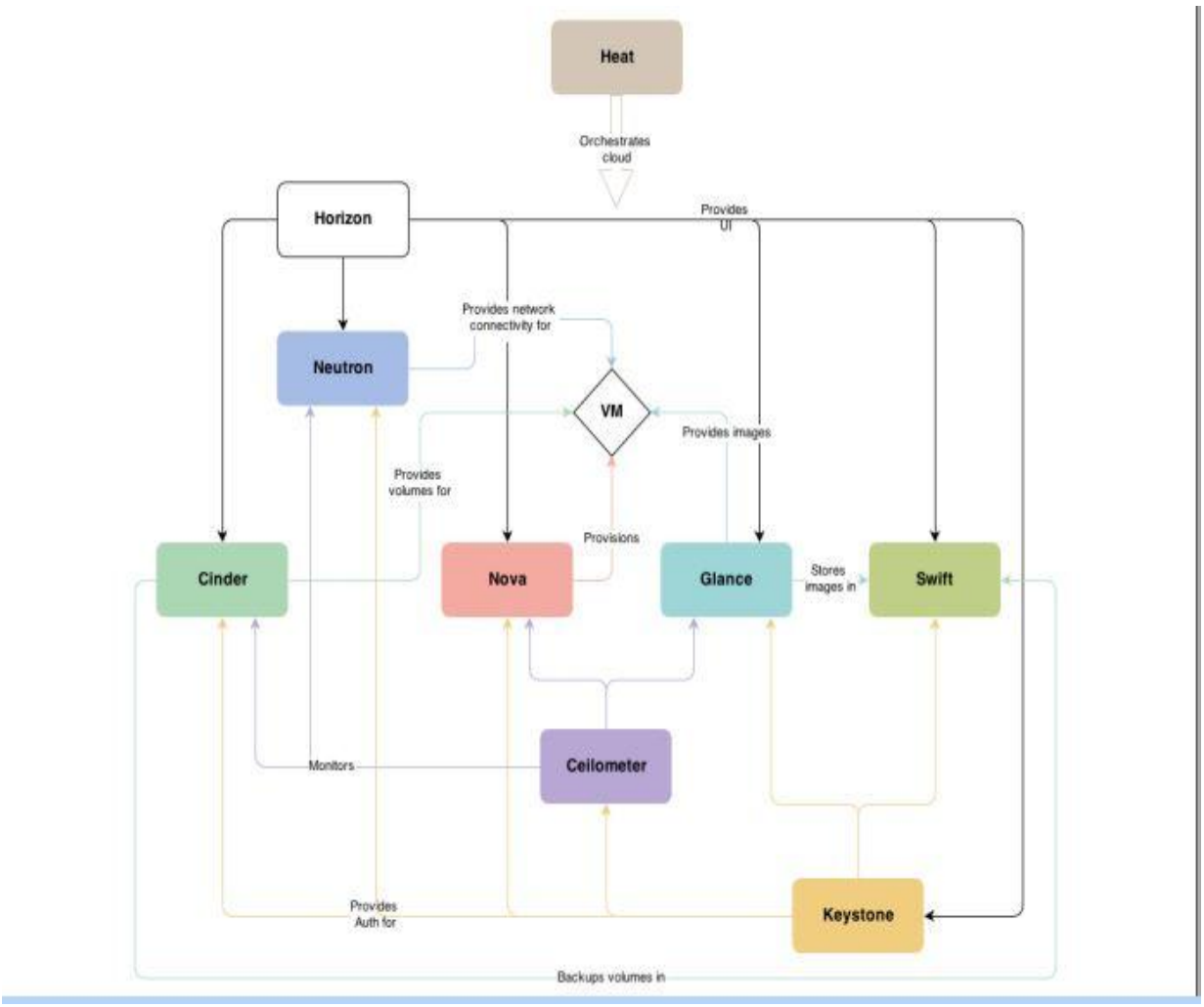
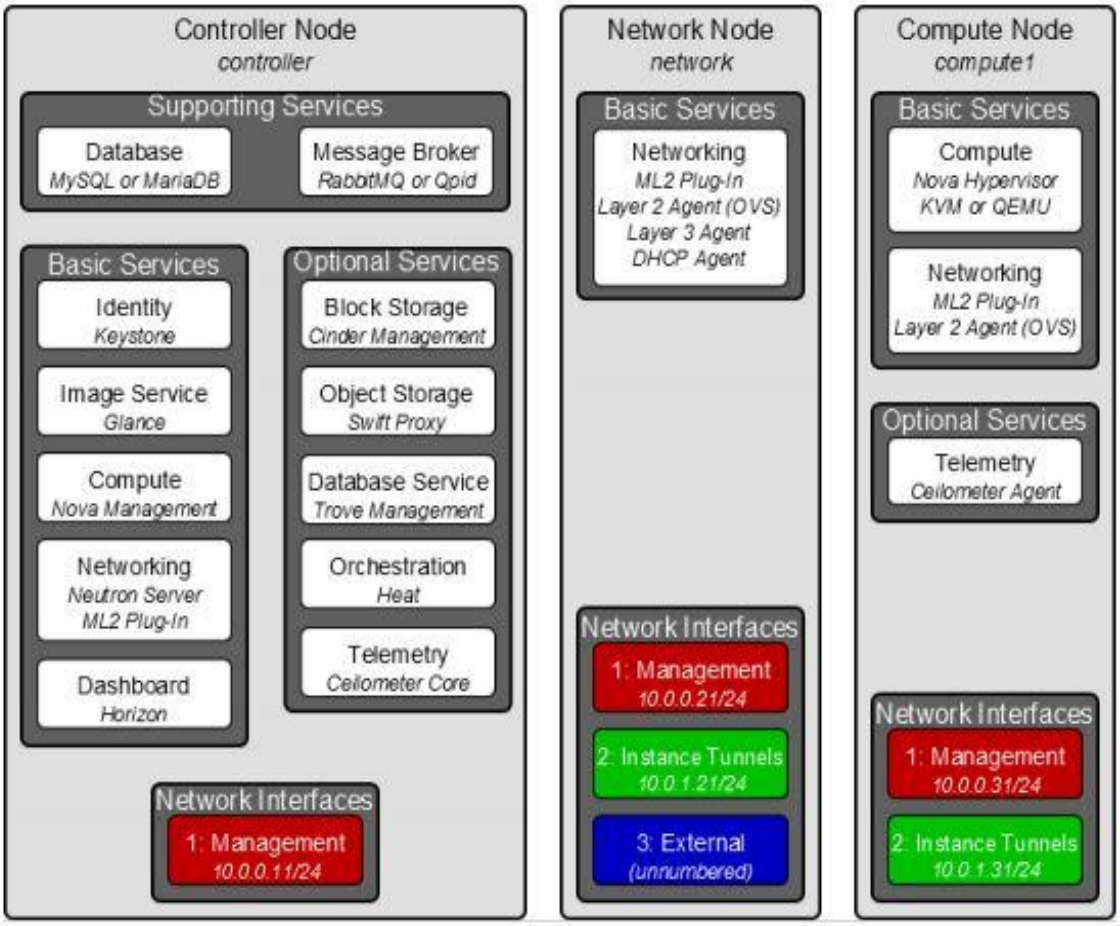
OpenStack Juno for RHEL、CentOS、Fedora 安装指南

一、体系架构

1、OpenStack 组件结构



2、OpenStack 三节点结构



二、基本环境

控制节点：1 处理器、2 GB 内存和 5 GB 的存储空间
网络节点：1 处理器、512 MB 内存和 5 GB 的存储空间
计算节点：1 处理器、2 GB 内存和 10 GB 的存储空间

网络
使用 OpenStack 网络（neutron）

控制器节点
配置网络：
1、 第一个接口配置为管理界面
IP address: 10.0.0.11
Network mask: 255.255.255.0 (or /24)
Default gateway: 10.0.0.1

2、 重启系统
reboot

配置名称解析：
1、设置控制器节点的主机名
2、编辑/etc/hosts 文件包含以下
controller
10.0.0.11 controller
network
10.0.0.21 network
compute1
10.0.0.31 compute1

网络节点
配置网络：
1、 第一个接口配置为管理界面
IP address: 10.0.0.21
Network mask: 255.255.255.0 (or /24)
Default gateway: 10.0.0.1

2、配置第二个界面实例隧道接口
IP address: 10.0.1.21
Network mask: 255.255.255.0 (or /24)

3、外部接口使用一个特殊的配置，不分配 IP 地址给它。第三个接口配置为外部接口。
INTERFACE_NAME 替换为实际的接口名称。例如，eth2 或 ens256。

编辑/etc/sysconfig/network-scripts / ifcfg-INTERFACE_NAME 文件包含以下（但不改变 HWADDR 和 UUID 键）：
DEVICE=INTERFACE_NAME
TYPE=Ethernet
ONBOOT="yes"
BOOTPROTO="none"

4、重启系统
reboot

配置名称解析：
1、设置网络节点的主机名
2、编辑/etc/hosts 文件包含以下
network
10.0.0.21 network

controller
10.0.0.11 controller
compute1
10.0.0.31 compute1

计算节点

配置网络

1、第一个接口配置为管理界面

IP address: 10.0.0.31
Network mask: 255.255.255.0 (or /24)
Default gateway: 10.0.0.1

请注意

额外的计算节点应该使用 10.0.0.32 10.0.0.33 等。

2、配置第二个界面实例隧道接口

IP address: 10.0.1.31
Network mask: 255.255.255.0 (or /24)

3、重启系统

reboot

配置名称解析：

1、compute1 设置为节点的主机名

2、编辑/etc/hosts 文件包含以下

compute1
10.0.0.31 compute1
controller
10.0.0.11 controller
network
10.0.0.21 network

验证连接

我们建议您验证节点之间的网络和互联网连接

1、 从控制器节点，ping 一个网站

 # ping -c 4 openstack.org

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

4 packets transmitted, 4 received, 0% packet loss, time 3022ms
rtt min/avg/max/mdev = 17.489/17.715/18.346/0.364 ms

2、 从控制器节点，ping 网络节点
 # ping -c 4 network
PING network (10.0.0.21) 56(84) bytes of data.
64 bytes from network (10.0.0.21): icmp_seq=1 ttl=64 time=0.263 ms
64 bytes from network (10.0.0.21): icmp_seq=2 ttl=64 time=0.202 ms
64 bytes from network (10.0.0.21): icmp_seq=3 ttl=64 time=0.203 ms
64 bytes from network (10.0.0.21): icmp_seq=4 ttl=64 time=0.202 ms
--- network 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、从控制器节点，ping 计算节点
 # 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
--- network ping statistics ---

4、从网络节点，ping 一个网站

5、 从网络节点，ping 控制节点

6、 从网络节点，ping 计算节点

7、 从计算节点，ping 一个网站

8、 从计算节点，ping 控制节点

9、从计算节点，ping 网络节点

网络时间协议（NTP）
控制节点
安装 NTP 服务
yum install ntp

配置 NTP 服务
默认情况下，控制器节点同步，通过公共服务器池的时间。但是，你可以选择编辑/etc/ntp.conf 中的文件来配置备用服务器，来为您的组织提供。

1、 编辑/etc/ntp.conf 中的文件
server NTP_SERVER iburst
restrict -4 default kod notrap nomodify
restrict -6 default kod notrap nomodify

用一个合适的更准确的主机名或 IP 地址的 NTP 服务器，替换 NTP_SERVER。

2、启动 NTP 服务，并将其配置为随系统自启动
systemctl enable ntpd.service
systemctl start ntpd.service

其他两个节点
安装 NTP 服务
yum install ntp

配置 NTP 服务
配置网络和计算节点，以引用控制器节点。
1、编辑/etc/ntp.conf 中的文件
server controller iburst

2、启动 NTP 服务，并将其配置为随系统自启动
systemctl enable ntpd.service
systemctl start ntpd.service

验证操作
我们建议您在进一步处理之前确认 NTP 是否同步。一些节点，特别是那些引用控制器的节点，可能需要几分钟同步。

1、控制器节点上运行此命令
 # ntpq -c peers
```
  remote refid st t when poll reach delay offset
  jitter
==============================================================================
*ntp-server1 192.0.2.11  2 u  169 1024  377    1.901   -0.611
  5.483
+ntp-server2 192.0.2.12  2 u  887 1024  377    0.922   -0.246
  2.864
```

2、控制器节点上运行此命令
 # ntpq -c assoc

```
ind assid status conf reach auth condition last_event cnt
===========================================================
  1 20487 961a yes yes none sys.peer sys_peer 1
  2 20488 941a yes yes none candidate sys_peer 1
```

在状态栏内容应说明 sys.peer 用于至少一个服务器。

3、其他节点上运行下面命令

```
# ntpq -c peers
  remote refid st t when poll reach delay offset
  jitter
==============================================================================
=====
====
*controller 192.0.2.21 3 u 47 64 37 0.308 -0.251
  0.079
```

4、其他节点上运行下面命令

```
# ntpq -c assoc
ind assid status conf reach auth condition last_event cnt
===========================================================
  1 21181 963a yes yes none sys.peer sys_peer 3
```

安全

OpenStack 的服务，支持各种安全方法，包括密码策略和加密。此外，配套服务，包括数据库服务器和消息代理至少支持安全性的密码。

为了简化安装过程，本指南只涉及适用的安全密码。

您可以手动创建安全密码或使用工具，如 pwgen 生成它们，或通过运行以下命令：

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

对于 OpenStack 的服务，本指南使用 SERVICE_PASS 引用服务帐户的密码和 SERVICE_DBPASS 引用数据库密码。

下面的表提供了所需要的密码和服务及其相关联的列表：

表 3 密码

Password name	Description
Database password (no variable used)	Root password for the database
RABBIT_PASS	Password of user guest of RabbitMQ
KEYSTONE_DBPASS	Database password of Identity service
DEMO_PASS	Password of user `demo`
ADMIN_PASS	Password of user `admin`

Password name	Description
GLANCE_DBPASS	Database password for Image Service
GLANCE_PASS	Password of Image Service user `glance`
NOVA_DBPASS	Database password for Compute service
NOVA_PASS	Password of Compute service user `nova`
DASH_DBPASS	Database password for the dashboard
CINDER_DBPASS	Database password for the Block Storage service
CINDER_PASS	Password of Block Storage service user `cinder`
NEUTRON_DBPASS	Database password for the Networking service
NEUTRON_PASS	Password of Networking service user `neutron`
HEAT_DBPASS	Database password for the Orchestration service
HEAT_PASS	Password of Orchestration service user `heat`
CEILOMETER_DBPASS	Database password for the Telemetry service
CEILOMETER_PASS	Password of Telemetry service user `ceilometer`
TROVE_DBPASS	Database password of Database service
TROVE_PASS	Password of Database Service user `trove`

数据库

控制器节点

安装和配置数据库服务器

1、安装包

```
# yum install mariadb mariadb-server  MySQL-python
```

2、编辑/etc/my.cnf 文件

A．在[mysqld]部分，设置 bind-address 的控制节点的管理网 IP 地址，以便通过管理网络访问其他节点：

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

B．在[mysqld]部分中，设置以下值启用有用的选项和 UTF-8 字符集：

```
[mysqld]
...
default-storage-engine = innodb
innodb_file_per_table
collation-server = utf8_general_ci
init-connect = 'SET NAMES utf8'
character-set-server = utf8
```

完成安装

1、启动数据库服务，并将其配置为随系统自启动

systemctl enable mariadb.service

systemctl start mariadb.service

2、固定数据库服务，包括选择合适的 root 帐户密码

mysql_secure_installation

OpenStack 软件包

请注意

禁用或移除任何自动更新服务，因为他们可能影响你的 OpenStack 环境。

在所有节点上执行这些过程。

1、安装 yum-plugin-priorities 包

yum install yum-plugin-priorities

2、安装 EPEL-release 组件

yum install http://dl.fedoraproject.org/pub/epel/7/x86_64/e/epel-release-7-2.noarch.rpm

OpenStack 存储库

1、安装 RDO-RELEASE-JUNO 包

yum install http://rdo.fedorapeople.org/openstack-juno/rdo-release-juno.rpm

完成安装

1、 升级你系统上的软件包

yum upgrade

请注意

如果在升级过程包括一个新的内核，重新启动您的系统将其激活。

2、RHEL 和 CentOS 默认启用了 SELinux。安装 SELinux 包自动管理 OpenStack 服务的安全策略

yum install openstack-selinux

Messaging 服务

安装 RabbitMQ 消息代理服务

yum install rabbitmq-server

配置消息代理服务

1、 启动消息代理服务，并将其配置为自启动

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

2、运行下面的命令，用合适的密码替换 RABBIT_PASS

```
# rabbitmqctl change_password guest RABBIT_PASS
Changing password for user "guest" ...
...done.
```

恭喜你，现在你已经准备好要安装 OpenStack 的服务了！

添加 Identity 服务

安装和配置

本节介绍如何在控制器节点，安装和配置 OpenStack 的身份服务。

配置 OpenStack 的身份服务，你必须创建一个数据库和一个管理令牌（token）。

1、创建数据库，请完成以下步骤

A．登陆数据库：

```
$ mysql -u root –p
```

B．创建 keystone 数据库：

```
CREATE DATABASE keystone;
```

C．授予适当的访问 Keystone 数据库的权限：

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

用合适的密码替换 KEYSTONE_DBPASS。

D．退出数据库

2、生成一个随机值作为初始配置过程中的管理令牌

```
# openssl rand -hex 10
```

安装和配置组件

1、 运行以下命令安装软件包

```
# yum install openstack-keystone python-keystoneclient
```

2、编辑/etc/keystone/keystone.conf 文件

A．在[DEFAULT]部分，定义初始管理的值：
[DEFAULT]
...
admin_token = ADMIN_TOKEN
使用您先前生成的随机值更换 ADMIN_TOKEN

B．在[database]部分，配置数据库访问：
[database]
...
connection = mysql://keystone: KEYSTONE_DBPASS@controller/keystone

用你选择的数据库密码替换 keystone_dbpass。

C.（可选）在[DEFAULT]部分，启用详细日志记录协助解决问题：
[DEFAULT]
...
verbose = True

3、创建通用的证书和密钥，限制访问相关文件
```
# keystone-manage pki_setup --keystone-user keystone --keystone-group
  keystone
# chown -R keystone:keystone /var/log/keystone
# chown -R keystone:keystone /etc/keystone/ssl
# chmod -R o-rwx /etc/keystone/ssl
```

4、填充 Keystone 身份服务数据库
```
# su -s /bin/sh -c "keystone-manage db_sync" keystone
```

完成安装

1、 启动身份服务和配置自启动
```
# systemctl enable openstack-keystone.service
# systemctl start openstack-keystone.service
```

