OpenStack分步安装【2013-12-29】

**一、OpenStack架构**

|  |  |  |
| --- | --- | --- |
| DashBoard | Horizon | 提供WEB界面 |
| Computer | Nova | 计算也就是虚拟机 |
| Networking | Neutron | 提供给nova网络支持 |
| Object Storage | Swift | 提供对象存储 |
| Block Storage | Cinder | 提供云硬盘给nova，同时备份到Swift |
| Identity Sservice | Keystone | 提供所有组件的认证 |
| Image Service | Glance | 提供给nova镜像服务 |
| Telemetry Service | Cellometer | 监控  cinder,neutron,nova,glance |
| Orchestration Service | Heat | 与AWS cloud兼容 |

组件对应关系：

图1：OpenStack的概念架构

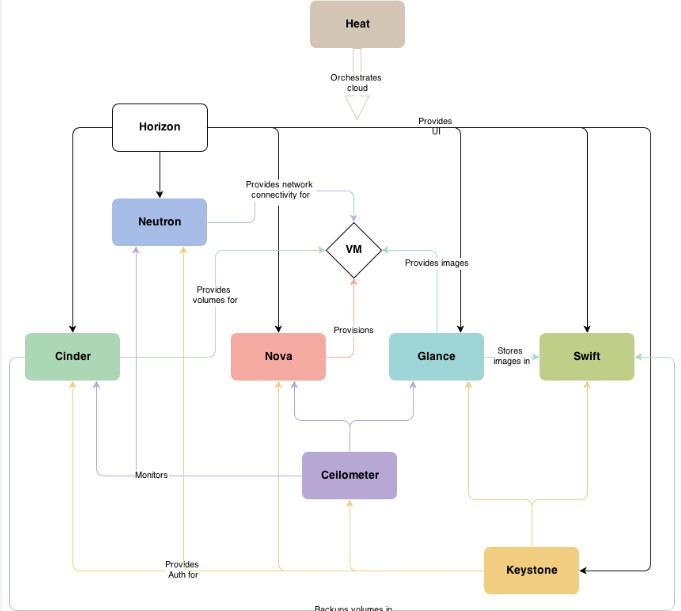


图2：Open Stack逻辑架构

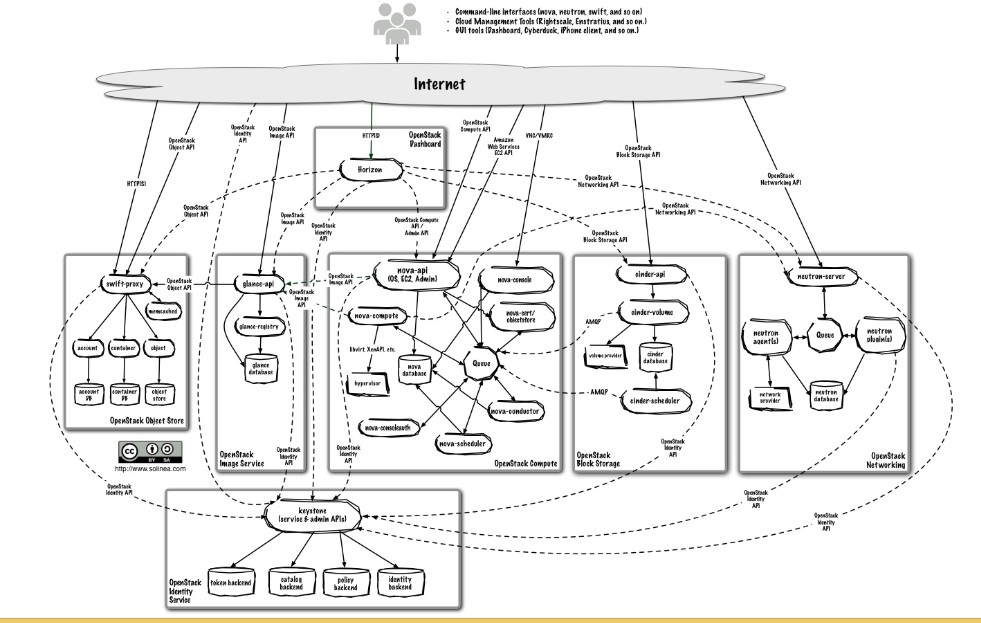
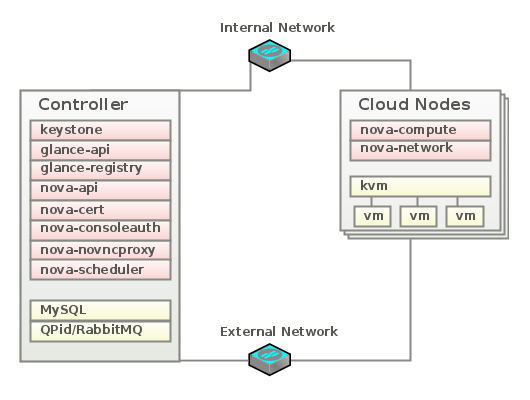


图3：OpenStack基本架构



**二、OpenStack基础操作系统配置**

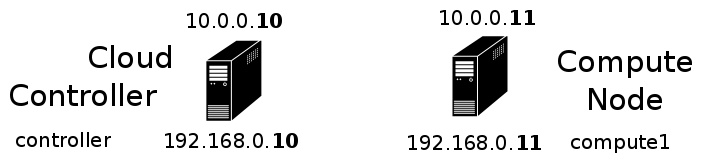
操作系统：

最好是使用64位操作系统，并开启虚拟化VT支持。因为32位的操作系统安装64位的虚拟机会报错。

网络：

两个网络，一个是外部网络，一个是内部节点通信网络

图1：官方基础架构

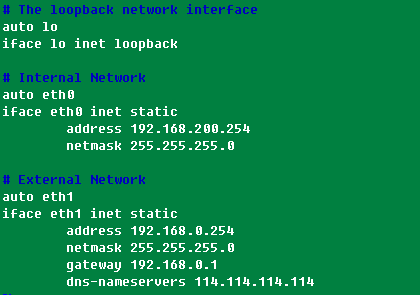


一、分配规则：

**200段：内部网络**

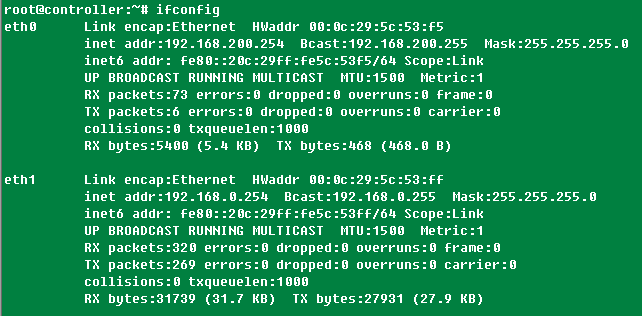
**0段：外部网络**

**# cat /etc/network/interfaces**

[](http://img1.51cto.com/attachment/201312/094848558.png)

重新启动使之生效：

**# service networking restart**



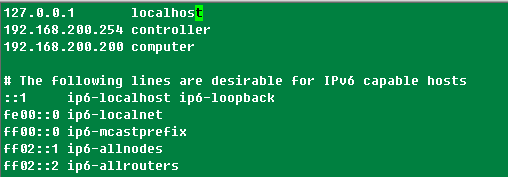
设置一个主机名,并永久生效：

**# hostname controller**

221451813.png

配置主机文件对应

**/etc/hosts**



同步时间：

**# apt-get install ntp**

**# /usr/sbin/ntpdate ntp.api.bz**

221451172.png

密码：

OpenStack各种服务都需要密码的，我们可以用密码生成器来生成密码。如百度和openssl



221453519.png

密码表：

|  |  |
| --- | --- |
| Database password | 数据库的root密码 |
| Rabbit\_pass | rabbit的guest密码 |
| KeyStone\_dbpass | keystone的数据库密码 |
| Admin\_pass | admin的密码，WEBUI密码 |
| Clance\_dbpass | Glance的数据库密码 |
| Nova\_dbpass | nova的数据库密码 |
| Nova\_pass | Nova的密码 |
| Dash\_dbpass | dash的数据库密码 |
| Cinder\_dbpass | cinder的数据库密码 |
| Neutron\_dbpass | Neutron的数据库密码 |
| Neutron\_pass | neutron的密码 |
| Heat\_dbpass | heat的数据库密码 |
| Heat\_pass | Heat的密码 |
| Cellometer\_dbpass | cellometer的数据库密码  监控 |
| Cellonmeter\_pass | collometer的密码 |

