CentOS7.2下安装部署OpenStack+KVM 云平台虚拟化环境详解

基础环境搭建

公司在 IDC 机房有两台很高配置的服务器，计划在上面部署 openstack 云平台虚拟化环境，用于承载后期开发测试和其他的一些对内业务。

以下对 OpenStack 的部署过程及其使用做一详细介绍，仅仅依据本人实际经验而述，如有不当，敬请指出~

****

1 OpenStack 介绍

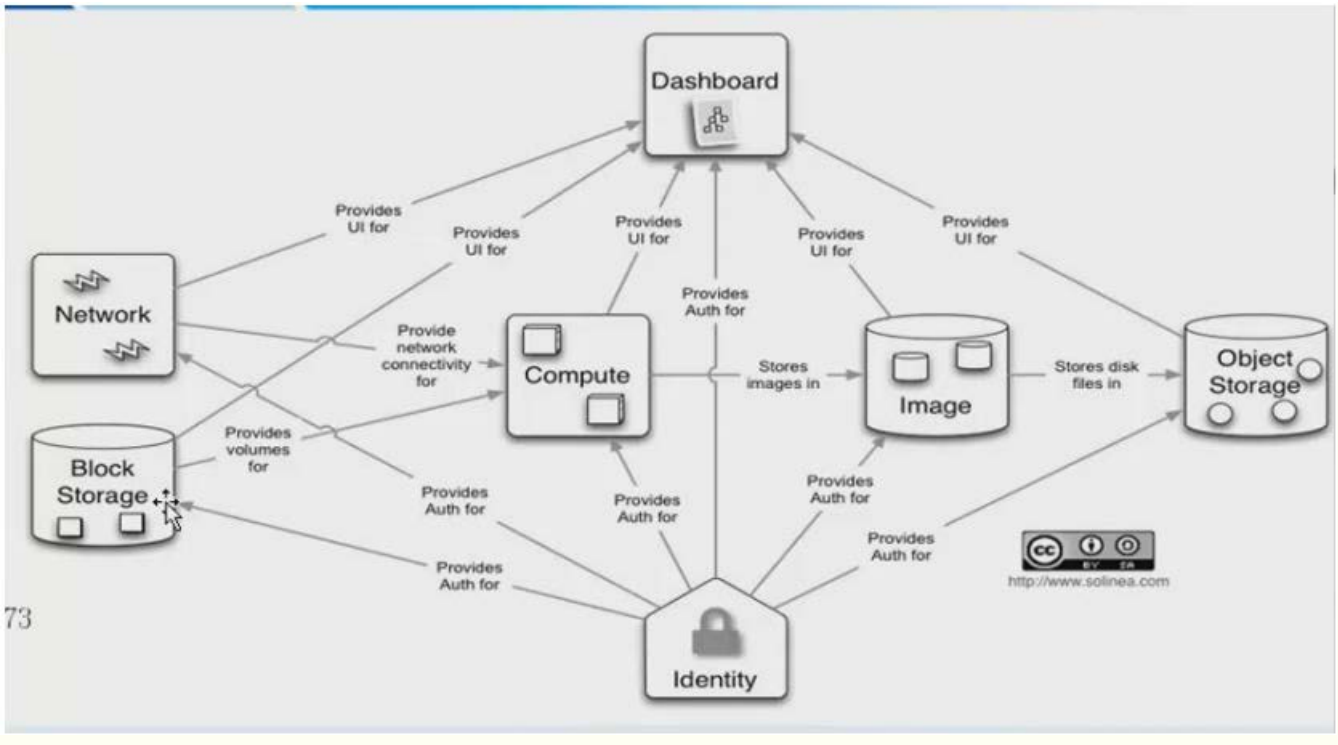
1.1 百度百科

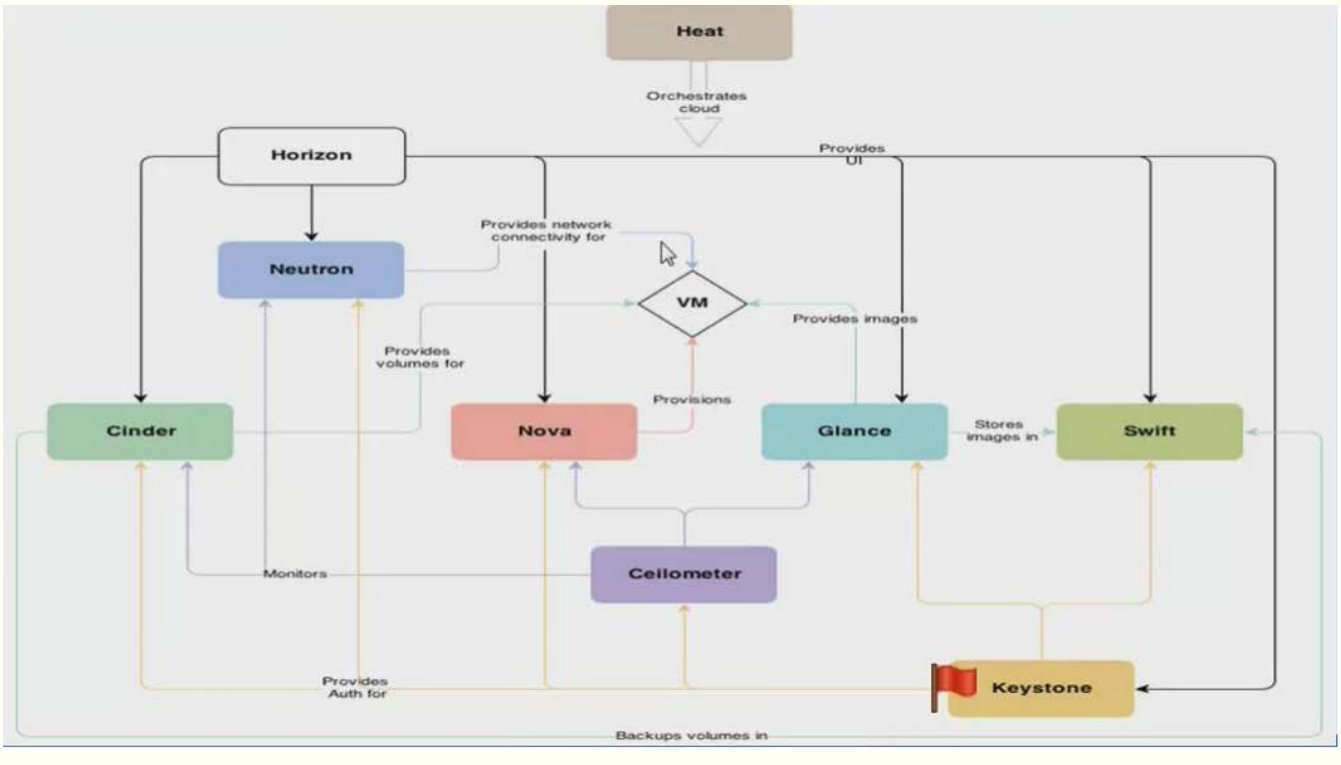
OpenStack 是一个由 NASA （美国国家航空航天局）和 Rackspace 合作研发并发起的，以 Apache 许可证授权的自由软件和开放源代码项目。

1.2 版本历史

Release Name	Release Date	Included Components
Austin	21 October 2010	Nova, Swift
Bexar	3 February 2011	Nova, Glance, Swift
Cactus	15 April 2011	Nova, Glance, Swift
Diablo	22 September 2011	Nova, Glance, Swift
Essex	5 April 2012	Nova, Glance, Swift, Horizon, Keystone
Folsom	27 September 2012	Nova, Glance, Swift, Horizon, Keystone, Quantum, Cinder
Grizzly	4 April 2013	Nova, Glance, Swift, Horizon, Keystone, Quantum, Cinder
Havana	17 October 2013	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder
Icehouse	April 2014	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, (More to be added)

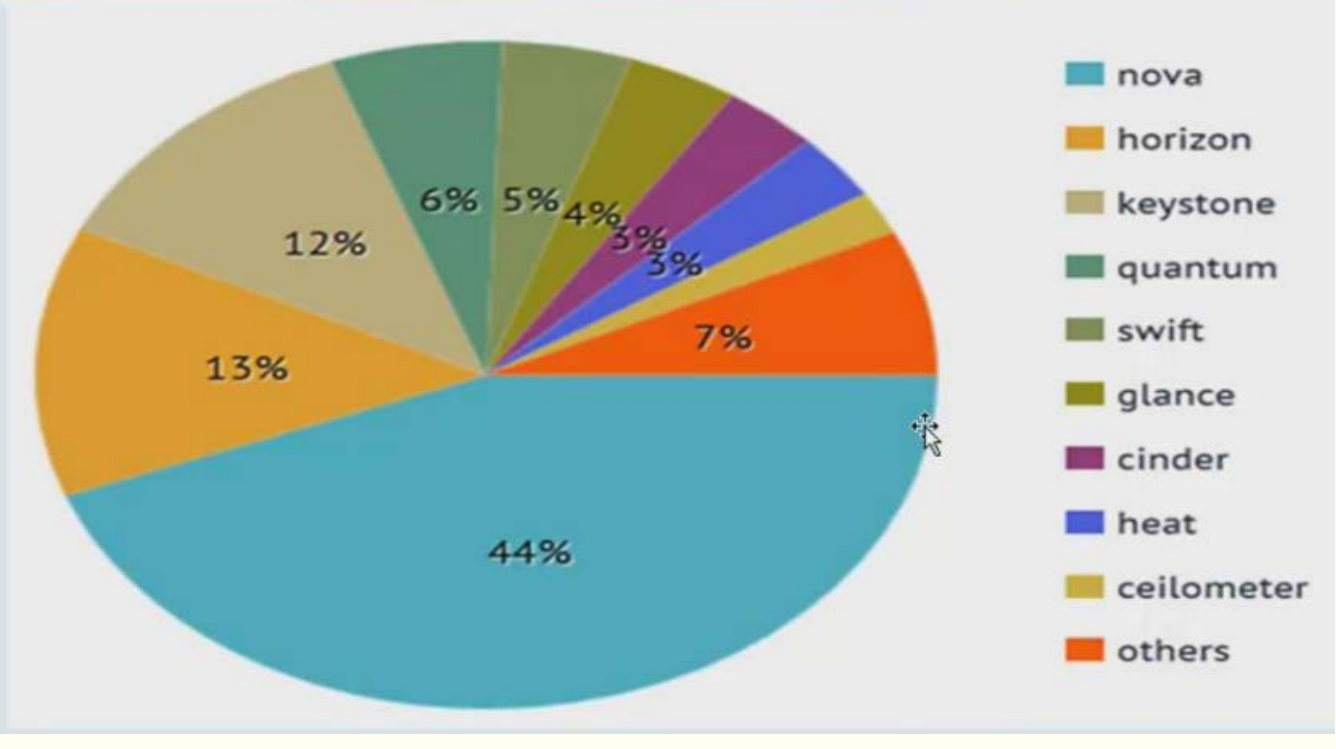
1.3 openstack 架构概念





1.4 openstack 各个服务名称对应

服务名称	项目名称	描述
Dashboard	Horizon	基于OpenStack API接口使用django开发的Web管理。
Compute	Nova	通过虚拟化技术提供计算资源池。
Networking	Neutron	实现了虚拟机的网络资源管理。
Storage（存储）		
Object Storage	Swift	对象存储，适用于“一次写入、多次读取”
Block Storage	Cinder	块存储，提供存储资源池
Shared Services（共享服务）		
Identity Service	Keystone	认证管理
Image Service	Glance	提供虚拟镜像的注册和存储管理
Telemetry	Ceilometer	提供监控和数据采集、计量服务
Higher-level services（高层服务）		
Orchestration	Heat	自动化部署的组件
Database Service	Trove	提供数据库应用服务



****

以下安装部署已经过测试，完全通过！

建议在物理机上部署 openstack，并且是 centos7 或 ubuntu 系统下，centos6x 的源里已不支持 openstack 部分组件下载了。

2 环境准备

openstack 主机名不能改，装的时候是什么就是什么， 运维标准化。

1、 CentOS 7.2 系统 2 台

node1 即作为控制节点，也作为计算节点；(即可以单机部署，单机部署时则下面记录的控制节点和计算节点的操作步骤都要在本机执行下)

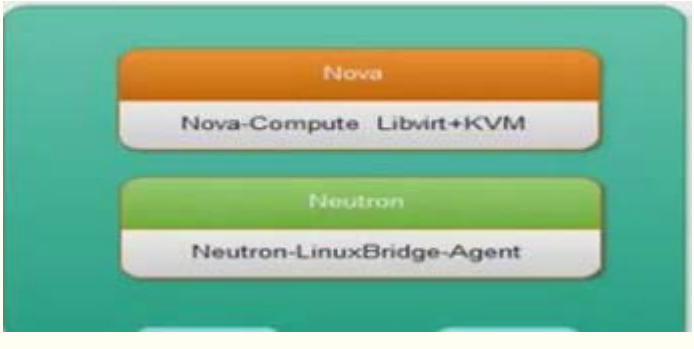
node2 就只是计算节点

控制节点去操控计算节点，计算节点上可以创建虚拟机

linux-node1.openstack 192.168.1.17 网卡 NAT em2 （外网 ip 假设是 58.68.250.17）(em2 是内网网卡，下面 neutron 配置文件里会设置到)
linux-node2.openstack 192.168.1.8 网卡 NAT em2

控制节点：linux-node1.openstack 192.168.1.17

计算节点：linux-node2.openstack 192.168.1.8



2.域名解析和关闭防火墙 （控制节点和计算节点都做）

/etc/hosts #主机名一开始设置好，后面就不能更改了，否则就会出问题！这里设置好 ip 与主机名的对应关系

```
192.168.1.17 linux-node1.openstack
192.168.1.8   linux-node2.openstack
```

关闭 selinux

```
sed -i 's#SELINUX=enforcing#SELINUX=disabled#g' /etc/sysconfig/selinux
setenforce 0
```

关闭 iptables

```
systemctl start firewalld.service
systemctl stop firewalld.service
systemctl disable firewalld.service
```

3 安装配置 OpenStack

官方文档 http://docs.openstack.org/

3.1 安装软件包

linux-node1.openstack 安装

****
***

```
#Base
yum install -y http://dl.fedoraproject.org/pub/epel/7/x86_64/e/epel-release-7-8.noarch.rpm
yum install -y centos-release-openstack-liberty
yum install -y python-openstackclient

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

##RabbitMQ
yum install -y rabbitmq-server

##Keystone
yum install -y openstack-keystone httpd mod_wsgi memcached python-memcached

##Glance
yum install -y openstack-glance python-glance python-glanceclient

##Nova
yum install -y openstack-nova-api openstack-nova-cert openstack-nova-conductor openstack-nova-con
sole openstack-nova-novncproxy openstack-nova-scheduler python-novaclient

##Neutron linux-node1.example.com
yum install -y openstack-neutron openstack-neutron-ml2 openstack-neutron-linuxbridge python-neutro
nclient ebtables ipset

##Dashboard
yum install -y openstack-dashboard

##Cinder
yum install -y openstack-cinder python-cinderclient
```

****

linux-node2.openstack 安装

```
##Base
yum install -y http://dl.fedoraproject.org/pub/epel/7/x86_64/e/epel-release-7-8.noarch.rpm
yum install centos-release-openstack-liberty
yum install python-openstackclient

##Nova linux-node2.openstack
yum install -y openstack-nova-compute sysfsutils

##Neutron linux-node2.openstack
yum install -y openstack-neutron openstack-neutron-linuxbridge ebtables ipset

##Cinder
yum install -y openstack-cinder python-cinderclient targetcli python-oslo-policy
```

****

3.2 设置时间同步、 关闭 selinux 和 iptables

在 linux-node1 上配置（ 只有 centos7 能用， 6 还用 ntp）

```
[root@linux-node1 ~]# yum install -y chrony
[root@linux-node1 ~]# vim /etc/chrony.conf
allow 192.168/16 #允许那些服务器和自己同步时间
[root@linux-node1 ~]# systemctl enable chronyd.service    #开机启动
[root@linux-node1 ~]# systemctl start chronyd.service
[root@linux-node1 ~]# timedatectl set-timezone Asia/Shanghai    #设置时区
[root@linux-node1 ~]# timedatectl status
Local time: Fri 2016-08-26 11:14:19 CST
Universal time: Fri 2016-08-26 03:14:19 UTC
RTC time: Fri 2016-08-26 03:14:19
Time zone: Asia/Shanghai (CST, +0800)
NTP enabled: yes
NTP synchronized: yes
```

```
RTC in local TZ: no
DST active: n/a
```

在 **linux-node2** 上配置

```
[root@linux-node2 ~]# yum install -y chrony
[root@linux-node2 ~]# vim /etc/chrony.conf
server 192.168.1.17 iburst #只留一行
[root@linux-node2 ~]# systemctl enable chronyd.service
[root@linux-node2 ~]# systemctl start chronyd.service
[root@linux-node2 ~]# timedatectl set-timezone Asia/Shanghai
[root@linux-node2 ~]# chronyc sources
```

3.3 安装及配置 mysql

```
[root@linux-node1 ~]# cp /usr/share/mysql/my-medium.cnf /etc/my.cnf                #或者是/usr/share/mariadb/my-medium.cnf
[mysqld]
default-storage-engine = innodb
innodb_file_per_table
collation-server = utf8_general_ci
init-connect = 'SET NAMES utf8'
character-set-server = utf8
[root@linux-node1 ~]# systemctl enable mariadb.service                             #Centos 7 里面 mysql 叫 maridb
[root@linux-node1 ~]# ln -s '/usr/lib/systemd/system/mariadb.service' '/etc/systemd/system/multi-user.target.wants/mariadb.service'
[root@linux-node1 ~]# mysql_install_db --datadir="/var/lib/mysql" --user="mysql"           #初始化数据库
[root@linux-node1 ~]# systemctl start mariadb.service
[root@linux-node1 ~]# mysql_secure_installation                                    #设置密码及初始化
```

密码 123456，一路 y 回车

创建数据库

```
[root@openstack-server ~]# mysql -p123456
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 5579
Server version: 5.5.50-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)]> CREATE DATABASE keystone;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY 'keystone';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY 'keystone';
MariaDB [(none)]> CREATE DATABASE glance;
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';
MariaDB [(none)]> CREATE DATABASE nova;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' IDENTIFIED BY 'nova';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' IDENTIFIED BY 'nova';
MariaDB [(none)]> CREATE DATABASE neutron;
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';
MariaDB [(none)]> CREATE DATABASE cinder;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' IDENTIFIED BY 'cinder';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' IDENTIFIED BY 'cinder';
MariaDB [(none)]> flush privileges;
MariaDB [(none)]> show databases;
+--------------------+
| Database |
```

```
+--------------------+
| information_schema |
| cinder             |
| glance             |
| keystone           |
| mysql              |
| neutron            |
| nova               |
| performance_schema |
+--------------------+
8 rows in set (0.00 sec)

MariaDB [(none)]>
```

修改下 mysql 的连接数，否则 openstack 后面的操作会报错：“ERROR 1040 (08004): Too many connections ”

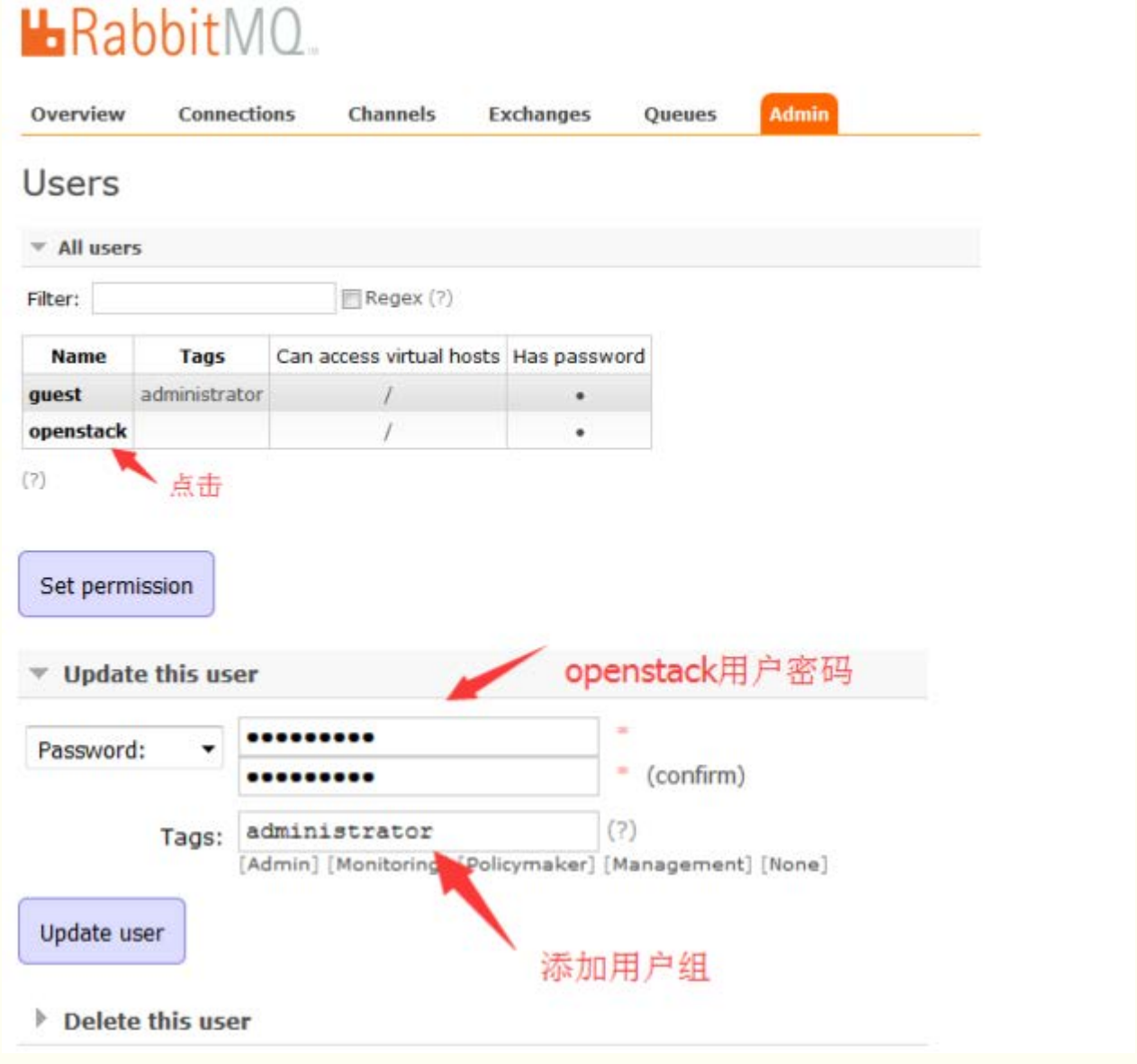
3.4 配置 rabbitmq

MQ 全称为 Message Queue, 消息队列（ MQ）是一种应用程序对应用程序的通信方法。应用程序通过读写出入队列的消息（针对应用程序的数据）来通信，而无需专用连接来链接它们。
消 息传递指的是程序之间通过在消息中发送数据进行通信，而不是通过直接调用彼此来通信，直接调用通常是用于诸如远程过程调用的技术。排队指的是应用程序通过 队列来通信。
队列的使用除去了接收和发送应用程序同时执行的要求。
RabbitMQ 是一个在 AMQP 基础上完整的，可复用的企业消息系统。他遵循 Mozilla Public License 开源协议。
启动 rabbitmq， 端口 5672，添加 openstack 用户

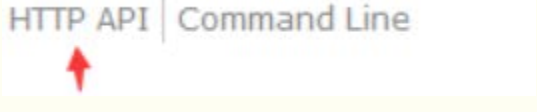
```
[root@linux-node1 ~]# systemctl enable rabbitmq-server.service
[root@linux-node1 ~]# ln -s '/usr/lib/systemd/system/rabbitmq-server.service' '/etc/systemd/system/multi-user.target.wants/rabbitmq-server.service'
[root@linux-node1 ~]# systemctl start rabbitmq-server.service
[root@linux-node1 ~]# rabbitmqctl add_user openstack openstack                    #添加用户及密码
[root@linux-node1 ~]# rabbitmqctl set_permissions openstack ".*" ".*" ".*"       #允许配置、写、读访问 openstack
[root@linux-node1 ~]# rabbitmq-plugins list                                      #查看支持的插件
.....
[ ] rabbitmq_management 3.6.2                                    #使用此插件实现 web 管理
.....
[root@linux-node1 ~]# rabbitmq-plugins enable rabbitmq_management #启动插件
The following plugins have been enabled:
mochiweb
webmachine
rabbitmq_web_dispatch
amqp_client
rabbitmq_management_agent
rabbitmq_management
Plugin configuration has changed. Restart RabbitMQ for changes to take effect.
[root@linux-node1 ~]# systemctl restart rabbitmq-server.service

[root@linux-node1 ~]#lsof -i:15672
```

访问 RabbitMQ,访问地址是 http://58.68.250.17:15672
默认用户名密码都是 guest，浏览器添加 openstack 用户到组并登陆测试，连不上情况一般是防火墙没有关闭所致！



之后退出使用 openstack 登录
如何使用 zabbix 监控？
左下角有 HTTP API 的介绍，可以实现 zabbix 的监控



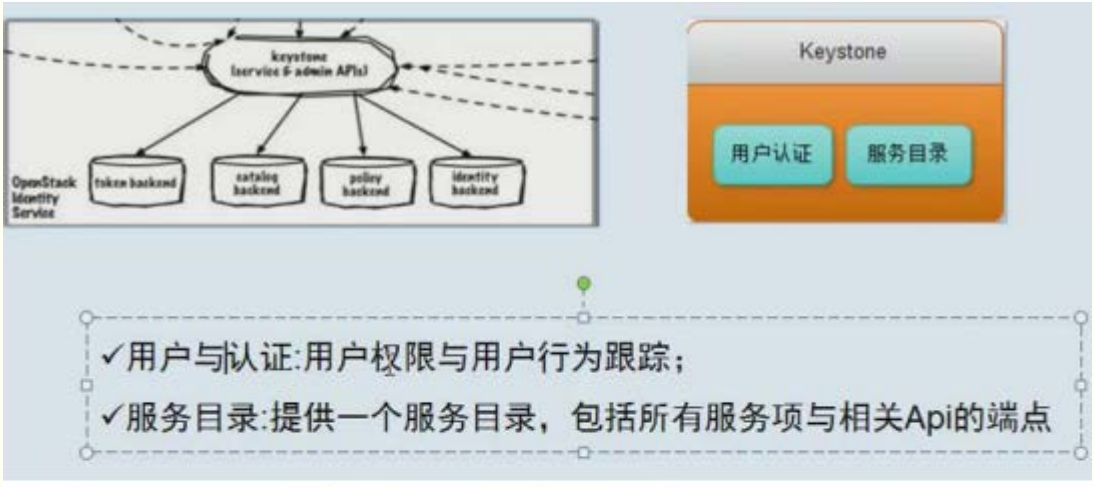
****

以上完成基础环境的配置，下面开始安装 openstack 的组件

3.5 配置 Keystone 验证服务

所有的服务，都需要在 keystone 上注册

3.5.1 Keystone 介绍



OpenStack中的概念	跟宾馆的类比
User	住宾馆的人
Credentials	开启房间的钥匙
Authentication	宾馆为了拒绝不必要的人进入宾馆，专门设置的机制，只有拥有要是的人才能进入
Token	也是一种钥匙，不过有点特别
Tenant	宾馆
Service	宾馆可以提供的服务类别，比如饮食类，娱乐类
Endpoint	具体的一种服务，比如游泳，棋牌
Role	VIP等级，VIP级别越高，享有越高的权限

3.5.2 配置 Keystone

端口 5000 和 35357

1、修改/etc/keystone/keystone.conf

取一个随机数

```
[root@linux-node1 ~]# openssl rand -hex 10
35d6e6f377a889571bcf
[root@linux-node1 ~]# cat /etc/keystone/keystone.conf|grep -v "^#"|grep -v "^$"
[DEFAULT]
admin_token = 35d6e6f377a889571bcf                    #设置 token，和上面产生的随机数值一致
verbose = true
[assignment]
[auth]
[cache]
[catalog]
[cors]
[cors.subdomain]
```

```
[credential]
[database]
connection = mysql://keystone:keystone@192.168.1.17/keystone                    #设置数据库连接 写到 database 下
[domain_config]
[endpoint_filter]
[endpoint_policy]
[eventlet_server]
[eventlet_server_ssl]
[federation]
[fernet_tokens]
[identity]
[identity_mapping]
[kvs]
[ldap]
[matchmaker_redis]
[matchmaker_ring]
[memcache]
servers = 192.168.1.17:11211
[oauth1]
[os_inherit]
[oslo_messaging_amqp]
[oslo_messaging_qpid]
[oslo_messaging_rabbit]
[oslo_middleware]
[oslo_policy]
[paste_deploy]
[policy]
[resource]
[revoke]
driver = sql
[role]
[saml]
[signing]
[ssl]
[token]
provider = uuid
driver = memcache
[tokenless_auth]
[trust]
```

