Openstack搭建

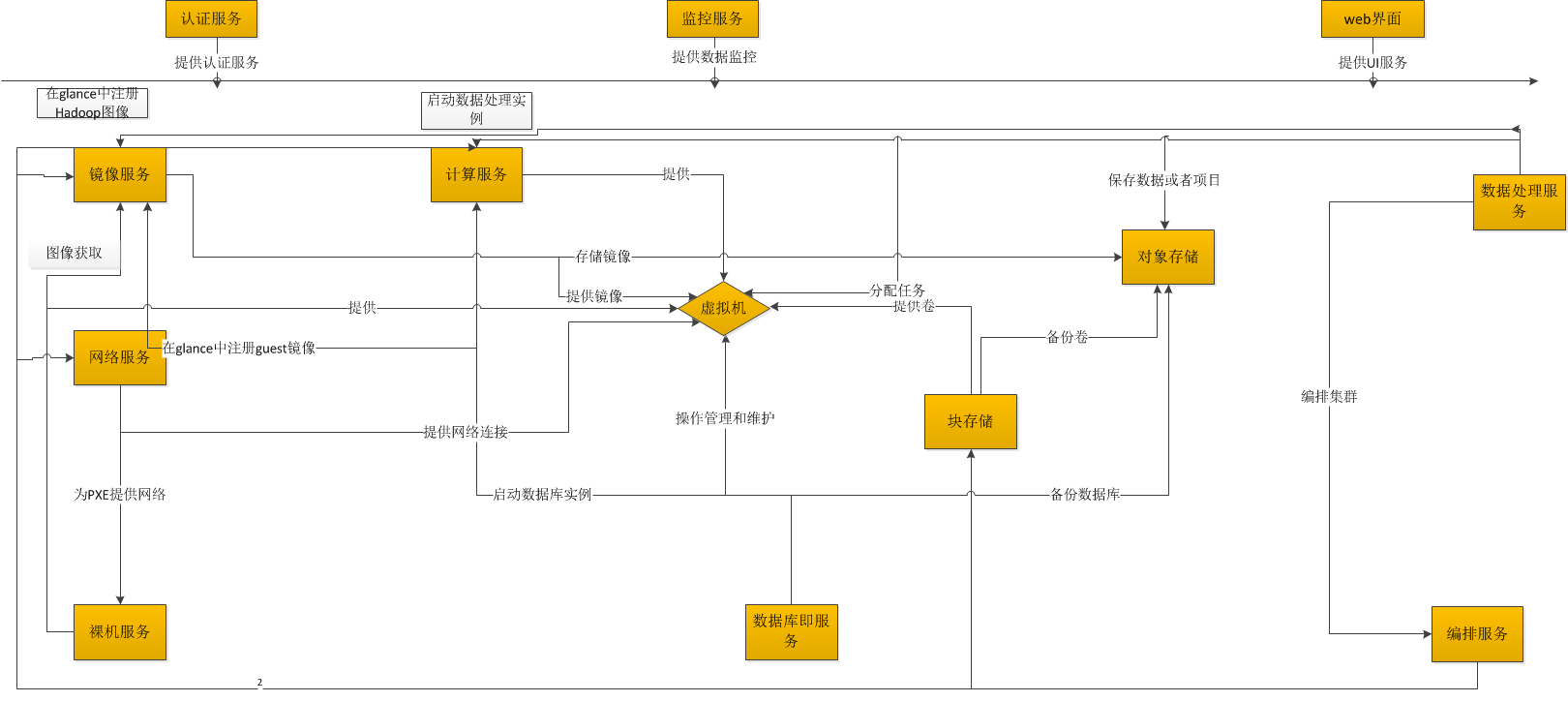
关于OpenStack

该系统由OpenStack的若干服务组成，这些服务根据您的云一起工作，服务包括该计算机的身份、网络、图像、存储块，对象存储、遥测、业务流程及数据库服务。任何项目的安装可以分开和独立的配置作为连接或实体。

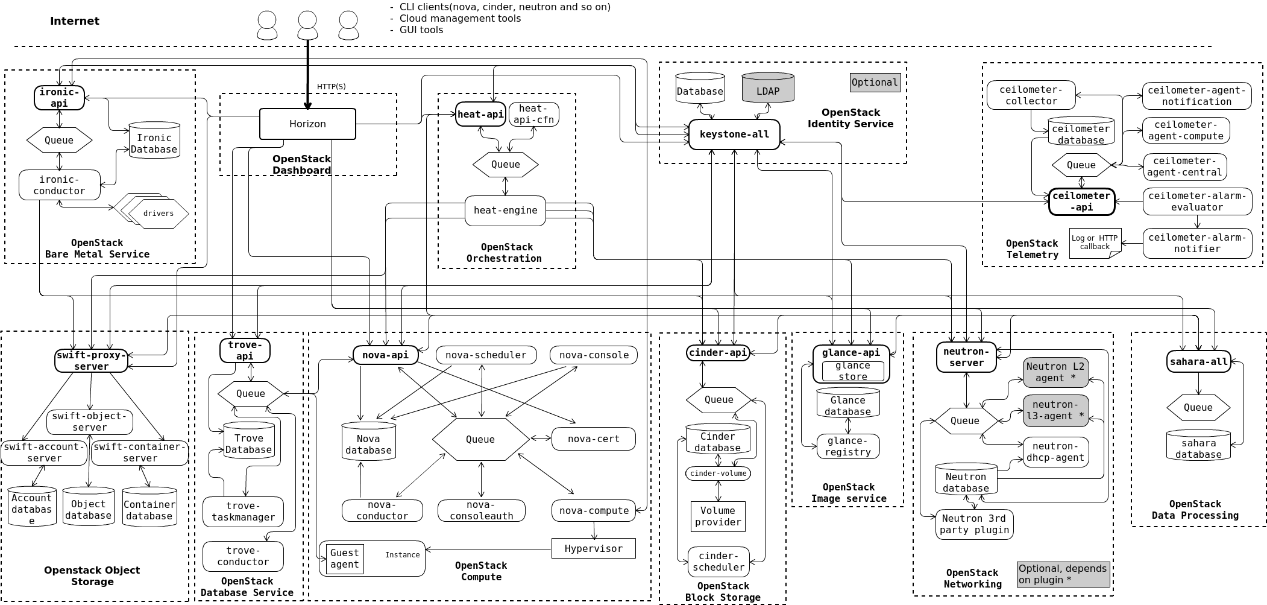
OpenStack各组件名称及功能

|  |  |  |
| --- | --- | --- |
| 项目名称 | 项目代码 | 描述 |
| Dashboard | Horizon | 为OpenStack提供了一个web管理界面，可在web界面创建实例等操作 |
| Network | Neutron | 实现了虚拟机的网络资源管理。 |
| Computer | Nova | 通过虚拟化技术提供计算资源池 |
| Identify Service | Keystone | 为其他服务提供管理身份验证，授权和端点目录 |
| Block Service | Cinder | 为运行的实例提供持久的存储 |
| Object Service | Swift | 是一个多租户对象存储系统，用于存储帐户，容器和对象的磁盘 |
| Image Service | Glance | Image服务使用户能够发现，注册和检索虚拟机映像 |
| Orchestration Service | Heat | 编排服务通过运行OpenStack API调用来生成正在运行的云应用程序，从而提供基于模板的业务流程来描述云应用程序。 |
| Bare Metal Service | Ironic | 裸机服务是一组组件集合，可以为管理和配置物理机器提供支持。 |
| Database Service | Trove | 数据库服务为关系数据库引擎和非关系数据库引擎提供了可扩展且可靠的云配置功能。用户可以快速轻松地使用数据库功能，而无需处理复杂的管理任务。云用户和数据库管理员可以根据需要配置和管理多个数据库实例。 |
| Metering & Data Collecting Service | Ceilometer | 提供监控和数据采集、计量服务 |
| Containers Service | Zun | Zun是Openstack中提供容器管理服务的组件 |
| Clustering Service | Senlin | Senlin是OpenStack云的集群服务。它创建并运行由其他OpenStack服务公开的同类对象群集。使相似对象的集合更容易编排。 |
| Shared File Systems Service | Manila | 为虚拟机提供文件存储。文件共享系统服务提供了管理和供应文件共享的抽象化。该服务还支持共享类型的管理以及驱动程序支持共享快照。 |
| Load-balancer Service | Octavia | Octavia是一款开源的运营商级负载均衡解决方案，旨在与OpenStack配合使用。 |

OpenStack架构图

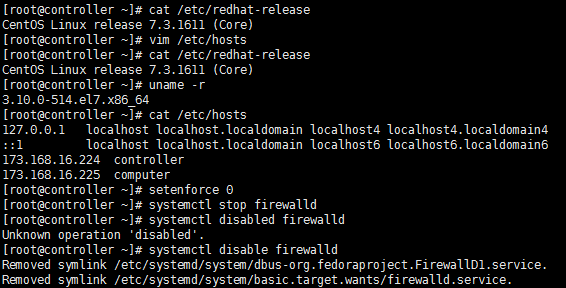


OpenStack逻辑体系结构



1. 基础环境搭建

1.查看环境配置，配置好名称解析，并关闭selinux和firewall，在计算节点也做相同配置



2.安装时间服务器

[root@controller ~]# yum install chrony

[root@controller ~]# sed -i '7a server 173.168.16.224 iburst' /etc/chrony.conf

[root@controller ~]# sed -i '8a allow 173.168.16.0/24' /etc/chrony.conf //为了使其他节点连接到控制器上的chrony守护程序



[root@controller ~]# systemctl enable chronyd

[root@controller ~]# systemctl start chronyd //启动并开机自启动chrony

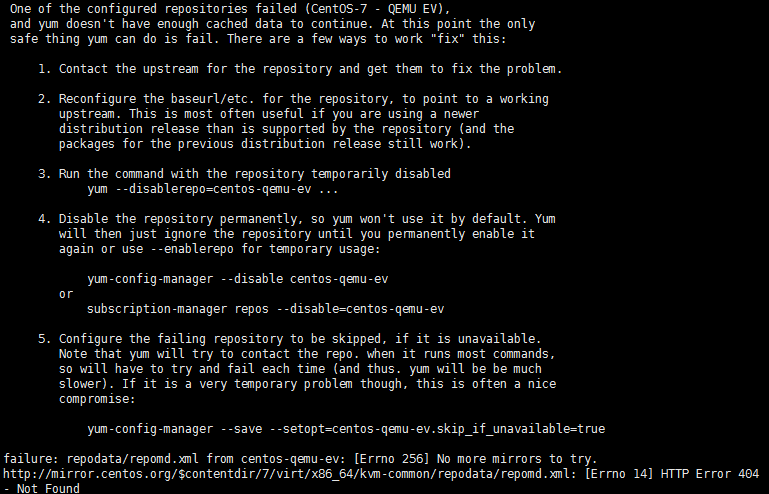


3.启用OpenStack仓库

[root@controller ]# yum install centos-release-openstack-pike //安装OpenStack存储库，所有节点都要安装

Yum upgrade //升级所有节点上的安装包

这里出现一个错误，yum源出问题了，执行下图的第四条和第五条命令即可



[root@controller ]# yum -y install python-openstackclient //OpenStack客户端，在所有节点上安装

