trove.db.sqlalchemy.migration [-]
  Upgrading mysql://trove:dbaasdb@controller/trove to version latest
```

> **Note**
>
> Ignore any deprecation messages in this output.

Finalize installation

1. Start the Database services and configure them to start when the system boots:

```
# systemctl enable openstack-trove-api.service \
  openstack-trove-taskmanager.service \
  openstack-trove-conductor.service

# systemctl start openstack-trove-api.service \
  openstack-trove-taskmanager.service \
  openstack-trove-conductor.service
```

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
  - Database service overview
  - Install and configure
  - Verify operation
  - Next steps
- Launch an instance
- Community support
- Glossary

OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Verify operation

UPDATED: 2016-08-13 23:58

Verify operation of the Database service.

> **Note**
>
> Perform these commands on the node where you installed trove.

1. Source the `admin` tenant credentials:

```
$ source admin-openrc.sh
```

2. Run the `trove list` command. You should see output similar to this:

```
$ trove list
+----+------+-----------+-------------------+--------+-----------+------+
| id | name | datastore | datastore_version | status | flavor_id | size |
+----+------+-----------+-------------------+--------+-----------+------+
+----+------+-----------+-------------------+--------+-----------+------+
```

3. Add a datastore to trove:

- Create a trove image.
  Create an image for the type of database you want to use, for example, MySQL, MongoDB, Cassandra.

  This image must have the trove guest agent installed.

- Upload the image to glance. Example:

```
$ glance image-create --name "mysqlTest" --disk-format qcow2 \
  --container-format bare \
  --file mysql-5.6.qcow2
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 51a8e6e5ff10b08f2c2ec2953f0a8086     |
| container_format | bare                                 |
| created_at       | 2016-04-08T15:15:41Z                 |
| disk_format      | qcow2                                |
| id               | 5caa76dd-f44b-4d01-a3b4-a111e27896be |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | mysqlTest                            |
| owner            | 0c0bd5e850c24893b48c4cc01e2a7986     |
| protected        | False                                |
| size             | 533790720                            |
| status           | active                               |
| tags             | []                                   |
| updated_at       | 2016-04-08T15:15:51Z                 |
| virtual_size     | None                                 |
| visibility       | private                              |
+------------------+--------------------------------------+
```

- Create a datastore. You need to create a separate datastore for each type of database you want to use, for example, MySQL, MongoDB, Cassandra. This example shows you how to create a datastore for a MySQL database:

```
# su -s /bin/sh -c "trove-manage \
  --config-file /etc/trove/trove.conf \
  datastore_update mysql ''" trove
...
Datastore 'mysql' updated.
```

4. Update the datastore to use the new image.

   This example shows you how to update a MySQL 5.6 datastore:

```
# su -s /bin/sh -c "trove-manage --config-file /etc/trove/trove.conf \
  datastore_version_update \
  mysql mysql-5.6 mysql glance_image_ID '' 1" trove
...
Datastore version 'mysql-5.6' updated.
```

5. Create a database instance.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
  - Database service overview
  - Install and configure
  - Verify operation
  - Next steps
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Next steps

UPDATED: 2016-08-13 23:58

Your OpenStack environment now includes Database services.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
  - Database service overview
  - Install and configure
  - Verify operation
  - Next steps
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Launch an instance

UPDATED: 2016-08-13 23:58

Contents

- Create virtual networks
- Create m1.nano flavor
- Generate a key pair
- Add security group rules
- Launch an instance
- Block Storage
- Orchestration
- Shared File Systems

This section creates the necessary virtual networks to support launching instances. Networking option 1 includes one provider (external) network with one instance that uses it. Networking option 2 includes one provider network with one instance that uses it and one self-service (private) network with one instance that uses it. The instructions in this section use command-line interface (CLI) tools on the controller node. For more information on the CLI tools, see the OpenStack End User Guide. To use the dashboard, see the OpenStack End User Guide.

Create virtual networks

Create virtual networks for the networking option that you chose in *Networking service*. If you chose option 1, create only the provider network. If you chose option 2, create the provider and self-service networks.

- Provider network
- Self-service network

After creating the appropriate networks for your environment, you can continue preparing the environment to launch an instance.

Create m1.nano flavor

The smallest default flavor consumes 512 MB memory per instance. For environments with compute nodes containing less than 4 GB memory, we recommend creating the `m1.nano` flavor that only requires 64 MB per instance. Only use this flavor with the CirrOS image for testing purposes.

```
$ openstack flavor create --id 0 --vcpus 1 --ram 64 --disk 1 m1.nano
+----------------------------+---------+
| Field                      | Value   |
+----------------------------+---------+
| OS-FLV-DISABLED:disabled   | False   |
| OS-FLV-EXT-DATA:ephemeral  | 0       |
| disk                       | 1       |
| id                         | 0       |
| name                       | m1.nano |
| os-flavor-access:is_public | True    |
| ram                        | 64      |
| rxtx_factor                | 1.0     |
| swap                       |         |
| vcpus                      | 1       |
+----------------------------+---------+
```

Generate a key pair

Most cloud images support *public key authentication* rather than conventional password authentication. Before launching an instance, you must add a public key to the Compute service.

1. Source the `demo` tenant credentials:

   ```
   $ . demo-openrc
   ```

2. Generate and add a key pair:

   ```
   $ ssh-keygen -q -N ""
   $ openstack keypair create --public-key ~/.ssh/id_rsa.pub mykey
   +-------------+-------------------------------------------------+
   | Field       | Value                                           |
   +-------------+-------------------------------------------------+
   | fingerprint | ee:3d:2e:97:d4:e2:6a:54:6d:0d:ce:43:39:2c:ba:4d |
   | name        | mykey                                           |
   | user_id     | 58126687cbcc4888bfa9ab73a2256f27                |
   ```

```
+-------------+-------------------------------------------------+
```

> **Note**
>
> Alternatively, you can skip the `ssh-keygen` command and use an existing public key.

3. Verify addition of the key pair:

```
$ openstack keypair list
+-------+-------------------------------------------------+
| Name  | Fingerprint                                     |
+-------+-------------------------------------------------+
| mykey | ee:3d:2e:97:d4:e2:6a:54:6d:0d:ce:43:39:2c:ba:4d |
+-------+-------------------------------------------------+
```

Add security group rules

By default, the `default` security group applies to all instances and includes firewall rules that deny remote access to instances. For Linux images such as CirrOS, we recommend allowing at least ICMP (ping) and secure shell (SSH).

- Add rules to the `default` security group:
  - Permit *ICMP* (ping):

```
$ openstack security group rule create --proto icmp default
+-----------------------+--------------------------------------+
| Field                 | Value                                |
+-----------------------+--------------------------------------+
| id                    | a1876c06-7f30-4a67-a324-b6b5d1309546 |
| ip_protocol           | icmp                                 |
| ip_range              | 0.0.0.0/0                            |
| parent_group_id       | b0d53786-5ebb-4729-9e4a-4b675016a958 |
| port_range            |                                      |
| remote_security_group |                                      |
+-----------------------+--------------------------------------+
```

  - Permit secure shell (SSH) access:

```
$ openstack security group rule create --proto tcp --dst-port 22 default
+-----------------------+--------------------------------------+
| Field                 | Value                                |
+-----------------------+--------------------------------------+
| id                    | 3d95e59c-e98d-45f1-af04-c750af914f14 |
| ip_protocol           | tcp                                  |
```

