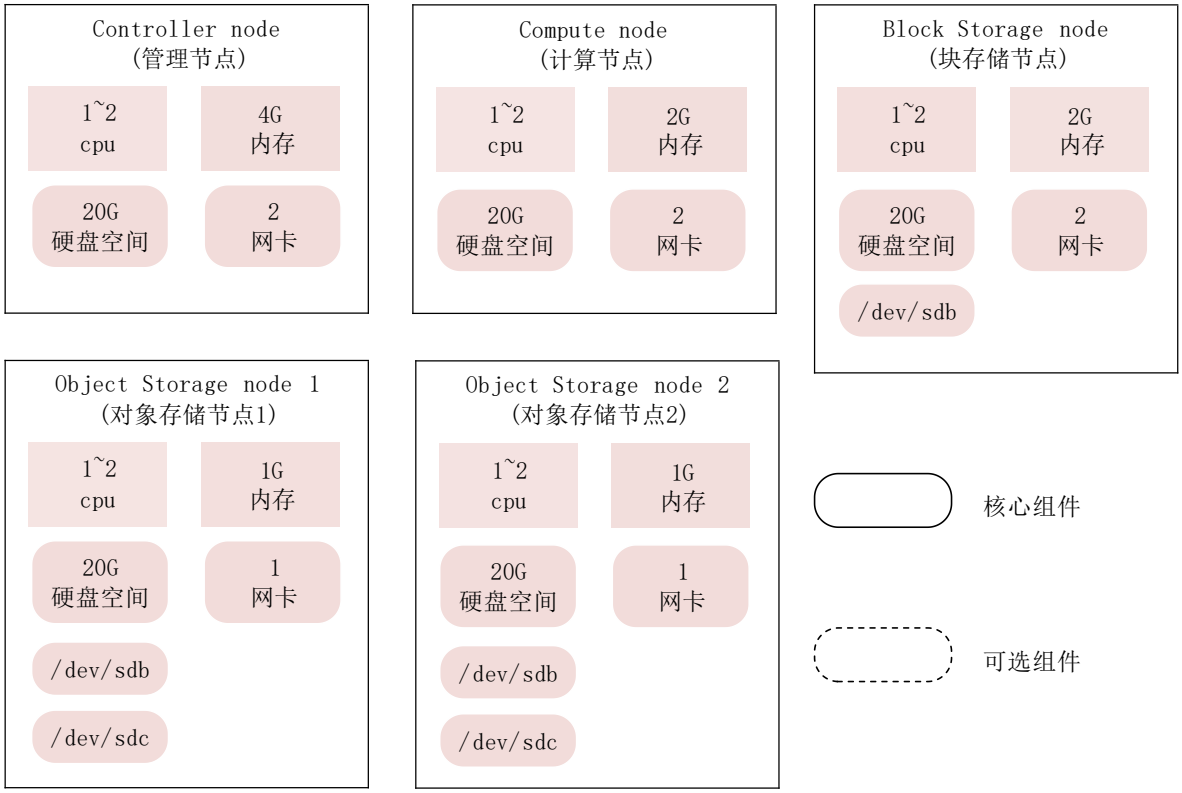
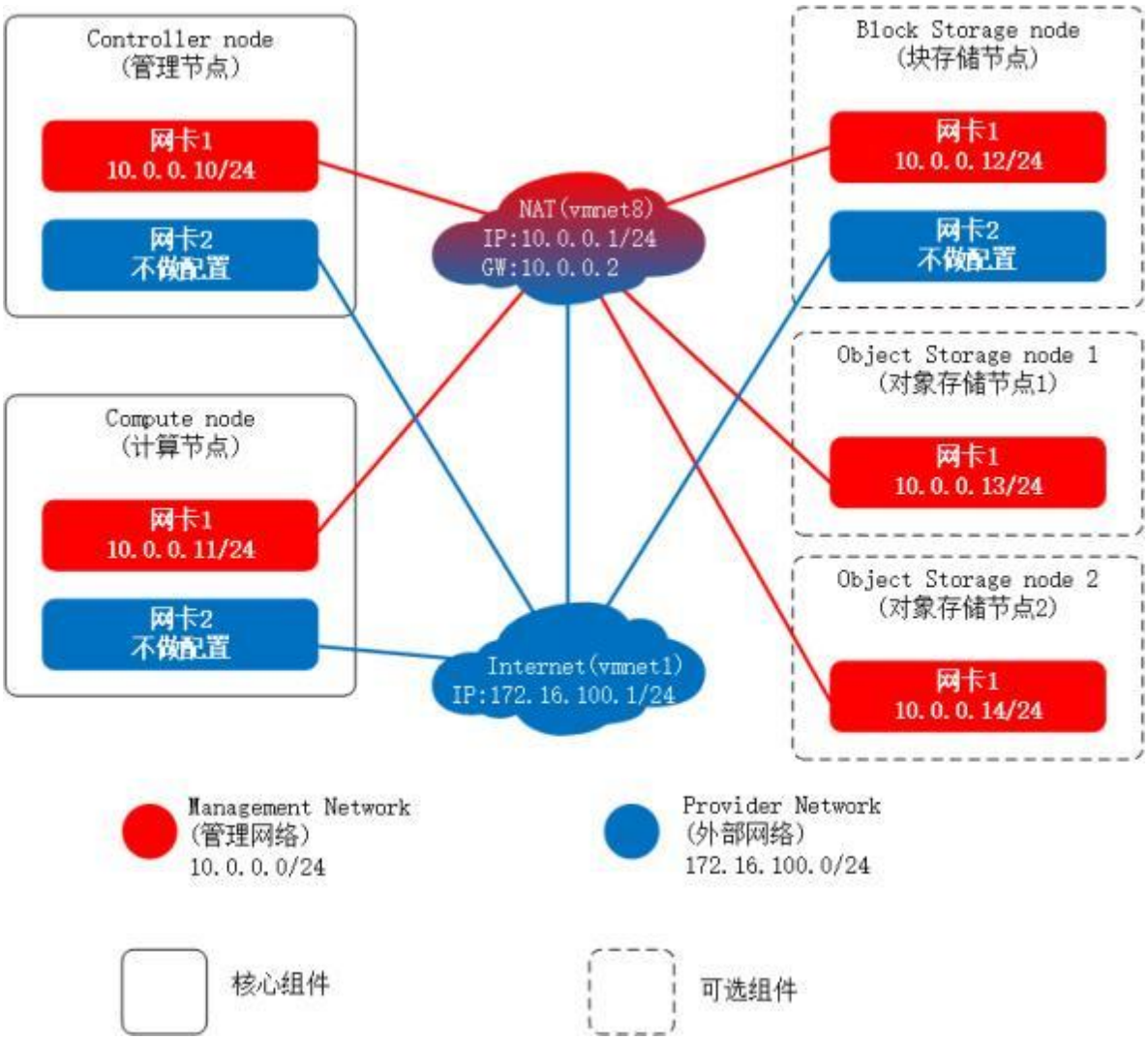
OpenStack(Mitaka) 安装指南

一、实验环境资源需求



二、实验环境拓扑



三、实验环境配置

1、安装Centos7.0

本指南要求Centos的版本一定是7.2.1511。

先安装一个虚拟机，安装时注意以下问题：

（1）CPU勾选虚拟化；

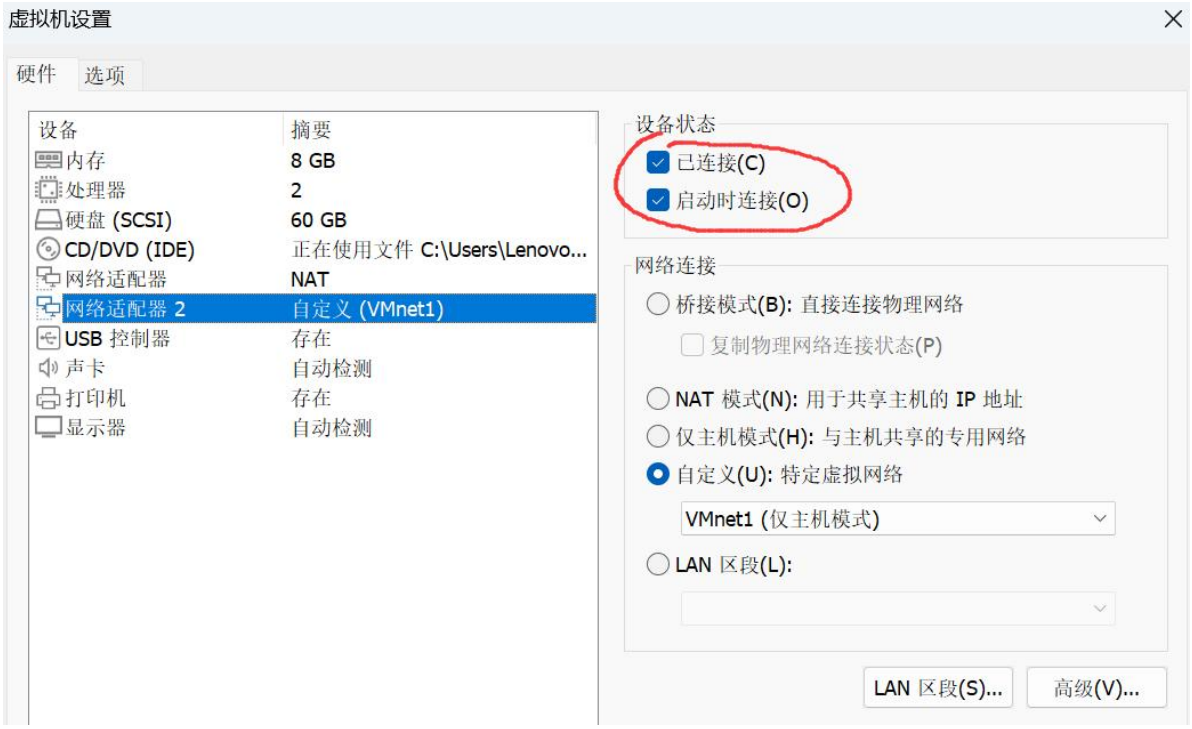
（2）添加两块网卡；

（3）安装时勾选服务器；

（4）安装操作系统时提示创建用户，要创建一个名字是openstack的用户；

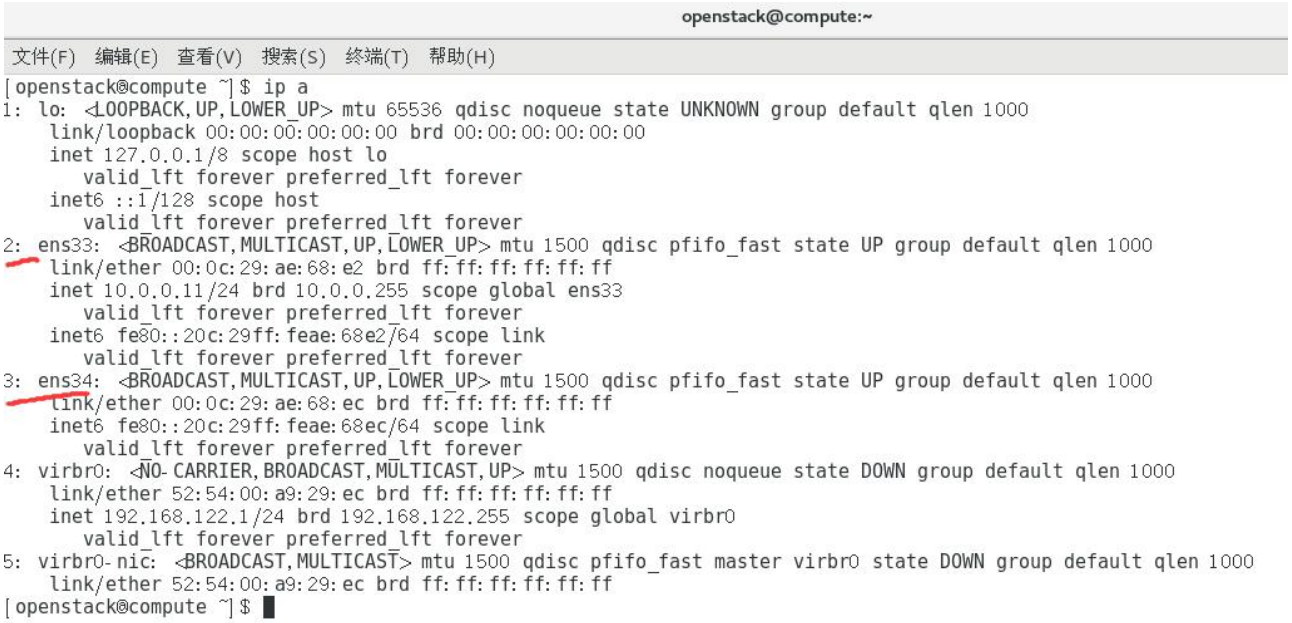
（5）采用克隆的方式生成其余虚拟机；

（6）克隆虚拟机打开前要编辑虚拟机配置修改网卡的MAC地址，并将所有虚拟机网卡的状态改为如下状态；



（7）启动虚拟机后查看虚拟机的名字，并修改两块网卡的配置文件，可参考下面的参数，但需注意网卡的IP不能冲突。

查看网卡



修改网卡ens33配置文件如下（仅供参考）

```
[openstack@compute ~]$ vi /etc/sysconfig/network-scripts/ifcfg-ens33
```

```
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens33
UUID=846c22d4-c791-4f18-ab60-9a3ee4238234
DEVICE=ens33
ONBOOT=yes
IPV6_PRIVACY=no
IPADDR=10.0.0.11
NETMASK=255.255.255.0
GATEWAY=10.0.0.2
DNS1=114.114.114.114
DNS2=8.8.8.8
~
```

修改网卡ens34配置文件如下（仅供参考）

```
[openstack@compute ~]$ vi /etc/sysconfig/network-scripts/ifcfg-ens34
```

```
文件(F)  编辑(E)  查看(V)  搜索(S)  终端(T)  帮
TYPE=Ethernet
BOOTPROTO="none"
NAME=ens34
DEVICE=ens34
ONBOOT=yes
~
~
~
```

（8）克隆出的虚拟机还要完成以下操作：清空/etc/udev/rules.d/70-persistent-net.rules文件；修改网卡配置文件中的uuid，只需改后面两组数据中的个别位即可；

（9）关闭NetworkManager服务。

可能遇到的问题：

（1）克隆的虚拟机解析不了域名，NetworkMananger的原因，启动该服务，生成一下/etc/resolve.conf文件，再彻底关掉该服务。

（2）第二块网卡的配置会影响到域名解析文件？etc/resolve.conf的自动生成，可以查资料禁止每次启动更新。

以下1~3操作在每台计算机上操作

1、如上图所示创建虚拟机，并修改主机名。例如：

```
# hostnamectl set-hostname controller
```

2、如上图所示配置红色网卡 IP 地址。蓝色网卡无需特殊配置，保持网卡激活状态即可。（改为修改ifcg-ens33文件）例如：

```
# nmcli connection add type ethernet con-name conn1 ifname ens33

# nmcli connection modify conn1 ipv4.method manual ipv4.addresses 10.0.0.10/24
ipv4.gateway 10.0.0.2

# nmcli connection up conn1
```

3、修改每个服务器的/etc/hosts 文件， 添加内容如下：

```
10.0.0.10   controller

10.0.0.11   compute

10.0.0.12   block

10.0.0.13   object1

10.0.0.14   object2
```

4、配置 controller node(管理节点)为时间服务器，并设置其他节点从管理节点同步时间。

```
# yum install chrony
```

1) 修改 controller node(管理节点)的/etc/chrony.conf 配置文件。添加如下内容：

```
allow 10.0.0.0/24

bindcmdaddress 0.0.0.0

local stratum 10
```

