

OpenStack-Newton部署指南

官方安装手册：http://docs.openstack.org/newton/zh_CN/install-guide-rdo/neutron.html

环境准备

以两台实验主机为例，node1作为控制节点，node2作为计算机节点。在控制节点上部署主要的项目组件，和认证管理组件，计算节点上部署必要的计算服务，用于创建虚拟机。

修改主机名同步系统时间

Openstack对环境一致性和时间一致性的要求比较高，在部署之前做好合理的规划可以方便后期维护。

1. 修改主机名：

```
hostname node1
echo "node1" >/etc/hostname
echo "172.16.10.31 node1" >>/etc/hosts
echo "172.16.10.32 node2" >>/etc/hosts
```

2. 在node1和node2上执行如下操作同步时间：

```
[root@node1 ~]# yum install ntpdate -y
[root@node1 ~]# ntpdate time1.aliyun.com
```

安装newton仓库与相关软件包

1. 在所有节点安装openstack的官方仓库:

```
# yum install centos-release-openstack-newton -y
```

2. 所有节点安装openstack客户端:

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

3. 安装 `openstack-selinux` 实现对包的自动管理:

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

在控制节点配置消息队列与Memcached

1. 配置消息队列:

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

2. 启动消息队列服务并将其配置为随系统启动:

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

3. 添加 `openstack` 用户,设置密码为openstack:

```
[root@node1 ~]# rabbitmqctl add_user openstack openstack
Creating user "openstack" ...
```

4. 给 `openstack` 用户设置读写权限:

```
[root@node1 ~]# rabbitmqctl set_permissions openstack ".*" ".*" ".*"
Setting permissions for user "openstack" in vhost "/" ...
```

5. 安装Memcached并启动服务（此处使用Memcached主要用于控制节点缓存身份认证的令牌）：

```
# yum install memcached python-memcached -y
# systemctl enable memcached.service
# systemctl start memcached.service
```

6. 打开RabbitMQ的监控插件，可以直接通过网页登录查看：

```
# rabbitmq-plugins enable rabbitmq_management
```

在控制节点安装OpenStack服务组件

1. 安装keystone相关组件：

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

2. 安装镜像服务组件：

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

3. 安装计算服务组件：

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

4. 安装网络服务组件:

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

在控制节点（node1）上部署数据库

1. 在控制节点安装部署数据库:

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

2. 创建并编辑 /etc/my.cnf.d/openstack.cnf, 并加入如下配置:

```
[mysqld]  
bind-address = 172.16.10.31  
  
default-storage-engine = innodb  
innodb_file_per_table  
max_connections = 4096  
collation-server = utf8_general_ci  
character-set-server = utf8
```

3. 启动数据库，并配置开机自启动：

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

4. 初始化数据库：

```
# mysql_secure_installation
```

5. 创建openstack服务授权表：

```
> create database keystone;
> grant all on keystone.* to 'keystone'@'localhost' identified by 'keystone';
> grant all on keystone.* to 'keystone'@'%' identified by 'keystone';
> create database glance;
> grant all on glance.* to 'glance'@'localhost' identified by 'glance';
> grant all on glance.* to 'glance'@'%' identified by 'glance';
> create database nova;
> grant all on nova.* to 'nova'@'localhost' identified by 'nova';
> grant all on nova.* to 'nova'@'%' identified by 'nova';
> create database nova_api;
> grant all on nova_api.* to 'nova'@'localhost' identified by 'nova';
> grant all on nova_api.* to 'nova'@'%' identified by 'nova';
> create database neutron;
> grant all on neutron.* to 'neutron'@'localhost' identified by 'neutron';
> grant all on neutron.* to 'neutron'@'%' identified by 'neutron';
```

查看创建的数据库是否正常：

```
MariaDB [(none)]> show databases;
```

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

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

```
| glance |
```

```
| information_schema |
```

```
| keystone |
```

```
| mysql |
```

```
| neutron |
```

```
| nova |
```

```
| nova_api |
```

```
| performance_schema |
```

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

```
8 rows in set (0.00 sec)
```

在计算节点安装计算服务和网络服务组件

1. 安装计算服务:

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

2. 安装网络服务:


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

检查服务环境

配置好基础环境之后，查看一下RabbitMQ和各项服务端口是否启动。

```
[root@node1 ~]# netstat -lnpt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 172.16.10.31:3306       0.0.0.0:*               LISTEN      12995/mysqld
tcp        0      0 127.0.0.1:11211        0.0.0.0:*               LISTEN      14017/memcached
tcp        0      0 0.0.0.0:4369           0.0.0.0:*               LISTEN      13332/epmd
tcp        0      0 0.0.0.0:22             0.0.0.0:*               LISTEN      936/sshd
tcp        0      0 0.0.0.0:15672          0.0.0.0:*               LISTEN      13352/beam
tcp        0      0 127.0.0.1:25           0.0.0.0:*               LISTEN      2077/master
tcp        0      0 0.0.0.0:25672          0.0.0.0:*               LISTEN      13352/beam
tcp6       0      0 :::11211               :::*                    LISTEN      14017/memcached
tcp6       0      0 :::4369                 :::*                    LISTEN      13332/epmd
tcp6       0      0 :::22                   :::*                    LISTEN      936/sshd
tcp6       0      0 :::1:25                 :::*                    LISTEN      2077/master
tcp6       0      0 :::5672                 :::*                    LISTEN      13352/beam
```

测试RabbitMQ监控服务可通过登录http://node1_ip:15672/，使用默认账号guest,密码guest进行查看。此处RabbitMQ的服务端口是5672。15672为RabbitMQ的web登录端口。

认证服务Keystone

Keystone主要提供用户认证和服务目录的功能。openstack的服务授权都需要在keystone上完成,keystone通过给授权的用户提供一个具有时间有效期的token,在用户token过期之后需要重新授权。服务目录则包含了所有服务项和与之相关的API端点。

- 用户认证: User, Project,Token, Role.
这里的Role就类似一个具有相同权限的用户组, Keystone通过这些机制来进行认证授权操作。
- 服务目录: service, endpoint. service 服务, 如 Nova, Glance, Swift. 一个服务可以确认当前用户是否具有访问资源的权限。
endpoint其实是一个url,每个url都对应一个服务的实例的访问地址, 并且具有public、private和admin这三种权限。public url可以被全局访问, private url只能被局域网访问, admin url被从常规的访问中分离。

Keystone组件配置

[参考官方文档点击此处](#)

1. 编辑文件 `/etc/keystone/keystone.conf` 并完成如下动作:
 - 配置数据库访问, 指定库、用户名、密码:

```
[database]
connection = mysql+pymysql://keystone:keystone@172.16.10.31/keystone
```

- 在 `[token]` 部分, 配置Fernet UUID令牌的提供者:

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

2. 初始化身份认证服务的数据库:

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

3. 初始化Fernet key:

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

4. 引导验证服务, 设置 Admin 账户密码为 admin:

```
# keystone-manage bootstrap --bootstrap-password admin \  
--bootstrap-admin-url http://172.16.10.31:35357/v3/ \  
--bootstrap-internal-url http://172.16.10.31:35357/v3/ \  
--bootstrap-public-url http://172.16.10.31:5000/v3/ \  
--bootstrap-region-id RegionOne
```

配置Apache HTTP服务

1. 编辑 `/etc/httpd/conf/httpd.conf` 文件，配置 `ServerName` 选项为控制节点：

```
ServerName 172.16.10.31
```

