OPENSTACK INSTALLATION GUIDE

ABSTRACT
Installation guide of OpenStack based on OpenStack web site documents and supplemental review

Davoud Mostafavi Amjad
1398/08/24

System preferences that we used to install Queens version of OpenStack

<table>
<tr><td rowspan="2">Virtualization technology</td><td colspan="2">XenServer 7.2

User/password
root/Dm1358!</td><td colspan="2">Core i7 (16 cores)-16GB Ram
256 SSD HDD</td></tr>
<tr><td colspan="4">http://dl.hellodigi.ir/dl.hellodigi.ir/dl/citrix/XenServer-7.2.0-install-cd.iso</td></tr>
<tr><td rowspan="2">Operating systems</td><td>CentOs 7</td><td>Controller :
6 cores- 6GB Ram- 128GB HDD

User/password
user/123321
root/123</td><td>Compute :
4 cores-4GB Ram- 128GB HDD

User/password
user/123321
root/123</td><td>Ubuntu 16.04:
4 cores-4GB Ram

User/password
user/123321
root/123</td></tr>
<tr><td colspan="3">https://www.centos.org/download/</td><td>https://ubuntu.com/download/desktop</td></tr>
<tr><td rowspan="2">Openstack</td><td colspan="4">All passwords set to : 123321
Except demo user that have Dm1358! Password</td></tr>
<tr><td colspan="4">https://docs.openstack.org/install-guide/index.html</td></tr>
</table>

Environment

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.

Security

THIS PAGE LAST UPDATED: 2019-11-11 12:17:29

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.

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 RabbitMQ user **openstack**

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 Compute service documentation for Pike, the Compute service documentation for Queens, or the Compute service documentation for Rocky 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.