2、 创建数据库表， 使用命令同步

```
[root@linux-node1 ~]# su -s /bin/sh -c "keystone-manage db_sync" keystone
No handlers could be found for logger "oslo_config.cfg"                    #出现这个信息，不影响后续操作！忽略~

[root@linux-node1 ~]# ll /var/log/keystone/keystone.log
-rw-r--r--. 1 keystone keystone 298370 Aug 26 11:36 /var/log/keystone/keystone.log      #之所以上面 su 切换是因为这个日志文件属主
[root@linux-node1 config]# mysql -h 192.168.1.17 -u keystone -p      #数据库检查表,生产环境密码不要用 keystone，改成复杂点的密码
```

3、 启动 memcached 和 apache

启动 memcached

```
[root@linux-node1 ~]# systemctl enable memcached
[root@linux-node1 ~]#ln -s '/usr/lib/systemd/system/memcached.service' '/etc/systemd/system/multi-user.target.wants/memcached.service'
[root@linux-node1 ~]# systemctl start memcached
```

配置 httpd

```
[root@linux-node1 ~]# vim /etc/httpd/conf/httpd.conf
ServerName 192.168.1.17:80
[root@linux-node1 ~]# cat /etc/httpd/conf.d/wsgi-keystone.conf
Listen 5000
Listen 35357
```

```
<VirtualHost *:5000>
WSGIDaemonProcess keystone-public processes=5 threads=1 user=keystone group=keystone display-
name=%{GROUP}
WSGIProcessGroup keystone-public
WSGIScriptAlias / /usr/bin/keystone-wsgi-public
WSGIApplicationGroup %{GLOBAL}
WSGIPassAuthorization On
<IfVersion >= 2.4>
ErrorLogFormat "%{cu}t %M"
</IfVersion>
ErrorLog /var/log/httpd/keystone-error.log
CustomLog /var/log/httpd/keystone-access.log combined
<Directory /usr/bin>
<IfVersion >= 2.4>
Require all granted
</IfVersion>
<IfVersion < 2.4>
Order allow,deny
Allow from all
</IfVersion>
</Directory>
</VirtualHost>

<VirtualHost *:35357>
WSGIDaemonProcess keystone-admin processes=5 threads=1 user=keystone group=keystone display
-name=%{GROUP}
WSGIProcessGroup keystone-admin
WSGIScriptAlias / /usr/bin/keystone-wsgi-admin
WSGIApplicationGroup %{GLOBAL}
WSGIPassAuthorization On
<IfVersion >= 2.4>
ErrorLogFormat "%{cu}t %M"
</IfVersion>
ErrorLog /var/log/httpd/keystone-error.log
CustomLog /var/log/httpd/keystone-access.log combined
<Directory /usr/bin>
<IfVersion >= 2.4>
Require all granted
</IfVersion>
<IfVersion < 2.4>
Order allow,deny
Allow from all
</IfVersion>
</Directory>
</VirtualHost>
```

启动 httpd

```
[root@linux-node1 config]# systemctl enable httpd
[root@linux-node1 config]#ln -s '/usr/lib/systemd/system/httpd.service' '/etc/systemd/system/multi-u
ser.target.wants/httpd.service'
[root@linux-node1 config]# systemctl start httpd
[root@linux-node1 ~]# netstat -lntup|grep httpd
tcp6 0 0 :::5000 :::* LISTEN 23632/httpd
tcp6 0 0 :::80 :::* LISTEN 23632/httpd
tcp6 0 0 :::35357 :::* LISTEN 23632/httpd
```

如果 http 起不来关闭 selinux 或者安装 yum install openstack-selinux

4、 创建 keystone 用户

临时设置 admin_token 用户的环境变量，用来创建用户

```
[root@linux-node1 ~]# export OS_TOKEN=35d6e6f377a889571bcf                    #上面产生的随机
数值
[root@linux-node1 ~]# export OS_URL=http://192.168.1.17:35357/v3
[root@linux-node1 ~]# export OS_IDENTITY_API_VERSION=3
```

创建 admin 项目---创建 admin 用户（密码 admin，生产不要这么玩） ---创建 admin 角色---把 admin 用户加入到 admin 项目赋予 admin 的角色（三个 admin 的位置：项目，用户，角色）

```
[root@linux-node1 ~]#openstack project create --domain default --description "Admin Project" admin
[root@linux-node1 ~]#openstack user create --domain default --password-prompt admin
[root@linux-node1 ~]#openstack role create admin
[root@linux-node1 ~]#openstack role add --project admin --user admin admin
```

创建一个普通用户 demo

```
[root@linux-node1 ~]#openstack project create --domain default --description "Demo Project" demo
[root@linux-node1 ~]#openstack user create --domain default --password=demo demo
[root@linux-node1 ~]#openstack role create user
[root@linux-node1 ~]#openstack role add --project demo --user demo user
```

创建 service 项目，用来管理其他服务用

```
[root@linux-node1 ~]#openstack project create --domain default --description "Service Project" servic
e
```

以上的名字都是固定的，不能改

查看创建的而用户和项目

```
[root@linux-node1 ~]# openstack user list
+----------------------------------+-------+
| ID | Name |
+----------------------------------+-------+
| b1f164577a2d43b9a6393527f38e3f75 | demo |
| b694d8f0b70b41d883665f9524c77766 | admin |
+----------------------------------+-------+
[root@linux-node1 ~]# openstack project list
+----------------------------------+---------+
| ID | Name |
+----------------------------------+---------+
| 604f9f78853847ac9ea3c31f2c7f677d | demo |
| 777f4f0108b1476eabc11e00dccaea9f | admin |
| aa087f62f1d44676834d43d0d902d473 | service |
+----------------------------------+---------+
```

5、注册 keystone 服务，以下**三种类型分别为公共的、内部的、管理的。**

```
[root@linux-node1 ~]#openstack service create --name keystone --description "OpenStack Identity" id
entity
[root@linux-node1 ~]#openstack endpoint create --region RegionOne identity public http://192.168.1.
17:5000/v2.0
[root@linux-node1 ~]#openstack endpoint create --region RegionOne identity internal http://192.168.
1.17:5000/v2.0
[root@linux-node1 ~]#openstack endpoint create --region RegionOne identity admin http://192.168.1.
17:35357/v2.0
[root@linux-node1 ~]# openstack endpoint list #查看
+----------------------------------+-----------+--------------+--------------+---------+----
-------+----------------------------------+
| ID | Region | Service Name | Service Type | Enabled |
Interface | URL |
+----------------------------------+-----------+--------------+--------------+---------+----
-------+----------------------------------+
| 011a24def8664506985815e0ed2f8fa5 | RegionOne | keystone | identity | True |
internal | http://192.168.1.17:5000/v2.0 |
| b0981cae6a8c4b3186edef818733fec6 | RegionOne | keystone | identity | True | public
| http://192.168.1.17:5000/v2.0 |
| c4e0c79c0a8142eda4d9653064563991 | RegionOne | keystone | identity | True | admin
| http://192.168.1.17:35357/v2.0 |
+----------------------------------+-----------+--------------+--------------+---------+----
-------+----------------------------------+
[root@linux-node1 ~]# openstack endpoint delete ID                    #使用这个命令删除
```

6、 验证，获取 token，只有获取到才能说明 keystone 配置成功

```
[root@linux-node1 ~]# unset OS_TOKEN
[root@linux-node1 ~]# unset OS_URL
[root@linux-node1 ~]# openstack --os-auth-url http://192.168.1.17:35357/v3 --os-project-domain-id
```

```
default --os-user-domain-id default --os-project-name admin --os-username admin --os-auth-type pas
sword token issue              #回车
Password: admin
+------------+----------------------------------+
| Field | Value |
+------------+----------------------------------+
| expires | 2015-12-17T04:22:00.600668Z |
| id | 1b530a078b874438aadb77af11ce297e |
| project_id | 777f4f0108b1476eabc11e00dccaea9f |
| user_id | b694d8f0b70b41d883665f9524c77766 |
+------------+----------------------------------+
```

使用环境变量来获取 token，环境变量在后面创建虚拟机时也需要用。
创建两个环境变量文件，使用时直接 source！！！（注意，下面两个 sh 文件所在的路径，在查看命令前都要 source 下，不然会报错！！）

```
[root@linux-node1 ~]# cat admin-openrc.sh
export OS_PROJECT_DOMAIN_ID=default
export OS_USER_DOMAIN_ID=default
export OS_PROJECT_NAME=admin
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=admin
export OS_AUTH_URL=http://192.168.1.17:35357/v3
export OS_IDENTITY_API_VERSION=3

[root@linux-node1 ~]# cat demo-openrc.sh
export OS_PROJECT_DOMAIN_ID=default
export OS_USER_DOMAIN_ID=default
export OS_PROJECT_NAME=demo
export OS_TENANT_NAME=demo
export OS_USERNAME=demo
export OS_PASSWORD=demo
export OS_AUTH_URL=http://192.168.1.17:5000/v3
export OS_IDENTITY_API_VERSION=3
[root@linux-node1 ~]# source admin-openrc.sh
[root@linux-node1 ~]# openstack token issue
+------------+----------------------------------+
| Field | Value |
+------------+----------------------------------+
| expires | 2015-12-17T04:26:08.625399Z |
| id | 58370ae3b9bb4c07a67700dd184ad3b1 |
16
| project_id | 777f4f0108b1476eabc11e00dccaea9f |
| user_id | b694d8f0b70b41d883665f9524c77766 |
+------------+----------------------------------+
```

3.6 配置 glance 镜像服务

3.6.1 glance 介绍

3.6.2 glance 配置

端口：

api 9191
registry 9292

1、修改/etc/glance/glance-api.conf 和/etc/glance/glance-registry.conf

```
[root@linux-node1 ~]# cat /etc/glance/glance-api.conf|grep -v "^#"|grep -v "^$"
[DEFAULT]
verbose=True
notification_driver = noop                              #galnce 不需要消息队列
[database]
connection=mysql://glance:glance@192.168.1.17/glance
[glance_store]
default_store=file
filesystem_store_datadir=/var/lib/glance/images/
[image_format]
[keystone_authtoken]
auth_uri = http://192.168.1.17:5000
auth_url = http://192.168.1.17:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = glance
password = glance
[matchmaker_redis]
[matchmaker_ring]
[oslo_concurrency]
[oslo_messaging_amqp]
[oslo_messaging_qpid]
[oslo_messaging_rabbit]
[oslo_policy]
[paste_deploy]
flavor=keystone
[store_type_location_strategy]
[task]
[taskflow_executor]

[root@linux-node1 ~]# cat /etc/glance/glance-registry.conf|grep -v "^#"|grep -v "^$"
[DEFAULT]
```

```
verbose=True
notification_driver = noop
[database]
connection=mysql://glance:glance@192.168.1.17/glance
[glance_store]
[keystone_authtoken]
auth_uri = http://192.168.1.17:5000
auth_url = http://192.168.1.17:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = glance
password = glance
[matchmaker_redis]
[matchmaker_ring]
[oslo_messaging_amqp]
[oslo_messaging_qpid]
[oslo_messaging_rabbit]
[oslo_policy]
[paste_deploy]
flavor=keystone
```

2、创建数据库表，同步数据库

```
[root@linux-node1 ~]# su -s /bin/sh -c "glance-manage db_sync" glance
[root@linux-node1 ~]# mysql -h 192.168.1.17 -uglance -p
```

3、 创建关于 glance 的 keystone 用户

```
[root@linux-node1 ~]# source admin-openrc.sh
[root@linux-node1 ~]# openstack user create --domain default --password=glance glance
[root@linux-node1 ~]# openstack role add --project service --user glance admin
```

4、启动 glance

```
[root@linux-node1 ~]#systemctl enable openstack-glance-api
[root@linux-node1 ~]#systemctl enable openstack-glance-registry
[root@linux-node1 ~]#systemctl start openstack-glance-api
[root@linux-node1 ~]#systemctl start openstack-glance-registry
[root@linux-node1 ~]# netstat -lnutp |grep 9191 #registry
tcp 0 0 0.0.0.0:9191 0.0.0.0:* LISTEN
24890/python2
[root@linux-node1 ~]# netstat -lnutp |grep 9292 #api
tcp 0 0 0.0.0.0:9292 0.0.0.0:* LISTEN
24877/python2
```

5、在 keystone 上注册

```
[root@linux-node1 ~]# source admin-openrc.sh
[root@linux-node1 ~]#openstack service create --name glance --description "OpenStack Image servic
e" image
[root@linux-node1 ~]#openstack endpoint create --region RegionOne image public http://192.168.1.1
7:9292
[root@linux-node1 ~]#openstack endpoint create --region RegionOne image internal http://192.168.1.
17:9292
[root@linux-node1 ~]#openstack endpoint create --region RegionOne image admin http://192.168.1.1
7:9292
```

6、添加 glance 环境变量并测试

```
[root@linux-node1 src]# echo "export OS_IMAGE_API_VERSION=2" | tee -a admin-openrc.sh demo-o
penrc.sh
[root@linux-node1 src]# glance image-list
+----+------+
| ID | Name |
+----+------+
+----+------+
```

7、 下载镜像并上传到 glance 【此处下载的 qcow2 格式镜像比较小，可以直接下载 ios 格式镜像，然后用 oz 工具制造】

```
[root@linux-node1 ~]# wget -q http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img
                          #也可以提前下载下来
[root@linux-node1 ~]# glance image-create --name "cirros" --file cirros-0.3.4-x86_64-disk.img --disk-format qcow2 --container-format bare --visibility public --progress
[=============================>] 100%
+------------------+--------------------------------------+
| Property | Value |
+------------------+--------------------------------------+
| checksum | ee1eca47dc88f4879d8a229cc70a07c6 |
| container_format | bare |
| created_at | 2015-12-17T04:11:02Z |
| disk_format | qcow2 |
| id | 2707a30b-853f-4d04-861d-e05b0f1855c8 |
| min_disk | 0 |
| min_ram | 0 |
| name | cirros |
| owner | 777f4f0108b1476eabc11e00dccaea9f |
| protected | False |
| size | 13287936 |
| status | active |
| tags | [] |
| updated_at | 2015-12-17T04:11:03Z |
| virtual_size | None |
| visibility | public |
+------------------+--------------------------------------+
```

下载 ios 格式镜像，需要用 OZ 工具制造 openstack 镜像，具体操作请见另一篇博客：

实际生产环境下，肯定要使用 ios 镜像进行制作了

或者直接下载 centos 的 qcow2 格式镜像进行上传，qcow2 格式镜像直接就可以在 openstack 里使用，不需要进行格式转换！
下载地址：http://cloud.centos.org/centos，可以到里面下载 centos5/6/7 的 qcow2 格式的镜像

```
[root@linux-node1 ~]#wget http://cloud.centos.org/centos/7/images/CentOS-7-x86_64-GenericCloud.qcow2
[root@linux-node1 ~]#glance image-create --name "CentOS-7-x86_64" --file CentOS-7-x86_64-GenericCloud.qcow2 --disk-format qcow2 --container-format bare --visibility public --progress
```

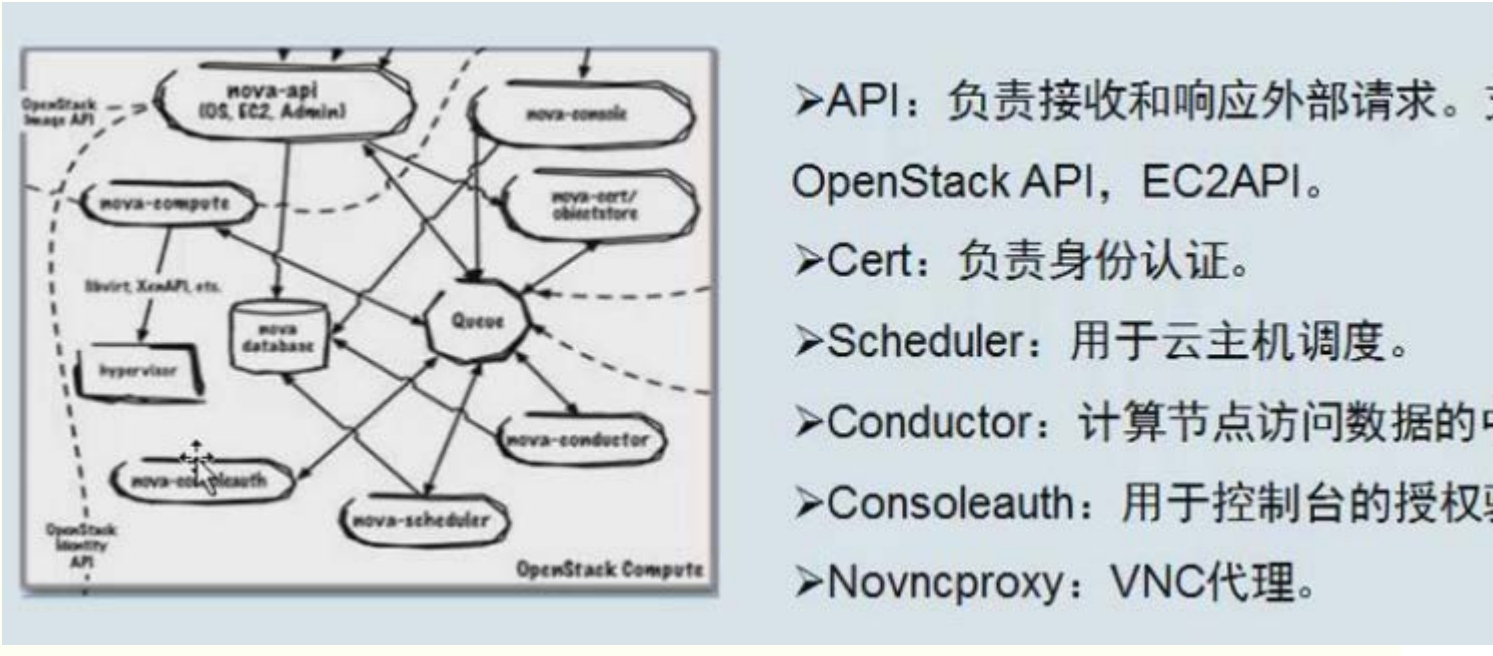
```
[root@linux-node1 ~]# glance image-list
+--------------------------------------+--------+
| ID | Name |
+--------------------------------------+--------+
| 2707a30b-853f-4d04-861d-e05b0f1855c8 | cirros |
+--------------------------------------+--------+

[root@linux-node1 ~]# ll /var/lib/glance/images/
总用量 12980
-rw-r-----. 1 glance glance 1569390592 Aug 26 12:50 35b36f08-eeb9-4a91-9366-561f0a308a1b
```

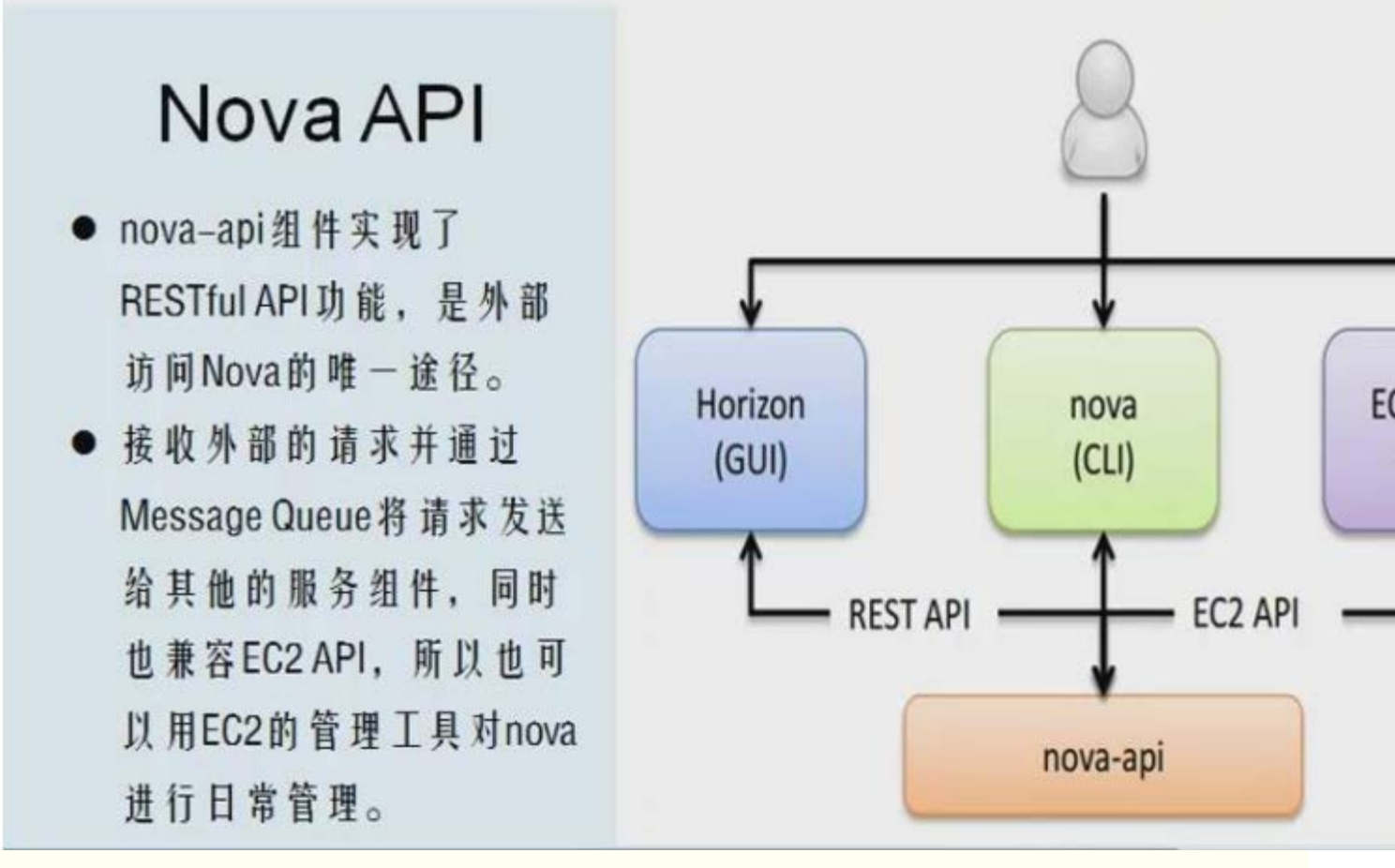
3.7 配置 nova 计算服务

3.7.1 nova 介绍

nova 必备的组件



nova API



nova scheduler

Nova scheduler

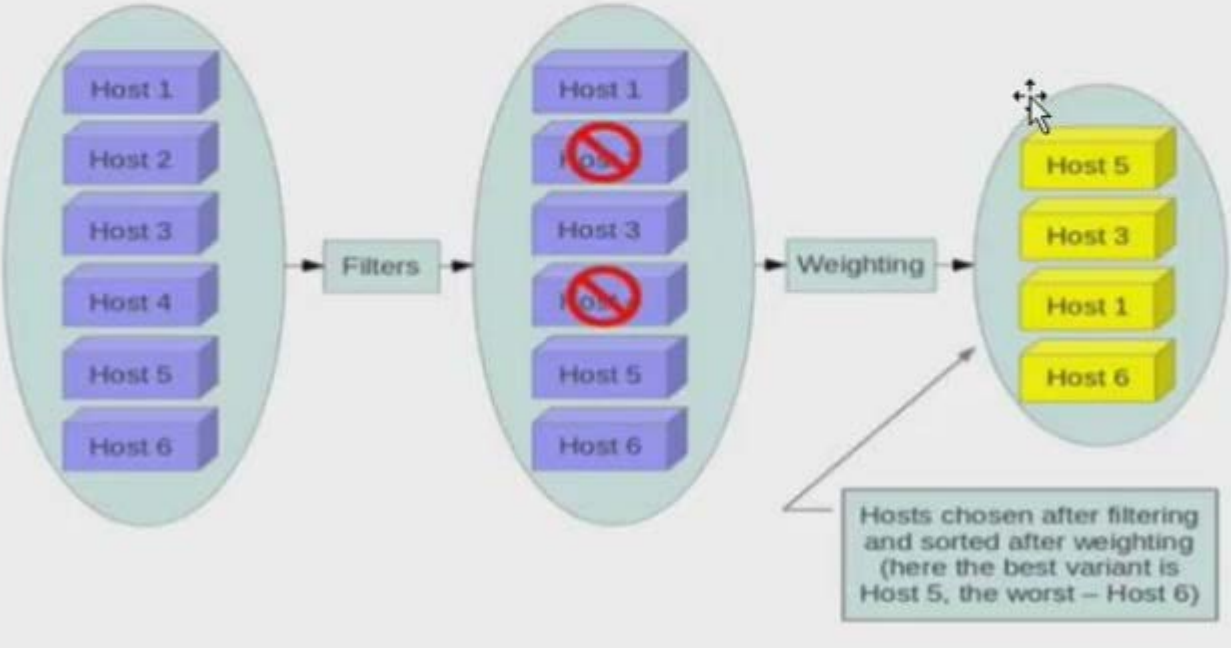
Nova Scheduler模块在openstack中的作用就是决策虚拟机建在哪个主机（计算节点）上。