2、默认情况下，身份服务存储在数据库中，无限过期的令牌积累大大增加了数据库的大小，可能会降低服务的性能，特别是在资源有限的环境中。我们建议您使用 cron 配置一个周期性任务，以清除过期的令牌：
```
# (crontab -l -u keystone 2>&1 | grep -q token_flush) || \
  echo '@hourly /usr/bin/keystone-manage token_flush >/var/log/keystone/
keystone-tokenflush.log 2>&1' \
  >> /var/spool/cron/keystone
```

创建租户、用户和角色

1、配置管理令牌
$ export OS_SERVICE_TOKEN=ADMIN_TOKEN

更换 ADMIN_TOKEN 与您在“安装和配置”一节中所产生的管理令牌。例如：
$ export OS_SERVICE_TOKEN=294a4c8a8a475f9b9836

2、 配置端点
$ export OS_SERVICE_ENDPOINT=http://controller:35357/v2.0

创建租户、用户和角色
1、创建一个 admin 租户、用户、角色
A．创建 admin 租户：
$ keystone tenant-create --name admin --description "Admin Tenant"
+-------------+----------------------------------+
| Property | Value |
+-------------+----------------------------------+
description	Admin Tenant
enabled	True
id	6f4c1e4cbfef4d5a8a1345882fbca110
name	admin

B．创建 admin 用户：
$ keystone user-create --name admin --pass ADMIN_PASS --
email EMAIL_ADDRESS
+----------+----------------------------------+
| Property | Value |
+----------+----------------------------------+
email	admin@example.com
enabled	True
id	ea8c352d253443118041c9c8b8416040
name	admin
username	admin

用合适的密码和电子邮件替换 ADMIN_PASS 和 EMAIL_ADDRESS。

C．创建 admin 角色：
$ keystone role-create --name admin
+----------+----------------------------------+
| Property | Value |
+----------+----------------------------------+
| id | bff3a6083b714fa29c9344bf8930d199 |
| name | admin |

D．添加 admin 租户、admin 用户和 admin 角色：

$ keystone user-role-add --tenant admin --user admin --role admin

E．在默认情况下，dashboard 限制用户访问_member_作用域。
创建_member_作用域：
$ keystone role-create --name _member_

```
+----------+----------------------------------+
| Property | Value |
+----------+----------------------------------+
| id | 0f198e94ffce416cbcbe344e1843eac8 |
| name | _member_ |
+----------+----------------------------------+
```

F．添加 admin 租户和用户到_member_角色：
$ keystone user-role-add --tenant admin --user admin --role _member_

2、创建一个典型环境中的 demo 租户和用户
A．创建 demo 租户：
$ keystone tenant-create --name demo --description "Demo Tenant"

```
+-------------+-----------------------------------+
| Property | Value |
+-------------+-----------------------------------+
| description | Demo Tenant |
| enabled | True |
| id | 4aa51bb942be4dd0ac0555d7591f80a6 |
| name | demo |
```

B．创建 demo 用户：
$ keystone user-create --name demo --pass DEMO_PASS --
email EMAIL_ADDRESS

```
+----------+----------------------------------+
| Property | Value |
+----------+----------------------------------+
| email | demo@example.com |
| enabled | True |
| id | 7004dfa0dda84d63aef81cf7f100af01 |
| name | demo |
| username | demo |
+----------+----------------------------------+
```

使用一个合适的密码和电子邮件替换 demo_pass 和 email_address。

C．添加 demo 租户和用户到_member_角色：
$ keystone user-role-add --tenant demo --user demo --role _member_

3、OpenStack 服务还需要一个租户、用户和角色与其他服务进行交互。
•创建 service 租户：

```
$ keystone tenant-create --name service --description "Service Tenant"
+-------------+----------------------------------+
| Property | Value |
+-------------+----------------------------------+
| description | Service Tenant |
| enabled | True |
| id | 6b69202e1bf846a4ae50d65bc4789122 |
| name | service |
```

创建 service entity 和 API 端点

1、身份服务管理你的 OpenStack 环境目录服务。服务使用此目录在您的环境中查找其他服务

创建用于 Identity service 的服务：

```
$ keystone service-create --name keystone --type identity \
  --description "OpenStack Identity"
+-------------+----------------------------------+
| Property | Value |
+-------------+----------------------------------+
| description | OpenStack Identity |
| enabled | True |
| id | 15c11a23667e427e91bc31335b45f4bd |
| name | keystone |
| type | identity |
+-------------+----------------------------------+
```

2、创建用于 Identity service 的 API 端点

```
$ keystone endpoint-create \
  --service-id $(keystone service-list | awk '/ identity / {print $2}') \
  --publicurl http://controller:5000/v2.0 \
  --internalurl http://controller:5000/v2.0 \
  --adminurl http://controller:35357/v2.0 \
  --region regionOne
+-------------+----------------------------------+
| Property | Value |
+-------------+----------------------------------+
| adminurl | http://controller:35357/v2.0 |
| id | 11f9c625a3b94a3f8e66bf4e5de2679f |
| internalurl | http://controller:5000/v2.0 |
| publicurl | http://controller:5000/v2.0 |
| region | regionOne |
| service_id | 15c11a23667e427e91bc31335b45f4bd |
```

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

验证操作

本节描述如何验证身份服务操作。

1、卸载 os_service_token 和 os_service_endpoint 环境变量

$ unset OS_SERVICE_TOKEN OS_SERVICE_ENDPOINT

2、作为 admin 租户和角色，请求认证令牌

**$ keystone --os-tenant-name admin --os-username admin --ospassword ADMIN_PASS **
--os-auth-url http://controller:35357/v2.0 token-get

3、作为 admin 租户和用户，验证列表

**$ keystone --os-tenant-name admin --os-username admin --ospassword ADMIN_PASS **
--os-auth-url http://controller:35357/v2.0 tenant-list

```
+----------------------------------+----------+---------+
| id | name | enabled |
+----------------------------------+----------+---------+
| 6f4c1e4cbfef4d5a8a1345882fbca110 | admin | True |
| 4aa51bb942be4dd0ac0555d7591f80a6 | demo | True |
| 6b69202e1bf846a4ae50d65bc4789122 | service | True |
+----------------------------------+----------+---------+
```

4、列出用户身份验证服务

**$ keystone --os-tenant-name admin --os-username admin --ospassword ADMIN_PASS **
--os-auth-url http://controller:35357/v2.0 user-list

```
+----------------------------------+---------+--------
+--------------------+
| id | name | enabled | email
 |
+----------------------------------+---------+--------
+--------------------+
| ea8c352d253443118041c9c8b8416040 | admin | True | admin@example.
com |
| 7004dfa0dda84d63aef81cf7f100af01 | demo | True | demo@example.com
 |
+----------------------------------+---------+--------
+--------------------+
```

5、列出角色身份验证服务

$ keystone --os-tenant-name admin --os-username admin --ospassword

**ADMIN_PASS **
--os-auth-url http://controller:35357/v2.0 role-list

```
+----------------------------------+----------+
| id | name |
+----------------------------------+----------+
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_ |
| bff3a6083b714fa29c9344bf8930d199 | admin |
```

6、作为 demo 租户和用户，请求认证令牌

**$ keystone --os-tenant-name demo --os-username demo --ospassword DEMO_PASS **
--os-auth-url http://controller:35357/v2.0 token-get

```
+-----------+----------------------------------+
| Property | Value |
+-----------+----------------------------------+
| expires | 2014-10-10T12:51:33Z |
| id | 1b87ceae9e08411ba4a16e4dada04802 |
| tenant_id | 4aa51bb942be4dd0ac0555d7591f80a6 |
| user_id | 7004dfa0dda84d63aef81cf7f100af01 |
+-----------+----------------------------------+
```

7、列出用户验证

**$ keystone --os-tenant-name demo --os-username demo --ospassword DEMO_PASS **
--os-auth-url http://controller:35357/v2.0 user-list

You are not authorized to perform the requested action, admin_required.
(HTTP 403)

创建 OpenStack 客户端环境脚本

为 admin 和 demo 租户和用户创建客户端环境脚本。本指南参考这些脚本加载客户端适当的凭据。

1、编辑 admin-openrc.sh 文件，添加以下内容

```
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_AUTH_URL=http://controller:35357/v2.0
```

使用你在用户身份管理服务选择的密码替换 admin_pass。

3、 编辑 demo-openrc.sh 文件，添加以下内容

```
export OS_TENANT_NAME=demo
export OS_USERNAME=demo
```

export OS_PASSWORD=DEMO_PASS
export OS_AUTH_URL=http://controller:5000/v2.0

用您选择的密码替换 DEMO_PASS 演示用户的身份服务。

加载客户端环境脚本
$ source admin-openrc.sh

三、添加 Image 服务

安装和配置
本节介绍如何安装和配置 Image 服务，代号为 glance，在控制器节点上。为简单起见，此存储镜像配置为本地文件系统。

配置的先决条件
在您安装和配置镜像服务之前，您必须创建一个数据库和身份服务终端凭据。
1、创建数据库，完成这些步骤
A．登陆数据库：
$ mysql -u root -p

B．创建 glance 数据库：
CREATE DATABASE glance;

C．授予权限：
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \
  IDENTIFIED BY ' GLANCE_DBPASS';
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' \
  IDENTIFIED BY ' GLANCE_DBPASS';
用一个合适的密码替换 GLANCE_DBPASS。

D．退出数据库

2、执行 admin 凭证脚本文件
$ source admin-openrc.sh

3、创建身份服务凭据
A．创建 glance 用户：
$ keystone user-create --name glance --pass GLANCE_PASS
+----------+----------------------------------+
| Property | Value |
+----------+----------------------------------+
| email | |

enabled	True
id	f89cca5865dc42b18e2421fa5f5cce66
name	glance
username	glance
+----------+----------------------------------
用一个合适的密码替换 GLANCE_PASS。

B．将 glance 用户链接到 service 租户和 admin 角色：
$ keystone user-role-add --user glance --tenant service --role admin

C．创建 glance 服务：
**$ keystone service-create --name glance --type image **
--description "OpenStack Image Service"
+-------------+----------------------------------+
| Property | Value |
+-------------+----------------------------------+
description	OpenStack Image Service
enabled	True
id	23f409c4e79f4c9e9d23d809c50fbacf
name	glance
type	image
+-------------+----------------------------------+

4、创建身份服务端点
**$ keystone endpoint-create **
**--service-id $(keystone service-list | awk '/ image / {print $2}') **
**--publicurl http://controller:9292 **
**--internalurl http://controller:9292 **
**--adminurl http://controller:9292 **
--region regionOne
+-------------+----------------------------------+
| Property | Value |
+-------------+----------------------------------+
adminurl	http://controller:9292
id	a2ee818c69cb475199a1ca108332eb35
internalurl	http://controller:9292
publicurl	http://controller:9292
region	regionOne
service_id	23f409c4e79f4c9e9d23d809c50fbacf
+-------------+----------------------------------+

安装和配置 Image 服务组件
1、安装包

yum install openstack-glance python-glanceclient

2、编辑/etc/glance/glance-api.conf 文件
A．在[database]部分，配置数据库访问：
[database]
...
connection = mysql://glance: GLANCE_DBPASS@controller/glance

GLANCE_DBPASS 替换为您选择的镜像服务数据库密码。

B．在[keystone_authtoken]和[paste_deploy]部分，配置身份服务访问：
[keystone_authtoken]
...
auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357
admin_tenant_name = service
admin_user = glance
admin_password = GLANCE_PASS
[paste_deploy]
...
flavor = keystone

GLANCE_PASS 替换为您选择的 glance 用户密码。

请注意
注释掉任何 auth_host 、auth_port、auth_protocol 选项，因为 identity_uri 选项将替换他们。

C.（可选）在[DEFAULT]部分，启用详细记录来帮助解决问题
[DEFAULT]
...
verbose = True

3、编辑/etc/glance/glance-registry.conf 文件
A．在[database]部分，配置数据库访问：
[database]
...
connection = mysql://glance: GLANCE_DBPASS@controller/glance

GLANCE_DBPASS 替换为您选择的镜像服务数据库密码。

B．在[keystone_authtoken]和[paste_deploy]部分，配置身份服务访问：
[keystone_authtoken]
...

auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357
admin_tenant_name = service
admin_user = glance
admin_password = GLANCE_PASS
[paste_deploy]
...
flavor = keystone

GLANCE_PASS 替换为您选择的 glance 用户身份服务密码。

请注意
注释掉任何 auth_host、 auth_port、auth_protocol 选项，因为 identity_uri 选项将替换他们。

C．在[DEFAULT]部分，启用详细的日志记录，来协助排除潜在的故障：
[DEFAULT]
...
verbose = True

4、填充 Image 服务数据库
su -s /bin/sh -c "glance-manage db_sync" glance

完成安装
•启动镜像服务和配置随系统启动：
systemctl enable openstack-glance-api.service
systemctl enable openstack-glance-registry.service
systemctl start openstack-glance-api.service
systemctl start openstack-glance-registry.service

验证操作
本节描述如何使用 CirrOS 验证镜像服务，一个小的 Linux 镜像可以帮助测试你的 OpenStack 部署。
1、创建和切换到一个本地临时目录
$ mkdir /tmp/images
$ cd /tmp/images

2、下载镜像到本地的临时目录中
$ wget http://cdn.download.cirros-cloud.net/0.3.3/cirros-0.3.3-x86_64-disk.img

3、执行 admin 凭证文件

$ source admin-openrc.sh

4、上传镜像到镜像服务

**$ glance image-create --name "cirros-0.3.3-x86_64" --file cirros-0.3.3-x86_64-disk.img **
--disk-format qcow2 --container-format bare --is-public True --progress

```
[==============================>] 100%
+------------------+--------------------------------------+
| Property | Value |
+------------------+--------------------------------------+
| checksum | 133eae9fb1c98f45894a4e60d8736619 |
| container_format | bare |
| created_at | 2014-10-10T13:14:42 |
| deleted | False |
| deleted_at | None |
| disk_format | qcow2 |
| id | acafc7c0-40aa-4026-9673-b879898e1fc2 |
| is_public | True |
| min_disk | 0 |
| min_ram | 0 |
| name | cirros-0.3.3-x86_64 |
| owner | ea8c352d253443118041c9c8b8416040 |
| protected | False |
| size | 13200896 |
| status | active |
| updated_at | 2014-10-10T13:14:43 |
| virtual_size | None |
+------------------+--------------------------------------+
```

5、确定镜像上传和验证属性

$ glance image-list

```
+--------------------------------------+---------------------
+-------------+-----------------+----------+--------+
| ID | Name | Disk Format
 | Container Format | Size | Status |
+--------------------------------------+---------------------
+-------------+-----------------+----------+--------+
| acafc7c0-40aa-4026-9673-b879898e1fc2 | cirros-0.3.3-x86_64 | qcow2
 | bare | 13200896 | active |
+--------------------------------------+---------------------
+-------------+-----------------+----------+--------+
```

6、删除本地临时目录

$ rm -r /tmp/images

四、添加计算服务

安装和配置控制器节点
配置的先决条件
1、创建数据库，完成这些步骤
A．登陆数据库：
$ mysql -u root –p

B．创建 Nova 数据库：
CREATE DATABASE nova;

C．授予权限：
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
  IDENTIFIED BY ' NOVA_DBPASS';
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
  IDENTIFIED BY ' NOVA_DBPASS';

用一个合适的密码替换 NOVA_DBPASS。

D．退出数据库

2、 执行 admin 凭证文件
$ source admin-openrc.sh

3、创建身份服务凭证，完成以下步骤
A．创建 nova 用户：
$ keystone user-create --name nova --pass NOVA_PASS
+----------+----------------------------------+
| Property | Value |
+----------+----------------------------------+
email	
enabled	True
id	387dd4f7e46d4f72965ee99c76ae748c
name	nova
username	nova

用一个合适的密码替换 NOVA_PASS。

B．Nova 用户链接到 service 租户和 admin 角色：
$ keystone user-role-add --user nova --tenant service --role admin

C．创建 nova 服务：

$ keystone service-create --name nova --type compute --description "OpenStack Compute"

```
+-------------+----------------------------------+
| Property | Value |
+-------------+----------------------------------+
| description | OpenStack Compute |
| enabled | True |
| id | 6c7854f52ce84db795557ebc0373f6b9 |
| name | nova |
| type | compute |
+-------------+----------------------------------+
```

4、创建计算服务端点

```
$ keystone endpoint-create \
  --service-id $(keystone service-list | awk '/ compute / {print $2}') \
  --publicurl http://controller:8774/v2/%\(tenant_id\)s \
  --internalurl http://controller:8774/v2/%\(tenant_id\)s \
  --adminurl http://controller:8774/v2/%\(tenant_id\)s \
  --region regionOne
```

```
+-------------+----------------------------------+
| Property | Value |
+-------------+----------------------------------+
| adminurl | http://controller:8774/v2/%(tenant_id)s |
| id | c397438bd82c41198ec1a9d85cb7cc74 |
| internalurl | http://controller:8774/v2/%(tenant_id)s |
| publicurl | http://controller:8774/v2/%(tenant_id)s |
| region | regionOne |
| service_id | 6c7854f52ce84db795557ebc0373f6b9 |
+-------------+----------------------------------+
```

安装和配置计算控制器组件

1、安装包

```
# yum install openstack-nova-api openstack-nova-cert openstack-novaconductor \
  openstack-nova-console openstack-nova-novncproxy openstack-novascheduler \
  python-novaclient
```

2、编辑/etc/nova/nova.conf 文件，并完成以下操作

A．在[database]部分，配置数据库访问：

```
[database]
...
connection = mysql://nova: NOVA_DBPASS@controller/nova
```

用您选择的密码替换 NOVA_DBPASS 计算数据库。

B．在[DEFAULT]部分，配置 RabbitMQ message broker 访问:
[DEFAULT]
...
rpc_backend = rabbit
rabbit_host = controller
rabbit_password = RABBIT_PASS

用您选择的 RabbitMQ 账户密码替换 RABBIT_PASS。

C．在[keystone_authtoken]部分，配置身份服务访问:
[keystone_authtoken]
...
auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357
admin_tenant_name = service
admin_user = nova
admin_password = NOVA_PASS

用您选择的 nova 用户密码替换 NOVA_PASS。

请注意
注释掉任何 auth_host 、auth_port、auth_protocol 选项，因为 identity_uri 选项将替换他们。

D．在[DEFAULT]部分，my_ip 选项配置为使用控制器节点管理接口的 IP 地址：
[DEFAULT]
...
my_ip = 10.0.0.11

E．在[DEFAULT]部分，VNC 代理配置为使用控制器节点管理接口的 IP 地址：
[DEFAULT]
...
vncserver_listen = 10.0.0.11
vncserver_proxyclient_address = 10.0.0.11