[root@controller ]# yum install openstack-selinux //安装 openstack-selinux软件包,在所有节点上安装

4.安装mariadb数据库

yum install mariadb mariadb-server python2-PyMySQL

[root@controller ~]# yum install mariadb mariadb-server python2-PyMySQL

创建并编辑该**/etc/my.cnf.d/openstack.cnf**文件并添加以下部分：

**[mysqld]**

bind-address = 173.168.16.224

default-storage-engine = innodb

innodb\_file\_per\_table = on

max\_connections = 4096

collation-server = utf8\_general\_ci

character-set-server = utf8

启动数据库

[root@controller my.cnf.d]# systemctl enable mariadb

[root@controller my.cnf.d]# systemctl start mariadb



初始化mariadb数据库，运行**mysql\_secure\_installation** 脚本来保护数据库服务。特别是，为数据库**root**帐户选择一个合适的密码

5.安装消息队列

[root@controller /]# yum install rabbitmq-server //安装软件包

启动消息队列服务并将其配置为在系统引导时启动

[root@controller /]# systemctl enable rabbitmq-server.service

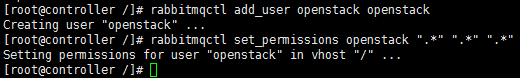
[root@controller /]# systemctl start rabbitmq-server.service



添加OpenStack用户

[root@controller /]# rabbitmqctl add\_user openstack openstack

允许用户进行配置，写入和读取访问 **openstack**



启用rabbitmq\_management插件

[root@controller /]# rabbitmq-plugins enable rabbitmq\_management

[root@controller /]# systemctl restart rabbitmq-server

6.安装memcached缓存服务

[root@controller /]# yum install memcached python-memcached //安装软件包

启动Memcached服务并将其配置为在系统引导时启动

[root@controller /]# systemctl enable memcached

[root@controller /]# systemctl start Memcached

7.安装ETCD

OpenStack服务可能使用Etcd，这是一个分布式可靠的键值存储，用于分布式密钥锁定，存储配置，跟踪服务的实时性和其他场景。

yum install etcd //安装软件包

编辑**/etc/etcd/etcd.conf**文件并设置**ETCD\_INITIAL\_CLUSTER**， **ETCD\_INITIAL\_ADVERTISE\_PEER\_URLS**，**ETCD\_ADVERTISE\_CLIENT\_URLS**， **ETCD\_LISTEN\_CLIENT\_URLS**控制器节点，以使经由管理网络通过其他节点的访问的管理IP地址

#[Member]

ETCD\_DATA\_DIR="/var/lib/etcd/default.etcd"

ETCD\_LISTEN\_PEER\_URLS=<http://173.168.16.224:2380>

ETCD\_LISTEN\_CLIENT\_URLS=<http://173.168.16.224:2379>

ETCD\_NAME="controller"

*#[Clustering]*

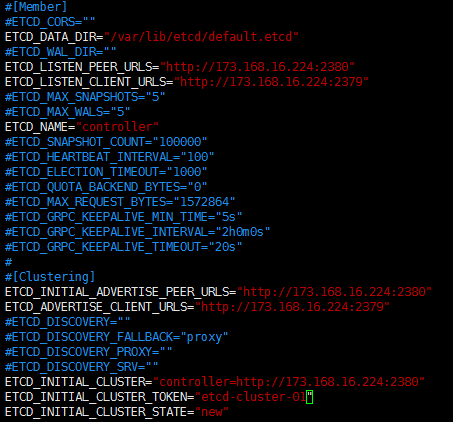
ETCD\_INITIAL\_ADVERTISE\_PEER\_URLS=<http://173.168.16.224:2380>

ETCD\_ADVERTISE\_CLIENT\_URLS=<http://173.168.16.224:2379>

ETCD\_INITIAL\_CLUSTER="controller=http://173.168.16.224:2380"

ETCD\_INITIAL\_CLUSTER\_TOKEN="etcd-cluster-01"

ETCD\_INITIAL\_CLUSTER\_STATE="new"



启用并启动etcd服务

[root@controller /]# systemctl enable etcd

[root@controller /]# systemctl start etcd

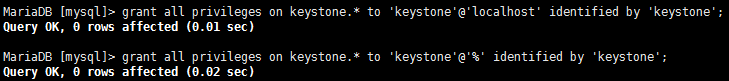
1. keystone认证服务搭建
2. 创建keystone数据库，并进行配置

mysql -uroot -p //进入mariadb数据库

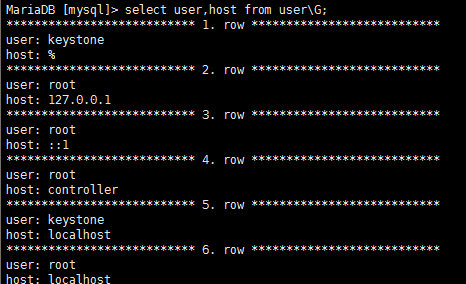
MariaDB [mysql]> create database keystone; //创建keystone数据库

MariaDB [mysql]> grant all privileges on keystone.\* to 'keystone'@'localhost' identified by 'keystone';

MariaDB [mysql]> grant all privileges on keystone.\* to 'keystone'@'%' identified by 'keystone';创建keystone数据库授权用户keystone;



MariaDB [mysql]> select user,host from user\G; //查看授权的用户



2.安装并配置keystone服务

[root@controller /]# yum install openstack-keystone httpd mod\_wsgi

编辑/etc/keystone/keystone.conf文件并完成以下操作：

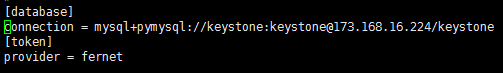
在该[database]部分中，配置数据库访问：

[database]

connection = mysql+pymysql://keystone:keystone@173.168.16.224/keystone在该[token]部分中，配置Fernet令牌提供程序

[token]

provider = fernet



初始化身份认证服务数据库

[root@controller /]# su -s /bin/sh -c "keystone-manage db\_sync" keystone



**注意：这里配置文件的database的格式一定要配置正确，否则会报错或者同步失败**

初始化Fernet密钥存储库

[root@controller /]# keystone-manage fernet\_setup --keystone-user keystone --keystone-group keystone

[root@controller /]# keystone-manage credential\_setup --keystone-user keystone --keystone-group keystone



创建身份服务api端点

[root@controller /]# keystone-manage bootstrap --bootstrap-password admin --bootstrap-admin-url http://controller:35357/v3/ --bootstrap-internal-url http://controller:5000/v3/ --bootstrap-public-url http://controller.5500/v3/ --bootstrap-region-id RegionOne



3.配置Apache HTTP服务器

编辑该/etc/httpd/conf/httpd.conf文件并配置该 ServerName选项以引用控制器节点

echo ServerName controller >> /etc/httpd/conf/httpd.conf

为mod\_wsgi模块添加配置文件,创建一个软连接到/etc/httpd/conf.d/

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

4.启动httpd服务且设置开机自启

[root@controller /]# systemctl enable httpd

[root@controller /]# systemctl start httpd

5.配置管理账户

export OS\_USERNAME=admin

export OS\_PASSWORD=admin

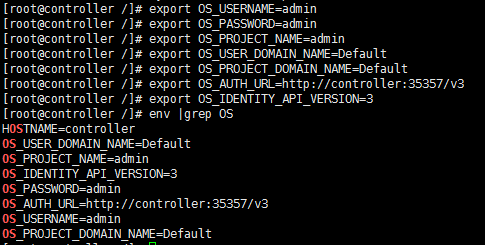
export OS\_PROJECT\_NAME=admin

export OS\_USER\_DOMAIN\_NAME=Default

export OS\_PROJECT\_DOMAIN\_NAME=Default

export OS\_AUTH\_URL=http://controller:35357/v3

export OS\_IDENTITY\_API\_VERSION=3



6.创建一个域，项目，用户和角色

身份服务为每个OpenStack服务提供身份验证服务。身份验证服务使用域，项目，用户和角色的组合。

创建服务项目

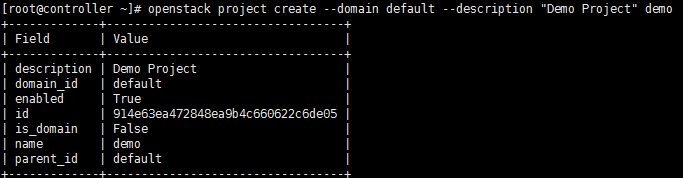
[root@controller ~]# openstack project create --domain default --description "Service Project" service



**注意：这里可能会出现‘Missing value auth-url required for auth plugin password’的报错，翻译报错之后尝试重新执行第五步，创建成功**

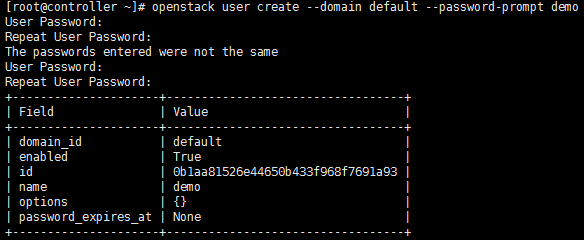
创建demo项目

[root@controller ~]# openstack project create --domain default --description "Demo Project" demo



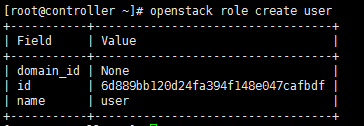
创建demo用户

[root@controller ~]# openstack user create --domain default --password-prompt demo



创建user角色

[root@controller ~]# openstack role create user

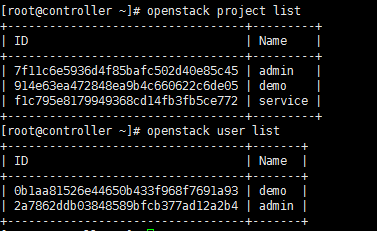


将user角色添加到demo项目和demo用户中

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

[root@controller ~]# openstack project list //查看当前项目列表

[root@controller ~]# openstack user list //查看当前用户列表



admin的项目和用户是上面的步骤就创建好了的

7.验证keystone操作

取消设置临时 变量OS\_AUTH\_URL和OS\_PASSWORD环境变量

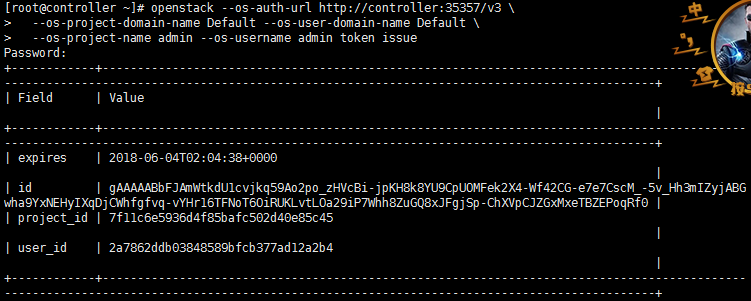
[root@controller ~]# unset OS\_AUTH\_URL OS\_PASSWORD

