OpenStack安装手册-CentOS

笔记本： openstack
创建时间： 2013/12/27 11:12 更新时间： 2014/1/5 16:59
URL： http://docs.openstack.org/havana/install-guide/install/yum/content/

基础系统配置

网络配置

```
Control
     |-----eth0:192.168.0.70
     |-----eth1:172.16.0.*

hostname controller
EX_IP=`ifconfig eth0|sed -n 's/\(.*:\)\(\([0-9]\{1,3\}\.\)\{3\}[0-9]\{1,3\}\)\
(.*Bcast:.*\)/\2/g;/\([0-9]\{1,3\}\.\)\{3\}/p'`
sed -i --posix "s/\(HOSTNAME=\)\(.*\)/\1`hostname`/g" /etc/sysconfig/network
echo -e "$EX_IP `hostname`\n192.168.0.71 compute1"  >> /etc/hosts

Compute
     |-----eth0:192.168.0.71
     |-----eth1:172.16.0.*

hostname compute1
EX_IP=`ifconfig eth0|sed -n 's/\(.*:\)\(\([0-9]\{1,3\}\.\)\{3\}[0-9]\{1,3\}\)\
(.*Bcast:.*\)/\2/g;/\([0-9]\{1,3\}\.\)\{3\}/p'`
sed -i --posix "s/\(HOSTNAME=\)\(.*\)/\1`hostname`/g" /etc/sysconfig/network
echo -e "$EX_IP `hostname`\n192.168.0.70 controller"  >> /etc/hosts
```

密码

产生一个密码

```
openssl rand -hex 10
c5aa299d3d5a746d0618
```

Password name

Description

Database password (no variable used)

数据库的root密码

KEYSTONE_DBPASS

Identity 服务的数据库密码

ADMIN_PASS

admin 用户的密码

GLANCE_DBPASS

Image 服务的数据库密码

GLANCE_PASS

Image 服务 glance 用户的密码

NOVA_DBPASS

Compute 服务的数据库密码

NOVA_PASS

Compute 服务 nova 用户的密码

DASH_DBPASS

dashboard 的数据库密码

CINDER_DBPASS

Block Storage 服务的数据库密码

CINDER_PASS

Storage 服务 cinder 用户的密码

NEUTRON_DBPASS

Networking 服务的密码

NEUTRON_PASS

Networking 服务 neutron 用户的密码

HEAT_DBPASS

Orchestration 服务的数据库密码

HEAT_PASS

Orchestration 服务 heat 用户的密码

CEILOMETER_DBPASS

Telemetry 服务的密码

CEILOMETER_PASS

Telemetry 服务 ceilometer 用户的密码

数据库

控制节点

```
useradd mysql
yum -y install bison bison-devel ncurses-devel cmake
wget http://www.percona.com/redir/downloads/Percona-Server-5.6/LATEST/source/Percona-Server-5.6.15-rel63.0.tar.gz
tar zxf Percona-Server-5.6.15-rel63.0.tar.gz
cd Percona-Server-5.6.15-rel63.0
CC=gcc CFLAGS="-DBIG_JOINS=1 -DHAVE_DLOPEN=1 -O3" CXX=g++ CXXFLAGS="-DBIG_JOINS=1 -DHAVE_DLOPEN=1 -felide-constructors -fno-rtti -O3"
cmake -DCMAKE_INSTALL_PREFIX=/usr/local/mysql \
-DMYSQL_UNIX_ADDR=/tmp/mysql.sock \
-DDEFAULT_CHARSET=utf8 \
-DDEFAULT_COLLATION=utf8_general_ci \
-DWITH_EXTRA_CHARSETS:STRING=all \
-DWITH_PARTITION_STORAGE_ENGINE:BOOL=1 \
-DWITH_BLACKHOLE_STORAGE_ENGINE:BOOL=1 \
-DINSTALL_LAYOUT:STRING=STANDALONE  \
-DWITH_MYISAM_STORAGE_ENGINE=1 \
-DWITH_INNOBASE_STORAGE_ENGINE=1 \
-DWITH_ARCHIVE_STORAGE_ENGINE:BOOL=1 \
-DWITH_FEDERATED_STORAGE_ENGINE:BOOL=1 \
```

```
-DWITH_READLINE=1 \
-DENABLED_LOCAL_INFILE=1 \
-DMYSQL_DATADIR=/var/mysql/data \
-DWITH_BIG_TABLES=1 \
-DWITH_DEBUG=0
make && make install

echo "/usr/local/mysql/lib" >> /etc/ld.so.conf && ldconfig

chmod +w /usr/local/mysql
chown -R mysql:mysql /usr/local/mysql

mkdir -p /var/mysql/
mkdir -p /var/mysql/data/
mkdir -p /var/mysql/log/
chown -R mysql:mysql /var/mysql/

cd support-files/
cp mysql.server /etc/init.d/mysqld
cp my-default.cnf /usr/local/mysql/my.cnf

/usr/local/mysql/scripts/mysql_install_db \
    --defaults-file=/usr/local/mysql/my.cnf \
    --basedir=/usr/local/mysql \
    --datadir=/var/mysql/data \
    --user=mysql

chmod +x /etc/init.d/mysqld
sed -i '/^# basedir/ cbasedir=/usr/local/mysql' /etc/init.d/mysqld
sed -i '/^# datadir/ cdatadir=/var/mysql/data' /etc/init.d/mysqld
sed -i '$askip-name-resolve' /usr/local/mysql/my.cnf
sed -i '$abasedir=/usr/local/mysql' /usr/local/mysql/my.cnf
sed -i '$adatadir=/var/mysql/data' /usr/local/mysql/my.cnf

echo 'export PATH=$PATH:/usr/local/mysql/bin' >> /etc/profile && . /etc/profile

[ -f /etc/my.cnf ] && mv /etc/my.cnf{,.rpm}
/etc/init.d/mysqld start

/usr/local/mysql/bin/mysql_secure_installation <<EOF

N
Y
Y
Y
Y
EOF
```

计算节点

```
yum -y install mysql MySQL-python
```

openstack 基础包

```
sed -i '/priority/d' /etc/yum.repos.d/*
yum -y install http://repos.fedorapeople.org/repos/openstack/openstack-havana/rdo-release-havana-6.noarch.rpm
yum -y install http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
yum -y install openstack-utils
yum -y upgrade
reboot
```

消息服务

```
yum -y install qpid-cpp-server memcached
```

```
sed -i --posix 's/\(^auth=\)\(.*\)/\1no/g' /etc/qpidd.conf
service qpidd start
chkconfig qpidd on
```

配置Identity 服务

Identity 服务概述

Identity 服务具有以下功能：

用户管理。跟踪用户及权限。

服务目录。提供一个可用的服务API的目录

要理解 Identity 服务必须了解以下概念：

用户（User）

用数字表示的使用OpenStack云服务的人、系统或服务。Identity 服务验证用户声明的传入的请求。用户可以登录并分配令牌来访问资源。用户可以直接分配给特定租户，就像用户包含在该租户。

凭证（Credentials）

由用户证明他们是谁的数据。例如在Identity服务中的 用户名和密码,用户名和API Key,或Identity服务提供的token。

身份验证（Authentication）

确认用户身份。身份验证服务确认用户传入的凭证信息。

这些凭据最初是用户名和密码或用户名和API Key。在响应这些凭证时，Identity 服务为用户发出一个 token，用户将在后续的请求中使用该 token 。

令牌（Token）

用于访问资源的任意文本。每个令牌都有一定的使用范围，用来限制可以访问哪些资源。令牌随时可以吊销，它只在有限的时间范围内可用。

虽然在此版本的Identity服务中支持基于token的身份验证，它的目的是为了在未来的版本中支持更多的协议，最重要的目的是可以集成服务，而不是希望成为一个完整的Identity存储和管理的解决方案。

租户（Tenant）

用于分组或分离资源和（或）Identity 对象的容器。依赖服务运营商，一个租户可以映射到一个消费者、账户、组织或项目。

服务（Service）

OpenStack服务,如计算(Nova),对象存储(Swift),或镜像服务(Glance)。提供一个或多个endpoints ，通过它用户可以访问资源以及执行操作。

端点（Endpoint）

一个通过网络访问的地址，通常是一个URL，从那里可以访问服务。

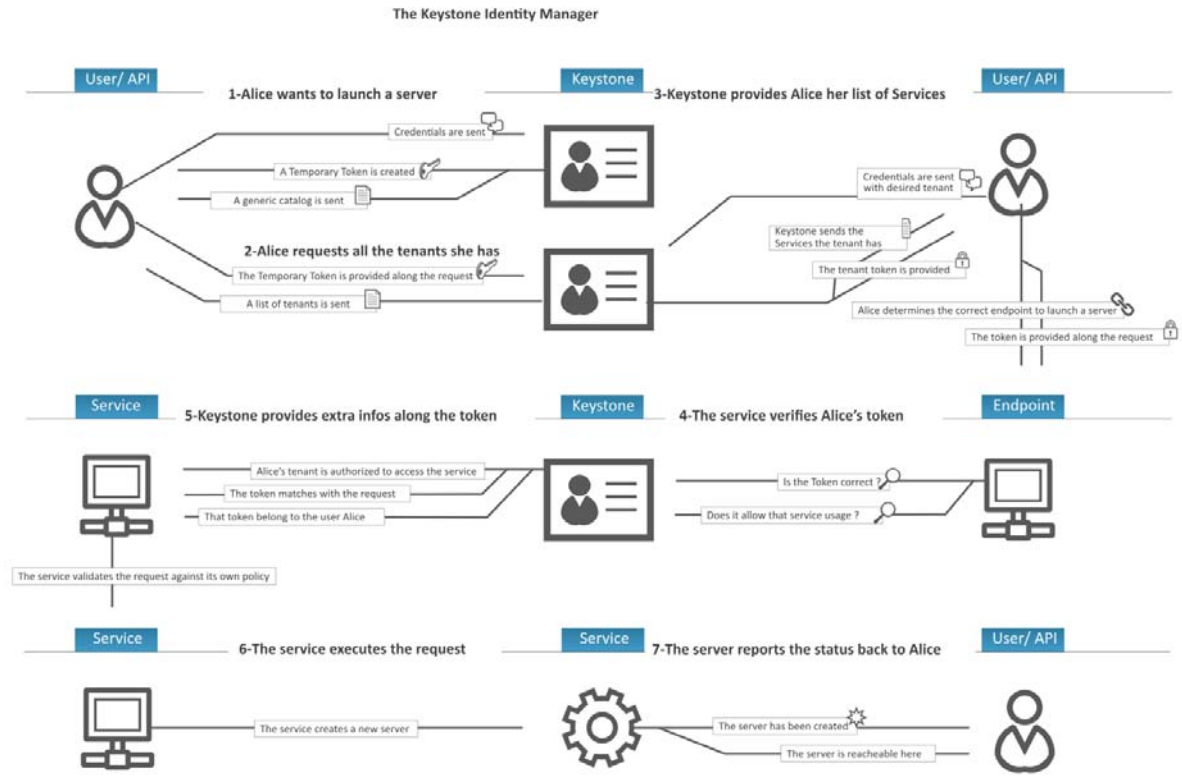
如果使用模板的扩展,您可以创建一个端点模板,这代表了模板的所有可消费服务的可用区域。

角色（Role）

一个用户承担的特性，使用户能够执行一组特定的操作。一个角色包含一组权限。用户担任该角色时继承这些权限。

在Identity服务中，发给用户的token包含一个角色的列表。由被用户调用的服务决定如何授予每个角色对每个操作或资源的访问权限。

下图展示了Identity服务的工作流程：



安装Identity服务

```
yum -y install openstack-keystone python-keystoneclient
openstack-config --set /etc/keystone/keystone.conf \
   sql connection mysql://keystone:openstack@controller/keystone

#因此次使用源码安装的mysql，所以需要取消对安装mysql-server的检测
sed -i '205,224s/^/#/g' /usr/bin/openstack-db

#为keystone 初始化（导入）数据库，并设置访问密码为openstack，使用空root密码登录数据库
openstack-db --init --service keystone --password openstack -r

ADMIN_TOKEN=$(openssl rand -hex 10)
echo $ADMIN_TOKEN
ebc3b5f98b9e115e8fee

openstack-config --set /etc/keystone/keystone.conf DEFAULT \
   admin_token $ADMIN_TOKEN

#默认情况下Keystone 使用PKI tokens,创建一个签名秘钥和证书：
keystone-manage pki_setup --keystone-user keystone --keystone-group keystone
chown -R keystone:keystone /etc/keystone/* /var/log/keystone/keystone.log

service openstack-keystone start
chkconfig openstack-keystone on
```

定义用户、租户和角色

在安装完Identity服务之后设置用户、租户和角色进行身份验证。这些都用于允许访问服务和端点，在下一节中描述。

通常你将使用用户名和密码向Identity服务进行身份验证，然而现在我们并没有创建任何用户，所以我们必须使用上一节中创建的授权令牌。

你可以使用--os-token参数或者设置环境变量OS_SERVICE_TOKEN。现在将使用OS_SERVICE_TOKEN和OS_SERVICE_ENDPOINT指向正在运行的Identity服务。

```
export OS_SERVICE_TOKEN=$ADMIN_TOKEN
export OS_SERVICE_ENDPOINT=http://controller:35357/v2.0
```

首先创建一个管理用户的租户和一个供其它服务使用的租户
```
keystone tenant-create --name=admin --description="Admin Tenant"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |           Admin Tenant           |
|   enabled   |               True               |
|      id     | e7559f20f52b4cd78504e1a929a409d4 |
|     name    |              admin               |
+-------------+----------------------------------+

keystone tenant-create --name=service --description="Service Tenant"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |          Service Tenant          |
|   enabled   |               True               |
|      id     | c09af41a86ae422bbb0181e2ec6a3b47 |
|     name    |             service              |
+-------------+----------------------------------+
```

创建管理员
```
keystone user-create --name=admin --pass=openstack --email=admin@example.com
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |        admin@example.com         |
| enabled  |               True               |
|    id    | aff212b9935d4ed088eb56daee5ffa20 |
|   name   |              admin               |
+----------+----------------------------------+
```

创建管理员角色
所有Openstack服务都可以通过在policy.json文件中指定权限，默认情况下admin角色可以访问大部分服务。
```
keystone role-create --name=admin
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|    id    | 8c774876221140cb942a98c3d6a2e001 |
|   name   |              admin               |
+----------+----------------------------------+
```

最后，你必须为用户添加角色。用户总是登录为租户，再将租户角色赋予用户。添加管理员角色使admin用户登录为租户admin。
```
keystone user-role-add --user=admin --tenant=admin --role=admin
```

定义服务和API端点
Identity 服务还跟踪Openstack服务的安装以及他们在网络上的位置。每个OpenStack服务的安装都需要运行这些命令：
    keystone service-create. 创建服务标识.
    keystone endpoint-create. 关联服务和API端点.
现在创建一个Identity服务本身的服务，以使keystone命令使用正常的身份验证，而不是使用token。

创建Identity服务的服务标识：
```
keystone service-create --name=keystone --type=identity \
  --description="Keystone Identity Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |    Keystone Identity Service     |
|      id     | cfc1e42731554202b0dd612393cf73a8 |
```

```
|     name    |             keystone             |
|     type    |             identity             |
+-------------+----------------------------------+
```

通过返回的服务ID指定Identity服务的API端点。当你指定了一个端点时，你为公共API、内部API和管理API提供了访问的url。在本指南中将使用主机名controller。Identity服务为管理API提供不同的端口。

```
keystone endpoint-create \
  --service-id=cfc1e42731554202b0dd612393cf73a8 \
  --publicurl=http://controller:5000/v2.0 \
  --internalurl=http://controller:5000/v2.0 \
  --adminurl=http://controller:35357/v2.0
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  |   http://controller:35357/v2.0   |
|      id     | 853448034b3f4927b88a6fbd2499a1e3 |
| internalurl |   http://controller:5000/v2.0    |
|  publicurl  |   http://controller:5000/v2.0    |
|    region   |            regionOne             |
|  service_id | cfc1e42731554202b0dd612393cf73a8 |
+-------------+----------------------------------+
```

当你安装其它openstack服务时，使用这些命令向Identity服务注册。

验证Identity服务安装

首先需要删除环境变量 OS_SERVICE_TOKEN 和 OS_SERVICE_ENDPOINT

```
unset OS_SERVICE_TOKEN OS_SERVICE_ENDPOINT
```

现在可以使用基于用户名和密码的身份验证

获取用户信息

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

获取用户的租户信息

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

还可以在命令行设置--os-*的变量。可以将admin的认证信息和admin的端点放到openrc.sh文件中

```
cat > openrc.sh <<EOF
export OS_USERNAME=admin
export OS_PASSWORD=openstack
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://controller:35357/v2.0
EOF

source openrc.sh

keystone token-get

keystone user-list
```

安装和配置openstack客户端

概述

你可以使用OpenStack命令行客户端运行简单的命令进行API调用。您可以从命令行或脚本自动化任务中运行这些命令。只要你提供OpenStack凭证,你可以在任何机器上运行这些命令。

在openstack内部，每个客户端都运行嵌入的curl命令发送api请求。OpenStack的api使用基于http协议的RESTful API，包括methods, URIs, media types 和 response codes.

这些开源Python客户机运行在Linux或Mac OS X系统上,很容易学习和使用。每个OpenStack服务有自己的命令行客户端。在某些客户端命令中，你可以指定debug参数来显示底层的发送API请求的命令。这是一个很好的熟悉OpenStack API调用的方法。

这些命令行客户端可用于各自的服务的api:

ceilometer (python-ceilometerclient). 度量服务的API客户端 用户创建和搜集OpenStack的度量值.

cinder (python-cinderclient). 块存储服务的API客户端 用于您创建和管理卷.

glance (python-glanceclient). 映像服务的API客户端 用于您创建和管理映像.

heat (python-heatclient). 流程控制服务的API客户端 可以从模板启动栈, 查看栈的运行细节（包括事件和资源），更新和删除堆栈。

keystone (python-keystoneclient). 身份验证服务的API客户端 用于创建和管理 users, tenants, roles, endpoints 以及 credentials.

neutron (python-neutronclient). 网络服务的API客户端 用于为客户机配置网络. 在上一版本中叫做 quantum.

nova (python-novaclient). 计算服务的API客户端机器扩展 用于创建和管理 images, instances 和 flavors.

swift (python-swiftclient). 对象存储服务的API客户端 可以搜集对象存储服务的统计信息, 列出项目清单, 更新元数据, 上传, 下载和删除文件存储.

安装OpenStack命令行客户端
所需软件

Prerequisite

Description

Python 2.6 or newer

目前客户端不支持 Python 3.

setuptools package

Mac OS X 系统默认安装

下载地址 http://pypi.python.org/pypi/setuptools.

pip package

通过包管理器安装:

Mac OS X.

```
$ sudo easy_install pip
```

Microsoft Windows. 确保 C:\Python27\Scripts 在环境变量 PATH 中声明, 然后使用 easy_install 命令安装:

```
C:\>easy_install pip
```

Ubuntu 12.04. 可以使用 dpkg 或者 aptitude 安装 python-novaclient 的二进制版本:

```
# aptitude install python-novaclient
```

Ubuntu.

```
# aptitude install python-pip
```

RHEL, CentOS, or Fedora. 可以在 RDO 源中使用yum安装客户端:

```
# yum install python-PROJECTclient
```

也可以使用pip来安装客户端:

```
# yum install python-pip
```

openSUSE 12.2 and earlier. 可以使用rpm或者zypper来安装python-novaclient:

```
# zypper install python-PROJECT
```

也可以使用pip来安装客户端:

```
# zypper install python-pip
```

openSUSE 12.3 and newer. 可以使用rpm或者zypper来安装python-novaclient:

```
# zypper install python-PROJECTclient
```

配置映像服务

映像服务概述

映像服务包含以下组件：

glance-api. 接受映像发现、检索、存储的API调用.

glance-registry. 存储、处理和检索映像的元数据. 元数据包含项目的大小和类型.

Database. 存储映像元数据. 可以根据你的喜好选择数据库. 大多数使用 MySQL 或者 SQlite.

Storage repository for image files. 在图1.2 “Logical architecture”中, 对象存储服务是映像村粗库. 然而你可以配置一个不同的存储库. 映像服务支持普通的文件系统, RADOS 块设备, Amazon S3, 和 HTTP. 其中某些只提供只读。

许多周期性的映像服务处理过程支持高速缓存。集群通过复制服务保障一致性和可用性。其它周期性的处理包括auditors、updates 和 reapers。

在概念架构一节的图中，映像服务在总体IAAS的中央。它接受来自终端用户或计算组件的映像或元数据的API请求，并且可以将磁盘文件存储到对象服务中。

安装映像服务

OpenStack的映像服务提供虚拟磁盘映像的注册服务。用户可以添加新的映像，或者从现有服务器存储映像的快照。使用快照进行备份并作为模板来启动新的服务器。你可以注册存储在对象存储或其它地方的映像。例如：你可以在普通的文件系统或外部WEB服务器来存储映像。

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

配置数据库的位置。glance-api和glance-registry有独立的配置文件

```
openstack-config --set /etc/glance/glance-api.conf \
   DEFAULT sql_connection mysql://glance:openstack@controller/glance
openstack-config --set /etc/glance/glance-registry.conf \
   DEFAULT sql_connection mysql://glance:openstack@controller/glance

openstack-db --init --service glance --password openstack -r

keystone user-create --name=glance --pass=glance \
   --email=glance@example.com
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |        glance@example.com        |
| enabled  |               True               |
|    id    | 5c281905b5db41c2b2d073e2f7ef7d30 |
|   name   |              glance              |
```

```
+----------+----------------------------------+
```
关联user(glance)，role(admin)，tenant(service)
```
keystone user-role-add --user=glance --tenant=service --role=admin
```

将认证信息添加到映像服务的配置文件
```
openstack-config --set /etc/glance/glance-api.conf keystone_authtoken \
   auth_host controller
openstack-config --set /etc/glance/glance-api.conf keystone_authtoken \
   admin_user glance
openstack-config --set /etc/glance/glance-api.conf keystone_authtoken \
   admin_tenant_name service
openstack-config --set /etc/glance/glance-api.conf keystone_authtoken \
   admin_password glance
openstack-config --set /etc/glance/glance-registry.conf keystone_authtoken auth_host
controller
openstack-config --set /etc/glance/glance-registry.conf \
   keystone_authtoken admin_user glance
openstack-config --set /etc/glance/glance-registry.conf \
   keystone_authtoken admin_tenant_name service
openstack-config --set /etc/glance/glance-registry.conf \
   keystone_authtoken admin_password glance
```

添加认证信息到 /etc/glance/glance-api-paste.ini 和 /etc/glance/glance-registry-paste.ini 文件。
在centos中安装包并不能正确创建配置文件，需要手动copy
```
cp /usr/share/glance/glance-api-dist-paste.ini /etc/glance/glance-api-paste.ini
cp /usr/share/glance/glance-registry-dist-paste.ini /etc/glance/glance-registry-
paste.ini
```

并在 [filter:authtoken] 后添加以下内容
```
cat >> /etc/glance/glance-registry-paste.ini <<EOF
auth_host=controller
admin_user=glance
admin_tenant_name=service
admin_password=glance
flavor=keystone
EOF

cat >> /etc/glance/glance-api-paste.ini <<EOF
auth_host=controller
admin_user=glance
admin_tenant_name=service
admin_password=glance
flavor=keystone
EOF
```

注册映像服务到Identity服务中，以使其它OpenStack服务可以找到它。注册服务和创建端点：
```
keystone service-create --name=glance --type=image \
  --description="Glance Image Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |       Glance Image Service       |
|      id     | 441371d1190e443aaeb82aedd6bb3d53 |
|     name    |              glance              |
|     type    |              image               |
+-------------+----------------------------------+

keystone endpoint-create \
  --service-id=441371d1190e443aaeb82aedd6bb3d53 \
  --publicurl=http://controller:9292 \
  --internalurl=http://controller:9292 \
  --adminurl=http://controller:9292
```

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  |      http://controller:9292      |
|      id     | 625d96e5e12d4a9fb14f80cd70c61106 |
| internalurl |      http://controller:9292      |
|  publicurl  |      http://controller:9292      |
|    region   |            regionOne             |
|  service_id | 441371d1190e443aaeb82aedd6bb3d53 |
+-------------+----------------------------------+
```

启动服务

```
service openstack-glance-api start
service openstack-glance-registry start
chkconfig openstack-glance-api on
chkconfig openstack-glance-registry on
```

验证映像服务安装

验证映像服务的安装你需要下载一个可以在OpenStack运行的虚拟机映像。例如：CirrOS是一个小的测试图像通常是用于测试OpenStack部署(CirrOS下载)。这里使用64位CirrOS QCOW2映像。