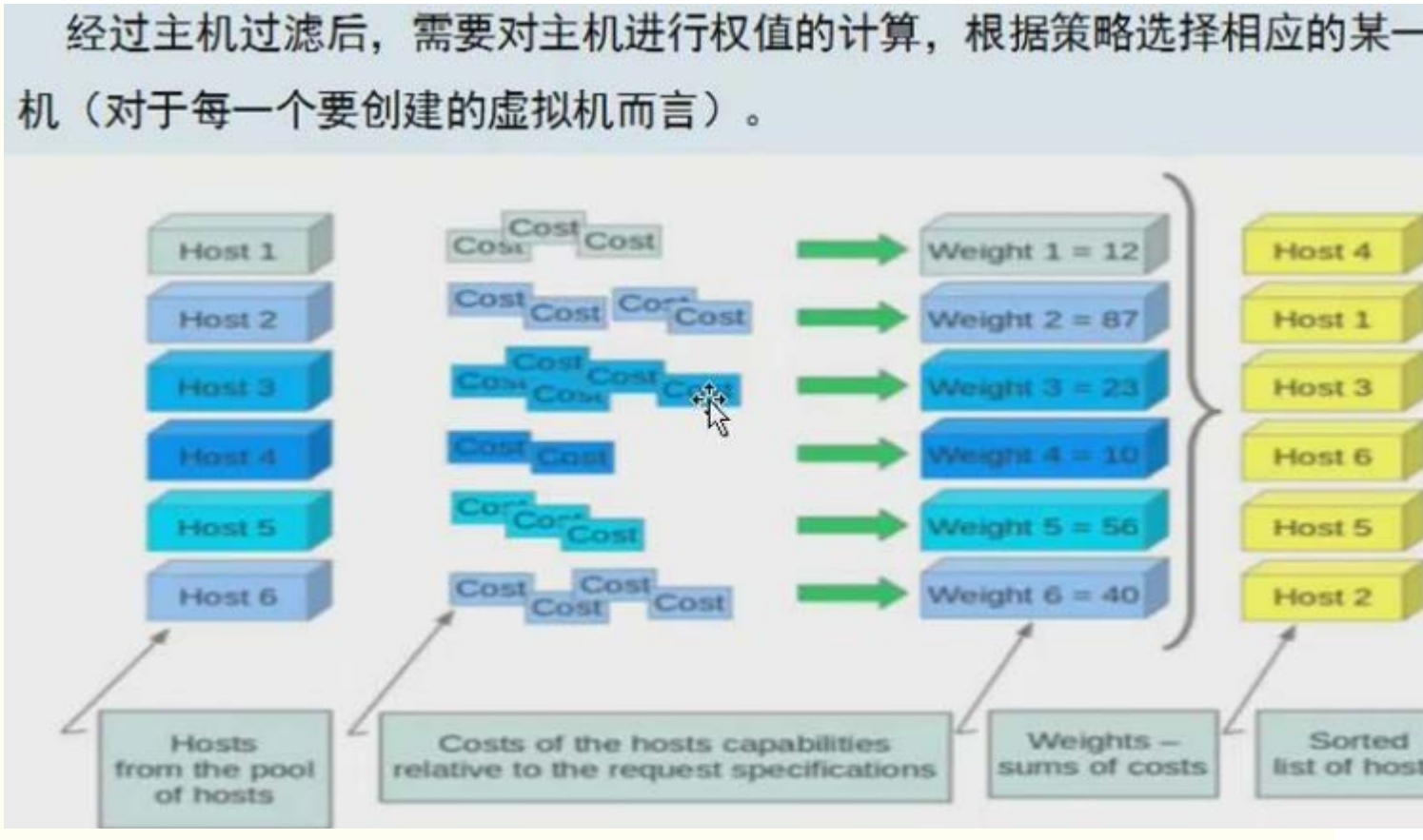
决策一个虚机应该调度到某物理节点，需要分两个步骤：

- 过滤（Fliter）
- 计算权值（Weight）

Nova Dashboard

Filter Scheduler首先得到未经过滤的主机列表，然后根据过滤属性，选择的计算节点主机。





3.7.2 Nova 控制节点配置

1、修改/etc/nova/nova.conf

```
[root@linux-node1 ~]# cat /etc/nova/nova.conf|grep -v "^#"|grep -v "^$"
[DEFAULT]
my_ip=192.168.1.17
enabled_apis=osapi_compute,metadata
auth_strategy=keystone
network_api_class=nova.network.neutronv2.api.API
linuxnet_interface_driver=nova.network.linux_net.NeutronLinuxBridgeInterfaceDriver
security_group_api=neutron
firewall_driver = nova.virt.firewall.NoopFirewallDriver
debug=true
verbose=true
rpc_backend=rabbit
allow_resize_to_same_host=True
scheduler_default_filters=RetryFilter,AvailabilityZoneFilter,RamFilter,ComputeFilter,ComputeCapabilitie
sFilter,ImagePropertiesFilter,ServerGroupAntiAffinityFilter,ServerGroupAffinityFilter
[api_database]
[barbican]
[cells]
[cinder]
[conductor]
[cors]
[cors.subdomain]
[database]
connection=mysql://nova:nova@192.168.1.17/nova
[ephemeral_storage_encryption]
[glance]
host=$my_ip
[guestfs]
[hyperv]
[image_file_url]
[ironic]
[keymgr]
[keystone_authtoken]
```

```
auth_uri = http://192.168.1.17:5000
auth_url = http://192.168.1.17:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = nova
password = nova
[libvirt]
virt_type=kvm                    #如果控制节点也作为计算节点（单机部署的话），这一行也添加上（这行是计算节点配置的）
[matchmaker_redis]
[matchmaker_ring]
[metrics]
[neutron]
url = http://192.168.1.17:9696
auth_url = http://192.168.1.17:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
region_name = RegionOne
project_name = service
username = neutron
password = neutron
service_metadata_proxy = True
metadata_proxy_shared_secret = neutron
lock_path=/var/lib/nova/tmp
[osapi_v21]
[oslo_concurrency]
[oslo_messaging_amqp]
[oslo_messaging_qpid]
[oslo_messaging_rabbit]
rabbit_host=192.168.1.17
rabbit_port=5672
rabbit_userid=openstack
rabbit_password=openstack
[oslo_middleware]
[rdp]
[serial_console]
[spice]
[ssl]
[trusted_computing]
[upgrade_levels]
[vmware]
[vnc]
novncproxy_base_url=http://58.68.250.17:6080/vnc_auto.html     #如果控制节点也作为计算节点（单机部署的话），这一行也添加上（这行是计算节点配置的），配置控制节点的公网 ip
vncserver_listen= $my_ip
vncserver_proxyclient_address= $my_ip
keymap=en-us        #如果控制节点也作为计算节点（单机部署的话），这一行也添加上（这行是计算节点配置的）
[workarounds]
[xenserver]
[zookeeper]

**********************************************************************
{网络部分为啥这么写：network_api_class=nova.network.neutronv2.api.API}
[root@linux-node1 ~]# ls /usr/lib/python2.7/site-packages/nova/network/neutronv2/api.py
/usr/lib/python2.7/site-packages/nova/network/neutronv2/api.py
这里面有一个 API 方法，其他配置类似
**********************************************************************
```

2、同步数据库

```
[root@linux-node1 ~]# su -s /bin/sh -c "nova-manage db sync" nova
[root@linux-node1 ~]# mysql -h 192.168.1.17 -unova -p 检查
```

3、创建 nova 的 keystone 用户

```
[root@linux-node1 ~]# openstack user create --domain default --password=nova nova
[root@linux-node1 ~]# openstack role add --project service --user nova admin
```

4、启动 nova 相关服务

```
[root@linux-node1 ~]#systemctl enable openstack-nova-api.service openstack-nova-cert.service open
stack-nova-consoleauth.service openstack-nova-scheduler.service openstack-nova-conductor.service o
penstack-nova-novncproxy.service
[root@linux-node1 ~]#systemctl start openstack-nova-api.service openstack-nova-cert.service openst
ack-nova-consoleauth.service openstack-nova-scheduler.service openstack-nova-conductor.service ope
nstack-nova-novncproxy.service
```

5、在 keystone 上注册

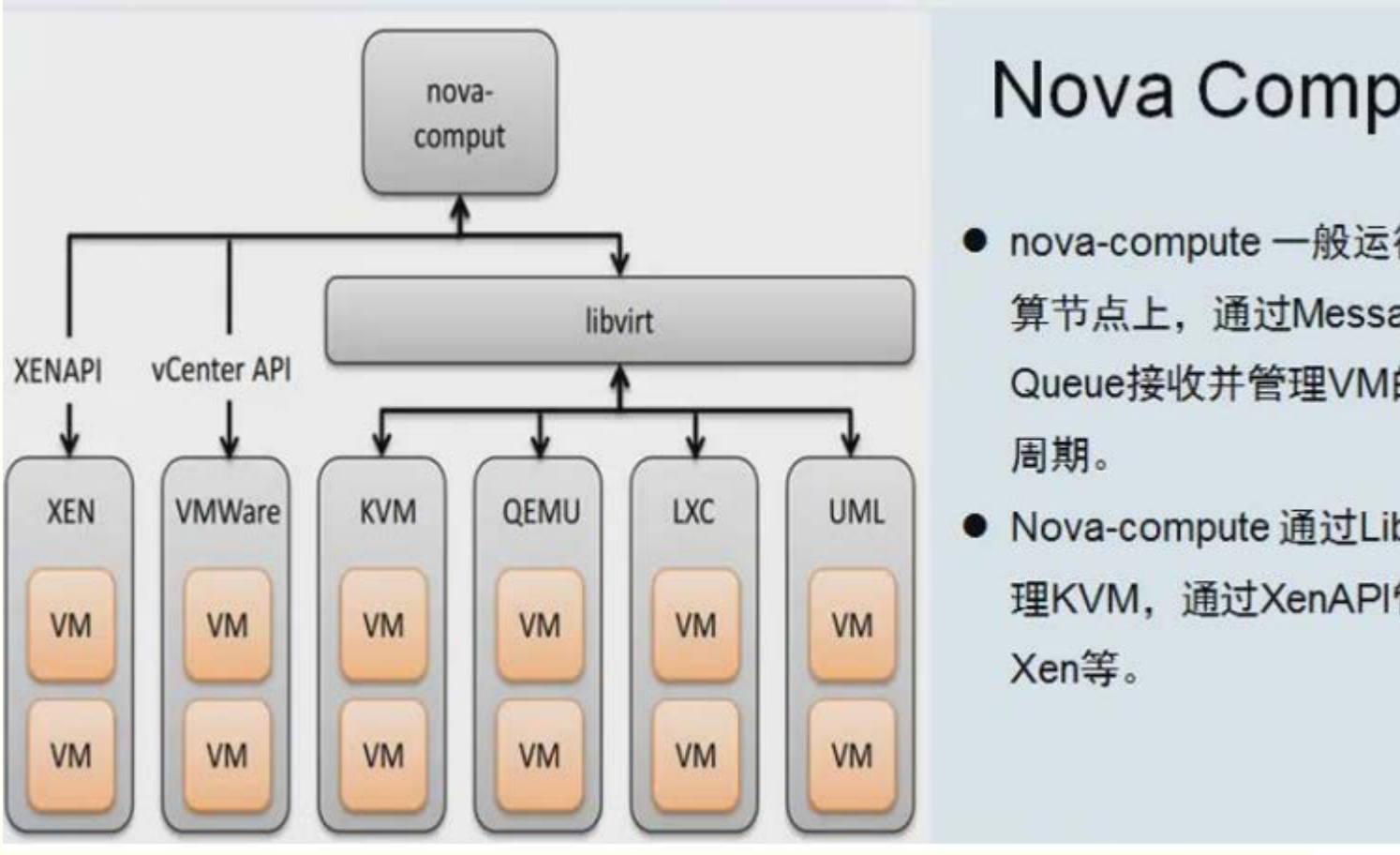
```
[root@linux-node1 ~]# source admin-openrc.sh
[root@linux-node1 ~]# openstack service create --name nova --description "OpenStack Compute" com
pute
[root@linux-node1 ~]# openstack endpoint create --region RegionOne compute public http://192.168.
1.17:8774/v2/%\(tenant_id\)s
[root@linux-node1 ~]# openstack endpoint create --region RegionOne compute internal http://192.16
8.1.17:8774/v2/%\(tenant_id\)s
[root@linux-node1 ~]# openstack endpoint create --region RegionOne compute admin http://192.168.
1.17:8774/v2/%\(tenant_id\)s
```

检查

```
[root@linux-node1 ~]# openstack host list
+--------------------------+-------------+----------+
| Host Name | Service | Zone |
+--------------------------+-------------+----------+
| linux-node1.oldboyedu.com | conductor | internal |
| linux-node1.oldboyedu.com | scheduler | internal |
| linux-node1.oldboyedu.com | consoleauth | internal |
| linux-node1.oldboyedu.com | cert | internal |
+--------------------------+-------------+----------+
```

3.7.3 nova 计算节点配置

1、 nova compute 介绍



2、修改配置文件/etc/nova/nova.conf

可以直接从 **node1** 拷贝到 **node2** 上

```
[root@linux-node1 ~]# scp /etc/nova/nova.conf 192.168.1.8:/etc/nova/
```

手动更改如下配置

```
[root@linux-node2 ~]# vim /etc/nova/nova.conf
my_ip=192.168.1.8
novncproxy_base_url=http://192.168.1.17:6080/vnc_auto.html
vncserver_listen=0.0.0.0
vncserver_proxyclient_address= $my_ip
keymap=en-us
[glance]
host=192.168.56.17
[libvirt]
virt_type=kvm                #虚拟机类型，默认是 kvm
```

3、启动服务

```
[root@linux-node2 ~]# systemctl enable libvirtd openstack-nova-compute
[root@linux-node2 ~]# systemctl start libvirtd openstack-nova-compute
```

4、在控制节点测试（计算节点上也行，需要环境变量）

```
[root@linux-node1 ~]# openstack host list
+-------------------------+-------------+----------+
| Host Name | Service | Zone |
+-------------------------+-------------+----------+
| linux-node1.oldboyedu.com | conductor | internal |
| linux-node1.oldboyedu.com | consoleauth | internal |
| linux-node1.oldboyedu.com | scheduler | internal |
| linux-node1.oldboyedu.com | cert | internal |
| linux-node2.oldboyedu.com | compute | nova |
+-------------------------+-------------+----------+

[root@linux-node1 ~]# nova image-list              #测试 glance 是否正常
+--------------------------------------+--------+--------+--------+
| ID | Name | Status | Server |
+--------------------------------------+--------+--------+--------+
| 2707a30b-853f-4d04-861d-e05b0f1855c8 | cirros | ACTIVE | |
```

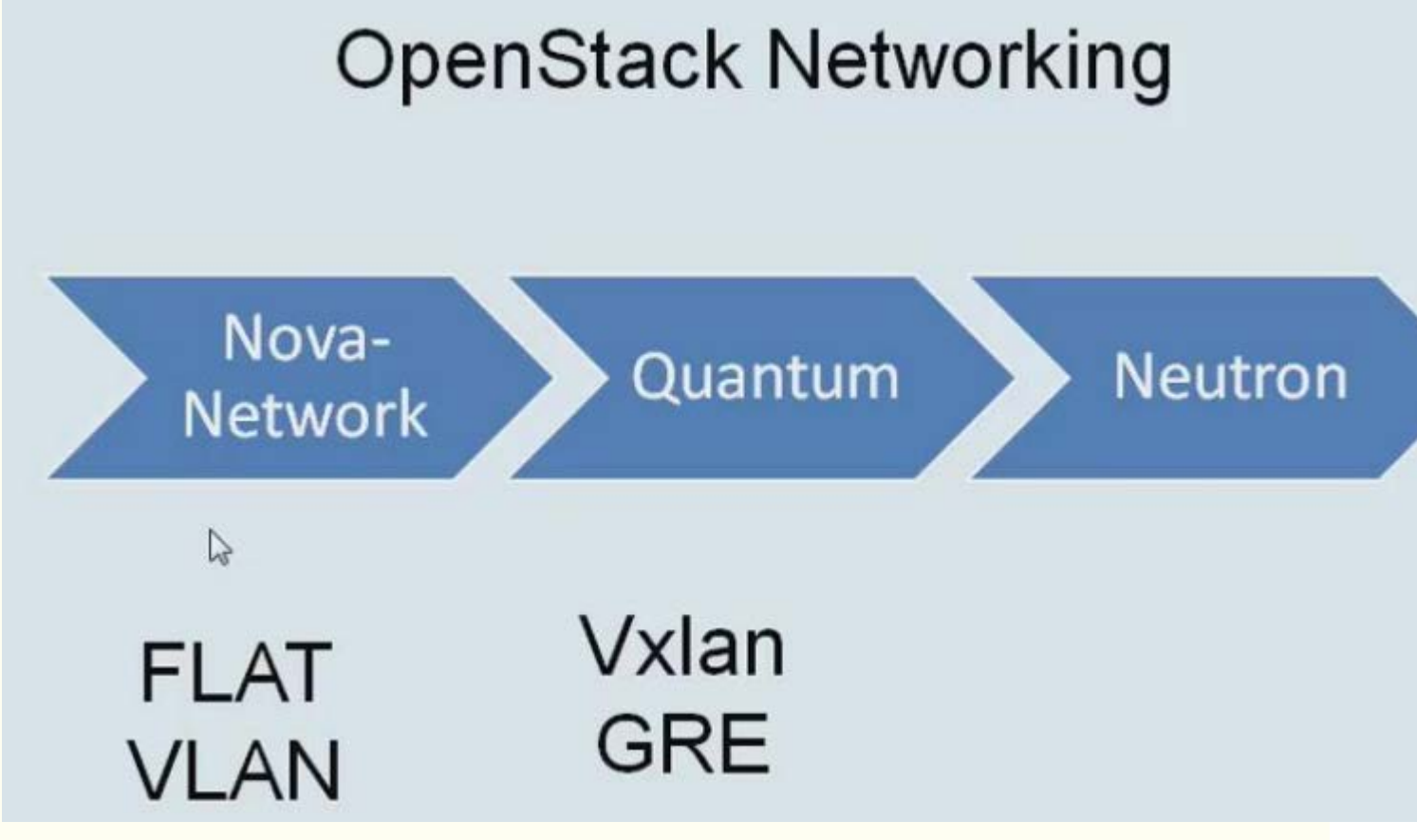
```
+----------------------------------+--------+--------+--------+
[root@linux-node1 ~]# nova endpoints                    #测试 keystone
WARNING: keystone has no endpoint in ! Available endpoints for this service:          #这一行告警不影响后面的操作
+-----------+----------------------------------+
| keystone | Value |
+-----------+----------------------------------+
| id | 02fed35802734518922d0ca2d672f469 |
| interface | internal |
| region | RegionOne |
| region_id | RegionOne |
| url | http://192.168.1.17:5000/v2.0 |
+-----------+----------------------------------+
+-----------+----------------------------------+
| keystone | Value |
+-----------+----------------------------------+
| id | 52b0a1a700f04773a220ff0e365dea45 |
| interface | public |
| region | RegionOne |
| region_id | RegionOne |
| url | http://192.168.1.17:5000/v2.0 |
+-----------+----------------------------------+
+-----------+----------------------------------+
| keystone | Value |
+-----------+----------------------------------+
| id | 88df7df6427d45619df192979219e65c |
| interface | admin |
| region | RegionOne |
| region_id | RegionOne |
| url | http://192.168.1.17:35357/v2.0 |
+-----------+----------------------------------+
WARNING: nova has no endpoint in ! Available endpoints for this service:
+-----------+-----------------------------------------------------------------+
| nova | Value |
+-----------+-----------------------------------------------------------------+
| id | 1a3115941ff54b7499a800c7c43ee92a |
| interface | internal |
| region | RegionOne |
| region_id | RegionOne |
| url | http://192.168.1.17:8774/v2/65a0c00638c247a0a274837aa6eb165f |
+-----------+-----------------------------------------------------------------+
+-----------+-----------------------------------------------------------------+
| nova | Value |
+-----------+-----------------------------------------------------------------+
| id | 5278f33a42754c9a8d90937932b8c0b3 |
| interface | admin |
| region | RegionOne |
| region_id | RegionOne |
| url | http://192.168.1.17:8774/v2/65a0c00638c247a0a274837aa6eb165f |
+-----------+-----------------------------------------------------------------+
+-----------+-----------------------------------------------------------------+
| nova | Value |
+-----------+-----------------------------------------------------------------+
| id | 8c4fa7b9a24949c5882949d13d161d36 |
| interface | public |
| region | RegionOne |
| region_id | RegionOne |
| url | http://192.168.1.17:8774/v2/65a0c00638c247a0a274837aa6eb165f |
+-----------+-----------------------------------------------------------------+
WARNING: glance has no endpoint in ! Available endpoints for this service:
+-----------+----------------------------------+
| glance | Value |
+-----------+----------------------------------+
```

```
| id | 31fbf72537a14ba7927fe9c7b7d06a65 |
| interface | admin |
| region | RegionOne |
| region_id | RegionOne |
| url | http://192.168.1.17:9292 |
+-----------+----------------------------------+
+-----------+----------------------------------+
| glance | Value |
+-----------+----------------------------------+
| id | be788b4aa2ce4251b424a3182d0eea11 |
| interface | public |
| region | RegionOne |
| region_id | RegionOne |
| url | http://192.168.1.17:9292 |
+-----------+----------------------------------+
+-----------+----------------------------------+
| glance | Value |
+-----------+----------------------------------+
| id | d0052712051a4f04bb59c06e2d5b2a0b |
| interface | internal |
| region | RegionOne |
| region_id | RegionOne |
| url | http://192.168.1.17:9292 |
+-----------+----------------------------------+
```

3.8 Neutron 网络服务

3.8.1 Neutron 介绍

neutron 由来

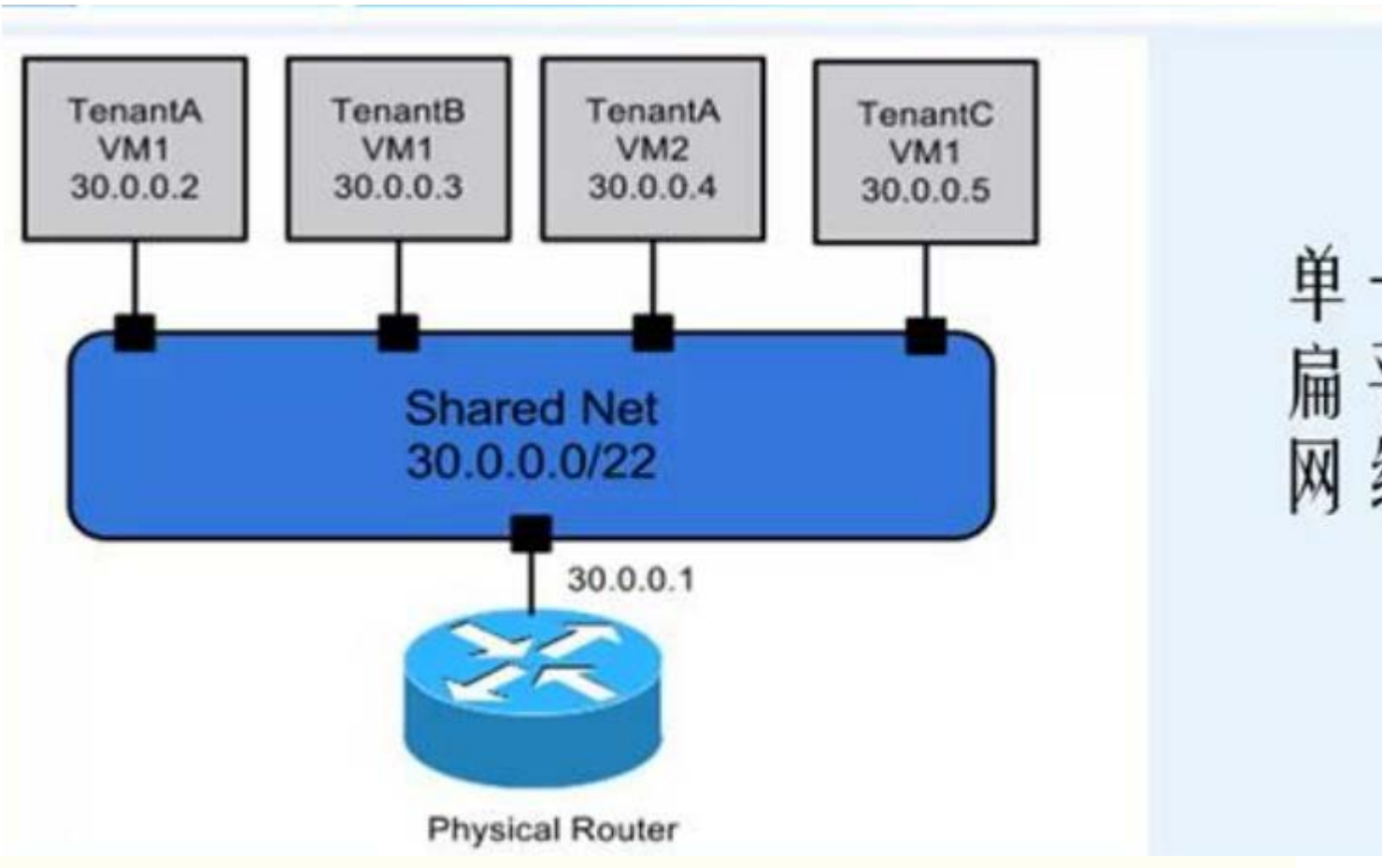


openstack 网络分类：

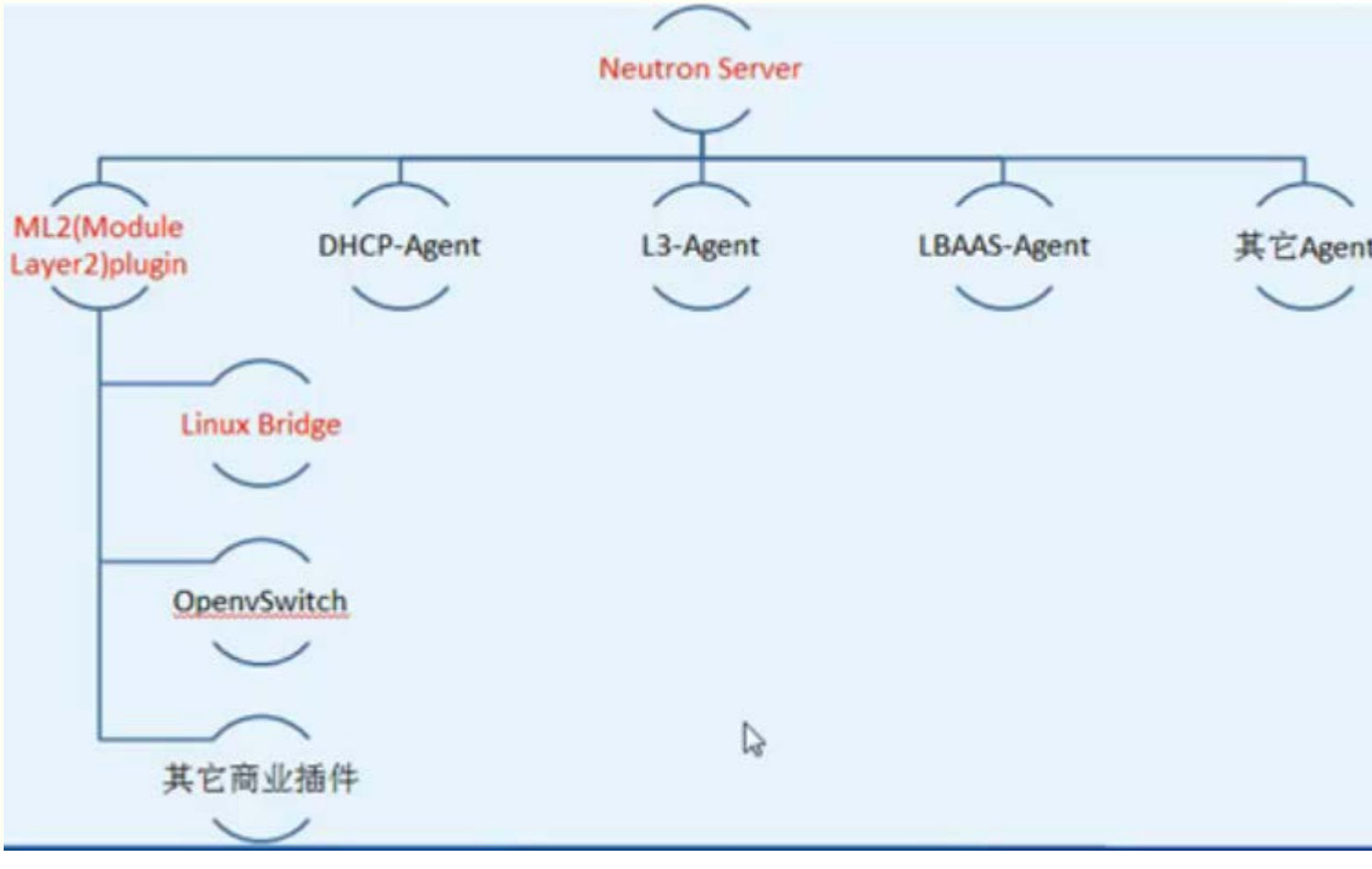
公共网络　向租户提供访问或者 API 调用
管理网络　云中物理机之间的通信
存储网络　云中存储的网络，如 ISCSI 或 GlusterFS 使用
服务网络　虚拟机内部使用的网络

OpenStack Networking

- **网络**：在实际的物理环境下，我们使用交换机或者集线器把多个计算机连接起来形网络。在Neutron的世界里，网络也是将多个不同的云主机连接起来。
- **子网**：在实际的物理环境下，在一个网络中。我们可以将网络划分成多为逻辑子网Neutron的世界里，子网也是隶属于网络下的。
- **端口**：是实际的物理环境下，每个子网或者每个网络，都有很多的端口，比如交换来供计算机连接。在Neutron的世界里端口也是隶属于子网下，云主机的网卡会对个端口上。
- **路由器**：在实际的网络环境下，不同网络或者不同逻辑子网之间如果需要进行通信通过路由器进行路由。在Neutron的实际里路由也是这个作用。用来连接不同的网子网。



Neutron 组件



ML2 可以实现下面多个网络插件的共存。
DHCP-Agent 分配 IP 地址。
L3-Agent 用来路由
LBASS-Agent 负载均衡

3.8.2 Neutron 控制节点配置（ 5 个配置文件）

1、修改**/etc/neutron/neutron.conf** 文件

```
[root@linux-node1 ~]# cat /etc/neutron/neutron.conf|grep -v "^#"|grep -v "^$"
[DEFAULT]
state_path = /var/lib/neutron
core_plugin = ml2
service_plugins = router
auth_strategy = keystone
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
nova_url = http://192.168.1.17:8774/v2
rpc_backend=rabbit
[matchmaker_redis]
[matchmaker_ring]
[quotas]
[agent]
[keystone_authtoken]
auth_uri = http://192.168.1.17:5000
auth_url = http://192.168.1.17:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = neutron
password = neutron
admin_tenant_name = %SERVICE_TENANT_NAME%
admin_user = %SERVICE_USER%
admin_password = %SERVICE_PASSWORD%
[database]
connection = mysql://neutron:neutron@192.168.1.17:3306/neutron
[nova]
auth_url = http://192.168.1.17:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
region_name = RegionOne
project_name = service
username = nova
password = nova
[oslo_concurrency]
lock_path = $state_path/lock
[oslo_policy]
[oslo_messaging_amqp]
[oslo_messaging_qpid]
[oslo_messaging_rabbit]
rabbit_host = 192.168.1.17
rabbit_port = 5672
rabbit_userid = openstack
rabbit_password = openstack
[qos]
```

