openstack.

Install Guide

Release Version: 15.0.0

OpenStack contributors

Oct 11, 2017

CONTENTS

ABSTRACT

The OpenStack system consists of several key services that are separately installed. These services work together depending on your cloud needs and include the Compute, Identity, Networking, Image, Block Storage, Object Storage, Telemetry, Orchestration, and Database services. 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 42.2 and SUSE Linux Enterprise Server 12 - for both SP1 and SP2 - through the Open Build Service Cloud repository.

Explanations of configuration options and sample configuration files are included.

This guide documents the OpenStack Ocata release.

Conventions

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.

Overview

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 complementary services. Each service offers an *Application Programming Interface (API)* that facilitates this integration.

This guide covers step-by-step deployment of the major OpenStack services using a functional example architecture suitable for new users of OpenStack with sufficient Linux experience. This guide is not intended to be used for production system installations, but to create a minimum proof-of-concept for the purpose of learning about OpenStack.

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.

Important: The example architecture used in this guide is a minimum configuration, and is not intended for production system installations. It is designed to provide a minimum proof-of-concept for the purpose of learning about OpenStack. For information on creating architectures for specific use cases, or how to determine which architecture is required, see the Architecture Design Guide.

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, OpenStack Operations Guide, and OpenStack Networking Guide.

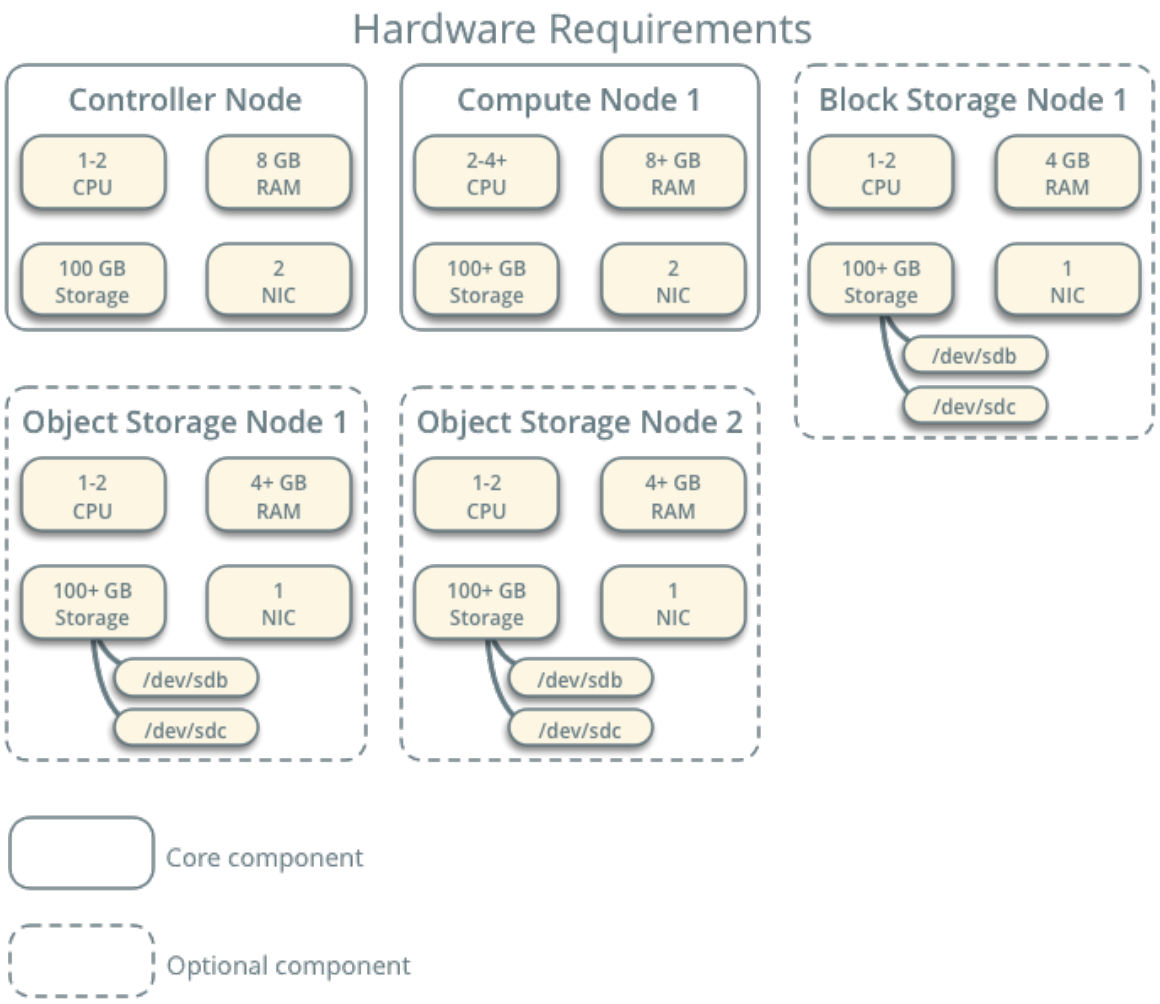


Fig. 1: **Hardware requirements**

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 the 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.

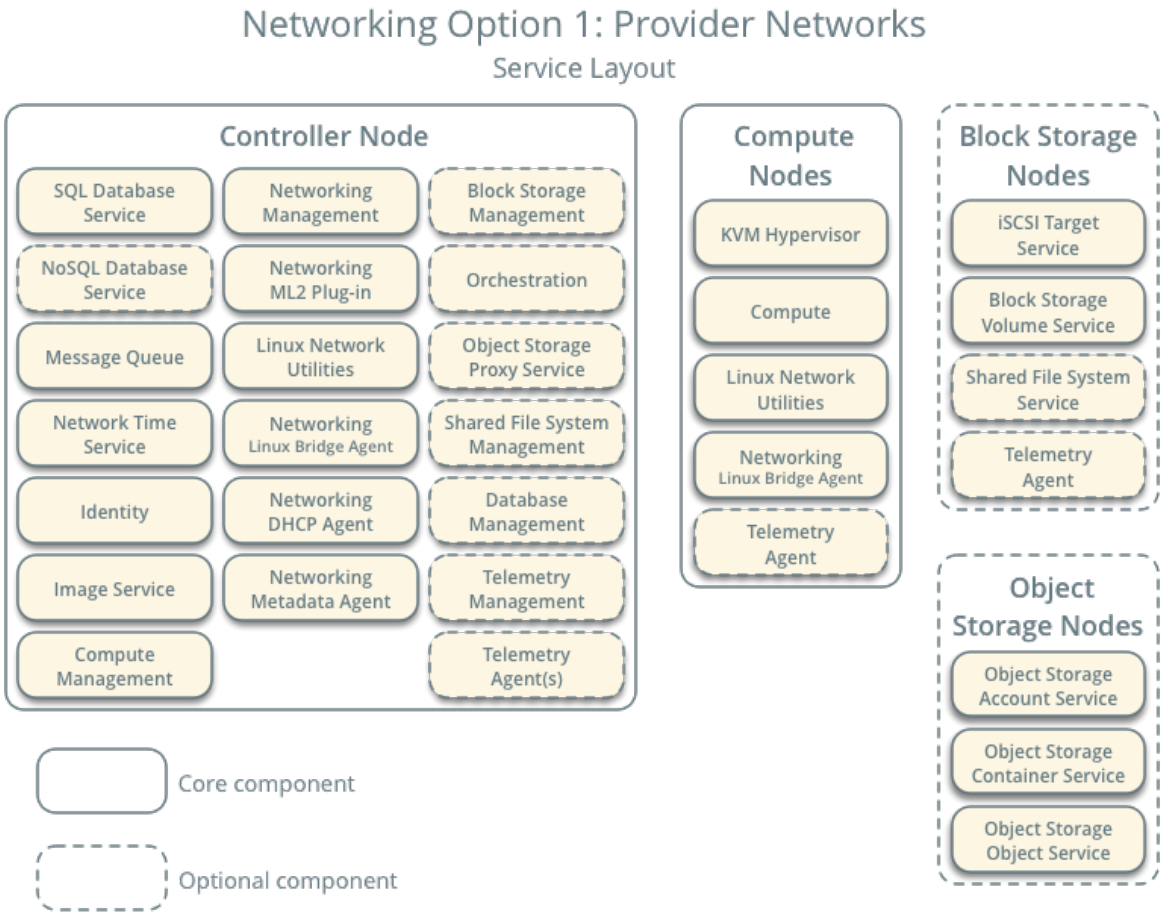
Networking

Choose one of the following virtual networking options.

Networking Option 1: Provider networks

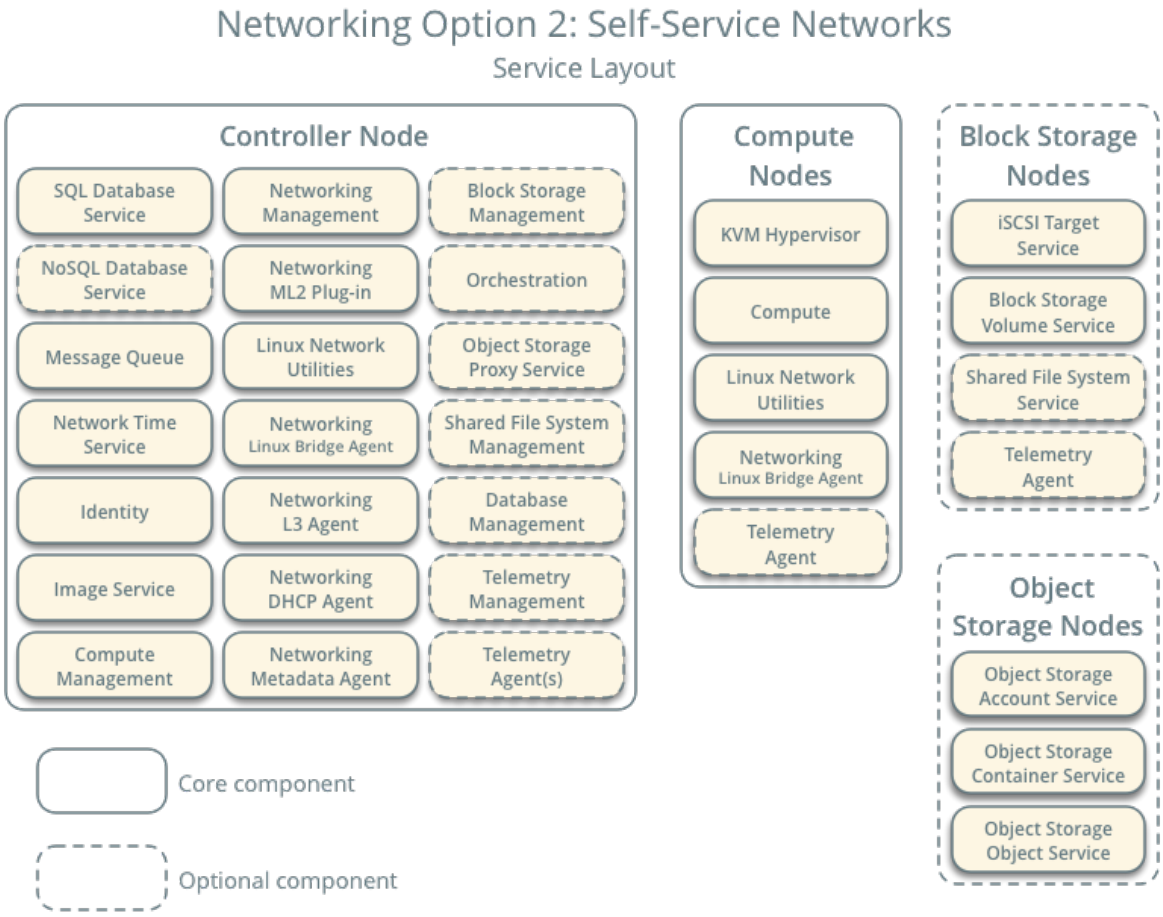
The provider networks option deploys the OpenStack Networking service in the simplest way possible with primarily layer-2 (bridging/switching) services and VLAN segmentation of networks. Essentially, it bridges virtual networks to physical networks and relies on physical network infrastructure for layer-3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

> **Warning:** 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 below 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.



Environment

This section explains how to configure the controller node 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 Installation Guide after configuring the appropriate nodes for it.

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.

The **`systemctl enable`** call on openSUSE outputs a warning message when the service uses SysV Init scripts instead of native systemd files. This warning can be ignored.

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 select 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

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

To ease the installation process, this guide only covers password security where applicable. You can create secure passwords manually, but the database connection string in services configuration file cannot accept special characters like "@". We recommend you 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.

Table 1: **Passwords**

Password name	Description
Database password (no variable used)	Root password for the database
`ADMIN_PASS`	Password of user `admin`
`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`
`KEYSTONE_DBPASS`	Database password of Identity service
`METADATA_SECRET`	Secret for the metadata proxy
`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`
`PLACEMENT_PASS`	Password of the Placement service user `placement`
`RABBIT_PASS`	Password of user guest of RabbitMQ

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 OpenStack Administrator Guide for more information.

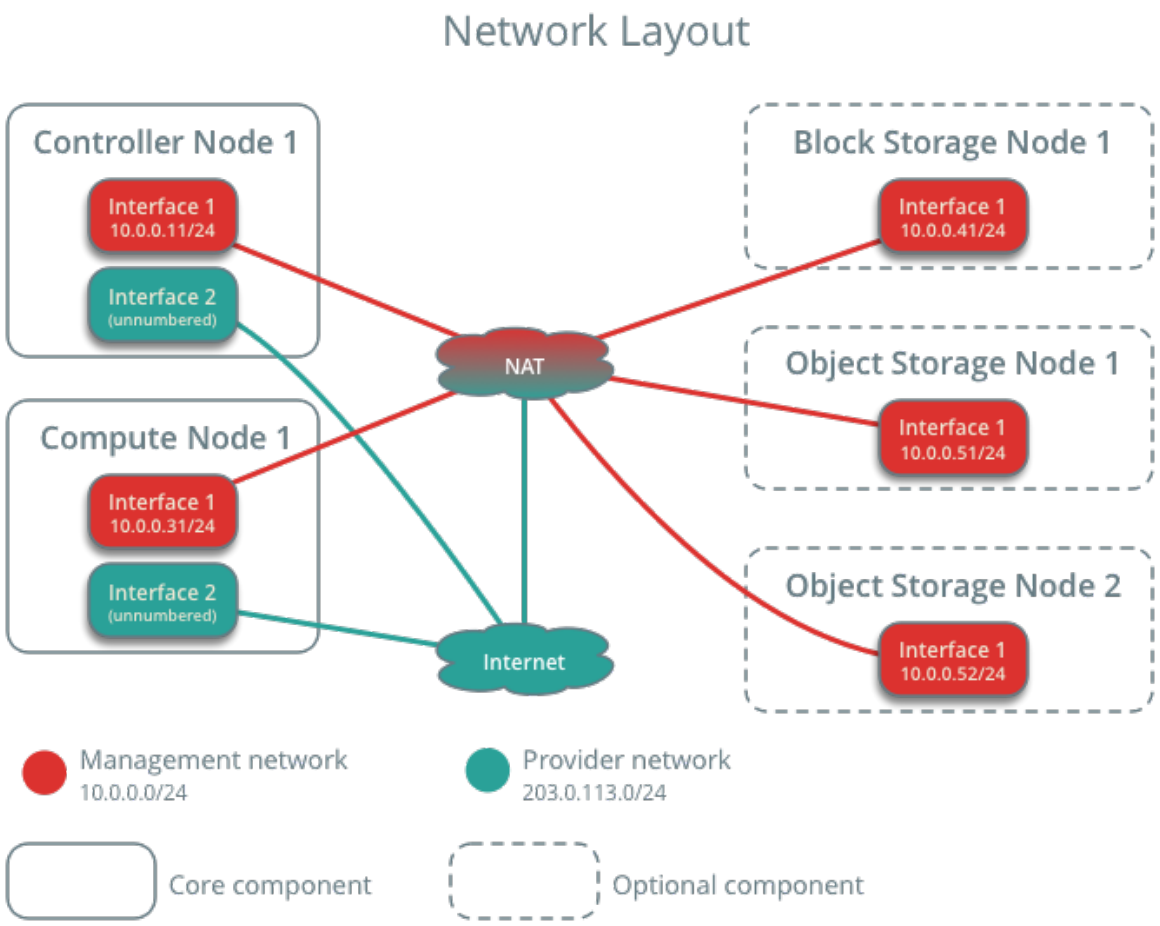
The Networking service assumes default values for kernel network parameters and modifies firewall rules. To avoid most issues during your initial installation, we recommend using a stock deployment of a supported distribution on your hosts. However, if you choose to automate deployment of your hosts,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

Host networking

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 SLES 12 or openSUSE 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 methods. 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 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. 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

Configure network interfaces

1. Configure the first interface as the management interface:

   IP address: 10.0.0.11

   Network mask: 255.255.255.0 (or /24)

   Default gateway: 10.0.0.1

2. The provider interface uses a special configuration without an IP address assigned to it. Configure the second interface as the provider interface:

   Replace `INTERFACE_NAME` with the actual interface name. For example, *eth1* or *ens224*.