```
mkdir images
cd images/
wget http://cdn.download.cirros-cloud.net/0.3.1/cirros-0.3.1-x86_64-disk.img

glance image-create --name="CirrOS 0.3.1" --disk-format=qcow2 \
  --container-format=bare --is-public=true < cirros-0.3.1-x86_64-disk.img
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | d972013792949d0d3ba628fbe8685bce     |
| container_format | bare                                 |
| created_at       | 2013-12-30T02:25:03                  |
| deleted          | False                                |
| deleted_at       | None                                 |
| disk_format      | qcow2                                |
| id               | 99529f7f-0760-4b4c-8f65-14e2f7348e4f |
| is_public        | True                                 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | CirrOS 0.3.1                         |
| owner            | None                                 |
| protected        | False                                |
| size             | 13147648                             |
| status           | active                               |
| updated_at       | 2013-12-30T02:25:04                  |
+------------------+--------------------------------------+

glance image-list
+--------------------------------------+--------------+-------------+-----------------
--+----------+--------+
| ID                                   | Name         | Disk Format | Container
Format | Size     | Status |
+--------------------------------------+--------------+-------------+-----------------
--+----------+--------+
| 99529f7f-0760-4b4c-8f65-14e2f7348e4f | CirrOS 0.3.1 | qcow2       | bare
| 13147648 | active |
+--------------------------------------+--------------+-------------+-----------------
--+----------+--------+
```

配置计算服务

计算服务概述

计算服务是云计算的结构控制器，这是IaaS的重要组成部分。可以用它来承载和管理云计算系统。主模

块使用python实现。
计算服务与认证服务交互进行身份验证，映像服务提供映像，仪表盘提供用户和管理界面。通过项目和用户限制对映像的访问；每个项目使用有限的配额（例如实例的数量）。
计算服务可以在标准硬件上进行横向扩展，根据需要下载映像启动实例。
计算服务由以下功能区域及其组件组成：
API

nova-api service. 接受并响应终端用户对计算API的调用. 支持OpenStack Compute API, Amazon EC2 API, 和一个为特殊用户执行管理操作的Admin API . 并且启动一些组织操作, 包括运行一个实例并应用一些策略

nova-api-metadata service. 接受来自实例的元数据请求. nova-api-metadata 服务通常只在使用 nova-network 安装的多主机模式下使用。详细信息参考管理员手册中的Metadata service

在Debian系统中,它包含在nova-api 包中, 并且可以通过debconf选择.

Compute core

nova-compute process. 一个可以通过 hypervisor APIs 接口创建和终止虚拟机的守护进程.例如, XenServer/XCP的XenAPI ,KVM 或QEMU的libvirt, VMware的VMwareAPI 等等. 它的处理是相当复杂的，但基本原理很简单:接受队列,并执行一系列的操作系统命令,像运行一个KVM实例,同时更新数据库中的状态。

nova-scheduler process. 在计算服务中最简单的代码. 从队列中取出一个虚拟机实例的请求，并选定运行实例的计算节点，然后启动实例.

nova-conductor module. 协调nova-compute 和 database的交互.旨在避免nova-compute直接访问数据库.nova-conductor模块可以横向扩展.但是不要在任何nova-compute节点部署它. 更多信息查看A new Nova service: nova-conductor.
Networking for VMs

nova-network worker daemon. 类似nova-compute, 它从队列中接受网络配置任务并执行操作,比如设置桥接端口或者改变iptables规则。 这个功能被迁移到OpenStack网络，作为一个独立的服务.

nova-dhcpbridge script. 使用dnsmasq dhcp-script等工具跟踪IP租约并更新数据库记录. 此功能被迁移到OpenStack网络. OpenStack网络服务提供了一个不同的脚本.

Console interface

nova-consoleauth daemon. 提供控制台代理的用户授权token. 查看nova-novncproxy 和nova-xvpnvcproxy.此服务必须运行控制台代理才能工作. 这两种代理可以在集群中运行单一的nova-consoleauth服务. 详细信息参考 nova-consoleauth.

nova-novncproxy daemon. 通过VNC访问实例的代理 . 支持基于浏览器的novnc客户端.

nova-console daemon. 在Grizzly版中弃用. 用nova-xvpnvncproxy 替代.

nova-xvpnvncproxy daemon. 一个通过vnc访问实例的代理. 支持为OpenStack设计的java客户端.

nova-cert daemon. 管理x509证书.

Image management (EC2 scenario)

nova-objectstore daemon. 提供了一个S3接口用于向映像服务注册映像. 主要用于必须支持euca2ools的安装. euca2ools 工具按照S3的语言与nova-objectstore交互, nova-objectstore转换S3请求为映像服务请求.

euca2ools client. 一组用于管理云资源的命令行解释器. 虽然不是OpenStack模块, 但你可以配置nova-api支持EC2接口. 更多信息请查看 Eucalyptus 2.0 Documentation .

Command-line clients and other interfaces

nova client. 允许用户作为租户、管理员或终端用户提交命令.

nova-manage client. 使用云管理员提交命令.

Other components

The queue. 在其它守护进程之间传递消息的消息中心. 通常使用RabbitMQ, 但可能是任何AMPQ消息队列,例如Apache Qpid 或Zero MQ.

SQL database. 存储大多数云基础设施的构建和运行时状态. 包括可用的实例类型, 使用中的实例, 有效的网络和项目. 理论上, OpenStack计算服务使用的SQL-Alchemy 支持任何数据库, 但是广泛使用的数据库只有 sqlite3(只适合测试和开发工作), MySQL, 以及PostgreSQL.

计算服务和OpenStack的其它服务进行交互: 身份认证服务提供认证, 映像服务提供映像, 以及 OpenStack dashboard 提供一个web界面.

安装计算控制器服务

计算控制器是一个服务集合，可以使你能够启动一个虚拟机实例。这些服务可以运行在相同或不同的节点。在本指南中，大多数服务在控制节点运行，启动虚拟机所需的服务运行在专用的计算节点。本节将向您展示如何在控制器节点上安装和配置这些服务。

```
yum -y install openstack-nova python-novaclient

openstack-config --set /etc/nova/nova.conf \
  database connection mysql://nova:openstack@controller/nova

openstack-config --set /etc/nova/nova.conf \
  DEFAULT rpc_backend nova.openstack.common.rpc.impl_qpid

openstack-config --set /etc/nova/nova.conf DEFAULT qpid_hostname controller

openstack-db --init --service nova --password openstack -r

IP=`ifconfig eth0|sed -n 's/\(.*:\)\(\([0-9]\{1,3\}\.\)\{3\}[0-9]\{1,3\}\)\
(.*Bcast:.*\)/\2/g;/\([0-9]\{1,3\}\.\)\{3\}/p'`
openstack-config --set /etc/nova/nova.conf DEFAULT my_ip $IP
openstack-config --set /etc/nova/nova.conf DEFAULT vncserver_listen $IP
openstack-config --set /etc/nova/nova.conf DEFAULT vncserver_proxyclient_address $IP

keystone user-create --name=nova --pass=nova --email=nova@example.com
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |         nova@example.com         |
| enabled  |               True               |
|    id    | 9e84ca627edf42c894730f2f9ae8d255 |
|   name   |               nova               |
+----------+----------------------------------+

keystone user-role-add --user=nova --tenant=service --role=admin

openstack-config --set /etc/nova/nova.conf DEFAULT auth_strategy keystone
openstack-config --set /etc/nova/nova.conf keystone_authtoken auth_host controller
openstack-config --set /etc/nova/nova.conf keystone_authtoken auth_protocol http
openstack-config --set /etc/nova/nova.conf keystone_authtoken auth_port 35357
openstack-config --set /etc/nova/nova.conf keystone_authtoken admin_user nova
openstack-config --set /etc/nova/nova.conf keystone_authtoken admin_tenant_name
service
openstack-config --set /etc/nova/nova.conf keystone_authtoken admin_password nova

编辑/etc/nova/api-paste.ini，在 [filter:authtoken]段paste.filter_factory =
keystoneclient.middleware.auth_token:filter_factory后添加:
cat >> /etc/nova/api-paste.ini <<EOF
auth_host = controller
```

```
auth_port = 35357
auth_protocol = http
auth_uri = http://controller:5000/v2.0
admin_tenant_name = service
admin_user = nova
admin_password = nova
EOF
```

确保/etc/nova/nova.conf文件中包含 api_paste_config=/etc/nova/api-paste.ini。

```
sed -i '/^#api_paste_config=/aapi_paste_config=/etc/nova/api-paste.ini'
/etc/nova/nova.conf

keystone service-create --name=nova --type=compute \
  --description="Nova Compute service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |       Nova Compute service       |
|      id     | d1c9655126c5426a85458eab8cb4364b |
|     name    |               nova               |
|     type    |             compute              |
+-------------+----------------------------------+

keystone endpoint-create \
  --service-id=d1c9655126c5426a85458eab8cb4364b \
  --publicurl=http://controller:8774/v2/%\(tenant_id\)s \
  --internalurl=http://controller:8774/v2/%\(tenant_id\)s \
  --adminurl=http://controller:8774/v2/%\(tenant_id\)s
+-------------+-----------------------------------------+
|   Property  |                  Value                  |
+-------------+-----------------------------------------+
|   adminurl  | http://controller:8774/v2/%(tenant_id)s |
|      id     |     3374f979ff3f420d97af5fe30d6db679    |
| internalurl | http://controller:8774/v2/%(tenant_id)s |
|  publicurl  | http://controller:8774/v2/%(tenant_id)s |
|    region   |                regionOne                |
|  service_id |     d1c9655126c5426a85458eab8cb4364b    |
+-------------+-----------------------------------------+

service openstack-nova-api start
service openstack-nova-cert start
service openstack-nova-consoleauth start
service openstack-nova-scheduler start
service openstack-nova-conductor start
service openstack-nova-novncproxy start
chkconfig openstack-nova-api on
chkconfig openstack-nova-cert on
chkconfig openstack-nova-consoleauth on
chkconfig openstack-nova-scheduler on
chkconfig openstack-nova-conductor on
chkconfig openstack-nova-novncproxy on

验证安装
nova image-list
+--------------------------------------+-------------+--------+--------+
| ID                                   | Name        | Status | Server |
+--------------------------------------+-------------+--------+--------+
| 99529f7f-0760-4b4c-8f65-14e2f7348e4f | CirrOS 0.3.1 | ACTIVE |        |
+--------------------------------------+-------------+--------+--------+
```

配置一个计算节点

在控制节点配置完计算服务后还必须配置一个计算节点。计算节点接收来自控制节点和虚拟机实例的请求。你可以在一个节点上运行所有服务，但在本手册中使用独立的系统。按照本节中的说明可以很容易的

通过增加额外的计算节点进行水平扩展。
计算服务依赖一个虚拟层来运行虚拟机实例，openstack支持不同的虚拟层，本手册使用kvm。
1.配置一个系统。参考Chapter 2, Basic operating system configuration。但请注意与控制节点的差异。
eth0配置一个不同的ip，本手册使用192.18.0.71.eth1不要配置静态地址. 使用OpenStack分配和配置IP地址的网络组件

主机名配置为compute1.使用uname -n验证配置. 确保每个主机的ip和主机名在/etc/hosts中生效

从控制节点同步. 按照“Network Time Protocol (NTP)”进行操作.

安装mysql客户端. 不需要安装mysql服务器及启动mysql服务.

启用openstack的发行版本安装包. 参考 “OpenStack packages”.
2.配置好系统后，安装计算服务需要的包
```
yum -y install openstack-nova-compute
```

3.修改nova.conf配置
```
openstack-config --set /etc/nova/nova.conf database connection
mysql://nova:NOVA_DBPASS@controller/nova
openstack-config --set /etc/nova/nova.conf DEFAULT auth_strategy keystone
openstack-config --set /etc/nova/nova.conf keystone_authtoken auth_host controller
openstack-config --set /etc/nova/nova.conf keystone_authtoken auth_protocol http
openstack-config --set /etc/nova/nova.conf keystone_authtoken auth_port 35357
openstack-config --set /etc/nova/nova.conf keystone_authtoken admin_user nova
openstack-config --set /etc/nova/nova.conf keystone_authtoken admin_tenant_name
service
openstack-config --set /etc/nova/nova.conf keystone_authtoken admin_password nova
```

4.配置Qpid相关选项
```
openstack-config --set /etc/nova/nova.conf \
  DEFAULT rpc_backend nova.openstack.common.rpc.impl_qpid

openstack-config --set /etc/nova/nova.conf DEFAULT qpid_hostname controller
```

5.提供远程访问实例的配置
```
IP=`ifconfig eth0|sed -n 's/\(.*:\)\(\([0-9]\{1,3\}\.\)\{3\}[0-9]\{1,3\}\)\
(.*Bcast:.*\)/\2/g;/\([0-9]\{1,3\}\.\)\{3\}/p'`
openstack-config --set /etc/nova/nova.conf DEFAULT my_ip $IP
openstack-config --set /etc/nova/nova.conf DEFAULT vnc_enabled True
openstack-config --set /etc/nova/nova.conf DEFAULT vncserver_listen 0.0.0.0
openstack-config --set /etc/nova/nova.conf DEFAULT vncserver_proxyclient_address $IP
openstack-config --set /etc/nova/nova.conf \
  DEFAULT novncproxy_base_url http://controller:6080/vnc_auto.html
```

6.指定连接的映像服务
```
openstack-config --set /etc/nova/nova.conf DEFAULT glance_host controller
```

7.修改/etc/nova/api-paste.ini中的[filter:authtoken]添加认证信息
在 [filter:authtoken]段paste.filter_factory =
keystoneclient.middleware.auth_token:filter_factory后添加:
```
cat >> /etc/nova/api-paste.ini <<EOF
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = nova
EOF

sed -i '/^#api_paste_config=/aapi_paste_config=/etc/nova/api-paste.ini'
/etc/nova/nova.conf
```

8.启动服务
```
service libvirtd start
service messagebus start
chkconfig libvirtd on
chkconfig messagebus on
service openstack-nova-compute start
chkconfig openstack-nova-compute on
```

启用网络

配置网络是一段令人困惑的经历。下面的例子配置了一个最简单的可用于生产环境的配置：在openstack计算的传统的网络中，采用扁平化网络及DHCP

这套配置采用了多主机功能，网络被配置为具有高可用性的分布式网络，因此网络控制器没有单点故障，这样需要在每个计算节点配置网络。

#如果需要配置完整的SDN网络，请查看Chapter 9, Install the Networking service.

1.安装所需的包
```
yum -y install openstack-nova-network
```

2.编辑nova.conf配置
```
openstack-config --set /etc/nova/nova.conf DEFAULT \
  network_manager nova.network.manager.FlatDHCPManager
openstack-config --set /etc/nova/nova.conf DEFAULT \
  firewall_driver nova.virt.libvirt.firewall.IptablesFirewallDriver
openstack-config --set /etc/nova/nova.conf DEFAULT network_size 254
openstack-config --set /etc/nova/nova.conf DEFAULT allow_same_net_traffic False
openstack-config --set /etc/nova/nova.conf DEFAULT multi_host True
openstack-config --set /etc/nova/nova.conf DEFAULT send_arp_for_ha True
openstack-config --set /etc/nova/nova.conf DEFAULT share_dhcp_address True
openstack-config --set /etc/nova/nova.conf DEFAULT force_dhcp_release True
openstack-config --set /etc/nova/nova.conf DEFAULT flat_interface eth1
openstack-config --set /etc/nova/nova.conf DEFAULT flat_network_bridge br100
openstack-config --set /etc/nova/nova.conf DEFAULT public_interface eth1
```

3.为计算节点上的实例提供元数据服务。只有在不运行nova-api服务的计算节点需要。
```
yum install openstack-nova-api
service openstack-nova-metadata-api start
chkconfig openstack-nova-metadata-api on
```

4.启动网络服务
```
service openstack-nova-network restart
chkconfig openstack-nova-network on
```

创建一个虚拟机可用的网络。对于整个安装仅需执行一次，不需要在每个计算节点执行。在控制节点运行nova network-create命令
```
source openrc.sh
nova network-create vmnet --fixed-range-v4=172.16.0.0/24 \
  --bridge-interface=br100 --multi-host=T
```

运行实例

计算服务配置完成后，你可以运行一个实例。OpenStack规定在计算服务器上每个实例就是一个虚拟机。下面的例子将展示如何使用下载的映像启动一个低配置的实例。

这个过程需要你具备：

运行命令需要安装nova的客户端库（如果不确定是否已安装，可以在控制节点运行）
设置指定认证信息的环境变量。参考“Verify the Identity Service installation”.
下载映像。参考 “Verify the Image Service installation”.
配置网络。参考 “Enable Networking”.

1.生成用于运行OpenStack实例所需的秘钥对。这些秘钥将添加到实例中，用于SSH无密码登录实例。这需要将必要的工具打包到映像中。更多信息查看 OpenStack Admin User Guide.
```
ssh-keygen
cd .ssh
nova keypair-add --pub_key id_rsa.pub mykey
```

你刚刚创建了mykey秘钥对，id_rsa的私钥保存在本地的~/.ssh中，你可以使用mykey发起对虚拟机实例的连接。查看有效的keypairs：

```
nova keypair-list
+-------+-------------------------------------------------+
| Name  | Fingerprint                                     |
+-------+-------------------------------------------------+
| mykey | 21:5f:5e:d9:36:2f:97:f1:20:88:c2:3d:15:3c:dd:cb |
+-------+-------------------------------------------------+
```

2.要运行一个实例，你必须指定运行实例所需的 flavor ID。 flavor 是资源配置的概要文件。例如，它指定了你的实例可以拥有对少虚拟cpu和多少内存。查看有效的概要文件：

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

3.获取用于实例的映像ID

```
nova image-list
+--------------------------------------+--------------+--------+--------+
| ID                                   | Name         | Status | Server |
+--------------------------------------+--------------+--------+--------+
| 99529f7f-0760-4b4c-8f65-14e2f7348e4f | CirrOS 0.3.1 | ACTIVE |        |
+--------------------------------------+--------------+--------+--------+
```

4.要使用ssh和ping，你需要配置安全组规则。参考OpenStack User Guide.

```
nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| tcp         | 22        | 22      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+

nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| icmp        | -1        | -1      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
```

5.运行实例

```
nova boot --flavor 1 --key_name mykey --image  99529f7f-0760-4b4c-8f65-14e2f7348e4f -
-security_group default cirrOS
+--------------------------------------+--------------------------------------+
| Property                             | Value                                |
+--------------------------------------+--------------------------------------+
| OS-EXT-STS:task_state                | scheduling                           |
| image                                | CirrOS 0.3.1                         |
| OS-EXT-STS:vm_state                  | building                             |
```

```
| OS-EXT-SRV-ATTR:instance_name        | instance-00000001                    |
| OS-SRV-USG:launched_at               | None                                 |
| flavor                               | m1.tiny                              |
| id                                   | 04bb092c-927a-4cda-9c1b-8964f6a2e468 |
| security_groups                      | [{u'name': u'default'}]              |
| user_id                              | aff212b9935d4ed088eb56daee5ffa20     |
| OS-DCF:diskConfig                    | MANUAL                               |
| accessIPv4                           |                                      |
| accessIPv6                           |                                      |
| progress                             | 0                                    |
| OS-EXT-STS:power_state               | 0                                    |
| OS-EXT-AZ:availability_zone          | nova                                 |
| config_drive                         |                                      |
| status                               | BUILD                                |
| updated                              | 2013-12-30T04:23:34Z                 |
| hostId                               |                                      |
| OS-EXT-SRV-ATTR:host                 | None                                 |
| OS-SRV-USG:terminated_at             | None                                 |
| key_name                             | mykey                                |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | None                                 |
| name                                 | cirrOS                               |
| adminPass                            | bfUN4rwGRuuv                         |
| tenant_id                            | e7559f20f52b4cd78504e1a929a409d4     |
| created                              | 2013-12-30T04:23:34Z                 |
| os-extended-volumes:volumes_attached | []                                   |
| metadata                             | {}                                   |
+--------------------------------------+--------------------------------------+
```

如果没有足够实例使用的内存，计算可以创建，但不能启动，实例的状态会被设置为ERROR。

6.启动实例后，可以使用nova list查看实例状态。它的状态将从 BUILD 变为 ACTIVE。

```
nova list
+--------------------------------------+----------+--------+------------+-----------
--+----------------+
| ID                                   | Name     | Status | Task State | Power
State | Networks       |
+--------------------------------------+----------+--------+------------+-----------
--+----------------+
| 04bb092c-927a-4cda-9c1b-8964f6a2e468 | cirrOS   | BUILD  | spawning   | NOSTATE
| vmnet=172.16.0.2 |
+--------------------------------------+----------+--------+------------+-----------
--+----------------+
$ nova list
+--------------------------------------+----------+--------+------------+-----------
--+----------------+
| ID                                   | Name     | Status | Task State | Power
State | Networks       |
+--------------------------------------+----------+--------+------------+-----------
--+----------------+
| 04bb092c-927a-4cda-9c1b-8964f6a2e468 | cirrOS   | ACTIVE | None       | Running
| vmnet=172.16.0.2 |
+--------------------------------------+----------+--------+------------+-----------
--+----------------+
```

查看实例详细信息：

```
nova show 04bb092c-927a-4cda-9c1b-8964f6a2e468
```

7.当实例已启动并且初始化以及你已经配安全组后，你可以不用使用密码，使用nova boot命令中指定的keypair登录到实例。

使用nova list可以获得实例的ip。

如果使用的是CirrOS的映像，你必须使用cirros用户登录。也可以使用密码 cubswin:) ，而不用ssh key。

```
ssh cirros@172.16.0.2
```

安装仪表盘

系统要求

在安装OpenStack仪表盘之前，你的系统必须满足以下需求：

已安装OpenStack计算，启用Identity服务管理用户和项目

注意身份服务和计算端点的url。

运行Identity服务的用户有sudo权限。因为apache服务不适用root用户启动，用户必须以拥有sudo权限的运行Identity服务的用户启动仪表盘

Python 2.6或2.7，必须支持Django。

OpenStack仪表盘可以安装在任何一个可以连接Identity服务的节点。

为用户提供以下信息，以方便用户通过本地浏览器访问仪表盘：

1.一个可以访问仪表盘的IP地址

2.一个可以登录仪表盘的账号密码

你的浏览器必须支持HTML5、cookie、JavaScript

如果在仪表盘使用VNC客户端，你的浏览器必须支持HTML5 Canvas以及HTML5 WebSockets.

查看关于noVNC及浏览器的详细信息，参考

https://github.com/kanaka/noVNC/blob/master/README.md及

https://github.com/kanaka/noVNC/wiki/Browser-support

安装仪表盘

如果你只安装了对象存储和身份验证服务，即使你安装了仪表盘它也是不可用的。

关于仪表盘部署的详细信息，请参考deployment topics in the developer documentation.

1.安装仪表盘及依赖服务

```
yum -y install memcached python-memcached mod_wsgi openstack-dashboard
```

2.修改/etc/openstack-dashboard/local_settings中的CACHES['default']，增加memcache地址

```
sed --posix -i "/\(^ *\)'BACKEND'/s/\(^ *'BACKEND' : '\)\(.*\)\
('\)/\1django.core.cache.backends.memcached.MemcachedCache\3,/g;/\(^
*\)'BACKEND'/a\\\t'LOCATION' : '127.0.0.1:11211'" \
/etc/openstack-dashboard/local_settings
```

改变配置后必须重启apache

可以使用其它服务器存储session，通常使用SESSION_ENGINE选择配置session的后端

可以修改/etc/openstack-dashboard/local_settings中的TIME_ZONE改变时区

3.修改ALLOWED_HOSTS，添加你访问仪表盘使用的地址

```
sed -i --posix "s/\(ALLOWED_HOSTS.*[\)\('localhost'\)\(\]\)/\1\2, '192.168.0.70'/g"
/etc/openstack-dashboard/local_settings
```

4.本指南假定仪表盘安装在controller节点，你可以很容易的部署在其它节点上，修改local_settings.py为合适的配置。

修改OPENSTACK_HOST为Identity服务节点：

```
sed -i --posix 's/\(^OPENSTACK_HOST.*"\)\(.*\)\("\)/\1controller\3/g' /etc/openstack-
dashboard/local_settings
```

5.启动服务

```
service httpd start
service memcached start
chkconfig httpd on
chkconfig memcached on
```

6.浏览器打开 http://192.168.0.70/dashboard

为仪表盘建立session存储

本地存储

本地内存存储是最快最简单的方式，因为它没有任何外部依赖，但是它有以下缺陷：

不能在多进程间共享

进程终止即丢失

本地内存存储为默认选项，因为它没有任何依赖。不建议用于生产或重要的开发工作。开启方式：

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
CACHES = {
          'BACKEND': 'django.core.cache.backends.locmem.LocMemCache'
}
```

Key-Values 存储
Memcached
Redis
你可以使用Memcached或Redis做外部缓存。这些程序可以提供持久性和共享存储，对小规模部署或开发非常有用。

Memcached
是一个高性能的分布式内存对象缓存系统，提供对任意类型的小型数据块的内存缓存。
依赖于：

- Memcached服务
- 安装python-memcached

启用方式

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

Redis
是一个基于BSD协议开源的高级key-value存储，通常被称为结构化数据服务器。
依赖于：

- Redis服务
- 安装python的redis模块以及 django-redis

启用方式：

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
CACHES = {
     "default": {
          "BACKEND": "redis_cache.cache.RedisCache",
          "LOCATION": "127.0.0.1:6379:1",
           "OPTIONS": {
               "CLIENT_CLASS": "redis_cache.client.DefaultClient",
          }
     }
}
```

数据库初始化及配置
数据库存储session提供持久化、可扩展、高性能及高可用。然而数据库存储session在高访问量的情况下是比较慢的，数据库的优化超出了本文档的范畴。

1.命令行登录Mysql

```
mysql -uroot -p
```

2.输入密码

3.创建数据库

```
CREATE DATABASE dash;
```

4.配置权限

```
GRANT ALL ON dash.* TO 'dash'@'%' IDENTIFIED BY 'dash';
GRANT ALL ON dash.* TO 'dash'@'localhost' IDENTIFIED BY 'dash';
```

5.退出mysql

6.修改local_settings

```
SESSION_ENGINE = 'django.core.cache.backends.db.DatabaseCache'
DATABASES = {
'default': {
# Database configuration here
'ENGINE': 'django.db.backends.mysql',
'NAME': 'dash',
'USER': 'dash',
'PASSWORD': 'dash',
'HOST': 'localhost',
'default-character-set': 'utf8'
}
}
```

7.同步数据库

```
/usr/share/openstack-dashboard/manage.py syncdb

You just installed Django's auth system, which means you don't have any superusers
defined.
Would you like to create one now? (yes/no): yes
Username (leave blank to use 'root'):
E-mail address: root@example.com
Password:
Password (again):
Superuser created successfully.
Installing custom SQL ...
Installing indexes ...
Installed 0 object(s) from 0 fixture(s)
```

8.在ubuntu中，如果想在重启apache时避免警告信息，可以创建一个黑洞目录：

```
sudo mkdir -p /var/lib/dash/.blackhole
```

9.重启apache

```
service httpd restart
chkconfig httpd on
```

10.在ubuntu中，重启nova-api服务以确保仪表盘连接api服务时没有错误

```
sudo restart nova-api
```

缓存数据库

为了减轻数据库查询时的性能问题，你可以使用Django cached_db session后端，它可以同时使用数据库和缓存系统，以执行高性能写入和高效检索

通过以下配置同时启用数据库和缓存：

