about 云
openstack 入门宝典

版本：0226

2015 年 02 月 26 日

序言

本书是参考官网，对官网的内容进行的翻译及个人理解。后面篇章是以实际操作为主，每条命令进行了验证。

是在零基础学习 openstack【完整中级篇】及 openstack 资源汇总基础的一个继续，我们将会继续出一系列的内容。对于新手来说，一篇部署 openstack 的文章显得有些庞大，到部署完毕，才发现问题不少。但是又不能很好的定位，在部署组件时，对组件进行简单介绍，部署完每一个组件之后，都会有相关的验证，这样我们都能知道是否安装成功，专门为新手入门提供帮助。

目录（原文内容）：

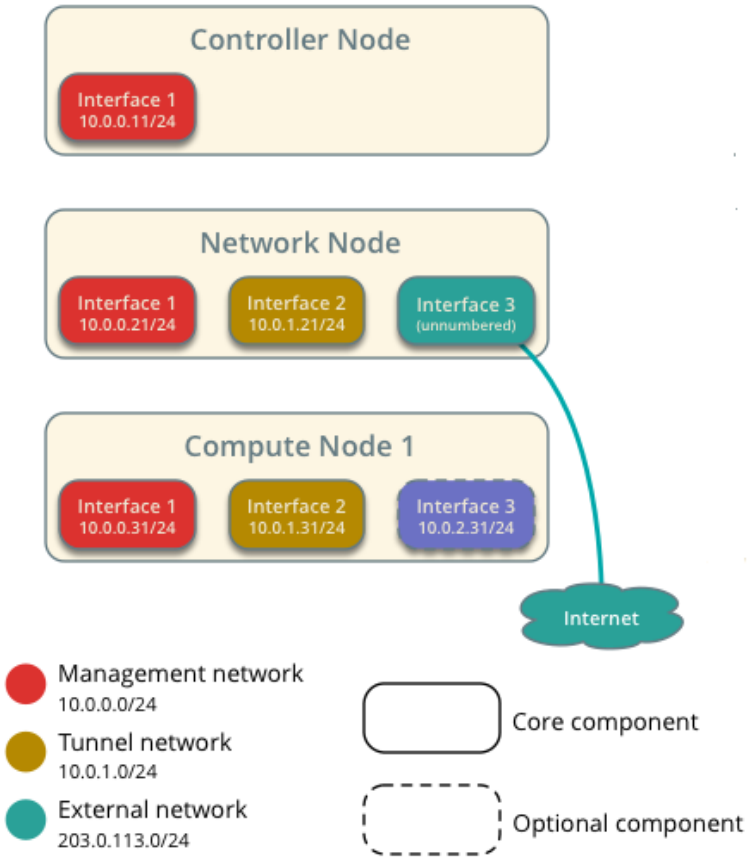
openstack【juno】入门 【准备篇】零：整体介绍

问题导读

1.对于 openstack 新节点，安装各个组件前，需要注意什么问题？

2.本文的网段是如何划分的？

网络规划：



环境：Ubuntu 14.04

包括 7 个节点，其中包括：

控制节点（1）

计算节点（1）

网络节点（1）

cinder 节点（1）

swift 节点（2）

网络划分：
10.0.0.0/24 为管理网络
10.0.1.0/24 为 tunnel 网络
203.0.113.0/24 为外部网络

cinder 节点
ip 地址：
10.0.0.41

swift 节点
swift1 ip 地址：
10.0.0.51

swift2 ip 地址：
10.0.0.51

新节点安装需要注意的问题

安装中需要注意的问题：就是我们完全根据教程安装，但是还是出错，也找不到原因。这是因为我们在安装新节点的过程中，没有安装 openstack 包。

具体如下

```
1.  apt-get install ubuntu-cloud-keyring
```

```
1.  echo "deb http://ubuntu-cloud.archive.canonical.com/ubuntu" \
2.    "trusty-updates/juno main" > /etc/apt/sources.list.d/cloudarchive-juno.list
```

```
1.  apt-get update && apt-get dist-upgrade
```

这是每个 openstack 新节点所必须安装的。否则可能出错了，也不知道哪里出问题了。

外部网络

我们部署 openstack，大多数都是使用虚拟机，在网络节点的外部网卡，我们需要注意，这个是不需要配置 ip 地址的。
同时由于每个虚拟机是需要联网的。所以我们需要在原先的网络规划的基础上，在增加一个上网的网卡。
虚拟机环境的搭建，可以参考
集群搭建必备，云技术基础：Linux 及虚拟化知识学习指导（hadoop、openstack）

openstack【juno】入门 【准备篇】一： Ubuntu14.04 远程连接（ssh 安装）

问题导读

ubuntu14.04 ssh 如何安装？
扩展：
1. Ubuntu14.04 与 ubuntu12.04 ssh 有什么区别？
2.远程连接，为什么安装 ssh

这是关于 openstack juno 部署的一系列文章，关于网络的配置，需要保证安装节点都能上网，并且 ping 通。如果这里不了解，相信搭建集群是非常困难的。
可以参考：
集群搭建必备，云技术基础：Linux 及虚拟化知识学习指导（hadoop、openstack）
当然，可以配合一些视频，等方式，达到网络互通的目的。这里由于时间的原因，以后在补充上，这里也有关于这方面知识，如果觉得 about 云对你有所帮助，亦可捐助 about 云，同时能够学到知识。
淘宝链接，下面我们从 ssh 开始讲 openstack juno 一系列的安装

ssh 是一种安全协议，主要用于给远程登录会话数据进行加密，保证数据传输的安全，现在介绍一下如何在 Ubuntu 14.04 上安装和配置 ssh

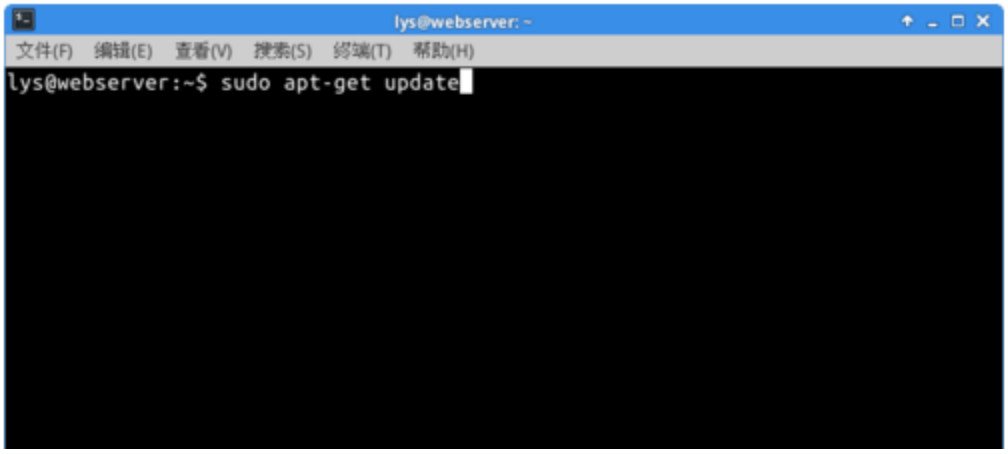
工具/原料

Ubuntu 14.04
putty v0.63

更新源列表

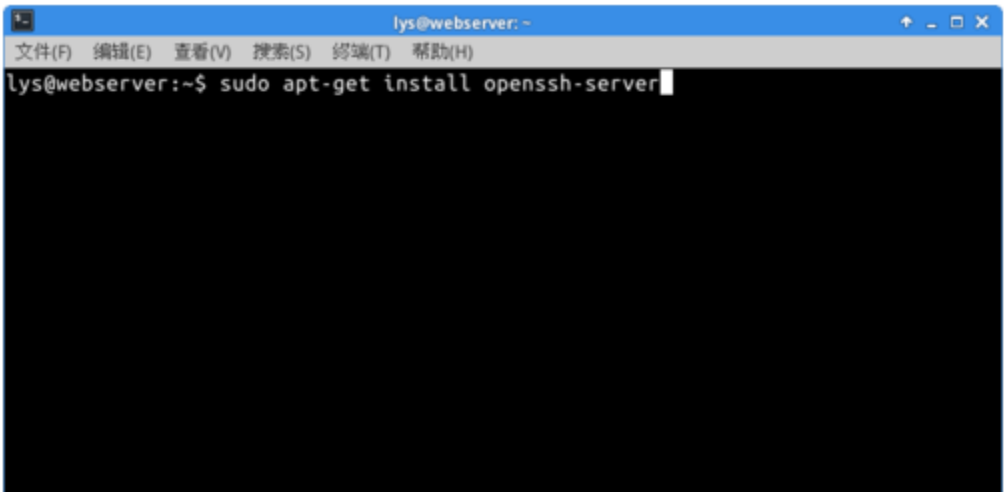
打开"终端窗口"，输入"sudo apt-get update"-->回车-->"输入当前登录用户的管理员密码"-->回车,就可以

了。



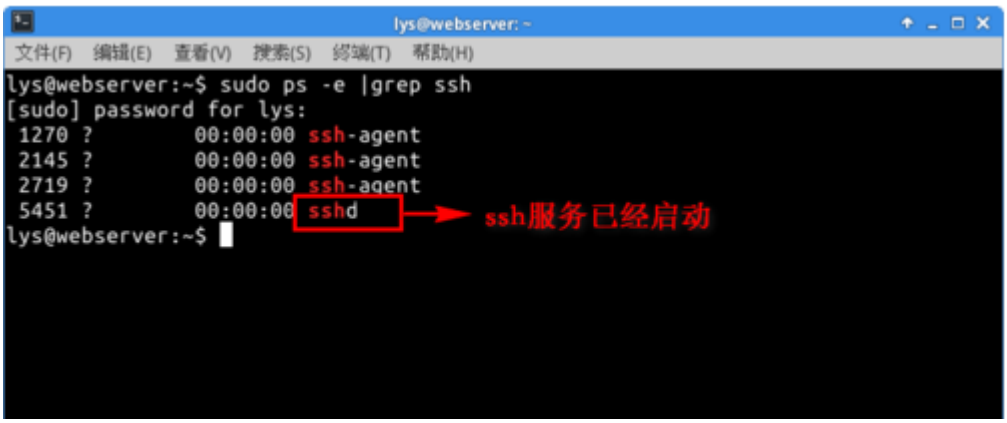
安装 ssh

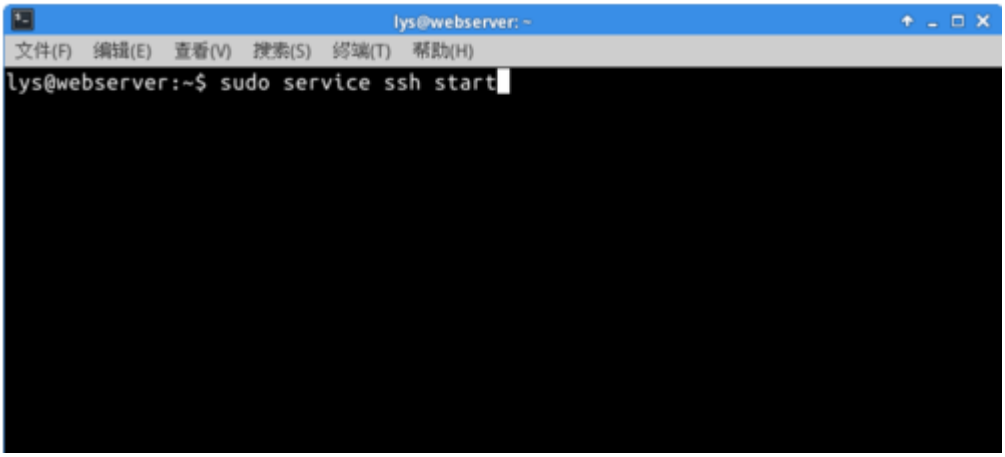
打开"终端窗口"，输入"sudo apt-get install openssh-server"-->回车-->输入"y"-->回车-->安装完成。



查看 ssh 服务是否启动

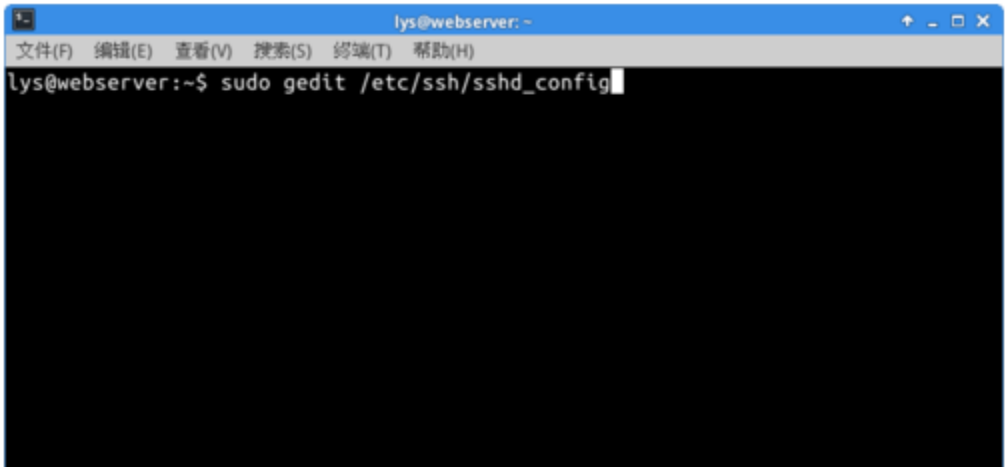
打开"终端窗口"，输入"sudo ps -e |grep ssh"-->回车-->有 sshd,说明 ssh 服务已经启动，如果没有启动，输入"sudo service ssh start"-->回车-->ssh 服务就会启动。

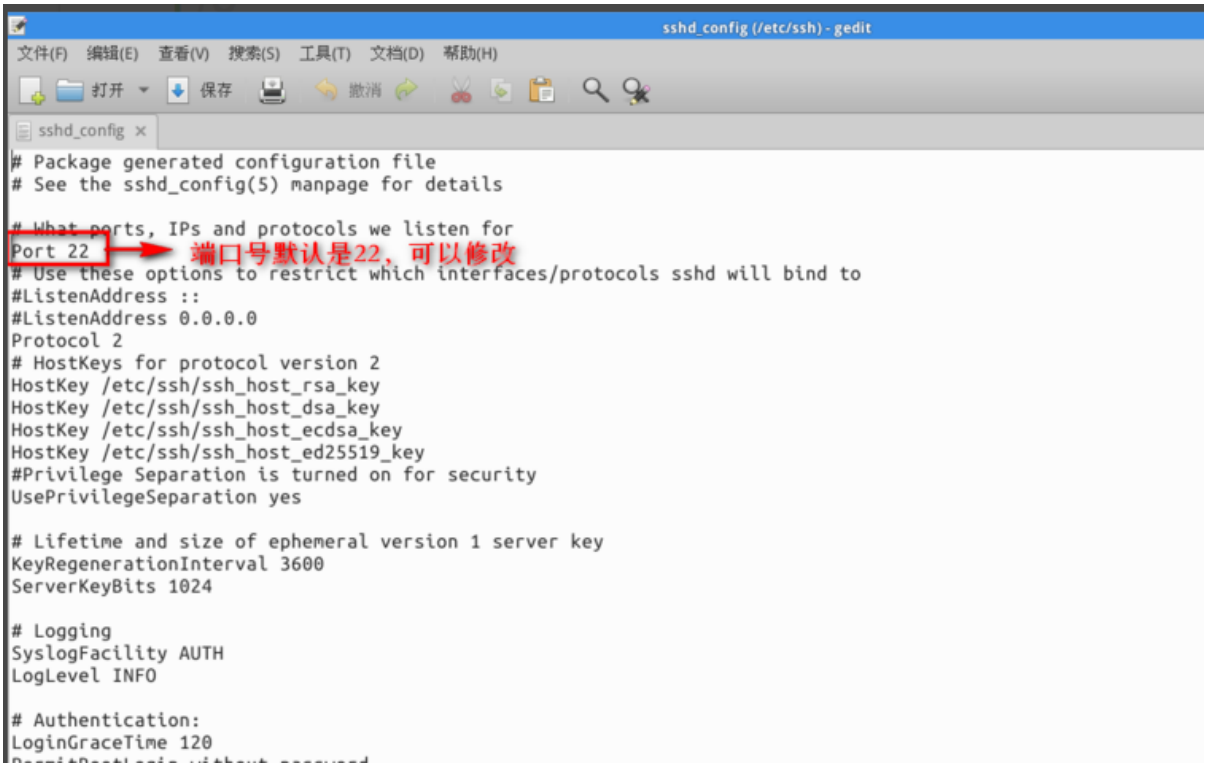


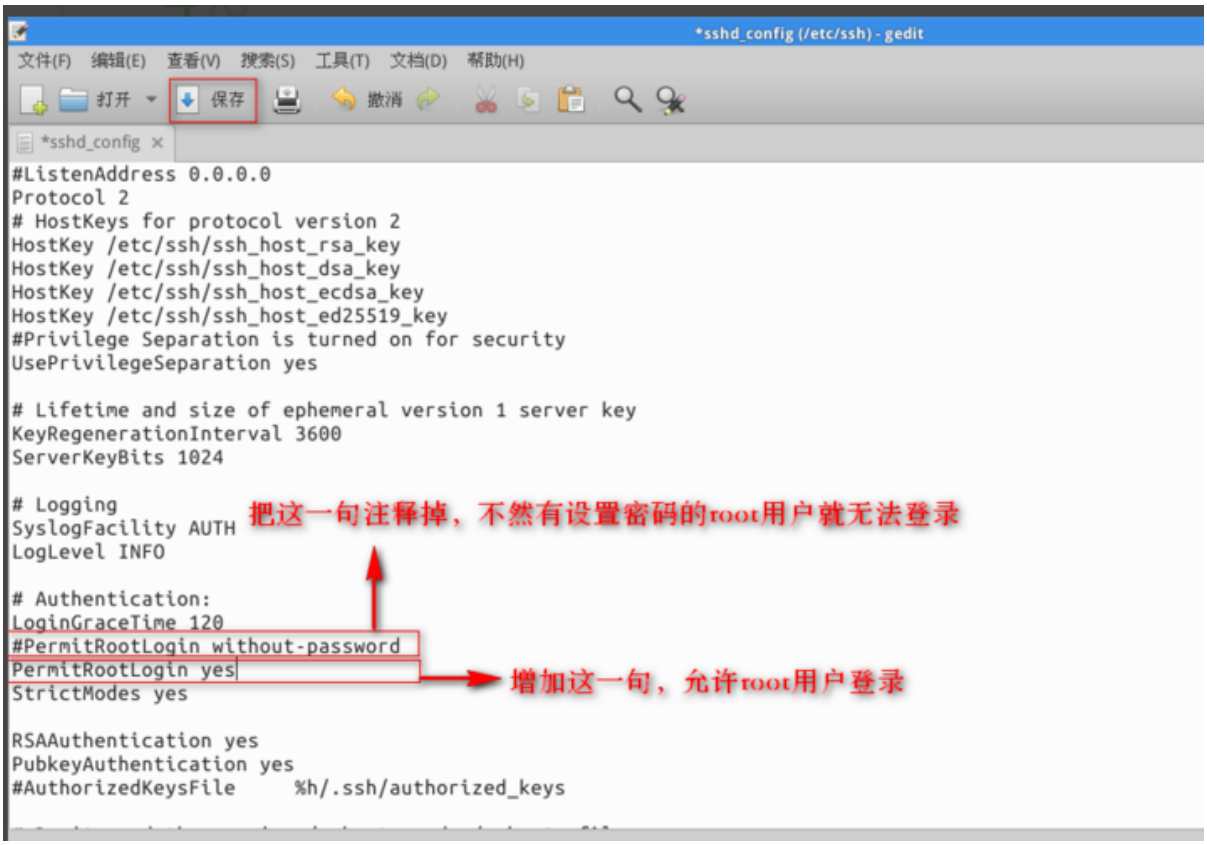


使用 gedit 修改配置文件"/etc/ssh/sshd_config"

打开"终端窗口"，输入"sudo gedit /etc/ssh/sshd_config"-->回车-->把配置文件中的"PermitRootLogin without-password"加一个"#"号,把它注释掉-->再增加一句"PermitRootLogin yes"-->保存，修改成功。




sshd_config (/etc/ssh) - gedit
文件(F) 编辑(E) 查看(V) 搜索(S) 工具(T) 文档(D) 帮助(H)
打开 保存 撤消
sshd_config
Package generated configuration file
See the sshd_config(5) manpage for details
What ports, IPs and protocols we listen for
Port 22
端口号默认是22，可以修改
Use these options to restrict which interfaces/protocols sshd will bind to
#ListenAddress ::
#ListenAddress 0.0.0.0
Protocol 2
HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes
Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 1024
Logging
SyslogFacility AUTH
LogLevel INFO
Authentication:
LoginGraceTime 120


*sshd_config (/etc/ssh) - gedit
文件(F) 编辑(E) 查看(V) 搜索(S) 工具(T) 文档(D) 帮助(H)
打开 保存 撤消
*sshd_config
#ListenAddress 0.0.0.0
Protocol 2
HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes
Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 1024
Logging
SyslogFacility AUTH
把这一句注释掉，不然有设置密码的root用户就无法登录
LogLevel INFO
Authentication:
LoginGraceTime 120
#PermitRootLogin without-password
PermitRootLogin yes
增加这一句，允许root用户登录
StrictModes yes
RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile %h/.ssh/authorized_keys

openstack【juno】入门 【准备篇】二：：NTP 安装

导读

由于 ntp 服务器时间的设置，网上错综复杂，资料很多，大家随便找个资料，这个不行，找另外的资料，总之对这个不是太了解，这里找了份资料，详细介绍了 ntp。万变不离其中，明白了原理，ntp 的配置就不在困难了。

这里面解决了一些疑问：

1.如何查看 ntp 是否配置成功？

2.如何了解 ntp 列出的参数的含义？

3.restrict 关键字的作用是什么？

控制节点

1.安装 ntp

```
1.  apt-get install ntp
```

2.修改配置文件

修改 /etc/ntp.conf，其中 NTP_SERVER 为 controller

```
1.  server NTP_SERVER iburst

2.  restrict -4 default kod notrap nomodify

3.  restrict -6 default kod notrap nomodify
```

，如果控制节点为 controller，直接使用下面即可

```
1.  server controller iburst

2.  restrict -4 default kod notrap nomodify

3.  restrict -6 default kod notrap nomodify
```

注意：

对于 restrict ,可以去掉 nopeer and noquery 这两个选项.注释掉其他 server 选项

如果 /var/lib/ntp/ntp.conf 文件存在，则移动

3.重启 NTP

```
1.  service ntp restart
```

其它节点（network，compute）

安装 NTP

```
1.  apt-get install ntp
```

修改配置文件 /etc/ntp.conf

```
1.  server controller iburst
```

如果文件存在，则移除 /var/lib/ntp/ntp.conf.dhcp

重启 NTP

```
1.  service ntp restart
```

查看是否成功（各节点运行）

```
1.  watch ntpq -p
```

上面操作过程中，可能有一些疑问，具体参考下面内容，更多参考
[让你真正明白 Linux NTP（包括历史、参数解释、安装配置）]

如果使用的是虚拟机，并且上面没有配置成功，则切换至 root 执行下面命令能够

```
1.  rm -rf /etc/localtime
2.  ln -s /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
3.
4.  yum install -y ntp
5.  ntpdate -d cn.pool.ntp.org
6.  date
```

贴出关键信息：

```
# watch ntpq -p
Every 2.0s: ntpq -p                                  Sat Jul  7 00:41:45 2007
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
+193.60.199.75   193.62.22.98     2 u   52   64  377     8.578   10.203 289.032
*mozart.musicbox 192.5.41.41      2 u   54   64  377    19.301  -60.218 292.411
```

现在我就来解释一下其中的含义

remote: 它指的就是本地机器所连接的远程 NTP 服务器

refid: 它指的是给远程服务器(e.g. 193.60.199.75)提供时间同步的服务器

st: 远程服务器的层级别（stratum）. 由于 NTP 是层型结构,有顶端的服务器,多层的 Relay Server 再到客户端. 所以服务器从高到低级别可以设定为 1-16. 为了减缓负荷和网络堵塞,原则上应该避免直接连接到级别为 1 的服务器的.

t: 这个.....我也不知道啥意思^_^

when: 我个人把它理解为一个计时器用来告诉我们还有多久本地机器就需要和远程服务器进行一次时间同步

poll: 本地机和远程服务器多少时间进行一次同步(单位为秒). 在一开始运行 NTP 的时候这个 poll 值会比较小,那样和服务器同步的频率也就增加了,可以尽快调整到正确的时间范围.之后 poll 值会逐渐增大,同步的频率也就会相应减小

reach: 这是一个八进制值,用来测试能否和服务器连接.每成功连接一次它的值就会增加

delay: 从本地机发送同步要求到服务器的 round trip time

offset: 这是个最关键的值，它告诉了我们本地机和服务器之间的时间差别. offset 越接近于 0,我们就和服务器的时间越接近

jitter: 这是一个用来做统计的值. 它统计了在特定个连续的连接数里 offset 的分布情况. 简单地说这个数值的绝对值越小我们和服务器的时间就越精确

NTP 安全设置(restrict)

运行一个 NTP Server 不需要占用很多的系统资源,所以也不用专门配置独立的服务器,就可以给许多 client 提供时间同步服务，但是一些基本的安全设置还是很有必要的

那么这里一个很简单的思路就是第一我们只允许局域网内一部分的用户连接到我们的服务器. 第二个就是这些 client 不能修改我们服务器上的时间

关于权限设定部分

权限的设定主要以 restrict 这个参数来设定，主要的语法为：

restrict IP 地址 mask 子网掩码 参数

其中 IP 可以是 IP 地址，也可以是 default ，default 就是指所有的 IP

参数有以下几个：

ignore ：关闭所有的 **NTP** 联机服务

nomodify：客户端不能更改服务端的时间参数，但是客户端可以通过服务端进行网络校时。

notrust ：客户端除非通过认证，否则该客户端来源将被视为不信任子网

noquery ：不提供客户端的时间查询

更多内容：

让你真正明白 **Linux NTP**（包括历史、参数解释、安装配置）

http://www.aboutyun.com/thread-11395-1-1.html

openstack【juno】入门 【准备篇】三：mysql（MariaDB）安装

问题导读

1.MariaDB 与 mysql 的关系是什么？
2.遇到 Checking for corrupt, not cleanly closed and upgrade needing tables.该如何解决？

安装 mysql 之前首先安装 OpenStack 库

```
1.  # apt-get install ubuntu-cloud-keyring
2.  # echo "deb http://ubuntu-cloud.archive.canonical.com/ubuntu" \
3.    "trusty-updates/juno main" > /etc/apt/sources.list.d/cloudarchive-juno.list
```

更新

```
1.  apt-get update && apt-get dist-upgrade
```

，如果不安装 openstack 库，直接安装 keystone，会 keystone 能够安装成功，但是 keystone 启动后，接着就会失败。造成 keystone 为 unknown instance

为什么产生 MariaDB

首先这里介绍一下，大家对 MariaDB 可能不太熟悉，MariaDB 数据库管理系统是 MySQL 的一个分支，主要由开源社区在维护，采用 GPL 授权许可。开发这个分支的原因之一是：甲骨文公司收购了 MySQL 后，有将 MySQL 闭源的潜在风险，因此社区采用分支的方式来避开这个风险。

根据官网文档：

安装

```
1.  apt-get install mariadb-server python-mysqldb
```

修改配置文件

/etc/mysql/my.cnf

找到 bind-address = 127.0.0.1
修改为下面：

```
1.  [mysqld]
2.  ...
3.  bind-address = 10.0.0.11
```

然后在新增如下内容：

```
1.  [mysqld]
2.  ...
3.  default-storage-engine = innodb
4.  innodb_file_per_table
5.  collation-server = utf8_general_ci
6.  init-connect = 'SET NAMES utf8'
7.  character-set-server = utf8
```

注意：不要带上[mysqld]

重启 mysql

```
1.  service mysql restart
```

输出如下信息

```
* Stopping MariaDB database server
mysqld                                                     [ OK ]
* Starting MariaDB database server
mysqld                                                     [ OK ]
* Checking for corrupt, not cleanly closed and upgrade needing tables.
```

这个只是个提示，告诉你在做什么。不管它

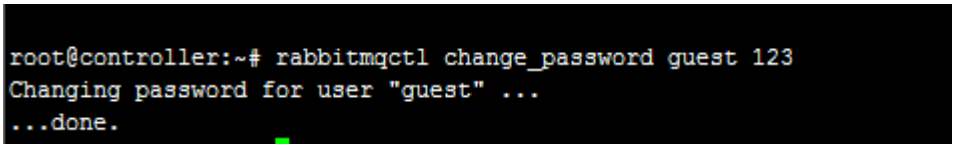
openstack【juno】入门 【准备篇】四：：RabbitMQ 安装

有的同学比较困惑 RabbitMQ 安装在什么节点：根据官网文档 RabbitMQ 安装在 controller 节点。
执行下面命令：

```
1.  apt-get install rabbitmq-server
```

修改密码为 123

```
1.  rabbitmqctl change_password guest  123
```



至此 openstack 安装准备工作完毕，开始安装

openstack【juno】入门 【keystone 篇】五：：Keystone 部署及介绍

我们安装部署 keystone，那么我们就需要了解 keystone 的作用，简单来讲，keystone 是 openstack 中验证组件。更多内容：
Keystone, Openstack 之魂

零基础学习 openstack【完整中级篇】及 openstack 资源汇总

下面我们开始安装和部署：

创建数据库，并授权

```
1.  mysql -u root -p
```

创建 keystone 数据库

```
1.  CREATE DATABASE keystone;
```

对 keystone 授权

```
1.  GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' \
2.    IDENTIFIED BY 'KEYSTONE_DBPASS';
```

```
1.  GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' \
2.    IDENTIFIED BY 'KEYSTONE_DBPASS';
```

KEYSTONE_DBPASS 可以自定义。
上面的含义：
实现了，对 keystone 用户实现了，本地和远程都可以访问

更多了解：openstack 外篇之认识 mysql 授权及一些操作

生成 token

```
1.  openssl rand -hex 10
```

我这里

```
1.  570f150cb897e793e58f
```

安装 keystone 包：

```
1.  apt-get install keystone python-keystoneclient
```

```
root@controller:~#  apt-get install keystone python-keystoneclient
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  gcc gcc-4.8 libasan0 libatomic1 libc-dev-bin libc6-dev libgcc-4.8-dev
  libgomp1 libitm1 libquadmath0 libtsan0 linux-libc-dev manpages-dev
  zlib1g-dev
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  alembic ieee-data libjs-jquery libjs-sphinxdoc libjs-underscore librabbitmq1
  libxslt1.1 libyaml-0-2 python-alembic python-amqp python-anyjson
  python-babel python-babel-localedata python-crypto python-decorator
  python-dns python-dogpile.cache python-dogpile.core python-eventlet
```

编辑 /etc/keystone/keystone.conf

```
1.  [DEFAULT]

2.  ...

3.  admin_token = ADMIN_TOKEN
```

这里修改如下

```
1.  admin_token =570f150cb897e793e58f
```

修改 [database]部分

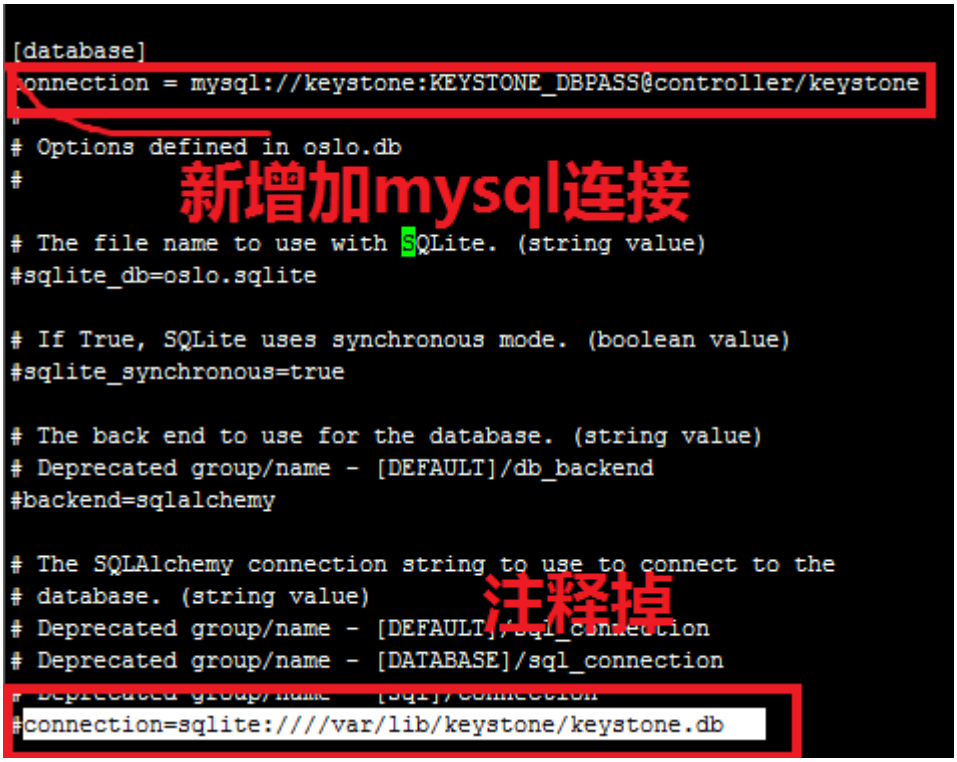
```
1.  [database]

2.  ...
```

```
3.  connection = mysql://keystone:KEYSTONE_DBPASS@controller/keystone
```

补充：

记得一定注释掉：

```
1.  connection=sqlite:////var/lib/keystone/keystone.db
```



修改 [token]部分

```
1.  [token]

2.  ...

3.  provider = keystone.token.providers.uuid.Provider

4.  driver = keystone.token.persistence.backends.sql.Token
```

修改[DEFAULT]部分

```
1.  [DEFAULT]

2.  ...

3.  verbose = True
```

填充 keystone

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

这里最好切换至 root 用户，否则会同步不成功。

同步成功，有如下信息：

```
root@controller:~# su -s /bin/sh -c "keystone-manage db_sync" keystone
2015-01-27 17:48:30.024 2376 INFO migrate.versioning.api [-] 33 -> 34...
2015-01-27 17:48:30.514 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.515 2376 INFO migrate.versioning.api [-] 34 -> 35...
2015-01-27 17:48:30.541 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.543 2376 INFO migrate.versioning.api [-] 35 -> 36...
2015-01-27 17:48:30.565 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.566 2376 INFO migrate.versioning.api [-] 36 -> 37...
2015-01-27 17:48:30.583 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.584 2376 INFO migrate.versioning.api [-] 37 -> 38...
2015-01-27 17:48:30.620 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.621 2376 INFO migrate.versioning.api [-] 38 -> 39...
2015-01-27 17:48:30.683 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.684 2376 INFO migrate.versioning.api [-] 39 -> 40...
2015-01-27 17:48:30.730 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.731 2376 INFO migrate.versioning.api [-] 40 -> 41...
2015-01-27 17:48:30.754 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.755 2376 INFO migrate.versioning.api [-] 41 -> 42...
2015-01-27 17:48:30.780 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.781 2376 INFO migrate.versioning.api [-] 42 -> 43...
2015-01-27 17:48:30.798 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.799 2376 INFO migrate.versioning.api [-] 43 -> 44...
2015-01-27 17:48:30.819 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.820 2376 INFO migrate.versioning.api [-] 44 -> 45...
2015-01-27 17:48:30.843 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.844 2376 INFO migrate.versioning.api [-] 45 -> 46...
2015-01-27 17:48:30.860 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.861 2376 INFO migrate.versioning.api [-] 46 -> 47...
2015-01-27 17:48:30.874 2376 INFO migrate.versioning.api [-] done
```

重启 keystone

```
1.  service keystone restart
```

删除 Ubuntu 包，创建的 SQLite 数据库

```
1.  rm -f /var/lib/keystone/keystone.db
```

为了效率设定定期清理过期 token

```
1.  # (crontab -l -u keystone 2>&1 | grep -q token_flush) || \
```

```
2.    echo '@hourly /usr/bin/keystone-manage
    token_flush >/var/log/keystone/keystone-tokenflush.log 2>&1' \
3.    >> /var/spool/cron/crontabs/keystone
```

openstack【juno】入门 【keystone 篇】六：：Keystone 使用及遇到问题解决办法

问题导读

1.keystone 的 ADMIN_TOKEN 该如何填写？
2.如何查看 keystone 数据库?
3.同步数据库遇到问题，可能有哪些原因？
4.keystone 创建 demo 租户，是否还需要创建 user 及 role？