2、 配置/etc/neutron/plugins/ml2/ml2_conf.ini

```
[root@linux-node1 ~]# cat /etc/neutron/plugins/ml2/ml2_conf.ini|grep -v "^#"|grep -v "^$"
[ml2]
```

```
type_drivers = flat,vlan,gre,vxlan,geneve
tenant_network_types = vlan,gre,vxlan,geneve
mechanism_drivers = openvswitch,linuxbridge
extension_drivers = port_security
[ml2_type_flat]
flat_networks = physnet1
[ml2_type_vlan]
[ml2_type_gre]
[ml2_type_vxlan]
[ml2_type_geneve]
[securitygroup]
enable_ipset = True
```

3、配置/etc/neutron/plugins/ml2/ linuxbridge_agent.ini

```
[root@linux-node1 ~]# cat /etc/neutron/plugins/ml2/linuxbridge_agent.ini|grep -v "^#"|grep -v "^$"
[linux_bridge]
physical_interface_mappings = physnet1:em2
[vxlan]
enable_vxlan = false
[agent]
prevent_arp_spoofing = True
[securitygroup]
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
enable_security_group = True
```

4、修改/etc/neutron/dhcp_agent.ini

```
[root@linux-node1 ~]# cat /etc/neutron/dhcp_agent.ini|grep -v "^#"|grep -v "^$"
[DEFAULT]
interface_driver = neutron.agent.linux.interface.BridgeInterfaceDriver
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
enable_isolated_metadata = true
[AGENT]
```

5、修改/etc/neutron/metadata_agent.ini

```
[root@linux-node1 ~]# cat /etc/neutron/metadata_agent.ini|grep -v "^#"|grep -v "^$"
[DEFAULT]
auth_uri = http://192.168.1.17:5000
auth_url = http://192.168.1.17:35357
auth_region = RegionOne
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = neutron
password = neutron
nova_metadata_ip = 192.168.1.17
metadata_proxy_shared_secret = neutron
admin_tenant_name = %SERVICE_TENANT_NAME%
admin_user = %SERVICE_USER%
admin_password = %SERVICE_PASSWORD%
[AGENT]
```

6、创建连接并创建 keystone 的用户

```
[root@linux-node1 ~]# ln -s /etc/neutron/plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
[root@linux-node1 ~]# openstack user create --domain default --password=neutron neutron
[root@linux-node1 ~]# openstack role add --project service --user neutron admin
```

7、更新数据库

```
[root@linux-node1 ~]# 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
```

8、注册 keystone

```
[root@linux-node1 ~]# source admin-openrc.sh
[root@linux-node1 ~]# openstack service create --name neutron --description "OpenStack Networking
" network
```

```
[root@linux-node1 ~]# openstack endpoint create --region RegionOne network public http://192.168.1.17:9696
[root@linux-node1 ~]# openstack endpoint create --region RegionOne network internal http://192.168.1.17:9696
[root@linux-node1 ~]# openstack endpoint create --region RegionOne network admin http://192.168.1.17:9696
```

9、 启动服务并检查

因为 neutron 和 nova 有联系，做 neutron 时修改 nova 的配置文件，上面 nova.conf 已经做了 neutron 的关联配置，所以要重启 openstack-nova-api 服务。

这里将 nova 的关联服务都一并重启了：

```
[root@linux-node1 ~]# systemctl restart openstack-nova-api.service openstack-nova-cert.service openstack-nova-consoleauth.service openstack-nova-scheduler.service openstack-nova-conductor.service openstack-nova-novncproxy.service
```

启动 neutron 相关服务

```
[root@linux-node1 ~]# systemctl enable neutron-server.service neutron-linuxbridge-agent.service neutron-dhcp-agent.service neutron-metadata-agent.service
[root@linux-node1 ~]# systemctl start neutron-server.service neutron-linuxbridge-agent.service neutron-dhcp-agent.service neutron-metadata-agent.service
```

检查

```
[root@linux-node1 ~]# neutron agent-list
+--------------------------------------+--------------------+-----------------+-------+---------------+---------------------------+
| id | agent_type | host | alive | admin_state_up | binary |
+--------------------------------------+--------------------+-----------------+-------+---------------+---------------------------+
| 385cebf9-9b34-4eca-b780-c515dbc7eec0 | Linux bridge agent | openstack-server | :-) | True | neutron-linuxbridge-agent |
| b3ff8ffe-1ff2-4659-b823-331def4e6a93 | DHCP agent | openstack-server | :-) | True | neutron-dhcp-agent |
| b5bed625-47fd-4e79-aa55-01cf8a8cc577 | Metadata agent | openstack-server | :-) | True | neutron-metadata-agent |
+--------------------------------------+--------------------+-----------------+-------+---------------+---------------------------+
```

查看注册信息

```
[root@openstack-server src]# openstack endpoint list
+----------------------------------+-----------+--------------+--------------+---------+-----------+-----------------------------------------+
| ID | Region | Service Name | Service Type | Enabled | Interface | URL |
+----------------------------------+-----------+--------------+--------------+---------+-----------+-----------------------------------------+
| 02fed35802734518922d0ca2d672f469 | RegionOne | keystone | identity | True | internal | http://192.168.1.17:5000/v2.0 |
| 1a3115941ff54b7499a800c7c43ee92a | RegionOne | nova | compute | True | internal | http://192.168.1.17:8774/v2/%(tenant_id)s |
| 31fbf72537a14ba7927fe9c7b7d06a65 | RegionOne | glance | image | True | admin | http://192.168.1.17:9292 |
| 5278f33a42754c9a8d90937932b8c0b3 | RegionOne | nova | compute | True | admin | http://192.168.1.17:8774/v2/%(tenant_id)s |
| 52b0a1a700f04773a220ff0e365dea45 | RegionOne | keystone | identity | True | public | http://192.168.1.17:5000/v2.0 |
| 88df7df6427d45619df192979219e65c | RegionOne | keystone | identity | True | admin | http://192.168.1.17:35357/v2.0 |
| 8c4fa7b9a24949c5882949d13d161d36 | RegionOne | nova | compute | True | public | http://192.168.1.17:8774/v2/%(tenant_id)s |
| be788b4aa2ce4251b424a3182d0eea11 | RegionOne | glance | image | True | public | http://192.168.1.17:9292 |
| c059a07fa3e141a0a0b7fc2f46ca922c | RegionOne | neutron | network | True | public | http://192.168.1.17:9696 |
| d0052712051a4f04bb59c06e2d5b2a0b | RegionOne | glance | image | True | internal | http://192.168.1.17:9292 |
| ea325a8a2e6e4165997b2e24a8948469 | RegionOne | neutron | network | True | internal | http://19
```

```
2.168.1.17:9696 |
| ffdec11ccf024240931e8ca548876ef0 | RegionOne | neutron | network | True | admin | http://192.16
8.1.17:9696 |
+----------------------------------+----------+-------------+-------------+---------+----------+---------
---------------------------------+
```

3.8.3 Neutron 计算节点配置

1、修改相关配置文件

从 node1 上直接拷贝

```
[root@linux-node1 ~]# scp /etc/neutron/neutron.conf 192.168.1.8:/etc/neutron/
[root@linux-node1 ~]# scp /etc/neutron/plugins/ml2/linuxbridge_agent.ini 192.168.1.8:/etc/neutron/
plugins/ml2/
[root@linux-node1 ~]# scp /etc/neutron/plugins/ml2/ml2_conf.ini 192.168.1.8:/etc/neutron/plugins/
ml2/
```

修改计算节点的 nova 配置文件中 neutron 部分， 并重启 openstack-nova-compute 服务， 因为上面 nova 计算节点也是从控制节点拷贝的，此处无需操作

2、 创建软连接并启动服务

```
[root@linux-node2 ~]# ln -s /etc/neutron/plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
[root@linux-node2 ~]# systemctl enable neutron-linuxbridge-agent.service
[root@linux-node2 ~]# systemctl start neutron-linuxbridge-agent.service
```

检查

```
[root@linux-node1 ~]# neutron agent-list
+--------------------------------------+--------------------+-----------------+-------+---------------+-----
-------------------+
| id | agent_type | host | alive | admin_state_up | binary |
+--------------------------------------+--------------------+-----------------+-------+---------------+-----
-------------------+
| 385cebf9-9b34-4eca-b780-c515dbc7eec0 | Linux bridge agent | openstack-server | :-) | True | neutr
on-linuxbridge-agent |
| b3ff8ffe-1ff2-4659-b823-331def4e6a93 | DHCP agent | openstack-server | :-) | True | neutron-dhcp-
agent |
| b5bed625-47fd-4e79-aa55-01cf8a8cc577 | Metadata agent | openstack-server | :-) | True | neutron-
metadata-agent |
+--------------------------------------+--------------------+-----------------+-------+---------------+-----
-------------------+
```

3.9 创建虚拟机

3.9.1 创建桥接网络

1、 创建网络

```
[root@linux-node1 ~]# source admin-openrc.sh                    #在哪个项目下创建虚拟机，这里选择在 dem
o 下创建；也可以在 admin 下
[root@linux-node1 ~]# neutron net-create flat --shared --provider:physical_network physnet1 --provi
der:network_type flat
```

2、 创建子网（填写宿主机的内网网关，下面 DNS 和内网网关可以设置成宿主机的内网 ip，下面 192.168.1.100-200 是分配给虚拟机的 ip 范围）

```
[root@linux-node1 ~]# neutron subnet-create flat 192.168.1.0/24 --name flat-subnet --allocation-pool
 start=192.168.1.100,end=192.168.1.200 --dns-nameserver 192.168.1.1 --gateway 192.168.1.1
```

3、 查看子网

```
[root@linux-node1 ~]# neutron net-list
+--------------------------------------+------+--------------------------------------------------+
| id | name | subnets |
+--------------------------------------+------+--------------------------------------------------+
| 1d9657f6-de9e-488f-911f-020c8622fe78 | flat | c53da14a-01fe-4f6c-8485-232489deaa6e 192.168.1.
0/24 |
+--------------------------------------+------+--------------------------------------------------+

[root@linux-node1 ~]# neutron subnet-list
+--------------------------------------+------------+--------------+--------------------------------------
-----------+
| id | name | cidr | allocation_pools |
```

```
+--------------------------------------+------------+--------------+--------------------------------------
----------+
| c53da14a-01fe-4f6c-8485-232489deaa6e | flat-subnet | 192.168.1.0/24 | {"start": "192.168.1.100",
 "end": "192.168.1.200"} |
+--------------------------------------+------------+--------------+--------------------------------------
----------+
```

需要关闭 VMware 的 dhcp

3.9.2 创建虚拟机（为 vm 分配内网 ip，后续利用 squid 代理或宿主机 NAT 端口转发进行对外或对内访问）

1、创建 key

```
[root@linux-node1 ~]# source demo-openrc.sh
```

（这是在 demo 账号下创建虚拟机；要是在 admin 账号下创建虚拟机，就用 source admin-openrc.sh）

```
[root@linux-node1 ~]# ssh-keygen -q -N ""
```

2、将公钥添加到虚拟机

```
[root@linux-node1 ~]# nova keypair-add --pub-key /root/.ssh/id_rsa.pub mykey
[root@linux-node1 ~]# nova keypair-list
+-------+-------------------------------------------------+
| Name | Fingerprint |
+-------+-------------------------------------------------+
| mykey | cd:7a:1e:cd:c0:43:9b:b1:f4:3b:cf:cd:5e:95:f8:00 |
+-------+-------------------------------------------------+
```

3、创建安全组

```
[root@linux-node1 ~]# nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
[root@linux-node1 ~]# nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
```

4、 创建虚拟机

查看支持的虚拟机类型

```
[root@linux-node1 ~]# nova flavor-list
+----+-----------+-----------+------+-----------+------+-------+------------+-----------+
| ID | Name | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
+----+-----------+-----------+------+-----------+------+-------+------------+-----------+
| 1 | m1.tiny | 512 | 1 | 0 | | 1 | 1.0 | True |
| 2 | m1.small | 2048 | 20 | 0 | | 1 | 1.0 | True |
| 3 | m1.medium | 4096 | 40 | 0 | | 2 | 1.0 | True |
| 4 | m1.large | 8192 | 80 | 0 | | 4 | 1.0 | True |
| 5 | m1.xlarge | 16384 | 160 | 0 | | 8 | 1.0 | True |
+----+-----------+-----------+------+-----------+------+-------+------------+-----------+
```

查看镜像

```
[root@linux-node1 ~]# nova image-list
+--------------------------------------+--------+--------+--------+
| ID | Name | Status | Server |
+--------------------------------------+--------+--------+--------+
| 2707a30b-853f-4d04-861d-e05b0f1855c8 | cirros | ACTIVE | |
+--------------------------------------+--------+--------+--------+
```

查看网络

```
[root@linux-node1 ~]# neutron net-list
+--------------------------------------+------+------------------------------------------------------+
| id | name | subnets |
+--------------------------------------+------+------------------------------------------------------+
| 1d9657f6-de9e-488f-911f-020c8622fe78 | flat | c53da14a-01fe-4f6c-8485-232489deaa6e 192.1
68.1.0/24 |
+--------------------------------------+------+------------------------------------------------------+
```

创建虚拟机 【这一步容易报错，一般都是由于上面的 nova.conf 配置填写有误所致】

```
[root@linux-node1 ~]# nova boot --flavor m1.tiny --image cirros --nic net-id=1d9657f6-de9e-488
f-911f-020c8622fe78 --security-group default --key-name mykey hello-instance
```

5、查看虚拟机

```
[root@linux-node1 ~]# nova list
+--------------------------------------+--------------+--------+------------+-------------+----------------
--+
| ID | Name | Status | Task State | Power State | Networks |
```

```
+--------------------------------------+--------------+--------+-----------+------------+-----------------
--+
| 7a6215ac-aea7-4e87-99a3-b62c06d4610e | hello-instance| ACTIVE | - | Running | flat=192.168.1.1
02 |
+--------------------------------------+--------------+--------+-----------+------------+-----------------
--+
```

****

如果要删除虚拟机（利用虚拟机 ID 进行删除）

```
[root@linux-node1 ~]# nova delete 7a6215ac-aea7-4e87-99a3-b62c06d4610e
```

****

```
[root@linux-node1 src]# nova list
+--------------------------------------+---------------+--------+-----------+------------+-----------------
---+
| ID | Name | Status | Task State | Power State | Networks |
+--------------------------------------+---------------+--------+-----------+------------+-----------------
---+
| 007db18f-ae3b-463a-b86d-9a8455a21e2d | hello-instance | ACTIVE | - | Running | flat=192.168.1.1
01 |
+--------------------------------------+---------------+--------+-----------+------------+-----------------
---+
```

[root@linux-node1 ~]# ssh cirros@192.168.1.101 登录查看

****

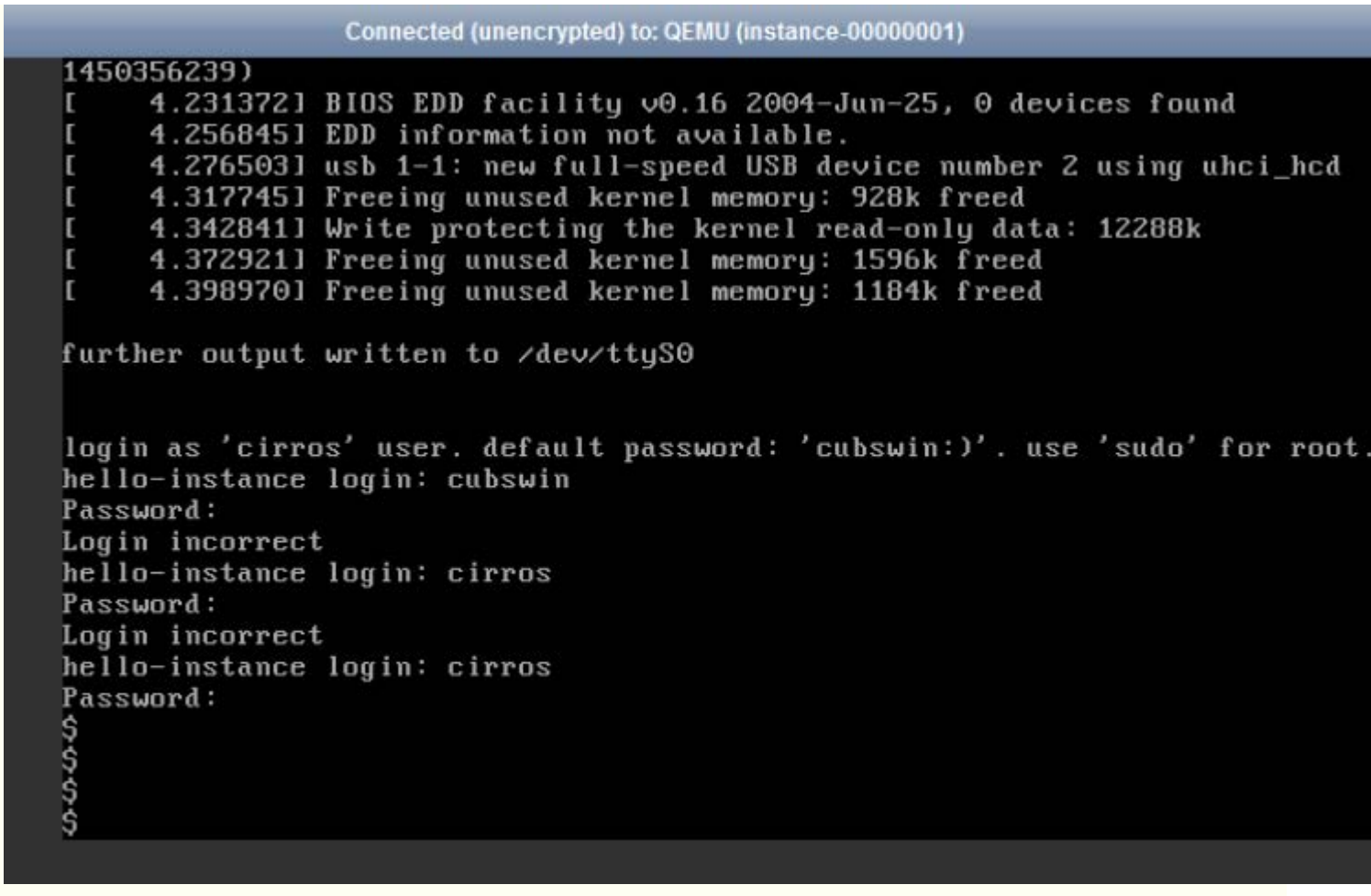
上面创建虚拟机的时候，openstack 在 neutron 组网内是采用 dhcp-agent 自动分配 ip 的!

可以在创建虚拟机的时候，指定固定 ip

****

6、 web 界面打开虚拟机

```
[root@linux-node1 ~]# nova get-vnc-console hello-instance novnc
+-------+-----------------------------------------------------------------------------------
-+
| Type | Url
| +
-------+-----------------------------------------------------------------------------------
-+
| novnc | http://58.68.250.17:6080/vnc_auto.html?token=303d5a78-c85f-4ed9-93b6-be9d5d28fba6 |
      #访问这个链接即可打开 vnc 界面
+-------+-----------------------------------------------------------------------------------
-+
```



4.0 安装 dashboard，登陆 web 管理界面

```
[root@linux-node1 ~]# yum install openstack-dashboard -y
[root@linux-node1 ~]# vim /etc/openstack-dashboard/local_settings          #按照下面几行进行配置修改
OPENSTACK_HOST = "192.168.1.17"                          #更改为 keystone 机器地址
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"                 #默认的角色
ALLOWED_HOSTS = ['*']                                    #允许所有主机访问
CACHES = {
'default': {
'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
'LOCATION': '192.168.1.17:11211',                        #连接 memcached
}
}
#CACHES = {
# 'default': {
# 'BACKEND': 'django.core.cache.backends.locmem.LocMemCache',
# }
#}
TIME_ZONE = "Asia/Shanghai"                              #设置时区
```

重启 httpd 服务

```
[root@linux-node1 ~]# systemctl restart httpd
```

web 界面登录访问 dashboard

http://58.68.250.17/dashboard/

用户密码 demo 或者 admin（管理员）

如果要修改 dashboard 的访问端口（比如将 80 端口改为 8080 端口），则需要修改下面两个配置文件：

1）vim /etc/httpd/conf/httpd.conf

将 80 端口修改为 8080 端口

```
Listen 8080
ServerName 192.168.1.17:8080
```

2）vim /etc/openstack-dashboard/local_settings #将下面两处的端口由 80 改为 8080

```
'from_port': '8080',
'to_port': '8080',
```

然后重启 http 服务：

```
systemctl restart httpd
```

如果开启了防火墙，还需要开通 8080 端口访问规则

这样，dashboard 访问 url：

http://58.68.250.17:8080/dashboard

前面建立了两个账号：admin 和 demo，两个账号都可以登陆 web！只不过， admin 是管理员账号， admin 登陆后可以看到其他账号下的状态
demo 等普通账号登陆后只能看到自己的状态
注意：
上面的 Rabbit 账号 admin 和 openstack 是消息队列的 web 登陆账号。
比如一下子要建立 10 个虚拟机的指令，但是当前资源处理不过来，就通过 Rabbit 进行排队！！

修改 OpenStack 中 dashboard 用户登陆密码的方法：

登陆 dashboard：

创建虚拟机的时候，我们可以自己定义虚拟机的类型（即配置）。

登陆 openstack 的 web 管理界面里进行自定义，也可以将之前的删除。

查看上传到 glance 的镜像

查看创建的虚拟机实例

自定义虚拟主机类型，设置如下：

（如果想让虚拟机有空闲磁盘空间，用于新建分区之用，则可以在这里分配临时磁盘）

创建云主机类型

主机类型信息 * | 云主机类型访问

名称 *

kvm002

云主机类型定义RAM和磁盘的大小、核数，以及其他资源，在用户部署实例的时候使用。

ID

auto

虚拟内核 *

2

内存 (MB) *

6144

根磁盘(GB) *

10

临时磁盘(GB)

0

Swap磁盘(MB)

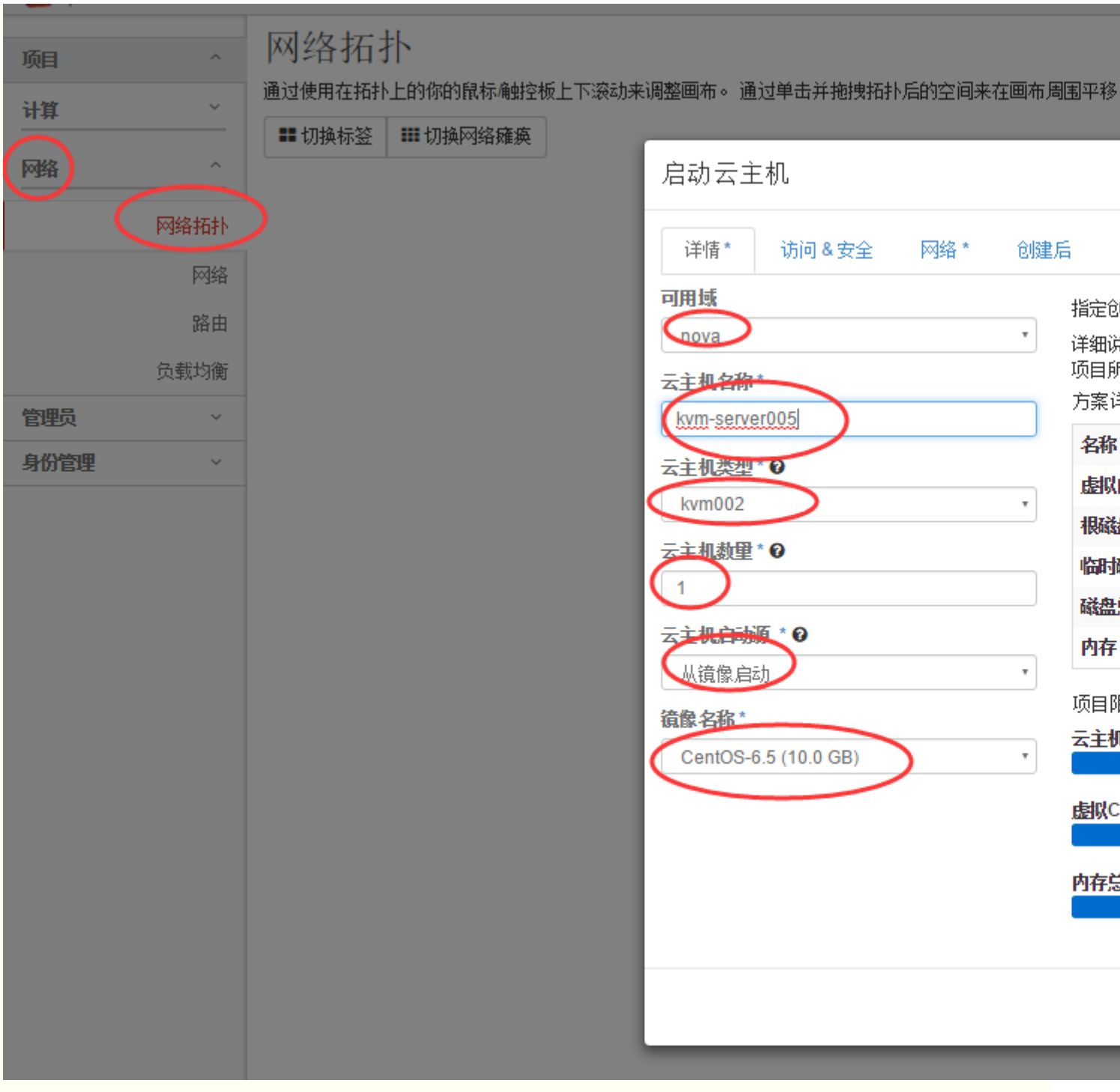
1000

取消 | 创建云主机类型

我创建了四个虚拟机实例，采用的是同一个虚拟主机类型（即上面的 kvm002），四个实例总共占用宿主机 40G 的空间。

登陆到 openstack，可以看到，左侧一共有四个标签栏：

可以登陆 dashboard 界面，在“计算”->“实例”里选择“启动云主机"或者“计算->网络->网络拓扑”里选择“启动虚拟机”就可以再创建一个虚拟机
也可以按照快照再启动（创建）一个虚拟机，不过这样启动起来的虚拟机是一个新的 ip（快照前的源虚拟机就要关机了）



查看实例，发现 kvm-server005 虚拟机已经创建成功了。默认创建后的 ip 是 dhcp 自动分配的，可以登陆虚拟机改成 static 静态 ip

在 openstack 中重启实例有两种，分别被称为“软重启”和“硬重启”。所谓的软重启会尝试正常关机并重启实例，硬重启会直接将实例“断电”并重启。也就是说硬重启会“关闭”电源。其具体命令如下：
默认情况下，如果您通过 nova 重启，执行的是软重启。

```
$ nova reboot SERVER
```

如果您需要执行硬重启，添加--hard 参数即可：

```
$ nova reboot --hard SERVER
```

nova 命令管理虚拟机：

```
$ nova list #查看虚拟机
$ nova stop [vm-name]或[vm-id] #关闭虚拟机
$ nova start [vm-name]或[vm-id] #启动虚拟机
$ nova suspend [vm-name]或[vm-id] #暂停虚拟机
$ nova resume [vm-name]或[vm-id] #启动暂停的虚拟机
$ nova delete [vm-name]或[vm-id] #删除虚拟机

$nova-manage service list    #检查服务是否正常

[root@openstack-server ~]# source /usr/local/src/admin-openrc.sh
[root@openstack-server ~]# nova list
+--------------------------------------+---------------+--------+------------+-------------+-----------------
---+
| ID | Name | Status | Task State | Power State | Networks |
+--------------------------------------+---------------+--------+------------+-------------+-----------------
---+
| 11e7ad7f-c0a8-482b-abca-3a4b7cfdd55d | hello-instance | ACTIVE | - | Running | flat=192.168.1.10
7 |
| 67f71703-c32c-4bf1-8778-b2a6600ad34a | kvm-server0 | ACTIVE | - | Running | flat=192.168.1.12
0 |
+--------------------------------------+---------------+--------+------------+-------------+-----------------
---+
[root@openstack-server ~]# ll /var/lib/nova/instances/          #下面是虚拟机的存放路径
total 8
drwxr-xr-x. 2 nova nova 85 Aug 29 15:22 11e7ad7f-c0a8-482b-abca-3a4b7cfdd55d
drwxr-xr-x. 2 nova nova 85 Aug 29 15:48 67f71703-c32c-4bf1-8778-b2a6600ad34a
drwxr-xr-x. 2 nova nova 80 Aug 29 15:40 _base
-rw-r--r--. 1 nova nova 39 Aug 29 16:44 compute_nodes
drwxr-xr-x. 2 nova nova 4096 Aug 29 13:58 locks
```