启动 chronyd 服务，并设置开机自动启动。

```
# systemctl enable chronyd.service

# systemctl start chronyd.service
```

2) 修改其他节点的/etc/chrony.conf 配置文件。添加如下内容：

```
server controller iburst
```

(删除或注释其他 server 项)

启动 chronyd 服务，并设置开机自动启动。

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

3）检验

```
# chronyc sources
```

Controller的测试结果

```
[root@controller openstack]# chronyc sources
210 Number of sources = 4
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^? 110.42.98.138                 3   8   100   662  -3524us[-5077us] +/-   67ms
^* ntp.jxust.edu.cn              2   6   377    22   +878us[+1073us] +/-   49ms
^- 81.16.177.123                 2   6   337    17    +39ms[   +39ms] +/-  234ms
^? ntp.wdc2.us.leaseweb.net      0   7     0     -     +0ns[    +0ns] +/-    0ns
```

其它节点的测试结果

```
[root@block openstack]# chronyc sources
210 Number of sources = 1
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* controller                    3   6    17    33    -39us[  -209us] +/-   50ms
```

5、关闭所有节点的防火墙服务

```
# systemctl disable firewalld
# systemctl stop firewalld
```

6、配置所有节点的 yum 源

1）如果可以上网，则删除或移走所有节点的/etc/yum.repo.d 目录下的所有文件，并创建新文件 a.repo，使其包含如下内容。

```
[openstack-mitaka]
name=OpenStack Mitaka Repository
baseurl=https://mirrors.tuna.tsinghua.edu.cn/centos-vault/7.2.1511/cloud/x86_64/openstack-mitaka/
gpgcheck=0
enabled=1

[base]
name=base
baseurl=https://mirrors.tuna.tsinghua.edu.cn/centos-vault/7.2.1511/os/x86_64/
enabled=1
gpgcheck=0

[extras]
```

```
name=extras
baseurl=https://mirrors.tuna.tsinghua.edu.cn/centos-vault/7.2.1511/extras/x86_64/
enabled=1
gpgcheck=0

[updates]
name=updates
baseurl=https://mirrors.tuna.tsinghua.edu.cn/centos-vault/7.2.1511/updates/x86_64/
enabled=1
gpgcheck=0
[Virt]
name=CentOS-$releasever - Base
baseurl=https://mirrors.tuna.tsinghua.edu.cn/centos-vault/7.2.1511/virt/x86_64/kvm-common/
gpgcheck=0
enabled=1
```

2) 如果不能上网，则在 Controller node(管理节点)添加包含 yum 仓库的磁盘文件，并在管理节点进行如下操作。

A. 将系统安装 DVD 镜像光盘挂在到/media 目录。

```
# mount /dev/cdrom /media/
```

B. 安装 vsftpd 服务，并启动服务。

```
# rpm -ivh /media/Packages/vsftpd-3.0.2-10.el7.x86_64.rpm
# systemctl start vsftpd.service
# systemctl enable vsftpd.service
```

C. 创建 yum 仓库磁盘挂载点，并挂载 yum 仓库磁盘。

```
# mkdir /var/ftp/yum
# echo "/dev/sdb1 /var/ftp/yum/ xfs defaults 0 0" >> /etc/fstab
# mount -a
```

D. 恢复 SELinux 上下文。

```
# restorecon -Rv /var/ftp/
```

3) 删除或移走所有节点/etc/yum.repos.d 目录中的文件，并在该目录中创建新文件 b.repo,包含如下内容：

```
[openstack-mitaka]
name=OpenStack Mitaka Repository
baseurl=ftp://controller/yum/openstack-mitaka/
gpgcheck=0
enabled=1

[base]
name=base
```

```
baseurl=ftp://controller/yum/base
enabled=1
gpgcheck=0

[extras]
name=extras
baseurl=ftp://controller/yum/extras/
enabled=1
gpgcheck=0

[updates]
name=updates
baseurl=ftp://controller/yum/updates/
enabled=1
gpgcheck=0
```

7、在所有节点安装软件包。

1) 更新所有软件包，如果更新了内核，请重启系统后再继续其他操作。

```
# yum clean all
# yum makecache

# yum upgrade  -y
```

2) 安装 OpenStack 客户端

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

3) 由于 CentOS 或 RHEL 的 SELinux 默认是打开的，因此需要安装 openstack-selinux 包来自动管理跟 openstack 服务有关的安全策略。

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

8、大多数 OpenStack 服务使用 SQL 数据库存储信息。数据库一般运行在 Controller node(管理节点)。在管理节点安装并配置 MariaDB 数据库组件。

1) 安装软件包。

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

2) 创建并编辑文件/etc/my.cnf.d/openstack.cnf。

A. 在[mysqld]小节， 设置 bind-address 配置项为管理节点的管理 IP 地址。

```
[mysqld]
...
bind-address = 10.0.0.10
```

B. 在[mysqld]小节， 添加下列有用的配置项，以及支持 UTF-8 字符集。

```
[mysqld]
...
default-storage-engine = innodb
innodb_file_per_table
max_connections = 4096
```

```
collation-server = utf8_general_ci
character-set-server = utf8
```

3) 完成安装

A. 启动数据库服务并设置开机自动启动。

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

B. 执行 mysql_secure_installation，设置数据库管理员 root 用户的密码。

```
# mysql_secure_installation
```

9、Telemetry 服务使用 NoSQL 数据库存储数据。该数据库一般运行在 Controller node(管理节点)上。在管理节点上安装并配置 MongoDB。

1) 安装 MongoDB 软件包

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

2) 编辑/etc/mongod.conf 文件

A. 配置 bind_ip 配置项的值为管理节点的管理 IP 地址。

```
bind_ip = 10.0.0.10
```

B. 默认情况下 MongoDB 创建 1G 大小的日志文件，并存放于/var/lib/mongodb/journal 目录中。如果你想缩小日志文件的大小到 128M 并限制总日志空间为 512M，则需要设置 smallfiles 配置项。

```
smallfiles = true
```

3) 完成安装

启动 MongoDB 服务并设置开机自动启动。

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

10、OpenStack 使用消息队列服务进行服务之间的协调和状态信息的同步。消息队列服务通常运行于 Controller node（管理节点）。OpenStack 支持多种消息队列服务，包括 RabbitMQ，Qpid 和 ZeroMQ。大多数 OpenStack 发行版支持 RabbitMQ。在管理节点安装 RabbitMQ 消息队列服务。

1) 安装软件包

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

2) 启动消息队列服务并设置开机自动启动。

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

3) 添加 openstack 用户

```
rabbitmqctl add_user openstack RABBIT_PASS
```

替换 *RABBIT_PASS*为一个合适的密码。

4) 为 openstack 用户赋予读和写访问权限。

```
rabbitmqctl set_permissions openstack ".*" ".*" ".*"
```

11、identity 服务身份认证机制使用 Memcached 缓存令牌。Memcached 服务通常运行于 Controller node(管理节点)。在管理节点安装 Memcached 服务。

1) 安装软件包。

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

2) 启动 Memcached 服务并设置开机自动启动。

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

四、安装和配置 Identity service(身份服务)

本章节介绍在 Controller node(管理节点)安装和配置身份服务。

一 先决条件

在安装配置 OpenStack 身份服务前，你必须创建一个数据库和管理员令牌。

1. 创建数据库，并完成下列操作

A. 使用数据库命令行客户端，以 root 身份登录数据库服务器。

```
# mysql -u root -p
```

B. 创建 keystone 数据库

```
CREATE DATABASE keystone;
```

C. 授予数据库用户 keystone 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';
```

替换 *KEYSTONE_DBPASS*为一个合适的密码。

2. 生成一个随机值，作为管理员的初始化令牌。

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

安装并配置组件

1. 安装软件包

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

2. 编辑/etc/keystone/keystone.conf 文件并完成下列操作：

   A. 在[default]小节， 定义管理员初始化令牌。

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

   替换 *ADMIN_TOKEN*为刚才产生的随机值。

   B. 在[database]小节，配置数据库访问：

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

   替换 *KEYSTONE_DBPASS*为合适的密码。

   C. 在[token]小节，配置使用 Fernet 技术提供令牌。

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

3. 初始化身份服务数据库：

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

4. 初始化 Fernet keys：

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

一 配置 Apache HTTP 服务

1. 编辑/etc/httpd/conf/httpd.conf 文件并配置 ServerName 配置项为管理节点的主机名：

```
ServerName controller
```

2. 创建/etc/httpd/conf.d/wsgi-keystone.conf 文件，包含下列内容：

```
Listen 5000
Listen 35357

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

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

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

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

一 完成安装

启动 Apache HTTP 服务并设置开机自动启动：

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

创建临时管理员令牌环境

1. 配置管理员令牌

```
# export OS_TOKEN=ADMIN_TOKEN
```

替换 *ADMIN_TOKEN*为刚才生成的随机值。例如：

```
# export OS_TOKEN=42d8da412a15d4f23587
```

2. 配置端点 URL：

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

3. 配置身份服务 API 版本：

```
# export OS_IDENTITY_API_VERSION=3
```

二 创建服务实体和 API 端点

身份服务提供一个目录服务和他们的位置。每一个向 OpenStack 环境添加的服务都需要一个服务实体和一些 API 端点，并将这些信息保存在目录中。

身份服务在你的 OpenStack 环境中管理一个包含所有服务的目录。服务们使用目录定位你环境中的其他服务是否可用。

1. 创建身份服务的服务实体：

```
# openstack service create --name keystone --description
"OpenStack Identity" identity
```

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Identity               |
| enabled     | True                             |
| id          | c8958889093c4be499bbde7bb120eb38 |
| name        | keystone                         |
| type        | identity                         |
+-------------+----------------------------------+
```

在你的 OpenStack 环境中，身份服务管理一个与服务相关联 API 端点的目录。服务们使用此目录来确定如何与你的环境中的其他服务通信。

2. 创建身份服务 API 端点：

```
# openstack endpoint create --region RegionOne identity public http://controller:5000/v3
```

```
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 969abe2d75984d0b8f4973207aca5aec |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | c8958889093c4be499bbde7bb120eb38 |
| service_name | keystone                         |
| service_type | identity                         |
| url          | http://controller:5000/v3        |
+--------------+----------------------------------+
```

```
# openstack endpoint create --region RegionOne identity internal http://controller:5000/v3
```

```
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
```

```
| id           | 2b71774910a047c592c6d560815bcee1 |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | c8958889093c4be499bbde7bb120eb38 |
| service_name | keystone                         |
| service_type | identity                         |
| url          | http://controller:5000/v3        |
+--------------+----------------------------------+
```

```
# openstack endpoint create --region RegionOne identity admin http://controller:35357/v3
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | c291916ea52d4a3793d0b0fa8dd66e30 |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | c8958889093c4be499bbde7bb120eb38 |
| service_name | keystone                         |
| service_type | identity                         |
| url          | http://controller:35357/v3       |
+--------------+----------------------------------+
```

一 创建域，项目，用户和角色

身份服务为每一个 OpenStack 服务提供认证服务。认证服务使用一个 domain(域),projects(项目(tenants(租户))), users(用户)和 roles(角色)的组合。

1. 创建 default 域:

```
# openstack domain create --description "Default Domain" default
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Default Domain                   |
```

```
| enabled     | True                             |
| id          | 5dfa4c3356474e97b7782eba0f46d710 |
| name        | default                          |
+-------------+----------------------------------+
```

2. 创建一个管理项目、用户和角色，用于在你的系统中行使管理操作：

A. 创建 admin 项目：

```
# openstack project create --domain default --description "Admin
Project" admin
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Admin Project                    |
| domain_id   | 5dfa4c3356474e97b7782eba0f46d710 |
| enabled     | True                             |
| id          | 782fda82df3449e7893f82c942efe1f8 |
| is_domain   | False                            |
| name        | admin                            |
| parent_id   | 5dfa4c3356474e97b7782eba0f46d710 |
+-------------+----------------------------------+
```

B. 创建 admin 用户：

```
# openstack user create --domain default --password-prompt admin
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | 5dfa4c3356474e97b7782eba0f46d710 |
| enabled   | True                             |
| id        | bff8caa7781245d9a6adcb9d14b3d70d |
| name      | admin                            |
+-----------+----------------------------------+
```

C. 创建 admin 角色：

```
# openstack role create admin
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
```

```
| domain_id | None                             |
| id        | 045723de2c83497481b1fcc5021b5b20 |
| name      | admin                            |
+-----------+----------------------------------+
```

D. 添加 admin 角色到 admin 项目和用户：

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

3. 本手册使用一个 service 项目，用来包含你环境中每一个服务的唯一用户。创建 service 项目：

```
# openstack project create --domain default --description "Service Project" service
```

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Service Project                  |
| domain_id   | 5dfa4c3356474e97b7782eba0f46d710 |
| enabled     | True                             |
| id          | 44bf5cda8ee54fd494a8a34fd7b34de9 |
| is_domain   | False                            |
| name        | service                          |
| parent_id   | 5dfa4c3356474e97b7782eba0f46d710 |
+-------------+----------------------------------+
```

4. 日常任务一般使用一个非特权项目和用户。在本手册中，创建 demo 项目和用户：

A. 创建 demo 项目：

```
# openstack project create --domain default --description "Demo Project" demo
```

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Demo Project                     |
| domain_id   | 5dfa4c3356474e97b7782eba0f46d710 |
| enabled     | True                             |
| id          | 8695acb7e10e4c44a25da3076b2671b4 |
```

```
| is_domain   | False                            |
| name        | demo                             |
| parent_id   | 5dfa4c3356474e97b7782eba0f46d710 |
+-------------+----------------------------------+
```

B. 创建 demo 用户：

```
# openstack user create --domain default --password-prompt demo
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | 5dfa4c3356474e97b7782eba0f46d710 |
| enabled   | True                             |
| id        | ac3905c764824555aa187363b3c6f7bb |
| name      | demo                             |
+-----------+----------------------------------+
```

C. 创建 user 角色：

```
# openstack role create user
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | 64f9ebb38e454d55bae8cb812f29b4b2 |
| name      | user                             |
+-----------+----------------------------------+
```

D. 添加 user 角色到 demo 项目和用户：

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

一 验证操作

在安装其他服务前，验证身份服务是否正常。

1. 由于安全的原因，关闭临时认证令牌机制。

   编辑/etc/keystone/keystone-paste.ini 文件并移除 [pipeline:public_api], [pipeline:admin_api], 和 [pipeline:api_v3]小节的 admin_token_auth 项。

2. 删除临时环境变量 OS_TOKEN 和 OS_URL：

```
# unset OS_TOKEN OS_URL
```

3. 使用 admin 用户，请求认证令牌：

```
# openstack --os-auth-url http://controller:35357/v3 --os-project-domain-name default --os-user-domain-name default --os-project-name admin --os-username admin token issue
Password:
+------------+-----------------------------------------------------------------+
| Field      | Value                                                           |
+------------+-----------------------------------------------------------------+
| expires    | 2016-07-19T18:17:19.491392Z                                     |
| id         | gAAAAABXjmCfunvxAKL8VIwCtSaD-                                   |
|            | 79938wIpTh6HL8yKeNYb7x87Iymcvc4v48boWwFQeRNO-                   |
|            | 5NFXu8JfLVrAnaUUlNE9QHEoh7QKGWd5zy3QPKnwq1ikCG_-                |
|            | VfBv2sIVWkNTqKxOrkDqTjTYTqVBOEswvQ631dzq_pzcfyB_VdCrfHAenf_aA   |
| project_id | 782fda82df3449e7893f82c942efe1f8                                |
| user_id    | bff8caa7781245d9a6adcb9d14b3d70d                                |
+------------+-----------------------------------------------------------------+
```

4. 使用 demo 用户，请求认证令牌：