F．在[glance]部分，配置镜像的位置服务：
[glance]
...
host = controller

G.（可选）在[DEFAULT]部分，启用详细日志记录协助故障排除。
[DEFAULT]

...
verbose = True

3、填充 Nova 计算数据库
su -s /bin/sh -c "nova-manage db sync" nova

完成安装

•启动计算服务和配置随系统自启动
systemctl enable openstack-nova-api.service
systemctl enable openstack-nova-cert.service
systemctl enable openstack-nova-consoleauth.service
systemctl enable openstack-nova-scheduler.service
systemctl enable openstack-nova-conductor.service
systemctl enable openstack-nova-novncproxy.service
systemctl start openstack-nova-api.service
systemctl start openstack-nova-cert.service
systemctl start openstack-nova-consoleauth.service
systemctl start openstack-nova-scheduler.service
systemctl start openstack-nova-conductor.service
systemctl start openstack-nova-novncproxy.service

安装和配置一个计算节点

该服务支持多种虚拟机监控程序（Hypervisor）或 vm 部署实例。为简单起见，本配置使用 QEMU 虚拟机监控程序的 KVM 扩展计算节点。支持虚拟机的硬件加速。你可以通过计算节点横向扩展您的环境。

安装和配置计算程序组件

1、安装包
yum install openstack-nova-compute

2、编辑/etc/nova/nova.conf 文件，并完成以下操作
A．在[DEFAULT]部分，配置 RabbitMQ message broker 访问：
[DEFAULT]
...
rpc_backend = rabbit
rabbit_host = controller
rabbit_password = RABBIT_PASS

用您选择的 RabbitMQ 账户密码替换 RABBIT_PASS。

B．在[keystone_authtoken]部分，配置身份服务访问：
[keystone_authtoken]

...
auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357
admin_tenant_name = service
admin_user = nova
admin_password = NOVA_PASS

用您选择的 nova 用户密码替换 NOVA_PASS。

请注意
注释掉任何 auth_host 、auth_port、auth_protocol 选项，因为 identity_uri 选项将替换他们。

C．在[DEFAULT]部分，配置 my_ip 选项
[DEFAULT]
...
my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS

在你的计算节点，使用管理接口的 IP 地址替换 management_interface_ip_address，通常 10.0.0.31 为例子中的第一个体系结构的节点。

D．在[DEFAULT]部分，启用和配置远程控制台访问：
[DEFAULT]
...
vnc_enabled = True
vncserver_listen = 0.0.0.0
vncserver_proxyclient_address = MANAGEMENT_INTERFACE_IP_ADDRESS
novncproxy_base_url = http://controller:6080/vnc_auto.html

服务器组件监听所有 IP 地址，代理组件只监听管理接口计算节点的 IP 地址。你可以使用一个 web 浏览器来访问远程主机计算节点上的实例。

用计算节点管理网络接口 IP 地址替换 MANAGEMENT_INTERFACE_IP_ADDRESS，通常 10.0.0.31 为例子中的第一个体系结构的节点。

E．在[glance]部分，配置镜像的位置服务
[glance]
...
host = controller

F．(可选)在[DEFAULT]部分，启用详细日志记录协助故障排除。
[DEFAULT]
...
verbose = True

完成安装

1、确定您的计算节点支持虚拟机硬件加速

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

如果该命令返回一个值或更多，则你的计算节点支持硬件加速，通常不需要额外的配置。

相反，如果该命令返回零个值，则计算节点不支持硬件加速，你必须配置 libvirt KVM 使用 QEMU。

在/etc/nova/nova.conf 文件，编辑[libvirt]部分如下：

[libvirt]

...

virt_type = qemu

2、启动计算服务并配置随系统自启动

systemctl enable libvirtd.service

systemctl start libvirtd.service

systemctl enable openstack-nova-compute.service

systemctl start openstack-nova-compute.service

验证操作

本节描述如何验证操作计算服务。

请注意

控制器节点上执行这些命令。

1、执行 admin 凭证文件

$ source admin-openrc.sh

2、 验证服务组件列表

$ nova service-list

```
+----+------------------+------------+----------+---------+------
+----------------------------+-----------------+
| Id | Binary | Host | Zone | Status | State |
  Updated_at | Disabled Reason |
+----+------------------+------------+----------+---------+------
+----------------------------+-----------------+
| 1 | nova-conductor | controller | internal | enabled | up |
  2014-09-16T23:54:02.000000 | - |
| 2 | nova-consoleauth | controller | internal | enabled | up |
  2014-09-16T23:54:04.000000 | - |
| 3 | nova-scheduler | controller | internal | enabled | up |
```

```
 2014-09-16T23:54:07.000000 | - |
| 4 | nova-cert | controller | internal | enabled | up |
 2014-09-16T23:54:00.000000 | - |
| 5 | nova-compute | compute1 | nova | enabled | up |
 2014-09-16T23:54:06.000000 | - |
+----+------------------+------------+----------+---------+------
+----------------------------+-------------------+
```

3、查看镜像列表服务

$ nova image-list

```
+--------------------------------------+-------------------------+--------
+--------+
| ID | Name | Status |
 Server |
+--------------------------------------+-------------------------+--------
+--------+
| acafc7c0-40aa-4026-9673-b879898e1fc2 | cirros-0.3.3-x86_64 | ACTIVE |
 |
+--------------------------------------+-------------------------+--------
+--------+
```

五、添加一个网络组件

使用 OpenStack 网络(neutron)

安装和配置控制器节点

配置的先决条件

1、 创建数据库

A．登录 MySQL：

```
$ mysql -u root –p
```

B．创建 neutron 数据库：

```
CREATE DATABASE neutron;
```

C．授予权限：

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

用一个合适的密码替换 NEUTRON_DBPASS。

D．退出数据库

2、 执行 admin 凭证文件
$ source admin-openrc.sh

3、 创建身份服务凭证
创建 neutron 用户:
$ keystone user-create --name neutron --pass NEUTRON_PASS

```
+----------+----------------------------------+
| Property | Value |
+----------+----------------------------------+
| email | |
| enabled | True |
| id | 7fd67878dcd04d0393469ef825a7e005 |
| name | neutron |
| username | neutron |
+----------+----------------------------------+
```

用一个合适的密码替换 NEUTRON_PASS。

B．neutron 用户链接到 service 租户和 admin 角色：
$ keystone user-role-add --user neutron --tenant service --role admin

C．创建 neutron 服务：
**$ keystone service-create --name neutron --type network **
--description "OpenStack Networking"

```
+-------------+----------------------------------+
| Property | Value |
+-------------+----------------------------------+
| description | OpenStack Networking |
| enabled | True |
| id | 6369ddaf99a447f3a0d41dac5e342161 |
| name | neutron |
| type | network |
+-------------+----------------------------------+
```

D．创建身份服务：

```
$ keystone endpoint-create \
  --service-id $(keystone service-list | awk '/ network / {print $2}')
  \
  --publicurl http://controller:9696 \
  --adminurl http://controller:9696 \
  --internalurl http://controller:9696 \
  --region regionOne
```

```
+-------------+----------------------------------+
| Property | Value |
+-------------+----------------------------------+
| adminurl | http://controller:9696 |
| id | fa18b41938a94bf6b35e2c152063ee21 |
| internalurl | http://controller:9696 |
| publicurl | http://controller:9696 |
| region | regionOne |
| service_id | 6369ddaf99a447f3a0d41dac5e342161 |
+-------------+----------------------------------+
```

安装网络组件

```
# yum install openstack-neutron openstack-neutron-ml2 python-neutronclient
  which
```

配置网络服务器组件

•编辑/etc/neutron/neutron.conf 文件，并完成以下步骤

A．在[database]部分，配置数据库访问：

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

NEUTRON_DBPASS 替换为您选择的数据库密码。

B．在[DEFAULT]部分，配置 RabbitMQ message broker 访问：

```
[DEFAULT]
...
rpc_backend = rabbit
rabbit_host = controller
rabbit_password = RABBIT_PASS
```

用您选择的 RabbitMQ 账户密码替换 RABBIT_PASS。

C．在[DEFAULT]和[keystone_authtoken]部分，配置身份服务访问：

```
[DEFAULT]
...
auth_strategy = keystone
[keystone_authtoken]
...
auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357
admin_tenant_name = service
admin_user = neutron
```

admin_password = NEUTRON_PASS

使用你选择的 neutron 用户密码替换 NEUTRON_PASS。

请注意
注释掉 auth_host、auth_port 和 auth_protocol，因为 identity_uri 选项将替换它们。

在[DEFAULT]部分，使模块化 2 层（ML2）插件和路由器的 IP 地址服务重叠：
[DEFAULT]
...
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True

E．在[DEFAULT]部分，配置网络通知计算网络拓扑结构变化：
[DEFAULT]
...
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
nova_url = http://controller:8774/v2
nova_admin_auth_url = http://controller:35357/v2.0
nova_region_name = regionOne
nova_admin_username = nova
nova_admin_tenant_id = SERVICE_TENANT_ID
nova_admin_password = NOVA_PASS

分别使用 service 租户(id)、nova 用户密码替换 SERVICE_TENANT_ID 和 NOVA_PASS。

获取 service 租户(id)：
$ source admin-openrc.sh
$ keystone tenant-get service
+-------------+----------------------------------+
| Property | Value |
+-------------+----------------------------------+
description	Service Tenant
enabled	True
id	f727b5ec2ceb4d71bad86dfc414449bf
name	service
+-------------+----------------------------------+

F．(可选)在[DEFAULT]部分，启用详细日志记录协助故障排除
[DEFAULT]
...
verbose = True

配置模块 2 层（ML2）插件

编辑/etc/neutron/plugins/ml2/ml2_conf.ini 文件，并完成以下步骤：

A．在[ML2]部分：

```
[ml2]
...
type_drivers = flat,gre
tenant_network_types  = gre
mechanism_drivers = openvswitch
```

B．在[ml2_type_gre]部分，配置隧道标识符(id)范围：

```
[ml2_type_gre]
...
tunnel_id_ranges = 1:1000
```

C．在[securitygroup]部分：

```
[securitygroup]
...
enable_security_group = True
enable_ipset = True
firewall_driver = neutron.agent.linux.iptables_firewall.
OVSHybridIptablesFirewallDriver
```

配置计算使用网络

•编辑/etc/nova/nova.conf 文件，并完成以下步骤

A．[DEFAULT]部分：

```
[DEFAULT]
...
network_api_class = nova.network.neutronv2.api.API
security_group_api = neutron
linuxnet_interface_driver = nova.network.linux_net.
LinuxOVSInterfaceDriver
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

B．在[neutron]部分，配置访问参数：

```
[neutron]
...
url = http://controller:9696
auth_strategy = keystone
admin_auth_url = http://controller:35357/v2.0
admin_tenant_name = service
admin_username = neutron
```

admin_password = NEUTRON_PASS
用您选择的 neutron 用户密码替换 NEUTRON_PASS。

完成安装

1、网络服务初始化脚本需要一个符号链接/etc/neutron/plugin.ini 插件配置文件中，打开指向名为 ML2 层的 /etc/neutron/plugins/ml2/ml2_conf.ini 文件。如果此符号链接不存在，就使用以下命令创建它

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

2、填充 neutron 数据库

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

3、重新启动计算服务

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

4、启动网络服务和配置随系统自启动

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

验证操作

请注意

控制器节点上执行这些命令。

1、执行 admin 凭证文件

```
$ source admin-openrc.sh
```

2、验证 neutron-server 过程

```
$ neutron ext-list
+-----------------------+-----------------------------------------------+
| alias | name |
+-----------------------+-----------------------------------------------+
| security-group | security-group |
| l3_agent_scheduler | L3 Agent Scheduler |
| ext-gw-mode | Neutron L3 Configurable external gateway mode |
| binding | Port Binding |
| provider | Provider Network |
| agent | agent |
| quotas | Quota management support |
```

dhcp_agent_scheduler	DHCP Agent Scheduler
l3-ha	HA Router extension
multi-provider	Multi Provider Network
external-net	Neutron external network
router	Neutron L3 Router
allowed-address-pairs	Allowed Address Pairs
extraroute	Neutron Extra Route
extra_dhcp_opt	Neutron Extra DHCP opts
dvr	Distributed Virtual Router
+----------------------+--+

安装和配置网络节点

网络节点主要处理内部和外部的路由以及虚拟网络的 DHCP 服务。

配置的先决条件

在安装和配置 OpenStack 网络之前，您必须配置特定的内核网络参数。

1、编辑/etc/sysctl.conf 文件，包含以下参数

net.ipv4.ip_forward=1
net.ipv4.conf.all.rp_filter=0
net.ipv4.conf.default.rp_filter=0

2、使配置生效

sysctl –p

安装网络组件

yum install openstack-neutron openstack-neutron-ml2 openstack-neutronopenvswitch ipset

配置网络常见的组件

常见的网络组件配置，包括身份验证机制、message broker、插件。

•编辑/etc/neutron/neutron.conf 文件，并完成以下步骤

A．在[database]部分，注释掉任何 connection 选项，因为网络节点不直接访问数据库。

B．在[DEFAULT]部分，配置 RabbitMQ message broker 访问：

[DEFAULT]
...
rpc_backend = rabbit
rabbit_host = controller
rabbit_password = RABBIT_PASS

用您选择的 RabbitMQ 账户密码替换 RABBIT_PASS。

C．在[DEFAULT]和[keystone_authtoken]部分，配置身份服务访问：

```
[DEFAULT]
...
auth_strategy = keystone
[keystone_authtoken]
...
auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357
admin_tenant_name = service
admin_user = neutron
admin_password = NEUTRON_PASS
```

NEUTRON_PASS 替换为您选择的 neutron 用户密码。

请注意

注释掉任何 auth_host 、auth_port、auth_protocol 选项，因为 identity_uri 选项将替换他们。

D．在[DEFAULT]部分，在模块化 2 层中打开名为（ML2）的插件，使路由器服务和 IP 地址重叠：

```
[DEFAULT]
...
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True
```

E．(可选)在[DEFAULT]部分，启用详细日志记录

```
[DEFAULT]
...
verbose = True
```

配置模块 2 层（ML2）插件

ML2 插件使用开放 VSwitch（OVS）机制（agent），为实例（VM）建立虚拟网络框架。

•编辑/etc/neutron/plugins/ml2/ml2_conf.ini 文件，并完成以下操作

A．在[ML2]部分：

```
[ml2]
...
type_drivers = flat,gre
tenant_network_types = gre
mechanism_drivers = openvswitch
```

B．在[ml2_type_flat]部分，配置外部网络：

[ml2_type_flat]
...
flat_networks = external

C．在[ml2_type_gre]部分，配置隧道标识符(id)范围：
[ml2_type_gre]
...
tunnel_id_ranges = 1:1000

D．[securitygroup]部分：
[securitygroup]
...
enable_security_group = True
enable_ipset = True
firewall_driver = neutron.agent.linux.iptables_firewall.
OVSHybridIptablesFirewallDriver

E．在(ovs)部分，配置 Open vSwitch(ovs)代理：
[ovs]
...
local_ip = INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS
tunnel_type = gre
enable_tunneling = True
bridge_mappings = external:br-ex

用网络节点上的隧道接口的实例 IP 地址替换
INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS。

配置第三层(L3)代理
第三层(L3)代理为虚拟网络提供路由服务。

•编辑/etc/neutron/l3_agent.ini 文件，并完成以下步骤
A．在[DEFAULT]部分，配置驱动程序和网络名称支持空间：
[DEFAULT]
...
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
use_namespaces = True
external_network_bridge = br-ex

B．(可选)在[DEFAULT]部分，启用详细日志记录：
[DEFAULT]
...
verbose = True

配置 DHCP 代理

DHCP 代理为虚拟网络提供 DHCP 服务。

1、编辑/etc/neutron/dhcp_agent.ini 文件

A．在[DEFAULT]部分，配置驱动程序和启用名称空间：

```
[DEFAULT]
...
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
use_namespaces = True
```

B．(可选)在[DEFAULT]部分，启用详细日志记录：

```
[DEFAULT]
...
verbose = True
```

2、(可不填)

A．编辑/etc/neutron/dhcp_agent.ini 文件：

•在[DEFAULT]部分，配置 dnsmasq 文件

```
[DEFAULT]
...
dnsmasq_config_file = /etc/neutron/dnsmasq-neutron.conf
```

B．创建和编辑/etc/neutron/dnsmasq-neutron.conf 文件：

•启用 DHCP MTU 选项(26)和配置可达 1454 个字节

```
dhcp-option-force=26,1454
```

C．杀死任何现有 dnsmasq 过程：

```
# pkill dnsmasq
```

配置元数据代理

元数据代理提供配置信息，如凭证实例。

1、编辑/etc/neutron/metadata_agent.ini 文件

A．在[DEFAULT]部分，配置访问参数：

```
[DEFAULT]
...
auth_url = http://controller:5000/v2.0
auth_region = regionOne
admin_tenant_name = service
admin_user = neutron
admin_password = NEUTRON_PASS
```

用您选择的 neutron 用户密码替换 NEUTRON_PASS。

B．在[DEFAULT]部分，配置元数据：
[DEFAULT]
...
nova_metadata_ip = controller

C．在[DEFAULT]部分，配置元数据代理共享密钥：
[DEFAULT]
...
metadata_proxy_shared_secret = METADATA_SECRET

使用一个合适的元数据代理密码替换 METADATA_SECRET。

D．在[DEFAULT]部分，启用详细日志记录：
[DEFAULT]
...
verbose = True

2、在控制器节点上，编辑/etc/nova/nova.conf 文件
•在[neutron]部分
[neutron]
...
service_metadata_proxy = True
metadata_proxy_shared_secret = METADATA_SECRET

使用你选择的元数据代理密码替换 METADATA_SECRET。

3、在控制器节点上，重新启动计算 API 服务
systemctl restart openstack-nova-api.service

配置 Open vSwitch(OVS)服务
1、启动 OVS 服务和配置随系统自启动
systemctl enable openvswitch.service
systemctl start openvswitch.service

2、添加外部桥接
ovs-vsctl add-br br-ex

3、添加一个端口到外部桥接，连接到物理外部网络
INTERFACE_NAME 替换为实际的接口名称。例如 eth2 或 ens256。
ovs-vsctl add-port br-ex INTERFACE_NAME

在你的环境中，暂时禁用客户外部网络接口
ethtool -K INTERFACE_NAME gro off

完成安装

1、网络服务初始化脚本/etc/neutron/plugin.ini 插件配置文件，链接指向名为 ML2 /etc/neutron/plugins/ml2/ml2_conf.ini 文件。如果此符号链接不存在，就创建它。

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

由于包装错误，Open vSwitch 代理初始化脚本显式地看起来，配置文件 /etc/neutron/plugin.ini 不是一个符号链接，插件配置文件中打开指向名为 ML2。运行以下命令来解决这个问题