2. 创建一个链接到 `/usr/share/keystone/wsgi-keystone.conf` 文件

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

3. 启动Apache HTTP,出现5000端口和35357端口说明配置成功:

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

4. 配置admin账户：

```
$ export OS_USERNAME=admin  
$ export OS_PASSWORD=admin  
$ export OS_PROJECT_NAME=admin  
$ export OS_USER_DOMAIN_NAME=default  
$ export OS_PROJECT_DOMAIN_NAME=default  
$ export OS_AUTH_URL=http://172.16.10.31:35357/v3  
$ export OS_IDENTITY_API_VERSION=3
```

创建域、项目、用户和角色

1. 创建 **service** 项目:

```
openstack project create --domain default \
--description "Service Project" service
+-----+-----+
| Field      | Value                                     |
+-----+-----+
| description | Service Project                         |
| domain_id   | default                                 |
| enabled     | True                                    |
| id          | 075fc30ab7d94b46870f08e88ea15bb6      |
| is_domain   | False                                  |
| name        | service                                |
| parent_id   | default                                 |
+-----+-----+
```

2. 创建一个常规无特权的项目和用户，用于执行非管理任务，以 **demo** 为例：

- 创建 **demo** 项目：

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

+-----+	+-----+	+-----+
Field	Value	
+-----+	+-----+	+-----+
description	Demo Project	
domain_id	default	
enabled	True	
id	b6549194db2342e9879eceed8de30cb9	
is_domain	False	
name	demo	
parent_id	default	
+-----+	+-----+	+-----+

- 创建 demo 用户，密码为demo:

```
openstack user create --domain default \  
--password-prompt demo  
User Password:  
Repeat User Password:
```

Field	Value
domain_id	default
enabled	True
id	291cdb4cd2314bf1954a64aa6bf688a0
name	demo
password_expires_at	None

- 创建 user 角色:

```
openstack role create user
```

Field	Value
domain_id	None
id	510d4fdaa6414c89abc6666c0ca3029d
name	user

- 添加 user 角色到 demo 项目和用户：

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

验证服务是否正常

1. 因为安全性的原因，关闭临时认证令牌机制：（此处如果处理不当会造成demo用户无法获取token的情况，此步可不操作）编辑 `/etc/keystone/keystone-paste.ini` 文件，从 `[pipeline:public_api]`，`[pipeline:admin_api]` 和 `[pipeline:api_v3]` 部分删除 `admin_token_auth`。
2. 撤销临时环境变量 `OS_AUTH_URL` 和 `OS_PASSWORD`：

```
# unset OS_AUTH_URL OS_PASSWORD
```

3. 使用 admin 用户，请求认证令牌，输入 admin 用户的密码：

```
openstack --os-auth-url http://172.16.10.31:35357/v3 \
--os-project-domain-name default --os-user-domain-name default \
--os-project-name admin --os-username admin token issue
Password:
```

Field	Value
expires	2017-01-06 08:59:46+00:00
id	gAAAAABYb05yewK_zsdeS5oNlTRdhU7a2DhH9BK0GBE08seYyx4Ydj1ES3acwfu2Tiv7E4E4YEMU2eebWEi0 zwE00ICulCY2nEYe38b-
	oATdTCMz28z0nG5z0PN0ZdDNbuMapgXisnwYVbrPrch60MlobUbvnjmMXLGx0I2P51vRc9sP9iN3jHw
project_id	c34d22ac34cb4c62a025295cd3d6540c
user_id	a89aec9a0224445a96025fe8ccf6c857

4. 作为 **demo** 用户, 请求认证令牌:

```
openstack --os-auth-url http://172.16.10.31:5000/v3 \
--os-project-domain-name default --os-user-domain-name default \
--os-project-name demo --os-username demo token issue
```

Password:

Field	Value
expires	2017-01-06 09:03:25+00:00
id	gAAAAABYb090ZJ_T1_9jFQWiuqQiMAstUef8zHRsI8APC6eZtNIMhSAfF2eozsqkgpYIx6ux5qNKWAIQu60a
	RWAniOT_dKVDyqPa7totIk9K0dt3DtVUp7JBM6u3hT4xc1E-
	rsnfNu2CTmDjGMkf_jmB1W6p8vXijfB3iRXtL5utlrXQRysalq0
project_id	b6549194db2342e9879eceed8de30cb9
user_id	291cdb4cd2314bf1954a64aa6bf688a0

“

提示: keystone中admin用户使用的35357端口, demo用户使用5000端口, 出现上面结果说明验证成功。

创建 OpenStack 客户端环境脚本

在客户端进行请求验证时, 需要指定一系列参数, 这些参数可以通过创建一个本地的环境变量, 在需要请求keystone时, 加载这些环境变量, 从而减少输入内容。创建 admin 和 demo 项目和用户创建客户端环境变量脚本。接下来的部分会引用这些脚本, 为客户端操作加载合

适的的凭证。

1. 编辑 `admin-openstack` 文件并且增加以下内容

```
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
export OS_AUTH_URL=http://172.16.10.31:35357/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

2. 编辑文件 `demo-openstack` 并添加如下内容:

```
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
export OS_AUTH_URL=http://172.16.10.31:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

3. 使用脚本, 尝试获取token:

```
[root@node1 ~]# . admin-openstack
[root@node1 ~]# openstack token issue
```

```
+-----+-----+
| Field      | Value                                                                 |
+-----+-----+
| expires    | 2017-01-06 09:33:30+00:00                                           |
| id         | gAAAAABYb1Za-97A69q8NZpltnkAa9iBfEzMehTrrDADVZZApUD6ynpTF2kvCl0orw0ca |
|            | --6Q0ngLv0Ha40Ho1TQFa68us59qADKGofzuVK-                             |
|            | ampSHb2fzPmBSSFx9i7SUJ6tESKPT3ppofP9L3xPxfp13jauBL1TdjXQ0zL7svy2xjVB39sDS_o |
| project_id | c34d22ac34cb4c62a025295cd3d6540c                                   |
| user_id    | a89aec9a0224445a96025fe8ccf6c857                                   |
+-----+-----+
```

镜像服务 Glance

[查看官方手册](#)

Glance由Glance-api和Glance-Registry以及image Storage三个组件组成。

- Glance-api: 接受云系统镜像的创建、删除和读取请求。通过接收REST API的请求，调用其他模块来完成镜像的查找，获取、上传、删除等操作。默认的监听端口为9292.
- Glance-Registry: 云镜像的注册服务。与mysql进行数据交互，用于存储和获取镜像的元数据。Glance数据库中有两张表，一张是image表，另一张是image property表。image表保存了镜像格式、大小等信息，image property表则主要保存镜像的定制化信息。glance-registry监听的端口为9191.
- Image storage: 是一个存储的接口层，严格来说它并不是属于glance,只是给glance提供调用的一个接口。通过这个接口，glance可

以获取镜像。image storage支持的存储有Amazon的S3、Openstack本身的Swift,还有如 ceph,sheepdog,GlusterFS等分布式存储, image storage是镜像保存和获取的接口, 由于仅仅是一个接口层, 具体的实现需要外部存储的支持。

在Keystone上注册Glance服务

1. 创建服务证书:

- 加载admin环境变量,创建glance用户:

```
# . admin-openstack
# openstack user create --domain default --password-prompt glance

User Password:
Repeat User Password:
+-----+-----+
| Field           | Value                               |
+-----+-----+
| domain_id       | default                             |
| enabled         | True                                |
| id              | f9a9e04fdcc544fcb77209461bd8eba7 |
| name            | glance                              |
| password_expires_at | None                                |
+-----+-----+
```

