# Openstack-M版安装部署

## 一、Openstack信息简介：

Openstack项目是一个开源的云计算平台，它为广大云平台提供了可大规模扩展的平台，全世界的云计算技术人员创造了这个项目，通过一组相互关联的服务来提供了Iaas基础解决方案，每一个服务都通过提供自身的API来提供服务，个人或者企业完全可以根据自身的需求来安装一部分或者全部的服务。

通过下面一张表格来描述一下当前openstack的各个组件及功能。

下面这张图完整的体现了openstack服务之间的联系（Conceptualarchitecture概念架构）：

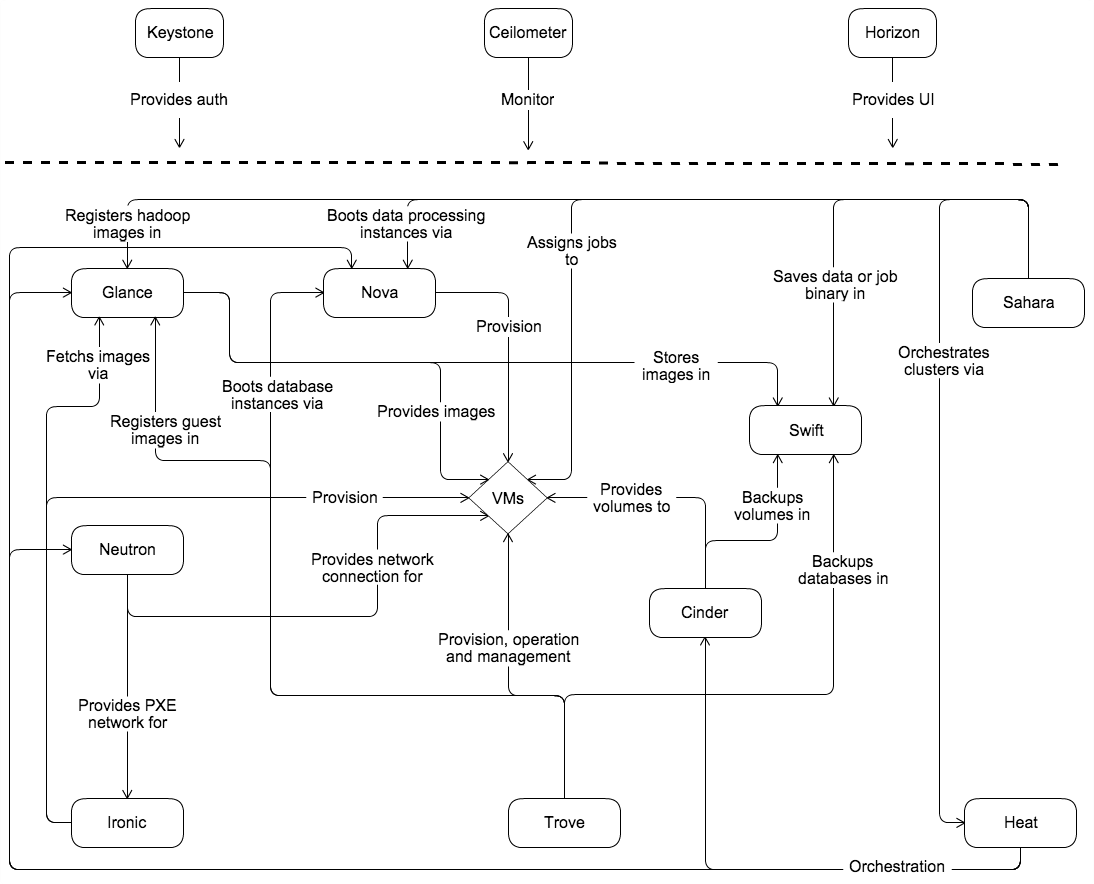


图1[openstack体系架构]