```
SESSION_ENGINE = "django.contrib.sessions.backends.cached_db"
```

Cookies

如果你使用Django1.4或更高版本，则signed_cookies后端避免了服务器的负载和扩展问题。

这个后端会将session数据存到cookie中，它存储在用户的浏览器。后端使用加密签名技术以确保数据传输不被篡改。这与加密技术不同，session数据对攻击者仍然可读。

这个引擎的优点是不需要任何依赖及基础设施的开销。只要放到一个正常的cookie中，它可以无限期存储。

最大的缺陷是它存储在用户机器上，并且使用网络传输，它也限制了最大可存储的session数据数量。

请查看Django cookie-based sessions。

创建块存储服务

块存储服务

它支持管理卷、卷快照及卷类型。它包含以下组件：

cinder-api. 接受api请求并将其路由到cinder-volume进程操作

cinder-volume. 响应读取和写入块存储数据库的请求，保持状态，通过消息队列与其它进程（如cinder-scheduler）交互。它可以通过驱动整合多种存储。

cinder-scheduler 守护进程. 与nova-scheduler类似，选择最优的块存储提供者节点创建卷。

消息队列. 块存储服务间的消息处理

块存储服务于计算服务交互，为实例提供卷。

配置块存储服务
本节描述如何在控制节点配置块存储服务，并假设第二个节点通过cinder-volume提供存储。关于如何配置第二个节点，请查看“Configure a Block Storage Service node”.

你可以配置OpenStack使用不同的存储系统，此处使用LVM.
1.安装块存储服务包
```
yum -y install openstack-cinder openstack-utils openstack-selinux
```

2.修改数据库配置
```
openstack-config --set /etc/cinder/cinder.conf \
  database connection mysql://cinder:cinder@controller/cinder
```

3.初始化数据库
```
openstack-db --init --service cinder --password openstack -r
```

4.创建用户
```
keystone user-create --name=cinder --pass=cinder --email=cinder@example.com
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |        cinder@example.com        |
| enabled  |               True               |
|    id    | 1c43be18f95748f4bf68bab931648a5f |
|   name   |              cinder              |
+----------+----------------------------------+

keystone user-role-add --user=cinder --tenant=service --role=admin
```

5.添加认证信息到/etc/cinder/api-paste.ini的[filter:authtoken]部分。
```
cat >> /etc/cinder/api-paste.ini <<EOF
auth_host=controller
auth_port = 35357
auth_protocol = http
admin_tenant_name=service
admin_user=cinder
admin_password=cinder
EOF
```

6.修改消息中间件配置
```
openstack-config --set /etc/cinder/cinder.conf \
  DEFAULT rpc_backend cinder.openstack.common.rpc.impl_qpid
openstack-config --set /etc/cinder/cinder.conf \
  DEFAULT qpid_hostname controller
```

7.注册服务
```
keystone service-create --name=cinder --type=volume \
  --description="Cinder Volume Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |      Cinder Volume Service       |
|      id     | b8c508cfa1f847afbc6887fd1484fe4a |
|     name    |              cinder              |
|     type    |              volume              |
+-------------+----------------------------------+

keystone endpoint-create \
  --service-id=b8c508cfa1f847afbc6887fd1484fe4a \
  --publicurl=http://controller:8776/v1/%\(tenant_id\)s \
  --internalurl=http://controller:8776/v1/%\(tenant_id\)s \
  --adminurl=http://controller:8776/v1/%\(tenant_id\)s
```

```
+-------------+-----------------------------------------+
|   Property  |                  Value                  |
+-------------+-----------------------------------------+
|   adminurl  | http://controller:8776/v1/%(tenant_id)s |
|      id     |     3dc8bd3fd615485dacd902183bebd8a9    |
| internalurl | http://controller:8776/v1/%(tenant_id)s |
|  publicurl  | http://controller:8776/v1/%(tenant_id)s |
|    region   |                regionOne                |
|  service_id |     b8c508cfa1f847afbc6887fd1484fe4a    |
+-------------+-----------------------------------------+
```

8.注册一个v2版本的API

```
keystone service-create --name=cinderv2 --type=volumev2 \
  --description="Cinder Volume Service V2"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |     Cinder Volume Service V2     |
|      id     | e9421b980ed24c88b7ade5aef5f184d7 |
|     name    |              cinderv2            |
|     type    |              volumev2            |
+-------------+----------------------------------+
keystone endpoint-create \
  --service-id=e9421b980ed24c88b7ade5aef5f184d7 \
  --publicurl=http://controller:8776/v2/%\(tenant_id\)s \
  --internalurl=http://controller:8776/v2/%\(tenant_id\)s \
  --adminurl=http://controller:8776/v2/%\(tenant_id\)s
+-------------+-----------------------------------------+
|   Property  |                  Value                  |
+-------------+-----------------------------------------+
|   adminurl  | http://controller:8776/v2/%(tenant_id)s |
|      id     |     6d052b506b7042b48acdce4946ed3831    |
| internalurl | http://controller:8776/v2/%(tenant_id)s |
|  publicurl  | http://controller:8776/v2/%(tenant_id)s |
|    region   |                regionOne                |
|  service_id |     e9421b980ed24c88b7ade5aef5f184d7    |
+-------------+-----------------------------------------+
```

9.启动服务

```
service openstack-cinder-api start
service openstack-cinder-scheduler start
chkconfig openstack-cinder-api on
chkconfig openstack-cinder-scheduler on
```

配置块存储服务节点

在控制节点安装好服务后，将第二个系统配置为块存储服务节点。该节点有一个用作卷存储的磁盘。可以使用不同的存储，本例展示如何配LVM.

1.参考Chapter 2, Basic operating system configuration配置一个系统。请注意以下与控制节点的区别：

设置主机名为block1.确保在每个节点配置了hosts

参考“Network Time Protocol (NTP)”从控制节点同步

2.创建一个LVM逻辑卷，本手册假定/dev/sdb用于LVM.

```
pvcreate /dev/sdb
vgcreate cinder-volumes /dev/sdb
```

3.在/etc/lvm/lvm.conf 中的devices部分添加条目，使lvm扫描虚拟机使用的设备。

你必须在块存储主机添加所需的LVM物理卷，运行pvdisplay 命令获取所需的卷

过滤规则中的每一项以a(accept)或r(reject)开始。所有块存储服务器上的物理卷以a开始，必须以"r/.*/"结束filter，以拒绝所有未列出的设备。

```
sed -i '/^ *filter/s/ ]$/, "a\/sdb\/", "r\/.*\/"&/g' /etc/lvm/lvm.conf
```

4.配置操作系统后安装openstack包

```
yum -y install openstack-cinder openstack-utils openstack-selinux --skip-broken

5.修改/etc/cinder/api-paste.ini，增加认证信息
cat >> /etc/cinder/api-paste.ini <<EOF
auth_host=controller
auth_port = 35357
auth_protocol = http
admin_tenant_name=service
admin_user=cinder
admin_password=cinder
EOF

6.配置Qpid支持
openstack-config --set /etc/cinder/cinder.conf \
  DEFAULT rpc_backend cinder.openstack.common.rpc.impl_qpid
openstack-config --set /etc/cinder/cinder.conf \
  DEFAULT qpid_hostname controller

7.配置mysql支持
openstack-config --set /etc/cinder/cinder.conf \
  database connection mysql://cinder:openstack@controller/cinder

8.配置iscsi target服务，添加cinder valumes 发现。
echo 'include /etc/cinder/volumes/*' >> /etc/tgt/targets.conf

9.启动服务
service openstack-cinder-volume start
service tgtd start
chkconfig openstack-cinder-volume on
chkconfig tgtd on
```

添加对象存储

对象存储服务

对象存储服务是通过RESTful HTTP API提供的高度可扩展和持久的低成本多租户的对象存储系统，用于大批量非结构化数据的存储。
它包含以下组件：
代理服务(swift-proxy-server)：接受对象存储API和原始HTTP请求以上传文件,修改元数据,并创建容器。它还提供web浏览文件及容器列表。
账户服务(swift-account-server)：管理对象存储服务的账户定义。
容器服务(swift-container-server)：管理容器或文件夹在对象存储服务中的映射。
对象服务(swift-object-server)：在存储节点管理实际对象，如文件。
大量的周期性处理。执行大型数据存储的日常任务。集群通过复制服务保障一致性和可用性。其它周期性过程包括auditors、 updaters和reapers.
配置WSGI中间件来处理身份验证，通常是Identity服务。

系统需求

硬件：OpenStack对象存储运行在商用硬件上。
当你只安装对象存储服务和Identity服务时,您不能使用仪表板,除非你也安装计算和映像服务。
表8.1　硬件建议

Server	推荐配置	Notes
对象存储服务器	处理器: 双四核　内存: 8 or 12 GB RAM 硬盘空间: 优化每GB成本　网络: 1个千兆网卡 (NIC)	磁盘空间的容量取决于你的容量需求. 你需要根据硬盘平均故障率优化每GB成本. 在Rackspace, 我们的存储服务器运行普通4U服务器 24块2T SATA硬盘 8核CPU. RAID不是必须的，也不推荐使用. 在使用RAID的情况下Swift磁盘模式可能是最差的, 使用 RAID 5 或 6 性能下降很快. 例如, Rackspace 运行云文件存储的服务器有24块 2T SATA 硬盘以及8核心CPU. 已经支持大多数单进程或并发服务. 这样可以有效利用多核.
对象存储容器/账	处理器: 双四核　内存: 8 or 12 GB RAM 网络: 1个千兆网卡	优化SQLite 数据库的IOPS.

户服务器	(NIC)	
对象存储代理服务器	处理器：双四核　网络：一块千兆网卡 (NIC)	较高的网络吞吐量为支持多种API请求提供了更好的性能。 优化代理服务器的CPU性能.代理服务是多CPU和网络I/O密集型服务. 如果使用10G的网络代理或者SSL代理，必须使用更高的CPU

操作系统： OpenStack对象存储可以运行在Ubuntu, RHEL, CentOS, Fedora, openSUSE 或 SLES
网络：内部建议使用1G或10G网络。对于OpenStack对象存储，外界连接到代理服务器和存储网络应使用隔离私有网络的公共网络连接。
数据库：在OpenStack对象存储中，SQLite数据库是OpenStack对象存储容器和账号管理程序的一部分。
权限：你可以使用root用户或者具有sudo权限的的用户安装OpenStack对象存储。

对象存储网络的设计
为节省网络资源，并确保网络管理员了解用于提供访问的API必要的存储网络及公网IP地址需求，本节提供建议和所需的最小配置。建议至少使用千兆网络。
本指南描述了以下网络：
一个必须的公共网络。连接到代理服务器。
一个必须的存储网络。集群之外无法访问，所有节点连接到这个网络。
一个可选的复制网络。集群之外无法访问，用于存储节点的复制流量。必须配置为环形网络。
默认情况下所有OpenStack的对象存储服务以及存储节点上的rsync守护进程都监听STORAGE_LOCAL_NET的ip地址
如果你配置了一个环形复制网络，账户,容器和对象服务器监听STORAGE_LOCAL_NET和STORAGE_REPLICATION_NET IP地址。rsync守护进程只监听STORAGE_REPLICATION_NET IP地址。

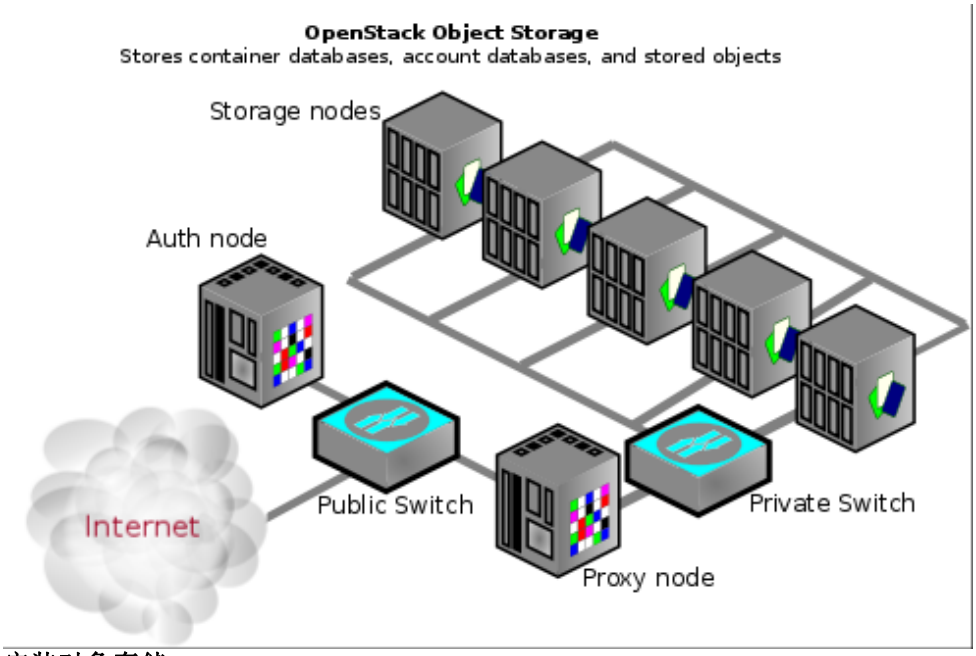
公共网络（公开可路由的IP范围）
　　提供公网IP访问到云基础设施中的API端点。
　　最低配置：每个代理服务器一个IP地址。

存储网络(RFC1918 IP范围,非公网地址)
　　管理对象存储基础架构中的所有服务器间的通信。
　　最低配置：每个存储节点和代理服务器一个IP地址。
　　推荐配置：同上，将网络位扩展到最大范围。如255或CIDR /24

复制网络(RFC1918 IP范围,非公网地址)
　　管理对象存储基础架构中的存储服务器之间复制相关的通信。
　　推荐配置：配置STORAGE_LOCAL_NET即可

对象存储架构示例
　　node： 运行一个或多个对象存储服务的主机
　　代理节点：运行代理服务
　　存储节点：运行账户、容器和对象服务
　　环：　　一组的OpenStack对象存储数据的物理设备之间的映射。
　　复制： 一个对象的副本。默认情况下，集群中存在三个副本。
　　Zone： 集群中逻辑独立的部分，有关独立的故障特性。

为增加可靠性和提高性能，你可以增加代理服务器。
本文描述了每个存储节点作为集群中的一个独立的区域，建议至少5个区域。区域是尽可能独立于其它节点的一组节点（不同的服务器、网络、电源甚至是地理位置）。环确保每个副本保存在一个独立的区域，下图显示了一个最小的配置：



安装对象存储

如果你正在安装一个新的服务器，你需要有一个操作系统的安装介质。
接下来的步骤假定你已经安装了OpenStack需要的包，参见OpenStack Packages.
本文描述了如何按照以下的节点安装一个集群：

一个代理节点运行swift-proxy-server进程。代理服务器代理请求到适当的存储节点。

五个存储节点运行swift-account-server、swift-container-server和swift-object-server进程，控制账户数据库的存储,容器数据库,以及实际的存储对象。

最初可以使用更少的存储节点,但生产集群建议至少五个。

一般安装步骤

1.安装swift核心文件及ssh

```
yum -y install openstack-swift openstack-swift-proxy \
  openstack-swift-account openstack-swift-container \
  openstack-swift-object memcached
```

2.在所有节点上创建配置文件目录

```
mkdir -p /etc/swift
chown -R swift:swift /etc/swift/
```

3.在所有节点创建配置文件 /etc/swift/swift.conf

```
cat >/etc/swift/swift.conf <<EOF
[swift-hash]
# random unique strings that can never change (DO NOT LOSE)
swift_hash_path_prefix = `od -t x8 -N 8 -A n </dev/random`
swift_hash_path_suffix = `od -t x8 -N 8 -A n </dev/random`
EOF
```

suffix的值应该使用一些随机的文本字符串，在环做hash散列时将作为种子数。此文件在集群中的所有节点都是相同的。
接下来,设置存储节点和代理节点。这个示例使用共同的身份服务进行身份验证。

安装和配置存储服务

对象存储可以工作在任何文件系统上，支持扩展属性（XATTRS）。Rackspace进行了大量的测试及压力测试后显示在swift应用中，XFS总体性能是最佳的。它也是唯一经过完整测试的文件系统。其它建议参考OpenStack Configuration Reference。

1.安装存储节点所需包

```
yum -y install openstack-swift-account openstack-swift-container \
  openstack-swift-object xfsprogs
```

2.将存储节点上的每一块将用于对象存储的设备格式化为XFS卷（本例中使用/dev/sdb）。使用单独的分区或驱动器。例如在具有12块硬盘的服务器中，你可以使用一到两块硬盘安装操作系统，这些硬盘不应该在此步骤中操作。其它的10到11块硬盘应该独立分区并用XFS进行格式化。

```
fdisk /dev/sdb <<EOF
n
p
1


w
EOF
mkfs.xfs /dev/sdb1
echo "/dev/sdb1 /srv/node/sdb1 xfs noatime,nodiratime,nobarrier,logbufs=8 0 0" >>
/etc/fstab
mkdir -p /srv/node/sdb1
mount /srv/node/sdb1
chown -R swift:swift /srv/node
```

3.创建/etc/rsyncd.conf

```
cat > /etc/rsyncd.conf <<EOF
uid = swift
gid = swift
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
address = 192.168.0.88
[account]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/account.lock
[container]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/container.lock
[object]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/object.lock
EOF
```

4.如果需要使用单独的网卡进行复制，需要将address 设置为指定的网卡ip

```
address = <STORAGE_REPLICATION_NET_IP>
```

5.编辑/etc/default/rsync

```
RSYNC_ENABLE = true
```

RHEL/CentOS 略过

6.启动rsync服务

```
service rsync start
```

RHEL/CentOS中为

```
rsync --config=/etc/rsyncd.conf --daemon
echo 'rsync --config=/etc/rsyncd.conf --daemon' >> /etc/rc.local
```

7.创建swift recon缓存目录并设置权限

```
mkdir -p /var/swift/recon
chown -R swift:swift /var/swift/recon
```

安装和配置代理节点

在keystone增加用户角色及注册服务、端点

```
keystone tenant-create --name=swift
```

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |                                  |
|   enabled   |               True               |
|      id     | c266a7fe732949cc85e518ff63088cdb |
|     name    |              swift               |
+-------------+----------------------------------+

keystone user-create --name=swift --pass=swift --email=swift@example.com
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |        swift@example.com         |
| enabled  |               True               |
|    id    | 74cf4997282c4d309ded0099f7bf7fb1 |
|   name   |              swift               |
+----------+----------------------------------+

keystone role-create --name=Member
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|    id    | 3f942ac4182b44ccab8c8554ffa52cba |
|   name   |              Member              |
+----------+----------------------------------+

keystone role-create --name=sysadmin
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|    id    | 4c850ed673ac4f3a9dc71817eb35ccf1 |
|   name   |             sysadmin             |
+----------+----------------------------------+

keystone user-role-add --user=swift --role=Member --tenant=swift
keystone user-role-add --user=swift --role=sysadmin --tenant=swift
keystone user-role-add --user=admin --role=admin --tenant=swift

keystone service-create --name=Swift --type=object-store --description="Swift Object
Store Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |    Swift Object Store Service    |
|      id     | a0f8f55924a446ea9d7c9cb3cb49e02d |
|     name    |              swift               |
|     type    |           object-store           |
+-------------+----------------------------------+

keystone endpoint-create  --service-id=a0f8f55924a446ea9d7c9cb3cb49e02d   \
--publicurl=http://192.168.0.88:8080/v1/AUTH_%\(tenant_id\)s   \
--internalurl=http://192.168.0.88:8080/v1/AUTH_%\(tenant_id\)s   \
--adminurl=http://192.168.0.88:8080/v1/AUTH_%\(tenant_id\)s
+-------------+-------------------------------------------------+
|   Property  |                      Value                      |
+-------------+-------------------------------------------------+
|   adminurl  | http://192.168.0.88:8080/v1/AUTH_%(tenant_id)s |
|      id     |         eb512f403bf0442d8f2b4deb937b4526        |
| internalurl | http://192.168.0.88:8080/v1/AUTH_%(tenant_id)s |
|  publicurl  | http://192.168.0.88:8080/v1/AUTH_%(tenant_id)s |
|    region   |                    regionOne                    |
|  service_id |         a0f8f55924a446ea9d7c9cb3cb49e02d        |
```

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

代理服务器需要为每个请求查找账户、容器或对象的位置，并正确的路由请求。代理服务也处理API请求。可以通过proxy-server.conf启用账户管理。
#对象存储进程以一个独立的用户和组运行，它被配置为swift:swift，默认用户是swift，在你的系统上可能不存在这个用户。

1.安装swift-proxy服务
```
yum -y install openstack-swift-proxy memcached openstack-utils python-swiftclient
python-keystone-auth-token
```

2.创建一个ssl自签名证书
```
cd /etc/swift
openssl req -new -x509 -nodes -out cert.crt -keyout cert.key
```

3.修改memcached配置，使其只监听本地网络(swift可连接即可)
增加/修改启动参数 -l
```
sed -i '/OPTIONS/s/"$/ -l 192.168.0.88&/' /etc/sysconfig/memcached
```

4.运行memcached
```
service memcached restart
```

5.仅需在RHEL/CentOS/Fedora系统配置：要设置对象存储进行token验证，用openstack-config命令设置swift proxy文件中Identity服务的admin token
```
ADMIN_TOKEN=41e0ef4c7ce65f3e084a
openstack-config --set /etc/swift/proxy-server.conf \
  filter:authtoken admin_token $ADMIN_TOKEN
openstack-config --set /etc/swift/proxy-server.conf \
  filter:authtoken auth_token $ADMIN_TOKEN
openstack-config --set /etc/swift/proxy-server.conf  filter:authtoken auth_host
controller
openstack-config --set /etc/swift/proxy-server.conf  filter:cache memcache_servers
192.168.0.88:11211
openstack-config --set /etc/swift/proxy-server.conf filter:authtoken
admin_tenant_name service
openstack-config --set /etc/swift/proxy-server.conf filter:authtoken admin_user swift
openstack-config --set /etc/swift/proxy-server.conf filter:authtoken admin_password
swift
```

6. /etc/swift/proxy-server.conf
```
[DEFAULT]
bind_port = 8080
workers = 8
user = swift
log_facility = LOG_LOCAL1
eventlet_debug = true


[pipeline:main]
pipeline = healthcheck cache authtoken keystone proxy-server

[app:proxy-server]
use = egg:swift#proxy
allow_account_management = true
account_autocreate = true


[filter:cache]
use = egg:swift#memcache
memcache_servers = 192.168.0.88:11211

[filter:catch_errors]
use = egg:swift#catch_errors
```

```
[filter:healthcheck]
use = egg:swift#healthcheck

[filter:keystone]
#use = egg:swift#keystoneauth
paste.filter_factory = keystone.middleware.swift_auth:filter_factory
operator_roles = Member, admin, SwiftOperator
is_admin = true
cache = swift.cache

[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
service_port = 5000
service_host = controller
auth_uri = http://controller:5000/
admin_tenant_name = service
admin_user = swift
admin_password = swift
auth_host = controller
auth_port = 35357
auth_protocol = http
signing_dir = /tmp/keystone-signing-swift
admin_token = 41e0ef4c7ce65f3e084a
auth_token = 41e0ef4c7ce65f3e084a

[filter:proxy-logging]
use = egg:swift#proxy_logging

#如果有多个memcached,可以在proxy-server.conf文件中的 [filter:cache]段添加多个，如：
10.1.2.3:11211,10.1.2.4:11211

7.创建signing_dir并设置权限
mkdir -p /home/swift/keystone-signing
chown -R swift:swift /home/swift/keystone-signing

8.创建账户、容器及对象环。参数18表示2^18，设置分区大小，该值确定了分区的大小。设置基于你期望的整个环分区能力的总量。3表示每个对象副本的数量，最后的值是限制一小时内最多移动分区不超过1次。
cd /etc/swift
swift-ring-builder account.builder create 18 3 1
swift-ring-builder container.builder create 18 3 1
swift-ring-builder object.builder create 18 3 1

9.为每个环的每个节点添加存储设备条目，语法如下：
swift-ring-builder account.builder add z<ZONE>-
<STORAGE_LOCAL_NET_IP>:6002[R<STORAGE_REPLICATION_NET_IP>:6005]/<DEVICE> 100
swift-ring-builder container.builder add z<ZONE>-
<STORAGE_LOCAL_NET_IP_1>:6001[R<STORAGE_REPLICATION_NET_IP>:6004]/<DEVICE> 100
swift-ring-builder object.builder add z<ZONE>-
<STORAGE_LOCAL_NET_IP_1>:6000[R<STORAGE_REPLICATION_NET_IP>:6003]/<DEVICE> 100
#如果你不适用专用网络进行同步复制，必须去掉STORAGE_REPLICATION_NET_IP参数
例如，如果Zone1 有一个存储节点的IP是10.0.0.1，存储节点有一个IP为10.0.1.1的专用复制网络，挂载点是/srv/node/sdb1，并且在rsyncd.conf中的路径为/srv/node,那么添加设备sdb1的命令是：
swift-ring-builder account.builder add z1-10.0.0.1:6002R10.0.1.1:6005/sdb1 100
swift-ring-builder container.builder add z1-10.0.0.1:6001R10.0.1.1:6005/sdb1 100
swift-ring-builder object.builder add z1-10.0.0.1:6000R10.0.1.1:6005/sdb1 100

本例中ip均为192.168.0.88
swift-ring-builder account.builder add z1-192.168.0.88:6002/sdb1 100
swift-ring-builder container.builder add z1-192.168.0.88:6001/sdb1 100
swift-ring-builder object.builder add z1-192.168.0.88:6000/sdb1 100
#如果有5个区域，每个区域一个节点，以1为zone的起点，对于额外增加的节点，区域应加1.
```

10.验证每个环的内容
```
swift-ring-builder account.builder
account.builder, build version 1
262144 partitions, 3.000000 replicas, 1 regions, 1 zones, 1 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    192.168.0.88  6002    192.168.0.88              6002
sdb1 100.00          0 -100.00

swift-ring-builder container.builder
container.builder, build version 1
262144 partitions, 3.000000 replicas, 1 regions, 1 zones, 1 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    192.168.0.88  6001    192.168.0.88              6001
sdb1 100.00          0 -100.00

swift-ring-builder object.builder
object.builder, build version 1
262144 partitions, 3.000000 replicas, 1 regions, 1 zones, 1 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    192.168.0.88  6000    192.168.0.88              6000
sdb1 100.00          0 -100.00
```

11.环再平衡
```
swift-ring-builder account.builder rebalance
swift-ring-builder container.builder rebalance
swift-ring-builder object.builder rebalance
#再平衡需要一些时间
```

12.将11步生成的 account.ring.gz, container.ring.gz object.ring.gz copy到每个存储节点和代理节点的/etc/swift中。

13.确保swift对配置文件的所有权
```
chown -R swift:swift /etc/swift
```

14.启动代理服务
```
service openstack-swift-proxy start
```

在存储节点启动服务
```
sed --posix -i 's/\(^bind_ip = \)\(.*\)/\1192.168.0.88/g'
/etc/swift/{account,object,container}-server.conf

service openstack-swift-object start
service openstack-swift-object-replicator start
service openstack-swift-object-updater start
service openstack-swift-object-auditor start
service openstack-swift-container start
service openstack-swift-container-replicator start
service openstack-swift-container-updater start
service openstack-swift-container-auditor start
service openstack-swift-account start
service openstack-swift-account-replicator start
service openstack-swift-account-reaper start
service openstack-swift-account-auditor start

for s in `ls /etc/init.d/openstack-swift-*`;do chkconfig `basename $s` on;done
```

```
#一个节点上启动所有swift服务可以使用以下命令：
swift-init main start

service rsyslog restart
service memcached restart
```