- 添加 admin 角色到 glance 用户和 service 项目上:

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

- 创建 glance 服务实体:

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

Field	Value
description	OpenStack Image
enabled	True
id	32f40933b6c14d8d860fe7ff8e1d9b6b
name	glance
type	image

2. 创建镜像服务的 API 端点,分别指定为public,internal,admin:

```
openstack endpoint create --region RegionOne \  
image public http://172.16.10.31:9292
```

Field	Value
enabled	True
id	f38f74ece80b4a4c988fdaf4bf6ee078
interface	public
region	RegionOne
region_id	RegionOne
service_id	32f40933b6c14d8d860fe7ff8e1d9b6b
service_name	glance
service_type	image
url	http://172.16.10.31:9292


```
openstack endpoint create --region RegionOne \  
image internal http://172.16.10.31:9292
```

Field	Value
enabled	True
id	21804b6e807e41e0b44a531a2cf3c3d9
interface	internal
region	RegionOne
region_id	RegionOne
service_id	32f40933b6c14d8d860fe7ff8e1d9b6b
service_name	glance
service_type	image
url	http://172.16.10.31:9292

```
openstack endpoint create --region RegionOne \
image admin http://172.16.10.31:9292
```

Field	Value
enabled	True
id	2adbe5c08f9a4392a39cca54b20e7fbb
interface	admin
region	RegionOne
region_id	RegionOne
service_id	32f40933b6c14d8d860fe7ff8e1d9b6b
service_name	glance
service_type	image
url	http://172.16.10.31:9292

3. 编辑文件 `/etc/glance/glance-api.conf` 并完成如下动作:

- 在 [database] 部分, 配置数据库访问:

```
[database]
connection = mysql+pymysql://glance:glance@172.16.10.31/glance
```

- 在 [keystone_authtoken] 和 [paste_deploy] 部分, 配置认证服务访问:

```
[keystone_auth_token]
auth_uri = http://172.16.10.31:5000
auth_url = http://172.16.10.31:35357
memcached_servers = 172.16.10.31:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = glance
password = glance
```

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

- 在 [glance_store] 部分，配置本地文件系统存储和镜像文件位置：

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

4. 编辑文件 `/etc/glance/glance-registry.conf` 并完成如下动作：

- 在 [database] 部分，配置数据库访问：

```
[database]
connection = mysql+pymysql://glance:glance@172.16.10.31/glance
```

- 在 [keystone_authtoken] 和 [paste_deploy] 部分，配置认证服务访问：

```
[keystone_authtoken]
auth_uri = http://172.16.10.31:5000
auth_url = http://172.16.10.31:35357
memcached_servers = 172.16.10.31:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = glance
password = glance
```

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

5. 写入镜像服务数据库：

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

6. 启动镜像服务并将其配置为随机启动（启动后会占用9292和9191端口）：

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

对执行结果进行验证

1. . 加载admin的环境变量，下载测试镜像：

```
source admin-openstack  
wget http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img
```

2. 创建一个 **cirros** 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	2017-01-06T11:27:52Z
disk_format	qcow2
file	/v2/images/b14c429c-6c18-4407-acba-565ddd829ec4/file
id	b14c429c-6c18-4407-acba-565ddd829ec4
min_disk	0
min_ram	0
name	cirros
owner	c34d22ac34cb4c62a025295cd3d6540c
protected	False
schema	/v2/schemas/image
size	13287936
status	active
tags	
updated_at	2017-01-06T11:27:53Z
virtual_size	None
visibility	public
+-----+	

3. 查看镜像属性:

```
[root@node1 ~]# openstack image list
```

ID	Name	Status
b14c429c-6c18-4407-acba-565ddd829ec4	cirros	active

计算服务 Nova

[参考官方文档](#)

在openstack的创建中，我们一般将Nova的计算节点组件放在需要创建虚拟机的主机上，而除了计算节点之外的其他Nova组件安装在控制节点上，计算节点只负责创建虚拟机。

Nova的服务组件：

- API:负责接收和响应外部请求。API接收的请求将会放到消息队列（rabbitMQ）中。是外部访问nova的唯一途径。
- Cert:负责身份认证EC2。
- Scheduler:用于云主机调度。决策虚拟机创建在哪个主机（计算节点）上
- Conductor: 计算节点访问数据库的中间件。
- Consoleauth:用于控制台的授权验证
- Novncproxy: VNC代理

在Keystone上注册nova服务

1. 创建服务证书

- 加载admin环境变量, 创建 nova 用户:

```
# source admin-openstack
# openstack user create --domain default \
  --password-prompt nova
```

User Password:

Repeat User Password:

+-----+-----+		
Field	Value	
+-----+-----+		
domain_id	default	
enabled	True	
id	520f61410ece4380a30b366513726f01	
name	nova	
password_expires_at	None	
+-----+-----+		

- 给 nova 用户添加 admin 角色:

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

- 创建 nova 服务实体:


```
# openstack service create --name nova \  
--description "OpenStack Compute" compute
```

Field	Value
description	OpenStack Compute
enabled	True
id	6940f1e002e14791809b3120edf51360
name	nova
type	compute

2. 创建 Compute 服务 API 端点：

```
openstack endpoint create --region RegionOne \  
compute public http://172.16.10.31:8774/v2.1/%(tenant_id)s
```

Field	Value
enabled	True
id	b5e091a8806244f5aae23776c1a365d3
interface	public
region	RegionOne
region_id	RegionOne
service_id	6940f1e002e14791809b3120edf51360
service_name	nova

```
| service_type | compute |
| url          | http://172.16.10.31:8774/v2.1/%(tenant_id)s |
+-----+
```

```
openstack endpoint create --region RegionOne \
compute internal http://172.16.10.31:8774/v2.1/%(tenant_id)s
```

```
+-----+
| Field      | Value |
+-----+
| enabled    | True  |
| id         | d03f880db1734abbbdc4e8ca07987ac8 |
| interface  | internal |
| region     | RegionOne |
| region_id  | RegionOne |
| service_id | 6940f1e002e14791809b3120edf51360 |
| service_name | nova |
| service_type | compute |
| url        | http://172.16.10.31:8774/v2.1/%(tenant_id)s |
+-----+
```

```
openstack endpoint create --region RegionOne \
compute admin http://172.16.10.31:8774/v2.1/%(tenant_id)s
```

```
+-----+
| Field      | Value |
+-----+
| enabled    | True  |
| id         | 51352a4e60d64daea5507fdde3911e11 |
| interface  | admin |
+-----+
```

```
| region      | RegionOne |
| region_id   | RegionOne |
| service_id  | 6940f1e002e14791809b3120edf51360 |
| service_name | nova      |
| service_type | compute   |
| url         | http://172.16.10.31:8774/v2.1/%(tenant_id)s |
+-----+-----+
```

计算节点配置Nova组件

1. 编辑 `/etc/nova/nova.conf` 文件并完成下面的操作：
 - 在 `[DEFAULT]` 部分，只启用计算和元数据API:

```
[DEFAULT]
```

```
enabled_apis = osapi_compute,metadata
```

- 在 `[api_database]` 和 `[database]` 部分，配置数据库的连接:

```
[api_database]
connection=mysql+pymysql://nova:nova@172.16.10.31/nova_api

[database]
connection=mysql+pymysql://nova:nova@172.16.10.31/nova
```

- 在 [DEFAULT] 部分，配置 RabbitMQ 消息队列访问权限：

```
[DEFAULT]
transport_url=rabbit://openstack:openstack@172.16.10.31
```