作为admin用户，请求身份验证令牌

[root@controller ~]# 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

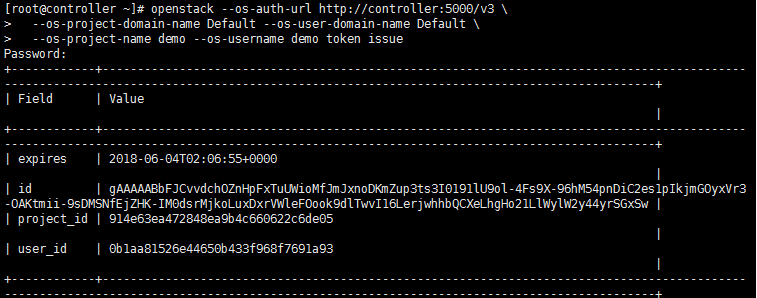


作为demo用户，请求身份验证令牌

[root@controller ~]# openstack --os-auth-url http://controller:5000/v3 \

> --os-project-domain-name Default --os-user-domain-name Default \

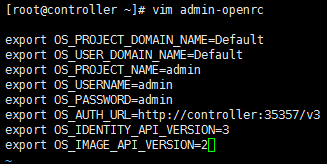
> --os-project-name demo --os-username demo token issue



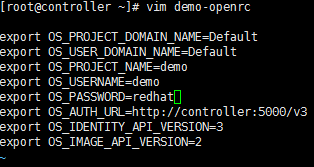
8.创建OpenStack客户端环境脚本

创建客户端环境的脚本admin和demo项目和用户

创建并编辑admin-openrc文件并添加以下内容：



创建并编辑demon-openrc文件并添加以下内容：



9.使用脚本

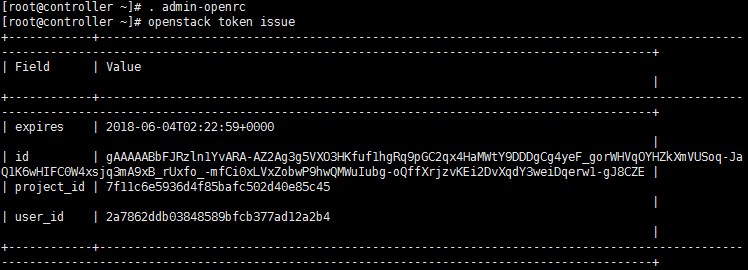
要以特定项目和用户身份运行客户端，只需在运行客户端环境脚本之前加载相关的客户端环境脚本即可。例如：

加载admin-openrc文件以使用Identity服务的位置以及admin项目和用户凭据填充环境变量：

[root@controller ~]# . admin-openrc

请求身份验证令牌：

[root@controller ~]# openstack token issue



至此OpenStack的keystone服务搭建完成

1. 图像服务（glance）

Image服务（glance）使用户能够发现，注册和检索虚拟机映像。它提供了一个 [REST](https://docs.openstack.org/glance/pike/glossary.html#term-restful) API，使您可以查询虚拟机图像元数据并检索实际图像。您可以将通过映像服务提供的虚拟机映像存储在各种位置，从简单的文件系统到对象存储系统（如OpenStack对象存储）。

OpenStack Image服务是基础架构即服务（IaaS）的核心。它接受磁盘或服务器映像的API请求，以及来自最终用户或OpenStack Compute组件的元数据定义。它还支持在各种存储库类型（包括OpenStack对象存储）上存储磁盘或服务器映像。

许多定期进程在OpenStack Image服务上运行以支持缓存。复制服务通过集群确保一致性和可用性。其他定期流程包括审计员，更新者和收割者。

OpenStack Image服务包含以下组件：

**一目了然的API**

接受图像API调用以进行图像发现，检索和存储。

**一目了然的注册表**

存储，处理和检索有关图像的元数据。元数据包括大小和类型等项目。

**数据库**

存储图像元数据，您可以根据自己的喜好选择数据库。大多数部署使用MySQL或SQLite。

**图像文件的存储库**

支持各种存储库类型，包括常规文件系统（或安装在glance-api控制器节点上的任何文件系统），Object Storage，RADOS块设备，VMware数据存储和HTTP。请注意，某些存储库仅支持只读用法。

**元数据定义服务**

用于供应商，管理员，服务和用户的通用API来有意义地定义他们自己的定制元数据。此元数据可用于不同类型的资源，如图像，工件，体积，风味和聚合。定义包括新属性的关键字，描述，约束和它可以关联的资源类型

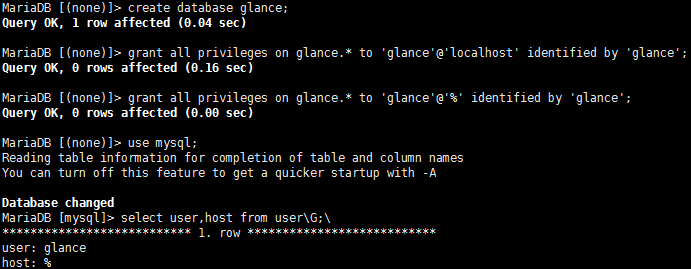
1. 创建glance数据库并授权

[root@controller ~]# mysql -uroot -p //进入数据库

MariaDB [(none)]> create database glance; //创建glance数据库

MariaDB [(none)]> grant all privileges on glance.\* to 'glance'@'localhost' identified by 'glance';

MariaDB [(none)]> grant all privileges on glance.\* to 'glance'@'%' identified by 'glance'; //创建glance数据库授权用户

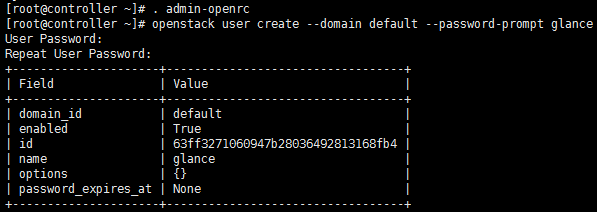


1. 创建glance用户和服务

[root@controller ~]# . admin-openrc //加载环境变量

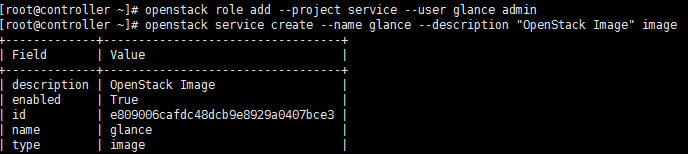
[root@controller ~]# openstack

ate --domain default --password-prompt glance //创建glance用户



[root@controller ~]# openstack role add --project service --user glance admin //将admin角色添加到glance用户和service项目中

[root@controller ~]# openstack service create --name glance --description "OpenStack Image" image //创建glance服务实体

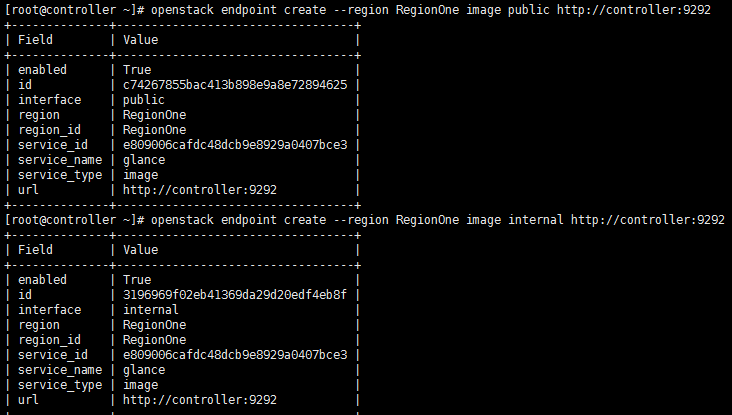


1. 创建图像服务api端点

[root@controller ~]# openstack endpoint create --region RegionOne image public <http://controller:9292>

[root@controller ~]# openstack endpoint create --region RegionOne image internal <http://controller:9292>

[root@controller ~]# openstack endpoint create --region RegionOne image admin <http://controller:9292>



1. 安装和配置的部件

[root@controller ~]# yum install openstack-glance //安装软件包

编辑/etc/glance/glance-api.conf文件并完成以下操作：

在该[database]部分中，配置数据库访问：

[database]

connection = mysql+pymysql://glance:glance@173.168.16.224/glance

在[keystone\_authtoken]和[paste\_deploy]部分中，配置身份服务访问：

[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

[paste\_deploy]

flavor = keystone

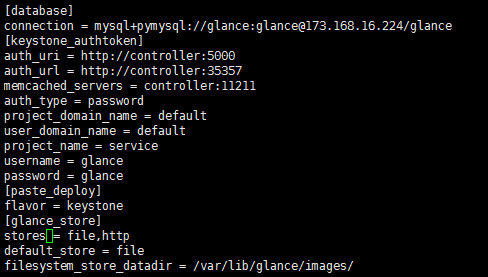
在该[glance\_store]部分中，配置本地文件系统存储和映像文件的位置

[glance\_store]

stores = file,http

default\_store = file

filesystem\_store\_datadir = /var/lib/glance/images/



编辑/etc/glance/glance-registry.conf文件并完成以下操作：

在该[database]部分中，配置数据库访问：

[database]

connection = mysql+pymysql://glance:glance@controller/glance

在[keystone\_authtoken]和[paste\_deploy]部分中，配置身份服务访问：

[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

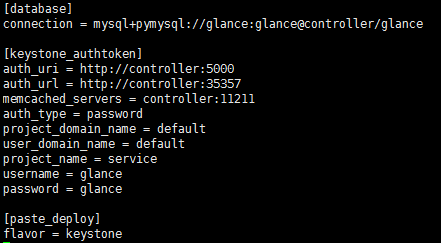
project\_name = service

username = glance

password = glance

[paste\_deploy]

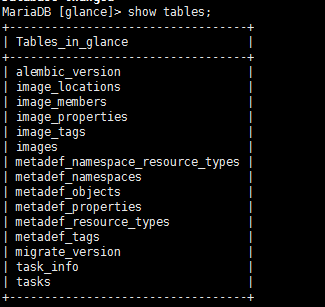
flavor = keystone



5.初始化glance数据库

[root@controller ~]# su -s /bin/sh -c "glance-manage db\_sync" glance //忽略输出的任何信息

[root@controller ~]# mysql -uglance -pglance -e "use glance;show tables;"|wc -l //验证是否同步成功



6.启动映像服务并将其配置为在系统引导时启动

[root@controller ~]# systemctl enable openstack-glance-api openstack-glance-registry

[root@controller ~]# systemctl start openstack-glance-api openstack-glance-registry

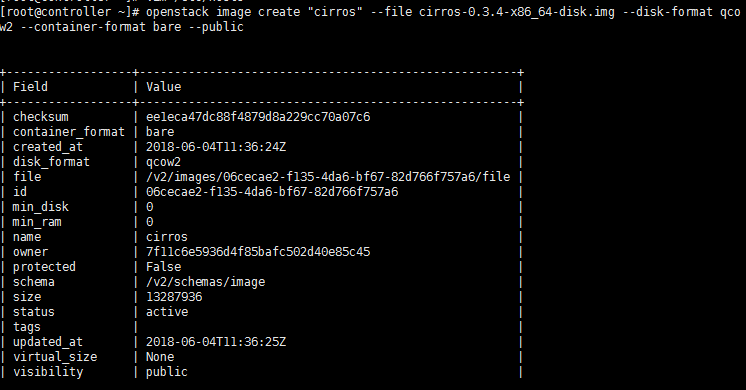
7.验证glance服务

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。

[root@controller ~]# . admin-openrc //加载环境变量

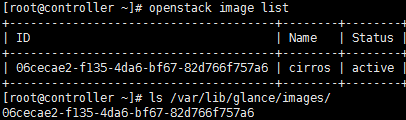
[root@controller ~]# wget <http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img> //下载源镜像