创建租户、用户、角色，网上资料还是比较多的。
这里参考管网，使用另外的方式

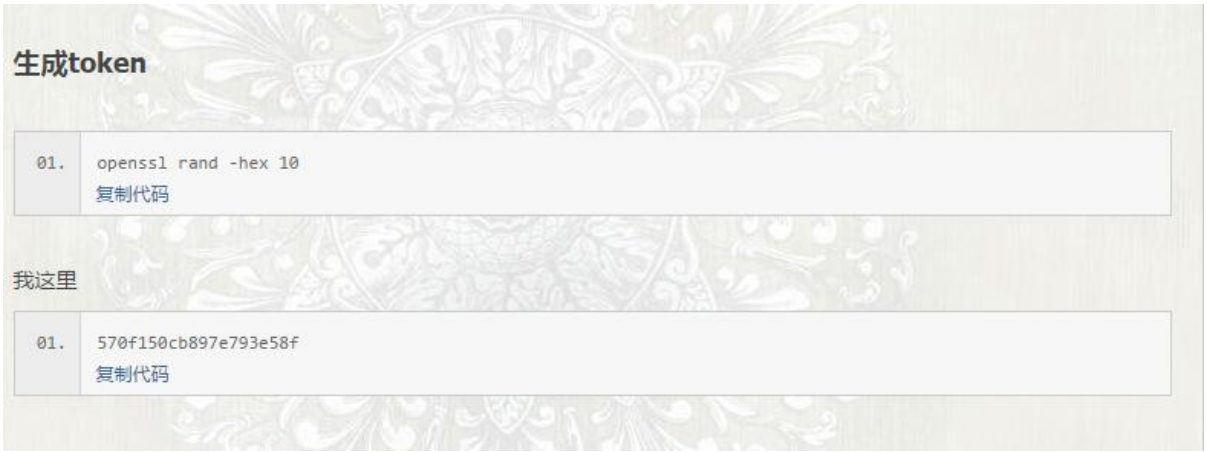
配置环境变量

配置 OS_SERVICE_TOKEN

这里的 ADMIN_TOKEN 是上篇产生的值

```
1.  export OS_SERVICE_TOKEN=ADMIN_TOKEN
```

openstack juno（ubuntu14.04）安装 5：Keystone 部署及介绍



上面替换为：

```
1.  export OS_SERVICE_TOKEN=570f150cb897e793e58f
```

配置 endpoint:

```
1.  export OS_SERVICE_ENDPOINT=http://controller:35357/v2.0
```

创建租户、用户、角色

```
1.  keystone tenant-create --name admin --description "Admin Tenant"
```

```
root@controller:~# keystone tenant-create --name admin --description "Admin Tenant"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |           Admin Tenant           |
|   enabled   |               True               |
|      id     | cc82e958254043529fd23eeaf06a5304 |
|     name    |              admin               |
+-------------+----------------------------------+
```

这里会遇到的问题

```
aboutyun@controller:~$ keystone tenant-create --name admin --description "admin Tenant"
An unexpected error prevented the server from fulfilling your request. (HTTP 500)
```

解决办法详细查看**遇到问题及解决办法**：

创建 admin 用户

```
1.  keystone user-create --name admin --pass ADMIN_PASS --email EMAIL_ADDRESS
```

```
root@controller:~# keystone user-create --name admin --pass ADMIN_PASS --email EMAIL_ADDRESS
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |          EMAIL_ADDRESS           |
| enabled  |               True               |
|    id    | f05d82730c234806868400777627ce4e |
|   name   |              admin               |
| username |              admin               |
+----------+----------------------------------+
```

创建 **admin** 角色

```
1.  keystone role-create --name admin
```

```
root@controller:~# keystone role-create --name admin
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|    id    | a967525a60414deb8f9e97330fe777de |
|   name   |              admin               |
+----------+----------------------------------+
```

创建 **Demo** 租户、用户、角色

```
1.  keystone tenant-create --name demo --description "Demo Tenant"
```

```
root@controller:~# keystone tenant-create --name demo --description "Demo Tenant"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |           Demo Tenant            |
|   enabled   |               True               |
|      id     | 9bc030e050c54a71ae72473ab8b08598 |
|     name    |              demo                |
+-------------+----------------------------------+
```

```
1.  keystone user-create --name demo --tenant demo --pass DEMO_PASS --email EMAIL_ADDRESS
```

```
root@controller:~# keystone user-create --name demo --tenant demo --pass DEMO_PASS --email EMAIL_ADDRESS
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |          EMAIL_ADDRESS           |
| enabled  |               True               |
|    id    | 51b14548af8b427eb1b4d90658e1b11d |
|   name   |               demo               |
| tenantId | 9bc030e050c54a71ae72473ab8b08598 |
| username |               demo               |
+----------+----------------------------------+
```

至此 demod 用户完毕注意：在创建 demo tenant 的同时，demo user,demo role 会自动创建

```
root@controller:~# keystone user-list
+----------------------------------+-------+---------+---------------+
|                id                |  name | enabled |     email     |
+----------------------------------+-------+---------+---------------+
| f05d82730c234806868400777627ce4e | admin |   True  | EMAIL_ADDRESS |
| 51b14548af8b427eb1b4d90658e1b11d |  demo |   True  | EMAIL_ADDRESS |
+----------------------------------+-------+---------+---------------+
root@controller:~# keystone tenant-list
+----------------------------------+-------+---------+
|                id                |  name | enabled |
+----------------------------------+-------+---------+
| cc82e958254043529fd23eeaf06a5304 | admin |   True  |
| 9bc030e050c54a71ae72473ab8b08598 |  demo |   True  |
+----------------------------------+-------+---------+
root@controller:~# keystone role-list
+----------------------------------+----------+
|                id                |   name   |
+----------------------------------+----------+
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_ |
| a967525a60414deb8f9e97330fe777de |  admin   |
+----------------------------------+----------+
```

🍒 🍒 🍒 🍒 🍒 🍒 🍒 🍒 🍒 🍒 🍒 🍒 🍒 🍒 🍒 🍒

遇到问题及解决办法：

问题 1

aboutyun@controller:~$ keystone tenant-create --name admin --description "admin Tenant"

An unexpected error prevented the server from fulfilling your request. (HTTP 500)

原因 1：

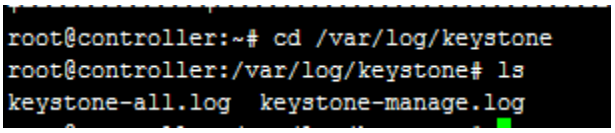
环境变量错误

记得重启后，执行下面命令

```
1.  export OS_SERVICE_TOKEN=570f150cb897e793e58f

2.  export OS_SERVICE_ENDPOINT=http://controller:35357/v2.0
```

原因 2：

查看日志：




```
1.  more keystone-all.log
```

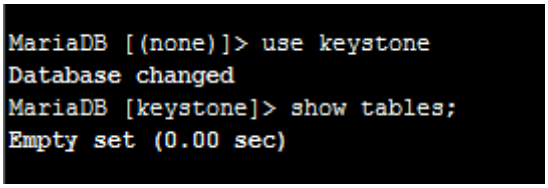
获取如下关键信息

```
1.  (OperationalError) no such table: project u'INSERT INTO project (id, name, domain_id, des

2.  cription, enabled, extra) VALUES (?, ?, ?, ?, ?, ?)' ('70c0487ba99743719d6721a34560fca2',

    u'admin', 'default', u'Admin Tenant', 1, '{}')
```

意思是说表不存在，进入 mysql

```
1.  mysql -uroot -p
```

```
1.  use keystone
```




```
1.  show tables;
```

竟然是空的，也就是没有同步成功。

执行命令，再次同步。

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

还需要输入密码，算了还是使用 root。

再次运行命令，还是不成功。

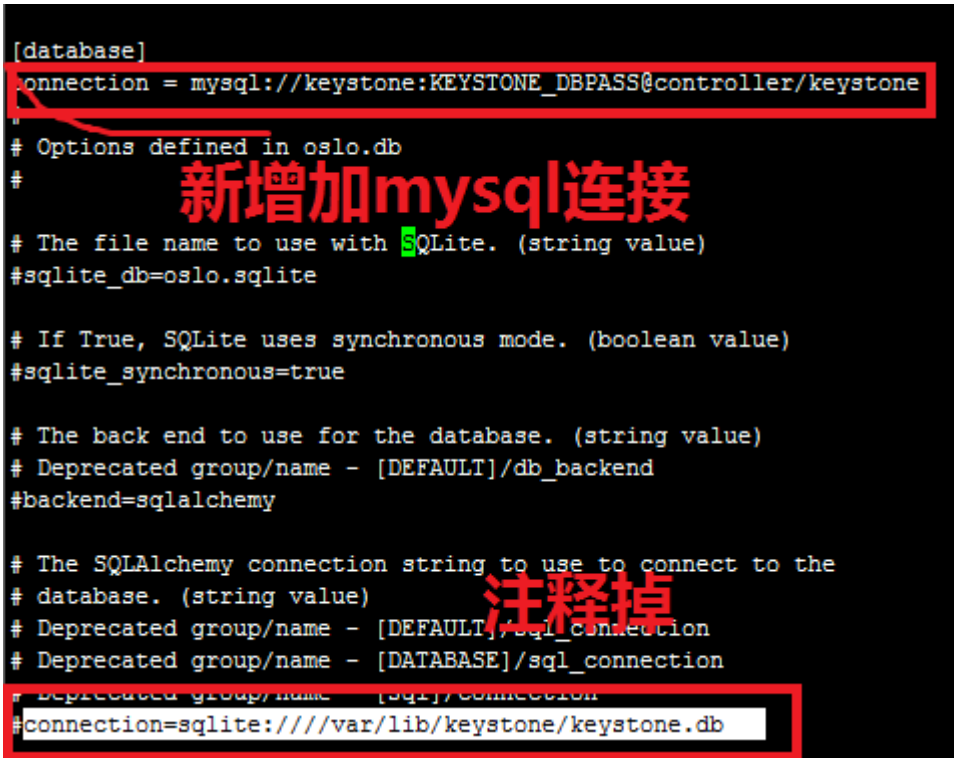
为什么那？还是从配置入手。

编辑 /etc/keystone/keystone.conf

数据库明明连接上了，为什么会不成功，

```
1.  [database]

2.  ...

3.  connection = mysql://keystone:KEYSTONE_DBPASS@controller/keystone
```

后来在配置文件中，发现了 sqllite，注释掉即可

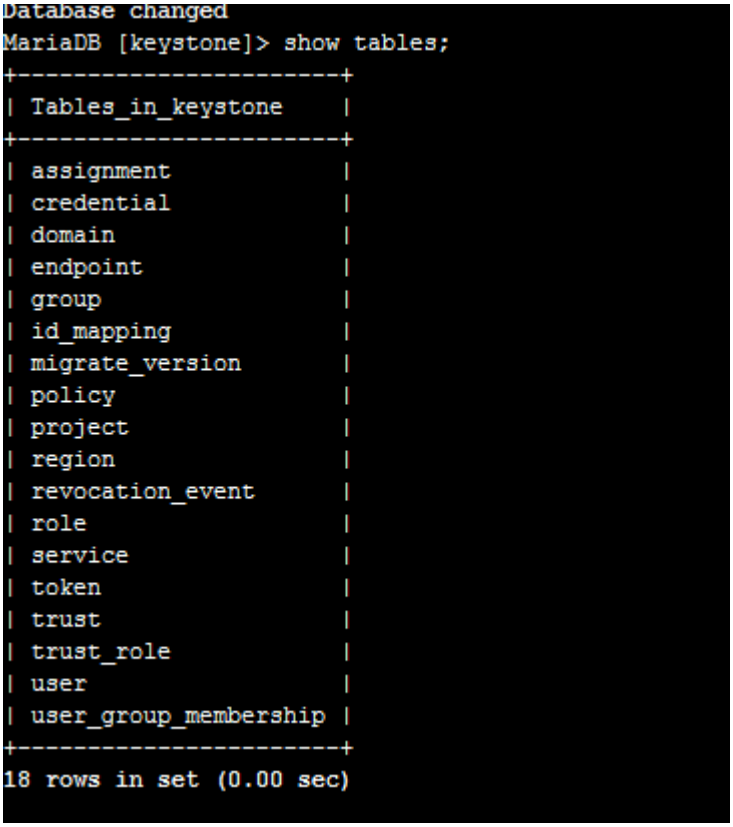


再次运行命令

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

```
root@controller:~# su -s /bin/sh -c "keystone-manage db_sync" keystone
2015-01-27 17:48:30.024 2376 INFO migrate.versioning.api [-] 33 -> 34...
2015-01-27 17:48:30.514 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.515 2376 INFO migrate.versioning.api [-] 34 -> 35...
2015-01-27 17:48:30.541 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.543 2376 INFO migrate.versioning.api [-] 35 -> 36...
2015-01-27 17:48:30.565 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.566 2376 INFO migrate.versioning.api [-] 36 -> 37...
2015-01-27 17:48:30.583 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.584 2376 INFO migrate.versioning.api [-] 37 -> 38...
2015-01-27 17:48:30.620 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.621 2376 INFO migrate.versioning.api [-] 38 -> 39...
2015-01-27 17:48:30.683 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.684 2376 INFO migrate.versioning.api [-] 39 -> 40...
2015-01-27 17:48:30.730 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.731 2376 INFO migrate.versioning.api [-] 40 -> 41...
2015-01-27 17:48:30.754 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.755 2376 INFO migrate.versioning.api [-] 41 -> 42...
2015-01-27 17:48:30.780 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.781 2376 INFO migrate.versioning.api [-] 42 -> 43...
2015-01-27 17:48:30.798 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.799 2376 INFO migrate.versioning.api [-] 43 -> 44...
2015-01-27 17:48:30.819 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.820 2376 INFO migrate.versioning.api [-] 44 -> 45...
2015-01-27 17:48:30.843 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.844 2376 INFO migrate.versioning.api [-] 45 -> 46...
2015-01-27 17:48:30.860 2376 INFO migrate.versioning.api [-] done
2015-01-27 17:48:30.861 2376 INFO migrate.versioning.api [-] 46 -> 47...
2015-01-27 17:48:30.874 2376 INFO migrate.versioning.api [-] done
```

同步成功，表不在为空



```
Database changed
MariaDB [keystone]> show tables;
+-----------------------+
| Tables_in_keystone    |
+-----------------------+
| assignment            |
| credential            |
| domain                |
| endpoint              |
| group                 |
| id_mapping            |
| migrate_version       |
| policy                |
| project               |
| region                |
| revocation_event      |
| role                  |
| service               |
| token                 |
| trust                 |
| trust_role            |
| user                  |
| user_group_membership |
+-----------------------+
18 rows in set (0.00 sec)
```

openstack【juno】入门 【keystone 篇】七：创建 service entity 和 API endpoint

问题导读

1.如何查看创建的服务？
2.通过哪个命令可以删除服务?
3.看图 API endpoints 中包含了哪些？

环境变量

在配置前，首先配置好环境变量

创建 service entity and API endpoints

```
1.  keystone service-create --name keystone --type identity \
2.    --description "OpenStack Identity"
```

```
root@controller:~# keystone service-create --name keystone --type identity \
>   --description "OpenStack Identity"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |        OpenStack Identity        |
|   enabled   |               True               |
|      id     | 97ad71e0a9994f4aa7303d09d2c01369 |
|     name    |             keystone             |
|     type    |             identity             |
+-------------+----------------------------------+
```

补充：

查看创建服务

```
keystone service-list
```

如果想删除则使用命令：

```
keystone service-delete id
```

****

创建 API endpoints

```
keystone endpoint-create \
  --service-id $(keystone service-list | awk '/ identity / {print $2}') \
  --publicurl http://controller:5000/v2.0 \
  --internalurl http://controller:5000/v2.0 \
  --adminurl http://controller:35357/v2.0 \
  --region regionOne
```

```
root@controller:~# keystone service-create --name keystone --type identity \
>   --description "OpenStack Identity"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |        OpenStack Identity        |
|   enabled   |               True               |
|      id     | 97ad71e0a9994f4aa7303d09d2c01369 |
|     name    |             keystone             |
|     type    |             identity             |
+-------------+----------------------------------+
root@controller:~# keystone endpoint-create \
>   --service-id $(keystone service-list | awk '/ identity / {print $2}') \
>   --publicurl http://controller:5000/v2.0 \
>   --internalurl http://controller:5000/v2.0 \
>   --adminurl http://controller:35357/v2.0 \
>   --region regionOne
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  |   http://controller:35357/v2.0   |
|      id     | 0be919feee754877aae876e8b512fed1 |
| internalurl |   http://controller:5000/v2.0    |
|  publicurl  |   http://controller:5000/v2.0    |
|    region   |            regionOne             |
|  service_id | 97ad71e0a9994f4aa7303d09d2c01369 |
+-------------+----------------------------------+
```

注意：

service_id | 97ad71e0a9994f4aa7303d09d2c01369 |是前面创建 service 所产生的 id

openstack【juno】入门 【keystone 篇】八：新手操作篇（验证操作篇）

问题导读

1.openstack 的环境变量的作用是什么？
2.openstack 不配置环境变量如何执行命令？
3.非管理员租户是否具有查看 user 的权限？

接上一篇
openstack juno（ubuntu14.04）安装 7：创建 service entity 和 API endpoint

很多新手，对于环境变量经常不理解，而且对于 openstack 的 dasboard 中用户名和密码也很模糊，如果刚接触 openstack，或许对于我所说的也是一知半解，这里只需要记住。openstack 的环境变量，也是 dasboard 登录的用户名和密码。

在 Linux 操作中，环境变量的配置主要是方便操作，同样 openstack 也是这样的.所以这里去掉环境变量的配置，让新手能够弄明白他们之间的区别。

首先，我们看看，环境变来那个，我们是如何配置的

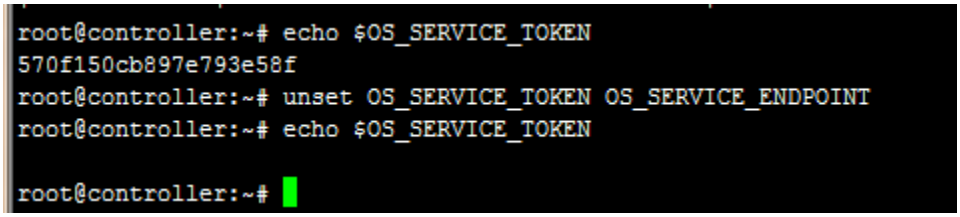
```
1.  export OS_SERVICE_TOKEN=570f150cb897e793e58f
```

```
1.  export OS_SERVICE_ENDPOINT=http://controller:35357/v2.0
```

去除环境变变量

下面，我们来去掉环境变量的配置

```
1.  unset OS_SERVICE_TOKEN OS_SERVICE_ENDPOINT
```



其它验证操作

admin 租户获取 token:

```
keystone --os-tenant-name admin --os-username admin --os-password ADMIN_PASS \
  --os-auth-url http://controller:35357/v2.0 token-get
```

这里需要注意的是 ADMIN_PASS 是我们创建租户的时候，设置的密码，我们在创建租户的时候，使用的是默认密码

```
+-----------+----------------------------------+
|  Property |              Value               |
+-----------+----------------------------------+
|  expires  |       2015-01-28T11:59:01Z       |
|     id    | 4e0c4d02d693454a9f59bc3d1152d2aa |
| tenant_id | cc82e958254043529fd23eeaf06a5304 |
|  user_id  | f05d82730c234806868400777627ce4e |
+-----------+----------------------------------+
```

列出租户(tenant)

```
keystone --os-tenant-name admin --os-username admin --os-password ADMIN_PASS \
  --os-auth-url http://controller:35357/v2.0 tenant-list
```

```
root@controller:~# keystone --os-tenant-name admin --os-username admin --os-password ADMIN_PASS \
>   --os-auth-url http://controller:35357/v2.0 tenant-list
+----------------------------------+---------+---------+
|                id                |   name  | enabled |
+----------------------------------+---------+---------+
| cc82e958254043529fd23eeaf06a5304 |  admin  |   True  |
| 9bc030e050c54a71ae72473ab8b08598 |   demo  |   True  |
| d27e6063531a467a8fbdd81168995c40 | service |   True  |
+----------------------------------+---------+---------+
```

列出用户（**user**）

```
1.  keystone --os-tenant-name admin --os-username admin --os-password ADMIN_PASS \
2.  >   --os-auth-url http://controller:35357/v2.0 user-list
```

```
root@controller:~# keystone --os-tenant-name admin --os-username admin --os-password ADMIN_PASS \
>   --os-auth-url http://controller:35357/v2.0 user-list
+----------------------------------+-------+---------+---------------+
|                id                |  name | enabled |     email     |
+----------------------------------+-------+---------+---------------+
| f05d82730c234806868400777627ce4e | admin |   True  | EMAIL_ADDRESS |
| 51b14548af8b427eb1b4d90658e1b11d |  demo |   True  | EMAIL_ADDRESS |
+----------------------------------+-------+---------+---------------+
```

列出角色（**role**）

```
1.  keystone --os-tenant-name admin --os-username admin --os-password ADMIN_PASS \
2.    --os-auth-url http://controller:35357/v2.0 role-list
```

```
root@controller:~# keystone --os-tenant-name admin --os-username admin --os-password ADMIN_PASS \
> --os-auth-url http://controller:35357/v2.0 role-list
+----------------------------------+----------+
|                id                |   name   |
+----------------------------------+----------+
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_ |
| a967525a60414deb8f9e97330fe777de |  admin   |
+----------------------------------+----------+
```

demo 租户获取令牌 token

```
1.  keystone --os-tenant-name demo --os-username demo --os-password DEMO_PASS \
2.  >   --os-auth-url http://controller:35357/v2.0 token-get
```

```
root@controller:~# keystone --os-tenant-name demo --os-username demo --os-password DEMO_PASS \
>   --os-auth-url http://controller:35357/v2.0 token-get
+-----------+----------------------------------+
|  Property |              Value               |
+-----------+----------------------------------+
|  expires  |       2015-01-28T12:10:51Z       |
|     id    | 32b4c008bd0b4973b9a1efd9f24c1110 |
| tenant_id | 9bc030e050c54a71ae72473ab8b08598 |
|  user_id  | 51b14548af8b427eb1b4d90658e1b11d |
+-----------+----------------------------------+
```

demo 用户无权查看 user，会报错误如下：

```
1.  keystone --os-tenant-name demo --os-username demo --os-password DEMO_PASS \
2.    --os-auth-url http://controller:35357/v2.0 user-list
```

You are not authorized to perform the requested action: admin_required (HTTP 403)

```
root@controller:~# keystone --os-tenant-name demo --os-username demo --os-password DEMO_PASS \
>   --os-auth-url http://controller:35357/v2.0 user-list
You are not authorized to perform the requested action: admin_required (HTTP 403)
root@controller:~# ^C
root@controller:~#
```

openstack【juno】入门【keystone 篇】九：创建 openstack 客户端环境变量脚本

问题导读

1.环境变量一般为什么格式？
2.不同用户环境变量端口是否一致？
3.openstack 环境变量都需要配置哪些信息？
4.如何才能保证配置信息生效？

接上一篇：
openstack juno（ubuntu14.04）安装 7.1：新手操作篇（验证操作篇）

创建 admin、demo 环境变量
openstack 支持 OpenRC 脚本文件

创建 admin 环境变量
编辑 admin-openrc.sh

把这个脚本放到自己想放的位置

```
1.  sudo nano  admin-openrc.sh
```

```
1.  export OS_TENANT_NAME=admin

2.  export OS_USERNAME=admin

3.  export OS_PASSWORD=ADMIN_PASS

4.  export OS_AUTH_URL=http://controller:35357/v2.0
```

生效

```
1.  source admin-openrc.sh
```

创建 demo 环境变量

编辑 demo-openrc.sh

```
1.  export OS_TENANT_NAME=demo
2.  export OS_USERNAME=demo
3.  export OS_PASSWORD=DEMO_PASS
4.  export OS_AUTH_URL=http://controller:5000/v2.0
```

生效

```
1.  source demo-openrc.sh
```

注意
上面两个环境两的端口是不一样的。

端口 35357 是管理员端口
端口 5000 是普通用户的功能，是最常用的

openstack【juno】入门 【glance 篇】十：glance 初步介绍

问题导读

1.glance 包含哪些组件？
2.glance-api、glance-registry 有什么区别？
3.图像文件存储库支持哪些格式？

OpenStack 镜像服务

openstack 是 Iaas 的核心，它接受用户和 opentack 组件请求磁盘或则镜像服务和镜像元数据。它支持多种多种存储类型，包括 openstack 对象存储（ Object Storage）。

运行在 OpenStack Image Service 的进程支持缓存。通过集群的备份确保一致性和可用性。其它周期性进程包括：
auditors, updaters, and reapers。
openstack glance（ Image Service ）包含下面组件

glance-api

接受 api 请求，并提供相应操作，包括显示，检索，存储

glance-registry
存储、处理、检索镜像的元数据，元数据包括例如镜像大小、类型等
注意：
glance-registry 是 OpenStack Image Service 内部服务，不对用户开放

数据库

可以选择自己喜欢的数据库存储镜像元数据，大多数使用 MySQL 或则 SQLite.

图像文件存储库

不同类型的库支持包括文件系统、对象存储、RADOS block devices、HTTP, 和 Amazon S3

注意：
一些库只支持只读

openstack【juno】入门 【glance 篇】十一：glance 安装配置

问题导读

1.如何创建 glance 用户？
2.安装 glance，需要做哪些准备？
3.如何验证 glance 数据库同步成功？
4.修改配置文件的过程中，有哪些需要注意的地方？
5.nano 编辑器如何使用？

接上一篇
openstack【juno】入门 【glance 篇】十：glance 初步介绍

这一篇描述了 glance 安装与配置，安装在控制节点，为了简单起见，配置存储镜像在本地文件系统

注意：1.这里使用的 nano 编辑器，对于这个编辑器很方便，如果想查找某个内容，使用 Ctrl+W 即可。更详细参考
Ubuntu 使用 nano 文本编辑器
2.在修改配置文件过程中，切记标记重复，例如[database]、[defaule]等

配置前准备

1.创建数据库
2.服务认证
3.API endpoints

1.创建数据库

a.进入数据库，并输入密码：

```
1.  mysql -u root -p
```

```
aboutyun@controller:~$ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 37
Server version: 5.5.40-MariaDB-0ubuntu0.14.04.1 (Ubuntu)

Copyright (c) 2000, 2014, Oracle, Monty Program Ab and others.

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

MariaDB [(none)]>
```

b.创建 glance 数据库

```
1.  CREATE DATABASE glance;
```

```
MariaDB [(none)]> CREATE DATABASE glance;
Query OK, 1 row affected (0.05 sec)
```

c.授权 glance 数据库，使得本地及远程都能访问

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

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

（GLANCE_DBPASS 上面密码可以自定义）

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \
    ->   IDENTIFIED BY 'GLANCE_DBPASS';
Query OK, 0 rows affected (0.17 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' \
    ->   IDENTIFIED BY 'GLANCE_DBPASS';
Query OK, 0 rows affected (0.01 sec)
```

d.退出 mysql

```
MariaDB [(none)]> exit
Bye
aboutyun@controller:~$
```

2.环境变量生效

```
1.  source admin-openrc.sh
```

如果这里不明白，可以查看此篇 openstack【juno】入门 【keystone 篇】九：创建 openstack 客户端环境变量脚本

3.创建 glance（凭据）用户

a.创建 glance 用户

```
1.  keystone user-create --name glance --pass GLANCE_PASS
```

```
aboutyun@controller:~$ keystone user-create --name glance --pass GLANCE_PASS
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |                                  |
| enabled  |               True               |
|    id    | 326758e13daf42ad84b4cf7d3d440077 |
|   name   |              glance              |
| username |              glance              |
+----------+----------------------------------+
```

b.给 glance 用户授予 admin role（角色）

```
1.  keystone user-role-add --user glance --tenant service --role admin
```

注意这条命令没有任何输出

```
aboutyun@controller:~$ keystone user-role-add --user glance --tenant service --role admin
aboutyun@controller:~$
```

c.创建 glance service 实例:

```
1.  keystone service-create --name glance --type image \
2.    --description "OpenStack Image Service"
```

```
aboutyun@controller:~$ keystone service-create --name glance --type image \
>   --description "OpenStack Image Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |     OpenStack Image Service      |
|   enabled   |               True               |
|      id     | 770c3ceda361482c9d1e57ab6367235b |
|     name    |              glance              |
|     type    |              image               |
+-------------+----------------------------------+
```

4.创建 Image 服务 API endpoints:

```
keystone endpoint-create \
  --service-id $(keystone service-list | awk '/ image / {print $2}') \
  --publicurl http://controller:9292 \
  --internalurl http://controller:9292 \
  --adminurl http://controller:9292 \
  --region regionOne
```

```
aboutyun@controller:~$ keystone endpoint-create \
>   --service-id $(keystone service-list | awk '/ image / {print $2}') \
>   --publicurl http://controller:9292 \
>   --internalurl http://controller:9292 \
>   --adminurl http://controller:9292 \
>   --region regionOne
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  |      http://controller:9292      |
|      id     | d32d8dbd11aa40bd89790fe944714371 |
| internalurl |      http://controller:9292      |
|  publicurl  |      http://controller:9292      |
|    region   |            regionOne             |
|  service_id | 770c3ceda361482c9d1e57ab6367235b |
+-------------+----------------------------------+
```

安装配置镜像服务组件

1.安装

```
1.  apt-get install glance python-glanceclient
```

```
aboutyun@controller:~$ apt-get install glance python-glanceclient
E: Could not open lock file /var/lib/dpkg/lock - open (13: Permission denied)
E: Unable to lock the administration directory (/var/lib/dpkg/), are you root?
aboutyun@controller:~$ su root
Password:
root@controller:/home/aboutyun# cd
root@controller:~# su aboutyun
aboutyun@controller:/root$ 123
123: command not found
aboutyun@controller:/root$ cd
aboutyun@controller:~$ sudo apt-get install glance python-glanceclient
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  gcc gcc-4.8 libasan0 libatomic1 libc-dev-bin libc6-dev libgcc-4.8-dev
  libgomp1 libitm1 libquadmath0 libtsan0 linux-libc-dev manpages-dev
  zlib1g-dev
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  glance-api glance-common glance-registry python-boto python-cinderclient
```

2.修改配置文件

/etc/glance/glance-api.conf

```
1.  sudo nano  /etc/glance/glance-api.conf
```

a.修改数据库连接（直接添加即可）

```
1.  [database]

2.  ...

3.  connection = mysql://glance:GLANCE_DBPASS@controller/glance
```

说明：如果想查找某个内容，比如[database]，使用 nano 快捷键即可查找

b.在 [keystone_authtoken] 和 [paste_deploy] 部分，修改配置:

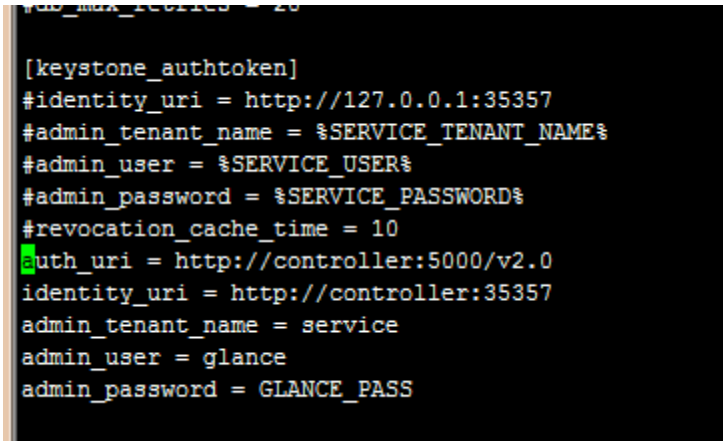
添加如下内容，切记不要 keystone_authtoken 标记重复

```
1.  [keystone_authtoken]

2.  ...

3.  auth_uri = http://controller:5000/v2.0

4.  identity_uri = http://controller:35357
```

```
admin_tenant_name = service
admin_user = glance
admin_password = GLANCE_PASS
```

注释掉下面内容

```
#identity_uri = http://127.0.0.1:35357
#admin_tenant_name = %SERVICE_TENANT_NAME%
#admin_user = %SERVICE_USER%
#admin_password = %SERVICE_PASSWORD%
#revocation_cache_time = 10
```



修改 flavor

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

c.修改 [glance_store] 部分，配置本地文件存储及存储路径

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

```
# ============ Filesystem Store Options ========================

# Directory that the Filesystem backend store
# writes image data to
default_store = file
filesystem_store_datadir = /var/lib/glance/images/
```

d.可选，帮助排除定位错误，使日志记录在[DEFAULT]部分

```
1.  [DEFAULT]

2.  ...

3.  verbose = True
```

```
[DEFAULT]
# Show more verbose log output (sets INFO log level output)
#verbose = False
verbose = True
# Show debugging output in logs (sets DEBUG log level output)
#debug = False
```

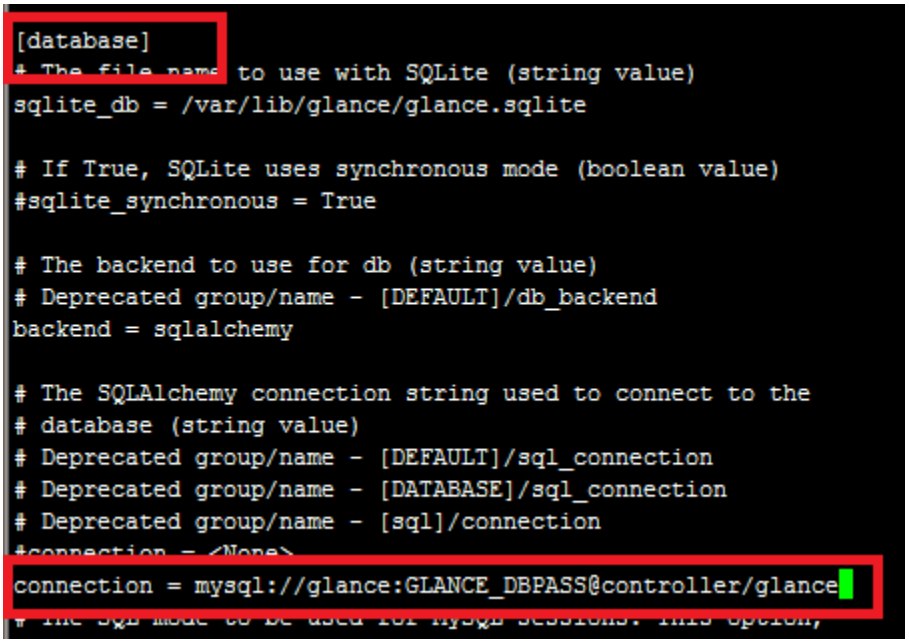
3.修改配置文件

a.修改 /etc/glance/glance-registry.conf 文件

```
1.  sudo nano  /etc/glance/glance-registry.conf
```

添加如下内容

```
1.  [database]

2.  ...