- 在 [DEFAULT] 和 [keystone_authtoken] 部分，配置认证服务访问：

```
[DEFAULT]
```

```
auth_strategy = keystone
```

```
[keystone_authtoken]
```

```
auth_uri = http://172.16.10.31:5000
```

```
auth_url = http://172.16.10.31:35357
```

```
memcached_servers = 172.16.10.31:11211
```

```
auth_type = password
```

```
project_domain_name = Default
```

```
user_domain_name = Default
```

```
project_name = service
```

```
username = nova
```

```
password = nova
```

- 在 `[DEFAULT]` 部分, 启用网络服务支持.(默认情况下, 计算服务使用内置的防火墙服务。由于网络服务包含了防火墙服务, 必须使用 `nova.virt.firewall.NoopFirewallDriver` 防火墙服务来禁用掉计算服务内置的防火墙服务):

```
[DEFAULT]
```

```
use_neutron = True
```

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

- 在 [vnc] 部分，配置VNC代理使用控制节点的管理接口IP地址：

```
[vnc]
```

```
vncserver_listen = 172.16.10.31  
vncserver_proxyclient_address = 172.16.56.31
```

- 在 [glance] 区域，配置镜像服务 API 的位置：

```
[glance]
```

```
api_servers = http://172.16.10.31:9292
```

- 在 [oslo_concurrency] 部分，配置锁路径：

```
[oslo_concurrency]
```

```
lock_path = /var/lib/nova/tmp
```

2. 同步数据库：

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

3. 启动nova服务并将其设置为随系统启动:

```
# 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
```

控制节点配置Nova组件

1. 编辑 `/etc/nova/nova.conf` 文件并完成下面的操作:
 - 在 `[DEFAULT]` 部分, 只启用计算和元数据API:

```
[DEFAULT]
```

```
enabled_apis = osapi_compute,metadata
```

- 在 `[DEFAULT]` 部分, 配置 `RabbitMQ` 消息队列访问权限:

```
[DEFAULT]
```

```
transport_url = rabbit://openstack:openstack@172.16.10.31
```

- 在 `[DEFAULT]` 和 `[keystone_auth_token]` 部分，配置认证服务访问：

```
[DEFAULT]
```

```
auth_strategy = keystone
```

```
[keystone_auth_token]
```

```
auth_uri = http://172.16.10.31:5000
```

```
auth_url = http://172.16.10.31:35357
```

```
memcached_servers = 172.16.10.31:11211
```

```
auth_type = password
```

```
project_domain_name = Default
```

```
user_domain_name = Default
```

```
project_name = service
```

```
username = nova
```

```
password = nova
```

- 在 `[DEFAULT]` 部分，启用网络服务支持：

[DEFAULT]

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

- 在 [vnc] 部分，启用并配置远程控制台访问：

[vnc]

```
enabled = True  
vncserver_listen = 0.0.0.0  
vncserver_proxyclient_address = 172.16.10.32  
novncproxy_base_url = http://172.16.10.31:6080/vnc_auto.html
```

- 在 [glance] 区域，配置镜像服务 API 的位置：

[glance]

```
api_servers = http://172.16.10.31:9292
```

- 在 [oslo_concurrency] 部分，配置锁路径：

```
[oslo_concurrency]
```

```
lock_path = /var/lib/nova/tmp
```

- 配置默认的虚拟机类型:

```
virt_type=kvm
```

2. 启动服务并验证

确定计算节点是支持虚拟机的硬件加速，返回值大于等于1，说明支持：

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

启动计算服务，并设置系统自启动：

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

验证nova服务是否正常

在控制节点加载admin的环境变量：

```
source admin-openstack
```

列出服务组件，以验证是否成功启动并注册了每个进程：

```
# nova service-list
```

Id	Binary	Host	Zone	Status	State	Updated_at	Disabled Reason
1	nova-scheduler	node1	internal	enabled	up	2017-01-12T11:13:01.000000	-
2	nova-consoleauth	node1	internal	enabled	up	2017-01-12T11:13:01.000000	-
3	nova-conductor	node1	internal	enabled	up	2017-01-12T11:13:01.000000	-
6	nova-compute	node2	nova	enabled	up	2017-01-12T11:13:01.000000	-

```
#openstack image list
```

ID	Name	Status
b14c429c-6c18-4407-acba-565ddd829ec4	cirros	active

也可以使用 `openstack compute service list` 命令，结果相同。

网络服务-Neutron

[参考官方文档](#)

Neutron由一个Neutron Server提供服务， 主要包含一些二层的插件， 如Linux Bridge， openvSwitch， DHCP-Agent, L3-Agent , LBAAS-Agent 和其他组件等。模拟了实际物理网络中的服务和协议。 Neutron有两种网络架构， 单一扁平网络和负杂的多网段网络， 这里以单一扁平网络为例。

配置控制节点

1. 在控制节点上创建服务证书， 完成这些步骤：
 - 创建 `neutron` 用户：

```
source admin-openstack
```

```
openstack user create --domain default --password-prompt neutron
```

User Password:

Repeat User Password:

+-----+-----+	
Field	Value
+-----+-----+	
domain_id	default
enabled	True
id	fe8f2e309bf24df4a0093f60228849e5
name	neutron
password_expires_at	None
+-----+-----+	

- 添加 **admin** 角色到 **neutron** 用户:

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

- 创建 **neutron** 服务实体:

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

```
+-----+-----+  
| Field      | Value                                     |  
+-----+-----+  
| description | OpenStack Networking                   |  
| enabled     | True                                   |  
| id          | fc5d229b7878429cada0303737da9b27    |  
| name        | neutron                               |  
| type        | network                               |  
+-----+-----+
```

2. 创建网络服务API端点:

```
openstack endpoint create --region RegionOne \  
network public http://172.16.10.31:9696
```

Field	Value
enabled	True
id	cca38f6d37c64acba35991340c2dd815
interface	public
region	RegionOne
region_id	RegionOne
service_id	fc5d229b7878429cada0303737da9b27
service_name	neutron
service_type	network
url	http://172.16.10.31:9696

```
openstack endpoint create --region RegionOne \  
network internal http://172.16.10.31:9696
```

Field	Value
enabled	True
id	b8dce2c9cd074b22ac7b58ae9881d963
interface	internal
region	RegionOne
region_id	RegionOne
service_id	fc5d229b7878429cada0303737da9b27
service_name	neutron
service_type	network
url	http://172.16.10.31:9696


```
openstack endpoint create --region RegionOne \
network admin http://172.16.10.31:9696
```

Field	Value
enabled	True
id	87955af92bb34a3bb71231efc336d17c
interface	admin
region	RegionOne
region_id	RegionOne
service_id	fc5d229b7878429cada0303737da9b27
service_name	neutron
service_type	network
url	http://172.16.10.31:9696

3. 计算节点配置提供者网络

编辑 `/etc/neutron/neutron.conf` 文件并完成如下操作：

- 在 [database] 部分，配置数据库访问：

```
[database]
```

```
connection = mysql+pymysql://neutron:neutron@172.16.10.31/neutron
```

- 在 [DEFAULT] 部分，启用ML2插件并禁用其他插件：

```
[DEFAULT]
```

```
core_plugin = ml2  
service_plugins =
```

- 在 [DEFAULT] 部分，配置 RabbitMQ 消息队列访问权限：

```
[DEFAULT]
```

```
transport_url = rabbit://openstack:openstack@172.16.10.31
```