[root@controller ~]# openstack image create "cirros" --file cirros-0.3.4-x86\_64-disk.img --disk-format qcow2 --container-format bare –public //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，使所有项目可以访问该镜像



**注意：这里我一直报500http内部错误，后来从谷歌上看到有人和我有一样的报错，原来是hosts文件里controller少了一个’l’**

[root@controller ~]# openstack image list //查看镜像列表



至此glance服务搭建成功

1. 安装和配置代码为nova的Compute服务。

OpenStack Compute包含以下区域及其组件：

**nova-api 服务**

接受并响应最终用户计算API调用。该服务支持OpenStack Compute API，Amazon EC2 API以及特殊用户执行管理操作的特殊管理API。它执行一些策略并启动大多数编排活动，例如运行实例。

**nova-api-metadata 服务**

接受来自实例的元数据请求。nova-api-metadata通常在nova-network 安装多主机模式下运行时使用该服务。有关详细信息，请参阅 计算管理员指南中的[元数据服务](https://docs.openstack.org/nova/pike/admin/networking-nova.html#metadata-service)。

**nova-compute 服务**

通过管理程序API创建和终止虚拟机实例的工作守护程序。例如：

* 用于XenServer / XCP的XenAPI
* KVM或QEMU的libvirt
* 适用于VMware的VMwareAPI

处理相当复杂。基本上，守护进程接受来自队列的动作并执行一系列系统命令，例如启动KVM实例并更新其在数据库中的状态。

**nova-placement-api 服务**

跟踪每个提供商的库存和使用情况。有关详情，请参阅[Placement API](https://docs.openstack.org/developer/nova/placement.html)。

**nova-scheduler 服务**

从队列中获取虚拟机实例请求，并确定它在哪个计算服务器主机上运行。

**nova-conductor**

调解nova-compute服务和数据库之间的交互。它消除了由nova-compute服务直接访问云数据库的情况 。该nova-conductor模块水平缩放。但是，请勿将其部署到nova-compute运行服务的节点上。有关更多信息，请参阅[配置选项中](https://docs.openstack.org/nova/pike/configuration/config.html)的conductor部分 。

**nova-consoleauth 守护进程**

为控制台代理提供的用户授权令牌。见 nova-novncproxy和nova-xvpvncproxy。此服务必须运行以使控制台代理正常工作。您可以在群集配置中针对单个nova-consoleauth服务运行任一类型的代理。有关信息，请参阅[关于nova-consoleauth](https://docs.openstack.org/nova/pike/admin/remote-console-access.html#about-nova-consoleauth)。

**nova-novncproxy 守护进程**

提供通过VNC连接访问正在运行的实例的代理。支持基于浏览器的novnc客户端。

**nova-spicehtml5proxy 守护进程**

提供通过SPICE连接访问正在运行的实例的代理。支持基于浏览器的HTML5客户端。

**nova-xvpvncproxy 守护进程**

提供通过VNC连接访问正在运行的实例的代理。支持OpenStack特定的Java客户端。

**队列**

守护进程之间传递消息的中心集线器。通常用[RabbitMQ](https://www.rabbitmq.com/)实现 ，也可以用另一个AMQP消息队列实现，例如[ZeroMQ](http://www.zeromq.org/)。

**SQL数据库**

存储云基础架构的大部分构建时间和运行时状态，其中包括：

* 可用的实例类型
* 正在使用的实例
* 可用的网络
* 项目

理论上，OpenStack Compute可以支持SQLAlchemy支持的任何数据库。通用数据库是用于测试和开发工作的SQLite3，MySQL，MariaDB和PostgreSQL。

A.在控制器节点上安装和配置代码为nova的Compute服务。

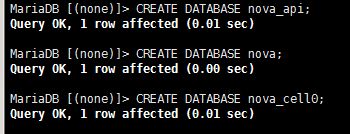
1.创建数据库并授权

[root@controller ~]# mysql -uroot -p //进入数据库

MariaDB [(none)]> CREATE DATABASE nova\_api; //创建nova\_api数据库

MariaDB [(none)]> CREATE DATABASE nova; //创建nova数据库

MariaDB [(none)]> CREATE DATABASE nova\_cell0; //创建nova\_ce110数据库



MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova\_api.\* TO 'nova'@'localhost' IDENTIFIED BY 'nova'//对nova\_api数据库进行用户授权

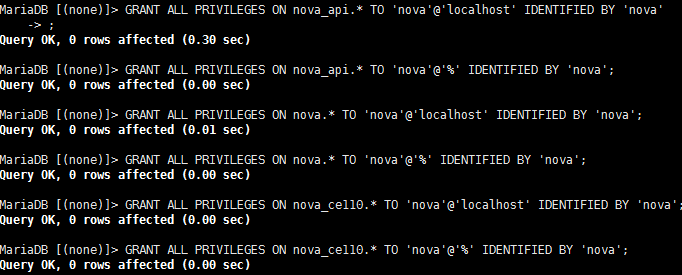
MariaDB [mysql]> GRANT ALL PRIVILEGES ON nova\_api.\* TO 'nova'@'%' IDENTIFIED BY 'nova';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.\* TO 'nova'@'localhost' IDENTIFIED BY 'nova'; //对nova数据库进行用户授权

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

MariaDB [mysql]> GRANT ALL PRIVILEGES ON nova\_ce110.\* TO 'nova'@'localhost' IDENTIFIED BY 'nova'; //对nova\_ce110数据库进行用户授权

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova\_ce110.\* TO 'nova'@'%' IDENTIFIED BY 'nova';



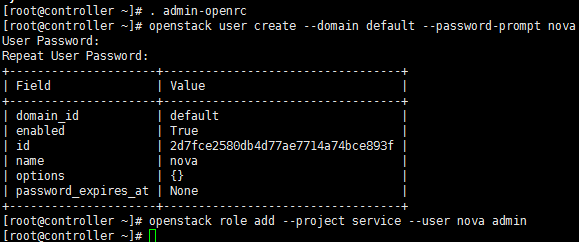
2.加载环境变量

[root@controller ~]# . admin-openrc

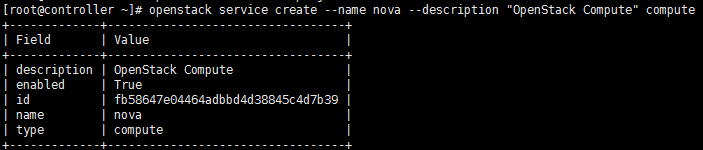
3.创建计算服务凭据

[root@controller ~]# openstack user create --domain default --password-prompt nova //创建nova用户

[root@controller ~]# openstack role add --project service --user nova admin //将admin角色添加到nova用户



[root@controller ~]# openstack service create --name nova --description "OpenStack Compute" compute //创建服务实体

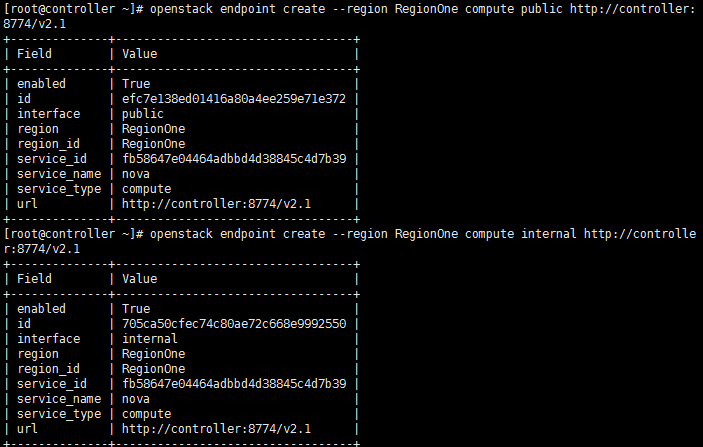


4. 创建计算服务API端点

[root@controller ~]# openstack endpoint create --region RegionOne compute public <http://controller:8774/v2.1> //

[root@controller ~]# openstack endpoint create --region RegionOne compute internal <http://controller:8774/v2.1>

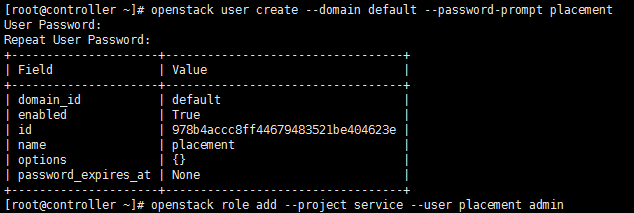
[root@controller ~]# openstack endpoint create --region RegionOne compute admin <http://controller:8774/v2.1>



1. 创建位置服务用户

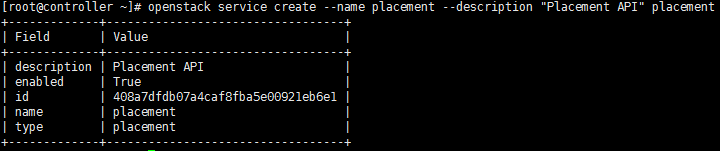
[root@controller ~]# openstack user create --domain default --password-prompt placement

[root@controller ~]# openstack role add --project service --user placement admin //将位置用户添加到具有管理员角色的服务项目中



1. 在服务目录中创建Placement API条目

[root@controller ~]# openstack service create --name placement --description "Placement API" placement

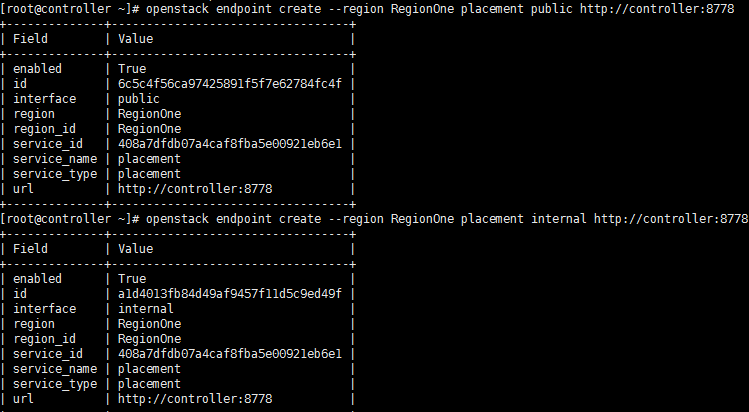


1. 创建Placement API服务端点

[root@controller ~]# openstack endpoint create --region RegionOne placement public [http://controller:8778](http://controller:87780)

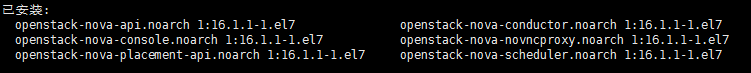
[root@controller ~]# openstack endpoint create --region RegionOne placement internal <http://controller:8778>

[root@controller ~]# openstack endpoint create --region RegionOne placement admin <http://controller:8778>



1. 安装并配置nova软件

yum install openstack-nova-api openstack-nova-conductor openstack-nova-console openstack-nova-novncproxy openstack-nova-scheduler openstack-nova-placement-api -y //安装nova相关组件