3.  connection = mysql://glance:GLANCE_DBPASS@controller/glance
```

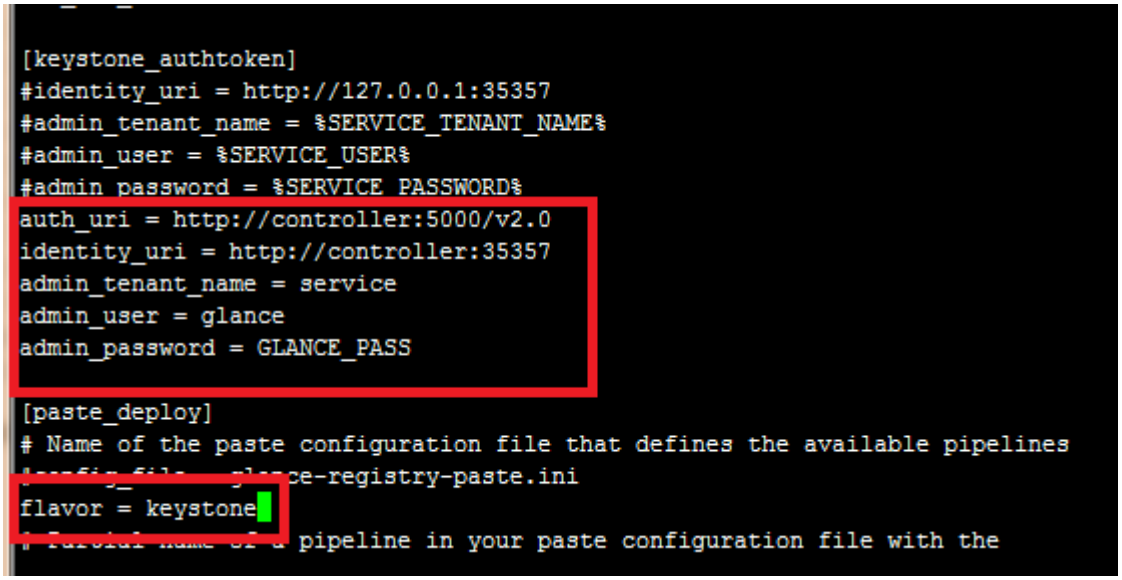


b.在 [keystone_authtoken] 和 [paste_deploy] 部分，添加如下内容：

```
1.  [keystone_authtoken]
2.  ...
3.  auth_uri = http://controller:5000/v2.0
4.  identity_uri = http://controller:35357
5.  admin_tenant_name = service
6.  admin_user = glance
7.  admin_password = GLANCE_PASS
8.
9.  [paste_deploy]
10. ...
11. flavor = keystone
```

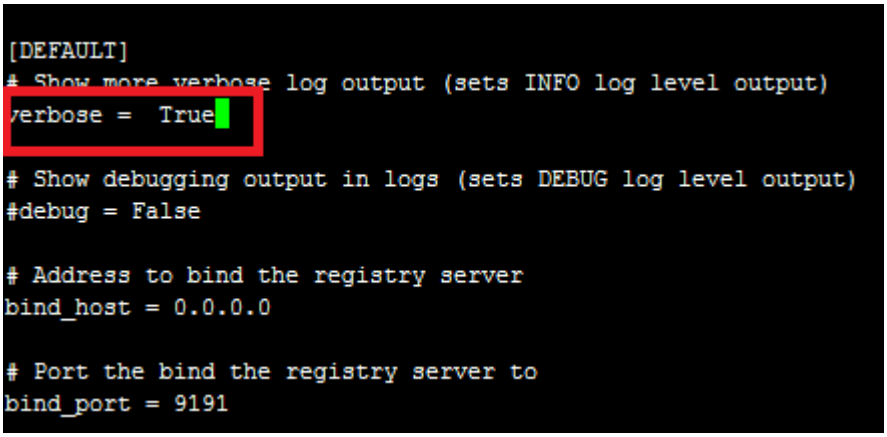
记得注释掉如下部分：

```
1.  #identity_uri = http://127.0.0.1:35357
2.  #admin_tenant_name = %SERVICE_TENANT_NAME%
3.  #admin_user = %SERVICE_USER%
4.  #admin_password = %SERVICE_PASSWORD%
```



c.可选，帮助排除定位错误，使日志记录在[DEFAULT]部分

```
1.  [DEFAULT]
2.  ...
3.  verbose = True
```



4.填充 the Image Service 数据库:

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

这个如果输入密码不生效，最好切换至 root 用户来执行。

进入 mysql

```
1.  mysql -uroot -p
```

```
1.  use glance
```

```
1.  show tables;
```

展示如下：同步成功

```
MariaDB [glance]> show tables;
+----------------------------------+
| Tables_in_glance                 |
+----------------------------------+
| image_locations                  |
| image_members                    |
| image_properties                 |
| image_tags                       |
| images                           |
| metadef_namespace_resource_types |
| metadef_namespaces               |
| metadef_objects                  |
| metadef_properties               |
| metadef_resource_types           |
| migrate_version                  |
| task_info                        |
| tasks                            |
+----------------------------------+
13 rows in set (0.00 sec)
```

完成安装

1.重启服务

```
1.  service glance-registry restart
```

```
1.  service glance-api restart
```

```
root@controller:~# service glance-registry restart
glance-registry stop/waiting
glance-registry start/running, process 19884
root@controller:~# service glance-api restart
glance-api stop/waiting
glance-api start/running, process 19900
```

2.可能 ubuntu 有安装的 SQLite 数据库.如果有的话，可以使用下面命令移除

```
1.  rm -f /var/lib/glance/glance.sqlite
```

openstack【juno】入门 【glance 篇】十二：glance 安装配置验证及相关操作

问题导读：

1.如何上传镜像

2.如何判断环境变量是否生效？

接上一篇：

openstack【juno】入门 【glance 篇】十：glance 安装配置

这一篇，主要使用 CirrOS 验证 glance 是否安装成功。CirrOS 是一个小的 Linux 镜像，帮助测试 openstack 部署

这一篇不是很顺利，遇到了一个 openstack ubuntu 的一个 bug

1.创建一个临时本地目录

```
1.  mkdir /tmp/images

1.  cd /tmp/images
```

2.下载镜像到本地目录

```
1.  wget http://cdn.download.cirros-cloud.net/0.3.3/cirros-0.3.3-x86_64-disk.img
```

3.环境变量生效

```
1.  source admin-openrc.sh
```

（需要找到 admin-openrc.sh）

4.上传镜像

```
1.  glance image-create --name "cirros-0.3.3-x86_64" --file cirros-0.3.3-x86_64-disk.img \
```

```
2.    --disk-format qcow2 --container-format bare --is-public True --progress
```

5.确认上传成功，并且可用

```
1.  glance image-list
```

6.移除本地临时目录（选操作，当然自己可以保留着）

```
1.  rm -r /tmp/images
```

遇到问题：

```
1.  glance image-create --name "cirros-0.3.3-x86_64" --file cirros-0.3.3-x86_64-disk.img \
2.    --disk-format qcow2 --container-format bare --is-public True --progress
```

global name '_' is not defined glance

原来环境变量的问题。source 已经 source 了，但是却没有起作用。最后使用绝招

```
1.  export OS_TENANT_NAME=admin
2.  export OS_USERNAME=admin
3.  export OS_PASSWORD=ADMIN_PASS
4.  export OS_AUTH_URL=http://controller:35357/v2.0
```

在 shell 中直接执行 shell。然后

通过

```
1.  echo $OS_TENANT_NAME
```

检验能否安装成功。

openstack【juno】入门 【nova 篇】十三（1）：nova 简单介绍

问题导读

1.openstack nova 的作用是什么？
2.你了解 nova 包含哪些组件？
扩展：
3.了解 nova api 及组件对使用 openstack 有什么作用？

相关内容：
Nova 各个组件介绍以及功能分析(逻辑架构，运行架构，开发架构以及数据库)

OpenStack Compute

使用 OpenStack Compute 管理云计算操作系统， OpenStack Compute 是 iaas 系统的主要部分。这一部分是用 python 来实现的。

OpenStackCompute 包括以下几个方面及其组件：

原文地址

API

nova-api 服务

接受并响应最终用户 compute API 调用。服务支持 OpenStack Compute API, the Amazon EC2 API, 和 Admin API 执行的权限. 它执行一些策略和一些初始化操作，比如运行一个实例。

nova-api-metadata 服务

接受实例元数据请求， nova-api-metadata 服务一般使用在安装 nova-network 多节点。更多细节查看 Metadata service

在 Debian 系统，它被包含在 nova-api 包中，可以通过 debconf 选择。

Compute core

nova-compute 服务

一个工作虚拟机实例进程，通过 hypervisor APIs 创建和终止。例如

- XenAPI for XenServer/XCP
- libvirt for KVM or QEMU
- VMwareAPI for VMware

处理是相当复杂的，最基本的，守护进程从队列和一系列系统命令操作，比如创建 KVM 实例，更新数据库状态

nova-scheduler 服务

决定实例运行在那个节点上

nova-conductor 模块

nova-conductor 在 nova-compute 服务 和 the database 之间，它使 nova-compute 服务无需直接访问云数据库（cloud database）。尽管如此，不要将
nova-conductor 部署在运行 nova-compute 服务的节点上

Networking for VMs

nova-network 工作进程

类似 nova-compute 服务，从队列中接受网络任务，执行任务，如设置网桥、改变防火墙规则
控制台界面，还包含下面进程

nova-consoleauth daemon

nova-consoleauth daemon
nova-novncproxy daemon
nova-spicehtml5proxy daemon
nova-xvpnvncproxy daemon
nova-cert daemon

Image management (EC2 scenario)
nova-objectstore daemon
euca2ools client
详细内容查看

命令行客户端和其它接口

nova client

作为管理员或则最终用户提交命令

其它组件

队列
进程之间通信，通常由 RabbitMQ，但可以用一个 AMQP 消息队列实现的，如 Apache Qpid orZero MQ

SQL 数据库
存储云基础设施的状态，包括

- 可用的类型。
- 中使用的实例。
- 可用网络
- 项目

从理论上来说， OpenStack Compute 支持任何数据库。通常 SQLite3 用来测试，开发如 MySQL, 和 PostgreSQL.

openstack【juno】入门 【nova 篇】十三（2）：安装配置计算服务

问题导读

1.计算节点、控制节点管理网络 ip 地址分别为什么？
2.如何判断是否支持虚拟机硬件加速按？
3.创建实例之前，需要注意什么问题？

准备

1.创建数据库，完成下面内容

a.登录 mysql

```
1.  mysql -u root -p
```

b.创建 nova 数据库

```
1.  CREATE DATABASE nova;
```

c.授权

```
1.  GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
2.    IDENTIFIED BY 'NOVA_DBPASS';
3.  GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
4.    IDENTIFIED BY 'NOVA_DBPASS';
```

d.退出 mysql

```
1.  exit
```

2.生效环境变量

```
1.  source admin-openrc.sh
```

内容如下：

```
1.  export OS_TENANT_NAME=admin

2.  export OS_USERNAME=admin

3.  export OS_PASSWORD=ADMIN_PASS

4.  export OS_AUTH_URL=http://controller:35357/v2.0
```

3.创建服务认证

a.创建一个 nova 用户

```
1.  keystone user-create --name nova --pass NOVA_PASS
```

这里使用默认密码，如果更改，在配置文件中需要做相应的修改。

b.给 nova 用户授予 admin 角色

```
1.  keystone user-role-add --user nova --tenant service --role admin
```

c.创建 nova 服务实例

```
1.  keystone service-create --name nova --type compute \

2.    --description "OpenStack Compute"
```

4.创建 nova 服务 API endpoints

```
1.  keystone endpoint-create \

2.    --service-id $(keystone service-list | awk '/ compute / {print $2}') \

3.    --publicurl http://controller:8774/v2/%\(tenant_id\)s \

4.    --internalurl http://controller:8774/v2/%\(tenant_id\)s \

5.    --adminurl http://controller:8774/v2/%\(tenant_id\)s \

6.    --region regionOne
```

安装配置计算控制组件

1.安装包

```
1.  apt-get install nova-api nova-cert nova-conductor nova-consoleauth \
2.    nova-novncproxy nova-scheduler python-novaclient
```

2.编辑文件，完成下面内容

```
1.  sudo nano  /etc/nova/nova.conf
```

a.在[database]部分，配置数据库连接

```
1.  [database]
2.  ...
3.  connection = mysql://nova:NOVA_DBPASS@controller/nova
```

b.在 [DEFAULT]部分，配置 RabbitMQ 消息代理访问

```
1.  [DEFAULT]
2.  ...
3.  rpc_backend = rabbit
4.  rabbit_host = controller
5.  rabbit_password = RABBIT_PASS
```

c. 在[DEFAULT] 和 [keystone_authtoken]部分，配置认证访问

```
1.  [DEFAULT]
2.  ...
3.  auth_strategy = keystone
4.
```

```
[keystone_authtoken]
...
auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357
admin_tenant_name = service
admin_user = nova
admin_password = NOVA_PASS
```

注释掉 auth_host, auth_port, 和 auth_protocol，因为 identity_uri 已经含有这些配置。

d.在 [DEFAULT]部分，配置控制节点，管理网络的 ip 地址 my_ip 选项

```
[DEFAULT]
...
my_ip = 10.0.0.11
```

e.在 [DEFAULT] 部分，配置 VNC 代理，使用控制节点管理网络 ip 地址

```
[DEFAULT]
...
vncserver_listen = 10.0.0.11
vncserver_proxyclient_address = 10.0.0.11
```

f.在[glance]部分，配置 image 服务

```
[glance]
...
host = controller
```

g.为排除故障，在[DEFAULT]部分，启用详细日志

```
[DEFAULT]
...
verbose = True
```

3.同步数据库

```
1.  su -s /bin/sh -c "nova-manage db sync" nova
```

完成安装

重启计算服务

```
1.  service nova-api restart
2.  service nova-cert restart
3.  service nova-consoleauth restart
4.  service nova-scheduler restart
5.  service nova-conductor restart
6.  service nova-novncproxy restart
```

如果存在 SQLite 数据库，则删除

```
1.  rm -f /var/lib/nova/nova.sqlite
```

安装配置计算 hypervisor 组件

1.安装包

```
1.  apt-get install nova-compute sysfsutils
```

2.编辑文件 /etc/nova/nova.conf，完成下面内容

a.在[DEFAULT]部分，配置 RabbitMQ 消息代理访问

```
1.  [DEFAULT]
2.  ...
3.  rpc_backend = rabbit
4.  rabbit_host = controller
5.  rabbit_password = RABBIT_PASS
```

b.在 [DEFAULT] 和 [keystone_authtoken] 部分，配置认证服务访问

```
1.  [DEFAULT]
2.  ...
3.  auth_strategy = keystone
4.
5.  [keystone_authtoken]
6.  ...
7.  auth_uri = http://controller:5000/v2.0
8.  identity_uri = http://controller:35357
9.  admin_tenant_name = service
10. admin_user = nova
11. admin_password = NOVA_PASS
```

注意：注释掉其它 auth_host, auth_port, 和 auth_protocol ，因为 identity_uri 已经配置

c.在 [DEFAULT]部分配置，配置计算节点管理网络 ip 地址，my_ip 选项

```
1.  [DEFAULT]
2.  ...
3.  my_ip = 10.0.0.31
```

（这里不要配置成控制节点管理网络 ip 地址）

d.在 [DEFAULT]部分，启用配置远程访问

格式：

```
1.  [DEFAULT]
2.  ...
3.  vnc_enabled = True
4.  vncserver_listen = 0.0.0.0
5.  vncserver_proxyclient_address = MANAGEMENT_INTERFACE_IP_ADDRESS
```

```
6.  novncproxy_base_url = http://controller:6080/vnc_auto.html
```

MANAGEMENT_INTERFACE_IP_ADDRESS 替换为计算节点管理网络 ip 地址，即为

```
1.  [DEFAULT]
2.  ...
3.  vnc_enabled = True
4.  vncserver_listen = 0.0.0.0
5.  vncserver_proxyclient_address = 10.0.0.31
6.  novncproxy_base_url = http://controller:6080/vnc_auto.html
```

这个服务组件侦听所有 IP 地址，代理组件仅监听计算节点管理网络 ip 地址。novncproxy_base_url 表示的是你可以通过浏览器远程访问计算节点产生的实例控制台【也就是我们自己启动的虚拟机】
值得注意的是：
controller 必须配置 hosts，如果没有配置，可以通过 ip 地址。

e.在 [glance]部分，配置 Image 服务的 host

```
1.  [glance]
2.  ...
3.  host = controller
```

f.为排除故障，在[DEFAULT]部分，启用详细日志记录

```
1.  [DEFAULT]
2.  ...
3.  verbose = True
```

完成安装

1.通过下面命令，测试是否支持虚拟机硬件加速

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

如果输出的不是 0，那么不需要额外配置
如果输出的是 0.则使用 QEMU 代替 KVM

编辑文件 etc/nova/nova-compute.conf，在 [libvirt]部分，修改如下

```
1.  [libvirt]
2.  ...
3.  virt_type = qemu
```

2.重启计算服务

```
1.  service nova-compute restart
```

3.如果存在 SQLite 数据库，则删除

```
1.  rm -f /var/lib/nova/nova.sqlite
```

核实操作

注意在控制节点执行

1.生效环境变量

```
1.  source admin-openrc.sh
```

2.列出创建成功的组件进程

```
1.  nova service-list
```

其中四个组件在控制节点，一个组件在计算节点

```
root@controller:~# nova service-list
+----+------------------+------------+----------+---------+-------+----------------------------+-----------------+
| Id | Binary           | Host       | Zone     | Status  | State | Updated_at                 | Disabled Reason |
+----+------------------+------------+----------+---------+-------+----------------------------+-----------------+
| 1  | nova-cert        | controller | internal | enabled | up    | 2015-02-26T04:30:05.000000 | -               |
| 2  | nova-consoleauth | controller | internal | enabled | up    | 2015-02-26T04:30:05.000000 | -               |
| 3  | nova-scheduler   | controller | internal | enabled | up    | 2015-02-26T04:30:06.000000 | -               |
| 4  | nova-conductor   | controller | internal | enabled | up    | 2015-02-26T04:30:06.000000 | -               |
| 5  | nova-compute     | compute    | nova     | enabled | up    | 2015-02-26T04:30:04.000000 | -               |
+----+------------------+------------+----------+---------+-------+----------------------------+-----------------+
root@controller:~# nova-manage service list
Binary           Host                                 Zone             Status     State Updated_At
nova-cert        controller                           internal         enabled    :-)   2015-02-26 04:30:25
nova-consoleauth controller                           internal         enabled    :-)   2015-02-26 04:30:25
nova-scheduler   controller                           internal         enabled    :-)   2015-02-26 04:30:26
nova-conductor   controller                           internal         enabled    :-)   2015-02-26 04:30:26
nova-compute     compute                              nova             enabled    :-)   2015-02-26 04:30:24
root@controller:~#
```

注意：

在创建实例之前，一定要查看上述服务是否正常，否则会造成虚拟机创建错误，比如实例获取不到 ip 地址，状态出现 error 等，这些进程起着非常重要的作用

如果遇到没有启动，可以执行下面命令

```
1.  service nova-api restart

2.  service nova-cert restart

3.  service nova-consoleauth restart

4.  service nova-scheduler restart

5.  service nova-conductor restart

6.  service nova-novncproxy restart
```

3.列出镜像服务目录

```
1.  nova image-list
```

```
+--------------------------------------+------------------+--------+--------------------------------------+
| ID                                   | Name             | Status |
Server                               |
+--------------------------------------+------------------+--------+--------------------------------------+
| 9ec0ceec-0629-406a-9069-159a8b59ea78 | cirros-0.3.3-x86_64 | ACTIVE |                                      |
+--------------------------------------+------------------+--------+--------------------------------------+
```

```
root@controller:~# nova image-list
+--------------------------------------+---------------------+--------+--------------------------------------+
| ID                                   | Name                | Status | Server                               |
+--------------------------------------+---------------------+--------+--------------------------------------+
| 2cf16e02-db12-458f-b004-d2c54985d1e0 | begin               | ACTIVE | 29e13b8a-9105-44aa-9b24-4cd9bef9a912 |
| 9ec0ceec-0629-406a-9069-159a8b59ea78 | cirros-0.3.3-x86_64 | ACTIVE |                                      |
+--------------------------------------+---------------------+--------+--------------------------------------+
```

这里多出一个，是因为后面做了镜像快照

openstack【juno】入门 【网络篇】十四：neutron 介绍

导读：

本文翻译之官网，翻译的并不好，但是翻译的过程，也是一个学习的过程，毕竟是第一手资料。英文较好的同学，或则不好的同学，都建议看看官网的原资料，这样理解才够深刻。

OpenStack 网络

openstack 网络允许创建和附加第三方网络插件，插件可以实现适应不同的网络设备和软件,使用的 openstack 部署和架构更加的灵活。

neutron-server

接受 api 请求，并选择适当的网络插件做出响应

openstack 网络插件和代理

创建和删除端口，创建网络和子网和提供 ip 地址。这些插件与代理不同，取决于使用的特定的云供应商和技术。openstack 网络插件和代理，是由思科的虚拟、物理 switches、NEC OpenFlow products, Open vSwitch, Linux bridging, Ryu Network Operating System, and the VMware NSX product.

通用代理 L3 (layer 3), DHCP (dynamic host IP addressing), 和 一个 plug-in agent.

消息队列

大多数用于在 neutron-server 于多种客户端之间传递信息，也作为一个数据库存储网络插件的状态。openstack 网络与 openstack 计算提供的实例连接相互作用。

网络概念

openstack neutron 在 openstack 环境中，管理所有虚拟网络及访问物理网络结构接入层方面（PNI）openstack neutron 能使网络租户创建网络拓扑，包括服务例如 firewalls, load balancers, and virtual private networks (VPNs).

openstack 网络包括 networks，子网，和路由对象的抽象，每个抽象路由功能模仿物理理由：网络包括子网，路由器连接不同子网及网络之间

每个路由器连接到网络，和网卡连接到子网。子网可以访问连接到相同路由的机器的其它子网
任何的设置，至少有一个外网。不像其他网络，这个外网不止是虚拟的定义网络，相反它可以通过外部网络访问 openstack。对于 openstack 的外部网卡，外部的物理网络都是可以访问的，在这个网络中，dhcp 是被禁用的。

除了外部网络，任何网络都有一个或多个内部网络，这些软件定义的网络直接连接到虚拟机。在任何给定网络的虚拟机，或则在子网中连接相同的路由，可以直接访问 VMs 连接的网络。
对于外部网络访问实例，实例访问外部网络，路由在网络之间是必须的。每个路由器连接到网络和网卡连接到子网，像物理路由器，子网访问其它子网的机器需要连接相同的路由，实例能访问外部网络通过路由。

此外，可以分配外网 ip 地址，对于内部网络的接口。凡是连接子网的，统称为接口。你可以给连接虚拟机端口分配 ip 地址。这样外部网络实体可以访问虚拟机。

openstack 网络也支持安全组，管理员能够自定防火墙规则在安全组中，一个实例可以属于一个或多个安全组实例，网络在安全组中设置的规则，阻止或则允许端口、端口的范围或则虚拟机流量类型。

每个插件,网络的使用都有自己的概念。而操作虚拟网络结构和 openstack 环境不是至关重要的，理解这些概念帮助你创建网络。所有的网络设备使用一个核心插件和安全组插件。此外 Firewall-as-a-Service (FWaaS) 和 Load-Balancer-as-a-Service (LBaaS) plug-ins 插件都是可以利用的

openstack【juno】入门 【网络篇】十五：neutron 安装部署（控制节点）

问题导读

1.neutron 什么时候同步数据库，与其它组件有什么不同？
2.keystone tenant-get service 作用是什么？
3.本文都配置了哪些网络插件？

接上一篇
openstack【juno】入门 【网络篇】十四：neutron 介绍

在安装配置 openstack neutron 之前，需要创建数据库、服务认证、 API endpoints.

安装准备

1.创建数据库

a.登录 mysql

```
1.  mysql -u root -p
```

b.创建 neutron 数据库:

```
1.  CREATE DATABASE neutron;
```

c.授权

```
1.  GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' \
2.    IDENTIFIED BY 'NEUTRON_DBPASS';
3.  GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' \
4.    IDENTIFIED BY 'NEUTRON_DBPASS';
```

d.推出 mysql

```
1.  exit
```

2.环境变量生效

```
1.  source admin-openrc.sh
```

内容为

```
1.  export OS_TENANT_NAME=admin

2.  export OS_USERNAME=admin

3.  export OS_PASSWORD=ADMIN_PASS

4.  export OS_AUTH_URL=http://controller:35357/v2.0
```

3.创建服务凭证

a.创建 neutron 用户:

```
1.  keystone user-create --name neutron --pass NEUTRON_PASS
```

```
root@controller:~# source admin-openrc.sh
root@controller:~# keystone user-create --name neutron --pass NEUTRON_PASS
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |                                  |
| enabled  |               True               |
|    id    | 28c0b8e16e0c48a2b97bc618494af3a7 |
|   name   |             neutron              |
| username |             neutron              |
+----------+----------------------------------+
```

b.给 neutron 用户授予 admin 角色

```
1.  keystone user-role-add --user neutron --tenant service --role admin
```

没有输出

c.创建 neutron 服务实例

```
1.  keystone service-create --name neutron --type network \
```

```
2.    --description "OpenStack Networking"
```

```
root@controller:~# keystone service-create --name neutron --type network \
> --description "OpenStack Networking"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |       OpenStack Networking       |
|   enabled   |               True               |
|      id     | f9eef7516e8447568367239309fdf444 |
|     name    |             neutron              |
|     type    |             network              |
+-------------+----------------------------------+
```

4.创建网路服务 API endpoints:

```
1.  keystone endpoint-create \
2.    --service-id $(keystone service-list | awk '/ network / {print $2}') \
3.    --publicurl http://controller:9696 \
4.    --adminurl http://controller:9696 \
5.    --internalurl http://controller:9696 \
6.    --region regionOne
```

```
root@controller:~# keystone endpoint-create \
> --service-id $(keystone service-list | awk '/ network / {print $2}') \
> --publicurl http://controller:9696 \
> --adminurl http://controller:9696 \
> --internalurl http://controller:9696 \
> --region regionOne
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  |      http://controller:9696      |
|      id     | c9d1fa7f71ef438dbf6237d912b56796 |
| internalurl |      http://controller:9696      |
|  publicurl  |      http://controller:9696      |
|    region   |            regionOne             |
|  service_id | f9eef7516e8447568367239309fdf444 |
+-------------+----------------------------------+
```

安装网络组件

安装组件

```
1.  apt-get install neutron-server neutron-plugin-ml2 python-neutronclient
```

```
root@controller:~# apt-get install neutron-server neutron-plugin-ml2 python-neutronclient
Reading package lists... Done
Building dependency tree
Reading state information... Done
python-neutronclient is already the newest version.
python-neutronclient set to manually installed.
The following extra packages will be installed:
  ipset libipset3 libmnl0 neutron-common python-jsonrpclib python-neutron
The following NEW packages will be installed:
  ipset libipset3 libmnl0 neutron-common neutron-plugin-ml2 neutron-server
  python-jsonrpclib python-neutron
0 upgraded, 8 newly installed, 0 to remove and 0 not upgraded.
Need to get 1,388 kB of archives.
After this operation, 13.1 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu/ trusty/main python-jsonrpclib all 0.1.3-1build1 [14.1 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu/ trusty/main libmnl0 amd64 1.0.3-3ubuntu1 [11.4 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu/ trusty/universe libipset3 amd64 6.20.1-1 [50.8 kB]
```

配置网络服务组件

网络服务器组件配置包括数据库，认证机制，消息代理，拓扑改变报警，和插件。

编辑文件 /etc/neutron/neutron.conf，完成下面下面配置

```
1.  sudo nano  /etc/neutron/neutron.conf
```

a.在 [database] 部分，配置数据库访问：

```
1.  [database]

2.  ...

3.  connection = mysql://neutron:NEUTRON_DBPASS@controller/neutron
```

b.在 [DEFAULT] 部分，配置 RabbitMQ 消息代理访问：

```
1.  [DEFAULT]

2.  ...

3.  rpc_backend = rabbit

4.  rabbit_host = controller

5.  rabbit_password = RABBIT_PASS
```

c.在 [DEFAULT] 和 [keystone_authtoken] 部分,配置认证服务

```
1.  [DEFAULT]
2.  ...
3.  auth_strategy = keystone
4.
5.  [keystone_authtoken]
6.  ...
7.  auth_uri = http://controller:5000/v2.0
8.  identity_uri = http://controller:35357
9.  admin_tenant_name = service
10. admin_user = neutron
11. admin_password = NEUTRON_PASS
```

注释掉其它 auth_host, auth_port, 和 auth_protocol，因为 identity_uri 可能被覆盖

```
[keystone_authtoken]
#auth_host = 127.0.0.1
#auth_port = 35357
#auth_protocol = http
#admin_tenant_name = %SERVICE_TENANT_NAME%
#admin_user = %SERVICE_USER%
#admin_password = %SERVICE_PASSWORD%
auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357
admin_tenant_name = service
admin_user = neutron
admin_password = NEUTRON_PASS
```

d.在[DEFAULT]部分配置 Modular Layer 2 (ML2) plug-in, router service, and overlapping IP addresses:

```
1.  [DEFAULT]
2.  ...
3.  core_plugin = ml2
4.  service_plugins = router
5.  allow_overlapping_ips = True
```

e.在 [DEFAULT] 部分，配置网络的网络计算拓扑变化通知：

```
1.  [DEFAULT]
2.  ...
3.  notify_nova_on_port_status_changes = True
```

```
notify_nova_on_port_data_changes = True
nova_url = http://controller:8774/v2
nova_admin_auth_url = http://controller:35357/v2.0
nova_region_name = regionOne
nova_admin_username = nova
nova_admin_tenant_id = SERVICE_TENANT_ID
nova_admin_password = NOVA_PASS
```

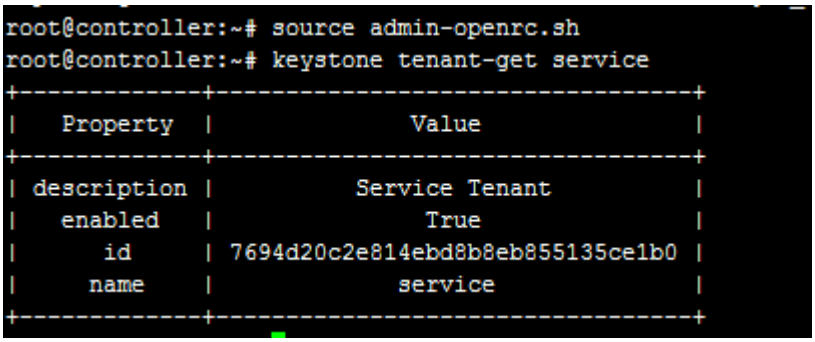
注意，这里 SERVICE_TENANT_ID 是 keystone service 租户 id 。nova_pass 是 nova 用户密码，这是使用的是默认密码

注意保存 service 租户的 tenant identifier (id):

```
source admin-openrc.sh
```

```
keystone tenant-get service
```

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |          Service Tenant          |
|   enabled   |               True               |
|      id     | 7694d20c2e814ebd8b8eb855135ce1b0 |
|     name    |             service              |
+-------------+----------------------------------+
```



上面配置替换为下面：

```
[DEFAULT]
...
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
nova_url = http://controller:8774/v2
nova_admin_auth_url = http://controller:35357/v2.0
nova_region_name = regionOne
nova_admin_username = nova
nova_admin_tenant_id = 7694d20c2e814ebd8b8eb855135ce1b0
nova_admin_password = NOVA_PASS
```

f.为帮助排除问题，在[DEFAULT]启用 verbose

```
[DEFAULT]
...
verbose = True
```

配置 Modular Layer 2 (ML2) 插件

ML2 插件使用 Open vSwitch (OVS)机制，为实例创建虚拟网络框架。尽管如此控制节点不需要 OVS 组件，因为它不处理实例网络互通

编辑文件 /etc/neutron/plugins/ml2/ml2_conf.ini，完成下面步骤

```
sudo nano  /etc/neutron/plugins/ml2/ml2_conf.ini
```

a.在 [ml2]部分，配置启用 flat 和 generic routing encapsulation (GRE) 网络驱动，GRE 租户网络和 ovs 驱动

```
[ml2]
...
type_drivers = flat,gre
tenant_network_types = gre
mechanism_drivers = openvswitch
```

注意：

一旦配置 ML2 插件，禁用网络驱动和重启，将会导致数据库不一致。

b.在 [ml2_type_gre] 部分，配置 tunnel identifier (id) 范围:

```
1.  [ml2_type_gre]
2.  ...
3.  tunnel_id_ranges = 1:1000
```

c.在 [securitygroup] 部分，配置安全组，ipset，配置 OVS 防火墙驱动

```
1.  [securitygroup]
2.  ...
3.  enable_security_group = True
4.  enable_ipset = True
5.  firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
```

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

# Controls if neutron security group is enabled or not.
# It should be false when you use nova security group.
# enable_security_group = True

# Use ipset to speed-up the iptables security groups. Enabling ipset support
# requires that ipset is installed on L2 agent node.
# enable_ipset = True
```

使用网络配置计算

默认情况下，分配包配置使用传统网络计算。你不许通过网络重新配置管理网络计算
编辑文件 /etc/nova/nova.conf，完成下面操作

```
1.  sudo nano  /etc/nova/nova.conf
```

a.在 [DEFAULT]默认部分，配置 api 和驱动

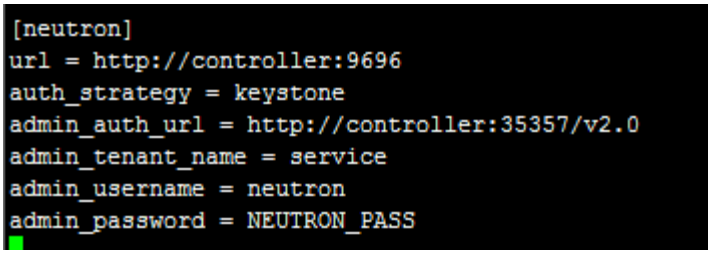
```
[DEFAULT]
...
network_api_class = nova.network.neutronv2.api.API
security_group_api = neutron
linuxnet_interface_driver = nova.network.linux_net.LinuxOVSInterfaceDriver
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

注意：
默认，计算使用内部防火墙网络，自从网络包含网络防火墙服务，你必须使用 nova.virt.firewall.NoopFirewallDriver 防火墙驱动停掉计算防火墙服务。

b.在 [neutron] 部分，配置访问参数

```
[neutron]
...
url = http://controller:9696
auth_strategy = keystone
admin_auth_url = http://controller:35357/v2.0
admin_tenant_name = service
admin_username = neutron
admin_password = NEUTRON_PASS
```

发现没有这个标签，直接添加了




```
[DEFAULT]
network_api_class = nova.network.neutronv2.api.API
security_group_api = neutron
linuxnet_interface_driver = nova.network.linux_net.LinuxOVSInterfaceDriver
firewall_driver = nova.virt.firewall.NoopFirewallDriver


```

```
8.  dhcpbridge_flagfile=/etc/nova/nova.conf
9.  dhcpbridge=/usr/bin/nova-dhcpbridge
10. logdir=/var/log/nova
11. state_path=/var/lib/nova
12. lock_path=/var/lock/nova
13. force_dhcp_release=True
14. libvirt_use_virtio_for_bridges=True
15. verbose=True
16. ec2_private_dns_show_ip=True
17. api_paste_config=/etc/nova/api-paste.ini
18. enabled_apis=ec2,osapi_compute,metadata
19. rpc_backend = rabbit
20. rabbit_host = controller
21. rabbit_password = RABBIT_PASS
22. auth_strategy = keystone
23. my_ip = 10.0.0.11
24. vncserver_listen = 10.0.0.11
25. vncserver_proxyclient_address = 10.0.0.11
26. verbose = True
27.
28. [database]
29. connection = mysql://nova:NOVA_DBPASS@controller/nova
30.
31. [keystone_authtoken]
32.
33. auth_uri = http://controller:5000/v2.0
34. identity_uri = http://controller:35357
35. admin_tenant_name = service
36. admin_user = nova
37. admin_password = NOVA_PASS
38.
39.
```

```
[glance]
host = controller
[neutron]
url = http://controller:9696
auth_strategy = keystone
admin_auth_url = http://controller:35357/v2.0
admin_tenant_name = service
admin_username = neutron
admin_password = NEUTRON_PASS
```

完成安装

1.同步数据库

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

注意：同步数据库最后，是因为脚本需要完成服务 和 插件的配置文件

2.重启计算服务

```
service nova-api restart
service nova-scheduler restart
service nova-conductor restart
```

```
root@controller:~#  service nova-api restart
nova-api stop/waiting
nova-api start/running, process 23427
root@controller:~#  service nova-scheduler restart
nova-scheduler stop/waiting
nova-scheduler start/running, process 23444
root@controller:~#  service nova-conductor restart
nova-conductor stop/waiting
nova-conductor start/running, process 23474
root@controller:~#
```

3.重启网络服务

```
1.  service neutron-server restart
```

验证是否成功

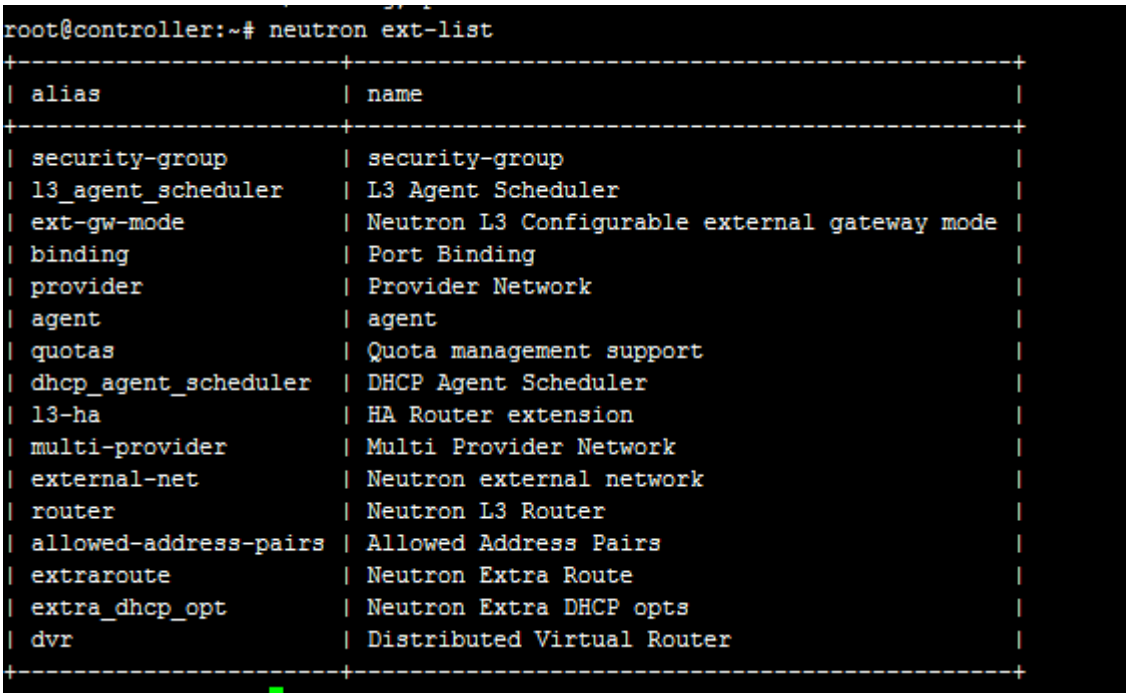
在控制节点上执行下面命令

1.生效环境变量

```
1.  source admin-openrc.sh
```

2.列出创建的加载扩展的 neutron-server 进程

```
1.  neutron ext-list
```



```
root@controller:~# neutron ext-list
+-----------------------+-----------------------------------------------+
| alias                 | name                                          |
+-----------------------+-----------------------------------------------+
| security-group        | security-group                                |
| l3_agent_scheduler    | L3 Agent Scheduler                            |
| ext-gw-mode           | Neutron L3 Configurable external gateway mode |
| binding               | Port Binding                                  |
| provider              | Provider Network                              |
| agent                 | agent                                         |
| quotas                | Quota management support                      |
| dhcp_agent_scheduler  | DHCP Agent Scheduler                          |
| l3-ha                 | HA Router extension                           |
| multi-provider        | Multi Provider Network                        |
| external-net          | Neutron external network                      |
| router                | Neutron L3 Router                             |
| allowed-address-pairs | Allowed Address Pairs                         |
| extraroute            | Neutron Extra Route                           |
| extra_dhcp_opt        | Neutron Extra DHCP opts                       |
| dvr                   | Distributed Virtual Router                    |
+-----------------------+-----------------------------------------------+
```

遇到问题：

不能连接 http://controller:9696/v2.0/extensions.json

root@controller:~# neutron ext-list

Unable to establish connection to http://controller:9696/v2.0/extensions.json

原因：

不能同步数据库

，同步即解决

```
1.  su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/neutron.conf \

2.    --config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade juno" neutron
```

openstack【juno】入门 【网络篇】十六：neutron 安装部署（网络节点）

问题导读

1.ovs 部署在哪个节点?
2.如何验证网络节点安装成功?
3.添加外部网桥网卡如何选择?