- 在 “[DEFAULT]” 和 “[keystone_authtoken]” 部分，配置认证服务访问：

```
[DEFAULT]
```

```
auth_strategy = keystone
```

```
[keystone_authtoken]
```

```
auth_uri = http://172.16.10.31:5000
```

```
auth_url = http://172.16.10.31:35357
```

```
memcached_servers = 172.16.10.31:11211
```

```
auth_type = password
```

```
project_domain_name = Default
```

```
user_domain_name = Default
```

```
project_name = service
```

```
username = neutron
```

```
password = neutron
```

- 在 [DEFAULT] 和 [nova] 部分，配置网络服务来通知计算节点的网络拓扑变化：

```
[DEFAULT]
```

```
notify_nova_on_port_status_changes = True  
notify_nova_on_port_data_changes = True
```

```
[nova]
```

```
auth_url = http://172.16.10.31:35357  
auth_type = password  
project_domain_name = Default  
user_domain_name = Default  
region_name = RegionOne  
project_name = service  
username = nova  
password = nova
```

- 在 [oslo_concurrency] 部分，配置锁路径：

```
[oslo_concurrency]
```

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

4. ML2插件使用Linuxbridge机制来为实例创建layer-2虚拟网络基础设施,编辑 `/etc/neutron/plugins/ml2/ml2_conf.ini` 文件并完成以下操作：

- 在 [m12] 部分，启用flat和VLAN网络,删除 local：

```
[m12]
```

```
type_drivers = flat,vlan,gre,vxlan,geneve
```

- 在 [m12] 部分，禁用私有网络：

```
[m12]
```

```
tenant_network_types =
```

- 在 [m12] 部分，启用Linuxbridge机制：

```
[m12]
```

```
mechanism_drivers = linuxbridge
```

- 在 [m12] 部分，启用端口安全扩展驱动：

```
[ml2]
```

```
extension_drivers = port_security
```

- 在 `[ml2_type_flat]` 部分，配置公共虚拟网络为flat网络：

```
[ml2_type_flat]
```

```
flat_networks = public
```

- 在 `[securitygroup]` 部分，启用 ipset 增加安全组的方便性：

```
[securitygroup]
```

```
enable_ipset = True
```

5. 配置Linuxbridge代理

Linuxbridge代理为实例建立layer-2虚拟网络并且处理安全组规则。编辑 `/etc/neutron/plugins/ml2/linuxbridge_agent.ini` 文件并且完成以下操作：

- 在 `[linux_bridge]` 部分，将公共虚拟网络和公共物理网络接口对应起来，这里自定义public网络，对应接口为eth0：

```
[linux_bridge]
physical_interface_mappings = public:eth0
```

- 在 `[vxlan]` 部分，禁止VXLAN覆盖网络：

```
[vxlan]
enable_vxlan = False
```

- 在 `[securitygroup]` 部分，启用安全组并配置 Linux 桥接 iptables 防火墙驱动：

```
[securitygroup]

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

6. 配置DHCP代理 DHCP agent为虚拟网络提供了DHCP服务，编辑 `/etc/neutron/dhcp_agent.ini` 文件并完成下面的操作：
在 `[DEFAULT]` 部分，配置Linuxbridge驱动接口，DHCP驱动并启用隔离元数据，这样在公共网络上的实例就可以通过网络来访问元数据

```
[DEFAULT]
```

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

7. 配置元数据代理

metadata agent提供了如实例验证之类的配置信息。编辑 `/etc/neutron/metadata_agent.ini`，在 `[DEFAULT]` 部分，配置元数据主机以及自定义共享密码：

```
[DEFAULT]
```

```
nova_metadata_ip = 172.16.10.31  
metadata_proxy_shared_secret = trying
```

8. 配置计算服务使用网络服务

编辑 `/etc/nova/nova.conf` 文件，在 `[neutron]` 部分，配置访问参数，启用元数据代理并设置密码(此处的密码和metadata密码对应)：


```
[neutron]
...
url = http://172.16.10.31:9696
auth_url = http://172.16.10.31:35357
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = neutron
password = neutron
service_metadata_proxy = True
metadata_proxy_shared_secret = trying
```

9. 完成安装并启动服务

- 网络服务初始化脚本需要一个超链接 `/etc/neutron/plugin.ini` 指向ML2插件配置文件 `/etc/neutron/plugins/ml2/ml2_conf.ini`。如果超链接不存在，使用下面的命令创建它：

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

- 同步数据库：

```
# 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
```

- 重启计算API 服务:

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

- 当系统启动时, 启动 Networking 服务并配置它启动, 值针对当前配置的提供者网络:

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

安装配置计算节点

1. 配置统一组件

编辑 `/etc/neutron/neutron.conf` 文件并完成如下操作:

- 在 [database] 部分，注释所有 connection 项，因为计算节点不直接访问数据库。
- 在 [DEFAULT] 部分，配置 RabbitMQ 消息队列访问权限：

```
[DEFAULT]
```

```
transport_url = rabbit://openstack:openstack@172.16.10.31
```

- 在 [DEFAULT] 和 [keystone_authtoken] 部分，配置认证服务访问：

```
[DEFAULT]
```

```
auth_strategy = keystone
```

```
[keystone_authtoken]
```

```
auth_uri = http://172.16.10.31:5000
```

```
auth_url = http://172.16.10.31:35357
```

```
memcached_servers = 172.16.10.31:11211
```

```
auth_type = password
```

```
project_domain_name = Default
```

```
user_domain_name = Default
```

```
project_name = service
```

```
username = neutron
```

```
password = neutron
```

- 在 [oslo_concurrency] 部分，配置锁路径：

```
[oslo_concurrency]

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

2. 配置提供者网络 Linuxbridge代理为实例建立layer-2虚拟网络并且处理安全组规则。编辑 /etc/neutron/plugins/ml2/linuxbridge_agent.ini 文件并且完成以下操作：

- 在 [linux_bridge] 部分，将公共虚拟网络和公共物理网络接口对应起来：

```
[linux_bridge]
physical_interface_mappings = public:eth0
```

- 在 [vxlan] 部分，禁止VXLAN覆盖网络：

```
[vxlan]
enable_vxlan = False
```

- 在 [securitygroup] 部分，启用安全组并配置 Linux 桥接 iptables 防火墙驱动：

```
[securitygroup]
```

```
enable_security_group = True
```

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

3. 配置计算服务使用网络服务 编辑 `/etc/nova/nova.conf` 文件并完成下面的操作:

- 在 `[neutron]` 部分, 配置访问参数:

```
[neutron]
```

```
url = http://172.16.10.31:9696
```

```
auth_url = http://172.16.10.31:35357
```

```
auth_type = password
```

```
project_domain_name = Default
```

```
user_domain_name = Default
```

```
region_name = RegionOne
```

```
project_name = service
```

```
username = neutron
```

```
password = neutron
```

4. 完成安装

- 重启计算服务:

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

- 启动Linuxbridge代理并配置它开机自启动:

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

5. 在计算节点验证服务是否正常

```
# source admin-openstack
```

- 列出加载的扩展来验证 `neutron-server` 进程是否正常启动:

```
# neutron ext-list
+-----+-----+
| alias                | name                |
+-----+-----+
| default-subnetpools  | Default Subnetpools |
| availability_zone     | Availability Zone    |
| network_availability_zone | Network Availability Zone |
| binding              | Port Binding        |
| agent               | agent               |
```

subnet_allocation	Subnet Allocation	
dhcp_agent_scheduler	DHCP Agent Scheduler	
tag	Tag support	
external-net	Neutron external network	
flavors	Neutron Service Flavors	
net-mtu	Network MTU	
network-ip-availability	Network IP Availability	
quotas	Quota management support	
provider	Provider Network	
multi-provider	Multi Provider Network	
address-scope	Address scope	
subnet-service-types	Subnet service types	
standard-attr-timestamp	Resource timestamps	
service-type	Neutron Service Type Management	
extra_dhcp_opt	Neutron Extra DHCP opts	
standard-attr-revisions	Resource revision numbers	
pagination	Pagination support	
sorting	Sorting support	
security-group	security-group	
rbac-policies	RBAC Policies	
standard-attr-description	standard-attr-description	
port-security	Port Security	
allowed-address-pairs	Allowed Address Pairs	
project-id	project_id field enabled	
+-----+	+-----+	+-----+

- 提供者网络使用 `openstack network agent list` 或者 `neutron agent-list` 命令来确认neutron服务代理都已经启动

(每个计算节点会有一个代理, 控制节点三个代理)

```
# neutron agent-list
```

```
+-----+-----+-----+-----+-----+-----+-----+
-----+
| id              | agent_type      | host | availability_zone | alive | admin_state_up | binary
|
+-----+-----+-----+-----+-----+-----+-----+
-----+
| 18777143-ded5-4af | Linux bridge    | node2 |                   |       | True            | neutron-
|
| 4-91ae-          | agent           |       |                   |       |                 |
linuxbridge-agent |
| ff3c30d88eee     |                 |       |                   |       |                 |
|
| 79527fbd-727c-438 | Linux bridge    | node1 |                   |       | True            | neutron-
|
| 8-9189-bc6c40bd6d | agent           |       |                   |       |                 |
linuxbridge-agent |
| 07               |                 |       |                   |       |                 |
|
| 7a7201f3-277e-    | Metadata agent  | node1 |                   |       | True            | neutron-
metadata-         |
| 4acb-            |                 |       |                   |       |                 |
|
| 8de3-e8bbdc02f485 |                 |       |                   |       |                 |
|
| f85b03ec-d9b6     | DHCP agent      | node1 | nova              |       | True            | neutron-
```



```
dhcp-agent |
| -4a5a-bbbe- | | | |
|
| 7053f2db2409 | | | |
|
+-----+-----+-----+-----+-----+-----+
-----+
```

“

提示： 每个配置文件的配置项一定要在顶格，之前不能有空格，如果出现错，可通过查看控制节点的所有服务日志，一般带有 **ERROR** 标示的重点分析，即可排除故障。

创建虚拟机

[参考官方文档](#)

创建一个虚拟的提供者网络

1. 创建网络：

```
# source admin-openstack
```

```
# openstack network create --share \  
--provider-physical-network public \  
--provider-network-type flat public
```

+-----+-----+		
Field	Value	
+-----+-----+		
admin_state_up	UP	
availability_zone_hints		
availability_zones		
created_at	2017-01-13T09:14:02Z	
description		
headers		
id	ba798366-8f3b-4561-9a9b-244fb3aa54de	
ipv4_address_scope	None	
ipv6_address_scope	None	
mtu	1500	
name	public	
port_security_enabled	True	
project_id	c34d22ac34cb4c62a025295cd3d6540c	
project_id	c34d22ac34cb4c62a025295cd3d6540c	
provider:network_type	flat	
provider:physical_network	public	
provider:segmentation_id	None	
revision_number	3	
router:external	Internal	
shared	True	
status	ACTIVE	

subnets		
tags	[]	
updated_at	2017-01-13T09:14:02Z	
+-----+		

2. 创建一个子网

```
openstack subnet create --network public \
--allocation-pool start=172.16.10.40,end=172.16.10.60 \
--dns-nameserver 172.16.0.1 --gateway 172.16.0.1 \
--subnet-range 172.16.0.0/16 public-instance
```

+-----+		
Field	Value	
+-----+		
allocation_pools	172.16.10.40-172.16.10.60	
cidr	172.16.0.0/16	
created_at	2017-01-13T09:20:59Z	
description		
dns_nameservers	172.16.0.1	
enable_dhcp	True	
gateway_ip	172.16.0.1	
headers		
host_routes		
id	7e48698b-6fa8-4f4e-a69f-43be676f189e	
ip_version	4	
ipv6_address_mode	None	

ipv6_ra_mode	None	
name	public-instance	
network_id	ba798366-8f3b-4561-9a9b-244fb3aa54de	
project_id	c34d22ac34cb4c62a025295cd3d6540c	
project_id	c34d22ac34cb4c62a025295cd3d6540c	
revision_number	2	
service_types	[]	
subnetpool_id	None	
updated_at	2017-01-13T09:20:59Z	
+-----+	+-----+	+-----+

创建一个nano类型的虚拟机

设置一个最小类型的测试虚拟机，测试运行环境是否正常：

```
# source admin-openstack
# 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    |
| properties      |         |
| ram             | 64      |
| rxtx_factor     | 1.0     |
| swap            |         |
| vcpus           | 1       |
+-----+-----+
```

生成一个键值对

在启动实例前，必须使用demo用户添加一个公共密钥到计算服务，这样当虚拟机启动后，控制节点就可以直接登录管理虚拟机，不需要密码。

```
# source demo-openstack
# ssh-keygen -q -N ""
# openstack keypair create --public-key ~/.ssh/id_rsa.pub mykey
```

```
+-----+-----+
| Field      | Value                                     |
+-----+-----+
| fingerprint | b2:2c:24:53:81:9a:77:e4:62:cb:2d:40:10:01:7e:86 |
| name        | mykey                                   |
| user_id     | 3cd88136979a4934a566746a35abb0a5          |
+-----+-----+
```

“

提示：如果此步出现401错误，需要排查demo的用户权限，使用 `source demo-openstack` 和 `openstack token issue` 看能否获取token,如果依旧是401，需要编辑 `/etc/keystone/keystone-paste.ini` 文件，从 `[pipeline:public_api]`，`[pipeline:admin_api]` 和 `[pipeline:api_v3]` 部分加上 `admin_token_auth`。然后删除已经创建的demo和user,并重新创建demo用户：

```
openstack role delete user
openstack user delete demo
openstack project delete demo
```

```
openstack project create --domain default \ --description "Demo Project" demo
openstack user create --domain default \ --password-prompt demo
openstack role create user
openstack role add --project demo --user demo user
```

验证公钥添加:

```
# openstack keypair list

+-----+-----+
| Name   | Fingerprint                                     |
+-----+-----+
| mykey  | b2:2c:24:53:81:9a:77:e4:62:cb:2d:40:10:01:7e:86 |
+-----+-----+
```

添加安全组规则

默认情况下, `default` 安全组适用于所有实例并且包括拒绝远程访问实例的防火墙规则。对诸如CirrOS这样的Linux镜像, 我们推荐至少允许ICMP (ping) 和安全shell(SSH)规则。

允许ICMP(PING)数据包:

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

Field	Value
created_at	2017-01-16T07:23:54Z
description	
direction	ingress
ethertype	IPv4
headers	
id	06c3cac4-74de-4119-af48-67f8294d1695
port_range_max	None
port_range_min	None
project_id	c091e1b9295748c185b7a381983f0b19
project_id	c091e1b9295748c185b7a381983f0b19
protocol	icmp
remote_group_id	None
remote_ip_prefix	0.0.0.0/0
revision_number	1
security_group_id	68b6970a-295b-4e0d-b3cb-96271c342560
updated_at	2017-01-16T07:23:54Z