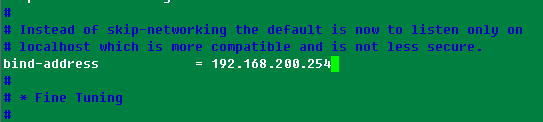
二、数据库：

我们用Mysql,因为openstack大部分用python写的，我们用python-mysqldb去连接数据库。

**#apt-get install python-mysqldb mysql-server**

其中要输入密码

更改绑定IP为外网IP：



查看一下是否绑定到位：

**# netstat -ntlp**



重启数据库

**# service mysql restart**

删除一些匿名用户连接数据库

**# mysql\_secure\_installation**

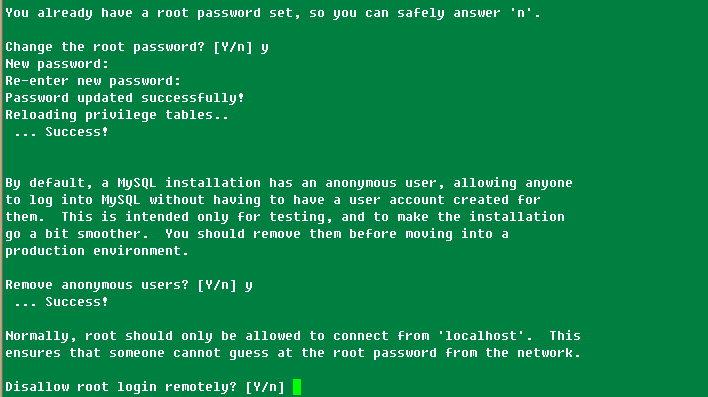
1、修改root密码

2、删除匿名用户

3、只允许root从本地登录

4、删除test数据库

5、马上刷新表的权限



三、OpenStack的包安装：

1、安装Ubuntu Clound云上面的havana包

**# apt-get install python-software-properties**

**# add-apt-repository cloud-archive:havana**

#需要按回车键

2、更新包仓库并重新启动操作系统

**# apt-get update && apt-get dist-upgrade**

**# reboot**

四、安装消息服务rabbitmq：

**#apt-get install rabbitmq-server**

更改密码，指定一个密码。

**#rabbitmqctl change\_password guest $rabbit\_pass**

我们设的密码是rabbit

095452912.png

**三、Openstack安装keystion(identity)**

Idenitity 服务理念：

执行以下功能：

用户管理，跟踪用户以及用户的权限

catalog服务：提供与他们的API catalog 服务

用户（user）

用户表示拥有用户名，密码，邮箱等帐号信息的自然人。

证书(Credentials)

证明资格和权力的东西，如用户名和密码，用户名和API KEY 或者服务本身的认证token

认证（Authentication）

确认用户身份的行为，经过上面描述的“证书”服务确认用户。

令牌（token）

一种认证方式，通过认证后发给连接端。

租户(tenant)

租户可以被理解为一个项目，团队或组织。你必须指定一个相应的租户(tenant)才可以申请OpenStack服务

服务（Service）

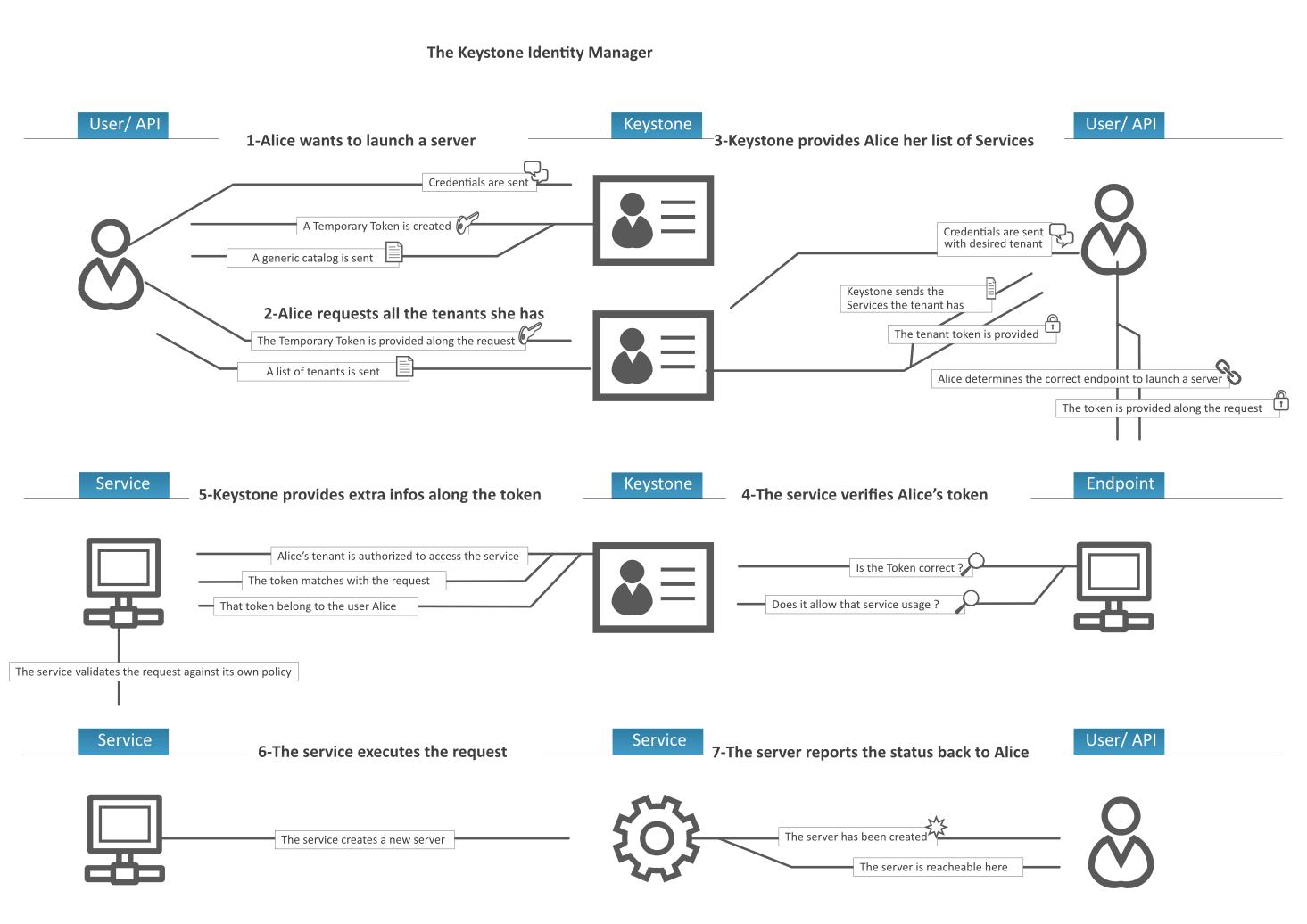
通过它用户可以访问资源并执行操作的一个终端。 即完成以上几步后就给一个终端供client使用。

终端（Endpoint）

通过网络访问的一个URL地址，可以获得你想要访问的服务，

角色（Role）

角色代表特定的租户中的用户用户操作权限



六个过程，alice任务请求创建一个虚拟机

alice发送凭据给keystone,keystone返回一个临时创建的tokent和一个通用的目录。

alice请求提供租户的列表，keystone返回一个他的租户的服务列表。

keystone发送给alice带有租房的凭据、租房的服务列表、租户的token

alice去终端连接服务，并提供一个请求的token,终端去问keystone是否是正确的令牌和是否允许使用这个服务

keystone提供“alice的租户允许访问服务，令牌与请求匹配并且这个令牌属于alice”给Service

Service执行服务请求并把执行状态发送给alice

安装identity服务

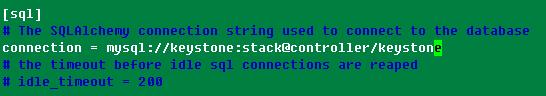
安装软件包

# apt-get install keystone

修改配置文件/etc/keystone/keystone.conf

使用数据库存储数据，我们改一下他的连接方式，默认使用sqlite我们要进行更改

connection = mysql://keystone:stack@controller/keystone



给keystone创建一个keystone数据库

mysql> create database keystone;

mysql> GRANT ALL PRIVILEGES ON keystone.\* TO 'keystone'@'localhost' IDENTIFIED BY 'stack’;