接上篇
openstack【juno】入门 【网络篇】十五：neutron 安装部署（控制节点）

安装配置网络节点

网络节点主要处理内部和外部的虚拟路由和 DHCP 服务。

准备

在配置 neutron 之前，必须配置一定的内核网络参数

1.修改文件 /etc/sysctl.conf，添加如下内容：

```
1.  sudo nano  /etc/sysctl.conf
```

```
1.  net.ipv4.ip_forward=1
```

```
2.  net.ipv4.conf.all.rp_filter=0
3.  net.ipv4.conf.default.rp_filter=0
```

2.生效

```
1.  sysctl -p
```

安装网络组件

```
1.  apt-get install neutron-plugin-ml2 neutron-plugin-openvswitch-agent \
2.    neutron-l3-agent neutron-dhcp-agent
```

配置网络组件

网络通用组件配置包括：认证机制、消息代理、和插件

编辑文件 /etc/neutron/neutron.conf 完成下面内容：

```
1.  sudo nano /etc/neutron/neutron.conf
```

a.在 [database]部分，注释掉连接数据 connection 部分，网络节点不直接访问数据库
这里如果没有对文件进行改动，不需要进行操作。

b.在 [DEFAULT] 部分，配置消息代理访问

```
1.  [DEFAULT]
2.  ...
3.  rpc_backend = rabbit
4.  rabbit_host = controller
5.  rabbit_password = RABBIT_PASS
```

记得替换 RABBIT_PASS，这里使用的是默认，如果在安装时进行了更改，则使用更改密码。

c.在 [DEFAULT] 和 [keystone_authtoken] 部分，配置身份服务访问：

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

[keystone_authtoken]
...
auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357
admin_tenant_name = service
admin_user = neutron
admin_password = NEUTRON_PASS
```

注意注释掉含有 auth_host, auth_port, and auth_protocol 的选项，以免新配置被覆盖

```
[keystone_authtoken]
#auth_host = 127.0.0.1
#auth_port = 35357
#auth_protocol = http
#admin_tenant_name = %SERVICE_TENANT_NAME%
#admin_user = %SERVICE_USER%
#admin_password = %SERVICE_PASSWORD%


auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357
admin_tenant_name = service
admin_user = neutron
admin_password = NEUTRON_PASS
```

d.在 [DEFAULT] 部分，启用 Modular Layer 2 (ML2)插件，路由服务，和 ip 地址

```
[DEFAULT]
...
core_plugin = ml2
```

```
service_plugins = router
allow_overlapping_ips = True
```

e.（可选）协助排除故障，详细记录在[DEFAULT]部分使：

```
[DEFAULT]
...
verbose = True
```

配置 Modular Layer 2 (ML2)插件

ML2 插件使用 Open vSwitch (OVS)机制为实例建立虚拟网络架构

修改文件/etc/neutron/plugins/ml2/ml2_conf.ini ，完成下面内容

```
sudo nano /etc/neutron/plugins/ml2/ml2_conf.ini
```

a.在 [ml2] 部分，启用 flat 和 generic routing encapsulation (GRE)网络类型驱动， GRE 租户网络，和 OVS 机制驱动:

```
[ml2]
...
type_drivers = flat,gre
tenant_network_types = gre
mechanism_drivers = openvswitch
```

b.在 [ml2_type_flat] 部分，配置 flat_networks 为 external

```
[ml2_type_flat]
...
flat_networks = external
```

c.在 [ml2_type_gre] 部分，配置 tunnel id 的范围

```
[ml2_type_gre]
...
tunnel_id_ranges = 1:1000
```

d.在 [securitygroup]部分，启用 安全组，ipset，和 配置 OVS 防火墙驱动:

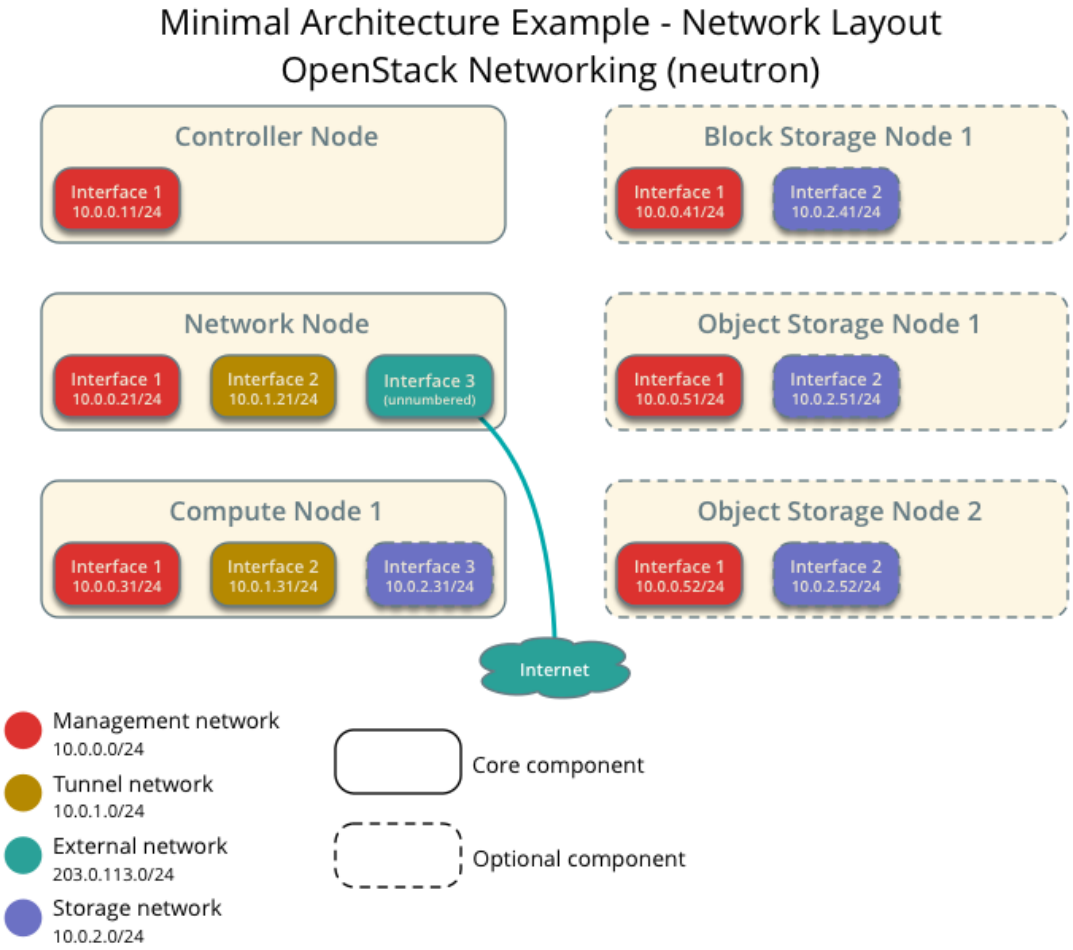
```
[securitygroup]
...
enable_security_group = True
enable_ipset = True
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
```

e.在 [ovs] 部分，启用 tunnels，配置本地 tunnel endpoint，和 bridge_mappings

bridge_mappings:map the external flat provider network to the br-ex external network bridge:

```
[ovs]
...
local_ip = INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS
enable_tunneling = True
bridge_mappings = external:br-ex
```

INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS 是网络节点上的实例隧道网络的 ip 地址



根据上图。ip 地址为：10.0.1.21

即替换为下面内容：

```
1.  [ovs]
2.  ...
3.  local_ip = 10.0.1.21
4.  enable_tunneling = True
5.  bridge_mappings = external:br-ex
```

[ovs]未找不到,直接添加

f.在 [agent] 部分，启动 GRE tunnels

```
1.  [agent]
```

```
2.  ...
3.  tunnel_types = gre
```

[agent]未找不到,直接添加

配置 Layer-3 (L3) 代理

Layer-3 (L3) agent 为虚拟网络提供路由服务

编辑文件 /etc/neutron/l3_agent.ini，完成下面内容

```
1.  sudo nano  /etc/neutron/l3_agent.ini
```

a.在 [DEFAULT] 部分，配置驱动，启用 network namespaces，配置外部网桥，使失效的路由器名称删除

```
1.  [DEFAULT]
2.  ...
3.  interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
4.  use_namespaces = True
5.  external_network_bridge = br-ex
6.  router_delete_namespaces = True
```

b.（可选）协助排除故障，详细记录在[DEFAULT]部分：

```
1.  [DEFAULT]
2.  ...
3.  verbose = True
```

配置 DHCP 代理

DHCP 代理为虚拟网络提供 DHCP 服务【如果这里配置错误，openstack 产生实例，会获取不到 ip 地址】

1.编辑文件 /etc/neutron/dhcp_agent.ini，完成下面内容。

a.在 [DEFAULT] 部分，配置驱动，启用命名空间和启用删除废弃的命名空间

```
1.  [DEFAULT]
2.  ...
3.  interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
4.  dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
5.  use_namespaces = True
6.  dhcp_delete_namespaces = True
```

b.可选为排除错误，启用 verbose 在[DEFAULT] 部分:

```
1.  [DEFAULT]
2.  ...
3.  verbose = True
```

2.可选

a.编辑文件 /etc/neutron/dhcp_agent.ini，完成下面内容

在 [DEFAULT]部分，启用 the dnsmasq 配置文件:

```
1.  [DEFAULT]
2.  ...
3.  dnsmasq_config_file = /etc/neutron/dnsmasq-neutron.conf
```

b.创建文件 /etc/neutron/dnsmasq-neutron.conf，并添加下面内容

```
1.  dhcp-option-force=26,1454
```

c.杀死已经存在的 dnsmasq 进程

```
1.  Kill any existing dnsmasq processes:
```

配置元数据代理

元数据代理提供配置信息，比如实例凭证.

注释：

metadata agent（元数据代理）

openstack 网络代理为实例提供元数据服务

1.编辑 /etc/neutron/metadata_agent.ini 文件，完成下面内容

```
1.  sudo nano /etc/neutron/metadata_agent.ini
```

a.在[DEFAULT] 部分，配置访问参数

```
1.  [DEFAULT]
2.  ...
3.  auth_url = http://controller:5000/v2.0
4.  auth_region = regionOne
5.  admin_tenant_name = service
6.  admin_user = neutron
7.  admin_password = NEUTRON_PASS
```

同样不要忘记 NEUTRON_PASS 替换密码，这里使用的是默认密码

不要忘记注释掉其它授权，以免被覆盖

```
1.  # The Neutron user information for accessing the Neutron API.
2.  #auth_url = http://localhost:5000/v2.0
3.  #auth_region = RegionOne
4.  # Turn off verification of the certificate for ssl
5.  # auth_insecure = False
6.  # Certificate Authority public key (CA cert) file for ssl
7.  # auth_ca_cert =
8.  #admin_tenant_name = %SERVICE_TENANT_NAME%
9.  #admin_user = %SERVICE_USER%
10. #admin_password = %SERVICE_PASSWORD%
```

```
11.
12. auth_url = http://controller:5000/v2.0
13. auth_region = regionOne
14. admin_tenant_name = service
15. admin_user = neutron
16. admin_password = NEUTRON_PASS
17.
```

b. 在[DEFAULT] 部分，配置元数据主机

```
1.  [DEFAULT]
2.  ...
3.  nova_metadata_ip = controller
```

复制代码

c.在[DEFAULT]部分，配置元数据代理共享密码：

```
1.  [DEFAULT]
2.  ...
3.  metadata_proxy_shared_secret = METADATA_SECRET
```

d.（可选）协助排除故障，详细记录在[DEFAULT]部分：

```
1.  [DEFAULT]
2.  ...
3.  verbose = True
```

2.在控制节点编辑文件/etc/nova/nova.conf，完成下面内容

```
1.  sudo nano /etc/nova/nova.conf
```

在[neutron]部分，启动元数据代理和配置密码

```
1.  [neutron]
2.  ...
3.  service_metadata_proxy = True
4.  metadata_proxy_shared_secret = METADATA_SECRET
```

METADATA_SECRET 替换为自己设置的密码，这里使用的是默认密码

3.在控制节点重启 api 服务

```
1.  service nova-api restart
```

配置 Open vSwitch (OVS) 服务

OVS 服务为实例提供底层虚拟网络框架，集成网桥 br-int 处理 OVS 内实例内部网络的流通。外部网桥 br-ex 处理 ovs 内实例外部网络的流通。外部网桥需要提供外部网络接口来访问实例外部网络。本质上，在你的环境中，连接着虚拟外部网络和物理外部网络。翻译不是太准确，请看英文原文，欢迎纠正。
英文原文

```
1.  The OVS service provides the underlying virtual networking framework for instances. The
    integration bridge br-int handles internal instance network traffic within OVS. The external
    bridge br-ex handles external instance network traffic within OVS. The external bridge
    requires a port on the physical external network interface to provide instances with external
    network access. In essence, this port connects the virtual and physical external networks
    in your environment.
```

下面是在网络节点

1.重启 ovs 服务

```
1.  service openvswitch-switch restart
```

2.添加外部网桥

```
1.  ovs-vsctl add-br br-ex
```

添加一个外部网桥连接外部网络的物理网卡

替换 INTERFACE_NAME 用实际网卡 ，本文为 eth2

```
1.  ovs-vsctl add-port br-ex INTERFACE_NAME
```

替换：

```
1.  ovs-vsctl add-port br-ex eth2
```

注意：
根据网络接口驱动程序，你需要停掉 generic receive offload (GRO)，来获得合适网络流量，在实例和外部网络之间。
在测试环境中，需要禁用 GRO。

```
1.  ethtool -K eth2 gro off
```

```
root@network:~# ethtool -K eth2 gro off
The program 'ethtool' is currently not installed. You can install it by typing:
apt-get install ethtool
root@network:~# apt-get install ethtool
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-3.13.0-32 linux-headers-3.13.0-32-generic
  linux-image-3.13.0-32-generic linux-image-extra-3.13.0-32-generic
Use 'apt-get autoremove' to remove them.
The following NEW packages will be installed:
  ethtool
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 90.9 kB of archives.
After this operation, 333 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu/ trusty/main ethtool amd64 1:3.13-1 [90.9 kB]
Fetched 90.9 kB in 3s (25.6 kB/s)
Selecting previously unselected package ethtool.
(Reading database ... 119592 files and directories currently installed.)
Preparing to unpack .../ethtool_1%3a3.13-1_amd64.deb ...
Unpacking ethtool (1:3.13-1) ...
Processing triggers for man-db (2.6.7.1-1ubuntu1) ...
Setting up ethtool (1:3.13-1) ...
root@network:~# ethtool -K eth2 gro off
```

完成安装

重启服务

```
1.  service neutron-plugin-openvswitch-agent restart
2.  service neutron-l3-agent restart
3.  service neutron-dhcp-agent restart
4.  service neutron-metadata-agent restart
```

输出如下内容：

```
1.  root@network:~# service neutron-plugin-openvswitch-agent restart
2.  neutron-plugin-openvswitch-agent stop/waiting
3.  neutron-plugin-openvswitch-agent start/running, process 19501
4.  root@network:~# service neutron-l3-agent restart
5.  stop: Unknown instance:
6.  neutron-l3-agent start/running, process 19521
```

```
root@network:~# service neutron-dhcp-agent restart
neutron-dhcp-agent stop/waiting
neutron-dhcp-agent start/running, process 19542
root@network:~# service neutron-metadata-agent restart
neutron-metadata-agent stop/waiting
neutron-metadata-agent start/running, process 19731
root@network:~# service neutron-l3-agent restart
neutron-l3-agent stop/waiting
neutron-l3-agent start/running, process 19768
root@network:~# service neutron-plugin-openvswitch-agent restart
```

检验安装

注意：

在控制节点执行下面命令

1.生效环境变量

```
source admin-openrc.sh
```

2.列出创建成功的 neutron 代理

```
neutron agent-list
```

输出如下内容：

```
+--------------------------------------+--------------------+---------+-------+----------------+---------------------------+
| id                                   | agent_type         | host    | alive | admin_state_up | binary                    |
```

```
3.  +--------------------------------------+--------------------+---------+-------+-------
    ---------+---------------------------+
4.  | 051543d3-4be3-45b9-a9a2-5bbd3e89a47c | Open vSwitch agent | network | :-)   |
    True           | neutron-openvswitch-agent |
5.  | 16bd7da4-b76e-4fbd-9e5f-92b52a8c70a5 | DHCP agent         | network | :-)   |
    True           | neutron-dhcp-agent        |
6.  | 2882756c-e965-4070-a105-9b408ef1cebc | L3 agent           | network | :-)   |
    True           | neutron-l3-agent          |
7.  | 5c4678e6-4096-44fe-95ad-5c4c26f2cc43 | Metadata agent     | network | :-)   |
    True           | neutron-metadata-agent    |
8.  +--------------------------------------+--------------------+---------+-------+-------
    ---------+---------------------------+
```

```
root@controller:~# neutron agent-list
+--------------------------------------+--------------------+---------+-------+----------------+---------------------------+
| id                                   | agent_type         | host    | alive | admin_state_up | binary                    |
+--------------------------------------+--------------------+---------+-------+----------------+---------------------------+
| 051543d3-4be3-45b9-a9a2-5bbd3e89a47c | Open vSwitch agent | network | :-)   | True           | neutron-openvswitch-agent |
| 16bd7da4-b76e-4fbd-9e5f-92b52a8c70a5 | DHCP agent         | network | :-)   | True           | neutron-dhcp-agent        |
| 2882756c-e965-4070-a105-9b408ef1cebc | L3 agent           | network | :-)   | True           | neutron-l3-agent          |
| 5c4678e6-4096-44fe-95ad-5c4c26f2cc43 | Metadata agent     | network | :-)   | True           | neutron-metadata-agent    |
+--------------------------------------+--------------------+---------+-------+----------------+---------------------------+
```

openstack【juno】入门 【网络篇】十七：neutron 安装部署（计算节点）

问题导读

1.计算节点管理网络为那个网段 ip？
2.网络节点如何实现不直接访问数据库？

接上篇
openstack【juno】入门 【网络篇】十六：neutron 安装部署（网络节点）

计算节点处理实例的连通性及安全组

安装准备

1.在配置 openstack neutron 之前，需要配置一定的核心网络参数

修改文件/etc/sysctl.conf，完成下面内容

```
1.  sudo nano /etc/sysctl.conf
```

```
1.  net.ipv4.conf.all.rp_filter=0
2.  net.ipv4.conf.default.rp_filter=0
```

2.生效修改内容

```
1.  sysctl -p
```

安装网络组件

```
1.  apt-get install neutron-plugin-ml2 neutron-plugin-openvswitch-agent
```

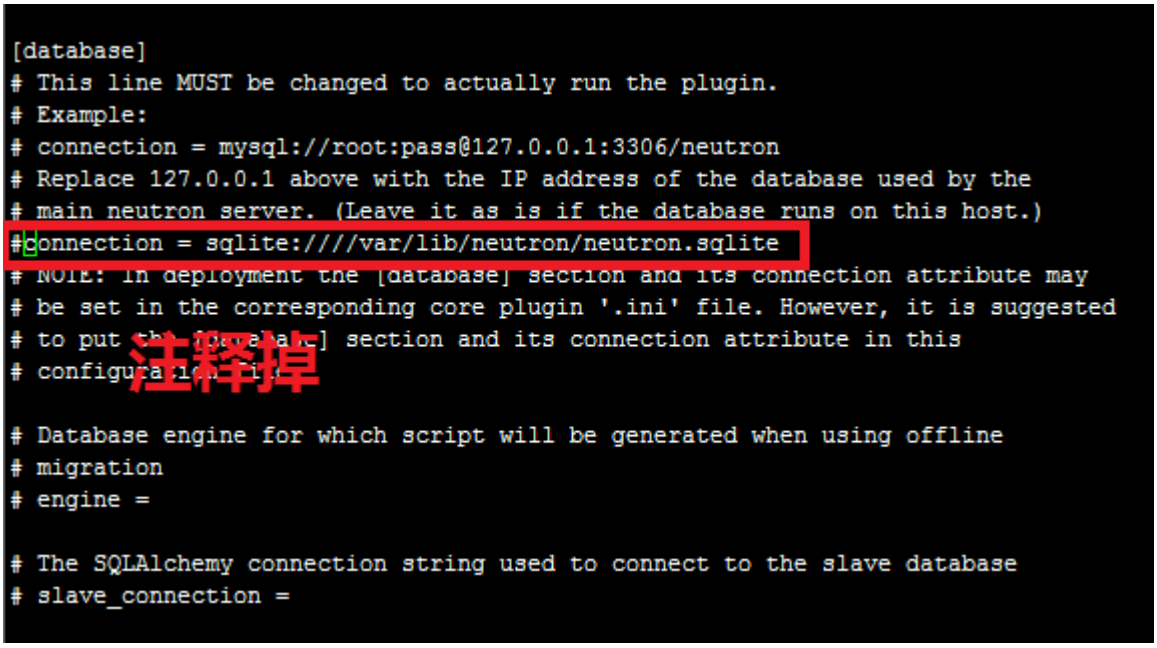
配置网络通用组件

网络组件的配置包括：认证机制、消息代理、和插件

修改文件 /etc/neutron/neutron.conf，完成下面内容：

```
1.  sudo nano  /etc/neutron/neutron.conf
```

a.在 [database] 部分，注释掉 connection 选项，因为网络组件不直接访问数据库



b.在 [DEFAULT] 部分，配置 RabbitMQ 消息代理访问

```
1.  [DEFAULT]
2.  ...
3.  rpc_backend = rabbit
4.  rabbit_host = controller
5.  rabbit_password = RABBIT_PASS
```

这里的密码采用的是默认

c.在 [DEFAULT] 和 [keystone_authtoken] 部分，配置身份服务访问

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

[keystone_authtoken]
...
auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357
admin_tenant_name = service
admin_user = neutron
admin_password = NEUTRON_PASS
```

NEUTRON_PASS 这里使用的是默认密码

```
[keystone_authtoken]
#auth_host = 127.0.0.1
#auth_port = 35357
#auth_protocol = http
#admin_tenant_name = %SERVICE_TENANT_NAME%
#admin_user = %SERVICE_USER%
#admin_password = %SERVICE_PASSWORD%
auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357
admin_tenant_name = service
admin_user = neutron
admin_password = NEUTRON_PASS
```

记得注释掉 auth_host, auth_port, and auth_protocol 选项，以免配置被覆盖

d.在 [DEFAULT] 部分，启用 Layer 2 (ML2) 插件，路由服务，overlapping IP addresses

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

e.（可选）协助排除故障，开启 verbose 记录在[DEFAULT]部分：

```
[DEFAULT]
...
verbose = True
```

配置 Modular Layer 2 (ML2) 插件

ML2 插件使用 Open vSwitch (OVS) 机制创建实例的虚拟网络
编辑文件 /etc/neutron/plugins/ml2/ml2_conf.ini ，完成下面内容：

```
sudo nano  /etc/neutron/plugins/ml2/ml2_conf.ini
```

a. 在[ml2]部分，启用 flat 和 generic routing encapsulation (GRE)网络类型驱动，GRE 租户网络和 OVS 机制驱动。

```
[ml2]
...
type_drivers = flat,gre
tenant_network_types = gre
mechanism_drivers = openvswitch
```

b.在 [ml2_type_gre] 部分，配置 tunnel id 范围

```
[ml2_type_gre]
...
tunnel_id_ranges = 1:1000
```

c. 在[securitygroup] 部分，启用安全组，启用 ipset，配置 OVS iptables，防火墙驱动

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

d.在 [ovs] 部分，启用 tunnels ，配置本地 tunnel endpoint:

```
1.  [ovs]
2.  ...
3.  local_ip = INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS
4.  enable_tunneling = True
```

替换 INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS，为计算节点实例隧道 ip.这里管理网络为 10.0.1.31，替换如下

```
1.  [ovs]
2.  ...
3.  local_ip =  10.0.1.31
4.  enable_tunneling = True
```

没有找到，直接添加

e.在 [agent] 部分，启用 GRE 隧道

```
1.  [agent]
2.  ...
3.  tunnel_types = gre
```

配置 Open vSwitch (OVS) 服务

ovs 服务为实例提供 underlying 虚拟网络服务框架。
重启 OVS 服务

```
1.  service openvswitch-switch restart
```

配置计算节点使用网络

默认情况下，分布式包配置使用传统网络，你必须重新配置计算节点管理网络
原文

```
1.  By default, distribution packages configure Compute to use legacy networking. You must
    reconfigure Compute to manage networks through Networking.
```

修改文件 /etc/nova/nova.conf，完成下面内容

```
1.  sudo  nano  /etc/nova/nova.conf
```

a.在 [DEFAULT]部分，配置 APIs 和驱动 ：

```
1.  [DEFAULT]
2.  ...
3.  network_api_class = nova.network.neutronv2.api.API
4.  security_group_api = neutron
5.  linuxnet_interface_driver = nova.network.linux_net.LinuxOVSInterfaceDriver
6.  firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

说明：
默认，计算使用内部防火墙服务，自从 neutron 包括防火墙服务，你必须使用 nova.virt.firewall.NoopFirewallDriver 防火墙驱动，禁用计算防火墙服务

b. 在[neutron] 部分，配置访问参数

```
1.  [neutron]
2.  ...
3.  url = http://controller:9696
4.  auth_strategy = keystone
5.  admin_auth_url = http://controller:35357/v2.0
6.  admin_tenant_name = service
7.  admin_username = neutron
8.  admin_password = NEUTRON_PASS
```

在身份认证 neutron 用户，替换 NEUTRON_PASS 密码，这里使用默认密码。

完成安装

1.重启计算服务

```
1.  service nova-compute restart
```

2.重启 Open vSwitch (OVS) 代理:

```
1.  service neutron-plugin-openvswitch-agent restart
```

验证安装

注意：这里是在控制节点安装

1.生效环境变量

```
1.  source admin-openrc.sh
```

2.列出创建的 neutron 代理

```
1.  neutron agent-list
```

```
root@controller:~# neutron agent-list
+--------------------------------------+--------------------+---------+-------+----------------+---------------------------+
| id                                   | agent_type         | host    | alive | admin_state_up | binary                    |
+--------------------------------------+--------------------+---------+-------+----------------+---------------------------+
| 051543d3-4be3-45b9-a9a2-5bbd3e89a47c | Open vSwitch agent | network | :-)   | True           | neutron-openvswitch-agent |
| 16bd7da4-b76e-4fbd-9e5f-92b52a8c70a5 | DHCP agent         | network | :-)   | True           | neutron-dhcp-agent        |
| 2882756c-e965-4070-a105-9b408ef1cebc | L3 agent           | network | :-)   | True           | neutron-l3-agent          |
| 40ee5d4d-c2b3-4ac7-b036-716d4e752a6b | Open vSwitch agent | compute | :-)   | True           | neutron-openvswitch-agent |
| 5c4678e6-4096-44fe-95ad-5c4c26f2cc43 | Metadata agent     | network | :-)   | True           | neutron-metadata-agent    |
+--------------------------------------+--------------------+---------+-------+----------------+---------------------------+
```

openstack【juno】入门 【网络篇】十八：创建实例化网络

问题导读

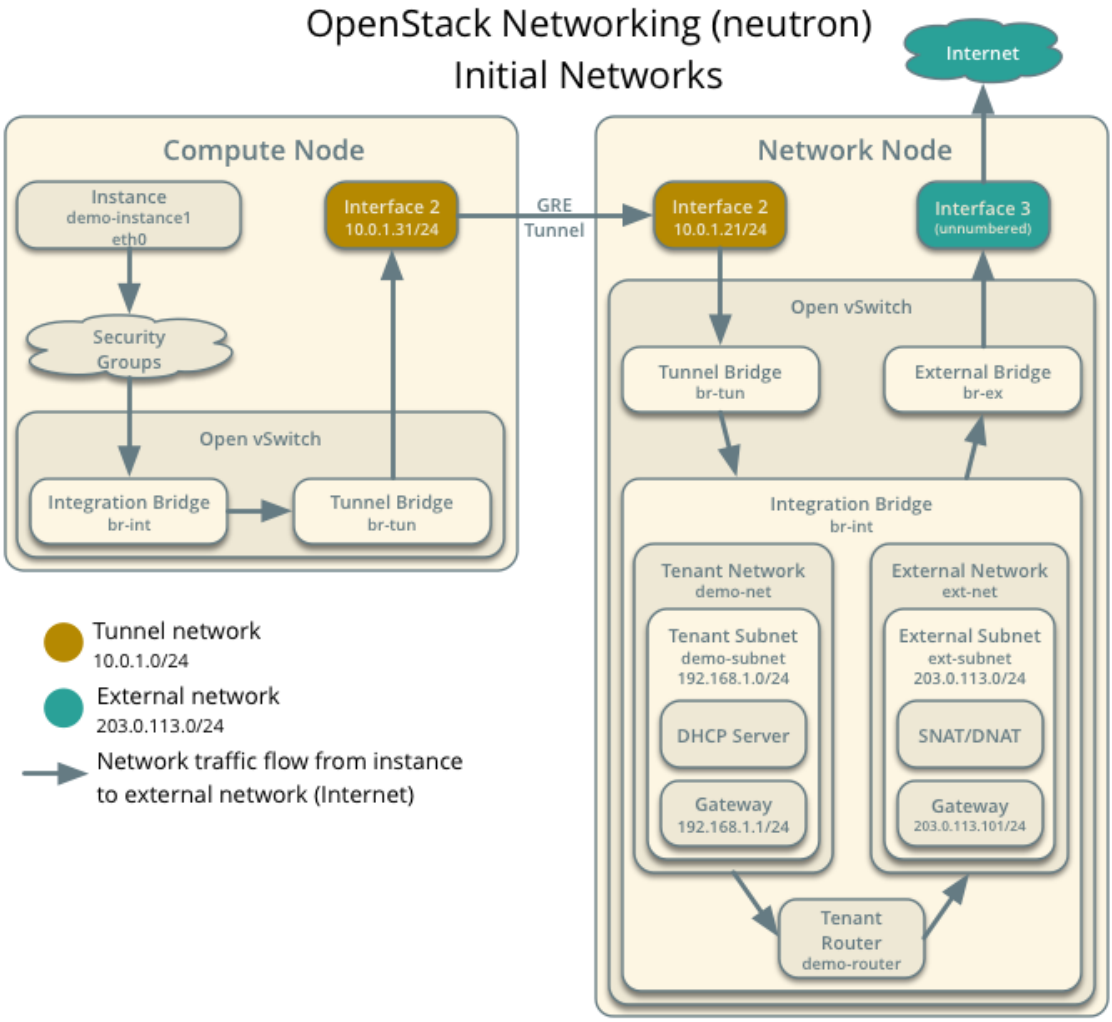
1.如何添加租户网络？
2.如何添加路由？
3.网络连通性如何验证？

接上篇：openstack【juno】入门 【网络篇】十七：neutron 安装部署（计算节点）

实例化网络结构

创建第一个实例之前，你必须创建实例连接所必需的虚拟网络基础设施，包括外网和租户网络。 参见图 6.1, “Initial networks”.创建完虚拟网络基础设施之后，我们建议验证网络连接性，在继续之前解决遇到的任何问题。图 6.1“Initial networks”可以看到初始化网络组件，展示了实例与外网或则 Internet 之间的流通。

图 6.1. Initial networks



外部网络

外部网络通常提供互联网访问实例。实例使用 NAT 允许互联网访问。你可以使用互联网访问内部实例，通过使用 floating IP 地址，和合适的安全组规则。admin 租户有这个网络，因为它提供外部网络访问多租户。注意：

执行下面命令在控制节点

创建外部网络

1.生效环境变量

```
1.  source admin-openrc.sh
```

2.创建网络

```
1.  neutron net-create ext-net --router:external True \
2.    --provider:physical_network external --provider:network_type flat
```

```
root@controller:~# neutron net-create ext-net --router:external True \
> --provider:physical_network external --provider:network_type flat
Created a new network:
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 0028b1d7-07c9-468c-abf2-ccd4dc17b240 |
| name                      | ext-net                              |
| provider:network_type     | flat                                 |
| provider:physical_network | external                             |
| provider:segmentation_id  |                                      |
| router:external           | True                                 |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | c52f15ccf48741c08fa8ba53b8b8ce01     |
+---------------------------+--------------------------------------+
root@controller:~#
```

```
1.  Created a new network:
2.  +---------------------------+--------------------------------------+
3.  | Field                     | Value                                |
4.  +---------------------------+--------------------------------------+
5.  | admin_state_up            | True                                 |
6.  | id                        | 0028b1d7-07c9-468c-abf2-ccd4dc17b240 |
7.  | name                      | ext-net                              |
8.  | provider:network_type     | flat                                 |
9.  | provider:physical_network | external                             |
10. | provider:segmentation_id  |                                      |
11. | router:external           | True                                 |
12. | shared                    | False                                |
13. | status                    | ACTIVE                               |
14. | subnets                   |                                      |
```

```
| tenant_id                | c52f15ccf48741c08fa8ba53b8b8ce01     |
+--------------------------+--------------------------------------+
```

就像一个物理网络，虚拟网络需要的子网分配给它的。这个网络与网络节点连接外网的物理网卡共享同一个子网和网关
你应该指定路由子网的 ip 地址范围及 floating ip 地址，以防止干扰外网的其它网络设备。

创建外网的子网

创建一个子网,形式如下：

```
neutron subnet-create ext-net --name ext-subnet \
  --allocation-pool start=FLOATING_IP_START,end=FLOATING_IP_END \
  --disable-dhcp --gateway EXTERNAL_NETWORK_GATEWAY EXTERNAL_NETWORK_CIDR
```

替换掉 FLOATING_IP_START，FLOATING_IP_END，分别是 floating ip 地址的开始地址和结束地址。替换掉 EXTERNAL_NETWORK_CIDR 子网关联的物理网络。替换 EXTERNAL_NETWORK_GATEWAY 与物理网络的网关。通常.1ip 地址。禁用子网 ip 地址，因为实例不直接连接外网，floating ip 需要手工分配。

举例：，使用 203.0.113.0/24 ， floating IP 地址范围 203.0.113.101 to 203.0.113.200:

```
neutron subnet-create ext-net --name ext-subnet \
  --allocation-pool start=203.0.113.101,end=203.0.113.200 \
  --disable-dhcp --gateway 203.0.113.1 203.0.113.0/24
```

```
root@controller:~# neutron subnet-create ext-net --name ext-subnet \
>   --allocation-pool start=203.0.113.101,end=203.0.113.200 \
>   --disable-dhcp --gateway 203.0.113.1 203.0.113.0/24
Created a new subnet:
+-------------------+-----------------------------------------------------+
| Field             | Value                                               |
+-------------------+-----------------------------------------------------+
| allocation_pools  | {"start": "203.0.113.101", "end": "203.0.113.200"} |
| cidr              | 203.0.113.0/24                                      |
| dns_nameservers   |                                                     |
| enable_dhcp       | False                                               |
| gateway_ip        | 203.0.113.1                                         |
| host_routes       |                                                     |
| id                | e6fab309-a8ff-4385-8b32-c942434414cf                |
| ip_version        | 4                                                   |
| ipv6_address_mode |                                                     |
| ipv6_ra_mode      |                                                     |
| name              | ext-subnet                                          |
| network_id        | 0028b1d7-07c9-468c-abf2-ccd4dc17b240                |
| tenant_id         | c52f15ccf48741c08fa8ba53b8b8ce01                    |
+-------------------+-----------------------------------------------------+
```

```
1.  Created a new subnet:
2.  +-------------------+-----------------------------------------------------+
3.  | Field             | Value                                               |
4.  +-------------------+-----------------------------------------------------+
5.  | allocation_pools  | {"start": "203.0.113.101", "end": "203.0.113.200"} |
6.  | cidr              | 203.0.113.0/24                                      |
7.  | dns_nameservers   |                                                     |
8.  | enable_dhcp       | False                                               |
9.  | gateway_ip        | 203.0.113.1                                         |
10. | host_routes       |                                                     |
11. | id                | e6fab309-a8ff-4385-8b32-c942434414cf                |
12. | ip_version        | 4                                                   |
13. | ipv6_address_mode |                                                     |
14. | ipv6_ra_mode      |                                                     |
15. | name              | ext-subnet                                          |
16. | network_id        | 0028b1d7-07c9-468c-abf2-ccd4dc17b240                |
17. | tenant_id         | c52f15ccf48741c08fa8ba53b8b8ce01                    |
```