```
# cp /usr/lib/systemd/system/neutron-openvswitch-agent.service \
  /usr/lib/systemd/system/neutron-openvswitch-agent.service.orig
# sed -i 's,plugins/openvswitch/ovs_neutron_plugin.ini,plugin.ini,g' \
  /usr/lib/systemd/system/neutron-openvswitch-agent.service
```

2、启动网络服务和配置随系统启动

```
# systemctl enable neutron-openvswitch-agent.service
# systemctl enable neutron-l3-agent.service
# systemctl enable neutron-dhcp-agent.service
# systemctl enable neutron-metadata-agent.service
# systemctl enable neutron-ovs-cleanup.service
# systemctl start neutron-openvswitch-agent.service
# systemctl start neutron-l3-agent.service
# systemctl start neutron-dhcp-agent.service
# systemctl start neutron-metadata-agent.service
```

验证操作

请注意

控制器节点上执行这些命令。

1、执行 admin 凭证文件

```
$ source admin-openrc.sh
```

2、查看 neutron 代理列表

```
$ neutron agent-list
+--------------------------------------+--------------------+---------
+-------+----------------+---------------------------+
| id | agent_type | host |
  alive | admin_state_up | binary |
+--------------------------------------+--------------------+---------
+-------+----------------+---------------------------+
| 30275801-e17a-41e4-8f53-9db63544f689  | Metadata agent | network
  | :-) | True | neutron-metadata-agent  |
| 4bd8c50e-7bad-4f3b-955d-67658a491a15  | Open vSwitch agent | network
  | :-) | True | neutron-openvswitch-agent  |
| 756e5bba-b70f-4715-b80e-e37f59803d20  | L3 agent | network
```

```
  | :-) | True | neutron-l3-agent |
| 9c45473c-6d6d-4f94-8df1-ebd0b6838d5f  | DHCP agent | network
  | :-) | True | neutron-dhcp-agent |
+--------------------------------------+--------------------+---------
+-------+---------------+-----------------------+
```

安装和配置计算节点

计算节点处理连接和安全组的实例。

配置先决条件

在安装和配置 OpenStack 网络之前，您必须配置特定的内核网络参数。

1、编辑/etc/sysctl.conf 文件包含以下参数

```
net.ipv4.conf.all.rp_filter=0
net.ipv4.conf.default.rp_filter=0
```

2、使配置生效

```
# sysctl –p
```

安装网络组件

```
# yum install openstack-neutron-ml2 openstack-neutron-openvswitch  ipset
```

配置网络常见的组件

常见的网络组件配置包括身份验证机制、message broker、插件。

•编辑/etc/neutron/neutron.conf 文件

A． 在[database] 部分，注释掉任何 connection 选项，因为计算节点不直接访问数据库。

B．在[DEFAULT]部分，配置 RabbitMQ message broker 访问：

```
[DEFAULT]
...
rpc_backend = rabbit
rabbit_host = controller
rabbit_password = RABBIT_PASS
```

用您选择的 RabbitMQ 帐户密码替换 RABBIT_PASS。

C．在[DEFAULT]和[keystone_authtoken]部分，配置身份服务访问：

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

[keystone_authtoken]
...
auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357
admin_tenant_name = service
admin_user = neutron
admin_password = NEUTRON_PASS

用您选择的 neutron 用户密码替换 NEUTRON_PASS。

请注意
注释掉任何 auth_host 、auth_port、auth_protocol 选项，因为 identity_uri 选项将替换他们。

D．在[DEFAULT]部分，模块化 2 层中打开名为（ML2）插件，使路由器服务和 IP 地址重叠：
[DEFAULT]
...
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True

E．（可选）
[DEFAULT]
...
verbose = True

配置模块 2 层（ML2）插件
插件（ML2）使用 Open vSwitch 中打开的名为 ML2(OVS)机制(agent)来构建虚拟网络的框架实例。

•编辑/etc/neutron/plugins/ml2/ml2_conf.ini 文件
A．在[ml2]部分：
[ml2]
...
type_drivers = flat,gre
tenant_network_types = gre
mechanism_drivers = openvswitch

B．在[ml2_type_gre]部分，配置隧道标识符(id)范围：
[ml2_type_gre]
...
tunnel_id_ranges = 1:1000

C．在[securitygroup]部分：
[securitygroup]
...
enable_security_group = True
enable_ipset = True
firewall_driver = neutron.agent.linux.iptables_firewall.
OVSHybridIptablesFirewallDriver

D．在(ovs)部分，配置 Open vSwitch(ov)代理：
[ovs]
...
local_ip = INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS
tunnel_type = gre
enable_tunneling = True

用计算节点上的实例隧道网络接口 IP 地址替换
INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS。

配置 Open vSwitch(ov)服务
•启动 OVS 服务和配置随机启动
systemctl enable openvswitch.service
systemctl start openvswitch.service

使用网络配置计算
•编辑/etc/nova/nova.conf 文件
A．在[DEFAULT]部分：
[DEFAULT]
...
network_api_class = nova.network.neutronv2.api.API
security_group_api = neutron
linuxnet_interface_driver = nova.network.linux_net.
LinuxOVSInterfaceDriver
firewall_driver = nova.virt.firewall.NoopFirewallDriver

B．在[neutron]部分，配置访问参数：
[neutron]
...
url = http://controller:9696
auth_strategy = keystone
admin_auth_url = http://controller:35357/v2.0
admin_tenant_name = service
admin_username = neutron

admin_password = NEUTRON_PASS

用您选择的 neutron 用户身份服务密码替换 NEUTRON_PASS。

完成安装

1、做软连接

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

由于包装错误，Open vSwitch 代理初始化脚本显式地看起来 Open vSwitch 插件配置文件不是一个符号链接，/etc/neutron/plugin.ini 插件配置文件中打开指向名为 ML2。运行以下命令来解决这个问题

```
# cp /usr/lib/systemd/system/neutron-openvswitch-agent.service \
  /usr/lib/systemd/system/neutron-openvswitch-agent.service.orig
# sed -i 's,plugins/openvswitch/ovs_neutron_plugin.ini,plugin.ini,g' \
  /usr/lib/systemd/system/neutron-openvswitch-agent.service
```

2、重启计算服务

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

3、启动 Open vSwitch(ovs)代理和配置随系统启动

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

验证操作

请注意

控制器节点上执行这些命令

1、执行 admin 凭证文件

```
$ source admin-openrc.sh
```

2、查看 neutron 代理列表

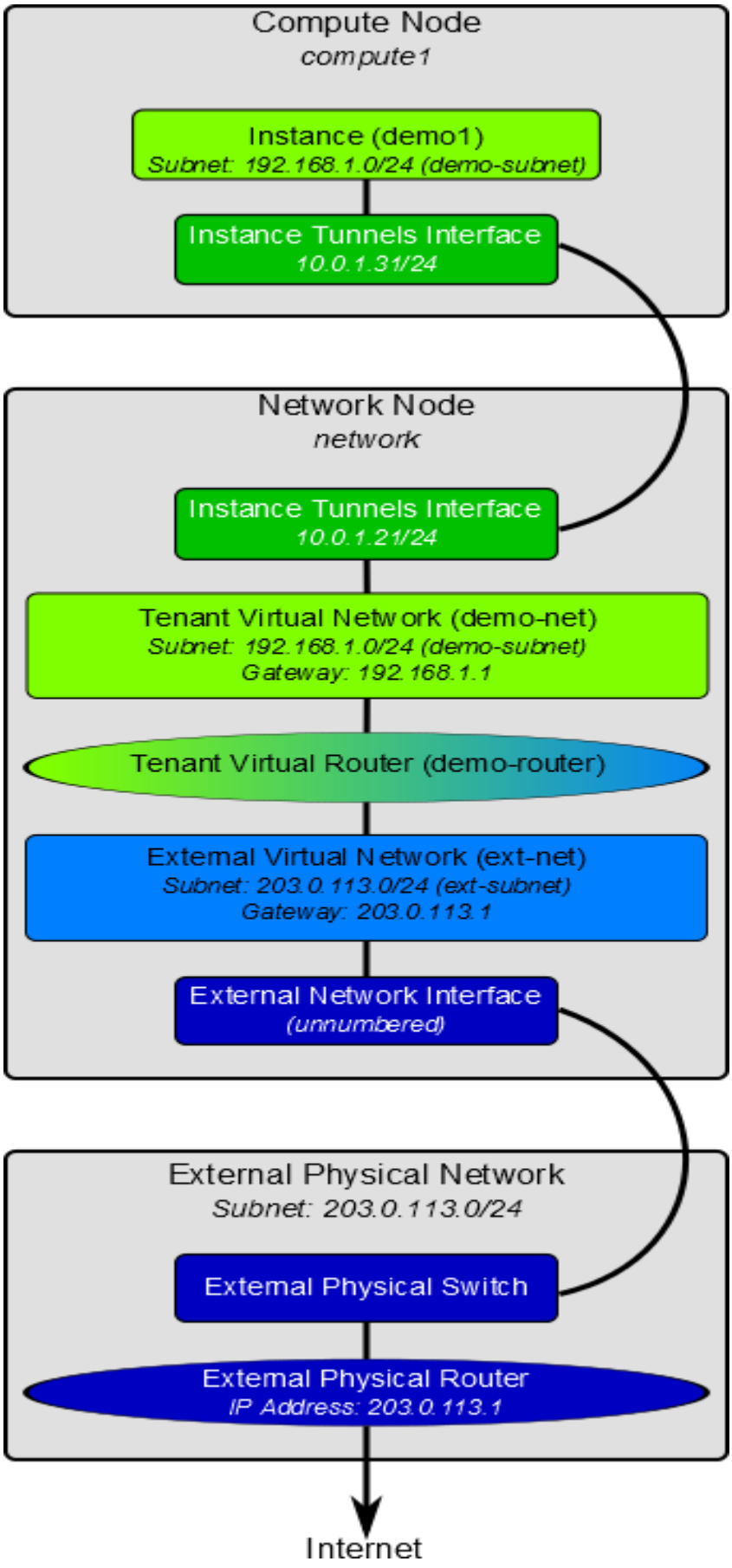
```
$ neutron agent-list
+--------------------------------------+--------------------+---------
+-------+----------------+---------------------------+
| id | agent_type | host |
  alive | admin_state_up | binary |
+--------------------------------------+--------------------+---------
+-------+----------------+---------------------------+
...
| a5a49051-05eb-4b4f-bfc7-d36235fe9131 | Open vSwitch agent | compute1
  | :-) | True | neutron-openvswitch-agent |
+--------------------------------------+--------------------+---------
```

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

创建初始网络

你的第一个实例启动前，你必须创造必要的实例连接的虚拟网络基础设施，包括外部网络和租户网络。在创建这个基础设施时，我们建议您在继续之前验证连接性和解决任何问题。

图 1 最初的网络


Compute Node
compute1
Instance (demo1)
Subnet: 192.168.1.0/24 (demo-subnet)
Instance Tunnels Interface
10.0.1.31/24
Network Node
network
Instance Tunnels Interface
10.0.1.21/24
Tenant Virtual Network (demo-net)
Subnet: 192.168.1.0/24 (demo-subnet)
Gateway: 192.168.1.1
Tenant Virtual Router (demo-router)
External Virtual Network (ext-net)
Subnet: 203.0.113.0/24 (ext-subnet)
Gateway: 203.0.113.1
External Network Interface
(unnumbered)
External Physical Network
Subnet: 203.0.113.0/24
External Physical Switch
External Physical Router
IP Address: 203.0.113.1
Internet

外部网络

外部网络通常提供互联网接入您的实例。默认情况下，这个网络只允许使用网络地址转换(NAT) 实例上网。您可以启用互联网接入使用浮动 IP 地址和个人实例，合适的安全组规则。admin 租户拥有这个网络，因为它提供了外部网络访问多个租户。您还必须启用共享允许访问这些租户。

请注意

控制器节点上执行这些命令。

创建外部网络

1、执行 admin 凭证文件

$ source admin-openrc.sh

2、创建网络

**$ neutron net-create ext-net --shared --router:external True **

--provider:physical_network external --provider:network_type flat

Created a new network:

```
+---------------------------+--------------------------------------+
| Field | Value |
+---------------------------+--------------------------------------+
| admin_state_up | True |
| id | 893aebb9-1c1e-48be-8908-6b947f3237b3 |
| name | ext-net |
| provider:network_type | flat |
| provider:physical_network | external |
| provider:segmentation_id | |
| router:external | True |
| shared | True |
| status | ACTIVE |
| subnets | |
| tenant_id | 54cd044c64d5408b83f843d63624e0d8 |
+---------------------------+--------------------------------------+
```

像一个物理网络，一个虚拟网络需要一个子网分配给它。外部网络共享相同的子网和网关。

网络节点上的外部接口。你应该指定一个独立子网、路由器和浮动 IP 地址来防止干扰其他外部网络设备。

创建一个外部网络子网

•创建子网：

$ neutron subnet-create ext-net --name ext-subnet \

--allocation-pool start=FLOATING_IP_START,end=FLOATING_IP_END \

--disable-dhcp --gateway EXTERNAL_NETWORK_GATEWAY EXTERNAL_NETWORK_CIDR

使用你想要分配的范围浮动 IP 地址取代 FLOATING_IP_START 、FLOATING_IP_END 第一个和最后一个 IP 地址。使用物理网络的子网取代 EXTERNAL_NETWORK_CIDR。使用外部物理网络的网关取代 EXTERNAL_NETWORK_GATEWAY，通常最后一个字节为“1”。你应该在这个子网中禁用 DHCP，因为实例不直接连接到外部网络和浮动 IP 地址，需要手动分配。

例如，使用范围为 203.0.113.0/24 的浮动 IP 地址
203.0.113.101　203.0.113.200：

```
$ neutron subnet-create ext-net --name ext-subnet \
  --allocation-pool start=203.0.113.101,end=203.0.113.200 \
  --disable-dhcp --gateway 203.0.113.1 203.0.113.0/24
Created a new subnet:
+-------------------
+-----------------------------------------------------------+
| Field | Value
  |
+-------------------
+-----------------------------------------------------------+
| allocation_pools | {"start": "203.0.113.101", "end": "203.0.113.200"}
  |
| cidr | 203.0.113.0/24
  |
| dns_nameservers |
  |
| enable_dhcp | False
  |
| gateway_ip | 203.0.113.1
  |
| host_routes |
  |
| id | 9159f0dc-2b63-41cf-bd7a-289309da1391
  |
| ip_version | 4
  |
| ipv6_address_mode |
  |
| ipv6_ra_mode |
  |
| name | ext-subnet
  |
| network_id | 893aebb9-1c1e-48be-8908-6b947f3237b3
  |
| tenant_id | 54cd044c64d5408b83f843d63624e0d8
```

|
+------------------

租户网络

租户网络提供内部网络访问实例。使用这种类型的网络架构访问其他租户。因为 demo 租户拥有这个网络，它只提供网络访问实例。

请注意

控制器节点上执行这些命令。

创建租户网络

1、执行 demo 凭证文件

$ source demo-openrc.sh

2、创建网络

$ neutron net-create demo-net

Created a new network:

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

| Field | Value |

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

| admin_state_up | True |

| id | ac108952-6096-4243-adf4-bb6615b3de28 |

| name | demo-net |

| router:external | False |

| shared | False |

| status | ACTIVE |

| subnets | |

| tenant_id | cdef0071a0194d19ac6bb63802dc9bae |

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

租户网络与外部网络一样，也需要一个子网。你可以指定任何有效子网，因为架构将使用租户网络。默认情况下，这个子网将使用 DHCP，这样你的实例可以获得 IP 地址。

租户网络上创建一个子网

•创建子网：

$ neutron subnet-create demo-net --name demo-subnet \

--gateway TENANT_NETWORK_GATEWAY TENANT_NETWORK_CIDR

使用你想要的网关和租户子网分别替换 TENANT_NETWORK_GATEWAY 和 TENANT_NETWORK_CIDR。

使用 192.168.1.0/24 示例：

**$ neutron subnet-create demo-net --name demo-subnet **
--gateway 192.168.1.1 192.168.1.0/24

Created a new subnet:

```
+-------------------
+-----------------------------------------------+-----------+
| Field | Value
  |
+-------------------
+-----------------------------------------------+-----------+
| allocation_pools | {"start": "192.168.1.2", "end": "192.168.1.254"}
  |
| cidr | 192.168.1.0/24
  |
| dns_nameservers |
  |
| enable_dhcp | True
  |
| gateway_ip | 192.168.1.1
  |
| host_routes |
  |
| id | 69d38773-794a-4e49-b887-6de6734e792d
  |
| ip_version | 4
  |
| ipv6_address_mode |
  |
| ipv6_ra_mode |
  |
| name | demo-subnet
  |
| network_id | ac108952-6096-4243-adf4-bb6615b3de28
  |
| tenant_id | cdef0071a0194d19ac6bb63802dc9bae
```

一个虚拟路由器通过两个或多个虚拟网络之间的网络流量。每个路由器需要一个或多个接口或网关提供特定的网络。

在本例中，您将创建一个路由器并附上你的租户和外部网络。

1、创建路由器

$ neutron router-create demo-router

Created a new router:

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

```
+-----------------------+--------------------------------------+
| admin_state_up | True |
| external_gateway_info | |
| id | 635660ae-a254-4feb-8993-295aa9ec6418 |
| name | demo-router |
| routes | |
| status | ACTIVE |
| tenant_id | cdef0071a0194d19ac6bb63802dc9bae |
+-----------------------+--------------------------------------+
```

2、demo 租户子网连接到路由器：

$ neutron router-interface-add demo-router demo-subnet

Added interface b1a894fd-aee8-475c-9262-4342afdc1b58 to router demorouter.

3、连接路由器到外部网络通过设置它作为网关：

$ neutron router-gateway-set demo-router ext-net

Set gateway for router demo-router

验证连接

我们建议您检查网络连接并进一步解决任何之前的问题。例子中，外部网络子网使用 203.0.113.0/24，租户路由器网关应该占据最低的浮动 IP 地址范围，即 IP 地址 203.0.113.101。如果你正确配置外部物理网络和虚拟网络，你应该能从任何主机上 ping 通您的外部物理网络 IP 地址。

请注意

如果你正在构建 OpenStack 节点作为虚拟机，您必须配置虚拟机监控程序允许在外部网络混杂模式。

验证网络连接

•ping 租户路由器网关：

$ ping -c 4 203.0.113.101