---

virsh 命令行管理虚拟机：

```
[root@openstack-server ~]# virsh list #查看虚拟机

 Id Name State
----------------------------------------------------
 9 instance-00000008 running
 41 instance-00000015 running
[root@openstack-server ~]# ll /etc/libvirt/qemu/ #虚拟机文件
total 16
-rw-------. 1 root root 4457 Aug 26 17:46 instance-00000008.xml
-rw-------. 1 root root 4599 Aug 29 15:40 instance-00000015.xml
drwx------. 3 root root 22 Aug 24 12:06 networks
```

其中：

virsh list #显示本地活动虚拟机
virsh list --all #显示本地所有的虚拟机（活动的+不活动的）
virsh define instance-00000015.xml #通过配置文件定义一个虚拟机（这个虚拟机还不是活动的）
virsh edit instance-00000015 # 编辑配置文件（一般是在刚定义完虚拟机之后）
virsh start instance-00000015 #启动名字为 ubuntu 的非活动虚拟机
virsh reboot instance-00000015 #重启虚拟机
virsh create instance-00000015.xml #创建虚拟机（创建后，虚拟机立即执行，成为活动主机）
virsh suspend instance-00000015 #暂停虚拟机
virsh resume instance-00000015 #启动暂停的虚拟机
virsh shutdown instance-00000015 #正常关闭虚拟机
virsh destroy instance-00000015 #强制关闭虚拟机
virsh dominfo instance-00000015 #显示虚拟机的基本信息
virsh domname 2 #显示 id 号为 2 的虚拟机名
virsh domid instance-00000015 #显示虚拟机 id 号
virsh domuuid instance-00000015 #显示虚拟机的 uuid
virsh domstate instance-00000015 #显示虚拟机的当前状态
virsh dumpxml instance-00000015 #显示虚拟机的当前配置文件（可能和定义虚拟机时的配置不同，因为当虚拟机启动时，需要给虚拟机分配 id 号、uuid、vnc 端口号等等）
virsh setmem instance-00000015 512000 #给不活动虚拟机设置内存大小
virsh setvcpus instance-00000015 4 # 给不活动虚拟机设置 cpu 个数
virsh save instance-00000015 a #将该 instance-00000015 虚拟机的运行状态存储到文件 a 中
virsh restore a #恢复被存储状态的虚拟机的状态，即便虚拟机被删除也可以恢复（如果虚拟机已经被 undefine 移除，那么恢复的虚拟机也只是一个临时的状态，关闭后自动消失）
virsh undefine instance-00000015 #移除虚拟机，虚拟机处于关闭状态后还可以启动，但是被该指令删除后不能启动。在虚拟机处于 Running 状态时，调用该指令，该指令暂时不生效，但是当虚拟机被关闭后，该指令生效移除该虚拟机，也可以在该指令生效之前调用 define+TestKVM.xml 取消该指令

注意：

virsh destroy instance-00000015 这条命令并不是真正的删除这个虚拟机，只是将这个虚拟机强制关闭了。可以通过该虚拟机的 xml 文件恢复。如下：

```
[root@kvm-server ~]# virsh list
 Id Name State
----------------------------------------------------
 1 dev-new-test2 running
 2 beta-new2 running
 5 test-server running
 8 ubuntu-test03 running
 9 elk-node1 running
 10 elk-node2 running
 11 ubuntu-test01 running
 12 ubuntu-test02 running
```

强制关闭虚拟机

```
[root@kvm-server ~]# virsh destroy ubuntu-test02
Domain ubuntu-test02 destroyed
```

发现 ubuntu-test02 虚拟机已经关闭了

```
[root@kvm-server ~]# virsh list
Id Name State
----------------------------------------------------
1 dev-new-test2 running
2 beta-new2 running
5 test-server running
8 ubuntu-test03 running
9 elk-node1 running
10 elk-node2 running
11 ubuntu-test01 running
```

但是该虚拟机的 xml 文件还在，可以通过这个文件恢复

```
[root@kvm-server ~]# ll /etc/libvirt/qemu/ubuntu-test02.xml
-rw------- 1 root root 2600 Dec 26 13:55 /etc/libvirt/qemu/ubuntu-test02.xml
[root@kvm-server ~]# virsh define /etc/libvirt/qemu/ubuntu-test02.xml #这只是重新添加了这个虚拟机，目前还不是活动的虚拟机，需要启动下
[root@kvm-server ~]# virsh start ubuntu-test02
Domain ubuntu-test02 started
[root@kvm-server ~]# virsh list
Id Name State
----------------------------------------------------
1 dev-new-test2 running
2 beta-new2 running
5 test-server running
8 ubuntu-test03 running
9 elk-node1 running
10 elk-node2 running
11 ubuntu-test01 running
12 ubuntu-test02 running
```

欢迎点击这里的链接进入精彩的Linux公社 网站

Linux公社（www.Linuxidc.com）于2006年9月25日注册并开通网站，Linux现在已经成为一种广受关注和支持的一种操作系统，IDC是互联网数据中心，LinuxIDC就是关于Linux的数据中心。

Linux公社是专业的Linux系统门户网站，实时发布最新Linux资讯，包括Linux、Ubuntu、Fedora、RedHat、红旗Linux、Linux教程、Linux认证、SUSE Linux、Android、Oracle、Hadoop、CentOS、MySQL、Apache、Nginx、Tomcat、Python、Java、C语言、OpenStack、集群等技术。

Linux公社（LinuxIDC.com）设置了有一定影响力的Linux专题栏目。

Linux公社 主站网址：www.linuxidc.com 旗下网站：www.linuxidc.net

包括：**Ubuntu 专题** **Fedora 专题** **Android 专题** **Oracle 专题** **Hadoop 专题** **RedHat 专题** **SUSE 专题** **红旗 Linux 专题** **CentOS 专题**

Linux 公社微信公众号：linuxidc_com

云硬盘等后续配置

1 虚拟机相关

1.1 虚拟机位置介绍

opens tack 上创建的虚拟机实例存放位置是/var/lib/nova/instances
如下，可以查看到虚拟机的 ID

```
[root@linux-node2 ~]# nova list
+--------------------------------------+-------------+--------+------------+-------------+-----------------
--+
| ID | Name | Status | Task State | Power State | Networks |
+--------------------------------------+-------------+--------+------------+-------------+-----------------
--+
| 980fd600-a4e3-43c6-93a6-0f9dec3cc020 | kvm-server001 | ACTIVE | - | Running | flat=192.168.1.1
10 |
| e7e05369-910a-4dcf-8958-ee2b49d06135 | kvm-server002 | ACTIVE | - | Running | flat=192.168.1.
111 |
| 3640ca6f-67d7-47ac-86e2-11f4a45cb705 | kvm-server003 | ACTIVE | - | Running | flat=192.168.1.1
12 |
| 8591baa5-88d4-401f-a982-d59dc2d14f8c | kvm-server004 | ACTIVE | - | Running | flat=192.168.1.
113 |
+--------------------------------------+-------------+--------+------------+-------------+-----------------
--+
[root@linux-node2 ~]# cd /var/lib/nova/instances/
[root@linux-node2 instances]# ll
total 8
drwxr-xr-x. 2 nova nova 85 Aug 30 17:16 3640ca6f-67d7-47ac-86e2-11f4a45cb705    #虚拟机的 ID
drwxr-xr-x. 2 nova nova 85 Aug 30 17:17 8591baa5-88d4-401f-a982-d59dc2d14f8c
drwxr-xr-x. 2 nova nova 85 Aug 30 17:15 980fd600-a4e3-43c6-93a6-0f9dec3cc020
drwxr-xr-x. 2 nova nova 69 Aug 30 17:15 _base
-rw-r--r--. 1 nova nova 39 Aug 30 17:17 compute_nodes      #计算节点信息
drwxr-xr-x. 2 nova nova 85 Aug 30 17:15 e7e05369-910a-4dcf-8958-ee2b49d06135
drwxr-xr-x. 2 nova nova 4096 Aug 30 17:15 locks #锁

[root@linux-node2 instances]# cd 3640ca6f-67d7-47ac-86e2-11f4a45cb705/
[root@linux-node2 3640ca6f-67d7-47ac-86e2-11f4a45cb705]# ll
total 6380
-rw-rw----. 1 qemu qemu 20856 Aug 30 17:17 console.log                #vnc 的终端输出
-rw-r--r--. 1 qemu qemu 6356992 Aug 30 17:43 disk                     #虚拟磁盘（不是全部，有后端文
件）
-rw-r--r--. 1 nova nova 162 Aug 30 17:16 disk.info                    #disk 详情
-rw-r--r--. 1 qemu qemu 197120 Aug 30 17:16 disk.swap
-rw-r--r--. 1 nova nova 2910 Aug 30 17:16 libvirt.xml                 #xml 配置，此文件在虚拟机启动
时动态生成的，改了也没卵用。

[root@linux-node2 3640ca6f-67d7-47ac-86e2-11f4a45cb705]# file disk
disk: QEMU QCOW Image (v3), has backing file (path /var/lib/nova/instances/_base/378396c387dd43
7ec61d59627fb3fa9a6), 10737418240 bytes      #disk 后端文件
[root@openstack-server 3640ca6f-67d7-47ac-86e2-11f4a45cb705]# qemu-img info disk
image: disk
file format: qcow2
virtual size: 10G (10737418240 bytes)
disk size: 6.1M
cluster_size: 65536
backing file: /var/lib/nova/instances/_base/378396c387dd437ec61d59627fb3fa9a67f857de
Format specific information:
compat: 1.1
lazy refcounts: false
```

disk 是写时复制的方式，后端文件不变，变动的文件放在 2.2M 的 disk 文件中，不变的在后端文件放置。 占用更小的空间。

2 安装配置 Horizon-dashboard（web 界面）

这个在前面这篇中已经配置过了，这里再赘述一下吧：
dashboard 通过 api 来通信的

2.1 安装配置 dashboard

1、安装

```
[root@linux-node1 ~]# yum install -y openstack-dashboard
```

2、 修改配置文件

```
[root@linux-node1 ~]# vim /etc/openstack-dashboard/local_settings
OPENSTACK_HOST = "192.168.1.17"                    #更改为 keystone 机器地址
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user" #默认的角色
ALLOWED_HOSTS = ['*']                              #允许所有主机访问
CACHES = {
'default': {
'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
'LOCATION': '192.168.1.17:11211',                  #连接 memcached
}
}
#CACHES = {
# 'default': {
# 'BACKEND': 'django.core.cache.backends.locmem.LocMemCache',
# }
#}
TIME_ZONE = "Asia/Shanghai"                        #设置时区
```

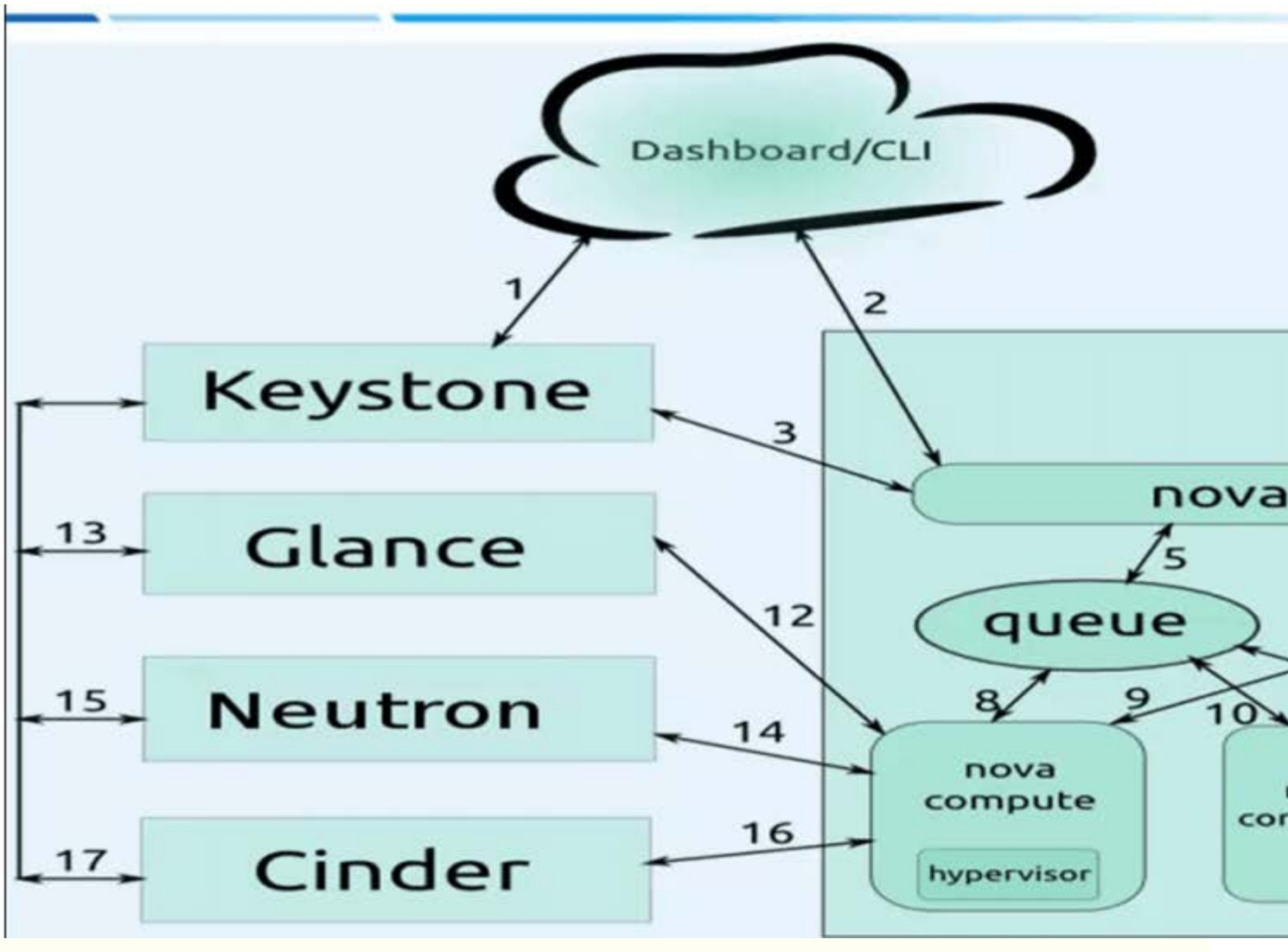
重启 httpd 服务

```
[root@linux-node1 ~]# systemctl restart httpd
```

web 界面登录访问 dashboard

http://58.68.250.17/dashboard/
用户密码 demo 或者 admin（管理员）

3 虚拟机创建流程（非常重要）



第一阶段：
1、用户通过 Dashboard 或者命令行，发送用户名密码给 Keystone 进行验证，验证成功后，返回 OS_TOKEN（令牌）
2、 Dashboard 或者命令行访问 nova-api，我要创建虚拟机
3、 nova-api 去找 keystone 验证确认。

第二阶段：nova 之间的组件交互
4、 nova-api 和 nova 数据库进行交互，记录
5-6、 nova-api 通过消息队列讲信息发送给 nova-scheduler
7、 nova-scheduler 收到消息后，和数据库进行交互，自己进行调度
8、 nova-scheduler 通过消息队列将信息发送给 nova-compute
9-11、 nova-compute 通过消息队列和 nova-conductor 通信，通过 nova-conductor 和数据库进行交互，获取相关信息。（图上有点问题）， nova-conductor 就是专门和数据库进行通信的。

第三阶段：
12、 nova-compute 发起 api 调用 Glance 获取镜像。
13、 Glance 去找 keystone 认证，认证成功后将镜像给 nova-compute
14、 nova-compute 找 Neutron 获取网络
15、 Neutron 去找 keystone 认证，认证后为 nova-compute 提供网络
16-17 同理

第四阶段：
nova-compute 通过 libvirt 调用 kvm 生成虚拟机
18.、 nova-compute 和底层的 hypervisor 进行交互，如果是使用的 kvm，则通过 libvirt 调用 kvm 去创建虚拟机，创建过程中 nova-api 会一直去数据库轮询查看虚拟机创建状态。

****
****

细节：

新的计算节点第一次创建虚拟机会慢

因为 glance 需要把镜像上传到计算节点上，即_bash 目录下，之后才会创建虚拟机

```
[root@linux-node2 _base]# pwd
/var/lib/nova/instances/_base
[root@openstack-server _base]# ll
total 10485764
-rw-r--r--. 1 nova qemu 10737418240 Aug 30 17:57 378396c387dd437ec61d59627fb3fa9a67f857de
-rw-r--r--. 1 nova qemu 1048576000 Aug 30 17:57 swap_1000
```

第一个虚拟机创建后，后续在创建其他的虚拟机时就快很多了。

创建虚拟机操作，具体见前面第一步。

****

4 cinder 云存储服务

4.1 存储的分类

1、块存储

磁盘

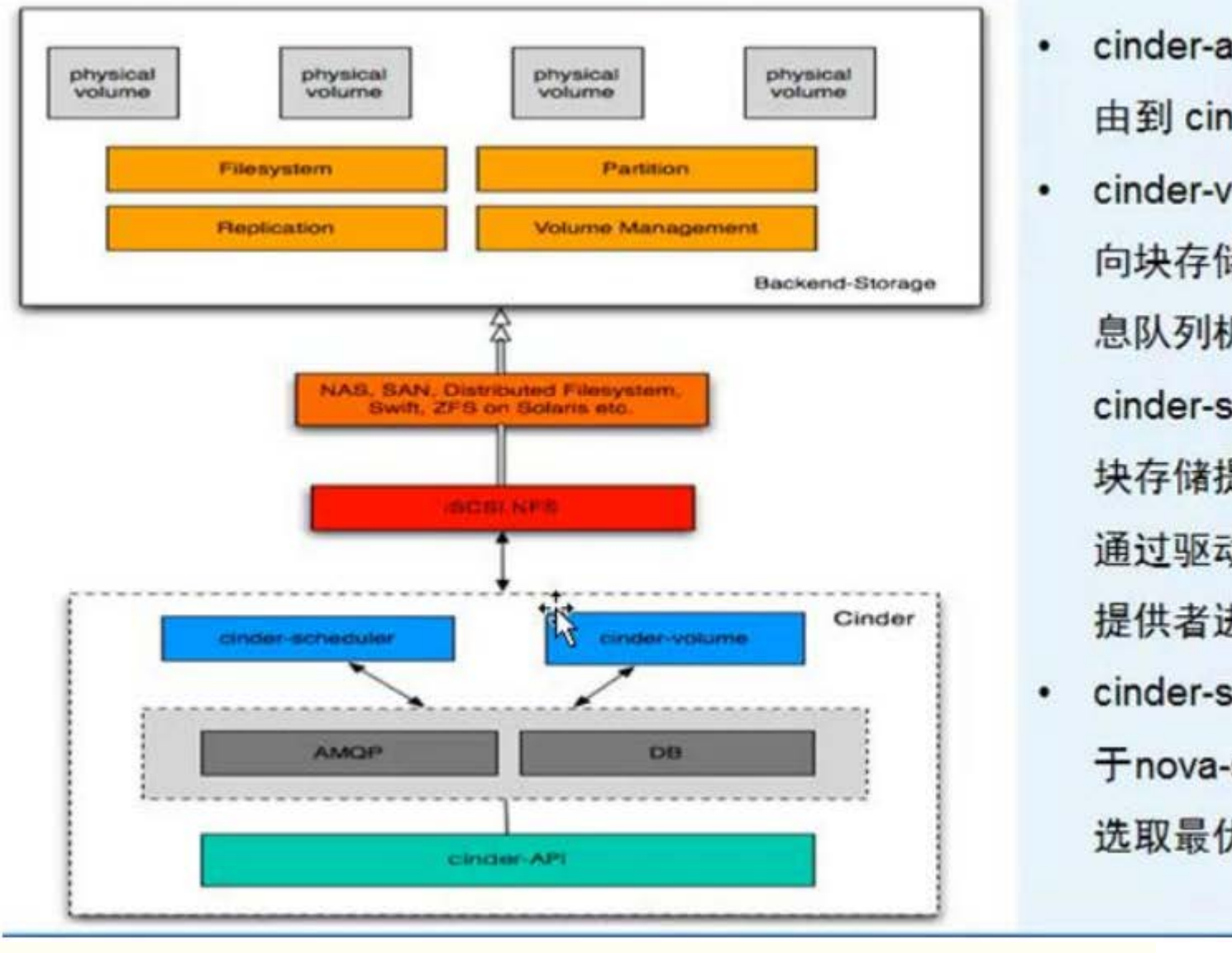
2、文件存储

nfs

3、对象存储

4.2 cinder 介绍

云硬盘



一般 cinder-api 和 cinder-scheduler 安装在控制节点上， cinder-volume 安装在存储节点上。

4.3 cinder 控制节点配置

1、安装软件包

控制节点

```
[root@linux-node1 ~]#yum install -y openstack-cinder python-cinderclient
```

计算节点

```
[root@linux-node2 ~]#yum install -y openstack-cinder python-cinderclient
```

2、 创建 cinder 的数据库

之前的一篇中已经创建了。

3、修改配置文件

```
[root@linux-node1 ~]# cat /etc/cinder/cinder.conf|grep -v "^#"|grep -v "^$"
[DEFAULT]
glance_host = 192.168.1.17
auth_strategy = keystone
rpc_backend = rabbit
[BRCD_FABRIC_EXAMPLE]
[CISCO_FABRIC_EXAMPLE]
[cors]
[cors.subdomain]
[database]
connection = mysql://cinder:cinder@192.168.1.17/cinder
[fc-zone-manager]
[keymgr]
[keystone_authtoken]
auth_uri = http://192.168.1.17:5000
auth_url = http://192.168.1.17:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = cinder
password = cinder
[matchmaker_redis]
[matchmaker_ring]
[oslo_concurrency]
lock_path = /var/lib/cinder/tmp
[oslo_messaging_amqp]
[oslo_messaging_qpid]
[oslo_messaging_rabbit]
rabbit_host = 192.168.1.17
rabbit_port = 5672
rabbit_userid = openstack
rabbit_password = openstack
[oslo_middleware]
[oslo_policy]
[oslo_reports]
[profiler]
```

在 nova 配置文件中添加

```
[root@linux-node1 ~]# vim /etc/nova/nova.conf
os_region_name=RegionOne                    #在[cinder]区域里添加
```

4、同步数据库

```
[root@linux-node1 ~]# su -s /bin/sh -c "cinder-manage db sync" cinder
..................
2016-08-30 18:27:20.204 67111 INFO migrate.versioning.api [-] done
2016-08-30 18:27:20.204 67111 INFO migrate.versioning.api [-] 59 -> 60...
2016-08-30 18:27:20.208 67111 INFO migrate.versioning.api [-] done
```

5、 创建 keystone 用户

```
[root@linux-node1 ~]# cd /usr/local/src/
[root@linux-node1 src]# source admin-openrc.sh
[root@linux-node1 src]# openstack user create --domain default --password-prompt cinder
User Password: #这里我设置的是 cinder
```

```
Repeat User Password:
+-----------+----------------------------------+
| Field | Value |
+-----------+----------------------------------+
| domain_id | default |
| enabled | True |
| id | 955a2e684bed4617880942acd69e1073 |
| name | cinder |
+-----------+----------------------------------+
[root@openstack-server src]# openstack role add --project service --user cinder admin
```

6、启动服务

```
[root@linux-node1 ~]# systemctl restart openstack-nova-api.service
[root@linux-node1 ~]# systemctl enable openstack-cinder-api.service openstack-cinder-scheduler.serv
ice
[root@linux-node1 ~]# systemctl start openstack-cinder-api.service openstack-cinder-scheduler.servic
e
```

7、在 keystone 上创建服务并注册

v1 和 v2 都要注册

```
[root@linux-node1 src]# source admin-openrc.sh
[root@linux-node1 src]# openstack service create --name cinder --description "OpenStack Block Stora
ge" volume
+-------------+----------------------------------+
| Field | Value |
+-------------+----------------------------------+
| description | OpenStack Block Storage |
| enabled | True |
| id | 7626bd9be54a444589ae9f8f8d29dc7b |
| name | cinder |
| type | volume |
+-------------+----------------------------------+
[root@linux-node1 src]# openstack service create --name cinderv2 --description "OpenStack Block Sto
rage" volumev2
+-------------+----------------------------------+
| Field | Value |
+-------------+----------------------------------+
| description | OpenStack Block Storage |
| enabled | True |
| id | 5680a0ce912b484db88378027b1f6863 |
| name | cinderv2 |
| type | volumev2 |
+-------------+----------------------------------+
[root@linux-node1 src]# openstack endpoint create --region RegionOne volume public http://192.168.
1.17:8776/v1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field | Value |
+--------------+-----------------------------------------+
| enabled | True |
| id | 10de5ed237d54452817e19fd65233ae6 |
| interface | public |
| region | RegionOne |
| region_id | RegionOne |
| service_id | 7626bd9be54a444589ae9f8f8d29dc7b |
| service_name | cinder |
| service_type | volume |
| url | http://192.168.1.17:8776/v1/%(tenant_id)s |
+--------------+-----------------------------------------+
[root@linux-node1 src]# openstack endpoint create --region RegionOne volume internal http://192.16
8.1.17:8776/v1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field | Value |
+--------------+-----------------------------------------+
| enabled | True |
```

```
| id | f706552cfb40471abf5d16667fc5d629 |
| interface | internal |
| region | RegionOne |
| region_id | RegionOne |
| service_id | 7626bd9be54a444589ae9f8f8d29dc7b |
| service_name | cinder |
| service_type | volume |
| url | http://192.168.1.17:8776/v1/%(tenant_id)s |
+-------------+-----------------------------------------+
[root@linux-node1 src]# openstack endpoint create --region RegionOne volume admin http://192.168.
1.17:8776/v1/%\(tenant_id\)s
+-------------+-----------------------------------------+
| Field | Value |
+-------------+-----------------------------------------+
| enabled | True |
| id | c9dfa19aca3c43b5b0cf2fe7d393efce |
| interface | admin |
| region | RegionOne |
| region_id | RegionOne |
| service_id | 7626bd9be54a444589ae9f8f8d29dc7b |
| service_name | cinder |
| service_type | volume |
| url | http://192.168.1.17:8776/v1/%(tenant_id)s |
+-------------+-----------------------------------------+
[root@linux-node1 src]# openstack endpoint create --region RegionOne volumev2 public http://192.16
8.1.17:8776/v2/%\(tenant_id\)s
+-------------+-----------------------------------------+
| Field | Value |
+-------------+-----------------------------------------+
| enabled | True |
| id | 9ac83d0fab134f889e972e4e7680b0e6 |
| interface | public |
| region | RegionOne |
| region_id | RegionOne |
| service_id | 5680a0ce912b484db88378027b1f6863 |
| service_name | cinderv2 |
| service_type | volumev2 |
| url | http://192.168.1.17:8776/v2/%(tenant_id)s |
+-------------+-----------------------------------------+
[root@linux-node1 src]# openstack endpoint create --region RegionOne volumev2 internal http://192.
168.1.17:8776/v2/%\(tenant_id\)s
+-------------+-----------------------------------------+
| Field | Value |
+-------------+-----------------------------------------+
| enabled | True |
| id | 9d18eac0868b4c49ae8f6198a029d7e0 |
| interface | internal |
| region | RegionOne |
| region_id | RegionOne |
| service_id | 5680a0ce912b484db88378027b1f6863 |
| service_name | cinderv2 |
| service_type | volumev2 |
| url | http://192.168.1.17:8776/v2/%(tenant_id)s |
+-------------+-----------------------------------------+
[root@linux-node1 src]# openstack endpoint create --region RegionOne volumev2 admin http://192.1
68.1.17:8776/v2/%\(tenant_id\)s
+-------------+-----------------------------------------+
| Field | Value |
+-------------+-----------------------------------------+
| enabled | True |
| id | 68c93bd6cd0f4f5ca6d5a048acbddc91 |
| interface | admin |
| region | RegionOne |
| region_id | RegionOne |
```

```
| service_id | 5680a0ce912b484db88378027b1f6863 |
| service_name | cinderv2 |
| service_type | volumev2 |
| url | http://192.168.1.17:8776/v2/%(tenant_id)s |
+-------------+-----------------------------------------+
```