GRANT ALL PRIVILEGES ON keystone.\* TO 'keystone'@'controller' IDENTIFIED BY 'stack';



wKiom1LGICKzduk1AAAiPXONPYg058.jpg

4、同步一下到数据库

# keystone-manage db\_sync

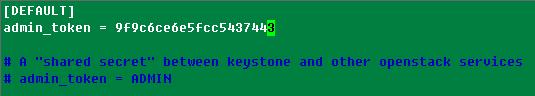
创建了一些空表



6、生成一个token，并放到配置文件中

wKiom1LGIDvCNjDxAAATL_6kryc273.jpg

/etc/keystone/keystone.conf



重启keystone服务

# service keystone restart

定义用户、租户、角色

1、定义环境变量

由于我们并没有用户，所以需要创建用户，但这本身就需要认证，所以我们增加怎么环境变量os\_server\_token os\_service\_endpoint

# export OS\_SERVICE\_TOKEN=9f9c6ce6e5fcc5437443

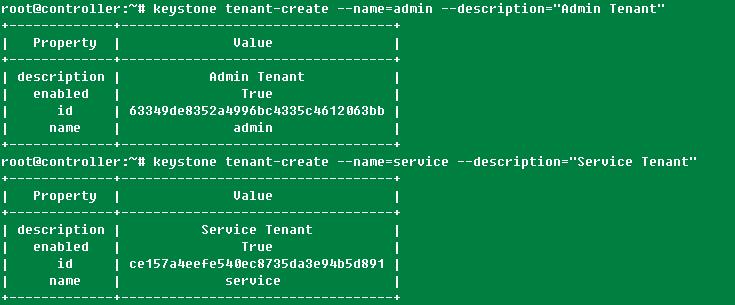
# export OS\_SERVICE\_ENDPOINT=http://controller:35357/v2.0

wKioL1LGIFLBLmcKAAAxv-Bir08254.jpg

创建一个admin和service的租户

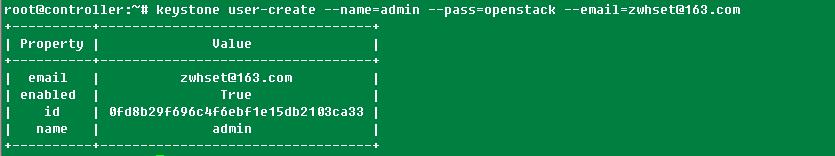
# keystone tenant-create --name=admin --description="Admin Tenant"

# keystone tenant-create --name=service --description="Service Tenant"



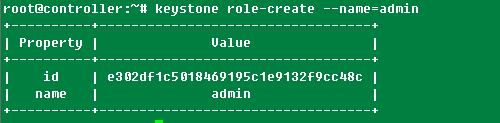
创建一个管理员用户为admin并设置一个密码并指定电子邮件,这是第三个密码了。记下来。

# keystone user-create --name=admin --pass=openstack --email=zwhset@163.com



创建一个管理员用记的角色admin，注意默认使用admin角色将允许大多数的服务。创建的角色应该映射到openstack服务的policy.json文件名中指定的角色

# keystone role-create --name=admin

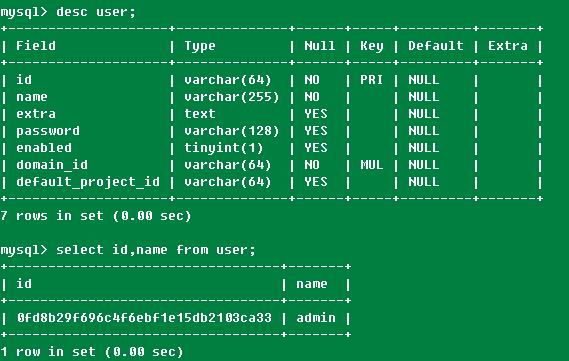


添加角色用户租户的对应关系

# keystone user-role-add --user=admin --tenant=admin --role=admin

wKioL1LGIH3AeskBAAAdgsXYSyc493.jpg

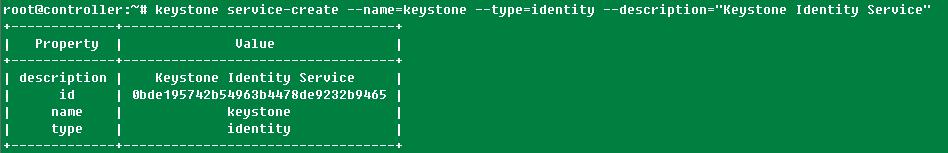
我们看下数据库



定义服务与API的端点

创建一个服务，服务ID是随机生成的

# keystone service-create --name=keystone --type=identity --description="Keystone Identity Service"



通过返回的服务ID指定身份证服务的API端点，你应该提供公共的API，内部API和管理API的URL。

煮酒品茶：注意填写你自己的ID号

# keystone endpoint-create   --service-id=0bde195742b54963b4478de9232b9465   --publicurl=http://controller:5000/v2.0   --internalurl=http://controller:5000/v2.0   --adminurl=http://controller:35357/v2.0

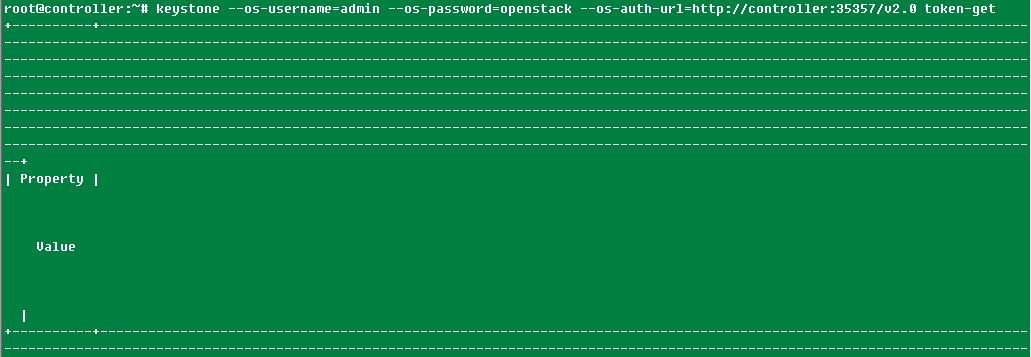


你安装了其它服务时，通过这个指定你的服务。

验证一下：

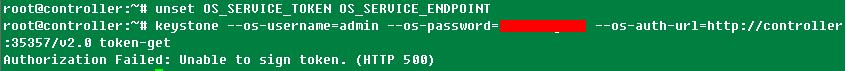
# unset OS\_SERVICE\_TOKEN OS\_SERVICE\_ENDPOINT

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

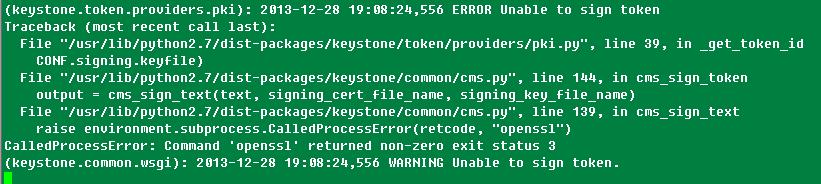


－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

如果发现500错误



查看日志：



更改配置文件：



好了，需要重启下keystone

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

重启下：

# service keystone restart

我们设置环境变量，就不需要每次都填验证参数了。

# export OS\_USERNAME=admin

# export OS\_PASSWORD=openstack

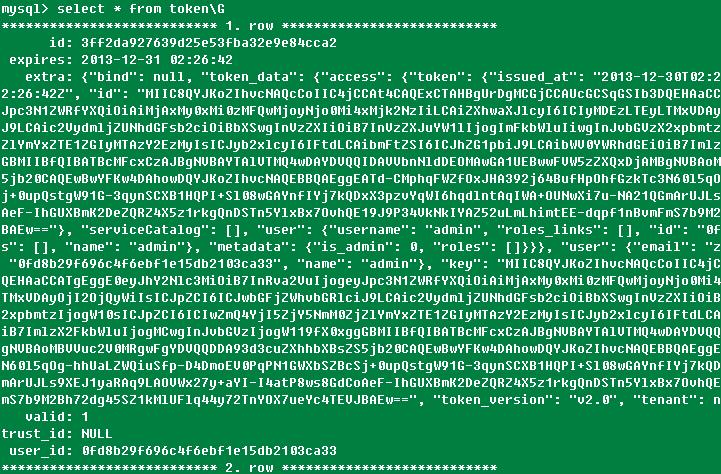
# export OS\_TENANT\_NAME=admin

# export OS\_AUTH\_URL=http://controller:35357/v2.0

查看一下token,事实上他的ID也太长了吧。



查了一下表：



更多可以看用户租户角色：

<http://docs.openstack.org/havana/install-guide/install/apt/content/keystone-users.html>

**四、OpenStack Glance安装**

Glance是OpenStack镜像服务，用来注册、登陆和检索虚拟机镜像。Glance服务提供了一个REST API，使你能够查询虚拟机镜像元数据和检索的实际镜像。通过镜像服务提供的虚拟机镜像可以存储在不同的位置，从简单的文件系统对象存储到类似OpeenStack对象存储系统。