```
PING 203.0.113.101 (203.0.113.101) 56(84) bytes of data.
64 bytes from 203.0.113.101: icmp_req=1 ttl=64 time=0.619 ms
64 bytes from 203.0.113.101: icmp_req=2 ttl=64 time=0.189 ms
64 bytes from 203.0.113.101: icmp_req=3 ttl=64 time=0.165 ms
64 bytes from 203.0.113.101: icmp_req=4 ttl=64 time=0.216 ms
--- 203.0.113.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.165/0.297/0.619/0.187 ms
```

七、添加 Dashboard

请注意
Dashboard 使用 VNC 客户端，浏览器必须支持 HTML5、Canvas 和 HTML5 WebSockets。
浏览器支持 noVNC 的详细信息，请参阅 https://github.com/kanaka/noVNC/blob/master/README.md 和 https://github.com/kanaka/noVNC/。

安装和配置
本节描述如何安装和配置控制器节点上的 Dashboard。

安装 Dashboard 组件
•安装包：
yum install openstack-dashboard httpd mod_wsgi memcached pythonmemcached

配置 dashboard
•编辑/etc/openstack-dashboard / local_settings 文件
A．在 OpenStack 服务控制器节点上配置使用 dashboard：
OPENSTACK_HOST = "controller"

B．允许所有主机访问 dashboard：
ALLOWED_HOSTS = ['*']

C．配置 memcached 会话存储服务：
```
CACHES = {
 'default': {
 'BACKEND': 'django.core.cache.backends.memcached.
MemcachedCache',
 'LOCATION': '127.0.0.1:11211',
 }
}
```

请注意
注释掉其他会话存储配置。

D．(可选)配置时区：
TIME_ZONE = "TIME_ZONE"
用一个合适的时区标识符替换 TIME_ZONE。

完成安装
1、在 RHEL 和 CentOS、SELinux 上允许 web 服务器配置为连接到 OpenStack 服务
setsebool -P httpd_can_network_connect on

2、由于包错误，dashboard CSS 无法正常加载。运行以下命令来解决这个问题

chown -R apache:apache /usr/share/openstack-dashboard/static

有关更多信息，请参见错误报告。

3、启动 web 服务器和会话存储服务，配置随系统启动
systemctl enable httpd.service memcached.service
systemctl start httpd.service memcached.service

验证操作
本节描述如何验证操作 dashboard。
1、使用 web 浏览器访问 dashboard：http://controller/dashboard。
2、使用 admin 或 demo 用户验证凭证。

八、添加块存储服务

OpenStack 块存储
OpenStack 块存储服务(cinder)将持久性存储（你可以理解为移动硬盘，随取随用）添加到一个虚拟机。块存储卷管理为 OpenStack 计算实例提供卷。服务还支持卷快照管理、数量和类型。

安装和配置控制器节点
本节介绍如何安装和配置块存储服务，代号 cinder，控制器节点上。这个可选服务至少需要一个额外的节点。

配置的先决条件
在安装和配置块存储服务之前，您必须创建一个数据库和身份服务凭证包括端点。
1、使用数据库
A．登录数据库：
$ mysql -u root –p

B．创建 cinder 库：
CREATE DATABASE cinder;

C．授予权限：
GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' \
  IDENTIFIED BY ' CINDER_DBPASS';
mysql> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' \
  IDENTIFIED BY ' CINDER_DBPASS';

用一个合适的 cinder 数据库密码替换 CINDER_DBPASS。

D．退出登录

2、执行 admin 凭证文件
$ source admin-openrc.sh

3、创建身份服务凭证，完成以下步骤
A．创建一个 cinder 用户：
$ keystone user-create --name cinder --pass CINDER_PASS
```
+----------+----------------------------------+
| Property | Value |
+----------+----------------------------------+
| email | |
| enabled | True |
| id | 881ab2de4f7941e79504a759a83308be |
| name | cinder |
| username | cinder |
+----------+----------------------------------+
```

用一个合适的 cinder 用户密码替换 CINDER_PASS。

B．cinder 用户链接到 service 租户和 admin 角色：
$ keystone user-role-add --user cinder --tenant service --role admin

C．创建 cinder 服务：
**$ keystone service-create --name cinder --type volume **
--description "OpenStack Block Storage"
```
+-------------+----------------------------------+
| Property | Value |
+-------------+----------------------------------+
| description | OpenStack Block Storage |
| enabled | True |
| id | 1e494c3e22a24baaafcaf777d4d467eb |
| name | cinder |
| type | volume |
+-------------+----------------------------------+
```
**$ keystone service-create --name cinderv2 --type volumev2 **
--description "OpenStack Block Storage"
```
+-------------+----------------------------------+
| Property | Value |
+-------------+----------------------------------+
| description | OpenStack Block Storage |
| enabled | True |
| id | 16e038e449c94b40868277f1d801edb5 |
| name | cinderv2 |
```

```
| type | volumev2 |
+-------------+----------------------------------+
```

请注意
块存储服务需要两个不同的服务支持 API 版本 1 和 2。

D．创建块存储服务端点：

```
$ keystone endpoint-create \
  --service-id $(keystone service-list | awk '/ volume / {print $2}')
  \
  --publicurl http://controller:8776/v1/%\(tenant_id\)s \
  --internalurl http://controller:8776/v1/%\(tenant_id\)s \
  --adminurl http://controller:8776/v1/%\(tenant_id\)s \
  --region regionOne
+-------------+----------------------------------+
| Property | Value |
+-------------+----------------------------------+
| adminurl | http://controller:8776/v1/%(tenant_id)s |
| id | d1b7291a2d794e26963b322c7f2a55a4 |
| internalurl | http://controller:8776/v1/%(tenant_id)s |
| publicurl | http://controller:8776/v1/%(tenant_id)s |
| region | regionOne |
| service_id | 1e494c3e22a24baaafcaf777d4d467eb |
+-------------+----------------------------------+
$ keystone endpoint-create \
  --service-id $(keystone service-list | awk '/ volumev2 / {print
  $2}') \
  --publicurl http://controller:8776/v2/%\(tenant_id\)s \
  --internalurl http://controller:8776/v2/%\(tenant_id\)s \
  --adminurl http://controller:8776/v2/%\(tenant_id\)s \
  --region regionOne
+-------------+----------------------------------+
| Property | Value |
+-------------+----------------------------------+
| adminurl | http://controller:8776/v2/%(tenant_id)s |
| id | 097b4a6fc8ba44b4b10d4822d2d9e076 |
| internalurl | http://controller:8776/v2/%(tenant_id)s |
| publicurl | http://controller:8776/v2/%(tenant_id)s |
| region | regionOne |
| service_id | 16e038e449c94b40868277f1d801edb5 |
+-------------+----------------------------------+
```

请注意
块存储服务需要两个不同的端点支持 API 版本 1 和 2。

安装和配置块存储控制器组件

1、安装包

yum install openstack-cinder python-cinderclient python-oslo-db

2、编辑/etc/cinder/cinder.conf 文件

A．在[database]部分，配置数据库访问：

[database]

...

connection = mysql://cinder: CINDER_DBPASS@controller/cinder

CINDER_DBPASS 替换为您选择的块存储数据库密码。

B．在[DEFAULT]部分，配置 RabbitMQ message broker 访问：

[DEFAULT]

...

rpc_backend = rabbit

rabbit_host = controller

rabbit_password = RABBIT_PASS

用您选择的 RABBIT 帐户密码替换 RABBIT_PASS。

C．在[DEFAULT]和[keystone_authtoken]部分，配置身份服务访问：

[DEFAULT]

...

auth_strategy = keystone

[keystone_authtoken]

...

auth_uri = http://controller:5000/v2.0

identity_uri = http://controller:35357

admin_tenant_name = service

admin_user = cinder

admin_password = CINDER_PASS

CINDER_PASS 替换为您选择的 cinder 用户密码。

请注意

注释掉任何 auth_host 、auth_port、auth_protocol 选项，因为 identity_uri 选项会替换他们。

D．在[DEFAULT]部分，my_ip 选项配置为控制器节点的管理接口 IP 地址：

[DEFAULT]

...

my_ip = 10.0.0.11

E．在[DEFAULT]部分，启用详细日志记录：
[DEFAULT]
...
verbose = True

3、填充 cinder 数据库
su -s /bin/sh -c "cinder-manage db sync" cinder

完成安装
•启动块存储服务和配置随系统启动
systemctl enable openstack-cinder-api.service openstack-cinderscheduler.service
systemctl start openstack-cinder-api.service openstack-cinder-scheduler.
service

配置一个块存储服务节点
控制器节点上配置服务后，配置一个块存储服务节点，其中包含服务卷的磁盘。您可以配置 OpenStack 使用不同的存储系统（如 ceph、gfs、nfs 等）。这个例子中，使用一个 LVM。

配置操作系统
控制器节点：
•设置主机名 block1 和使用 10.0.0.41 管理网络接口 IP 地址。确保控制器节点的 IP 地址、主机名和块存储服务节点/etc/hosts 文件中列出每个系统。

使用“网络时间协议(NTP)”，从控制器节点保持时间同步。

创建一个逻辑卷
1、 安装 LVM 包
yum install lvm2

请注意
默认一些发行版包括 LVM。

2、 启动 LVM 的元数据服务和配置随系统启动
systemctl enable lvm2-lvmetad.service
systemctl start lvm2-lvmetad.service

3、创建 LVM 物理卷和卷组。本指南假定第二个磁盘/dev/sdb 被用于此目的
pvcreate /dev/sdb
vgcreate cinder-volumes /dev/sdb

4、在/etc/lvm/lvm. conf 文件部分，添加 r/.*/防止 LVM 扫描设备所使用的虚拟机
devices {

...
filter = ["a/sda1/", "a/sdb/", "r/.*/"]
...
}

请注意
在这个例子中，/dev/sda1 卷的操作系统为节点驻留，虽然/dev/sdb 留给 cinder-volumes。

安装和配置块存储服务节点组件
1、安装块存储服务
yum install openstack-cinder targetcli python-oslo-db MySQL-python

2、编辑/etc/cinder/cinder.conf 文件
A．在[database]部分，配置数据库访问：
[database]
...
connection = mysql://cinder: CINDER_DBPASS@controller/cinder

CINDER_DBPASS 替换为您选择的块存储数据库密码。

B．在[DEFAULT]部分，配置 RabbitMQ message broker 访问：
[DEFAULT]
...
rpc_backend = rabbit
rabbit_host = controller
rabbit_password = RABBIT_PASS

用您选择的 RABBIT 账户密码替换 RABBIT_PASS

C．在[DEFAULT]和[keystone_authtoken]部分，配置身份服务访问：
[DEFAULT]
...
auth_strategy = keystone
[keystone_authtoken]
...
auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357
admin_tenant_name = service
admin_user = cinder
admin_password = CINDER_PASS

CINDER_PASS 替换为您选择的 cinder 用户密码。

请注意
注释掉任何 auth_host、 auth_port、auth_protocol 选项，因为 identity_uri 选项将替换他们。

D．在[DEFAULT]部分，配置 my_ip 选择：
[DEFAULT]
...
my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS

用存储节点的管理网络接口 IP 地址替换 MANAGEMENT_INTERFACE_IP_ADDRESS，通常为 10.0.0.41 例子中的第一个体系结构的节点。

E．在[DEFAULT]部分，配置镜像的位置服务：
[DEFAULT]
...
glance_host = controller

F．在[DEFAULT]部分，块存储配置为使用 iSCSI lioadm 服务：
[DEFAULT]
...
iscsi_helper = lioadm

G．在[DEFAULT]部分，启用详细日志记录：
[DEFAULT]
...
verbose = True

完成安装
1、使用目标服务
systemctl enable target.service

2、开始目标服务
systemctl start target.service

3、启动和配置 cinder volume 服务随系统启动
systemctl enable openstack-cinder-volume.service
systemctl start openstack-cinder-volume.service

验证操作
本节描述如何通过创建一个卷，验证操作块存储服务。

请注意
控制器节点上执行这些命令。

1、执行 demo 凭证文件
$ source demo-openrc.sh

2、创建一个 1GB 的卷
$ cinder create --display-name demo-volume1 1

```
+---------------------+--------------------------------------+
| Property | Value |
+---------------------+--------------------------------------+
| attachments | [] |
| availability_zone | nova |
| bootable | false |
| created_at | 2014-10-14T23:11:50.870239 |
| display_description | None |
| display_name | demo-volume1 |
| encrypted | False |
| id | 158bea89-07db-4ac2-8115-66c0d6a4bb48 |
| metadata | {} |
| size | 1 |
| snapshot_id | None |
| source_volid | None |
| status | creating |
| volume_type | None |
+---------------------+--------------------------------------+
```

3、验证创建的卷
$ cinder list

```
--------------------------------------+-----------+----------------+-----
+-------------+----------+------------+
| ID | Status | Display Name | Size |
 Volume Type | Bootable | Attached to |
+--------------------------------------+-----------+----------------+-----
+-------------+----------+------------+
| 158bea89-07db-4ac2-8115-66c0d6a4bb48 | available | demo-volume1 | 1 |
 None | false | |
+--------------------------------------+-----------+----------------+-----
+-------------+----------+------------+
```

如果状态意味着不可用，则在/var/log/cinder 检查日志获取更多信息。

九、添加对象存储（Swift）

表 1 硬件建议

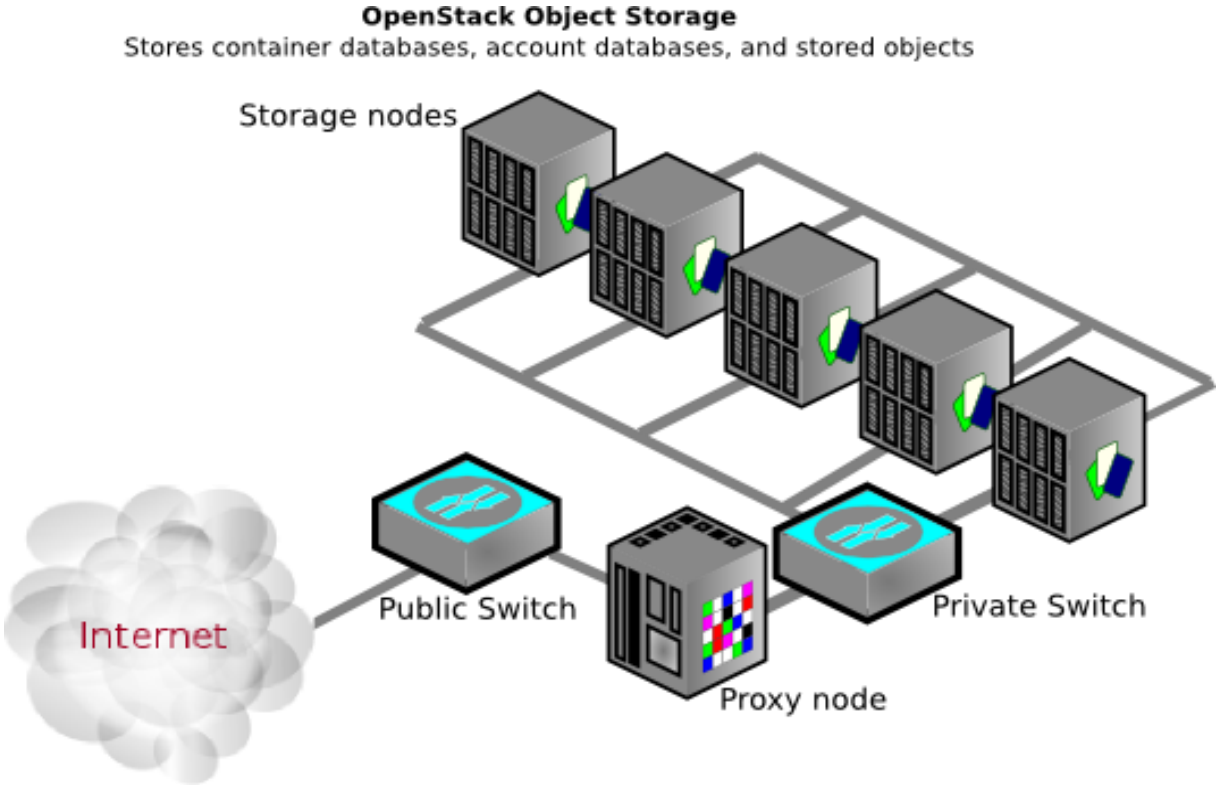
Server	Recommended Hardware	Notes
Object Storage object servers	Processor: dual quad core Memory: 8 or 12 GB RAM Disk space: optimized for cost per GB Network: one 1 GB Network Interface Card (NIC)	The amount of disk space depends on how much you can fit into the rack efficiently. You want to optimize these for best cost per GB while still getting industry-standard failure rates. At Rackspace, our storage servers are currently running fairly generic 4U servers with 24 2T SATA drives and 8 cores of processing power. RAID on the storage drives is not required and not recommended. Swift's disk usage pattern is the worst case possible for RAID, and performance degrades very quickly using RAID 5 or 6. As an example, Rackspace runs Cloud Files storage servers with 24 2T SATA drives and 8 cores of processing power. Most services support either a worker or concurrency value in the settings. This allows the services to make effective use of the cores available.
Object Storage container/account servers	Processor: dual quad core Memory: 8 or 12 GB RAM Network: one 1 GB Network Interface Card (NIC)	Optimized for IOPS due to tracking with SQLite databases.
Object Storage proxy server	Processor: dual quad core Network: one 1 GB Network Interface Card (NIC)	Higher network throughput offers better performance for supporting many API requests. Optimize your proxy servers for best CPU performance. The Proxy Services are more CPU and network I/O intensive. If you are using 10 GB networking to the proxy, or are terminating SSL traffic at the proxy, greater CPU power is required.

示例架构

在生产环境中，对象存储服务至少需要两个代理节点和 5 个存储节点。为简单起见，本指南使用最少的架构与代理服务运行在现有 OpenStack 控制器节点和两个存储节点。然而，这些概念仍然适用如下。

- 节点（Node）：一个主机运行一个或多个 OpenStack 对象存储服务。
- 节点代理（Proxy node）：代理服务。
- 存储节点（Storage node）：运行账户、容器和对象服务。包含了 SQLite 数据库。
- 环（Ring）：一组 OpenStack 对象存储数据之间的映射到物理设备。
- 复制（Replica）：复制一个对象。默认情况下集群中维护三份。
- 区域(可选)：一个逻辑上独立的集群，与独立的故障特征。
- 地区(可选)：一个逻辑独立上的集群部分，代表不同的物理位置，如城市或国家。类似于区域，但代表的是物理位置集群部分而不是逻辑部分。

增加可靠性和性能，您可以添加额外的代理服务器。
图 2 显示了一个架构最小的生产环境



安装对象存储
本文演示了如何通过使用以下类型的节点安装一个集群。

•一个代理节点运行 swift-proxy-server 流程。代理服务器代理请求到适当的存储节点中。
•五个存储节点运行流程控制存储器的帐户数据库、容器数据库、以及实际的存储对象。

请注意

最初可以使用更少的存储节点，但建议至少五个生产集群。

一般的安装步骤

1、创建一个 Swift 用户，使用 service 租户并给用户 admin 角色