可以看说，整个生态系统外围掌控整个系统，内部各个服务相互协作提供各种功能，并且系统伸缩性大，扩展性高。

## [openstack环境准备]

软件：vmareworkstation 11

操作系统：CentOS7 3.10.0-327.e17.x86\_64

2台:一台作为控制节点controller，一台作为计算节点compute。

基本设置：

1. 每一个都选择2张网卡，一张对外提供服务，需要连接互联网，可以是桥接或者nat模式。一张提供内部网络，作为openstack内部联系的网络，在VMware选择host only模式。
2. 修改对应的主机名，并在hosts做主机与IP的解析，
3. 做时间同步，可以是本地，也可以是时间服务器，在centos7中改为chrony，调整完毕后设置开机启动。
4. 注！：方便起见，把防火墙和selinux全部关闭，确保网络畅通。

到这一步，基本上的环境配置就搞定了，我相信有余力去学习openstack，那对linux有了基本的 理解，上述步骤应该能搞定。最后重启系统准备安装。

## [openstack基本安装]

在centos上默认包含openstack版本，可以直接用yum安装最新的版本不用费力去寻找相关的源码包。

[root@ceshiapp\_2~]# yum install centos-release-openstack-mitaka.noarch -y

安装openstack客户端

[root@ceshiapp\_2~]# yum install -y python-openstackclient

openstack自带有openstack-selinux来管理openstack服务之间的安装性能，这里，为了实验顺利，这里选择忽略，实际情况可以根据需求选择是否安装。

注！：如没有特殊说明以下安装说明都是在controller中执行。

数据库以及nosql安装

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

#yum安装数据库,用于到时候创建服务的数据库 port：3306

[root@controller~]mysql \_secure\_installation #安全配置向导，包括root密码等。

[root@controllermy.cnf.d]# vim openstack.cnf

[mysqld]

......

bind-address= 0.0.0.0 #这个地址表示能够让所有的地址能远程访问。

default-storage-engine= innodb

innodb\_file\_per\_table

max\_connections= 4096

collation-server= utf8\_general\_ci

character-set-server= utf8

#修改配置文件，定义字符集和引擎，其他可按需修改。

这是yum安装的数据库，是mariadb是兼容mysql的，也可以选择源码编译安装，安装过程这里不详细讲了。

MariaDB[(none)]> grant all on \*.\* to root@'%' identified by "root" ;

这里先添加远程登录的root用户。

[root@controllermy.cnf.d]# yum install -y mongodb-server mongodb

安装mongodb数据库，提高web访问的性能。Port：27017

[root@controllermy.cnf.d]# vim /etc/mongod.conf

bind\_ip = 192.168.10.200 #将IP绑定在控制节点上。

......

[root@controllermy.cnf.d]# systemctl enable mongod.service

Createdsymlink from /etc/systemd/system/multi-user.target.wants/mongod.service to/usr/lib/systemd/system/mongod.service.

[root@controllermy.cnf.d]# systemctl start mongod.service

设置开机自启动。

[root@controllermy.cnf.d]#yum install memcached python-memcached

[root@controllermy.cnf.d]# systemctl enable memcached.service

Createdsymlink from /etc/systemd/system/multi-user.target.wants/memcached.service to /usr/lib/systemd/system/memcached.service.

[root@controllermy.cnf.d]# systemctl start memcached.service

安装memcached并设置开机启动。Port：11211 memcache用于缓存tokens

[rabbitmq消息队列安装与配置]

为了保持openstack项目松耦合，扁平化架构的特性，因此消息队列在整个架构中扮演了交通枢纽的作用，所以消息队列对openstack来说是必不可少的。不一定适用RabbitMQ，也可以是其他产品。

[root@controllermy.cnf.d]# yum install -y rabbitmq-server

[root@controllermy.cnf.d]# systemctl enable rabbitmq-server.service

Createdsymlink from/etc/systemd/system/multi-user.target.wants/rabbitmq-server.service to/usr/lib/systemd/system/rabbitmq-server.service.

[root@controllermy.cnf.d]# systemctl start rabbitmq-server.service

[root@controllermy.cnf.d]# rabbitmqctl add\_user openstack openstack

Creatinguser "openstack" ...

[root@controllermy.cnf.d]# rabbitmqctl set\_permissions openstack ".\*" ".\*"".\*"

Settingpermissions for user "openstack" in vhost "/" ...