对象存储服务安装后的工作

验证安装

你可以在代理服务器或者任何有权限访问Identity服务的服务器上运行这些命令

1.export swift管理员密码，设置一个添加到proxy-server.conf的Identity服务管理员密码并的变量。你可以参考*OpenStack User Guide*中描述的openrc文件中的OS_USERNAME变量

```
export OS_PASSWORD=swift
export OS_AUTH_URL=http://controller:5000/v2.0
#例如proxy-server.conf中使用swift作为ADMIN_PASS，如果你不想将admin password存到shell历史，可以运行以下命令
export SWIFT_PROXY_CONF=/etc/swift/proxy-server.conf export OS_PASSWORD=$( grep admin_password ${SWIFT_PROXY_CONF} | awk '{ print $NF }' )
```

2.运行一下命令验证安装

```
swift -V 2.0 -A $OS_AUTH_URL -U service:swift -K $OS_PASSWORD stat
   Account: AUTH_c09af41a86ae422bbb0181e2ec6a3b47
Containers: 0
   Objects: 0
     Bytes: 0
Content-Type: text/plain; charset=utf-8
X-Timestamp: 1388474969.68056
X-Put-Timestamp: 1388474969.68056

 curl -s -d '{"auth": {"tenantName": "admin", "passwordCredentials":{"username": "admin", "password": "openstack"}}}' -H "Content-type: application/json" http://controller:35357/v2.0/tokens | python -mjson.tool
```

3.上传文件到容器

```
swift -V 2.0 -A $OS_AUTH_URL -U service:swift -K $OS_PASSWORD upload myfiles cert.key
swift -V 2.0 -A $OS_AUTH_URL -U service:swift -K $OS_PASSWORD upload myfiles cert.crt
```

4.从myfiles容器下载文件

```
swift -V 2.0 -A $OS_AUTH_URL -U service:swift -K $OS_PASSWORD download myfiles
```

增加代理服务器

对于可靠性，你可以增加代理服务器。你可以像配置第一个节点一样配置一个额外的代理节点，但需要额外的步骤。

如果你有两个以上的节点，必须配置负载均衡。你的存储端点（客户端连接到的存储）也发生改变。你可以使用不同的负载均衡策略。例如你可以使用DNS轮询，或者在两个代理前端增加软件或硬件的负载均衡器，存储的url指向负载均衡器。

配置一个初始节点，然后用以下步骤增加一个节点。

1.更新/etc/swift/proxy-server.conf文件中的memcached服务器列表。

```
[filter:cache]
use = egg:swift#memcache
memcache_servers = <PROXY_LOCAL_NET_IP>:11211
```

2.copy环信息到所有节点，包括新增加的代理节点。确保环信息中包括所有的存储节点。

3.同步所有节点后，确保/etc/swift中存在admin的秘钥，并且环文件的权限正确。

安装网络服务

网络方面的考虑

OpenStack网络驱动范围从软件桥到完全控制的某些硬件交换设备。本手册主要讲解Open vSwitch设备，本文提出的理论大多适用于其它机制，更多信息请参考OpenStack Configuration Reference的Networking一章

```
#如果你已经使用nova-network配置了网络，这里的配置将覆盖nova-network配置。
```

Neutron 概述

类似Nova 网络， Neutron管理OpenStack的软件定义网络。然而不像Nova网络，你可以为高级虚拟网络拓扑配置Neutron，如为每个租户配置专用网络等。

Neutron有以下抽象对象： 网络、子网和路由器。每一个对象模拟对应的物理设备：网络包含子网，路由器路由不同网络和子网之间的流量。

任何Neutron至少包含一个外部网络。这个网络不像其它网络，不是一个实际的网络定义。它代表 OpenStack 访问外部网络的部分。Neutron的外部网络上的IP地址是外部网络的任何人都可以访问的。由于这种网络仅仅是外部网络的一部分，所以要在此网络禁用DHCP。

除了外部网络，任何Neutron都拥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。只有在指定内部网络上的虚拟机或那些通过类似路由器连接的子网可以访问虚拟机直连的网络。

外部网络访问虚拟机必须通过路由器，反之亦然。每个路由器都有连接到子网的网关，并且有连接到子网的许多接口。像物理路由器一样，一个子网可以访问连接到同一路由器的其它子网，机器可以通过网关路由器访问外网。

此外，你可以为内部网络分配外网ip的端口。每当有一个网络连接，这个连接即被称为端口。你可以将虚拟机与外网ip的端口关联，这样外部网络的实体就可以访问虚拟机。

Neutron 也支持安全组，安全组管理员可以组中定义防火墙规则。一个虚拟机可以属于一个或多个安全组，Neutron可以应用这些安全组来禁止或允许端口、端口范围或者虚拟机的流量类型。

Neutron 的每个插件都有自己的概念。 虽然理解这些概念对于操作Neutron不是至关重要的，但它有助于帮助你配置Neutron。所有的Neutron安装都使用一个核心插件和安全组插件（或者是无操作的安全组插件）。此外，还可以使用Firewall-as-a-service (FWaaS) 和Load-balancing-as-a-service (LBaaS) 插件。

OpenvSwitch概述

OpenvSwitch是一个最受欢迎的核心插件，OpenvSwitch由桥接和端口构成。端口指可以连接到其它东西如物理接口或电缆。桥上任何端口的数据都与其它端口共享。桥可以通过Open vSwitch virtual patch cables或者Linux virtual Ethernet cables (veth)进行连接。此外桥作为linux的端口出现，所以可以为它分配IP地址。

在Neutron中集成的桥叫做bt-int。直接连到虚拟机或相关服务。外部桥叫做br-ex，连接到外部网络。最后OpenvSwitch的VLAN插件配置使桥与物理网络关联。

除了定义桥，OpenvSwitch还支持OpenFlow，这允许你定义网络流规则。某些配置中使用这些规则来传输VLAN之间的数据包。

OpenvSwitch的一些配置使用网络命名空间，使Linux的分组适配器称为唯一的命名空间，这对于其它命名空间是不可见的，这使得同一网络节点可以管理多个Neutron路由器。

使用OpenvSwitch，你可以使用两种不同的技术创建虚拟网络：GRE或者VLAN。

Generic Routing Encapsulation (GRE)在许多VPN技术中使用。它为不同的路由信息封装IP数据包，创建全新的数据包。当新数据包到达目的地时，它被解包，然后交由底层进行分组路由。在OpenvSwitch中使用GRE,Neutron创建GRE隧道，这些隧道就像是桥的端口和在不同的系统中桥之间的桥，这允许计算和网络节点充当路由器。

Virtual LANs (VLANs)修改以太网报头，它添加一个4字节的vlan标记范围从1到4094（0和4095是个特殊的标签，加工所有数据，相当于一个未标记的包）。网卡、交换机、路由器知道如何解释vlan标记，OpenvSwitch也一样。
一个标记vlan的数据包只能与标记为该vlan的设备共享，即使所有设备在同一网络。

大多数常见的安全组支持IPTables或OpenvSwitch插件混合使用。它使用IPTables和OpenFlow规则组合。使用 IPTables可以在Linux中创建防火墙NAT规则。该工具采用了复杂的规则体系和规则链，以适应Neutron安全组所需的复杂规则。

安装网络服务

在为各个节点配置网络之前，你必须创建所需的OpenStack组件：用户、服务和数据库以及一个或多个端

点。当你在控制节点完成这些步骤后，安装本手册配置OpenStack网络节点。
1.创建Neutron数据库
```
mysql -u root -e "CREATE DATABASE neutron;GRANT ALL PRIVILEGES ON neutron.* TO
'neutron'@'localhost' IDENTIFIED BY 'openstack';GRANT ALL PRIVILEGES ON neutron.* TO
'neutron'@'%' IDENTIFIED BY 'openstack'"
```

2. 创建所需的用户、服务及端点
```
keystone user-create --name=neutron --pass=neutron --email=neutron@example.com
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |       neutron@example.com        |
| enabled  |               True               |
|    id    | 9859c06f57d641feab26fea918a72cda |
|   name   |             neutron              |
+----------+----------------------------------+

keystone user-role-add --user=neutron --tenant=service --role=admin
keystone service-create --name=neutron --type=network \
     --description="OpenStack Networking Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |   OpenStack Networking Service   |
|      id     | 0464095d4e9b4f109b044d9b72ce3805 |
|     name    |             neutron              |
|     type    |             network              |
+-------------+----------------------------------+

keystone endpoint-create \
     --service-id=0464095d4e9b4f109b044d9b72ce3805 \
     --publicurl=http://controller:9696 \
     --adminurl=http://controller:9696 \
     --internalurl=http://controller:9696
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  |      http://controller:9696      |
|      id     | ad6fa5211b2d4f10aabe7d07b0c60d1c |
| internalurl |      http://controller:9696      |
|  publicurl  |      http://controller:9696      |
|    region   |            regionOne             |
|  service_id | 0464095d4e9b4f109b044d9b72ce3805 |
+-------------+----------------------------------+
```

在一个专用节点上安装网络服务
在开始设置一台机器作为一个专门的网络节点之前，专用的网络节点需要有一个MGMT_INTERFACE NIC，一个DATA_INTERFACE NIC和一个EXTERNAL_INTERFACE NIC.
管理网络处理节点之间的沟通，数据网络处理出入虚拟机的流量，外部网络连接到网络节点及可选的控制器节点，这样你的虚拟机才可以连接到外网。
所有网卡必须有静态IP，然而数据网卡和外部网卡有个特殊的设置。查看网络插件的详细信息参考“Install and configure the Networking plug-ins”.

默认情况下， RHEL使用system-config-firewall 自动配置防火墙规则，这个图形界面（以及在最后加-tui的接口）为你配置了一个基本的iptables防火墙，当你使用网络时你应该禁用它，除非你熟悉底层网络技术。默认情况下它阻止多种类型的网络流量。
当你设置好OpenStack 网络后，你可以重新配置和启用这个工具。然而禁用这个工具使调试网络问题更加容易。

1.在网络节点安装网络服务
```
yum -y install openstack-neutron
```

2.配置网络服务开机启动

```
for s in neutron-{dhcp,metadata,l3}-agent; do chkconfig $s on; done
```

3.配置/etc/sysctl.conf，使网络节点可以调整虚拟机流量

```
cat >> /etc/sysctl.conf <<EOF
net.ipv4.ip_forward=1
net.ipv4.conf.all.rp_filter=0
net.ipv4.conf.default.rp_filter=0
EOF

sysctl -p
```

重启网络（建议，非必须）

```
service network restart
```

4.编辑/etc/neutron/neutron.conf,配置Neutron 使用Keystone进行身份验证

a.启用keystone

```
sed -i '/^# auth_strategy =/aauth_strategy = keystone' /etc/neutron/neutron.conf
```

b.修改认证配置

```
sed -i '/\[keystone_authtoken\]/aauth_host = controller\nauth_port =
35357\nauth_protocol = http\nadmin_tenant_name = service\nadmin_user =
neutron\nadmin_password = neutron' /etc/neutron/neutron.conf
```

5.启用[agent]段的root_helper

```
sed -i --posix 's/\(^[# ]*\)\(root_helper.*\)/\2/g' /etc/neutron/neutron.conf
```

6.修改 DEFAULT 段，启用消息服务

```
sed -i '1arpc_backend = neutron.openstack.common.rpc.impl_qpid\nqpid_hostname =
controller\nqpid_port = 5672\nqpid_username = guest\nqpid_password = guest'
/etc/neutron/neutron.conf
```

7.配置数据库连接

```
openstack-config --set /etc/neutron/neutron.conf  database connection
mysql://neutron:openstack@controller/neutron
```

控制节点创建数据库

```
mysql -uroot -e "create database neutron;grant all on neutron.* to neutron@'%'
identified by 'openstack';"
```

8.编辑/etc/neutron/api-paste.ini，修改 [filter:authtoken]配置

```
openstack-config --set /etc/neutron/api-paste.ini filter:authtoken auth_host
controller
openstack-config --set /etc/neutron/api-paste.ini filter:authtoken auth_uri
http://controller:5000
openstack-config --set /etc/neutron/api-paste.ini filter:authtoken admin_tenant_name
service
openstack-config --set /etc/neutron/api-paste.ini filter:authtoken admin_user neutron
openstack-config --set /etc/neutron/api-paste.ini filter:authtoken admin_password
neutron
```

9.安装和配置网络插件。OpenStack使用这个插件完成软件定义网络。请查看instructions的说明，然后回到这里。

现在你已经配置了一个插件，是时候配置其余的部分了。

1.要在软件定义的网络上运行DHCP，OpenStack网络支持几种不同的插件。然而，一般使用dnsmasq。修改/etc/neutron/dhcp_agent.ini ：

```
sed -i '/^# dhcp_driver =/adhcp_driver = neutron.agent.linux.dhcp.Dnsmasq'
/etc/neutron/dhcp_agent.ini
```

2.允许虚拟机访问计算元数据信息，必须配置并启用网络元数据代理。该代理将作为计算元数据的代理。

在控制器上编辑 /etc/nova/nova.conf来定义一个秘钥，它将用于计算服务和网络元数据代理之间使用。

```
openstack-config --set /etc/nova/nova.conf DEFAULT
neutron_metadata_proxy_shared_secret 1234567890
openstack-config --set /etc/nova/nova.conf DEFAULT \
    service_neutron_metadata_proxy true
```

重启nova-api 服务

```
service openstack-nova-api restart
```

在网络节点修改元数据代理的配置

```
openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT \
    auth_url http://controller:5000/v2.0
openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT \
    auth_region regionOne
openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT \
    admin_tenant_name service
openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT \
    admin_user neutron
openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT \
    admin_password neutron
openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT \
    nova_metadata_ip controller
openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT \
    metadata_proxy_shared_secret 1234567890
#auth_region的值是区分大小写的，必须与Keystone中定义的端点region 相同
```

3.重启网络服务

```
ln -s /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini /etc/neutron/plugin.ini

service neutron-server restart
service neutron-dhcp-agent restart
service neutron-l3-agent restart
service neutron-metadata-agent restart
```

也重启一下网络插件代理，例如openvSwitch：

```
service neutron-openvswitch-agent restart
```

4.然后配置计算节点、控制节点，配置基础网络。

instructions的说明如下：

安装openvswitch插件

1.安装openvswitch及其依赖

```
yum -y install openstack-neutron-openvswitch
```

2.启动openvswitch

```
service openvswitch start
chkconfig openvswitch on
```

3.不管你使用哪种网络技术，你必须添加连接到虚拟机的br-int,以及连接外网的br-ex

```
ovs-vsctl add-br br-int
ovs-vsctl add-br br-ex
```

4.添加一个端口到 br-ex (eth2为EXTERNAL_INTERFACE)

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

5.配置eth2（EXTERNAL_INTERFACE）为混杂模式，并且不配置IP.此外你需要重新创建一个br-ex接口的配置文件，并且配置为原eth2的IP.

```
vi /etc/sysconfig/network-scripts/ifcfg-eth2
DEVICE=eth2
TYPE=Ethernet
ONBOOT=no
BOOTPROTO=none
PROMISC=yes
```

6.新建/etc/sysconfig/network-scripts/ifcfg-br-ex
vi /etc/sysconfig/network-scripts/ifcfg-br-ex
DEVICE=br-ex
TYPE=Bridge
ONBOOT=no
BOOTPROTO=none
IPADDR=192.168.222.10
NETMASK=255.255.255.0
#GATEWAY=192.168.222.1

7.无论你使用OpenvSwitch的哪种网络技术，你都必须配置一些常用选项。配置L3和DHCP使用OVS及命名空间。编辑/etc/neutron/l3_agent.ini 和 /etc/neutron/dhcp_agent.ini，修改如下：
sed -i '1ainterface_driver =
neutron.agent.linux.interface.OVSInterfaceDriver\nuse_namespaces = True'
/etc/neutron/{l3,dhcp}_agent.ini

某些特殊的内核中必须启用veth支持。比如最近的RHEL和CentOS，只有部分支持命名空间。编辑刚才的文件，修改以下配置：
sed -i '3aovs_use_veth = True' /etc/neutron/{l3,dhcp}_agent.ini

8.同时，你也必须告诉Neutron使用OVS，编辑 /etc/neutron/neutron.conf。
sed -i '/core_plugin/acore_plugin =
neutron.plugins.openvswitch.ovs_neutron_plugin.OVSNeutronPluginV2'
/etc/neutron/neutron.conf

9.选择一种虚拟网络技术。neutron支持GRE隧道、VLAN以及VXLAN。本手册展示如何配置GRE及VLAN.
GRE隧道的配置是比较简单的，因为它不需要物理网络任何特殊的配置。然而这个协议很难过在物理网络上过滤流量。此外，这个配置不使用命名空间，你只能为每个网络节点配置一个路由器。但是你可以启用命名空间及veth，本小节中描述了如何在OVS中配置VLAN。

另一方面，VLAN 标记修改了以太网包的报头，你可以在物理网络以正常的方式过滤数据包。然而并非所有网卡都支持，vlan标记一定程度上增加了包的大小。你可能需要在物理网络上进行额外的配置，以确保Neutron的vlan不干扰你网络上的其它vlan，以及各网络节点的物理硬件不剥离vlan标记。

#尽管本手册中的示例默认使用网络命名空间，如果它们出现问题或你的内核不支持，你可以在
/etc/neutron/l3_agent.ini 和 /etc/neutron/dhcp_agent.ini中禁用它们：
use_namespaces = False

修改/etc/neutron/neutron.conf禁止IP地址重复
sed -i '/^# allow_overlapping_ips =/aallow_overlapping_ips = False'
/etc/neutron/neutron.conf

#当禁用网络命名空间时，你的每个网络节点只能有一个路由器，并且不支持IP地址重复。

10.配置防火墙插件。如果你不希望执行防火墙规则（OpenStack中称为安全组），你可以使用
neutron.agent.firewall.NoopFirewall，如果不是，你可以选择一个防火墙插件。最常见的是结合OVS-Iptables驱动，但你也可以使用Firewall-as-a-Service 驱动。编
辑/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini文件：
sed -i '/\[securitygroup\]/afirewall_driver =
neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver'
/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini

#警告：你必须使用防火墙，哪怕是No-Op。否则Horizon 和其它OpenStack服务不能获取或设置虚拟机启动选项。

11.配置OVS插件开机启动
chkconfig neutron-openvswitch-agent on

12.现在回到OVS 配置说明。

配置OVS使用GRE隧道
1.配置OVS使用GRE隧道，需要在/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini中配置br-int，br-tun以及数据接口隧道的IP

```
sed -i '/^\[ovs\]/atenant_network_type = gre\ntunnel_id_ranges =
1:1000\nenable_tunneling = True\nintegration_bridge = br-int\ntunnel_bridge = br-
tun\nlocal_ip = 172.16.0.254' /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini
```

2.回到OVS 配置说明

配置OVS 使用vlan
1.配置OVS使用VLAN

```
sed -i '/^\[ovs\]/atenant_network_type = vlan\nnetwork_vlan_ranges =
physnet1:1:4094\nbridge_mappings = physnet1:br-int'
/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini
```

2.创建 DATA_INTERFACE，并将物理网卡添加到 DATA_INTERFACE

```
ovs-vsctl add-br br-int
ovs-vsctl add-port br-int eth1
```

3.按照 EXTERNAL_INTERFACE 配置到 br-ex的方式配置DATA_INTERFACE，但是不要打开混杂模式。

```
cat > /etc/sysconfig/network-scripts/ifcfg-eth1 <<EOF
DEVICE=eth1
TYPE=Ethernet
ONBOOT=yes
BOOTPROTO=none
PROMISC=no
EOF

cat > /etc/sysconfig/network-scripts/ifcfg-br-int <<EOF
DEVICE=br-int
TYPE=Bridge
ONBOOT=no
BOOTPROTO=none
IPADDR=172.16.0.254
NETMASK=255.255.255.0
EOF
```

4.回到OVS 配置说明

专门的计算节点上启用网络支持

#本节设置任意运行nova-compute组件，但不运行完整网络栈的节点。
#警告：默认情况下，在RHEL中system-config-firewall 工具自动配置了系统防火墙，这个图形界面允许你配置一个iptables作为基本的防火墙。当你使用Neutron时应该禁用它，除非你特别熟悉基础网络技术。例如默认情况下它组织Neutron网络的重要流量。
当你成功为OpenStack配置了Neutron网络时，你可以重新启用和配置它，然而在Neutron中禁用它将使网络问题的调试更加容易。

1.禁用数据包目的地筛选（路由验证）使网络服务路由流量到虚拟机中。

```
cat >> /etc/sysctl.conf <<EOF
net.ipv4.conf.all.rp_filter=0
net.ipv4.conf.default.rp_filter=0
EOF

sysctl -p
```

2.安装和配置网络插件组件。当你设置好网络节点后安装和配置网络插件，查看“Install and configure Neutron plug-ins on a dedicated compute node”.

3.配置Neutron核心。

```
openstack-config --set /etc/neutron/neutron.conf keystone_authtoken auth_host
controller
openstack-config --set /etc/neutron/neutron.conf keystone_authtoken admin_tenant_name
service
openstack-config --set /etc/neutron/neutron.conf keystone_authtoken admin_user
neutron
```

```
openstack-config --set /etc/neutron/neutron.conf keystone_authtoken admin_password
neutron
openstack-config --set /etc/neutron/neutron.conf keystone_authtoken auth_url
http://controller:35357/v2.0
openstack-config --set /etc/neutron/neutron.conf keystone_authtoken auth_strategy
keystone
openstack-config --set /etc/neutron/neutron.conf keystone_authtoken rpc_backend
neutron.openstack.common.rpc.impl_qpid
openstack-config --set /etc/neutron/neutron.conf keystone_authtoken qpid_hostname
controller
```

4.配置[agent]部分

```
openstack-config --set /etc/neutron/neutron.conf agent root_helper "sudo neutron-
rootwrap /etc/neutron/rootwrap.conf"
```

5.配置[database]

```
openstack-config --set /etc/neutron/neutron.conf database connection =
mysql://neutron:openstack@controller/neutron
```

6.编辑/etc/neutron/api-paste.ini，配置 [filter:authtoken]

```
openstack-config --set /etc/neutron/api-paste.ini filter:authtoken
paste.filter_factory keystoneclient.middleware.auth_token:filter_factory
openstack-config --set /etc/neutron/api-paste.ini filter:authtoken auth_host
controller
openstack-config --set /etc/neutron/api-paste.ini filter:authtoken admin_tenant_name
service
openstack-config --set /etc/neutron/api-paste.ini filter:authtoken admin_user neutron
openstack-config --set /etc/neutron/api-paste.ini filter:authtoken admin_password
neutron
```

7.配置OpenStack计算使用OpenStack网络服务。

```
openstack-config --set /etc/nova/nova.conf DEFAULT network_api_class
nova.network.neutronv2.api.API
openstack-config --set /etc/nova/nova.conf DEFAULT neutron_url http://controller:9696
openstack-config --set /etc/nova/nova.conf DEFAULT neutron_auth_strategy keystone
openstack-config --set /etc/nova/nova.conf DEFAULT neutron_admin_tenant_name service
openstack-config --set /etc/nova/nova.conf DEFAULT neutron_admin_username neutron
openstack-config --set /etc/nova/nova.conf DEFAULT neutron_admin_password neutron
openstack-config --set /etc/nova/nova.conf DEFAULT neutron_admin_auth_url
http://controller:35357/v2.0
openstack-config --set /etc/nova/nova.conf DEFAULT linuxnet_interface_driver
nova.network.linux_net.LinuxOVSInterfaceDriver
openstack-config --set /etc/nova/nova.conf DEFAULT firewall_driver
nova.virt.firewall.NoopFirewallDriver
openstack-config --set /etc/nova/nova.conf DEFAULT security_group_api neutron
```

#当你配置网络节点和计算节点时，无论选择哪种防火墙驱动，你必须编辑/etc/nova/nova.conf设置防火墙驱动为 nova.virt.firewall.NoopFirewallDriver。因为OpenStack网络控制防火墙，这个语句告诉OpenStack计算不使用防火墙。

#如果你希望网络支持防火墙，编辑/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini设置firewall_driver选项为一个防火墙插件。例如OVS:

```
openstack-config --set /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini \
securitygroup firewall_driver
neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
```

#如果你不想在计算或网络中使用防火墙，编辑两个配置文件，并设置firewall_driver=nova.virt.firewall.NoopFirewallDriver，还需要注释或删除掉/etc/nova/nova.conf中的security_group_api=neutron。
否则当你使用nova list命令时，你将收到一个错误：The server has either erred or is incapable of performing the requested operation. 可能返回(HTTP 500)

8.重启计算服务和网络插件

```
service openstack-nova-compute restart
service neutron-openvswitch-agent restart
chkconfig neutron-openvswitch-agent on
```

在计算节点安装openvSwitch
1.安装OpenvSwitch插件
```
yum -y install openstack-neutron-openvswitch
```

2.启动openvswitch
```
service openvswitch start
chkconfig openvswitch on
```

3.无论你使用OpenvSwitch的哪种技术，你都必须配置一些常用的配置。你必须添加一个br-int的集成网桥，它连接到虚拟机。
```
ovs-vsctl add-br br-int
```

4.你必须配置一些常用的选项。必须配置网络核心使用OVS。
```
openstack-config --set /etc/neutron/neutron.conf DEFAULT core_plugin
neutron.plugins.openvswitch.ovs_neutron_plugin.OVSNeutronPluginV2
openstack-config --set /etc/neutron/neutron.conf DEFAULT api_paste_config
/etc/neutron/api-paste.ini
openstack-config --set /etc/neutron/neutron.conf DEFAULT rpc_backend
neutron.openstack.common.rpc.impl_qpid
```

5.配置你设置网络节点时选择的网络类型： GRE tunneling 或者 VLANs.
在计算节点配置OVS使用GRE tunneling
1.告诉OVS插件使用GRE隧道使用的集成网桥br-int、隧道网桥br-tun以及DATA_INTERFACE的本地IP
```
openstack-config --set  /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini ovs
tenant_network_type gre
openstack-config --set  /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini ovs
tunnel_id_ranges 1:1000
openstack-config --set  /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini ovs
enable_tunneling True
openstack-config --set  /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini ovs
integration_bridge br-int
openstack-config --set  /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini ovs
tunnel_bridge br-tun
openstack-config --set  /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini ovs
local_ip 172.16.0.253
```