```
| ip_range              | 0.0.0.0/0                                |
| parent_group_id       | b0d53786-5ebb-4729-9e4a-4b675016a958 |
| port_range            | 22:22                                    |
| remote_security_group |                                          |
+-----------------------+------------------------------------------+
```

Launch an instance

If you chose networking option 1, you can only launch an instance on the provider network. If you chose networking option 2, you can launch an instance on the provider network and the self-service network.

- Launch an instance on the provider network
- Launch an instance on the self-service network

Block Storage

If your environment includes the Block Storage service, you can create a volume and attach it to an instance.

- Block Storage

Orchestration

If your environment includes the Orchestration service, you can create a stack that launches an instance.

- Orchestration

Shared File Systems

If your environment includes the Shared File Systems service, you can create a share and mount it in an instance:

- Shared File Systems

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
  - Create virtual networks
  - Create m1.nano flavor
  - Generate a key pair
  - Add security group rules
  - Launch an instance
  - Block Storage
  - Orchestration
  - Shared File Systems
- Community support
- Glossary

OpenStack

Projects

OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Community support

UPDATED: 2016-08-13 23:58

Contents

- Documentation
- ask.openstack.org
- OpenStack mailing lists
- The OpenStack wiki
- The Launchpad Bugs area
- The OpenStack IRC channel
- Documentation feedback
- OpenStack distribution packages

The following resources are available to help you run and use OpenStack. The OpenStack community constantly improves and adds to the main features of OpenStack, but if you have any questions, do not hesitate to ask. Use the following resources to get OpenStack support, and troubleshoot your installations.

Documentation

For the available OpenStack documentation, see docs.openstack.org.

To provide feedback on documentation, join and use the openstack-docs@lists.openstack.org mailing list at OpenStack Documentation Mailing List, or report a bug.

The following books explain how to install an OpenStack cloud and its associated components:

- Installation Guide for openSUSE Leap 42.1 and SUSE Linux Enterprise Server 12 SP1

Installation Guide for Red Hat Enterprise Linux 7 and CentOS 7
- Installation Guide for Ubuntu 14.04 (LTS)

The following books explain how to configure and run an OpenStack cloud:

- Architecture Design Guide
- Administrator Guide
- Configuration Reference
- Operations Guide
- Networking Guide
- High Availability Guide
- Security Guide
- Virtual Machine Image Guide

The following books explain how to use the OpenStack dashboard and command-line clients:

- API Guide
- End User Guide
- Command-Line Interface Reference

The following documentation provides reference and guidance information for the OpenStack APIs:

- API Complete Reference (HTML)
- API Complete Reference (PDF)

The following guide provides how to contribute to OpenStack documentation:

- Documentation Contributor Guide

ask.openstack.org

During the set up or testing of OpenStack, you might have questions about how a specific task is completed or be in a situation where a feature does not work correctly. Use the ask.openstack.org site to ask questions and get answers. When you visit the https://ask.openstack.org site, scan the recently asked questions to see whether your question has already been answered. If not, ask a new question. Be sure to give a clear, concise summary in the title and provide as much detail as possible in the description. Paste in your command output or stack traces, links to screen shots, and any other information which might be useful.

OpenStack mailing lists

A great way to get answers and insights is to post your question or problematic scenario to the OpenStack mailing list. You can learn from and help others who might have similar issues. To subscribe or view the archives, go to http://lists.openstack.org/cgi-bin/mailman/listinfo/openstack. If you are interested in the other mailing lists for specific projects or development, refer to Mailing Lists.

The OpenStack wiki

The OpenStack wiki contains a broad range of topics but some of the information can be difficult to find or is a few pages deep. Fortunately, the wiki search feature enables you to search by title or content. If you search for specific information, such as about networking or OpenStack Compute, you can find a large amount of relevant material. More is being added all the time, so be sure to check back often. You can find the search box in the upper-right corner of any

OpenStack wiki page.

The Launchpad Bugs area

The OpenStack community values your set up and testing efforts and wants your feedback. To log a bug, you must sign up for a Launchpad account at https://launchpad.net/+login. You can view existing bugs and report bugs in the Launchpad Bugs area. Use the search feature to determine whether the bug has already been reported or already been fixed. If it still seems like your bug is unreported, fill out a bug report.

Some tips:

- Give a clear, concise summary.
- Provide as much detail as possible in the description. Paste in your command output or stack traces, links to screen shots, and any other information which might be useful.
- Be sure to include the software and package versions that you are using, especially if you are using a development branch, such as, `"Kilo release" vs git commit bc79c3ecc55929bac585d04a03475b72e06a3208`.
- Any deployment-specific information is helpful, such as whether you are using Ubuntu 14.04 or are performing a multi-node installation.

The following Launchpad Bugs areas are available:

- Bugs: OpenStack Block Storage (cinder)
- Bugs: OpenStack Compute (nova)
- Bugs: OpenStack Dashboard (horizon)
- Bugs: OpenStack Identity (keystone)
- Bugs: OpenStack Image service (glance)
- Bugs: OpenStack Networking (neutron)
- Bugs: OpenStack Object Storage (swift)
- Bugs: Application catalog (murano)
- Bugs: Bare metal service (ironic)
- Bugs: Clustering service (senlin)
- Bugs: Containers service (magnum)
- Bugs: Data processing service (sahara)
- Bugs: Database service (trove)
- Bugs: Deployment service (fuel)
- Bugs: DNS service (designate)
- Bugs: Key Manager Service (barbican)
- Bugs: Monitoring (monasca)
- Bugs: Orchestration (heat)
- Bugs: Rating (cloudkitty)
- Bugs: Shared file systems (manila)
- Bugs: Telemetry (ceilometer)
- Bugs: Telemetry v3 (gnocchi)
- Bugs: Workflow service (mistral)
- Bugs: Messaging service (zaqar)
- Bugs: OpenStack API Documentation (developer.openstack.org)
- Bugs: OpenStack Documentation (docs.openstack.org)

The OpenStack IRC channel

The OpenStack community lives in the #openstack IRC channel on the Freenode network. You can hang out, ask questions, or get immediate feedback for urgent and pressing issues. To install an IRC client or use a browser-based client, go to https://webchat.freenode.net/. You can also use Colloquy (Mac OS X, http://colloquy.info/), mIRC (Windows, http://www.mirc.com/), or XChat (Linux). When you are in the IRC channel and want to share code or command output, the generally accepted method is to use a Paste Bin. The OpenStack project has one at http://paste.openstack.org. Just paste your longer amounts of text or logs in the web form and you get a URL that you can paste into the channel. The OpenStack IRC channel is `#openstack` on `irc.freenode.net`. You can find a list of all OpenStack IRC channels at https://wiki.openstack.org/wiki/IRC.

Documentation feedback

To provide feedback on documentation, join and use the openstack-docs@lists.openstack.org mailing list at OpenStack Documentation Mailing List, or report a bug.

OpenStack distribution packages

The following Linux distributions provide community-supported packages for OpenStack:

- Debian: https://wiki.debian.org/OpenStack
- CentOS, Fedora, and Red Hat Enterprise Linux: https://www.rdoproject.org/
- openSUSE and SUSE Linux Enterprise Server: https://en.opensuse.org/Portal:OpenStack
- Ubuntu: https://wiki.ubuntu.com/ServerTeam/CloudArchive

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
  - Documentation
  - ask.openstack.org
  - OpenStack mailing lists
  - The OpenStack wiki
  - The Launchpad Bugs area
  - The OpenStack IRC channel
  - Documentation feedback
  - OpenStack distribution packages
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

Glossary

UPDATED: 2016-08-13 23:58

Contents

0-9
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V

W
X
Z

This glossary offers a list of terms and definitions to define a vocabulary for OpenStack-related concepts.

To add to OpenStack glossary, clone the openstack/openstack-manuals repository and update the source file `doc/common/glossary.rst` through the OpenStack contribution process.

0-9

6to4
: A mechanism that allows IPv6 packets to be transmitted over an IPv4 network, providing a strategy for migrating to IPv6.

A

absolute limit
: Impassable limits for guest VMs. Settings include total RAM size, maximum number of vCPUs, and maximum disk size.

access control list
: A list of permissions attached to an object. An ACL specifies which users or system processes have access to objects. It also defines which operations can be performed on specified objects. Each entry in a typical ACL specifies a subject and an operation. For instance, the ACL entry `(Alice, delete)` for a file gives Alice permission to delete the file.

access key
: Alternative term for an Amazon EC2 access key. See EC2 access key.

account
: The Object Storage context of an account. Do not confuse with a user account from an authentication service, such as Active Directory, /etc/passwd, OpenLDAP, OpenStack Identity, and so on.

account auditor
: Checks for missing replicas and incorrect or corrupted objects in a specified Object Storage account by running queries against the back-end SQLite database.

account database
: A SQLite database that contains Object Storage accounts and related metadata and that the accounts server accesses.

account reaper
: An Object Storage worker that scans for and deletes account databases and that the account server has marked for deletion.

account server
: Lists containers in Object Storage and stores container information in the account database.

account service
: An Object Storage component that provides account services such as list, create, modify, and audit. Do not confuse with OpenStack Identity service, OpenLDAP, or similar user-account services.

accounting
: The Compute service provides accounting information through the event notification and system usage data facilities.

ACL
: See access control list.

active/active configuration

In a high-availability setup with an active/active configuration, several systems share the load together and if one fails, the load is distributed to the remaining systems.

Active Directory
: Authentication and identity service by Microsoft, based on LDAP. Supported in OpenStack.

active/passive configuration
: In a high-availability setup with an active/passive configuration, systems are set up to bring additional resources online to replace those that have failed.

address pool
: A group of fixed and/or floating IP addresses that are assigned to a project and can be used by or assigned to the VM instances in a project.

admin API
: A subset of API calls that are accessible to authorized administrators and are generally not accessible to end users or the public Internet. They can exist as a separate service (keystone) or can be a subset of another API (nova).

administrator
: The person responsible for installing, configuring, and managing an OpenStack cloud.

admin server
: In the context of the Identity service, the worker process that provides access to the admin API.

Advanced Message Queuing Protocol (AMQP)
: The open standard messaging protocol used by OpenStack components for intra-service communications, provided by RabbitMQ, Qpid, or ZeroMQ.

Advanced RISC Machine (ARM)
: Lower power consumption CPU often found in mobile and embedded devices. Supported by OpenStack.

alert
: The Compute service can send alerts through its notification system, which includes a facility to create custom notification drivers. Alerts can be sent to and displayed on the horizon dashboard.

allocate
: The process of taking a floating IP address from the address pool so it can be associated with a fixed IP on a guest VM instance.

Amazon Kernel Image (AKI)
: Both a VM container format and disk format. Supported by Image service.

Amazon Machine Image (AMI)
: Both a VM container format and disk format. Supported by Image service.

Amazon Ramdisk Image (ARI)
: Both a VM container format and disk format. Supported by Image service.

Anvil
: A project that ports the shell script-based project named DevStack to Python.

Apache
: The Apache Software Foundation supports the Apache community of open-source software projects. These projects provide software products for the public good.

Apache License 2.0
: All OpenStack core projects are provided under the terms of the Apache License 2.0 license.

Apache Web Server
: The most common web server software currently used on the Internet.

API endpoint
: The daemon, worker, or service that a client communicates with to access an API. API endpoints can provide any number of services, such as authentication, sales data, performance meters, Compute VM commands, census data, and so on.

API extension
: Custom modules that extend some OpenStack core APIs.

API extension plug-in
: Alternative term for a Networking plug-in or Networking API extension.

API key
: Alternative term for an API token.

API server
: Any node running a daemon or worker that provides an API endpoint.

API token
: Passed to API requests and used by OpenStack to verify that the client is authorized to run the requested operation.

API version
: In OpenStack, the API version for a project is part of the URL. For example, `example.com/nova/v1/foobar`.

applet
: A Java program that can be embedded into a web page.

Application Programming Interface (API)
: A collection of specifications used to access a service, application, or program. Includes service calls, required parameters for each call, and the expected return values.

Application Catalog service
: OpenStack project that provides an application catalog service so that users can compose and deploy composite environments on an application abstraction level while managing the application lifecycle. The code name of the project is murano.

application server
: A piece of software that makes available another piece of software over a network.

Application Service Provider (ASP)
: Companies that rent specialized applications that help businesses and organizations provide additional services with lower cost.

Address Resolution Protocol (ARP)
: The protocol by which layer-3 IP addresses are resolved into layer-2 link local addresses.

arptables
: Tool used for maintaining Address Resolution Protocol packet filter rules in the Linux kernel firewall modules. Used along with iptables, ebtables, and ip6tables in Compute to provide firewall services for VMs.

associate
: The process associating a Compute floating IP address with a fixed IP address.

Asynchronous JavaScript and XML (AJAX)
: A group of interrelated web development techniques used on the client-side to create asynchronous web applications. Used extensively in horizon.

ATA over Ethernet (AoE)
: A disk storage protocol tunneled within Ethernet.

attach
: The process of connecting a VIF or vNIC to a L2 network in Networking. In the context of Compute, this process connects a storage volume to an instance.

attachment (network)
: Association of an interface ID to a logical port. Plugs an interface into a port.

auditing
: Provided in Compute through the system usage data facility.

auditor
: A worker process that verifies the integrity of Object Storage objects, containers, and accounts. Auditors is the collective term for the Object Storage account auditor, container auditor, and object auditor.

Austin
: The code name for the initial release of OpenStack. The first design summit took place in Austin, Texas, US.

auth node
: Alternative term for an Object Storage authorization node.

authentication
: The process that confirms that the user, process, or client is really who they say they are through private key, secret token, password, fingerprint, or similar method.

authentication token
: A string of text provided to the client after authentication. Must be provided by the user or process in subsequent requests to the API endpoint.

AuthN
: The Identity service component that provides authentication services.

authorization
: The act of verifying that a user, process, or client is authorized to perform an action.

authorization node
: An Object Storage node that provides authorization services.

AuthZ
: The Identity component that provides high-level authorization services.

Auto ACK
: Configuration setting within RabbitMQ that enables or disables message acknowledgment. Enabled by default.

auto declare
: A Compute RabbitMQ setting that determines whether a message exchange is automatically created when the program starts.

availability zone
: An Amazon EC2 concept of an isolated area that is used for fault tolerance. Do not confuse with an OpenStack Compute zone or cell.

AWS
: Amazon Web Services.

AWS CloudFormation template
: AWS CloudFormation allows AWS users to create and manage a collection of related resources. The Orchestration service supports a CloudFormation-compatible format (CFN).

B

back end
: Interactions and processes that are obfuscated from the user, such as Compute volume mount, data transmission to an iSCSI target by a daemon, or Object Storage object integrity checks.

back-end catalog
: The storage method used by the Identity service catalog service to store and retrieve information about API endpoints that are available to the client. Examples include an SQL database, LDAP database, or KVS back end.

back-end store
: The persistent data store used to save and retrieve information for a service, such as lists of Object Storage objects, current state of guest VMs, lists of user names, and so on. Also, the method that the Image service uses to get and store VM images. Options include Object Storage, local file system, S3, and HTTP.

backup restore and disaster recovery as a service
: The OpenStack project that provides integrated tooling for backing up, restoring, and recovering file systems, instances, or database backups. The project name is freezer.

bandwidth
: The amount of available data used by communication resources, such as the Internet. Represents the amount of data that is used to download things or the amount of data available to download.

barbican
: Code name of the key management service for OpenStack.

bare
: An Image service container format that indicates that no container exists for the VM image.

Bare Metal service
: OpenStack project that provisions bare metal, as opposed to virtual, machines. The code name for the project is ironic.

base image
An OpenStack-provided image.

Bell-LaPadula model
A security model that focuses on data confidentiality and controlled access to classified information. This model divide the entities into subjects and objects. The clearance of a subject is compared to the classification of the object to determine if the subject is authorized for the specific access mode. The clearance or classification scheme is expressed in terms of a lattice.

Benchmark service
OpenStack project that provides a framework for performance analysis and benchmarking of individual OpenStack components as well as full production OpenStack cloud deployments. The code name of the project is rally.

Bexar
A grouped release of projects related to OpenStack that came out in February of 2011. It included only Compute (nova) and Object Storage (swift). Bexar is the code name for the second release of OpenStack. The design summit took place in San Antonio, Texas, US, which is the county seat for Bexar county.

binary
Information that consists solely of ones and zeroes, which is the language of computers.

bit
A bit is a single digit number that is in base of 2 (either a zero or one). Bandwidth usage is measured in bits per second.

bits per second (BPS)
The universal measurement of how quickly data is transferred from place to place.

block device
A device that moves data in the form of blocks. These device nodes interface the devices, such as hard disks, CD-ROM drives, flash drives, and other addressable regions of memory.

block migration
A method of VM live migration used by KVM to evacuate instances from one host to another with very little downtime during a user-initiated switchover. Does not require shared storage. Supported by Compute.

Block Storage service
The OpenStack core project that enables management of volumes, volume snapshots, and volume types. The project name of Block Storage is cinder.

Block Storage API
An API on a separate endpoint for attaching, detaching, and creating block storage for compute VMs.

BMC
Baseboard Management Controller. The intelligence in the IPMI architecture, which is a specialized micro-controller that is embedded on the motherboard of a computer and acts as a server. Manages the interface between system management software and platform hardware.

bootable disk image
A type of VM image that exists as a single, bootable file.

Bootstrap Protocol (BOOTP)
A network protocol used by a network client to obtain an IP address from a configuration server. Provided in Compute through the dnsmasq daemon when using either the FlatDHCP manager or VLAN manager network manager.

Border Gateway Protocol (BGP)
The Border Gateway Protocol is a dynamic routing protocol that connects autonomous systems. Considered the backbone of the Internet, this protocol connects disparate networks to form a larger network.

browser
Any client software that enables a computer or device to access the Internet.

builder file
Contains configuration information that Object Storage uses to reconfigure a ring or to re-create it from scratch after a serious failure.

bursting
The practice of utilizing a secondary environment to elastically build instances on-demand when the primary

environment is resource constrained.

button class
: A group of related button types within horizon. Buttons to start, stop, and suspend VMs are in one class. Buttons to associate and disassociate floating IP addresses are in another class, and so on.

byte
: Set of bits that make up a single character; there are usually 8 bits to a byte.

C

CA
: Certificate Authority or Certification Authority. In cryptography, an entity that issues digital certificates. The digital certificate certifies the ownership of a public key by the named subject of the certificate. This enables others (relying parties) to rely upon signatures or assertions made by the private key that corresponds to the certified public key. In this model of trust relationships, a CA is a trusted third party for both the subject (owner) of the certificate and the party relying upon the certificate. CAs are characteristic of many public key infrastructure (PKI) schemes.

cache pruner
: A program that keeps the Image service VM image cache at or below its configured maximum size.

Cactus
: An OpenStack grouped release of projects that came out in the spring of 2011. It included Compute (nova), Object Storage (swift), and the Image service (glance). Cactus is a city in Texas, US and is the code name for the third release of OpenStack. When OpenStack releases went from three to six months long, the code name of the release changed to match a geography nearest the previous summit.

CADF
: Cloud Auditing Data Federation (CADF) is a specification for audit event data. CADF is supported by OpenStack Identity.

CALL
: One of the RPC primitives used by the OpenStack message queue software. Sends a message and waits for a response.

capability
: Defines resources for a cell, including CPU, storage, and networking. Can apply to the specific services within a cell or a whole cell.

capacity cache
: A Compute back-end database table that contains the current workload, amount of free RAM, and number of VMs running on each host. Used to determine on which host a VM starts.

capacity updater
: A notification driver that monitors VM instances and updates the capacity cache as needed.

CAST
: One of the RPC primitives used by the OpenStack message queue software. Sends a message and does not wait for a response.

catalog
: A list of API endpoints that are available to a user after authentication with the Identity service.

catalog service
: An Identity service that lists API endpoints that are available to a user after authentication with the Identity service.

ceilometer
: The project name for the Telemetry service, which is an integrated project that provides metering and measuring facilities for OpenStack.

cell
: Provides logical partitioning of Compute resources in a child and parent relationship. Requests are passed from parent cells to child cells if the parent cannot provide the requested resource.

cell forwarding

A Compute option that enables parent cells to pass resource requests to child cells if the parent cannot provide the requested resource.

cell manager
: The Compute component that contains a list of the current capabilities of each host within the cell and routes requests as appropriate.

CentOS
: A Linux distribution that is compatible with OpenStack.

Ceph
: Massively scalable distributed storage system that consists of an object store, block store, and POSIX-compatible distributed file system. Compatible with OpenStack.

CephFS
: The POSIX-compliant file system provided by Ceph.

certificate authority
: A simple certificate authority provided by Compute for cloudpipe VPNs and VM image decryption.

Challenge-Handshake Authentication Protocol (CHAP)
: An iSCSI authentication method supported by Compute.

chance scheduler
: A scheduling method used by Compute that randomly chooses an available host from the pool.

changes since
: A Compute API parameter that downloads changes to the requested item since your last request, instead of downloading a new, fresh set of data and comparing it against the old data.

Chef
: An operating system configuration management tool supporting OpenStack deployments.

child cell
: If a requested resource such as CPU time, disk storage, or memory is not available in the parent cell, the request is forwarded to its associated child cells. If the child cell can fulfill the request, it does. Otherwise, it attempts to pass the request to any of its children.

cinder
: A core OpenStack project that provides block storage services for VMs.

CirrOS
: A minimal Linux distribution designed for use as a test image on clouds such as OpenStack.

Cisco neutron plug-in
: A Networking plug-in for Cisco devices and technologies, including UCS and Nexus.

cloud architect
: A person who plans, designs, and oversees the creation of clouds.

cloud computing
: A model that enables access to a shared pool of configurable computing resources, such as networks, servers, storage, applications, and services, that can be rapidly provisioned and released with minimal management effort or service provider interaction.

cloud controller
: Collection of Compute components that represent the global state of the cloud; talks to services, such as Identity authentication, Object Storage, and node/storage workers through a queue.

cloud controller node
: A node that runs network, volume, API, scheduler, and image services. Each service may be broken out into separate nodes for scalability or availability.

Cloud Data Management Interface (CDMI)
: SINA standard that defines a RESTful API for managing objects in the cloud, currently unsupported in OpenStack.

Cloud Infrastructure Management Interface (CIMI)
: An in-progress specification for cloud management. Currently unsupported in OpenStack.

cloud-init
: A package commonly installed in VM images that performs initialization of an instance after boot using information

that it retrieves from the metadata service, such as the SSH public key and user data.

cloudadmin
: One of the default roles in the Compute RBAC system. Grants complete system access.

Cloudbase-Init
: A Windows project providing guest initialization features, similar to cloud-init.

cloudpipe
: A compute service that creates VPNs on a per-project basis.

cloudpipe image
: A pre-made VM image that serves as a cloudpipe server. Essentially, OpenVPN running on Linux.

Clustering service
: The OpenStack project that OpenStack project that implements clustering services and libraries for the management of groups of homogeneous objects exposed by other OpenStack services. The project name of Clustering service is senlin.

CMDB
: Configuration Management Database.

congress
: OpenStack project that provides the Governance service.

command filter
: Lists allowed commands within the Compute rootwrap facility.

Common Internet File System (CIFS)
: A file sharing protocol. It is a public or open variation of the original Server Message Block (SMB) protocol developed and used by Microsoft. Like the SMB protocol, CIFS runs at a higher level and uses the TCP/IP protocol.

community project
: A project that is not officially endorsed by the OpenStack Foundation. If the project is successful enough, it might be elevated to an incubated project and then to a core project, or it might be merged with the main code trunk.

compression
: Reducing the size of files by special encoding, the file can be decompressed again to its original content. OpenStack supports compression at the Linux file system level but does not support compression for things such as Object Storage objects or Image service VM images.