编辑/etc/nova/nova.conf文件并完成以下操作：

在该[DEFAULT]部分中，仅启用计算API和元数据API：

enable\_apis = osapi\_compute,metadata

在[api\_database]和[database]部分中，配置数据库访问：

[api\_database]

connection = mysql+pymysql://nova:nova@controller/nova\_api

[database]

connection = mysql+pymysql://nova:nova@controller/nova

在该[DEFAULT]部分中，配置RabbitMQ消息队列访问

transport\_url = rabbit://openstack:openstack@controller

在[api]和[keystone\_authtoken]部分中，配置身份服务访问

[api]

auth\_strategy = keystone

[keystone\_authtoken]

auth\_uri = http://controller:5000

auth\_url = http://controller:35357

memcached\_servers = controller:11211

auth\_type = password

project\_domain\_name = default

user\_domain\_name = default

project\_name = service

username = nova

password = nova

在该[DEFAULT]部分中，将该my\_ip选项配置为使用控制器节点的管理接口IP地址：

my\_ip = 173.168.16.224

在本[DEFAULT]节中，启用对网络服务的支持：

use\_neutron = True

firewall\_driver = nova.virt.firewall.NoopFirewallDriver

在本[vnc]节中，将VNC代理配置为使用控制器节点的管理接口IP地址：

[vnc]

enabled = true

vncserver\_listen = $my\_ip

vncserver\_proxyclient\_address = $my\_ip

在该[glance]部分中，配置Image Service API的位置：

[glance]

api\_servers = http://controller:9292

在该[oslo\_concurrency]部分中，配置锁定路径

[oslo\_concurrency]

lock\_path = /var/lib/nova/tmp

在该[placement]部分中，配置Placement API：

[placement]

os\_region\_name = RegionOne

project\_domain\_name = Default

project\_name = service

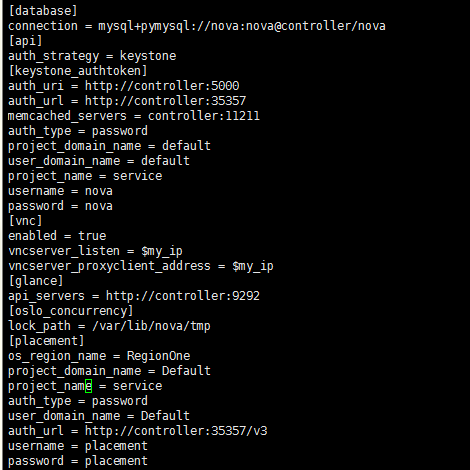
auth\_type = password

user\_domain\_name = Default

auth\_url = http://controller:35357/v3

username = placement

password = placement



由于软件包bug,您必须启用对Placement API的访问权限，方法是将以下配置添加到 /etc/httpd/conf.d/00-nova-placement-api.conf：

<Directory /usr/bin>

<IfVersion >= 2.4>

Require all granted

</IfVersion>

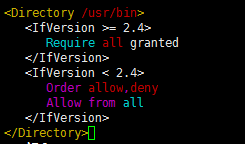
<IfVersion < 2.4>

Order allow,deny

Allow from all

</IfVersion>

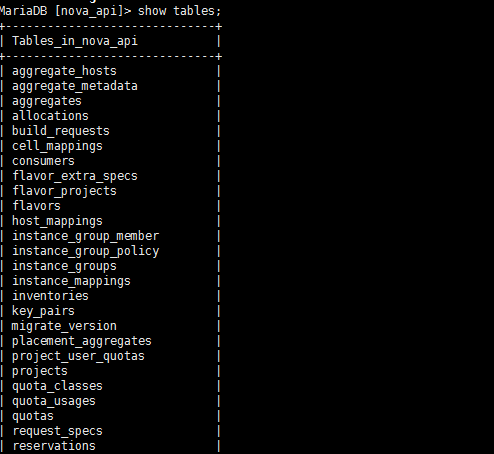
</Directory>



[root@controller ~]# systemctl restart httpd //重启httpd服务

9.初始化nova数据库

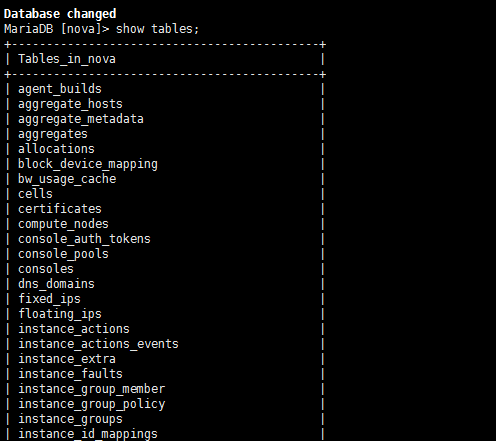
[root@controller ~]# su -s /bin/sh -c "nova-manage api\_db sync" nova //初始化nova\_api数据库



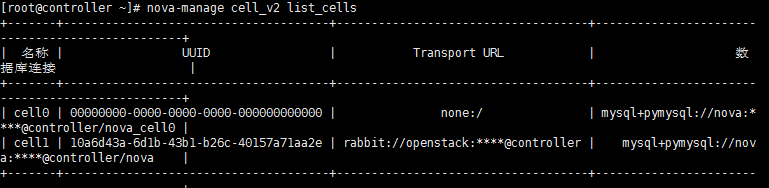
[root@controller ~]# su -s /bin/sh -c "nova-manage cell\_v2 map\_cell0" nova //注册ce110数据库

[root@controller ~]# su -s /bin/sh -c "nova-manage cell\_v2 create\_cell --name=cell1 --verbose" nova //创建ce110的单元格

[root@controller ~]# su -s /bin/sh -c "nova-manage db sync" nova //初始化nova数据库，忽略其中的输出信息

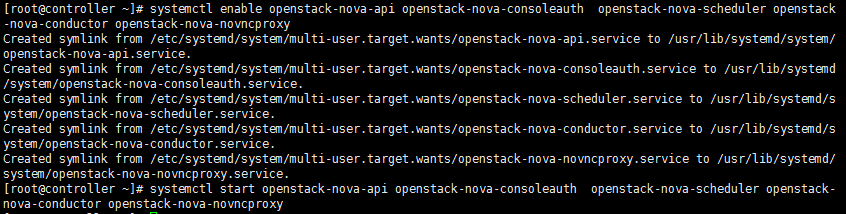


[root@controller ~]# nova-manage cell\_v2 list\_cells //验证nova cell0和cell1是否正确注册



[root@controller ~]# systemctl enable openstack-nova-api openstack-nova-consoleauth openstack-nova-scheduler openstack-nova-conductor openstack-nova-novncproxy

[root@controller ~]# systemctl start openstack-nova-api openstack-nova-consoleauth openstack-nova-scheduler openstack-nova-conductor openstack-nova-novncproxy //启动计算服务并将其配置为在系统引导时启动



B．在计算节点上安装和配置代码为nova的Compute服务。

1.在计算节点上安装和配置

yum install openstack-nova-compute //安装软件包，这里出现一个报错缺少一个rpm包，某位大佬丢给我一个rpm包地址得以解决

rpm -ivh https://repos.fedorapeople.org/repos/openstack/openstack-pike/rdo-release-pike-1.noarch.rpm

编辑/etc/nova/nova.conf文件并完成以下操作：

在该[DEFAULT]部分中，仅启用计算API和元数据API和配置RabbitMQ消息队列访问：：

[DEFAULT]

enabled\_apis = osapi\_compute,metadata

transport\_url = rabbit://openstack:openstack@controller

在[api]和[keystone\_authtoken]部分中，配置身份服务访问

[api]

auth\_strategy = keystone

[keystone\_authtoken]

auth\_uri = http://controller:5000

auth\_url = http://controller:35357

memcached\_servers = controller:11211

auth\_type = password

project\_domain\_name = default

user\_domain\_name = default

project\_name = service

username = nova

password = nova

在该[DEFAULT]部分中，配置my\_ip选项

my\_ip = 173.168.16.225

在本[DEFAULT]节中，启用对网络服务的支持：

use\_neutron = True

firewall\_driver = nova.virt.firewall.NoopFirewallDriver

在本[vnc]节中，启用并配置远程控制台访问：

[vnc]

enabled = True

vncserver\_listen = 0.0.0.0

vncserver\_proxyclient\_address = $my\_ip

novncproxy\_base\_url = <http://controller:6080/vnc_auto.html>

在该[glance]部分中，配置Image Service API的位置：

[glance]

api\_servers = <http://controller:9292>

在该[oslo\_concurrency]部分中，配置锁定路径：

[oslo\_concurrency]

lock\_path = /var/lib/nova/tmp

在该[placement]部分中，配置Placement API：

[placement]

os\_region\_name = RegionOne

project\_domain\_name = Default

project\_name = service

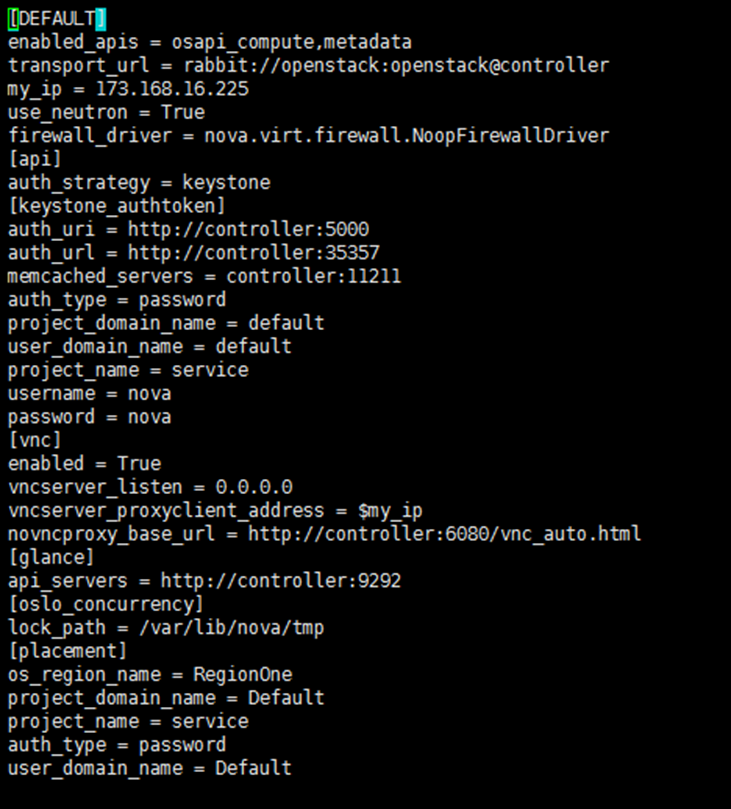
auth\_type = password

user\_domain\_name = Default

auth\_url = http://controller:35357/v3

username = placement

password = placement



2.完成安装

egrep -c '(vmx|svm)' /proc/cpuinfo //检查计算节点是否支持虚拟机的硬件加速如果返回值不为0，说明支持，计算节点不支持硬件加速，您必须配置libvirt为使用QEMU而不是KVM



按如下方式编辑文件中的[libvirt]部分/etc/nova/nova.conf：

[libvirt]

virt\_type = qemu



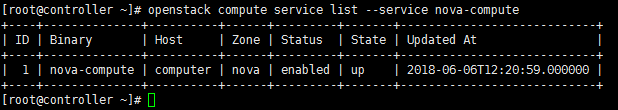
[root@computer ~]# systemctl enable libvirtd.service openstack-nova-compute.service