```
$ keystone user-create --name swift --pass SWIFT_PASS
$ keystone user-role-add --user swift --tenant service --role admin
```

用一个合适的 Swift 用户密码替换 SWIFT_PASS。

2、创建一个对象存储服务

```
$ keystone service-create --name swift --type object-store \
  --description "OpenStack Object Storage"
+-------------+----------------------------------+
| Property | Value |
+-------------+----------------------------------+
| description | OpenStack Object Storage |
| id | eede9296683e4b5ebfa13f5166375ef6 |
| name | swift |
| type | object-store |
+-------------+----------------------------------+
```

3、指定一个 API 端点对象存储服务，通过使用返回的服务 ID。指定一个端点为公共 API 提供 url、内部 API、管理 API。在本指南中，控制器使用主机名

```
$ keystone endpoint-create \
  --service-id $(keystone service-list | awk '/ object-store / {print
  $2}') \
  --publicurl 'http://controller:8080/v1/AUTH_%(tenant_id)s' \
  --internalurl 'http://controller:8080/v1/AUTH_%(tenant_id)s' \
  --adminurl http://controller:8080 \
  -region regionOne
+-------------+-----------------------------------------+
| Property | Value |
+-------------+-----------------------------------------+
| adminurl | http://controller:8080/ |
| id | 9e3ce428f82b40d38922f242c095982e |
| internalurl | http://controller:8080/v1/AUTH_%(tenant_id)s |
| publicurl | http://controller:8080/v1/AUTH_%(tenant_id)s |
| region | regionOne |
| service_id | eede9296683e4b5ebfa13f5166375ef6 |
+-------------+-----------------------------------------+
```

4、在所有节点上创建配置目录

```
# mkdir -p /etc/swift
```

5、在所有节点上创建/etc/swift/swift.conf
[swift-hash]
# random unique string that can never change (DO NOT LOSE)
swift_hash_path_prefix = xrfuniounenqjnw
swift_hash_path_suffix = fLIbertYgibbitZ

请注意
在/etc/swift/swift.conf 前缀和后缀的值应被设置为散列值，此文件必须在集群中的每个节点上一样！

接下来，设置您的存储节点和代理节点。本示例使用身份服务为共同的认证部件。

安装和配置存储节点

请注意
对象存储的工作原理上，它支持扩展属性的任何文件系统（XATTRS）。 XFS 显示了最佳的整体性能。

经过大量的测试和基准测试，在 Rackspace 公司这也是已经彻底测试的唯一文件系统。

1、安装存储节点程序包
yum install openstack-swift-account openstack-swift-container \
  openstack-swift-object xfsprogs xinetd

2、在每个设备节点上，建立 XFS 容量（/dev/sdb 为例）用于存储。每个驱动一个分区。例如，在 12 盘的服务器中可以使用一个或两个磁盘操作系统。其他 10 或 11 盘应该分配一个分区，然后格式化 XFS。
fdisk /dev/sdb
mkfs.xfs /dev/sdb1
echo "/dev/sdb1 /srv/node/sdb1 xfs noatime,nodiratime,nobarrier,logbufs=
8 0 0" >> /etc/fstab
mkdir -p /srv/node/sdb1
mount /srv/node/sdb1
chown -R swift:swift /srv/node

3、创建 /etc/rsyncd.conf
uid = swift
gid = swift
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
address = STORAGE_LOCAL_NET_IP
[account]
max connections = 2

path = /srv/node/
read only = false
lock file = /var/lock/account.lock
[container]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/container.lock
[object]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/object.lock

4、(可选)如果你想要单独的 rsync 和流量复制网络，则设置 STORAGE_REPLICATION_NET_IP 代替 STORAGE_LOCAL_NET_IP
address = STORAGE_REPLICATION_NET_IP

5、在/etc/xinetd.d/rsync 编辑以下行
disable = no

6、启动 xinetd 服务
systemctl start xinetd.service

请注意
rsync 服务不需要身份验证，所以在本地运行它的私用网络。

7、设置 bind_ip 配置选项 storage_local_net_ip。
在 /etc/swift 目录中，编辑 account-server.conf、acontainer-server.conf、object-server.conf 文件。在每个文件的[DEFAULT]部分，更新如下
[DEFAULT]
bind_ip = STORAGE_LOCAL_NET_IP
...

8、确保 Swift 用户拥有所有配置文件
chown -R swift:swift /etc/swift

9、创建快速缓存目录并设置其权限
mkdir -p /var/swift/recon
chown -R swift:swift /var/swift/recon

安装和配置代理节点
代理服务器的容器考虑每个请求和查找位置，并将请求路由正确。代理服务器还

处理 API 请求。通过配置/etc/swift/proxy-server.conf 文件启用帐户管理。

请注意
对象存储进程运行在一个单独的用户和组，设定的配置选项称为 swift:swift。默认的用户是 swift。

1、安装 swift-proxy 服务
yum install openstack-swift-proxy memcached python-swiftclient pythonkeystone-auth-token

2、编辑/etc/sysconfig/memcached 文件
OPTIONS="-l PROXY_LOCAL_NET_IP"

3、启动 memcached 服务和配置随系统自启动
systemctl enable memcached.service
systemctl start memcached.service

4、编辑文件/etc/swift/proxy-server.conf
```
[DEFAULT]
bind_port = 8080
user = swift
[pipeline:main]
pipeline = catch_errors gatekeeper healthcheck proxy-logging cache
  authtoken keystoneauth proxy-logging proxy-server
[app:proxy-server]
use = egg:swift#proxy
allow_account_management = true
account_autocreate = true
[filter:keystoneauth]
use = egg:swift#keystoneauth
operator_roles = _member_,admin,swiftoperator
[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
# Delaying the auth decision is required to support token-less
# usage for anonymous referrers ('.r:*').
delay_auth_decision = true
# auth_* settings refer to the Keystone server
auth_protocol = http
auth_host = controller
auth_uri = http://controller:5000

# the service tenant and swift username and password created in Keystone
admin_tenant_name = service
admin_user = swift
```

admin_password = SWIFT_PASS
[filter:healthcheck]
use = egg:swift#healthcheck
[filter:cache]
use = egg:swift#memcache
set log_name = cache
[filter:catch_errors]
use = egg:swift#catch_errors
[filter:gatekeeper]
use = egg:swift#gatekeeper
[filter:proxy-logging]
use = egg:swift#proxy_logging

请注意
如果你运行多个 memcache 服务器，把多个 IP：端口列表，配置在 /etc/swift/proxy-server.conf 文件的[filter:cache]部分：
10.1.2.3:11211,10.1.2.4:11211
只有代理服务器使用 memcache。

警告
keystoneclient.middleware.auth_token：您必须配置 auth_uri 公共身份端点。否则，客户可能无法验证管理端点。

5、创建帐户、容器和对象的环（rings）。创建几个参数，值 18 的参数表示 2 ^18，表示该分区的最大值。你希望你的整个环使用存储总量。值 3 表示每个对象的副本数目，最后的值“1”是小时数来限制移动分区活动不止一次。
cd /etc/swift
swift-ring-builder account.builder create 18 3 1
swift-ring-builder container.builder create 18 3 1
swift-ring-builder object.builder create 18 3 1

6、在每个节点上的每一个存储设备添加条目，每个环。
swift-ring-builder account.builder add
zZONE-STORAGE_LOCAL_NET_IP:6002[RSTORAGE_REPLICATION_NET_IP:6005]/DEVICE
100

swift-ring-builder container.builder add
zZONE-STORAGE_LOCAL_NET_IP_1:6001[RSTORAGE_REPLICATION_NET_IP:6004]/DEVICE
100

swift-ring-builder object.builder add

zZONE-STORAGE_LOCAL_NET_IP_1:6000[RSTORAGE_REPLICATION_NET_IP:6003]/DEVICE
100

请注意
如果你不想使用专用网络复制，你必须省略可选参数STORAGE_REPLICATION_NET_IP。

例如，如果一个 IP 地址为 10.0.0.1 的存储节点表示在区域 1 分区存储，复制网络节点地址为 10.0.1.1。这个分区的/srv/node/sdb1 挂载点和/etc/rsyncd.conf 路径是/srv/node/，那么 sdb1 命令是：
swift-ring-builder account.builder add z1-10.0.0.1:6002R10.0.1.1:6005/sdb1 100
swift-ring-builder container.builder add z1-10.0.0.1:6001R10.0.1.1:6004/sdb1 100
swift-ring-builder object.builder add z1-10.0.0.1:6000R10.0.1.1:6003/sdb1 100

请注意
如果你假定五区，每个区的一个节点在启动区 1。为每一个额外的节点，增加区 1。

7、验证每个环内容
swift-ring-builder account.builder
swift-ring-builder container.builder
swift-ring-builder object.builder

8、平衡环
swift-ring-builder account.builder rebalance
swift-ring-builder container.builder rebalance
swift-ring-builder object.builder rebalance

请注意
平衡环可能需要一些时间。

9、 account.ring.gz、container.ring.gz 和 object.ring.gz 文件复制到每个代理服务器和存储节点的/etc/swift 中。

10、确保 Swift 用户拥有所有配置文件
chown -R swift:swift /etc/swift

11、启动代理服务和配置随系统启动
systemctl enable openstack-swift-proxy.service
systemctl start openstack-swift-proxy.service

存储节点上启动服务

现在你可以在每个存储节点上开始服务，运行以下命令：

```
# for service in \
openstack-swift-object openstack-swift-object-replicator
openstack-swiftobject-updater openstack-swift-object-auditor \
  openstack-swift-container openstack-swift-container-replicator
openstackswift-container-updater openstack-swift-container-auditor \
  openstack-swift-account openstack-swift-account-replicator
openstack-swiftaccount-reaper openstack-swift-account-auditor; do \
  systemctl enable $service.service; systemctl start $service.service; done
```

请注意

迅速启动所有服务，运行以下命令：

```
# swift-init all start
```

了解 swift-init 命令，运行：

```
$ man swift-init
```

验证安装

您可以运行这些命令从代理服务器或其他服务器访问身份服务。

1、执行 admin 凭证文件

```
$ source admin-openrc.sh
```

2、运行以下 swift 命令

```
$ swift stat
Account: AUTH_11b9758b7049476d9b48f7a91ea11493
Containers: 0
  Objects: 0
  Bytes: 0
Content-Type: text/plain; charset=utf-8
X-Timestamp: 1381434243.83760
X-Trans-Id: txdcdd594565214fb4a2d33-0052570383
X-Put-Timestamp: 1381434243.83760
```

3、运行以下命令来上传文件到一个容器中。创建 test.txt 和 test2.txt。

```
$ swift upload myfiles test.txt
$ swift upload myfiles test2.txt
```

4、运行以下 Swift 命令从 myfiles 容器下载所有文件

```
$ swift download myfiles
test2.txt [headers 0.267s, total 0.267s, 0.000s MB/s]
test.txt [headers 0.271s, total 0.271s, 0.000s MB/s]
```

添加另一个代理服务器

1、在/etc/swift/proxy-server.conf 文件更新 memcache 服务器列表。对于添加的代理服务器。如果你运行多个服务器的内存缓存，在每个代理服务器的配置文件中使用多个 IP：端口列表这种模式。

```
10.1.2.3:11211,10.1.2.4:11211
[filter:cache]
use = egg:swift#memcache
memcache_servers = PROXY_LOCAL_NET_IP:11211
```

2、环信息复制到所有节点，包括新的代理节点。同时，确保所有存储节点环信息。

3、你全部同步节点后，确保管理员在/etc/swift 对于环文件所有权是正确的。

十、添加编排模块（Heat）

安装和配置业务流程

本节将介绍在控制器节点如何安装和配置业务流程模块，代号为 heat。

配置的先决条件

在安装和配置业务流程之前，您必须创建一个数据库和身份服务凭证端点。

1、创建数据库，完成这些步骤

A．登录：

```
$ mysql -u root –p
```

B．创建库：

```
CREATE DATABASE heat;
```

C．授权：

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

用一个合适的 heat 数据库密码替换 HEAT_DBPASS。

D．退出数据库

2、执行 admin 凭证文件

```
$ source admin-openrc.sh
```

3、创建服务身份凭证，完成以下步骤
A．创建 Heat 用户：
$ keystone user-create --name heat --pass HEAT_PASS

```
+----------+----------------------------------+
| Property | Value |
+----------+----------------------------------+
| email | |
| enabled | True |
| id | 7fd67878dcd04d0393469ef825a7e005 |
| name | heat |
| username | heat |
+----------+----------------------------------+
```

用一个合适的 heat 用户密码替换 HEAT_PASS。

B．Heat 用户链接到 service 租户和 admin 角色：

```
$ keystone user-role-add --user heat --tenant service --role admin
```

请注意
这个命令没有提供输出。

C．创建 heat_stack_user 和 heat_stack_owner 角色：

```
$ keystone role-create --name heat_stack_user
$ keystone role-create --name heat_stack_owner
```

默认情况下，用户创建的编制使用 heat_stack_user 角色。

D．创建 Heat 和 heat-cfn 服务：

```
$ keystone service-create --name heat --type orchestration \
  --description "Orchestration"
+-------------+----------------------------------+
| Property | Value |
+-------------+----------------------------------+
| description | Orchestration |
| enabled | True |
| id | 031112165cad4c2bb23e84603957de29 |
| name | heat |
| type | orchestration |
+-------------+----------------------------------+
$ keystone service-create --name heat-cfn --type cloudformation \
  --description "Orchestration"
+-------------+----------------------------------+
| Property | Value |
+-------------+----------------------------------+
```

description	Orchestration
enabled	True
id	297740d74c0a446bbff867acdccb33fa
name	heat-cfn
type	cloudformation
+-------------+----------------------------------+

E．创建服务端点：

**$ keystone endpoint-create **
--service-id $(keystone service-list | awk '/ orchestration / {print
**$2}') **
**--publicurl http://controller:8004/v1/%\(tenant_id\)s **
**--internalurl http://controller:8004/v1/%\(tenant_id\)s **
**--adminurl http://controller:8004/v1/%\(tenant_id\)s **
--region regionOne

+-------------+----------------------------------+
| Property | Value |
+-------------+----------------------------------+
adminurl	http://controller:8004/v1/%(tenant_id)s
id	f41225f665694b95a46448e8676b0dc2
internalurl	http://controller:8004/v1/%(tenant_id)s
publicurl	http://controller:8004/v1/%(tenant_id)s
region	regionOne
service_id	031112165cad4c2bb23e84603957de29
+-------------+----------------------------------+

**$ keystone endpoint-create **
--service-id $(keystone service-list | awk '/ cloudformation /
**{print $2}') **
**--publicurl http://controller:8000/v1 **
**--internalurl http://controller:8000/v1 **
**--adminurl http://controller:8000/v1 **
--region regionOne

+-------------+----------------------------------+
| Property | Value |
+-------------+----------------------------------+
adminurl	http://controller:8000/v1
id	f41225f665694b95a46448e8676b0dc2
internalurl	http://controller:8000/v1
publicurl	http://controller:8000/v1
region	regionOne
service_id	297740d74c0a446bbff867acdccb33fa
+-------------+----------------------------------+

安装和配置业务流程组件

1、运行以下命令安装包

```
# yum install openstack-heat-api openstack-heat-api-cfn openstack-heatengine \
  python-heatclient
```

2、编辑/etc/heat/heat.conf 文件

A．在[database]部分，配置数据库访问：

```
[database]
...
connection = mysql://heat: HEAT_DBPASS@controller/heat
```

用您选择的 heat 数据库密码替换 HEAT_DBPASS。

B．在[DEFAULT]部分，配置 RabbitMQ message broker 访问：

```
[DEFAULT]
...
rpc_backend = rabbit
rabbit_host = controller
rabbit_password = RABBIT_PASS
```

用您选择的 RabbitMQ 账户密码替换 RABBIT_PASS。

C．在[keystone_authtoken]和[ec2authtoken]部分，配置身份服务访问：

```
[keystone_authtoken]
...
auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357
admin_tenant_name = service
admin_user = heat
admin_password = HEAT_PASS

[ec2authtoken]
...
auth_uri = http://controller:5000/v2.0
```

用您选择的 Heat 用户密码替换 HEAT_PASS。

请注意

注释掉任何 auth_host 、auth_port、auth_protocol 选项，因为 identity_uri 选项将替换他们。

D．在[DEFAULT]部分，配置元数据等网址：

```
[DEFAULT]
...
```

heat_metadata_server_url = http://controller:8000
heat_waitcondition_server_url = http://controller:8000/v1/
waitcondition

E．(可选)在[DEFAULT]部分，启用详细日志记录
[DEFAULT]
...
verbose = True

3、填充编排数据库
su -s /bin/sh -c "heat-manage db_sync" heat

完成安装
•启动编排服务和配置随系统启动
systemctl enable openstack-heat-api.service
systemctl enable openstack-heat-api-cfn.service
systemctl enable openstack-heat-engine.service
systemctl start openstack-heat-api.service
systemctl start openstack-heat-api-cfn.service
systemctl start openstack-heat-engine.service

验证操作
本节描述如何验证操作编制模块(heat)。
1、执行 demo 凭证文件
$ source demo-openrc.sh

2、创建一个测试模板 test-stack.yml 文件包含以下内容
heat_template_version: 2013-05-23
description: Test Template
parameters:
  ImageID:
  type: string
  description: Image use to boot a server
  NetID:
  type: string
  description: Network ID for the server
resources:
  server1:

type: OS::Nova::Server
  properties:
  name: "Test server"

```
  image: { get_param: ImageID }
  flavor: "m1.tiny"
  networks:
  - network: { get_param: NetID }
outputs:
  server1_private_ip:
  description: IP address of the server in the private network
  value: { get_attr: [ server1, first_address ] }
```

3、使用 stack-create 命令创建一个堆栈的模板

```
$ NET_ID=$(nova net-list | awk '/ demo-net / { print $2 }')
$ heat stack-create -f test-stack.yml \
  -P "ImageID=cirros-0.3.3-x86_64;NetID=$NET_ID" testStack
+--------------------------------------+------------+-----------------
+---------------------+
| id | stack_name | stack_status |
  creation_time |
+--------------------------------------+------------+-----------------
+---------------------+
| 477d96b4-d547-4069-938d-32ee990834af  | testStack | CREATE_IN_PROGRESS |
  2014-04-06T15:11:01Z |
+--------------------------------------+------------+-----------------
```

4、使用 stack-list 命令来验证堆栈