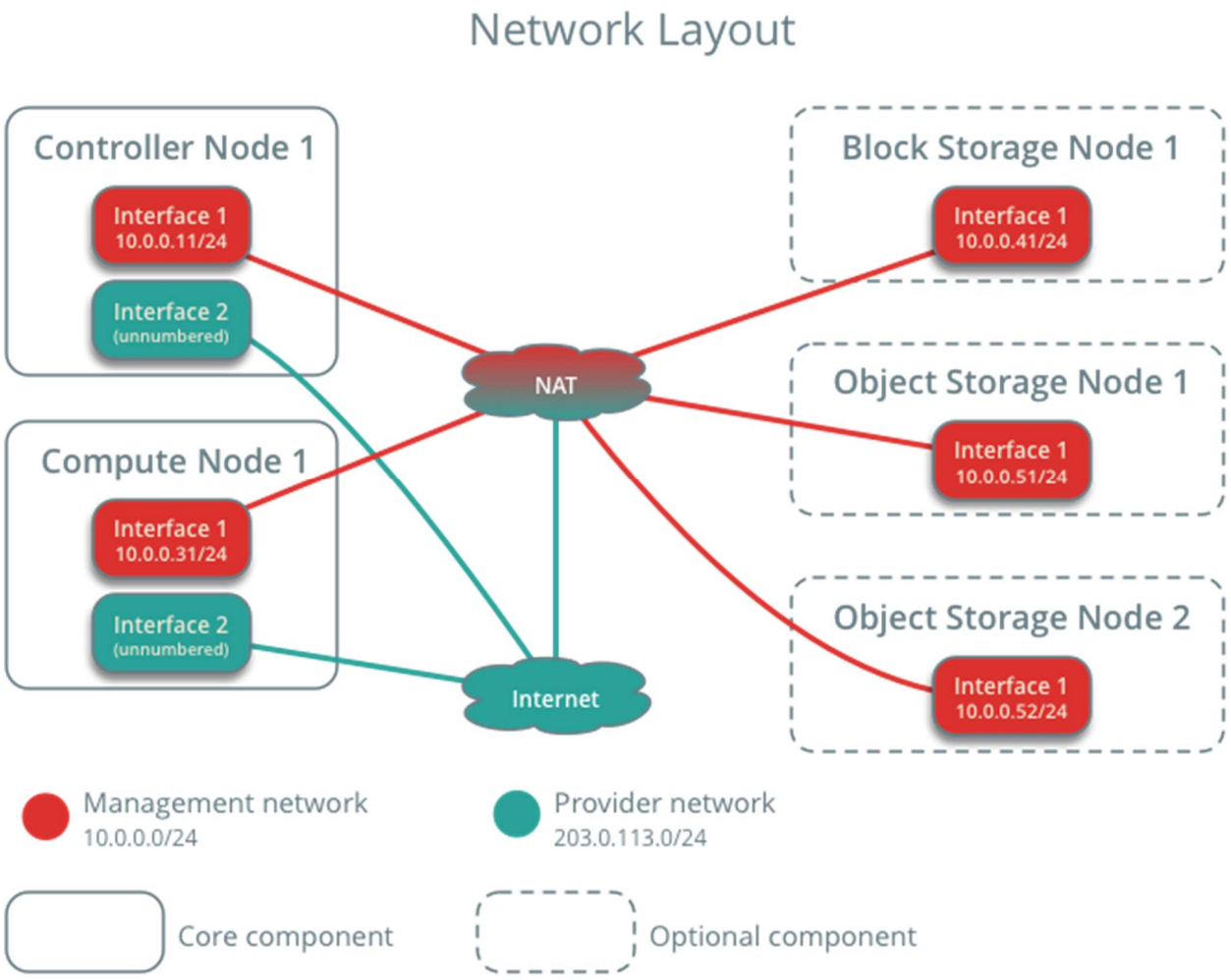
Host networking

THIS PAGE LAST UPDATED: 2019-11-11 12:17:29

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

All nodes require Internet access for administrative purposes such as package installation, security updates, DNS, and NTP. In most cases, nodes should obtain Internet access through the management network interface. To highlight the importance of network separation, the example architectures use private address space for the management network and assume that the physical network infrastructure provides Internet access via NAT or other 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
RHEL, CentOS and SUSE distributions enable a restrictive firewall by default. Ubuntu does not. For more information about securing your environment, refer to the OpenStack Security Guide.

Controller node

THIS PAGE LAST UPDATED: 2019-11-11 12:17:29

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*.

   For RHEL or CentOS:

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

     Do not change the **HWADDR** and **UUID** keys.

```
DEVICE=INTERFACE_NAME
TYPE=Ethernet
ONBOOT="yes"
BOOTPROTO="none"
```

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

THIS PAGE LAST UPDATED: 2019-11-11 12:17:29

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*.

   For RHEL or CentOS:

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

     Do not change the **HWADDR** and **UUID** keys.

```
DEVICE=INTERFACE_NAME
TYPE=Ethernet
ONBOOT="yes"
BOOTPROTO="none"
```

   For SUSE:

   - 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)

THIS PAGE LAST UPDATED: 2019-11-11 12:17:29

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

THIS PAGE LAST UPDATED: 2019-11-11 12:17:29

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 docs.openstack.org

PING files02.openstack.org (23.253.125.17) 56(84) bytes of data.

64 bytes from files02.openstack.org (23.253.125.17): icmp_seq=1 ttl=43 time=12
5 ms

64 bytes from files02.openstack.org (23.253.125.17): icmp_seq=2 ttl=43 time=12
5 ms

64 bytes from files02.openstack.org (23.253.125.17): icmp_seq=3 ttl=43 time=12
5 ms

64 bytes from files02.openstack.org (23.253.125.17): icmp_seq=4 ttl=43 time=12
5 ms



--- files02.openstack.org ping statistics ---

4 packets transmitted, 4 received, 0% packet loss, time 3004ms

rtt min/avg/max/mdev = 125.192/125.282/125.399/0.441 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

RHEL, CentOS and SUSE distributions enable 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)

THIS PAGE LAST UPDATED: 2019-11-11 12:17:29

To properly synchronize services among nodes, you can install Chrony, an implementation of NTP. 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

THIS PAGE LAST UPDATED: 2019-11-11 12:17:29

Perform these steps on the controller node.