[root@computer ~]# systemctl start libvirtd.service openstack-nova-compute.service //启动Compute服务（包括其依赖项）并将其配置为在系统引导时自动启动，这里出现一个小插曲，防火墙开着，服务一直起不来

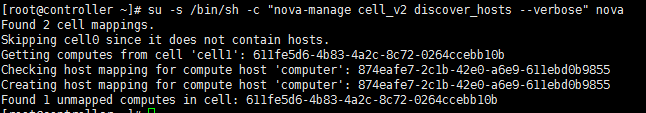
3.将计算节点添加到单元数据库[¶](https://docs.openstack.org/nova/pike/install/compute-install-rdo.html#add-the-compute-node-to-the-cell-database)（在控制节点运行）

[root@controller ~]# . admin-openrc //加载环境变量

[root@controller ~]# openstack compute service list --service nova-compute //确认数据库中是否有计算主机



[root@controller ~]# su -s /bin/sh -c "nova-manage cell\_v2 discover\_hosts --verbose" nova //发现计算主机



当您添加新的计算节点时，您必须在控制器节点上运行以注册这些新的计算节点。或者，您可以在以下位置设置适当的间隔： /etc/nova/nova.conf



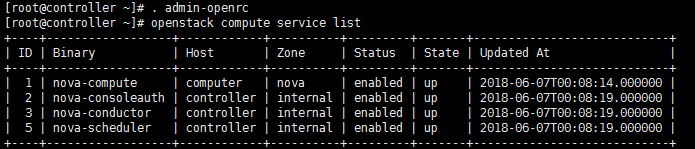
4.验证computer服务的操作（控制节点运行）

加载环境变量

[root@controller ~]# . admin-openrc

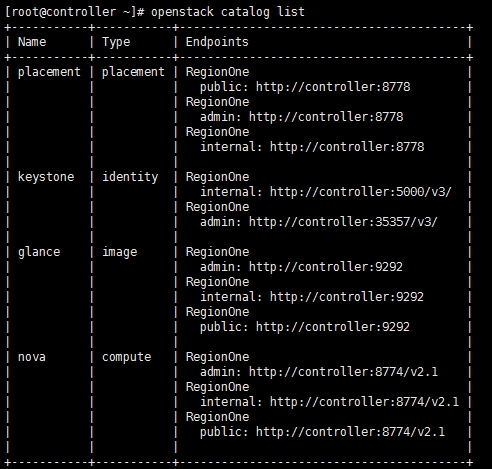
列出服务组件以验证每个进程的成功启动和注册

[root@controller ~]# openstack compute service list



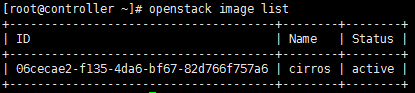
列出身份服务中的API端点以验证与身份服务的连接

[root@controller ~]# openstack catalog list



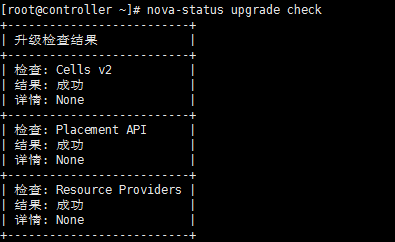
列出Image服务中的图像以验证与Image服务的连接性：

[root@controller ~]# openstack image list



检查cells和placement API是否成功运行：

[root@controller ~]# nova-status upgrade check



至此，nova服务部分已经安装完成了

1. 网络服务NeuTron

A.安装并配置控制器节点

1.配置neutron数据库

[root@controller ~]# mysql -uroot -p //进入mariadb数据库

MariaDB [(none)]> CREATE DATABASE neutron; //创建neutron数据库

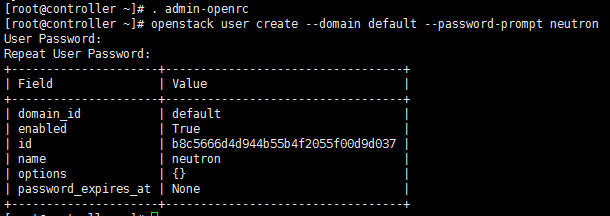
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.\* TO 'neutron'@'localhost' IDENTIFIED BY 'neutron';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.\* TO 'neutron'@'%' IDENTIFIED BY 'neutron'; //创建neutron的授权用户

2.创建服务凭据

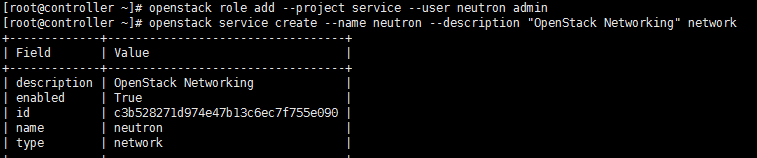
[root@controller ~]# . admin-openrc //加载环境变量

[root@controller ~]# openstack user create --domain default --password-prompt neutron //创建neutron用户



[root@controller ~]# openstack role add --project service --user neutron admin //将admin角色添加到neutron用户

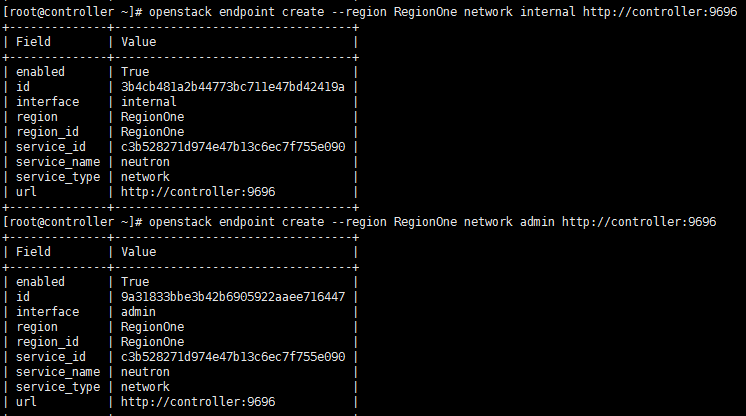
[root@controller ~]# openstack service create --name neutron --description "OpenStack Networking" network //创建neutron服务实体



[root@controller ~]# openstack endpoint create --region RegionOne network public <http://controller:9696>

[root@controller ~]# openstack endpoint create --region RegionOne network internal <http://controller:9696>

[root@controller ~]# openstack endpoint create --region RegionOne network admin <http://controller:9696> //创建网络服务的api端点



3.安装并配置neutron网络，这里有两种网络选项，我选择的是提供商网络

[root@controller ~]# yum install openstack-neutron openstack-neutron-ml2 openstack-neutron-linuxbridge ebtables //在控制节点安装neutron相关组件

编辑/etc/neutron/neutron.conf文件并完成以下操作：

在该[database]部分中，配置数据库访问：

[database]

connection = mysql+pymysql://neutron:neutron@controller/neutron

在本[DEFAULT]节中，启用Modular Layer 2（ML2）插件并禁用其他插件：

core\_plugin = ml2

service\_plugins = //等号后面什么都不加就是禁用其他插件

在该[DEFAULT]部分中，配置RabbitMQ 消息队列访问：

transport\_url = rabbit://openstack:openstack@controller

在[DEFAULT]和[keystone\_authtoken]部分中，配置身份服务访问：

[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

在[DEFAULT]和[nova]部分中，配置网络以通知计算网络拓扑更改：

[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

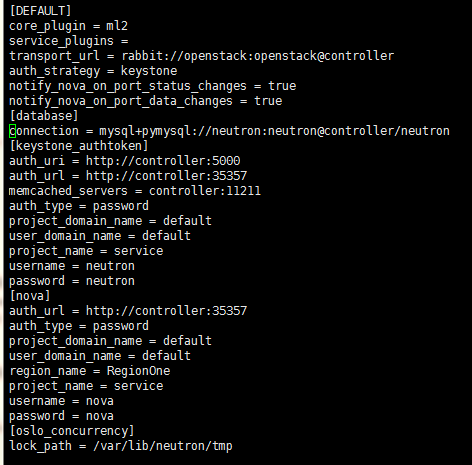
password = nova

在该[oslo\_concurrency]部分中，配置锁定路径：

[oslo\_concurrency]

lock\_path = /var/lib/neutron/tmp

完整配置如下图所示



ML2插件使用Linux桥接机制为实例构建第2层（桥接和交换）虚拟网络基础结构。

编辑/etc/neutron/plugins/ml2/ml2\_conf.ini文件并完成以下操作：

在本[ml2]节中，启用flat和VLAN网络:

[ml2]

type\_drivers = flat,vlan

在本[ml2]节中，禁用自助服务网络：

tenant\_network\_types =

在本[ml2]节中，启用Linux桥接机制：

mechanism\_drivers = linuxbridge

在该[ml2]部分中，启用端口安全扩展驱动程序：

extension\_drivers = port\_security

在本[ml2\_type\_flat]节中，将提供者虚拟网络配置为扁平网络：

[ml2\_type\_flat]

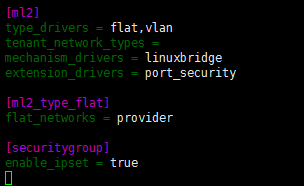
flat\_networks = provider

在本[securitygroup]节中，使用ipset来提高安全组规则的效率：

[securitygroup]

enable\_ipset = true

完整配置如下图所示：



Linux网桥代理为实例构建第2层（桥接和交换）虚拟网络基础架构并处理安全组。

编辑/etc/neutron/plugins/ml2/linuxbridge\_agent.ini文件并完成以下操作：

在本[linux\_bridge]节中，将提供者虚拟网络映射到提供者物理网络接口：

[linux\_bridge]

physical\_interface\_mappings = provider:ens33 //将ens33替换成自己本身的网卡接口

在该[vxlan]部分中，禁用VXLAN覆盖网络：

[vxlan]

enable\_vxlan = false

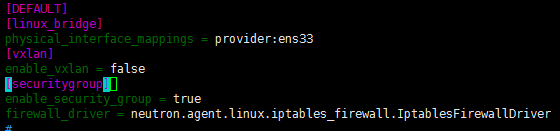
在本[securitygroup]节中，启用安全组并配置Linux网桥iptables防火墙驱动程序：