2.现在回到OVS 配置

在计算节点配置OVS使用VLAN
1.配置OVS使用VLAN
```
openstack-config --set  /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini ovs
tenant_network_type vlan
openstack-config --set  /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini ovs
network_vlan_ranges physnet1:1:4094
openstack-config --set  /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini ovs
bridge_mappings physnet1:br-int
```

2.以类似网络节点的配置方式创建一个桥，并将DATA_INTERFACE物理端口加入DATA_INTERFACE桥
```
ovs-vsctl add-br br-int
ovs-vsctl add-port br-int eth1
```

3.现在回到OVS 配置
##############################

在独立的控制节点安装网络支持
#这个节点运行Neutron的控制组件，但不运行任何能够提供相关功能(如插件代理或L3代理)的组件。如果你希望整合计算节点或控制器节点，请按照说明应用于计算节点。

#警告：默认情况下，在RHEL中system-config-firewall 工具自动配置了系统防火墙，这个图形界面允许你配置一个iptables作为基本的防火墙。当你使用Neutron时应该禁用它，除非你特别熟悉基础网络技术。例如默认情况下它组织Neutron网络的重要流量。
当你成功为OpenStack配置了Neutron网络时，你可以重新启用和配置它，然而在Neutron中禁用它将使网络问题的调试更加容易。

1.安装neutron服务器、python库和neutron命令行接口
```
yum -y install openstack-neutron python-neutron python-neutronclient
```

2.配置Neutron核心组件。
```
openstack-config --set  /etc/neutron/neutron.conf keystone_authtoken auth_host
controller
openstack-config --set  /etc/neutron/neutron.conf keystone_authtoken
admin_tenant_name service
openstack-config --set  /etc/neutron/neutron.conf keystone_authtoken admin_user
neutron
openstack-config --set  /etc/neutron/neutron.conf keystone_authtoken admin_password
NEUTRON_PASS
openstack-config --set  /etc/neutron/neutron.conf keystone_authtoken auth_url
http://controller:35357/v2.0
openstack-config --set  /etc/neutron/neutron.conf keystone_authtoken auth_strategy
keystone

openstack-config --set  /etc/neutron/neutron.conf DEFAULT rpc_backend
neutron.openstack.common.rpc.impl_qpid
openstack-config --set  /etc/neutron/neutron.conf DEFAULT qpid_hostname controller
openstack-config --set  /etc/neutron/neutron.conf DEFAULT qpid_port 5672
openstack-config --set  /etc/neutron/neutron.conf DEFAULT qpid_username guest
openstack-config --set  /etc/neutron/neutron.conf DEFAULT qpid_password guest
```

3.编辑[agent]
```
openstack-config --set  /etc/neutron/neutron.conf agent root_helper sudo neutron-
rootwrap /etc/neutron/rootwrap.conf
```

4.配置[database]
```
openstack-config --set  /etc/neutron/neutron.conf database connection
mysql://neutron:openstack@controller/neutron
```

5.编辑/etc/neutron/api-paste.ini,配置认证
```
openstack-config --set  /etc/neutron/api-paste.ini filter:authtoken
paste.filter_factory keystoneclient.middleware.auth_token:filter_factory
openstack-config --set  /etc/neutron/api-paste.ini filter:authtoken admin_tenant_name
service
openstack-config --set  /etc/neutron/api-paste.ini filter:authtoken admin_user
neutron
openstack-config --set  /etc/neutron/api-paste.ini filter:authtoken admin_password
neutron
```

6.在这个节点上配置网络插件。
尽管这个节点不运行任何提供潜在功能的插件，但neutron-server服务必须知道哪些插件是在运行的，因为它是插件的接口

7.告诉nova关于Neutron的信息。确切的说，你必须告诉nova Neutron控制网络和防火墙。
```
openstack-config --set /etc/nova/nova.conf DEFAULT network_api_class
nova.network.neutronv2.api.API
openstack-config --set /etc/nova/nova.conf DEFAULT neutron_url http://controller:9696
openstack-config --set /etc/nova/nova.conf DEFAULT neutron_auth_strategy keystone
openstack-config --set /etc/nova/nova.conf DEFAULT neutron_admin_tenant_name service
openstack-config --set /etc/nova/nova.conf DEFAULT neutron_admin_username neutron
openstack-config --set /etc/nova/nova.conf DEFAULT neutron_admin_password neutron
openstack-config --set /etc/nova/nova.conf DEFAULT neutron_admin_auth_url
http://controller:35357/v2.0
openstack-config --set /etc/nova/nova.conf DEFAULT linuxnet_interface_driver
```

```
nova.network.linux_net.LinuxOVSInterfaceDriver
openstack-config --set /etc/nova/nova.conf DEFAULT firewall_driver
nova.virt.firewall.NoopFirewallDriver
openstack-config --set /etc/nova/nova.conf DEFAULT security_group_api neutron

#当你配置网络节点和计算节点时，无论选择哪种防火墙驱动，你必须编辑/etc/nova/nova.conf设置防火墙为 No-op,这是nova防火墙。因为Neutron控制防火墙，这个语句告诉Nova不使用防火墙。

#如果你希望网络支持防火墙，编辑/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini设置firewall_driver选项为一个防火墙插件。例如OVS:
openstack-config --set /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini \
securitygroup firewall_driver
neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver

#如果你不想在计算或网络中使用防火墙，编辑两个配置文件，并设置
firewall_driver=nova.virt.firewall.NoopFirewallDriver，还需要注释或删除
掉/etc/nova/nova.conf中的security_group_api=neutron。
否则当你使用nova list命令时，你将收到一个错误：The server has either erred or is
incapable of performing the requested operation. 可能返回(HTTP 500)

8.启动neutro-server
ln -s /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini /etc/neutron/plugin.ini
service neutron-server start
chkconfig neutron-server on

#确保服务启动成功。如果你得到丢失plugin.ini文件的错误，创建一个链接文
件/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini 到/etc/neutron/plugin.ini

在独立的控制器节点安装Neutron 插件
1.安装openvswitch插件
yum -y install openstack-neutron-openvswitch

2.无论使用openvswitch的哪种网络技术，你必须配置一些常用选项。你必须配置网络使用OVS。
openstack-config --set /etc/neutron/neutron.conf DEFAULT core_plugin
neutron.plugins.openvswitch.ovs_neutron_plugin.OVSNeutronPluginV2

3.配置ovs插件的网络类型：GRE隧道或VLAN
控制节点不需要运行openvSwitch或openvSwitch代理

4.返回OVS手册

在控制节点配置OVS插件使用GRE隧道
1.配置ovs插件使用GRE隧道
openstack-config --set /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini ovs
tenant_network_type gre
openstack-config --set /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini ovs
tunnel_id_ranges  1:1000
openstack-config --set /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini ovs
enable_tunneling  True

2.回到OVS配置手册

在控制节点配置OVS插件使用vlan
1.配置OVS使用vlan
openstack-config --set /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini ovs
tenant_network_type vlan
openstack-config --set /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini ovs
network_vlan_ranges physnet1:1:4094

2.返回OVS 配置手册
################################

创建一个基本的Neutron网络
```

#替换SPECIAL_OPTIONS为你选择的网络插件选项。查看here检查你的插件需要的任何特殊选项。

1.创建ext-net外部网络。这个网络是外部网络的一部分，虚拟机不直接连接到这个网络；它们连接到内部网络。出去的流量通过neutron路由到外部网络。此外，可能从ext-net子网分配一个浮动IP地址到虚拟机，所以外部网络可以联系它们。Neutron恰当的路由流量。

```
neutron net-create ext-net -- --router:external=True SPECIAL_OPTIONS
```

2.创建与EXTERNAL_INTERFACE相同的子网和网关。它不能有DHCP，因为它是外网的一部分。

```
neutron subnet-create ext-net \
  --allocation-pool start=FLOATING_IP_START,end=FLOATING_IP_END \
  --gateway=EXTERNAL_INTERFACE_GATEWAY --enable_dhcp=False \
  EXTERNAL_INTERFACE_CIDR
```

3.创建一个或多个初始租户：

```
keystone tenant-create --name DEMO_TENANT
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |                                  |
|   enabled   |               True               |
|      id     | 6fd51dd30c3c43c489296d56c92181e5 |
|     name    |           DEMO_TENANT            |
+-------------+----------------------------------+
```

查看“Define users, tenants, and roles”了解细节。

4.创建一个连接到外网的路由器。该路由器路由流量到恰当的内部子网。你可以为一个指定的租户创建：增加参数--tenant-id，值为 DEMO_TENANT_ID

```
keystone tenant-list | grep DEMO_TENANT | awk '{print $2;}'
6fd51dd30c3c43c489296d56c92181e5
```

创建路由

```
neutron router-create ext-to-int --tenant-id 6fd51dd30c3c43c489296d56c92181e5
Created a new router:
+-----------------------+--------------------------------------+
| Field                 | Value                                |
+-----------------------+--------------------------------------+
| admin_state_up        | True                                 |
| external_gateway_info |                                      |
| id                    | e7732943-9e3e-4c3b-ac95-c3a2adbca3a3 |
| name                  | ext-to-int                           |
| status                | ACTIVE                               |
| tenant_id             | 6fd51dd30c3c43c489296d56c92181e5     |
+-----------------------+--------------------------------------+
```

5.通过设置网关将路由器作为ext-net连接到ext-net ：

```
neutron router-gateway-set EXT_TO_INT_ID EXT_NET_ID
```

6.为DEMO_TENANT创建一个内部网络

```
neutron net-create --tenant-id 6fd51dd30c3c43c489296d56c92181e5 demo-net
Created a new network:
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | e414aac9-a1a1-4b77-8b9f-ed8861971d03 |
| name                      | demo-net                             |
| provider:network_type     | gre                                  |
| provider:physical_network |                                      |
| provider:segmentation_id  | 1                                    |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
```

```
| tenant_id                 | 6fd51dd30c3c43c489296d56c92181e5     |
+---------------------------+--------------------------------------+

neutron subnet-create --name demo-net-sub --tenant-id
6fd51dd30c3c43c489296d56c92181e5 demo-net 172.16.0.0/24 --gateway 172.16.0.1
Created a new subnet:
+------------------+----------------------------------------------------+
| Field            | Value                                              |
+------------------+----------------------------------------------------+
| allocation_pools | {"start": "172.16.0.2", "end": "172.16.0.254"}     |
| cidr             | 172.16.0.0/24                                      |
| dns_nameservers  |                                                    |
| enable_dhcp      | True                                               |
| gateway_ip       | 172.16.0.1                                         |
| host_routes      |                                                    |
| id               | feb72054-8cc5-4ed5-b38b-96c17e0da787               |
| ip_version       | 4                                                  |
| name             | demo-net-sub                                       |
| network_id       | e414aac9-a1a1-4b77-8b9f-ed8861971d03               |
| tenant_id        | 6fd51dd30c3c43c489296d56c92181e5                   |
+------------------+----------------------------------------------------+

neutron router-interface-add ext-to-int demo-net-sub
Added interface 5dc56841-04b4-4059-ad09-7aab4bd5f590 to router ext-to-int.
```

7.其余的步骤是检查插件的特殊选项页。现在回到OVS配置。

Neutron部署示例

单一扁平网络

Install

Configure logical network

Use case: single flat network

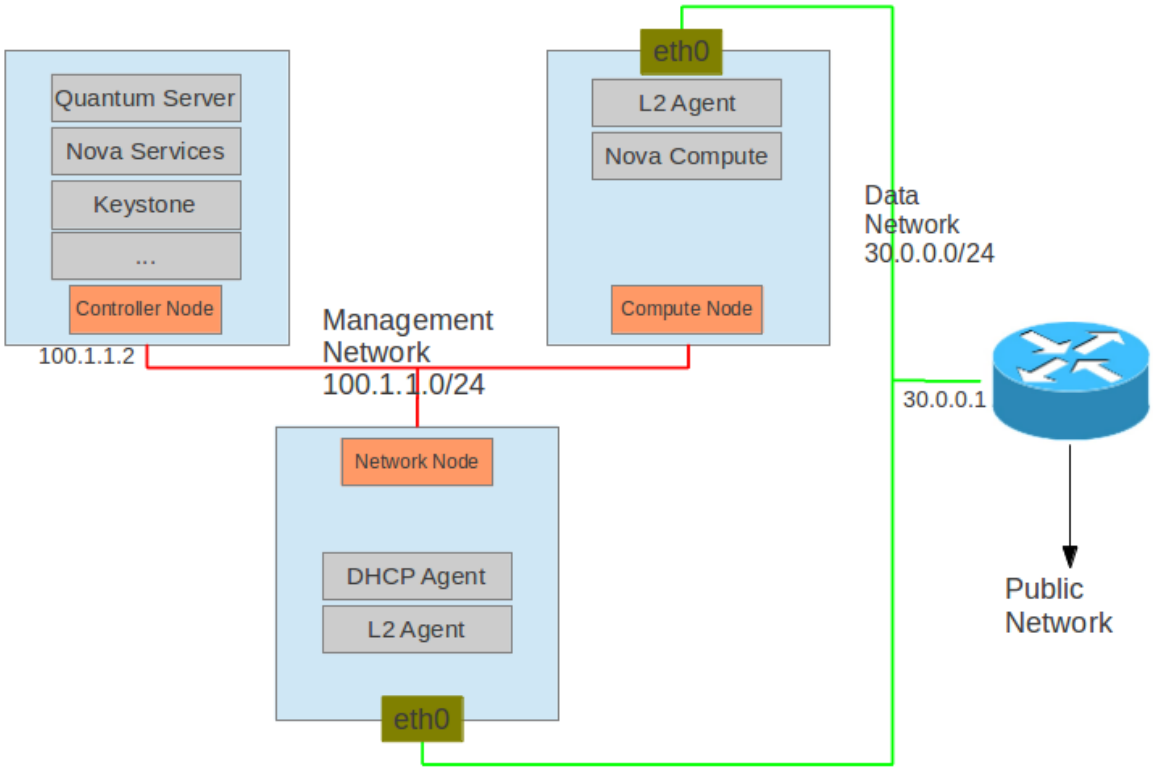
Use case: multiple flat network

Use case: mixed flat and private network

本节介绍如何安装OpenStack的网络服务及其组件，用于在单一的平面网络的使用情况。

如下图展示的配置。为了简单起见，所有的节点都应该有一个用于管理的接口和一个或多个用于虚拟机流量的接口。管理网络是100.1.1.0/24,控制节点使用100.1.1.2。该示例使用OpenvSwitch插件和代理。

#你可以修改这个配置以使用另外的插件和代理。



下面描述了一些节点的配置：

节点	描述
控制节点	运行网络服务、Identity服务和部署虚拟机所需的计算服务（例如：nova-api、nova-scheduler）。此节点至少有一个用于管理的网络接口。主机名是controller，其它节点都需要解析到控制节点的IP. #nova-network服务不应该运行，它被网络服务取代。要删除一个网络使用nova-manage network delete命令 #删除网络前必须先使用network-disassociate命令解除与项目的关联。
计算节点	运行L2网络代理和运行虚拟机的计算服务（nova-compute和其它依赖的nova服务）。此节点至少有两个网络接口，一个与控制器通信的管理网络，另外一个处理虚拟机的网络流量。虚拟机从DHCP代理获取IP地址
网络节点	运行L2网络代理和DHCP代理服务。DHCP代理为虚拟机网络分配IP地址。这个节点至少有两个网络接口，一个用于与控制节点通信，一个用于处理数据网络上的虚拟机流量。
路由	路由有个30.0.0.1的ip，它是所有虚拟机的默认网关。路由器必须能够访问公共网络。

此示例需要以下先决条件：
控制节点

1. 相关的计算服务已经安装、配置并运行。
2. Glance已经安装、配置并运行，且有可用的映像。
3. OpenStack Identity已经安装、配置并运行。网络用户neutron已经关联到租户service。
4. 附件服务：
   RabbitMQ以默认的账号密码运行。
   MySQL已运行（账号root，密码root）

计算节点

1. 计算服务已安装并配置。
2. 创建ovs_neutron数据库。
3. 更新网络配置文件 /etc/neutron/neutron.conf，

安装
控制节点与网络节点

1.安装网络服务器

2.创建ovs_neutron数据库

3.更新网络配置文件 /etc/neutron/neutron.conf，选择一个网络插件并配置一个必须的 Identity 服务用户。

```
[DEFAULT]
core_plugin =
neutron.plugins.openvswitch.ovs_neutron_plugin.OVSNeutronPluginV2
control_exchange = neutron
rabbit_host = controller
rabbit_password = RABBIT_PASS
notification_driver = neutron.openstack.common.notifier.rabbit_notifier

[database]
connection = mysql://neutron:NEUTRON_DBPASS@controller/neutron
[keystone_authtoken]
admin_tenant_name=service
admin_user=neutron
admin_password=NEUTRON_PASS
```

4.更新插件配置文件/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini：

```
[ovs]
network_vlan_ranges = physnet1
bridge_mappings = physnet1:br-eth0
```

5.启动网络服务

计算节点的计算

1.安装nova-compute服务。

2.更新计算配置文件/etc/nova/nova.conf。在最后添加以下内容

```
network_api_class=nova.network.neutronv2.api.API
neutron_admin_username=neutron
neutron_admin_password=NEUTRON_PASS
neutron_admin_auth_url=http://controller:35357/v2.0/
neutron_auth_strategy=keystone
neutron_admin_tenant_name=service
neutron_url=http://controller:9696/
```

3.重启计算服务

计算和网络节点 L2代理

1.安装并启动openvswitch

2.安装L2 代理（Neutron openvswitch代理）

3.为openvswitch增加一个集成网桥

```
ovs-vsctl add-br br-int
```

4.更新/etc/neutron/neutron.conf

```
[DEFAULT]
core_plugin =
neutron.plugins.openvswitch.ovs_neutron_plugin.OVSNeutronPluginV2
control_exchange = neutron
rabbit_host = controller
rabbit_password = RABBIT_PASS
notification_driver = neutron.openstack.common.notifier.rabbit_notifier
[database]
connection = mysql://neutron:NEUTRON_DBPASS@controller/neutron
```

5.更新 /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini

```
[ovs]
network_vlan_ranges = physnet1
```

bridge_mappings = physnet1:br-eth0

6.创建一个软连接，不然neutron-server 无法启动
ln -s /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini /etc/neutron/plugin.ini

7.创建一个br-eth0的网桥以使用eth0处理节点间的通信
ovs-vsctl add-br br-eth0
ovs-vsctl add-port br-eth0 eth0

8.启动L2代理

网络节点DHCP代理
1.安装DHCP代理
2.更新 /etc/neutron/neutron.conf
[DEFAULT]
core_plugin = neutron.plugins.openvswitch.ovs_neutron_plugin.OVSNeutronPluginV2
control_exchange = neutron
rabbit_host = controller
rabbit_password = RABBIT_PASS
notification_driver = neutron.openstack.common.notifier.rabbit_notifier

3.更新 /etc/neutron/dhcp_agent.ini
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver

4.启动DHCP代理

配置逻辑网络
在网络节点执行以下命令。
确保存在以下变量，所有客户端都使用以下变量访问身份服务。

```
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://controller:5000/v2.0/
```

1.获取租户ID(稍后将会用到租户ID)

```
keystone tenant-list
+----------------------------------+---------+---------+
|                id                |   name  | enabled |
+----------------------------------+---------+---------+
| 247e478c599f45b5bd297e8ddbbc9b6a | TenantA |   True  |
| 2b4fec24e62e4ff28a8445ad83150f9d | TenantC |   True  |
| 3719a4940bf24b5a8124b58c9b0a6ee6 | TenantB |   True  |
| 5fcfbc3283a142a5bb6978b549a511ac |   demo  |   True  |
| b7445f221cda4f4a8ac7db6b218b1339 |  admin  |   True  |
+----------------------------------+---------+---------+
```

2.获取用户信息

```
keystone user-list
+----------------------------------+-------+---------+-------------------+
|                id                |  name | enabled |       email       |
+----------------------------------+-------+---------+-------------------+
| 5a9149ed991744fa85f71e4aa92eb7ec |  demo |   True  |                   |
| 5b419c74980d46a1ab184e7571a8154e | admin |   True  | admin@example.com |
| 8e37cb8193cb4873a35802d257348431 | UserC |   True  |                   |
| c11f6b09ed3c45c09c21cbbc23e93066 | UserB |   True  |                   |
| ca567c4f6c0942bdac0e011e97bddbe3 | UserA |   True  |                   |
+----------------------------------+-------+---------+-------------------+
```

3.为demo租户创建一个内部共享网络（租户ID为 b7445f221cda4f4a8ac7db6b218b1339）

```
neutron net-create --tenant-id $TENANT_ID sharednet1 --shared --provider:network_type
```

```
flat \
  --provider:physical_network physnet1
Created a new network:
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 04457b44-e22a-4a5c-be54-a53a9b2818e7 |
| name                      | sharednet1                           |
| provider:network_type     | flat                                 |
| provider:physical_network | physnet1                             |
| provider:segmentation_id  |                                      |
| router:external           | False                                |
| shared                    | True                                 |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | b7445f221cda4f4a8ac7db6b218b1339     |
+---------------------------+--------------------------------------+
```

4.在网络上创建一个子网

```
neutron subnet-create --tenant-id $TENANT_ID sharednet1 30.0.0.0/24
Created a new subnet:
+-------------------+------------------------------------------------+
| Field             | Value                                          |
+-------------------+------------------------------------------------+
| allocation_pools  | {"start": "30.0.0.2", "end": "30.0.0.254"}     |
| cidr              | 30.0.0.0/24                                    |
| dns_nameservers   |                                                |
| enable_dhcp       | True                                           |
| gateway_ip        | 30.0.0.1                                       |
| host_routes       |                                                |
| id                | b8e9a88e-ded0-4e57-9474-e25fa87c5937           |
| ip_version        | 4                                              |
| name              |                                                |
| network_id        | 04457b44-e22a-4a5c-be54-a53a9b2818e7           |
| tenant_id         | 5fcfbc3283a142a5bb6978b549a511ac               |
+-------------------+------------------------------------------------+
```

5.为 tenantA创建一个服务器

```
nova --os-tenant-name TenantA --os-username UserA --os-password password \
  --os-auth-url=http://localhost:5000/v2.0 boot --image tty --flavor 1 \
  --nic net-id=04457b44-e22a-4a5c-be54-a53a9b2818e7 TenantA_VM1

nova --os-tenant-name TenantA --os-username UserA --os-password password \
--os-auth-url=http://localhost:5000/v2.0  list
+--------------------------------------+-------------+--------+---------------------+
| ID                                   | Name        | Status | Networks            |
+--------------------------------------+-------------+--------+---------------------+
| 09923b39-050d-4400-99c7-e4b021cdc7c4 | TenantA_VM1 | ACTIVE | sharednet1=30.0.0.3 |
+--------------------------------------+-------------+--------+---------------------+
```

6.ping租户TenantA 的服务器

```
ip addr flush eth0
ip addr add 30.0.0.201/24 dev br-eth0
ping 30.0.0.3
```

7.在租户TenantA 的服务器中ping公共网络

```
ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_req=1 ttl=64 time=1.74 ms
64 bytes from 192.168.1.1: icmp_req=2 ttl=64 time=1.50 ms
64 bytes from 192.168.1.1: icmp_req=3 ttl=64 time=1.23 ms
^C
```

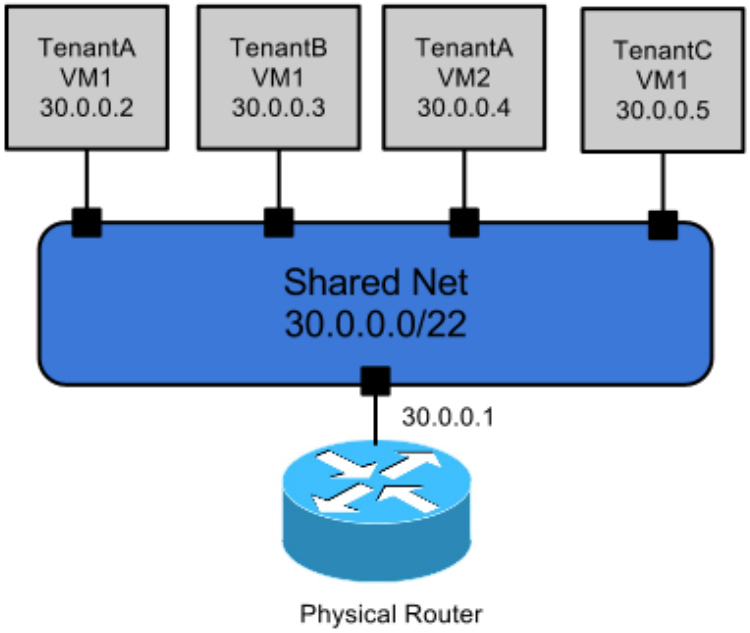
```
--- 192.168.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.234/1.495/1.745/0.211 ms
```

#192.168.1.1是路由器连接公共网络端口的IP

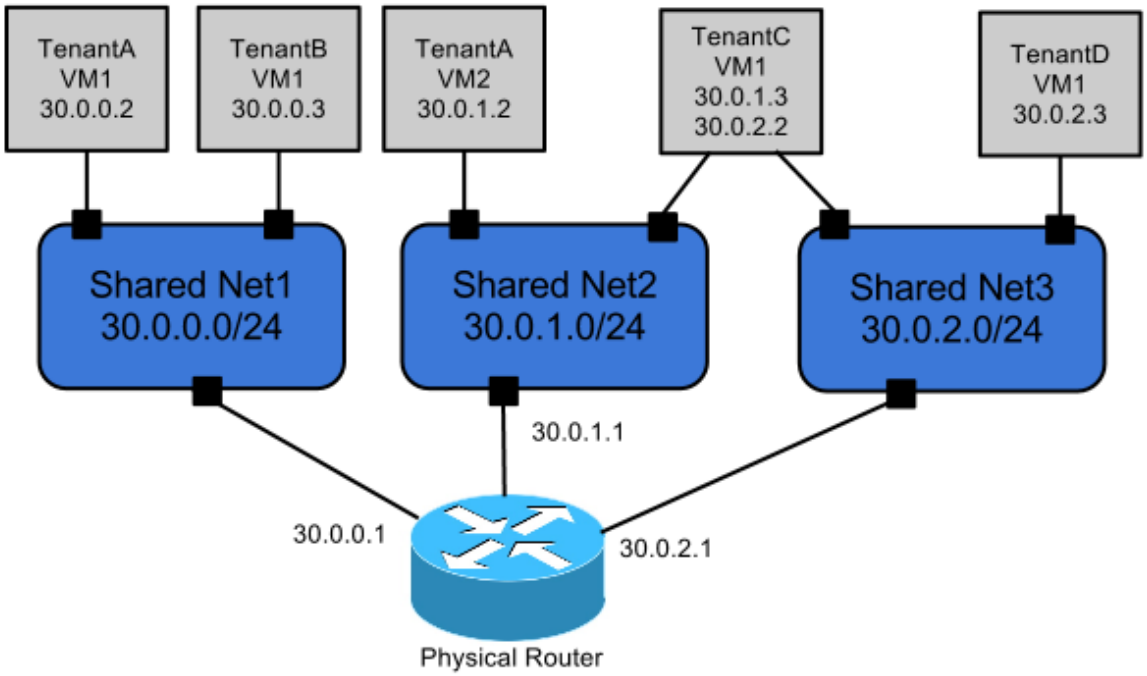
8.用类似的命令为其它租户创建服务器。因为所有虚拟机共享同一子网，所以它们可以互相访问。