#安装rabbidmq，创建用户并设置允许读写操作。

[Identity service—keystone]

接下来是openstack第一个验证服务——keystone服务安装，它为项目管理提供了一系列重要的验证服务和服务目录。openstack的其他服务安装之后都需要再keystone进行验证过，之后能够跟踪所有在局域网内安装好的openstack服务。

安装keystone之前先配置好数据库环境。

[root@controller~]# mysql -uroot -p

MariaDB[(none)]> CREATE DATABASE keystone;

QueryOK, 1 row affected (0.00 sec)

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

QueryOK, 0 rows affected (0.09 sec)

MariaDB[(none)]> grant all on keystone.\* to keystone@'%' identified by'keystone';

QueryOK, 0 rows affected (0.00 sec)

MariaDB[(none)]> flush privileges;

QueryOK, 0 rows affected (0.05 sec)

#生成admin tokens

[root@controller~]# openssl rand -hex 10

bc6aec97cc5e93a009b1

安装keystone httpd wsgi

WSGI是Web Server Gateway Interface的缩写，是python定义的web服务器或者框架的一种接口，位于web应用程序和web服务器之间。

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

修改keystone配置文件

[root@controller~]# vim /etc/keystone/keystone.conf

[default]

[DEFAULT]

...

admin\_token= bc6aec97cc5e93a009b1 #这里填之前生成的admintokens

[database]

......

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

[token]

......

provider= fernet #控制tokens验证、建设和撤销的操作方式

[root@controller~]# sh -c "keystone-manage db\_sync" keystone #设置数据库同步

[root@controller~]# keystone-manage fernet\_setup --keystone-user keystone --keystone-groupkeystone #生成fernetkeys

[root@controller~]# vim /etc/httpd/conf/httpd.conf #修改配置文件

...

ServerAdmincontroller

...

接下来配置虚拟主机

[root@controller~]# vim /etc/httpd/conf.d/wsgi-keystone.conf

|  |
| --- |
| WSGIScriptAlias //usr/bin/keystone-wsgi-public  WSGIApplicationGroup %{GLOBAL}  WSGIPassAuthorization On  ErrorLogFormat "%{cu}t %M"  ErrorLog /var/log/httpd/keystone-error.log  CustomLog/var/log/httpd/keystone-access.log combined  Require all granted  WSGIDaemonProcess keystone-adminprocesses=5 threads=1 user=keystone group=keystone display-na  me=%{GROUP}  WSGIProcessGroup keystone-admin  WSGIScriptAlias //usr/bin/keystone-wsgi-admin  WSGIApplicationGroup %{GLOBAL}  WSGIPassAuthorization On  ErrorLogFormat "%{cu}t %M"  ErrorLog /var/log/httpd/keystone-error.log  CustomLog/var/log/httpd/keystone-access.log combined  Require all granted |

设置开机自启动。

[root@controller~]# systemctl enable httpd.service

[root@controller~]# systemctl start httpd.service

配置上述生成的admin token

[root@controller~]# export OS\_TOKEN=bc6aec97cc5e93a009b1

配置endpoint URL

[root@controller~]# export OS\_URL=http://controller:35357/v3

Identity 版本

[root@controller~]# export OS\_IDENTITY\_API\_VERSION=3

创建identity service 。

[root@controller~]# openstack service create --namekeystone --description "OpenStack Identity" identity

验证服务管理着openstack项目中所有服务的API endpoint，服务之间的相互联系取决于验证服务中的API endpoint。根据用户不同，它提供了三种形式的API endpoint，分别是admin、internal、public。admin默认允许管理用户和租户，internal和public不允许管理用户和租户。因为在openstack中一般都会设置为2张网卡，一张对外提供服务，一张是内部交互提供的网络，因此对外一般使用public API 能够使得用户对他们自己的云进行管理。Admin API 一般对管理员提供服务，而internal API 是针对用户对自己所使用opservice服务进行管理。因此一般一个openstack服务都会创建多个API，针对不用的用户提供不同等级的服务。

[root@controller~]# openstack endpoint create --region RegionOne identity publichttp://controller:5000/v3

[root@controller~]# openstack endpoint create --region RegionOne identity internalhttp://controller:5000/v3

[root@controller~]# openstack endpoint create --region RegionOne identity adminhttp://controller:35357/v3