Compute service
: The OpenStack core project that provides compute services. The project name of Compute service is nova.

Compute API
: The nova-api daemon provides access to nova services. Can communicate with other APIs, such as the Amazon EC2 API.

compute controller
: The Compute component that chooses suitable hosts on which to start VM instances.

compute host
: Physical host dedicated to running compute nodes.

compute node
: A node that runs the nova-compute daemon that manages VM instances that provide a wide range of services, such as web applications and analytics.

Compute service
: Name for the Compute component that manages VMs.

compute worker
: The Compute component that runs on each compute node and manages the VM instance lifecycle, including run, reboot, terminate, attach/detach volumes, and so on. Provided by the nova-compute daemon.

concatenated object
: A set of segment objects that Object Storage combines and sends to the client.

conductor
: In Compute, conductor is the process that proxies database requests from the compute process. Using conductor improves security because compute nodes do not need direct access to the database.

consistency window
: The amount of time it takes for a new Object Storage object to become accessible to all clients.

console log
: Contains the output from a Linux VM console in Compute.

container
: Organizes and stores objects in Object Storage. Similar to the concept of a Linux directory but cannot be nested. Alternative term for an Image service container format.

container auditor
: Checks for missing replicas or incorrect objects in specified Object Storage containers through queries to the SQLite back-end database.

container database
: A SQLite database that stores Object Storage containers and container metadata. The container server accesses this database.

container format
: A wrapper used by the Image service that contains a VM image and its associated metadata, such as machine state, OS disk size, and so on.

container server
: An Object Storage server that manages containers.

Containers service
: OpenStack project that provides a set of services for management of application containers in a multi-tenant cloud environment. The code name of the project name is magnum.

container service
: The Object Storage component that provides container services, such as create, delete, list, and so on.

content delivery network (CDN)
: A content delivery network is a specialized network that is used to distribute content to clients, typically located close to the client for increased performance.

controller node
: Alternative term for a cloud controller node.

core API
: Depending on context, the core API is either the OpenStack API or the main API of a specific core project, such as Compute, Networking, Image service, and so on.

core service
: An official OpenStack service defined as core by DefCore Committee. Currently, consists of Block Storage service (cinder), Compute service (nova), Identity service (keystone), Image service (glance), Networking service (neutron), and Object Storage service (swift).

cost
: Under the Compute distributed scheduler, this is calculated by looking at the capabilities of each host relative to the flavor of the VM instance being requested.

credentials
: Data that is only known to or accessible by a user and used to verify that the user is who he says he is. Credentials are presented to the server during authentication. Examples include a password, secret key, digital certificate, and fingerprint.

Cross-Origin Resource Sharing (CORS)
: A mechanism that allows many resources (for example, fonts, JavaScript) on a web page to be requested from another domain outside the domain from which the resource originated. In particular, JavaScript's AJAX calls can use the XMLHttpRequest mechanism.

Crowbar
: An open source community project by Dell that aims to provide all necessary services to quickly deploy clouds.

current workload
: An element of the Compute capacity cache that is calculated based on the number of build, snapshot, migrate, and resize operations currently in progress on a given host.

customer
: Alternative term for tenant.

customization module
: A user-created Python module that is loaded by horizon to change the look and feel of the dashboard.

D

daemon
: A process that runs in the background and waits for requests. May or may not listen on a TCP or UDP port. Do not confuse with a worker.

DAC
: Discretionary access control. Governs the ability of subjects to access objects, while enabling users to make policy decisions and assign security attributes. The traditional UNIX system of users, groups, and read-write-execute permissions is an example of DAC.

Dashboard
: The web-based management interface for OpenStack. An alternative name for horizon.

data encryption
: Both Image service and Compute support encrypted virtual machine (VM) images (but not instances). In-transit data encryption is supported in OpenStack using technologies such as HTTPS, SSL, TLS, and SSH. Object Storage does not support object encryption at the application level but may support storage that uses disk encryption.

database ID
: A unique ID given to each replica of an Object Storage database.

database replicator
: An Object Storage component that copies changes in the account, container, and object databases to other nodes.

Database service
: An integrated project that provide scalable and reliable Cloud Database-as-a-Service functionality for both relational and non-relational database engines. The project name of Database service is trove.

Data Processing service
: OpenStack project that provides a scalable data-processing stack and associated management interfaces. The code name for the project is sahara.

data store
: A database engine supported by the Database service.

deallocate
: The process of removing the association between a floating IP address and a fixed IP address. Once this association is removed, the floating IP returns to the address pool.

Debian
: A Linux distribution that is compatible with OpenStack.

deduplication
: The process of finding duplicate data at the disk block, file, and/or object level to minimize storage use—currently unsupported within OpenStack.

default panel
: The default panel that is displayed when a user accesses the horizon dashboard.

default tenant
: New users are assigned to this tenant if no tenant is specified when a user is created.

default token
: An Identity service token that is not associated with a specific tenant and is exchanged for a scoped token.

delayed delete
: An option within Image service so that an image is deleted after a predefined number of seconds instead of immediately.

delivery mode

Setting for the Compute RabbitMQ message delivery mode; can be set to either transient or persistent.

denial of service (DoS)
: Denial of service (DoS) is a short form for denial-of-service attack. This is a malicious attempt to prevent legitimate users from using a service.

deprecated auth
: An option within Compute that enables administrators to create and manage users through the `nova-manage` command as opposed to using the Identity service.

designate
: Code name for the DNS service project for OpenStack.

Desktop-as-a-Service
: A platform that provides a suite of desktop environments that users access to receive a desktop experience from any location. This may provide general use, development, or even homogeneous testing environments.

developer
: One of the default roles in the Compute RBAC system and the default role assigned to a new user.

device ID
: Maps Object Storage partitions to physical storage devices.

device weight
: Distributes partitions proportionately across Object Storage devices based on the storage capacity of each device.

DevStack
: Community project that uses shell scripts to quickly build complete OpenStack development environments.

DHCP
: Dynamic Host Configuration Protocol. A network protocol that configures devices that are connected to a network so that they can communicate on that network by using the Internet Protocol (IP). The protocol is implemented in a client-server model where DHCP clients request configuration data, such as an IP address, a default route, and one or more DNS server addresses from a DHCP server.

DHCP agent
: OpenStack Networking agent that provides DHCP services for virtual networks.

Diablo
: A grouped release of projects related to OpenStack that came out in the fall of 2011, the fourth release of OpenStack. It included Compute (nova 2011.3), Object Storage (swift 1.4.3), and the Image service (glance). Diablo is the code name for the fourth release of OpenStack. The design summit took place in the Bay Area near Santa Clara, California, US and Diablo is a nearby city.

direct consumer
: An element of the Compute RabbitMQ that comes to life when a RPC call is executed. It connects to a direct exchange through a unique exclusive queue, sends the message, and terminates.

direct exchange
: A routing table that is created within the Compute RabbitMQ during RPC calls; one is created for each RPC call that is invoked.

direct publisher
: Element of RabbitMQ that provides a response to an incoming MQ message.

disassociate
: The process of removing the association between a floating IP address and fixed IP and thus returning the floating IP address to the address pool.

disk encryption
: The ability to encrypt data at the file system, disk partition, or whole-disk level. Supported within Compute VMs.

disk format
: The underlying format that a disk image for a VM is stored as within the Image service back-end store. For example, AMI, ISO, QCOW2, VMDK, and so on.

dispersion
: In Object Storage, tools to test and ensure dispersion of objects and containers to ensure fault tolerance.

distributed virtual router (DVR)

Mechanism for highly-available multi-host routing when using OpenStack Networking (neutron).

Django
: A web framework used extensively in horizon.

DNS
: Domain Name System. A hierarchical and distributed naming system for computers, services, and resources connected to the Internet or a private network. Associates a human-friendly names to IP addresses.

DNS record
: A record that specifies information about a particular domain and belongs to the domain.

DNS service
: OpenStack project that provides scalable, on demand, self service access to authoritative DNS services, in a technology-agnostic manner. The code name for the project is designate.

dnsmasq
: Daemon that provides DNS, DHCP, BOOTP, and TFTP services for virtual networks.

domain
: An Identity API v3 entity. Represents a collection of projects, groups and users that defines administrative boundaries for managing OpenStack Identity entities. On the Internet, separates a website from other sites. Often, the domain name has two or more parts that are separated by dots. For example, yahoo.com, usa.gov, harvard.edu, or mail.yahoo.com. Also, a domain is an entity or container of all DNS-related information containing one or more records.

Domain Name System (DNS)
: A system by which Internet domain name-to-address and address-to-name resolutions are determined. DNS helps navigate the Internet by translating the IP address into an address that is easier to remember. For example, translating 111.111.111.1 into www.yahoo.com. All domains and their components, such as mail servers, utilize DNS to resolve to the appropriate locations. DNS servers are usually set up in a master-slave relationship such that failure of the master invokes the slave. DNS servers might also be clustered or replicated such that changes made to one DNS server are automatically propagated to other active servers. In Compute, the support that enables associating DNS entries with floating IP addresses, nodes, or cells so that hostnames are consistent across reboots.

download
: The transfer of data, usually in the form of files, from one computer to another.

DRTM
: Dynamic root of trust measurement.

durable exchange
: The Compute RabbitMQ message exchange that remains active when the server restarts.

durable queue
: A Compute RabbitMQ message queue that remains active when the server restarts.

Dynamic Host Configuration Protocol (DHCP)
: A method to automatically configure networking for a host at boot time. Provided by both Networking and Compute.

Dynamic HyperText Markup Language (DHTML)
: Pages that use HTML, JavaScript, and Cascading Style Sheets to enable users to interact with a web page or show simple animation.

E

east-west traffic
: Network traffic between servers in the same cloud or data center. See also north-south traffic.

EBS boot volume
: An Amazon EBS storage volume that contains a bootable VM image, currently unsupported in OpenStack.

ebtables
: Filtering tool for a Linux bridging firewall, enabling filtering of network traffic passing through a Linux bridge. Used in Compute along with arptables, iptables, and ip6tables to ensure isolation of network communications.

EC2
: The Amazon commercial compute product, similar to Compute.

EC2 access key
: Used along with an EC2 secret key to access the Compute EC2 API.

EC2 API
: OpenStack supports accessing the Amazon EC2 API through Compute.

EC2 Compatibility API
: A Compute component that enables OpenStack to communicate with Amazon EC2.

EC2 secret key
: Used along with an EC2 access key when communicating with the Compute EC2 API; used to digitally sign each request.

Elastic Block Storage (EBS)
: The Amazon commercial block storage product.

encryption
: OpenStack supports encryption technologies such as HTTPS, SSH, SSL, TLS, digital certificates, and data encryption.

endpoint
: See API endpoint.

endpoint registry
: Alternative term for an Identity service catalog.

encapsulation
: The practice of placing one packet type within another for the purposes of abstracting or securing data. Examples include GRE, MPLS, or IPsec.

endpoint template
: A list of URL and port number endpoints that indicate where a service, such as Object Storage, Compute, Identity, and so on, can be accessed.

entity
: Any piece of hardware or software that wants to connect to the network services provided by Networking, the network connectivity service. An entity can make use of Networking by implementing a VIF.

ephemeral image
: A VM image that does not save changes made to its volumes and reverts them to their original state after the instance is terminated.

ephemeral volume
: Volume that does not save the changes made to it and reverts to its original state when the current user relinquishes control.

Essex
: A grouped release of projects related to OpenStack that came out in April 2012, the fifth release of OpenStack. It included Compute (nova 2012.1), Object Storage (swift 1.4.8), Image (glance), Identity (keystone), and Dashboard (horizon). Essex is the code name for the fifth release of OpenStack. The design summit took place in Boston, Massachusetts, US and Essex is a nearby city.

ESXi
: An OpenStack-supported hypervisor.

ETag
: MD5 hash of an object within Object Storage, used to ensure data integrity.

euca2ools
: A collection of command-line tools for administering VMs; most are compatible with OpenStack.

Eucalyptus Kernel Image (EKI)
: Used along with an ERI to create an EMI.

Eucalyptus Machine Image (EMI)
: VM image container format supported by Image service.

Eucalyptus Ramdisk Image (ERI)

Used along with an EKI to create an EMI.

evacuate
: The process of migrating one or all virtual machine (VM) instances from one host to another, compatible with both shared storage live migration and block migration.

exchange
: Alternative term for a RabbitMQ message exchange.

exchange type
: A routing algorithm in the Compute RabbitMQ.

exclusive queue
: Connected to by a direct consumer in RabbitMQ—Compute, the message can be consumed only by the current connection.

extended attributes (xattr)
: File system option that enables storage of additional information beyond owner, group, permissions, modification time, and so on. The underlying Object Storage file system must support extended attributes.

extension
: Alternative term for an API extension or plug-in. In the context of Identity service, this is a call that is specific to the implementation, such as adding support for OpenID.

external network
: A network segment typically used for instance Internet access.

extra specs
: Specifies additional requirements when Compute determines where to start a new instance. Examples include a minimum amount of network bandwidth or a GPU.

F

FakeLDAP
: An easy method to create a local LDAP directory for testing Identity and Compute. Requires Redis.

fan-out exchange
: Within RabbitMQ and Compute, it is the messaging interface that is used by the scheduler service to receive capability messages from the compute, volume, and network nodes.

federated identity
: A method to establish trusts between identity providers and the OpenStack cloud.

Fedora
: A Linux distribution compatible with OpenStack.

Fibre Channel
: Storage protocol similar in concept to TCP/IP; encapsulates SCSI commands and data.

Fibre Channel over Ethernet (FCoE)
: The fibre channel protocol tunneled within Ethernet.

fill-first scheduler
: The Compute scheduling method that attempts to fill a host with VMs rather than starting new VMs on a variety of hosts.

filter
: The step in the Compute scheduling process when hosts that cannot run VMs are eliminated and not chosen.

firewall
: Used to restrict communications between hosts and/or nodes, implemented in Compute using iptables, arptables, ip6tables, and ebtables.

FWaaS
: A Networking extension that provides perimeter firewall functionality.

fixed IP address
: An IP address that is associated with the same instance each time that instance boots, is generally not accessible to

end users or the public Internet, and is used for management of the instance.

Flat Manager
: The Compute component that gives IP addresses to authorized nodes and assumes DHCP, DNS, and routing configuration and services are provided by something else.

flat mode injection
: A Compute networking method where the OS network configuration information is injected into the VM image before the instance starts.

flat network
: Virtual network type that uses neither VLANs nor tunnels to segregate tenant traffic. Each flat network typically requires a separate underlying physical interface defined by bridge mappings. However, a flat network can contain multiple subnets.

FlatDHCP Manager
: The Compute component that provides dnsmasq (DHCP, DNS, BOOTP, TFTP) and radvd (routing) services.

flavor
: Alternative term for a VM instance type.

flavor ID
: UUID for each Compute or Image service VM flavor or instance type.

floating IP address
: An IP address that a project can associate with a VM so that the instance has the same public IP address each time that it boots. You create a pool of floating IP addresses and assign them to instances as they are launched to maintain a consistent IP address for maintaining DNS assignment.

Folsom
: A grouped release of projects related to OpenStack that came out in the fall of 2012, the sixth release of OpenStack. It includes Compute (nova), Object Storage (swift), Identity (keystone), Networking (neutron), Image service (glance), and Volumes or Block Storage (cinder). Folsom is the code name for the sixth release of OpenStack. The design summit took place in San Francisco, California, US and Folsom is a nearby city.

FormPost
: Object Storage middleware that uploads (posts) an image through a form on a web page.

freezer
: OpenStack project that provides backup restore and disaster recovery as a service.

front end
: The point where a user interacts with a service; can be an API endpoint, the horizon dashboard, or a command-line tool.

G

gateway
: An IP address, typically assigned to a router, that passes network traffic between different networks.

generic receive offload (GRO)
: Feature of certain network interface drivers that combines many smaller received packets into a large packet before delivery to the kernel IP stack.

generic routing encapsulation (GRE)
: Protocol that encapsulates a wide variety of network layer protocols inside virtual point-to-point links.

glance
: A core project that provides the OpenStack Image service.

glance API server
: Processes client requests for VMs, updates Image service metadata on the registry server, and communicates with the store adapter to upload VM images from the back-end store.

glance registry
: Alternative term for the Image service image registry.

global endpoint template
: The Identity service endpoint template that contains services available to all tenants.

GlusterFS
: A file system designed to aggregate NAS hosts, compatible with OpenStack.

golden image
: A method of operating system installation where a finalized disk image is created and then used by all nodes without modification.

Governance service
: OpenStack project to provide Governance-as-a-Service across any collection of cloud services in order to monitor, enforce, and audit policy over dynamic infrastructure. The code name for the project is congress.

Graphic Interchange Format (GIF)
: A type of image file that is commonly used for animated images on web pages.

Graphics Processing Unit (GPU)
: Choosing a host based on the existence of a GPU is currently unsupported in OpenStack.

Green Threads
: The cooperative threading model used by Python; reduces race conditions and only context switches when specific library calls are made. Each OpenStack service is its own thread.

Grizzly
: The code name for the seventh release of OpenStack. The design summit took place in San Diego, California, US and Grizzly is an element of the state flag of California.

Group
: An Identity v3 API entity. Represents a collection of users that is owned by a specific domain.

guest OS
: An operating system instance running under the control of a hypervisor.

H

Hadoop
: Apache Hadoop is an open source software framework that supports data-intensive distributed applications.

Hadoop Distributed File System (HDFS)
: A distributed, highly fault-tolerant file system designed to run on low-cost commodity hardware.

handover
: An object state in Object Storage where a new replica of the object is automatically created due to a drive failure.

hard reboot
: A type of reboot where a physical or virtual power button is pressed as opposed to a graceful, proper shutdown of the operating system.

Havana
: The code name for the eighth release of OpenStack. The design summit took place in Portland, Oregon, US and Havana is an unincorporated community in Oregon.

heat
: An integrated project that aims to orchestrate multiple cloud applications for OpenStack.

Heat Orchestration Template (HOT)
: Heat input in the format native to OpenStack.

health monitor
: Determines whether back-end members of a VIP pool can process a request. A pool can have several health monitors associated with it. When a pool has several monitors associated with it, all monitors check each member of the pool. All monitors must declare a member to be healthy for it to stay active.

high availability (HA)
: A high availability system design approach and associated service implementation ensures that a prearranged level of operational performance will be met during a contractual measurement period. High availability systems seeks to

minimize system downtime and data loss.

horizon
: OpenStack project that provides a dashboard, which is a web interface.

horizon plug-in
: A plug-in for the OpenStack dashboard (horizon).

host
: A physical computer, not a VM instance (node).

host aggregate
: A method to further subdivide availability zones into hypervisor pools, a collection of common hosts.

Host Bus Adapter (HBA)
: Device plugged into a PCI slot, such as a fibre channel or network card.

hybrid cloud
: A hybrid cloud is a composition of two or more clouds (private, community or public) that remain distinct entities but are bound together, offering the benefits of multiple deployment models. Hybrid cloud can also mean the ability to connect colocation, managed and/or dedicated services with cloud resources.

Hyper-V
: One of the hypervisors supported by OpenStack.

hyperlink
: Any kind of text that contains a link to some other site, commonly found in documents where clicking on a word or words opens up a different website.

Hypertext Transfer Protocol (HTTP)
: An application protocol for distributed, collaborative, hypermedia information systems. It is the foundation of data communication for the World Wide Web. Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text. HTTP is the protocol to exchange or transfer hypertext.

Hypertext Transfer Protocol Secure (HTTPS)
: An encrypted communications protocol for secure communication over a computer network, with especially wide deployment on the Internet. Technically, it is not a protocol in and of itself; rather, it is the result of simply layering the Hypertext Transfer Protocol (HTTP) on top of the TLS or SSL protocol, thus adding the security capabilities of TLS or SSL to standard HTTP communications. most OpenStack API endpoints and many inter-component communications support HTTPS communication.

hypervisor
: Software that arbitrates and controls VM access to the actual underlying hardware.

hypervisor pool
: A collection of hypervisors grouped together through host aggregates.

I

IaaS
: Infrastructure-as-a-Service. IaaS is a provisioning model in which an organization outsources physical components of a data center, such as storage, hardware, servers, and networking components. A service provider owns the equipment and is responsible for housing, operating and maintaining it. The client typically pays on a per-use basis. IaaS is a model for providing cloud services.

Icehouse
: The code name for the ninth release of OpenStack. The design summit took place in Hong Kong and Ice House is a street in that city.

ICMP
: Internet Control Message Protocol, used by network devices for control messages. For example, `ping` uses ICMP to test connectivity.

ID number
: Unique numeric ID associated with each user in Identity, conceptually similar to a Linux or LDAP UID.

Identity API
: Alternative term for the Identity service API.

Identity back end
: The source used by Identity service to retrieve user information; an OpenLDAP server, for example.

identity provider
: A directory service, which allows users to login with a user name and password. It is a typical source of authentication tokens.

Identity service
: The OpenStack core project that provides a central directory of users mapped to the OpenStack services they can access. It also registers endpoints for OpenStack services. It acts as a common authentication system. The project name of Identity is keystone.

Identity service API
: The API used to access the OpenStack Identity service provided through keystone.

IDS
: Intrusion Detection System.

image
: A collection of files for a specific operating system (OS) that you use to create or rebuild a server. OpenStack provides pre-built images. You can also create custom images, or snapshots, from servers that you have launched. Custom images can be used for data backups or as “gold” images for additional servers.

Image API
: The Image service API endpoint for management of VM images.

image cache
: Used by Image service to obtain images on the local host rather than re-downloading them from the image server each time one is requested.

image ID
: Combination of a URI and UUID used to access Image service VM images through the image API.

image membership
: A list of tenants that can access a given VM image within Image service.

image owner
: The tenant who owns an Image service virtual machine image.

image registry
: A list of VM images that are available through Image service.

Image service
: An OpenStack core project that provides discovery, registration, and delivery services for disk and server images. The project name of the Image service is glance.

Image service API
: Alternative name for the glance image API.

image status
: The current status of a VM image in Image service, not to be confused with the status of a running instance.

image store
: The back-end store used by Image service to store VM images, options include Object Storage, local file system, S3, or HTTP.

image UUID
: UUID used by Image service to uniquely identify each VM image.

incubated project
: A community project may be elevated to this status and is then promoted to a core project.

ingress filtering
: The process of filtering incoming network traffic. Supported by Compute.

INI
: The OpenStack configuration files use an INI format to describe options and their values. It consists of sections and key value pairs.

injection
: The process of putting a file into a virtual machine image before the instance is started.

instance
: A running VM, or a VM in a known state such as suspended, that can be used like a hardware server.

instance ID
: Alternative term for instance UUID.

instance state
: The current state of a guest VM image.

instance tunnels network
: A network segment used for instance traffic tunnels between compute nodes and the network node.

instance type
: Describes the parameters of the various virtual machine images that are available to users; includes parameters such as CPU, storage, and memory. Alternative term for flavor.

instance type ID
: Alternative term for a flavor ID.

instance UUID
: Unique ID assigned to each guest VM instance.

interface
: A physical or virtual device that provides connectivity to another device or medium.

interface ID
: Unique ID for a Networking VIF or vNIC in the form of a UUID.

Internet protocol (IP)