```
# openstack --os-auth-url http://controller:5000/v3 --os-project-domain-name default --os-user-domain-name default --os-project-name demo --os-username demo token issue
Password:
+------------+-----------------------------------------------------------------+
| Field      | Value                                                           |
+------------+-----------------------------------------------------------------+
| expires    | 2016-07-19T18:17:06.152134Z                                     |
| id         | gAAAAABXjmCSLfjx_R3hO-zL4fWLtHQRn4PTgYHrxchfVecmVh4MLWnSoCF_bOh-  |
|            | rZnid7AXwEkUsdHBP9OuBaNPaY6qtU_l5qEv-iPRXGBImA4UGO5jsAXEz4pI6HpG1PNgyN |
|            | QcZpgh57OcEiHpiuxGgq1e60sBRPzWgaHf1mgui5Fn42c_iKI                |
| project_id | 8695acb7e10e4c44a25da3076b2671b4                                |
| user_id    | ac3905c764824555aa187363b3c6f7bb                                |
+------------+-----------------------------------------------------------------+
```

一 创建脚本

为 admin 和 demo 项目和用户创建客户端环境脚本。本手册后续部分将使用这些脚本加载用户凭据。

1. 编辑 admin-poenrc 文件，并添加下列内容：

```
export OS_PROJECT_DOMAIN_NAME=default
export OS_USER_DOMAIN_NAME=default

export OS_PROJECT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS

export OS_AUTH_URL=http://controller:35357/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

替换 *ADMIN_PASS*为身份服务中 admin 用户的密码。

2. 编辑 demo-poenrc 文件，并添加下列内容：

```
export OS_PROJECT_DOMAIN_NAME=default
export OS_USER_DOMAIN_NAME=default

export OS_PROJECT_NAME=demo
export OS_USERNAME=demo
export OS_PASSWORD=DEMO_PASS

export OS_AUTH_URL=http://controller:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

替换 *DEMO_PASS*为身份服务中 demo 用户的密码。

一 使用脚本

1. 加载 admin-openrc 文件用来填充身份服务中 admin 项目和用户的用户凭据到环境变量：

```
# . admin-openrc
```

2. 请求认证令牌：

```
# openstack token issue
```

```
+------------+-----------------------------------------------------------------+
| Field      | Value                                                           |
+------------+-----------------------------------------------------------------+
| expires    | 2016-07-19T18:33:01.690214Z                                     |
| id         | gAAAAABXjmRNOdWsAK-                                             |
|            | VHVmmC_cv4hANIbWTbLOBcgw20SnM6Zzi9HevJPzNcLrf4rIZHoBRlh2L1eWOkWUWfE- |
|            | sOsZOAA-PVTJzpSO2xivaXRw8AMS_1i008-SOrk8g6lwSCZcGbHn4JbO8       |
|            | -ABOgE68mKoXDzqDZEu31mUCN-KF1Y8o8NP3UCk                         |
```

```
| project_id | 782fda82df3449e7893f82c942efe1f8                                 |
| user_id    | bff8caa7781245d9a6adcb9d14b3d70d                                 |
+------------+------------------------------------------------------------------+
```

五、安装和配置 Image service(镜像服务)

本章介绍在 Controller node(管理节点)安装和配置镜像服务，代号(glance)。出于简单的目的， 本次使用本地文件系统存储镜像。

一 先决条件

1. 创建数据库，完成下列步骤：

   A. 使用数据库命令行客户端，以 root 身份登录数据库服务器。

   ```
   # mysql -u root -p
   ```

   B. 创建 glance 数据库

   ```
   CREATE DATABASE glance;
   ```

   C. 授予数据库用户 glance 访问 glance 数据库的权限。

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

   替换 *GLANCE_DBPASS*为一个合适的密码。

   D. 退出数据库

2. 执行 admin 凭据脚本，以便以 admin 身份执行后续命令：

   ```
   # . admin-openrc
   ```

3. 创建服务凭据，完成下列操作：

   A. 创建 glance 用户

```
# openstack user create --domain default --password-prompt glance
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | 5dfa4c3356474e97b7782eba0f46d710 |
| enabled   | True                             |
| id        | 27c8dd63af4b4e4d937e0ad7aaf7fb4e |
| name      | glance                           |
+-----------+----------------------------------+
```

B. 添加 admin 角色到 glance 用户和 service 项目

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

C. 创建 glance 服务实体：

```
# openstack service create --name glance --description "OpenStack Image"
image
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Image                  |
| enabled     | True                             |
| id          | 6d30b3ddfeee41cdbcc7e476302ce377 |
| name        | glance                           |
| type        | image                            |
+-------------+----------------------------------+
```

4. 创建镜像服务 API 端点：

```
# openstack endpoint create --region RegionOne image public
http://controller:9292
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 39612eb4fe26445d8459167decfffcd9 |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 6d30b3ddfeee41cdbcc7e476302ce377 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
```

```
+--------------+----------------------------------+
# openstack endpoint create --region RegionOne image internal http://controller:9292
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 8bccf423496f4df3abb21f28c15ac5c1 |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 6d30b3ddfeee41cdbcc7e476302ce377 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+
# openstack endpoint create --region RegionOne image admin http://controller:9292
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | bffd4d92f66a4f629dac68797ad9abd6 |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 6d30b3ddfeee41cdbcc7e476302ce377 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+
```

一 安装和配置组件

1. 安装软件包

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

2. 编辑/etc/glance/glance-api.conf 文件并完成下列操作：

A. 在[database]小节，配置数据库访问：

```
[database]
```

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

替换 *GLANCE_DBPASS*为你的镜像服务数据库用户 glance 的密码。

B. 在[keystone_authtoken]和[paste_deploy]小节配置身份服务访问信息：

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

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

替换 *GLANCE_DBPASS*为认证服务中 glance 用户的密码。

C. 在[glance_store]小节，配置使用本地系统存储和镜像文件存储路径：

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

3. 编辑/etc/glance/glance-registry.conf 文件并完成下列操作：

A. 在[database]小节，配置数据库访问：

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

替换 *GLANCE_DBPASS*为你的镜像服务数据库用户 glance 的密码。

B. 在[keystone_authtoken]和[paste_deploy]小节，配置身份服

务访问信息：

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

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

替换 *GLANCE_DBPASS*为认证服务中 glance 用户的密码。

4. 初始化镜像服务数据库

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

完成安装

启动镜像服务并设置开机自动启动：

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

确认安装

使用 CirrOS 镜像确认镜像服务是否安装正常。CirrOS 是一个小型 Linux 镜像，可以用来测试你的 OpenStack 环境。

1. 执行 admin 凭据脚本，以便以 admin 身份执行后续命令：

```
# . admin-openrc
```

2. 下载镜像文件

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

3. 上传镜像文件到镜像服务，使用 QCOW2 磁盘格式，bare 容器格式，添加公共可见选项，是所有项目可以访问该镜像：

```
# openstack image create "cirros" --file cirros-0.3.4-x86_64-disk.img --disk-format qcow2 --container-format bare --public
+------------------+------------------------------------------------------+
| Field            | Value                                                |
+------------------+------------------------------------------------------+
| checksum         | ee1eca47dc88f4879d8a229cc70a07c6                     |
| container_format | bare                                                 |
| created_at       | 2016-07-21T03:51:22Z                                 |
| disk_format      | qcow2                                                |
| file             | /v2/images/25c308d5-9056-4b6d-ae7c-5e83dde5be39/file |
| id               | 25c308d5-9056-4b6d-ae7c-5e83dde5be39                 |
| min_disk         | 0                                                    |
| min_ram          | 0                                                    |
| name             | cirros                                               |
| owner            | 782fda82df3449e7893f82c942efe1f8                     |
| protected        | False                                                |
| schema           | /v2/schemas/image                                    |
| size             | 13287936                                             |
| status           | active                                               |
| tags             |                                                      |
| updated_at       | 2016-07-21T03:51:22Z                                 |
| virtual_size     | None                                                 |
| visibility       | public                                               |
+------------------+------------------------------------------------------+
```

4. 确认镜像已经上传并验证属性：

```
# openstack image list
+--------------------------------------+--------+--------+
| ID                                   | Name   | Status |
+--------------------------------------+--------+--------+
| 25c308d5-9056-4b6d-ae7c-5e83dde5be39 | cirros | active |
+--------------------------------------+--------+--------+
```

六、安装和配置 Compute Service(计算服务)

使用 OpenStack 计算服务托管和管理云计算系统。OpenStack 计算服务是基础架构即服务(IaaS)系统的重要组成部分。

✧ 安装并配置管理节点

本章介绍在 Controller node(管理节点)安装和配置计算服务，代号 nova。

— 先决条件

在安装和配置计算服务前，必须创建数据库，服务凭据和 API 端点。

1. 创建数据库并完成下列步骤:

   A. 使用数据库命令行客户端，以 root 身份登录数据库服务器。

   ```
   # mysql -u root -p
   ```

   B. 创建 nova_api 和 nova 数据库:

   ```
   CREATE DATABASE nova_api;
   ```

   ```
   CREATE DATABASE nova;
   ```

   C. 创建数据库用户 nova，并授予数据库用户 nova 访问 nova_api 和 nova 数据库的权限。

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

   替换 *NOVA_DBPASS*为一个合适的密码。

D. 退出数据库

2. 执行 admin 凭据脚本，以便以 admin 身份执行后续命令:

```
# . admin-openrc
```

3. 创建服务凭据，并完成下列步骤:

A. 创建 nova 用户

```
# openstack user create --domain default --password-prompt nova
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | 5dfa4c3356474e97b7782eba0f46d710 |
| enabled   | True                             |
| id        | 8c929b0659e643178c6e0cee9374f829 |
| name      | nova                             |
+-----------+----------------------------------+
```

B. 添加 admin 角色到 nova 用户和 service 项目

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

C. 创建 nova 服务实体:

```
# openstack service create --name nova --description
"OpenStack Compute" compute
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Compute                |
| enabled     | True                             |
| id          | 87a337ab88794172b80b9b029e0b19c7 |
| name        | nova                             |
| type        | compute                          |
+-------------+----------------------------------+
```

D. 创建计算服务的 API 端点:

```
# openstack endpoint create --region RegionOne compute
```

```
public http://controller:8774/v2.1/%\(tenant_id\)s
```

```
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 9ec2813c6eed4db5ba5cd16ec89f0bc1        |
| interface    | public                                  |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | 87a337ab88794172b80b9b029e0b19c7        |
| service_name | nova                                    |
| service_type | compute                                 |
| url          | http://controller:8774/v2.1/%(tenant_id)s |
+--------------+-----------------------------------------+
```

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

```
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 471f6880773c49d1a79703040565f6d4        |
| interface    | internal                                |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | 87a337ab88794172b80b9b029e0b19c7        |
| service_name | nova                                    |
| service_type | compute                                 |
| url          | http://controller:8774/v2.1/%(tenant_id)s |
+--------------+-----------------------------------------+
```

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

```
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 4d85d883deb84903ad2a3e3efcb2ca26        |
| interface    | admin                                   |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
```

```
| service_id    | 87a337ab88794172b80b9b029e0b19c7                |
| service_name  | nova                                            |
| service_type  | compute                                         |
| url           | http://controller:8774/v2.1/%(tenant_id)s       |
+---------------+-------------------------------------------------+
```

安装配置组件

1. 安装软件包

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

2. 编辑/etc/nova/nova.conf 文件并完成下列操作:

A. 在[DEFAULT]小节，只启用 compute 和 metadata 的 API。

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

B. 在[api_database]和[database]小节，配置数据库访问:

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

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

替换 *NOVA_DBPASS*为你的计算服务数据库用户 nova 的密码。

C. 在[DEFAULT]在[oslo_messaging_rabbit]小节配置 RabbitMQ 消息队列访问:

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

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

```
rabbit_password = RABBIT_PASS
```

替换 *RABBIT_PASS*为 RabbitMQ 用户 openstack 的密码。

D. 在[DEFAULT]和[keystone_authtoken]小节配置身份服务访问信息：

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

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

替换 *NOVA_PASS*为身份服务中用户 nova 的密码。

E. 在[DEFAULT]小节， 配置 my_ip 配置项为管理节点的管理接口 IP 地址。

```
[DEFAULT]
...
my_ip = 10.0.0.10
```

F. 在[DEFAULT]小节， 启用支持 neutron 网络服务：

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

G. 在[vnc]小节，配置 vnc 代理，使用管理节点的管理接口 IP 地

址：

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

H. 在[glance]小节， 配置镜像服务 API 的位置：

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

I. 在[oslo_concurrency]小节，配置锁路径：

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

3. 初始化计算服务数据库：

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

一 完成安装

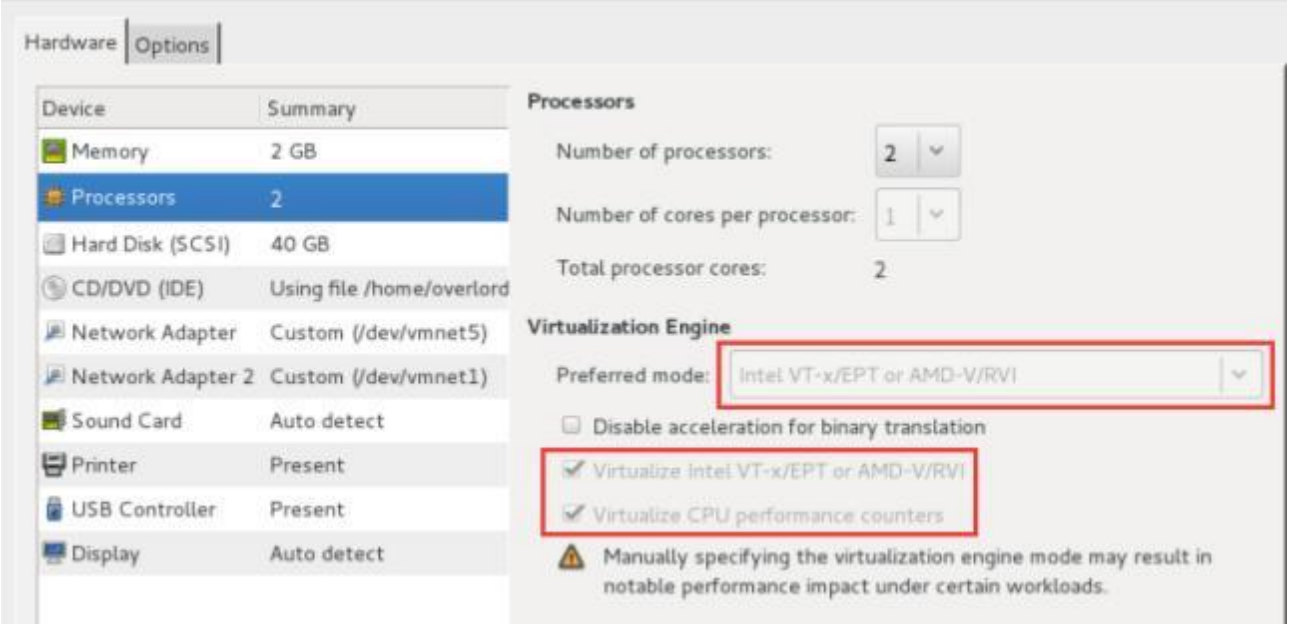
启动计算服务并设置开机自动运行：

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

✧ 安装和配置计算节点

本章介绍在 Compute node(计算节点)安装和配置计算服务。

计算节点所在的 VMware workstation 虚拟机需要打开 CPU 虚拟化支持。如图所示:



如果你的虚拟机没有打开，请先关闭虚拟机，打开 CPU 虚拟化支持后再开机。

— 安装和配置组件

1. 安装软件包:

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

说明：如果出现下面的错误，请在a.repo文件最后增加以下内容。