   - Edit the `/etc/sysconfig/network/ifcfg-INTERFACE_NAME` file to contain the following:

     ```
     STARTMODE='auto'
     BOOTPROTO='static'
     ```

3. Reboot the system to activate the changes.

Configure name resolution

1. Set the hostname of the node to `controller`.
2. Edit the `/etc/hosts` file to contain the following:

   ```
   # controller
   10.0.0.11       controller

   # compute1
   10.0.0.31       compute1

   # block1
   10.0.0.41       block1

   # object1
   10.0.0.51       object1

   # object2
   10.0.0.52       object2
   ```

Warning: Some distributions add an extraneous entry in the `/etc/hosts` file that resolves the actual hostname to another loopback IP address such as `127.0.1.1`. You must comment out or remove this entry to prevent name resolution problems. **Do not remove the 127.0.0.1 entry.**

Note: This guide includes host entries for optional services in order to reduce complexity should you choose to deploy them.

Compute node

Configure network interfaces

1. Configure the first interface as the management interface:

   IP address: 10.0.0.31

   Network mask: 255.255.255.0 (or /24)

   Default gateway: 10.0.0.1

   Note: Additional compute nodes should use 10.0.0.32, 10.0.0.33, and so on.

2. The provider interface uses a special configuration without an IP address assigned to it. Configure the second interface as the provider interface:

   Replace `INTERFACE_NAME` with the actual interface name. For example, *eth1* or *ens224*.

   - Edit the `/etc/sysconfig/network/ifcfg-INTERFACE_NAME` file to contain the following:

     ```
     STARTMODE='auto'
     BOOTPROTO='static'
     ```

3. Reboot the system to activate the changes.

Configure name resolution

1. Set the hostname of the node to `compute1`.
2. Edit the `/etc/hosts` file to contain the following:

```
# controller
10.0.0.11       controller

# compute1
10.0.0.31       compute1

# block1
10.0.0.41       block1

# object1
10.0.0.51       object1
```

```
# object2
10.0.0.52       object2
```

Warning: Some distributions add an extraneous entry in the `/etc/hosts` file that resolves the actual hostname to another loopback IP address such as `127.0.1.1`. You must comment out or remove this entry to prevent name resolution problems. **Do not remove the 127.0.0.1 entry.**

Note: This guide includes host entries for optional services in order to reduce complexity should you choose to deploy them.

Block storage node (Optional)

If you want to deploy the Block Storage service, configure one additional storage node.

Configure network interfaces

- Configure the management interface:
  - IP address: `10.0.0.41`
  - Network mask: `255.255.255.0` (or `/24`)
  - Default gateway: `10.0.0.1`

Configure name resolution

1. Set the hostname of the node to `block1`.
2. Edit the `/etc/hosts` file to contain the following:

```
# controller
10.0.0.11       controller

# compute1
10.0.0.31       compute1

# block1
10.0.0.41       block1

# object1
10.0.0.51       object1

# object2
10.0.0.52       object2
```

Warning: Some distributions add an extraneous entry in the `/etc/hosts` file that resolves the actual hostname to another loopback IP address such as `127.0.1.1`. You must comment out or remove this entry to prevent name resolution problems. **Do not remove the 127.0.0.1 entry.**

Note: This guide includes host entries for optional services in order to reduce complexity should you choose to deploy them.

3. Reboot the system to activate the changes.

Verify connectivity

We recommend that you verify network connectivity to the Internet and among the nodes before proceeding further.

1. From the *controller* node, test access to the Internet:

```
# ping -c 4 openstack.org

PING openstack.org (174.143.194.225) 56(84) bytes of data.
64 bytes from 174.143.194.225: icmp_seq=1 ttl=54 time=18.3 ms
64 bytes from 174.143.194.225: icmp_seq=2 ttl=54 time=17.5 ms
64 bytes from 174.143.194.225: icmp_seq=3 ttl=54 time=17.5 ms
64 bytes from 174.143.194.225: icmp_seq=4 ttl=54 time=17.4 ms

--- openstack.org ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3022ms
rtt min/avg/max/mdev = 17.489/17.715/18.346/0.364 ms
```

2. From the *controller* node, test access to the management interface on the *compute* node:

```
# ping -c 4 compute1

PING compute1 (10.0.0.31) 56(84) bytes of data.
64 bytes from compute1 (10.0.0.31): icmp_seq=1 ttl=64 time=0.263 ms
64 bytes from compute1 (10.0.0.31): icmp_seq=2 ttl=64 time=0.202 ms
64 bytes from compute1 (10.0.0.31): icmp_seq=3 ttl=64 time=0.203 ms
64 bytes from compute1 (10.0.0.31): icmp_seq=4 ttl=64 time=0.202 ms

--- compute1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.202/0.217/0.263/0.030 ms
```

3. From the *compute* node, test access to the Internet:

```
# ping -c 4 openstack.org

PING openstack.org (174.143.194.225) 56(84) bytes of data.
64 bytes from 174.143.194.225: icmp_seq=1 ttl=54 time=18.3 ms
64 bytes from 174.143.194.225: icmp_seq=2 ttl=54 time=17.5 ms
64 bytes from 174.143.194.225: icmp_seq=3 ttl=54 time=17.5 ms
64 bytes from 174.143.194.225: icmp_seq=4 ttl=54 time=17.4 ms
```

```
--- openstack.org ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3022ms
rtt min/avg/max/mdev = 17.489/17.715/18.346/0.364 ms
```

4. From the *compute* node, test access to the management interface on the *controller* node:

```
# ping -c 4 controller

PING controller (10.0.0.11) 56(84) bytes of data.
64 bytes from controller (10.0.0.11): icmp_seq=1 ttl=64 time=0.263 ms
64 bytes from controller (10.0.0.11): icmp_seq=2 ttl=64 time=0.202 ms
64 bytes from controller (10.0.0.11): icmp_seq=3 ttl=64 time=0.203 ms
64 bytes from controller (10.0.0.11): icmp_seq=4 ttl=64 time=0.202 ms

--- controller ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.202/0.217/0.263/0.030 ms
```

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.

Network Time Protocol (NTP)

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

Perform these steps on the controller node.

Install and configure components

1. Install the packages:

```
# zypper install chrony
```

2. Edit the `/etc/chrony.conf` file and add, change, or remove these keys as necessary for your environment:

```
server NTP_SERVER iburst
```

Replace `NTP_SERVER` with the hostname or IP address of a suitable more accurate (lower stratum) NTP server. The configuration supports multiple `server` keys.

Note: By default, the controller node synchronizes the time via a pool of public servers. However, you can optionally configure alternative servers such as those provided by your organization.

---

3. To enable other nodes to connect to the chrony daemon on the controller node, add this key to the `/etc/chrony.conf` file:

```
allow 10.0.0.0/24
```

   If necessary, replace `10.0.0.0/24` with a description of your subnet.

4. Start the NTP service and configure it to start when the system boots:

```
# systemctl enable chronyd.service
# systemctl start chronyd.service
```

Other nodes

Other nodes reference the controller node for clock synchronization. Perform these steps on all other nodes.

Install and configure components

1. Install the packages:

```
# zypper install chrony
```

2. Edit the `/etc/chrony.conf` file and comment out or remove all but one `server` key. Change it to reference the controller node:

```
server controller iburst
```

3. Start the NTP service and configure it to start when the system boots:

```
# systemctl enable chronyd.service
# systemctl start chronyd.service
```

Verify operation

We recommend that you verify NTP synchronization before proceeding further. Some nodes, particularly those that reference the controller node, can take several minutes to synchronize.

1. Run this command on the *controller* node:

```
# chronyc sources

  210 Number of sources = 2
  MS Name/IP address         Stratum Poll Reach LastRx Last sample
  ===============================================================================
  ^- 192.0.2.11                    2   7    12   137  -2814us[-3000us] +/-   43ms
  ^* 192.0.2.12                    2   6   177    46    +17us[  -23us] +/-   68ms
```

   Contents in the *Name/IP address* column should indicate the hostname or IP address of one or more NTP servers. Contents in the *MS* column should indicate * for the server to which the NTP service is currently synchronized.

2. Run the same command on *all other* nodes:

```
# chronyc sources

  210 Number of sources = 1
  MS Name/IP address         Stratum Poll Reach LastRx Last sample
  ===============================================================================
  ^* controller                   3    9   377   421    +15us[  -87us] +/-   15ms
```

Contents in the *Name/IP address* column should indicate the hostname of the controller node.

OpenStack packages

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

Note: The set up of OpenStack packages described here needs to be done on all nodes: controller, compute, and Block Storage 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.

Enable the OpenStack repository

- Enable the Open Build Service repositories based on your openSUSE or SLES version:

  On openSUSE:

  ```
  # zypper addrepo -f obs://Cloud:OpenStack:Ocata/openSUSE_Leap_42.2 Ocata
  ```

  Note: The openSUSE distribution uses the concept of patterns to represent collections of packages. If you selected 'Minimal Server Selection (Text Mode)' during the initial installation, you may be presented with a dependency conflict when you attempt to install the OpenStack packages. To avoid this, remove the minimal_base-conflicts package:

  ```
  # zypper rm patterns-openSUSE-minimal_base-conflicts
  ```

  On SLES:

  ```
  # zypper addrepo -f obs://Cloud:OpenStack:Ocata/SLE_12_SP2 Ocata
  ```

  Note: The packages are signed by GPG key `D85F9316`. You should verify the fingerprint of the imported GPG key before using it.

```
Key Name:         Cloud:OpenStack OBS Project <Cloud:OpenStack@build.opensuse.org>
Key Fingerprint:  35B34E18 ABC1076D 66D5A86B 893A90DA D85F9316
Key Created:      2015-12-16T16:48:37 CET
Key Expires:      2018-02-23T16:48:37 CET
```

Finalize the installation

1. Upgrade the packages on all nodes:

```
# zypper refresh && zypper dist-upgrade
```

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

2. Install the OpenStack client:

```
# zypper install python-openstackclient
```

SQL database

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:

```
# zypper install mariadb-client mariadb python-PyMySQL
```

2. Create and edit the `/etc/my.cnf.d/openstack.cnf` file and complete the following actions:

   - Create a `[mysqld]` section, and set the `bind-address` key to the management IP address of the controller node to enable access by other nodes via the management network. Set additional keys to enable useful options and the UTF-8 character set:

```
[mysqld]
bind-address = 10.0.0.11

default-storage-engine = innodb
innodb_file_per_table = on
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 mysql.service
# systemctl start mysql.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
```

Message queue

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.

The message queue runs on the controller node.

Install and configure components

1. Install the package:

```
# zypper 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" ...
```

   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 "/" ...
```

Memcached

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:

```
# zypper install memcached python-python-memcached
```

2. Edit the `/etc/sysconfig/memcached` file and complete the following actions:

   - Configure the service to use the management IP address of the controller node. This is to enable access by other nodes via the management network:

```
MEMCACHED_PARAMS="-l 127.0.0.1"
```

   Note: Change the existing line `MEMCACHED_PARAMS="-l 127.0.0.1,::1"`.

Finalize installation

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

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

Identity service

Identity service overview

The OpenStack *Identity service* provides a single point of integration for managing authentication, authorization, and a catalog of services.

The Identity service is typically the first service a user interacts with. Once authenticated, an end user can use their identity to access other OpenStack services. Likewise, other OpenStack services leverage the Identity service to ensure users are who they say they are and discover where other services are within the deployment. The Identity service can also integrate with some external user management systems (such as LDAP).

Users and services can locate other services by using the service catalog, which is managed by the Identity service. As the name implies, a service catalog is a collection of available services in an OpenStack deployment. Each service can have one or many endpoints and each endpoint can be one of three types: admin, internal, or public. In a production environment, different endpoint types might reside on separate networks exposed to 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 types and the default `RegionOne` region. Together, regions, services, and endpoints created within the Identity service comprise the service catalog for a deployment. Each OpenStack service in your deployment needs a service entry with corresponding endpoints stored in the Identity service. This can all be done after the Identity service has been installed and configured.

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.

Install and configure

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

Prerequisites

Before you install and configure the Identity service, you must create a database.

Note: Before you begin, ensure you have the most recent version of `python-pyasn1` installed.

---

1. Use the database access client to connect to the database server as the `root` user:

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

2. Create the `keystone` database:

```
MariaDB [(none)]> CREATE DATABASE keystone;
```

3. Grant proper access to the `keystone` database:

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' \
IDENTIFIED BY 'KEYSTONE_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' \
IDENTIFIED BY 'KEYSTONE_DBPASS';
```

   Replace `KEYSTONE_DBPASS` with a suitable password.

4. Exit the database access client.

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.

Note: Starting with the Newton release, SUSE OpenStack packages are shipping with the upstream default configuration files. For example `/etc/keystone/keystone.conf`, with customizations in `/etc/keystone/keystone.conf.d/010-keystone.conf`. While the following instructions modify the default configuration file, adding a new file in `/etc/keystone/keystone.conf.d` achieves the same result.

1. Run the following command to install the packages:

```
# zypper install openstack-keystone apache2-mod_wsgi
```

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

   - 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.

     Note: Comment out or remove any other `connection` options in the `[database]` section.

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

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

3. Populate the Identity service database:

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

4. Initialize Fernet key repositories:

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

5. Bootstrap the Identity service:

```
# keystone-manage bootstrap --bootstrap-password ADMIN_PASS \
  --bootstrap-admin-url http://controller:35357/v3/ \
  --bootstrap-internal-url http://controller:5000/v3/ \
  --bootstrap-public-url http://controller:5000/v3/ \
  --bootstrap-region-id RegionOne
```

   Replace `ADMIN_PASS` with a suitable password for an administrative user.

Configure the Apache HTTP server

1. Edit the `/etc/sysconfig/apache2` file and configure the `APACHE_SERVERNAME` option to reference the controller node:

```
APACHE_SERVERNAME="controller"
```

2. Create the `/etc/apache2/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 display-name=%{GROUP}
    WSGIProcessGroup keystone-public
    WSGIScriptAlias / /usr/bin/keystone-wsgi-public
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    ErrorLogFormat "%{cu}t %M"
    ErrorLog /var/log/apache2/keystone.log
    CustomLog /var/log/apache2/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 display-name=%{GROUP}
    WSGIProcessGroup keystone-admin
    WSGIScriptAlias / /usr/bin/keystone-wsgi-admin
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    ErrorLogFormat "%{cu}t %M"
    ErrorLog /var/log/apache2/keystone.log
    CustomLog /var/log/apache2/keystone_access.log combined

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

3. Recursively change the ownership of the `/etc/keystone` directory:

```
# chown -R keystone:keystone /etc/keystone
```

Finalize the installation

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

```
# systemctl enable apache2.service
# systemctl start apache2.service
```

2. Configure the administrative account

```
$ export OS_USERNAME=admin
$ export OS_PASSWORD=ADMIN_PASS
$ export OS_PROJECT_NAME=admin
```

```
$ export OS_USER_DOMAIN_NAME=Default
$ export OS_PROJECT_DOMAIN_NAME=Default
$ export OS_AUTH_URL=http://controller:35357/v3
$ export OS_IDENTITY_API_VERSION=3
```

Replace `ADMIN_PASS` with the password used in the `keystone-manage bootstrap` command in *keystone-install-configure*.

Create a domain, projects, users, and roles

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

1. 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   | default                          |
| enabled     | True                             |
| id          | 24ac7f19cd944f4cba1d77469b2a73ed |
| is_domain   | False                            |
| name        | service                          |
| parent_id   | default                          |
+-------------+----------------------------------+
```

2. 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   | default                          |
| enabled     | True                             |
| id          | 231ad6e7ebba47d6a1e57e1cc07ae446 |
| is_domain   | False                            |
| name        | demo                             |
| parent_id   | default                          |
+-------------+----------------------------------+
```

   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           | default                          |
| enabled             | True                             |
| id                  | aeda23aa78f44e859900e22c24817832 |
| name                | demo                             |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

- Create the `user` role:

```
$ openstack role create user

+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | 997ce8d05fc143ac97d83fdfb5998552 |
| name      | user                             |
+-----------+----------------------------------+
```

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

```
$ 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.

Verify operation

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_AUTH_URL` and `OS_PASSWORD` environment variable:

```
$ unset OS_AUTH_URL OS_PASSWORD
```

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.

Create OpenStack client environment 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.

Note: The paths of the client environment scripts are unrestricted. For convenience, you can place the scripts in any location, however ensure that they are accessible.

---

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

   Note: The OpenStack client also supports using a `clouds.yaml` file. For more information, see the os-client-config.

   ```
   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. Create and 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    | ac33776331494012961f6c0d92d79dc16                                    |
+------------+----------------------------------------------------------------------+
```

Image service

Image service overview

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. Keep in mind that since the `file` back end is often local to a controller node, it is not typically suitable for a multi-node glance deployment.

For information on requirements for other back ends, see Configuration Reference.

The OpenStack Image service is central to Infrastructure-as-a-Service (IaaS) as shown in get_started_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 (or any filesystem mounted on the glance-api controller node), Object Storage, RADOS block devices, VMware datastore, and HTTP. 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.

Install and configure

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:

     ```
     MariaDB [(none)]> CREATE DATABASE glance;
     ```

   - Grant proper access to the `glance` database:

     ```
     MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \
       IDENTIFIED BY 'GLANCE_DBPASS';
     MariaDB [(none)]> 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           | default                          |
     ```

```
| enabled             | True                             |
| id                  | 3f4e777c4062483ab8d9edd7dff829df |
| name                | glance                           |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

- 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.