: Principal communications protocol in the internet protocol suite for relaying datagrams across network boundaries.

Internet Service Provider (ISP)
: Any business that provides Internet access to individuals or businesses.

Internet Small Computer System Interface (iSCSI)
: Storage protocol that encapsulates SCSI frames for transport over IP networks.

ironic
: OpenStack project that provisions bare metal, as opposed to virtual, machines.

IOPS
: IOPS (Input/Output Operations Per Second) are a common performance measurement used to benchmark computer storage devices like hard disk drives, solid state drives, and storage area networks.

IP address
: Number that is unique to every computer system on the Internet. Two versions of the Internet Protocol (IP) are in use for addresses: IPv4 and IPv6.

IP Address Management (IPAM)
: The process of automating IP address allocation, deallocation, and management. Currently provided by Compute, melange, and Networking.

IPL
: Initial Program Loader.

IPMI
: Intelligent Platform Management Interface. IPMI is a standardized computer system interface used by system administrators for out-of-band management of computer systems and monitoring of their operation. In layman's terms, it is a way to manage a computer using a direct network connection, whether it is turned on or not; connecting to the hardware rather than an operating system or login shell.

ip6tables
: Tool used to set up, maintain, and inspect the tables of IPv6 packet filter rules in the Linux kernel. In OpenStack Compute, ip6tables is used along with arptables, ebtables, and iptables to create firewalls for both nodes and VMs.

ipset
: Extension to iptables that allows creation of firewall rules that match entire "sets" of IP addresses simultaneously. These sets reside in indexed data structures to increase efficiency, particularly on systems with a large quantity of

rules.

iptables
: Used along with arptables and ebtables, iptables create firewalls in Compute. iptables are the tables provided by the Linux kernel firewall (implemented as different Netfilter modules) and the chains and rules it stores. Different kernel modules and programs are currently used for different protocols: iptables applies to IPv4, ip6tables to IPv6, arptables to ARP, and ebtables to Ethernet frames. Requires root privilege to manipulate.

IQN
: iSCSI Qualified Name (IQN) is the format most commonly used for iSCSI names, which uniquely identify nodes in an iSCSI network. All IQNs follow the pattern iqn.yyyy-mm.domain:identifier, where 'yyyy-mm' is the year and month in which the domain was registered, 'domain' is the reversed domain name of the issuing organization, and 'identifier' is an optional string which makes each IQN under the same domain unique. For example, 'iqn.2015-10.org.openstack.408ae959bce1'.

iSCSI
: The SCSI disk protocol tunneled within Ethernet, supported by Compute, Object Storage, and Image service.

ISO9660
: One of the VM image disk formats supported by Image service.

itsec
: A default role in the Compute RBAC system that can quarantine an instance in any project.

J

Java
: A programming language that is used to create systems that involve more than one computer by way of a network.

JavaScript
: A scripting language that is used to build web pages.

JavaScript Object Notation (JSON)
: One of the supported response formats in OpenStack.

Jenkins
: Tool used to run jobs automatically for OpenStack development.

jumbo frame
: Feature in modern Ethernet networks that supports frames up to approximately 9000 bytes.

Juno
: The code name for the tenth release of OpenStack. The design summit took place in Atlanta, Georgia, US and Juno is an unincorporated community in Georgia.

K

Kerberos
: A network authentication protocol which works on the basis of tickets. Kerberos allows nodes communication over a non-secure network, and allows nodes to prove their identity to one another in a secure manner.

kernel-based VM (KVM)
: An OpenStack-supported hypervisor. KVM is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V), ARM, IBM Power, and IBM zSeries. It consists of a loadable kernel module, that provides the core virtualization infrastructure and a processor specific module.

Key Manager service
: OpenStack project that produces a secret storage and generation system capable of providing key management for services wishing to enable encryption features. The code name of the project is barbican.

keystone
: The project that provides OpenStack Identity services.

Kickstart
: A tool to automate system configuration and installation on Red Hat, Fedora, and CentOS-based Linux distributions.

Kilo
: The code name for the eleventh release of OpenStack. The design summit took place in Paris, France. Due to delays in the name selection, the release was known only as K. Because `k` is the unit symbol for kilo and the reference artifact is stored near Paris in the Pavillon de Breteuil in Sèvres, the community chose Kilo as the release name.

L

large object
: An object within Object Storage that is larger than 5 GB.

Launchpad
: The collaboration site for OpenStack.

Layer-2 network
: Term used in the OSI network architecture for the data link layer. The data link layer is responsible for media access control, flow control and detecting and possibly correcting errors that may occur in the physical layer.

Layer-3 network
: Term used in the OSI network architecture for the network layer. The network layer is responsible for packet forwarding including routing from one node to another.

Layer-2 (L2) agent
: OpenStack Networking agent that provides layer-2 connectivity for virtual networks.

Layer-3 (L3) agent
: OpenStack Networking agent that provides layer-3 (routing) services for virtual networks.

Liberty
: The code name for the twelfth release of OpenStack. The design summit took place in Vancouver, Canada and Liberty is the name of a village in the Canadian province of Saskatchewan.

libvirt
: Virtualization API library used by OpenStack to interact with many of its supported hypervisors.

Lightweight Directory Access Protocol (LDAP)
: An application protocol for accessing and maintaining distributed directory information services over an IP network.

Linux bridge
: Software that enables multiple VMs to share a single physical NIC within Compute.

Linux Bridge neutron plug-in
: Enables a Linux bridge to understand a Networking port, interface attachment, and other abstractions.

Linux containers (LXC)
: An OpenStack-supported hypervisor.

live migration
: The ability within Compute to move running virtual machine instances from one host to another with only a small service interruption during switchover.

load balancer
: A load balancer is a logical device that belongs to a cloud account. It is used to distribute workloads between multiple back-end systems or services, based on the criteria defined as part of its configuration.

load balancing
: The process of spreading client requests between two or more nodes to improve performance and availability.

LBaaS
: Enables Networking to distribute incoming requests evenly between designated instances.

Logical Volume Manager (LVM)
: Provides a method of allocating space on mass-storage devices that is more flexible than conventional partitioning schemes.

M

magnum
: Code name for the OpenStack project that provides the Containers Service.

management API
: Alternative term for an admin API.

management network
: A network segment used for administration, not accessible to the public Internet.

manager
: Logical groupings of related code, such as the Block Storage volume manager or network manager.

manifest
: Used to track segments of a large object within Object Storage.

manifest object
: A special Object Storage object that contains the manifest for a large object.

manila
: OpenStack project that provides shared file systems as service to applications.

maximum transmission unit (MTU)
: Maximum frame or packet size for a particular network medium. Typically 1500 bytes for Ethernet networks.

mechanism driver
: A driver for the Modular Layer 2 (ML2) neutron plug-in that provides layer-2 connectivity for virtual instances. A single OpenStack installation can use multiple mechanism drivers.

melange
: Project name for OpenStack Network Information Service. To be merged with Networking.

membership
: The association between an Image service VM image and a tenant. Enables images to be shared with specified tenants.

membership list
: A list of tenants that can access a given VM image within Image service.

memcached
: A distributed memory object caching system that is used by Object Storage for caching.

memory overcommit
: The ability to start new VM instances based on the actual memory usage of a host, as opposed to basing the decision on the amount of RAM each running instance thinks it has available. Also known as RAM overcommit.

message broker
: The software package used to provide AMQP messaging capabilities within Compute. Default package is RabbitMQ.

message bus
: The main virtual communication line used by all AMQP messages for inter-cloud communications within Compute.

message queue
: Passes requests from clients to the appropriate workers and returns the output to the client after the job completes.

Message service
: OpenStack project that aims to produce an OpenStack messaging service that affords a variety of distributed application patterns in an efficient, scalable and highly-available manner, and to create and maintain associated Python libraries and documentation. The code name for the project is zaqar.

Metadata agent
: OpenStack Networking agent that provides metadata services for instances.

Meta-Data Server (MDS)
: Stores CephFS metadata.

migration
: The process of moving a VM instance from one host to another.

mistral

OpenStack project that provides the Workflow service.

Mitaka
: The code name for the thirteenth release of OpenStack. The design summit took place in Tokyo, Japan. Mitaka is a city in Tokyo.

monasca
: OpenStack project that provides a Monitoring service.

multi-host
: High-availability mode for legacy (nova) networking. Each compute node handles NAT and DHCP and acts as a gateway for all of the VMs on it. A networking failure on one compute node doesn't affect VMs on other compute nodes.

multinic
: Facility in Compute that allows each virtual machine instance to have more than one VIF connected to it.

murano
: OpenStack project that provides an Application catalog.

Modular Layer 2 (ML2) neutron plug-in
: Can concurrently use multiple layer-2 networking technologies, such as 802.1Q and VXLAN, in Networking.

Monitor (LBaaS)
: LBaaS feature that provides availability monitoring using the `ping` command, TCP, and HTTP/HTTPS GET.

Monitor (Mon)
: A Ceph component that communicates with external clients, checks data state and consistency, and performs quorum functions.

Monitoring
: The OpenStack project that provides a multi-tenant, highly scalable, performant, fault-tolerant Monitoring-as-a-Service solution for metrics, complex event processing, and logging. It builds an extensible platform for advanced monitoring services that can be used by both operators and tenants to gain operational insight and visibility, ensuring availability and stability. The project name is monasca.

multi-factor authentication
: Authentication method that uses two or more credentials, such as a password and a private key. Currently not supported in Identity.

MultiNic
: Facility in Compute that enables a virtual machine instance to have more than one VIF connected to it.

N

Nebula
: Released as open source by NASA in 2010 and is the basis for Compute.

netadmin
: One of the default roles in the Compute RBAC system. Enables the user to allocate publicly accessible IP addresses to instances and change firewall rules.

NetApp volume driver
: Enables Compute to communicate with NetApp storage devices through the NetApp OnCommand Provisioning Manager.

network
: A virtual network that provides connectivity between entities. For example, a collection of virtual ports that share network connectivity. In Networking terminology, a network is always a layer-2 network.

NAT
: Network Address Translation; Process of modifying IP address information while in transit. Supported by Compute and Networking.

network controller
: A Compute daemon that orchestrates the network configuration of nodes, including IP addresses, VLANs, and

bridging. Also manages routing for both public and private networks.

Network File System (NFS)
: A method for making file systems available over the network. Supported by OpenStack.

network ID
: Unique ID assigned to each network segment within Networking. Same as network UUID.

network manager
: The Compute component that manages various network components, such as firewall rules, IP address allocation, and so on.

network namespace
: Linux kernel feature that provides independent virtual networking instances on a single host with separate routing tables and interfaces. Similar to virtual routing and forwarding (VRF) services on physical network equipment.

network node
: Any compute node that runs the network worker daemon.

network segment
: Represents a virtual, isolated OSI layer-2 subnet in Networking.

Newton
: The code name for the fourteenth release of OpenStack. The design summit took place in Austin, Texas, US. The release is named after “Newton House” which is located at 1013 E. Ninth St., Austin, TX. which is listed on the National Register of Historic Places.

NTP
: Network Time Protocol; Method of keeping a clock for a host or node correct via communication with a trusted, accurate time source.

network UUID
: Unique ID for a Networking network segment.

network worker
: The `nova-network` worker daemon; provides services such as giving an IP address to a booting nova instance.

Networking service
: A core OpenStack project that provides a network connectivity abstraction layer to OpenStack Compute. The project name of Networking is neutron.

Networking API
: API used to access OpenStack Networking. Provides an extensible architecture to enable custom plug-in creation.

neutron
: A core OpenStack project that provides a network connectivity abstraction layer to OpenStack Compute.

neutron API
: An alternative name for Networking API.

neutron manager
: Enables Compute and Networking integration, which enables Networking to perform network management for guest VMs.

neutron plug-in
: Interface within Networking that enables organizations to create custom plug-ins for advanced features, such as QoS, ACLs, or IDS.

Nexenta volume driver
: Provides support for NexentaStor devices in Compute.

No ACK
: Disables server-side message acknowledgment in the Compute RabbitMQ. Increases performance but decreases reliability.

node
: A VM instance that runs on a host.

non-durable exchange
: Message exchange that is cleared when the service restarts. Its data is not written to persistent storage.

non-durable queue

Message queue that is cleared when the service restarts. Its data is not written to persistent storage.

non-persistent volume
: Alternative term for an ephemeral volume.

north-south traffic
: Network traffic between a user or client (north) and a server (south), or traffic into the cloud (south) and out of the cloud (north). See also east-west traffic.

nova
: OpenStack project that provides compute services.

Nova API
: Alternative term for the Compute API.

nova-network
: A Compute component that manages IP address allocation, firewalls, and other network-related tasks. This is the legacy networking option and an alternative to Networking.

O

object
: A BLOB of data held by Object Storage; can be in any format.

object auditor
: Opens all objects for an object server and verifies the MD5 hash, size, and metadata for each object.

object expiration
: A configurable option within Object Storage to automatically delete objects after a specified amount of time has passed or a certain date is reached.

object hash
: Uniquely ID for an Object Storage object.

object path hash
: Used by Object Storage to determine the location of an object in the ring. Maps objects to partitions.

object replicator
: An Object Storage component that copies an object to remote partitions for fault tolerance.

object server
: An Object Storage component that is responsible for managing objects.

Object Storage service
: The OpenStack core project that provides eventually consistent and redundant storage and retrieval of fixed digital content. The project name of OpenStack Object Storage is swift.

Object Storage API
: API used to access OpenStack Object Storage.

Object Storage Device (OSD)
: The Ceph storage daemon.

object versioning
: Allows a user to set a flag on an Object Storage container so that all objects within the container are versioned.

Ocata
: The code name for the fifteenth release of OpenStack. The design summit will take place in Barcelona, Spain. Ocata is a beach north of Barcelona.

Oldie
: Term for an Object Storage process that runs for a long time. Can indicate a hung process.

Open Cloud Computing Interface (OCCI)
: A standardized interface for managing compute, data, and network resources, currently unsupported in OpenStack.

Open Virtualization Format (OVF)
: Standard for packaging VM images. Supported in OpenStack.

Open vSwitch

Open vSwitch is a production quality, multilayer virtual switch licensed under the open source Apache 2.0 license. It is designed to enable massive network automation through programmatic extension, while still supporting standard management interfaces and protocols (for example NetFlow, sFlow, SPAN, RSPAN, CLI, LACP, 802.1ag).

Open vSwitch (OVS) agent
: Provides an interface to the underlying Open vSwitch service for the Networking plug-in.

Open vSwitch neutron plug-in
: Provides support for Open vSwitch in Networking.

OpenLDAP
: An open source LDAP server. Supported by both Compute and Identity.

OpenStack
: OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a data center, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface. OpenStack is an open source project licensed under the Apache License 2.0.

OpenStack code name
: Each OpenStack release has a code name. Code names ascend in alphabetical order: Austin, Bexar, Cactus, Diablo, Essex, Folsom, Grizzly, Havana, Icehouse, Juno, Kilo, Liberty, and Mitaka. Code names are cities or counties near where the corresponding OpenStack design summit took place. An exception, called the Waldon exception, is granted to elements of the state flag that sound especially cool. Code names are chosen by popular vote.

openSUSE
: A Linux distribution that is compatible with OpenStack.

operator
: The person responsible for planning and maintaining an OpenStack installation.

optional service
: An official OpenStack service defined as optional by DefCore Committee. Currently, consists of Dashboard (horizon), Telemetry service (Telemetry), Orchestration service (heat), Database service (trove), Bare Metal service (ironic), and so on.

Orchestration service
: An integrated project that orchestrates multiple cloud applications for OpenStack. The project name of Orchestration is heat.

orphan
: In the context of Object Storage, this is a process that is not terminated after an upgrade, restart, or reload of the service.

Oslo
: OpenStack project that produces a set of Python libraries containing code shared by OpenStack projects.

P

parent cell
: If a requested resource, such as CPU time, disk storage, or memory, is not available in the parent cell, the request is forwarded to associated child cells.

partition
: A unit of storage within Object Storage used to store objects. It exists on top of devices and is replicated for fault tolerance.

partition index
: Contains the locations of all Object Storage partitions within the ring.

partition shift value
: Used by Object Storage to determine which partition data should reside on.

path MTU discovery (PMTUD)
: Mechanism in IP networks to detect end-to-end MTU and adjust packet size accordingly.

pause
: A VM state where no changes occur (no changes in memory, network communications stop, etc); the VM is frozen but not shut down.

PCI passthrough
: Gives guest VMs exclusive access to a PCI device. Currently supported in OpenStack Havana and later releases.

persistent message
: A message that is stored both in memory and on disk. The message is not lost after a failure or restart.

persistent volume
: Changes to these types of disk volumes are saved.

personality file
: A file used to customize a Compute instance. It can be used to inject SSH keys or a specific network configuration.

Platform-as-a-Service (PaaS)
: Provides to the consumer the ability to deploy applications through a programming language or tools supported by the cloud platform provider. An example of Platform-as-a-Service is an Eclipse/Java programming platform provided with no downloads required.

plug-in
: Software component providing the actual implementation for Networking APIs, or for Compute APIs, depending on the context.

policy service
: Component of Identity that provides a rule-management interface and a rule-based authorization engine.

pool
: A logical set of devices, such as web servers, that you group together to receive and process traffic. The load balancing function chooses which member of the pool handles the new requests or connections received on the VIP address. Each VIP has one pool.

pool member
: An application that runs on the back-end server in a load-balancing system.

port
: A virtual network port within Networking; VIFs / vNICs are connected to a port.

port UUID
: Unique ID for a Networking port.

preseed
: A tool to automate system configuration and installation on Debian-based Linux distributions.

private image
: An Image service VM image that is only available to specified tenants.

private IP address
: An IP address used for management and administration, not available to the public Internet.

private network
: The Network Controller provides virtual networks to enable compute servers to interact with each other and with the public network. All machines must have a public and private network interface. A private network interface can be a flat or VLAN network interface. A flat network interface is controlled by the flat_interface with flat managers. A VLAN network interface is controlled by the `vlan_interface` option with VLAN managers.

project
: Projects represent the base unit of “ownership” in OpenStack, in that all resources in OpenStack should be owned by a specific project. In OpenStack Identity, a project must be owned by a specific domain.

project ID
: User-defined alphanumeric string in Compute; the name of a project.

project VPN
: Alternative term for a cloudpipe.

promiscuous mode
: Causes the network interface to pass all traffic it receives to the host rather than passing only the frames addressed to it.

protected property
: Generally, extra properties on an Image service image to which only cloud administrators have access. Limits which user roles can perform CRUD operations on that property. The cloud administrator can configure any image property as protected.

provider
: An administrator who has access to all hosts and instances.

proxy node
: A node that provides the Object Storage proxy service.

proxy server
: Users of Object Storage interact with the service through the proxy server, which in turn looks up the location of the requested data within the ring and returns the results to the user.

public API
: An API endpoint used for both service-to-service communication and end-user interactions.

public image
: An Image service VM image that is available to all tenants.

public IP address
: An IP address that is accessible to end-users.

public key authentication
: Authentication method that uses keys rather than passwords.

public network
: The Network Controller provides virtual networks to enable compute servers to interact with each other and with the public network. All machines must have a public and private network interface. The public network interface is controlled by the `public_interface` option.

Puppet
: An operating system configuration-management tool supported by OpenStack.

Python
: Programming language used extensively in OpenStack.

Q

QEMU Copy On Write 2 (QCOW2)
: One of the VM image disk formats supported by Image service.

Qpid
: Message queue software supported by OpenStack; an alternative to RabbitMQ.

quarantine
: If Object Storage finds objects, containers, or accounts that are corrupt, they are placed in this state, are not replicated, cannot be read by clients, and a correct copy is re-replicated.

Quick EMUlator (QEMU)
: QEMU is a generic and open source machine emulator and virtualizer. One of the hypervisors supported by OpenStack, generally used for development purposes.

quota
: In Compute and Block Storage, the ability to set resource limits on a per-project basis.

R

RabbitMQ
: The default message queue software used by OpenStack.

Rackspace Cloud Files
: Released as open source by Rackspace in 2010; the basis for Object Storage.

RADOS Block Device (RBD)
: Ceph component that enables a Linux block device to be striped over multiple distributed data stores.

radvd
: The router advertisement daemon, used by the Compute VLAN manager and FlatDHCP manager to provide routing services for VM instances.

rally
: OpenStack project that provides the Benchmark service.

RAM filter
: The Compute setting that enables or disables RAM overcommitment.

RAM overcommit
: The ability to start new VM instances based on the actual memory usage of a host, as opposed to basing the decision on the amount of RAM each running instance thinks it has available. Also known as memory overcommit.

rate limit
: Configurable option within Object Storage to limit database writes on a per-account and/or per-container basis.

raw
: One of the VM image disk formats supported by Image service; an unstructured disk image.

rebalance
: The process of distributing Object Storage partitions across all drives in the ring; used during initial ring creation and after ring reconfiguration.

reboot
: Either a soft or hard reboot of a server. With a soft reboot, the operating system is signaled to restart, which enables a graceful shutdown of all processes. A hard reboot is the equivalent of power cycling the server. The virtualization platform should ensure that the reboot action has completed successfully, even in cases in which the underlying domain/VM is paused or halted/stopped.

rebuild
: Removes all data on the server and replaces it with the specified image. Server ID and IP addresses remain the same.

Recon
: An Object Storage component that collects meters.

record
: Belongs to a particular domain and is used to specify information about the domain. There are several types of DNS records. Each record type contains particular information used to describe the purpose of that record. Examples include mail exchange (MX) records, which specify the mail server for a particular domain; and name server (NS) records, which specify the authoritative name servers for a domain.

record ID
: A number within a database that is incremented each time a change is made. Used by Object Storage when replicating.

Red Hat Enterprise Linux (RHEL)
: A Linux distribution that is compatible with OpenStack.

reference architecture
: A recommended architecture for an OpenStack cloud.

region
: A discrete OpenStack environment with dedicated API endpoints that typically shares only the Identity (keystone) with other regions.

registry
: Alternative term for the Image service registry.

registry server
: An Image service that provides VM image metadata information to clients.

Reliable, Autonomic Distributed Object Store
: (RADOS)

A collection of components that provides object storage within Ceph. Similar to OpenStack Object Storage.

Remote Procedure Call (RPC)
: The method used by the Compute RabbitMQ for intra-service communications.

replica
: Provides data redundancy and fault tolerance by creating copies of Object Storage objects, accounts, and containers so that they are not lost when the underlying storage fails.

replica count
: The number of replicas of the data in an Object Storage ring.

replication
: The process of copying data to a separate physical device for fault tolerance and performance.

replicator
: The Object Storage back-end process that creates and manages object replicas.

request ID
: Unique ID assigned to each request sent to Compute.

rescue image
: A special type of VM image that is booted when an instance is placed into rescue mode. Allows an administrator to mount the file systems for an instance to correct the problem.

resize
: Converts an existing server to a different flavor, which scales the server up or down. The original server is saved to enable rollback if a problem occurs. All resizes must be tested and explicitly confirmed, at which time the original server is removed.

RESTful
: A kind of web service API that uses REST, or Representational State Transfer. REST is the style of architecture for hypermedia systems that is used for the World Wide Web.

ring
: An entity that maps Object Storage data to partitions. A separate ring exists for each service, such as account, object, and container.

ring builder
: Builds and manages rings within Object Storage, assigns partitions to devices, and pushes the configuration to other storage nodes.

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. Supported in both Identity and Compute and can be configured using the horizon dashboard.

role
: A personality that a user assumes to perform a specific set of operations. A role includes a set of rights and privileges. A user assuming that role inherits those rights and privileges.

role ID
: Alphanumeric ID assigned to each Identity service role.

rootwrap