Install and configure components

1. Install the packages:

   For Ubuntu:

   For RHEL or CentOS:

```
# yum install chrony
```

2. Edit the **chrony.conf** file and add, change, or remove the following keys as necessary for your environment.

   For RHEL, CentOS, or SUSE, edit the **/etc/chrony.conf** file:

```
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 same **chrony.conf** file mentioned above:

```
allow 10.0.0.0/24
```

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

4. Restart the NTP service:

For RHEL, CentOS, or SUSE:

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

Other nodes

THIS PAGE LAST UPDATED: 2019-11-11 12:17:29

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

Install and configure components

1. Install the packages.

For RHEL or CentOS:

```
# yum install chrony
```

2. Configure the **chrony.conf** file and comment out or remove all but one **server** key. Change it to reference the controller node.

For RHEL, CentOS, or SUSE, edit the **/etc/chrony.conf** file:

```
server controller iburst
```

3. Comment out the **`pool 2.debian.pool.ntp.org offline iburst`** line.
4. Restart the NTP service.

   For RHEL, CentOS, or SUSE:

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

Verify operation

THIS PAGE LAST UPDATED: 2019-11-11 12:17:29

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] +/-   4
3ms
  ^* 192.0.2.12                    2   6   177    46    +17us[  -23us] +/-   6
8ms
```

   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 for RHEL and CentOS

THIS PAGE LAST UPDATED: 2019-11-11 12:17:29

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.

Prerequisites

Warning

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

Note

The following steps apply to RHEL only. CentOS does not require these steps.

1. When using RHEL, it is assumed that you have registered your system using Red Hat Subscription Management and that you have the **`rhel-7-server-rpms`** repository enabled by default.

   For more information on registering the system, see the Red Hat Enterprise Linux 7 System Administrator's Guide.

2. In addition to **`rhel-7-server-rpms`**, you also need to have the **`rhel-7-server-optional-rpms`**, **`rhel-7-server-extras-rpms`**, and **`rhel-7-server-rh-common-rpms`** repositories enabled:

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

Enable the OpenStack repository

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

  When installing the Rocky release, run:

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

  When installing the Queens release, run:

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

  When installing the Pike release, run:

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

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

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

  The RDO repository RPM installs the latest available OpenStack release.

Finalize the installation

1. Upgrade the packages on all nodes:

```
# yum upgrade
```

   Note

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

2. Install the OpenStack client:

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

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

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

SQL database

THIS PAGE LAST UPDATED: 2019-11-11 12:17:29

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.

Note
If you see **Too many connections** or **Too many open files** error log messages on OpenStack services, verify that maximum number of connection settings are well applied to your environment. In MariaDB, you may also need to change open_files_limit configuration.

SQL database for RHEL and CentOS

THIS PAGE LAST UPDATED: 2019-11-11 12:17:29

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

Install and configure components

1. Install the packages:

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

2. Create and edit the **/etc/my.cnf.d/openstack.cnf** file (backup existing configuration files in **/etc/my.cnf.d/** if needed) 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 mariadb.service
# systemctl start mariadb.service
```

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

```
# mysql_secure_installation
```

Message queue for RHEL and CentOS

THIS PAGE LAST UPDATED: 2019-11-11 12:17:29

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:

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

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

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

3. Add the **openstack** user:

```
# rabbitmqctl add_user openstack RABBIT_PASS

Creating user "openstack" ...
```

   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 for RHEL and CentOS

THIS PAGE LAST UPDATED: 2019-11-11 12:17:29

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

Install and configure components

1. Install the packages:

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

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:

```
OPTIONS="-l 127.0.0.1,::1,controller"
```

> Note
>
> Change the existing line **OPTIONS="-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
```

Etcd for RHEL and CentOS

THIS PAGE LAST UPDATED: 2019-11-11 12:17:29

OpenStack services may use Etcd, a distributed reliable key-value store for distributed key locking, storing configuration, keeping track of service live-ness and other scenarios.

The etcd service runs on the controller node.

Install and configure components

1. Install the package:

```
# yum install etcd
```

2. Edit the **/etc/etcd/etcd.conf** file and set the **ETCD_INITIAL_CLUSTER**, **ETCD_INITIAL_ADVERTISE_PEER_URLS**, **ETCD_ADVERTISE_CLIENT_URLS**, **ETCD_LISTEN_CLIENT_URLS** to the management IP address of the controller node to enable access by other nodes via the management network:

```
#[Member]
ETCD_DATA_DIR="/var/lib/etcd/default.etcd"
```

```
ETCD_LISTEN_PEER_URLS="http://10.0.0.11:2380"
ETCD_LISTEN_CLIENT_URLS="http://10.0.0.11:2379"
ETCD_NAME="controller"
#[Clustering]
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://10.0.0.11:2380"
ETCD_ADVERTISE_CLIENT_URLS="http://10.0.0.11:2379"
ETCD_INITIAL_CLUSTER="controller=http://10.0.0.11:2380"
ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster-01"
ETCD_INITIAL_CLUSTER_STATE="new"
```