Note: Starting with the Newton release, SUSE OpenStack packages are shipping with the upstream default configuration files. For example `/etc/glance/glance-api.conf` or `/etc/glance/glance-registry.conf`, with customizations in `/etc/glance/glance-api.conf.d/` or `/etc/glance/glance-registry.conf.d/`. While the following instructions modify the default configuration files, adding new files in `/etc/glance/glance-api.conf.d` or `/etc/glance/glance-registry.conf.d` achieves the same result.

1. Install the packages:

```
# zypper install openstack-glance \
  openstack-glance-api openstack-glance-registry
```

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.

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
```

Verify operation

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/0.3.5/cirros-0.3.5-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-0.3.5-x86_64-disk.img \
  --disk-format qcow2 --container-format bare \
  --public

+------------------+------------------------------------------------------+
| Field            | 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 Create or update an image (glance) in the `OpenStack User Guide`.

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

Compute service

Compute service overview

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-placement-api service Tracks the inventory and usage of each provider. For details, see Placement API.

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-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.

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 SQLAlchemy supports. Common databases are SQLite3 for test and development work, MySQL, MariaDB, and PostgreSQL.

Install and configure controller node

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`, `nova`, and `nova_cell0` databases:

     ```
     MariaDB [(none)]> CREATE DATABASE nova_api;
     MariaDB [(none)]> CREATE DATABASE nova;
     MariaDB [(none)]> CREATE DATABASE nova_cell0;
     ```

   - Grant proper access to the databases:

     ```
     MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' \
       IDENTIFIED BY 'NOVA_DBPASS';
     MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' \
       IDENTIFIED BY 'NOVA_DBPASS';

     MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
       IDENTIFIED BY 'NOVA_DBPASS';
     MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
       IDENTIFIED BY 'NOVA_DBPASS';

     MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'localhost' \
       IDENTIFIED BY 'NOVA_DBPASS';
     MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* 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. Create the Compute service credentials:

   - Create the `nova` user:

```
$ openstack user create --domain default --password-prompt nova

User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 8a7dbf5279404537b1c7b86c033620fe |
| name                | nova                             |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

- 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 API service endpoints:

```
$ openstack endpoint create --region RegionOne \
  compute public http://controller:8774/v2.1

+--------------+------------------------------------------+
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | 3c1caa473bfe4390a11e7177894bcc7b         |
| interface    | public                                   |
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | 060d59eac51b4594815603d75a00aba2         |
| service_name | nova                                     |
| service_type | compute                                  |
| url          | http://controller:8774/v2.1              |
+--------------+------------------------------------------+
```

```
$ openstack endpoint create --region RegionOne \
  compute internal http://controller:8774/v2.1

+--------------+------------------------------------------+
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | e3c918de680746a586eac1f2d9bc10ab         |
| interface    | internal                                 |
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | 060d59eac51b4594815603d75a00aba2         |
| service_name | nova                                     |
| service_type | compute                                  |
| url          | http://controller:8774/v2.1              |
+--------------+------------------------------------------+

$ openstack endpoint create --region RegionOne \
  compute admin http://controller:8774/v2.1

+--------------+------------------------------------------+
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | 38f7af91666a47cfb97b4dc790b94424         |
| interface    | admin                                    |
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | 060d59eac51b4594815603d75a00aba2         |
| service_name | nova                                     |
| service_type | compute                                  |
| url          | http://controller:8774/v2.1              |
+--------------+------------------------------------------+
```

5. Create a Placement service user using your chosen `PLACEMENT_PASS`:

```
$ openstack user create --domain default --password-prompt placement

User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | fa742015a6494a949f67629884fc7ec8 |
| name                | placement                        |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

6. Add the Placement user to the service project with the admin role:

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

Note: This command provides no output.

7. Create the Placement API entry in the service catalog:

```
$ openstack service create --name placement --description "Placement API" placement
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Placement API                    |
| enabled     | True                             |
| id          | 2d1a27022e6e4185b86adac4444c495f |
| name        | placement                        |
| type        | placement                        |
+-------------+----------------------------------+
```

8. Create the Placement API service endpoints:

```
$ openstack endpoint create --region RegionOne placement public http://controller:8778
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 2b1b2637908b4137a9c2e0470487cbc0 |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 2d1a27022e6e4185b86adac4444c495f |
| service_name | placement                        |
| service_type | placement                        |
| url          | http://controller:8778           |
+--------------+----------------------------------+

$ openstack endpoint create --region RegionOne placement internal http://
↪controller:8778
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 02bcda9a150a4bd7993ff4879df971ab |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 2d1a27022e6e4185b86adac4444c495f |
| service_name | placement                        |
| service_type | placement                        |
| url          | http://controller:8778           |
+--------------+----------------------------------+

$ openstack endpoint create --region RegionOne placement admin http://controller:8778
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
```

```
| id           | 3d71177b9e0f406f98cbff198d74b182 |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 2d1a27022e6e4185b86adac4444c495f |
| service_name | placement                        |
| service_type | placement                        |
| url          | http://controller:8778           |
+--------------+----------------------------------+
```

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:

```
# zypper install openstack-nova-api openstack-nova-scheduler \
  openstack-nova-conductor openstack-nova-consoleauth \
  openstack-nova-novncproxy openstack-nova-placement-api \
  iptables
```

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]` section, configure `RabbitMQ` message queue access:

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

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

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

```
[api]
# ...
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]
enabled = true
# ...
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/run/nova
```

- In the `[placement]` section, configure the Placement API:

```
[placement]
# ...
os_region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:35357/v3
username = placement
password = PLACEMENT_PASS
```

Replace `PLACEMENT_PASS` with the password you choose for the `placement` user in the Identity service. Comment out any other options in the `[placement]` section.

3. Populate the nova-api database:

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

Note: Ignore any deprecation messages in this output.

4. Register the `cell0` database:

```
# su -s /bin/sh -c "nova-manage cell_v2 map_cell0" nova
```

5. Create the `cell1` cell:

```
# su -s /bin/sh -c "nova-manage cell_v2 create_cell --name=cell1 --verbose" nova
109e1d4b-536a-40d0-83c6-5f121b82b650
```

6. Populate the nova database:

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

7. Verify nova cell0 and cell1 are registered correctly:

```
# nova-manage cell_v2 list_cells
+-------+--------------------------------------+
| Name  | UUID                                 |
+-------+--------------------------------------+
| cell1 | 109e1d4b-536a-40d0-83c6-5f121b82b650 |
| cell0 | 00000000-0000-0000-0000-000000000000 |
+-------+--------------------------------------+
```

Finalize installation

- Enable the placement API Apache vhost:

```
# mv /etc/apache2/vhosts.d/nova-placement-api.conf.sample /etc/apache2/vhosts.d/nova-
↪placement-api.conf
# systemctl reload apache2.service
```

- 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
```

Install and configure a compute node

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:

```
# zypper install openstack-nova-compute genisoimage qemu-kvm libvirt
```

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 `[DEFAULT]` section, set the `compute_driver`:

```
[DEFAULT]
# ...
compute_driver = libvirt.LibvirtDriver
```

- In the `[DEFAULT]` section, configure `RabbitMQ` message queue access:

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

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

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

```
[api]
# ...
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/run/nova
```

- In the `[placement]` section, configure the Placement API:

```
[placement]
# ...
os_region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:35357/v3
username = placement
password = PLACEMENT_PASS
```

Replace `PLACEMENT_PASS` with the password you choose for the `placement` user in the Identity service. Comment out any other options in the `[placement]` section.

3. Ensure the kernel module `nbd` is loaded.

```
# modprobe nbd
```

4. Ensure the module loads on every boot by adding `nbd` to the `/etc/modules-load.d/nbd.conf` file.

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
```

Note: If the `nova-compute` service fails to start, check `/var/log/nova/nova-compute.log`. The error message `AMQP server on controller:5672 is unreachable` likely indicates that the firewall on the controller node is preventing access to port 5672. Configure the firewall to open port 5672 on the controller node and restart `nova-compute` service on the compute node.

Add the compute node to the cell database

Important: Run the following commands on the **controller** node.

---

1. Source the admin credentials to enable admin-only CLI commands, then confirm there are compute hosts in the database:

```
$ . admin-openrc

$ openstack hypervisor list
+----+---------------------+-----------------+-----------+-------+
| ID | Hypervisor Hostname | Hypervisor Type | Host IP   | State |
+----+---------------------+-----------------+-----------+-------+
|  1 | compute1            | QEMU            | 10.0.0.31 | up    |
+----+---------------------+-----------------+-----------+-------+
```

2. Discover compute hosts:

```
# su -s /bin/sh -c "nova-manage cell_v2 discover_hosts --verbose" nova

Found 2 cell mappings.
Skipping cell0 since it does not contain hosts.
Getting compute nodes from cell 'cell1': ad5a5985-a719-4567-98d8-8d148aaae4bc
Found 1 computes in cell: ad5a5985-a719-4567-98d8-8d148aaae4bc
Checking host mapping for compute host 'compute': fe58ddc1-1d65-4f87-9456-bc040dc106b3
Creating host mapping for compute host 'compute': fe58ddc1-1d65-4f87-9456-bc040dc106b3
```

Note: When you add new compute nodes, you must run `nova-manage cell_v2 discover_hosts`

on the controller node to register those new compute nodes. Alternatively, you can set an appropriate interval in `/etc/nova/nova.conf`:

```
[scheduler]
discover_hosts_in_cells_interval = 300
```

Verify operation

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.

3. List API endpoints in the Identity service to verify connectivity with the Identity service:

Note: Below endpoints list may differ depending on the installation of OpenStack components.

```
$ openstack catalog list

+-----------+-----------+-----------------------------------------+
| Name      | Type      | Endpoints                               |
+-----------+-----------+-----------------------------------------+
```

```
| keystone  | identity  | RegionOne                                |
|           |           |   public: http://controller:5000/v3/    |
|           |           | RegionOne                                |
|           |           |   internal: http://controller:5000/v3/   |
|           |           | RegionOne                                |
|           |           |   admin: http://controller:35357/v3/     |
|           |           |                                          |
| glance    | image     | RegionOne                                |
|           |           |   admin: http://controller:9292          |
|           |           | RegionOne                                |
|           |           |   public: http://controller:9292         |
|           |           | RegionOne                                |
|           |           |   internal: http://controller:9292       |
|           |           |                                          |
| nova      | compute   | RegionOne                                |
|           |           |   admin: http://controller:8774/v2.1     |
|           |           | RegionOne                                |
|           |           |   internal: http://controller:8774/v2.1  |
|           |           | RegionOne                                |
|           |           |   public: http://controller:8774/v2.1    |
|           |           |                                          |
| placement | placement | RegionOne                                |
|           |           |   public: http://controller:8778         |
|           |           | RegionOne                                |
|           |           |   admin: http://controller:8778          |
|           |           | RegionOne                                |
|           |           |   internal: http://controller:8778       |
|           |           |                                          |
+-----------+-----------+------------------------------------------+
```

Note: Ignore any warnings in this output.

4. List images in the Image service to verify connectivity with the Image service:

```
$ openstack image list

+--------------------------------------+-------------+-------------+
| ID                                   | Name        | Status      |
+--------------------------------------+-------------+-------------+
| 9a76d9f9-9620-4f2e-8c69-6c5691fae163 | cirros      | active      |
+--------------------------------------+-------------+-------------+
```

5. Check the cells and placement API are working successfully:

```
# nova-status upgrade check

+---------------------------+
| Upgrade Check Results     |
+---------------------------+
| Check: Cells v2           |
| Result: Success           |
| Details: None             |
+---------------------------+
| Check: Placement API      |
| Result: Success           |
```

```
| Details: None             |
+---------------------------+
| Check: Resource Providers |
| Result: Success           |
| Details: None             |
+---------------------------+
```

Networking service

Networking service overview

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 Plug and unplug ports, create networks or subnets, and provide 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.

Networking (neutron) concepts

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 projects 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.

Install and configure controller node

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:

     ```
     MariaDB [(none)] CREATE DATABASE neutron;
     ```

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

     ```
     MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' \
       IDENTIFIED BY 'NEUTRON_DBPASS';
     MariaDB [(none)]> 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           | default                          |
| enabled             | True                             |
| id                  | fdb0f541e28141719b6a43c8944bf1fb |
| name                | neutron                          |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

- 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

Install and configure the Networking components on the *controller* node.

Install the components

```
# zypper install --no-recommends openstack-neutron \
  openstack-neutron-server openstack-neutron-linuxbridge-agent \
  openstack-neutron-dhcp-agent openstack-neutron-metadata-agent \
  bridge-utils
```

Configure the server component

The Networking server component configuration includes the database, authentication mechanism, message queue, topology change notifications, 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, configure database access:

    ```
    [database]
    # ...
    connection = mysql+pymysql://neutron:NEUTRON_DBPASS@controller/neutron
    ```

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

    Note: Comment out or remove any other `connection` options in the `[database]` section.

  - In the `[DEFAULT]` section, enable the Modular Layer 2 (ML2) plug-in and disable additional plug-ins:

    ```
    [DEFAULT]
    # ...
    core_plugin = ml2
    service_plugins =
    ```

  - In the `[DEFAULT]` section, configure `RabbitMQ` message queue access:

    ```
    [DEFAULT]
    # ...
    transport_url = rabbit://openstack:RABBIT_PASS@controller
    ```

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 `[DEFAULT]` and `[nova]` sections, configure Networking to notify Compute of network topology changes:

```
[DEFAULT]
# ...
notify_nova_on_port_status_changes = true
notify_nova_on_port_data_changes = true

[nova]
# ...
auth_url = http://controller:35357
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = nova
password = NOVA_PASS
```

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

Configure the Modular Layer 2 (ML2) plug-in

The ML2 plug-in uses the Linux bridge mechanism to build layer-2 (bridging and switching) virtual networking infrastructure for instances.

- Edit the `/etc/neutron/plugins/ml2/ml2_conf.ini` file and complete the following actions:
  – In the `[ml2]` section, enable flat and VLAN networks:

```
[ml2]
# ...
type_drivers = flat,vlan
```

– In the `[ml2]` section, disable self-service networks:

```
[ml2]
# ...
tenant_network_types =
```

– In the `[ml2]` section, enable the Linux bridge mechanism:

```
[ml2]
# ...
mechanism_drivers = linuxbridge
```

> **Warning:** After you configure the ML2 plug-in, removing values in the `type_drivers` option can lead to database inconsistency.

– In the `[ml2]` section, enable the port security extension driver:

```
[ml2]
# ...
extension_drivers = port_security
```

– In the `[ml2_type_flat]` section, configure the provider virtual network as a flat network:

```
[ml2_type_flat]
# ...
flat_networks = provider
```

– In the `[securitygroup]` section, enable *ipset* to increase efficiency of security group rules:

```
[securitygroup]
# ...
enable_ipset = true
```

Configure the Linux bridge agent

The Linux bridge agent builds layer-2 (bridging and switching) virtual networking infrastructure for instances and handles security groups.

- Edit the `/etc/neutron/plugins/ml2/linuxbridge_agent.ini` file and complete the following actions:

  – In the `[linux_bridge]` section, map the provider virtual network to the provider physical network interface:

  ```
  [linux_bridge]
  physical_interface_mappings = provider:PROVIDER_INTERFACE_NAME
  ```

  Replace `PROVIDER_INTERFACE_NAME` with the name of the underlying provider physical network interface. See *Host networking* for more information.

– In the `[vxlan]` section, disable VXLAN overlay networks:

```
[vxlan]
enable_vxlan = false
```

– In the `[securitygroup]` section, enable security groups and configure the Linux bridge *iptables* firewall driver:

```
[securitygroup]
# ...
enable_security_group = true
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

Configure the DHCP agent

The *DHCP agent* provides DHCP services for virtual networks.

- Edit the `/etc/neutron/dhcp_agent.ini` file and complete the following actions:

  – In the `[DEFAULT]` section, configure the Linux bridge interface driver, Dnsmasq DHCP driver, and enable isolated metadata so instances on provider networks can access metadata over the network:

```
[DEFAULT]
# ...
interface_driver = linuxbridge
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
enable_isolated_metadata = true
```

Return to *Networking controller node configuration*.

Networking Option 2: Self-service networks

Install and configure the Networking components on the *controller* node.

Install the components

```
# zypper install --no-recommends openstack-neutron \
  openstack-neutron-server openstack-neutron-linuxbridge-agent \
  openstack-neutron-l3-agent openstack-neutron-dhcp-agent \
  openstack-neutron-metadata-agent bridge-utils
```

Configure the server component

- Edit the `/etc/neutron/neutron.conf` file and complete the following actions:

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

```
[database]
# ...
connection = mysql+pymysql://neutron:NEUTRON_DBPASS@controller/neutron
```

Replace NEUTRON_DBPASS with the password you chose for the database.

Note: Comment out or remove any other connection options in the [database] section.