```
$ heat stack-list
+--------------------------------------+------------+---------------
+---------------------+
| id | stack_name | stack_status |
  creation_time |
+--------------------------------------+------------+---------------
+---------------------+
| 477d96b4-d547-4069-938d-32ee990834af  | testStack | CREATE_COMPLETE |
  2014-04-06T15:11:01Z |
+--------------------------------------+------------+---------------
+---------------------+
```

十一 添加遥测模块（ceilometer）

安装和配置控制器节点

本节描述了如何安装和配置遥测模块，代号 ceilometer，在控制器节点上，遥测模块在您的环境中使用独立的代理收集测量每个 OpenStack 服务。

配置的先决条件

在安装和配置遥测之前，您必须安装 MongoDB，创建一个 MongoDB 数据库，创建身份服务凭证包括端点。

1、安装 MongoDB 包

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

2、编辑/etc/mongodb.conf 文件，并完成以下操作

A. 配置 bind_ip 使用控制器节点的管理接口 IP 地址：

```
bind_ip = 10.0.0.11
```

B. 默认情况下，MongoDB 会在/var/lib/mongodb/journal 中创建 1 GB 日志文件。如果你想减少每个日志文件的大小为 128 MB，限制总日记空间消耗为 512 MB，则设置 smallfiles 值：

```
smallfiles = true
```

您还可以禁用日志记录。更多信息，请参阅 MongoDB 手册。

C. 启动 MongoDB 服务和配置随系统启动：

```
# service mongod start
# chkconfig mongod on
```

3、创建 ceilometer 数据库：

```
# mongo --host controller --eval '
  db = db.getSiblingDB("ceilometer");
  db.addUser({user: "ceilometer",
  pwd: "CEILOMETER_DBPASS",
  roles: [ "readWrite", "dbAdmin" ]})'
```

用一个合适的 ceilometer 数据库密码替换 CEILOMETER_DBPASS。

4、执行 admin 凭证文件

```
$ source admin-openrc.sh
```

5、创建身份服务凭证

A. 创建 ceilometer 用户：

```
$ keystone user-create --name ceilometer --pass CEILOMETER_PASS
```

用一个合适的 ceilometer 用户密码替换 CEILOMETER_PASS。

B. ceilometer 用户链接到 service 租户和 admin 角色：

```
$ keystone user-role-add --user ceilometer --tenant service --role
  admin
```

C．创建 ceilometer 服务：

```
$ keystone service-create --name ceilometer --type metering \
  --description "Telemetry"
```

D．创建服务端点：

```
$ keystone endpoint-create \
  --service-id $(keystone service-list | awk '/ metering / {print
  $2}') \
  --publicurl http://controller:8777  \
  --internalurl http://controller:8777  \
  --adminurl http://controller:8777  \
  --region regionOne
```

安装和配置遥测模块组件

1、安装包

```
# yum install openstack-ceilometer-api openstack-ceilometer-collector \
  openstack-ceilometer-notification openstack-ceilometer-central
  openstack-ceilometer-alarm \
  python-ceilometerclient
```

2、生成一个随机值作为 ceilometer 的密码

```
# openssl rand -hex 10
```

3、编辑/etc/ceilometer/ceilometer.conf 文件

A．在[database]部分，配置数据库访问：

```
[database]
...
connection = mongodb://ceilometer:  CEILOMETER_DBPASS@controller:27017/
ceilometer
```

用您选择的 ceilometer 数据库密码替换 CEILOMETER_DBPASS。

B．在[DEFAULT]部分，配置 RabbitMQ message broker 访问：

```
[DEFAULT]
...
rpc_backend  = rabbit
rabbit_host = controller
rabbit_password = RABBIT_PASS
```

用您选择的 RabbitMQ 账户密码替换 RABBIT_PASS。

C．在[DEFAULT]和[keystone_authtoken]部分，配置你的身份服务访问：

```
[DEFAULT]
```

...
auth_strategy = keystone
[keystone_authtoken]
...
auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357
admin_tenant_name = service
admin_user = ceilometer
admin_password = CEILOMETER_PASS

用您选择的 celiometer 用户密码更换 CEILOMETER_PASS。

请注意
注释掉任何 auth_host 、auth_port、auth_protocol 选项，因为 identity_uri 选项将替换他们。

D．在[service_credentials]部分，配置服务凭证：
[service_credentials]
...
os_auth_url = http://controller:5000/v2.0
os_username = ceilometer
os_tenant_name = service
os_password = CEILOMETER_PASS

用您选择的 celiometer 用户密码更换 CEILOMETER_PASS。

E．在[publisher]部分，配置 celiometer 的密钥：
[publisher]
...
metering_secret = METERING_SECRET

使用您在上一步中生成的随机值，更换 METERING_SECRET。

完成安装
•启动遥测服务和配置随系统启动
service openstack-ceilometer-api start
service openstack-ceilometer-notification start
service openstack-ceilometer-central start
service openstack-ceilometer-collector start
service openstack-ceilometer-alarm-evaluator start
service openstack-ceilometer-alarm-notifier start
chkconfig openstack-ceilometer-api on
chkconfig openstack-ceilometer-notification on
chkconfig openstack-ceilometer-central on

chkconfig openstack-ceilometer-collector on
chkconfig openstack-ceilometer-alarm-evaluator on
chkconfig openstack-ceilometer-alarm-notifier on

安装计算代理遥测

此部分介绍如何在计算节点上安装和配置运行代理。

配置的先决条件

1、安装包

yum install openstack-ceilometer-compute python-ceilometerclient pythonpecan

2、编辑/etc/nova/nova.conf 配置文件，在[DEFAULT]部分添加以下行

```
[DEFAULT]
...
instance_usage_audit = True
instance_usage_audit_period = hour
notify_on_state_change = vm_and_task_state
notification_driver = nova.openstack.common.notifier.rpc_notifier
notification_driver = ceilometer.compute.nova_notifier
```

3、重启计算服务（Nova）

service openstack-nova-compute restart

配置计算代理遥测

编辑/etc/ceilometer/ceilometer.conf 文件，并完成以下操作：

1、在[publisher]部分，设置密钥用于遥测服务节点：

```
[publisher]
# Secret value for signing metering messages (string value)
metering_secret = CEILOMETER_TOKEN
```

使用先前创建的 controller 令牌，更换 CEILOMETER_TOKEN。

2、在[DEFAULT]部分，配置 RabbitMQ 的访问：

```
[DEFAULT]
rabbit_host = controller
rabbit_password = RABBIT_PASS
```

用您选择的 RabbitMQ 帐户密码替换 RABBIT_PASS。

3、在[keystone_authtoken]部分，配置身份服务访问

```
[keystone_authtoken]
auth_uri = http://controller:5000/v2.0
```

identity_uri = http://controller:35357
admin_tenant_name = service
admin_user = ceilometer
admin_password = CEILOMETER_PASS

使用您选择的 CEILOMETER 数据库密码更换 CEILOMETER_PASS。

请注意
注释掉 auth_host、auth_port 和 auth_protocol，因为它们会由 identity_uri 和 auth_uri 代替。

4、在[service_credentials]部分，配置服务凭据：
[service_credentials]
os_auth_url = http://controller:5000/v2.0
os_username = ceilometer
os_tenant_name = service
os_password = CEILOMETER_PASS
os_endpoint_type = internalURL

用您选择的 CEILOMETER 用户密码替换 CEILOMETER_PASS。

完成安装
•启动服务和配置随系统启动
service openstack-ceilometer-compute start
chkconfig openstack-ceilometer-compute on

配置遥测镜像服务
1、获取镜像样本
编辑/etc/glance/glance-api.conf 文件，修改[DEFAULT]部分：
notification_driver = messaging
rpc_backend = rabbit
rabbit_host = controller
rabbit_password = RABBIT_PASS

2、重新启动镜像服务与新的设置
service openstack-glance-api restart
service openstack-glance-registry restart

添加块存储服务代理遥测
1、要检索卷的样品，必须配置块存储服务将通知发送到总线。
编辑/etc/cinder/cinder.conf 文件，设置[DEFAULT]部分：

control_exchange = cinder
notification_driver = cinder.openstack.common.notifier.rpc_notifier

2、重新启动块存储服务
在控制器节点：
service openstack-cinder-api restart
service openstack-cinder-scheduler restart

在卷节点：
service openstack-cinder-volume restart

配置遥测对象存储服务
1、将对象存储在代理服务器上安装 Python-ceilometerclient 包
yum install python-ceilometerclient

2、要检索对象存储的统计数据，遥测服务需要访问对象存储 ResellerAdmin，为您的 os_username 用户和 os_tenant_name 租户：
$ keystone role-create --name ResellerAdmin
+----------+----------------------------------+
| Property | Value |
+----------+----------------------------------+
| id | 462fa46c13fd4798a95a3bfbe27b5e54 |
| name | ResellerAdmin |
+----------+----------------------------------+
**$ keystone user-role-add --tenant service --user ceilometer **
--role 462fa46c13fd4798a95a3bfbe27b5e54

3、您还必须添加如下行到/etc/swift/proxy-server.conf 文件
[filter:ceilometer]
use = egg:ceilometer#swift

4、 添加如下行到[pipeline:main]部分
[pipeline:main]
pipeline = healthcheck cache authtoken keystoneauth ceilometer proxyserver

5、添加 ceilometer 系统组的 Swift 系统用户到对象存储访问文件 ceilometer.conf 中
usermod -a -G ceilometer swift

6、添加 ResellerAdmin 到同一文件的 operator_roles 参数中
operator_roles = Member,admin,swiftoperator,_member_,ResellerAdmin

7、重新启动其新设置的服务
service openstack-swift-proxy restart

验证遥测安装

为了测试遥测安装，从镜像服务下载镜像，并且使用该 ceilometer 命令显示使用统计。

1、使用 ceilometer meter-list 命令来测试遥测访问

$ ceilometer meter-list

```
+------------+-------+-------+------------------------------------+
| Name | Type | Unit | Resource ID | User
 ID | Project ID |
+------------+-------+-------+------------------------------------+
+---------+----------------------------------+
| image | gauge | image | acafc7c0-40aa-4026-9673-b879898e1fc2 | None
 | efa984b0a914450e9a47788ad330699d |
| image.size | gauge | B | acafc7c0-40aa-4026-9673-b879898e1fc2 | None
 | efa984b0a914450e9a47788ad330699d |
+------------+-------+-------+------------------------------------+
```

2、从镜像服务下载镜像

```
$ glance image-download "cirros-0.3.3-x86_64" > cirros.img
```

3、调用 ceilometer meter-list 命令再次验证下载

$ ceilometer meter-list

```
+---------------+-------+-------+------------------------------------+
+---------+----------------------------------+
| Name | Type | Unit | Resource ID |
 User ID | Project ID |
+---------------+-------+-------+------------------------------------+
+---------+----------------------------------+
| image | gauge | image | acafc7c0-40aa-4026-9673-b879898e1fc2 |
 None | efa984b0a914450e9a47788ad330699d |
| image.download | delta | B | acafc7c0-40aa-4026-9673-b879898e1fc2 |
 None | efa984b0a914450e9a47788ad330699d |
| image.serve | delta | B | acafc7c0-40aa-4026-9673-b879898e1fc2 |
 None | efa984b0a914450e9a47788ad330699d |
| image.size | gauge | B | acafc7c0-40aa-4026-9673-b879898e1fc2 |
 None | efa984b0a914450e9a47788ad330699d |
+---------------+-------+-------+------------------------------------+
```

4、您现在可以得到各种使用统计

$ ceilometer statistics -m image.download -p 60

```
+--------+---------------------+---------------------+------
+------------+------------+------------+------------+----------
+----------------------------+----------------------------+
| Period | Period Start | Period End | Count | Min
```

| Max | Sum | Avg | Duration | Duration Start
| Duration End |
+--------+--------------------+--------------------+------
+------------+------------+-----------+-----------+---------
+--------------------------+----------------------------+
| 60 | 2013-11-18T18:08:50 | 2013-11-18T18:09:50 | 1 | 13167616.0
| 13167616.0 | 13167616.0 | 13167616.0 | 0.0 | 2013-11-18T18:09:05.
334000 | 2013-11-18T18:09:05.334000 |
+--------+--------------------+--------------------+------

十二、添加 Database 服务

数据库服务概述

数据库服务在高的性能水平提供资源隔离，并自动管理复杂的任务，例如部署、配置、打补丁、备份、恢复和监控。

本实施适用于使用数据库服务的高级别流程：

1、 管理员配置 OpenStack 使用下面的基础设施

步骤：

A．安装数据库服务。

B．每个类型的数据库镜像。例如，一个 MySQL，另一个 MongoDB。

D． 使用 trove-manage 命令导入镜像，并将它们提供给租户。

2、OpenStack 终端用户使用以下步骤部署数据库服务。

A．使用 trove create 命令来创建数据库服务实例。

B.使用 trove list 命令来获取实例 ID，其次是 trove show 命令来获得它的 IP 地址。

C．使用典型的数据库存取命令访问数据库服务实例。例如，与 MySQL：

$ mysql -u myuser -p -h TROVE_IP_ADDRESS mydb

数据库服务包括以下组件：

python-troveclient ：

command-line client

A CLI that communicates with the trove-api component.

trove-api：

component Provides an OpenStack-native RESTful API that supports JSON to provision and manage Trove instances.

trove-conductor ：

service Runs on the host, and receives messages from guest instances that want to update information on the host.

trove-taskmanager ：

service Instruments the complex system flows that support provisioning instances,

managing the lifecycle of instances,
and performing operations on instances.

trove-guestagent：

service Runs within the guest instance. Manages and performs
operations on the database itself.

安装数据库服务

此过程安装控制器节点上的数据库模块。

本节假定你已经有一个工作的 OpenStack 环境，至少安装了以下组件：Compute、Image Service、 Identity。

•如果你想要做备份和恢复，还需要对象存储。

•如果你想在块存储卷提供数据存储，还需要块存储。

安装在控制器节点上的数据库模块：

1、安装所需的软件包

yum install openstack-trove python-troveclient

2、准备 OpenStack

A．执行 admin 凭证文件：

$ source ~/admin-openrc.sh

B．创建计算使用的标识服务来验证 trove 用户。使用 service 租户，给用户 admin 角色：

$ keystone user-create --name trove --pass TROVE_PASS

$ keystone user-role-add --user trove --tenant service --role admin

用一个合适的 trove 用户密码替换 TROVE_PASS。

3、编辑配置文件

- trove.conf
- trove-taskmanager.conf
- trove-conductor.conf

A．编辑上述每个文件的[DEFAULT]部分，并设置适当的值作为 OpenStack 的服务 URL、日志记录和消息配置，以及 SQL 连接：

[DEFAULT]

log_dir = /var/log/trove

trove_auth_url = http://controller:5000/v2.0

nova_compute_url = http://controller:8774/v2

cinder_url = http://controller:8776/v1

swift_url = http://controller:8080/v1/AUTH_

sql_connection = mysql://trove: TROVE_DBPASS@controller/trove

notifier_queue_hostname = controller

B．配置数据库模块使用 RabbitMQ 消息代理。每个文件的[DEFAULT]部分，配置如下：

```
[DEFAULT]
...
rpc_backend = rabbit
rabbit_host = controller
rabbit_password = RABBIT_PASS
```

4、编辑 API-paste.ini 文件的[filter:authtoken]部分：

```
[filter:authtoken]
auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357
admin_user = trove
admin_password = ADMIN_PASS
admin_tenant_name = service
signing_dir = /var/cache/trove
```

5、编辑 trove.conf 文件，使其包含默认数据存储适当的值和网络标签的正则表达式如下所示：

```
[DEFAULT]
default_datastore = mysql
....
# Config option for showing the IP address that nova doles out
add_addresses = True
network_label_regex = ^NETWORK_LABEL$
....
```

6、编辑 trove-taskmanager.conf 文件，使其包含所需的设置连接到 OpenStack 计算服务，如下所示：

```
[DEFAULT]
....
# Configuration options for talking to nova via the novaclient.
# These options are for an admin user in your keystone config.
# It proxy's the token received from the user to send to nova via this
  admin users creds,
# basically acting like the client via that proxy token.
nova_proxy_admin_user = admin
nova_proxy_admin_pass = ADMIN_PASS
nova_proxy_admin_tenant_name = service
taskmanager_manager = trove.taskmanager.manager.Manager
...
```

7、准备 trove 数据库：

```
$ mysql -u root -p
mysql> CREATE DATABASE trove;
mysql> GRANT ALL PRIVILEGES ON trove.* TO trove@'localhost' \
IDENTIFIED BY ' TROVE_DBPASS';
mysql> GRANT ALL PRIVILEGES ON trove.* TO trove@'%' \
IDENTIFIED BY ' TROVE_DBPASS';
```

8、准备数据库服务
A. 初始化数据库：

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

B. 创建数据存储。你需要为每种类型的单独的数据存储使用各种数据库，例如，MySQL、MongoDB、Cassandra。这个例子说明了如何创建一个 MySQL 数据库的数据存储：

```
# su -s /bin/sh -c "trove-manage datastore_update mysql '''" trove
```

9、创建一个 trove 镜像。
你要创建镜像的可使用的数据库类型。例如，MySQL、MongoDB 和 Cassandra。

此镜像必须安装 trove guest 代理，并且它必须配置连接到您的 OpenStack 环境 trove-guestagent.conf 文件中。请遵循这些步骤
您使用的是建立镜像间的实例：
•添加以下行到 trove-guestagent.conf：

```
rabbit_host = controller
rabbit_password = RABBIT_PASS
nova_proxy_admin_user = admin
nova_proxy_admin_pass = ADMIN_PASS
nova_proxy_admin_tenant_name = service
trove_auth_url = http://controller:35357/v2.0
```

10、更新使用新的镜像数据存储，使用 trove-manage 命令。
这个例子说明如何创建一个 MySQL5.5 的数据存储：

```
# trove-manage --config-file /etc/trove/trove.conf
  datastore_version_update \
  mysql mysql-5.5 mysql glance_image_ID mysql-server-5.5 1
```

11、您必须注册身份服务到数据库模块，这样其他的 OpenStack 服务才能够找到它。注册服务，并指定端点：

```
$ keystone service-create --name trove --type database \
  --description "OpenStack Database Service"
$ keystone endpoint-create \
  --service-id $(keystone service-list | awk '/ trove / {print $2}') \
  --publicurl http://controller:8779/v1.0/%\(tenant_id\)s \
  --internalurl http://controller:8779/v1.0/%\(tenant_id\)s \
```

```
  --adminurl http://controller:8779/v1.0/%\(tenant_id\)s  \
  --region regionOne
```

12、启动数据库服务，并将其配置为自启动