为了简单，本安装镜像服务使用普通文件系统做为后端，也就是说上传镜像将被存储在一个目录里，这个目录是/var/lib/glance/image,你要确保这个目录提供足够的空间，然后再来存储虚拟机的镜像和快照。大小怎么着自己看着办。

镜像服务组件：

Glance-API 承受镜像API的探索，检索和存储镜像的API调用。

Glance-registry 存储，处理和检索有关镜像的元数据，元数据大小、类型等等。

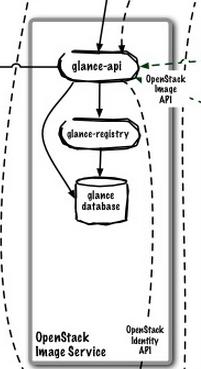
Database 存储镜像元数据，你可以选择你喜欢的数据库，现在比较多的是mysql和sqlite

镜像文件存储在存储库中，你可以选择不同的存储库，比如正常文件系统、 RADOS、Amazon S3 HTTP,也有一些提供只读使用，所以我们可以使用不同的分布式文件系统来扩展。

Object Storage [**{Ring proxy object container account replication updaters auditors}**](http://search.t.qq.com/index.php?pos=242&sort=2&k=Ring%20proxy%20object%20container%20account%20replication%20updaters%20auditors) auditors检查**[{object,container,account}](http://search.t.qq.com/index.php?pos=242&sort=2&k=object,container,account" \t "_blank)**updaters失败更新**[{container account}](http://search.t.qq.com/index.php?pos=242&sort=2&k=container%20account" \t "_blank)** account处理container列表 container处理对象列表 object文件所在地 Ring名称与物理位置映射。

它接受来自最终用户和Nova的镜像和镜像元数据的API请求，并且可以存储在对象存储服务的磁盘文件上。

逻辑图：



安装镜像服务：

glance服务工作在虚拟磁盘镜像上面，用户可以添加一个新镜像，或者从现有的服务中存储快照，使用快照来备份或者做为其它服务器的模版，你可以注册镜像在对象存储或者在其它的文件系统中，例如可以使用WEB服务或者NFS，MFS等。

1、安装镜像服务包

# apt-get install glance python-glanceclient

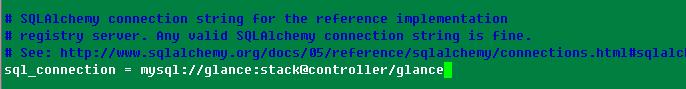
2、配置数据位置，glance-api和glance-registry都有自己的配置文件，路径分别是

/etc/glance/glance-api.conf

/etc/glance/glance-registry.conf修改[default]选项，连接mysql

两个文件都要改

#sql\_connection = mysql://glance:stack@controller/glance



3、删除rm /var/lib/glance/glance.sqlite文件，避免ubuntu使用sqlite连接数据库

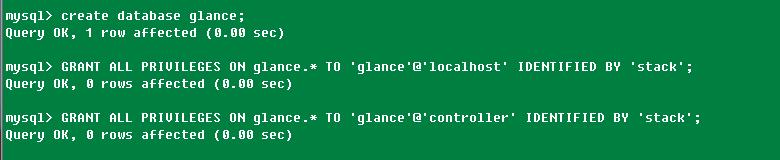
# rm /var/lib/glance/glance.sqlite

4、创建数据库和连接方式，又是一个密码，同样记住了。

mysql> create database glance;

mysql> GRANT ALL PRIVILEGES ON glance.\* TO 'glance'@'localhost' IDENTIFIED BY 'stack';

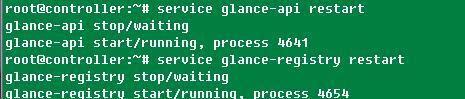
mysql> GRANT ALL PRIVILEGES ON glance.\* TO 'glance'@'controller' IDENTIFIED BY 'stack';



5、重启下

# service glance-api restart

# service glance-registry restart



6、用镜像服务创建数据表

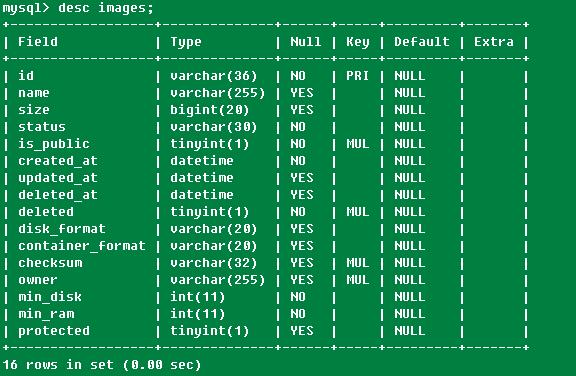
# glance-manage db\_sync

wKioL1LLTOXCCL_eAAAQVe61818830.jpg

查下一下有多少表：

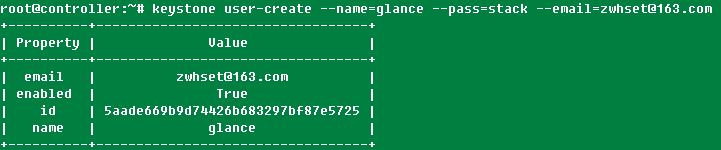


看一下images的表结构：



6、创建一个用户glance在keystone，使用密码和邮件，并把他添加到服务的租户里去。记住咯，又一个密码。

# keystone user-create --name=glance --pass=stack [--email=zwhset@163.com](mailto:--email=zwhset@163.com)



7、把镜像服务注册到keystone里去，修改配置文件。

修改glance-api.comf 和glance-registry.conf

[keystone\_authtoken]

auth\_host = controller

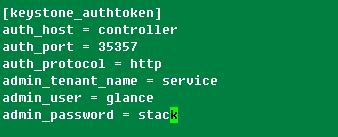
auth\_port = 35357

auth\_protocol = http

admin\_tenant\_name = service

admin\_user = glance

admin\_password = stack



还是一样两个都要改

8、把凭证添加到下面两个文件中去

/etc/glance/glance-api-paste.ini

/etc/glance/glance-registry-paste.ini

[filter:authtoken]

paste.filter\_factory = keystoneclient.middleware.auth\_token:filter\_factory

delay\_auth\_decision = true

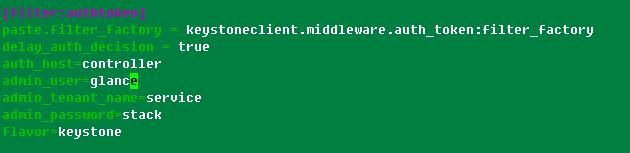
auth\_host=controller

admin\_user=glance

admin\_tenant\_name=service

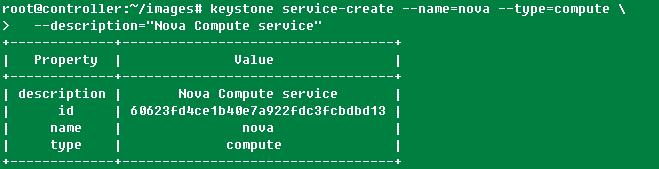
admin\_password=stack

flavor=keystone



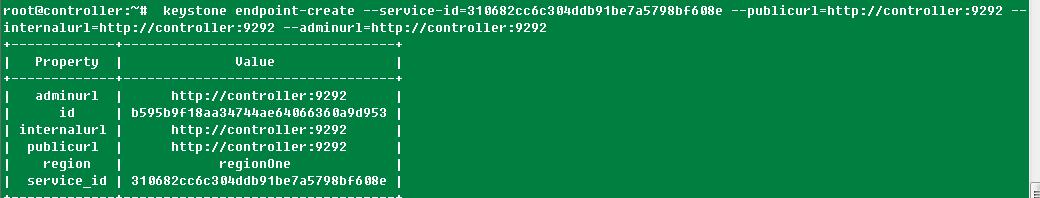
9、在keystone上面注册镜像服务让其它服务可以看到并且创建终端。

# keystone service-create --name=glance --type=image --description="Glance Image Service"



10、使用刚返回的ID创建终端服务。

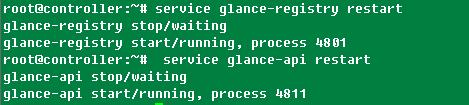
#  keystone endpoint-create --service-id=310682cc6c304ddb91be7a5798bf608e --publicurl=http://controller:9292 --internalurl=http://controller:9292 --adminurl=http://controller:9292