– In the [DEFAULT] section, enable the Modular Layer 2 (ML2) plug-in, router service, and overlapping IP addresses:

```
[DEFAULT]
# ...
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = true
```

– In the [DEFAULT] section, configure RabbitMQ message queue access:

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

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 [DEFAULT] and [nova] sections, configure Networking to notify Compute of network topology changes:

```
[DEFAULT]
# ...
notify_nova_on_port_status_changes = true
notify_nova_on_port_data_changes = true

[nova]
# ...
auth_url = http://controller:35357
```

```
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = nova
password = NOVA_PASS
```

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

Configure the Modular Layer 2 (ML2) plug-in

The ML2 plug-in uses the Linux bridge mechanism to build layer-2 (bridging and switching) virtual networking infrastructure for instances.

- Edit the `/etc/neutron/plugins/ml2/ml2_conf.ini` file and complete the following actions:
  - In the `[ml2]` section, enable flat, VLAN, and VXLAN networks:

```
[ml2]
# ...
type_drivers = flat,vlan,vxlan
```

  - In the `[ml2]` section, enable VXLAN self-service networks:

```
[ml2]
# ...
tenant_network_types = vxlan
```

  - In the `[ml2]` section, enable the Linux bridge and layer-2 population mechanisms:

```
[ml2]
# ...
mechanism_drivers = linuxbridge,l2population
```

> **Warning:** After you configure the ML2 plug-in, removing values in the `type_drivers` option can lead to database inconsistency.

> **Note:** The Linux bridge agent only supports VXLAN overlay networks.

  - In the `[ml2]` section, enable the port security extension driver:

```
[ml2]
# ...
extension_drivers = port_security
```

  - In the `[ml2_type_flat]` section, configure the provider virtual network as a flat network:

```
[ml2_type_flat]
# ...
flat_networks = provider
```

- In the `[ml2_type_vxlan]` section, configure the VXLAN network identifier range for self-service networks:

```
[ml2_type_vxlan]
# ...
vni_ranges = 1:1000
```

- In the `[securitygroup]` section, enable *ipset* to increase efficiency of security group rules:

```
[securitygroup]
# ...
enable_ipset = true
```

Configure the Linux bridge agent

The Linux bridge agent builds layer-2 (bridging and switching) virtual networking infrastructure for instances and handles security groups.

- Edit the `/etc/neutron/plugins/ml2/linuxbridge_agent.ini` file and complete the following actions:
  - In the `[linux_bridge]` section, map the provider virtual network to the provider physical network interface:

```
[linux_bridge]
physical_interface_mappings = provider:PROVIDER_INTERFACE_NAME
```

    Replace `PROVIDER_INTERFACE_NAME` with the name of the underlying provider physical network interface. See *Host networking* for more information.

  - In the `[vxlan]` section, enable VXLAN overlay networks, configure the IP address of the physical network interface that handles overlay networks, and enable layer-2 population:

```
[vxlan]
enable_vxlan = true
local_ip = OVERLAY_INTERFACE_IP_ADDRESS
l2_population = true
```

    Replace `OVERLAY_INTERFACE_IP_ADDRESS` with the IP address of the underlying physical network interface that handles overlay networks. The example architecture uses the management interface to tunnel traffic to the other nodes. Therefore, replace `OVERLAY_INTERFACE_IP_ADDRESS` with the management IP address of the controller node. See *Host networking* for more information.

  - In the `[securitygroup]` section, enable security groups and configure the Linux bridge *iptables* firewall driver:

```
[securitygroup]
# ...
enable_security_group = true
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

Configure the layer-3 agent

The *Layer-3 (L3) agent* provides routing and NAT services for self-service virtual networks.

- Edit the `/etc/neutron/l3_agent.ini` file and complete the following actions:
  - In the `[DEFAULT]` section, configure the Linux bridge interface driver and external network bridge:

```
[DEFAULT]
# ...
interface_driver = linuxbridge
```

Configure the DHCP agent

The *DHCP agent* provides DHCP services for virtual networks.

- Edit the `/etc/neutron/dhcp_agent.ini` file and complete the following actions:
  - In the `[DEFAULT]` section, configure the Linux bridge interface driver, Dnsmasq DHCP driver, and enable isolated metadata so instances on provider networks can access metadata over the network:

```
[DEFAULT]
# ...
interface_driver = linuxbridge
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
enable_isolated_metadata = true
```

Return to *Networking controller node configuration*.

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 the Compute service to use the Networking service

- 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

Note: SLES enables apparmor by default and restricts dnsmasq. You need to either completely disable apparmor or disable only the dnsmasq profile:

```
# ln -s /etc/apparmor.d/usr.sbin.dnsmasq /etc/apparmor.d/disable/
# systemctl restart apparmor
```

---

1. Restart the Compute API service:

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

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

   For both networking options:

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

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

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

Install and configure compute node

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

Install the components

```
# zypper install --no-recommends \
  openstack-neutron-linuxbridge-agent bridge-utils
```

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]` section, configure `RabbitMQ` message queue access:

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

    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.

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 the Compute service to use the Networking service*.

Networking Option 1: Provider networks

Configure the Networking components on a *compute* node.

Configure the Linux bridge agent

The Linux bridge agent builds layer-2 (bridging and switching) virtual networking infrastructure for instances and handles security groups.

- Edit the `/etc/neutron/plugins/ml2/linuxbridge_agent.ini` file and complete the following actions:
  - In the `[linux_bridge]` section, map the provider virtual network to the provider physical network interface:

```
[linux_bridge]
physical_interface_mappings = provider:PROVIDER_INTERFACE_NAME
```

    Replace `PROVIDER_INTERFACE_NAME` with the name of the underlying provider physical network interface. See *Host networking* for more information.
  - In the `[vxlan]` section, disable VXLAN overlay networks:

```
[vxlan]
enable_vxlan = false
```

  - In the `[securitygroup]` section, enable security groups and configure the Linux bridge *iptables* firewall driver:

```
[securitygroup]
# ...
enable_security_group = true
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

Return to *Networking compute node configuration*.

Networking Option 2: Self-service networks

Configure the Networking components on a *compute* node.

Configure the Linux bridge agent

The Linux bridge agent builds layer-2 (bridging and switching) virtual networking infrastructure for instances and handles security groups.

- Edit the `/etc/neutron/plugins/ml2/linuxbridge_agent.ini` file and complete the following actions:
  - In the `[linux_bridge]` section, map the provider virtual network to the provider physical network interface:

```
[linux_bridge]
physical_interface_mappings = provider:PROVIDER_INTERFACE_NAME
```

Replace `PROVIDER_INTERFACE_NAME` with the name of the underlying provider physical network interface. See *Host networking* for more information.

– In the `[vxlan]` section, enable VXLAN overlay networks, configure the IP address of the physical network interface that handles overlay networks, and enable layer-2 population:

```
[vxlan]
enable_vxlan = true
local_ip = OVERLAY_INTERFACE_IP_ADDRESS
l2_population = true
```

Replace `OVERLAY_INTERFACE_IP_ADDRESS` with the IP address of the underlying physical network interface that handles overlay networks. The example architecture uses the management interface to tunnel traffic to the other nodes. Therefore, replace `OVERLAY_INTERFACE_IP_ADDRESS` with the management IP address of the compute node. See *Host networking* for more information.

– In the `[securitygroup]` section, enable security groups and configure the Linux bridge *iptables* firewall driver:

```
[securitygroup]
# ...
enable_security_group = true
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

Return to *Networking compute node configuration*.

Configure the Compute service to use the Networking service

- 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. The Networking service initialization scripts expect the variable `NEUTRON_PLUGIN_CONF` in the `/etc/sysconfig/neutron` file to reference the ML2 plug-in configuration file. Ensure that the `/etc/sysconfig/neutron` file contains the following:

```
NEUTRON_PLUGIN_CONF="/etc/neutron/plugins/ml2/ml2_conf.ini"
```

2. Restart the Compute service:

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

3. Start the Linux Bridge agent and configure it to start when the system boots:

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

Verify operation

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:

```
$ openstack extension list --network

+---------------------------+---------------------------+----------------------------+
| Name                      | Alias                     | Description                |
+---------------------------+---------------------------+----------------------------+
| Default Subnetpools       | default-subnetpools       | Provides ability to mark   |
|                           |                           | and use a subnetpool as    |
|                           |                           | the default                |
| Availability Zone         | availability_zone         | The availability zone      |
|                           |                           | extension.                 |
| Network Availability Zone | network_availability_zone | Availability zone support  |
|                           |                           | for network.               |
| Port Binding              | binding                   | Expose port bindings of a  |
|                           |                           | virtual port to external   |
|                           |                           | application                |
| agent                     | agent                     | The agent management       |
|                           |                           | extension.                 |
| Subnet Allocation         | subnet_allocation         | Enables allocation of      |
|                           |                           | subnets from a subnet pool |
| DHCP Agent Scheduler      | dhcp_agent_scheduler      | Schedule networks among    |
|                           |                           | dhcp agents                |
| Tag support               | tag                       | Enables to set tag on      |
|                           |                           | resources.                 |
| Neutron external network  | external-net              | Adds external network      |
|                           |                           | attribute to network       |
|                           |                           | resource.                  |
| Neutron Service Flavors   | flavors                   | Flavor specification for   |
|                           |                           | Neutron advanced services  |
| Network MTU               | net-mtu                   | Provides MTU attribute for |
|                           |                           | a network resource.        |
| Network IP Availability   | network-ip-availability   | Provides IP availability   |
|                           |                           | data for each network and  |
```

```
|                            |                            | subnet.                      |
| Quota management support   | quotas                     | Expose functions for         |
|                            |                            | quotas management per        |
|                            |                            | tenant                       |
| Provider Network           | provider                   | Expose mapping of virtual    |
|                            |                            | networks to physical         |
|                            |                            | networks                     |
| Multi Provider Network     | multi-provider             | Expose mapping of virtual    |
|                            |                            | networks to multiple         |
|                            |                            | physical networks            |
| Address scope              | address-scope              | Address scopes extension.    |
| Subnet service types       | subnet-service-types       | Provides ability to set      |
|                            |                            | the subnet service_types     |
|                            |                            | field                        |
| Resource timestamps        | standard-attr-timestamp    | Adds created_at and          |
|                            |                            | updated_at fields to all     |
|                            |                            | Neutron resources that       |
|                            |                            | have Neutron standard        |
|                            |                            | attributes.                  |
| Neutron Service Type       | service-type               | API for retrieving service   |
| Management                 |                            | providers for Neutron        |
|                            |                            | advanced services            |
| Tag support for            | tag-ext                    | Extends tag support to       |
| resources: subnet,         |                            | more L2 and L3 resources.    |
| subnetpool, port, router   |                            |                              |
| Neutron Extra DHCP opts    | extra_dhcp_opt             | Extra options                |
|                            |                            | configuration for DHCP.      |
|                            |                            | For example PXE boot         |
|                            |                            | options to DHCP clients      |
|                            |                            | can be specified (e.g.       |
|                            |                            | tftp-server, server-ip-      |
|                            |                            | address, bootfile-name)      |
| Resource revision numbers  | standard-attr-revisions    | This extension will          |
|                            |                            | display the revision         |
|                            |                            | number of neutron            |
|                            |                            | resources.                   |
| Pagination support         | pagination                 | Extension that indicates     |
|                            |                            | that pagination is           |
|                            |                            | enabled.                     |
| Sorting support            | sorting                    | Extension that indicates     |
|                            |                            | that sorting is enabled.     |
| security-group             | security-group             | The security groups          |
|                            |                            | extension.                   |
| RBAC Policies              | rbac-policies              | Allows creation and          |
|                            |                            | modification of policies     |
|                            |                            | that control tenant access   |
|                            |                            | to resources.                |
| standard-attr-description  | standard-attr-description  | Extension to add             |
|                            |                            | descriptions to standard     |
|                            |                            | attributes                   |
| Port Security              | port-security              | Provides port security       |
| Allowed Address Pairs      | allowed-address-pairs      | Provides allowed address     |
|                            |                            | pairs                        |
| project_id field enabled   | project-id                 | Extension that indicates     |
|                            |                            | that project_id field is     |
|                            |                            | enabled.                     |
+----------------------------+----------------------------+------------------------------+
```

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

- List agents to verify successful launch of the neutron agents:

```
$ openstack network agent list

+--------------------------------------+--------------------+------------+------------
↪-------+-------+-------+---------------------------+
| ID                                   | Agent Type         | Host       |
↪Availability Zone | Alive | State | Binary                    |
+--------------------------------------+--------------------+------------+------------
↪-------+-------+-------+---------------------------+
| 0400c2f6-4d3b-44bc-89fa-99093432f3bf | Metadata agent     | controller | None
↪       | True  | UP    | neutron-metadata-agent    |
| 83cf853d-a2f2-450a-99d7-e9c6fc08f4c3 | DHCP agent         | controller | nova
↪       | True  | UP    | neutron-dhcp-agent        |
| ec302e51-6101-43cf-9f19-88a78613cbee | Linux bridge agent | compute    | None
↪       | True  | UP    | neutron-linuxbridge-agent |
| fcb9bc6e-22b1-43bc-9054-272dd517d025 | Linux bridge agent | controller | None
↪       | True  | UP    | neutron-linuxbridge-agent |
+--------------------------------------+--------------------+------------+------------
↪-------+-------+-------+---------------------------+
```

The output should indicate three agents on the controller node and one agent on each compute node.

Networking Option 2: Self-service networks

- List agents to verify successful launch of the neutron agents:

```
$ openstack network agent list

+--------------------------------------+--------------------+------------+------------
↪-------+-------+-------+---------------------------+
| ID                                   | Agent Type         | Host       |
↪Availability Zone | Alive | State | Binary                    |
+--------------------------------------+--------------------+------------+------------
↪-------+-------+-------+---------------------------+
| f49a4b81-afd6-4b3d-b923-66c8f0517099 | Metadata agent     | controller | None
↪       | True  | UP    | neutron-metadata-agent    |
| 27eee952-a748-467b-bf71-941e89846a92 | Linux bridge agent | controller | None
↪       | True  | UP    | neutron-linuxbridge-agent |
| 08905043-5010-4b87-bba5-aedb1956e27a | Linux bridge agent | compute1   | None
↪       | True  | UP    | neutron-linuxbridge-agent |
| 830344ff-dc36-4956-84f4-067af667a0dc | L3 agent           | controller | nova
↪       | True  | UP    | neutron-l3-agent          |
| dd3644c9-1a3a-435a-9282-eb306b4b0391 | DHCP agent         | controller | nova
↪       | True  | UP    | neutron-dhcp-agent        |
+--------------------------------------+--------------------+------------+------------
↪-------+-------+-------+---------------------------+
```

The output should indicate four agents on the controller node and one agent on each compute node.

Next steps

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.

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 OpenStack Networking Guide.

Dashboard

Install and configure

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

The only core service required by the dashboard is the Identity service. You can use the dashboard in combination with other services, such as Image service, Compute, and Networking. You can also use the dashboard in environments with stand-alone services such as Object Storage.

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:

```
# zypper install openstack-dashboard
```

2. Configure the web server:

```
# cp /etc/apache2/conf.d/openstack-dashboard.conf.sample \
  /etc/apache2/conf.d/openstack-dashboard.conf
# a2enmod rewrite
```

3. Edit the `/srv/www/openstack-dashboard/openstack_dashboard/local/local_settings.py` file and complete the following actions:

   - Configure the dashboard to use OpenStack services on the `controller` node:

```
OPENSTACK_HOST = "controller"
```

- Allow your hosts to access the dashboard:

```
ALLOWED_HOSTS = ['one.example.com', 'two.example.com']
```

Note: `ALLOWED_HOSTS` can also be `['*']` to accept all hosts. This may be useful for development work, but is potentially insecure and should not be used in production. See Django documentation for further information.

- 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 apache2.service memcached.service
```

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

Verify operation

Verify operation of the dashboard.

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

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

Next steps

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 and configure 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.

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.

Block Storage service

Block Storage service overview

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.

Install and configure controller node

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:

     ```
     MariaDB [(none)]> CREATE DATABASE cinder;
     ```

   - Grant proper access to the `cinder` database:

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' \
  IDENTIFIED BY 'CINDER_DBPASS';
MariaDB [(none)]> 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           | default                          |
| enabled             | True                             |
| id                  | 9d7e33de3e1a498390353819bc7d245d |
| name                | cinder                           |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

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

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

Note: This command provides no output.