```
错误：错误：软件包：1:openstack-nova-compute-17.0.7-1.el7.noarch (openstack-queens)
           需要：qemu-kvm-rhev >= 2.10.0
 您可以尝试添加 --skip-broken 选项来解决该问题
 您可以尝试执行：rpm -Va --nofiles --nodigest
```

```
[Virt]
name=CentOS-$releasever - Base
baseurl=https://mirrors.tuna.tsinghua.edu.cn/centos-vault/7.5.1804/virt/x86_64/kvm-common/
gpgcheck=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
```

2. 编辑/etc/nova/nova.conf 文件，并完成下列操作：

A. 在[DEFAULT]和[oslo_messaging_rabbit]小节，配置 RabbitMQ 消息队列访问：

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

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

替换 *RABBIT_PASS*为 RabbitMQ 用户 openstack 的密码。

B. 在[DEFAULT]和[keystone_authtoken]小节，配置身份服务访问信息：

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

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

替换 *NOVA_PASS*为身份服务用户 nova 的密码。

C. 在[DEFAULT]小节，配置 my_ip 配置项：

```
[DEFAULT]

...

my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
```

替换 *MANAGEMENT_INTERFACE_IP_ADDRESS* 为计算节点管理接口的 IP 地址。在本例中，计算节点的管理接口 IP 为 10.0.0.11。

D. 在[DEFAULT]小节，启用支持 neutron 网络服务：

```
[DEFAULT]
```

```
...
use_neutron = True
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

E. 在[vnc]小节，启用并配置远程控制访问信息：

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

注：如果 controller 不能解析到管理节点的 IP 地址，请修改你的 DNS 或 host 文件。

F. 在[glance]小节， 配置镜像服务 API 的位置：

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

G. 在[oslo_concurrency]小节，配置锁路径：

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

一 完成安装

1. 探测你的计算节点是否支持硬件虚拟机化：

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

如果结果大于或等于 1，则表示你的计算节点支持硬件虚拟化。如果等于 0，则表示不支持，那么你必须配置 libvirt，用 QEMU 代替 KVM。

编辑/etc/nova/nova.conf 文件中的[libvirt]小节：

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

2. 启动计算服务及其依赖服务，并这是开机自动运行：

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

验证操作

在管理节点进行下列操作。

1. 执行 admin 凭据脚本，以便以 admin 身份执行后续命令：

```
# . admin-openrc
```

2. 通过列出服务组件，确认每一个进程已经成功启动和注册：

```
# openstack compute service list
```

```
+----+------------------+------------+----------+---------+-------+----------------------------+
| Id | Binary           | Host       | Zone     | Status  | State | Updated At                 |
+----+------------------+------------+----------+---------+-------+----------------------------+
|  1 | nova-consoleauth | controller | internal | enabled | up    | 2016-07-21T07:58:47.000000 |
|  2 | nova-conductor   | controller | internal | enabled | up    | 2016-07-21T07:58:45.000000 |
|  3 | nova-scheduler   | controller | internal | enabled | up    | 2016-07-21T07:58:48.000000 |
|  6 | nova-compute     | compute1   | nova     | enabled | up    | 2016-07-21T07:58:48.000000 |
+----+------------------+------------+----------+---------+-------+----------------------------+
```

七、安装配置 Networking Service(网络服务)

OpenStack 网络服务(neutron)，管理所有网络方面的内容。包括虚拟网络基础架构(VNI)和接入层方面的物理网络基础架构(PNI)。

✧ 安装和配置管理节点

本章介绍在 Controller node(管理节点)安装和配置网络服务。

— 先决条件

在配置 Openstack Networking(neutron) service 之前，必须创建数据库，服务凭据和 API 端点。

1. 创建数据库，并完成下列步骤:

   A. 使用数据库命令行客户端，以 root 身份登录数据库服务器。

   ```
   # mysql -u root  -p
   ```

   B. 创建 neutron 数据库

   ```
   CREATE DATABASE neutron;
   ```

   C. 创建数据库用户 neutron，并授予数据库用户 neutron 访问 neutron 数据库的权限。

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

   替换 *NEUTRON_DBPASS*为一个合适的密码。

   D. 退出数据库

2. 执行 admin 凭据脚本，以便以 admin 身份执行后续命令:

```
# . admin-openrc
```

3. 创建服务凭据，并完成下列步骤：

A. 创建 neutron 用户：

```
# openstack user create --domain default --password-prompt neutron
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | 5dfa4c3356474e97b7782eba0f46d710 |
| enabled   | True                             |
| id        | 7d95183bac4a4282a18ca4ba8c580a50 |
| name      | neutron                          |
+-----------+----------------------------------+
```

B. 添加 admin 角色到 neutron 用户和 service 项目

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

C. 创建 neutron 服务实体：

```
# openstack service create --name neutron --description "OpenStack
Networking" network
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Networking             |
| enabled     | True                             |
| id          | e8ea198b90ea4136a3baff61d28ec385 |
| name        | neutron                          |
| type        | network                          |
+-------------+----------------------------------+
```

4. 创建网络服务的 API 端点：

```
# openstack endpoint create --region RegionOne network public
http://controller:9696
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
```

```
| id           | e9f335a9f72f40fbac3ed346f8d6144c |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | e8ea198b90ea4136a3baff61d28ec385 |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+
```

```
# openstack endpoint create --region RegionOne network internal http://controller:9696
```

```
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 274be25144b140fbaebf35e064bd33bb |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | e8ea198b90ea4136a3baff61d28ec385 |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+
```

```
# openstack endpoint create --region RegionOne network admin http://controller:9696
```

```
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 7f87357bf54c4ab3ad97b07e2e1a56d5 |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | e8ea198b90ea4136a3baff61d28ec385 |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+
```

一 安装并配置服务组件

在 controller node(管理节点)安装和配置网络组件。

1. 安装组件

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

2. 编辑/etc/neutron/neutron.conf 文件并完成下列操作：

A. 在[database]小节，配置数据库访问信息：

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

替换 *NEUTRON_DBPASS* 为你的网络服务数据库用户 neutron 的密码。

B. 在[DEFAULT]小节，启用二层模块(ML2)插件，路由服务和重叠地址功能：

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

C. 在[DEFAULT]和[oslo_messaging_rabbit]小节，配置 RabbitMQ 消息队列访问信息：

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

[oslo_messaging_rabbit]
```

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

替换 *RABBIT_PASS* 为 RabbitMQ 用户 openstack 的密码。

D. 在[DEFAULT]和[keystone_authtoken]小节，配置身份服务访问信息：

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

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

替换 *NEUTRON_PASS* 为身份服务用户 neutron 的密码。

E. 在[DEFAULT]和[nova]小节，配置当网络拓扑发生改变时向计算服务发送网络通知。

```
[DEFAULT]
...
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True

[nova]
...
auth url = http://controller:35357
auth_type = password
project domain name = default
user_domain_name = default
region_name = RegionOne
project_name = service
```

```
username = nova
password = NOVA_PASS
```

替换 *NOVA_PASS*为身份服务用户 nova 的密码。

F. 在[oslo_concurrency]小节，配置锁路径：

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

配置二层(ML2)模块插件

ML2 插件使用 Linux bridge 机制为云主机建立二层虚拟网络基础。

1. 编辑/etc/neutron/plugins/ml2/ml2_conf.ini 文件并完成下列操作：

A. 在[ml2]小节，启用 flat,VLAN 和 VXLAN 网络：

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

B. 在[ml2]小节，启用 VXLAN 为用户自定义网络：

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

C. 在[ml2]小节，启用 Linux bridge 和 layer-2 population 机制：

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

D. 在[ml2]小节，启用端口安全扩展驱动：

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

E. 在[ml2_type_flat]小节，配置 provider 虚拟网络使用 flat 网络：

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

F. 在[ml2_type_vxlan]小节，配置自定义 VXLAN 网络的 id 范围：

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

G. 在[securitygroup]小节，启用 ipset 增强安全组的工作效率：

```
[securitygroup]
...
enable_ipset = True
```

一 配置 Linux bridge agent

Linux bridge agent 为云主机和处理安全组建立二层虚拟网络基础。

1. 编辑/etc/neutron/plugins/ml2/linuxbridge_agent.ini 文件并完成下列操作：

A. 在[linux_bridge]小节，映射 provider 虚拟网络到 provider 物

理网络接口：

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

替换 *PROVIDER_INTERFACE_NAME*为 provider 物理网络接口的名字。

B. 在[vxlan]小节，启用 VXLAN 覆盖网络，配置处理覆盖网络物理网络接口的 IP 地址。启用 layer-2 population:

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

替换*OVERLAY_INTERFACE_IP_ADDRESS*为管理节点管理接口的 IP 地址。

C. 在[securitygroup]小节， 启用安全组并配置 Linux bridge iptables 防火墙驱动：

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

配置三层代理

Layer-3 (L3) agent 为自定义虚拟网络提供路由和 NAT 服务。

1. 编辑/etc/neutron/l3_agent.ini 文件并完成下列操作：

在[DEFAULT]小节， 配置 Linux bridge 接口驱动和外部网络网桥：

```
[DEFAULT]
...
interface_driver = neutron.agent.linux.interface.BridgeInterfaceDriver
external_network_bridge =
```

一 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 文件并完成下列操作：

   在[DEFAULT]小节， 配置 Linux bridge 接口驱动，Dnsmasq DHCP 驱动并启用 isolated metadata，以便云主机可以通过 provider 网络访问元数据：

```
[DEFAULT]
...
interface_driver = neutron.agent.linux.interface.BridgeInterfaceDriver
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
enable_isolated_metadata = True
```

一 配置元数据代理

metadata agent(元数据代理)提供配置信息，例如云主机的凭据。

1. 编辑/etc/neutron/metadata_agent.ini 文件并完成下列操作：

   在[DEFAULT]小节， 配置元数据主机和共享秘钥：

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

   替换 *METADATA_SECRET*为一个合适的密码。

一 配置计算节点使用 neutron 网络

1. 编辑/etc/nova/nova.conf 文件并完成下列操作:

   在[neutron]小节， 配置访问参数， 启用元数据代理， 并配置共享秘钥:

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

service_metadata_proxy = True
metadata_proxy_shared_secret = METADATA_SECRET
```

   替换 *NEUTRON_PASS*为身份服务中 neutron 用户的密码。

   替换 *METADATA_SECRET*为/etc/neutron/metadata_agent.ini 文件中相同的密码。

完成安装

1. 网络服务初始化脚本/etc/neutron/plugin.ini 实际上是一个链接文件，它指向 ML2 插件的配置文件/etc/neutron/plugins/ml2/ml2_conf.ini，如果该链接文件不存在，则需要使用下面的命令创建:

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

2. 初始化数据库

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

3. 重启计算节点的 API 服务

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

4. 启动网络服务并配置开机自动运行：

```
# systemctl enable neutron-server.service neutron-linuxbridge-agent.service neutron-dhcp-agent.service neutron-metadata-agent.service neutron-l3-agent.service
```

```
# systemctl start neutron-server.service neutron-linuxbridge-agent.service neutron-dhcp-agent.service neutron-metadata-agent.service neutron-l3-agent.service
```

✧ 安装和配置计算节点

计算节点负责处理云主机的连接性和安全组。在 compute node(计算节点)完成下列操作。

安装组件

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

配置公共组件

网络服务公共组件配置包括认证机制，消息队列和插件。

1. 编辑/etc/neutron/neutron.conf 文件并完成下列操作：

   A. 在[database]小节， 注释掉所有 connection 配置项。因为计算节点不需要直接访问数据库。

   B. 在[DEFAULT]和[oslo_messaging_rabbit]小节，配置 RabbitMQ 消息队列访问信息：

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

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

```
rabbit_password = RABBIT_PASS
```

替换 *RABBIT_PASS*为 RabbitMQ 用户 openstack 的密码。

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

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

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

替换 *NEUTRON_PASS*为身份服务中 neutron 用户的密码。

D. 在[oslo_concurrency]小节， 配置锁路径：

```
[oslo_concurrency]

...

lock_path = /var/lib/neutron/tmp
```

一 配置 Linux bridge 代理

Linux bridge 代理为云主机建立二层虚拟网络基础结构并处理安全组。

1. 编辑/etc/neutron/plugins/ml2/linuxbridge_agent.ini 文件并完成下列操作：

A. 在[linux_bridge]小节，映射 provider 虚拟网络到 provider 物理网络接口：

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

替换 *PROVIDER_INTERFACE_NAME*为 provider 网络的物理网络接口。

B. 在[vxlan]小节，启用 VXLAN 覆盖网络，配置处理覆盖网络物理网络接口的 IP 地址。启用 layer-2 population:

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

替换*OVERLAY_INTERFACE_IP_ADDRESS*为计算节点管理接口的 IP 地址。

C. 在[securitygroup]小节，启用安全组并配置 Linux bridge iptables 防火墙驱动：

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

一 配置计算节点使用 neutron 网络

1. 编辑/etc/nova/nova.conf 文件并完成下列操作：

在[neutron]小节，配置访问参数：

```
[neutron]
...
url = http://controller:9696
auth_url = http://controller:35357
auth_type = password
project_domain_name = default
```

```
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
```

替换 *NEUTRON_PASS* 为身份服务中 neutron 用户的密码。

完成安装

1. 重启计算服务：

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

2. 启动 Linux bridge 代理并设置开机自动运行：

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

验证操作

1. 执行 admin 凭据脚本，以便以 admin 身份执行后续命令：

```
# . admin-openrc
```

2. 列出加载的扩展模块，确认 neutron-server 服务进程成功启动：

```
# neutron ext-list
```

```
+---------------------------+-----------------------------------------------+
| alias                     | name                                          |
+---------------------------+-----------------------------------------------+
| default-subnetpools       | Default Subnetpools                           |
| network-ip-availability   | Network IP Availability                       |
| network_availability_zone | Network Availability Zone                     |
| auto-allocated-topology   | Auto Allocated Topology Services              |
| ext-gw-mode               | Neutron L3 Configurable external gateway mode |
| binding                   | Port Binding                                  |
| agent                     | agent                                         |
| subnet_allocation         | Subnet Allocation                             |
| l3_agent_scheduler        | L3 Agent Scheduler                            |
| tag                       | Tag support                                   |
| external-net              | Neutron external network                      |
| net-mtu                   | Network MTU                                   |
```

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

3. 列出代理确认 neutron 代理成功启动：

```
# neutron agent-list
```

```
+--------------------------------------+--------------------+-----------+-------------------+-------+----------------+---------------------------+
| id                                   | agent_type         | host      | availability_zone | alive | admin_state_up | binary                    |
+--------------------------------------+--------------------+-----------+-------------------+-------+----------------+---------------------------+
| 1d510f9c-1029-409c-be64-3d89712616b4 | Metadata agent     | controller |                   | :-)   | True           | neutron-metadata-agent    |
| 1e319932-5726-47f3-9f86-5dc55aa6435d | L3 agent           | controller | nova              | :-)   | True           | neutron-l3-agent          |
| a51d8a6f-47b7-42be-a9c4-7d6018551beb | Linux bridge agent | compute1   |                   | :-)   | True           | neutron-linuxbridge-agent |
| a63c1e95-f708-47a6-a11a-9de0fd9cb2e0 | DHCP agent         | controller | nova              | :-)   | True           | neutron-dhcp-agent        |
| fb739c3d-39d0-471f-b861-3b25ba903cb3 | Linux bridge agent | controller |                   | :-)   | True           | neutron-linuxbridge-agent |
+--------------------------------------+--------------------+-----------+-------------------+-------+----------------+---------------------------+
```

一 修改 VMware workstation 虚拟机和系统配置

因为 VMware workstation 虚拟机默认开启网卡 MAC 地址检查，该功能将导致云主机与 provider 网络通信异常。物理设备不会出现此异常。因此需要对虚拟机的配置文件做修改。关闭虚拟机网卡的 MAC 地址检查。将所以虚拟机关机，并修改所有虚拟机的 vmx 配置文件， 添加如下配置项：

```
ethernet0.checkMACAddress = "FALSE"
ethernet1.checkMACAddress = "FALSE"
```