使用示例：单一平面网络
最简单的用例是单一网络。这是个共享网络，意思是它通过网络API对所有租户可见。租户虚拟机只有一个网卡，从与该网络相关的子网获取一个固定的地址。这个用例基本上对应由计算提供的FlatManager和FlatDHCPManager模型。不支持浮动IP地址。

这种网络类型通常由openStack管理员创建直接映射到数据中心现有的物理网络（称为运营商网络）。这需要运营商的数据中心有一个物理路由器作为虚拟机的网关连接到外网。对于外部网络的每个子网，都必须在OpenStack之外手动配置物理路由器的网关。

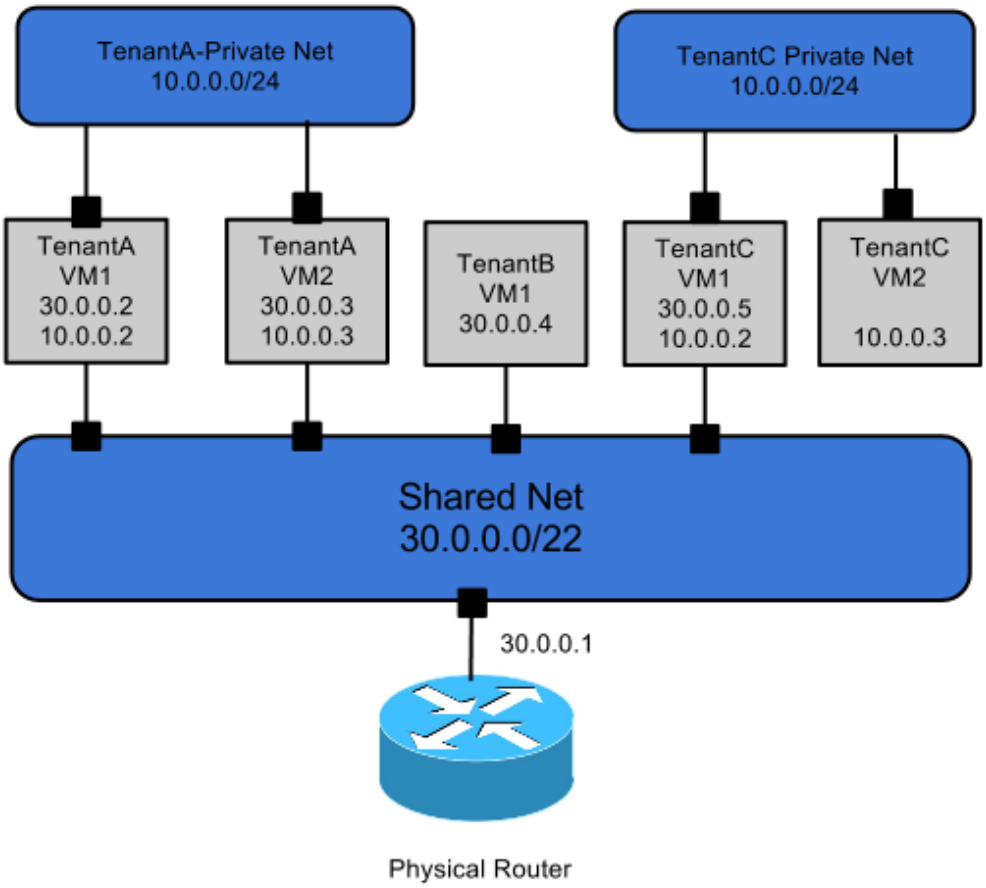


使用示例：多节点平面网络
这个用例类似上面的单一平面网络，除了租户可以通过网络API可见多个共享网络和可以选择接入哪个（或哪几个）网络。



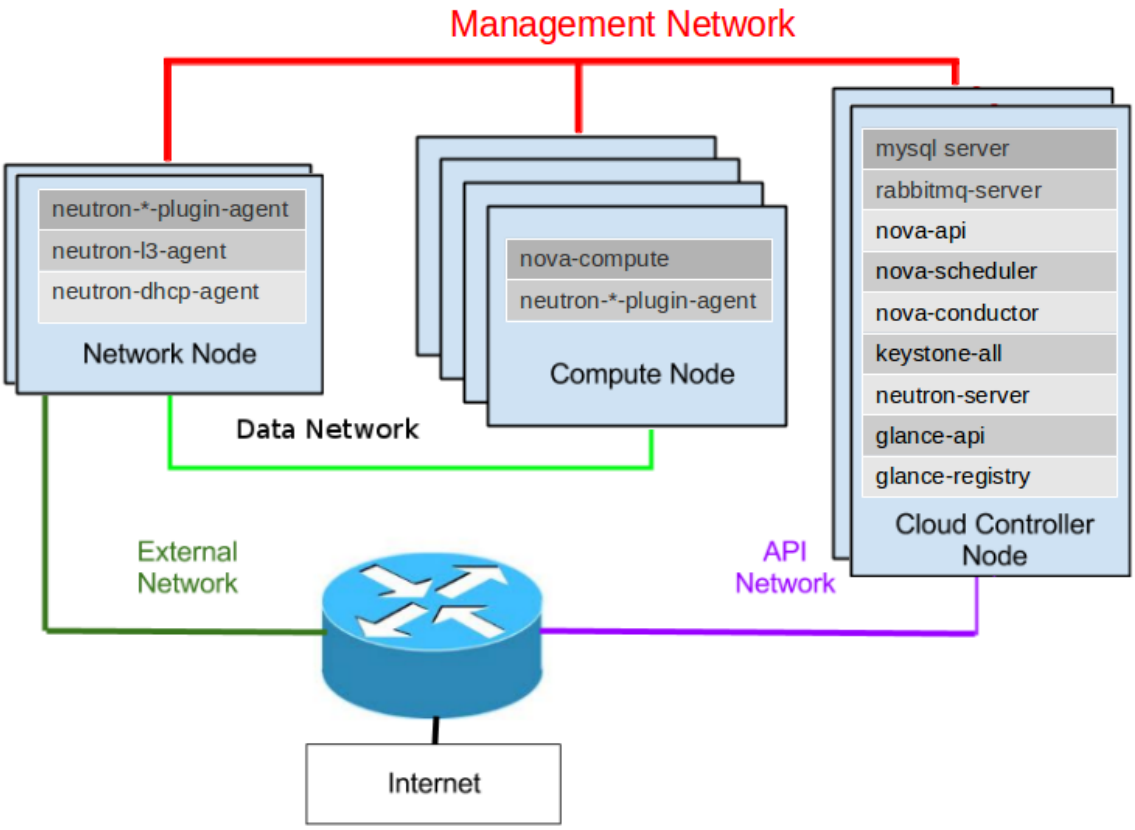
使用示例：混合平面网络和私有网络
这个用例是上面平面网络的扩展用例，除了能通过OpenStack网络API可见的一个或多个共享网络外租户也可以配置基本的私有网络（仅对租户的用户可见）。

创建的虚拟机可以拥有共享或私有网络的多块网卡。这允许虚拟机拥有多块网卡创建多层拓扑结构。这使得虚拟机可以作为网关，以提供服务。如路由、NAT以及负载均衡



为私有网络提供路由器
[Install]
[Logical network configuration]
[Use case: provider router with private networks]

本节介绍如何安装OpenStack的网络服务及其组件，用于单一路由器的用例：为私有网络提供一个路由器
下图展示的配置：



#因为在一个节点运行DHCP和L3代理，所以你必须将两个配置文件中的use_namespaces设置为True（默认值）。
这些节点的配置包括：
表9.1

节点	描述
控制器	运行网络服务、身份服务以及部署虚拟机所需的计算服务。 这个服务至少有两个网络接口。第一个接口连接用于计算和网络节点的通信的管理网络。第二个接口连接API或公共网络。
计算	运行计算和L2代理。 该节点没有接入公共网络。 该节点至少有一个网络接口，用于通过管理网络与控制节点通信。虚拟机从这个网络的DHCP代理接收IP地址。
网络	运行L2、DHCP、L3代理。 这个节点可以访问公共网络。DHCP代理在网络上为虚拟机分配IP地址。L3代理完成NAT和允许虚拟机访问公网。 这个节点有： 一个通过管理网络与控制节点通信的网络接口 一个在数据网络上管理虚拟机流量的接口 一个连接外部网络网关的接口

安装
Controller
Network node
Compute Node

控制节点
安装和配置控制器节点
1.运行此命令

```
yum install openstack-neutron
```

2.配置网络服务

编辑/etc/neutron/neutron.conf，增加这些行：

```
core_plugin = neutron.plugins.openvswitch.ovs_neutron_plugin.OVSNeutronPluginV2
auth_strategy = keystone
fake_rabbit = False
rabbit_password = RABBIT_PASS
[database]
connection = mysql://neutron:NEUTRON_DBPASS@controller/neutron
```

编辑/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini：

```
[ovs]
tenant_network_type = vlan
network_vlan_ranges = physnet1:100:2999
```

编辑 /etc/neutron/api-paste.ini：

```
admin_tenant_name = service
admin_user = neutron
admin_password = NEUTRON_PASS
```

3.启动服务

```
service neutron-server restart
```

网络节点

安装和配置网络节点

1.安装包

```
yum install openstack-neutron-openvswitch \
openstack-neutron
```

2.启动OpenvSwitch

```
service openvswitch start
chkconfig openvswitch on
```

3.添加一个集成网桥

```
ovs-vsctl add-br br-int
```

4.更新 /etc/neutron/neutron.conf

```
openstack-config --set /etc/neutron/neutron.conf \
  DEFAULT qpid_hostname controller
openstack-config --set /etc/neutron/neutron.conf \
   database connection mysql://neutron:NEUTRON_DBPASS@controller:3306/neutron
```

5.更新 /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini

```
[ovs]
tenant_network_type=vlan
network_vlan_ranges = physnet1:1:4094
bridge_mappings = physnet1:br-eth1
```

6.创建网桥br-eth1，所有虚拟机间的通信都通过br-eth1

```
ovs-vsctl add-br br-eth1
ovs-vsctl add-port br-eth1 eth1
```

7.创建外部网桥

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

8.编辑 /etc/neutron/l3_agent.ini ，增加

```
[DEFAULT]
auth_url = http://controller:35357/v2.0
admin_tenant_name = service
admin_user = neutron
admin_password = NEUTRON_PASS
metadata_ip = controller
use_namespaces = True
```

9.编辑 /etc/neutron/api-paste.ini，增加
```
[DEFAULT]
auth_host = controller
admin_tenant_name = service
admin_user = neutron
admin_password = NEUTRON_PASS
```

10.编辑/etc/neutron/dhcp_agent.ini，增加
```
use_namespaces = True
```

11.启动网络服务
```
service neutron-openvswitch-agent start
service neutron-dhcp-agent start
service neutron-l3-agent start
chkconfig neutron-openvswitch-agent on
chkconfig neutron-dhcp-agent on
chkconfig neutron-l3-agent on
```

12.启动neutron-ovs-cleanup服务，这个服务需要开机启动并确保网络有对创建和管理tap设备有完全的控制权。
```
chkconfig neutron-ovs-cleanup on
```

计算节点
安装和配置计算节点
1.安装包
```
yum install openstack-neutron-openvswitch
```

2.启动openvSwitch
```
service openvswitch start
chkconfig openvswitch on
```

3.创建集成网桥
```
ovs-vsctl add-br br-int
```

4.创建一个br-eth1的网桥，所有虚拟机之间的通信都通过此网桥
```
ovs-vsctl add-br br-eth1
ovs-vsctl add-port br-eth1 eth1
```

5.编辑/etc/neutron/neutron.conf，增加
```
openstack-config --set /etc/neutron/neutron.conf \
  DEFAULT qpid_hostname controller
openstack-config --set /etc/neutron/neutron.conf \
   database connection mysql://neutron:NEUTRON_DBPASS@controller:3306/neutron
```

6.编辑/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini，增加
```
[ovs]
tenant_network_type = vlan
network_vlan_ranges = physnet1:1:4094
bridge_mappings = physnet1:br-eth1
```

7.启动网络服务
```
service neutron-openvswitch-agent start
chkconfig neutron-openvswitch-agent on
```

逻辑网络配置
在网络节点上运行这些命令
确保以下这些环境变量已经设置，不同的客户端使用这些变量访问身份服务。
创建一个novarc文件
```
export OS_TENANT_NAME=provider_tenant
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
```

```
export OS_AUTH_URL="http://controller:5000/v2.0/"
export OS_SERVICE_ENDPOINT="http://controller:35357/v2.0"
export OS_SERVICE_TOKEN=password
```

声明变量

```
source novarc; echo "source novarc">>.bashrc
```

admin用户代表tenant_A创建了网络和子网，tenant_A的用户也可以完成这些步骤。

配置内部网络

1.获取租户ID(稍后会用到租户ID)

```
keystone tenant-list
+----------------------------------+--------------------+---------+
|                id                |        name        | enabled |
+----------------------------------+--------------------+---------+
| 48fb81ab2f6b409bafac8961a594980f |  provider_tenant   |   True  |
| cbb574ac1e654a0a992bfc0554237abf |      service       |   True  |
| e371436fe2854ed89cca6c33ae7a83cd | invisible_to_admin |   True  |
| e40fa60181524f9f9ee7aa1038748f08 |      tenant_A      |   True  |
+----------------------------------+--------------------+---------+
```

2.为租户tenant_A创建一个网络net1(租户id为 e40fa60181524f9f9ee7aa1038748f08)

```
neutron net-create --tenant-id $TENANT_ID net1
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | e99a361c-0af8-4163-9feb-8554d4c37e4f |
| name                      | net1                                 |
| provider:network_type     | vlan                                 |
| provider:physical_network | physnet1                             |
| provider:segmentation_id  | 1024                                 |
| router:external           | False                                |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | e40fa60181524f9f9ee7aa1038748f08     |
+---------------------------+--------------------------------------+
```

3.为net1创建一个子网（稍后将会用到子网ID）

```
neutron subnet-create --tenant-id $TENANT_ID net1 10.5.5.0/24
+-------------------+------------------------------------------------+
| Field             | Value                                          |
+-------------------+------------------------------------------------+
| allocation_pools  | {"start": "10.5.5.2", "end": "10.5.5.254"}     |
| cidr              | 10.5.5.0/24                                    |
| dns_nameservers   |                                                |
| enable_dhcp       | True                                           |
| gateway_ip        | 10.5.5.1                                       |
| host_routes       |                                                |
| id                | c395cb5d-ba03-41ee-8a12-7e792d51a167           |
| ip_version        | 4                                              |
| name              |                                                |
| network_id        | e99a361c-0af8-4163-9feb-8554d4c37e4f           |
| tenant_id         | e40fa60181524f9f9ee7aa1038748f08               |
+-------------------+------------------------------------------------+
```

用户必须以管理员角色完成这些步骤，这个过程中使用的admin用户来自provider_tenant。

配置一个路由和扩展网络

1.创建一个路由router1 （稍后将用到路由ID）

```
neutron router-create router1
```

```
+-----------------------+--------------------------------------+
| Field                 | Value                                |
+-----------------------+--------------------------------------+
| admin_state_up        | True                                 |
| external_gateway_info |                                      |
| id                    | 685f64e7-a020-4fdf-a8ad-e41194ae124b |
| name                  | router1                              |
| status                | ACTIVE                               |
| tenant_id             | 48fb81ab2f6b409bafac8961a594980f     |
+-----------------------+--------------------------------------+
```

这里没有使用--tenant-id指定租户id，所以这个路由创建到了租户provider_tenant。

2.为路由 router1 和网络net1 附属的子网创建一个接口

```
neutron router-interface-add $ROUTER_ID $SUBNET_ID
Added interface to router 685f64e7-a020-4fdf-a8ad-e41194ae124b
```

可以为其他租户的其它网络重复此步骤创建更多接口

3.创建一个 ext_net的外部网络

```
neutron net-create ext_net --router:external=True
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 8858732b-0400-41f6-8e5c-25590e67ffeb |
| name                      | ext_net                              |
| provider:network_type     | vlan                                 |
| provider:physical_network | physnet1                             |
| provider:segmentation_id  | 1                                    |
| router:external           | True                                 |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | 48fb81ab2f6b409bafac8961a594980f     |
+---------------------------+--------------------------------------+
```

4.创建一个可以浮动ip的子网

```
#这个子网禁用DHCP
neutron subnet-create ext_net \
--allocation-pool start=7.7.7.130,end=7.7.7.150 \
--gateway 7.7.7.1 7.7.7.0/24 --disable-dhcp
+-------------------+------------------------------------------------------+
| Field             | Value                                                |
+-------------------+------------------------------------------------------+
| allocation_pools  | {"start": "7.7.7.130", "end": "7.7.7.150"}           |
| cidr              | 7.7.7.0/24                                           |
| dns_nameservers   |                                                      |
| enable_dhcp       | False                                                |
| gateway_ip        | 7.7.7.1                                              |
| host_routes       |                                                      |
| id                | aef60b55-cbff-405d-a81d-406283ac6cff                 |
| ip_version        | 4                                                    |
| name              |                                                      |
| network_id        | 8858732b-0400-41f6-8e5c-25590e67ffeb                 |
| tenant_id         | 48fb81ab2f6b409bafac8961a594980f                     |
+-------------------+------------------------------------------------------+
```

5.设置路由器到外部网络的网关

```
neutron router-gateway-set $ROUTER_ID $EXTERNAL_NETWORK_ID
Set gateway for router 685f64e7-a020-4fdf-a8ad-e41194ae124b
```

用户从tenant_A完成了这些操作，所以环境变量的认证与之前的步骤是不同的。

分配浮动IP
1.你可以在虚拟机启动之后分配一个浮动ip。找到分配给虚拟机的端口ID($PORT_ID),如下：

```
nova list
+--------------------------------------+--------+--------+---------------+
|                  ID                  |  Name  | Status |    Networks   |
+--------------------------------------+--------+--------+---------------+
| 1cdc671d-a296-4476-9a75-f9ca1d92fd26 | testvm | ACTIVE | net1=10.5.5.3 |
+--------------------------------------+--------+--------+---------------+

neutron port-list -- --device_id 1cdc671d-a296-4476-9a75-f9ca1d92fd26
+--------------------------------------+------+-------------------+-----------------
-------------------------------------------------------------+
| id                                   | name | mac_address       | fixed_ips
|
+--------------------------------------+------+-------------------+-----------------
-------------------------------------------------------------+
| 9aa47099-b87b-488c-8c1d-32f993626a30 |      | fa:16:3e:b4:d6:6c | {"subnet_id":
"c395cb5d-ba03-41ee-8a12-7e792d51a167", "ip_address": "10.5.5.3"} |
+--------------------------------------+------+-------------------+-----------------
-------------------------------------------------------------+
```

2.分配一个浮动IP（使用$FLOATING_ID）

```
neutron floatingip-create ext_net
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| fixed_ip_address    |                                      |
| floating_ip_address | 7.7.7.131                            |
| floating_network_id | 8858732b-0400-41f6-8e5c-25590e67ffeb |
| id                  | 40952c83-2541-4d0c-b58e-812c835079a5 |
| port_id             |                                      |
| router_id           |                                      |
| tenant_id           | e40fa60181524f9f9ee7aa1038748f08     |
+---------------------+--------------------------------------+
```

3.为虚拟机的端口分配浮动ip

```
neutron floatingip-associate $FLOATING_ID $PORT_ID
Associated floatingip 40952c83-2541-4d0c-b58e-812c835079a5
```

4.显示浮动ip

```
neutron floatingip-show $FLOATING_ID
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| fixed_ip_address    | 10.5.5.3                             |
| floating_ip_address | 7.7.7.131                            |
| floating_network_id | 8858732b-0400-41f6-8e5c-25590e67ffeb |
| id                  | 40952c83-2541-4d0c-b58e-812c835079a5 |
| port_id             | 9aa47099-b87b-488c-8c1d-32f993626a30 |
| router_id           | 685f64e7-a020-4fdf-a8ad-e41194ae124b |
| tenant_id           | e40fa60181524f9f9ee7aa1038748f08     |
+---------------------+--------------------------------------+
```

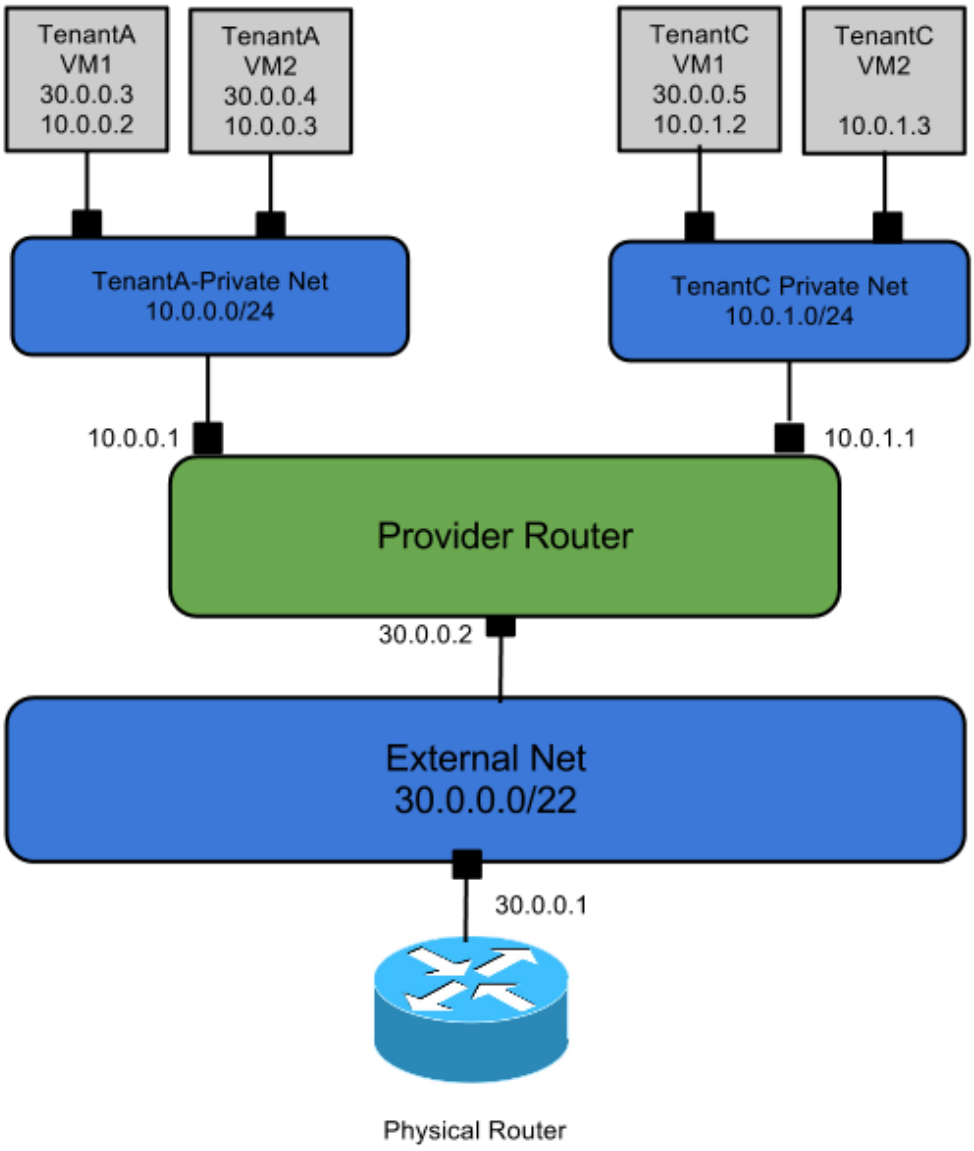
5.测试浮动IP

```
ping 7.7.7.131
PING 7.7.7.131 (7.7.7.131) 56(84) bytes of data.
64 bytes from 7.7.7.131: icmp_req=2 ttl=64 time=0.152 ms
64 bytes from 7.7.7.131: icmp_req=3 ttl=64 time=0.049 ms
```

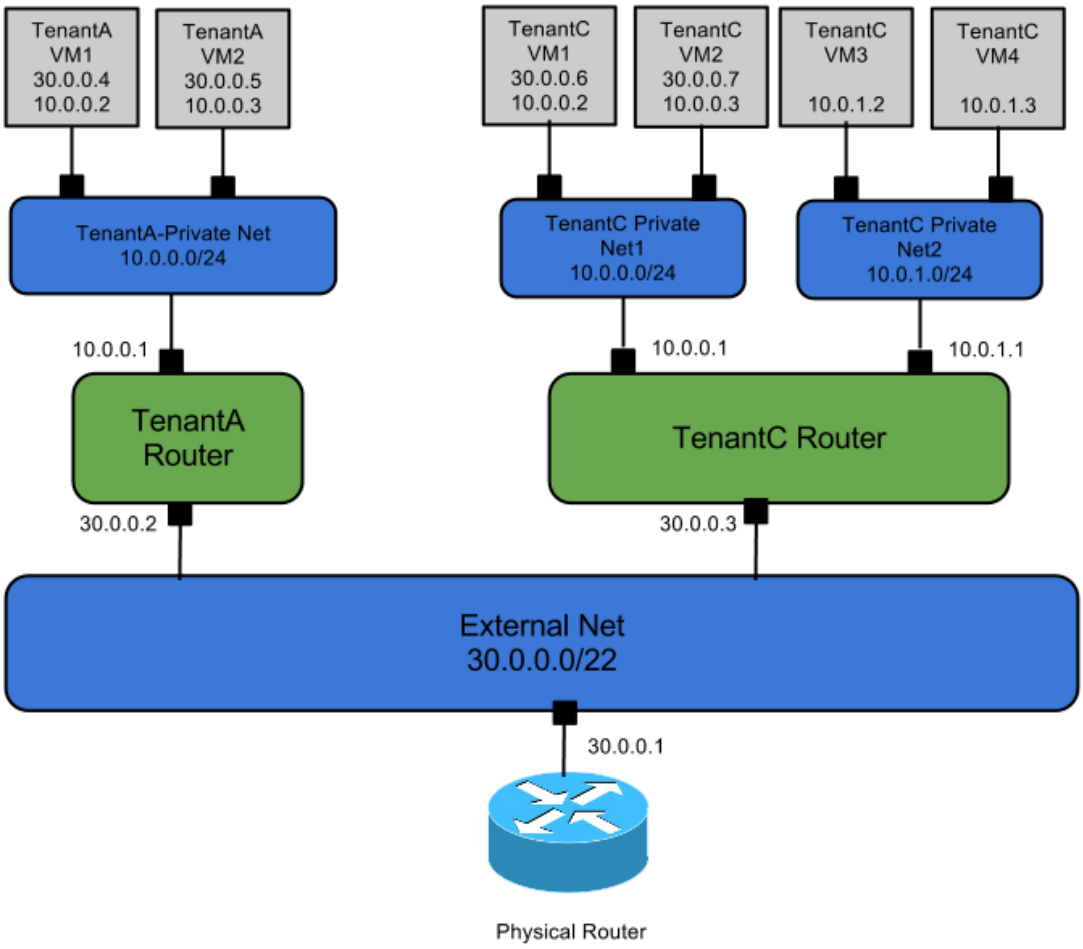
使用示例：为私有网络提供路由器
这个用例为每个租户提供一个或多个私有网络通过OpenStack网络路由器连接到外网。当租户得到一个网络时，这种架构用计算中的VlanManager映射到相同的逻辑网络拓扑（尽管网络不需要vlan）。租户只能使用网络API看到分配给每个租户的私有网络。API中创建的路由器对象归云管理员所有。