查看注册信息：

```
[root@linux-node1 src]# openstack endpoint list
+----------------------------------+----------+-------------+-------------+---------+----------+-----------------------------------------+
| ID | Region | Service Name | Service Type | Enabled | Interface | URL |
+----------------------------------+----------+-------------+-------------+---------+----------+-----------------------------------------+
| 02fed35802734518922d0ca2d672f469 | RegionOne | keystone | identity | True | internal | http://192.168.1.17:5000/v2.0 |
| 10de5ed237d54452817e19fd65233ae6 | RegionOne | cinder | volume | True | public | http://192.168.1.17:8776/v1/%(tenant_id)s |
| 1a3115941ff54b7499a800c7c43ee92a | RegionOne | nova | compute | True | internal | http://192.168.1.17:8774/v2/%(tenant_id)s |
| 31fbf72537a14ba7927fe9c7b7d06a65 | RegionOne | glance | image | True | admin | http://192.168.1.17:9292 |
| 5278f33a42754c9a8d90937932b8c0b3 | RegionOne | nova | compute | True | admin | http://192.168.1.17:8774/v2/%(tenant_id)s |
| 52b0a1a700f04773a220ff0e365dea45 | RegionOne | keystone | identity | True | public | http://192.168.1.17:5000/v2.0 |
| 68c93bd6cd0f4f5ca6d5a048acbddc91 | RegionOne | cinderv2 | volumev2 | True | admin | http://192.168.1.17:8776/v2/%(tenant_id)s |
| 88df7df6427d45619df192979219e65c | RegionOne | keystone | identity | True | admin | http://192.168.1.17:35357/v2.0 |
| 8c4fa7b9a24949c5882949d13d161d36 | RegionOne | nova | compute | True | public | http://192.168.1.17:8774/v2/%(tenant_id)s |
| 9ac83d0fab134f889e972e4e7680b0e6 | RegionOne | cinderv2 | volumev2 | True | public | http://192.168.1.17:8776/v2/%(tenant_id)s |
| 9d18eac0868b4c49ae8f6198a029d7e0 | RegionOne | cinderv2 | volumev2 | True | internal | http://192.168.1.17:8776/v2/%(tenant_id)s |
| be788b4aa2ce4251b424a3182d0eea11 | RegionOne | glance | image | True | public | http://192.168.1.17:9292 |
| c059a07fa3e141a0a0b7fc2f46ca922c | RegionOne | neutron | network | True | public | http://192.168.1.17:9696 |
| c9dfa19aca3c43b5b0cf2fe7d393efce | RegionOne | cinder | volume | True | admin | http://192.168.1.17:8776/v1/%(tenant_id)s |
| d0052712051a4f04bb59c06e2d5b2a0b | RegionOne | glance | image | True | internal | http://192.168.1.17:9292 |
| ea325a8a2e6e4165997b2e24a8948469 | RegionOne | neutron | network | True | internal | http://192.168.1.17:9696 |
| f706552cfb40471abf5d16667fc5d629 | RegionOne | cinder | volume | True | internal | http://192.168.1.17:8776/v1/%(tenant_id)s |
| ffdec11ccf024240931e8ca548876ef0 | RegionOne | neutron | network | True | admin | http://192.168.1.17:9696 |
+----------------------------------+----------+-------------+-------------+---------+----------+-----------------------------------------+
```

4.4 cinder 存储节点配置

1、 使用 ISCSI 方式创建云硬盘

计算节点添加硬盘并创建 VG

```
[root@linux-node2 ~]# df -h
Filesystem Size Used Avail Use% Mounted on
/dev/sda2 100G 44G 57G 44% /
devtmpfs 10G 0 10G 0% /dev
tmpfs 10G 0 10G 0% /dev/shm
tmpfs 10G 90M 10G 1% /run
tmpfs 10G 0 10G 0% /sys/fs/cgroup
/dev/sda1 197M 127M 71M 65% /boot
```

```
tmpfs 6.3G 0 6.3G 0% /run/user/0
/dev/sda5 811G 33M 811G 1% /home
```

由于这里我的计算节点上没有多余的硬盘和空间了
所以考虑将上面的 home 分区卸载，拿来做云硬盘

卸载 home 分区前，将 home 分区下的数据备份。
等到 home 卸载后，再创建/home 目录，将备份数据拷贝到/home 下

```
[root@linux-node2 ~]# umount /home
[root@linux-node2 ~]# df -h
Filesystem Size Used Avail Use% Mounted on
/dev/sda2 100G 44G 57G 44% /
devtmpfs 10G 0 10G 0% /dev
tmpfs 10G 0 10G 0% /dev/shm
tmpfs 10G 90M 10G 1% /run
tmpfs 10G 0 10G 0% /sys/fs/cgroup
/dev/sda1 197M 127M 71M 65% /boot
tmpfs 6.3G 0 6.3G 0% /run/user/0
```

```
[root@linux-node2 ~]# fdisk -l

Disk /dev/sda: 999.7 GB, 999653638144 bytes, 1952448512 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x000b2db8

Device Boot Start End Blocks Id System
/dev/sda1 * 2048 411647 204800 83 Linux
/dev/sda2 411648 210126847 104857600 83 Linux
/dev/sda3 210126848 252069887 20971520 82 Linux swap / Solaris
/dev/sda4 252069888 1952448511 850189312 5 Extended
/dev/sda5 252071936 1952448511 850188288 83 Linux
```

这样，home 分区卸载的/dev/sda5 可以拿来做 lvm

```
[root@linux-node2 ~]# vim /etc/lvm/lvm.conf
filter = [ "a/sda5/", "r/.*/"]
```

其中：a 表示同意， r 是不同意

上面的 home 分区没有做 lvm，设备名是/dev/sda5，则/etc/lvm/lvm.conf 可以如上设置。

如果 home 分区做了 lvm，“df -h”命令查看 home 分区的设备名比如是/dev/mapper/centos-home
那么/etc/lvm/lvm.conf 这里就要这样配置了:
filter = ["a|^/dev/mapper/centos-home$|", "r|.*/|"]

```
[root@linux-node2 ~]# pvcreate /dev/sda5
WARNING: xfs signature detected on /dev/sda5 at offset 0. Wipe it? [y/n]: y
Wiping xfs signature on /dev/sda5.
Physical volume "/dev/sda5" successfully created
[root@linux-node2 ~]# vgcreate cinder-volumes /dev/sda5
Volume group "cinder-volumes" successfully created
```

2、修改配置文件

```
[root@linux-node1 ~]# scp /etc/cinder/cinder.conf 192.168.1.8:/etc/cinder/cinder.conf
```

需要更改

```
[root@linux-node2 ~]# vim /etc/cinder/cinder.conf
enabled_backends = lvm              #在[DEFAULT]区域添加

[lvm]                    #文件底部添加 lvm 区域设置
volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
volume_group = cinder-volumes
```

```
iscsi_protocol = iscsi
iscsi_helper = lioadm
```

3、启动服务

```
[root@linux-node2 ~]#systemctl enable openstack-cinder-volume.service target.service
[root@linux-node2 ~]#systemctl start openstack-cinder-volume.service target.service
```

4.5 创建云硬盘

1、在控制节点上检查

时间不同步可能会出现 down 的状态，

```
[root@linux-node1 ~]# systemctl restart chronyd
[root@linux-node1 ~]# source admin-openrc.sh
[root@openstack-server ~]# cinder service-list
+------------------+--------------------+------+---------+-------+----------------------------+-----------------+
| Binary | Host | Zone | Status | State | Updated_at | Disabled Reason |
+------------------+--------------------+------+---------+-------+----------------------------+-----------------+
| cinder-scheduler | openstack-server | nova | enabled | up | 2016-08-31T07:50:06.000000 | - |
| cinder-volume    | openstack-server@lvm | nova | enabled | up | 2016-08-31T07:50:08.000000 | - |
+------------------+--------------------+------+---------+-------+----------------------------+-----------------+
```

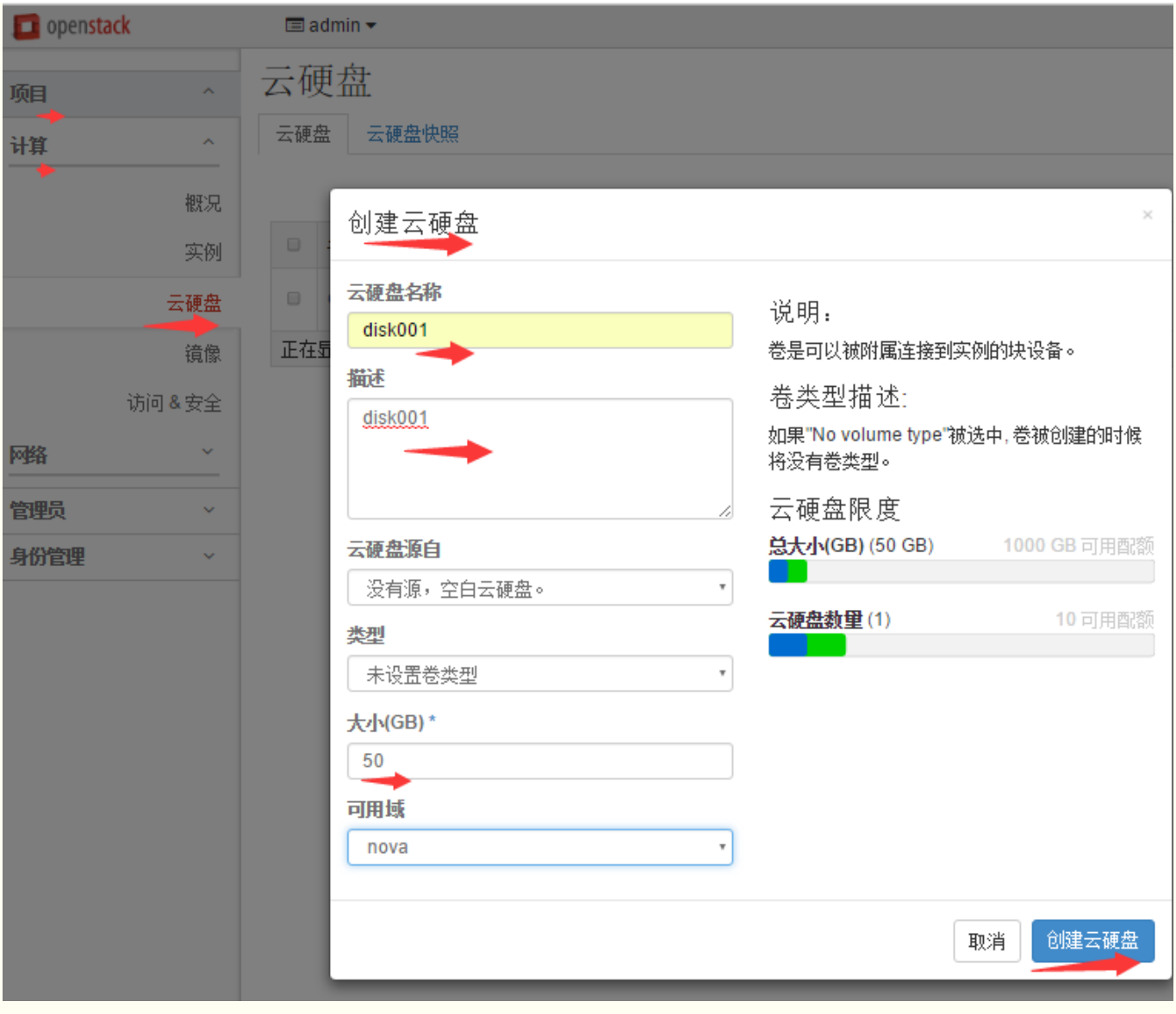
这个时候，退出 openstack 的 dashboard，再次登录！

就可以在左侧栏的“计算”里看见“云硬盘”了

2、使用 dashboard 创建云硬盘

（注意：可以利用已有的虚拟机做快照**(快照做好后，这台做快照的虚拟机就会关机，需要之后再手动启动)**，然后就能利用快照进行创建/启动虚拟机）

(注意：通过快照创建的虚拟机，默认是没有 ip 的，需要做下修改。修改参考 webvirtmgr 中克隆虚拟机后的修改方法)

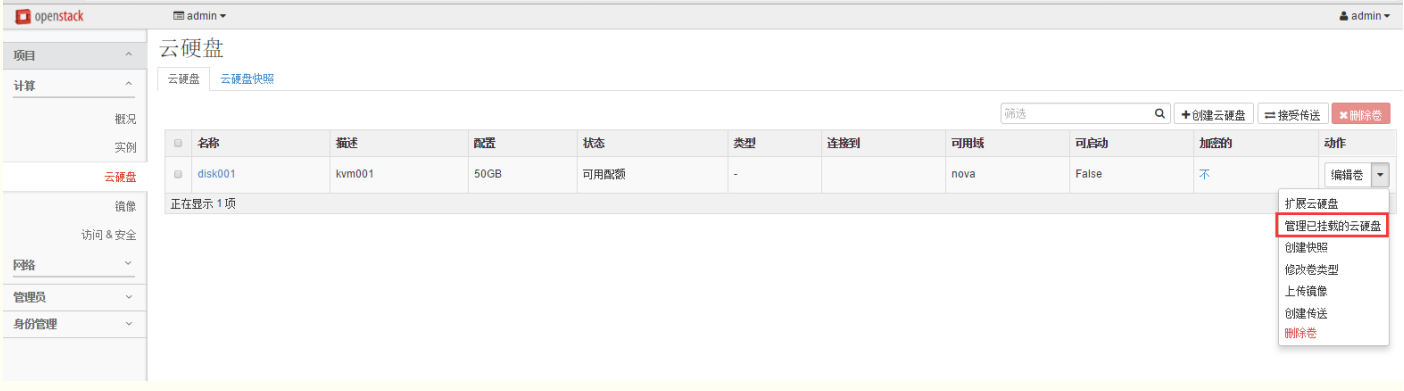


此时可以在计算节点上查看到：

```
[root@linux-node2 ~]# lvdisplay
--- Logical volume ---
LV Path /dev/cinder-volumes/volume-efb1d119-e006-41a8-b695-0af9f8d35063
LV Name volume-efb1d119-e006-41a8-b695-0af9f8d35063
VG Name cinder-volumes
LV UUID aYztLC-jljz-esGh-UTco-KxtG-ipce-Oinx9j
LV Write Access read/write
LV Creation host, time openstack-server, 2016-08-31 15:55:05 +0800
LV Status available
# open 0
LV Size 50.00 GiB
Current LE 12800
Segments 1
Allocation inherit
Read ahead sectors auto
```

- currently set to 256
Block device 253:0

下面可将创建的云硬盘挂载到相应的虚拟机上了！



登陆虚拟机 kvm-server001 查看，就能发现挂载的云硬盘了。挂载就能直接用了。

```
[root@kvm-server001 ~]# fdisk -l

Disk /dev/vda: 10.7 GB, 10737418240 bytes
16 heads, 63 sectors/track, 20805 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00046e27

.............

Disk /dev/vdc: 53.7 GB, 53687091200 bytes
16 heads, 63 sectors/track, 104025 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
```

格式化连接过来的云硬盘

```
[root@kvm-server001 ~]# mkfs.ext4 /dev/vdc
mke2fs 1.41.12 (17-May-2010)
.............
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

创建挂载目录/data

```
[root@kvm-server001 ~]# mkdir /data
```

然后挂载

```
[root@kvm-server001 ~]# mount /dev/vdc /data
[root@kvm-server001 ~]# df -h
Filesystem Size Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol00 8.2G 737M 7.1G 10% /
tmpfs 2.9G 0 2.9G 0% /dev/shm
/dev/vda1 194M 28M 156M 16% /boot
/dev/vdc 50G 180M 47G 1% /data
```

----------特别说明下----------

由于制作的虚拟机的根分区很小，可以把挂载的云硬盘制作成 lvm，扩容到根分区上（根分区也是 lvm）

操作记录如下：

```
[root@localhost ~]# fdisk -l
.............
.............
Disk /dev/vdc: 161.1 GB, 161061273600 bytes #这是挂载的云硬盘
16 heads, 63 sectors/track, 312076 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

[root@localhost ~]# df -h
Filesystem Size Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
8.1G 664M 7.0G 9% /                    #vm 的根分区，可以进行手动 lvm 扩容
tmpfs 2.9G 0 2.9G 0% /dev/shm
/dev/vda1 190M 37M 143M 21% /boot
```

首先将挂载下来的云硬盘制作新分区

```
[root@localhost ~]# fdisk /dev/vdc
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0x3256d3cb.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.
```

```
Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
switch off the mode (command 'c') and change display units to
sectors (command 'u').

Command (m for help): p

Disk /dev/vdc: 161.1 GB, 161061273600 bytes
16 heads, 63 sectors/track, 312076 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x3256d3cb

Device Boot Start End Blocks Id System

Command (m for help): n
Command action
e extended
p primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-312076, default 1):
Using default value 1
Last cylinder, +cylinders or +size{K,M,G} (1-312076, default 312076):
Using default value 312076

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
[root@localhost ~]# fdisk /dev/vdc

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
switch off the mode (command 'c') and change display units to
sectors (command 'u').

Command (m for help): p

Disk /dev/vdc: 161.1 GB, 161061273600 bytes
16 heads, 63 sectors/track, 312076 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x3256d3cb

Device Boot Start End Blocks Id System
/dev/vdc1 1 312076 157286272+ 83 Linux
```

开始进行根分区的 **lvm** 扩容：

```
[root@localhost ~]# pvcreate /dev/vdc1
Physical volume "/dev/vdc1" successfully created

[root@localhost ~]# lvdisplay
--- Logical volume ---
LV Path /dev/VolGroup00/LogVol01
LV Name LogVol01
VG Name VolGroup00
LV UUID xtykaQ-3ulO-XtF0-BUqB-Pure-LH1n-O2zF1Z
LV Write Access read/write
LV Creation host, time localhost.localdomain, 2016-09-05 22:21:00 -0400
LV Status available
# open 1
LV Size 1.50 GiB
Current LE 48
Segments 1
```

```
Allocation inherit
Read ahead sectors auto
- currently set to 256
Block device 253:0

--- Logical volume ---
LV Path /dev/VolGroup00/LogVol00    #这是虚拟机的根分区的 lvm 逻辑卷，就是给这个扩容
LV Name LogVol00
VG Name VolGroup00
LV UUID 7BW8Wm-4VSt-5GzO-sIew-D1OI-pqLP-eXgM80
LV Write Access read/write
LV Creation host, time localhost.localdomain, 2016-09-05 22:21:00 -0400
LV Status available
# open 1
LV Size 8.28 GiB
Current LE 265
Segments 1
Allocation inherit
Read ahead sectors auto
- currently set to 256
Block device 253:1

[root@localhost ~]# vgdisplay
--- Volume group ---
VG Name VolGroup00
System ID
Format lvm2
Metadata Areas 1
Metadata Sequence No 5
VG Access read/write
VG Status resizable
MAX LV 0
Cur LV 2
Open LV 2
Max PV 0
Cur PV 1
Act PV 1
VG Size 9.78 GiB
PE Size 32.00 MiB
Total PE 313
Alloc PE / Size 313 / 9.78 GiB
Free PE / Size 0 / 0                                    #VolGroup00 这个卷组没有剩余空间了，需
要 vg 进行自身扩容
VG UUID tEEreQ-O2HZ-rm9d-vS8Y-VemY-D7uY-qAYdWU

[root@localhost ~]# vgextend VolGroup00 /dev/vdc1          #vg 扩容
Volume group "VolGroup00" successfully extended

[root@localhost ~]# vgdisplay #vg 扩容后再次查看
--- Volume group ---
VG Name VolGroup00
System ID
Format lvm2
Metadata Areas 2
Metadata Sequence No 6
VG Access read/write
VG Status resizable
MAX LV 0
Cur LV 2
Open LV 2
Max PV 0
Cur PV 2
Act PV 2
VG Size 159.75 GiB
```

PE Size 32.00 MiB
Total PE 5112
Alloc PE / Size 313 / 9.78 GiB
Free PE / Size 4799 / 149.97 GiB #发现剩余空间有了 149.97G
VG UUID tEEreQ-O2HZ-rm9d-vS8Y-VemY-D7uY-qAYdWU

在上面查询可知的 vg 所有的剩余空间全部增加给逻辑卷/dev/VolGroup00/LogVol00
[root@localhost ~]# lvextend -l +4799 /dev/VolGroup00/LogVol00
Size of logical volume VolGroup00/LogVol00 changed from 8.28 GiB (265 extents) to 158.25 GiB (5064 extents).
Logical volume LogVol00 successfully resized.

修改逻辑卷大小后，通过 resize2fs 来修改文件系统的大小
[root@localhost ~]# resize2fs /dev/VolGroup00/LogVol00
resize2fs 1.41.12 (17-May-2010)
Filesystem at /dev/VolGroup00/LogVol00 is mounted on /; on-line resizing required
old desc_blocks = 1, new_desc_blocks = 10
Performing an on-line resize of /dev/VolGroup00/LogVol00 to 41484288 (4k) blocks.
The filesystem on /dev/VolGroup00/LogVol00 is now 41484288 blocks long.

再次查看，根分区已经扩容了！！
[root@localhost ~]# df -h
Filesystem Size Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
156G 676M 148G 1% /
tmpfs 2.9G 0 2.9G 0% /dev/shm
/dev/vda1 190M 37M 143M 21% /boot

****

云硬盘添加是热添加

注意：
虚拟机上发现的云硬盘格式化并挂载到如 /data 目录下

删除云硬盘需要先卸载【不仅在虚拟机上卸载，在 dashboard 界面里也要卸载】

可以在虚拟机上对连接的云硬盘做 lvm 逻辑卷，以便以后不够用时，可以再加硬盘做 lvm 扩容，无缝扩容！

如下，虚拟机 kvm-server001 连接了一块 100G 的云硬盘
现对这 100G 的硬盘分区，制作 lvm
[root@kvm-server001 ~]# fdisk -l

```
Disk /dev/vda: 10.7 GB, 10737418240 bytes
16 heads, 63 sectors/track, 20805 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00046e27
.....

Disk /dev/vdc: 107.4 GB, 107374182400 bytes
16 heads, 63 sectors/track, 208050 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
```

先制作分区

```
[root@kvm-server001 ~]# fdisk /dev/vdc
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0x4e0d7808.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Command (m for help): p

Disk /dev/vdc: 107.4 GB, 107374182400 bytes
16 heads, 63 sectors/track, 208050 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x4e0d7808

Device Boot Start End Blocks Id System

Command (m for help): n
Command action
e extended
p primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-208050, default 1):                    #回车
Using default value 1
Last cylinder, +cylinders or +size{K,M,G} (1-208050, default 208050):      #回车，即使用全部剩余空间
创建新分区
Using default value 208050

Command (m for help): p

Disk /dev/vdc: 107.4 GB, 107374182400 bytes
16 heads, 63 sectors/track, 208050 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x4e0d7808

Device Boot Start End Blocks Id System
/dev/vdc1 1 208050 104857168+ 83 Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.

[root@kvm-server001 ~]# pvcreate /dev/vdc1                #制作 pv
Physical volume "/dev/vdc1" successfully created
[root@kvm-server001 ~]# vgcreate vg0 /dev/vdc1            #制作 vg
Volume group "vg0" successfully created
[root@kvm-server001 ~]# vgdisplay                         #查看 vg 大小
```

```
--- Volume group ---
VG Name vg0
System ID
Format lvm2
Metadata Areas 1
Metadata Sequence No 1
VG Access read/write
VG Status resizable
MAX LV 0
Cur LV 0
Open LV 0
Max PV 0
Cur PV 1
Act PV 1
VG Size 100.00 GiB
PE Size 4.00 MiB
Total PE 25599
Alloc PE / Size 0 / 0
Free PE / Size 25599 / 100.00 GiB
VG UUID UIsTAe-oUzt-3atO-PVTw-0JUL-7Z8s-XVppIH

[root@kvm-server001 ~]# lvcreate -L +99.99G -n lv0 vg0              #lv 逻辑卷大小不能超过 vg 大小
Rounding up size to full physical extent 99.99 GiB
Logical volume "lv0" created
[root@kvm-server001 ~]# mkfs.ext4 /dev/vg0/lv0                      #格式化 lvm 逻辑卷
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
6553600 inodes, 26212352 blocks
1310617 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=4294967296
800 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
4096000, 7962624, 11239424, 20480000, 23887872

Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 20 mounts or
180 days, whichever comes first. Use tune2fs -c or -i to override.