: A feature of Compute that allows the unprivileged “nova” user to run a specified list of commands as the Linux root user.

round-robin scheduler
: Type of Compute scheduler that evenly distributes instances among available hosts.

router
: A physical or virtual network device that passes network traffic between different networks.

routing key
: The Compute direct exchanges, fanout exchanges, and topic exchanges use this key to determine how to process a message; processing varies depending on exchange type.

RPC driver
: Modular system that allows the underlying message queue software of Compute to be changed. For example, from

RabbitMQ to ZeroMQ or Qpid.

rsync
: Used by Object Storage to push object replicas.

RXTX cap
: Absolute limit on the amount of network traffic a Compute VM instance can send and receive.

RXTX quota
: Soft limit on the amount of network traffic a Compute VM instance can send and receive.

S

S3
: Object storage service by Amazon; similar in function to Object Storage, it can act as a back-end store for Image service VM images.

sahara
: OpenStack project that provides a scalable data-processing stack and associated management interfaces.

SAML assertion
: Contains information about a user as provided by the identity provider. It is an indication that a user has been authenticated.

scheduler manager
: A Compute component that determines where VM instances should start. Uses modular design to support a variety of scheduler types.

scoped token
: An Identity service API access token that is associated with a specific tenant.

scrubber
: Checks for and deletes unused VMs; the component of Image service that implements delayed delete.

secret key
: String of text known only by the user; used along with an access key to make requests to the Compute API.

secure shell (SSH)
: Open source tool used to access remote hosts through an encrypted communications channel, SSH key injection is supported by Compute.

security group
: A set of network traffic filtering rules that are applied to a Compute instance.

segmented object
: An Object Storage large object that has been broken up into pieces. The re-assembled object is called a concatenated object.

self-service
: For IaaS, ability for a regular (non-privileged) account to manage a virtual infrastructure component such as networks without involving an administrator.

SELinux
: Linux kernel security module that provides the mechanism for supporting access control policies.

senlin
: OpenStack project that provides a Clustering service.

server
: Computer that provides explicit services to the client software running on that system, often managing a variety of computer operations. A server is a VM instance in the Compute system. Flavor and image are requisite elements when creating a server.

server image
: Alternative term for a VM image.

server UUID
: Unique ID assigned to each guest VM instance.

service
: An OpenStack service, such as Compute, Object Storage, or Image service. Provides one or more endpoints through which users can access resources and perform operations.

service catalog
: Alternative term for the Identity service catalog.

service ID
: Unique ID assigned to each service that is available in the Identity service catalog.

service provider
: A system that provides services to other system entities. In case of federated identity, OpenStack Identity is the service provider.

service registration
: An Identity service feature that enables services, such as Compute, to automatically register with the catalog.

service tenant
: Special tenant that contains all services that are listed in the catalog.

service token
: An administrator-defined token used by Compute to communicate securely with the Identity service.

session back end
: The method of storage used by horizon to track client sessions, such as local memory, cookies, a database, or memcached.

session persistence
: A feature of the load-balancing service. It attempts to force subsequent connections to a service to be redirected to the same node as long as it is online.

session storage
: A horizon component that stores and tracks client session information. Implemented through the Django sessions framework.

share
: A remote, mountable file system in the context of the Shared File Systems. You can mount a share to, and access a share from, several hosts by several users at a time.

share network
: An entity in the context of the Shared File Systems that encapsulates interaction with the Networking service. If the driver you selected runs in the mode requiring such kind of interaction, you need to specify the share network to create a share.

Shared File Systems API
: A Shared File Systems service that provides a stable RESTful API. The service authenticates and routes requests throughout the Shared File Systems service. There is python-manilaclient to interact with the API.

Shared File Systems service
: An OpenStack service that provides a set of services for management of shared file systems in a multi-tenant cloud environment. The service is similar to how OpenStack provides block-based storage management through the OpenStack Block Storage service project. With the Shared File Systems service, you can create a remote file system and mount the file system on your instances. You can also read and write data from your instances to and from your file system. The project name of the Shared File Systems service is manila.

shared IP address
: An IP address that can be assigned to a VM instance within the shared IP group. Public IP addresses can be shared across multiple servers for use in various high-availability scenarios. When an IP address is shared to another server, the cloud network restrictions are modified to enable each server to listen to and respond on that IP address. You can optionally specify that the target server network configuration be modified. Shared IP addresses can be used with many standard heartbeat facilities, such as keepalive, that monitor for failure and manage IP failover.

shared IP group
: A collection of servers that can share IPs with other members of the group. Any server in a group can share one or more public IPs with any other server in the group. With the exception of the first server in a shared IP group, servers must be launched into shared IP groups. A server may be a member of only one shared IP group.

shared storage
: Block storage that is simultaneously accessible by multiple clients, for example, NFS.

Sheepdog
: Distributed block storage system for QEMU, supported by OpenStack.

Simple Cloud Identity Management (SCIM)
: Specification for managing identity in the cloud, currently unsupported by OpenStack.

Single-root I/O Virtualization (SR-IOV)
: A specification that, when implemented by a physical PCIe device, enables it to appear as multiple separate PCIe devices. This enables multiple virtualized guests to share direct access to the physical device, offering improved performance over an equivalent virtual device. Currently supported in OpenStack Havana and later releases.

Service Level Agreement (SLA)
: Contractual obligations that ensure the availability of a service.

SmokeStack
: Runs automated tests against the core OpenStack API; written in Rails.

snapshot
: A point-in-time copy of an OpenStack storage volume or image. Use storage volume snapshots to back up volumes. Use image snapshots to back up data, or as “gold” images for additional servers.

soft reboot
: A controlled reboot where a VM instance is properly restarted through operating system commands.

Software Development Lifecycle Automation service
: OpenStack project that aims to make cloud services easier to consume and integrate with application development process by automating the source-to-image process, and simplifying app-centric deployment. The project name is solum.

SolidFire Volume Driver
: The Block Storage driver for the SolidFire iSCSI storage appliance.

solum
: OpenStack project that provides a Software Development Lifecycle Automation service.

SPICE
: The Simple Protocol for Independent Computing Environments (SPICE) provides remote desktop access to guest virtual machines. It is an alternative to VNC. SPICE is supported by OpenStack.

spread-first scheduler
: The Compute VM scheduling algorithm that attempts to start a new VM on the host with the least amount of load.

SQL-Alchemy
: An open source SQL toolkit for Python, used in OpenStack.

SQLite
: A lightweight SQL database, used as the default persistent storage method in many OpenStack services.

stack
: A set of OpenStack resources created and managed by the Orchestration service according to a given template (either an AWS CloudFormation template or a Heat Orchestration Template (HOT)).

StackTach
: Community project that captures Compute AMQP communications; useful for debugging.

static IP address
: Alternative term for a fixed IP address.

StaticWeb
: WSGI middleware component of Object Storage that serves container data as a static web page.

storage back end
: The method that a service uses for persistent storage, such as iSCSI, NFS, or local disk.

storage node
: An Object Storage node that provides container services, account services, and object services; controls the account databases, container databases, and object storage.

storage manager

A XenAPI component that provides a pluggable interface to support a wide variety of persistent storage back ends.

storage manager back end
: A persistent storage method supported by XenAPI, such as iSCSI or NFS.

storage services
: Collective name for the Object Storage object services, container services, and account services.

strategy
: Specifies the authentication source used by Image service or Identity. In the Database service, it refers to the extensions implemented for a data store.

subdomain
: A domain within a parent domain. Subdomains cannot be registered. Subdomains enable you to delegate domains. Subdomains can themselves have subdomains, so third-level, fourth-level, fifth-level, and deeper levels of nesting are possible.

subnet
: Logical subdivision of an IP network.

SUSE Linux Enterprise Server (SLES)
: A Linux distribution that is compatible with OpenStack.

suspend
: Alternative term for a paused VM instance.

swap
: Disk-based virtual memory used by operating systems to provide more memory than is actually available on the system.

swauth
: An authentication and authorization service for Object Storage, implemented through WSGI middleware; uses Object Storage itself as the persistent backing store.

swift
: An OpenStack core project that provides object storage services.

swift All in One (SAIO)
: Creates a full Object Storage development environment within a single VM.

swift middleware
: Collective term for Object Storage components that provide additional functionality.

swift proxy server
: Acts as the gatekeeper to Object Storage and is responsible for authenticating the user.

swift storage node
: A node that runs Object Storage account, container, and object services.

sync point
: Point in time since the last container and accounts database sync among nodes within Object Storage.

sysadmin
: One of the default roles in the Compute RBAC system. Enables a user to add other users to a project, interact with VM images that are associated with the project, and start and stop VM instances.

system usage
: A Compute component that, along with the notification system, collects meters and usage information. This information can be used for billing.

T

Telemetry service
: An integrated project that provides metering and measuring facilities for OpenStack. The project name of Telemetry is ceilometer.

TempAuth
: An authentication facility within Object Storage that enables Object Storage itself to perform authentication and

authorization. Frequently used in testing and development.

Tempest
: Automated software test suite designed to run against the trunk of the OpenStack core project.

TempURL
: An Object Storage middleware component that enables creation of URLs for temporary object access.

tenant
: A group of users; used to isolate access to Compute resources. An alternative term for a project.

Tenant API
: An API that is accessible to tenants.

tenant endpoint
: An Identity service API endpoint that is associated with one or more tenants.

tenant ID
: Unique ID assigned to each tenant within the Identity service. The project IDs map to the tenant IDs.

token
: An alpha-numeric string of text used to access OpenStack APIs and resources.

token services
: An Identity service component that manages and validates tokens after a user or tenant has been authenticated.

tombstone
: Used to mark Object Storage objects that have been deleted; ensures that the object is not updated on another node after it has been deleted.

topic publisher
: A process that is created when a RPC call is executed; used to push the message to the topic exchange.

Torpedo
: Community project used to run automated tests against the OpenStack API.

transaction ID
: Unique ID assigned to each Object Storage request; used for debugging and tracing.

transient
: Alternative term for non-durable.

transient exchange
: Alternative term for a non-durable exchange.

transient message
: A message that is stored in memory and is lost after the server is restarted.

transient queue
: Alternative term for a non-durable queue.

TripleO
: OpenStack-on-OpenStack program. The code name for the OpenStack Deployment program.

trove
: OpenStack project that provides database services to applications.

U

Ubuntu
: A Debian-based Linux distribution.

unscoped token
: Alternative term for an Identity service default token.

updater
: Collective term for a group of Object Storage components that processes queued and failed updates for containers and objects.

user
: In OpenStack Identity, entities represent individual API consumers and are owned by a specific domain. In

OpenStack Compute, a user can be associated with roles, projects, or both.

user data
: A blob of data that the user can specify when they launch an instance. The instance can access this data through the metadata service or config drive. Commonly used to pass a shell script that the instance runs on boot.

User Mode Linux (UML)
: An OpenStack-supported hypervisor.

V

VIF UUID
: Unique ID assigned to each Networking VIF.

VIP
: The primary load balancing configuration object. Specifies the virtual IP address and port where client traffic is received. Also defines other details such as the load balancing method to be used, protocol, and so on. This entity is sometimes known in load-balancing products as a virtual server, vserver, or listener.

Virtual Central Processing Unit (vCPU)
: Subdivides physical CPUs. Instances can then use those divisions.

Virtual Disk Image (VDI)
: One of the VM image disk formats supported by Image service.

VXLAN
: A network virtualization technology that attempts to reduce the scalability problems associated with large cloud computing deployments. It uses a VLAN-like encapsulation technique to encapsulate Ethernet frames within UDP packets.

Virtual Hard Disk (VHD)
: One of the VM image disk formats supported by Image service.

virtual IP
: An Internet Protocol (IP) address configured on the load balancer for use by clients connecting to a service that is load balanced. Incoming connections are distributed to back-end nodes based on the configuration of the load balancer.

virtual machine (VM)
: An operating system instance that runs on top of a hypervisor. Multiple VMs can run at the same time on the same physical host.

virtual network
: An L2 network segment within Networking.

virtual networking
: A generic term for virtualization of network functions such as switching, routing, load balancing, and security using a combination of VMs and overlays on physical network infrastructure.

Virtual Network Computing (VNC)
: Open source GUI and CLI tools used for remote console access to VMs. Supported by Compute.

Virtual Network InterFace (VIF)
: An interface that is plugged into a port in a Networking network. Typically a virtual network interface belonging to a VM.

virtual port
: Attachment point where a virtual interface connects to a virtual network.

virtual private network (VPN)
: Provided by Compute in the form of cloudpipes, specialized instances that are used to create VPNs on a per-project basis.

virtual server
: Alternative term for a VM or guest.

virtual switch (vSwitch)

Software that runs on a host or node and provides the features and functions of a hardware-based network switch.

virtual VLAN
: Alternative term for a virtual network.

VirtualBox
: An OpenStack-supported hypervisor.

VLAN manager
: A Compute component that provides dnsmasq and radvd and sets up forwarding to and from cloudpipe instances.

VLAN network
: The Network Controller provides virtual networks to enable compute servers to interact with each other and with the public network. All machines must have a public and private network interface. A VLAN network is a private network interface, which is controlled by the `vlan_interface` option with VLAN managers.

VM disk (VMDK)
: One of the VM image disk formats supported by Image service.

VM image
: Alternative term for an image.

VM Remote Control (VMRC)
: Method to access VM instance consoles using a web browser. Supported by Compute.

VMware API
: Supports interaction with VMware products in Compute.

VMware NSX Neutron plug-in
: Provides support for VMware NSX in Neutron.

VNC proxy
: A Compute component that provides users access to the consoles of their VM instances through VNC or VMRC.

volume
: Disk-based data storage generally represented as an iSCSI target with a file system that supports extended attributes; can be persistent or ephemeral.

Volume API
: Alternative name for the Block Storage API.

volume controller
: A Block Storage component that oversees and coordinates storage volume actions.

volume driver
: Alternative term for a volume plug-in.

volume ID
: Unique ID applied to each storage volume under the Block Storage control.

volume manager
: A Block Storage component that creates, attaches, and detaches persistent storage volumes.

volume node
: A Block Storage node that runs the cinder-volume daemon.

volume plug-in
: Provides support for new and specialized types of back-end storage for the Block Storage volume manager.

volume worker
: A cinder component that interacts with back-end storage to manage the creation and deletion of volumes and the creation of compute volumes, provided by the cinder-volume daemon.

vSphere
: An OpenStack-supported hypervisor.

W

weighting
: A Compute process that determines the suitability of the VM instances for a job for a particular host. For example, not

enough RAM on the host, too many CPUs on the host, and so on.

weight
: Used by Object Storage devices to determine which storage devices are suitable for the job. Devices are weighted by size.

weighted cost
: The sum of each cost used when deciding where to start a new VM instance in Compute.

worker
: A daemon that listens to a queue and carries out tasks in response to messages. For example, the cinder-volume worker manages volume creation and deletion on storage arrays.

Workflow service
: OpenStack project that provides a simple YAML-based language to write workflows, tasks and transition rules, and a service that allows to upload them, modify, run them at scale and in a highly available manner, manage and monitor workflow execution state and state of individual tasks. The code name of the project is mistral.

X

Xen
: Xen is a hypervisor using a microkernel design, providing services that allow multiple computer operating systems to execute on the same computer hardware concurrently.

Xen API
: The Xen administrative API, which is supported by Compute.

Xen Cloud Platform (XCP)
: An OpenStack-supported hypervisor.

Xen Storage Manager Volume Driver
: A Block Storage volume plug-in that enables communication with the Xen Storage Manager API.

XenServer
: An OpenStack-supported hypervisor.

XFS
: High-performance 64-bit file system created by Silicon Graphics. Excels in parallel I/O operations and data consistency.

Z

zaqar
: OpenStack project that provides a message service to applications.

ZeroMQ
: Message queue software supported by OpenStack. An alternative to RabbitMQ. Also spelled 0MQ.

Zuul
: Tool used in OpenStack development to ensure correctly ordered testing of changes in parallel.

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary
  - 0-9
  - A
  - B
  - C
  - D
  - E
  - F
  - G
  - H
  - I
  - J
  - K
  - L
  - M
  - N
  - O

- P
- Q
- R
- S
- T
- U
- V
- W
- X
- Z

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack

SEARCH

Search

UPDATED: 2016-08-13 23:58

Search

Please activate JavaScript to enable the search functionality.

From here you can search these documents. Enter your search words into the box below and click "search". Note that the search function will automatically search for all of the words. Pages containing fewer words won't appear in the result list.

search

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

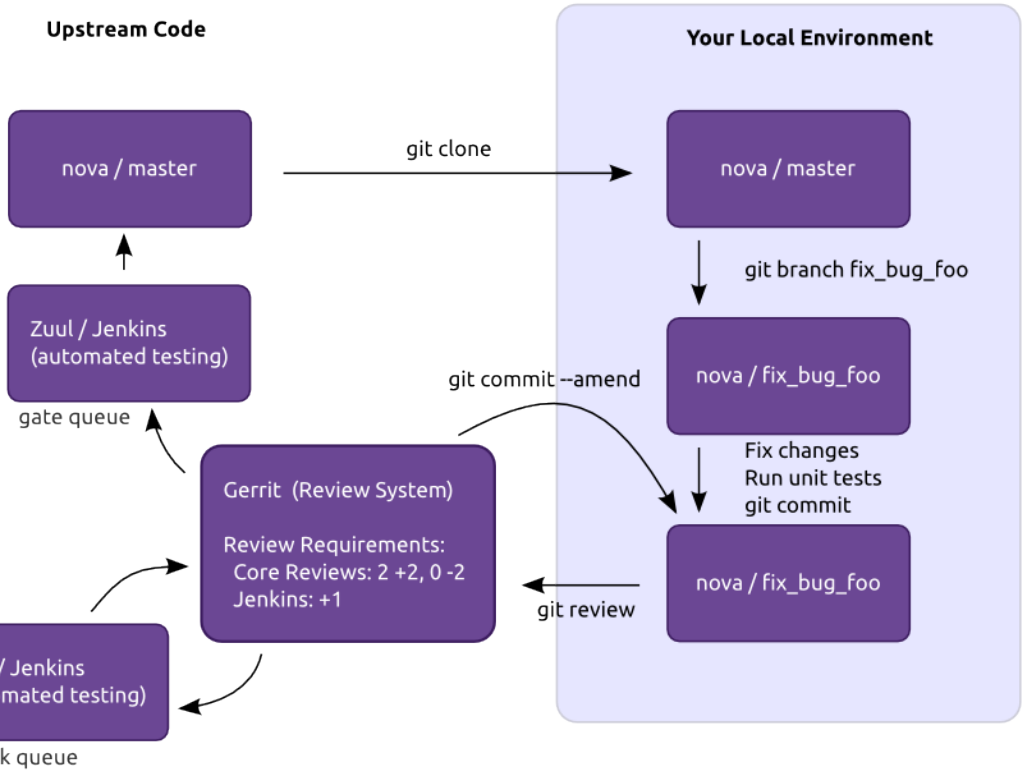
The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

Table Of Contents

Developer's Guide
- Quick Reference
- Getting Started
  - Account Setup
    - Sign the appropriate individual contributor License Agreement
    - Add an SSH key to your Gerrit account
    - Install git-review
  - Cloning
  - Making Changes
  - Submitting Changes
  - Project Bots
  - Working Collaboratively
  - Reviewing
  - Recognizing Work
  - Advanced Git
  - Advanced Gerrit
  - Tracking Changes
  - Testing
  - Debugging
  - Publishing
  - Publishing Documentation
  - Publishing Packages
  - Publishing Artifacts
  - Publishing Logs
  - Publishing Translations
  - Publishing Stable Branches
  - Publishing Binaries
  - Publishing Images
  - Publishing Containers
  - Publishing Wheels
  - Publishing Eggs
  - Publishing Jars
  - Publishing Gems
  - Publishing Charts
  - Publishing Modules
  - Publishing Plugins
  - Publishing Extensions
  - Publishing Drivers
  - Publishing Libraries
  - Publishing Clients
  - Publishing Servers
  - Publishing Agents
  - Publishing Daemons
  - Publishing Services
  - Publishing Tools
  - Publishing Scripts
  - Publishing Configs
  - Publishing Templates
  - Publishing Examples
  - Publishing Tests
  - Publishing Benchmarks
  - Publishing Fixtures
  - Publishing Mocks
  - Publishing Stubs
  - Publishing Fakes
  - Publishing Doubles
  - Publishing Spies
  - Publishing Dummies
  - Publishing Placeholders
  - Publishing Samples
  - Publishing Demos
  - Publishing Prototypes
  - Publishing Proofs of Concept
  - Publishing Experiments
  - Publishing Research
  - Publishing Papers
  - Publishing Theses
  - Publishing Dissertations
  - Publishing Books
  - Publishing Articles
  - Publishing Blogs
  - Publishing Wikis
  - Publishing Forums
  - Publishing Mailing Lists
  - Publishing Chats
  - Publishing Videos
  - Publishing Podcasts
  - Publishing Webinars
  - Publishing Courses
  - Publishing Tutorials
  - Publishing Workshops
  - Publishing Conferences
  - Publishing Meetups
  - Publishing Hackathons
  - Publishing Sprints
  - Publishing Retrospectives
  - Publishing Postmortems
  - Publishing Incidents
  - Publishing Outages
  - Publishing Maintenance
  - Publishing Upgrades
  - Publishing Migrations
  - Publishing Backups
  - Publishing Restores
  - Publishing Disaster Recovery
  - Publishing Business Continuity
  - Publishing Security
  - Publishing Compliance
  - Publishing Audits
  - Publishing Reports
  - Publishing Metrics
  - Publishing Dashboards
  - Publishing Alerts
  - Publishing Notifications
  - Publishing Logs
  - Publishing Traces
  - Publishing Profiles
  - Publishing Debugging
  - Publishing Troubleshooting
  - Publishing Support
  - Publishing Documentation
  - Publishing Training
  - Publishing Certification
  - Publishing Licensing
  - Publishing Trademarks
  - Publishing Branding
  - Publishing Marketing
  - Publishing Sales
  - Publishing Partnerships
  - Publishing Alliances
  - Publishing Ecosystem
  - Publishing Community
  - Publishing Governance
  - Publishing Policies
  - Publishing Procedures
  - Publishing Guidelines
  - Publishing Standards
  - Publishing Specifications
  - Publishing Requirements
  - Publishing Roadmaps
  - Publishing Strategies
  - Publishing Plans
  - Publishing Goals
  - Publishing Objectives
  - Publishing Key Results
  - Publishing Metrics
  - Publishing Dashboards
  - Publishing Alerts
  - Publishing Notifications
  - Publishing Logs
  - Publishing Traces
  - Publishing Profiles
  - Publishing Debugging
  - Publishing Troubleshooting
  - Publishing Support
  - Publishing Documentation
  - Publishing Training
  - Publishing Certification
  - Publishing Licensing
  - Publishing Trademarks
  - Publishing Branding
  - Publishing Marketing
  - Publishing Sales
  - Publishing Partnerships
  - Publishing Alliances
  - Publishing Ecosystem
  - Publishing Community
  - Publishing Governance
  - Publishing Policies
  - Publishing Procedures
  - Publishing Guidelines
  - Publishing Standards
  - Publishing Specifications
  - Publishing Requirements
  - Publishing Roadmaps
  - Publishing Strategies
  - Publishing Plans
  - Publishing Goals
  - Publishing Objectives
  - Publishing Key Results

Developer's Guide

Quick Reference



Getting Started

The goal of this document is to walk you through the concepts and specifics that should be understood while contributing to OpenStack.

Prior to contributing to an OpenStack source code repository a few steps need to be completed. This document covers the steps that get you started, such as creating a few accounts on required websites, signing a contributor license agreement, uploading an ssh key, and installing git-review.

Account Setup

You'll need a Launchpad account, since this is how the Web interface for the Gerrit Code Review system will identify you. This is also useful for automatically crediting bug fixes to you when you address them with your code commits.

If you haven't already, join The OpenStack Foundation (it's free and currently required for all code contributors, though there is work in progress which may remove this requirement in the future). Your member level needs to be *Foundation Member* (not Community Member). Among other privileges, this also allows you to vote in elections and run for elected positions within The OpenStack Project. When signing up for Foundation Membership, make sure to give the same email address you'll use for code contributions, since this will need to match your preferred email address in Gerrit.

- Installing git-review
- Starting Work on a New Project
- Accessing Gerrit over HTTPS
- Development Workflow
  - Working on Bugs
  - Working on Specifications and Blueprints
  - Starting a Change
  - Committing a Change
  - Using Signed-off-by
  - Running

Visit https://review.openstack.org/ and click the `Sign In` link at the top-right corner of the page. Log in with your Launchpad ID.

The first time you sign into OpenStack's Gerrit (review.openstack.org), you will be prompted to "Select a unique username:". You can enter your Launchpad username here, or something else if you want. Type carefully, as once set it cannot be changed. This is the username you will eventually use to submit changes to Gerrit and to perform authenticated queries through its API.

Because Gerrit uses Launchpad OpenID single sign-on, you won't need a separate password for Gerrit, and once you log in to one of Launchpad, Gerrit, or any number of other OpenStack services, you won't have to enter your password for the others.

Sign the appropriate Individual Contributor License Agreement

Note

You need to have completed all the steps above before you'll be able to sign an Individual Contributor License Agreement.

Unless you are an U.S. Government Employee (see below), agree to the Individual Contributor License Agreement and provide contact information. The full text of the agreement will be displayed before you can enter "I AGREE" below it, but it's also anonymously available if you want to preview the OpenStack ICLA now. Your full name and email address will be public (since they also appear in repository commit logs) and the latter needs to match the user.email in your Git configuration. The other contact information (postal address, phone numbers) will be kept confidential and is only used as a fallback record in the unlikely event The OpenStack Foundation needs to reach you directly over code contribution related matters. This contact information can also be easily updated later if desired, but make sure the primary email address always matches the one you set for your OpenStack Foundation Membership – otherwise Gerrit will give you an error message and refuse to accept your contact information.

Employees of the U.S. Government do not sign the Individual CLA. Instead, someone with authority to sign on behalf of your agency should sign the U.S. Government Contributor License Agreement. Please contact the OpenStack Foundation to initiate this process.

If you are contributing on behalf of a company or organization, you still need to sign the ICLA above but someone at your company or organization also needs to sign the Corporate Contributor License Agreement providing a list of people authorized to commit code to OpenStack. If you need a printable copy of the Corporate CLA please email community managers. Check How to update the CCLA to provide changes to such list. A list of current companies and organizations with an existing Corporate CLA is available for your review.

You'll also want to upload an SSH key to Gerrit at review.openstack.org while you're at it, so that you'll be able to commit changes for review later. This is different from adding a key to Launchpad.

Ensure that you have run these steps to let git know about your email address:

```
git config --global user.name "Firstname Lastname"
git config --global user.email "your email@youremail.com"
```

To check your git configuration:

```
git config --list
```

Installing git-review

We recommend using the `git-review` tool which is a git subcommand that handles all the details of working with Gerrit, the code review system used in OpenStack development. Before you start work, make sure you have git-review installed on your system.

On Ubuntu Precise (12.04) and later, git-review is included in the distribution, so install it as any other package:

- g Unit Tests
- Previewing a Change
- Submitting a Change for Review
- Updating a Change
- Understanding Changes and Patchsets
- Squashing Changes
- Adding a Dependency
- Cross-Repository Dependenci


```
apt-get install git-review
```

On Fedora 16 and later, git-review is included into the distribution, so install it as any other package:

```
yum install git-review
```

On Red Hat Enterprise Linux, you must first enable the EPEL repository, then install it as any other package:

```
yum install git-review
```

On openSUSE 12.2 and later, git-review is included in the distribution under the name python-git-review, so install it as any other package:

```
zypper in python-git-review
```

On Mac OS X, or most other Unix-like systems, you may install it with pip:

```
pip install git-review
```

If you run into trouble, you can refer to the git-review readme file.

All of git-review's interactions with Gerrit are sequences of normal git commands. If you want to know more about what it's doing, just add -v to the options and it will print out all of the commands it's running.

Starting Work on a New Project

Clone a repository in the usual way, for example:

```
git clone https://git.openstack.org/openstack/<projectname>.git
```

You may want to ask git-review to configure your repository to know about Gerrit at this point. If you don't, it will do so the first time you submit a change for review, but you probably want to do this ahead of time so the Gerrit Change-Id commit hook gets installed. To do so:

```
cd <projectname>
git review -s
```

Git-review checks that you can log in to Gerrit with your ssh key. It assumes that your Gerrit/Launchpad user name is the same as the current running user. If that doesn't work, it asks for your Gerrit/Launchpad user name. You can avoid that question by configuring git to use your Gerrit username, as follows:

```
git config --global gitreview.username yourgerritusername
```

If you don't remember your Gerrit user name go to the settings page on gerrit to check it out (it's not your email address).

> **Note**
>
> You can verify the SSH host keys for review.openstack.org on https://review.openstack.org/#/settings/ssh-keys

We have a tutorial: *Learn the Gerrit Workflow in the Sandbox*. If this is your first time contributing to OpenStack, we strongly suggest you follow this tutorial.

Accessing Gerrit over HTTPS

Git-review normally communicates with Gerrit using SSH over port 29418 with no further configuration needed. However, if you suspect that ssh over non-standards ports might be blocked (or you need to access the web using https) then you can configure git-review to use an https endpoint instead of ssh. Keep in mind that you will need to generate an HTTP password in Gerrit to use this connection. You should run the following command before "git review -s":

es
- G
  - a
  - t
  - e
  - R
  - i
  - p
  - e
  - l
  - i
  - n
  - e
- C
  - h
  - e
  - c
  - k
  - P
  - i
  - p
  - e
  - l
  - i
  - n
  - e
- M
  - u
  - l
  - t
  - i
  - p
  - l
  - e
  - C
  - h
  - a
  - n
  - g
  - e
  - s
- C
  - v
  - i
  - e
  - s
- Li
  - m
  - i
  - t
  - a
  - t
  - i
  - o
  - n
  - s
  - a
  - n
  - d
  - C
  - a
  - v
  - e
  - a
  - t
  - s
- Code Review
  - Autom

```
git remote add gerrit https://<username>@review.openstack.org/<umbrella repository name>/<repository name>.git
```

In case you had already tried to setup git-review and it failed, it might be necessary to remove the Gerrit remote from git:

```
git remote rm gerrit
```

Development Workflow

Working on Bugs

Bug reports for a project are generally tracked on Launchpad at https://bugs.launchpad.net/<projectname>. Contributors may review these reports regularly when looking for work to complete.

There are 4 key tasks with regards to bugs that anyone can do:

1. Confirm new bugs: When a bug is filed, it is set to the "New" status. A "New" bug can be marked "Confirmed" once it has been reproduced and is thus confirmed as genuine.
2. Solve inconsistencies: Make sure bugs are Confirmed, and if assigned that they are marked "In Progress"
3. Review incomplete bugs: See if information that caused them to be marked "Incomplete" has been provided, determine if more information is required and provide reminders to the bug reporter if they haven't responded after 2-4 weeks.
4. Review stale In Progress bugs: Work with assignee of bugs to determine if the bug is still being worked on, if not, unassign them and mark them back to Confirmed or Triaged.

Learn more about working with bugs for various projects at:

https://wiki.openstack.org/wiki/BugTriage

Bug statuses are documented here:

https://wiki.openstack.org/wiki/Bugs

If you find a bug that you wish to work on, you may assign it to yourself. When you upload a review, include the bug in the commit message for automatic updates back to Launchpad. The following options are available:

```
Closes-Bug: #######
Partial-Bug: #######
Related-Bug: #######
```

Also see the Including external references section of the OpenStack Git Commit Good Practices wiki page.

Working on Specifications and Blueprints

Many OpenStack project teams have a <projectteam>-specs repository which is used to hold approved design specifications for additions and changes to the project team's code repositories.

The layout of the repository will typically be something like:

```
specs/<release>/
```

It may also have subdirectories to make clear which specifications are approved and which have already been implemented:

specs/<release>/approved specs/<release>/implemented

You can typically find an example spec in `specs/template.rst`.

ated Testing
- Peer Review
- Work in Progress
- Merging
- Project Gating

Previous topic

Infrastructure User Manual

Next topic

Python Developer's Guide

Project Source

Project Source

This Page

Show Source

Quick search

Go

Enter search terms or a module, class or function name.

Check the repository for the project team you're working on for specifics about repository organization.

Specifications are proposed for a given release by adding them to the `specs/<release>` directory and posting it for review. The implementation status of a blueprint for a given release can be found by looking at the blueprint in Launchpad. Not all approved blueprints will get fully implemented.

Specifications have to be re-proposed for every release. The review may be quick, but even if something was previously approved, it should be re-reviewed to make sure it still makes sense as written.

Historically, Launchpad blueprints were used to track the implementation of these significant features and changes in OpenStack. For many project teams, these Launchpad blueprints are still used for tracking the current status of a specification. For more information, see the Blueprints wiki page.

View all approved project team's specifications at http://specs.openstack.org/.

Starting a Change

Once your local repository is set up as above, you must use the following workflow.

Make sure you have the latest upstream changes:

```
git remote update
git checkout master
git pull --ff-only origin master
```

Create a topic branch to hold your work and switch to it. If you are working on a blueprint, name your topic branch `bp/BLUEPRINT` where BLUEPRINT is the name of a blueprint in Launchpad (for example, `bp/authentication`). The general convention when working on bugs is to name the branch `bug/BUG-NUMBER` (for example, `bug/1234567`). Otherwise, give it a meaningful name because it will show up as the topic for your change in Gerrit:

```
git checkout -b TOPIC-BRANCH
```

Committing a Change

Git commit messages should start with a short 50 character or less summary in a single paragraph. The following paragraph(s) should explain the change in more detail.

If your changes addresses a blueprint or a bug, be sure to mention them in the commit message using the following syntax:

```
Implements: blueprint BLUEPRINT
Closes-Bug: ####### (Partial-Bug or Related-Bug are options)
```

For example:

```
Adds keystone support

...Long multiline description of the change...

Implements: blueprint authentication
Closes-Bug: #123456
Change-Id: I4946a16d27f712ae2adf8441ce78e6c0bb0bb657
```

Note that in most cases the Change-Id line should be automatically added by a Gerrit commit hook installed by git-review. If you already made the commit and the Change-Id was not added, do the Gerrit setup step and run: `git commit --amend`. The commit hook will automatically add the Change-Id when you finish amending the commit message, even if you don't actually make any changes. Do not change the Change-Id when amending a change as that will confuse Gerrit.

Make your changes, commit them, and submit them for review:

```
git commit -a
```

Note

Do not check in changes on your master branch. Doing so will cause merge commits when you pull new upstream changes, and merge commits will not be accepted by Gerrit.

Using Signed-off-by

OpenStack projects do not currently require the use of a `Signed-off-by` header as a CLA is used, instead. However, you are welcome to include `Signed-off-by` in your commits. By doing so, you are certifying that the following is true:

```
Developer's Certificate of Origin 1.1

By making a contribution to this project, I certify that:

(a) The contribution was created in whole or in part by me and I
    have the right to submit it under the open source license
    indicated in the file; or

(b) The contribution is based upon previous work that, to the best
    of my knowledge, is covered under an appropriate open source
    license and I have the right under that license to submit that
    work with modifications, whether created in whole or in part
    by me, under the same open source license (unless I am
    permitted to submit under a different license), as indicated
    in the file; or

(c) The contribution was provided directly to me by some other
    person who certified (a), (b) or (c) and I have not modified
    it.

(d) I understand and agree that this project and the contribution
    are public and that a record of the contribution (including all
    personal information I submit with it, including my sign-off) is
    maintained indefinitely and may be redistributed consistent with
    this project or the open source license(s) involved.
```

A `Signed-off-by` header takes the following form in a commit message:

```
Signed-off-by: Full Name <email@example.com>
```

If you add the `-s` option to `git commit`, this header will be added automatically:

```
git commit -s
```

Running Unit Tests

Before submitting your change, you should test it. To learn how to run python based unit tests in OpenStack projects see Running Python Unit Tests

Previewing a Change

Before submitting your change, you should make sure that your change does not contain the files or lines you do not explicitly change:

```
git show
```

Submitting a Change for Review

Once you have committed a change to your local repository, all you need to do to send it to Gerrit for code review is run:

```
git review
```

When that completes, automated tests will run on your change and other developers will peer review it.

Updating a Change

If the code review process suggests additional changes, make and amend the changes to the

existing commit. Leave the existing Change-Id: footer in the commit message as-is. Gerrit knows that this is an updated patchset for an existing change:

```
git commit -a --amend
git review
```

Understanding Changes and Patch Sets

It's important to understand how Gerrit handles changes and patch sets. Gerrit combines the Change-Id in the commit message, the project, and the target branch to uniquely identify a change.

A new patch set is determined by any modification in the commit hash. When a change is initially pushed up it only has one patch set. When an update is done for that change, `git commit --amend` will change the most current commit's hash because it is essentially a new commit with the changes from the previous state combined with the new changes added. Since it has a new commit hash, once a `git review` is successfully processed, a new patch set appears in Gerrit.

Since a patch set is determined by a modification in the commit hash, many git commands will cause new patch sets. Three common ones that do this are:

- `git commit --amend`
- `git rebase`
- `git cherry-pick`

As long as you leave the "Change-Id" line in the commit message alone and continue to propose the change to the same target branch, Gerrit will continue to associate the new commit with the already existing change, so that reviewers are able to see how the change evolves in response to comments.

Squashing Changes

If you have made many small commits, you should squash them so that they do not show up in the public repository. Remember: each commit becomes a change in Gerrit, and must be approved separately. If you are making one "change" to the project, squash your many "checkpoint" commits into one commit for public consumption. Here's how:

```
git checkout master
git pull origin master
git checkout TOPIC-BRANCH
git rebase -i master
```

Use the editor to squash any commits that should not appear in the public history. If you want one change to be submitted to Gerrit, you should only have one "pick" line at the end of this process. After completing this, you can prepare your public commit message(s) in your editor. You start with the commit message from the commit that you picked, and it should have a Change-Id line in the message. Be sure to leave that Change-Id line in place when editing.

Once the commit history in your branch looks correct, run git review to submit your changes to Gerrit.

Adding a Dependency

When you want to start new work that is based on the commit under the review, you can add the commit as a dependency.

Fetch change under review and check out branch based on that change:

```
git review -d $PARENT_CHANGE_NUMBER
git checkout -b $DEV TOPIC BRANCH
```

Edit files, add files to git:

```
git commit -a
git review
```

Note

git review rebases the existing change (the dependency) and the new commit if there is a conflict against the branch they are being proposed to. Typically this is desired behavior as merging cannot happen until these conflicts are resolved. If you don't want to deal with new patchsets in the existing change immediately you can pass the `-R` option to git review in the last step above to prevent rebasing. This requires future rebasing to resolve conflicts.

If the commit your work depends on is updated, and you need to get the latest patchset from the depended commit, you can do the following.

Fetch and checkout the parent change:

```
git review -d $PARENT CHANGE NUMBER
```

Cherry-pick your commit on top of it:

```
git review -x $CHILD CHANGE NUMBER
```

Submit rebased change for review:

```
git review
```

The note for the previous example applies here as well. Typically you want the rebase behavior in git review. If you would rather postpone resolving merge conflicts you can use git review `-R` as the last step above.

Cross-Repository Dependencies

If your change has a dependency on a change outside of that repository, like a change for another repository or some manual setup, you have to ensure that the change merge at the right time.

For a change depending on a manual setup, mark your change with the "Work in Progress" label until the manual setup is done. A core reviewer might also block an important change with a -2 so that it does not get merged accidentally before the manual setup is done.

If your change has a dependency on a change in another repository, you can use cross-repo dependencies (CRD) in Zuul:

- To use them, include "Depends-On: <gerrit-change-id>" in the footer of your commit message. Use the full Change-ID ('I' + 40 characters). A patch can also depend on multiple changes as explained in *Multiple Changes*.
- These are one-way dependencies only – do not create a cycle.

Gate Pipeline

When Zuul sees CRD changes, it serializes them in the usual manner when enqueuing them into a pipeline. This means that if change A depends on B, then when they are added to the gate pipeline, B will appear first and A will follow. If tests for B fail, both B and A will be removed from the pipeline, and it will not be possible for A to merge until B does.

Note that if changes with CRD do not share a change queue (such as the "integrated gate"), then Zuul is unable to enqueue them together, and the first will be required to merge before the second is enqueued.

Check Pipeline

When changes are enqueued into the check pipeline, all of the related dependencies (both normal git-dependencies that come from parent commits as well as CRD changes) appear in a dependency graph, as in the gate pipeline. This means that even in the check pipeline, your change will be tested with its dependency. So changes that were previously unable to be fully tested until a related change landed in a different repo may now be tested together from the start.

All of the changes are still independent (so you will note that the whole pipeline does not share a graph as in the gate pipeline), but for each change tested, all of its dependencies are visually connected to it, and they are used to construct the git references that Zuul uses when testing. When looking at this graph on the Zuul status page, you will note that the dependencies show up as grey dots, while the actual change tested shows up as red or green. This is to indicate that the grey changes are only there to establish dependencies. Even if one of the dependencies is also being tested, it will show up as a grey dot when used as a dependency, but separately and additionally will appear as its own red or green dot for its test.

Multiple Changes

A Gerrit change ID may refer to multiple changes (on multiple branches of the same project, or even multiple projects). In these cases, Zuul will treat all of the changes with that change ID as dependencies. So if you say that a tempest change Depends-On a change ID that has changes in nova master and nova stable/juno, then when testing the tempest change, both nova changes will be applied, and when deciding whether the tempest change can merge, both changes must merge ahead of it.

A change may depend on more than one Gerrit change ID as well. So it is possible for a change in tempest to depend on a change in devstack and a change in nova. Simply add more "Depends-On:" lines to the footer.

Cycles

If a cycle is created by use of CRD, Zuul will abort its work very early. There will be no message in Gerrit and no changes that are part of the cycle will be enqueued into any pipeline. This is to protect Zuul from infinite loops. The developers hope that they can improve this to at least leave a message in Gerrit in the future. But in the meantime, please be cognizant of this and do not create dependency cycles with Depends-On lines.

Limitations and Caveats

Keep in mind that these dependencies are dependencies on changes in other repositories. Thus, a Depends-on only enforces an ordering but is not visible otherwise especially in these cases:

- Changes for the CI infrastructure like changes `openstack-infra/project-config` are never tested in a production simulated environment. So, if one of the changes adjusts the job definitions or creates a new job, a Depends-On will not test the new definition, the CI infrastructure change needs to merge to master and be in production to be fully evaluated.

- If a test job installs packages from PyPI and not via source, be aware that the package from PyPI will always be used, a Depends-On will not cause a modified package to be used instead of installing from PyPI.

  As an example, if you are testing a change in python-novaclient that needs a change in python-keystoneclient, you add a Depends-On in the python-novaclient change. If a python-novaclient job installs python-keystoneclient from PyPI, the Depends-On will not have any effect since the PyPI version is used. If a python-novaclient job installs python-keystoneclient from source, the checked out source will have the change applied.

Do not add a Depends-On an abandoned change, your change will never merge.

If you backport a change to another branch, the gerrit change ID stays the same. If you add a Depends-On using the Gerrit change ID of the patch that subsequently was backported, the patch with the Depends-On is now also dependent on the backported change. This might be desirable for some changes and a surprise for others.

A change that is dependent on another can be approved before the dependent change merges. If the repositories share the gate queue, it will merge automatically after the dependent change merged. But if the repositories do not share the gate queue, it will not merge automatically when the dependent change has merged, even a `recheck` will not help. Zuul waits for a status change

and does not see it. The change needs another approval or a toggle of the approval, toggle means removing the approval and readding it again.

Code Review

Log in to https://review.openstack.org/ to see proposed changes, and review them.

To provide a review for a proposed change in the Gerrit UI, click on the Review button (it will be next to the buttons that will provide unified or side-by-side diffs in the browser). In the code review, you can add a message, as well as a vote (+1,0,-1).

It's also possible to add comments to specific lines in the file, for giving context to the comment. For that look at the diff of changes done in the file (click the file name), and click on the line number for which you want to add the inline comment. After you add one or more inline comments, you still have to send the Review message (see above, with or without text and vote). Prior to sending the inline comments in a review comment the inline comments are stored as Drafts in your browser. Other reviewers can only see them after you have submitted them as a comment on the patchset.

Any OpenStack developer may propose or comment on a change (including voting +1/0/-1 on it). OpenStack project teams have a policy requiring two positive reviews from core reviewers. A vote of +2 is allowed from core reviewers, and should be used to indicate that they are a core reviewer and are leaving a vote that should be counted as such.

When a review has two +2 reviews and one of the core team believes it is ready to be merged, he or she should leave a +1 vote in the "Approved" category. You may do so by clicking the "Review" button again, with or without changing your code review vote and optionally leaving a comment. When a +1 Approved review is received, Jenkins will run tests on the change, and if they pass, it will be merged.

A green checkmark indicates that the review has met the requirement for that category. Under "Code-Review", only one +2 gets the green check.

For more details on reviews in Gerrit, check the Gerrit documentation.

Automated Testing

When a new patchset is uploaded to Gerrit, that project's "check" tests are run on the patchset by Jenkins. Once completed the test results are reported to Gerrit by Jenkins in the form of a Verified: +/-1 vote. After code reviews have been completed and a change receives an Approved: +1 vote that project's "gate" tests are run on the change by Jenkins. Jenkins reports the results of these tests back to Gerrit in the form of a Verified: +/-2 vote. Code merging will only occur after the gate tests have passed successfully and received a Verified: +2. You can view the state of tests currently being run on the Zuul Status page.

If a change fails tests in Jenkins, please follow the steps below:

1. Jenkins leaves a comment in the review with links to the log files for the test run. Follow those links and examine the output from the test. It will include a console log, and in the case of unit tests, HTML output from the test runner, or in the case of a devstack-gate test, it may contain quite a large number of system logs. For jobs in the post queue, logs are found at `http://logs.openstack.org/<first two characters of commit SHA>/<commit SHA>`. For example, if a change is committed with the sha 'deadbeef123456', the logs will be found at `http://logs.openstack.org/de/deadbeef123456`.
2. Examine the console log or other relevant log files to determine the cause of the error. If it is related to your change, you should fix the problem and upload a new patchset. Do not use "recheck".
3. It may be the case that the problem is due to non-deterministic behavior unrelated to your change that has already merged. In this situation, you can help other developers and focus the attention of QA, CI, and developers working on a fix by performing the following steps:

1. Visit http://status.openstack.org/elastic-recheck/ to see if one of the bugs listed there matches the error you've seen. If your error isn't there, then:
2. Identify which project or projects are affected, and search for a related bug on Launchpad. You can search for bugs affecting all OpenStack Projects here: https://bugs.launchpad.net/openstack/ If you do not find an existing bug, file a new one (be sure to include the error message and a link to the logs for the failure). If the problem is due to an infrastructure problem (such as Jenkins or Gerrit), file (or search for) the bug against the openstack-gate project.

4. To re-run check or gate jobs, leave a comment on the review with the form "recheck".
5. If a nice message from Elastic Recheck didn't show up in your change when a test in a gate job failed, and you've identified a bug to recheck against, you can help out by writing an elastic-recheck query for the bug.

A patchset has to be approved to run tests in the gate pipeline. If the patchset has failed in the gate pipeline (it will have been approved to get into the gate pipeline) a recheck will first run the check jobs and if those pass, it will again run the gate jobs. There is no way to only run the gate jobs, the check jobs will first be run again.

More information on debugging automated testing failures can be found in the following recordings:

- Tales From The Gate
- Debugging Failures in the OpenStack Gate

Peer Review

Anyone can be a reviewer: participating in the review process is a great way to learn about OpenStack social norms and the development processes. Some things are necessary to keep in mind when doing code reviews:

1. The code should comply with everything in that project's *HACKING.rst* file, if it has one. If the project reuses nova's hacking guidelines, then it may have a "hacking" section in its *tox.ini* file in which case much of this is already checked automatically for you by the continuous integration system.
2. The code should be 'pythonic' and look like the code around it, to make the code more uniform and easier to read.
3. Commit message and change break-up:

    1. Learn the best practices for git commit messages.
    2. Use the "DocImpact" tag on changes that affect documentation.
    3. Use the "SecurityImpact" tag on changes that should get the attention of the OpenStack Security Group (OSSG) for additional review.
    4. Use the "UpgradeImpact" tag on changes which require configuration changes to be mentioned in the release notes.
    5. Use the "APIImpact" tag on changes impacting API stability, this tag will aid in gaining the attention of the OpenStack API Working Group for additional review.
    6. If the change fixes a bug, it should include the bug number. For example, add the line "Closes-Bug: 1234".
    7. If the change implements a feature, it should reference a blueprint. The blueprint should be approved before the change is merged. For example, add the line "Blueprint: my-blueprint."

4. Test case implementation (Mock vs. Mox):

    1. New test cases should be implemented using Mock. It is part of the Python standard library in Python 3 and as such is the preferred method for OpenStack.
    2. Exceptions can be made for tests added where Mox was already in use, or any other situation where using Mock would cause excessive difficulty for some reason. However, note that using mox does not support python 3 and mox3 has known to intermittently fail in py34 jobs, so it should be avoided if python 3 compatibility is a goal of the project being tested.
    3. There is no need to convert existing Mox test cases to Mock, but if you are changing a

Mox test case anyway, please consider converting it to Mock at the same time.

5. About Python 3 compatibility:

    1. It is preferred for new code to use package six. When it is possible we should be use *six.text_type* or *six.text_binary* to cast or test value for unicode or str.
    2. Use of *six.iteritems* without clear justification should be avoided. If a *dict* will be very large, and the program will be expected to keep many such objects resident, then that should be stated in comments whenever *six.iteritems* is used. Otherwise, migrate the code to use *.items()*.
    3. Unit tests should be written in mock which supports python 3. mox does not support python 3 and mox3 is a limited port which intermittently fails in py34 jobs due to races.

6. The code should comply with the community logging standards.

7. General flow:

    1. Review is a conversation that works best when it flows back and forth. Submitters need to be responsive to questions asked in comments, even if the score is *+0* from the reviewer. Likewise, reviewers should not use a negative score to elicit a response if they are not sure the patch should be changed before merging.

        For example, if there is a patch submitted which a reviewer cannot fully understand because there are changes that aren't documented in the commit message or code documentation, this is a good time to issue a negative score. Patches need to be clear in their commit message and documentation.

        As a counter-example, a patch which is making use of a new library, which the reviewer has never used before, should not elicit a negative score from the reviewer with a question like "Is this library using standard python sockets for communication?" That is a question the reviewer can answer themselves, and which should not hold up the review process while the submitter explains things. Either the author or a reviewer should try to add a review comment answering such questions, unless they indicate a need to better extend the commit message, code comments, docstrings or accompanying documentation files.

    2. In almost all cases, a negative review should be accompanied by clear instructions for the submitter how they might fix the patch.

There may be more specific items to be aware of inside the projects' documentation for contributors.

Contributors may notice a review that has several +1's from other reviewers, passes the functional tests, etc. but the code still has not been merged. As only core contributors can approve code for merging, you can help things along by getting a core developer's attention in IRC (never on the mailing lists) and letting them know there is a changeset with lots of positive reviews and needs final approval.

Work in Progress

To get early feedback on a change which is not fully finished yet, you can submit a change to Gerrit and mark it as "Work in Progress" (WIP).

> **Note**
>
> The OpenStack Gerrit system does not support drafts, use "Work in Progress" instead.

To do so, after submitting a change to Gerrit in usual way (`git review`), You should go to Gerrit, and do Code Review of your own change while setting "Workflow" vote to "`-1`", which marks the change as WIP.

This allows others to review the change, while at the same time blocking it from being merged, as

you already plan to continue working on it.

Note

After uploading a new patchset, this -1 (WIP) vote disappears. So if you still plan to do additional changes, do not forget to set Workflow to -1 on the new patchset.

Merging

Once a change has been approved and passed the gate jobs, Gerrit automatically merges the latest patchset.

Each patchset gets merged to the head of the branch before testing it. If Gerrit cannot merge a patchset, it will give a -1 review and add a comment notifying of merge failure.

Each time a change merges, the "merge-check" pipeline verifies that all open changes on the same project are still mergeable. If any job is not mergeable, Jenkins will give a -1 review and add a comment notifying of merge failure.

After a change is merged, project-specific post jobs are run. Most often the post jobs publish documentation, run coverage, or send strings to the translation server.

Project Gating

Project gating refers to the process of running regression tests before a developer's patchset is merged. The intent of running regression tests is to validate that new changes submitted against the source code repository will not introduce new bugs. Gating prevents regressions by ensuring that a series of tests pass successfully before allowing a patchset to be merged into the mainline of development.

The system used for gating is Zuul, which listens to the Gerrit event stream and is configured with YAML files to define a series of tests to be run in response to an event.

The jobs in the gate queue are executed once a core reviewer approves a change (using a +1 Workflow vote) and a verified +1 vote exist. When approving, at least one +2 Code-Review vote needs to exist (can be given by core reviewer when approving). The convention is that two +2 Code-Reviews are needed for approving.

Once all of the jobs report success on an approved patchset in the configured gate pipeline, then Gerrit will merge the code into trunk.

Besides running the gate tests, the gate pipeline determines the order of changes to merge across multiple projects. The changes are tested and merged in this order, so that for each change the state of all other repositories can be identified.

Additional information about project gating and Zuul can be found in the Zuul documentation, located at: http://docs.openstack.org/infra/zuul/gating.html

© Copyright 2016, OpenStack Contributors. Freely licensed under CC BY 3.0. Propose changes to the infra-manual git repo. Last updated on Tue Aug 9 12:27:52 2016, commit 876e3bc. Created using Sphinx 1.1.3.

openstack SEARCH

Mitaka release

UPDATED: 2016-08-16 23:14

Contents

- Configuration Reference
- High Availability Guide
- Installation Guide
- Networking Guide
- Operations Guide
- User Guides
- Virtual Machine Image Guide
- Command-Line Interface Reference
- Architecture Design Guide
- API Guides
- Training Guides
- Translations

- Tracking of release notes in the `releasenotes` directory.

Configuration Reference

- Completed RST conversion.
- Documented Message service (zaqar).

High Availability Guide

- Added the Highly available Shared File Systems API section.
- Improved Pacemaker/Corosync cluster installation and configuration details.
- Documented the Pacemaker cluster manager and Keepalived architecture details and limitations.
- Added the MariaDB Galera cluster installation, configuration, and management details.
- Improved the RabbitMQ section.

Installation Guide

- Updated configuration for all services.
- Added Shared File Systems (manila) content.
- Added Database service (trove) content.

Networking Guide

- Documentation of some of the new features in Mitaka.
- New content including documentation for LBaaS, DNS integration, and macvtap ml2 driver.

Operations Guide

- Added the Shared File Systems chapter.

User Guides

- Reorganised the Admin User Guide content together with the Cloud Admin Guide content to create a new Administrator Guide.
- Approximately one third of the Administrator Guide chapters received a thorough edit for style and consistency, following the contributor guide standard.
- Troubleshooting chapters now have consistent formatting, which is a step toward improved troubleshooting sections.
- The Admin User Guide content has been removed from OpenStack Docs since all files have been reorganised into the Administrator Guide.

Virtual Machine Image Guide

- Completed RST conversion.

Command-Line Interface Reference

- Completed RST conversion.
- Documented that individual CLIs are deprecated in favor of the common OpenStack client.
- Marked Identity API v2 as deprecated.
- Added senlin, monasca, and cloudkitty clients.
- Removed tuskar client because of retirement.

Architecture Design Guide

- Completed RST conversion.

API Guides

- New, cleaner developer.openstack.org landing page.
- API Quick Start converted to RST with theme styling to match.
- Compute API Guide now built from nova source tree.
- Draft swagger files now built to http://developer.openstack.org/draft/swagger/
- Created templates for writing API guides for projects teams available in projects repositories.
- Released fairy-slipper, a migration tool for WADL to RST plus API reference information.

Training Guides

- Added the Upstream Training Archives (the list of past global and local Upstream Training events).
- Added the bug report links to each slide and the landing page.
- Enabled translation.

Translations

- Japanese
  - Published the Networking Guide.
  - Published the Upstream Training.
- Korean
  - Published the Upstream Training.
- German
  - Published the Upstream Training.

UPDATED: 2016-08-16 23:14

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Mitaka release
  - Configuration Reference
  - High Availability Guide
  - Installation Guide
  - Networking Guide
  - Operations Guide
  - User Guides
  - Virtual Machine Image Guide
  - Command-Line Interface Reference
  - Architecture Design Guide
  - API Guides
  - Training Guides
  - Translations
- Next release: Newton

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation

Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

OpenStack Installation Guide for openSUSE and SUSE Linux Enterprise

UPDATED: 2016-08-13 23:58

OpenStack Installation Guide for openSUSE and SUSE Linux Enterprise
Abstract
Contents
Appendix
Glossary
Search in this guide

The OpenStack system consists of several key services that are separately installed. These services work together depending on your cloud needs. These services include Compute service, Identity service, Networking service, Image service, Block Storage service, Object Storage service, Telemetry service, Orchestration service, and Database service. You can install any of these projects separately and configure them stand-alone or as connected entities.

This guide will show you how to install OpenStack by using packages on openSUSE Leap 42.1 and SUSE Linux Enterprise Server 12 SP1 through the Open Build Service Cloud repository.

Explanations of configuration options and sample configuration files are included.

This guide documents OpenStack Mitaka release.

- Conventions
  - Notices
  - Command prompts
- Overview
  - Example architecture
  - Networking

- Environment
  - Security
  - Host networking
  - Network Time Protocol (NTP)
  - OpenStack packages
  - SQL database
  - NoSQL database
  - Message queue
  - Memcached
- Identity service
  - Identity service overview
  - Install and configure
  - Create the service entity and API endpoints
  - Create a domain, projects, users, and roles
  - Verify operation
  - Create OpenStack client environment scripts
- Image service
  - Image service overview
  - Install and configure
  - Verify operation
- Compute service
  - Compute service overview
  - Install and configure controller node
  - Install and configure a compute node
  - Verify operation
- Networking service
  - Networking service overview
  - Networking (neutron) concepts
  - Install and configure controller node