   - Create the `cinderv2` and `cinderv3` service entities:

```
$ openstack service create --name cinderv2 \
  --description "OpenStack Block Storage" volumev2

+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Block Storage          |
| enabled     | True                             |
| id          | eb9fd245bdbc414695952e93f29fe3ac |
| name        | cinderv2                         |
| type        | volumev2                         |
+-------------+----------------------------------+
```

```
$ openstack service create --name cinderv3 \
  --description "OpenStack Block Storage" volumev3
```

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Block Storage          |
| enabled     | True                             |
| id          | ab3bbbef780845a1a283490d281e7fda |
| name        | cinderv3                         |
| type        | volumev3                         |
+-------------+----------------------------------+
```

Note: The Block Storage services require two service entities.

4. Create the Block Storage service API endpoints:

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

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

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

```
$ openstack endpoint create --region RegionOne \
  volumev3 public http://controller:8776/v3/%\(project_id\)s

+--------------+------------------------------------------+
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | 03fa2c90153546c295bf30ca86b1344b         |
| interface    | public                                   |
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | ab3bbbef780845a1a283490d281e7fda         |
| service_name | cinderv3                                 |
| service_type | volumev3                                 |
| url          | http://controller:8776/v3/%(project_id)s |
+--------------+------------------------------------------+

$ openstack endpoint create --region RegionOne \
  volumev3 internal http://controller:8776/v3/%\(project_id\)s

+--------------+------------------------------------------+
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | 94f684395d1b41068c70e4ecb11364b2         |
| interface    | internal                                 |
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | ab3bbbef780845a1a283490d281e7fda         |
| service_name | cinderv3                                 |
| service_type | volumev3                                 |
| url          | http://controller:8776/v3/%(project_id)s |
+--------------+------------------------------------------+

$ openstack endpoint create --region RegionOne \
  volumev3 admin http://controller:8776/v3/%\(project_id\)s

+--------------+------------------------------------------+
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | 4511c28a0f9840c78bacb25f10f62c98         |
| interface    | admin                                    |
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | ab3bbbef780845a1a283490d281e7fda         |
```

```
| service_name | cinderv3                                  |
| service_type | volumev3                                  |
| url          | http://controller:8776/v3/%(project_id)s |
+--------------+-------------------------------------------+
```

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

Install and configure components

1. Install the packages:

```
# zypper install openstack-cinder-api openstack-cinder-scheduler
```

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]` section, configure `RabbitMQ` message queue access:

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

     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
```

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
```

Install and configure a storage node

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:

```
# zypper install lvm2
```

- (Optional) If you intend to use non-raw image types such as QCOW2 and VMDK, install the QEMU package:

```
# zypper install qemu
```

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-volumes` 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:

```
# zypper install openstack-cinder-volume tgt
```

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]` section, configure `RabbitMQ` message queue access:

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

     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 = tgtadm
```

- 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
```

3. Create the `/etc/tgt/conf.d/cinder.conf` file with the following data:

```
include /var/lib/cinder/volumes/*
```

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 tgtd.service
# systemctl start openstack-cinder-volume.service tgtd.service
```

Verify operation

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:

```
$ openstack volume service list

+------------------+------------+------+---------+-------+----------------------------
↪+
| Binary           | Host       | Zone | Status  | State | Updated_at                 
↪|
+------------------+------------+------+---------+-------+----------------------------
↪+
| cinder-scheduler | controller | nova | enabled | up    | 2016-09-30T02:27:41.000000
↪|
| cinder-volume    | block@lvm  | nova | enabled | up    | 2016-09-30T02:27:46.000000
↪|
+------------------+------------+------+---------+-------+----------------------------
↪+
```

Next steps

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

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 controller nodes, compute nodes, or standalone storage nodes.

For more information, see the Configuration Reference.

Additional services

Installation and configuration of additional OpenStack services is documented in separate, project-specific installation guides.

Bare Metal service (ironic)

The Bare Metal service is a collection of components that provides support to manage and provision physical machines.

Installation and configuration is documented in the Bare Metal installation guide.

Container Infrastructure Management service (magnum)

The Container Infrastructure Management service (magnum) is an OpenStack API service making container orchestration engines (COE) such as Docker Swarm, Kubernetes and Mesos available as first class resources in OpenStack.

Installation and configuration is documented in the Container Infrastructure Management installation guide.

Database service (trove)

The Database service (trove) provides cloud provisioning functionality for database engines.

Installation and configuration is documented in the Database installation guide.

DNS service (designate)

The DNS service (designate) provides cloud provisioning functionality for DNS Zones and Recordsets.

Installation and configuration is documented in the DNS installation guide.

Key Manager service (barbican)

The Key Manager service provides a RESTful API for the storage and provisioning of secret data such as passphrases, encryption keys, and X.509 certificates.

Installation and configuration is documented in the Key Manager installation guide.

Messaging service (zaqar)

The Messaging service allows developers to share data between distributed application components performing different tasks, without losing messages or requiring each component to be always available.

Installation and configuration is documented in the Messaging installation guide.

Object Storage services (swift)

The Object Storage services (swift) work together to provide object storage and retrieval through a REST API.

Installation and configuration is documented in the Object Storage installation guide.

Orchestration service (heat)

The Orchestration service (heat) uses a Heat Orchestration Template (HOT) to create and manage cloud resources.

Installation and configuration is documented in the Orchestration installation guide.

Shared File Systems service (manila)

The Shared File Systems service (manila) provides coordinated access to shared or distributed file systems.

Installation and configuration is documented in the Shared File Systems installation guide.

Telemetry Alarming services (aodh)

The Telemetry Alarming services trigger alarms when the collected metering or event data break the defined rules.

Installation and configuration is documented in the Telemetry Alarming installation guide.

Telemetry Data Collection service (ceilometer)

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.

Installation and configuration is documented in the Telemetry Data Collection installation guide.

Launch an instance

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. However, you can follow the instructions on any host that the tools are installed. 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

Before launching an instance, you must create the necessary virtual network infrastructure. For networking option 1, an instance uses a provider (external) network that connects to the physical network infrastructure via layer-2 (bridging/switching). This network includes a DHCP server that provides IP addresses to instances.

The `admin` or other privileged user must create this network because it connects directly to the physical network infrastructure.

Note: The following instructions and diagrams use example IP address ranges. You must adjust them for your particular environment.

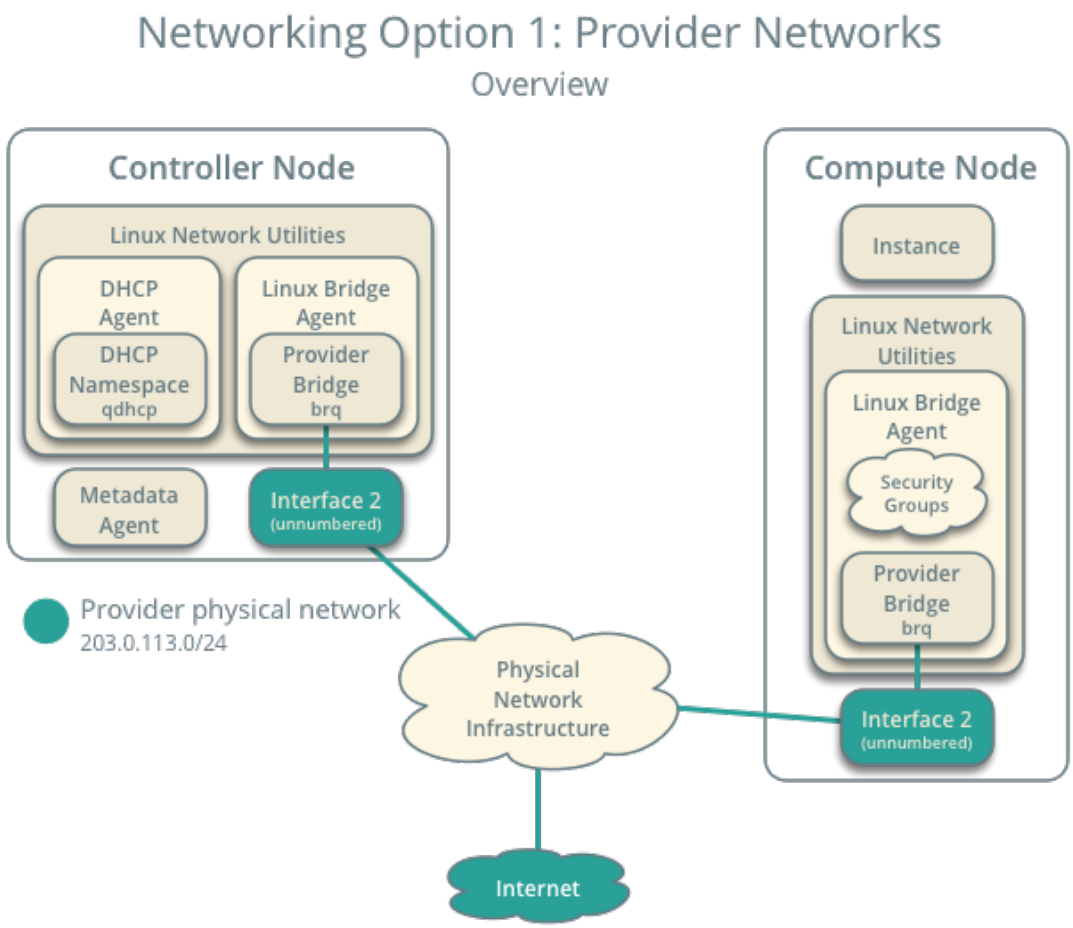


Fig. 2: **Networking Option 1: Provider networks - Overview**

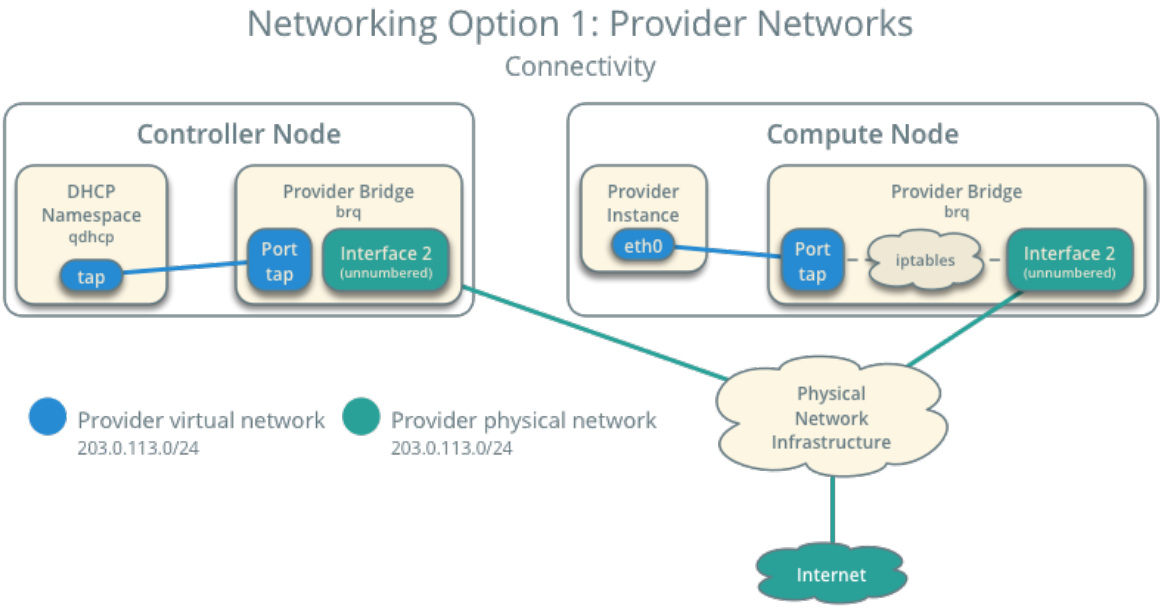


Fig. 3: **Networking Option 1: Provider networks - Connectivity**

Create the provider network

1. On the controller node, source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

2. Create the network:

```
$ openstack network create  --share --external \
  --provider-physical-network provider \
  --provider-network-type flat provider

Created a new network:

+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | UP                                   |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2017-03-14T14:37:39Z                 |
| description               |                                      |
| dns_domain                | None                                 |
| id                        | 54adb94a-4dce-437f-a33b-e7e2e7648173 |
| ipv4_address_scope        | None                                 |
| ipv6_address_scope        | None                                 |
| is_default                | None                                 |
| mtu                       | 1500                                 |
| name                      | provider                             |
| port_security_enabled     | True                                 |
| project_id                | 4c7f48f1da5b494faaa66713686a7707     |
| provider:network_type     | flat                                 |
| provider:physical_network | provider                             |
| provider:segmentation_id  | None                                 |
| qos_policy_id             | None                                 |
| revision_number           | 3                                    |
| router:external           | External                             |
| segments                  | None                                 |
| shared                    | True                                 |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| updated_at                | 2017-03-14T14:37:39Z                 |
+---------------------------+--------------------------------------+
```

The `--share` option allows all projects to use the virtual network.

The `--external` option defines the virtual network to be external. If you wish to create an internal network, you can use `--internal` instead. Default value is `internal`.

The `--provider-physical-network provider` and `--provider-network-type flat` options connect the flat virtual network to the flat (native/untagged) physical network on the `eth1` interface on the host using information from the following files:

`ml2_conf.ini`:

```
[ml2_type_flat]
flat_networks = provider
```

`linuxbridge_agent.ini`:

```
[linux_bridge]
physical_interface_mappings = provider:eth1
```

3. Create a subnet on the network:

```
$ openstack subnet create --network provider \
  --allocation-pool start=START_IP_ADDRESS,end=END_IP_ADDRESS \
  --dns-nameserver DNS_RESOLVER --gateway PROVIDER_NETWORK_GATEWAY \
  --subnet-range PROVIDER_NETWORK_CIDR provider
```

Replace `PROVIDER_NETWORK_CIDR` with the subnet on the provider physical network in CIDR notation.

Replace `START_IP_ADDRESS` and `END_IP_ADDRESS` with the first and last IP address of the range within the subnet that you want to allocate for instances. This range must not include any existing active IP addresses.

Replace `DNS_RESOLVER` with the IP address of a DNS resolver. In most cases, you can use one from the `/etc/resolv.conf` file on the host.

Replace `PROVIDER_NETWORK_GATEWAY` with the gateway IP address on the provider network, typically the ".1" IP address.

Example

The provider network uses 203.0.113.0/24 with a gateway on 203.0.113.1. A DHCP server assigns each instance an IP address from 203.0.113.101 to 203.0.113.250. All instances use 8.8.4.4 as a DNS resolver.

```
$ openstack subnet create --network provider \
  --allocation-pool start=203.0.113.101,end=203.0.113.250 \
  --dns-nameserver 8.8.4.4 --gateway 203.0.113.1 \
  --subnet-range 203.0.113.0/24 provider

Created a new subnet:
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| allocation_pools  | 203.0.113.101-203.0.113.250          |
| cidr              | 203.0.113.0/24                       |
| created_at        | 2016-11-02T20:45:04Z                 |
| description       |                                      |
| dns_nameservers   | 8.8.4.4                              |
| enable_dhcp       | True                                 |
| gateway_ip        | 203.0.113.1                          |
| headers           |                                      |
| host_routes       |                                      |
| id                | 2c65ef8c-a5f3-4f51-94c1-4df0daaaab5c |
| ip_version        | 4                                    |
| ipv6_address_mode | None                                 |
| ipv6_ra_mode      | None                                 |
| name              | provider                             |
| network_id        | 9793a02d-4f05-40d2-a280-407c48db0161 |
| project_id        | 7e188c33604d4b02ae0a99b5da68cae0     |
| revision_number   | 2                                    |
| service_types     | []                                   |
| subnetpool_id     | None                                 |
| updated_at        | 2016-11-02T20:45:04Z                 |
+-------------------+--------------------------------------+
```

Return to *Launch an instance - Create virtual networks*.

Self-service network

If you chose networking option 2, you can also create a self-service (private) network that connects to the physical network infrastructure via NAT. This network includes a DHCP server that provides IP addresses to instances. An instance on this network can automatically access external networks such as the Internet. However, access to an instance on this network from external networks such as the Internet requires a *floating IP address*.

The `demo` or other unprivileged user can create this network because it provides connectivity to instances within the `demo` project only.

Warning: You must *create the provider network* before the self-service network.

Note: The following instructions and diagrams use example IP address ranges. You must adjust them for your particular environment.