11、重启glance服务以生效刚修改的

# service glance-registry restart

#  service glance-api restart



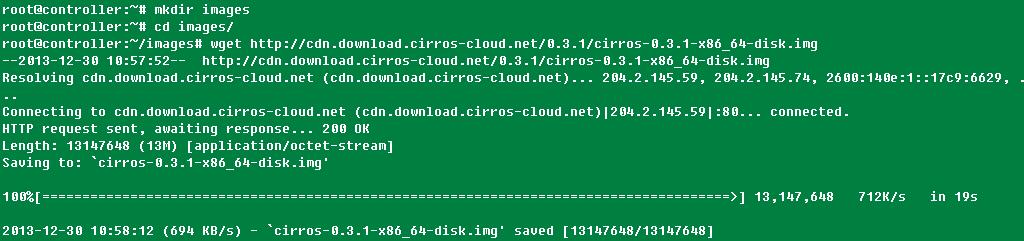
验证镜像服务的安装

1、下载一个13M镜像测试。

# mkdir images

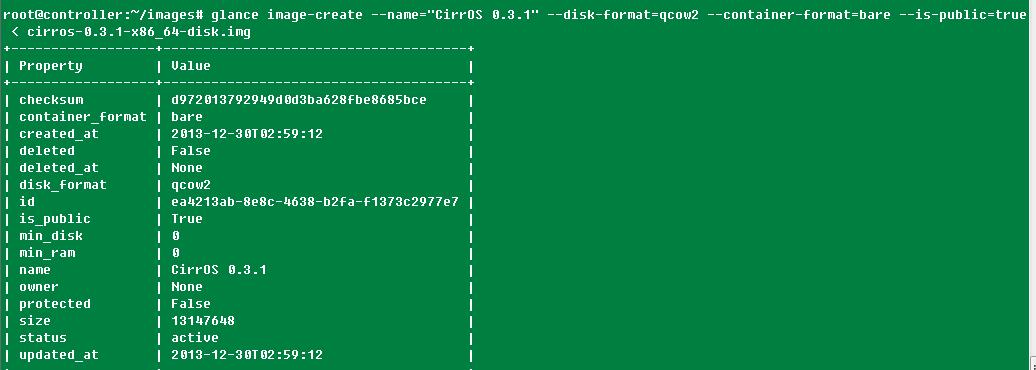
# cd images/

# wget http://cdn.download.cirros-cloud.net/0.3.1/cirros-0.3.1-x86\_64-disk.img



2、上传镜像到镜像服务

# 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



选项说明：

# glance image-create --name=imageLabel --disk-format=fileFormat   --container-format=containerFormat --is-public=accessValue < imageFile

imageLable:镜像的名称，随意写。

fileFormat:指定图像的格式，有效的格式为qcow2,raw,vhd,vmdk,vdi,iso,aki,ari和ami

查看格式：

wKiom1LLTVCg4lzmAAAllYdteq0778.jpg

containerFormat:指定container的格式,有效的格式有bare,ovf,aki,ari 和ami

imageFile：指定镜像文件

3、查看镜像列表



**五、OpenStack安装Nova**

计算服务

计算服务是云算机控制器，它是IaaS系统的主要部分，用它来承载和管理云计算系统，主模块是用python来实现的，与keystone验证，对于图像服务以及dashboard和管理接口，获取镜像是通过项目和用户配额限制，如实例数量，

组件：

API

Nova-api请求和回复一个用户计算的API，支持openstack 计算API,Amazon EC2 api和Admin Api对特权用户执行管理操作，另外启动最新的任务，如运行一个实例，安装一个虚拟机

Nova-api-metadata 接受来自实例的元数据请求

Comput core

Nova-comput 进程，一个守护进程，通过虚拟机管理程序的API创建和终止虚拟机实例，如xenapi for xenserver/xcp libvirt适用于kvm 或qemu vmwareapi 适用于vmware

Nova-scheduler 进程，调度，从消息队列中取一个虚拟机实例的请求，并执行在其虚拟主机上。

Nova-conductor 模式，介于nova-computer和database之间，设计目的在于消除直接nova-computer直接访问云数据库。

Networking for VMs

Nova-network 工作守护进程，类nova-computer它接受消息队列中的任务闭幕式执行，如设立桥接接口或者更改iptables规则，

Nova-dhcpbridge 脚本，跟踪IP地址租约，并通过使用dnsmasq的DHCP脚本记录在数据库中。

Console interface

Noca-consoleauth 守护进程，由用户的console控制台代理提供授权tokens，两种模式nova-novncproxy和nova-xvpnvcproxy

nova-novncproxy 守护进程，提供了VNC连接访问正在运行的实例的代表，基于novnc客户浏览器

Nova-console 已经弃用，被 nova-xvpnvncproxy取代

nova-xvpnvncproxy 守护进程，通过vnc连接访问正在运行的实例代理，支持专门设计的Openstack的java客户端

Nova-cert 守护进程，管理x509证书

Image management

nova-objectstore 提供了一个用于注册S3接口的镜像服务，主要必须支持euca2ools安装，nova-objectstore转换S3请求镜像服务的请求

Euca2ools 客户端，一组命令解释器来管理云资源，可以配置nova-api来支持EC2接口。

Command-line clients and other interfaces

Nova 客户端 使用户、租客管理员提交命令

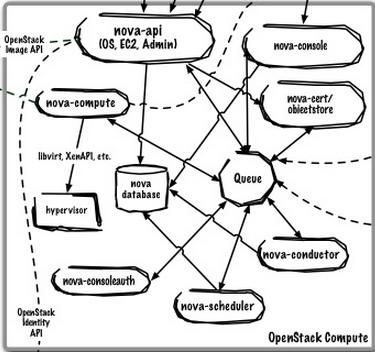
Nova-manage 客户端，使云管理员提交命令

Other components

The queue 消息队列 rabbitmq,apache qpid zeromq

Sql database 数据库mysql,sqlite,PostgreSQL

逻辑图：



安装

可以配置在一个节点或多个节点。

1、安装包

# apt-get install nova-novncproxy novnc nova-api   nova-ajax-console-proxy nova-cert nova-conductor   nova-consoleauth nova-doc nova-scheduler   python-novaclient

2、配置连接数据库/etc/nova/nova.conf和注册keystone服务

[database]

connection = mysql://nova:stack@controller/nova

[keystone\_authtoken]

auth\_host = controller

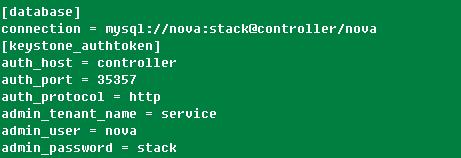
auth\_port = 35357

auth\_protocol = http

admin\_tenant\_name = service

admin\_user = nova

admin\_password = stack



3、配置使用消息队列rabbitmq

rpc\_backend = nova.rpc.impl\_kombu

rabbit\_host = controller

rabbit\_password = rabbit

wKioL1LGdWqSb0JnAAAWQEOva0s728.jpg

4、删除默认数据库

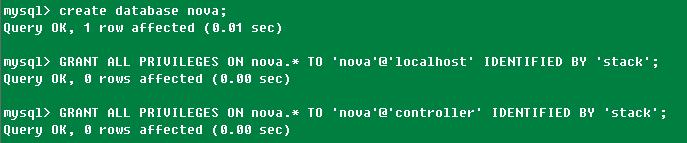
# rm /var/lib/nova/nova.sqlite

5、创建数据库以及授权访问

mysql> create database nova;

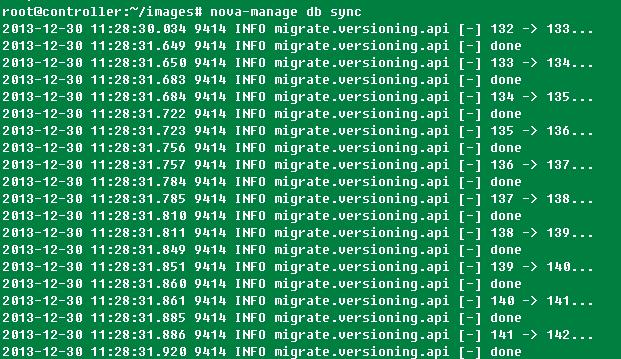
mysql> GRANT ALL PRIVILEGES ON nova.\* TO 'nova'@'localhost' IDENTIFIED BY 'stack';

mysql> GRANT ALL PRIVILEGES ON nova.\* TO 'nova'@'controller' IDENTIFIED BY 'stack';