接下来是创建默认的域、工程（租户）、角色、用户以及绑定。

[root@controller~]# openstack domain create --description "Default Domain" default

[root@controller~]# openstack project create --domain default \

> --description "Admin Project"admin

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

UserPassword:

RepeatUser Password:#密码admin

[root@controller~]# openstack role add --project admin --user admin admin

对demo工程、用户、角色进行绑定。

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

[root@controller~]# openstack project create --domain default \

> --description "Demo Project" demo

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

UserPassword:

RepeatUser Password: #demo

[root@controller~]# openstack role add --project demo --user demo user

测试验证服务是否安装有效。

1. 在keystone-paste.ini中的这3个字段中删除admin\_token\_auth

|  |
| --- |
| [root@controller~]# vim /etc/keystone/keystone-paste.ini  [pipeline:public\_api]  ......admin\_token\_auth  [pipeline:admin\_api]  ......  [pipeline:api\_v3]  ...... |

1. 删除刚才设置的OS\_token和OS\_URL

Unset OS\_TOKEN OS\_URL

然后开始验证，没报错说明成功了。

[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 admintoken issue

Password:#admin

验证demo用户：

[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 demotoken issue

Password:#demo

最后创建user脚本

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://controller:35357/v3

export OS\_IDENTITY\_API\_VERSION=3

export OS\_IMAGE\_API\_VERSION=2

这里如果是多个台服务器配置的话。这样的脚本就会方便你配置管理员的API。主要还是起到了方便的作用。

[Image service—glance]

依旧先创建数据库以及相关用户。

[root@controller~]# mysql -u root -p

MariaDB[(none)]> CREATE DATABASE glance;

QueryOK, 1 row affected (0.04 sec)

MariaDB[(none)]> GRANT ALL PRIVILEGES ON glance.\* TO 'glance'@'localhost' \

->IDENTIFIED BY 'glance';

QueryOK, 0 rows affected (0.28 sec)

MariaDB[(none)]> GRANT ALL PRIVILEGES ON glance.\* TO 'glance'@'%' \

->IDENTIFIED BY 'glance';

QueryOK, 0 rows affected (0.01 sec)

MariaDB[(none)]> flush privileges;

QueryOK, 0 rows affected (0.08 sec)

[root@controller~]# . admin-openrc注！如果在其他服务器上就需要执行这个脚本，来配置admin api

接来下创建glance用户以及绑定。

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

UserPassword:

RepeatUser Password: #glance

将admin 角色增加到 service工程和 glance用户。

[root@controller~]# openstack role add --project service --user glance admin

[root@controller~]# openstack service create --name glance \

> --description "OpenStack Image"image

创建image服务的三种接口

[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>

安装openstack-glance并修改配置文件

[root@controller~]# yum install -y openstack-glance

[root@controller~]# vim /etc/glance/glance-api.conf

|  |
| --- |
| ......  [database]  ......  connection= mysql+pymysql://glance:glance@controller/glance  [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  [glance\_store]  ...  stores= file,http  default\_store= file  filesystem\_store\_datadir= /var/lib/glance/images/  [root@controller~]# vim /etc/glance/glance-registry.conf  [database]  ......  connection= mysql+pymysql://glance:glance@controller/glance  [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 |

做数据库同步。输出信息忽略。

[root@controller ~]# su -s /bin/sh -c "glance-managedb\_sync" glance

Option"verbose" from group "DEFAULT" is deprecated forremoval. Its value may be silentlyignored in the future.

/usr/lib/python2.7/site-packages/oslo\_db/sqlalchemy/enginefacade.py:1056:OsloDBDeprecationWarning: EngineFacade is deprecated; please useo slo\_db.sqlalchemy.enginefacade

expire\_on\_commit=expire\_on\_commit,\_conf=conf)

/usr/lib/python2.7/site-packages/pymysql/cursors.py:146:Warning: Duplicate index 'ix\_image\_properties\_image\_id\_name' defined on thetable 'glance.image\_properties'. This is deprecated and will be disallowed in afuture release.

result = self.\_query(query)

开机启动。

[root@controller~]# systemctl enable openstack-glance-api.service \

> openstack-glance-registry.service

Createdsymlink from/etc/systemd/system/multi-user.target.wants/openstack-glance-api.service to/usr/lib/systemd/system/openstack-glance-api.service.