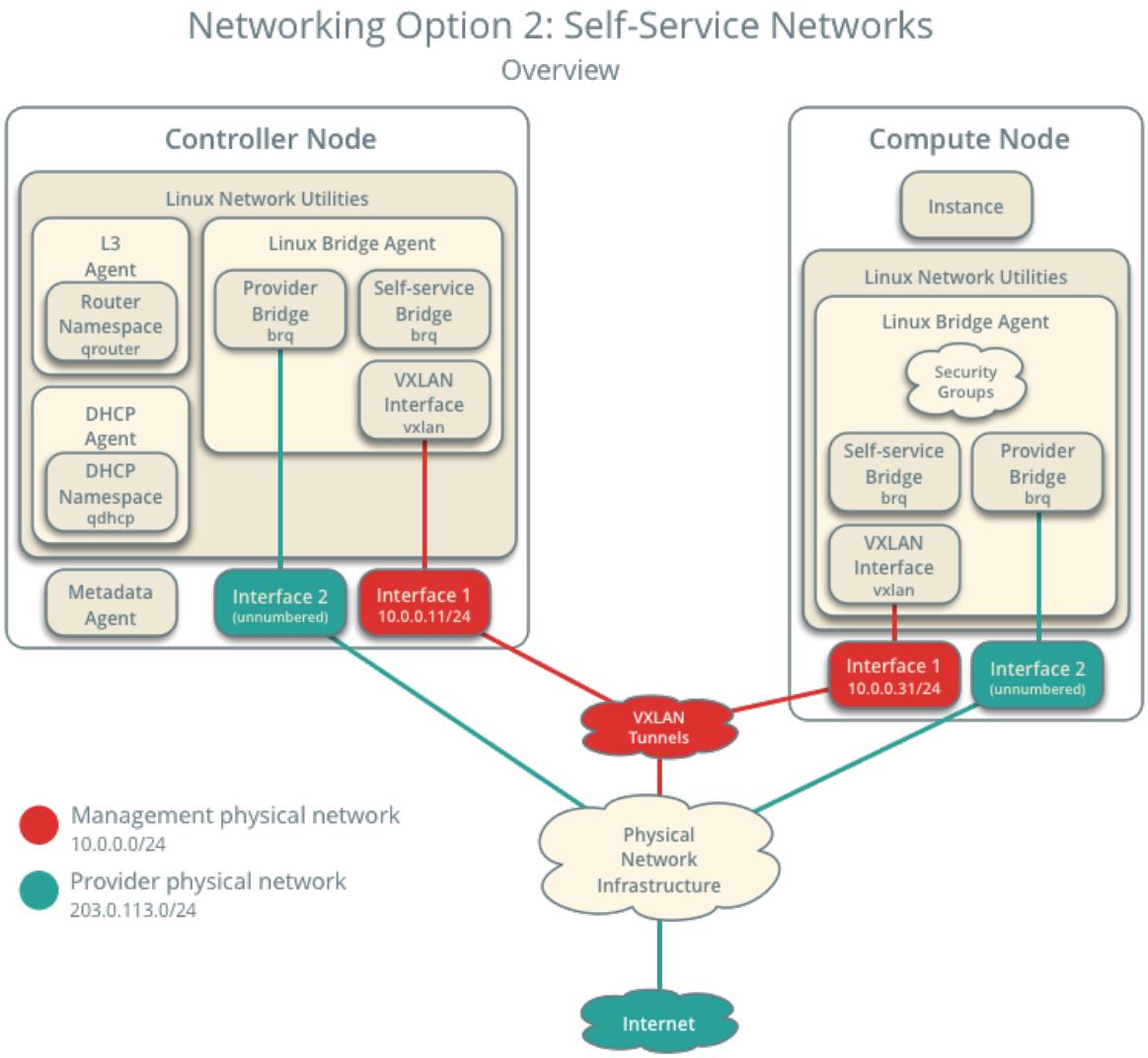


Fig. 4: **Networking Option 2: Self-service networks - Overview**

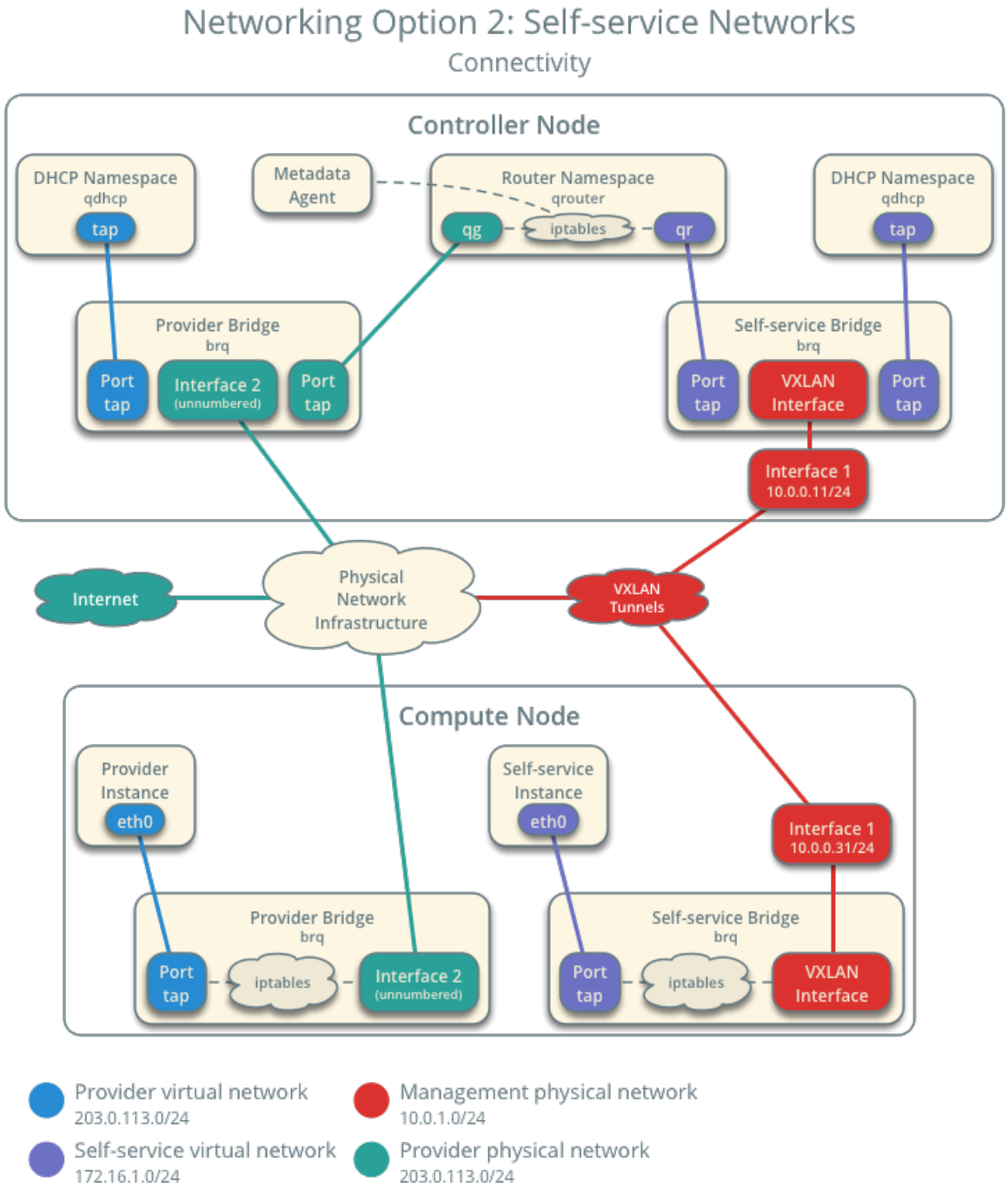


Fig. 5: **Networking Option 2: Self-service networks - Connectivity**

Create the self-service network

1. On the controller node, source the `demo` credentials to gain access to user-only CLI commands:

```
$ . demo-openrc
```

2. Create the network:

```
$ openstack network create selfservice

Created a new network:
+-------------------------+--------------------------------------+
| Field                   | Value                                |
+-------------------------+--------------------------------------+
| admin_state_up          | UP                                   |
| availability_zone_hints |                                      |
| availability_zones      |                                      |
| created_at              | 2016-11-04T18:20:59Z                 |
| description             |                                      |
| headers                 |                                      |
| id                      | 7c6f9b37-76b4-463e-98d8-27e5686ed083 |
| ipv4_address_scope      | None                                 |
| ipv6_address_scope      | None                                 |
| mtu                     | 1450                                 |
| name                    | selfservice                          |
| port_security_enabled   | True                                 |
| project_id              | 3828e7c22c5546e585f27b9eb5453788     |
| project_id              | 3828e7c22c5546e585f27b9eb5453788     |
| revision_number         | 3                                    |
| router:external         | Internal                             |
| shared                  | False                                |
| status                  | ACTIVE                               |
| subnets                 |                                      |
| tags                    | []                                   |
| updated_at              | 2016-11-04T18:20:59Z                 |
+-------------------------+--------------------------------------+
```

Non-privileged users typically cannot supply additional parameters to this command. The service automatically chooses parameters using information from the following files:

`ml2_conf.ini`:

```
[ml2]
tenant_network_types = vxlan

[ml2_type_vxlan]
vni_ranges = 1:1000
```

3. Create a subnet on the network:

```
$ openstack subnet create --network selfservice \
  --dns-nameserver DNS_RESOLVER --gateway SELFSERVICE_NETWORK_GATEWAY \
  --subnet-range SELFSERVICE_NETWORK_CIDR selfservice
```

Replace `DNS_RESOLVER` with the IP address of a DNS resolver. In most cases, you can use one from the `/etc/resolv.conf` file on the host.

Replace `SELFSERVICE_NETWORK_GATEWAY` with the gateway you want to use on the self-service network, typically the ".1" IP address.

Replace `SELFSERVICE_NETWORK_CIDR` with the subnet you want to use on the self-service network. You can use any arbitrary value, although we recommend a network from RFC 1918.

Example

The self-service network uses 172.16.1.0/24 with a gateway on 172.16.1.1. A DHCP server assigns each

instance an IP address from 172.16.1.2 to 172.16.1.254. All instances use 8.8.4.4 as a DNS resolver.

```
$ openstack subnet create --network selfservice \
  --dns-nameserver 8.8.4.4 --gateway 172.16.1.1 \
  --subnet-range 172.16.1.0/24 selfservice

Created a new subnet:
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| allocation_pools  | 172.16.1.2-172.16.1.254              |
| cidr              | 172.16.1.0/24                        |
| created_at        | 2016-11-04T18:30:54Z                 |
| description       |                                      |
| dns_nameservers   | 8.8.4.4                              |
| enable_dhcp       | True                                 |
| gateway_ip        | 172.16.1.1                           |
| headers           |                                      |
| host_routes       |                                      |
| id                | 5c37348e-e7da-439b-8c23-2af47d93aee5 |
| ip_version        | 4                                    |
| ipv6_address_mode | None                                 |
| ipv6_ra_mode      | None                                 |
| name              | selfservice                          |
| network_id        | b9273876-5946-4f02-a4da-838224a144e7 |
| project_id        | 3828e7c22c5546e585f27b9eb5453788     |
| project_id        | 3828e7c22c5546e585f27b9eb5453788     |
| revision_number   | 2                                    |
| service_types     | []                                   |
| subnetpool_id     | None                                 |
| updated_at        | 2016-11-04T18:30:54Z                 |
+-------------------+--------------------------------------+
```

Create a router

Self-service networks connect to provider networks using a virtual router that typically performs bidirectional NAT. Each router contains an interface on at least one self-service network and a gateway on a provider network.

The provider network must include the `router:external` option to enable self-service routers to use it for connectivity to external networks such as the Internet. The `admin` or other privileged user must include this option during network creation or add it later. In this case, the `router:external` option was set by using the `--external` parameter when creating the `provider` network.

1. On the controller node, source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

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

```
$ . demo-openrc
```

3. Create the router:

```
$ openstack router create router

Created a new router:
```

```
+-------------------------+--------------------------------------+
| Field                   | Value                                |
+-------------------------+--------------------------------------+
| admin_state_up          | UP                                   |
| availability_zone_hints |                                      |
| availability_zones      |                                      |
| created_at              | 2016-11-04T18:32:56Z                 |
| description             |                                      |
| external_gateway_info   | null                                 |
| flavor_id               | None                                 |
| headers                 |                                      |
| id                      | 67324374-396a-4db6-9443-c70be167a42b |
| name                    | router                               |
| project_id              | 3828e7c22c5546e585f27b9eb5453788     |
| project_id              | 3828e7c22c5546e585f27b9eb5453788     |
| revision_number         | 2                                    |
| routes                  |                                      |
| status                  | ACTIVE                               |
| updated_at              | 2016-11-04T18:32:56Z                 |
+-------------------------+--------------------------------------+
```

4. Add the self-service network subnet as an interface on the router:

```
$ neutron router-interface-add router selfservice

Added interface bff6605d-824c-41f9-b744-21d128fc86e1 to router router.
```

5. Set a gateway on the provider network on the router:

```
$ neutron router-gateway-set router provider

Set gateway for router router
```

Verify operation

We recommend that you verify operation and fix any issues before proceeding. The following steps use the IP address ranges from the network and subnet creation examples.

1. On the controller node, source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

2. List network namespaces. You should see one `qrouter` namespace and two `qdhcp` namespaces.

```
$ ip netns

qrouter-89dd2083-a160-4d75-ab3a-14239f01ea0b
qdhcp-7c6f9b37-76b4-463e-98d8-27e5686ed083
qdhcp-0e62efcd-8cee-46c7-b163-d8df05c3c5ad
```

3. List ports on the router to determine the gateway IP address on the provider network:

```
$ neutron router-port-list router

+--------------------------------------+------+-------------------+------------------
↪-------------------------+
```

```
| id                                   | name | mac_address       | fixed_ips
↪                       |
+--------------------------------------+------+-------------------+-------------------
↪----------------------+
| bff6605d-824c-41f9-b744-21d128fc86e1 |      | fa:16:3e:2f:34:9b | {"subnet_id":
↪                       |
|                                      |      |                   | "3482f524-8bff-
↪4871-80d4-5774c2730728",  |
|                                      |      |                   | "ip_address":
↪"172.16.1.1"}              |
| d6fe98db-ae01-42b0-a860-37b1661f5950 |      | fa:16:3e:e8:c1:41 | {"subnet_id":
↪                       |
|                                      |      |                   | "5cc70da8-4ee7-
↪4565-be53-b9c011fca011",  |
|                                      |      |                   | "ip_address":
↪"203.0.113.102"}           |
+--------------------------------------+------+-------------------+-------------------
↪----------------------+
```

4. Ping this IP address from the controller node or any host on the physical provider network:

```
$ ping -c 4 203.0.113.102

PING 203.0.113.102 (203.0.113.102) 56(84) bytes of data.
64 bytes from 203.0.113.102: icmp_req=1 ttl=64 time=0.619 ms
64 bytes from 203.0.113.102: icmp_req=2 ttl=64 time=0.189 ms
64 bytes from 203.0.113.102: icmp_req=3 ttl=64 time=0.165 ms
64 bytes from 203.0.113.102: icmp_req=4 ttl=64 time=0.216 ms

--- 203.0.113.102 ping statistics ---
rtt min/avg/max/mdev = 0.165/0.297/0.619/0.187 ms
```

Return to *Launch an instance - Create virtual networks*.

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` project credentials:

```
$ . demo-openrc
```

2. Generate a key pair and add a public key:

```
$ 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                                |
```

```
+-------------------+--------------------------------------+
| created_at        | 2016-10-05T09:52:31Z                 |
| description       |                                      |
| direction         | ingress                              |
| ethertype         | IPv4                                 |
| headers           |                                      |
| id                | 6ee8d630-9803-4d3d-9aea-8c795abbedc2 |
| port_range_max    | None                                 |
| port_range_min    | None                                 |
| project_id        | 77ae8d7104024123af342ffb0a6f1d88     |
| project_id        | 77ae8d7104024123af342ffb0a6f1d88     |
| protocol          | icmp                                 |
| remote_group_id   | None                                 |
| remote_ip_prefix  | 0.0.0.0/0                            |
| revision_number   | 1                                    |
| security_group_id | 4ceee3d4-d2fe-46c1-895c-382033e87b0d |
| updated_at        | 2016-10-05T09:52:31Z                 |
+-------------------+--------------------------------------+
```

– Permit secure shell (SSH) access:

```
$ openstack security group rule create --proto tcp --dst-port 22 default

+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| created_at        | 2016-10-05T09:54:50Z                 |
| description       |                                      |
| direction         | ingress                              |
| ethertype         | IPv4                                 |
| headers           |                                      |
| id                | 3cd0a406-43df-4741-ab29-b5e7dcb7469d |
| port_range_max    | 22                                   |
| port_range_min    | 22                                   |
| project_id        | 77ae8d7104024123af342ffb0a6f1d88     |
| project_id        | 77ae8d7104024123af342ffb0a6f1d88     |
| protocol          | tcp                                  |
| remote_group_id   | None                                 |
| remote_ip_prefix  | 0.0.0.0/0                            |
| revision_number   | 1                                    |
| security_group_id | 4ceee3d4-d2fe-46c1-895c-382033e87b0d |
| updated_at        | 2016-10-05T09:54:50Z                 |
+-------------------+--------------------------------------+
```

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

Determine instance options

To launch an instance, you must at least specify the flavor, image name, network, security group, key, and instance name.

1. On the controller node, source the `demo` credentials to gain access to user-only CLI commands:

```
$ . demo-openrc
```

2. A flavor specifies a virtual resource allocation profile which includes processor, memory, and storage.

   List available flavors:

```
$ openstack flavor list

+----+---------+-----+------+-----------+-------+-----------+
| ID | Name    | RAM | Disk | Ephemeral | VCPUs | Is Public |
+----+---------+-----+------+-----------+-------+-----------+
| 0  | m1.nano |  64 |    1 |         0 |     1 | True      |
+----+---------+-----+------+-----------+-------+-----------+
```

   Note: You can also reference a flavor by ID.

3. List available images:

```
$ openstack image list

+--------------------------------------+--------+--------+
| ID                                   | Name   | Status |
+--------------------------------------+--------+--------+
| 390eb5f7-8d49-41ec-95b7-68c0d5d54b34 | cirros | active |
+--------------------------------------+--------+--------+
```

   This instance uses the `cirros` image.

4. List available networks:

```
$ openstack network list

+--------------------------------------+-------------+--------------------------------
↪-------+
| ID                                   | Name        | Subnets                       
↪       |
+--------------------------------------+-------------+--------------------------------
↪-------+
| 4716ddfe-6e60-40e7-b2a8-42e57bf3c31c | selfservice | 2112d5eb-f9d6-45fd-906e-
↪7cabd38b7c7c |
| b5b6993c-ddf9-40e7-91d0-86806a42edb8 | provider    | 310911f6-acf0-4a47-824e-
↪3032916582ff |
+--------------------------------------+-------------+--------------------------------
↪-------+
```

This instance uses the `provider` provider network. However, you must reference this network using the ID instead of the name.