```
18. +------------------+----------------------------------------------------+
```

复制代码

创建租户网络

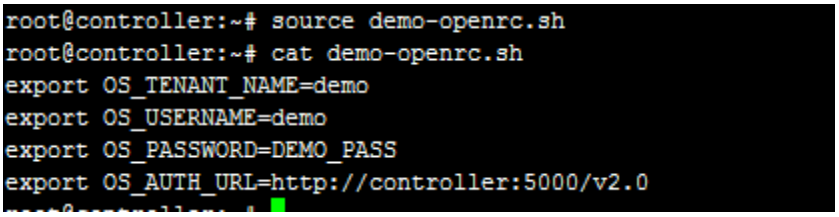
租户网络提供内部网络访问实例，这种结构使与其它租户分开。demo 租户拥有自己的网络，因为内部它提供网络访问实例。

注意：

在**控制节点**，执行下面命令

1.生效 demo 环境变量

```
1.  source demo-openrc.sh
```



demo-openrc.sh 的内容如下：

```
1.  export OS_TENANT_NAME=demo

2.  export OS_USERNAME=demo

3.  export OS_PASSWORD=DEMO_PASS

4.  export OS_AUTH_URL=http://controller:5000/v2.0
```

2.创建网络

```
1.  neutron net-create demo-net
```

```
root@controller:~# neutron net-create demo-net
Created a new network:
+-----------------+--------------------------------------+
| Field           | Value                                |
+-----------------+--------------------------------------+
| admin_state_up  | True                                 |
| id              | 299e4203-135e-4305-8a9f-cdd4f4bda185 |
| name            | demo-net                             |
| router:external | False                                |
| shared          | False                                |
| status          | ACTIVE                               |
| subnets         |                                      |
| tenant_id       | bfb31e69f05b44cd89e1336c09042e2f     |
+-----------------+--------------------------------------+
```

像外部网络一样，租户网络需要添加子网，你可以指定任何的有效子网，因为这个结构是租户网络结构隔离。默认子网使用 dhcp，因此你的实例，可以得到 ip 地址。

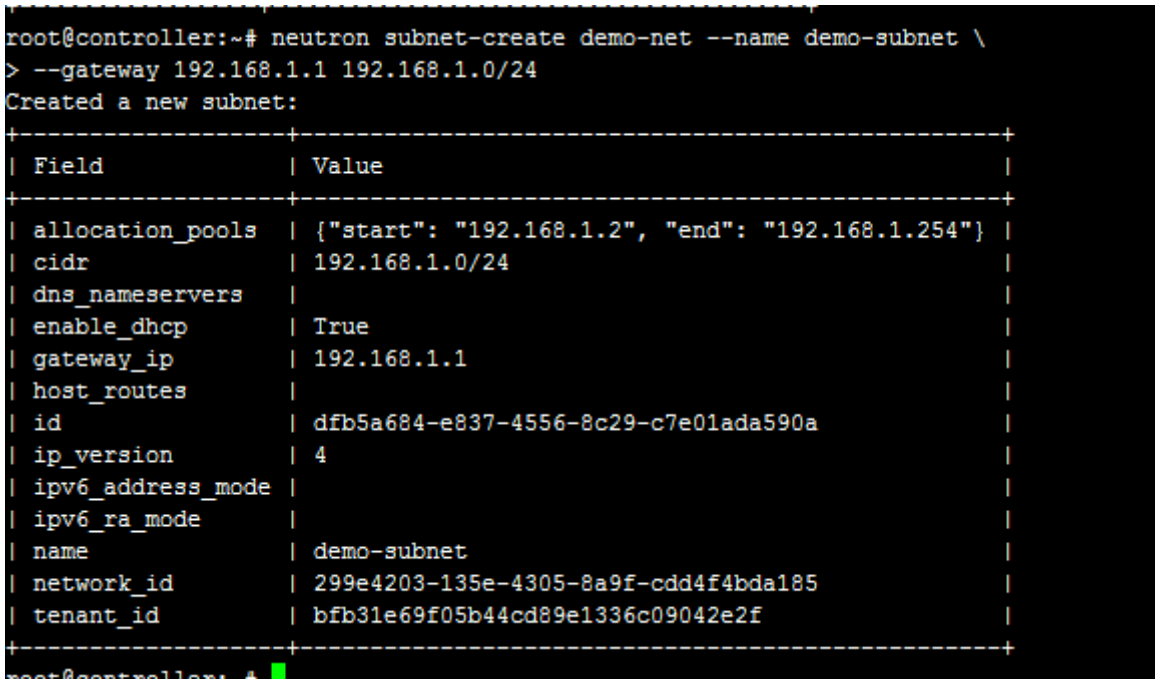
创建租户子网

创建子网

```
1.  neutron subnet-create demo-net --name demo-subnet \
2.    --gateway TENANT_NETWORK_GATEWAY TENANT_NETWORK_CIDR
```

替换子网 TENANT_NETWORK_CIDR 成你想更换的租户网络，和 TENANT_NETWORK_GATEWAY 替换成你想更换的网关。通常是.1ip 地址。
例如使用 192.168.1.0/24:

```
1.  neutron subnet-create demo-net --name demo-subnet \
2.    --gateway 192.168.1.1 192.168.1.0/24
```



```
root@controller:~# neutron subnet-create demo-net --name demo-subnet \
> --gateway 192.168.1.1 192.168.1.0/24
Created a new subnet:
+-------------------+------------------------------------------------------+
| Field             | Value                                                |
+-------------------+------------------------------------------------------+
| allocation_pools  | {"start": "192.168.1.2", "end": "192.168.1.254"}     |
| cidr              | 192.168.1.0/24                                       |
| dns_nameservers   |                                                      |
| enable_dhcp       | True                                                 |
| gateway_ip        | 192.168.1.1                                          |
| host_routes       |                                                      |
| id                | dfb5a684-e837-4556-8c29-c7e01ada590a                 |
| ip_version        | 4                                                    |
| ipv6_address_mode |                                                      |
| ipv6_ra_mode      |                                                      |
| name              | demo-subnet                                          |
| network_id        | 299e4203-135e-4305-8a9f-cdd4f4bda185                 |
| tenant_id         | bfb31e69f05b44cd89e1336c09042e2f                     |
+-------------------+------------------------------------------------------+
```

```
1.  Created a new subnet:
2.  +-------------------+------------------------------------------------------+
3.  | Field             | Value                                                |
4.  +-------------------+------------------------------------------------------+
5.  | allocation_pools  | {"start": "192.168.1.2", "end": "192.168.1.254"}     |
6.  | cidr              | 192.168.1.0/24                                       |
7.  | dns_nameservers   |                                                      |
8.  | enable_dhcp       | True                                                 |
9.  | gateway_ip        | 192.168.1.1                                          |
10. | host_routes       |                                                      |
11. | id                | dfb5a684-e837-4556-8c29-c7e01ada590a                 |
12. | ip_version        | 4                                                    |
13. | ipv6_address_mode |                                                      |
14. | ipv6_ra_mode      |                                                      |
15. | name              | demo-subnet                                          |
16. | network_id        | 299e4203-135e-4305-8a9f-cdd4f4bda185                 |
17. | tenant_id         | bfb31e69f05b44cd89e1336c09042e2f                     |
```

```
18. +-------------------+-----------------------------------------------+
```

一个或则多个网络由路由来连通。一个路由需要一个或则多个网卡和网关来提供访问指定的网络。这种情况下，你需要创建一个路由附加到租户和外部网络。

在租户网络创建路由和附加外网和租户网络到路由

1.创建路由

```
1.  neutron router-create demo-router
```

```
root@controller:~# neutron router-create demo-router
Created a new router:
+-----------------------+--------------------------------------+
| Field                 | Value                                |
+-----------------------+--------------------------------------+
| admin_state_up        | True                                 |
| external_gateway_info |                                      |
| id                    | 0481ede7-93e6-40c7-aa08-cc2c040a639e |
| name                  | demo-router                          |
| routes                |                                      |
| status                | ACTIVE                               |
| tenant_id             | bfb31e69f05b44cd89e1336c09042e2f     |
+-----------------------+--------------------------------------+
```

```
1.  root@controller:~# neutron router-create demo-router
2.  Created a new router:
3.  +-----------------------+--------------------------------------+
4.  | Field                 | Value                                |
5.  +-----------------------+--------------------------------------+
6.  | admin_state_up        | True                                 |
7.  | external_gateway_info |                                      |
8.  | id                    | 0481ede7-93e6-40c7-aa08-cc2c040a639e |
9.  | name                  | demo-router                          |
10. | routes                |                                      |
11. | status                | ACTIVE                               |
12. | tenant_id             | bfb31e69f05b44cd89e1336c09042e2f     |
13. +-----------------------+--------------------------------------+
```

2.附加路由到 demo 租户子网

```
1.  neutron router-interface-add demo-router demo-subnet
```

输出如下内容：

```
1.  Added interface 1dd67bdd-5a2c-416b-a3f7-25661f78ba57 to router demo-router.
```

3.连接路由器到外部网络通过设置为网关：

```
1.  neutron router-gateway-set demo-router ext-net
```

输出如下内容：

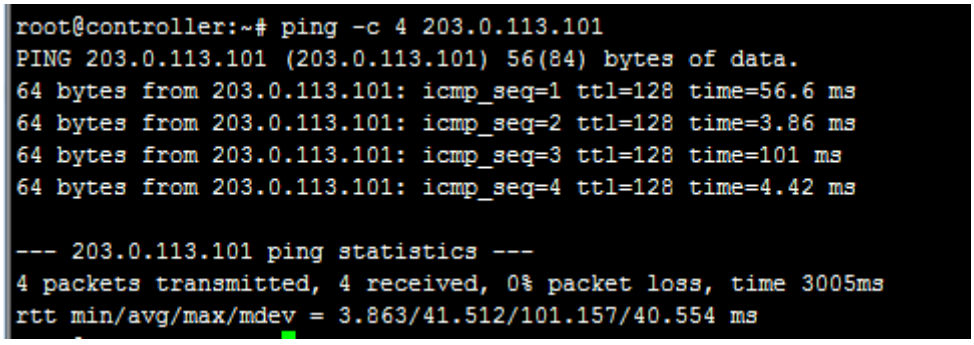
```
1.  Set gateway for router demo-router
```

验证安装

我们建议在进行下一步前，验证网络连通性，一下外部子网的例子使用 203.0.113.0/24,在 floating ip 地址范围内，路由网关应该是最低的 ip 地址， 203.0.113.101.如果你配置了虚拟机网络和外部物理网络正确，你应该可以 ping ip 地址从外部网络。说明：
如果你创建的 openstack 网络是虚拟机，你必须配置 hypervisor 允许在外部网络上设置混杂模式。

验证连连通性

ping 路由网关




```
1.  root@controller:~# ping -c 4 203.0.113.101
```

```
2.  PING 203.0.113.101 (203.0.113.101) 56(84) bytes of data.
3.  64 bytes from 203.0.113.101: icmp_seq=1 ttl=128 time=4.76 ms
4.  64 bytes from 203.0.113.101: icmp_seq=2 ttl=128 time=3.87 ms
5.  64 bytes from 203.0.113.101: icmp_seq=3 ttl=128 time=3.86 ms
6.  64 bytes from 203.0.113.101: icmp_seq=4 ttl=128 time=3.73 ms
7.
8.  --- 203.0.113.101 ping statistics ---
9.  4 packets transmitted, 4 received, 0% packet loss, time 3007ms
10. rtt min/avg/max/mdev = 3.737/4.059/4.765/0.415 ms
11. root@controller:~# ^C
12. root@controller:~# ping -c 4 203.0.113.101
13. PING 203.0.113.101 (203.0.113.101) 56(84) bytes of data.
14. 64 bytes from 203.0.113.101: icmp_seq=1 ttl=128 time=56.6 ms
15. 64 bytes from 203.0.113.101: icmp_seq=2 ttl=128 time=3.86 ms
16. 64 bytes from 203.0.113.101: icmp_seq=3 ttl=128 time=101 ms
17. 64 bytes from 203.0.113.101: icmp_seq=4 ttl=128 time=4.42 ms
18.
19. --- 203.0.113.101 ping statistics ---
20. 4 packets transmitted, 4 received, 0% packet loss, time 3005ms
21. rtt min/avg/max/mdev = 3.863/41.512/101.157/40.554 ms
```

在 window 下 ping

```
1.  ping 203.0.113.101
```


C:\Users\Administrator>ping 203.0.113.101
正在 Ping 203.0.113.101 具有 32 字节的数据:
来自 203.0.113.101 的回复: 字节=32 时间=3ms TTL=248
来自 203.0.113.101 的回复: 字节=32 时间=3ms TTL=248
来自 203.0.113.101 的回复: 字节=32 时间=3ms TTL=248
来自 203.0.113.101 的回复: 字节=32 时间=3ms TTL=248
203.0.113.101 的 Ping 统计信息:
数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
最短 = 3ms, 最长 = 3ms, 平均 = 3ms
C:\Users\Administrator>

openstack【juno】入门 【dashboard 篇】十九：添加 dashboard

问题导读

1.dashboard 的用户名和密码是如何产生的？
2.ubuntu14.04 是否满足 dashboard 安装要求？

由于 ubuntu 本身满足 dashboard 的安装要求，所以下面的安装要求，我们简单了解即可

内容：

openstack dashboard 也被称为 Horizon，是一个 web 界面，使用管理员和用户能够管理 openstack 不同的资源和服务
dashboard 通过 OpenStack APIs 操作 openstack 云计算控制器
Horizon 允许定制自己的商标
Horizon 提供了核心类和可重复使用的木板和工具
这个部署使用的是 Apache web server.

系统要求

在安装 openstack dashboard 之前，你必须满足下面需求

- openstack 计算安装，启用用户身份认证服务和项目管理
- 注意身份服务的 urls 和 Compute endpoints.
- 用户认证服务具有 sudo 的权限。因为 Apache 不能使用 root 用户服务，用户必须运行 dashboard 作为 sudo 权限身份认证服务。

- Python 2.6 或则 2.7，Python 版本必须支持 Django。Python 版本应该运行在任何系统，包括 Mac OS，不同的平台安装可能不一样。

然后在一个节点上安装和配置 dashboard 可以连接身份认证服务

用下面信息的用户，因此他们能够通过本地机器的浏览器访问 dashboard

- 通过公共 ip 地址，可以访问 dashboard
- 输入用户和密码

你的浏览器，必须支持 HTML5 ，启用 cookies 和 JavaScript

注意：
使用带有 dashboard 的 VNC 客户端，浏览器必须支持 HTML5 画布和 HTML5 WebSockets.

更多细节：
https://github.com/kanaka/noVNC/blob/master/README.md,
和 https://github.com/kanaka/noVNC/wiki/Browser-support,

安装和配置

这一部分描述了，在控制节点上如何安装和配置 dashboard 。
在开始之前，必须满足系统要求，dashboard 依赖的核心功能包括 Identity、 Image Service, Compute, 和 Networking (neutron) 或则 legacy networking (nova-network). 独立的服务比如 Object Storage，不能使用 dashboard。更多信息参考 developer documentation.
核实系统要求：

ubuntu 14.04 有自带 python

```
1.  python -V
```

```
aboutyun@controller:~$ python -V
Python 2.7.6
```

如果没有是正常 ubuntu，应该都满足系统要求

安装配置 dashboard 组件

```
1.  apt-get install openstack-dashboard apache2 libapache2-mod-wsgi memcached python-memcache
```

注意：Ubuntu 安装 openstack-dashboard-ubuntu-theme 包作为依赖。一些用户报告以前发布的版本有问题。如果你遇到问题，移除这个包，恢复到原先的 openstack 包。
移除可以执行下面命令（这里执行了下面命令）

```
1.  dpkg --purge openstack-dashboard-ubuntu-theme
2.
```

配置 dashboard

修改文件 /etc/openstack-dashboard/local_settings.py，完成下面内容

```
1.  sudo nano  /etc/openstack-dashboard/local_settings.py
```

a.配置 openstack 服务 dashboard，运行在控制节点

```
1.  OPENSTACK_HOST = "controller"
```

b.允许任何主机访问 dashboard

```
1.  ALLOWED_HOSTS = ['*']
```

c.配置缓存会话存储服务：

```
1.  CACHES = {
2.     'default': {
3.         'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
4.         'LOCATION': '127.0.0.1:11211',
5.     }
6.  }
```

注意：注释掉其它 session 存储配置
这里不需要修改

d.可选，配置其它 time zone

```
1.  TIME_ZONE = "TIME_ZONE"
```

替换 TIME_ZONE 为一个适当时区标准，更多信息查看 list of time zones.这里不需要修改，默认为

```
1.  TIME_ZONE = "UTC"
```

完成安装

重启 web server 和 session 存储服务:

```
1.  service apache2 restart
```

注意：记得这个是在非 root 用户下

```
1.  aboutyun@controller:~$ service apache2 restart

2.  * Restarting web server apache2
```

```
aboutyun@controller:~$ service apache2 restart
 * Restarting web server apache2
```

注意：下面命令是在 root 用户下

```
1.

2.  service memcached restart
```

验证安装

1.在浏览器输入下面网址

```
1.  http://controller/horizon
```

如果没有配置 hosts，在输入 ip 地址

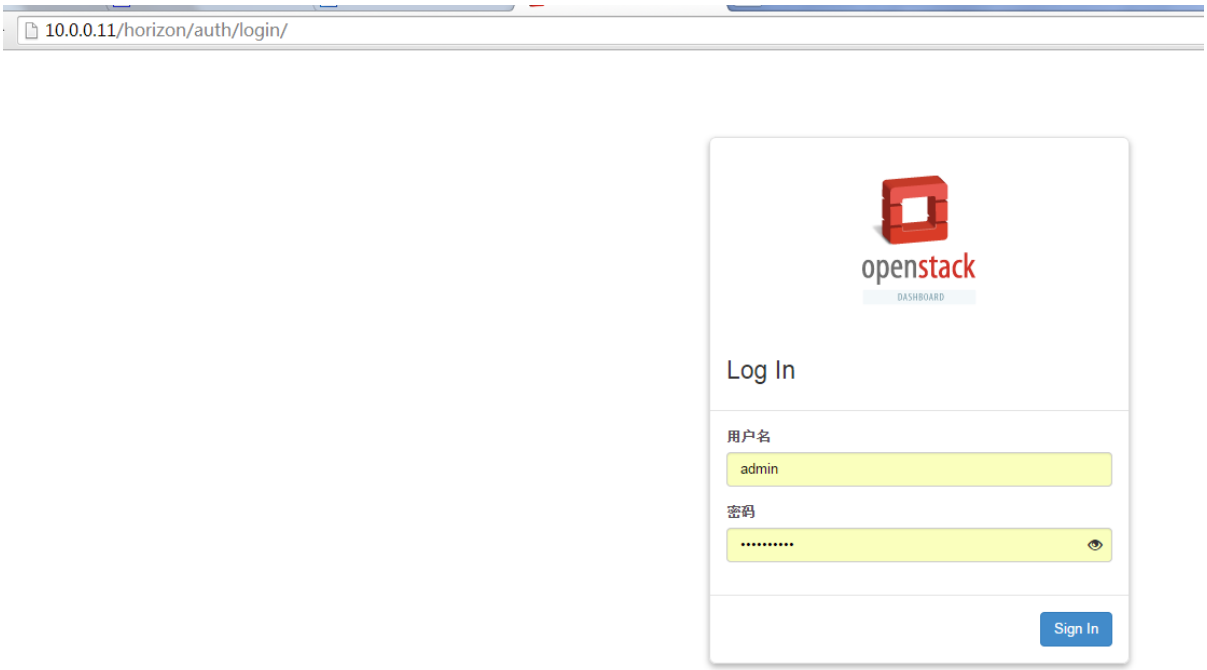
2.使用 admin 或则 demo 用户登录

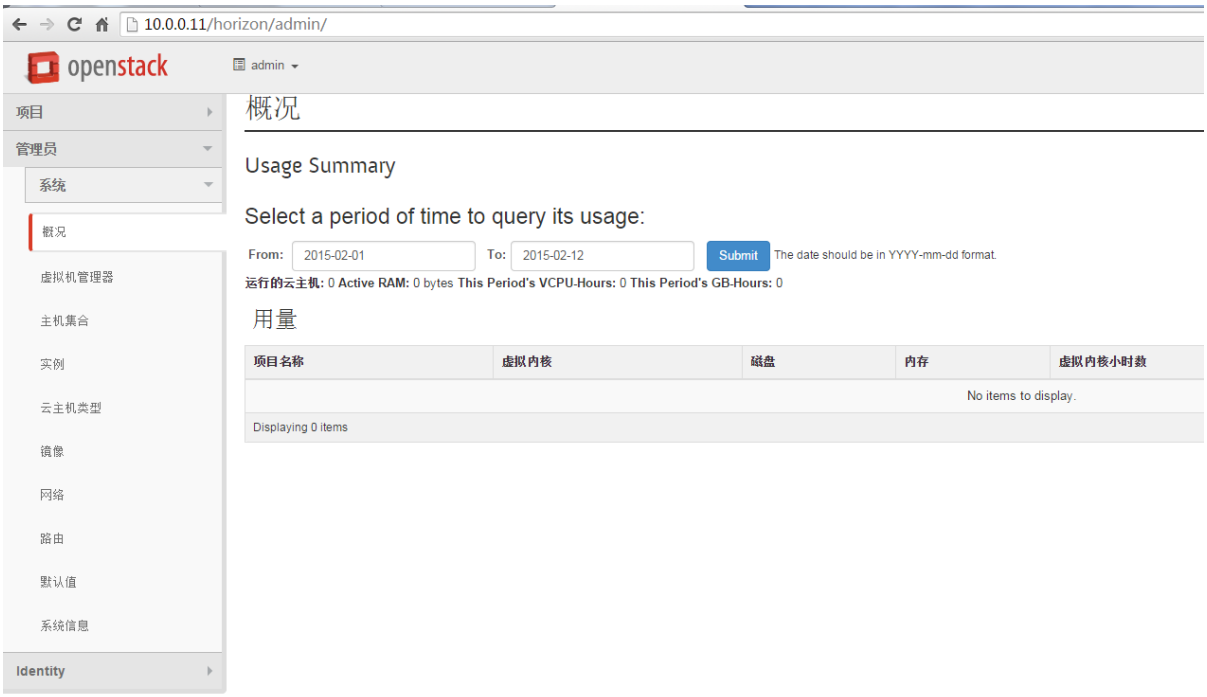
信息如下

```
1.  export OS_TENANT_NAME=admin

2.  export OS_USERNAME=admin

3.  export OS_PASSWORD=ADMIN_PASS

4.  export OS_AUTH_URL=http://controller:35357/v2.0

5.

6.
```

```
#export OS_TENANT_NAME=demo
#export OS_USERNAME=demo
#export OS_PASSWORD=DEMO_PASS
#export OS_AUTH_URL=http://controller:5000/v2.0
```

如下图所示：





注释：很多新手不知道这个用户名和密码是怎么来的，这个是由 keystone 来创建的。命令如下：

```
1.  keystone user-create --name admin --pass ADMIN_PASS --email EMAIL_ADDRESS
```

详细参考

登录 openstack 界面 dashboard 的用户名和密码是什么？

openstack【juno】入门 【cinder 篇】二十：cinder 介绍及安装配置【控制节点】

问题导读

1.OpenStack Block Storage 又被称为什么？

2.块存储服务为什么需要两个不同的服务实例？

3.两个服务实例与 api endpoint 之间是什么关系？

接上篇 openstack【juno】入门 【dashboard 篇】十九：添加 dashboard

内容：

- OpenStack Block Storage（cinder）
- 在控制节点安装配置
- 安装配置 cinder node

OpenStack Block 存储服务为实例通过不同后台提供 block 存储设备。 Block 存储 API 和调度服务运行在控制节点。volume 服务运行在一个或多个存储节点。cinder 为实例提供本地存储或则 SAN/NAS 后台适当的驱动。更多信息查看 Configuration Reference.

注意：
本章没有备份 manager，因为他依赖于对象服务。

OpenStack 块存储

openstack 块存储服务（cinder）为虚拟机增加持久性存储，块存储为管理卷提供基础设施，与 openstack 计

算提供的实例存储相互作用。这个也启用了 volume 快照管理，和 volume 类型。

块存储由以下内容组成：

cinder-api

接受一些请求，并把他们发送给 cinder-volume

cinder-volume

直接与块存储服务相互作用，处理比如 cinder-scheduler（cinder 调度），它通过消息队列直接与这些过程相互作用。cinder-volume 服务响应读写请求。cinder-volume 通过驱动与可以使用不同供应商的存储设置

cinder-scheduler 守护进程

选择最优节点存储创建卷，与 nova-scheduler 组件类似

消息队列

块存储之间传递信息

安装配置控制节点

这里描述怎么安装和配置存储服务，代号为 cinder，在控制节点上。这个服务至少需要一个存储节点为实例提供卷。

准备

在安装配置 cinder 之前，必须创建数据库、服务认证、和 API endpoints

1.创建数据库，完成下面内容

a.进入 mysql

```
1.  mysql -u root -p
```

b.创建 cinder 数据库

```
1.  CREATE DATABASE cinder;
```

c.授权

```
1.  GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' \
2.    IDENTIFIED BY 'CINDER_DBPASS';
3.  GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' \
4.    IDENTIFIED BY 'CINDER_DBPASS';
```

CINDER_DBPASS 可以自定义

d.退出 mysql

```
1.  exit
```

操作如下：

```
MariaDB [(none)]> CREATE DATABASE cinder;
Query OK, 1 row affected (0.00 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' \
    -> IDENTIFIED BY 'CINDER_DBPASS';
Query OK, 0 rows affected (0.15 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' \
    -> IDENTIFIED BY 'CINDER_DBPASS';
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> exit
Bye
aboutyun@controller:~$
```

2.生效环境变量

```
1.  source admin-openrc.sh
```

3.创建 keystone 认证

a.创建 cinder 用户

```
1.  keystone user-create --name cinder --pass CINDER_PASS
```

```
root@controller:~# keystone user-create --name cinder --pass CINDER_PASS
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |                                  |
| enabled  |               True               |
|    id    | c8f92fac64824cd0b581856eb72b63b6 |
|   name   |              cinder              |
| username |              cinder              |
+----------+----------------------------------+
root@controller:~#
```

b.给 cinder 用户授予 admin 角色

```
1.  keystone user-role-add --user cinder --tenant service --role admin
```

这里不会有输出内容，所以看不出输出信息

c.创建 cinder 服务实例

```
1.  keystone service-create --name cinder --type volume \
2.    --description "OpenStack Block Storage"
```

```
root@controller:~# keystone service-create --name cinder --type volume \
> --description "OpenStack Block Storage"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |     OpenStack Block Storage      |
|   enabled   |               True               |
|      id     | c2bdacccb5fc48fd9fe447c4a5ae430f |
|     name    |              cinder              |
|     type    |              volume              |
+-------------+----------------------------------+
```

```
1.  keystone service-create --name cinderv2 --type volumev2 \
2.    --description "OpenStack Block Storage"
```

```
root@controller:~# keystone service-create --name cinderv2 --type volumev2 \
>   --description "OpenStack Block Storage"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |     OpenStack Block Storage      |
|   enabled   |               True               |
|      id     | e165f4099ac6460f9a0f85b35b6cb1d0 |
|     name    |             cinderv2             |
|     type    |             volumev2             |
+-------------+----------------------------------+
```

说明：
块存储服务需要两个不同的服务实例支持 api 版本 1 和版本 2

4.创建块存储服务 API endpoints

```
keystone endpoint-create \
  --service-id $(keystone service-list | awk '/ volume / {print $2}') \
  --publicurl http://controller:8776/v1/%\(tenant_id\)s \
  --internalurl http://controller:8776/v1/%\(tenant_id\)s \
  --adminurl http://controller:8776/v1/%\(tenant_id\)s \
  --region regionOne
```

```
root@controller:~# keystone endpoint-create \
> --service-id $(keystone service-list | awk '/ volume / {print $2}') \
> --publicurl http://controller:8776/v1/%\(tenant_id\)s \
> --internalurl http://controller:8776/v1/%\(tenant_id\)s \
> --adminurl http://controller:8776/v1/%\(tenant_id\)s \
> --region regionOne
+-------------+-----------------------------------------+
|   Property  |                  Value                  |
+-------------+-----------------------------------------+
|   adminurl  | http://controller:8776/v1/%(tenant_id)s |
|      id     |     77d7efa9c1114a3a8a6f17dc958ff9f9    |
| internalurl | http://controller:8776/v1/%(tenant_id)s |
|  publicurl  | http://controller:8776/v1/%(tenant_id)s |
|    region   |                regionOne                |
|  service_id |     c2bdacccb5fc48fd9fe447c4a5ae430f    |
+-------------+-----------------------------------------+
```

```
keystone endpoint-create \
  --service-id $(keystone service-list | awk '/ volumev2 / {print $2}') \
  --publicurl http://controller:8776/v2/%\(tenant_id\)s \
  --internalurl http://controller:8776/v2/%\(tenant_id\)s \
  --adminurl http://controller:8776/v2/%\(tenant_id\)s \
  --region regionOne
```

```
root@controller:~# keystone endpoint-create \
>   --service-id $(keystone service-list | awk '/ volumev2 / {print $2}') \
>   --publicurl http://controller:8776/v2/%\(tenant_id\)s \
>   --internalurl http://controller:8776/v2/%\(tenant_id\)s \
>   --adminurl http://controller:8776/v2/%\(tenant_id\)s \
>   --region regionOne
+-------------+-----------------------------------------+
|   Property  |                  Value                  |
+-------------+-----------------------------------------+
|   adminurl  | http://controller:8776/v2/%(tenant_id)s |
|      id     |     f88f3d80ba17439b8a1fe6ec69ab118c    |
| internalurl | http://controller:8776/v2/%(tenant_id)s |
|  publicurl  | http://controller:8776/v2/%(tenant_id)s |
|    region   |                regionOne                |
|  service_id |     e165f4099ac6460f9a0f85b35b6cb1d0    |
+-------------+-----------------------------------------+
```

说明：

块存储服务需要两个不同的 endpoints ， 支持 api 版本 1 和版本 2

安装和配置块存储控制器组件

1.下载并安装

```
1.  apt-get install cinder-api cinder-scheduler python-cinderclient
```

2.编辑文件 /etc/cinder/cinder.conf，完成下面内容

```
1.  sudo nano  /etc/cinder/cinder.conf
```

a.在 [database]部分，配置数据库访问

```
1.  [database]
2.  ...
3.  connection = mysql://cinder:CINDER_DBPASS@controller/cinder
```

没有找到[database],这里直接添加

b.在 [DEFAULT]部分，配置 RabbitMQ 消息代理访问

```
1.  [DEFAULT]
2.  ...
3.  rpc_backend = rabbit
4.  rabbit_host = controller
5.  rabbit_password = RABBIT_PASS
```

RABBIT_PASS 这里使用的是默认密码

c.在 [DEFAULT] 和 [keystone_authtoken] 部分，配置认证访问

```
1.  [DEFAULT]
2.  ...
3.  auth_strategy = keystone
4.
5.  [keystone_authtoken]
6.  ...
7.  auth_uri = http://controller:5000/v2.0
8.  identity_uri = http://controller:35357
9.  admin_tenant_name = service
10. admin_user = cinder
11. admin_password = CINDER_PASS
```

注意：注释掉其他 auth_host, auth_port, 和 auth_protocol 选项，以免配置被覆盖。

d.在 [DEFAULT]部分，配置控制节点管理网络 ip 地址

```
1.  [DEFAULT]
2.  ...
3.  my_ip = 10.0.0.11
```

e.（可选）为排除故障，在 [DEFAULT]部分启用 verbose 日志记录

```
1.  [DEFAULT]

2.  ...

3.  verbose = True
```

3.同步数据库

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

```
root@controller:~# su -s /bin/sh -c "cinder-manage db sync" cinder
2015-02-13 03:21:02.661 22992 INFO migrate.versioning.api [-] 0 -> 1...
2015-02-13 03:21:03.998 22992 INFO migrate.versioning.api [-] done
2015-02-13 03:21:03.999 22992 INFO migrate.versioning.api [-] 1 -> 2...
2015-02-13 03:21:04.283 22992 INFO migrate.versioning.api [-] done
2015-02-13 03:21:04.284 22992 INFO migrate.versioning.api [-] 2 -> 3...
2015-02-13 03:21:04.421 22992 INFO migrate.versioning.api [-] done
2015-02-13 03:21:04.422 22992 INFO migrate.versioning.api [-] 3 -> 4...
2015-02-13 03:21:04.698 22992 INFO 004_volume_type_to_uuid [-] Created foreign key volume_type_extra_specs_ibfk_1
2015-02-13 03:21:04.702 22992 INFO migrate.versioning.api [-] done
2015-02-13 03:21:04.704 22992 INFO migrate.versioning.api [-] 4 -> 5...
2015-02-13 03:21:04.729 22992 INFO migrate.versioning.api [-] done
2015-02-13 03:21:04.730 22992 INFO migrate.versioning.api [-] 5 -> 6...
2015-02-13 03:21:04.752 22992 INFO migrate.versioning.api [-] done
2015-02-13 03:21:04.753 22992 INFO migrate.versioning.api [-] 6 -> 7...
2015-02-13 03:21:04.879 22992 INFO migrate.versioning.api [-] done
2015-02-13 03:21:04.879 22992 INFO migrate.versioning.api [-] 7 -> 8...
2015-02-13 03:21:04.895 22992 INFO migrate.versioning.api [-] done
2015-02-13 03:21:04.896 22992 INFO migrate.versioning.api [-] 8 -> 9...
2015-02-13 03:21:04.935 22992 INFO migrate.versioning.api [-] done
2015-02-13 03:21:04.936 22992 INFO migrate.versioning.api [-] 9 -> 10...
2015-02-13 03:21:04.958 22992 INFO migrate.versioning.api [-] done
2015-02-13 03:21:04.958 22992 INFO migrate.versioning.api [-] 10 -> 11...
2015-02-13 03:21:04.989 22992 INFO migrate.versioning.api [-] done
2015-02-13 03:21:04.989 22992 INFO migrate.versioning.api [-] 11 -> 12...
```

完成安装

1.重启服务

```
1.  service cinder-scheduler restart

2.  service cinder-api restart
```

```
root@controller:~# service cinder-scheduler restart
cinder-scheduler stop/waiting
cinder-scheduler start/running, process 23018
root@controller:~# service cinder-api restart
cinder-api stop/waiting
cinder-api start/running, process 23040
```

2.如果有 SQLite 数据库，则移除

```
1.  rm -f /var/lib/cinder/cinder.sqlite
```

openstack【juno】入门 【cinder 篇】二十一：安装配置块存储节点（cinder）

问题导读

1.如何让 Linux 新增硬盘生效被 openstak cinder 使用？
2.对于任何一个新增的 openstack 节点，如果不安装 openstack 包，是否可以？
3.如何验证创建的 volume 是否有效？

接上篇：openstack【juno】入门 【cinder 篇】二十：cinder 介绍及安装配置【控制节点】

准备

这一部分描述怎么样安装配置 cinder，为了简单起见，这个配置引用了空的本地存储设备 /dev/sdb 的存储节点。它包含一个分区表和一个分区/dev/sdb1 暂满整个设备。这个规定使用 LVM 驱动的逻辑卷。通过 iSCSI 提供给实例。按照下面说明，做一些改动。

安装 openstack 包

```
1.  apt-get install ubuntu-cloud-keyring
```

```
1.  echo "deb http://ubuntu-cloud.archive.canonical.com/ubuntu" \
2.    "trusty-updates/juno main" > /etc/apt/sources.list.d/cloudarchive-juno.list
```

更新源

```
1.  apt-get update && apt-get dist-upgrade
```

####################################

1.配置管理网络网卡

IP address: 10.0.0.41

```
1.  sudo nano /etc/network/interfaces
```

这里配置如下：

```
1.  auto eth0
2.  iface eth0 inet static
3.  address 10.0.0.41
4.  netmask 255.255.255.0
5.  gateway 10.0.0.2
6.  dns-nameservers 8.8.8.8
```

2.修改 hosts 为 block

```
1.  sudo nano /etc/hosts
```

修改如下：

```
1.  127.0.0.1       localhost
2.  #127.0.1.1      ubuntu
3.  10.0.0.41 block
```

3.修改 hostname

```
1.  sudo nano /etc/hostname
```

内容：

```
1.  block
```

4.重启

修改完毕，重启

```
1.  sudo reboot
```

5.安装 NTP

```
1.  apt-get install ntp
```

修改配置文件

```
1.  sudo nano  /etc/ntp.conf
```

添加如下内容：

```
1.  server controller iburst
```

（如果没有配置 host,则直接写 ip 地址，10.0.0.11 ）

其他 server 注释掉

删除 ntp.conf.dhcp

为防止 ntp.conf.dhcp 文件存在，执行下面命令

```
1.  rm /var/lib/ntp/ntp.conf.dhcp
```

重启 NTP

```
1.  service ntp restart
```

查看

```
1.  ntpq -c peers
```

```
root@block:~# ntpq -c peers
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 10.0.0.11       .INIT.          16 u  126 1024    0    0.000    0.000   0.000
```

6.安装 LVM 包

创建物理卷/dev/sdb1

```
1.  pvcreate /dev/sdb1
```

```
root@block:~# pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created
```

这里如果遇到 Device /dev/sdb1 not found (or ignored by filtering)？，查看

openstack【juno】入门 【cinder 外篇】Device /dev/sdb1 not found (or ignored byfiltering)

需要补充的知识：
openstack 存储基础 1：浅谈 Linux 磁盘存储管理

openstack 存储基础 2：Linux LVM 硬盘管理及 LVM 扩容

当然上面你也可以使用其它 device。

7.创建 LVM 卷组 cinder-volumes

```
1.  vgcreate cinder-volumes /dev/sdb1
```

```
root@block:~# vgcreate cinder-volumes /dev/sdb1
  Volume group "cinder-volumes" successfully created
root@block:~#
```

存储节点服务在卷组里创建逻辑卷

8.lvm 扫描修改

只有实例才能访问块存储卷,底层操作系统管理关于卷的管理设备，默认 LVM 卷扫描工具会为块存储扫描 /dev 目录。如果租户使用 lvm，扫描工具会检测这些卷和缓存他们，这会造成各种各样的问题，必须重新配置 lvm 扫描设备。

编辑 /etc/lvm/lvm.conf 文件完成下面内容

```
1.  sudo nano  /etc/lvm/lvm.conf
```

a.在 devices 部分，添加一个筛选器接受/dev/sdb 设备并拒绝所有其他设备：

```
1.  devices {
2.  ...
3.  filter = [ "a/sdb/", "r/.*/"]
```

```
devices {

    # Where do you want your volume groups to appear ?
    dir = "/dev"

    filter = [ "a/sdb/", "r/.*/"]
    # An array of directories that contain the device nodes you wish
    # to use with LVM2.
    scan = [ "/dev" ]

    # If set, the cache of block device nodes with all associated symlinks
    # will be constructed out of the existing udev database content.
    # This avoids using and opening any inapplicable non-block devices or
    # subdirectories found in the device directory. This setting is applied
    # to udev-managed device directory only, other directories will be scanned
    # fully. LVM2 needs to be compiled with udev support for this setting to
```

说明：

array 数组里包含可以访问和拒绝选项，里面使用正则表达式来表示。这个 array 数组，必须以 r/.*/结束拒绝其他设备。你可以使用 vgs -vvvv 命令，测试过滤器。

提示：

如果你的存储节点底层操作系统使用 LVM，也必须添加相关的设备到过滤器。例如：如果操作系统包含 /dev/sda 设备，

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

类似如下：

如果你的计算节点底层操作系统使用 LVM，你必须修改 /etc/lvm/lvm.conf 文件，在这些底层操作系统磁盘的节点上，例如操作系统包含/dev/sda

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

安装配置块存储卷组件

1.安装

```
1.  apt-get install cinder-volume python-mysqldb -y
```

2.编辑文件 /etc/cinder/cinder.conf 完成下面内容

```
1.  sudo nano /etc/cinder/cinder.conf
```

a.在[database]部分，配置数据库访问

```
1.  [database]
2.  ...
3.  connection = mysql://cinder:CINDER_DBPASS@controller/cinder
```

密码需要替换成自己的密码，这里使用的是默认。

b.在 [DEFAULT]部分，配置 RabbitMQ 消息代理

```
1.  [DEFAULT]
2.  ...
3.  rpc_backend = rabbit
4.  rabbit_host = controller
5.  rabbit_password = RABBIT_PASS
```

c.在[DEFAULT] 和 [keystone_authtoken] 部分，配置身份认证访问

```
1.  [DEFAULT]
2.  ...
3.  auth_strategy = keystone
4.
5.  [keystone_authtoken]
6.  ...
7.  auth_uri = http://controller:5000/v2.0
8.  identity_uri = http://controller:35357
9.  admin_tenant_name = service
10. admin_user = cinder
11. admin_password = CINDER_PASS
```

同样这里使用的是默认密码 CINDER_PASS，如果自定义，则需要替换。

注意：

替换掉 auth_host, auth_port, 和 auth_protocol，防止配置被覆盖。

d.在[DEFAULT]部分，配置 my_ip 选项

```
1.  [DEFAULT]

2.  ...