Createdsymlink from /etc/systemd/system/multi-user.target.wants/openstack-glance-registry.serviceto /usr/lib/systemd/system/openstack-glance-registry.service.

[root@controller~]# systemctl start openstack-glance-api.service \

> openstack-glance-registry.service

下载一个镜像并创建，然后可以list查看是否成功。

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

[root@controller~]# openstack image create "cirros" \

> --file cirros-0.3.4-x86\_64-disk.img \

> --disk-format qcow2 --container-format bare\

> --public

镜像active成功。

[root@controller~]# openstack image list

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

|ID | Name| Status |

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

|7f715e8d-6f29-4e78-ab6e-d3b973d20cf7 | cirros | active |

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

[Compute-service nova]

依旧为计算服务创建数据库。这里要为nova创建2个数据库。

MariaDB[(none)]> CREATE DATABASE nova\_api;CREATE DATABASE nova;

QueryOK, 1 row affected (0.09 sec)

QueryOK, 1 row affected (0.00 sec)

MariaDB[(none)]> GRANT ALL PRIVILEGES ON nova\_api.\* TO 'nova'@'localhost' \

->IDENTIFIED BY 'nova';

QueryOK, 0 rows affected (0.15 sec)

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

QueryOK, 0 rows affected (0.00 sec)

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

QueryOK, 0 rows affected (0.00 sec)

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

QueryOK, 0 rows affected (0.00 sec)

创建用户并绑定admin角色

[root@controller~]# . admin-openrc

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

UserPassword:

RepeatUser Password: #nova

[root@controller~]# openstack role add --project service --user nova admin

创建nova计算服务。

[root@controller~]# openstack service create --name nova \

> --description "OpenStack Compute"compute

创建nova的api。

[root@controller~]# openstack endpoint create --region RegionOne \

> compute public [http://controller:8774/v2.1/%\(tenant\_id\)s](http://controller:8774/v2.1/%/(tenant_id/)s)

[root@controller~]# openstack endpoint create --region RegionOne compute internalhttp://controller:8774/v2.1/%\(tenant\_id\)s

[root@controller~]# openstack endpoint create --region RegionOne compute adminhttp://controller:8774/v2.1/%\(tenant\_id\)s

安装nova组件。

[root@controller~]# yum install -y openstack-nova-api openstack-nova-conductor \

openstack-nova-consoleopenstack-nova-novncproxy \

openstack-nova-scheduler

修改配置文件，修改内容比较多，需要仔细比对一下。

[root@controller~]# vim /etc/nova/nova.conf

|  |
| --- |
| [default]  ......  enabled\_apis= osapi\_compute,metadata  rpc\_backend=rabbit  auth\_strategy=keystone  my\_ip=192.168.10.200# 设置内网IP  use\_neutron=true#支持网络服务  firewall\_driver=nova.virt.libvirt.firewall.NoopFirewallDriver  [api\_database]  ......  connection=mysql+pymysql://nova:nova@controller/nova\_api  [database]  ......  connection=mysql+pymysql://nova:nova@controller/nova  [oslo\_messaging\_rabbit]  ......  rabbit\_host= controller  rabbit\_userid= openstack  rabbit\_password= openstack #之间创建的密码。  [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 #之前创建的用户密码  [vnc]  ......  vncserver\_listen= $my\_ip  vncserver\_proxyclient\_address= $my\_ip  [glance]  .......  api\_servers= <http://controller:9292>  [oslo\_concurrency]  ......  lock\_path= /var/lib/nova/tmp |

同步数据库,忽略输出。

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

[root@controller~]# su -s /bin/sh -c "nova-manage db sync" nova

设置开机启动。

systemctlenable openstack-nova-api.service \

openstack-nova-consoleauth.serviceopenstack-nova-scheduler.service \

openstack-nova-conductor.serviceopenstack-nova-novncproxy.service

#systemctl start openstack-nova-api.service \

openstack-nova-consoleauth.serviceopenstack-nova-scheduler.service \

openstack-nova-conductor.serviceopenstack-nova-novncproxy.service

配置compute节点

[root@compute ~]# yum install -ycentos-release-openstack-mitaka #否则直接安装openstack-nova-compute失败