注：两个网卡的虚拟机添加两项，一个网卡的虚拟机添加第一项。

如果使用 Linux 版本的 VMware workstation，还需要将相关网卡设备的权限设置为 666,命令如下：

```
chmod 666 /dev/vmnet*
```

创建 provider 网络

1. 在管理节点执行 admin 凭据脚本，以便以 admin 身份执行后续命令：

```
# . admin-openrc
```

2. 创建网络：

```
# neutron net-create --shared --provider:physical_network
provider --provider:network_type flat provider
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2016-07-21T16:32:36                  |
| description               |                                      |
| id                        | 19fd8bb4-c17a-4af4-a16e-fd85e9c0e05b |
| ipv4_address_scope        |                                      |
| ipv6_address_scope        |                                      |
| mtu                       | 1500                                 |
| name                      | provider                             |
| port_security_enabled     | True                                 |
| provider:network_type     | flat                                 |
| provider:physical_network | provider                             |
| provider:segmentation_id  |                                      |
| router:external           | False                                |
| shared                    | True                                 |
| status                    | ACTIVE                               |
```

```
| subnets                   |                                      |
| tags                      |                                      |
| tenant_id                 | 782fda82df3449e7893f82c942efe1f8     |
| updated_at                | 2016-07-21T16:32:36                  |
+---------------------------+--------------------------------------+
```

3. 在 provider 网络上创建子网：

```
# neutron subnet-create --name provider --allocation-pool start=172.16.100.101,end=172.16.100.250 --dns-nameserver 202.106.0.20 --gateway 172.16.100.1 provider 172.16.100.0/24
```

```
+-------------------+-------------------------------------------------------+
| Field             | Value                                                 |
+-------------------+-------------------------------------------------------+
| allocation_pools  | {"start": "172.16.100.101", "end": "172.16.100.250"}  |
| cidr              | 172.16.100.0/24                                       |
| created_at        | 2016-07-21T16:32:43                                   |
| description       |                                                       |
| dns_nameservers   | 202.106.0.20                                          |
| enable_dhcp       | True                                                  |
| gateway_ip        | 172.16.100.1                                          |
| host_routes       |                                                       |
| id                | 326b4690-51c3-485a-819b-a2c660ae3eba                  |
| ip_version        | 4                                                     |
| ipv6_address_mode |                                                       |
| ipv6_ra_mode      |                                                       |
| name              | provider                                              |
| network_id        | 19fd8bb4-c17a-4af4-a16e-fd85e9c0e05b                  |
| subnetpool_id     |                                                       |
| tenant_id         | 782fda82df3449e7893f82c942efe1f8                      |
| updated_at        | 2016-07-21T16:32:43                                   |
+-------------------+-------------------------------------------------------+
```

创建自定义网络

1. 在管理节点，执行 demo 凭据脚本， 以便以 demo 身份执行后续命令：

```
# . demo-openrc
```

2. 创建网络

```
# neutron net-create selfservice
```

```
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2016-07-21T16:36:25                  |
| description               |                                      |
| id                        | 7ba9e684-b133-4c27-a63e-4fc7a5e39034 |
| ipv4_address_scope        |                                      |
| ipv6_address_scope        |                                      |
| mtu                       | 1450                                 |
| name                      | selfservice                          |
| port_security_enabled     | True                                 |
| router:external           | False                                |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tags                      |                                      |
| tenant_id                 | 8695acb7e10e4c44a25da3076b2671b4     |
| updated_at                | 2016-07-21T16:36:25                  |
+---------------------------+--------------------------------------+
```

3. 在自定义网络上创建子网

```
# neutron subnet-create --name selfservice --dns-nameserver 202.106.0.20 --gateway 192.168.111.1 selfservice 192.168.111.0/24
```

```
+-------------------+-------------------------------------------------+
| Field             | Value                                           |
+-------------------+-------------------------------------------------+
| allocation_pools  | {"start": "192.168.111.2", "end": "192.168.111.254"} |
| cidr              | 192.168.111.0/24                                |
| created_at        | 2016-07-21T16:38:30                             |
| description       |                                                 |
| dns_nameservers   | 202.106.0.20                                    |
| enable_dhcp       | True                                            |
| gateway_ip        | 192.168.111.1                                   |
| host_routes       |                                                 |
| id                | bc052f13-b07e-4954-bb19-c9b36aaea142            |
```

```
| ip_version          | 4                                                  |
| ipv6_address_mode   |                                                    |
| ipv6_ra_mode        |                                                    |
| name                | selfservice                                        |
| network_id          | 7ba9e684-b133-4c27-a63e-4fc7a5e39034               |
| subnetpool_id       |                                                    |
| tenant_id           | 8695acb7e10e4c44a25da3076b2671b4                   |
| updated_at          | 2016-07-21T16:38:30                                |
+---------------------+----------------------------------------------------+
```

一 创建一个路由器

1. 在管理节点，执行 admin 凭据脚本，以便以 admin 身份执行后续命令：

```
# . admin-openrc
```

2. 添加 router:external 选项到 provider 网络：

```
# neutron net-update provider --router:external
Updated network: provider
```

3. 执行 demo 凭据脚本，以便以 demo 身份执行后续命令：

```
# . demo-openrc
```

4. 创建路由器

```
# neutron router-create router
Created a new router:
+-------------------------+--------------------------------------+
| Field                   | Value                                |
+-------------------------+--------------------------------------+
| admin_state_up          | True                                 |
| availability_zone_hints |                                      |
| availability_zones      |                                      |
| description             |                                      |
| external_gateway_info   |                                      |
| id                      | 268f625c-b7ce-4b17-b84a-39a32c180a50 |
| name                    | router                               |
| routes                  |                                      |
```

```
| status                    | ACTIVE                           |
| tenant_id                 | 8695acb7e10e4c44a25da3076b2671b4 |
+---------------------------+----------------------------------+
```

5. 将自定义网络的子网连接到路由器的接口：

```
# neutron router-interface-add router selfservice
Added interface 049fd487-866c-49db-9425-9e3a65d61ec2 to router router.
```

6. 设置 provider 网络为路由器的网关：

```
# neutron router-gateway-set router provider
Set gateway for router router
```

一 验证操作

1. 在管理节点， 执行 admin 凭据脚本，以便以 admin 身份执行后续命令：

```
# . admin-openrc
```

2. 列出网络命名空间。你因该看到一个 qrouter 命名空间和两个 qdhcp 命名空间：

```
# ip netns
qrouter-268f625c-b7ce-4b17-b84a-39a32c180a50 (id: 2)
qdhcp-7ba9e684-b133-4c27-a63e-4fc7a5e39034 (id: 1)
qdhcp-19fd8bb4-c17a-4af4-a16e-fd85e9c0e05b (id: 0)
```

3. 列出路由器端口，查看 provider 网络分配给路由器的网关 IP：

```
#  neutron router-port-list router
+--------------------------+------+-------------------+-------------------------------+
| id                       | name | mac_address       | fixed_ips                     |
+--------------------------+------+-------------------+-------------------------------+
| 049fd487-866c-           |      | fa:16:3e:51:fc:28 | {"subnet_id": "bc052f13-b07 |
| 49db-9425-9e3a65d61ec2   |      |                   | e-4954-bb19-c9b36aaea142",  |
|                          |      |                   | "ip_address":               |
```

```
|                              |      |                   | "192.168.111.1"}                |
| 2cedee5a-e26a-               |      | fa:16:3e:1f:f2:7a | {"subnet_id":                   |
| 4b91-bd49-43ee1c6b6803       |      |                   | "326b4690-51c3-485a-819b-       |
|                              |      |                   | a2c660ae3eba",                  |
|                              |      |                   | "ip_address":                   |
|                              |      |                   | "172.16.100.102"}               |
+------------------------------+------+-------------------+---------------------------------+
```

4. 从 provider 网络其他主机 ping 该地址：

```
$ ping 172.16.100.102 -c 4
PING 172.16.100.102 (172.16.100.102) 56(84) bytes of data.
64 bytes from 172.16.100.102: icmp_seq=1 ttl=64 time=0.491 ms
64 bytes from 172.16.100.102: icmp_seq=2 ttl=64 time=0.386 ms
64 bytes from 172.16.100.102: icmp_seq=3 ttl=64 time=0.376 ms
64 bytes from 172.16.100.102: icmp_seq=4 ttl=64 time=0.405 ms

--- 172.16.100.102 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.376/0.414/0.491/0.049 ms
```

八、安装和配置 Dashboard

Dashboard(horizon)是一个 web 界面，可以通过它以管理员或普通用户身份管理 OpenStack 资源和服务。

本章介绍在 controller node(管理节点)安装和配置 dashboard。

安装和配置组件

1. 安装软件包

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

2. 编辑/etc/openstack-dashboard/local_settings 文件，并完成下列操作：

A. 配置 dashboard 使用管理节点的 Openstack 服务：

```
OPENSTACK_HOST = "controller"
```

B. 允许所有客户端访问 dashboard：

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

C. 配置 memcached 会话存储服务：

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

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

D. 启用身份服务 API 版本 3：

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

E. 启用支持多域：

```
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
```

F. 配置 API 版本：

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

G. 配置 default 为你在 dashboard 创建的用户的默认域：

```
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "default"
```

H. 配置 user 为你在 dashboard 创建用户的默认角色：

```
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
```

I. 配置时区：

```
TIME_ZONE = "Asia/Shanghai"
```

完成安装

重启 web 服务和会话存储服务：

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

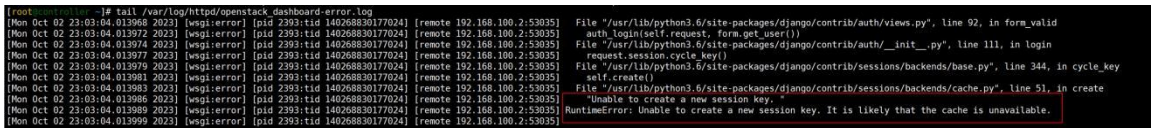
验证操作

使用http://controller/dashboard访问 dashboard。

认证使用 admin 或 demo 用户。域为 default。

故障排除：登录dashboard提示以下错误

解决方案：
在命令行查看日志信息。



问题原因：

无法创建新的session key，cache不可用。

解决方法：

修改dashboard中SESSION_ENGINE的配置，将存储方式由cache改为file

```
# vim /etc/openstack-dashboard/local_settings
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
修改为以下代码
SESSION_ENGINE = 'django.contrib.sessions.backends.file'
```

重启服务

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

九、安装和配置 Block Storage 服务(块存储服务)

Block Storage service(cinder)为云主机提供块存储设备(云硬盘)。

✧ 安装和配置管理节点

本节介绍在管理节点安装配置块存储服务。

— 先决条件

在安装和配置块存储服务之前，必须创建数据库、服务凭据和 API 端点。

1. 创建数据库，并完成下列步骤：

   A. 使用数据库命令行客户端，以 root 身份登录数据库服务器。

   `# mysql -u root -p`

   B. 创建 cinder 数据库

   `CREATE DATABASE cinder;`

   C. 创建数据库用户 cinder，并授予数据库用户 cinder 访问 cinder 数据库的权限。

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

替换 *CINDER_DBPASS* 为一个合适的密码。

D. 退出数据库

2. 执行 admin 凭据脚本，以便以 admin 身份执行后续命令：

```
# . admin-openrc
```

3. 创建服务凭据，并完成下列步骤：

A. 创建 cinder 用户

```
# openstack user create --domain default --password-prompt cinder
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | 5dfa4c3356474e97b7782eba0f46d710 |
| enabled   | True                             |
| id        | 7f3f2cdff2f74bf0b9d3b57f90299975 |
| name      | cinder                           |
+-----------+----------------------------------+
```

B. 添加 admin 角色到 cinder 用户和 service 项目：

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

C. 创建 cinder 和 cinderv2 服务实体：

```
# openstack service create --name cinder --description "OpenStack Block
Storage" volume
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Block Storage          |
| enabled     | True                             |
```

```
| id          | 4ea24af2e20746d4a49519ff64f8c4c0 |
| name        | cinder                           |
| type        | volume                           |
+-------------+----------------------------------+
# openstack service create --name cinderv2 --description "OpenStack Block Storage" volumev2
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Block Storage          |
| enabled     | True                             |
| id          | 64d1fcc62cde47d7b0787cbeec942aa6 |
| name        | cinderv2                         |
| type        | volumev2                         |
+-------------+----------------------------------+
```

4. 创建块存储服务 API 端点:

```
# openstack endpoint create --region RegionOne volume public http://controller:8776/v1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 8bc0206a30e2404a81eeab3c6ec9be61        |
| interface    | public                                  |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | 4ea24af2e20746d4a49519ff64f8c4c0        |
| service_name | cinder                                  |
| service_type | volume                                  |
| url          | http://controller:8776/v1/%(tenant_id)s |
+--------------+-----------------------------------------+
# openstack endpoint create --region RegionOne volume internal http://controller:8776/v1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 9bb37a3217e44db880d317dc1a3c3cb8        |
| interface    | internal                                |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | 4ea24af2e20746d4a49519ff64f8c4c0        |
| service_name | cinder                                  |
```

```
| service type | volume                                       |
| url          | http://controller:8776/v1/%(tenant_id)s      |
+--------------+----------------------------------------------+
# openstack endpoint create --region RegionOne volume admin http://controller:8776/v1/%\(tenant_id\)s
+--------------+----------------------------------------------+
| Field        | Value                                        |
+--------------+----------------------------------------------+
| enabled      | True                                         |
| id           | ab86bae27bc94f69b3bbbf0d47e25b53             |
| interface    | admin                                        |
| region       | RegionOne                                    |
| region_id    | RegionOne                                    |
| service_id   | 4ea24af2e20746d4a49519ff64f8c4c0             |
| service_name | cinder                                       |
| service_type | volume                                       |
| url          | http://controller:8776/v1/%(tenant_id)s      |
+--------------+----------------------------------------------+
# openstack endpoint create --region RegionOne volumev2 public http://controller:8776/v2/%\(tenant_id\)s
+--------------+----------------------------------------------+
| Field        | Value                                        |
+--------------+----------------------------------------------+
| enabled      | True                                         |
| id           | 6318f85f592d414e81d19dc1269d0eed             |
| interface    | public                                       |
| region       | RegionOne                                    |
| region_id    | RegionOne                                    |
| service_id   | 64d1fcc62cde47d7b0787cbeec942aa6             |
| service_name | cinderv2                                     |
| service_type | volumev2                                     |
| url          | http://controller:8776/v2/%(tenant_id)s      |
+--------------+----------------------------------------------+
# openstack endpoint create --region RegionOne volumev2 internal http://controller:8776/v2/%\(tenant_id\)s
+--------------+----------------------------------------------+
| Field        | Value                                        |
+--------------+----------------------------------------------+
| enabled      | True                                         |
| id           | 1a79867098fb4bc78c6ae2470b7f4776             |
| interface    | internal                                     |
| region       | RegionOne                                    |
| region_id    | RegionOne                                    |
| service_id   | 64d1fcc62cde47d7b0787cbeec942aa6             |
```

```
| service name | cinderv2                                      |
| service_type | volumev2                                      |
| url          | http://controller:8776/v2/%(tenant_id)s       |
+--------------+-----------------------------------------------+
# openstack endpoint create --region RegionOne volumev2 admin
http://controller:8776/v2/%\(tenant_id\)s
+--------------+-----------------------------------------------+
| Field        | Value                                         |
+--------------+-----------------------------------------------+
| enabled      | True                                          |
| id           | 888d079eafba4c2e973626811d405f8a              |
| interface    | admin                                         |
| region       | RegionOne                                     |
| region_id    | RegionOne                                     |
| service_id   | 64d1fcc62cde47d7b0787cbeec942aa6              |
| service_name | cinderv2                                      |
| service_type | volumev2                                      |
| url          | http://controller:8776/v2/%(tenant_id)s       |
+--------------+-----------------------------------------------+
```