```
# service openstack-trove-api start
# service openstack-trove-taskmanager start
# service openstack-trove-conductor  start
# chkconfig openstack-trove-api on
# chkconfig openstack-trove-taskmanager on
# chkconfig openstack-trove-conductor  on
```

验证数据库服务的安装

要验证数据库服务已安装并配置正确，尝试执行 Trove 命令

1、来源 demo-openrc.sh 文件

```
$ source ~/demo-openrc.sh
```

2、检索实例 trove 列表

```
$ trove list
```

你应该看到的输出与此类似：

```
+----+------+-----------+-------------------+--------+-----------+------+
| id | name | datastore | datastore_version | status | flavor_id | size |
+----+------+-----------+-------------------+--------+-----------+------+
```

3、假设你已经创建了你想要的镜像数据库类型，并又更新了数据存储使用的镜像，你现在可以创建一个实例 Trove（数据库）。要做到这一点，使用 Trove 创建命令。

这个例子说明如何创建一个 MySQL 数据库

```
$ trove create name 2 --size=2 --databases DBNAME \
  --users USER: PASSWORD --datastore_version mysql-5.5 \
  --datastore mysql
```

十三 、添加数据处理服务

数据处理服务（sahara）

OpenStack 的（sahara）数据处理服务宗旨，是指为用户提供简单的，以提供数据处理（Hadoop、Spark）集群通过指定几个参数，如 Hadoop 的版本，集群拓扑，节点的硬件细节。后用户填写所有参数，该数据处理服务在按需工作节点上部署集群只需要几分钟。同时 sahara 提供添加/删除方法扩展已配置的集群。

安装数据处理服务（sahara）

此过程在控制器节点上安装数据处理服务（sahara）。

1、安装所需的软件包
yum install openstack-sahara python-saharaclient

2、编辑/etc/sahara/sahara.conf 配置文件
A．首先，在[database]部分编辑 connection 参数。这里提供的 URL 应该指向一个空数据库。例如，MySQL 数据库连接字符串将是：
connection = mysql://sahara: SAHARA_DBPASS@controller/sahara

B．切换到[keystone_authtoken]部分。该 auth_uri 参数应该指定公共 API 端点。identity_uri 应该指向管理员身份 API 端点。例如：
auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357

C．接下来指定 admin_user、admin_password 和 admin_tenant_name。这些参数必须指定在给定的 admin 角色中。这些凭据允许 sahara 验证授权的用户。

D．切换到[DEFAULT]部分。进入到网络参数，如果您使用 neutron 联网，设置 use_neutron=true。否则，如果您正在使用 nova-network，则设置参数为 false。

E．如果你想增加日志记录级别、故障排除，还要再配置两个参数：verbose 和 debug。如果前者设置为 true，sahara 将开始写 INFO 以上级别的日志。如果 debug 设置为 true，sahara 会写所有的日志，包括 DEBUG 的。

3、如果您使用数据处理服务，MySQL 数据库用于存储大数据。在 sahara 内部的数据库二进制文件，您必须配置允许的最大规模数据包。
编辑 my.cnf 文件和更改参数：
[mysqld]
max_allowed_packet = 256M

重新启动 MySQL 服务器。

4、创建数据库模式
sahara-db-manage --config-file /etc/sahara/sahara.conf upgrade head

5、您必须注册服务标识，使其他的 OpenStack 服务能够找到数据处理服务。如下，注册服务，并指定端点：
```
$ keystone service-create --name sahara --type data_processing \
  --description "Data processing service"
$ keystone endpoint-create \
  --service-id $(keystone service-list | awk '/ sahara / {print $2}') \
  --publicurl http://controller:8386/v1.1/%\(tenant_id\)s  \
  --internalurl http://controller:8386/v1.1/%\(tenant_id\)s  \
```

```
  --adminurl http://controller:8386/v1.1/%\(tenant_id\)s  \
  --region regionOne
```

6、启动 sahara 服务

```
# systemctl start openstack-sahara-all
```

7、（可选）使数据处理服务开机启动

```
# systemctl enable openstack-sahara-all
```

验证数据处理服务的安装

为了验证数据处理服务（sahara）已安装并正确配置，尝试利用 sahara 的客户群名单。

1、来源 demo 租户凭据

```
$ source demo-openrc.sh
```

2.检索 sahara 集群列表

```
$ sahara cluster-list
```

你应该看到如下输出

```
+------+----+--------+------------+
| name | id | status | node_count |
+------+----+--------+------------+
```

十四、启动一个实例

启动 OpenStack 的网络实例(neutron)

生成密钥对

云镜像支持公钥认证，而不是传统的用户名/密码验证。要启动一个实例，您必须生成一个 public/private 密钥对。使用 ssh-keygen 并添加公共密钥到你的 OpenStack 环境中。

1、来源 demo 租户凭据

```
$ source demo-openrc.sh
```

2、生成密钥对

```
$ ssh-keygen
```

3、添加公钥到您的 OpenStack 环境中

```
$ nova keypair-add --pub-key ~/.ssh/id_rsa.pub demo-key
```

注意
该命令不提供输出。

4、验证此公钥
$ nova keypair-list

```
+----------+-------------------------------------------------+
| Name | Fingerprint |
+----------+-------------------------------------------------+
| demo-key | 6c:74:ec:3a:08:05:4e:9e:21:22:a6:dd:b2:62:b8:28 |
+----------+-------------------------------------------------+
```

启动一个实例
要启动一个实例，则必须至少指定 flavor、镜像名称、网络，安全组、密钥和实例名称。

1、一个 flavor 指定一个虚拟资源分配信息，包括处理器、存储器和存储。
列出可用的 flavor：
$ nova flavor-list

```
+----+-----------+-----------+------+-----------+------+-------
+-------------+-----------+
| ID | Name | Memory_MB | Disk | Ephemeral | Swap | VCPUs |
 RXTX_Factor | Is_Public |
+----+-----------+-----------+------+-----------+------+-------
+-------------+-----------+
| 1 | m1.tiny | 512 | 1 | 0 | | 1 | 1.0
 | True |
| 2 | m1.small | 2048 | 20 | 0 | | 1 | 1.0
 | True |
| 3 | m1.medium | 4096 | 40 | 0 | | 2 | 1.0
 | True |
| 4 | m1.large | 8192 | 80 | 0 | | 4 | 1.0
 | True |
| 5 | m1.xlarge | 16384 | 160 | 0 | | 8 | 1.0
 | True |
+----+-----------+-----------+------+-----------+------+-------
```

你的第一个实例使用 m1.tiny 的 flavor。

注意
您也可以通过 ID 引用 flavor。

2、列出可用的镜像
$ nova image-list

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

```
| ID | Name | Status |
 Server |
+--------------------------------------+------------------+--------
+--------+
| acafc7c0-40aa-4026-9673-b879898e1fc2 | cirros-0.3.3-x86_64 | ACTIVE |
  |
+--------------------------------------+------------------+--------
```

你的第一个实例使用 cirros-0.3.3-x86_64 镜像。

3、可用网络列表

$ neutron net-list

```
+--------------------------------------+----------+-------------------+
| id | name | subnets
  |
+--------------------------------------+----------+-------------------+
| 3c612b5a-d1db-498a-babb-a4c50e344cb1 | demo-net |
20bcd3fd-5785-41feac42-55ff884e3180 192.168.1.0/24 |
| 9bce64a3-a963-4c05-bfcd-161f708042d1 | ext-net | b54a8d85-b434-4e85-
a8aa-74873841a90d 203.0.113.0/24 |
+--------------------------------------+----------+-------------------+
```

你的第一个实例使用 demo-net 租户网络。但是，您必须引用网络 ID 而不是使用名称。

4、列出可用的安全组

$ nova secgroup-list

```
+--------------------------------------+---------+-------------+
| Id | Name | Description |
+--------------------------------------+---------+-------------+
| ad8d4ea5-3cad-4f7d-b164-ada67ec59473 | default | default |
+--------------------------------------+---------+-------------+
```

你的第一个实例使用默认的安全组。默认情况下，此安全组实现了一个防火墙，阻止远程访问实例。如果你想允许远程访问您的实例，你可启动它，然后配置远程访问。

5、启动实例

更换 DEMO_NET_ID 与 demo-net 租户网络的 ID。

$ nova boot --flavor m1.tiny --image cirros-0.3.3-x86_64 --nic netid=DEMO_NET_ID
****
** --security-group default --key-name demo-key demo-instance1**

```
+--------------------------------------+-----------------------------------+
| Property | Value
```

```
  |
+----------------------------------
+-------------------------------------------------------+
| OS-DCF:diskConfig | MANUAL
  |
| OS-EXT-AZ:availability_zone | nova
  |
| OS-EXT-STS:power_state | 0
  |
| OS-EXT-STS:task_state | scheduling
  |
| OS-EXT-STS:vm_state | building
  |
| OS-SRV-USG:launched_at | -
  |
| OS-SRV-USG:terminated_at | -
  |
| accessIPv4 |
  |
| accessIPv6 |
  |
| adminPass | vFW7Bp8PQGNo
  |
| config_drive |
  |
| created | 2014-04-09T19:24:27Z
  |
| flavor | m1.tiny (1)
  |
| hostId |
  |
| id |
 05682b91-81a1-464c-8f40-8b3da7ee92c5 |
| image | cirros-0.3.3-x86_64
 (acafc7c0-40aa-4026-9673-b879898e1fc2) |
| key_name | demo-key
  |
| metadata | {}
  |
| name | demo-instance1
  |
| os-extended-volumes:volumes_attached | []
  |
| progress | 0
```

|
| security_groups | default
|
| status | BUILD
|
| tenant_id | 7cf50047f8df4824bc76c2fdf66d11ec
|
| updated | 2014-04-09T19:24:27Z
|
| user_id | 0e47686e72114d7182f7569d70c519c9

6、检查实例的状态

$ nova list

```
+--------------------------------------+----------------+-------
+------------+-------------+----------------------+
| ID | Name | Status | Task
 State | Power State | Networks |
+--------------------------------------+----------------+-------
+------------+-------------+----------------------+
| 05682b91-81a1-464c-8f40-8b3da7ee92c5 | demo-instance1 | ACTIVE | -
 | Running | demo-net=192.168.1.3 |
+--------------------------------------+----------------+-------
```

从 BUILD 状态变更为 ACTIVE 时，表示您的实例完成构建过程。

使用虚拟控制台访问您的实例

•获取虚拟网络计算（VNC）会话 URL，从 Web 浏览器访问您的实例

$ nova get-vnc-console demo-instance1 novnc

```
+-------------------------------------------------------------------------------
+
| Type | Url
 |
+-------------------------------------------------------------------------------
+
| novnc |
http://controller:6080/vnc_auto.html?token=2f6dd985-f906-4bfcb566-e87ce656375
b |
```

请注意

如果您 Web 浏览器中运行的主机无法解析控制器主机名，你可以用你的控制器节点上的管理接口 IP 地址替换控制器主机名。该 CirrOS 镜像包括传统的用户名/密码认证和提供这些凭证在登录提示。登录到 CirrOS 后，我们建议你使用 ping

来验证网络连接。

验证 demo-net 租户网关：

$ ping -c 4 192.168.1.1

```
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_req=1 ttl=64 time=0.357 ms
64 bytes from 192.168.1.1: icmp_req=2 ttl=64 time=0.473 ms
64 bytes from 192.168.1.1: icmp_req=3 ttl=64 time=0.504 ms
64 bytes from 192.168.1.1: icmp_req=4 ttl=64 time=0.470 ms
--- 192.168.1.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2998ms
rtt min/avg/max/mdev = 0.357/0.451/0.504/0.055 ms
```

验证 ext-net 外网：

$ ping -c 4 openstack.org

```
PING openstack.org (174.143.194.225) 56(84) bytes of data.
64 bytes from 174.143.194.225: icmp_req=1 ttl=53 time=17.4 ms
64 bytes from 174.143.194.225: icmp_req=2 ttl=53 time=17.5 ms
64 bytes from 174.143.194.225: icmp_req=3 ttl=53 time=17.7 ms
64 bytes from 174.143.194.225: icmp_req=4 ttl=53 time=17.5 ms
--- openstack.org ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 17.431/17.575/17.734/0.143 ms
```

远程访问您的实例

1、添加默认的安全组规则

A．许可 ICMP（ping）：

$ nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0

```
+-------------+-----------+---------+-----------+-------------+
| IP Protocol | From Port | To Port | IP Range | Source Group |
+-------------+-----------+---------+-----------+-------------+
| icmp | -1 | -1 | 0.0.0.0/0 | |
+-------------+-----------+---------+-----------+-------------+
```

B．允许 SSH 访问：

$ nova secgroup-add-rule default tcp 22 22 0.0.0.0/0

```
+-------------+-----------+---------+-----------+-------------+
| IP Protocol | From Port | To Port | IP Range | Source Group |
+-------------+-----------+---------+-----------+-------------+
| tcp | 22 | 22 | 0.0.0.0/0 | |
+-------------+-----------+---------+-----------+-------------+
```

2、创建 ext-net 外部网络上的浮动 IP 地址

$ neutron floatingip-create ext-net

Created a new floatingip:

```
+---------------------+--------------------------------------+
| Field | Value |
+---------------------+--------------------------------------+
| fixed_ip_address | |
| floating_ip_address | 203.0.113.102 |
| floating_network_id | 9bce64a3-a963-4c05-bfcd-161f708042d1 |
| id | 05e36754-e7f3-46bb-9eaa-3521623b3722 |
| port_id | |
| router_id | |
| status | DOWN |
| tenant_id | 7cf50047f8df4824bc76c2fdf66d11ec |
+---------------------+--------------------------------------+
```

3、关联您的实例浮动 IP 地址

$ nova floating-ip-associate demo-instance1 203.0.113.102

请注意

这个命令没有提供输出。

4、检查你的浮动 IP 地址状态

$ nova list

```
+--------------------------------------+----------------+--------+------------+-------------+-----------------------+
| ID | Name | Status | Task State | Power State | Networks |
+--------------------------------------+----------------+--------+------------+-------------+-----------------------+
| 05682b91-81a1-464c-8f40-8b3da7ee92c5 | demo-instance1 | ACTIVE | - | Running | demo-net=192.168.1.3, 203.0.113.102 |
+--------------------------------------+----------------+--------+------------+-------------+-----------------------+
```

5、从控制器节点或外部网络上的任何主机使用 ping 来验证网络连接

$ ping -c 4 203.0.113.102

```
PING 203.0.113.102 (203.0.113.112) 56(84) bytes of data.
64 bytes from 203.0.113.102: icmp_req=1 ttl=63 time=3.18 ms
64 bytes from 203.0.113.102: icmp_req=2 ttl=63 time=0.981 ms
64 bytes from 203.0.113.102: icmp_req=3 ttl=63 time=1.06 ms
64 bytes from 203.0.113.102: icmp_req=4 ttl=63 time=0.929 ms
--- 203.0.113.102 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 0.929/1.539/3.183/0.951 ms
```

6、从控制器节点或外部的任何主机上使用 SSH 访问您的实例网络

$ ssh cirros@203.0.113.102

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

请注意

如果你的主机不包含在前面步骤中创建的公钥/私钥对，SSH 将提示输入与 cirros 关联的默认用户/密码。

附加一个块存储卷到您的实例

如果您的环境包括块数据存储服务，您可以附加一个卷到实例。

1、执行 demo 租户凭据

```
$ source demo-openrc.sh
```

2、列出卷

$ nova volume-list

```
+--------------------------------------+-----------+--------------+-----
+-------------+-------------+
| ID | Status | Display Name | Size |
  Volume Type | Attached to |
+--------------------------------------+-----------+--------------+-----
+-------------+-------------+
| 158bea89-07db-4ac2-8115-66c0d6a4bb48 | available | demo-volume1 | 1 |
  None | |
+--------------------------------------+-----------+--------------+-----
```

3、将 demo-volume1 卷附加到 demo-instance1 实例

$ nova volume-attach demo-instance1 158bea89-07db-4ac2-8115-66c0d6a4bb48

```
+----------+--------------------------------------+
| Property | Value |
+----------+--------------------------------------+
| device | /dev/vdb |
| id | 158bea89-07db-4ac2-8115-66c0d6a4bb48 |
| serverId | 05682b91-81a1-464c-8f40-8b3da7ee92c5 |
| volumeId | 158bea89-07db-4ac2-8115-66c0d6a4bb48 |
+----------+--------------------------------------+
```

请注意

你必须参考使用卷 id，而不是名字。

4、列出卷

$ nova volume-list

```
+--------------------------------------+-----------+--------------+-----
+-------------+-----------------------------------+
| ID | Status | Display Name | Size |
 Volume Type | Attached to |
+--------------------------------------+-----------+--------------+-----
+-------------+-----------------------------------+
| 158bea89-07db-4ac2-8115-66c0d6a4bb48 | in-use | demo-volume1 | 1 |
 None | 05682b91-81a1-464c-8f40-8b3da7ee92c5 |
+--------------------------------------+-----------+--------------+-----
+-------------+-----------------------------------+
```

demo-volume1 卷状态应该标明好 demo-instance1 实例的 ID。

5、从控制器节点或任何外部主机上使用 SSH 访问您的实例网络，并使用 fdisk 命令来验证是否存在/dev/vdb 块存储设备

$ ssh cirros@203.0.113.102

$ sudo fdisk -l

```
Disk /dev/vda: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders, total 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
  Device Boot Start End Blocks Id System
/dev/vda1 * 16065 2088449 1036192+ 83 Linux
Disk /dev/vdb: 1073 MB, 1073741824 bytes
16 heads, 63 sectors/track, 2080 cylinders, total 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
Disk /dev/vdb doesn't contain a valid partition table
```

请注意

你必须创建一个分区表和文件系统的使用量。

如果您的实例不启动或看起来像您预期的那样工作，更多信息，请参见 OpenStack 操作指南或使用其他选项寻求帮助。

我们希望您的环境工作！

——完

请注意
本翻译文档，省略了传统网络（nova-network）部分。原因很简单——随着时代的过去，其终将成为历史。

本文档翻译自官网：
OpenStack Installation Guide for Red Hat Enterprise Linux 7, CentOS 7, and Fedora 20

后语
我为什么要翻译这篇文档呢？
原因在于，我希望能通过翻译，成为学习 OpenStack 的一种方法。

本文档翻译历时一个月，基本是用晚上下班时间。虽经吾竭力避免最基本的翻译、语法等错误，但可能仍存在类似错误。然而，请原谅我这个英语从未及格过的人吧。

如果你阅读该翻译文档时有什么疑问或想法，欢迎使用如下联系方式之一。
Author：小徐
Date：2014.12.3
QQ：1757794282
Email：xiaoxu790@126.com
My Website：http://chaoxu.sinaapp.com

discovered by you!