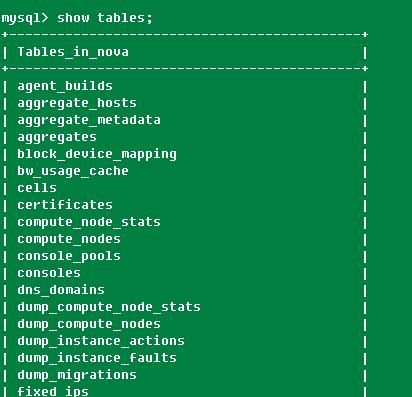


6、同步计算服务数据表

# nova-manage db sync



查看数据库的表



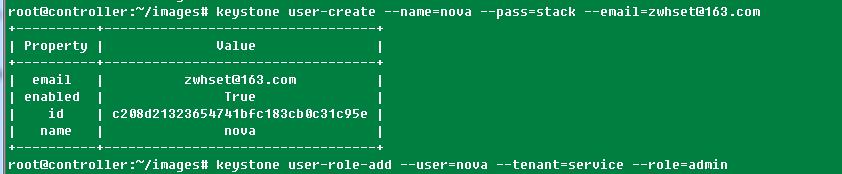
7、设制自己的IP以及其它VNC监控的项



8、在keystone中注册nova

# keystone user-create --name=nova --pass=stack [--email=zwhset@163.com](mailto:--email=zwhset@163.com)

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



9、配置使用keystone验证以及更改nova\_pass密码 [DEFAULT]项里

auth\_strategy=keystone

wKioL1LGdcjTomj_AAAIpKJ6E40332.jpg

10、添加认证方式和密码等文件/etc/nova/api-paste.ini

[filter:authtoken]

paste.filter\_factory = keystoneclient.middleware.auth\_token:filter\_factory

auth\_host = controller

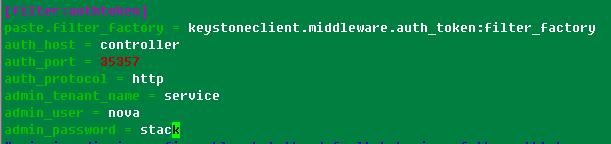
auth\_port = 35357

auth\_protocol = http

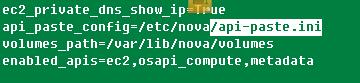
admin\_tenant\_name = service

admin\_user = nova

admin\_password = stack

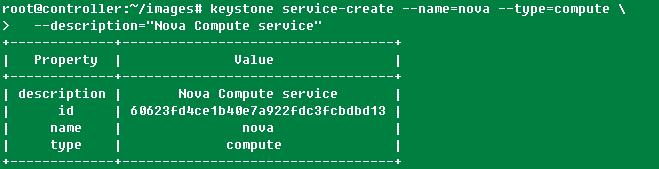


注意：在nova.conf文件里有这个文件的指向：



11、注册keystone服务让其它服务找到他，并且创建终端。

# keystone service-create --name=nova --type=compute --description="Nova Compute service"



使用刚才的ID创建终端，

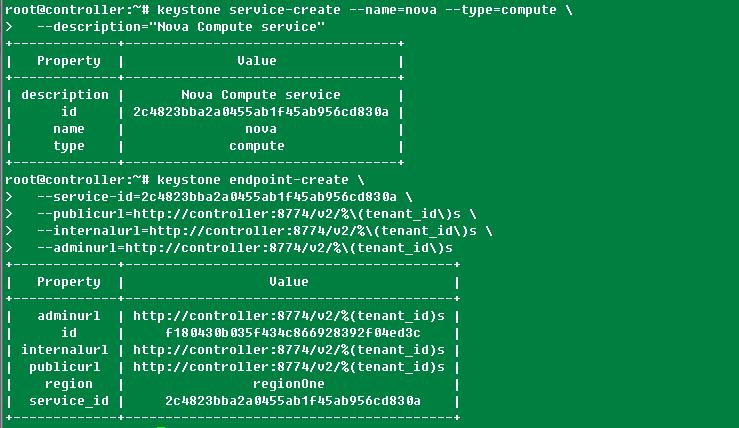
keystone endpoint-create \

>   --service-id=2c4823bba2a0455ab1f45ab956cd830a \

>   --publicurl=http://controller:8774/v2/%\(tenant\_id\)s \

>   --internalurl=http://controller:8774/v2/%\(tenant\_id\)s \

>   --adminurl=http://controller:8774/v2/%\(tenant\_id\)s



12、重启计算服务

# service nova-api restart

# service nova-cert restart

# service nova-consoleauth restart

# service nova-scheduler restart

# service nova-conductor restart

# service nova-novncproxy restart



13、列出计算可用的镜像



**六、OpenStack配置计算结点**

在另一台机器上配置计算结点：

IP：192.168.200.200

Hosts:

$ sudo vim /etc/hosts

[](http://s3.51cto.com/wyfs02/M02/0A/03/wKioL1LMt3mTteCjAAAhyBzYy6U812.jpg)

修改主机名：

[wKiom1LMt4bBf_NHAAASSVsOc2s155.jpg](http://s3.51cto.com/wyfs02/M01/0A/03/wKiom1LMt4bBf_NHAAASSVsOc2s155.jpg)

1、安装包

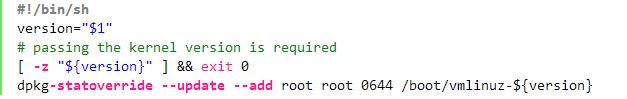
apt-get install nova-compute-kvm python-guestfs

2、有个BUG，你必须读取当前kernel

# dpkg-statoverride  --update --add root root 0644 /boot/vmlinuz-$(uname -r)

或者使用这个来覆盖以后的内核更新,并把文件弄成可执行

#cat /etc/kernel/postinst.d/statoverride

[](http://s3.51cto.com/wyfs02/M00/0A/03/wKioL1LMt3rC6rvcAAAtDYH_cVk688.jpg)

#  chmod +x /etc/kernel/postinst.d/statoverride

3、编辑配置文件注册keystone和数据库连接方式,

#cat /etc/nova/nova.conf

[](http://s3.51cto.com/wyfs02/M02/0A/03/wKiom1LMt4agUUzNAAAdHeWpp1A405.jpg)

4、使用消息队列，使用[DEFAULT]项

#cat /etc/nova/nova.conf

[wKioL1LMt3rz0WDoAAAVLH74ZFA952.jpg](http://s3.51cto.com/wyfs02/M01/0A/03/wKioL1LMt3rz0WDoAAAVLH74ZFA952.jpg)

5、提供远程控制台的使用[DEFAULT]项

#cat /etc/nova/nova.conf

[](http://s3.51cto.com/wyfs02/M00/0A/03/wKiom1LMt4bwp-arAAAor5bIXR0577.jpg)

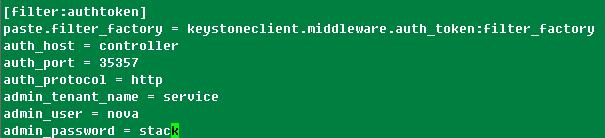
6、指定镜像服务的地址[DEFAULT]项

#cat /etc/nova/nova.conf

[wKioL1LMt-ryaDxBAAAJYf5kBso991.jpg](http://s3.51cto.com/wyfs02/M02/0A/03/wKioL1LMt-ryaDxBAAAJYf5kBso991.jpg)

7、添加凭据

# cat /etc/nova/api-paste.ini

[](http://s3.51cto.com/wyfs02/M02/0A/03/wKioL1LMt3yx-BqlAAA_GoTxAJQ300.jpg)

8、删除表,也是因为ubuntu默认使用sqlite有关系

# rm /var/lib/nova/nova.sqlite

9、重启服务

[wKiom1LMt4jARiIpAAAhp0X77po102.jpg](http://s3.51cto.com/wyfs02/M01/0A/03/wKiom1LMt4jARiIpAAAhp0X77po102.jpg)