安装和配置组件

1. 安装软件包

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

2. 编辑/etc/cinder/cinder.conf 文件并完成下列操作：

A. 在[database]小节，配置数据库访问：

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

替换 *CINDER_DBPASS*为数据库用户 cinder 的密码。

B. 在[DEFAULT]和[oslo_messaging_rabbit]小节，配置 RabbitMQ 消息队列访问信息：

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

[oslo_messaging_rabbit]
```

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

替换 *RABBIT_PASS*为 RabbitMQ 用户 openstack 的密码。

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

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

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

替换 *CINDER_PASS*为身份服务用户 cinder 的密码。

D. 在[DEFAULT]小节， 配置 my_ip 配置项为管理节点管理接口的 IP 地址：

```
[DEFAULT]
...
my_ip = 10.0.0.10
```

E. 在[oslo_concurrency]小节， 配置锁路径：

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

3. 初始化块存储的数据库

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

一 配置计算组件使用块存储

编辑/etc/nova/nova.conf 文件，添加下列内容：

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

完成安装

1. 重启计算组件的 API 服务：

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

2. 启动块存储服务并设置开机自动运行：

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

✧ 安装和配置一个存储节点

本节介绍为块存储节点安装配置块存储服务。块存储节点服务器需要至少额外添加一块新磁盘，比如/dev/sdb。

先决条件

1. 安装支持软件包

A. 安装 LVM 软件包

```
# yum install lvm2 -y
```

B. 启动 LVM metadata 服务并配置开机自动运行：

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

2. 创建 LVM 物理卷/dev/sdb

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

3. 创建 LVM 卷组 cinder-volumes：

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

4. 为了提升系统工作效率，使其 LVM 只扫描 cinder 使用的设置。编辑/etc/lvm/lvm.conf 文件，并完成下列操作：

在 devices 小节，添加过滤器允许/dev/sda(如果系统安装时使用 LVM 机制则需要允许 sda)和/dev/sdb，并拒绝所有其他设备：

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

二 安装和配置组件

1. 安装软件包

```
# yum install openstack-cinder targetcli python-keystonemiddleware -y
```

2. 编辑/etc/cinder/cinder.conf 文件并完成下列操作：

A. 在[database]小节，配置数据库访信息：

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

替换 *CINDER_DBPASS* 为数据库用户 cinder 的密码。

B. 在[DEFAULT]和[oslo_messaging_rabbit]小节，配置 RabbitMQ

消息队列访问信息:

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

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

替换 *RABBIT_PASS*为 RabbitMQ 用户 openstack 的密码。

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

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

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

替换 *CINDER_PASS*为身份服务用户 cinder 的密码。

D. 在[DEFAULT]小节， 配置 my_ip 配置项:

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

替换 *MANAGEMENT_INTERFACE_IP_ADDRESS* 为块存储节点管理接

口的 IP 地址。在本案例中为 10.0.0.12

E. 在[lvm]小节，配置 LVM 后端驱动， 使用 cinder-volumes 卷，使用 iSCSI 协议和适当的 iSCSI 服务：

```
[lvm]
...
volume_driver                                                    =
cinder.volume.drivers.lvm.LVMVolumeDriver
volume_group = cinder-volumes
iscsi_protocol = iscsi
iscsi_helper = lioadm
```

F. 在[DEFAULT]小节， 启用 LVM 后端

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

G. 在[DEFAULT]小节， 配置镜像服务 API 的位置：

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

H. 在[oslo_concurrency]小节， 配置锁路径:

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

— 完成安装

启动块存储卷服务及其依赖服务。并设置它们开机自动运行:

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

— 验证操作

1. 在管理节点执行 admin 凭据脚本，以便以 admin 身份执行后续命令:

```
# . admin-openrc
```

2. 列出服务组件确认每个进程成功启动:

```
# cinder service-list
```

```
+------------------+------------+------+---------+-------+----------------------------+-----------------+
|      Binary      |    Host    | Zone |  Status | State |         Updated_at         | Disabled Reason |
+------------------+------------+------+---------+-------+----------------------------+-----------------+
| cinder-scheduler | controller | nova | enabled |   up  | 2016-07-22T03:41:53.000000 |        -        |
|  cinder-volume   | block1@lvm | nova | enabled |   up  | 2016-07-22T03:41:54.000000 |        -        |
+------------------+------------+------+---------+-------+----------------------------+-----------------+
```

十、安装和配置 Shared File Syatems Servcie(共享文件系统服务)

OpenStack 共享文件系统服务(manila)为云主机提供文件存储服务。

✧ 安装和配置管理节点

本节介绍在管理节点安装和配置文件共享服务。

在安装和配置共享文件系统服务前，必须创建数据库、服务凭据和 API 端点。

1. 创建数据库。并完成下列步骤:

   A. 使用数据库命令行客户端，以 root 身份登录数据库服务器。

   ```
   # mysql -u root -p
   ```

   B. 创建 manila 数据库

```
CREATE DATABASE manila;
```

C. 创建数据库用户 manila，并授予数据库用户 manila 访问manila 数据库的权限。

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

替换 *MANILA_DBPASS*为一个合适的密码。

2. 执行 admin 凭据脚本，以便以 admin 身份执行后续命令：

```
# . admin-openrc
```

3. 创建服务凭据，并完成下列步骤：

A. 创建 manila 用户

```
# openstack user create --domain default --password-prompt manila
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | 5dfa4c3356474e97b7782eba0f46d710 |
| enabled   | True                             |
| id        | c44f332c894245bfbba6dcc4af6809f5 |
| name      | manila                           |
+-----------+----------------------------------+
```

B. 添加 admin 角色到 manila 用户和 service 项目：

```
# openstack role add --project service --user manila admin
```

C. 创建 manila 和 manila2 服务实体：

```
# openstack service create --name manila --description "OpenStack Shared File Systems" share
+-------------+----------------------------------+
| Field       | Value                            |
```

```
+-------------+----------------------------------+
| description | OpenStack Shared File Systems    |
| enabled     | True                             |
| id          | 3da670b961d14d91aa5fb18bd43de1c0 |
| name        | manila                           |
| type        | share                            |
+-------------+----------------------------------+
# openstack service create --name manilav2 --description "OpenStack Shared File Systems" sharev2
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Shared File Systems    |
| enabled     | True                             |
| id          | 214073059d474140968915f16224ea1c |
| name        | manilav2                         |
| type        | sharev2                          |
+-------------+----------------------------------+
```

4. 创建共享文件系统服务 API 端点:

```
# openstack endpoint create --region RegionOne share public http://controller:8786/v1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | c97c206b78bf45e5bce121e4606b5a68        |
| interface    | public                                  |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | 3da670b961d14d91aa5fb18bd43de1c0        |
| service_name | manila                                  |
| service_type | share                                   |
| url          | http://controller:8786/v1/%(tenant_id)s |
+--------------+-----------------------------------------+
# openstack endpoint create --region RegionOne share internal http://controller:8786/v1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 4bc3b9a3a6a048b7b0e57e75bf04c1f1        |
| interface    | internal                                |
| region       | RegionOne                               |
```

```
| region id    | RegionOne                                |
| service_id   | 3da670b961d14d91aa5fb18bd43de1c0         |
| service_name | manila                                   |
| service_type | share                                    |
| url          | http://controller:8786/v1/%(tenant_id)s  |
+--------------+------------------------------------------+
# openstack endpoint create --region RegionOne share admin http://controller:8786/v1/%\(tenant_id\)s
+--------------+------------------------------------------+
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | 1c51b71dfaf74e4c81510bc16aa95837         |
| interface    | admin                                    |
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | 3da670b961d14d91aa5fb18bd43de1c0         |
| service_name | manila                                   |
| service_type | share                                    |
| url          | http://controller:8786/v1/%(tenant_id)s  |
+--------------+------------------------------------------+
# openstack endpoint create --region RegionOne sharev2 public http://controller:8786/v2/%\(tenant_id\)s
+--------------+------------------------------------------+
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | 59f15fd00bcd4277979ba74ac08832f9         |
| interface    | public                                   |
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | 214073059d474140968915f16224ea1c         |
| service_name | manilav2                                 |
| service_type | sharev2                                  |
| url          | http://controller:8786/v2/%(tenant_id)s  |
+--------------+------------------------------------------+
# openstack endpoint create --region RegionOne sharev2 internal http://controller:8786/v2/%\(tenant_id\)s
+--------------+------------------------------------------+
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | c37639b902564e75adac3b3320a7ff96         |
| interface    | internal                                 |
```

```
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | 214073059d474140968915f16224ea1c         |
| service_name | manilav2                                 |
| service_type | sharev2                                  |
| url          | http://controller:8786/v2/%(tenant_id)s  |
+--------------+------------------------------------------+
# openstack endpoint create --region RegionOne sharev2 admin http://controller:8786/v2/%\(tenant_id\)s
+--------------+------------------------------------------+
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | 9f7a78ba5dd84d6384220c600b869f6b         |
| interface    | admin                                    |
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | 214073059d474140968915f16224ea1c         |
| service_name | manilav2                                 |
| service_type | sharev2                                  |
| url          | http://controller:8786/v2/%(tenant_id)s  |
+--------------+------------------------------------------+
```

一 安装和配置组件

1. 安装软件包

```
# yum install openstack-manila python-manilaclient -y
```

2. 编辑/etc/manila/manila.conf 文件并完成下列操作:

A. 在[database]小节，配置数据库访问信息:

```
[database]
...
connection = mysql+pymysql://manila:MANILA_DBPASS@controller/manila
```

替换 *MANILA_DBPASS*为数据库用户 manila 的密码。

B. 在[DEFAULT]和[oslo_messaging_rabbit]小节，配置 RabbitMQ 消息队列访问信息:

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

```
rpc_backend = rabbit

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

替换 *RABBIT_PASS*为 RabbitMQ 用户 openstack 的密码。

C. 在[DEFAULT]小节， 设置下列配置值：

```
[DEFAULT]

...

default_share_type = default_share_type

rootwrap_config = /etc/manila/rootwrap.conf
```

D. 在[DEFAULT]和[keystone_authtoken]小节，配置身份服务访问信息：

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

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

替换 *MANILA_PASS*为身份服务用户 manila 的密码。

E. 在[DEFAULT]小节， 配置 my_ip 配置项为管理节点管理接口的 IP 地址：

```
[DEFAULT]
...
my_ip = 10.0.0.10
```

F. 在[oslo_concurrency]小节， 配置锁路径：

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

3. 初始化共享文件系统的数据库

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

完成安装

启动共享文件系统服务并配置开机自动运行：

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

✧ 安装和配置一个共享节点

本节介绍为共享文件系统服务安装和配置一个共享节点(利用块存储节点)

安装和配置组件

1. 安装软件包：

```
# yum install openstack-manila-share python2-PyMySQL -y
```

2. 编辑/etc/manial/manila.conf 文件，并完成下列操作：

A. 在[database]小节，配置数据库访问信息：

```
[database]

...

connection = mysql://manila:MANILA_DBPASS@controller/manila
```

替换 *MANILA_DBPASS*为数据库用户 manila 的密码。

B. 在[DEFAULT]和[oslo_messaging_rabbit]小节，配置 RabbitMQ 消息队列访问信息：

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

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

替换 *RABBIT_PASS*为 RabbitMQ 用户 openstack 的密码。

C. 在[DEFAULT]小节， 设置下列配置值：

```
[DEFAULT]

...

default_share_type = default_share_type

rootwrap_config = /etc/manila/rootwrap.conf
```

D. 在[DEFAULT]和[keystone_authtoken]小节，配置身份服务访问信息：

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

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

替换 *MANILA_PASS*为身份服务用户 manila 的密码。

E. 在[DEFAULT]小节， 配置 my_ip 配置项：

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

替换 *MANAGEMENT_INTERFACE_IP_ADDRESS* 为共享节点管理接口的 IP 地址。在本案例中为：10.0.0.12。

F. 在[oslo_concurrency]小节， 配置锁路径：

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

✧ 配置共享服务管理支持选项

在块存储节点完成下列操作.

一 先决条件

1. 安装网络服务组件

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

2. 编辑/etc/neutron/neutron.conf 文件并完成下列操作：

A. 在[database]小节， 注释掉所有 connection 配置项。因为计算节点不需要直接访问数据库。

B. 在[DEFAULT]和[oslo_messaging_rabbit]小节，配置 RabbitMQ 消息队列访问信息：

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

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

替换 *RABBIT_PASS*为 RabbitMQ 用户 openstack 的密码。

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

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

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

替换 *NEUTRON_PASS*为身份服务中 neutron 用户的密码。

D. 在[oslo_concurrency]小节， 配置锁路径：

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

3. 编辑/etc/neutron/plugins/ml2/linuxbridge_agent.ini 文件并完成下列操作：

A. 在[linux_bridge]小节，映射 provider 虚拟网络到 provider 物理网络接口：

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

替换 *PROVIDER_INTERFACE_NAME*为 provider 网络的物理网络接口。

B. 在[vxlan]小节，启用 VXLAN 覆盖网络，配置处理覆盖网络物理网络接口的 IP 地址。启用 layer-2 population:

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

替换*OVERLAY_INTERFACE_IP_ADDRESS*为计算节点管理接口的 IP 地址。

C. 在[securitygroup]小节， 启用安全组并配置 Linux bridge iptables 防火墙驱动：

```
[securitygroup]
...
```

```
enable_security_group = True
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

4. 启动 Linux bridge 代理并设置开机自动运行：

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

配置组件

编辑/etc/manila/manila.conf 文件并完成下列操作：

1. 在[DEFAULT]小节， 启用 generic 驱动和 NFS/CIFS 协议：

```
[DEFAULT]
...
enabled_share_backends = generic
enabled_share_protocols = NFS,CIFS
```

2. 在[neutron],[nova]和[cinder]小节，为他们的服务启用认证：

```
[neutron]
...
url = http://controller:9696
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS

[nova]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
```

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

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

替换*红色字体*为身份服务中相应用户的密码。

3. 在[generic]小节， 配置 generic 驱动：

```
[generic]
share_backend_name = GENERIC
share_driver = manila.share.drivers.generic.GenericShareDriver
driver_handles_share_servers = True
service_instance_flavor_id = 100
service_image_name = manila-service-image
service_instance_user = manila
service_instance_password = manila
interface_driver = manila.network.linux.interface.BridgeInterfaceDriver
```

完成安装

启动共享文件系统服务及其依赖服务，并配置他们开机自动运行：

```
# systemctl enable openstack-manila-share.service
# systemctl start openstack-manila-share.service
```

验证操作

1. 在管理节点执行 admin 凭据脚本，以便以 admin 身份执行后续命令：

```
# . admin-openrc
```

2. 列出服务组件确认每一个进程成功启动:

```
# manila service-list
```

```
+----+-----------------+---------------+------+---------+-------+----------------------------+
| Id | Binary          | Host          | Zone | Status  | State | Updated_at                 |
+----+-----------------+---------------+------+---------+-------+----------------------------+
| 1  | manila-scheduler | controller    | nova | enabled | up    | 2016-07-22T05:44:02.000000 |
| 2  | manila-share     | block1@generic | nova | enabled | up    | 2016-07-22T05:44:12.000000 |
+----+-----------------+---------------+------+---------+-------+----------------------------+
```

十一、安装和配置 Object Storage service(对象存储服务)

OpenStack 对象存储是一个多租户的对象存储系统，具备高度可扩展性。可以利用 RESTful HTTP API 以低廉的成本管理大量非结构化数据。

✧ 安装和配置管理节点

本节介绍在管理节点安装和配置代理服务。

— 先决条件

1. 执行 admin 凭据脚本，以便以 admin 身份执行后续命令:

```
# . admin-openrc
```

2. 创建身份服务凭据，并完成下列步骤:

A. 创建 swift 用户

```
# openstack user create --domain default --password-prompt swift
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | 5dfa4c3356474e97b7782eba0f46d710 |
| enabled   | True                             |
```

```
| id          | 54398f2d9c7b431e8ae8aee7be46bdc6 |
| name        | swift                            |
+-----------+----------------------------------+
```

B. 添加 admin 角色到 swift 用户和 service 项目：

```
# openstack role add --project service --user swift admin
```

C. 创建 swift 服务实体：

```
# openstack service create --name swift --description "OpenStack Object Storage" object-store
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Object Storage         |
| enabled     | True                             |
| id          | 7a85f9ad7e004f129a9a65d4e62ae2ca |
| name        | swift                            |
| type        | object-store                     |
+-------------+----------------------------------+
```

3. 创建对象存储服务 API 端点：

```
# openstack endpoint create --region RegionOne object-store public http://controller:8080/v1/AUTH_%\(tenant_id\)s
+--------------+-------------------------------------------+
| Field        | Value                                     |
+--------------+-------------------------------------------+
| enabled      | True                                      |
| id           | 73b3765d373c40a78df62637d44a3f1c          |
| interface    | public                                    |
| region       | RegionOne                                 |
| region_id    | RegionOne                                 |
| service_id   | 7a85f9ad7e004f129a9a65d4e62ae2ca          |
| service_name | swift                                     |
| service_type | object-store                              |
| url          | http://controller:8080/v1/AUTH_%(tenant_id)s |
+--------------+-------------------------------------------+
# openstack endpoint create --region RegionOne object-store internal http://controller:8080/v1/AUTH_%\(tenant_id\)s
+--------------+-------------------------------------------+
| Field        | Value                                     |
+--------------+-------------------------------------------+
| enabled      | True                                      |
| id           | 5493d5e29614472a9c0618af6260c2c4          |
```

```
| interface    | internal                                  |
| region       | RegionOne                                 |
| region_id    | RegionOne                                 |
| service_id   | 7a85f9ad7e004f129a9a65d4e62ae2ca          |
| service_name | swift                                     |
| service_type | object-store                              |
| url          | http://controller:8080/v1/AUTH_%(tenant_id)s |
+--------------+-------------------------------------------+
```

```
# openstack endpoint create --region RegionOne object-store admin http://controller:8080/v1
```

```
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | f7d08190b1f14757944149d30f231656 |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 7a85f9ad7e004f129a9a65d4e62ae2ca |
| service_name | swift                            |
| service_type | object-store                     |
| url          | http://controller:8080/v1        |
+--------------+----------------------------------+
```

一 安装和配置组件

1. 安装软件包：

```
# yum install openstack-swift-proxy python-swiftclient python-keystoneclient python-keystonemiddleware memcached -y
```

2. 从对象存储获资源仓库获取代理服务配置文件：

```
# curl -o /etc/swift/proxy-server.conf https://git.openstack.org/cgit/openstack/swift/plain/etc/proxy-server.conf-sample?h=stable/mitaka
```

3. 编辑/etc/swift/proxy-server.conf 文件并完成下列操作：

A. 在[DEFAULT]小节， 配置端口绑定、用户和配置目录：

```
[DEFAULT]
...
bind_port = 8080
```

```
user = swift

swift_dir = /etc/swift
```

B. 在[pipeline:main]小节，移除 tempurl 和 tempauth 模块，并添加 authtoken 和 keystoneauth 模块：

```
[pipeline:main]
pipeline = catch_errors gatekeeper healthcheck proxy-logging cache container_sync bulk ratelimit authtoken keystoneauth container-quotas account-quotas slo dlo versioned_writes proxy-logging proxy-server
```

C. 在[app:proxy-server]小节， 启用自动创建账号功能：

```
[app:proxy-server]

use = egg:swift#proxy

...

account_autocreate = True
```

D. 在[filter:keystoneauth]小节，配置操作角色：

```
[filter:keystoneauth]

use = egg:swift#keystoneauth

...

operator_roles = admin,user
```

E. 在[filter:authtoken]小节， 配置身份服务访问信息：

```
[filter:authtoken]
paste.filter_factory = keystonemiddleware.auth_token:filter_factory
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
```

```
username = swift
password = SWIFT_PASS
delay_auth_decision = True
```

替换 *SWIFT_PASS*为身份服务中 swift 用户的密码。

F. 在[filter:cache]小节，配置 memcached 位置：

```
[filter:cache]
use = egg:swift#memcache
...
memcache_servers = controller:11211
```

✧ 安装和配置存储节点

本节介绍安装和配置对象存储节点，以及操作账户，容器和对象服务。出于简单的目的，推荐使用两个对象存储节点，每个节点添加两块新磁盘,例如/dev/sdb 和/dev/sdc。

— 先决条件

在对象存储节点安装和配置对象存储服务前，必须准备存储设备。

注：下列操作需要在两个对象存储节点上完成。

1. 安装支持工具包：

```
# yum install xfsprogs rsync -y
```

2. 格式化/dev/sdb 和/dev/sdc 设备为 XFS 文件系统：

```
# mkfs.xfs /dev/sdb
# mkfs.xfs /dev/sdc
```

3. 创建挂载点目录：

```
# mkdir -p /srv/node/sdb
# mkdir -p /srv/node/sdc
```

4. 编辑/etc/fstab 文件并添加下列内容:

```
/dev/sdb /srv/node/sdb xfs noatime,nodiratime,nobarrier,logbufs=8 0 2
/dev/sdc /srv/node/sdc xfs noatime,nodiratime,nobarrier,logbufs=8 0 2
```

5. 挂载设备:

```
# mount /srv/node/sdb
```

```
# mount /srv/node/sdc
```

6. 创建并编辑/etc/rsyncd.conf 文件,包含下列内容:

```
uid = swift
gid = swift

log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
address = MANAGEMENT_INTERFACE_IP_ADDRESS

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

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

[object]
max connections = 2
path = /srv/node/
read only = False
lock file = /var/lock/object.lock
```

替换*MANAGEMENT_INTERFACE_IP_ADDRESS*为对象存储节点管理接口的 IP 地址。

7. 启动 rsyncd 服务并设置开机自动运行:

```
# systemctl enable rsyncd.service
```

```
# systemctl start rsyncd.service
```

一 安装和配置组件

注：下列操作需要在两个对象存储节点上完成。

1. 安装软件包：

```
# yum install openstack-swift-account openstack-swift-container openstack-swift-object -y
```

2. 从对象存储资源仓库获取账号，容器和对象服务配置文件：

```
# curl -o /etc/swift/account-server.conf
https://git.openstack.org/cgit/openstack/swift/plain/etc/account-server.conf-sample?h=stable/mitaka
# curl -o /etc/swift/container-server.conf
https://git.openstack.org/cgit/openstack/swift/plain/etc/container-server.conf-sample?h=stable/mitaka
# curl -o /etc/swift/object-server.conf
https://git.openstack.org/cgit/openstack/swift/plain/etc/object-server.conf-sample?h=stable/mitaka
```

3. 编辑/etc/swift/account-server.conf 文件并完成下列操作：

A. 在[DEFAULT]小节， 配置绑定 IP 地址，绑定端口，用户，配置目录和挂在目录：

```
[DEFAULT]
...
bind_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
bind_port = 6002
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = True
```

替换 *MANAGEMENT_INTERFACE_IP_ADDRESS* 为对象存储节点的管理接口 IP 地址。

B. 在[pipeline:main]小节， 启用适当的模块：

```
[pipeline:main]
pipeline = healthcheck recon account-server
```

C. 在[filter:recon]小节，配置 recon(meters)缓存目录：

```
[filter:recon]
use = egg:swift#recon
...
recon_cache_path = /var/cache/swift
```

4. 编辑/etc/swift/container-server.conf 文件并完成下列操作：

A. 在[DEFAULT]小节， 配置绑定 IP 地址，绑定端口， 用户和配置目录和挂载点目录：

```
[DEFAULT]
...
bind_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
bind_port = 6001
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = True
```

替换 *MANAGEMENT_INTERFACE_IP_ADDRESS* 为对象存储节点管理接口的 IP 地址。

B. 在[pipeline:main]小节，启用适当的模块：

```
[pipeline:main]
pipeline = healthcheck recon container-server
```

C. 在[filter:recon]小节，配置 recon(meters)缓存目录：

```
[filter:recon]
use = egg:swift#recon
...
recon_cache_path = /var/cache/swift
```

5. 编辑/etc/swift/object-server.conf 文件并完成下列操作：

A. 在[DEFAULT]小节， 配置绑定 IP 地址，绑定端口， 用户和配置目录和挂载点目录：

```
[DEFAULT]
...
bind_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
bind_port = 6000
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = True
```

替换 *MANAGEMENT_INTERFACE_IP_ADDRESS* 为对象存储节点管理接口的 IP 地址。

B. 在[pipeline:main]小节，启用适当的模块：

```
[pipeline:main]
```

```
pipeline = healthcheck recon object-server
```

C. 在[filter:recon]小节，配置 recon(meters)缓存和锁目录：

```
[filter:recon]
use = egg:swift#recon
...
recon_cache_path = /var/cache/swift
recon_lock_path = /var/lock
```

6. 确保挂载点有正确的权限：

```
# chown -R swift:swift /srv/node
```

7. 创建 recon 目录并确认权限正确：

```
# mkdir -p /var/cache/swift
# chown -R root:swift /var/cache/swift
# chmod -R 775 /var/cache/swift
```

✧ 创建和分发初始环

在启动对象存储服务之前，必须创建并初始化账户、容器和对象的环。

注意：在管理节点完成下列操作。

— 创建账户环

账户服务使用账户环来维护容器列表。

1. 用 cd 命令切换到/etc/swift 目录。
2. 创建基础 account.builder 文件：

```
# swift-ring-builder account.builder create 10 3 1
```

3. 添加每一个对象存储节点到环：

```
# swift-ring-builder account.builder add --region 1 --zone 1 --ip STORAGE_NODE_MANAGEMENT_INTERFACE_IP_ADDRESS --port 6002 --device DEVICE_NAME --weight DEVICE_WEIGHT
```

替换 *STORAGE_NODE_MANAGEMENT_INTERFACE_IP_ADDRESS* 为对象存储节点管理接口的 IP 地址。

替换 *DEVICE_NAME*为存储设备的名字，例如 sdb。

替换 *DEVICE_WEIGHT*为整数权重值，例如 100。

在本案例中，具体命令如下：

```
# swift-ring-builder account.builder add --region 1 --zone 1 --ip 10.0.0.13 --port 6002 --device sdb --weight 100
Device d0r1z1-10.0.0.13:6002R10.0.0.13:6002/sdb_"" with 100.0 weight got id 0
# swift-ring-builder account.builder add --region 1 --zone 1 --ip 10.0.0.13 --port 6002 --device sdc --weight 100
Device d1r1z1-10.0.0.13:6002R10.0.0.13:6002/sdc_"" with 100.0 weight got id 1
# swift-ring-builder account.builder add --region 1 --zone 1 --ip 10.0.0.14 --port 6002 --device sdb --weight 100
Device d2r1z1-10.0.0.14:6002R10.0.0.14:6002/sdb_"" with 100.0 weight got id 2
# swift-ring-builder account.builder add --region 1 --zone 1 --ip 10.0.0.14 --port 6002 --device sdc --weight 100
Device d3r1z1-10.0.0.14:6002R10.0.0.14:6002/sdc_"" with 100.0 weight got id 3
```

4. 确认环内容

```
# swift-ring-builder account.builder
account.builder, build version 4
1024 partitions, 3.000000 replicas, 1 regions, 1 zones, 4 devices, 100.00 balance, 0.00 dispersion
The minimum number of hours before a partition can be reassigned is 1 (0:00:00 remaining)
The overload factor is 0.00% (0.000000)
Ring file account.ring.gz not found, probably it hasn't been written yet
Devices:    id  region  zone      ip address  port  replication ip  replication port      name weight partitions balance flags meta
             0       1     1     10.0.0.13  6002        10.0.0.13              6002       sdb 100.00          0 -100.00
             1       1     1     10.0.0.13  6002        10.0.0.13              6002       sdc 100.00          0 -100.00
             2       1     1     10.0.0.14  6002        10.0.0.14              6002       sdb 100.00          0 -100.00
             3       1     1     10.0.0.14  6002        10.0.0.14              6002       sdc 100.00          0 -100.00
```

5. 重新平衡环：

```
# swift-ring-builder account.builder rebalance
```

```
Reassigned 3072 (300.00%) partitions. Balance is now 0.00.  Dispersion is now 0.00
```

一 创建容器环

容器服务使用容器环维护对象列表，但是它不跟踪对象的位置。

1. 用 cd 命令切换到/etc/swift 目录。

2. 创建基本 container.builder 文件：

```
# swift-ring-builder container.builder create 10 3 1
```

3. 添加每一个对象存储节点到环：

   在本案例中，具体命令如下：

```
# swift-ring-builder container.builder add --region 1 --zone 1 --ip 10.0.0.13
--port 6001 --device sdb --weight 100
Device d0r1z1-10.0.0.13:6001R10.0.0.13:6001/sdb_"" with 100.0 weight got id 0
# swift-ring-builder container.builder add --region 1 --zone 1 --ip 10.0.0.13
--port 6001 --device sdc --weight 100
Device d1r1z1-10.0.0.13:6001R10.0.0.13:6001/sdc_"" with 100.0 weight got id 1
# swift-ring-builder container.builder add --region 1 --zone 1 --ip 10.0.0.14
--port 6001 --device sdb --weight 100
Device d2r1z1-10.0.0.14:6001R10.0.0.14:6001/sdb_"" with 100.0 weight got id 2
# swift-ring-builder container.builder add --region 1 --zone 1 --ip 10.0.0.14
--port 6001 --device sdc --weight 100
Device d3r1z1-10.0.0.14:6001R10.0.0.14:6001/sdc_"" with 100.0 weight got id 3
```

4. 确认环的内容：

```
# swift-ring-builder container.builder
container.builder, build version 4
1024 partitions, 3.000000 replicas, 1 regions, 1 zones, 4 devices, 100.00 balance, 0.00 dispersion
The minimum number of hours before a partition can be reassigned is 1 (0:00:00 remaining)
The overload factor is 0.00% (0.000000)
Ring file container.ring.gz not found, probably it hasn't been written yet
Devices:    id  region  zone      ip address  port  replication ip  replication port      name weight partitions balance
flags meta
             0       1     1       10.0.0.13  6001       10.0.0.13              6001       sdb 100.00          0 -100.00
             1       1     1       10.0.0.13  6001       10.0.0.13              6001       sdc 100.00          0 -100.00
             2       1     1       10.0.0.14  6001       10.0.0.14              6001       sdb 100.00          0 -100.00
             3       1     1       10.0.0.14  6001       10.0.0.14              6001       sdc 100.00          0 -100.00
```

5. 重新平衡环：

```
# swift-ring-builder container.builder rebalance
Reassigned 3072 (300.00%) partitions. Balance is now 0.00.  Dispersion is now 0.00
```

一 创建对象环

对象服务使用对象环维护本地设备物理位置列表。

1. 使用 cd 命令切换到/etc/swift 目录。

2. 创建基本 object.builder 文件：

```
# swift-ring-builder object.builder create 10 3 1
```

3. 添加所有对象存储节点到环：

   在本案例中，具体命令如下：