Note: If you chose option 2, the output should also contain the `selfservice` self-service network.

5. List available security groups:

```
$ openstack security group list

+--------------------------------------+---------+-----------------------+-----------
↪----------------------+
| ID                                   | Name    | Description           | Project
↪                      |
+--------------------------------------+---------+-----------------------+-----------
↪----------------------+
| dd2b614c-3dad-48ed-958b-b155a3b38515 | default | Default security group |
↪a516b957032844328896baa01e0f906c |
+--------------------------------------+---------+-----------------------+-----------
↪----------------------+
```

This instance uses the `default` security group.

Launch the instance

1. Launch the instance:

   Replace `PROVIDER_NET_ID` with the ID of the `provider` provider network.

Note: If you chose option 1 and your environment contains only one network, you can omit the `--nic` option because OpenStack automatically chooses the only network available.

```
$ openstack server create --flavor m1.nano --image cirros \
  --nic net-id=PROVIDER_NET_ID --security-group default \
  --key-name mykey provider-instance

+-------------------------------------+--------------------------------------------
↪-+
| Property                            | Value
↪ |
+-------------------------------------+--------------------------------------------
↪-+
| OS-DCF:diskConfig                   | MANUAL
↪ |
| OS-EXT-AZ:availability_zone         | nova
↪ |
| OS-EXT-STS:power_state              | 0
↪ |
| OS-EXT-STS:task_state               | scheduling
↪ |
| OS-EXT-STS:vm_state                 | building
↪ |
| OS-SRV-USG:launched_at              | -
↪ |
```

```
| OS-SRV-USG:terminated_at              | -                                                 
↪ |
| accessIPv4                            |                                                   
↪ |
| accessIPv6                            |                                                   
↪ |
| adminPass                             | hdF4LMQqC5PB                                      
↪ |
| config_drive                          |                                                   
↪ |
| created                               | 2015-09-17T21:58:18Z                              
↪ |
| flavor                                | m1.nano                                           
↪ |
| hostId                                |                                                   
↪ |
| id                                    | 181c52ba-aebc-4c32-a97d-2e8e82e4eaaf              
↪ |
| image                                 | cirros (38047887-61a7-41ea-9b49-
↪27987d5e8bb9) |
| key_name                              | mykey                                             
↪ |
| metadata                              | {}                                                
↪ |
| name                                  | provider-instance                                 
↪ |
| os-extended-volumes:volumes_attached | []                                                
↪ |
| progress                              | 0                                                 
↪ |
| security_groups                       | default                                           
↪ |
| status                                | BUILD                                             
↪ |
| tenant_id                             | f5b2ccaa75ac413591f12fcaa096aa5c                  
↪ |
| updated                               | 2015-09-17T21:58:18Z                              
↪ |
| user_id                               | 684286a9079845359882afc3aa5011fb                  
↪ |
+---------------------------------------+---------------------------------------------
↪-+
```

2. Check the status of your instance:

```
$ openstack server list

+--------------------------------------+------------------+--------+-----------------
↪-------+------------+
| ID                                   | Name             | Status | Networks        
↪       | Image Name |
+--------------------------------------+------------------+--------+-----------------
↪-------+------------+
| 181c52ba-aebc-4c32-a97d-2e8e82e4eaaf | provider-instance | ACTIVE | provider=203.0.
↪113.103 | cirros     |
+--------------------------------------+------------------+--------+-----------------
↪-------+------------+
```

The status changes from `BUILD` to `ACTIVE` when the build process successfully completes.

Access the instance using the virtual console

1. Obtain a *Virtual Network Computing (VNC)* session URL for your instance and access it from a web browser:

```
$ openstack console url show provider-instance

+-------+-------------------------------------------------------------------------------
↪----+
| Field | Value                                                                         
↪    |
+-------+-------------------------------------------------------------------------------
↪----+
| type  | novnc                                                                         
↪    |
| url   | http://controller:6080/vnc_auto.html?token=5eeccb47-525c-4918-ac2a-
↪3ad1e9f1f493 |
+-------+-------------------------------------------------------------------------------
↪----+
```

Note: If your web browser runs on a host that cannot resolve the `controller` host name, you can replace `controller` with the IP address of the management interface on your controller node.

The CirrOS image includes conventional user name/password authentication and provides these credentials at the login prompt. After logging into CirrOS, we recommend that you verify network connectivity using `ping`.

2. Verify access to the provider physical network gateway:

```
$ ping -c 4 203.0.113.1

PING 203.0.113.1 (203.0.113.1) 56(84) bytes of data.
64 bytes from 203.0.113.1: icmp_req=1 ttl=64 time=0.357 ms
64 bytes from 203.0.113.1: icmp_req=2 ttl=64 time=0.473 ms
64 bytes from 203.0.113.1: icmp_req=3 ttl=64 time=0.504 ms
64 bytes from 203.0.113.1: icmp_req=4 ttl=64 time=0.470 ms

--- 203.0.113.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2998ms
rtt min/avg/max/mdev = 0.357/0.451/0.504/0.055 ms
```

3. Verify access to the internet:

```
$ ping -c 4 openstack.org

PING openstack.org (174.143.194.225) 56(84) bytes of data.
64 bytes from 174.143.194.225: icmp_req=1 ttl=53 time=17.4 ms
64 bytes from 174.143.194.225: icmp_req=2 ttl=53 time=17.5 ms
64 bytes from 174.143.194.225: icmp_req=3 ttl=53 time=17.7 ms
64 bytes from 174.143.194.225: icmp_req=4 ttl=53 time=17.5 ms

--- openstack.org ping statistics ---
```

```
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 17.431/17.575/17.734/0.143 ms
```

Access the instance remotely

1. Verify connectivity to the instance from the controller node or any host on the provider physical network:

```
$ ping -c 4 203.0.113.103

PING 203.0.113.103 (203.0.113.103) 56(84) bytes of data.
64 bytes from 203.0.113.103: icmp_req=1 ttl=63 time=3.18 ms
64 bytes from 203.0.113.103: icmp_req=2 ttl=63 time=0.981 ms
64 bytes from 203.0.113.103: icmp_req=3 ttl=63 time=1.06 ms
64 bytes from 203.0.113.103: icmp_req=4 ttl=63 time=0.929 ms

--- 203.0.113.103 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 0.929/1.539/3.183/0.951 ms
```

2. Access your instance using SSH from the controller node or any host on the provider physical network:

```
$ ssh cirros@203.0.113.103

The authenticity of host '203.0.113.102 (203.0.113.102)' can't be established.
RSA key fingerprint is ed:05:e9:e7:52:a0:ff:83:68:94:c7:d1:f2:f8:e2:e9.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '203.0.113.102' (RSA) to the list of known hosts.
```

If your instance does not launch or seem to work as you expect, see the Instance Boot Failures section in OpenStack Operations Guide for more information or use one of the *many other options* to seek assistance. We want your first installation to work!

Return to *Launch an instance*.

Launch an instance on the self-service network

Determine instance options

To launch an instance, you must at least specify the flavor, image name, network, security group, key, and instance name.

1. On the controller node, source the `demo` credentials to gain access to user-only CLI commands:

```
$ . demo-openrc
```

2. A flavor specifies a virtual resource allocation profile which includes processor, memory, and storage.

   List available flavors:

```
$ openstack flavor list

+----+---------+-----+------+-----------+-------+-----------+
| ID | Name    | RAM | Disk | Ephemeral | VCPUs | Is Public |
```

```
+----+---------+-----+------+-----------+-------+-----------+
| 0  | m1.nano |  64 |    1 |         0 |     1 | True      |
+----+---------+-----+------+-----------+-------+-----------+
```

Note: You can also reference a flavor by ID.

3. List available images:

```
$ openstack image list

+--------------------------------------+--------+--------+
| ID                                   | Name   | Status |
+--------------------------------------+--------+--------+
| 390eb5f7-8d49-41ec-95b7-68c0d5d54b34 | cirros | active |
+--------------------------------------+--------+--------+
```

This instance uses the `cirros` image.

4. List available networks:

```
$ openstack network list

+--------------------------------------+-------------+--------------------------------
↪------+
| ID                                   | Name        | Subnets                        
↪      |
+--------------------------------------+-------------+--------------------------------
↪------+
| 4716ddfe-6e60-40e7-b2a8-42e57bf3c31c | selfservice | 2112d5eb-f9d6-45fd-906e-
↪7cabd38b7c7c |
| b5b6993c-ddf9-40e7-91d0-86806a42edb8 | provider    | 310911f6-acf0-4a47-824e-
↪3032916582ff |
+--------------------------------------+-------------+--------------------------------
↪------+
```

This instance uses the `selfservice` self-service network. However, you must reference this network using the ID instead of the name.

5. List available security groups:

```
$ openstack security group list

+--------------------------------------+---------+------------------------+
| ID                                   | Name    | Description            |
+--------------------------------------+---------+------------------------+
| dd2b614c-3dad-48ed-958b-b155a3b38515 | default | Default security group |
+--------------------------------------+---------+------------------------+
```

This instance uses the `default` security group.

6. Launch the instance:

Replace `SELFSERVICE_NET_ID` with the ID of the `selfservice` network.

```
$ openstack server create --flavor m1.nano --image cirros \
  --nic net-id=SELFSERVICE_NET_ID --security-group default \
```

```
  --key-name mykey selfservice-instance

+--------------------------------------+--------------------------------------+
| Field                                | Value                                |
+--------------------------------------+--------------------------------------+
| OS-DCF:diskConfig                    | MANUAL                               |
| OS-EXT-AZ:availability_zone          |                                      |
| OS-EXT-STS:power_state               | 0                                    |
| OS-EXT-STS:task_state                | scheduling                           |
| OS-EXT-STS:vm_state                  | building                             |
| OS-SRV-USG:launched_at               | None                                 |
| OS-SRV-USG:terminated_at             | None                                 |
| accessIPv4                           |                                      |
| accessIPv6                           |                                      |
| addresses                            |                                      |
| adminPass                            | 7KTBYHSjEz7E                         |
| config_drive                         |                                      |
| created                              | 2016-02-26T14:52:37Z                 |
| flavor                               | m1.nano                              |
| hostId                               |                                      |
| id                                   | 113c5892-e58e-4093-88c7-e33f502eaaa4 |
| image                                | cirros (390eb5f7-8d49-41ec-95b7-68c0d |
|                                      | 5d54b34)                             |
| key_name                             | mykey                                |
| name                                 | selfservice-instance                 |
| os-extended-volumes:volumes_attached | []                                   |
| progress                             | 0                                    |
| project_id                           | ed0b60bf607743088218b0a533d5943f     |
| properties                           |                                      |
| security_groups                      | [{u'name': u'default'}]              |
| status                               | BUILD                                |
| updated                              | 2016-02-26T14:52:38Z                 |
| user_id                              | 58126687cbcc4888bfa9ab73a2256f27     |
+--------------------------------------+--------------------------------------+
```

7. Check the status of your instance:

```
$ openstack server list

+--------------------------------------+----------------------+--------+--------------
↪----------+
| ID                                   | Name                 | Status | Networks
↪          |
+--------------------------------------+----------------------+--------+--------------
↪----------+
| 113c5892-e58e-4093-88c7-e33f502eaaa4 | selfservice-instance | ACTIVE |
↪selfservice=172.16.1.3 |
| 181c52ba-aebc-4c32-a97d-2e8e82e4eaaf | provider-instance    | ACTIVE | provider=203.
↪0.113.103 |
+--------------------------------------+----------------------+--------+--------------
↪----------+
```

The status changes from `BUILD` to `ACTIVE` when the build process successfully completes.

Access the instance using a virtual console

1. Obtain a *Virtual Network Computing (VNC)* session URL for your instance and access it from a web browser:

```
$ openstack console url show selfservice-instance

+-------+-------------------------------------------------------------------------------
↪----+
| Field | Value                                                                         
↪    |
+-------+-------------------------------------------------------------------------------
↪----+
| type  | novnc                                                                         
↪    |
| url   | http://controller:6080/vnc_auto.html?token=5eeccb47-525c-4918-ac2a-
↪3ad1e9f1f493 |
+-------+-------------------------------------------------------------------------------
↪----+
```

Note: If your web browser runs on a host that cannot resolve the `controller` host name, you can replace `controller` with the IP address of the management interface on your controller node.

The CirrOS image includes conventional user name/password authentication and provides these credentials at the login prompt. After logging into CirrOS, we recommend that you verify network connectivity using `ping`.

2. Verify access to the self-service network gateway:

```
$ ping -c 4 172.16.1.1

PING 172.16.1.1 (172.16.1.1) 56(84) bytes of data.
64 bytes from 172.16.1.1: icmp_req=1 ttl=64 time=0.357 ms
64 bytes from 172.16.1.1: icmp_req=2 ttl=64 time=0.473 ms
64 bytes from 172.16.1.1: icmp_req=3 ttl=64 time=0.504 ms
64 bytes from 172.16.1.1: icmp_req=4 ttl=64 time=0.470 ms

--- 172.16.1.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2998ms
rtt min/avg/max/mdev = 0.357/0.451/0.504/0.055 ms
```

3. Verify access to the internet:

```
$ ping -c 4 openstack.org

PING openstack.org (174.143.194.225) 56(84) bytes of data.
64 bytes from 174.143.194.225: icmp_req=1 ttl=53 time=17.4 ms
64 bytes from 174.143.194.225: icmp_req=2 ttl=53 time=17.5 ms
64 bytes from 174.143.194.225: icmp_req=3 ttl=53 time=17.7 ms
64 bytes from 174.143.194.225: icmp_req=4 ttl=53 time=17.5 ms

--- openstack.org ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 17.431/17.575/17.734/0.143 ms
```

Access the instance remotely

1. Create a *floating IP address* on the provider virtual network:

```
$ openstack floating ip create provider

+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| created_at          | 2017-01-20T17:29:16Z                 |
| description         |                                      |
| fixed_ip_address    | None                                 |
| floating_ip_address | 203.0.113.104                        |
| floating_network_id | b5b6993c-ddf9-40e7-91d0-86806a42edb8 |
| headers             |                                      |
| id                  | 88b4d06a-d794-4406-affd-6ffa2bcf1e2a |
| port_id             | None                                 |
| project_id          | ed0b60bf607743088218b0a533d5943f     |
| revision_number     | 1                                    |
| router_id           | None                                 |
| status              | DOWN                                 |
| updated_at          | 2017-01-20T17:29:16Z                 |
+---------------------+--------------------------------------+
```

2. Associate the floating IP address with the instance:

```
$ openstack server add floating ip selfservice-instance 203.0.113.104
```

Note: This command provides no output.

3. Check the status of your floating IP address:

```
$ openstack server list

+--------------------------------------+----------------------+--------+--------------
↪-------------------------+
| ID                                   | Name                 | Status | Networks
↪                         |
+--------------------------------------+----------------------+--------+--------------
↪-------------------------+
| 113c5892-e58e-4093-88c7-e33f502eaaa4 | selfservice-instance | ACTIVE |
↪selfservice=172.16.1.3, 203.0.113.104 |
| 181c52ba-aebc-4c32-a97d-2e8e82e4eaaf | provider-instance    | ACTIVE | provider=203.
↪0.113.103               |
+--------------------------------------+----------------------+--------+--------------
↪-------------------------+
```

4. Verify connectivity to the instance via floating IP address from the controller node or any host on the provider physical network:

```
$ ping -c 4 203.0.113.104

PING 203.0.113.104 (203.0.113.104) 56(84) bytes of data.
64 bytes from 203.0.113.104: icmp_req=1 ttl=63 time=3.18 ms
64 bytes from 203.0.113.104: icmp_req=2 ttl=63 time=0.981 ms
```

```
64 bytes from 203.0.113.104: icmp_req=3 ttl=63 time=1.06 ms
64 bytes from 203.0.113.104: icmp_req=4 ttl=63 time=0.929 ms

--- 203.0.113.104 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 0.929/1.539/3.183/0.951 ms
```

5. Access your instance using SSH from the controller node or any host on the provider physical network:

```
$ ssh cirros@203.0.113.104

The authenticity of host '203.0.113.104 (203.0.113.104)' can't be established.
RSA key fingerprint is ed:05:e9:e7:52:a0:ff:83:68:94:c7:d1:f2:f8:e2:e9.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '203.0.113.104' (RSA) to the list of known hosts.
```

If your instance does not launch or seem to work as you expect, see the Instance Boot Failures section in OpenStack Operations Guide for more information or use one of the *many other options* to seek assistance. We want your first installation to work!

Return to *Launch an instance*.

Block Storage

If your environment includes the Block Storage service, you can create a volume and attach it to an instance.