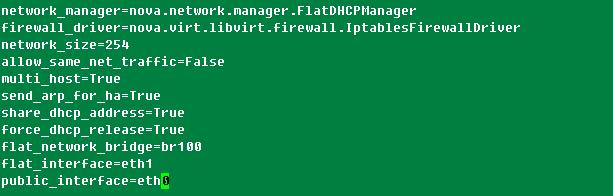
启用网络：

配置网络用DHCP，使用了多主机功能，主要是怕网络控制器成为一个单点故障，所以每个计算结点都配置网络。

1、安装nova-network的包，不用在控制节点也安装这个包。

# apt-get install nova-network

2、编辑nova.conf文件定义网络模式，还是默认项，eth1做内网，eth0做外网。



3、重启网络服务

wKiom1LMt4myEVHXAAAgnvruyfQ816.jpg

4、创建虚拟机可用的网络，只要运行一次就好，不用在每个机器上运行

wKioL1LMt3yCcPdPAAAyp4z0xkA735.jpg

出错了，因为我们重启了虚拟主机，环境变量失效了，我们重新定义下。

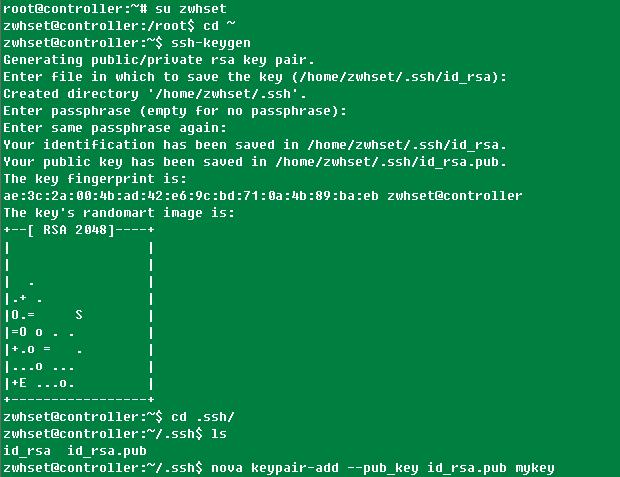


wKioL1LMt3yg8s3lAAAkiLufsI8149.jpg

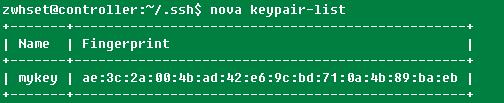
启动一个实例

我们来创建一个低资源的实例

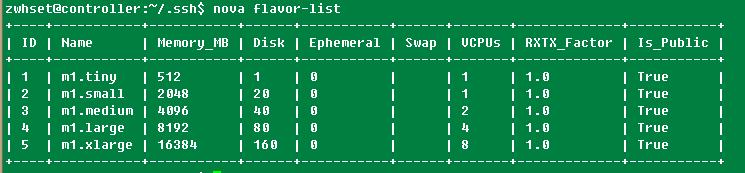
1、无密码登陆



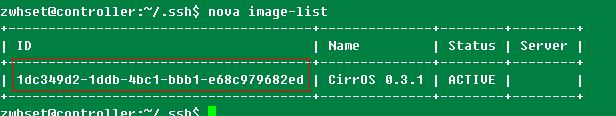
查看密钥对



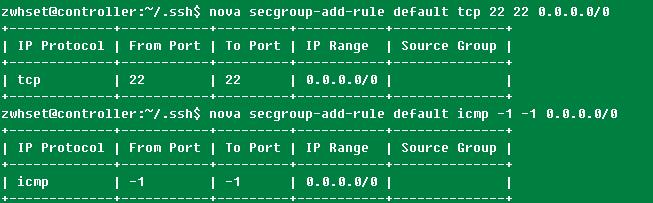
2、是创建一个实例，你必须要知道flavor的ID，他显示了你可以用的CPU，内存等信息，查看有哪些flavor(flavor 就跟菜单一样，你选哪个菜单由你决定)



3、获取镜像ID用来创建实例



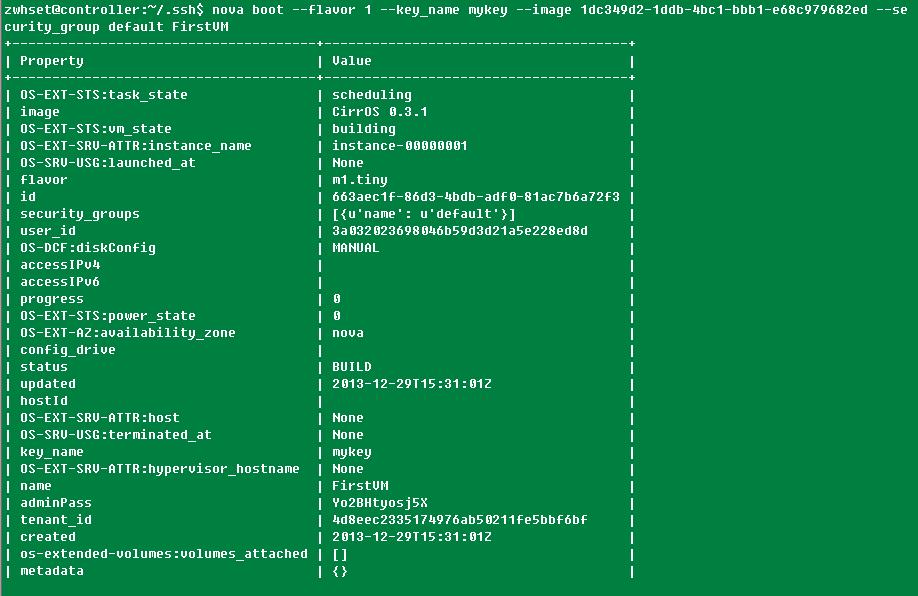
4、要使用ssh和ping就必须要配置安全组的规则



5、启动实例

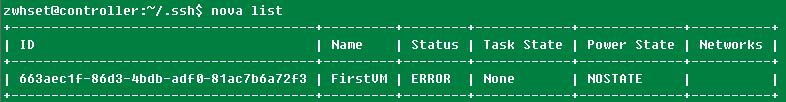
nova boot --flavor 1 --key\_name mykey --image 1dc349d2-1ddb-4bc1-bbb1-e68c979682ed --security\_group default FirstVM

格式： nova boot --flavor flavorType --key\_name keypairName --image ID newInstanceName

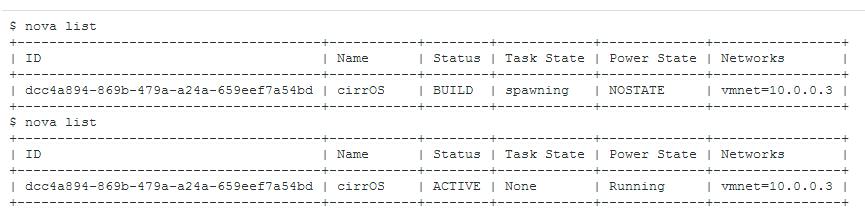


注意：如果没有足够的可用内存，虚拟机可以创建，但不能启动。

6、使用nova list来查看虚拟机实例，状态分为两种build和active



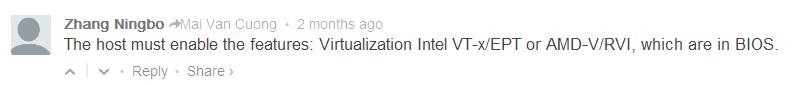
正常状态：

[](http://s3.51cto.com/wyfs02/M00/0A/03/wKiom1LMt4vTaxTMAACGU65ZN6U726.jpg)

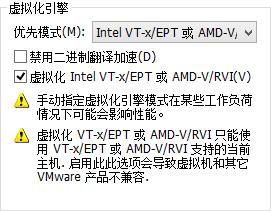
查看nova-computer日志发现

wKioL1LMt37QGt84AABSdVot0dY639.jpg

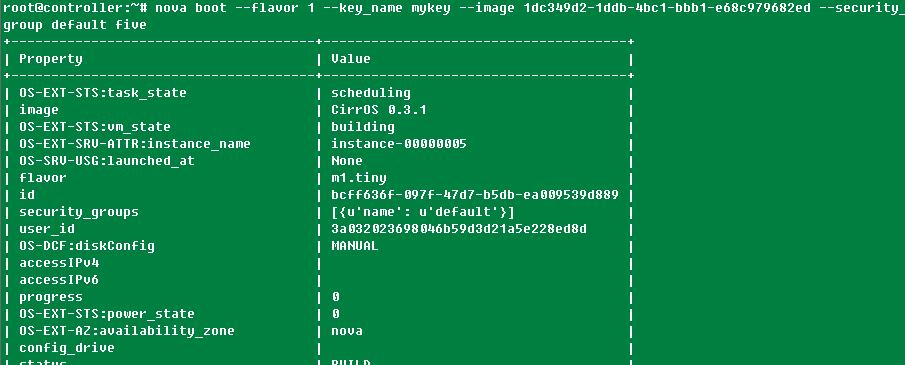
有人回复主机必须启用VT-x/EPT，我们是虚拟机做的测试，那么我们尝试更改虚拟机选项。

[](http://s3.51cto.com/wyfs02/M02/0A/03/wKiom1LMuUyDjYmWAAArSqNtgn4832.jpg)