[root@compute ~]# yum install -y openstack-nova-compute

修改配置文件

[root@compute~]# vim /etc/nova/nova.conf

[DEFAULT]

......

rpc\_backend= rabbit

auth\_strategy= keystone

my\_ip=192.168.10.201

firewall\_driver=nova.virt.libvirt.firewall.NoopFirewallDriver

use\_neutron=true

[glance]

......

api\_servers=http://controller:9292

[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

[libvirt]

......

virt\_type=qemu#egrep -c '(vmx|svm)' /proc/cpuinfo 如果显示为0 则需要修改成qemu

[oslo\_concurrency]

......

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

[oslo\_messaging\_rabbit]

......

rabbit\_host= controller

rabbit\_userid= openstack

rabbit\_password= openstack

[vnc]

......

enabled=true

vncserver\_listen=0.0.0.0

vncserver\_proxyclient\_address=$my\_ip

novncproxy\_base\_url=http://controller:6080/vnc\_auto.html

然后设置开机自启动。

systemctlenable libvirtd.service openstack-nova-compute.service

systemctlstart libvirtd.service openstack-nova-compute.service

验证结果，在compute上操作。4个up为ok

[root@compute~]# . admin-openrc

[root@compute~]# openstack compute service list

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

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

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

| 1 | nova-conductor | controller | internal | enabled | up | 2016-08-16T14:30:42.000000 |

| 2 | nova-scheduler | controller | internal | enabled | up | 2016-08-16T14:30:33.000000 |

| 3 | nova-consoleauth | controller | internal| enabled | up |2016-08-16T14:30:39.000000 |

| 6 | nova-compute | compute | nova| enabled | up |2016-08-16T14:30:33.000000 |

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

[Network service—neutron ]

Neutron最主要是功能是负责虚拟环境下的网络，但它之前并不叫neutron，是叫quantum，因为被注册然后才改名的。网络在openstack中最复杂的功能，配置也是最繁琐的，从官方文档中单独把网络向导列出来写一大篇文章就可以看出来。从L1到L7都有所涉及，openstack所有的服务都是通过网络练习起来。像glance一样，neutron本身并提供网络服务，它的大部分功能都是通过Plugin提供的，除了DHCP和L3-agent，这些信息在配置服务的过程中能体现出来。

Neutron本身有2种模式，一种是provider networks，指的是运营商提供网络，可以说是外网，还有一种是Self-service networks，这是私人网络，可以说内网，但是第二种网络模式包含第一种，所以基本上都会选择第二种部署方式，对外提供服务用外网，对内使用内网，因此部署openstack一般都会使用2张网卡。

在controller上配置数据库，赋权等操作。

[root@controller~]# mysql -u root -p

MariaDB[(none)]> CREATE DATABASE neutron;

QueryOK, 1 row affected (0.09 sec)

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

QueryOK, 0 rows affected (0.36 sec)

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

QueryOK, 0 rows affected (0.00 sec)

MariaDB[(none)]> flush privileges;

QueryOK, 0 rows affected (0.09 sec)

创建neutron用户，将admin角色绑定到用户上。

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

UserPassword:

RepeatUser Password: #neutron

[root@controller~]# openstack role add --project service --user neutron admin

创建neutron网络服务以及三种API接口。

[root@controller~]# openstack service create --name neutron \

> --description "OpenStackNetworking" network

[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>

网络配置，安装组件。

[root@controller~]# yum install openstack-neutron openstack-neutron-ml2 openstack-neutron-linuxbridge ebtables -y

修改neutron配置文件。

[root@controller~]# vim /etc/neutron/neutron.conf

|  |
| --- |
| [default]  ......  core\_plugin= ml2  service\_plugins= router  allow\_overlapping\_ips= True  auth\_strategy=keystone  rpc\_backend=rabbit  notify\_nova\_on\_port\_status\_changes= True  notify\_nova\_on\_port\_data\_changes= True  [database]  ......  connection= mysql+pymysql://neutron:neutron@controller/neutron  [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  [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  [oslo\_concurrency]  ......  lock\_path= /var/lib/neutron/tmp |