3.  my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
```

替换为下面形式

```
1.  [DEFAULT]

2.  ...

3.  my_ip = 10.0.0.41
```

```
root@block:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:50:56:20:c4:db brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.41/24 brd 10.0.0.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe20:c4db/64 scope link
       valid_lft forever preferred_lft forever
```

e.在 [DEFAULT] 部分，配置 image 服务的位置

```
1.  [DEFAULT]

2.  ...

3.  glance_host = controller
```

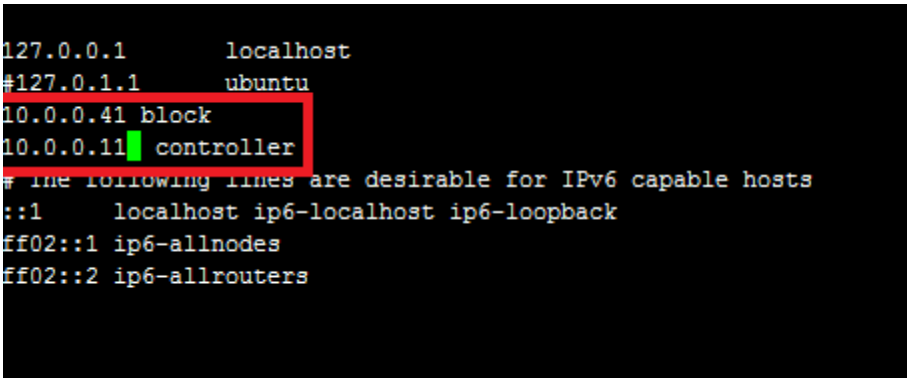
记得这里配置 hosts，如果之前没有配置，则补上

##

打开文件

```
1.  sudo nano /etc/hosts
```

添加如下即可



##

f.为排除故障，启用 verbose logging 在[DEFAULT] 部分

```
1. [DEFAULT]
2. ...
3. verbose = True
```

##############################

贴出配置文件如下

```
1. [DEFAULT]
2. rootwrap_config = /etc/cinder/rootwrap.conf
3. api_paste_confg = /etc/cinder/api-paste.ini
4. iscsi_helper = tgtadm
5. volume_name_template = volume-%s
6. volume_group = cinder-volumes
7. verbose = True
8. auth_strategy = keystone
9. state_path = /var/lib/cinder
10. lock_path = /var/lock/cinder
11. volumes_dir = /var/lib/cinder/volumes
12.
13. rpc_backend = rabbit
14. rabbit_host = controller
15. rabbit_password = RABBIT_PASS
16.
```

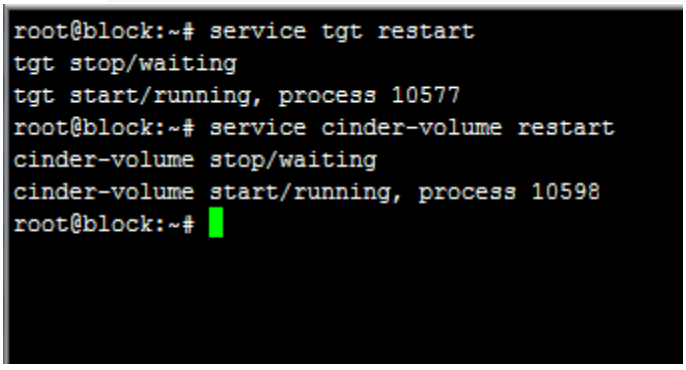
```
auth_strategy = keystone

my_ip = 10.0.0.41

glance_host = controller

verbose = True

[database]
connection = mysql://cinder:CINDER_DBPASS@controller/cinder
[keystone_authtoken]
auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357
admin_tenant_name = service
admin_user = cinder
admin_password = CINDER_PASS
```

##

完成安装

1.重启块存储 volume 服务，包括它的依赖

```
service tgt restart
service cinder-volume restart
```



2.SQLite 如果存在，则移除

```
1.  rm -f /var/lib/cinder/cinder.sqlite
```

验证安装

这里描述了怎样验证块存储服务创建 volume，注意执行下面命令在**控制节点**

1.环境变量生效

```
1.  source admin-openrc.sh
```

内容如下

```
1.  export OS_TENANT_NAME=admin

2.  export OS_USERNAME=admin

3.  export OS_PASSWORD=ADMIN_PASS

4.  export OS_AUTH_URL=http://controller:35357/v2.0
```

2.列出安装成功的服务组件

```
1.  cinder service-list
```

```
root@controller:~# cinder service-list
+------------------+------------+------+---------+-------+----------------------------+-----------------+
|      Binary      |    Host    | Zone |  Status | State |         Updated_at         | Disabled Reason |
+------------------+------------+------+---------+-------+----------------------------+-----------------+
| cinder-scheduler | controller | nova | enabled |   up  | 2015-02-16T16:31:53.000000 |       None      |
|  cinder-volume   |   block    | nova | enabled |   up  | 2015-02-16T16:31:52.000000 |       None      |
+------------------+------------+------+---------+-------+----------------------------+-----------------+
```

3.生效 **demo** 租户，作为一个非管理员租户执行下面步骤

```
1.  source demo-openrc.sh
```

内容如下：

```
1.  export OS_TENANT_NAME=demo

2.  export OS_USERNAME=demo

3.  export OS_PASSWORD=DEMO_PASS

4.  export OS_AUTH_URL=http://controller:5000/v2.0
```

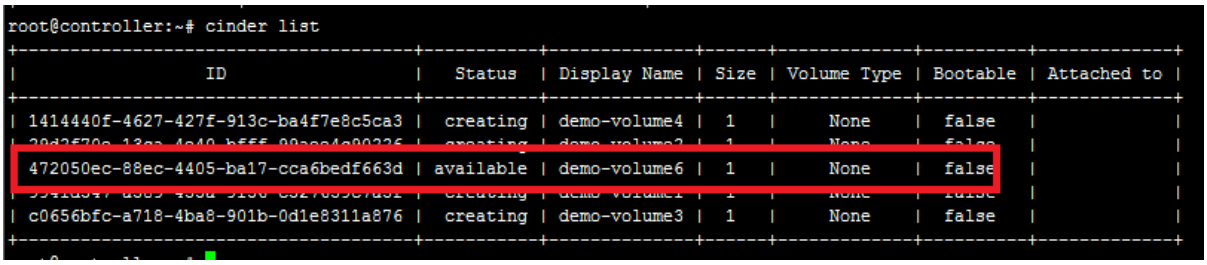
4.创建一个 1 GB volume:

```
1.  cinder create --display-name demo-volume6 1
```

```
root@controller:~#  cinder create --display-name demo-volume6 1
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|     attachments     |                  []                  |
|  availability_zone  |                nova                  |
|       bootable      |               false                  |
|      created_at     |      2015-02-17T05:21:13.500164      |
| display_description |                None                  |
|     display_name    |             demo-volume6             |
|      encrypted      |               False                  |
|          id         | 472050ec-88ec-4405-ba17-cca6bedf663d |
|       metadata      |                  {}                  |
|         size        |                  1                   |
|     snapshot_id     |                None                  |
|     source_volid    |                None                  |
|        status       |              creating                |
|     volume_type     |                None                  |
+---------------------+--------------------------------------+
```

5.验证 volume 有效性

```
1.  cinder list
```



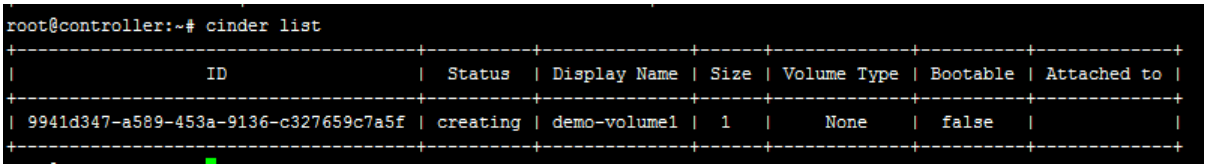
由于刚开始没有安装 openstack 包，所以导致前面创建错误，后来补上，问题得到解决

##

6.出现错误：

问题 1：openstack 包未安装

```
1.  cinder list
1.  +--------------------------------------+----------+--------------+------+-------------
    +----------+-------------+
2.  |                  ID                  |  Status  | Display Name | Size | Volume Type |
    Bootable | Attached to |
3.  +--------------------------------------+----------+--------------+------+-------------
    +----------+-------------+
4.  | 9941d347-a589-453a-9136-c327659c7a5f | creating | demo-volume1
    |  1   |     None    |  false   |             |
5.  +--------------------------------------+----------+--------------+------+-------------
    +----------+-------------+
```



查看日志：

```
1.  2015-02-16 09:26:13.585 1659 ERROR oslo.messaging._drivers.common
    [req-0d06d502-d895-4c89-8f47-dde86f696bcc 9695ee1a49a342aa9a78890896b99605
```

```
bfb31e69f05b44cd89e1336c09042e2f - - -] ['Traceback (most recent call last):\n', '  File
"/usr/lib/python2.7/dist-packages/oslo/messaging/rpc/dispatcher.py", line 133, in
_dispatch_and_reply\n    incoming.message))\n', '  File
"/usr/lib/python2.7/dist-packages/oslo/messaging/rpc/dispatcher.py", line 176, in
_dispatch\n    return self._do_dispatch(endpoint, method, ctxt, args)\n', '  File
"/usr/lib/python2.7/dist-packages/oslo/messaging/rpc/dispatcher.py", line 122, in
_do_dispatch\n    result = getattr(endpoint, method)(ctxt, **new_args)\n', "TypeError:
create_volume() got an unexpected keyword argument 'source_replicaid'\n"]
```

经过查找资料：

https://bugs.launchpad.net/cinder/+bug/1389687

原因为没有安装 Ubuntu 云 archive keyring 和 库:

解决办法：

```
1.  apt-get install ubuntu-cloud-keyring
1.  echo "deb http://ubuntu-cloud.archive.canonical.com/ubuntu" \
2.    "trusty-updates/juno main" > /etc/apt/sources.list.d/cloudarchive-juno.list
```

```
1.  apt-get update && apt-get dist-upgrade
```

问题 2：新增加硬盘未生效

在我们安装 cinder 的过程中：官网地址

在执行到创建 LVM 物理卷：

```
1.  pvcreate /dev/sdb1
```

遇到下面错误

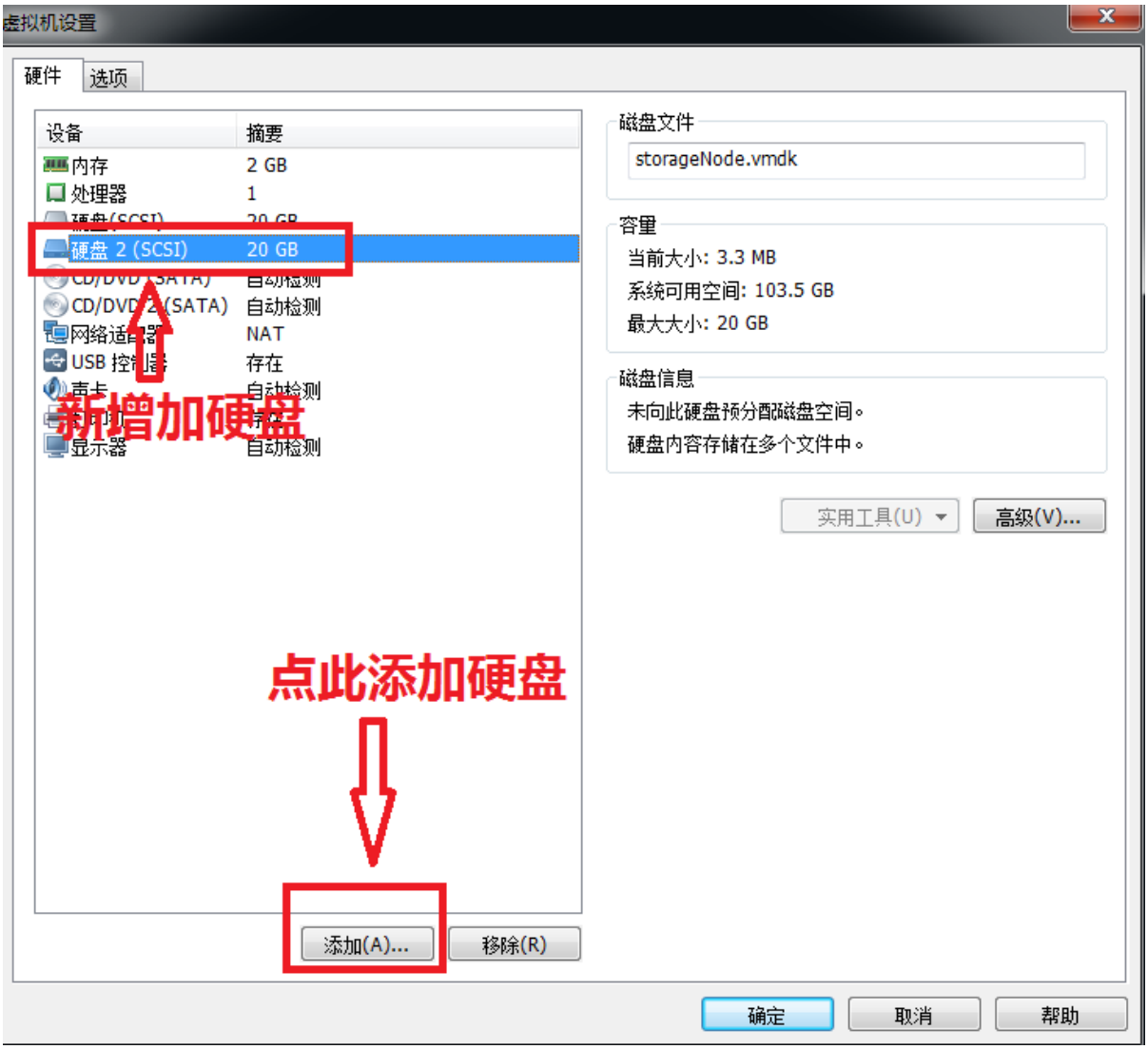
```
1.  Device /dev/sdb1 not found (or ignored by filtering).
```

这是由于我们只是挂载了硬盘，但是并没有格式化使用。所以下面交给大家怎么使用起来

```
1.  fdisk /dev/sdb1
2.  #Type in the followings:
3.  n
4.  p
5.  1
6.  ENTER
```

```
7.  ENTER

8.  t

9.  8e

10. w
```

对于上面的 /dev/sdb1 是如何来的那。这里由于使用的是虚拟机，是通过虚拟机增加的第二块硬盘：如下图：



增加硬盘之后，我们通过

```
1.  fdisk -l
```

命令，可以查看到

所以我们通过

下面命令，即可完成：

```
1.  fdisk /dev/sdb1
2.  #Type in the followings:
3.  n
4.  p
5.  1
6.  ENTER
7.  ENTER
8.  t
9.  8e
10. w
```

更多信息参考：

[openstack 存储基础 1：浅谈 Linux 磁盘存储管理]

openstack 存储基础 2：Linux LVM 硬盘管理及 LVM 扩容

openstack【juno】入门 【swift 篇】二十二：对象存储安装配置【控制节点】

问题导读

1.swift 包含哪些组件？
2.如果创建 swift 账户？
3.swift 能否单独使用？
4.如果下载 swift 文件？

openstack【juno】入门 【cinder 篇】二十一：安装配置块存储节点（cinder）

简介：

openstack swift 是一个多租户对象存储系统。它的高扩展性，以最小的成本管理大量的非结构化数据。它包括下面组件：

代理服务器【Proxy servers (swift-proxy-server)】

接受 openstack 对象存储 api 和 raw HTTP 请求上传文件，修改元数据，和创建容器。它也存储文件和通过浏览器查看列表。为提高性能，通常使用可选的高速缓存

账户服务器【Account servers (swift-account-server)】
管理对象存储账户

容器服务器【Container servers (swift-container-server)】
在 swift 中，管理容器或则文件的映射

对象服务器【Object servers (swift-object-server)】

管理实际对象，比如在存储节点的文件

Various periodic 进程
在大数量存储执行内部任务，replication 服务，确保集群一致性和高可用性。其它周期进程包括

WSGI 中间件

处理身份验证，通常是 OpenStack 身份。

部署向导

这一节描述怎么安装和配置代理服务，处理账户，container，对象服务操作请求，在存储节点上。为了简单起见，在控制节点上安装和配置代理服务。
尽管如此你可以在连接到存储节点的其它节点上安装代理服务。额外的，你可以安装配置代理服务到其它多节点，提高性能和冗余。更多信息查看

准备

代理服务依赖认证授权机制，比如身份认证服务。尽管如此，不像其他服务，它也提供内部认证机制，没有 openstack 其它服务也能进行操作。为了简单起见，这里使用认证服务。在配置对象服务之前，你必须创建认证服务和 API endpoints.

注意：
对象存储服务在控制节点不使用 SQL 数据库

1.创建身份认证服务，完成下面步骤

生效环境变量

```
1.  source admin-openrc.sh
```

内容如下：

```
1.  export OS_TENANT_NAME=admin

2.  export OS_USERNAME=admin

3.  export OS_PASSWORD=ADMIN_PASS

4.  export OS_AUTH_URL=http://controller:35357/v2.0
```

```
root@controller:~# cat admin-openrc.sh
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_AUTH_URL=http://controller:35357/v2.0
```

a.创建一个 swift 用户

```
1.  keystone user-create --name swift --pass SWIFT_PASS
```

```
root@controller:~# keystone user-create --name swift --pass SWIFT_PASS
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |                                  |
| enabled  |               True               |
|    id    | 4f72faf7fcca42cf868ae4fd9b30c40c |
|   name   |              swift               |
| username |              swift               |
+----------+----------------------------------+
```

b.授予 admin 角色

```
1.  keystone user-role-add --user swift --tenant service --role admin
```

执行命令后，这里没有输出内容

c.创建 swift 服务实例

```
1.  keystone service-create --name swift --type object-store \

2.    --description "OpenStack Object Storage"
```

```
root@controller:~# keystone service-create --name swift --type object-store \
>   --description "OpenStack Object Storage"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |     OpenStack Object Storage     |
|   enabled   |               True               |
|      id     | 8e10fbbb9ed34e2e8207b7697a1322a8 |
|     name    |              swift               |
|     type    |           object-store           |
+-------------+----------------------------------+
```

2.创建对象服务 API endpoints:

```
1.  keystone endpoint-create \

2.    --service-id $(keystone service-list | awk '/ object-store / {print $2}') \

3.    --publicurl 'http://controller:8080/v1/AUTH_%(tenant_id)s' \

4.    --internalurl 'http://controller:8080/v1/AUTH_%(tenant_id)s' \

5.    --adminurl http://controller:8080 \
```

```
6.    --region regionOne
```

```
root@controller:~# keystone endpoint-create \
>   --service-id $(keystone service-list | awk '/ object-store / {print $2}') \
>   --publicurl 'http://controller:8080/v1/AUTH_%(tenant_id)s' \
>   --internalurl 'http://controller:8080/v1/AUTH_%(tenant_id)s' \
>   --adminurl http://controller:8080 \
>   --region regionOne
+-------------+-----------------------------------------+
|   Property  |                  Value                  |
+-------------+-----------------------------------------+
|   adminurl  |          http://controller:8080         |
|      id     |      5ce5edd9822845a9be69d5db1aa6c81c   |
| internalurl | http://controller:8080/v1/AUTH_%(tenant_id)s |
|  publicurl  | http://controller:8080/v1/AUTH_%(tenant_id)s |
|    region   |                regionOne                |
|  service_id |      8e10fbbb9ed34e2e8207b7697a1322a8   |
+-------------+-----------------------------------------+
```

安装配置控制节点组件

1.安装

```
1.  apt-get install swift swift-proxy python-swiftclient python-keystoneclient \

2.    python-keystonemiddleware memcached
```

注意：完成 openstack 环境及 openstack 其它包

2.创建/etc/swift 目录

```
1.  mkdir /etc/swift
```

3.从对象存储服务源库获取代理服务配置文件

进入目录/etc/swift

```
1.  cd /etc/swift
```

，执行下面命令

```
curl -o /etc/swift/proxy-server.conf \
  https://raw.githubusercontent.com/openstack/swift/stable/juno/etc/proxy-server.conf-sample
```

##############################
如果 curl 不存在，则直接安装后，在执行命令，安装如下

```
apt-get install curl
```

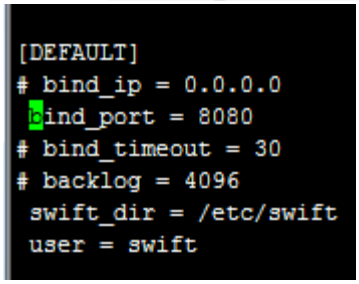
##############################

4.编辑文件 /etc/swift/proxy-server.conf，完成下面内容

```
sudo nano  /etc/swift/proxy-server.conf
```

a.在 [DEFAULT]部分，配置绑定的端口，用户和配置目录

```
[DEFAULT]
...
bind_port = 8080
user = swift
swift_dir = /etc/swift
```



b.在 [pipeline:main] 部分，启用 appropriate modules

```
[pipeline:main]
pipeline = authtoken cache healthcheck keystoneauth proxy-logging proxy-server
```

```
[pipeline:main]
#pipeline = catch_errors gatekeeper healthcheck proxy-logging cache container_sync bulk
pipeline = authtoken cache healthcheck keystoneauth proxy-logging proxy-server
```

注意：
更多信息在其他模块启用附加功能，查看部署向导

c.在[app:proxy-server]部分，启用账户管理

```
1.  [app:proxy-server]
2.  ...
3.  allow_account_management = true
4.  account_autocreate = true
```

d.在[filter:keystoneauth] 部分，配置操作的角色

```
1.  [filter:keystoneauth]
2.  use = egg:swift#keystoneauth
3.  ...
4.  operator_roles = admin,_member_
```

注意：
你可能需要取消这部分注释，你会看到【[filter:keystoneauth]】被注释掉了，所以需要取消掉

```
#
 [filter:keystoneauth]
use = egg:swift#keystoneauth
operator_roles = admin,_member_
# use = egg:swift#keystoneauth
```

e.在[filter:authtoken]部分，配置身份认证访问

```
1.  [filter:authtoken]
2.  paste.filter_factory = keystonemiddleware.auth_token:filter_factory
3.  ...
4.  auth_uri = http://controller:5000/v2.0
5.  identity_uri = http://controller:35357
```

```
6.  admin_tenant_name = service
7.  admin_user = swift
8.  admin_password = SWIFT_PASS
9.  delay_auth_decision = true
```

```
#
[filter:authtoken]
paste.filter_factory = keystonemiddleware.auth_token:filter_factory
auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357
admin_tenant_name = service
admin_user = swift
admin_password = SWIFT_PASS
delay_auth_decision = true

# paste.filter_factory = keystonemiddleware.auth_token:filter_factory
# auth_host = keystonehost
# auth_port = 35357
# auth_protocol = http
# auth_uri = http://keystonehost:5000/
# admin_tenant_name = service
# admin_user = swift
# admin_password = password
# delay_auth_decision = 1
# cache = swift.cache
# include_service_catalog = False
#
```

注意：

1.你可能需要取消掉注释掉这部分【为了保证正确性，这里直接添加】

2.注释掉其他 auth_host, auth_port, 和 auth_protocol，以免配置被覆盖

f.在 [filter:cache]部分，配置 memcached 位置

```
1.  [filter:cache]
2.  ...
3.  memcache_servers = 127.0.0.1:11211
```

openstack【juno】入门 【swift 篇】二十三：安装配置 swift 节点

问题导读

1.存储节点需要添加几块网卡，几块硬盘？
2.如何让新增加硬盘生效？
3.存储节点都进行了哪些配置？
4.object1 和 object2 配置有什么不同？

这一部分描述了安装配置存储节点，操作账户，容器，对象服务。为了简单起见，这个配置引用了两个存储节点，每一个都包含了两个空本地块存储设备。每一个设备, /dev/sdb 和 /dev/sdc,包含合适的分区表。尽管如此，swift 指出文件系统的可扩展属性。
测试和基准测试表明最佳的性能和可靠性在 xfs.更多 swift 信息，可以查看部署向导

这里新增两个存储节点 object1 和 object2，这个如果会使用虚拟机，相信不困难，这里就不在演示了。
注意：这里以 object1 的配置为例，object2 配置只需要更换 ip 即可。

安装准备

你必须配置每一个节点，在安装配置对象服务。相应的控制节点，每个存储节点，在管理网络包含一个网卡。每个存储节点，可以包含两块网卡，第二块网卡作为独立的备份网卡。更多信息查看 docs.openstack.org

安装 openstack 包

```
1.  apt-get install ubuntu-cloud-keyring
```

```
1.  echo "deb http://ubuntu-cloud.archive.canonical.com/ubuntu" \
```

```
2.    "trusty-updates/juno main" > /etc/apt/sources.list.d/cloudarchive-juno.list
```

更新源

```
1.  apt-get update && apt-get dist-upgrade
```

####################################

下面是以配置 objetc1 为例，object2 只要更换管理网络 ip 地址即可 objetc1 ip 地址

```
1.  10.0.0.51
```

objetc2 ip 地址

```
1.  10.0.0.52
```

1.配置管理

1.配置管理网络 ip 和 hostname

objetc1

ip 配置

打开文件

```
1.  sudo nano /etc/network/interfaces
```

添加如下内容：

```
1.  auto eth0

2.  iface eth0 inet static
```

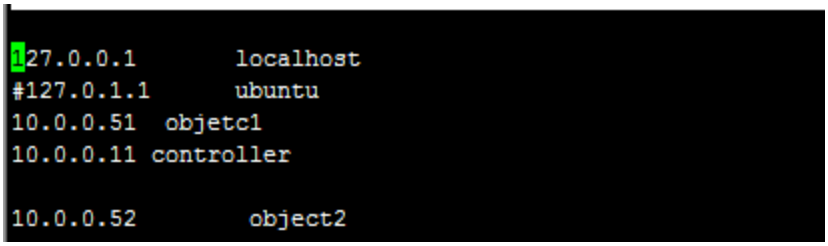
```
3.  address 10.0.0.51
4.  netmask 255.255.255.0
5.  gateway 10.0.0.2
6.  dns-nameservers 8.8.8.8
```

hosts 配置,编辑文件

```
1.  sudo nano /etc/hosts
```

贴入如下内容：

```
1.  127.0.0.1       localhost
2.  #127.0.1.1      ubuntu
3.  10.0.0.51  objetc1
4.  10.0.0.11 controller
5.  10.0.0.52        object2
```



objetc2

这里只是 ip 地址不同而已，其它均相同

```
1.  auto eth0
2.  iface eth0 inet static
3.  address 10.0.0.51
4.  netmask 255.255.255.0
5.  gateway 10.0.0.2
6.  dns-nameservers 8.8.8.8
```

2.安装 NTP

编辑文件

```
1.  sudo nano /etc/ntp.conf
```

添加如下内容：

```
1.  server controller iburst
```

注释掉其他 server

重启 NTP

```
1.  service ntp restart
```

更多内容参考：

openstack【juno】入门 【准备篇】二：：NTP 安装

3.安装支持实用包

```
1.  apt-get install xfsprogs rsync
```

4.格式化 /dev/sdb1 和 /dev/sdc1 分区表，作为 XFS

```
1.  mkfs.xfs /dev/sdb1

2.  mkfs.xfs /dev/sdc1
```

首先这里需要在虚拟机添加两块硬盘，然后使用 fdisk 命令，对其进行划分区。详细参考：解决：Disk /dev/sdb doesn't contain a valid partition table

否则会出现下面问题

```
1.  mkfs.xfs /dev/sdb1

2.  /dev/sdb1: No such file or directory
```

5.创建挂载点目录结构

```
1.  mkdir -p /srv/node/sdb1

2.  mkdir -p /srv/node/sdc1
```

6.编辑文件 /etc/fstab 添加如下内容

```
1.  sudo nano /etc/fstab
```

添加下面内容：

```
1.  /dev/sdb1 /srv/node/sdb1 xfs noatime,nodiratime,nobarrier,logbufs=8 0 2

2.  /dev/sdc1 /srv/node/sdc1 xfs noatime,nodiratime,nobarrier,logbufs=8 0 2
```

挂载：

```
1.  mount /srv/node/sdb1

2.  mount /srv/node/sdc1
```

7.编辑 /etc/rsyncd.conf 文件，添加如下内容

```
1.  sudo nano /etc/rsyncd.conf
```

添加内容如下：

```
1.  uid = swift

2.  gid = swift

3.  log file = /var/log/rsyncd.log

4.  pid file = /var/run/rsyncd.pid

5.  address = MANAGEMENT_INTERFACE_IP_ADDRESS

6.