Block Storage

Create a volume

1. Source the `demo` credentials to perform the following steps as a non-administrative project:

```
$ . demo-openrc
```

2. Create a 1 GB volume:

```
$ openstack volume create --size 1 volume1

+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| attachments         | []                                   |
| availability_zone   | nova                                 |
| bootable            | false                                |
| consistencygroup_id | None                                 |
| created_at          | 2016-03-08T14:30:48.391027           |
| description         | None                                 |
| encrypted           | False                                |
| id                  | a1e8be72-a395-4a6f-8e07-856a57c39524 |
| multiattach         | False                                |
| name                | volume1                              |
| properties          |                                      |
| replication_status  | disabled                             |
| size                | 1                                    |
```

```
| snapshot_id         | None                                 |
| source_volid        | None                                 |
| status              | creating                             |
| type                | None                                 |
| updated_at          | None                                 |
| user_id             | 684286a9079845359882afc3aa5011fb     |
+---------------------+--------------------------------------+
```

3. After a short time, the volume status should change from `creating` to `available`:

```
$ openstack volume list

+--------------------------------------+--------------+-----------+------+-------------
↪-+
| ID                                   | Display Name | Status    | Size | Attached
↪to |
+--------------------------------------+--------------+-----------+------+-------------
↪-+
| a1e8be72-a395-4a6f-8e07-856a57c39524 | volume1      | available |    1 |
↪ |
+--------------------------------------+--------------+-----------+------+-------------
↪-+
```

Attach the volume to an instance

1. Attach a volume to an instance:

```
$ openstack server add volume INSTANCE_NAME VOLUME_NAME
```

Replace `INSTANCE_NAME` with the name of the instance and `VOLUME_NAME` with the name of the volume you want to attach to it.

Example

Attach the `volume1` volume to the `provider-instance` instance:

```
$ openstack server add volume provider-instance volume1
```

Note: This command provides no output.

---

2. List volumes:

```
$ openstack volume list

+--------------------------------------+--------------+--------+------+---------------
↪-----------------------------+
| ID                                   | Display Name | Status | Size | Attached to
↪                             |
+--------------------------------------+--------------+--------+------+---------------
↪-----------------------------+
| a1e8be72-a395-4a6f-8e07-856a57c39524 | volume1      | in-use |    1 | Attached to
↪provider-instance on /dev/vdb  |
+--------------------------------------+--------------+--------+------+---------------
↪-----------------------------+
```

3. Access your instance using SSH and use the `fdisk` command to verify presence of the volume as the `/dev/vdb` block storage device:

```
$ sudo fdisk -l

Disk /dev/vda: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders, total 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

   Device Boot      Start         End      Blocks   Id  System
/dev/vda1   *       16065     2088449     1036192+  83  Linux

Disk /dev/vdb: 1073 MB, 1073741824 bytes
16 heads, 63 sectors/track, 2080 cylinders, total 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

Disk /dev/vdb doesn't contain a valid partition table
```

Note: You must create a file system on the device and mount it to use the volume.

For more information about how to manage volumes, see the Manage volumes in OpenStack End User Guide.

Return to *Launch an instance*.

Orchestration

If your environment includes the Orchestration service, you can create a stack that launches an instance.

For more information, see the Orchestration installation guide.

Shared File Systems

If your environment includes the Shared File Systems service, you can create a share and mount it in an instance.

For more information, see the Shared File Systems installation guide.

Community support

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, join our IRC channel `#openstack-doc` on the freenode IRC network, or report a bug.

The following books explain how to install an OpenStack cloud and its associated components:

- Installation Tutorial for openSUSE Leap 42.2 and SUSE Linux Enterprise Server 12 SP2
- Installation Tutorial for Red Hat Enterprise Linux 7 and CentOS 7
- Installation Tutorial for Ubuntu 16.04 (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:

- End User Guide
- Command-Line Interface Reference

The following documentation provides reference and guidance information for the OpenStack APIs:

- API Guide

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 Ask OpenStack 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 the general OpenStack mailing list. 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. 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: Container Infrastructure Management 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), mIRC (Windows), 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 Paste. 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 on the IRC page on the wiki.

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

Glossary

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 (ACL) 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.

Active Directory Authentication and identity service by Microsoft, based on LDAP. Supported in OpenStack.

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/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.

Address Resolution Protocol (ARP) The protocol by which layer-3 IP addresses are resolved into layer-2 link local addresses.

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).

admin server In the context of the Identity service, the worker process that provides access to the admin API.

administrator The person responsible for installing, configuring, and managing an OpenStack cloud.

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 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.

aodh Part of the OpenStack *Telemetry service*; provides alarming functionality.

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 Catalog service (murano) The 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.

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 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.

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 CloudFormation template AWS CloudFormation allows Amazon Web Services (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, locally mounted file system, RADOS block devices, VMware datastore, and HTTP.

Backup, Restore, and Disaster Recovery service (freezer) The project that provides integrated tooling for backing up, restoring, and recovering file systems, instances, or database backups.

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 Manager service*.

bare An Image service container format that indicates that no container exists for the VM image.

Bare Metal service (ironic) The OpenStack service that provides a service and associated libraries capable of managing and provisioning physical machines in a security-aware and fault-tolerant manner.

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 divides 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 (rally) OpenStack project that provides a framework for performance analysis and benchmarking of individual OpenStack components as well as full production OpenStack cloud deployments.

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 API An API on a separate endpoint for attaching, detaching, and creating block storage for compute VMs.

Block Storage service (cinder) The OpenStack service that implement services and libraries to provide on-demand, self-service access to Block Storage resources via abstraction and automation on top of other block storage devices.

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

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.

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 Part of the OpenStack *Telemetry service*; gathers and stores metrics from other OpenStack services.

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 (CA) 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. In OpenStack, a simple certificate authority is 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 Codename for *Block Storage service*.

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 Auditing Data Federation (CADF) Cloud Auditing Data Federation (CADF) is a specification for audit event data. CADF is supported by OpenStack Identity.

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 (senlin) The project that implements clustering services and libraries for the management of groups of homogeneous objects exposed by other OpenStack services.

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.

Common Libraries (oslo) The project that produces a set of python libraries containing code shared by OpenStack projects. The APIs provided by these libraries should be high quality, stable, consistent, documented and generally applicable.

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 API (Nova 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 (nova) The OpenStack core project that implements services and associated libraries to provide massively-scalable, on-demand, self-service access to compute resources, including bare metal, virtual machines, and containers.

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.

congress Code name for the *Governance service*.

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 Infrastructure Management service (magnum) The project which provides a set of services for provisioning, scaling, and managing container orchestration engines.

container server An Object Storage server that manages containers.

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.

CRL A Certificate Revocation List (CRL) in a PKI model is a list of certificates that have been revoked. End entities presenting these certificates should not be trusted.

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 SUSE that aims to provide all necessary services to quickly deploy and manage 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 project.

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.

Dashboard (horizon) OpenStack project which provides an extensible, unified, web-based user interface for all OpenStack services.

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.

Data loss prevention (DLP) software Software programs used to protect sensitive information and prevent it from leaking outside a network boundary through the detection and denying of the data transportation.

Data Processing service (sahara) OpenStack project that provides a scalable data-processing stack and associated management interfaces.

data store A database engine supported by the Database service.

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 (trove) An integrated project that provides scalable and reliable Cloud Database-as-a-Service functionality for both relational and non-relational database engines.

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 dashboard.

default project New users are assigned to this project if no project is specified when a user is created.

default token An Identity service token that is not associated with a specific project 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*.

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 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.

Discretionary Access Control (DAC) 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.

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 record A record that specifies information about a particular domain and belongs to the domain.

DNS service (designate) OpenStack project that provides scalable, on demand, self service access to authoritative DNS services, in a technology-agnostic manner.

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.

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 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. 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.

encapsulation The practice of placing one packet type within another for the purposes of abstracting or securing data. Examples include GRE, MPLS, or IPsec.

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.

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.

FireWall-as-a-Service (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 project 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 Code name for the *Backup, Restore, and Disaster Recovery service*.

front end The point where a user interacts with a service; can be an API endpoint, the 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 Codename for the *Image service*.

glance API server Alternative name for the *Image API*.

glance registry Alternative term for the Image service *image registry*.

global endpoint template The Identity service endpoint template that contains services available to all projects.

GlusterFS A file system designed to aggregate NAS hosts, compatible with OpenStack.

gnocchi Part of the OpenStack *Telemetry service*; provides an indexer and time-series database.

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 (congress) The project that provides Governance-as-a-Service across any collection of cloud services in order to monitor, enforce, and audit policy over dynamic infrastructure.

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.

HAProxy Provides a high availability load balancer and proxy server for TCP and HTTP-based applications that spreads requests across multiple servers.

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.

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.

heat Codename for the *Orchestration service*.

Heat Orchestration Template (HOT) Heat input in the format native to OpenStack.

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 seek to minimize system downtime and data loss.

horizon Codename for the *Dashboard*.

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

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.

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 (keystone) The project that facilitates API client authentication, service discovery, distributed multi-project authorization, and auditing. It provides a central directory of users mapped to the OpenStack services they can access. It also registers endpoints for OpenStack services and acts as a common authentication system.

Identity service API The API used to access the OpenStack Identity service provided through keystone.

IETF Internet Engineering Task Force (IETF) is an open standards organization that develops Internet standards, particularly the standards pertaining to TCP/IP.

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. 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.

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 projects that can access a given VM image within Image service.

image owner The project who owns an Image service virtual machine image.

image registry A list of VM images that are available through Image service.

Image service (glance) The OpenStack service that provide services and associated libraries to store, browse, share, distribute and manage bootable disk images, other data closely associated with initializing compute resources, and metadata definitions.

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, locally mounted file system, RADOS block devices, VMware datastore, 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.

Infrastructure Optimization service (watcher) OpenStack project that aims to provide a flexible and scalable resource optimization service for multi-project OpenStack-based clouds.

Infrastructure-as-a-Service (IaaS) 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.

ingress filtering The process of filtering incoming network traffic. Supported by Compute.

INI format 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.

Input/Output Operations Per Second (IOPS) IOPS are a common performance measurement used to benchmark computer storage devices like hard disk drives, solid state drives, and storage area networks.

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.

Intelligent Platform Management Interface (IPMI) 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.

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 Control Message Protocol (ICMP) A network protocol used by network devices for control messages. For example, **`ping`** uses ICMP to test connectivity.

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. Supported by Compute, Object Storage, and Image service.

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.

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.

ironic Codename for the *Bare Metal service*.

iSCSI Qualified Name (IQN) 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'.

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.

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 (barbican) The project that produces a secret storage and generation system capable of providing key management for services wishing to enable encryption features.

keystone Codename of the *Identity service*.

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 kilogram 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 (L2) agent OpenStack Networking agent that provides layer-2 connectivity for virtual networks.

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 (L3) agent OpenStack Networking agent that provides layer-3 (routing) services for virtual networks.

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.

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 Unix-like computer operating system assembled under the model of free and open-source software development and distribution.

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.

Load-Balancer-as-a-Service (LBaaS) Enables Networking to distribute incoming requests evenly between designated instances.

Load-balancing service (octavia) The project that aims to rovide scalable, on demand, self service access to load-balancer services, in technology-agnostic manner.

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 *Containers Infrastructure Management 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 Codename for OpenStack *Shared File Systems service*.

manila-share Responsible for managing Shared File System Service devices, specifically the back-end devices.

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 project. Enables images to be shared with specified projects.

membership list A list of projects 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 (zaqar) The project that provides a 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.

Meta-Data Server (MDS) Stores CephFS metadata.

Metadata agent OpenStack Networking agent that provides metadata services for instances.

migration The process of moving a VM instance from one host to another.

mistral Code name for *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.

Modular Layer 2 (ML2) neutron plug-in Can concurrently use multiple layer-2 networking technologies, such as 802.1Q and VXLAN, in Networking.

monasca Codename for OpenStack *Monitoring*.

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 (monasca) The OpenStack service that provides a multi-project, highly scalable, performant, fault-tolerant monitoring-as-a-service solution for metrics, complex event processing and logging. To

build an extensible platform for advanced monitoring services that can be used by both operators and projects to gain operational insight and visibility, ensuring availability and stability.

multi-factor authentication Authentication method that uses two or more credentials, such as a password and a private key. Currently not supported in Identity.

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 Codename for the *Application Catalog service*.

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.

Network Address Translation (NAT) 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.

Network Service Header (NSH) Provides a mechanism for metadata exchange along the instantiated service path.

Network Time Protocol (NTP) 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 API (Neutron API) API used to access OpenStack Networking. Provides an extensible architecture to enable custom plug-in creation.

Networking service (neutron) The OpenStack project which implements services and associated libraries to provide on-demand, scalable, and technology-agnostic network abstraction.

neutron Codename for OpenStack *Networking service*.

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.

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.

Nexenta volume driver Provides support for NexentaStor devices in Compute.

NFV Orchestration Service (tacker) OpenStack service that aims to implement Network Function Virtualization (NFV) orchestration services and libraries for end-to-end life-cycle management of network services and Virtual Network Functions (VNFs).

Nginx An HTTP and reverse proxy server, a mail proxy server, and a generic TCP/UDP proxy server.

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 Codename for OpenStack *Compute service*.

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 Unique 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 API API used to access OpenStack *Object Storage*.

Object Storage Device (OSD) The Ceph storage daemon.

Object Storage service (swift) The OpenStack core project that provides eventually consistent and redundant storage and retrieval of fixed digital content.

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.

Octavia Code name for the *Load-balancing service*.

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, Mitaka, Newton, Ocata, Pike, and Queens. 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 (heat) The OpenStack service which orchestrates composite cloud applications using a declarative template format through an OpenStack-native REST API.

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 Codename for the *Common Libraries project*.

P

panko Part of the OpenStack *Telemetry service*; provides event storage.

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.

Pike The code name for the sixteenth release of OpenStack. The design summit will take place in Boston, Massachusetts, US. The release is named after the Massachusetts Turnpike, abbreviated commonly as the Mass Pike, which is the easternmost stretch of Interstate 90.

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.

policy-based routing (PBR) Provides a mechanism to implement packet forwarding and routing according to the policies defined by the network administrator.

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 projects.

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 Unique ID assigned to each project by the Identity service.

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 projects.

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.

Quality of Service (QoS) The ability to guarantee certain network or storage requirements to satisfy a Service Level Agreement (SLA) between an application provider and end users. Typically includes performance requirements like networking bandwidth, latency, jitter correction, and reliability as well as storage performance in Input/Output Operations Per Second (IOPS), throttling agreements, and performance expectations at peak load.

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.

Queens The code name for the seventeenth release of OpenStack. The design summit will take place in Sydney, Australia. The release is named after the Queens Pound river in the South Coast region of New South Wales.

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 Codename for 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 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 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 dashboard.

role ID Alphanumeric ID assigned to each Identity service role.

Root Cause Analysis (RCA) service (Vitrage) OpenStack project that aims to organize, analyze and visualize OpenStack alarms and events, yield insights regarding the root cause of problems and deduce their existence before they are directly detected.

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

sahara Codename for the *Data Processing service*.

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 project.

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 boot Process whereby the system firmware validates the authenticity of the code involved in the boot process.

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 Code name for the *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 Function Chain (SFC) For a given service, SFC is the abstracted view of the required service functions and the order in which they are to be applied.

service ID Unique ID assigned to each service that is available in the Identity service catalog.

Service Level Agreement (SLA) Contractual obligations that ensure the availability of a service.

service project Special project that contains all services that are listed in the 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 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 service*. 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 service* 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 (manila) The service that provides a set of services for management of shared file systems in a multi-project cloud environment, 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.

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.

Simple Protocol for Independent Computing Environments (SPICE) SPICE provides remote desktop access to guest virtual machines. It is an alternative to VNC. SPICE is supported 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.

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 (solum) 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.

Software-defined networking (SDN) Provides an approach for network administrators to manage computer network services through abstraction of lower-level functionality.

SolidFire Volume Driver The Block Storage driver for the SolidFire iSCSI storage appliance.

solum Code name for the *Software Development Lifecycle Automation service*.

spread-first scheduler The Compute VM scheduling algorithm that attempts to start a new VM on the host with the least amount of load.

SQLAlchemy 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 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 node An Object Storage node that provides container services, account services, and object services; controls the account databases, container databases, and object storage.

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 Codename for OpenStack *Object Storage service*.

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

tacker Code name for the *NFV Orchestration service*

Telemetry service (telemetry) The OpenStack project which collects measurements of the utilization of the physical and virtual resources comprising deployed clouds, persists this data for subsequent retrieval and analysis, and triggers actions when defined criteria are met.

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 projects.

tenant endpoint An Identity service API endpoint that is associated with one or more projects.

tenant ID An alternative term for *project ID*.

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 project 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 Codename for OpenStack *Database service*.

trusted platform module (TPM) Specialized microprocessor for incorporating cryptographic keys into devices for authenticating and securing a hardware platform.

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.

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.

Virtual Extensible LAN (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 address (VIP) 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 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 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 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.

Vitrage Code name for the *Root Cause Analysis service*.

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

Watcher Code name for the *Infrastructure Optimization service*.

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.

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.

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 (mistral) The OpenStack service 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.

X

X.509 X.509 is the most widely used standard for defining digital certificates. It is a data structure that contains the subject (entity) identifiable information such as its name along with its public key. The certificate can contain a few other attributes as well depending upon the version. The most recent and standard version of X.509 is v3.

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 Codename for the *Message service*.

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.

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

K

P

Q

R

S

T

U

V

W

X

Z