```
# swift-ring-builder object.builder add --region 1 --zone 1 --ip 10.0.0.13 --
port 6000 --device sdb --weight 100
Device d0r1z1-10.0.0.13:6000R10.0.0.13:6000/sdb_"" with 100.0 weight got id 0
# swift-ring-builder object.builder add --region 1 --zone 1 --ip 10.0.0.13 --
port 6000 --device sdc --weight 100
Device d1r1z1-10.0.0.13:6000R10.0.0.13:6000/sdc_"" with 100.0 weight got id 1
# swift-ring-builder object.builder add --region 1 --zone 1 --ip 10.0.0.14 --
port 6000 --device sdb --weight 100
Device d2r1z1-10.0.0.14:6000R10.0.0.14:6000/sdb_"" with 100.0 weight got id 2
# swift-ring-builder object.builder add --region 1 --zone 1 --ip 10.0.0.14 --
port 6000 --device sdc --weight 100
Device d3r1z1-10.0.0.14:6000R10.0.0.14:6000/sdc_"" with 100.0 weight got id 3
```

4. 确认环的内容：

```
# swift-ring-builder object.builder
object.builder, build version 4
1024 partitions, 3.000000 replicas, 1 regions, 1 zones, 4 devices, 100.00 balance, 0.00 dispersion
The minimum number of hours before a partition can be reassigned is 1 (0:00:00 remaining)
The overload factor is 0.00% (0.000000)
Ring file object.ring.gz not found, probably it hasn't been written yet
Devices:    id  region  zone      ip address  port  replication ip  replication port      name weight partitions balance
flags meta
             0       1     1       10.0.0.13  6000       10.0.0.13              6000       sdb 100.00          0 -100.00
             1       1     1       10.0.0.13  6000       10.0.0.13              6000       sdc 100.00          0 -100.00
             2       1     1       10.0.0.14  6000       10.0.0.14              6000       sdb 100.00          0 -100.00
             3       1     1       10.0.0.14  6000       10.0.0.14              6000       sdc 100.00          0 -100.00
```

5. 重新平衡环：

```
# swift-ring-builder object.builder rebalance
Reassigned 3072 (300.00%) partitions. Balance is now 0.00.  Dispersion is now 0.00
```

分发环配置文件

复制 account.ring.gz, container.ring.gz,和 object.ring.gz 文件到每一个对象存储节点和运行代理服务的节点（管理节点）的/etc/swift 目录。

完成安装

1. 在管理节点，从对象存储资源仓库获取/etc/swift/swift.conf 文件。

```
# curl -o /etc/swift/swift.conf
https://git.openstack.org/cgit/openstack/swift/plain/etc/swift.conf-sample?h=stable/mitaka
```

2. 编辑/etc/swift/swift.conf 文件，并完成下列操作：

   A. 在[swift-hash]小节，为你的环境配置 hash path prefix 和 suffix

   ```
   [swift-hash]
   ...
   swift_hash_path_suffix = HASH_PATH_SUFFIX
   swift_hash_path_prefix = HASH_PATH_PREFIX
   ```

   替换 *HASH_PATH_SUFFIX* 和 *HASH_PATH_PREFIX* 为合适的内容，不一样就行。

   B. 在[storage-policy:0]小节， 配置默认存储策略：

```
[storage-policy:0]
...
name = Policy-0
default = yes
```

3. 复制 swift.conf 文件到每一个对象存储节点和运行代理服务的节点的/etc/swift 目录中。
4. 在所有节点，确保配置目录有正确的权限：

```
# chown -R root:swift /etc/swift
```

5. 在管理节点和任何其他运行代理服务的节点，启用对象代理服务及其依赖服务，并设置开机自动运行：

```
# systemctl enable openstack-swift-proxy.service memcached.service
# systemctl start openstack-swift-proxy.service memcached.service
```

6. 在对象存储节点，启动对象存储服务并设置他们开机自动运行：

```
# systemctl enable openstack-swift-account.service openstack-swift-account-auditor.service openstack-swift-account-reaper.service openstack-swift-account-replicator.service
# systemctl start openstack-swift-account.service openstack-swift-account-auditor.service openstack-swift-account-reaper.service openstack-swift-account-replicator.service
# systemctl enable openstack-swift-container.service openstack-swift-container-auditor.service openstack-swift-container-replicator.service openstack-swift-container-updater.service
# systemctl start openstack-swift-container.service openstack-swift-container-auditor.service openstack-swift-container-replicator.service openstack-swift-container-updater.service
# systemctl enable openstack-swift-object.service openstack-swift-object-auditor.service openstack-swift-object-replicator.service openstack-swift-object-updater.service
# systemctl start openstack-swift-object.service openstack-swift-object-auditor.service openstack-swift-object-replicator.service openstack-swift-object-updater.service
```

一 确认操作

在管理节点完成下列操作。

1. 如果对象存储节点未关闭 SELinux，则需要执行下列命令：

```
# chcon -R system_u:object_r:swift_data_t:s0 /srv/node
```

2. 执行 demo 凭据脚本，以便以 demo 身份执行后续命令：

```
# . demo-openrc
```

3. 显示服务状态

```
# swift stat
```

```
           Account: AUTH_8695acb7e10e4c44a25da3076b2671b4
        Containers: 0
           Objects: 0
             Bytes: 0
X-Put-Timestamp: 1469176355.03990
       X-Timestamp: 1469176355.03990
        X-Trans-Id: tx07289d3264f6414ca3f08-005791da21
      Content-Type: text/plain; charset=utf-8
```

4. 创建 container1 容器

```
# openstack container create container1
```

```
+---------------------------------------+-----------+---------------------------------------+
| account                               | container | x-trans-id                            |
+---------------------------------------+-----------+---------------------------------------+
| AUTH_8695acb7e10e4c44a25da3076b26 | container1 | tx3db71ce4a17946879da50-005791da62 |
| 71b4                                  |           |                                       |
+---------------------------------------+-----------+---------------------------------------+
```

5. 上传一个测试文件到 containiner1 容器：

```
# openstack object create container1 cirros-0.3.4-x86_64-disk.img
```

```
+------------------------------+-----------+----------------------------------+
| object                       | container | etag                             |
+------------------------------+-----------+----------------------------------+
| cirros-0.3.4-x86_64-disk.img | container1 | ee1eca47dc88f4879d8a229cc70a07c6 |
+------------------------------+-----------+----------------------------------+
```

*cirros-0.3.4-x86_64-disk.img*为当前目录下文件，可替换为其他

文件。

6. 列出 containerl 容器中的文件：

```
# openstack object list containerl
+----------------------------+
| Name                       |
+----------------------------+
| cirros-0.3.4-x86_64-disk.img |
+----------------------------+
```

7. 从 containerl 容器中下载测试文件：

```
# openstack object save containerl cirros-0.3.4-x86_64-disk.img
```

可替换 *cirros-0.3.4-x86_64-disk.img* 为你容器里有的其他文件。

十二、启动实例

一 创建云主机类型

创建一个有 1 核 CPU，64M 内存，1G 硬盘空间， 名为 m1.nano 的云主机类型：

```
# . admin-openrc
```

```
# openstack flavor create --id 0 --vcpus 1 --ram 64 --disk 1 m1.nano
+----------------------------+---------+
| Field                      | Value   |
+----------------------------+---------+
| OS-FLV-DISABLED:disabled   | False   |
| OS-FLV-EXT-DATA:ephemeral  | 0       |
| disk                       | 1       |
| id                         | 0       |
| name                       | m1.nano |
| os-flavor-access:is_public | True    |
| ram                        | 64      |
| rxtx_factor                | 1.0     |
```

```
| swap                        |          |
| vcpus                       | 1        |
+-----------------------------+----------+
```

一 创建一个密钥对

1. 执行 demo 凭据脚本，以便以 demo 身份执行后续命令:

```
# . demo-openrc
```

2. 生成并添加密钥对

```
# ssh-keygen -q -N ""
```

```
# openstack keypair create --public-key ~/.ssh/id_rsa.pub mykey
```

```
+-------------+-------------------------------------------------+
| Field       | Value                                           |
+-------------+-------------------------------------------------+
| fingerprint | a5:8d:b6:08:f8:04:39:64:81:6b:78:26:af:45:64:5f |
| name        | mykey                                           |
| user_id     | ac3905c764824555aa187363b3c6f7bb                |
+-------------+-------------------------------------------------+
```

3. 确认添加的秘钥对

```
# openstack keypair list
```

```
+-------+-------------------------------------------------+
| Name  | Fingerprint                                     |
+-------+-------------------------------------------------+
| mykey | a5:8d:b6:08:f8:04:39:64:81:6b:78:26:af:45:64:5f |
+-------+-------------------------------------------------+
```

添加安全组规则

默认情况下，有个名为 default 安全组。他拒绝其他 ip 远程访问云主机。这里建议允许 ICMP(ping)和 ssh。

1. 添加规则到 default 安全组

A. 允许 ICMP(ping)

```
# openstack security group rule create --proto icmp default
```

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

```
| Field                 | Value                                |
+-----------------------+--------------------------------------+
| id                    | 93ad4a7f-45ce-48d8-b7a4-d005a8768f9c |
| ip_protocol           | icmp                                 |
| ip_range              | 0.0.0.0/0                            |
| parent_group_id       | 4dec2182-e616-4414-80c6-54f84799d628 |
| port_range            |                                      |
| remote_security_group |                                      |
+-----------------------+--------------------------------------+
```

B. 允许 SSH 访问

```
# openstack security group rule create --proto tcp --dst-port 22 default
+-----------------------+--------------------------------------+
| Field                 | Value                                |
+-----------------------+--------------------------------------+
| id                    | 7e9faefd-7186-46fb-97f2-5edbc5f8d61a |
| ip_protocol           | tcp                                  |
| ip_range              | 0.0.0.0/0                            |
| parent_group_id       | 4dec2182-e616-4414-80c6-54f84799d628 |
| port_range            | 22:22                                |
| remote_security_group |                                      |
+-----------------------+--------------------------------------+
```

启动一个实例

确定实例选项

1. 执行 demo 凭据脚本，以便以 demo 身份执行后续命令：

```
# . demo-openrc
```

2. 列出可用的云主机类型

```
# openstack flavor list
```

```
+----+-----------+-------+------+-----------+-------+-----------+
| ID | Name      |   RAM | Disk | Ephemeral | VCPUs | Is Public |
+----+-----------+-------+------+-----------+-------+-----------+
| 0  | m1.nano   |    64 |    1 |         0 |     1 | True      |
| 1  | m1.tiny   |   512 |    1 |         0 |     1 | True      |
| 2  | m1.small  |  2048 |   20 |         0 |     1 | True      |
| 3  | m1.medium |  4096 |   40 |         0 |     2 | True      |
| 4  | m1.large  |  8192 |   80 |         0 |     4 | True      |
| 5  | m1.xlarge | 16384 |  160 |         0 |     8 | True      |
+----+-----------+-------+------+-----------+-------+-----------+
```

本实例选择 m1.nano 类型。

3. 列出可用的镜像：

```
# openstack image list
```

```
+--------------------------------------+--------+--------+
| ID                                   | Name   | Status |
+--------------------------------------+--------+--------+
| 25c308d5-9056-4b6d-ae7c-5e83dde5be39 | cirros | active |
+--------------------------------------+--------+--------+
```

本实例选择 cirros 镜像

4. 列出可用的网络：

```
# openstack network list
```

```
+--------------------------------------+-------------+--------------------------------------+
| ID                                   | Name        | Subnets                              |
+--------------------------------------+-------------+--------------------------------------+
| 19fd8bb4-c17a-4af4-a16e-fd85e9c0e05b | provider    | 326b4690-51c3-485a-819b-a2c660ae3eba |
| 7ba9e684-b133-4c27-a63e-4fc7a5e39034 | selfservice | bc052f13-b07e-4954-bb19-c9b36aaea142 |
+--------------------------------------+-------------+--------------------------------------+
```

本实例选择 selfservice 网络。

5. 列出可用的安全组：

```
# openstack security group list
```

```
+--------------------------------------+---------+------------------------+----------------------------------+
| ID                                   | Name    | Description            | Project                          |
+--------------------------------------+---------+------------------------+----------------------------------+
| 4dec2182-e616-4414-80c6-54f84799d628 | default | Default security group | 8695acb7e10e4c44a25da3076b2671b4 |
+--------------------------------------+---------+------------------------+----------------------------------+
```

本实例使用 default 安全组

6. 启动实例

替换 *SELFSERVICE_NET_ID*为 selfservice 网络的 ID。

```
# openstack server create --flavor m1.tiny --image cirros
```

```
--nic net-id=SELFSERVICE_NET_ID --security-group default -
-key-name mykey selfservice-instance
```

Field	Value
OS-DCF:diskConfig	MANUAL
OS-EXT-AZ:availability_zone	
OS-EXT-STS:power_state	0
OS-EXT-STS:task_state	scheduling
OS-EXT-STS:vm_state	building
OS-SRV-USG:launched_at	None
OS-SRV-USG:terminated_at	None
accessIPv4	
accessIPv6	
addresses	
adminPass	i7DHJjpShuc4
config_drive	
created	2016-07-22T09:08:43Z
flavor	m1.nano (0)
hostId	
id	3949765c-a1b4-4f91-8f6c-79c297a67559
image	cirros (25c308d5-9056-4b6d-ae7c-5e83dde5be39)
key_name	mykey
name	selfservice-instance
os-extended-volumes:volumes_attached	[]
progress	0
project_id	8695acb7e10e4c44a25da3076b2671b4
properties	
security_groups	[{u'name': u'default'}]
status	BUILD
updated	2016-07-22T09:09:05Z
user_id	ac3905c764824555aa187363b3c6f7bb

7. 查看你的实例状态：

```
# openstack server list
```

ID	Name	Status	Networks
3949765c-a1b4-4f91	selfservice-instance	ACTIVE	selfservice=192.168.111.

```
| -8f6c-79c297a67559      |                      |        | 3                       |
+-------------------------+----------------------+--------+-------------------------+
```

一 使用虚拟终端访问实例

1. 获取你的实例的 vnc 会话 URL，并通过浏览器访问：

```
# openstack console url show selfservice-instance
```

```
+-------+-----------------------------------------------------------------+
| Field | Value                                                           |
+-------+-----------------------------------------------------------------+
| type  | novnc                                                           |
| url   | http://controller:6080/vnc_auto.html?token=804f2d7c-6252-442c-b822-aaafdb2aece3 |
+-------+-----------------------------------------------------------------+
```

2. 登录实例系统后，确认可以访问自定义网络的网关：

```
$ ping 192.168.111.1 -c 4
```

远程访问实例

1. 在 provider 虚拟网络中创建浮动 IP：

```
# openstack ip floating create provider
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| fixed_ip    | None                                 |
| id          | 2d5bc301-04b6-4aaf-9c01-0af73492a003 |
| instance_id | None                                 |
| ip          | 172.16.100.103                       |
| pool        | provider                             |
+-------------+--------------------------------------+
```

2. 分配浮动 IP 到实例

```
# openstack ip floating add 172.16.100.103 selfservice-instance
```

替换 *172.16.100.103*为你刚才创建的浮动 IP

3. 查看你的实例的浮动 IP

```
# openstack server list
```

```
+----------------------+---------------------+---------+-----------------------+
| ID                   | Name                | Status  | Networks              |
+----------------------+---------------------+---------+-----------------------+
| 3949765c-a1b4-4f91   | selfservice-instance | SHUTOFF | selfservice=192.168.111 |
| -8f6c-79c297a67559   |                     |         | .3, 172.16.100.103    |
+----------------------+---------------------+---------+-----------------------+
```

4. 从管理节点或能够连接 provider 物理网络的其他机器确认实例的浮动 IP 可以通信：

```
# ping  -c  172.16.100.103
```

5. 在管理节点或能够连接 provider 物理网络的其他机器使用 ssh 远程访问实例：

```
# ssh cirros@172.16.100.103
```