7.  [account]
```

```
8.  max connections = 2
9.  path = /srv/node/
10. read only = false
11. lock file = /var/lock/account.lock
12.
13. [container]
14. max connections = 2
15. path = /srv/node/
16. read only = false
17. lock file = /var/lock/container.lock
18.
19. [object]
20. max connections = 2
21. path = /srv/node/
22. read only = false
23. lock file = /var/lock/object.lock
```

替换存储节点管理网络 ip 地址,替换为下面内容：

```
1.  uid = swift
2.  gid = swift
3.  log file = /var/log/rsyncd.log
4.  pid file = /var/run/rsyncd.pid
5.  address = 10.0.0.51
6.
7.  [account]
8.  max connections = 2
9.  path = /srv/node/
10. read only = false
11. lock file = /var/lock/account.lock
12.
13. [container]
14. max connections = 2
```

```
15. path = /srv/node/
16. read only = false
17. lock file = /var/lock/container.lock
18.
19. [object]
20. max connections = 2
21. path = /srv/node/
22. read only = false
23. lock file = /var/lock/object.lock
```

注意：rsync 服务不需要认证，所以考虑运行一个专用网络。

8.编辑文件 /etc/default/rsync 文件，启用 rsync 服务

```
1. sudo nano /etc/default/rsync
```

添加下面内容

```
1. RSYNC_ENABLE=true
```

9.启动 rsync 服务

```
1. service rsync start
```

安装配置存储节点组件

注意：下面在存储节点执行

1.安装包

```
1. apt-get install swift swift-account swift-container swift-object
```

2.从库中下载，并保存 accounting, container, 和 对象服务配置文件

```
curl -o /etc/swift/account-server.conf \
  https://raw.githubusercontent.com/openstack/swift/stable/juno/etc/account-server.conf-sample
```

```
curl -o /etc/swift/container-server.conf \
  https://raw.githubusercontent.com/openstack/swift/stable/juno/etc/container-server.conf-sample
```

```
curl -o /etc/swift/object-server.conf \
  https://raw.githubusercontent.com/openstack/swift/stable/juno/etc/object-server.conf-sample
```

3.编辑文件 /etc/swift/account-server.conf，完成下面内容

```
sudo nano /etc/swift/account-server.conf
```

a.在 [DEFAULT] 部分，配置绑定 ip 地址，端口，用户，配置文件目录，和挂载点木兰

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

替换为下面内容

```
[DEFAULT]
```

```
2.  ...
3.  bind_ip = 10.0.0.51
4.  bind_port = 6002
5.  user = swift
6.  swift_dir = /etc/swift
7.  devices = /srv/node
```

配置文件中 bind_port 已经存在，这里记得把其他注释掉

b.在[pipeline:main]部分，启用 appropriate 模块

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

文件中已经存在，可以使用默认

c.在 [filter:recon] 部分，配置缓存目录

```
1.  [filter:recon]
2.  ...
3.  recon_cache_path = /var/cache/swift
```

4.编辑文件 /etc/swift/container-server.conf ，完成下面内容：

```
1.  sudo nano  /etc/swift/container-server.conf
```

a.在 [DEFAULT]部分，配置绑定 ip 地址，端口，用户，配置目录，和挂载点

```
1.  [DEFAULT]
2.  ...
3.  bind_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
4.  bind_port = 6001
5.  user = swift
6.  swift_dir = /etc/swift
7.  devices = /srv/node
```

替换为如下内容

```
1.  [DEFAULT]
```

```
2.  ...
3.  bind_ip = 10.0.0.51
4.  bind_port = 6001
5.  user = swift
6.  swift_dir = /etc/swift
7.  devices = /srv/node
```

b.在[pipeline:main]部分，启用 appropriate 模块

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

启用更多额外功能参考：部署向导

c.在 [filter:recon]部分，配置缓存目录

```
1.  [filter:recon]
2.  ...
3.  recon_cache_path = /var/cache/swift
```

5.编辑文件/etc/swift/object-server.conf ，完成下面内容：

```
1.  sudo nano /etc/swift/object-server.conf
```

a.在 [DEFAULT]部分，配置绑定 ip 地址，绑定端口，用户，配置目录，和挂载点目录

```
1.  [DEFAULT]
2.  ...
3.  bind_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
4.  bind_port = 6000
5.  user = swift
6.  swift_dir = /etc/swift
7.  devices = /srv/node
```

MANAGEMENT_INTERFACE_IP_ADDRESS 替换为

```
[DEFAULT]
...
bind_ip = 10.0.0.51
bind_port = 6000
user = swift
swift_dir = /etc/swift
devices = /srv/node
```

b.在 [pipeline:main] 部分，启用 appropriate 模块

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

启用更多额外功能参考：部署向导

c.在 [filter:recon]部分，配置缓存目录

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

6.给挂载点目录合适的权限

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

7.创建 recon（侦查） 目录，并授权

```
mkdir -p /var/cache/swift
chown -R swift:swift /var/cache/swift
```

openstack【juno】入门【swift 篇】二十四：创建初始 rings

问题导读

1.如何形成环?
2.从本文遇到的错误，配置文件应该注意什么问题？
3.swift-init all start 的作用是什么？

在开始使用对象存储服务之前，你必须创建初始账户，容器，和 对象 rings。环生成器所创建的配置文件，每个节点使用确定和部署的存储架构。
为了简单起见，这个向导使用一个地区的区 2 ^10（1024）最大分区，每个对象 3 个副本，和至少一小时时间移动分区不止一次。对象存储一个分区代表一个存储设备上的目录，而不是一个传统分区表，更多内容，查看部署向导

账户环

账户服务器使用账户环，保存容器列表

创建环

注意：

执行下面命令在 控制节点

1.进入 /etc/swift 目录

```
1.  cd /etc/swift
```

2.创建 account.builder

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

```
root@controller:/etc/swift# swift-ring-builder account.builder create 10 3 1
root@controller:/etc/swift# ls
account.builder  backups  proxy-server.conf
```

3.添加每一个存储节点到环

```
1.  swift-ring-builder account.builder \
2.    add r1z1-STORAGE_NODE_MANAGEMENT_INTERFACE_IP_ADDRESS:6002/DEVICE_NAME DEVICE_WEIGHT
```

替换存储节点管理网络 ip 地址 STORAGE_NODE_MANAGEMENT_INTERFACE_IP_ADDRESS，替换存储节点设备名称 DEVICE_NAME ，例如在存储节点使用的 /dev/sdb1 存储设备，替换为下面内容

```
1.  swift-ring-builder account.builder add r1z1-10.0.0.51:6002/sdb1 100
```

```
1.  swift-ring-builder account.builder add r1z1-10.0.0.51:6002/sdc1 100
```

```
1.  swift-ring-builder account.builder add r1z1-10.0.0.52:6002/sdb1 100
```

```
1.  swift-ring-builder account.builder add r1z1-10.0.0.52:6002/sdc1 100
```

在每个存储节点使用上面命令

4.核实 ring 内容

```
1.  swift-ring-builder account.builder
```

```
root@controller:/etc/swift# swift-ring-builder account.builder
account.builder, build version 4
1024 partitions, 3.000000 replicas, 1 regions, 1 zones, 4 devices, 100.00 balance
The minimum number of hours before a partition can be reassigned is 1
Devices:    id  region  zone      ip address  port  replication ip  replication port      name weight partitions balance meta
             0       1     1       10.0.0.51  6002       10.0.0.51              6002      sdb1 100.00          0 -100.00
             1       1     1       10.0.0.52  6002       10.0.0.52              6002      sdb1 100.00          0 -100.00
             2       1     1       10.0.0.51  6002       10.0.0.51              6002      sdc1 100.00          0 -100.00
             3       1     1       10.0.0.52  6002       10.0.0.52              6002      sdc1 100.00          0 -100.00
root@controller:/etc/swift#
```

5.负载均衡 ring

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

这个过程可能需要时间

容器环

容器服务器使用容器环保存对象服务列表。尽管如此，它不跟踪对象的存储位置。

创建环

在**控制节点**，完成下面内容

1.进入 /etc/swift 目录

```
1.  cd  /etc/swift
```

2.创建 container.builder 文件

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

3.添加存储节点到环

格式如下：

```
1.  swift-ring-builder container.builder \
2.    add r1z1-STORAGE_NODE_MANAGEMENT_INTERFACE_IP_ADDRESS:6001/DEVICE_NAME DEVICE_WEIGHT
```

分别替换 STORAGE_NODE_MANAGEMENT_INTERFACE_IP_ADDRESS 和 DEVICE_NAME

替换为下面，变换为下面四种形式

```
1.  swift-ring-builder container.builder add r1z1-10.0.0.51:6001/sdb1 100
```

```
1.  swift-ring-builder container.builder add r1z1-10.0.0.51:6001/sdc1 100
```

```
1.  swift-ring-builder container.builder add r1z1-10.0.0.52:6001/sdb1 100
```

```
1.  swift-ring-builder container.builder add r1z1-10.0.0.52:6001/sdc1 100
```

```
root@controller:/etc/swift# swift-ring-builder container.builder add r1z1-10.0.0.51:6001/sdb1 100
Device d0r1z1-10.0.0.51:6001R10.0.0.51:6001/sdb1_"" with 100.0 weight got id 0
root@controller:/etc/swift#  swift-ring-builder container.builder add r1z1-10.0.0.51:6001/sdc1 100
Device d1r1z1-10.0.0.51:6001R10.0.0.51:6001/sdc1_"" with 100.0 weight got id 1
root@controller:/etc/swift#  swift-ring-builder container.builder add r1z1-10.0.0.52:6001/sdb1 100
Device d2r1z1-10.0.0.52:6001R10.0.0.52:6001/sdb1_"" with 100.0 weight got id 2
root@controller:/etc/swift#  swift-ring-builder container.builder add r1z1-10.0.0.52:6001/sdc1 100
Device d3r1z1-10.0.0.52:6001R10.0.0.52:6001/sdc1_"" with 100.0 weight got id 3
root@controller:/etc/swift#
```

4.核实环的内容

```
1.  swift-ring-builder container.builder
```

```
root@controller:/etc/swift# swift-ring-builder container.builder
container.builder, build version 4
1024 partitions, 3.000000 replicas, 1 regions, 1 zones, 4 devices, 100.00 balance
The minimum number of hours before a partition can be reassigned is 1
Devices:    id  region  zone      ip address  port  replication ip  replication port      name weight partitions balance meta
             0       1     1       10.0.0.51  6001       10.0.0.51              6001      sdb1 100.00          0 -100.00
             1       1     1       10.0.0.51  6001       10.0.0.51              6001      sdc1 100.00          0 -100.00
             2       1     1       10.0.0.52  6001       10.0.0.52              6001      sdb1 100.00          0 -100.00
             3       1     1       10.0.0.52  6001       10.0.0.52              6001      sdc1 100.00          0 -100.00
root@controller:/etc/swift#
```

5.Rebalance 环

```
1.  swift-ring-builder container.builder rebalance
```

对象环

对象服务器使用对象环保存本地对象列表

创建环

注意下面命令在**控制节点**执行

1.进入目录

```
1.  cd /etc/swift
```

2.创建文件 object.builder

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

3.添加节点到环

格式如下

```
1.  swift-ring-builder object.builder \
2.    add r1z1-STORAGE_NODE_MANAGEMENT_INTERFACE_IP_ADDRESS:6000/DEVICE_NAME DEVICE_WEIGHT
```

替换为下面形式，执行下面四个命令

```
1.  swift-ring-builder object.builder add r1z1-10.0.0.51:6000/sdb1 100
```

```
1.  swift-ring-builder object.builder add r1z1-10.0.0.51:6000/sdc1 100
```

```
1.  swift-ring-builder object.builder add r1z1-10.0.0.52:6000/sdb1 100
```

```
1.  swift-ring-builder object.builder add r1z1-10.0.0.52:6000/sdc1 100
```

```
root@controller:/etc/swift# swift-ring-builder object.builder create 10 3 1
root@controller:/etc/swift# swift-ring-builder object.builder add r1z1-10.0.0.51:6000/sdb1 100
Device d0r1z1-10.0.0.51:6000R10.0.0.51:6000/sdb1_"" with 100.0 weight got id 0
root@controller:/etc/swift# swift-ring-builder object.builder add r1z1-10.0.0.51:6000/sdc1 100
Device d1r1z1-10.0.0.51:6000R10.0.0.51:6000/sdc1_"" with 100.0 weight got id 1
root@controller:/etc/swift# swift-ring-builder object.builder add r1z1-10.0.0.52:6000/sdb1 100
Device d2r1z1-10.0.0.52:6000R10.0.0.52:6000/sdb1_"" with 100.0 weight got id 2
root@controller:/etc/swift# swift-ring-builder object.builder add r1z1-10.0.0.52:6000/sdc1 100
Device d3r1z1-10.0.0.52:6000R10.0.0.52:6000/sdc1_"" with 100.0 weight got id 3
root@controller:/etc/swift#
```

4.核实环内容

```
1.  swift-ring-builder object.builder
```

```
root@controller:/etc/swift# swift-ring-builder object.builder
object.builder, build version 4
1024 partitions, 3.000000 replicas, 1 regions, 1 zones, 4 devices, 100.00 balance
The minimum number of hours before a partition can be reassigned is 1
Devices:    id  region  zone      ip address  port  replication ip  replication port      name weight partitions balance meta
             0       1     1       10.0.0.51  6000       10.0.0.51              6000      sdb1 100.00          0 -100.00
             1       1     1       10.0.0.51  6000       10.0.0.51              6000      sdc1 100.00          0 -100.00
             2       1     1       10.0.0.52  6000       10.0.0.52              6000      sdb1 100.00          0 -100.00
             3       1     1       10.0.0.52  6000       10.0.0.52              6000      sdc1 100.00          0 -100.00
root@controller:/etc/swift#
```

5.Rebalance 环

```
1.  swift-ring-builder object.builder rebalance
```

分发配置文件

复制 account.ring.gz, container.ring.gz, 和 object.ring.gz 文件到/etc/swift 目录（每个存储节点和其它任何运行代理服务的额外节点）

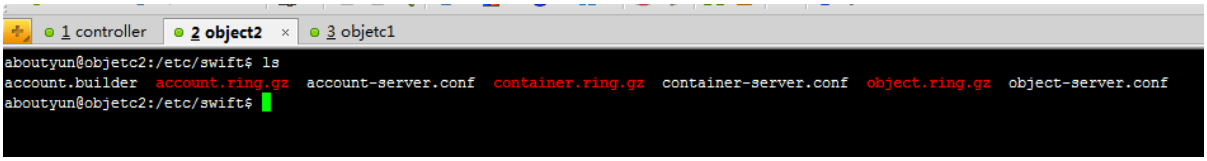
这些包在控制节点：

```
root@controller:/etc/swift# ls
account.builder  account.ring.gz  backups  container.builder  container.ring.gz  object.builder  object.ring.gz  proxy-server.conf
```

分别分发到 object1 和 object2

如果没有配置 object 的 hosts，可以直接执行下面命令。
scp container.ring.gz aboutyun@10.0.0.51:/etc/swift
如果没有操作权限，则按照下面 scp container.ring.gz aboutyun@10.0.0.51:~/
然后通过
cp 命令完成分发。
当然也可以自己写脚本。这里方法很多，只要能达到目的即可。

分发后内容如下：



完成安装

配置 hashes 和默认存储策略

在控制节点进行下面操作

1.从对象存储资源库，保存 /etc/swift/swift.conf 文件

```
1.  curl -o /etc/swift/swift.conf \

2.    https://raw.githubusercontent.com/openstack/swift/stable/juno/etc/swift.conf-sample
```

2.编辑文件，完成下面内容

```
1.  sudo nano /etc/swift/swift.conf
```

a.在[swift-hash] 部分，为环境配置哈希路径 prefix 和 suffix

格式如下：

```
1.  [swift-hash]

2.  ...

3.  swift_hash_path_suffix = HASH_PATH_PREFIX

4.  swift_hash_path_prefix = HASH_PATH_SUFFIX
```

替换为自定义字符

```
1.  [swift-hash]
2.  ...
3.  swift_hash_path_suffix = about
4.  swift_hash_path_prefix = yun
```

记得在相应的节点，创建路径

b.在 [storage-policy:0] 部分，配置默认存储策略

```
1.  [storage-policy:0]
2.  ...
3.  name = Policy-0
4.  default = yes
```

3.复制文件 swift.conf 到每个存储节点 /etc/swift 目录和其它运行代理服务的节点。

先远程复制

```
1.  scp swift.conf  aboutyun@10.0.0.51:~/
2.  scp swift.conf  aboutyun@10.0.0.52:~/
```

然后复制到相应的目录。

```
1.  sudo cp swift.conf /etc/swift
```

4.授权（所有节点）

```
1.  chown -R swift:swift /etc/swift
```

5.在控制节点，和其它运行代理服务的节点，重启存储节点服务包括它的依赖

```
1.  service memcached restart
2.  service swift-proxy restart
```

6.在存储节点，启动存储节点服务

```
1.  swift-init all start
```

输出如下信息：

```
1.  root@controller:~# swift-init all start
2.  Unable to locate config for container-updater
3.  Unable to locate config for account-auditor
4.  Unable to locate config for object-replicator
5.  Unable to locate config for container-sync
6.  Unable to locate config for container-replicator
7.  Unable to locate config for object-auditor
8.  Unable to locate config for object-expirer
9.  Unable to locate config for container-auditor
10. Unable to locate config for container-server
11. Unable to locate config for object-server
12. Unable to locate config for account-reaper
13. proxy-server running (3939 - /etc/swift/proxy-server.conf)
14. proxy-server already started...
15. Unable to locate config for account-replicator
16. Unable to locate config for object-updater
17. Unable to locate config for container-reconciler
18. Unable to locate config for account-server
```

```
root@controller:/home/aboutyun# cd
root@controller:~# swift-init all start
Unable to locate config for container-updater
Unable to locate config for account-auditor
Unable to locate config for object-replicator
Unable to locate config for container-sync
Unable to locate config for container-replicator
Unable to locate config for object-auditor
Unable to locate config for object-expirer
Unable to locate config for container-auditor
Unable to locate config for container-server
Unable to locate config for object-server
Unable to locate config for account-reaper
proxy-server running (3939 - /etc/swift/proxy-server.conf)
proxy-server already started...
Unable to locate config for account-replicator
Unable to locate config for object-updater
Unable to locate config for container-reconciler
Unable to locate config for account-server
root@controller:~#
```

注意：

存储节点运行了很多存储节点服务， swift-init 使他们更容易管理

遇到问题

问题 1：配置文件含有空格

service swift-proxy restart

```
root@controller:/etc/swift# service swift-proxy restart
stop: Unknown instance:
start: Job failed to start
```

执行 swift-init all start，报了如下错误：

```
1.  Error trying to load config from /etc/swift/proxy-server.conf: File contains parsing errors:
    /etc/swift/proxy-server.conf
2.          [line  2]: ' bind_ip = 0.0.0.0\n'
3.          [line  3]: ' bind_port = 8080\n'
4.          [line  6]: ' swift_dir = /etc/swift\n'
5.          [line  7]: ' user = swift\n'
```

原因：含有空格

问题 2：keystoneauth 标记不能识别

```
1.    File "/usr/bin/swift-proxy-server", line 23, in <module>
2.      sys.exit(run_wsgi(conf_file, 'proxy-server', **options))
3.    File "/usr/lib/python2.7/dist-packages/swift/common/wsgi.py", line 445, in run_wsgi
4.      loadapp(conf_path, global_conf=global_conf)
5.    File "/usr/lib/python2.7/dist-packages/swift/common/wsgi.py", line 354, in loadapp
6.      ctx = loadcontext(loadwsgi.APP, conf_file, global_conf=global_conf)
7.    File "/usr/lib/python2.7/dist-packages/swift/common/wsgi.py", line 338, in loadcontext
8.      global_conf=global_conf)
9.    File "/usr/lib/python2.7/dist-packages/paste/deploy/loadwsgi.py", line 296, in
    loadcontext
10.     global_conf=global_conf)
11.   File "/usr/lib/python2.7/dist-packages/paste/deploy/loadwsgi.py", line 320, in
    _loadconfig
12.     return loader.get_context(object_type, name, global_conf)
13.   File "/usr/lib/python2.7/dist-packages/swift/common/wsgi.py", line 61, in get_context
14.     object_type, name=name, global_conf=global_conf)
15.   File "/usr/lib/python2.7/dist-packages/paste/deploy/loadwsgi.py", line 450, in
    get_context
16.     global_additions=global_additions)
17.   File "/usr/lib/python2.7/dist-packages/paste/deploy/loadwsgi.py", line 562, in
    _pipeline_app_context
18.     for name in pipeline[:-1]]
19.   File "/usr/lib/python2.7/dist-packages/swift/common/wsgi.py", line 61, in get_context
```

```
20.     object_type, name=name, global_conf=global_conf)
21.   File "/usr/lib/python2.7/dist-packages/paste/deploy/loadwsgi.py", line 408, in
    get_context
22.     object_type, name=name)
23.   File "/usr/lib/python2.7/dist-packages/paste/deploy/loadwsgi.py", line 587, in
    find_config_section
24.     self.filename))
25. LookupError: No section 'keystoneauth' (prefixed by 'filter') found in config
    /etc/swift/proxy-server.conf
```

```
Traceback (most recent call last):
  File "/usr/bin/swift-proxy-server", line 23, in <module>
    sys.exit(run_wsgi(conf_file, 'proxy-server', **options))
  File "/usr/lib/python2.7/dist-packages/swift/common/wsgi.py", line 445, in run_wsgi
    loadapp(conf_path, global_conf=global_conf)
  File "/usr/lib/python2.7/dist-packages/swift/common/wsgi.py", line 354, in loadapp
    ctx = loadcontext(loadwsgi.APP, conf_file, global_conf=global_conf)
  File "/usr/lib/python2.7/dist-packages/swift/common/wsgi.py", line 338, in loadcontext
    global_conf=global_conf)
  File "/usr/lib/python2.7/dist-packages/paste/deploy/loadwsgi.py", line 296, in loadcontext
    global_conf=global_conf)
  File "/usr/lib/python2.7/dist-packages/paste/deploy/loadwsgi.py", line 320, in _loadconfig
    return loader.get_context(object_type, name, global_conf)
  File "/usr/lib/python2.7/dist-packages/swift/common/wsgi.py", line 61, in get_context
    object_type, name=name, global_conf=global_conf)
  File "/usr/lib/python2.7/dist-packages/paste/deploy/loadwsgi.py", line 450, in get_context
    global_additions=global_additions)
  File "/usr/lib/python2.7/dist-packages/paste/deploy/loadwsgi.py", line 562, in _pipeline_app_context
    for name in pipeline[:-1]]
  File "/usr/lib/python2.7/dist-packages/swift/common/wsgi.py", line 61, in get_context
    object_type, name=name, global_conf=global_conf)
  File "/usr/lib/python2.7/dist-packages/paste/deploy/loadwsgi.py", line 408, in get_context
    object_type, name=name)
  File "/usr/lib/python2.7/dist-packages/paste/deploy/loadwsgi.py", line 587, in find_config_section
    self.filename))
LookupError: No section 'keystoneauth' (prefixed by 'filter') found in config /etc/swift/proxy-server.conf
```

解决：
含有空格，造成不能识别配置文件。

总结：

在执行 service swift-proxy restart 的时候，报错 Unknown instance，没有任何日志，然后执行 swift-init all start 的时候，产生了上面两个错误。两个错误均是配置文件自带，并且去掉注释，因此喜欢手打，不使用粘帖复制的方式，一定要谨慎操作，否则含有空格等，很有可能造成配置文件的配置项不能被识别。

openstack【juno】入门 【swift 篇】二十五：验证安装（控制节点）

问题导读

1.swift stat 的作用是什么？

2.swift 如何上传一个文件？

3.如何查看 swift 列表？

在**控制节点**执行下面命令：

1.环境变量生效

```
1.  source demo-openrc.sh
```

2.显示服务状态

```
1.  swift stat
```

输出下面信息

```
1.        Account: AUTH_bfb31e69f05b44cd89e1336c09042e2f
2.       Containers: 0
3.          Objects: 0
4.            Bytes: 0
5.     Content-Type: text/plain; charset=utf-8
6.      X-Timestamp: 1424577028.33577
7.       X-Trans-Id: tx0163866e80294069b81d3-0054e95204
8.  X-Put-Timestamp: 1424577028.33577
```

```
root@controller:~# swift stat
        Account: AUTH_bfb31e69f05b44cd89e1336c09042e2f
     Containers: 0
        Objects: 0
          Bytes: 0
   Content-Type: text/plain; charset=utf-8
    X-Timestamp: 1424577028.33577
     X-Trans-Id: tx0163866e80294069b81d3-0054e95204
X-Put-Timestamp: 1424577028.33577
root@controller:~#
```

3.上传一个测试文件

这里创建了一个 testSwift 文件

```
1.  swift upload demo-container1 testSwift
```

```
root@controller:~#  swift upload demo-container1 testSwift
testSwift
root@controller:~#
```

4.列出容器

```
1.  swift list
```

```
root@controller:~# swift list
demo-container1
root@controller:~#
```

5.下载文件

```
1.  swift download demo-container1 testSwift
```

遇到问题

执行命令：

```
1.  swift list
```

产生如下错误：

```
1.  Traceback (most recent call last):
2.    File "/usr/bin/swift", line 10, in <module>
3.      sys.exit(main())
4.    File "/usr/lib/python2.7/dist-packages/swiftclient/shell.py", line 1287, in main
5.      globals()['st_%s' % args[0]](parser, argv[1:], output)
6.    File "/usr/lib/python2.7/dist-packages/swiftclient/shell.py", line 492, in st_stat
7.      stat_result = swift.stat()
8.    File "/usr/lib/python2.7/dist-packages/swiftclient/service.py", line 427, in stat
9.      raise SwiftError('Account not found', exc=err)
10. swiftclient.service.SwiftError: 'Account not found'
```

不能定位错误，使用--debug，如下命令

```
1.  swift --debug stat
```

输出下面信息

```
1.  DEBUG:keystoneclient.auth.identity.v2:Making authentication request to
    http://controller:5000/v2.0/tokens
2.  INFO:urllib3.connectionpool:Starting new HTTP connection (1): controller
3.  DEBUG:urllib3.connectionpool:Setting read timeout to None
4.  DEBUG:urllib3.connectionpool:"POST /v2.0/tokens HTTP/1.1" 200 2799
5.  DEBUG:iso8601.iso8601:Parsed 2015-02-20T05:52:16Z into {'tz_sign': None,
    'second_fraction': None, 'hour': u'05', 'daydash': u'20', 'tz_hour': None, 'month': None,
    'timezone': u'Z', 'second': u'16', 'tz_minute': None, 'year': u'2015', 'separator': u'T',
```

```
    'monthdash': u'02', 'day': None, 'minute': u'52'} with default timezone
    <iso8601.iso8601.Utc object at 0x7f0ddb06ba10>
6.  DEBUG:iso8601.iso8601:Got u'2015' for 'year' with default None
7.  DEBUG:iso8601.iso8601:Got u'02' for 'monthdash' with default 1
8.  DEBUG:iso8601.iso8601:Got 2 for 'month' with default 2
9.  DEBUG:iso8601.iso8601:Got u'20' for 'daydash' with default 1
10. DEBUG:iso8601.iso8601:Got 20 for 'day' with default 20
11. DEBUG:iso8601.iso8601:Got u'05' for 'hour' with default None
12. DEBUG:iso8601.iso8601:Got u'52' for 'minute' with default None
13. DEBUG:iso8601.iso8601:Got u'16' for 'second' with default None
14. INFO:urllib3.connectionpool:Starting new HTTP connection (1): controller
15. DEBUG:urllib3.connectionpool:Setting read timeout to None
16. DEBUG:urllib3.connectionpool:"HEAD /v1/AUTH_bfb31e69f05b44cd89e1336c09042e2f HTTP/1.1"
    503 0
17. INFO:swiftclient:REQ: curl -i
    http://controller:8080/v1/AUTH_bfb31e69f05b44cd89e1336c09042e2f -I -H "X-Auth-Token:
    e0712594a90d47d398dc3e435648b0f6"
18. INFO:swiftclient:RESP STATUS: 503 Internal Server Error
19. INFO:swiftclient:RESP HEADERS: [('date', 'Fri, 20 Feb 2015 04:52:16 GMT'),
    ('content-length', '0'), ('content-type', 'text/html; charset=UTF-8'), ('x-trans-id',
    'txf9179cecb3f84d2d89515-0054e6bd80')]
20. DEBUG:urllib3.connectionpool:Setting read timeout to None
21. DEBUG:urllib3.connectionpool:"HEAD /v1/AUTH_bfb31e69f05b44cd89e1336c09042e2f HTTP/1.1"
    503 0
22. INFO:swiftclient:REQ: curl -i
    http://controller:8080/v1/AUTH_bfb31e69f05b44cd89e1336c09042e2f -I -H "X-Auth-Token:
    e0712594a90d47d398dc3e435648b0f6"
23. INFO:swiftclient:RESP STATUS: 503 Internal Server Error
24. INFO:swiftclient:RESP HEADERS: [('date', 'Fri, 20 Feb 2015 04:52:17 GMT'),
    ('content-length', '0'), ('content-type', 'text/html; charset=UTF-8'), ('x-trans-id',
    'tx564066dbdf9c479a80317-0054e6bd81')]
25. DEBUG:urllib3.connectionpool:Setting read timeout to None
```

```
26. DEBUG:urllib3.connectionpool:"HEAD /v1/AUTH_bfb31e69f05b44cd89e1336c09042e2f HTTP/1.1"
    503 0
27. INFO:swiftclient:REQ: curl -i
    http://controller:8080/v1/AUTH_bfb31e69f05b44cd89e1336c09042e2f -I -H "X-Auth-Token:
    e0712594a90d47d398dc3e435648b0f6"
28. INFO:swiftclient:RESP STATUS: 503 Internal Server Error
29. INFO:swiftclient:RESP HEADERS: [('date', 'Fri, 20 Feb 2015 04:52:19 GMT'),
    ('content-length', '0'), ('content-type', 'text/html; charset=UTF-8'), ('x-trans-id',
    'tx8026db12471043108f8aa-0054e6bd83')]
30. DEBUG:urllib3.connectionpool:Setting read timeout to None
31. DEBUG:urllib3.connectionpool:"HEAD /v1/AUTH_bfb31e69f05b44cd89e1336c09042e2f HTTP/1.1"
    503 0
32. INFO:swiftclient:REQ: curl -i
    http://controller:8080/v1/AUTH_bfb31e69f05b44cd89e1336c09042e2f -I -H "X-Auth-Token:
    e0712594a90d47d398dc3e435648b0f6"
33. INFO:swiftclient:RESP STATUS: 503 Internal Server Error
34. INFO:swiftclient:RESP HEADERS: [('date', 'Fri, 20 Feb 2015 04:52:23 GMT'),
    ('content-length', '0'), ('content-type', 'text/html; charset=UTF-8'), ('x-trans-id',
    'tx672d65e6e5cc4bcb81a99-0054e6bd87')]
35. DEBUG:urllib3.connectionpool:Setting read timeout to None
36. DEBUG:urllib3.connectionpool:"HEAD /v1/AUTH_bfb31e69f05b44cd89e1336c09042e2f HTTP/1.1"
    503 0
37. INFO:swiftclient:REQ: curl -i
    http://controller:8080/v1/AUTH_bfb31e69f05b44cd89e1336c09042e2f -I -H "X-Auth-Token:
    e0712594a90d47d398dc3e435648b0f6"
38. INFO:swiftclient:RESP STATUS: 503 Internal Server Error
39. INFO:swiftclient:RESP HEADERS: [('date', 'Fri, 20 Feb 2015 04:52:31 GMT'),
    ('content-length', '0'), ('content-type', 'text/html; charset=UTF-8'), ('x-trans-id',
    'tx3b40cb1889c7403494571-0054e6bd8f')]
40. DEBUG:urllib3.connectionpool:Setting read timeout to None
41. DEBUG:urllib3.connectionpool:"HEAD /v1/AUTH_bfb31e69f05b44cd89e1336c09042e2f HTTP/1.1"
    503 0
```

```
INFO:swiftclient:REQ: curl -i http://controller:8080/v1/AUTH_bfb31e69f05b44cd89e1336c09042e2f -I -H "X-Auth-Token: e0712594a90d47d398dc3e435648b0f6"
INFO:swiftclient:RESP STATUS: 503 Internal Server Error
INFO:swiftclient:RESP HEADERS: [('date', 'Fri, 20 Feb 2015 04:52:47 GMT'), ('content-length', '0'), ('content-type', 'text/html; charset=UTF-8'), ('x-trans-id', 'tx38758acf7fe74c5cb1597-0054e6bd9f')]
ERROR:swiftclient:Account HEAD failed: http://controller:8080/v1/AUTH_bfb31e69f05b44cd89e1336c09042e2f 503 Internal Server Error
Traceback (most recent call last):
  File "/usr/lib/python2.7/dist-packages/swiftclient/client.py", line 1236, in _retry
    rv = func(self.url, self.token, *args, **kwargs)
  File "/usr/lib/python2.7/dist-packages/swiftclient/client.py", line 521, in head_account
    http_response_content=body)
ClientException: Account HEAD failed: http://controller:8080/v1/AUTH_bfb31e69f05b44cd89e1336c09042e2f 503 Internal Server Error
Traceback (most recent call last):
  File "/usr/bin/swift", line 10, in <module>
    sys.exit(main())
  File "/usr/lib/python2.7/dist-packages/swiftclient/shell.py", line 1287, in main
    globals()['st_%s' % args[0]](parser, argv[1:], output)
  File "/usr/lib/python2.7/dist-packages/swiftclient/shell.py", line 492, in st_stat
    stat_result = swift.stat()
  File "/usr/lib/python2.7/dist-packages/swiftclient/service.py", line 427, in stat
    raise SwiftError('Account not found', exc=err)
swiftclient.service.SwiftError: 'Account not found'
```

下面错误是核心

```
1.  INFO:swiftclient:REQ: curl -i
    http://controller:8080/v1/AUTH_bfb31e69f05b44cd89e1336c09042e2f -I -H "X-Auth-Token:
    e0712594a90d47d398dc3e435648b0f6"
2.  INFO:swiftclient:RESP STATUS: 503 Internal Server Error
```

意思是：请求服务器资源，不能从服务器获取。

这里面有两方面错误：

要么是

问题可能性

1.账户错误

2.服务器错误

验证：

1.账户错误，创建了另外一个 swift 代理节点，结果还是一样。

2.服务器错误

这个有两个节点 object1 和 object2

因此 object1 和 object2 是关键。但是并不知道到底是哪里的错误。

因此采用谷歌、百度。

答案如下：

1.配置文件错误，即授权错误

auth_host, auth_port, 和 auth_protocol 错误。

auth_host 不能访问，本文是 10.0.0.11，这个都是相互通信的，如果使用的是私有 ip，可能产生问题。

本系列使用下面 url 来代替上面，使用的是 controller

auth_uri = http://controller:5000/v2.0，所以不会这里的问题。

2.object 节点发生改变， 下面三个包则需要重新复制到其他节点

account.ring.gz

container.ring.gz

object.ring.gz

上面方案是从百度谷歌得到，但是都没有起作用。

##

问题的原因：

由于这里使用的是虚拟机，object1 配置完毕，对于 object2 直接使用的克隆的方式。

所以对 object2 做了如下核实措施：

1.mac 地址重新生成

2.hostname 在控制节点、object1 和 object2 节点都必须包含如下内容：

```
1.  10.0.0.51       object1
2.  10.0.0.52       object2
```

3.配置文件管理 ip，都必须替换成 10.0.0.52.

再次执行 swift stat，问题得到解决。

openstack【juno】入门 【实例篇】二十六：创建实例(neutron)

问题导读

1.如何创建实例?
2.openstack 默认几个 flavor?
3.如何查看虚拟机状态?

1.生成密钥对

大多数 cloud images 支持公钥授权而不是传统的用户名密码授权。在创建公钥前，你必须使用 ssh-keygen 生成公/私钥对。添加公钥到 openstack 环境中。

1.生效 demo 租户凭据

```
1.  source admin-openrc.sh
```

内容如下：

```
1.  export OS_TENANT_NAME=admin
2.  export OS_USERNAME=admin
3.  export OS_PASSWORD=ADMIN_PASS
4.  export OS_AUTH_URL=http://controller:35357/v2.0
```

2.生成密钥对

```
1.  ssh-keygen
```

3.添加公玥到 openstack 环境

```
1.  nova keypair-add --pub-key ~/.ssh/id_rsa.pub demo-key
```

4.核实添加的公玥

```
1.  nova keypair-list
```

输出下列内容：

```
1.  root@controller:~# nova keypair-list
2.  +----------+-------------------------------------------------+
3.  | Name     | Fingerprint                                     |
4.  +----------+-------------------------------------------------+
5.  | demo-key | a1:23:cb:63:c6:0e:67:14:da:7e:89:a2:97:a4:3f:f9 |
6.  +----------+-------------------------------------------------+
```

```
root@controller:/var/log/neutron# nova keypair-list
+----------+-------------------------------------------------+
| Name     | Fingerprint                                     |
+----------+-------------------------------------------------+
| demo-key | a1:23:cb:63:c6:0e:67:14:da:7e:89:a2:97:a4:3f:f9 |
+----------+-------------------------------------------------+
```

创建一个实例

创建实例，你至少指定 flavor，镜像名称，网络，安全组, key，和实例名称.

1.一个 flavor 指定一个虚拟资源分配文件包括处理器、内存、存储

列出可用 flavor

```
1.  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      |
+----+-----------+-----------+------+-----------+------+-------+-------------+--------
---+
```

```
root@controller:/var/log/neutron# 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      |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
```

第一个实例，可以使用 m1.tinyflavor.

2.列出可用镜像

```
nova image-list
```

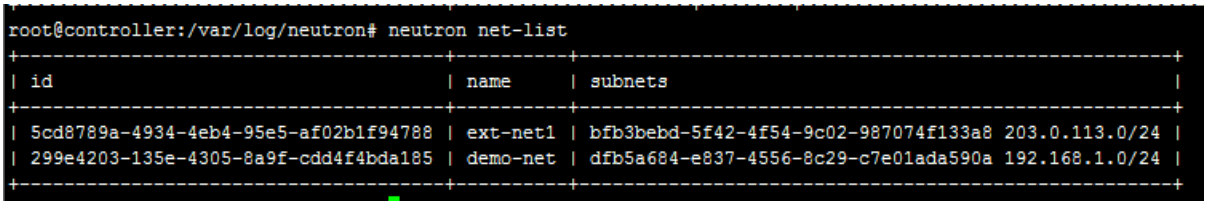
```
root@controller:/var/log/neutron# nova image-list
+--------------------------------------+---------------------+--------+--------------------------------------+
| ID                                   | Name                | Status | Server                               |
+--------------------------------------+---------------------+--------+--------------------------------------+
| 2cf16e02-db12-458f-b004-d2c54985d1e0 | begin               | ACTIVE | 29e13b8a-9105-44aa-9b24-4cd9bef9a912 |
| 9ec0ceec-0629-406a-9069-159a8b59ea78 | cirros-0.3.3-x86_64 | ACTIVE |                                      |
+--------------------------------------+---------------------+--------+--------------------------------------+
```

```
+--------------------------------------+--------------------+--------+--------+
| ID                                   | Name               | Status | Server |
+--------------------------------------+--------------------+--------+--------+
| 9ec0ceec-0629-406a-9069-159a8b59ea78 | cirros-0.3.3-x86_64 | ACTIVE |        |
+--------------------------------------+--------------------+--------+--------+
```

第一个实例使用 cirros-0.3.3-x86_64 镜像。

3.列出可用网络

```
neutron net-list
```




```
+--------------------------------------+----------+----------------------------------
------------------+
| id                                   | name                |
subnets
                  |
+--------------------------------------+----------+----------------------------------
------------------+
| 0028b1d7-07c9-468c-abf2-ccd4dc17b240 | ext-net      |
e6fab309-a8ff-4385-8b32-c942434414cf                            |
| 299e4203-135e-4305-8a9f-cdd4f4bda185 | demo-net  |
dfb5a684-e837-4556-8c29-c7e01ada590a 192.168.1.0/24 |
+--------------------------------------+----------+----------------------------------
------------------+
```

第一个实例使用 demo-net 租户网络，尽管如此，你需要使用 id，而不是使用 name

4.列出有效安全组

```
nova secgroup-list
+--------------------------------------+---------+-------------+
| Id                                   | Name    | Description |
+--------------------------------------+---------+-------------+
| 177deb74-7479-40c3-a332-4811bc9326bf | default | default     |
+--------------------------------------+---------+-------------+
```

第一个实例，使用默认安全组，这个安全组类似防火墙，阻止远程访问实例。如果拒绝远程访问实例，创建它，然后配置远程访问

5.创建实例

格式如下

```
nova boot --flavor m1.tiny --image cirros-0.3.3-x86_64 --nic net-id=DEMO_NET_ID \
  --security-group default --key-name demo-key demo-instance1
```

需要替换的内容：

DEMO_NET_ID 这里为：

```
nova boot --flavor m1.tiny --image cirros-0.3.3-x86_64 --nic
net-id=6e0c30ab-1c32-4e8e-b08f-84cdf2521713 \
--security-group default --key-name demo-key demo-instance1
```

6.检测实例状态