修改ML2配置文件

[root@controller~]# vim /etc/neutron/plugins/ml2/ml2\_conf.ini

|  |
| --- |
| [ml2]  ......  type\_drivers= flat,vlan,vxlan  tenant\_network\_types= vxlan  mechanism\_drivers= linuxbridge,l2population  extension\_drivers= port\_security  [ml2\_type\_flat]  ......  flat\_networks= provider  [ml2\_type\_vxlan]  ...  vni\_ranges= 1:1000  [securitygroup]  ...  enable\_ipset= True |

设置桥接代理配置文件。

[root@controller~]# vim /etc/neutron/plugins/ml2/linuxbridge\_agent.ini

|  |
| --- |
| [linux\_bridge]  ......  physical\_interface\_mappings=provider:eno33554984#内网网卡  [vxlan]  ......  enable\_vxlan= true  local\_ip= 192.168.10.200 #controller ip  l2\_population= true  [securitygroup]  ......  enable\_security\_group= true  firewall\_driver=neutron.agent.linux.iptables\_firewall.IptablesFirewallDriver |

修改Layer-3 代理配置文件。

[root@controller~]# vim /etc/neutron/l3\_agent.ini

|  |
| --- |
| [DEFAULT]  ......  interface\_driver= neutron.agent.linux.interface.BridgeInterfaceDriver  external\_network\_bridge=  修改DHCP代理配置文件。  [root@controller~]# vim /etc/neutron/dhcp\_agent.ini  [DEFAULT]  ......  interface\_driver= neutron.agent.linux.interface.BridgeInterfaceDriver  dhcp\_driver= neutron.agent.linux.dhcp.Dnsmasq  enable\_isolated\_metadata = true  修改元数据代理配置文件。  vim/etc/neutron/metadata\_agent.ini  [DEFAULT]  ......  nova\_metadata\_ip= controller  metadata\_proxy\_shared\_secret= meta #元数据共享的一个密码，之后会用到。 |

在nova配置文件中修改neutron的参数。

[root@controller~]# vim /etc/nova/nova.conf

|  |
| --- |
| [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= meta |

设置软连接

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

同步数据库。

[root@controllerneutron]# su -s /bin/sh -c "neutron-db-manage --config-file/etc/neutron/neutron.conf \

> --config-file /etc/neutron/plugins/ml2/ml2\_conf.iniupgrade head" neutron

出现OK成功。

重启nova-api服务

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

设置开机启动并启动neutron服务。

#systemctl enable neutron-server.service \

neutron-linuxbridge-agent.serviceneutron-dhcp-agent.service \

neutron-metadata-agent.service

#systemctl enable neutron-l3-agent.service

#systemctl start neutron-server.service \

neutron-linuxbridge-agent.serviceneutron-dhcp-agent.service \

neutron-metadata-agent.service

#systemctlstart neutron-l3-agent.service

配置完控制节点，配置compute节点的网络服务。

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

修改neutron配置文件

[root@compute~]# vim /etc/neutron/neutron.conf

|  |
| --- |
| [DEFAULT]  ......  rpc\_backend= rabbit  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  [oslo\_concurrency]  .....  lock\_path= /var/lib/neutron/tmp  [oslo\_messaging\_rabbit]  ......  rabbit\_host= controller  rabbit\_userid= openstack  rabbit\_password= openstack |

修改linux桥接代理的配置文件

[root@compute~]# vim /etc/neutron/plugins/ml2/linuxbridge.ini

|  |
| --- |
| [linux\_bridge]  ......  physical\_interface\_mappings= provider:eno33554984  [securitygroup]  .....  firewall\_driver= neutron.agent.linux.iptables\_firewall.IptablesFirewallDriver  enable\_security\_group= true  [vxlan]  ......  enable\_vxlan= true  local\_ip= 192.168.10.201 #内网IP  l2\_population= true |

修改nova中neutron模块的配置信息。

[root@compute~]# vim /etc/nova/nova.conf

|  |
| --- |
| [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  重启compute服务。  #systemctlrestart openstack-nova-compute.service  设置neutron开机启动。  #systemctlenable neutron-linuxbridge-agent.service  #systemctl start neutron-linuxbridge-agent.service |