[root@kvm-server001 ~]# mkdir /data                                 #创建挂载目录
[root@kvm-server001 ~]# mount /dev/vg0/lv0 /data                    #挂载 lvm
[root@kvm-server001 ~]# df -h
Filesystem Size Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol00 8.2G 842M 7.0G 11% /
tmpfs 2.9G 0 2.9G 0% /dev/shm
/dev/vda1 194M 28M 156M 16% /boot
/dev/mapper/vg0-lv0 99G 188M 94G 1% /data
```

****

背景：

由于计算节点内网网关不存在，所以 vm 不能通过桥接模式自行联网了。

要想使安装后的 vm 联网，还需要我们手动进行些特殊配置：

（1）计算节点部署 squid 代理环境，即 vm 对外的访问请求通过计算节点机 squid 代理出去。

（2）vm 对内的访问请求通过计算节点的 iptables 进行 nat 端口转发，web 应用请求可以利用 nginx 或 haproxy 进行代理转发。

下面说的是 http 方式的 squid 代理；
如果是 https 的 squid 代理，可以参考我的另一篇技术博客内容：
http://www.linuxidc.com/Linux/2017-02/140398.htm

（1）

1）计算节点上的操作：
yum 命令直接在线安装 squid
[root@linux-node2 ~]# yum install squid
安装完成后，修改 squid.conf 文件中的内容，修改之前可以先备份该文件
[root@linux-node2 ~]# cd /etc/squid/
[root@linux-node2 squid]# cp squid.conf squid.conf_bak
[root@linux-node2 squid]# vim squid.conf
http_access allow all
http_port 192.168.1.17:3128
cache_dir ufs /var/spool/squid 100 16 256

然后执行下面命令，进行 squid 启动前测试
[root@linux-node2 squid]# squid -k parse
2016/08/31 16:53:36| Startup: Initializing Authentication Schemes ...
.....
2016/08/31 16:53:36| Initializing https proxy context

在第一次启动之前或者修改了 cache 路径之后，需要重新初始化 cache 目录。
[root@kvm-linux-node2 squid]# squid -z
2016/08/31 16:59:21 kid1| /var/spool/squid exists
2016/08/31 16:59:21 kid1| Making directories in /var/spool/squid/00
.....

如果有下面报错：
2016/09/06 15:19:23 kid1| No cache_dir stores are configured.

解决办法：
vim squid.conf
cache_dir ufs /var/spool/squid 100 16 256 #打开这行的注释

#ll /var/spool/squid 确保这个目录存在

再次 squid -z 初始化就 ok 了

[root@kvm-linux-node2 squid]# systemctl enable squid
Created symlink from /etc/systemd/system/multi-user.target.wants/squid.service to /usr/lib/systemd/system/squid.service.
[root@kvm-server001 squid]# systemctl start squid
[root@kvm-server001 squid]# lsof -i:3128
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
squid 62262 squid 16u IPv4 4275294 0t0 TCP openstack-server:squid (LISTEN)

如果计算节点开启了 iptables 防火墙规则
这里我的 centos7.2 系统上设置了 iptables（关闭默认的 firewalle）
则还需要在/etc/sysconfig/iptables 里添加下面一行：
-A INPUT -s 192.168.1.0/24 -p tcp -m state --state NEW -m tcp --dport 3128 -j ACCEPT

我这里防火墙配置如下：
[root@linux-node2 squid]# cat /etc/sysconfig/iptables
sample configuration for iptables service
you can edit this manually or use system-config-firewall
please do not ask us to add additional ports/services to this default configuration
*filter
:INPUT ACCEPT [0:0]

```
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -s 192.168.1.0/24 -p tcp -m state --state NEW -m tcp --dport 3128 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 6080 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 10050 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

然后重启 iptables 服务

```
[root@linux-node2 ~]# systemctl restart iptables.service        #最后重启防火墙使配置生效
[root@linux-node2 ~]# systemctl enable iptables.service         #设置防火墙开机启动
```

2）下面是虚拟机上的 squid 配置：

只需要在系统环境变量配置文件/etc/profile 里添加下面一行即可（在文件底部添加）

```
[root@kvm-server001 ~]# vim /etc/profile
.....
export http_proxy=http://192.168.1.17:3128

[root@kvm-server001 ~]# source /etc/profile                    #使上面的配置生效
```

测试虚拟机是否能对外访问：

```
[root@kvm-server001 ~]# curl http://www.baidu.com                          #能正常对外访问

[root@kvm-server001 ~]# yum list                                           #yum 能正常在线使用

[root@kvm-server001 ~]# wget http://my.oschina.net/mingpeng/blog/293744 #能正常在线下载
```

这样，虚拟机的对外请求就可以通过 squid 顺利代理出去了！

这里，squid 代理的是 http 方式，如果是 https 方式的 squid 代理，可以参考我的另一篇博客：http://www.linuxidc.com/Linux/2017-02/140398.htm

****

（2）

1）下面说下虚拟机的对内请求的代理配置：

NAT 端口转发，可以参考我的另一篇博客内容： http://www.linuxidc.com/Linux/2016-10/136589p4.htm l

在计算节点（即虚拟机的宿主机）上配置 iptables 规则：

```
[root@linux-node2 ~]# cat iptables
# sample configuration for iptables service
# you can edit this manually or use system-config-firewall
# please do not ask us to add additional ports/services to this default configuration
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -s 192.168.1.0/24 -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -s 192.168.1.0/24 -p tcp -m state --state NEW -m tcp --dport 3128 -j ACCEPT          #开放
squid 代理端口
-A INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT                             #开放
dashboard 访问端口
-A INPUT -p tcp -m state --state NEW -m tcp --dport 6080 -j ACCEPT                           #开放
```

控制台 vnc 访问端口
-A INPUT -p tcp -m state --state NEW -m tcp --dport 15672 -j ACCEPT #开放
RabbitMQ 访问端口
-A INPUT -p tcp -m state --state NEW -m tcp --dport 10050 -j ACCEPT
#-A INPUT -j REJECT --reject-with icmp-host-prohibited #注意，这两行要注释掉！不然，开启这两行后，虚拟机之间就相互 ping 不通了！
#-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT

说明：
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
这两条的意思是在 INPUT 表和 FORWARD 表中拒绝所有其他不符合上述任何一条规则的数据包。并且发送一条 host prohibited 的消息给被拒绝的主机。
这个是 iptables 的默认策略，可以删除这两行，并且配置符合自己需求的策略。

这两行策略开启后，宿主机和虚拟机之间的 ping 无阻碍
但虚拟机之间就相互 ping 不通了，因为 vm 之间 ping 要经过宿主机，这两条规则阻碍了他们之间的通信！删除即可~

重启虚拟机
这样，开启防火墙后，宿主机和虚拟机，虚拟机之间都可以相互 ping 通~
[root@linux-node2 ~]# systemctl restart iptables.service

****
openstack 私有云环境，在一个计算节点上创建的虚拟机，其实就是一个局域网内的机器群了。
如上述在宿主机上开启防火墙，一番设置后，虚拟机和宿主机之间/同一个节点下的虚拟机之间/虚拟机和宿主机同一内网段内的机器之间都是可以相互连接的，即能相互 ping 通
****

2）虚拟机的 web 应用的代理部署

两种方案（宿主机上部署 nginx 或 haproxy）：

a.采用 nginx 的反向代理。即将各个域名解析到宿主机 ip，在 nginx 的 vhost 里配置，然后通过 proxy_pass 代理转发到虚拟机上。

b.采用 haproxy 代理。也是将各个域名解析到宿主机 ip，然后通过域名进行转发规则的设置。

这样，就能保证通过宿主机的 80 端口，将各个域名的访问请求转发给相应的虚拟机了。

nginx 反向代理，可以参考下面两篇博客：

http://www.linuxidc.com/Linux/2017-02/140399.htm

http://www.linuxidc.com/Linux/2017-02/140400.htm

****

nginx 反向代理思路：
在宿主机上启动 nginx 的 80 端口，根据不通域名进行转发；后端的虚拟机上 vhost 下不同域名的配置要启用不同的端口了~

比如：
在宿主机上下面两个域名的代理配置（其他域名配置同理）

```
[root@linux-node1 vhosts]# cat www.world.com.conf
upstream 8080 {
server 192.168.1.150:8080;
}

server {
listen 80;
```

```
server_name www.world.com;
location / {
proxy_store off;
proxy_redirect off;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header Host $http_host;
proxy_pass http://8080;
}
}

[root@linux-node1 vhosts]# cat www.tech.com.conf
upstream 8081 {
server 192.168.1.150:8081;
}

server {
listen 80;
server_name www.tech.com;
location / {
proxy_store off;
proxy_redirect off;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header Host $http_host;
proxy_pass http://8081;
}
}
```

即 www.world.com 和 www.tech.com 域名都解析到宿主机的公网 ip 上，然后：
访问 http://www.world.com 的请求就被宿主机代理到后端虚拟机 192.168.1.150 的 8080 端口上，即在虚拟机上这个域名配置的是 8080 端口；
访问 http://www.tech.com 的请求就被宿主机代理到后端虚拟机 192.168.1.150 的 8081 端口上，即在虚拟机上这个域名配置的是 8081 端口；

要是后端虚拟机配置了多个域名，那么其他域名的配置和上面是一样的~~~

另外：
最好在**代理服务器**和**后端真实服务器**上做 host 映射（/etc/hosts 文件里将各个域名指定对应到 127.0.0.1），不然，可能代理后访问域名有问题~~

由于宿主机上做 web 应用的代理转发，需要用到 80 端口。
80 端口已被 dashboard 占用，这里需要修改下 dashboard 的访问端口，比如改为 8080 端口
则需要做如下修改：
1）vim /etc/httpd/conf/httpd.conf
将 80 端口修改为 8080 端口

```
Listen 8080
ServerName 192.168.1.8:8080
```

2）vim /etc/openstack-dashboard/local_settings #将下面两处的端口由 80 改为 8080

```
'from_port': '8080',
'to_port': '8080',
```

3）防火墙添加 8080 端口访问规则

```
-A INPUT -p tcp -m state --state NEW -m tcp --dport 8080 -j ACCEPT
```

然后重启 http 服务：

```
#systemctl restart httpd
```

这样，dashboard 访问 url：
http://58.68.250.17:8080/dashboard

欢迎点击这里的链接进入精彩的Linux公社 网站

Linux公社（www.Linuxidc.com）于2006年9月25日注册并开通网站，Linux现在已经成为一种广受关注和支持的一种操作系统，IDC是互联网数据中心，LinuxIDC就是关于Linux的数据中心。

Linux公社是专业的Linux系统门户网站，实时发布最新Linux资讯，包括Linux、Ubuntu、Fedora、RedHat、红旗Linux、Linux教程、Linux认证、SUSE Linux、Android、Oracle、Hadoop、CentOS、MySQL、Apache、Nginx、Tomcat、Python、Java、C语言、OpenStack、集群等技术。

Linux公社（LinuxIDC.com）设置了有一定影响力的Linux专题栏目。

Linux公社 主站网址：www.linuxidc.com 旗下网站：www.linuxidc.net

包括：**Ubuntu 专题** **Fedora 专题** **Android 专题** **Oracle 专题** **Hadoop 专题** **RedHat 专题** **SUSE 专题** **红旗 Linux 专题** **CentOS 专题**

Linux 公社微信公众号：linuxidc_com

为虚拟机指定固定ip

openstack 在 neutron 组网模式下默认采用 DHCP-Agent 模块给虚拟机自动分配 ip

现在想给虚拟机指定固定 ip，即创建虚拟机的时候指定固定 ip。

现分析如下：

背景

1、我们目前使用 openstack+docker 来搭建自己的私有云
2、openstack 有两种网络环境搭建模式，一种是功能较简单的 nova-network，一种是 neutron 方案
3、neutron 方案代表着未来的趋势，提供更多高级的功能(例如路由功能和负载均衡服务等)，更加方便用户去自定义自己的虚拟化网络
4、在已有的几个集群中，我们在线下开发测试环境中搭建了 neutron 方案的 openstack 集群

目的

neutron 的设计理念是 ip 分配应当资源池化，因此在默认的 dashboard 操作界面上，只能为每个虚拟机指定特定的子网，虚拟机启动时会自动分配该子网可用的 ip 资源。但是，在很多开发测试场景下，我们还是需要为指定启动的虚拟机配备一个固定的 ip，比如需要反复创建、删除虚拟机，这个时候就会希望虚拟机的 ip 不变，方便测试用例的编写，固定的 ip 地址也有利于 CMDB 的管理。

前置条件

所有的操作都基于 openstack 项目提供的 NeutronClient 和 NovaClient 工具，请确保这两个工具已经安装，可以登录 openstack 的集群管理节点，在命令行界面执行 nova 和 neutron 命令，看是否有帮助提示。执行这两个工具，还需要预先获得权限，例如在管理节点上执行这两个命令的话，首先要执行以下命令，执行后就会在环境变量中保存有 admin 用户的权限信息

```
[root@openstack-server src]# source admin-openrc.sh
```

neutron 通过修改 dhcp 服务器的配置文件实现给指定虚拟机配置固定 ip，因此要首先要确保虚拟机准备连接的子网的 DHCP 功能已开启？
查看子网的详细信息，确认子网的 dhcp 功能已经开启

```
[root@openstack-server src]# neutron subnet-list
+--------------------------------------+------------+---------------+--------------------------------------------
----------+
| id | name | cidr | allocation_pools |
+--------------------------------------+------------+---------------+--------------------------------------------
----------+
| c53da14a-01fe-4f6c-8485-232489deaa6e | flat-subnet | 192.168.1.0/24 | {"start": "192.168.1.100",
 "end": "192.168.1.200"} |
+--------------------------------------+------------+---------------+--------------------------------------------
----------+

[root@openstack-server src]# neutron subnet-show c53da14a-01fe-4f6c-8485-232489deaa6e
+------------------+------------------------------------------------+
| Field | Value |
+------------------+------------------------------------------------+
| allocation_pools | {"start": "192.168.1.100", "end": "192.168.1.200"} |
| cidr | 192.168.1.0/24 |
| dns_nameservers | 192.168.1.17 |
| enable_dhcp | True |
| gateway_ip | 192.168.1.17 |
| host_routes | |
| id | c53da14a-01fe-4f6c-8485-232489deaa6e |
| ip_version | 4 |
| ipv6_address_mode | |
| ipv6_ra_mode | |
| name | flat-subnet |
| network_id | 1d9657f6-de9e-488f-911f-020c8622fe78 |
| subnetpool_id | |
| tenant_id | 65a0c00638c247a0a274837aa6eb165f |
+------------------+------------------------------------------------+
```

如上，“enable_dhcp”是“True”，表明子网的 dhcp 功能已经开启。

****

如果子网的 dhcp 功能没有开启，可以手动设置进行开启该功能！

```
[root@openstack-server src]# neutron subnet-update --enable-dhcp c53da14a-01fe-4f6c-8485-232489deaa6e
```

****

下面详细说下，创建虚拟机的时候，指定固定 ip 的方法：

1）创建一个和指定子网相关联的端口，并为该端口配置一个固定 ip，具体命令格式：

```
#neutron port-create --fixed-ip subnet_id=SUBNET_ID,ip_address=IP_FROM_POOL --name PORT_NAME NETWORK_ID
```

说明：

固定 ip 由自己决定分配哪一个 ip，只要这个 ip 在子网的可用 ip 范围内，且该 ip 还未被使用即可。

PORT_NAME 自行命名

NETWORK_ID 可在前面的 neutron subnet-show 命令的执行结果中找到。

```
[root@openstack-server src]# neutron port-create --fixed-ip subnet_id=c53da14a-01fe-4f6c-8485-232489deaa6e,ip_address=192.168.1.101 --name kvm-server001 1d9657f6-de9e-488f-911f-020c8622fe78
Created a new port:
+-----------------------+--------------------------------------------------------------------------------------+
| Field | Value |
+-----------------------+--------------------------------------------------------------------------------------+
| admin_state_up | True |
| allowed_address_pairs | |
| binding:host_id | |
| binding:profile | {} |
| binding:vif_details | {} |
| binding:vif_type | unbound |
| binding:vnic_type | normal |
| device_id | |
| device_owner | |
| dns_assignment | {"hostname": "host-192-168-1-101", "ip_address": "192.168.1.101", "fqdn": "host-192-168-1-101.openstacklocal."} |
| dns_name | |
| fixed_ips | {"subnet_id": "c53da14a-01fe-4f6c-8485-232489deaa6e", "ip_address": "192.168.1.101"} |
| id | 8cc0b915-773d-45b7-9c3a-0e8198818637 |
| mac_address | fa:16:3e:ce:bf:a5 |
| name | kvm-server001 |
| network_id | 1d9657f6-de9e-488f-911f-020c8622fe78 |
| port_security_enabled | True |
| security_groups | 050a6341-57c5-4b01-bc79-09efd9931d9c |
| status | DOWN |
| tenant_id | 65a0c00638c247a0a274837aa6eb165f |
+-----------------------+--------------------------------------------------------------------------------------+
```

2）

启动虚拟机，并在参数中指定要将虚拟机绑定到刚创建的 port 上，这样虚拟机就会被自动配置 port 已设置的 ip，具体命令格式：

```
#nova boot --flavor FLAVOR_ID --image IMAGE_ID --nic port-id=PORT_ID INSTANCE_NAME
```

接着，创建虚拟机，指定固定 ip：【具体参考本文前面里面创建虚拟机的步骤】

```
[root@openstack-server src]# nova boot --flavor kvm002 --image CentOS-6.5 --nic port-id=8cc0b915-773d-45b7-9c3a-0e8198818637 --security-group default --key-name mykey kvm-server001
+--------------------------------------+-----------------------------------------------+
| Property | Value |
+--------------------------------------+-----------------------------------------------+
| OS-DCF:diskConfig | MANUAL |
| OS-EXT-AZ:availability_zone | |
| OS-EXT-SRV-ATTR:host | - |
| OS-EXT-SRV-ATTR:hypervisor_hostname | - |
| OS-EXT-SRV-ATTR:instance_name | instance-00000017 |
| OS-EXT-STS:power_state | 0 |
| OS-EXT-STS:task_state | scheduling |
| OS-EXT-STS:vm_state | building |
| OS-SRV-USG:launched_at | - |
| OS-SRV-USG:terminated_at | - |
| accessIPv4 | |
| accessIPv6 | |
| adminPass | mFAKr7auzXv8 |
| config_drive | |
| created | 2016-08-30T08:47:06Z |
| flavor | kvm002 (938dd195-ad12-4750-836f-bc8a29a3f7ed) |
| hostId | |
| id | 1a611deb-8560-43fb-a267-cf51c48da709 |
| image | CentOS-6.5 (508db9d4-6c9f-459d-8782-065ee8b6f2c2) |
| key_name | mykey |
| metadata | {} |
| name | kvm-server001 |
| os-extended-volumes:volumes_attached | [] |
| progress | 0 |
| security_groups | default |
| status | BUILD |
| tenant_id | 65a0c00638c247a0a274837aa6eb165f |
| updated | 2016-08-30T08:47:06Z |
| user_id | b29da729de0b4ac2b3be9b519817a2b9 |
+--------------------------------------+-----------------------------------------------+
```

查看创建的虚拟机，发现 ip 已经是固定的了！

```
[root@openstack-server src]# nova list
+--------------------------------------+--------------+--------+------------+-------------+------------------
--+
| ID | Name | Status | Task State | Power State | Networks |
+--------------------------------------+--------------+--------+------------+-------------+------------------
--+
| 1a611deb-8560-43fb-a267-cf51c48da709 | kvm-server001 | ACTIVE | - | Running | flat=192.168.1.
101 | |
+--------------------------------------+--------------+--------+------------+-------------+------------------
--+
```

创建好后，使用镜像里的 root 密码登陆虚拟机，发现 ip 是上面固定的 ip 了！

这个时候，也可以手动修改网卡配置：由 dhcp 修改成 static 静态方式。

然后重启网卡和虚拟机后，ip 不会再变成其他的地址的~~

****

openstack 私有云环境，在一个计算节点上创建的虚拟机，其实就是一个局域网内的机器群了。

虚拟机和宿主机之间 / 同一个节点下的虚拟机之间 / 虚拟机和宿主机同一内网段内的机器之间都是可以相互连接的，即能相互 ping 通

****

如果不采用上述方法：即创建虚拟机的时候，不指定固定 ip，默认用 dhcp 自动分配 ip 方式创建虚拟机。

特别注意：
如果创建虚拟机时不按照上面指定 ip 操作，那么创建虚拟机后，可登陆机器修改配置网卡文件，**将 dhcp 方式改为 static 方式！**
但是，**ip 不能修改为其他 ip 地址，必须修改为 dhcp 自动为其分配的地址！**
要是修改为其他地址，则就和其他虚拟机和同网段内的机器 ping 不通了！
具体是什么原因导致以及解决方案，有待后续排查~~~

如下，虚拟机 kvm-server005 创建后，ip 是 dhcp 自动分配的：192.168.1.123

云主机名称	镜像名称	IP 地址	配置	值对	状态
kvm-server005	CentOS-6.5	192.168.1.123	kvm002	mykey	运行

登陆 kvm-server005 虚拟机本机进行修改：

```
[root@kvm-server005 ~]# cd /etc/sysconfig/network-scripts/
[root@kvm-server005 network-scripts]# cat ifcfg-eth0
DEVICE="eth0"
BOOTPROTO="dhcp"
IPV6INIT="yes"
NM_CONTROLLED="yes"
ONBOOT="yes"
TYPE="Ethernet"
UUID="db795113-37af-407a-9f78-62f49e26d5c2"
```

改为 static 静态 ip 方式

```
[root@kvm-server005 network-scripts]# cat ifcfg-eth0
DEVICE="eth0"
BOOTPROTO="static"
IPADDR=192.168.1.123
NETMASK=255.255.255.0
GATEWAY=192.168.1.17
IPV6INIT="yes"
NM_CONTROLLED="yes"
ONBOOT="yes"
TYPE="Ethernet"
UUID="db795113-37af-407a-9f78-62f49e26d5c2"
```

重启网卡，ip 就改为静态 ip 了

用OZ工具制作openstack镜像

在部署 openstack 云平台环境的时候，需要上传镜像到 glance。

首先下载 iso 镜像，这里下载了 centos6.5 镜像，放到/usr/local/src 目录下
然后用 OZ 工具制作 openstack 的镜像

**********安装 libvirt 虚拟机软件**********
[root@openstack-server src]# **yum install qemu-kvm libvirt libvirt-python libguestfs-tools virt-install**
[root@openstack-server src]# **systemctl enable libvirtd && systemctl start libvirtd**
****

[root@openstack-server src]#**yum install -y oz libguestfs-tools**

```
[root@openstack-server src]# pwd
/usr/local/src
[root@openstack-server src]# ll CentOS-6.5-x86_64-bin-DVD1.iso #下载的 iso 镜像
-rw-r--r--. 1 root root 4467982336 Nov 29 2013 CentOS-6.5-x86_64-bin-DVD1.iso
[root@openstack-server src]# cat CentOS6u5-x86_64.tdl #创建 tdl 文件
<template>
<name>CentOS6u5-x86_64</name>
<description>CentOS6u5-x86_64 template</description>
<os>
<name>CentOS-6</name>
<version>5</version>
<arch>x86_64</arch>
<rootpw>PASSWORD</rootpw>          #这个是虚拟机创建好后，root 的登陆密码，密码是在这个镜像里定义的！
<install type='iso'>
<iso>file:///usr/local/src/CentOS-6.5-x86_64-bin-DVD1.iso</iso>
</install>
</os>
<commands>              #centos7 系统里没有/boot/grub/grub.conf 文件，这块可以不用写，但写了也无妨。这里我是 centos7，也写了
<command name='console'>
sed -i 's/ rhgb//g' /boot/grub/grub.conf
sed -i 's/ quiet//g' /boot/grub/grub.conf
sed -i 's/ console=tty0 / serial=tty0 console=ttyS0,115200n8 /g' /boot/grub/grub.conf
</command>
</commands>
</template>
```

在用 OZ 工具制作 openstack 镜像，有报错：
[root@openstack-server src]# **oz-install -u -d3 CentOS6u5-x86_64.tdl** #报错如下：

```
.................
raise oz.OzException.OzException("Could not find a libvirt bridge. Please run 'virsh net-start default' to start the default libvirt network, or see http://github.com/clalancette/oz/wiki/Oz-Network-Configuration for more information")
oz.OzException.OzException: Could not find a libvirt bridge. Please run 'virsh net-start default' to start the default libvirt network, or see http://github.com/clalancette/oz/wiki/Oz-Network-Configuration for more information
```

```
[root@openstack-server src]# virsh net-start default
error: failed to get network 'default'
error: Network not found: no network with matching name 'default'
```

查阅资料，说是因为 default 网络不存在
回想了一下，我开始在捣鼓 openstack 的时候，可能用命令删除了 default，也或许没有启动 default 网络
因为在不同的环境下，default.xml 的存放路径不同，这里笔者以自己的 centos7 为例

```
[root@openstack-server src]# find / -name "default.xml"
/etc/libvirt/qemu/networks/autostart/default.xml
/etc/libvirt/qemu/networks/default.xml
/usr/share/backgrounds/default.xml
/usr/share/libvirt/networks/default.xml
[root@openstack-server src]# virsh net-define /usr/share/libvirt/networks/default.xml
Network default defined from /usr/share/libvirt/networks/default.xml

[root@openstack-server src]# virsh net-start default
Network default started

[root@openstack-server src]# virsh net-list
Name State Autostart Persistent
----------------------------------------------------------
default active no yes

[root@openstack-server src]#
```

然后接着再进行 openstack 镜像的制作：

```
[root@openstack-server src]# oz-install -u -d3 CentOS6u5-x86_64.tdl
libvirt bridge name is virbr0
Libvirt type is kvm
Name: CentOS6u5-x86_64, UUID: 0a9b1d18-f517-40ae-9de9-1fd6101878e2
MAC: 52:54:00:fb:0b:c9, distro: CentOS-6
update: 5, arch: x86_64, diskimage: /var/lib/libvirt/images/CentOS6u5-x86_64.dsk
nicmodel: virtio, clockoffset: utc
mousetype: ps2, disk_bus: virtio, disk_dev: vda
icicletmp: /var/lib/oz/icicletmp/CentOS6u5-x86_64, listen_port: 36050
Original ISO path: /var/lib/oz/isos/CentOS-65x86_64-iso.iso
Modified ISO cache: /var/lib/oz/isos/CentOS-65x86_64-iso-oz.iso
Output ISO path: /var/lib/libvirt/images/CentOS6u5-x86_64-iso-oz.iso
ISO content path: /var/lib/oz/isocontent/CentOS6u5-x86_64-iso
Checking for guest conflicts with CentOS6u5-x86_64
Generating install media
Attempting to get the lock for /var/lib/oz/isos/CentOS-65x86_64-iso.iso
Got the lock for /var/lib/oz/isos/CentOS-65x86_64-iso.iso
Fetching the original media
Fetching the original install media from file:///usr/local/src/CentOS-6.5-x86_64-bin-DVD1.iso
15kB of 4363264kB
10255kB of 4363264kB
20494kB of 4363264kB
30734kB of 4363264kB
40973kB of 4363264kB
51212kB of 4363264kB
.....
.....
.....
Cleaning up guestfs handle for CentOS6u5-x86_64
Syncing
Unmounting all
Libvirt XML was written to CentOS6u5-x86_64Aug_30_2016-13:47:18
```

注意：

镜像制作完默认会存放到/var/lib/libvirt/images/ 目录下，可以在/etc/oz/oz.cfg 配置文件中进行修改路径。

```
[root@openstack-server src]# cd /var/lib/libvirt/images/
[root@openstack-server images]# pwd
/var/lib/libvirt/images
[root@openstack-server images]# ll
total 1087336
-rw-rw-rw-. 1 root root 10737418240 Aug 30 13:47 CentOS6u5-x86_64.dsk
[root@openstack-server images]# virt-sysprep --add CentOS6u5-x86_64.dsk
[ 0.0] Examining the guest ...
[ 5.0] Performing "abrt-data" ...
```

```
.....
[ 5.0] Setting a random seed
[ 5.0] Performing "lvm-uuids" ...
```

查看文件信息

```
[root@openstack-server images]# qemu-img info CentOS6u5-x86_64.dsk
image: CentOS6u5-x86_64.dsk
file format: raw
virtual size: 10G (10737418240 bytes)
disk size: 1.0G
```

上传镜像到 Glance

```
[root@openstack-server images]# glance image-create --name "CentOS-6.5" --file /var/lib/libvirt/images/CentOS6u5-x86_64.dsk --disk-format qcow2 --container-format bare --visibility public --progress
[==============================>] 100%

+------------------+--------------------------------------+
| Property | Value |
+------------------+--------------------------------------+
| checksum | 2d16e5ef687fead34fa801aafe37f058 |
| container_format | bare |
| created_at | 2016-08-30T06:03:22Z |
| disk_format | qcow2 |
| id | 508db9d4-6c9f-459d-8782-065ee8b6f2c2 |
| min_disk | 0 |
| min_ram | 0 |
| name | CentOS-6.5 |
| owner | 65a0c00638c247a0a274837aa6eb165f |
| protected | False |
| size | 10737418240 |
| status | active |
| tags | [] |
| updated_at | 2016-08-30T06:04:26Z |
| virtual_size | None |
| visibility | public |
+------------------+--------------------------------------+
```

查看镜像

```
[root@openstack-server images]# glance image-list
+--------------------------------------+------------+
| ID | Name |
+--------------------------------------+------------+
| 508db9d4-6c9f-459d-8782-065ee8b6f2c2 | CentOS-6.5 |
+--------------------------------------+------------+
```

登陆 openstack 界面，发现可以查看到上面已经上传到 glance 里面的镜像。上面使用 OZ 工具制作的 openstack 所需要的 Centos6/Centos7 镜像的默认大小是 10G（如下图）。

注意：

如果不想用 OZ 工具制作，可以直接下载 centos 的 qcow2 格式镜像

下载地址：http://cloud.centos.org/centos 【有 centos6/7 的镜像】

比如：下载 centos7 的 qcow2 格式镜像
#wget http://cloud.centos.org/centos/7/images/CentOS-7-x86_64-GenericCloud.qcow2

上传到 galnce
#glance image-create --name "CentOS-7-x86_64" --disk-format qcow2 --container-format bare --file CentOS-7-x86_64-GenericCloud.qcow2 --visibility public --progress

---

当然，也可以使用 OZ 工具制作 openstack 需要的 qcow2 的 Centos7 镜像

```
[root@openstack-server src]# pwd
/usr/local/src
[root@openstack-server src]# ls CentOS-7-x86_64-DVD-1511.iso
CentOS-7-x86_64-DVD-1511.iso
[root@openstack-server src]# ls CentOS-7-x86_64.tdl
CentOS-7-x86_64.tdl
[root@openstack-server src]# cat CentOS-7-x86_64.tdl
<template>
<name>CentOS-7-x86_64</name>
<description>CentOS-7-x86_64 template</description>
<os>
```

```
<name>CentOS-7</name>
<version>2</version>
<arch>x86_64</arch>
<rootpw>PASSWORD</rootpw>
<install type='iso'>
<iso>file:///usr/local/src/CentOS-7-x86_64-DVD-1511.iso</iso>
</install>
</os>
<commands>
<command name='console'>
sed -i 's/ rhgb//g' /boot/grub/grub.conf
sed -i 's/ quiet//g' /boot/grub/grub.conf
sed -i 's/ console=tty0 / serial=tty0 console=ttyS0,115200n8 /g' /boot/grub/grub.conf
</command>
</commands>
</template>


[root@openstack-server src]# oz-install -u -d3 CentOS-7-x86_64.tdl
```