这个模型自支持使用浮动IP将公网地址分配给虚拟机；路由器从外部网络将公网地址映射到私有网络的固定IP地址。主机没有浮动ip仍然可以创建到外部网络的出站连接，因为路由器执行SNAT到路由器的外部IP。物理路由器的ip用作外部网络的网关，所以运营商有一个公网的默认路由。

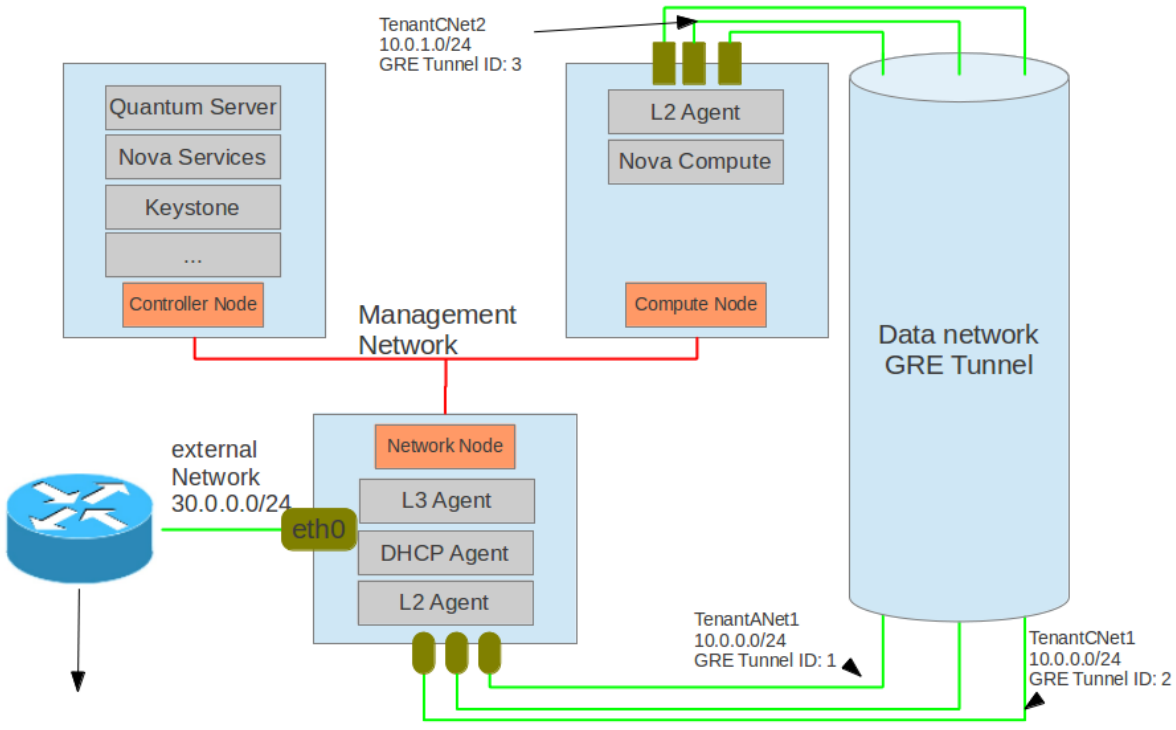
这个路由器提供私有网络间的L3连接。租户可以连接其它租户的实例，除非使用额外的过滤器（例如安全组）。在使用单一的路由器时，租户网络的IP地址不能复用。为解决这个问题，管理员可以代表租户创建私有网络。



私有网络的每租户路由器
Install
Configure logical network
Use case: per-tenant routers with private networks
本节介绍如何为有租户路由的私有网络用例安装OpenStack网络服务及其组件。



它使用下图的配置



如图所示的安装包括：

每个节点有一个用于管理流量的接口
使用openvswitch插件
所有代理使用GRE隧道传输数据
在外部网络上已配置了浮动IP和路由器网关，以及连接浮动IP和路由器网关端口的外部物理路由器。

#因为这个示例在一个节点运行DHCP代理和L3代理，你必须将所有代理的use_namespace配置为True，默认值为True。
下表是节点描述

节点	描述
控制器节点	运行网络、身份验证及部署虚拟机所需的所有计算服务（如nova-api,nova-scheduler）。这个节点至少有一个连接到管理网络的接口。主机名为controlnode，其它节点必须解析它的IP。 #nova-network必须停止运行，它被网络服务取代。
计算节点	运行L2网络代理和运行虚拟机的计算服务（nova-compute和依赖可选的其它nova服务）。节点至少有两个网络接口，一个连接与控制节点通信的管理网络，一个连接用于虚拟机通信的数据网络，虚拟机在这个网络通过dhcp代理获取IP地址
网络节点	运行L2网络代理、dhcp代理和L3代理。这个节点可以访问外网。DHCP代理为数据网络上的虚拟机分配IP（从技术上说，地址由网络服务器分配，并由DHCP代理分发）。这个节点至少拥有两个网络接口。一个连接与控制节点通信的管理网络，一个连接外部网络。数据网络设置为GRE隧道。
路由器	路由器有一个30.0.0.1的ip，这是所有虚拟机的默认网关。这个路由器必须可以访问外部网络。

此用例假设一下配置：
控制器节点
1.相关计算服务安装并运行
2.Glance已安装并运行，此外必须有一个名为tty的映像。
3.身份服务已安装并运行，有一个名为neutron，关联租户为service使用NEUTRON_PASS密码的用户。
4.额外的服务：
以默认用户Guest和密码运行的RabbitMQ
MySQL server（用户名密码为root.root）

计算节点
已安装和配置计算

安装
控制器节点的网络服务
1.安装网络服务器
2.创建数据库ovs_neutron
3.更新网络配置文件/etc/neutron/neutron.conf，选择网络插件和配置身份服务用户

```
[DEFAULT]
core_plugin = neutron.plugins.openvswitch.ovs_neutron_plugin.OVSNeutronPluginV2
control_exchange = neutron
rabbit_host = controller
rabbit_password = RABBIT_PASS
notification_driver = neutron.openstack.common.notifier.rabbit_notifier
[database]
connection = mysql://neutron:NEUTRON_DBPASS@controller:3306/neutron
[keystone_authtoken]
admin_tenant_name=service
admin_user=neutron
admin_password=NEUTRON_PASS
```

4.更新插件配置文件 /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini

```
[ovs]
tenant_network_type = gre
tunnel_id_ranges = 1:1000
enable_tunneling = True
```

5.启动网络服务
网络服务可以作为一个操作系统服务。启动服务的命令依赖你的操作系统。以下命令直接启动网络服务：

```
neutron-server --config-file /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini
\
  --config-file /etc/neutron/neutron.conf
```

计算节点配置
1.安装计算服务
2.更新计算服务配置文件/etc/nova/nova.conf，确保文件结尾有以下行：

```
network_api_class=nova.network.neutronv2.api.API
neutron_admin_username=neutron
neutron_admin_password=NEUTRON_PASS
neutron_admin_auth_url=http://controlnode:35357/v2.0/
neutron_auth_strategy=keystone
neutron_admin_tenant_name=service
neutron_url=http://controlnode:9696/
```

3.重启相关计算服务

计算和网络节点的L2代理
1.安装openvswitch
2.安装L2代理（Neutron openvswitch agent）
3.新建一个集成网桥

```
ovs-vsctl add-br br-int
```

4.更新网络配置文件 /etc/neutron/neutron.conf

```
[DEFAULT]
core_plugin = neutron.plugins.openvswitch.ovs_neutron_plugin.OVSNeutronPluginV2
control_exchange = neutron
rabbit_host = controller
rabbit_password = RABBIT_PASS
notification_driver = neutron.openstack.common.notifier.rabbit_notifier
[database]
connection = mysql://neutron:NEUTRON_DBPASS@controller:3306/neutron
```

5.更新插件配置文件 /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini
计算节点：

```
[ovs]
tenant_network_type = gre
tunnel_id_ranges = 1:1000
enable_tunneling = True
local_ip = 9.181.89.202
```

网络节点：

```
[ovs]
tenant_network_type = gre
tunnel_id_ranges = 1:1000
enable_tunneling = True
local_ip = 9.181.89.203
```

6.创建集成网桥br-int

```
ovs-vsctl --may-exist add-br br-int
```

7.启动L2代理
网络的openvswitch L2 代理可以作为一个操作系统服务。启动命令依赖你的操作系统，下面的命令直接启动服务：

```
neutron-openvswitch-agent --config-file
```

```
/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini \
  --config-file /etc/neutron/neutron.conf
```

网络节点DHCP代理
1.安装DHCP代理
2.更新网络配置文件 /etc/neutron/neutron.conf

```
[DEFAULT]
core_plugin = neutron.plugins.openvswitch.ovs_neutron_plugin.OVSNeutronPluginV2
control_exchange = neutron
rabbit_host = controller
rabbit_password = RABBIT_PASS
notification_driver = neutron.openstack.common.notifier.rabbit_notifier
allow_overlapping_ips = True
```

设置 allow_overlapping_ips是因为TenantA 和TenantC使用重叠的子网。

3.更新/etc/neutron/dhcp_agent.ini

```
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
```

4.启动DHCP代理
网络的DHCP代理可以作为一个操作系统服务，启动命令依赖你的操作系统，下面的命令直接启动服务：

```
neutron-dhcp-agent --config-file /etc/neutron/neutron.conf \
  --config-file /etc/neutron/dhcp_agent.ini
```

网络节点L3代理
1.安装L3代理
2.新建一个外部网桥

```
ovs-vsctl add-br br-ex
```

3.添加一个物理接口到外部网桥，去往外部网络的连接将通过此网桥：

```
ovs-vsctl add-port br-ex eth0
```

4.更新L3配置文件/etc/neutron/l3_agent.ini

```
[DEFAULT]
interface_driver=neutron.agent.linux.interface.OVSInterfaceDriver
use_namespaces=True
```

设置use_namespaces选项（默认为True）是因为TenantA和TenantC使用重叠的子网以及路由器都托管在一个L3代理网络节点

5.启动L3代理
L3代理可以作为一个操作系统服务，启动命令依赖于你的操作系统，下面的命令直接启动服务：

```
neutron-l3-agent --config-file /etc/neutron/neutron.conf \
--config-file /etc/neutron/l3_agent.ini
```

配置逻辑网络
这些命令在网络节点运行
确保配置了下面这些变量，其它客户端通过这些访问身份服务

```
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://controller:5000/v2.0/
```

1.获取租户ID（稍后将会用到）

```
keystone tenant-list
+----------------------------------+---------+---------+
|                id                |   name  | enabled |
+----------------------------------+---------+---------+
| 247e478c599f45b5bd297e8ddbbc9b6a | TenantA |   True  |
| 2b4fec24e62e4ff28a8445ad83150f9d | TenantC |   True  |
| 3719a4940bf24b5a8124b58c9b0a6ee6 | TenantB |   True  |
| 5fcfbc3283a142a5bb6978b549a511ac |   demo  |   True  |
| b7445f221cda4f4a8ac7db6b218b1339 |  admin  |   True  |
+----------------------------------+---------+---------+
```

2.获取用户信息
keystone user-list
```
+----------------------------------+-------+---------+-------------------+
|                id                |  name | enabled |       email       |
+----------------------------------+-------+---------+-------------------+
| 5a9149ed991744fa85f71e4aa92eb7ec |  demo |   True  |                   |
| 5b419c74980d46a1ab184e7571a8154e | admin |   True  | admin@example.com |
| 8e37cb8193cb4873a35802d257348431 | UserC |   True  |                   |
| c11f6b09ed3c45c09c21cbbc23e93066 | UserB |   True  |                   |
| ca567c4f6c0942bdac0e011e97bddbe3 | UserA |   True  |                   |
+----------------------------------+-------+---------+-------------------+
```

3.使用admin用户创建外部网络和它的子网
neutron net-create Ext-Net --provider:network_type local --router:external true
Created a new network:
```
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 2c757c9e-d3d6-4154-9a77-336eb99bd573 |
| name                      | Ext-Net                              |
| provider:network_type     | local                                |
| provider:physical_network |                                      |
| provider:segmentation_id  |                                      |
| router:external           | True                                 |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | b7445f221cda4f4a8ac7db6b218b1339     |
+---------------------------+--------------------------------------+
```

neutron subnet-create Ext-Net 30.0.0.0/24 --disable-dhcp
Created a new subnet:
```
+-------------------+------------------------------------------------+
| Field             | Value                                          |
+-------------------+------------------------------------------------+
| allocation_pools | {"start": "30.0.0.2", "end": "30.0.0.254"}      |
| cidr              | 30.0.0.0/24                                    |
| dns_nameservers   |                                                |
| enable_dhcp       | False                                          |
| gateway_ip        | 30.0.0.1                                       |
| host_routes       |                                                |
| id                | ba754a55-7ce8-46bb-8d97-aa83f4ffa5f9           |
| ip_version        | 4                                              |
| name              |                                                |
| network_id        | 2c757c9e-d3d6-4154-9a77-336eb99bd573           |
| tenant_id         | b7445f221cda4f4a8ac7db6b218b1339               |
+-------------------+------------------------------------------------+
```

provider:network_type local 意思是没有通过运营商网络实现这个网络。router:external true意思是外部网络创建后你可以创建浮动IP和路由器网关端口。

4.添加一个外部网络ip到br-ex。
br-ex是外部网络网桥，为br-ex添加一个30.0.0.100/24并且从网络节点ping虚拟机额浮动ip
```
ip addr add 30.0.0.100/24 dev br-ex
ip link set br-ex up
```

5.服务TenantA
为TenantA创建一个私有网络，子网，服务路由及浮动IP
a.创建网络
```
neutron --os-tenant-name TenantA --os-username UserA --os-password password \
 --os-auth-url=http://localhost:5000/v2.0 net-create TenantA-Net
Created a new network:
```

```
+-----------------+--------------------------------------+
| Field           | Value                                |
+-----------------+--------------------------------------+
| admin_state_up  | True                                 |
| id              | 7d0e8d5d-c63c-4f13-a117-4dc4e33e7d68 |
| name            | TenantA-Net                          |
| router:external | False                                |
| shared          | False                                |
| status          | ACTIVE                               |
| subnets         |                                      |
| tenant_id       | 247e478c599f45b5bd297e8ddbbc9b6a     |
+-----------------+--------------------------------------+
```

然后你可以使用admin用户查询提供的网络信息

```
neutron net-show TenantA-Net
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 7d0e8d5d-c63c-4f13-a117-4dc4e33e7d68 |
| name                      | TenantA-Net                          |
| provider:network_type     | gre                                  |
| provider:physical_network |                                      |
| provider:segmentation_id  | 1                                    |
| router:external           | False                                |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | 247e478c599f45b5bd297e8ddbbc9b6a     |
+---------------------------+--------------------------------------+
```

这个网络有一个GRE隧道ID（例如provider:segmentation_id）1.

b.在 TenantA-Net网络上创建一个子网

```
neutron --os-tenant-name TenantA --os-username UserA --os-password password \
  --os-auth-url=http://localhost:5000/v2.0 subnet-create TenantA-Net 10.0.0.0/24
Created a new subnet:
+-------------------+--------------------------------------------+
| Field             | Value                                      |
+-------------------+--------------------------------------------+
| allocation_pools  | {"start": "10.0.0.2", "end": "10.0.0.254"} |
| cidr              | 10.0.0.0/24                                |
| dns_nameservers   |                                            |
| enable_dhcp       | True                                       |
| gateway_ip        | 10.0.0.1                                   |
| host_routes       |                                            |
| id                | 51e2c223-0492-4385-b6e9-83d4e6d10657       |
| ip_version        | 4                                          |
| name              |                                            |
| network_id        | 7d0e8d5d-c63c-4f13-a117-4dc4e33e7d68       |
| tenant_id         | 247e478c599f45b5bd297e8ddbbc9b6a           |
+-------------------+--------------------------------------------+
```

c.为租户TenantA创建一个服务器

```
nova --os-tenant-name TenantA --os-username UserA --os-password password \
  --os-auth-url=http://localhost:5000/v2.0 boot --image tty --flavor 1 \
  --nic net-id=7d0e8d5d-c63c-4f13-a117-4dc4e33e7d68 TenantA_VM1

nova --os-tenant-name TenantA --os-username UserA --os-password password \
   --os-auth-url=http://localhost:5000/v2.0 list
+--------------------------------------+-------------+--------+----------------------
+
| ID                                   | Name        | Status | Networks
|
```

```
+--------------------------------------+------------+--------+----------------------
+
| 7c5e6499-7ef7-4e36-8216-62c2941d21ff | TenantA_VM1 | ACTIVE | TenantA-Net=10.0.0.3
|
+--------------------------------------+------------+--------+----------------------
+
```

重要的是你要明白不要直接使用Ext-Net附件实例。你从外部网络访问必须使用一个浮动IP。

d.为 TenantA创建和配置一个路由器

```
neutron --os-tenant-name TenantA --os-username UserA --os-password password \
  --os-auth-url=http://localhost:5000/v2.0 router-create TenantA-R1
Created a new router:
+-----------------------+--------------------------------------+
| Field                 | Value                                |
+-----------------------+--------------------------------------+
| admin_state_up        | True                                 |
| external_gateway_info |                                      |
| id                    | 59cd02cb-6ee6-41e1-9165-d251214594fd |
| name                  | TenantA-R1                           |
| status                | ACTIVE                               |
| tenant_id             | 247e478c599f45b5bd297e8ddbbc9b6a     |
+-----------------------+--------------------------------------+

neutron --os-tenant-name TenantA --os-username UserA --os-password password \
  --os-auth-url=http://localhost:5000/v2.0 router-interface-add \
  TenantA-R1 51e2c223-0492-4385-b6e9-83d4e6d10657
```

TenantA-R1添加一个接口

```
neutron --os-tenant-name TenantA --os-username UserA --os-password password \
  --os-auth-url=http://localhost:5000/v2.0 \
  router-gateway-set TenantA-R1 Ext-Net
```

6.TenantA_VM1关联浮动IP

a.创建一个浮动IP

```
neutron --os-tenant-name TenantA --os-username UserA --os-password password \
  --os-auth-url=http://localhost:5000/v2.0 floatingip-create Ext-Net
Created a new floatingip:
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| fixed_ip_address    |                                      |
| floating_ip_address | 30.0.0.2                             |
| floating_network_id | 2c757c9e-d3d6-4154-9a77-336eb99bd573 |
| id                  | 5a1f90ed-aa3c-4df3-82cb-116556e96bf1 |
| port_id             |                                      |
| router_id           |                                      |
| tenant_id           | 247e478c599f45b5bd297e8ddbbc9b6a     |
+---------------------+--------------------------------------+
```

b.获取ID为7c5e6499-7ef7-4e36-8216-62c2941d21ff的虚拟机的端口ID

```
neutron --os-tenant-name TenantA --os-username UserA --os-password password \
  --os-auth-url=http://localhost:5000/v2.0 port-list -- \
  --device_id 7c5e6499-7ef7-4e36-8216-62c2941d21ff
+--------------------------------------+------+-------------------+-----------------
-------------------------------------------------------------+
| id                                   | name | mac_address       | fixed_ips
|
+--------------------------------------+------+-------------------+-----------------
-------------------------------------------------------------+
| 6071d430-c66e-4125-b972-9a937c427520 |      | fa:16:3e:a0:73:0d | {"subnet_id":
"51e2c223-0492-4385-b6e9-83d4e6d10657", "ip_address": "10.0.0.3"} |
+--------------------------------------+------+-------------------+-----------------
```

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

c.为虚拟机端口分配浮动IP

```
neutron --os-tenant-name TenantA --os-username UserA --os-password password \
  --os-auth-url=http://localhost:5000/v2.0 floatingip-associate \
  5a1f90ed-aa3c-4df3-82cb-116556e96bf1 6071d430-c66e-4125-b972-9a937c427520
Associated floatingip 5a1f90ed-aa3c-4df3-82cb-116556e96bf1

neutron floatingip-list
+--------------------------------------+------------------+---------------------+----
----------------------------------+
| id                                   | fixed_ip_address | floating_ip_address |
port_id                              |
+--------------------------------------+------------------+---------------------+----
----------------------------------+
| 5a1f90ed-aa3c-4df3-82cb-116556e96bf1 | 10.0.0.3         | 30.0.0.2            |
6071d430-c66e-4125-b972-9a937c427520 |
+--------------------------------------+------------------+---------------------+----
----------------------------------+
```

7.从 TenantA的虚拟机ping公网

在我的环境中，与物理路由器连接的公共网络是192.168.1.0/24，并且也连接到外部网络30.0.0.0/24。使用浮动IP和虚拟路由器，你可以在TenantA的服务器ping通公共网络：

```
ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_req=1 ttl=64 time=1.74 ms
64 bytes from 192.168.1.1: icmp_req=2 ttl=64 time=1.50 ms
64 bytes from 192.168.1.1: icmp_req=3 ttl=64 time=1.23 ms
^C
--- 192.168.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.234/1.495/1.745/0.211 ms
```

8.ping TenantA服务器的浮动IP

```
ping 30.0.0.2
PING 30.0.0.2 (30.0.0.2) 56(84) bytes of data.
64 bytes from 30.0.0.2: icmp_req=1 ttl=63 time=45.0 ms
64 bytes from 30.0.0.2: icmp_req=2 ttl=63 time=0.898 ms
64 bytes from 30.0.0.2: icmp_req=3 ttl=63 time=0.940 ms
^C
--- 30.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 0.898/15.621/45.027/20.793 ms
```

9.为TenantA创建其它服务器。

你可以为TenantA创建更多的服务器并为它们分配浮动IP。

10.服务TenantC

为TenantC创建10.0.0.0/24和10.0.1.0/24两个私有网络，一些服务器和连接到这些子网的一个路由器及一些浮动IP。

a.为TenantC创建网络和子网：

```
neutron --os-tenant-name TenantC --os-username UserC --os-password password \
  --os-auth-url=http://localhost:5000/v2.0 net-create TenantC-Net1
neutron --os-tenant-name TenantC --os-username UserC --os-password password \
   --os-auth-url=http://localhost:5000/v2.0 subnet-create TenantC-Net1 \
    10.0.0.0/24 --name TenantC-Subnet1
neutron --os-tenant-name TenantC --os-username UserC --os-password password \
  --os-auth-url=http://localhost:5000/v2.0 net-create TenantC-Net2
neutron --os-tenant-name TenantC --os-username UserC --os-password password \
  --os-auth-url=http://localhost:5000/v2.0 subnet-create TenantC-Net2 \
  10.0.1.0/24 --name TenantC-Subnet2
```

之后可以用admin用户查询提供的网络信息

```
neutron net-show TenantC-Net1
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 91309738-c317-40a3-81bb-bed7a3917a85 |
| name                      | TenantC-Net1                         |
| provider:network_type     | gre                                  |
| provider:physical_network |                                      |
| provider:segmentation_id  | 2                                    |
| router:external           | False                                |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   | cf03fd1e-164b-4527-bc87-2b2631634b83 |
| tenant_id                 | 2b4fec24e62e4ff28a8445ad83150f9d     |
+---------------------------+--------------------------------------+

neutron net-show TenantC-Net2
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 5b373ad2-7866-44f4-8087-f87148abd623 |
| name                      | TenantC-Net2                         |
| provider:network_type     | gre                                  |
| provider:physical_network |                                      |
| provider:segmentation_id  | 3                                    |
| router:external           | False                                |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   | 38f0b2f0-9f98-4bf6-9520-f4abede03300 |
| tenant_id                 | 2b4fec24e62e4ff28a8445ad83150f9d     |
+---------------------------+--------------------------------------+
```

你可以看到GRE隧道ID（它是provider:segmentation_id）2和3。也需要注意网络ID和子网ID因为你需要使用它们创建虚拟机和路由器。

b.在TenantC-Net1为TenantC创建服务器TenantC-VM1

```
nova --os-tenant-name TenantC --os-username UserC --os-password password  \
  --os-auth-url=http://localhost:5000/v2.0 boot --image tty --flavor 1 \
  --nic net-id=91309738-c317-40a3-81bb-bed7a3917a85 TenantC_VM1
```

c.在TenantC-Net2为TenantC创建服务器TenanC-VM3

```
nova --os-tenant-name TenantC --os-username UserC --os-password password \
  --os-auth-url=http://localhost:5000/v2.0 boot --image tty --flavor 1 \
  --nic net-id=5b373ad2-7866-44f4-8087-f87148abd623 TenantC_VM3
```

d.TenantC服务器列表

```
nova --os-tenant-name TenantC --os-username UserC --os-password password \
  --os-auth-url=http://localhost:5000/v2.0 list
+--------------------------------------+-------------+--------+----------------------
-+
| ID                                   | Name        | Status | Networks
|
+--------------------------------------+-------------+--------+----------------------
-+
| b739fa09-902f-4b37-bcb4-06e8a2506823 | TenantC_VM1 | ACTIVE | TenantC-Net1=10.0.0.3
|
| 17e255b2-b14f-48b3-ab32-5df36566d2e8 | TenantC_VM3 | ACTIVE | TenantC-Net2=10.0.1.3
|
+--------------------------------------+-------------+--------+----------------------
-+
```