验证neutron服务,出现5个笑脸说明OK。

[root@controller~]# neutron agent-list

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

|id | agent\_type | host| availability\_zone | alive | admin\_state\_up | binary |

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

|7476f2bf- | DHCP agent | controller | nova | :-)| True | neutron-dhcp- |

|dced-44d6-8a | | | | | | agent |

|a0-2aab951b1 | | | | | | |

|961 | | | | | | |

|82319909-a4b | [Linux](https://www.2cto.com/os/linux/" \t "https://www.2cto.com/net/201608/_blank) bridge | compute| | :-) | True | neutron- |

|2-47d9-ae42- | agent | | | | | linuxbridge- |

|288282ad3972 | | | | | | agent |

|b465f93f- | Metadata | controller | | :-) | True| neutron- |

|90c3-4cde- | agent | | | | | metadata- |

|949c- | | | | | | agent |

|47c205a74f65 | | | | | | |

|e662d628-327 | L3 agent | controller| nova | :-) | True | neutron-l3-ag |

|b-4695-92e1- | | | | | | ent |

|76da38806b05 | | | | | | |

|fa84105f-d49 | Linux bridge | controller | | :-) | True | neutron- |

|7-4b64-9a53- | agent | | | | | linuxbridge- |

|3b1f074e459e | | | | | | agent |

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

[root@compute~]# neutron ext-list

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

|alias | name |

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

|default-subnetpools | DefaultSubnetpools |

|network-ip-availability | Network IPAvailability |

|network\_availability\_zone | Network Availability Zone |

|auto-allocated-topology | AutoAllocated Topology Services|

|ext-gw-mode | Neutron L3Configurable external gateway mode |

|binding | PortBinding |

|agent | agent |

|subnet\_allocation | SubnetAllocation |

|l3\_agent\_scheduler | L3 AgentScheduler |

|tag | Tagsupport|

|external-net | Neutronexternal network |

|net-mtu | NetworkMTU |

|availability\_zone | AvailabilityZone |

|quotas | Quotamanagement support |

|l3-ha | HA Routerextension |

|provider | ProviderNetwork |

|multi-provider | MultiProvider Network |

|address-scope | Addressscope |

|extraroute | Neutron ExtraRoute |

|timestamp\_core | Time StampFields addition for core resources |

|router | Neutron L3Router |

|extra\_dhcp\_opt | Neutron ExtraDHCP opts |

| dns-integration | DNS Integration |

|security-group |security-group|

|dhcp\_agent\_scheduler | DHCP AgentScheduler |

|router\_availability\_zone | RouterAvailability Zone |

|rbac-policies | RBACPolicies|

|standard-attr-description | standard-attr-description |

|port-security | PortSecurity |

|allowed-address-pairs | AllowedAddress Pairs |

|dvr | DistributedVirtual Router |

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

[Dashboard-horizon]

其实将neutron安装完毕以后openstack项目最基本的已经能够运行起来了，但是为了操作更加方便直观，我们再安装WebUI，就是horizon。

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

[root@controller~]# vim /etc/openstack-dashboard/local\_settings

|  |
| --- |
| ......  OPENSTACK\_HOST= "controller"  ALLOWED\_HOSTS= ['\*',]  OPENSTACK\_KEYSTONE\_URL= "http://%s:5000/v3.0" % OPENSTACK\_HOST  OPENSTACK\_API\_VERSIONS= {  "identity": 3,  "volume": 2,  "compute": 2,  }  OPENSTACK\_KEYSTONE\_MULTIDOMAIN\_SUPPORT= True  OPENSTACK\_KEYSTONE\_DEFAULT\_DOMAIN= "default"  OPENSTACK\_KEYSTONE\_DEFAULT\_ROLE= "user"  TIME\_ZONE= "UTC"#时区  SESSION\_ENGINE= 'django.contrib.sessions.backends.cache'  CACHES= {  'default': {  'BACKEND':'django.core.cache.backends.memcached.MemcachedCache',  'LOCATION': 'controller:11211',  }  } |

重启httpd 和memcached服务。

[root@controller~]# systemctl restart httpd.service

[root@controller~]# systemctl restart memcached.service

最后打开http://controller/dashboard登录