后面的步骤跟上面制作 Centos6.5 版本的操作一样

下面是 centos6.8 版本镜像制作的 tdl 文件模板：

```
[root@openstack-server src]# pwd
/usr/local/src
[root@openstack-server src]# ls CentOS-6.8-x86_64-bin-DVD1.iso
CentOS-6.8-x86_64-bin-DVD1.iso

[root@openstack-server src]# cat CentOS6u8-x86_64.tdl
<template>
<name>CentOS6u8-x86_64</name>
<description>CentOS6u8-x86_64 template</description>
<os>
<name>CentOS-6</name>
<version>8</version>
<arch>x86_64</arch>
<rootpw>PASSWORD</rootpw>
<install type='iso'>
<iso>file:///usr/local/src/CentOS-6.8-x86_64-bin-DVD1.iso</iso>
</install>
</os>
<commands>
<command name='console'>
sed -i 's/ rhgb//g' /boot/grub/grub.conf
sed -i 's/ quiet//g' /boot/grub/grub.conf
sed -i 's/ console=tty0 / serial=tty0 console=ttyS0,115200n8 /g' /boot/grub/grub.conf
</command>
</commands>
</template>

[root@openstack-server src]# oz-install -u -d3 CentOS6u8-x86_64.tdl
```

后面的步骤跟上面制作 Centos6.5 版本的操作一样

下面说下使用 OZ 工具制作 openstack 虚拟化环境下的 ubuntu 版本镜像的过程记录：

下面以 ubuntu12.04 版本为例，tdl 内容参考 OZ 在 github 上的模板样式。
https://github.com/rcbops/oz-image-build/tree/master/templates

```
[root@openstack-server src]# pwd
/usr/local/src
```

```
[root@openstack-server src]# ls ubuntu-12.04-server-amd64.iso
ubuntu-12.04-server-amd64.iso
```

下面 ubuntu 的 tdl 模板可直接使用~

```
[root@openstack-server src]# cat ubuntu-12.04_x86_64.tdl
<template>
<name>ubuntu-12.04_x86_64</name>
<description>Ubuntu 12.04 15GB template</description>
<disk>
<size>15</size>                                    #镜像大小为 15G
</disk>
<os>
<name>Ubuntu</name>
<version>12.04</version>
<arch>x86_64</arch>
<rootpw>ROOT-PW_CHANGE-ME!!!</rootpw>              #root 账号登录密码
<install type='iso'>
<iso>file:///usr/local/src/ubuntu-12.04-server-amd64.iso</iso>
</install>
</os>
<commands>
<command name='console'>
sed -i 's/splash//g' /etc/default/grub
sed -i 's/quiet/console=ttyS0/g' /etc/default/grub
/usr/sbin/update-grub
</command>
<command name='update'>
apt-get update
apt-get -y upgrade
echo "cloud-init cloud-init/datasources string NoCloud, OVF, Ec2" > /tmp/debconf-selections
/usr/bin/debconf-set-selections /tmp/debconf-selections
rm -f /tmp/debconf-selections
apt-get -y install cloud-init
apt-get clean
/usr/sbin/useradd -m stack
echo "stack ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers
sed -i 's/^user: ubuntu/user: stack/g' /etc/cloud/cloud.cfg
echo -n > /etc/udev/rules.d/70-persistent-net.rules
echo -n > /lib/udev/rules.d/75-persistent-net-generator.rules
</command>
</commands>
</template>


[root@openstack-server src]# oz-install -u -d3 ubuntu-12.04_x86_64.tdl
.............
.............

Waiting for ubuntu-12.04_x86_64 to finish installing, 1200/1200
Waiting for ubuntu-12.04_x86_64 to finish installing, 1190/1200
Waiting for ubuntu-12.04_x86_64 to finish installing, 1180/1200
Waiting for ubuntu-12.04_x86_64 to finish installing, 1170/1200

.............
.............
Unmounting all
Libvirt XML was written to ubuntu-12.04_x86_64Oct_31_2016-23:01:36

[root@openstack-server src]# cd /var/lib/oz/isos/
[root@openstack-server isos]# pwd
/var/lib/oz/isos
[root@openstack-server isos]# ls                   #发现 oz 下已经有了 ubuntu12.04 镜像
Ubuntu12.04x86_64-iso.iso

[root@openstack-server src]# cd /var/lib/libvirt/images/
[root@openstack-server images]# ls
```

```
ubuntu-12.04_x86_64.dsk
[root@openstack-server images]# virt-sysprep --add ubuntu-12.04_x86_64.dsk
[ 0.0] Examining the guest ...
.....
[ 5.0] Performing "lvm-uuids" .

[root@openstack-server images]# qemu-img info ubuntu-12.04_x86_64.dsk
image: ubuntu-12.04_x86_64.dsk
file format: raw
virtual size: 15G (16106127360 bytes)
disk size: 1.3G

[root@openstack-server images]# glance image-create --name "ubuntu-12.04" --file /var/lib/libvirt/images/ubuntu-12.04_x86_64.dsk --disk-format qcow2 --container-format bare --visibility public --progress
[==============================>] 100%
+-----------------+--------------------------------------+
| Property | Value |
+-----------------+--------------------------------------+
| checksum | 15d25f4da354d8fbd5a248fc01894ceb |
| container_format | bare |
| created_at | 2016-10-31T15:25:29Z |
| disk_format | qcow2 |
| id | 042073da-e6cb-4b0b-97dd-1d5ef5be236a |
| min_disk | 0 |
| min_ram | 0 |
| name | ubuntu-12.04 |
| owner | 0cd3632df93d48d6b2c24c67f70e56b8 |
| protected | False |
| size | 64424509440 |
| status | active |
| tags | [] |
| updated_at | 2016-10-31T15:38:09Z |
| virtual_size | None |
| visibility | public |
+-----------------+--------------------------------------+

[root@linux-node2 images]# glance image-list
+--------------------------------------+--------------------+
| ID | Name |
+--------------------------------------+--------------------+
| 042073da-e6cb-4b0b-97dd-1d5ef5be236a | ubuntu-12.04 |
+--------------------------------------+--------------------+
```

登录 openstack 界面，发现上面上传到 openstack 的 ubuntu12.04 镜像已经有了，镜像大小为 15G。

问题解决

一、

```
[root@openstack-server ~]# nova list
ERROR (CommandError): You must provide a username or user id via --os-username, --os-user-id, env[OS_USERNAME] or env[OS_USER_ID]
```

解决办法：

source 一下环境变量来获取 token

```
[root@openstack-server src]# source admin-openrc.sh
[root@openstack-server src]# nova list
+--------------------------------------+-------------+--------+------------+-------------+------------------+
| ID | Name | Status | Task State | Power State | Networks |
+--------------------------------------+-------------+--------+------------+-------------+------------------+
| 056c37a4-3e94-4280-96dd-5bc003469647 | kvmserver-01 | ACTIVE | - | Running | flat=192.168.1.121 |
+--------------------------------------+-------------+--------+------------+-------------+------------------+
```

二、

在进行 keystone 认证的时候，出现下面的报错：

```
Maximum lock attempts on _lockusertokens-0b75ff3ce97a40caa6bf6753c962c82b occurred. (Disable debug mode to suppress these details.) (HTTP 500) (Request-ID: req-bb1c465e-33bc-412a-8ac2-c0d0a7e168ad)
```

或者

```
An unexpected error prevented the server from fulfilling your request. (HTTP 500) (Request-ID: req-bb1c465e-33bc-412a-8ac2-c0d0a7e168ad)
```

问题产生的可能原因及解决办法：

1）admin_token 用户环境变量错误引起的。

解决办法：

```
[root@linux-node1 ~]# export OS_TOKEN=35d6e6f377a889571bcf
[root@linux-node1 ~]# export OS_URL=http://192.168.1.17:35357/v3
[root@linux-node1 ~]# export OS_IDENTITY_API_VERSION=3
```

2）keystone 数据同步失败，没有获取到正确的 token 值引起的。

解决办法：

删除 mysql 数据库中的 keystone 库，重新创建 keystone 库；

重新生成一个 token，在 keystone.conf 中重新指定 admin_token 对应的新的 token 指；

然后初始化 keystone 数据，重启 memcaced 服务，重新进行 keystone 的认证；

3）获取的 token 无效引起的。

解决办法：修改 keystone.conf 文件中的[token]区域。

之前的配置，将 token 存放在 memcache 缓存中

```
[token]
provider = uuid              //或者是 provider = keystone.token.providers.uuid.Provider
driver = memcache            //或者是 driver = keystone.token.persistence.backends.memcache.Token
```

修改后的配置，将 token 存放在 sql 数据库中

```
[token]
provider = uuid
driver = keystone.token.persistence.backends.sql.Token
```

说明下：

keystone 默认使用 SQL 数据库存储 token，token 默认有效期为 1 天（24h）。

Openstack 中每个组件执行的每次命令（请求）都需要 token 来验证，每次访问都会创建 token，增长速度非常快，token 表数据也会越来越多。

随着时间的推移，无效的记录越来越多，企业私有云几天的量就可以几万条、几十万条。这么多无效的 token 导致针对 token 表的 SQL 语句变慢，性能也会变差！

这就需要手动写个定时脚本来清理 token 表。

但是更好的方法当然是将 token 存放在 memcache 缓存中，利用 memcached 特性，自动删除不使用的缓存：

```
[memcache]
```

```
servers = localhost:11211
[token]
provider = keystone.token.providers.uuid.Provider
driver = keystone.token.persistence.backends.memcache.Token
```

默认的配置是：

```
[token]
provider = keystone.token.providers.uuid.Provider
driver = keystone.token.persistence.backends.sql.Token
```

三、

No handlers could be found for logger "oslo_config.cfg"

原因：日志文件的配置项错误

解决办法：

在/etc/keystone/keystone.conf、/etc/nova/nova.conf 配置文件里修改 logdir 为：

log_dir=/var/log/keystone

log_dir=/var/log/nova

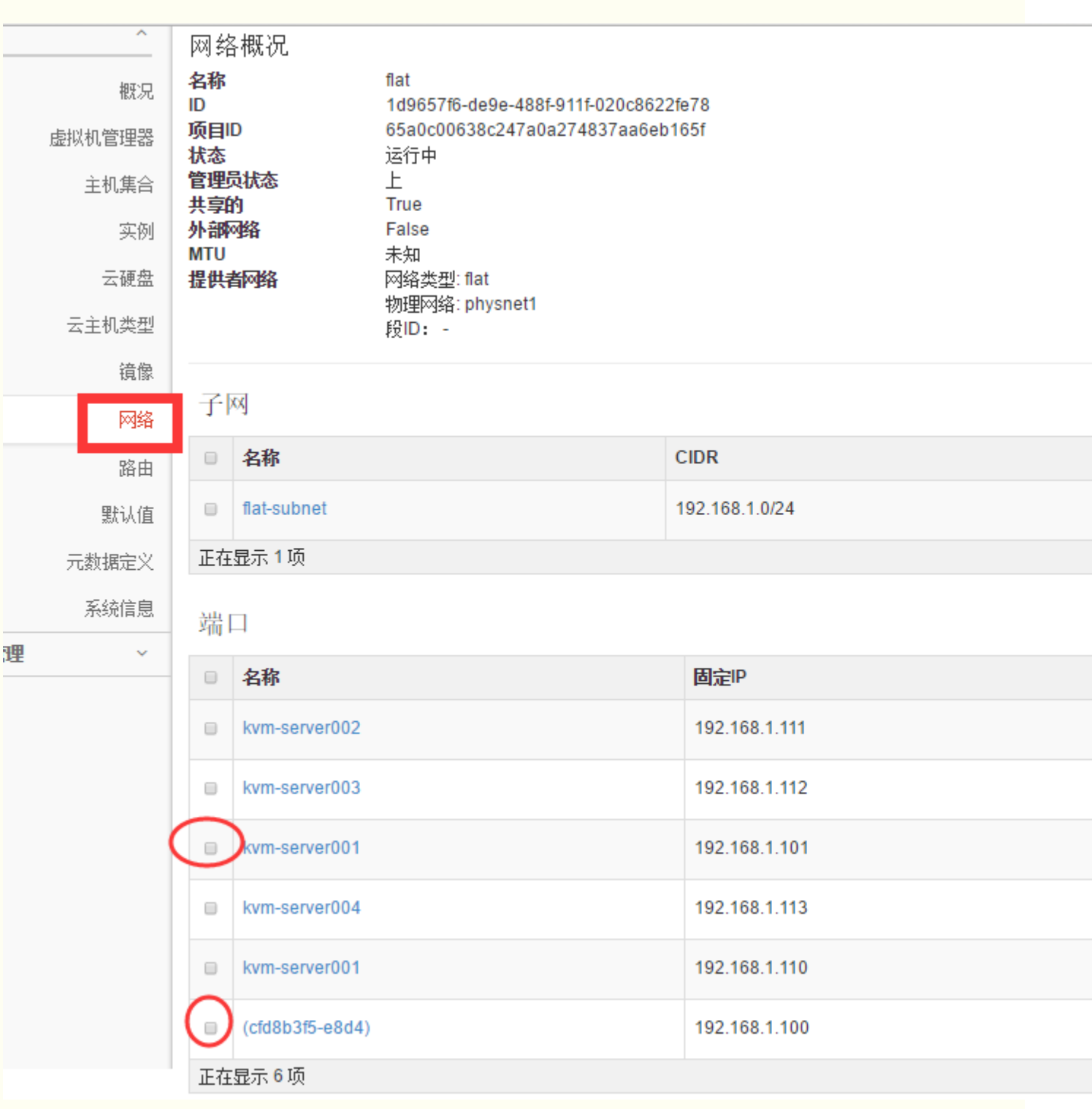
四、

之前在 openstack 里创建的虚拟机，后面删除了。但是再创建虚拟机并设置和之前删除的虚拟机一样的 ip 的时候，就报错说这个 ip 已经被占用了！

但是之前创建的虚拟机已经删除了，这是为什么？

这是因为虚拟机虽然删除了，但是所删除虚拟机在 neutron 组网内的 ip 还没有被真正释放出来。

需要登陆 openstack 的 web 界面，到“网络”部分里面进行删除：



五、

在上传镜像或查看镜像的时候，报错：

```
[root@linux-node1 ~]# glance image-list
500 Internal Server Error: The server has either erred or is incapable of performing the requested operation. (HTTP 500)
```

测试登陆数据库，发现登陆失败！

```
[root@linux-node1 ~]# mysql -u glance -h 192.168.1.17 -p
Enter password:
ERROR 1040 (08004): Too many connections
```

解决办法，这也是 centos7 下修改 mysql 连接数的做法：

1）临时修改

```
MariaDB [(none)]> set GLOBAL max_connections=1000;
Query OK, 0 rows affected (0.00 sec)
MariaDB [(none)]> show variables like "max_connections";
+-----------------+-------+
```

```
| Variable_name | Value |
+----------------+-------+
| max_connections | 1000 |
+----------------+-------+
1 row in set (0.00 sec)
```

2）永久修改：

配置/etc/my.cnf

[mysqld]新添加一行如下参数：

max_connections=1000

重启 mariadb 服务，再次查看 mariadb 数据库最大连接数，可以看到最大连接数是 214，并非我们设置的 1000。

```
MariaDB [(none)]> show variables like 'max_connections';
+----------------+-------+
| Variable_name | Value |
+----------------+-------+
| max_connections | 214 |
+----------------+-------+
```

这是由于 mariadb 有默认打开文件数限制。可以通过配置/usr/lib/systemd/system/mariadb.service 来调大打开文件数目。

配置/usr/lib/systemd/system/mariadb.service

[Service]新添加两行如下参数：

LimitNOFILE=10000

LimitNPROC=10000

重新加载系统服务，并重启 mariadb 服务

```
systemctl --system daemon-reload
systemctl restart mariadb.service
```

再次查看 mariadb 数据库最大连接数，可以看到最大连接数已经是 1000

```
MariaDB [(none)]> show variables like 'max_connections';
+----------------+-------+
| Variable_name | Value |
+----------------+-------+
| max_connections | 1000 |
+----------------+-------+
```

六、

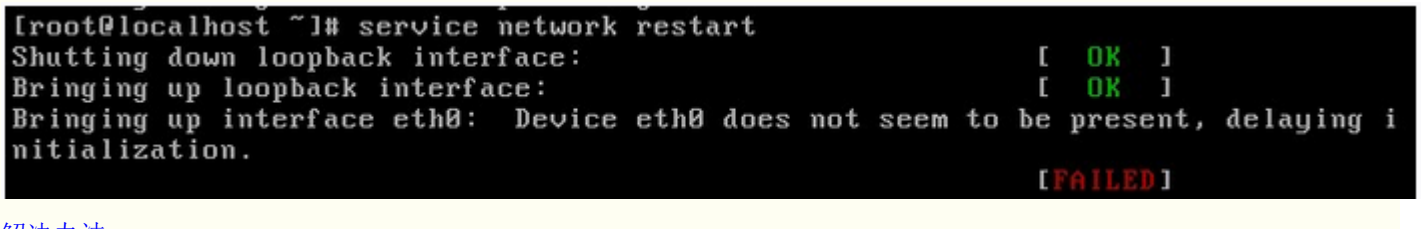
openstack 创建虚拟机后，vnc 进入 vm 后发现：

ifconfig 命令没有看到 eth0 信息

```
[root@localhost ~]# ifconfig
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)
```

重启网卡又报下面错误。

```
[root@localhost ~]# service network restart
Shutting down loopback insterface: [ OK ]
Bringing up loopback insterface:  [ OK ]
Bringing up interface eth0: Device eth0 does not seem to be present,delaying initialization. [FAILED]
```



解决办法：

首先，打开/etc/udev/rules.d/70-persistent-net.rules 内容如下面例子所示：

```
[root@localhost ~]# cat /etc/udev/rules.d/70-persistent-net.rules
```

```
# This file was automatically generated by the /lib/udev/write_net_rules
# program, run by the persistent-net-generator.rules rules file.
#
# You can modify it, as long as you keep each rule on a single
# line, and change only the value of the NAME= key.

# PCI device 0x1af4:0x1000 (virtio-pci)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="52:54:00:f0:f9:1c", ATTR
{type}=="1", KERNEL=="eth*", NAME="eth0"

# PCI device 0x1af4:0x1000 (virtio-pci)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="fa:16:3e:e9:ad:89", ATTR
{type}=="1", KERNEL=="eth*", NAME="eth1"
```

记录下，eth1 网卡的 mac 地址 fa:16:3e:e9:ad:89
接下来，打开/etc/sysconfig/network-scripts/ifcfg-eth0

```
[root@localhost ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth0"
BOOTPROTO="dhcp"
HWADDR="52:54:00:F0:F9:1C"
IPV6INIT="yes"
NM_CONTROLLED="yes"
ONBOOT="yes"
TYPE="Ethernet"
UUID="af2a1e56-a502-42e9-9359-5c7dd6b1e1e9"
```

将 ifcfg-eth0 文件下的 DEVICE 设备名称改为“eth1”
将 mac 地址即 HWADDR 改为上面记录的 eth1 的地址：fa:16:3e:e9:ad:89
即修改后：

```
[root@localhost ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth1"
BOOTPROTO="dhcp"
HWADDR="fa:16:3e:e9:ad:89"
IPV6INIT="yes"
NM_CONTROLLED="yes"
ONBOOT="yes"
TYPE="Ethernet"
UUID="af2a1e56-a502-42e9-9359-5c7dd6b1e1e9"
```

最后，重启网卡

```
[root@localhost ~]# /etc/init.d/network restart
Shutting down interface eth0: [ OK ]
Shutting down loopback interface: [ OK ]
Bringing up loopback interface: [ OK ]
Bringing up interface eth0:
Determining IP information for eth1... done.
[ OK ]
```

然后查看，发现 ifconfig 后有 ip 信息了！

```
[root@localhost ~]# ifconfig
eth1 Link encap:Ethernet HWaddr FA:16:3E:E9:AD:89
inet addr:192.168.1.102 Bcast:192.168.1.255 Mask:255.255.255.0
inet6 addr: fe80::f816:3eff:fee9:ad89/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:678 errors:0 dropped:0 overruns:0 frame:0
TX packets:183 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:78238 (76.4 KiB) TX bytes:27488 (26.8 KiB)

lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
```

```
collisions:0 txqueuelen:0
RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)
```

七、

openstack 上创建 vm 实例后，状态为 ERROR 问题解决

问题说明：

在 openstack 上创建虚拟机，之前已顺利创建了 n 个 centos6.8 镜像的 vm

现在用 ubuntu14.04 镜像创建 vm，**发现 vm 创建后的状态为 ERROR！**

1）终端命令行操作 vm 创建

```
[root@linux-node2 src]# nova boot --flavor kvm002 --image ubuntu-14.04 --nic net-id=3a5cef6e-2c12
-4f26-938c-5d343edc91b3 --security-group default --key-name mykey kvm-ubuntu01
+--------------------------------------+----------------------------------------------------+
| Property | Value |
+--------------------------------------+----------------------------------------------------+
| OS-DCF:diskConfig | MANUAL |
| OS-EXT-AZ:availability_zone | |
| OS-EXT-SRV-ATTR:host | - |
| OS-EXT-SRV-ATTR:hypervisor_hostname | - |
| OS-EXT-SRV-ATTR:instance_name | instance-00000006 |
| OS-EXT-STS:power_state | 0 |
| OS-EXT-STS:task_state | scheduling |
| OS-EXT-STS:vm_state | building |
| OS-SRV-USG:launched_at | - |
| OS-SRV-USG:terminated_at | - |
| accessIPv4 | |
| accessIPv6 | |
| adminPass | 97FNEj25qDHw |
| config_drive | |
| created | 2016-10-31T06:01:15Z |
| flavor | kvm002 (38d2c062-3fc5-4fc8-9bef-3cf16a7cf6d0) |
| hostId | |
| id | 898363d4-b5df-4603-80f3-299bba76f79c |
| image | ubuntu-14.04 (25fa5e72-5e10-4500-905a-82eda30dca21) |
| key_name | mykey |
| metadata | {} |
| name | kvm-ubuntu01 |
| os-extended-volumes:volumes_attached | [] |
| progress | 0 |
| security_groups | default |
| status | BUILD |
| tenant_id | 0cd3632df93d48d6b2c24c67f70e56b8 |
| updated | 2016-10-31T06:01:15Z |
| user_id | 52ba7917bb284af7ad6ac313b7e8e948 |
+--------------------------------------+----------------------------------------------------+
```

创建后，发现 vm 的状态是 ERROR

```
[root@linux-node2 src]# nova list
+--------------------------------------+-------------+--------+------------+-------------+----------+
| ID | Name | Status | Task State | Power State | Networks |
+--------------------------------------+-------------+--------+------------+-------------+----------+
| 898363d4-b5df-4603-80f3-299bba76f79c | kvm-ubuntu01 | ERROR | - | NOSTATE | |
+--------------------------------------+-------------+--------+------------+-------------+----------+
```

2）尝试在 openstack 的 dashboard 界面里创建 vm(即：“计算”->“实例"->"启动云主机")，**如果 vm 创建失败，则会显示错误信息**。

如上图，报错信息：

Flavor's disk is too small for requested image. Flavor disk is 16106127360 bytes, image is 2147483648 0 bytes.].

说明创建 vm 时所使用的 Flavor(云主机类型)的磁盘空间不满足 image 镜像要求！本案例是说 kvm002（15G）的根磁盘不满足 ubuntu-14.04（openstack 界面-“镜像”查看此镜像大小是 25G）镜像大小。

查看创建 vm 所使用的 Flavor 的类型

解决办法：

调整对应 Flavor 类型的跟磁盘大小(即："系统"->"云主机类型"->"编辑云主机类型"->"主机类型信息")。如下，由原来的 15G 调整到 30G！

最后，再次尝试创建 vm：

```
[root@linux-node2 src]# nova boot --flavor kvm002 --image ubuntu-14.04 --nic net-id=3a5cef6e-2c12
-4f26-938c-5d343edc91b3 --security-group default --key-name mykey kvm-ubuntu01
+--------------------------------------+------------------------------------------------+
| Property | Value |
+--------------------------------------+------------------------------------------------+
| OS-DCF:diskConfig | MANUAL |
| OS-EXT-AZ:availability_zone | |
| OS-EXT-SRV-ATTR:host | - |
| OS-EXT-SRV-ATTR:hypervisor_hostname | - |
| OS-EXT-SRV-ATTR:instance_name | instance-00000006 |
| OS-EXT-STS:power_state | 0 |
| OS-EXT-STS:task_state | scheduling |
| OS-EXT-STS:vm_state | building |
| OS-SRV-USG:launched_at | - |
| OS-SRV-USG:terminated_at | - |
| accessIPv4 | |
| accessIPv6 | |
| adminPass | 97FNEj25qDHw |
| config_drive | |
| created | 2016-10-31T06:01:15Z |
| flavor | kvm002 (38d2c062-3fc5-4fc8-9bef-3cf16a7cf6d0) |
| hostId | |
| id | 898363d4-b5df-4603-80f3-299bba76f79c |
| image | ubuntu-14.04 (25fa5e72-5e10-4500-905a-82eda30dca21) |
| key_name | mykey |
| metadata | {} |
| name | kvm-ubuntu01 |
| os-extended-volumes:volumes_attached | [] |
| progress | 0 |
| security_groups | default |
| status | BUILD |
| tenant_id | 0cd3632df93d48d6b2c24c67f70e56b8 |
| updated | 2016-10-31T06:01:15Z |
| user_id | 52ba7917bb284af7ad6ac313b7e8e948 |
+--------------------------------------+------------------------------------------------+
```

创建虚拟机后，发现 vm 可以正常启动了！

```
[root@linux-node2 src]# nova list
+--------------------------------------+-------------+--------+------------+-------------+------------------
-+
| ID | Name | Status | Task State | Power State | Networks |
+--------------------------------------+-------------+--------+------------+-------------+------------------
-+
| 729dd327-3447-42b9-b9cb-e7ef4a38b725 | kvm-ubuntu01 | ACTIVE | - | Running | flat=192.168.
1.120 |
+--------------------------------------+-------------+--------+------------+-------------+------------------
-+
```

问题：在一个计算节点上创建虚拟机，创建前几个虚拟机都没问题，但是再创建第 n 个虚拟机时就失败，报错如下：
创建云主机，状态错误，无法启动，提示 NoValidHost：**No valid host was found. There are not enough hosts available.**

查看 nova-conductor.log，如下：

```
[root@linux-node2 nova]# pwd
/var/log/nova
[root@linux-node2 nova]# tail -100 nova-conductor.log
.....................

2016-11-01 01:28:38.889 51843 WARNING nova.scheduler.utils [req-9eb2b8ec-216b-4073-95bd-1fbb
51844faf 52ba7917bb284af7ad6ac313b7e8e948 0cd3632df93d48d6b2c24c67f70e56b8 - - -] Failed to
compute_task_build_instances: No valid host was found. There are not enough hosts available.
Traceback (most recent call last):

File "/usr/lib/python2.7/site-packages/oslo_messaging/rpc/server.py", line 142, in inner
return func(*args, **kwargs)

File "/usr/lib/python2.7/site-packages/nova/scheduler/manager.py", line 84, in select_destinations
filter_properties)

File "/usr/lib/python2.7/site-packages/nova/scheduler/filter_scheduler.py", line 90, in select_destinatio
ns
raise exception.NoValidHost(reason=reason)

NoValidHost: No valid host was found. There are not enough hosts available.
```

2016-11-01 01:28:38.889 51843 WARNING nova.scheduler.utils [req-9eb2b8ec-216b-4073-95bd-1fbb51844faf 52ba7917bb284af7ad6ac313b7e8e948 0cd3632df93d48d6b2c24c67f70e56b8 - - -] [instance: 2211eeb4-9d06-4b15-ac15-69cdabe280ff] Setting instance to ERROR state.

这个问题产生的很大原因有：
1）计算节点的内存不足、CPU 资源不够、硬盘空间资源不足造成的；将云主机类型规格调小点，发现就能创建成功。
2）网络配置不正确，造成创建虚拟机的时候获取 ip 失败；网络不通或防火墙引起。
3）openstack-nova-compute 服务状态问题。可以尝试重启控制节点的 nova 相关服务和计算节点的 openstack-nova-compute 服务；详细检查控制节点和计算节点的 nova.conf 配置是否有不当配置。
4）这个报错问题的原因很多，具体要查看/var/log/nova 下的日志详细分析。重点是 nova-compute.log、nova-conductor.log 日志

在部署 openstack 虚拟机的时候，要注意以下几点：
（1）控制节点和计算节点在部署前，需要在/etc/hosts 里面对主机映射，并且后面不能轻易更改，否则会出问题！
（2）mysql 的连接数要调大！否则在操作过程中会由于 mysql 连接数过多而中断！比如设置 mysql 连接数为 1000，mysql 命令为 set GLOBAL max_connections=1000;
（3）在创建 vm 的时候，要保证 openstack 节点的内存够用。
（4）所使用的 Flavor 云主机类型配置的根磁盘要满足 image 镜像的空间。