记住服务器ID，因为稍后你将用到它们。

e.确保服务器获取它们的IP.
你可以使用vnc登陆虚拟机检查它们是否获取了ip。如果没有，你必须确保网络组建正常运行以及GRE隧道可以工作。

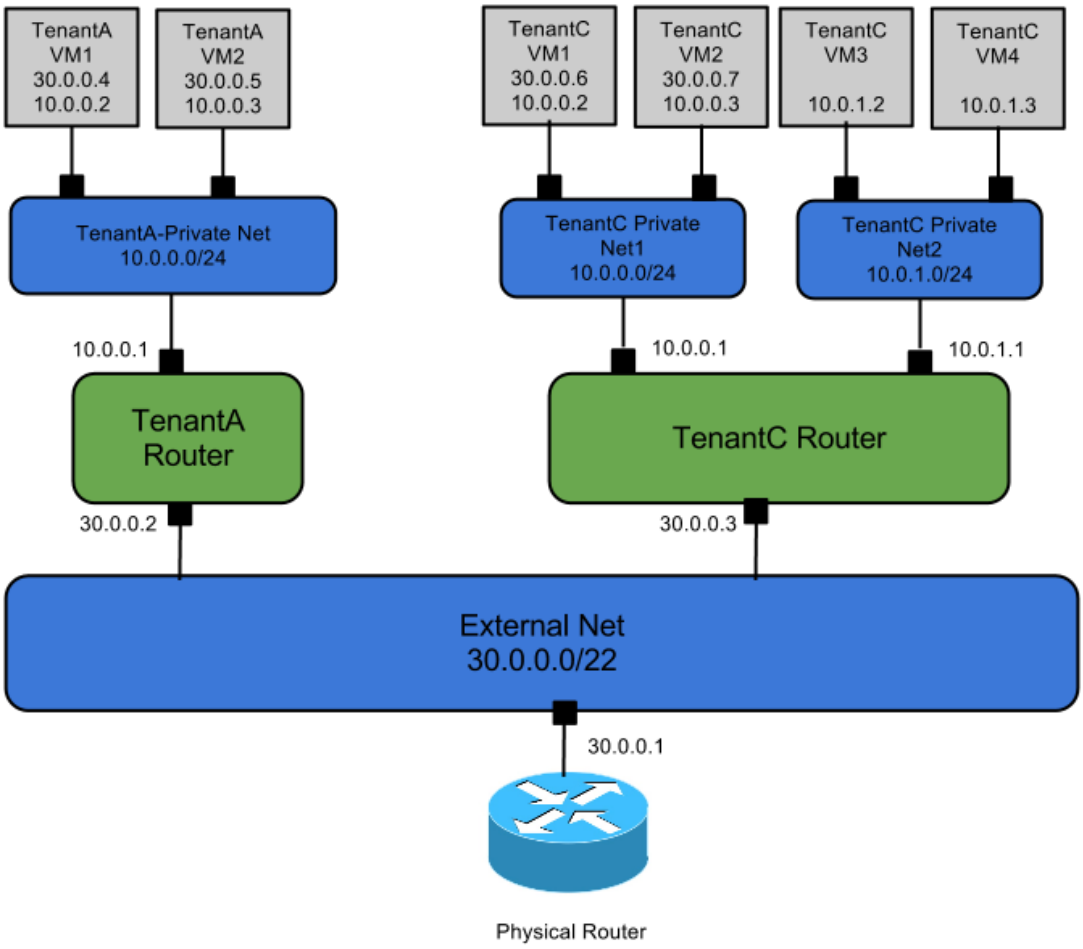
f.为TenantC创建和配置一个路由器。

```
neutron --os-tenant-name TenantC --os-username UserC --os-password password \
  --os-auth-url=http://localhost:5000/v2.0 router-create TenantC-R1

neutron --os-tenant-name TenantC --os-username UserC --os-password password \
  --os-auth-url=http://localhost:5000/v2.0 router-interface-add \
  TenantC-R1 cf03fd1e-164b-4527-bc87-2b2631634b83
neutron --os-tenant-name TenantC --os-username UserC --os-password password \
  --os-auth-url=http://localhost:5000/v2.0 router-interface-add \
  TenantC-R1 38f0b2f0-9f98-4bf6-9520-f4abede03300

neutron --os-tenant-name TenantC --os-username UserC --os-password password \
  --os-auth-url=http://localhost:5000/v2.0 \
  router-gateway-set TenantC-R1 Ext-Net
```

g.检查点。从TenantC的服务器中ping
因为路由器有两个子网，所以虚拟机可以互相ping通。因为路由器的网关接口已经配置，所以TenantC的服务器可以ping通外部网络。如192.168.1.1, 30.0.0.1等

i.为TenantC的服务器分配浮动IP
你可以使用TenantA部分中类似的命令

使用示例：每个私有网络配置路由器
这个用例代表了一些更高级的路由场景，在该场景中，每个租户至少有一个路由器，以及潜在的通过网络api创建额外的路由器。租户可以创建自己的网络，上联到潜在的路由器。这个模型允许租户定义、多层应用，路由器后面的每一层是一个独立的网络。因为有多个路由器，租户子网可以重叠，没有冲突。因为通过SNAT或浮动IP访问外部网络。每个路由器上行和浮动IP是从外部网络分配的子网。



添加Orchestration 服务

Orchestration 服务概述

Orchestration 提供了基于模板的编排，通过调用OpenStack API来生成云应用的应用程序。软件集成了OpenStack的其它核心组件到一个文件模板系统。使用模板可以创建大多数OpenStack资源类型，如实例、浮动IP、卷、安全组、用户等。同时提供了一些高级功能，如实例高可用，实例水平扩展和嵌入式堆栈。通过与OpenStack其它核心项目的高度集成，所有OpenStack核心项目将获得更大的用户群。

这个服务允许部署人员直接与流程服务或其它自定义插件集成。

业务流程服务包含以下组件：
heat 命令行客户端。与运行AWS CloudFormation API的heat-api通信。开发人员还可以直接使用编排服务的 REST API。
heat-api组件。提供一个OpenStack原生的REST API处理使用RPC发送到heat-engine的API请求。
heat-api-cfn组件。提供了一个AWS Query API兼容AWS CloudFormation和处理使用RPC发送到heat-engine的API请求。
heat-engine 从模板启动Orchestrates 和为API的消费者提供事件回调

安装Orchestration 服务

1.在控制器节点安装Orchestration 服务

```
yum -y install openstack-heat-api openstack-heat-engine openstack-heat-api-cfn
```

2.在配置文件中指定存储Orchestration服务数据的数据库位置。

```
openstack-config --set /etc/heat/heat.conf \
  DEFAULT sql_connection mysql://heat:openstack@controller/heat
```

3.以root用户登录，并创建一个heat数据库和用户

```
mysql -u root -e "CREATE DATABASE heat;GRANT ALL PRIVILEGES ON heat.* TO
'heat'@'localhost' IDENTIFIED BY 'openstack';GRANT ALL PRIVILEGES ON heat.* TO
'heat'@'%' IDENTIFIED BY 'openstack';"
```

4.创建heat服务表
heat-manage db_sync

忽略DeprecationWarning 警告

5.创建heat 用户
keystone user-create --name=heat --pass=heat --email=heat@example.com
```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |         heat@example.com         |
| enabled  |               True               |
|    id    | 4e774740ccbf4a7a892ad82aa58dddf7 |
|   name   |               heat               |
+----------+----------------------------------+
```

keystone user-role-add --user=heat --tenant=service --role=admin

6.修改配置文件/etc/heat/heat.conf的 [keystone_authtoken] 和[ec2_authtoken] 段，添加 Orchestration 的认证信息：
```
[keystone_authtoken]
auth_host = controller
auth_port = 35357
auth_protocol = http
auth_uri = http://controller:5000/v2.0
admin_tenant_name = service
admin_user = heat
admin_password = heat
[ec2_authtoken]
auth_uri = http://controller:5000/v2.0
keystone_ec2_uri = http://controller:5000/v2.0/ec2tokens
```

7.在Identity服务中注册Heat 和CloudFormation API以便其它OpenStack服务可以找到这些API。注册服务并指定端点：
keystone service-create --name=heat --type=orchestration \
  --description="Heat Orchestration API"
```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |      Heat Orchestration API      |
|      id     | 42f90a5f339f4652b3c806a5cc94f750 |
|     name    |               heat               |
|     type    |          orchestration           |
+-------------+----------------------------------+
```

8.使用服务id创建端点。
keystone endpoint-create \
  --service-id=42f90a5f339f4652b3c806a5cc94f750 \
  --publicurl=http://controller:8004/v1/%\(tenant_id\)s \
  --internalurl=http://controller:8004/v1/%\(tenant_id\)s \
  --adminurl=http://controller:8004/v1/%\(tenant_id\)s
```
+-------------+-----------------------------------------+
|   Property  |                  Value                  |
+-------------+-----------------------------------------+
|   adminurl  | http://controller:8004/v1/%(tenant_id)s |
|      id     |     6baa5166e8564bc59e6ac7797dbb06c4    |
| internalurl | http://controller:8004/v1/%(tenant_id)s |
|  publicurl  | http://controller:8004/v1/%(tenant_id)s |
|    region   |                regionOne                |
|  service_id |     42f90a5f339f4652b3c806a5cc94f750    |
+-------------+-----------------------------------------+
```

9.注册heat-cfn服务
```
keystone service-create --name=heat-cfn --type=cloudformation \
  --description="Heat CloudFormation API"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |     Heat CloudFormation API      |
|      id     | a34a7387b09d40e8a859d994cad9c813 |
|     name    |             heat-cfn             |
|     type    |          cloudformation          |
+-------------+----------------------------------+
```

10.使用heat-cfn服务ID创建端点
```
keystone endpoint-create \
  --service-id=a34a7387b09d40e8a859d994cad9c813 \
  --publicurl=http://controller:8000/v1 \
  --internalurl=http://controller:8000/v1 \
  --adminurl=http://controller:8000/v1
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  |    http://controller:8000/v1     |
|      id     | 066d32f9c42a4e519165f8a28a540fd3 |
| internalurl |    http://controller:8000/v1     |
|  publicurl  |    http://controller:8000/v1     |
|    region   |            regionOne             |
|  service_id | a34a7387b09d40e8a859d994cad9c813 |
+-------------+----------------------------------+
```

11.启动服务
```
service openstack-heat-api start
service openstack-heat-api-cfn start
service openstack-heat-engine start
chkconfig openstack-heat-api on
chkconfig openstack-heat-api-cfn on
chkconfig openstack-heat-engine on
```

验证Orchestration 安装
验证服务是否正确安装和配置，必须设置正确的认证信息。
```
source openrc.sh
```

创建和管理栈
从一个示例模板创建栈
1.创建一个栈或模板，从example template file，运行下面的命令：
```
wget http://fedorapeople.org/groups/heat/prebuilt-jeos-images/F17-x86_64-
cfntools.qcow2

nova keypair-add HEAT_KEY > heat_key.priv
chmod 600 heat_key.priv

glance image-create --name="F17-x86_64-cfntools" --disk-format=qcow2   --container-
format=bare --is-public=true < F17-x86_64-cfntools.qcow2
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | afab0f79bac770d61d24b4d0560b5f70     |
| container_format | bare                                 |
| created_at       | 2014-01-03T07:31:07                  |
| deleted          | False                                |
| deleted_at       | None                                 |
| disk_format      | qcow2                                |
| id               | 22c2ebc7-deed-451f-b8e6-ab1063b70206 |
| is_public        | True                                 |
```

```
| min_disk          | 0                                    |
| min_ram           | 0                                    |
| name              | F17-x86_64-cfntools                  |
| owner             | None                                 |
| protected         | False                                |
| size              | 476704768                            |
| status            | active                               |
| updated_at        | 2014-01-03T07:31:26                  |
+-------------------+--------------------------------------+

heat stack-create mystack --template-file=/root/heat-
templates/cfn/F17/WordPress_Single_Instance.template \
--
parameters="InstanceType=m1.large;DBUsername=USERNAME;DBPassword=PASSWORD;KeyName=HEA
T_KEY;LinuxDistribution=F17"
+--------------------------------------+------------+--------------------+----------
-----------+
| id                                   | stack_name | stack_status       |
creation_time        |
+--------------------------------------+------------+--------------------+----------
-----------+
| e5b63e6a-4a7e-40c9-818d-5d9b8933c4e5 | mystack    | CREATE_IN_PROGRESS | 2014-01-
03T07:33:14Z |
+--------------------------------------+------------+--------------------+----------
-----------+
```

2.你还可以使用stack-create验证一个模板文件，但不创建栈。

```
heat stack-create mystack --template-file=/root/heat-
templates/cfn/F17/WordPress_Single_Instance.template
```

如果验证失败将返回错误信息。

获取栈的信息

你可以使用一些命令查看栈的状态和历史

查看当前用户可见的栈

```
heat stack-list
+--------------------------------------+--------------+-----------------+----------
-----------+
| id           | stack_name    | stack_status    | creation_time        |
+--------------------------------------+--------------+-----------------+----------
-----------+
| 4c712026-dcd5-4664-90b8-0915494c1332 | mystack       | CREATE_COMPLETE | 2013-04-
03T23:22:08Z |
| 7edc7480-bda5-4e1c-9d5d-f567d3b6a050 | my-otherstack | CREATE_FAILED   | 2013-04-
03T23:28:20Z |
+--------------------------------------+--------------+-----------------+----------
-----------+
```

查看栈的详细信息

```
heat stack-show mystack
```

栈包含资源的集合

列出资源极其状态：

```
heat resource-list mystack
+---------------------+--------------------+-----------------+----------------------+
| logical_resource_id | resource_type      | resource_status | updated_time         |
+---------------------+--------------------+-----------------+----------------------+
| WikiDatabase        | AWS::EC2::Instance | CREATE_COMPLETE | 2013-04-03T23:25:56Z |
+---------------------+--------------------+-----------------+----------------------+
```

显示栈指定资源的详细信息

```
heat resource-show mystack WikiDatabase
```

有些资源关联的元数据可以在资源的整个生命周期中变化

```
heat resource-metadata mystack WikiDatabase
```

栈整个生命周期中的一系列事件

```
heat event-list mystack
+---------------------+----+----------------------+-----------------+-------------
--------+
| logical_resource_id | id | resource_status_reason | resource_status | event_time
|
+---------------------+----+----------------------+-----------------+-------------
--------+
| WikiDatabase        | 1  | state changed          | IN_PROGRESS     | 2013-04-
03T23:22:09Z |
| WikiDatabase        | 2  | state changed          | CREATE_COMPLETE | 2013-04-
03T23:25:56Z |
+---------------------+----+----------------------+-----------------+-------------
--------+
```

显示特定事件的细节

```
heat event-show WikiDatabase 1
```

更新栈

从修改过得模板更新一个已存在的栈

```
heat stack-update mystack --template-
file=/path/to/heat/templates/WordPress_Single_Instance_v2.template
     --
parameters="InstanceType=m1.large;DBUsername=wp;DBPassword=verybadpassword;KeyName=he
at_key;LinuxDistribution=F17"
+--------------------------------------+--------------+-----------------+----------
-----------+
| id           | stack_name    | stack_status    | creation_time        |
+--------------------------------------+--------------+-----------------+----------
-----------+
| 4c712026-dcd5-4664-90b8-0915494c1332 | mystack       | UPDATE_COMPLETE | 2013-04-
03T23:22:08Z |
| 7edc7480-bda5-4e1c-9d5d-f567d3b6a050 | my-otherstack | CREATE_FAILED   | 2013-04-
03T23:28:20Z |
+--------------------------------------+--------------+-----------------+----------
-----------+
```

有些资源会就地更新，有些资源将替换为新的资源

添加Telemetry 服务

OpenStack telemetry服务：

有效地收集关于CPU和网络成本的计量数据。
轮询服务或基础设施通过发送监控通知搜集数据。
配置搜集数据的类型，以满足不同的操作需求。通过REST API访问和插入统计数据。
使用额外的插件扩展架构，以搜集自定义数据。
产生不能否定的签名计量信息。

该系统有以下基本组件：

一个计算代理(ceilometer-agent-compute)。运行在每个计算节点，统计资源占用率。未来可能有其它类型的代理，但现在我们专注于开发计算代理。
一个中央代理(ceilometer-agent-central)。运行在中央管理服务器上，轮询统计资源的资源利用率信息，不依赖实例或计算节点。
一个收集器(ceilometer-collector)。在一个或多个中央管理服务器运行，监控消息队列（来自代理的通知和计量数据）。处理过的通知消息变成了计量信息，并使用合适的标题发回消息总线。遥测信息不经过修改直接存入数据库。
一个报警通知(ceilometer-alarm-notifier)。运行在一个或多个中央管理服务器，基于一组统计样本评估的阀值设置报警。
一个数据存储。一个能够处理并发写入(来自一个或多个统计器实例)和读取(来自API服务器)的数据库。
一个API服务器(ceilometer-api)。运行于一个或多个中央管理服务器，对数据存储的数据提供访问。

这些服务使用标准的OpenStack消息总线进行通信。只有收集器和API服务器访问数据存储。

安装Telemetry 服务

OpenStack遥测是一个API服务,提供了一个收集器和一系列不同的代理。你可以在计算节点等安装这些代理服务之前，你必须在控制节点执行这些步骤安装核心组件。

1.在控制节点安装遥测服务。

```
yum -y install openstack-ceilometer-api openstack-ceilometer-collector openstack-
ceilometer-central python-ceilometerclient
```

2.遥测服务使用数据库来存储信息。在配置文件中指定数据库的位置。这个例子使用在控制节点上的MongoDB：

```
yum -y install mongodb-server mongodb
```

3.启动MongoDB

```
service mongod start
chkconfig mongod on
```

4.创建ceilometer 数据库和用户：

```
mongo
> use ceilometer
> db.addUser( { user: "ceilometer",
              pwd: "openstack",
              roles: [ "readWrite", "dbAdmin" ]
            } )
```

5.配置遥测服务，以使用数据库

```
openstack-config --set /etc/ceilometer/ceilometer.conf \
  database connection mongodb://ceilometer:openstack@controller:27017/ceilometer
```

6.你必须定义一个在遥测服务节点之间共享的秘钥。使用openssl生成一个随机的token并存储到配置文件中：

```
ADMIN_TOKEN=$(openssl rand -hex 10)
07ca7d74d8729ef24371
echo $ADMIN_TOKEN
openstack-config --set /etc/ceilometer/ceilometer.conf publisher_rpc metering_secret
$ADMIN_TOKEN
```

7.在Identity服务中创建一个遥测服务使用的用户ceilometer 。使用service 租户和admin角色：

```
keystone user-create --name=ceilometer --pass=ceilometer --
email=ceilometer@example.com
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |      ceilometer@example.com      |
| enabled  |               True               |
|    id    | 578011e935a240f38efcdfb2a7411e8f |
|   name   |            ceilometer            |
+----------+----------------------------------+

keystone user-role-add --user=ceilometer --tenant=service --role=admin
```

8.添加认证信息到 Telemetry 服务的配置文件：

```
openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken auth_host
controller
openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken admin_user
ceilometer
openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken
admin_tenant_name service
openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken
auth_protocol http
openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken
```

admin_password ceilometer

9.在Identity服务中注册Telemetry 服务，以使其它OpenStack服务可以找到它。使用 keystone 命令注册服务和指定端点：

```
keystone service-create --name=ceilometer --type=metering \
  --description="Ceilometer Telemetry Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |   Ceilometer Telemetry Service   |
|      id     | 507eb578d0ae4d018a420b23378ca9e9 |
|     name    |            ceilometer            |
|     type    |             metering             |
+-------------+----------------------------------+
```

10.记住该服务返回的id，创建端点需要使用它：

```
keystone endpoint-create \
  --service-id=507eb578d0ae4d018a420b23378ca9e9 \
  --publicurl=http://controller:8777/ \
  --internalurl=http://controller:8777/ \
  --adminurl=http://controller:8777/
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  |     http://controller:8777/      |
|      id     | a48ffb9350f44813b9c825591c553d12 |
| internalurl |     http://controller:8777/      |
|  publicurl  |     http://controller:8777/      |
|    region   |            regionOne             |
|  service_id | 507eb578d0ae4d018a420b23378ca9e9 |
+-------------+----------------------------------+
```

11.启动penstack-ceilometer-api, openstack-ceilometer-central 和openstack-ceilometer-collector服务：

修复服务脚本的bug

```
sed -i '74s/rh_status_q/start/g' /etc/init.d/openstack-ceilometer-*
service openstack-ceilometer-api start
service openstack-ceilometer-central start
service openstack-ceilometer-collector start
chkconfig openstack-ceilometer-api on
chkconfig openstack-ceilometer-central on
chkconfig openstack-ceilometer-collector on
```

为遥测服务安装计算代理

这个过程的细节是如何在计算节点安装代理

1.在计算节点安装遥测服务

```
yum -y update python-mako --disablerepo=\* --enablerepo=openstack-havana
yum -y install python-webtest --disablerepo=\* --enablerepo=openstack-havana
yum -y install openstack-ceilometer-compute
```

2.配置/etc/nova/nova.conf中的选项

```
openstack-config --set /etc/nova/nova.conf DEFAULT instance_usage_audit True
openstack-config --set /etc/nova/nova.conf DEFAULT instance_usage_audit_period hour
openstack-config --set /etc/nova/nova.conf DEFAULT notify_on_state_change
vm_and_task_state
openstack-config --set /etc/nova/nova.conf DEFAULT notification_driver
nova.openstack.common.notifier.rpc_notifier
openstack-config --set /etc/nova/nova.conf DEFAULT notification_driver
ceilometer.compute.nova_driver
```

3.设置前面定义的秘钥。遥测服务节点共享这个秘钥：

```
openstack-config --set /etc/ceilometer/ceilometer.conf publisher_rpc metering_secret
$ADMIN_TOKEN
```

4.启动服务
```
service openstack-ceilometer-compute start
chkconfig openstack-ceilometer-compute on
```

为遥测服务添加映像服务代理
1.为了检索映像样本，你必须配置映像服务向总线发通知。
```
openstack-config --set /etc/ceilometer/ceilometer.conf DEFAULT notifier_strategy qpid
```

2.重启映像服务，应用新的设置
```
service openstack-glance-api restart
service openstack-glance-registry restart
```

为遥测服务添加块存储服务代理
1.为检索卷样本，你必须配置块存储服务向总线发送通知。
```
openstack-config --set /etc/cinder/cinder.conf DEFAULT control_exchange cinder
openstack-config --set /etc/cinder/cinder.conf DEFAULT notification_driver
cinder.openstack.common.notifier.rpc_notifier
```

2.重启块存储服务应用新的配置
```
service openstack-cinder-api restart
service openstack-cinder-volume restart
```

为遥测服务添加对象存储代理
1.为检索对象存储统计资料，遥测服务需要以 ResellerAdmin 角色访问对象存储。为你租户 os_tenant_name 的用户os_username 赋予这个角色。
```
keystone role-create --name=ResellerAdmin
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|    id    | 35d9ae7c38fb4d938ed8150794f30922 |
|   name   |          ResellerAdmin           |
+----------+----------------------------------+

keystone user-role-add --tenant service --user ceilometer --role
35d9ae7c38fb4d938ed8150794f30922
```

2.你必须添加一个遥测中间件来处理对象存储的出入流量。在/etc/swift/proxy-server.conf文件增加这些行：
```
[filter:ceilometer]
use = egg:ceilometer#swift
```

3.在同一个文件增加ceilometer 配置到 pipeline
```
[pipeline:main]
pipeline = healthcheck cache authtoken keystoneauth ceilometer proxy-server
```

4.重启对象服务
```
service openstack-swift-proxy-server restart
```

验证遥测服务安装
要测试遥测服务的安装，从映像服务下载映像，并从遥测服务查看统计数据。
1.使用ceilometer meter-list测试访问遥测服务
```
ceilometer meter-list
+------------+-------+-------+--------------------------------------+---------+------
----------------------------+
| Name       | Type  | Unit  | Resource ID                          | User ID |
Project ID                       |
+------------+-------+-------+--------------------------------------+---------+------
----------------------------+
| image      | gauge | image | 9e5c2bee-0373-414c-b4af-b91b0246ad3b | None    |
e66d97ac1b704897853412fc8450f7b9 |
| image.size | gauge | B     | 9e5c2bee-0373-414c-b4af-b91b0246ad3b | None    |
e66d97ac1b704897853412fc8450f7b9 |
```

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

2.从glance下载一个映像
glance image-download "CirrOS 0.3.1" > cirros.img

3.重新运行ceilometer meter-list命令，下载已被遥测服务发现并存储。
ceilometer meter-list

```
+----------------+-------+-------+--------------------------------------+---------+--
---------------------------------+
| Name           | Type  | Unit  | Resource ID                          | User ID |
Project ID                       |
+----------------+-------+-------+--------------------------------------+---------+--
---------------------------------+
| image          | gauge | image | 9e5c2bee-0373-414c-b4af-b91b0246ad3b | None    |
e66d97ac1b704897853412fc8450f7b9 |
| image.download | delta | B     | 9e5c2bee-0373-414c-b4af-b91b0246ad3b | None    |
e66d97ac1b704897853412fc8450f7b9 |
| image.serve    | delta | B     | 9e5c2bee-0373-414c-b4af-b91b0246ad3b | None    |
e66d97ac1b704897853412fc8450f7b9 |
| image.size     | gauge | B     | 9e5c2bee-0373-414c-b4af-b91b0246ad3b | None    |
e66d97ac1b704897853412fc8450f7b9 |
+----------------+-------+-------+--------------------------------------+---------+--
---------------------------------+
```

4.你可以通过statistics 获取不同的测量值
$ ceilometer statistics -m image.download -p 60

```
+--------+---------------------+---------------------+-------+------------+---------
--+------------+------------+---------+----------------------------+----------------
------------+
| Period | Period Start        | Period End          | Count | Min        | Max
| Sum        | Avg        | Duration | Duration Start             | Duration End
|
+--------+---------------------+---------------------+-------+------------+---------
--+------------+------------+---------+----------------------------+----------------
------------+
| 60     | 2013-11-18T18:08:50 | 2013-11-18T18:09:50 | 1     | 13147648.0 |
13147648.0 | 13147648.0 | 13147648.0 | 0.0      | 2013-11-18T18:09:05.334000 | 2013-
11-18T18:09:05.334000 |
+--------+---------------------+---------------------+-------+------------+---------
--+------------+------------+---------+----------------------------+----------------
------------+
```