添加允许安全 shell (SSH) 的访问:


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

Field	Value
created_at	2017-01-16T07:24:15Z
description	
direction	ingress
ethertype	IPv4
headers	
id	bbd0ee53-ec8d-43b5-b838-ac6e72bee300
port_range_max	22
port_range_min	22
project_id	c091e1b9295748c185b7a381983f0b19
project_id	c091e1b9295748c185b7a381983f0b19
protocol	tcp
remote_group_id	None
remote_ip_prefix	0.0.0.0/0
revision_number	1
security_group_id	68b6970a-295b-4e0d-b3cb-96271c342560
updated_at	2017-01-16T07:24:15Z

创建虚拟机

列出可用的资源和实例对象：

```
# source demo-openstack
```

```
# openstack flavor list
```

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

```
# openstack image list
```

```
+-----+-----+-----+
| ID                  | Name   | Status |
+-----+-----+-----+
| b14c429c-6c18-4407-acba-565ddd829ec4 | cirros | active |
+-----+-----+-----+
```

```
# openstack network list
```

```
+-----+-----+-----+-----+
| ID                  | Name   | Subnets |
+-----+-----+-----+-----+
| ba798366-8f3b-4561-9a9b-244fb3aa54de | public | 7e48698b-6fa8-4f4e-a69f-43be676f189e |
+-----+-----+-----+-----+
```

```
# openstack security group list
```

```
+-----+-----+-----+-----+-----+
| ID                  | Name   | Description | Project |
+-----+-----+-----+-----+-----+
| 68b6970a-295b-4e0d-b3cb-96271c342560 | default | Default security group | c091e1b9295748c185b7a381983f0b19 |
+-----+-----+-----+-----+-----+
```

启动云主机,指定网络ID:

```
openstack server create --flavor m1.nano --image cirros \  
  --nic net-id=ba798366-8f3b-4561-9a9b-244fb3aa54de --security-group default \  
  --key-name mykey public-instance
```

Field	Value
OS-DCF:diskConfig	MANUAL
OS-EXT-AZ:availability_zone	
OS-EXT-STS:power_state	NOSTATE
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	c7VpL6tQPDvo
config_drive	
created	2017-01-16T07:48:49Z
flavor	m1.nano (0)
hostId	
id	024be3b2-2dd4-4a62-8ddc-e4f4244f9dd8
image	cirros (b14c429c-6c18-4407-acba-565ddd829ec4)
key_name	mykey
name	public-instance

```

| os-extended-volumes:volumes_attached | [] |
| progress | 0 |
| project_id | c091e1b9295748c185b7a381983f0b19 |
| properties | |
| security_groups | [{u'name': u'default'}] |
| status | BUILD |
| updated | 2017-01-16T07:48:50Z |
| user_id | 3cd88136979a4934a566746a35abb0a5 |
+-----+-----+

```

如果虚拟机启动成功，可以通过 `openstack server list` 命令查看：

```

# openstack server list
+-----+-----+-----+-----+-----+
| ID | Name | Status | Networks | Image Name |
+-----+-----+-----+-----+-----+
| 024be3b2-2dd4-4a62-8ddc-e4f4244f9dd8 | public-instance | ACTIVE | public=172.16.10.49 | cirros |
+-----+-----+-----+-----+-----+

```

此时，通过在控制节点上使用ssh命令，即可登录cirros镜像系统，由于之前已经做过密码验证，所以不需要密码：

```

[root@node1 ~]# ssh cirros@172.16.10.49
$ ls
$

```

如果通过网页VNC管理虚拟机，可以执行如下命令，获取管理链接：

```
# openstack console url show public-instance
+-----+-----+
| Field | Value |
+-----+-----+
| type  | novnc |
| url   | http://172.16.10.31:6080/vnc_auto.html?token=1bfda8ba-0e05-447b-936d-dea6f1eb9780 |
+-----+-----+
```

在浏览器上直接打开此链接，即可对虚拟机进行操作。

Dashboard-Horizon

[参考官方文档](#)

OpenStack的管理界面是由Horizon提供的，使用django编写。这个组件可以安装在任何节点上（除控制节点，此版本可能与控制节点某些组件不兼容），可以通过界面交互的模式对openstack进行管理，只要修改对应的配置即可。Horizon不需要与数据库直接交流，所以不需要配置数据库。

计算节点上安装软件包并配置

1. 安装软件包：

```
yum install openstack-dashboard
```

2. 编辑文件 /etc/openstack-dashboard/local_settings 并完成如下动作：

- 配置仪表盘要连接的控制节点，以使用 OpenStack 服务：

```
OPENSTACK_HOST = "172.16.10.31"
```

- 允许所有主机访问仪表板：

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

- （略）配置 memcached 会话存储服务，并将其他的会话存储服务配置注释（如果本机没有memcache服务此步可以不做什么修改，否则无法登录）：

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': '127.0.0.1:11211',
    },
}
```

- 启用第3版认证API:

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

- 启用对域的支持

```
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
```

- 配置API版本:

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

- 通过仪表盘创建用户时的默认域配置为 default :

```
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "default"
```

- 通过仪表盘创建的用户默认角色配置为 user :

```
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
```

- 如果您选择网络参数1, 禁用支持3层网络服务以防止报错(如果使用的3层网络服务这里需要改为ture):

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

- 可以选择性地配置时区:

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

3. 重启http服务(和memcached):


```
systemctl restart httpd.service
```

4. 验证是否能登录Horizon <http://172.16.10.32/dashboard>

块存储-Cinder

Cinder提供了多种块存储的解决方案，可以使用多种后端驱动，如NAS/SAN,NFS.ISCSI,Ceph,GlusterFS等。

配置cinder控制节点

1. 创建cinder相关的数据库：

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

2. 验证创建的数据库是否有权限：

```
[root@node1 ~]# mysql -h 172.16.10.31 -ucinder -pcinder
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 202
Server version: 10.1.18-MariaDB MariaDB Server

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+-----+
| Database          |
+-----+
| cinder             |
| information_schema |
+-----+
2 rows in set (0.01 sec)
```

3. 要创建服务证书，完成这些步骤：

- 创建一个 cinder 用户：

```
[root@node1 ~]# source admin-openstack
[root@node1 ~]# openstack user create --domain default --password-prompt cinder
User Password:
Repeat User Password:
```

+-----+-----+		
Field	Value	
+-----+-----+		
domain_id	default	
enabled	True	
id	eff967a16585478789a085a144b46050	
name	cinder	
password_expires_at	None	
+-----+-----+		

- 添加 admin 角色到 cinder 用户上:

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

- 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	5ea612137d5a4fda80908f20b750e6d9
name	cinder
type	volume

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

Field	Value
description	OpenStack Block Storage
enabled	True
id	2be6fda30dc74ec69aa3b35fd263fca6
name	cinderv2
type	volumev2

4. 创建块设备存储服务v1和v2的 API 入口点:

```
# openstack endpoint create --region RegionOne \  
volume public http://172.16.10.31:8776/v1/%(tenant_id)s  
+-----+-----+  
| Field      | Value                                         |  
+-----+-----+  
| enabled    | True                                         |  
| id         | b23b45cbb8f14ed2880f7cfb103499d8          |  
| interface  | public                                       |  
| region     | RegionOne                                   |  
| region_id  | RegionOne                                   |  
| service_id | 5ea612137d5a4fda80908f20b750e6d9          |  
| service_name | cinder                                     |  
| service_type | volume                                       |  
| url        | http://172.16.10.31:8776/v1/%(tenant_id)s |  
+-----+-----+
```