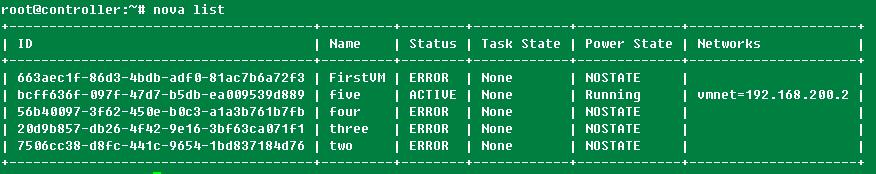
我们在虚拟机上启用试试：

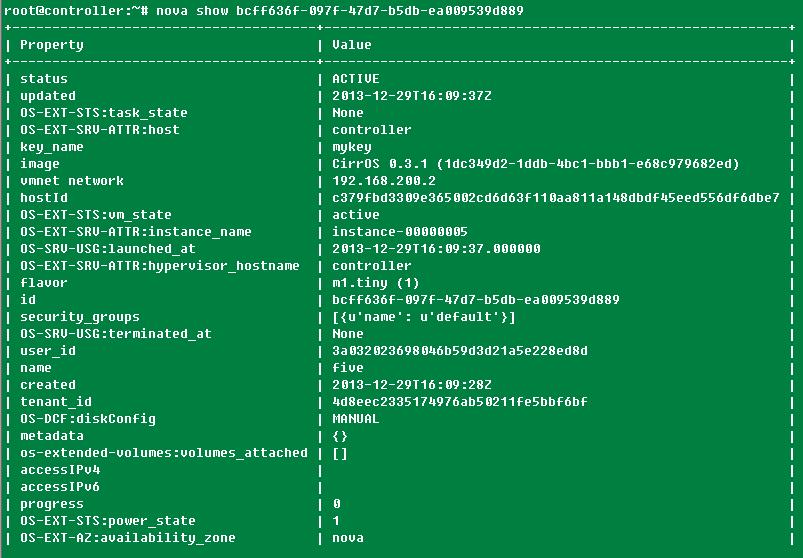


再次创建一个



现在就跑起来了，同样我们可以查看详细信息





7、经过实例启动并初始化并且配置了安全组后，你可以使用nova boot 命令指定密钥对到该实例，使用nova list获取实例的IP地址，你并不要私钥，因为它是存储在默认的位置~/.ssh/.id\_rsa

注意，如果是使用cirrOS image，你就必须用cirros登陆，而不是用root,你不需要ssh key可以用 cubswin:) 作为密码登陆

Ssh cirros@192.168.200.

**对Python-memcache分布式散列和调用的实现**

**煮酒品茶：对python-memcache进行学习，把分布式HASH算法加进去，不说线上自己玩玩的程序可以加到里面去。memcached读存数据就这些东西，看着补。**

分布式一致性HASH算法：memcache\_ring.py

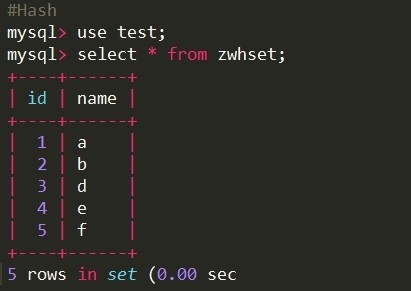
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | #coding:utf8  import hash\_ring  import memcache  memcache\_servers = [          '127.0.0.1:11211']  weights = {          '127.0.0.1:11211':1}  ring = hash\_ring.HashRing(memcache\_servers,weights)  #if value is null then get else set  def mc(key,value="Null-0"):      server\_node = ring.get\_node(key)      mc = memcache.Client([server\_node],debug=1)      if value == "Null-0":          return mc.get(key)      else:          return mc.set(key,value) |

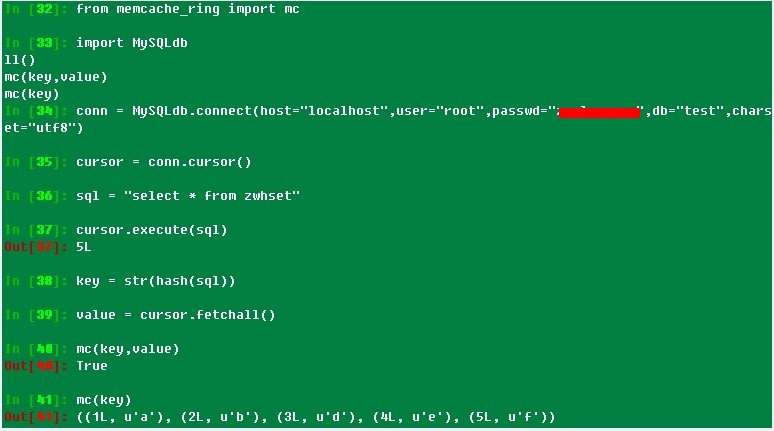
一致性hash读取数据：

[](http://s3.51cto.com/wyfs02/M00/11/7D/wKioL1LSeDuRXANVAADW8yi1_eA271.jpg)

从数据库中读取数据

sql:

[](http://s3.51cto.com/wyfs02/M02/11/82/wKioL1LTX1vTNIezAAB9FTHMdz0194.jpg)

[](http://s3.51cto.com/wyfs02/M01/11/7D/wKiom1LSeC-xCtr5AAG-lqNwfrk454.jpg)

可利用起来的程序，稍改动加一些try之类的就可以用到自己的程序玩了。else下key **=**str(hash(sql))可以去掉。不知道为啥不好册。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | #coding:utf8  from memcache\_ring import mc  import MySQLdb  #如果在memcache中就不查数据库，不在就取出来并存一份  sql = "select \* from zwhset where id=100"  key = str(hash(sql))  #查数据库  def select\_sql(sql):      conn = MySQLdb.connect(host="localhost",user="root",passwd="",db="test",charset="utf8")      cursor = conn.cursor()      cursor.execute(sql)      value = cursor.fetchall()      #如果没有查到数据，则原值返回      if not value:          return value      else:          key = str(hash(sql))          #把存储的结果给调用程序          return mc(key,value)  #读数据，先看memcached里有没有，有就直接返回memcached查的值，没有就查数据库，  #如果数据库也返回空的话原值返回，如果有值就写memcached,然后把value返回  if not mc(key):      select\_sql(sql)  else:      mc(key) |

**Openstack-make-centos-image**

1、建立虚拟机（硬盘镜像选对）

2、在虚拟机上做一些云的要求（有脚本）

3、上传镜像

4、测试

|  |
| --- |
| #!/bin/bash  # add the EPEL repo and update  rpm -Uvh http://download.fedoraproject.org/pub/epel/6/x86\_64/epel-release-6-8.noarch.rpm  yum -y update  # install cloud-init  yum -y install cloud-init  yum -y install rsync  # add the ec2-user (default for cloud-init)  adduser ec2-user  mkdir /home/ec2-user/.ssh/  chown -R ec2-user.ec2-user /home/ec2-user/.ssh  # patch up groups  sed -i '/^wheel:/ s/$/ec2-user/' /etc/group  # fix up sudoers  sed -i '/Defaults    requiretty/d' /etc/sudoers  sed -i '/## Same thing without a password/{n;d}' /etc/sudoers  sed -i '/# Same thing without a password/a \  %wheel   ALL=(ALL)   NOPASSWD: ALL' /etc/sudoers  # hack up sshd\_config  sed -i 's/PasswordAuthentication yes/PasswordAuthentication no/g' /etc/ssh/sshd\_config  # clean up the network interface stuff  rm /etc/udev/rules.d/70-persistent-net.rules  sed -i '/HWADDR/d' /etc/sysconfig/network-scripts/ifcfg-eth0  sed -i '/UUID/d' /etc/sysconfig/network-scripts/ifcfg-eth0  # graft up grub  sed -i 's/timeout=5/timeout=1/g' /boot/grub/menu.lst  sed -i '/hiddenmenu/a \  serial –unit=0 –speed=115200 \  terminal –timeout=10 console serial' /boot/grub/menu.lst  sed -i '/^\skernel/ s/$/ console=tty0 console=ttyS0,115200n8/' /boot/grub/menu.lst  # wipe the passwords  passwd -l root  passwd -l ec2-user  # say something cute in /etc/motd  echo "CentOS image built using BlueChipTek's OpenStack guide." >> /etc/motd  echo "" >> /etc/motd  echo "More guides on OpenStack are at http://openstack.bluechiptek.com/" >> /etc/motd  echo "" >> /etc/motd  echo "@StackGeek" >> /etc/motd  # notify we're halting  echo "Halting instance in 5 seconds!"  sleep 5  # halt the instance  halt |

**OpenStack踩坑记录**

Ubuntu 12.04:不能创建用户和配额的问题：

bug页：https://bugs.launchpad.net/openstack-cisco/+bug/1167483

# vim /usr/share/openstack-dashboard/openstack\_dashboard/local/local\_settings.py

将值调整 OPENSTACK\_KEYSTONE\_DEFAULT\_ROLE = "\_member\_" 原来的值是"MEMBER"