[securitygroup]

enable\_security\_group = true

firewall\_driver = neutron.agent.linux.iptables\_firewall.IptablesFirewallDriver

完整配置如下图所示：



DHCP代理为虚拟网络提供DHCP服务。

编辑/etc/neutron/dhcp\_agent.ini文件并完成以下操作：

在本[DEFAULT]节中，配置Linux网桥接口驱动程序，Dnsmasq DHCP驱动程序，并启用隔离的元数据，以便提供商网络上的实例可以通过网络访问元数据：

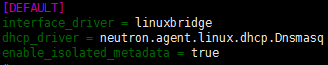
[DEFAULT]

interface\_driver = linuxbridge

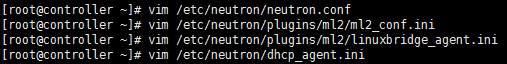
dhcp\_driver = neutron.agent.linux.dhcp.Dnsmasq

enable\_isolated\_metadata = true

完整配置如下图所示：



在这里，我们一共修改了四个配置文件，分别是



4.配置元数据代理

元数据代理为实例提供配置信息，例如凭据。

编辑/etc/neutron/metadata\_agent.ini文件并完成以下操作：

在该[DEFAULT]部分中，配置元数据主机和共享密钥：

[DEFAULT]

nova\_metadata\_host = controller

metadata\_proxy\_shared\_secret = neutron

完整配置如下图所示：



5.配置nova使用neutron服务

编辑/etc/nova/nova.conf文件并执行以下操作：

在该[neutron]部分中，配置访问参数，启用元数据代理并配置秘钥：

[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

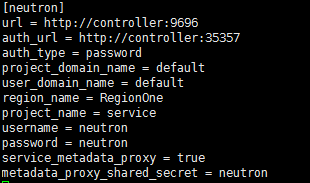
username = neutron

password = neutron

service\_metadata\_proxy = true

metadata\_proxy\_shared\_secret = neutron

完整配置如下图所示：



6.完成控制节点的neutron安装

网络服务初始化脚本需要/etc/neutron/plugin.ini指向ML2插件配置文件的符号链接/etc/neutron/plugins/ml2/ml2\_conf.ini。如果此符号链接不存在，请使用以下命令创建它：

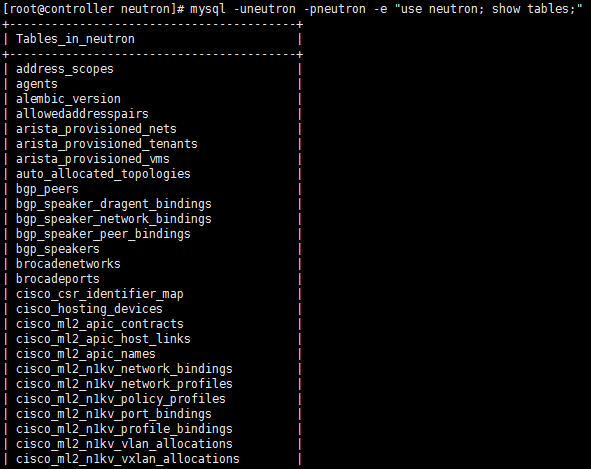
[root@controller ~] ln -s /etc/neutron/plugins/ml2/ml2\_conf.ini /etc/neutron/plugin.ini

初始化neutron数据库

[root@controller neutron]# 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

**这里我同步的时候出现了一个报错，猜想应该是配置问题，结果看了一遍又一遍配置文件没找出来错误，后来谷歌关键字得知我把[database]部分的”connection = mysql+pymysql://neutron:neutron@controller/neutron”写到了[DEFAULT]部分，还是不够细心啊**

[root@controller neutron]# mysql -uneutron -pneutron -e "use neutron; show tables;" //查看同步的数据库



重启计算API服务

[root@controller neutron]# systemctl restart openstack-nova-api.service

启动网络服务并将其配置为系统启动时启动

[root@controller ]# systemctl enable neutron-server.service neutron-linuxbridge-agent.service neutron-dhcp-agent.service neutron-metadata-agent.service

[root@controller neutron]# systemctl start neutron-server.service neutron-linuxbridge-agent.service neutron-dhcp-agent.service neutron-metadata-agent.service

B．在计算节点安装配置

1.安装并配置相关组件

[root@computer ~]# yum install openstack-neutron-linuxbridge ebtables ipset

2.配置通用组件

网络通用组件配置包括认证机制，消息队列和插件。

编辑/etc/neutron/neutron.conf文件并完成以下操作：

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

在该[DEFAULT]部分中，配置RabbitMQ 消息队列访问：

[DEFAULT]

transport\_url = rabbit://openstack:openstack@controller

在[DEFAULT]和[keystone\_authtoken]部分中，配置身份服务访问：

[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

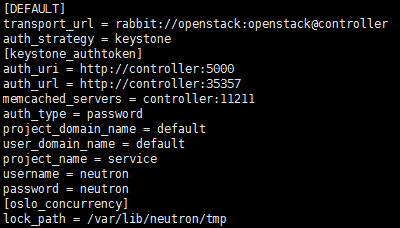
password = neutron

在该[oslo\_concurrency]部分中，配置锁定路径：

[oslo\_concurrency]

lock\_path = /var/lib/neutron/tmp

完整配置如下图所示：



3.配置网络选项

这里与控制器节点选择的相同网络选项，都是提供商网络

Linux网桥代理为实例构建第2层（桥接和交换）虚拟网络基础架构并处理安全组。

编辑/etc/neutron/plugins/ml2/linuxbridge\_agent.ini文件并完成以下操作：

在本[linux\_bridge]节中，将提供者虚拟网络映射到提供者物理网络接口：

[linux\_bridge]

physical\_interface\_mappings = provider:ens33 //将ens33替换为自己的网卡接口

在该[vxlan]部分中，禁用VXLAN覆盖网络：

[vxlan]

enable\_vxlan = false

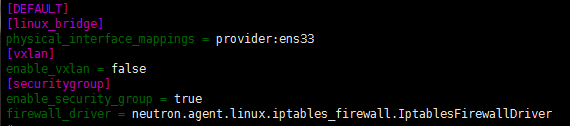
在本[securitygroup]节中，启用安全组并配置Linux网桥iptables防火墙驱动程序：

[securitygroup]

enable\_security\_group = true

firewall\_driver = neutron.agent.linux.iptables\_firewall.IptablesFirewallDriver

完整配置如下图所示：



4.配置计算服务已使用网络服务

编辑/etc/nova/nova.conf文件并完成以下操作：

在该[neutron]部分中，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

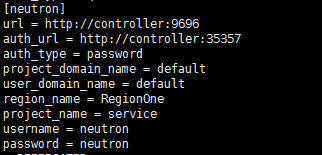
region\_name = RegionOne

project\_name = service

username = neutron

password = neutron

完整配置如下图所示：



5.完成安装

重新启动计算服务：

[root@computer ~]# systemctl restart openstack-nova-compute.service

启动Linux桥代理并将其配置为在系统引导时启动：

[root@computer ~]# systemctl enable neutron-linuxbridge-agent.service

[root@computer ~]# systemctl start neutron-linuxbridge-agent.service

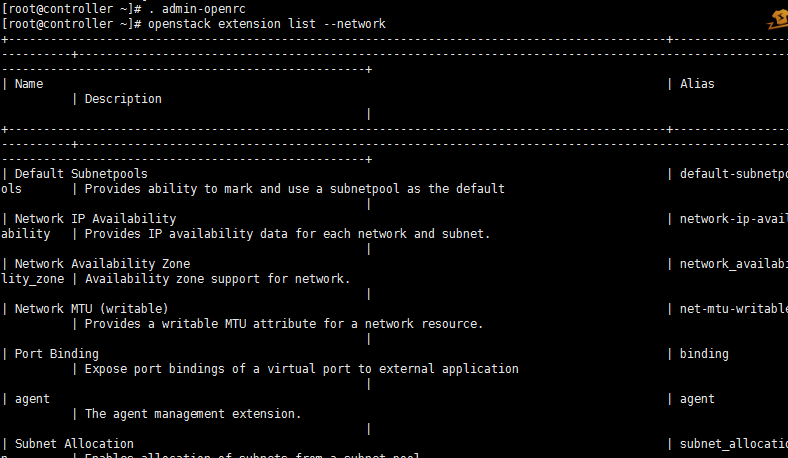
6.验证操作

加载环境变量

[root@controller ~]# . admin-openrc

列出加载的扩展以验证neutron-server过程的成功启动 ：

[root@controller ~]# openstack extension list –network



验证 neutron agent成功

[root@controller ~]# openstack network agent list



至此，neutron服务安装完成

1. 创建一个虚拟机实例
2. 创建虚拟网络

在启动实例之前，您必须创建必要的虚拟网络基础结构。对于提供商网络，实例使用通过第2层（桥接/交换）连接到物理网络基础设施的提供商（外部）网络。该网络包括一个为实例提供IP地址的DHCP服务器。

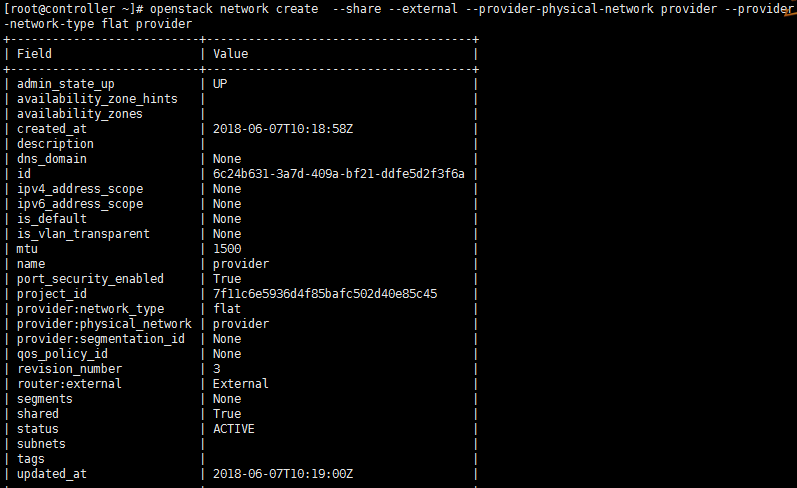
该admin或其他特权用户必须创建此网络，因为它直接连接到物理网络基础结构。

加载环境变量

[root@controller ~]# . admin-openrc

创建网络

[root@controller ~]# openstack network create --share --external --provider-physical-network provider --provider-network-type flat provider //--share选项允许所有项目使用虚拟网络，**--external**选项将虚拟网络定义为外部。如果你想创建一个内部网络，你可以使用**--internal**。默认值是**internal**。



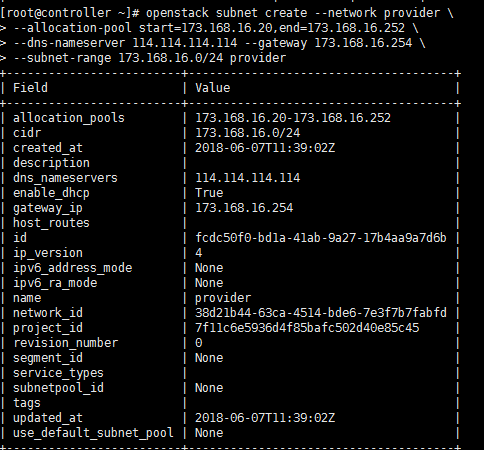
在网络上创建一个子网

[root@controller ~]# openstack subnet create --network provider \

> --allocation-pool start=173.168.16.20,end=173.168.16.252 \

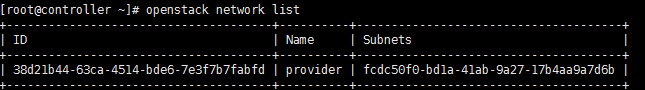
> --dns-nameserver 114.114.114.114 --gateway 173.168.16.254 \

> --subnet-range 173.168.16.0/24 provider //创建了一个地址池为173.168.16.20-173.168.16.252，DNS地址为114.114.114.114，网关为173.168.16.254的子网