```
# openstack endpoint create --region RegionOne \
volume internal http://172.16.10.31:8776/v1/%(tenant_id)s
```

Field	Value
enabled	True
id	6a397b2c391d4d12b64fcb8ec2040107
interface	internal
region	RegionOne
region_id	RegionOne
service_id	5ea612137d5a4fda80908f20b750e6d9
service_name	cinder
service_type	volume
url	http://172.16.10.31:8776/v1/%(tenant_id)s

```
# openstack endpoint create --region RegionOne volume admin http://172.16.10.31:8776/v1/%(tenant_id)s
```

+-----+-----+-----+	
Field	Value
+-----+-----+-----+	
enabled	True
id	d9f3c821793f464e901a09ac51b4ef84
interface	admin
region	RegionOne
region_id	RegionOne
service_id	5ea612137d5a4fda80908f20b750e6d9
service_name	cinder
service_type	volume
url	http://172.16.10.31:8776/v1/%(tenant_id)s
+-----+-----+-----+	

```
# openstack endpoint create --region RegionOne \  
volumev2 public http://172.16.10.31:8776/v2/%(tenant_id)s
```

Field	Value
enabled	True
id	6e3ffaa6e306450398dd1c81dd4c96a2
interface	public
region	RegionOne
region_id	RegionOne
service_id	2be6fda30dc74ec69aa3b35fd263fca6
service_name	cinderv2
service_type	volumev2
url	http://172.16.10.31:8776/v2/%(tenant_id)s


```
# openstack endpoint create --region RegionOne volumev2 internal http://172.16.10.31:8776/v2/%\
(tenant_id)s
```

Field	Value
enabled	True
id	8a9c0d3e643d4bd0b2d643a818dcbdd2
interface	internal
region	RegionOne
region_id	RegionOne
service_id	2be6fda30dc74ec69aa3b35fd263fca6
service_name	cinderv2
service_type	volumev2
url	http://172.16.10.31:8776/v2/%(tenant_id)s

```
# openstack endpoint create --region RegionOne volumev2 admin http://172.16.10.31:8776/v2/%(tenant_id)s
```

Field	Value
enabled	True
id	428f95053d1f4298981297b44dca37ce
interface	admin
region	RegionOne
region_id	RegionOne
service_id	2be6fda30dc74ec69aa3b35fd263fca6
service_name	cinderv2
service_type	volumev2
url	http://172.16.10.31:8776/v2/%(tenant_id)s

5. 安装并配置组件:

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

6. 编辑 `/etc/cinder/cinder.conf`，同时完成如下动作:

- 在 [database] 部分，配置数据库访问:

```
[database]
```

```
connection = mysql+pymysql://cinder:cinder@172.16.10.31/cinder
```

- 在 [DEFAULT] 部分，配置 RabbitMQ 消息队列访问权限：

```
[DEFAULT]
```

```
transport_url = rabbit://openstack:openstack@172.16.10.31
```

- 在 [DEFAULT] 和 [keystone_authtoken] 部分，配置认证服务访问：

```
[DEFAULT]
```

```
auth_strategy = keystone
```

```
[keystone_authtoken]
```

```
auth_uri = http://172.16.10.31:5000
```

```
auth_url = http://172.16.10.31:35357
```

```
memcached_servers = 172.16.10.31:11211
```

```
auth_type = password
```

```
project_domain_name = Default
```

```
user_domain_name = Default
```

```
project_name = service
```

```
username = cinder
```

```
password = cinder
```

- 在 [oslo_concurrency] 部分，配置锁路径：

```
[oslo_concurrency]
```

```
lock_path = /var/lib/cinder/tmp
```

7. 配置iscsi进程监听的IP:

```
iscsi_ip_address = 172.16.10.31
```

8. 初始化块设备服务的数据库:

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

9. 配置计算服务使用块存储, 修改 `/etc/nova/nova.conf`

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

10. 完成安装

- 重启nova-api:

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

- 设置开机自启动, 并启动cinder:

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

配置存储节点

此处使用控制节点做为存储节点，在实际的生产中，可以单独使用一个节点或集群作为存储。

1. 安装支持的工具包，并启动服务：

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

2. 存储节点需要有单独的存储磁盘来提供数据存储，此处为控制节点新加的一个磁盘sdb,在sdb上创建LVM卷：

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

3. 创建 LVM 卷组 cinder-volumes,块存储服务会在这个卷组中创建逻辑卷：

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

4. 编辑 `/etc/lvm/lvm.conf` 文件，在 `devices` 部分，添加一个过滤器：

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

如果您的存储节点在操作系统磁盘上使用了 LVM，您还必需添加相关的设备到过滤器中。例如，如果 /dev/sda 设备包含操作系统(a 表示 **accept**,b表示 **reject**):

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

5. 在存储节点安装软件包（此处为控制节点）：

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

6. 编辑 /etc/cinder/cinder.conf，同时完成如下动作（如果控制节点和存储节点是同一节点，此步不用操作）：

- 在 [database] 部分，配置数据库访问：

```
[database]  
  
connection = mysql+pymysql://cinder:cinder@172.16.10.31/cinder
```

- 在 [DEFAULT] 部分，配置 **RabbitMQ** 消息队列访问权限：

```
[DEFAULT]
```

```
transport_url = rabbit://openstack:openstack@172.16.10.31
```

- 在 `[DEFAULT]` 和 `[keystone_authtoken]` 部分，配置认证服务访问：

```
[DEFAULT]
```

```
auth_strategy = keystone
```

```
[keystone_authtoken]
```

```
auth_uri = http://172.16.10.31:5000
```

```
auth_url = http://172.16.10.31:35357
```

```
memcached_servers = 172.16.10.31:11211
```

```
auth_type = password
```

```
project_domain_name = Default
```

```
user_domain_name = Default
```

```
project_name = service
```

```
username = cinder
```

```
password = cinder
```

- 配置 `iscsi_ip_address` 选项为存储节点IP：


```
iscsi_ip_address = 172.16.10.31
```

- 添加 [lvm] 区域并加入如下内容:

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

- 在 [DEFAULT] 部分, 启用 LVM 后端:

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

- 在 [DEFAULT] 区域, 配置镜像服务 API 的位置:

```
[DEFAULT]  
  
glance_api_servers = http://172.16.10.31:9292
```

- 在 [oslo_concurrency] 部分，配置锁路径：

```
[oslo_concurrency]
```

```
lock_path = /var/lib/cinder/tmp
```

7. 启动服务

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

8. 验证操作 在控制节点查看服务组件是否运行正常

```
[root@node1 ~]# source admin-openstack
[root@node1 ~]# openstack volume service list
```

Binary	Host	Zone	Status	State	Updated At
cinder-scheduler	node1	nova	enabled	up	2017-01-17T04:53:03.000000
cinder-volume	node1@lvm	nova	enabled	up	2017-01-17T04:53:04.000000

9. 使用demo登录Horizon,在管理界面添加卷，连接卷到虚拟机,登录虚拟机，格式化新连接的磁盘vdb。

```
sudo mkfs.ext4 /dev/vdb  
sudo mount /dev/vdb /tmp  
sudo echo "this file on vdb" >> /tmp/test.txt  
sudo umount /tmp
```

10. 在管理界面断开此卷，可以尝试将写有内容的此卷连接到其他的虚拟机上，依旧可以读取之前写入的内容。