  - Install and configure compute node
  - Verify operation
  - Next steps
- Dashboard
  - Install and configure
  - Verify operation
  - Next steps
- Block Storage service
  - Block Storage service overview
  - Install and configure controller node
  - Install and configure a storage node
  - Verify operation
  - Next steps
- Shared File Systems service
  - Shared File Systems service overview
  - Install and configure controller node
  - Install and configure a share node
  - Verify operation
  - Next steps
- Object Storage service
  - Object Storage service overview
  - Install and configure the controller node

    - Install and configure the storage nodes
    - Create and distribute initial rings
    - Finalize installation
    - Verify operation
    - Next steps
- Orchestration service
    - Orchestration service overview
    - Install and configure
    - Verify operation
    - Next steps
- Telemetry service
    - Telemetry service overview
    - Install and configure
    - Enable Image service meters
    - Enable Compute service meters
    - Enable Block Storage meters
    - Enable Object Storage meters
    - Alarming service
    - Verify operation
    - Next steps
- Database service
    - Database service overview
    - Install and configure
    - Verify operation
    - Next steps
- Launch an instance
    - Create virtual networks
    - Create m1.nano flavor
    - Generate a key pair
    - Add security group rules
    - Launch an instance
    - Block Storage
    - Orchestration
    - Shared File Systems
- Community support
- Glossary
- *Search Page*

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups

Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

OpenStack Installation Guide for Ubuntu

UPDATED: 2016-08-13 23:58

OpenStack Installation Guide for Ubuntu
Abstract
Contents
Appendix
Glossary
Search in this guide

The OpenStack system consists of several key services that are separately installed. These services work together depending on your cloud needs. These services include Compute service, Identity service, Networking service, Image service, Block Storage service, Object Storage service, Telemetry service, Orchestration service, and Database service. You can install any of these projects separately and configure them stand-alone or as connected entities.

This guide will walk through an installation by using packages available through Canonical's Ubuntu Cloud archive repository.

Explanations of configuration options and sample configuration files are included.

This guide documents OpenStack Mitaka release.

- Conventions
  - Notices
  - Command prompts
- Overview
  - Example architecture
  - Networking
- Environment

    - Security
    - Host networking
    - Network Time Protocol (NTP)
    - OpenStack packages
    - SQL database
    - NoSQL database
    - Message queue
    - Memcached
- Identity service
    - Identity service overview
    - Install and configure
    - Create the service entity and API endpoints
    - Create a domain, projects, users, and roles
    - Verify operation
    - Create OpenStack client environment scripts
- Image service
    - Image service overview
    - Install and configure
    - Verify operation
- Compute service
    - Compute service overview
    - Install and configure controller node
    - Install and configure a compute node
    - Verify operation
- Networking service
    - Networking service overview
    - Networking (neutron) concepts
    - Install and configure controller node
    - Install and configure compute node
    - Verify operation
    - Next steps
- Dashboard
    - Install and configure
    - Verify operation
    - Next steps
- Block Storage service
    - Block Storage service overview
    - Install and configure controller node
    - Install and configure a storage node
    - Verify operation
    - Next steps
- Shared File Systems service
    - Shared File Systems service overview
    - Install and configure controller node
    - Install and configure a share node
    - Verify operation
    - Next steps
- Object Storage service
    - Object Storage service overview
    - Install and configure the controller node
    - Install and configure the storage nodes

    - Create and distribute initial rings
    - Finalize installation
    - Verify operation
    - Next steps
- Orchestration service
    - Orchestration service overview
    - Install and configure
    - Verify operation
    - Next steps
- Telemetry service
    - Telemetry service overview
    - Install and configure
    - Enable Image service meters
    - Enable Compute service meters
    - Enable Block Storage meters
    - Enable Object Storage meters
    - Alarming service
    - Verify operation
    - Next steps
- Database service
    - Database service overview
    - Install and configure
    - Verify operation
    - Next steps
- Launch an instance
    - Create virtual networks
    - Create m1.nano flavor
    - Generate a key pair
    - Add security group rules
    - Launch an instance
    - Block Storage
    - Orchestration
    - Shared File Systems
- Community support
- Glossary
- *Search Page*

UPDATED: 2016-08-13 23:58

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Overview
- Environment
- Identity service
- Image service
- Compute service
- Networking service
- Dashboard
- Block Storage service
- Shared File Systems service
- Object Storage service
- Orchestration service
- Telemetry service
- Database service
- Launch an instance
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups

Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

OpenStack Administrator Guide

UPDATED: 2016-08-16 23:13

Abstract
Contents
Appendix
Glossary
Search in this guide

Abstract

OpenStack offers open source software for OpenStack administrators to manage and troubleshoot an OpenStack cloud.

This guide documents OpenStack Mitaka, and OpenStack Liberty releases.

Contents

- Conventions
  - Notices
  - Command prompts
- Get started with OpenStack
  - Conceptual architecture
  - Logical architecture
  - OpenStack services

    - Feedback
- Identity management
    - Identity concepts
    - Certificates for PKI
    - Configure the Identity service with SSL
    - Domain-specific configuration
    - External authentication with Identity
    - Integrate Identity with LDAP
    - Keystone tokens
    - Configure Identity service for token binding
    - Fernet - Frequently Asked Questions
    - Use trusts
    - Caching layer
    - Start the Identity service
    - Example usage and Identity features
    - Authentication middleware with user name and password
    - Identity API protection with role-based access control (RBAC)
    - Troubleshoot the Identity service
- Dashboard
    - Customize the dashboard
    - Set up session storage for the Dashboard
    - Create and manage images
    - Create and manage roles
    - Manage instances
    - Manage flavors
    - Manage volumes and volume types
    - Manage shares and share types
    - View and manage quotas
    - View cloud resources
    - Create and manage host aggregates
    - Launch and manage stacks using the Dashboard
- Compute
    - System architecture
    - Images and instances
    - Networking with nova-network
    - System administration
    - Troubleshoot Compute
- Object Storage
    - Introduction to Object Storage
    - Features and benefits
    - Object Storage characteristics
    - Components
    - Ring-builder
    - Cluster architecture
    - Replication
    - Large object support
    - Object Auditor
    - Erasure coding
    - Account reaper
    - Configure tenant-specific image locations with Object Storage
    - Object Storage monitoring

    - System administration for Object Storage
    - Troubleshoot Object Storage
- Block Storage
    - Increase Block Storage API service throughput
    - Manage volumes
    - Troubleshoot your installation
- Shared File Systems
    - Introduction
    - Key concepts
    - Share management
    - Migrate shares
    - Share types
    - Share snapshots
    - Security services
    - Consistency groups
    - Share replication
    - Multi-storage configuration
    - Networking
    - Troubleshoot Shared File Systems service
- Networking
    - Introduction to Networking
    - Networking architecture
    - Plug-in configurations
    - Configure neutron agents
    - Configure Identity service for Networking
    - Advanced configuration options
    - Scalable and highly available DHCP agents
    - Use Networking
    - Advanced features through API extensions
    - Advanced operational features
    - Authentication and authorization
- Telemetry
    - System architecture
    - Data collection
    - Data retrieval
    - Alarms
    - Measurements
    - Events
    - Troubleshoot Telemetry
    - Telemetry best practices
- Database
    - Introduction
    - Create a data store
    - Configure a cluster
- Bare Metal
    - Introduction
    - System architecture
    - Bare Metal deployment
    - Use Bare Metal
    - Troubleshooting
- Orchestration

    - Introduction
    - Orchestration authorization model
    - Stack domain users
- OpenStack command-line clients
    - Overview
    - Install the OpenStack command-line clients
    - Discover the version number for a client
    - Set environment variables using the OpenStack RC file
    - Manage projects, users, and roles
    - Manage project security
    - Manage services
    - Manage images
    - Manage volumes
    - Manage shares
    - Manage flavors
    - Manage the OpenStack environment
    - Manage quotas
    - Analyze log files
    - Manage Block Storage scheduling
- Cross-project features
    - Cross-origin resource sharing

Appendix

- Community support

Glossary

- Glossary

Search in this guide

- *Search Page*

UPDATED: 2016-08-16 23:13

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Get started with OpenStack
- Identity management
- Dashboard
- Compute
- Object Storage
- Block Storage
- Shared File Systems
- Networking
- Telemetry
- Database
- Bare Metal
- Orchestration
- OpenStack command-line clients
- Cross-project features
- Community support
- Glossary

Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack

SEARCH

OpenStack High Availability Guide

UPDATED: 2016-08-16 23:15

Abstract
Contents
Appendix
Glossary
Search in this guide

Abstract

This guide describes how to install and configure OpenStack for high availability. It supplements the OpenStack Installation Guides and assumes that you are familiar with the material in those guides.

This guide documents OpenStack Mitaka, and OpenStack Liberty releases.

Warning

This guide is a work-in-progress and changing rapidly while we continue to test and enhance the guidance. Please note where there are open “to do” items and help where you are able.

Contents

- Conventions
  - Notices
  - Command prompts
- Introduction to OpenStack high availability
  - High availability concepts
  - Overview of highly-available controllers
  - Overview of high availability storage
  - Overview of highly-available compute nodes
  - High availability for other components
- Basic environment for high availability
  - Hardware considerations for high availability
  - Install operating system on each node
  - Configure NTP
  - Install memcached
- Shared services for high availability
  - Database (Galera Cluster) for high availability
  - Messaging service for high availability
- Configuring the controller for high availability
  - Pacemaker cluster stack
  - Configure the VIP
  - HAProxy
  - Memcached
  - Highly available Identity API
  - Highly available Telemetry API
- Networking services for high availability
  - Run Networking DHCP agent
  - Run Networking L3 agent
  - Run Networking metadata agent
  - Run Networking LBaaS agent
- Configuring Storage for high availability
  - Highly available Image API
  - Highly available Block Storage API
  - Highly available Shared File Systems API
  - Storage back end
- Configuring the compute node for high availability
  - Configure high availability on compute nodes

Appendix

- Community support

Glossary

- Glossary

Search in this guide

- *Search Page*

UPDATED: 2016-08-16 23:15

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Introduction to OpenStack high availability
- Basic environment for high availability
- Shared services for high availability
- Configuring the controller for high availability
- Networking services for high availability
- Configuring Storage for high availability
- Configuring the compute node for high availability
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

OpenStack Operations Guide

This book offers hard-earned experience from OpenStack operators who have run OpenStack in production for six months or longer. They've gathered their notes, shared their stories, and learned from each other. We invite you to join in the quest for best practices in OpenStack cloud operations. You can read the latest updates or download the new O'Reilly published versions below.

Get The Book!

Current HTML | O'Reilly .Mobi | O'Reilly .Epub

OpenStack Operations Guide

SET UP AND MANAGE YOUR OPENSTACK CLOUD

Tom Fifield, Diane Fleming, Anne Gentle,
Lorin Hochstein, Jonathan Proulx,
Everett Toews & Joe Topjian

If you're interested in OpenStack books, please take a look at the OpenStack Security Guide , also written in a five-day book sprint.

This work is licensed under a Creative Commons Attribution 3.0 United States License .

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing .

openstack SEARCH

OpenStack Security Guide

UPDATED: 2016-08-16 07:46

Abstract
Contents
Appendix
Glossary
Search in this guide

Abstract

This book provides best practices and conceptual information about securing an OpenStack cloud.

This guide documents OpenStack Mitaka, OpenStack Liberty, and OpenStack Kilo releases.

Contents

- Conventions
  - Notices
  - Command prompts
- Introduction
  - Acknowledgements
  - Why and how we wrote this book
  - Introduction to OpenStack

    - Security boundaries and threats
    - Selecting supporting software
    - Introduction to case studies
- System documentation
    - System documentation requirements
    - Case studies
- Management
    - Continuous systems management
    - Integrity life-cycle
    - Management interfaces
    - Case studies
- Secure communication
    - Introduction to TLS and SSL
    - TLS proxies and HTTP services
    - Secure reference architectures
    - Case studies
- API endpoints
    - API endpoint configuration recommendations
    - Case studies
- Identity
    - Authentication
    - Authentication methods
    - Authorization
    - Policies
    - Tokens
    - Domains
    - Federated keystone
    - Case studies
    - Checklist
- Dashboard
    - Domain names, dashboard upgrades, and basic web server configuration
    - HTTPS, HSTS, XSS, and SSRF
    - Front-end caching and session back end
    - Static media
    - Passwords
    - Secret key
    - Cookies
    - Cross Origin Resource Sharing (CORS)
    - Debug
    - Case studies
    - Checklist
- Compute
    - Hypervisor selection
    - Hardening the virtualization layers
    - Hardening Compute deployments
    - Vulnerability awareness
    - How to select virtual consoles
    - Case studies
    - Checklist
- Block Storage
    - Volume Wiping

    - Checklist
- Shared File Systems
    - Introduction
    - Network and security models
    - Security services
    - Share access control
    - Share type access control
    - Policies
    - Checklist
- Networking
    - Networking architecture
    - Networking services
    - Networking services security best practices
    - Securing OpenStack networking services
    - Case studies
    - Checklist
- Object Storage
    - First thing to secure: the network
    - Securing services: general
    - Securing storage services
    - Securing proxy services
    - Object Storage authentication
    - Other notable items
- Message queuing
    - Messaging security
    - Case studies
- Data processing
    - Introduction to Data processing
    - Deployment
    - Configuration and hardening
    - Case studies
- Databases
    - Database back end considerations
    - Database access control
    - Database transport security
    - Case studies
- Tenant data privacy
    - Data privacy concerns
    - Data encryption
    - Key management
    - Case studies
- Instance security management
    - Security services for instances
    - Case studies
- Monitoring and logging
    - Forensics and incident response
    - Case studies
- Compliance
    - Compliance overview
    - Understanding the audit process
    - Compliance activities

- Certification and compliance statements
- Privacy
- Case studies

Appendix

- Security Checklist
- Community support

Glossary

- Glossary

Search in this guide

- *Search Page*

UPDATED: 2016-08-16 07:46

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Introduction
- System documentation
- Management
- Secure communication

- API endpoints
- Identity
- Dashboard
- Compute
- Block Storage
- Shared File Systems
- Networking
- Object Storage
- Message queuing
- Data processing
- Databases
- Tenant data privacy
- Instance security management
- Monitoring and logging
- Compliance
- Security Checklist
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines

Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

OpenStack Virtual Machine Image Guide

UPDATED: 2016-08-16 23:15

Abstract
Contents
Appendix
Glossary
Search in this guide

Abstract

This guide describes how to obtain, create, and modify virtual machine images that are compatible with OpenStack.

Contents

- Conventions
  - Notices
  - Command prompts
- Introduction
  - Disk and container formats for images
  - Image metadata
- Get images
  - CentOS
  - CirrOS (test)

    - Debian
    - Fedora
    - Microsoft Windows
    - Ubuntu
    - openSUSE and SUSE Linux Enterprise Server
    - Red Hat Enterprise Linux
- Image requirements
    - Linux
    - Disk partitions and resize root partition on boot (cloud-init)
    - No hard-coded MAC address information
    - Ensure ssh server runs
    - Disable firewall
    - Access instance by using ssh public key (cloud-init)
    - Process user data and other metadata (cloud-init)
    - Ensure image writes boot log to console
    - Paravirtualized Xen support in the kernel (Xen hypervisor only)
    - Manage the image cache
- Modify images
    - guestfish
    - guestmount
    - virt-* tools
    - Loop devices, kpartx, network block devices
- Create images manually
    - Verify the libvirt default network is running
    - Use the virt-manager X11 GUI
    - Use virt-install and connect by using a local VNC client
    - Example: CentOS image
    - Example: Ubuntu image
    - Example: Fedora image
    - Example: Microsoft Windows image
    - Example: FreeBSD image
- Tool support for image creation
    - Diskimage-builder
    - Oz
    - VeeWee
    - Packer
    - image-bootstrap
    - imagefactory
    - KIWI
    - SUSE Studio
    - virt-builder
- Converting between image formats
    - qemu-img convert: raw, qcow2, qed, vdi, vmdk, vhd
    - VBoxManage: VDI (VirtualBox) to raw
- Image sharing
    - Sharing an image

Appendix

- Community support

Glossary

- Glossary

Search in this guide

- *Search Page*

UPDATED: 2016-08-16 23:15

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Introduction
- Get images
- Image requirements
- Modify images
- Create images manually
- Tool support for image creation
- Converting between image formats
- Image sharing
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

OpenStack Architecture Design Guide

UPDATED: 2016-08-16 23:14

Abstract
Contents
Appendix
Glossary
Search in this guide

Abstract

To reap the benefits of OpenStack, you should plan, design, and architect your cloud properly, taking user's needs into account and understanding the use cases.

Contents

- Conventions
  - Notices
  - Command prompts
- Introduction
  - Intended audience
  - How this book is organized
  - Why and how we wrote this book

    - Methodology
- Security and legal requirements
    - Legal requirements
    - Security
    - Security domains
    - Hypervisor-security
    - Baremetal security
    - Networking security
    - Multi-site security
    - OpenStack components
- General purpose
    - User requirements
    - Technical considerations
    - Operational considerations
    - Architecture
    - Prescriptive example
- Compute focused
    - Technical considerations
    - Operational considerations
    - Architecture
    - Prescriptive examples
- Storage focused
    - Technical considerations
    - Operational Considerations
    - Architecture
    - Prescriptive Examples
- Network focused
    - User requirements
    - Technical considerations
    - Operational considerations
    - Architecture
    - Prescriptive examples
- Multi-site
    - User requirements
    - Technical considerations
    - Operational considerations
    - Architecture
    - Prescriptive examples
- Hybrid
    - User requirements
    - Technical considerations
    - Architecture
    - Operational considerations
    - Prescriptive examples
- Massively scalable
    - User requirements
    - Technical considerations
    - Operational considerations
- Specialized cases
    - Multi-hypervisor example
    - Specialized networking example

    - Software-defined networking
    - Desktop-as-a-Service
    - OpenStack on OpenStack
    - Specialized hardware
- References

Appendix

- Community support

Glossary

- Glossary

Search in this guide

- *Search Page*

UPDATED: 2016-08-16 23:14

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Introduction
- Security and legal requirements
- General purpose

- Compute focused
- Storage focused
- Network focused
- Multi-site
- Hybrid
- Massively scalable
- Specialized cases
- References
- Community support
- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.

openstack SEARCH

OpenStack Networking Guide

UPDATED: 2016-08-13 23:56

Abstract
Contents
Appendix
Glossary
Search in this guide

Abstract

This guide targets OpenStack administrators seeking to deploy and manage OpenStack Networking (neutron).

This guide documents the OpenStack Mitaka release.

Contents

- Conventions
  - Notices
  - Command prompts
- Introduction to networking
  - Basic networking
  - Network components
  - Tunnel technologies

    - Network namespaces
    - Network address translation
- Introduction to OpenStack Networking (neutron)
    - Overview and components
    - Service and component hierarchy
    - Additional features
- Configuration
    - Server
    - ML2 plug-in
    - Name resolution for instances
- Deployment scenarios
    - Scenario: Classic with Open vSwitch
    - Scenario: Classic with Linux Bridge
    - Scenario: Classic with Macvtap
    - Scenario: High Availability using Distributed Virtual Routing (DVR)
    - Scenario: High availability using VRRP (L3HA) with Open vSwitch
    - Scenario: High availability using VRRP (L3HA) with Linux Bridge
    - Scenario: Provider networks with Open vSwitch
    - Scenario: Provider networks with Linux bridge
- Migration
    - Database migration
    - Migrate legacy nova-network to OpenStack Networking (neutron)
    - Legacy to DVR
    - Legacy to L3 HA
- Miscellaneous
    - Disabling libvirt networking
    - Adding high availability for DHCP
    - Neutron purge
- Advanced configuration
    - MTU considerations
    - Distributed Virtual Router SNAT HA configuration
    - Role-Based Access Control (RBAC)
    - Subnet pools
    - Address scopes
    - Native Open vSwitch firewall driver
    - Load Balancer as a Service (LBaaS)
    - FWaaS
    - VPNaaS
    - Service chaining
    - Using OpenStack Networking with QoS
    - Group policy
    - Debugging
    - Using OpenStack Networking with IPv6
    - Using SR-IOV functionality
    - IPAM configuration
    - Using availability zone
    - BGP dynamic routing
    - DNS integration in OpenStack Networking
    - Network IP Availability plug-in extension
    - Tagging resources

Appendix

- Community support

Glossary

- Glossary

Search in this guide

- *Search Page*

UPDATED: 2016-08-13 23:56

Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License. See all OpenStack Legal Documents.

FOUND AN ERROR? REPORT A BUG

QUESTIONS?

SEARCH

OpenStack Documentation

Contents

- Conventions
- Introduction to networking
- Introduction to OpenStack Networking (neutron)
- Configuration
- Deployment scenarios
- Migration
- Miscellaneous
- Advanced configuration
- Community support

- Glossary

OpenStack

Projects
OpenStack Security
Common Questions
Blog
News

Community

User Groups
Events
Jobs
Companies
Contribute

Documentation

OpenStack Manuals
Getting Started
API Documentation
Wiki

Branding & Legal

Logos & Guidelines
Trademark Policy
Privacy Policy
OpenStack CLA

Stay In Touch

The OpenStack project is provided under the Apache 2.0 license. Openstack.org is powered by Rackspace Cloud Computing.