查看网络列表

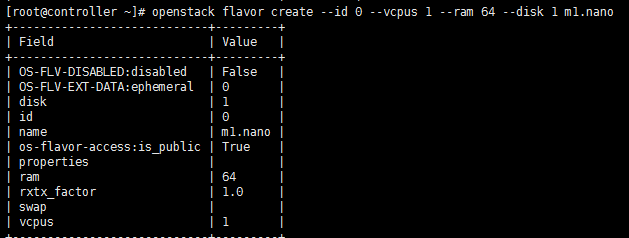
[root@controller ~]# openstack network list



1. 创建m1.nano规格的主机

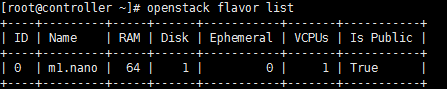
最小的默认规格消耗每个实例512 MB的内存。对于计算节点内存小于4 GB的环境，我们建议创建**m1.nano**每个实例仅需要64 MB 的风格。为了测试目的，请仅将CirrOS图像用于此规格。

[root@controller ~]# openstack flavor create --id 0 --vcpus 1 --ram 64 --disk 1 m1.nano //创建了主机id为0，CPU数量为1，内存64M（默认单位是M），磁盘容量为1G的主机



查看创建的主机

[root@controller ~]# openstack flavor list



1. 创建密钥对

大多数云图像支持[公钥认证，](https://docs.openstack.org/install-guide/common/glossary.html#term-public-key-authentication)而不是传统的密码认证。在启动实例之前，您必须将公钥添加到Compute服务。

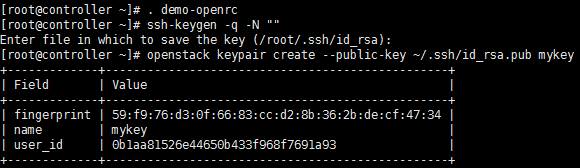
加载demo环境变量

[root@controller ~]# . demo-openrc

生成密钥对并添加公钥

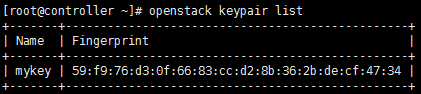
[root@controller ~]# ssh-keygen -q -N "" //创建密钥对

[root@controller ~]# openstack keypair create --public-key ~/.ssh/id\_rsa.pub mykey //添加公钥，或者不用执行上一条命令，直接使用现有的公钥



验证密钥对添加

[root@controller ~]# openstack keypair list



1. 添加安全组规则

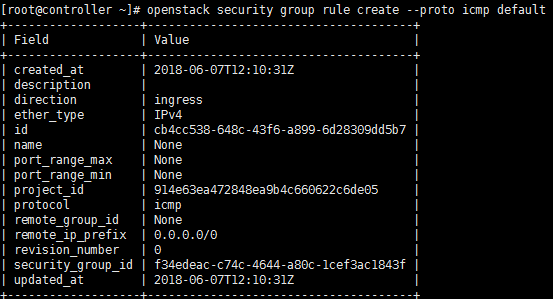
默认情况下，**default**安全组适用于所有实例，并包含拒绝远程访问实例的防火墙规则。对于像CirrOS这样的Linux映像，我们建议至少允许ICMP（ping）和安全shell（SSH）。

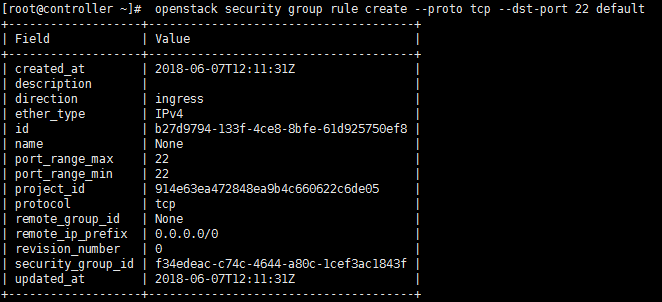
向**default**安全组添加规则，允许icmp（ping）：

[root@controller ~]# openstack security group rule create --proto icmp default

允许安全shell（SSH）访问：

[root@controller ~]# openstack security group rule create --proto tcp --dst-port 22 default





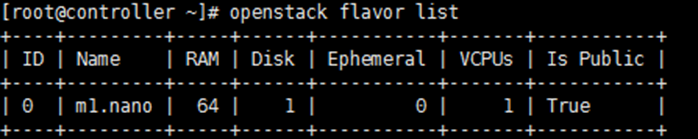
1. 启动一个实例

加载demo环境变量

[root@controller ~]# . demo-openrc

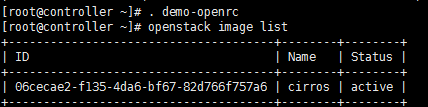
列出创建的实例

[root@controller ~]# openstack flavor list



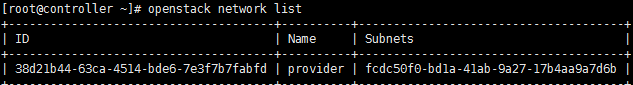
列出可用的镜像

[root@controller ~]# openstack image list



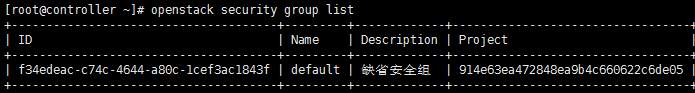
列出可用网络

[root@controller ~]# openstack network list



列出可用的安全组

[root@controller ~]# openstack security group list

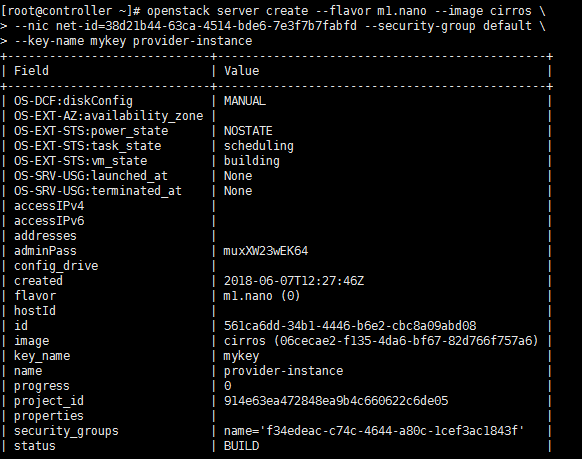


启动实例

[root@controller ~]# openstack server create --flavor m1.nano --image cirros \

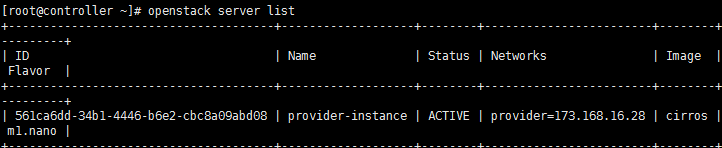
> --nic net-id=38d21b44-63ca-4514-bde6-7e3f7b7fabfd --security-group default \

> --key-name mykey provider-instance //net-id为创建的网络id



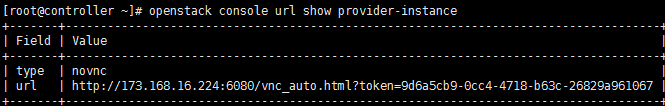
查看实例的状态

[root@controller ~]# openstack server list

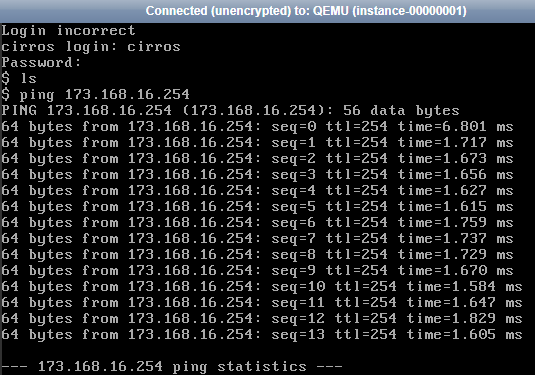


1. 使用虚拟控制台访问实例

[root@controller ~]# openstack console url show provider-instance //可以在web浏览器上输入下面的URL进行vnc连接



验证对提供商物理网络网关的访问权限

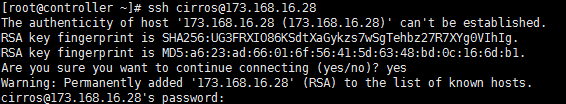


验证对外网的访问



远程连接虚拟机（账号cirros,密码‘cubswin；）’）

[root@controller ~]# ssh [cirros@173.168.16.28](mailto:cirros@173.168.16.28)



1. 安装horizon服务

本节介绍如何在控制器节点上安装和配置仪表板。

仪表板所需的唯一核心服务是身份服务。您可以将仪表板与其他服务结合使用，例如图像服务，计算和网络。您还可以在具有独立服务（如对象存储）的环境中使用仪表板。

本节假定使用Apache HTTP服务器和Memcached服务正确安装，配置和操作Identity Service。

1. 安装软件包

[root@controller ~]# yum install openstack-dashboard

1. 编辑配置文件修改配置

编辑 /etc/openstack-dashboard/local\_settings 文件并完成以下操作：

配置仪表板以在controller节点上使用OpenStack服务 ：

OPENSTACK\_HOST = "controller"

允许所有主机访问仪表板：

ALLOWED\_HOSTS = ['\*', 'localhost']

配置memcached会话存储服务：（这里注释掉其他的会话存储配置）

SESSION\_ENGINE = 'django.contrib.sessions.backends.cache'

CACHES = {

'default': {

'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',

'LOCATION': 'controller:11211',

}

}

启用版本为3的认证 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,

}

配置Default为您通过仪表板创建的用户的默认域：

OPENSTACK\_KEYSTONE\_DEFAULT\_DOMAIN = "Default"

配置user为您通过仪表板创建的用户的默认角色：

OPENSTACK\_KEYSTONE\_DEFAULT\_ROLE = "user"

如果您选择了网络选项1，请禁用对三层网络服务的支持：

OPENSTACK\_NEUTRON\_NETWORK = {

'enable\_router': False,

'enable\_quotas': False,

'enable\_ipv6': False,

'enable\_distributed\_router': False,

'enable\_ha\_router': False,

'enable\_lb': False,

'enable\_firewall': False,

'enable\_vpn': False,

'enable\_fip\_topology\_check': False,

}

配置时区

TIME\_ZONE = "Asia/Shanghai"

**注意：对配置文件里添加新配置的同时要对以前相同的配置进行注释，否则会造成重启http服务报错**

1. 完成安装

[root@controller ~]# systemctl restart httpd.service memcached.service //重新启动Web服务器和会话存储服务

1. 验证操作

在浏览器输入173.168.16.224/dashboard访问dashboard

**这里我一直访问不了，查看日志看到Script timed out before returning headers: django.wsgi，百度这条关键字，发现需要修改dashboard配置文件，编辑etc/httpd/conf.d/openstack-dashboard.conf，在WSGISocketPrefix run/wsgi下面加一行代码：   
WSGIApplicationGroup %{GLOBAL}   
保存，重启httpd服务。**

**后来可以访问了，但是登录老是失败，出现下图的报错，查看日志，发现这么一条信息，"Unable to create a new session key. " “Unable to create a new session key. It is likely that the cache is unavailable.”,百度一下修改/etc/openstack-dashboard/local\_settings文件中的SESSION\_ENGINE = 'django.contrib.sessions.backends.cache'，将cache修改为file，重启httpd服务即可成功访问**



****