Finalize installation

1. Enable and start the etcd service:

```
# systemctl enable etcd
# systemctl start etcd
```

Install OpenStack services

The installation of individual OpenStack services is covered in the Project Installation Guides that are available at the following locations:

- OpenStack Installation Guides for Train

- OpenStack Installation Guides for Stein
- OpenStack Installation Guides for Rocky
- OpenStack Installation Guides for Queens
- OpenStack Installation Guides for Pike

Minimal deployment for Queens

At a minimum, you need to install the following services. Install the services in the order specified below:

- Identity service – keystone installation for Queens
- Image service – glance installation for Queens
- Compute service – nova installation for Queens
- Networking service – neutron installation for Queens

We advise to also install the following components after you have installed the minimal deployment services:

- Dashboard – horizon installation for Queens
- Block Storage service – cinder installation for Queens

Keystone Installation Tutorial for Red Hat Enterprise Linux and CentOS

UPDATED: 2019-11-14 17:15

Abstract

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

Explanations of configuration options and sample configuration files are included.

Note

The Training Labs scripts provide an automated way of deploying the cluster described in this Installation Guide into VirtualBox or KVM VMs. You will need a desktop computer or a laptop with at least 8 GB memory and 20 GB free storage running Linux, MaOS, or Windows. Please see the OpenStack Training Labs.

This guide documents the OpenStack Queens release.

Warning

This guide is a work-in-progress and is subject to updates frequently. Pre-release packages have been used for testing, and some instructions may not work with final versions. Please help us make this guide better by reporting any errors you encounter.

Contents

Identity service overview

UPDATED: 2019-11-14 17:15

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

UPDATED: 2019-11-14 17:15

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.

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.

1. Run the following command to install the packages:

```
# yum install openstack-keystone httpd 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/keyston
e
```

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 keyst
one

# keystone-manage credential_setup --keystone-user keystone --keystone-group k
eystone
```

5. Bootstrap the Identity service:

> Note
>
> Before the Queens release, keystone needed to be run on two separate ports to accomodate the Identity v2 API which ran a separate admin-only service commonly on port 35357. With the removal of the v2 API, keystone can be run on the same port for all interfaces.

```
# keystone-manage bootstrap --bootstrap-password ADMIN_PASS \
  --bootstrap-admin-url http://controller:5000/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/httpd/conf/httpd.conf** file and configure the **ServerName** option to reference the controller node:

```
ServerName controller
```

2. Create a link to the **/usr/share/keystone/wsgi-keystone.conf** file:

```
# ln -s /usr/share/keystone/wsgi-keystone.conf /etc/httpd/conf.d/
```

Finalize the installation

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

```
# systemctl enable httpd.service
# systemctl start httpd.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:5000/v3
$ export OS_IDENTITY_API_VERSION=3
```

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

Create a domain, projects, users, and roles

UPDATED: 2019-11-14 17:15

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

1. Although the “default” domain already exists from the *keystone-manage bootstrap* step in this guide, a formal way to create a new domain would be:

```
$ openstack domain create --description "An Example Domain" example

+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
```

```
| description | An Example Domain                |
| enabled     | True                             |
| id          | 2f4f80574fd84fe6ba9067228ae0a50c |
| name        | example                          |
+-------------+----------------------------------+
```

2. 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                          |
+-------------+----------------------------------+
```

3. 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** project and user:

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

Note

This command provides no output.

Note

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

Verify operation

UPDATED: 2019-11-14 17:15

Verify operation of the Identity service before installing other services.

Note

Perform these commands on the controller node.

1. Unset the temporary **OS_AUTH_URL** and **OS_PASSWORD** environment variable:

```
$ unset OS_AUTH_URL OS_PASSWORD
```

2. 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.

3. 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

UPDATED: 2019-11-14 17:15

The previous sections 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 and located in a secure place appropriate for your deployment, as they do contain sensitive credentials.

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:5000/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 overview

UPDATED: 2019-09-10 15:48

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). 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.

Note
An OpenStack Community Goal in the Pike release was Control Plane API endpoints deployment via WSGI. As currently constituted, however, glance-api is **not suitable** to be run in such a configuration. Instead we recommend that Glance be run in the traditional manner as a standalone server. See the "Known Issues" section of the Glance Release Notes for the Pike and Queens releases for more information.

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.

Note
The Glance Registry Service and its APIs have been DEPRECATED in the Queens release and are subject to removal at the beginning of the 'S' development cycle, following the OpenStack standard deprecation policy.

For more information, see the Glance specification document Actually Deprecate the Glance Registry.

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 (Red Hat)

UPDATED: 2019-09-10 15:48

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'@'localhos
   t' 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.

1. Install the packages:

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

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

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

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

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

```
[keystone_authtoken]
# ...
auth_uri = http://controller:5000
auth_url = http://controller:5000
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:

> Note
>
> The Glance Registry Service and its APIs have been DEPRECATED in the Queens release and are subject to removal at the beginning of the 'S' development cycle, following the OpenStack standard deprecation policy.
>
> For more information, see the Glance specification document Actually Deprecate the Glance Registry.

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:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
```

```
username = glance
password = GLANCE_PASS

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

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

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

4. Populate the Image service database:

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

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

Finalize installation

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

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

Compute service overview

UPDATED: 2019-11-05 14:16

Todo

Update a lot of the links in here.

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. 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 Compute 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 the **`conductor`** section in the Configuration Options.

`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 for Red Hat Enterprise Linux and CentOS

UPDATED: 2019-11-05 14:16

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'@'localhos
t' \
  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'@'localh
ost' \
  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

5. +--------------+-----------------------------------------+
| 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              |
+--------------+------------------------------------------+
```

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

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.

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

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

7. Create the Placement API service endpoints:

```
$ openstack endpoint create --region RegionOne placement public http://control
ler: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://contr
oller: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://controll
er: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:

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

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_url = http://controller:5000/v3
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]
o  enabled = true
# ...
server_listen = $my_ip
server_proxyclient_address = $my_ip
```

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

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

In the **[oslo_concurrency]** section, configure the lock path:

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

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:5000/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.

Due to a packaging bug, you must enable access to the Placement API by adding the following configuration to **/etc/httpd/conf.d/00-nova-placement-api.conf**:

```
<Directory /usr/bin>
   <IfVersion >= 2.4>
      Require all granted
   </IfVersion>
   <IfVersion < 2.4>
      Order allow,deny
      Allow from all
   </IfVersion>
</Directory>
```

Restart the httpd service:

```
# systemctl restart httpd
```

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" no
va
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

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 for Red Hat Enterprise Linux and CentOS

UPDATED: 2019-11-05 14:16

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

Note

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

Install and configure components

Note

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

1. Install the packages:

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

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

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

    - 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_url = http://controller:5000/v3
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
server_listen = 0.0.0.0
server_proxyclient_address = $my_ip
novncproxy_base_url = http://controller:6080/vnc_auto.html
```

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

Note

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

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

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

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

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

- 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:5000/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.

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 compute service list --service nova-compute

+----+-------+--------------+------+-------+---------+-----------------------
----+

| ID | Host  | Binary       | Zone | State | Status  | Updated At
|

+----+-------+--------------+------+-------+---------+-----------------------
----+

| 1  | node1 | nova-compute | nova | up    | enabled | 2017-04-14T15:30:44.000
000 |

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

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-bc04
0dc106b3
Creating host mapping for compute host 'compute': fe58ddc1-1d65-4f87-9456-bc04
0dc106b3
```

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

UPDATED: 2019-11-05 14:16

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:5000/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 |
6. | Result: Success           |
| Details: None             |
```

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

Networking service overview

UPDATED: 2019-11-14 05:14

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

UPDATED: 2019-11-14 05:14

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

UPDATED: 2019-11-14 05:14

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'@'localh
ost' \
  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
- Networking Option 2: Self-service networks

Configure the metadata agent

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

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

```
[DEFAULT]
# ...
nova_metadata_host = 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

Note

The Nova compute service must be installed to complete this step. For more details see the compute install guide found under the *Installation Guides* section of the docs website.

- 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.

Remove openstack term from beginning of all codes below and change the openstack-neutron.service to neutron-server.service

---

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
```

Networking Option 1: Provider networks

UPDATED: 2019-11-14 05:14

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

Install the components

```
# yum install openstack-neutron openstack-neutron-ml2 \
  openstack-neutron-linuxbridge ebtables
```

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.