```
nova list
```

```
root@controller:/var/log/neutron# nova list
+--------------------------------------+---------------+--------+------------+-------------+---------------------+
| ID                                   | Name          | Status | Task State | Power State | Networks            |
+--------------------------------------+---------------+--------+------------+-------------+---------------------+
| c9b27530-1116-4c04-a371-43a949efd876 | demo-instance1 | ACTIVE | -          | Running     | demo-net=192.168.1.7 |
+--------------------------------------+---------------+--------+------------+-------------+---------------------+
```

这里遇到了创建实例不成功，并且 ip 地址也并没有显示。在后面会做一些相关说明

7.使用虚拟控制台访问实例

为实例获取一个 Virtual Network Computing (VNC) session url，并通过浏览器访问

```
1.  nova get-vnc-console demo-instance1 novnc
```

由于我这里有多个 demo-instance1，所以这里获取不到，使用 id 的方式获取。

```
1.  nova get-vnc-console instance-ID novnc
```

```
1.  nova get-vnc-console 29e13b8a-9105-44aa-9b24-4cd9bef9a912 novnc
```

```
1.  +-------+-----------------------------------------------------------------------------
    ----+
2.  | Type  |
    Url                                                                          |
3.  +-------+-----------------------------------------------------------------------------
    ----+
4.  | novnc |
    http://controller:6080/vnc_auto.html?token=58707002-24c3-4c3f-adc7-6ecbd1c97a53 |
5.  +-------+-----------------------------------------------------------------------------
    ----+
```

获取的 url 为 http://controller:6080/vnc_auto.html?token=58707002-24c3-4c3f-adc7-6ecbd1c97a53 ，这里需要注意本地一定要配置 host

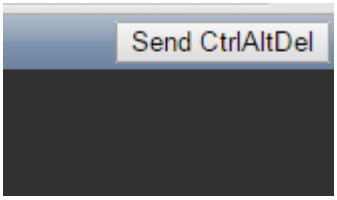
否则不能访问，获知直接使用 ip 为

http://10.0.0.11:6080/vnc_auto.html?token=58707002-24c3-4c3f-adc7-6ecbd1c97a53

也可以通过界面登录：

http://10.0.0.11/horizon

找到实例，如果控制台没有相应，则单击，下面图标即可



登录的用户名和密码都会有提示：用户名为：

cirros

密码为：

cubswin:)

验证 demo-net 租户网关

```
1.  ping -c 4 192.168.1.1
```

```
$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: seq=0 ttl=64 time=2.474 ms
64 bytes from 192.168.1.1: seq=1 ttl=64 time=1.546 ms
64 bytes from 192.168.1.1: seq=2 ttl=64 time=1.084 ms
64 bytes from 192.168.1.1: seq=3 ttl=64 time=1.122 ms

--- 192.168.1.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 1.084/1.556/2.474 ms
$
```

```
1.  ping -c 203.0.113.101
```

```
$ ping 203.0.113.101
PING 203.0.113.101 (203.0.113.101): 56 data bytes
64 bytes from 203.0.113.101: seq=0 ttl=64 time=1.703 ms
64 bytes from 203.0.113.101: seq=1 ttl=64 time=1.290 ms

--- 203.0.113.101 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 1.290/1.496/1.703 ms
$
```

##

问题：

1.创建实例失败

Status 处于 BUILD 状态，TASK State 处于 Scheduling 状态，
由于实例的创建是由 nova 主要来完成的，因此 nova 的各个进程，一定要起来。
所以通过，查看有 6 个进程，结果好几个出了问题。

```
1.  nova-manage service list
```

这时候，就找到了问题的症结，剩下的就是到/var/log/nova 下面查看日志了。

```
1.  service nova-api restart

2.  service nova-cert restart

3.  service nova-consoleauth restart

4.  service nova-scheduler restart

5.  service nova-conductor restart

6.  service nova-novncproxy restart

7.
```

然后重启，上面的服务。下面服务都正常，才可创建实例

2.openstack 创建实例，获取不到 ip 地址，不能 ping 通网关

ping -c 4 192.168.1.1
ping 不通网关，这里的原因很多，通过查看日志
应该是

```
1.  ERROR neutron.plugins.openvswitch.agent.ovs_neutron_agent

    [req-54efa1c5-7808-4d75-a047-522c0e6ee052 None] Error while processing VIF ports
```

错误，更多内容：

```
1.  2015-02-23 18:22:12.346 1979 ERROR neutron.plugins.openvswitch.agent.ovs_neutron_agent
    [req-54efa1c5-7808-4d75-a047-522c0e6ee052 None] Error while processing VIF ports
2.  2015-02-23 18:22:12.346 1979 TRACE neutron.plugins.openvswitch.agent.ovs_neutron_agent
    Traceback (most recent call last):
3.  2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent   File
    "/usr/lib/python2.7/dist-packages/neutron/plugins/openvswitch/agent/ovs_neutron_agent.
    py", line 1406, in rpc_loop
4.  2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent     ovs_restarted)
5.  2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent   File
    "/usr/lib/python2.7/dist-packages/neutron/plugins/openvswitch/agent/ovs_neutron_agent.
    py", line 1205, in process_network_ports
6.  2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent     port_info.get('updated',
    set()))
7.  2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent   File
    "/usr/lib/python2.7/dist-packages/neutron/agent/securitygroups_rpc.py", line 333, in
    setup_port_filters
8.  2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent     self.prepare_devices_filter(ne
    w_devices)
9.  2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent   File
    "/usr/lib/python2.7/dist-packages/neutron/agent/securitygroups_rpc.py", line 202, in
    decorated_function
10. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent     return func(self, *args,
    **kwargs)
```

```
11. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent   File
    "/usr/lib/python2.7/dist-packages/neutron/agent/securitygroups_rpc.py", line 210, in
    prepare_devices_filter
12. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent     if self.use_enhanced_rpc:
13. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent   File
    "/usr/lib/python2.7/dist-packages/neutron/agent/securitygroups_rpc.py", line 178, in
    use_enhanced_rpc
14. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent     self._check_enhanced_rpc_is_su
    pported_by_server())
15. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent   File
    "/usr/lib/python2.7/dist-packages/neutron/agent/securitygroups_rpc.py", line 184, in
    _check_enhanced_rpc_is_supported_by_server
16. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent     self.context, devices=[])
17. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent   File
    "/usr/lib/python2.7/dist-packages/neutron/agent/securitygroups_rpc.py", line 103, in
    security_group_info_for_devices
18. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent     version='1.2')
19. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent   File
    "/usr/lib/python2.7/dist-packages/neutron/common/log.py", line 34, in wrapper
20. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent     return method(*args, **kwargs)
```

```
21. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent   File
    "/usr/lib/python2.7/dist-packages/neutron/common/rpc.py", line 161, in call
22. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent     context, msg, rpc_method='call',
    **kwargs)
23. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent   File
    "/usr/lib/python2.7/dist-packages/neutron/common/rpc.py", line 187, in __call_rpc_method
24. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent     return func(context,
    msg['method'], **msg['args'])
25. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent   File
    "/usr/lib/python2.7/dist-packages/oslo/messaging/rpc/client.py", line 152, in call
26. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent     retry=self.retry)
27. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent   File
    "/usr/lib/python2.7/dist-packages/oslo/messaging/transport.py", line 90, in _send
28. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent     timeout=timeout, retry=retry)
29. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent   File
    "/usr/lib/python2.7/dist-packages/oslo/messaging/_drivers/amqpdriver.py", line 408, in
    send
30. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent     retry=retry)
31. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent   File
    "/usr/lib/python2.7/dist-packages/oslo/messaging/_drivers/amqpdriver.py", line 397, in
    _send
```

```
32. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent     result =
    self._waiter.wait(msg_id, timeout)
33. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent   File
    "/usr/lib/python2.7/dist-packages/oslo/messaging/_drivers/amqpdriver.py", line 285, in
    wait
34. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent     reply, ending =
    self._poll_connection(msg_id, timeout)
35. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent   File
    "/usr/lib/python2.7/dist-packages/oslo/messaging/_drivers/amqpdriver.py", line 235, in
    _poll_connection
36. 2015-02-23 18:22:12.346 1979 TRACE
    neutron.plugins.openvswitch.agent.ovs_neutron_agent     % msg_id)
37. 2015-02-23 18:22:12.346 1979 TRACE neutron.plugins.openvswitch.agent.ovs_neutron_agent
    MessagingTimeout: Timed out waiting for a reply to message ID
    cef18a2f5cea4036967ddef86eed4cf1
38. 2015-02-23 18:22:12.346 1979 TRACE neutron.plugins.openvswitch.agent.ovs_neutron_agent
39. 2015-02-23 18:22:12.404 1979 INFO neutron.plugins.openvswitch.agent.ovs_neutron_agent
    [req-54efa1c5-7808-4d75-a047-522c0e6ee052 None] Agent out of sync with plugin!
40. 2015-02-23 18:22:12.708 1979 INFO neutron.agent.securitygroups_rpc
    [req-54efa1c5-7808-4d75-a047-522c0e6ee052 None] Preparing filters for devices
    set([u'92df61ad-21b3-4697-9658-7299193ebc97'])
41. 2015-02-23 18:22:14.019 1979 INFO neutron.plugins.openvswitch.agent.ovs_neutron_agent
    [req-54efa1c5-7808-4d75-a047-522c0e6ee052 None] Port
    92df61ad-21b3-4697-9658-7299193ebc97 updated. Details: {u'profile': {},
    u'admin_state_up': True, u'network_id': u'299e4203-135e-4305-8a9f-cdd4f4bda185',
    u'segmentation_id': 1, u'device_owner': u'compute:None', u'physical_network': None,
    u'mac_address': u'fa:16:3e:50:2d:43', u'device':
    u'92df61ad-21b3-4697-9658-7299193ebc97', u'port_id':
```

```
    u'92df61ad-21b3-4697-9658-7299193ebc97', u'fixed_ips': [{u'subnet_id':
    u'dfb5a684-e837-4556-8c29-c7e01ada590a', u'ip_address': u'192.168.1.2'}],
    u'network_type': u'gre'}
42. 2015-02-23 18:22:14.021 1979 INFO neutron.plugins.openvswitch.agent.ovs_neutron_agent
    [req-54efa1c5-7808-4d75-a047-522c0e6ee052 None] Assigning 1 as local vlan for
    net-id=299e4203-135e-4305-8a9f-cdd4f4bda185
43. 2015-02-23 18:22:14.467 1979 INFO neutron.plugins.openvswitch.agent.ovs_neutron_agent
    [req-54efa1c5-7808-4d75-a047-522c0e6ee052 None] Configuration for device
    92df61ad-21b3-4697-9658-7299193ebc97 completed.
```

为什么会产生上面的问题，原因真不少。

比如你的网卡 ip 地址是否有问题。你的 dhpc agent 是否有问题，RabbitMQ 是否正常。

这些都会产生获取不到 ip 的情况。

我这里则是 dhcp agent 的问题。

进行下面配置（网络节点）

1.编辑文件 /etc/neutron/dhcp_agent.ini，完成下面内容。

a.在 [DEFAULT] 部分，配置驱动，启用命名空间和启用删除废弃的命名空间

```
1.  [DEFAULT]
2.  ...
3.  interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
4.  dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
5.  use_namespaces = True
6.  dhcp_delete_namespaces = True
```

更多内容参考

openstack【juno】入门 【网络篇】十六：neutron 安装部署（网络节点）

openstack【juno】入门 【总结篇】二十七：openstack 排除故障及常见问题记录

问题导读

1.如何使用 neutron 命令？
2.如何查看具体某个命令的参数解释？
3.如何查看 ovs 创建网桥

创建实例失败：

首先用下面命令查看服务是否正常

```
1.  nova-manage service list
```

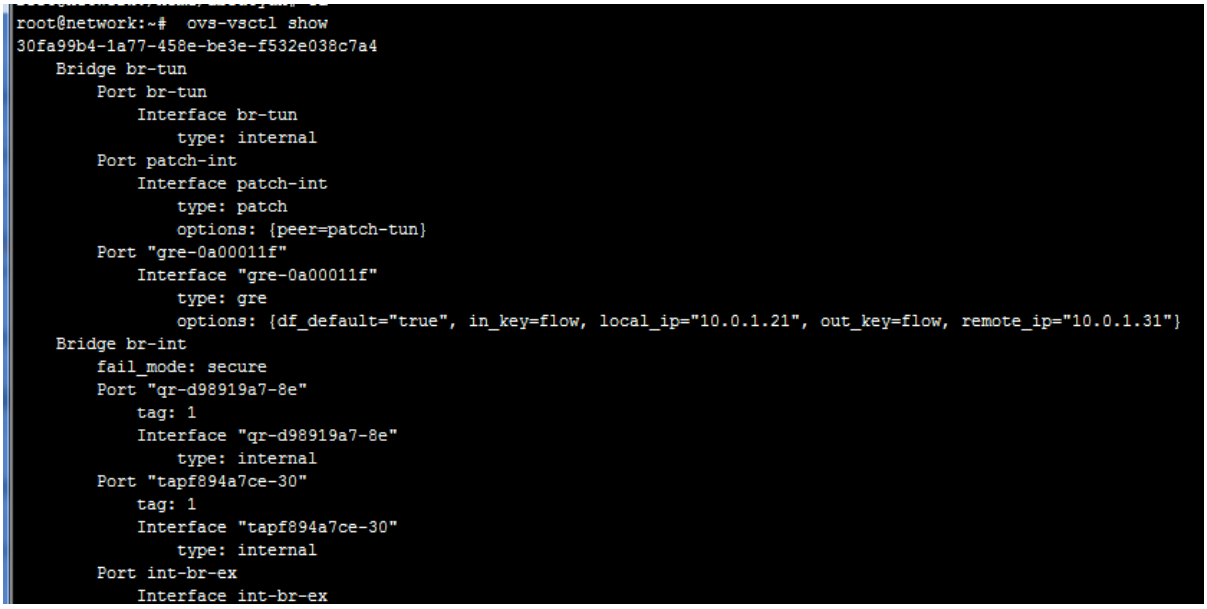
如果不正常，则使用下面命令重启，如果还不行，则查看日志，

```
1.  service nova-api restart
2.  service nova-cert restart
3.  service nova-consoleauth restart
4.  service nova-scheduler restart
5.  service nova-conductor restart
6.  service nova-novncproxy restart
```

对网络的判断：

可以查看网桥

```
1.  ovs-vsctl show
```



如果不会使用 ovs，则可以使用

```
1.  ovs-vsctl  --help
```

如果看不懂，则查看 openstack Juno neutron 必读系列文章：

网络流量查看：

```
1.  udhcpc eth0
```

neutron 命令的使用

```
1.  neutron --help
```

会输出很多内容

```
1.  --version            show program's version number and exit
2.    -v, --verbose, --debug
3.                         Increase verbosity of output and show tracebacks on
4.                         errors. You can repeat this option.
5.    -q, --quiet          Suppress output except warnings and errors.
6.    -h, --help           Show this help message and exit.
```

```
7.    -r NUM, --retries NUM
8.                          How many times the request to the Neutron server
9.                          should be retried if it fails.
10.   --os-service-type <os-service-type>
11.                         Defaults to env[OS_NETWORK_SERVICE_TYPE] or network.
12.   --os-endpoint-type <os-endpoint-type>
13.                         Defaults to env[OS_ENDPOINT_TYPE] or publicURL.
14.   --service-type <service-type>
15.                         DEPRECATED! Use --os-service-type.
16.   --endpoint-type <endpoint-type>
17.                         DEPRECATED! Use --os-endpoint-type.
18.   --os-auth-strategy <auth-strategy>
19.                         DEPRECATED! Only keystone is supported.
20.   --os-auth-url <auth-url>
21.                         Authentication URL, defaults to env[OS_AUTH_URL].
22.   --os-tenant-name <auth-tenant-name>
23.                         Authentication tenant name, defaults to
24.                         env[OS_TENANT_NAME].
25.   --os-project-name <auth-project-name>
26.                         Another way to specify tenant name. This option is
27.                         mutually exclusive with --os-tenant-name. Defaults to
28.                         env[OS_PROJECT_NAME].
29.   --os-tenant-id <auth-tenant-id>
30.                         Authentication tenant ID, defaults to
31.                         env[OS_TENANT_ID].
32.   --os-project-id <auth-project-id>
33.                         Another way to specify tenant ID. This option is
34.                         mutually exclusive with --os-tenant-id. Defaults to
35.                         env[OS_PROJECT_ID].
36.   --os-username <auth-username>
37.                         Authentication username, defaults to env[OS_USERNAME].
38.   --os-user-id <auth-user-id>
```

```
                        Authentication user ID (Env: OS_USER_ID)
    --os-user-domain-id <auth-user-domain-id>
                        OpenStack user domain ID. Defaults to
                        env[OS_USER_DOMAIN_ID].
    --os-user-domain-name <auth-user-domain-name>
                        OpenStack user domain name. Defaults to
                        env[OS_USER_DOMAIN_NAME].
    --os-project-domain-id <auth-project-domain-id>
                        Defaults to env[OS_PROJECT_DOMAIN_ID].
    --os-project-domain-name <auth-project-domain-name>
                        Defaults to env[OS_PROJECT_DOMAIN_NAME].
    --os-cert <certificate>
                        Path of certificate file to use in SSL connection.
                        This file can optionally be prepended with the private
                        key. Defaults to env[OS_CERT].
    --os-cacert <ca-certificate>
                        Specify a CA bundle file to use in verifying a TLS
                        (https) server certificate. Defaults to
                        env[OS_CACERT].
    --os-key <key>       Path of client key to use in SSL connection. This
                        option is not necessary if your key is prepended to
                        your certificate file. Defaults to env[OS_KEY].
    --os-password <auth-password>
                        Authentication password, defaults to env[OS_PASSWORD].
    --os-region-name <auth-region-name>
                        Authentication region name, defaults to
                        env[OS_REGION_NAME].
    --os-token <token>    Authentication token, defaults to env[OS_TOKEN].
    --http-timeout <seconds>
                        Timeout in seconds to wait for an HTTP response.
                        Defaults to env[OS_NETWORK_TIMEOUT] or None if not
                        specified.
```

```
71.    --os-url <url>        Defaults to env[OS_URL].
72.    --insecure            Explicitly allow neutronclient to perform "insecure"
73.                          SSL (https) requests. The server's certificate will
74.                          not be verified against any certificate authorities.
75.                          This option should be used with caution.
76.
77. Commands for API v2.0:
78.    agent-delete                  Delete a given agent.
79.    agent-list                    List agents.
80.    agent-show                    Show information of a given agent.
81.    agent-update                  Update a given agent.
82.    cisco-credential-create       Creates a credential.
83.    cisco-credential-delete       Delete a  given credential.
84.    cisco-credential-list         List credentials that belong to a given tenant.
85.    cisco-credential-show         Show information of a given credential.
86.    cisco-network-profile-create  Creates a network profile.
87.    cisco-network-profile-delete  Delete a given network profile.
88.    cisco-network-profile-list    List network profiles that belong to a given tenant.
89.    cisco-network-profile-show    Show information of a given network profile.
90.    cisco-network-profile-update  Update network profile's information.
91.    cisco-policy-profile-list     List policy profiles that belong to a given tenant.
92.    cisco-policy-profile-show     Show information of a given policy profile.
93.    cisco-policy-profile-update   Update policy profile's information.
94.    complete                      print bash completion command
95.    dhcp-agent-list-hosting-net   List DHCP agents hosting a network.
96.    dhcp-agent-network-add        Add a network to a DHCP agent.
97.    dhcp-agent-network-remove     Remove a network from a DHCP agent.
98.    ext-list                      List all extensions.
99.    ext-show                      Show information of a given resource.
100.   firewall-create               Create a firewall.
101.   firewall-delete               Delete a given firewall.
102.   firewall-list                 List firewalls that belong to a given tenant.
```

```
103.  firewall-policy-create         Create a firewall policy.
104.  firewall-policy-delete         Delete a given firewall policy.
105.  firewall-policy-insert-rule    Insert a rule into a given firewall policy.
106.  firewall-policy-list           List firewall policies that belong to a given tenant.
107.  firewall-policy-remove-rule    Remove a rule from a given firewall policy.
108.  firewall-policy-show           Show information of a given firewall policy.
109.  firewall-policy-update         Update a given firewall policy.
110.  firewall-rule-create           Create a firewall rule.
111.  firewall-rule-delete           Delete a given firewall rule.
112.  firewall-rule-list             List firewall rules that belong to a given tenant.
113.  firewall-rule-show             Show information of a given firewall rule.
114.  firewall-rule-update           Update a given firewall rule.
115.  firewall-show                  Show information of a given firewall.
116.  firewall-update                Update a given firewall.
117.  floatingip-associate           Create a mapping between a floating IP and a fixed IP.
118.  floatingip-create              Create a floating IP for a given tenant.
119.  floatingip-delete              Delete a given floating IP.
120.  floatingip-disassociate        Remove a mapping from a floating IP to a fixed IP.
121.  floatingip-list                List floating IPs that belong to a given tenant.
122.  floatingip-show                Show information of a given floating IP.
123.  gateway-device-create          Create a network gateway device.
124.  gateway-device-delete          Delete a given network gateway device.
125.  gateway-device-list            List network gateway devices for a given tenant.
126.  gateway-device-show            Show information for a given network gateway device.
127.  gateway-device-update          Update a network gateway device.
128.  help                          print detailed help for another command
129.  ipsec-site-connection-create   Create an IPsec site connection.
130.  ipsec-site-connection-delete   Delete a given IPsec site connection.
131.  ipsec-site-connection-list     List IPsec site connections that belong to a given tenant.
132.  ipsec-site-connection-show     Show information of a given IPsec site connection.
133.  ipsec-site-connection-update   Update a given IPsec site connection.
134.  l3-agent-list-hosting-router   List L3 agents hosting a router.
```

```
135.  l3-agent-router-add           Add a router to a L3 agent.
136.  l3-agent-router-remove        Remove a router from a L3 agent.
137.  lb-agent-hosting-pool         Get loadbalancer agent hosting a pool.
138.  lb-healthmonitor-associate    Create a mapping between a health monitor and a pool.
139.  lb-healthmonitor-create       Create a healthmonitor.
140.  lb-healthmonitor-delete       Delete a given healthmonitor.
141.  lb-healthmonitor-disassociate Remove a mapping from a health monitor to a pool.
142.  lb-healthmonitor-list         List healthmonitors that belong to a given tenant.
143.  lb-healthmonitor-show         Show information of a given healthmonitor.
144.  lb-healthmonitor-update       Update a given healthmonitor.
145.  lb-member-create              Create a member.
146.  lb-member-delete              Delete a given member.
147.  lb-member-list                List members that belong to a given tenant.
148.  lb-member-show                Show information of a given member.
149.  lb-member-update              Update a given member.
150.  lb-pool-create                Create a pool.
151.  lb-pool-delete                Delete a given pool.
152.  lb-pool-list                  List pools that belong to a given tenant.
153.  lb-pool-list-on-agent         List the pools on a loadbalancer agent.
154.  lb-pool-show                  Show information of a given pool.
155.  lb-pool-stats                 Retrieve stats for a given pool.
156.  lb-pool-update                Update a given pool.
157.  lb-vip-create                 Create a vip.
158.  lb-vip-delete                 Delete a given vip.
159.  lb-vip-list                   List vips that belong to a given tenant.
160.  lb-vip-show                   Show information of a given vip.
161.  lb-vip-update                 Update a given vip.
162.  meter-label-create            Create a metering label for a given tenant.
163.  meter-label-delete            Delete a given metering label.
164.  meter-label-list              List metering labels that belong to a given tenant.
165.  meter-label-rule-create       Create a metering label rule for a given label.
166.  meter-label-rule-delete       Delete a given metering label.
```

```
167.  meter-label-rule-list          List metering labels that belong to a given label.
168.  meter-label-rule-show          Show information of a given metering label rule.
169.  meter-label-show               Show information of a given metering label.
170.  nec-packet-filter-create       Create a packet filter for a given tenant.
171.  nec-packet-filter-delete       Delete a given packet filter.
172.  nec-packet-filter-list         List packet filters that belong to a given tenant.
173.  nec-packet-filter-show         Show information of a given packet filter.
174.  nec-packet-filter-update       Update packet filter's information.
175.  net-create                     Create a network for a given tenant.
176.  net-delete                     Delete a given network.
177.  net-external-list               List external networks that belong to a given tenant.
178.  net-gateway-connect            Add an internal network interface to a router.
179.  net-gateway-create             Create a network gateway.
180.  net-gateway-delete             Delete a given network gateway.
181.  net-gateway-disconnect         Remove a network from a network gateway.
182.  net-gateway-list               List network gateways for a given tenant.
183.  net-gateway-show               Show information of a given network gateway.
184.  net-gateway-update             Update the name for a network gateway.
185.  net-list                       List networks that belong to a given tenant.
186.  net-list-on-dhcp-agent         List the networks on a DHCP agent.
187.  net-show                       Show information of a given network.
188.  net-update                     Update network's information.
189.  nuage-netpartition-create      Create a netpartition for a given tenant.
190.  nuage-netpartition-delete      Delete a given netpartition.
191.  nuage-netpartition-list        List netpartitions that belong to a given tenant.
192.  nuage-netpartition-show        Show information of a given netpartition.
193.  port-create                     Create a port for a given tenant.
194.  port-delete                     Delete a given port.
195.  port-list                       List ports that belong to a given tenant.
196.  port-show                       Show information of a given port.
197.  port-update                     Update port's information.
198.  queue-create                    Create a queue.
```

```
199.  queue-delete                  Delete a given queue.
200.  queue-list                    List queues that belong to a given tenant.
201.  queue-show                    Show information of a given queue.
202.  quota-delete                  Delete defined quotas of a given tenant.
203.  quota-list                    List quotas of all tenants who have non-default quota
    values.
204.  quota-show                    Show quotas of a given tenant.
205.  quota-update                  Define tenant's quotas not to use defaults.
206.  router-create                  Create a router for a given tenant.
207.  router-delete                  Delete a given router.
208.  router-gateway-clear           Remove an external network gateway from a router.
209.  router-gateway-set             Set the external network gateway for a router.
210.  router-interface-add           Add an internal network interface to a router.
211.  router-interface-delete        Remove an internal network interface from a router.
212.  router-list                    List routers that belong to a given tenant.
213.  router-list-on-l3-agent        List the routers on a L3 agent.
214.  router-port-list               List ports that belong to a given tenant, with specified
    router.
215.  router-show                    Show information of a given router.
216.  router-update                  Update router's information.
217.  security-group-create          Create a security group.
218.  security-group-delete          Delete a given security group.
219.  security-group-list            List security groups that belong to a given tenant.
220.  security-group-rule-create     Create a security group rule.
221.  security-group-rule-delete     Delete a given security group rule.
222.  security-group-rule-list       List security group rules that belong to a given tenant.
223.  security-group-rule-show       Show information of a given security group rule.
224.  security-group-show            Show information of a given security group.
225.  security-group-update          Update a given security group.
226.  service-provider-list          List service providers.
227.  subnet-create                  Create a subnet for a given tenant.
228.  subnet-delete                  Delete a given subnet.
```

```
229.  subnet-list                  List subnets that belong to a given tenant.
230.  subnet-show                  Show information of a given subnet.
231.  subnet-update                Update subnet's information.
232.  vpn-ikepolicy-create         Create an IKE policy.
233.  vpn-ikepolicy-delete         Delete a given IKE policy.
234.  vpn-ikepolicy-list           List IKE policies that belong to a tenant.
235.  vpn-ikepolicy-show           Show information of a given IKE policy.
236.  vpn-ikepolicy-update         Update a given IKE policy.
237.  vpn-ipsecpolicy-create       Create an IPsec policy.
238.  vpn-ipsecpolicy-delete       Delete a given IPsec policy.
239.  vpn-ipsecpolicy-list         List ipsecpolicies that belongs to a given tenant
    connection.
240.  vpn-ipsecpolicy-show         Show information of a given IPsec policy.
241.  vpn-ipsecpolicy-update       Update a given IPsec policy.
242.  vpn-service-create           Create a VPN service.
243.  vpn-service-delete           Delete a given VPN service.
244.  vpn-service-list             List VPN service configurations that belong to a given
    tenant.
245.  vpn-service-show             Show information of a given VPN service.
246.  vpn-service-update           Update a given VPN service.
```

关键这些命令该如何使用：

例如我们想查看网络

```
1.  neutron net-list
```

```
root@controller:~# neutron net-list
+--------------------------------------+----------+--------------------------------------------------+
| id                                   | name     | subnets                                          |
+--------------------------------------+----------+--------------------------------------------------+
| d12d09bf-fbe4-43e8-81d1-5ecc38b43f81 | demo-net | 0e6512ab-f494-490e-8cd8-03ac4c54a179 192.168.1.0/24 |
| a6155046-7aa3-4c8e-8d36-7bcf12cf6864 | ext-net  | 19302ba6-9990-4e18-b5ad-a41a57c3b319 203.0.113.0/24 |
+--------------------------------------+----------+--------------------------------------------------+
```

如果我们查看某个特定的信息，则使用 show 命令

比如我们想查看上述网络的 demo-net 具体信息。其他的依次类推

```
1.  neutron net-show  demo-net
```

```
root@controller:~# neutron net-show  demo-net
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | d12d09bf-fbe4-43e8-81d1-5ecc38b43f81 |
| name                      | demo-net                             |
| provider:network_type     | gre                                  |
| provider:physical_network |                                      |
| provider:segmentation_id  | 1                                    |
| router:external           | False                                |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   | 0e6512ab-f494-490e-8cd8-03ac4c54a179 |
| tenant_id                 | bfb31e69f05b44cd89e1336c09042e2f     |
+---------------------------+--------------------------------------+
```

而有些比较复杂，比如我们想删除路由的某个端口，
我们不知道后面跟什么参数，只看到了命令，该如何处理

```
1.  router-interface-delete
```

使用下面命令

```
1.  neutron  router-interface-delete --help
```

列出下面参数：
也就是需要一个路由的 id，以及 SUBNET|subnet=SUBNET|port=PORT 这些参数都是可以的

```
1.  usage: neutron router-interface-delete [-h] [--request-format {json,xml}]
2.                                         router-id INTERFACE
3.
4.  Remove an internal network interface from a router.
5.
6.  positional arguments:
7.    router-id             ID of the router.
8.    INTERFACE             The format is "SUBNET|subnet=SUBNET|port=PORT". Either
9.                          a subnet or port must be specified. Both ID and name
10.                         are accepted as SUBNET or PORT. Note that "subnet="
11.                         can be omitted when specifying a subnet.
12.
```

```
13. optional arguments:
14.   -h, --help            show this help message and exit
15.   --request-format {json,xml}
16.                         The XML or JSON request format.
```

```
root@controller:~# neutron router-interface-delete --help
usage: neutron router-interface-delete [-h] [--request-format {json,xml}]
                                       router-id INTERFACE

Remove an internal network interface from a router.

positional arguments:
  router-id             ID of the router.
  INTERFACE             The format is "SUBNET|subnet=SUBNET|port=PORT". Either
                        a subnet or port must be specified. Both ID and name
                        are accepted as SUBNET or PORT. Note that "subnet="
                        can be omitted when specifying a subnet.

optional arguments:
  -h, --help            show this help message and exit
  --request-format {json,xml}
                        The XML or JSON request format.
```

这样我们根据英文含义及参数详细信息，就会使用 neutron 的命令了

openstack【juno】入门 【总结篇】二十八：keystone 及网络总结

问题导读

1.你认为 openstack 的 keystone 的作用是什么？
2.总结 keystone 与我们传统的验证区别在什么地方？
3.登录 dashboard 的用户名和密码是由谁来创建的？
4.你对 openstack 的网络了解多少？

keystone 认识

这里是对 keystone 有所了解，但是又不太懂 keystone 到底是什么同学的一个解惑。
它包含了几个概念，这里不做讨论，可以参考
什么是 Keystone，Keystone 概念介绍

Keystone, Openstack 之魂

这里只介绍下面内容：

他们是什么，是怎么来的：
这个是 admin 租户，是怎么来的那，这里是有 keystone 来创建的详细见：
openstack【juno】入门 【keystone 篇】六：：Keystone 使用及遇到问题解决办法

```
1.  export OS_TENANT_NAME=admin

2.  export OS_USERNAME=admin

3.  export OS_PASSWORD=ADMIN_PASS

4.  export OS_AUTH_URL=http://controller:35357/v2.0
```

那么 keystone 创建它能用来做什么，下面我们用 web 界面，或许能够更加的直观。

我们这系列文章管理网络控制节点 10.0.0.11,并且同时添加了 dashboard，如果没有添加，可以参考

openstack【juno】入门 【dashboard 篇】十九：添加 dashboard

我们输入

```
1.  http://10.0.0.11/horizon
```

看到下面界面，然后，我们输入的用户名和密码是什么那，其实就是

用户名：admin

密码：ADMIN_PASS

这时候你是否对 keystone 创建的 admin 有所了解的了。那么我们在 Linux 下的操作又是什么？

```
1.  source admin-openrc.sh
```

内容：

```
1.  export OS_TENANT_NAME=admin

2.  export OS_USERNAME=admin

3.  export OS_PASSWORD=ADMIN_PASS

4.  export OS_AUTH_URL=http://controller:35357/v2.0
```

其实这个等同于我们在界面的登录操作。

上面的是管理员帐户，那么我们的租户帐户是什么那？

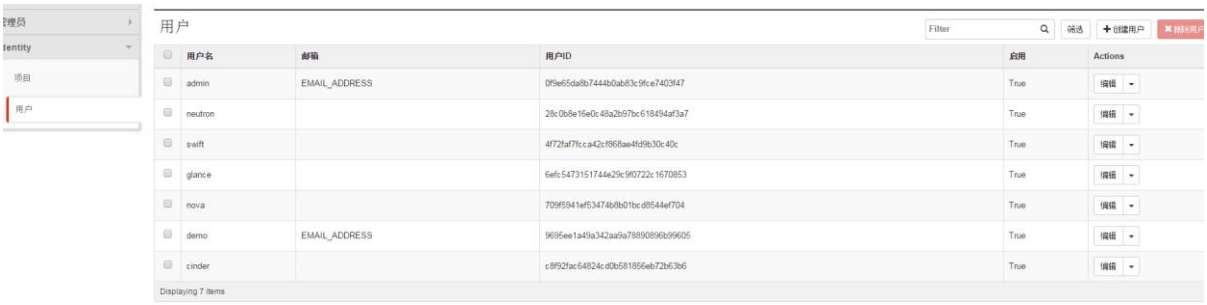
道理是一样的，我们输入

demo 帐户，同样可以登录，只不过看到的内容不一样而已，如果做过传统的开发，这里相信一点就通了。

```
1.  export OS_TENANT_NAME=demo

2.  export OS_USERNAME=demo

3.  export OS_PASSWORD=DEMO_PASS

4.  export OS_AUTH_URL=http://controller:5000/v2.0
```

那么我们还有那些帐户，比如 glance 帐户

这些都是可以登录使用的，那么到底有多少帐户
我们可以登录 admin 帐户进行查看



网络总结

网络显然比 keystone 更复杂一些，包括 ovs 的使用，neutron 的使用等等。 相关内容，例如：
openstack【juno】入门 【总结篇】二十七：openstack 排除故障及常见问题记录
推荐比较经典的文章
openstack Juno neutron 必读 1：在 Mac 上部署 Juno 版本 OpenStack 四节点环境介绍

openstack Juno neutron 必读 2：Neutron 深入学习之 OVS + GRE 之 Compute node

openstack Juno neutron 必读 3：Neutron 深入探索之 OVS + GRE 之 完整网络流程 篇

相关文章：
零基础学习 **openstack**【完整中级篇】及 **openstack** 资源汇总
http://www.aboutyun.com/thread-10306-1-1.html

文章会随时更新，如有错误，欢迎批评指正，在此基础上会对其扩展录制视频，可以预购，亦可捐助 about 云

作为新手入门入门，同时还包括 openstack 零基础开发。视频及文章都会陆续更新，包括后面的 k 版本。

如果想关注 about 云，可以通过下面方式：

1.加入 about 云群 371358502、322273151、90371779，云计算大数据爱好者群

2.关注腾讯认证空间

about 云腾讯认证空间

3.惯用手机的用户可关注 about 云微信地址：

搜索：
wwwaboutyuncom

4.关注微博：

新浪微博

5.邮件订阅

邮件订阅

2014 过去，2015 年来临，2014 年，about 云也有一年多，云技术、大数据迅速火了起来。过去的一年中，我们都尽其所能的为大家提供技术性实践资料、文章、视频。但是总的来说，还是不够。我们将一如既往的，出一些技术性文章。引导新手入门及学习，同时希望大数据、云技术爱好者多多支持 about 云，同时我们将会出一些视频，也欢迎捐助 about 云。

在 IT 行业，技术不断更新换代，不断抛陈出新，5 年不学习新知识，就会 out 了。about 云创立之初，面对资料的缺乏，与大家共同学习，相互分享学习经验，帮助了很多初学者入门，about 云在 2015 年将会更努力，成为更多人的大数据、云技术学习分享基地。

同时希望接触 about 云，知道 about 云，了解 about 云的人，about 云能传播给大家一种学习精神----活到老，学到老。

2015 年，about 云为了目标将会不断的学习，努力、坚持，帮助更多的人，更多的大数据、云技术爱好者。