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

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

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.IptablesFirewall
Driver
```

- Ensure your Linux operating system kernel supports network bridge filters by verifying all the following **`sysctl`** values are set to **`1`**:

```
net.bridge.bridge-nf-call-iptables
net.bridge.bridge-nf-call-ip6tables
```

To enable networking bridge support, typically the **`br_netfilter`** kernel module needs to be loaded. Check your operating system's documentation for additional details on enabling this module.

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

UPDATED: 2019-11-14 05:14

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

Install the components

```
# yum install openstack-neutron openstack-neutron-ml2 \
  openstack-neutron-linuxbridge ebtables
```

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.

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

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

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.IptablesFirewall
Driver
```

- Ensure your Linux operating system kernel supports network bridge filters by verifying all the following **sysctl** values are set to **1**:

```
net.bridge.bridge-nf-call-iptables
net.bridge.bridge-nf-call-ip6tables
```

  To enable networking bridge support, typically the **br_netfilter** kernel module needs to be loaded. Check your operating system's documentation for additional details on enabling this module.

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*.

Install and configure compute node

UPDATED: 2019-11-14 05:14

The compute node handles connectivity and security groups for instances.

Install the components

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

Configure the common component

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

Note

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

- Edit the **/etc/neutron/neutron.conf** file and complete the following actions:
    - In the **[database]** section, comment out any **connection** options because compute nodes do not directly access the database.
    - In the **[DEFAULT]** 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 **[oslo_concurrency]** section, configure the lock path:

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

Configure networking options

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

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

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. Restart the Compute service:

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

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

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

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

Networking Option 1: Provider networks

UPDATED: 2019-11-14 05:14

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.IptablesFirewall
Driver
```

- Ensure your Linux operating system kernel supports network bridge filters by verifying all the following **sysctl** values are set to **1**:

```
net.bridge.bridge-nf-call-iptables

net.bridge.bridge-nf-call-ip6tables
```

To enable networking bridge support, typically the **br_netfilter** kernel module needs to be loaded. Check your operating system's documentation for additional details on enabling this module.

Return to *Networking compute node configuration*

Networking Option 2: Self-service networks

UPDATED: 2019-11-14 05:14

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.IptablesFirewall
Driver
```

- Ensure your Linux operating system kernel supports network bridge filters by verifying all the following **sysctl** values are set to **1**:

```
net.bridge.bridge-nf-call-iptables
net.bridge.bridge-nf-call-ip6tables
```

To enable networking bridge support, typically the **br_netfilter** kernel module needs to be loaded. Check your operating system's documentation for additional details on enabling this module.

Return to *Networking compute node configuration*.

Installation Guide (Dashboard)

UPDATED: 2019-10-09 11:31

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.

System Requirements

UPDATED: 2019-10-09 11:31

The Queens release of horizon has the following dependencies.

- Python 2.7
- Django 1.11
    - Django 1.8 to 1.10 are also supported. Their support will be dropped in the Rocky release.
- An accessible keystone endpoint
- All other services are optional. Horizon supports the following services as of the Queens release. If the keystone endpoint for a service is configured, horizon detects it and enables its support automatically.
    - cinder: Block Storage
    - glance: Image Management
    - neutron: Networking
    - nova: Compute
    - swift: Object Storage
    - Horizon also supports many other OpenStack services via plugins. For more information, see the Plugin Registry.

Install and configure for Red Hat Enterprise Linux and CentOS

UPDATED: 2019-10-09 11:31

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.

Install and configure components

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

1. Install the packages:

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

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

```
OPENSTACK_HOST = "controller"
```

    - Allow 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 https://docs.djangoproject.com/en/dev/ref/settings/#allowed-hosts 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.

3. Add the following line to **`/etc/httpd/conf.d/openstack-dashboard.conf`** if not included.

```
WSGIApplicationGroup %{GLOBAL}
```

Finalize installation

- Restart the web server and session storage service:

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

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

Verify operation for Red Hat Enterprise Linux and CentOS

UPDATED: 2019-10-09 11:31

Verify operation of the dashboard.

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

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

If you want to install cinder block storage service, visit this address:

https://docs.openstack.org/cinder/queens/install/

References

https://docs.openstack.org/install-guide/index.html