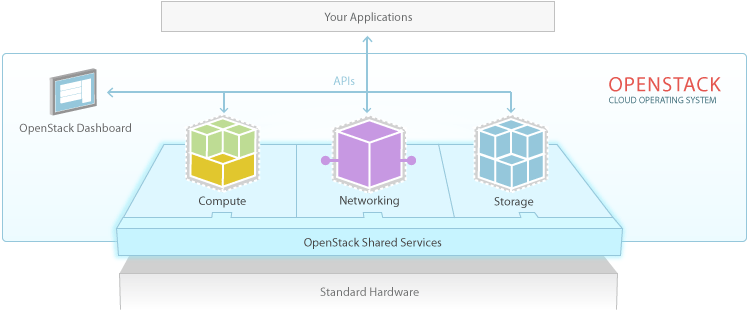
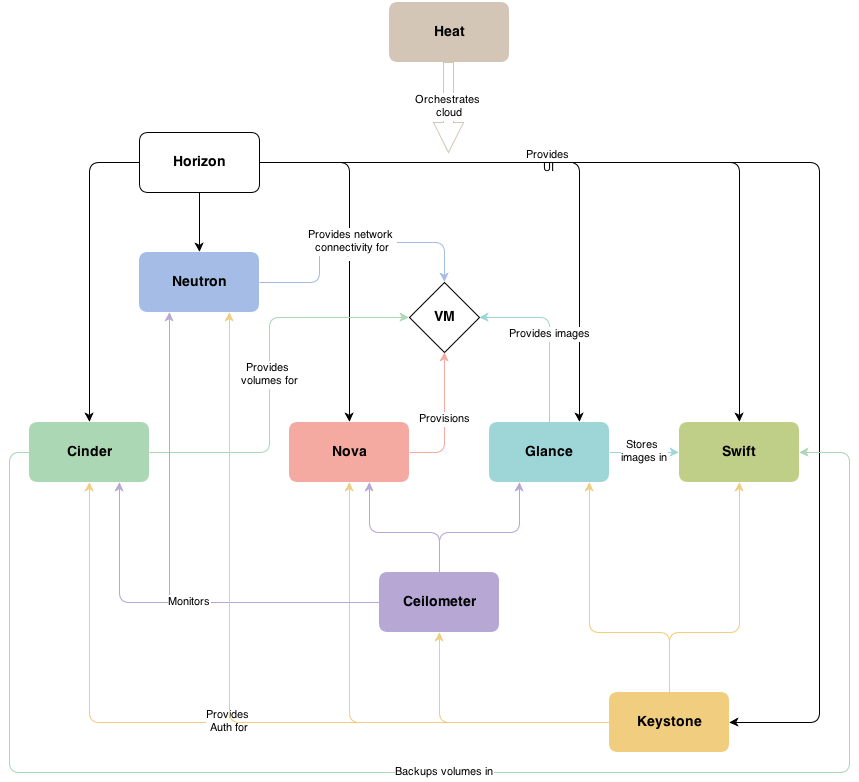
# OpenStack

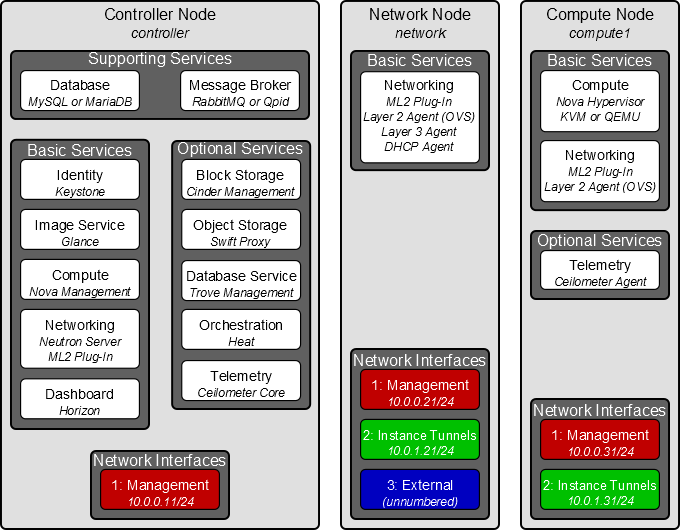
## 概述



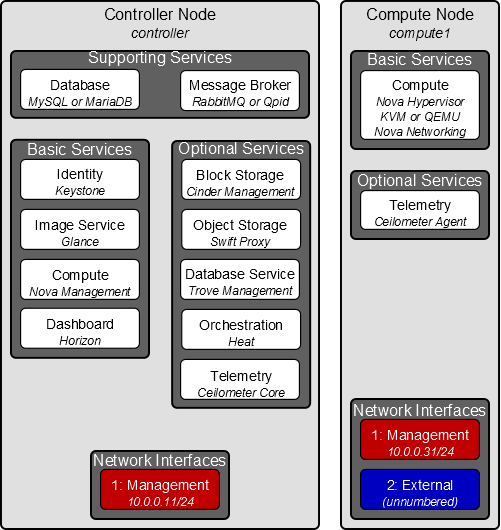
|  |  |  |
| --- | --- | --- |
| Service | Project name | Description |
| Dashboard | Horizon | 与底层OpenStack服务交互的Web应用，例如启动实例，分配IP，配置访问权限等 |
| Compute | Nova | 管理实例的生命周期，比如根据要求分配、调度、撤销虚拟机 |
| Networking | Neutron | 让网络连接成为一个服务，为用户提供API以定义网络。采用可插拔结构以支持多个网络生产商和技术 |
| Storage | | |
| Object Storage | Swift | 通过一个RESTful，基于HTTP的API保存和回收非结构化数据，通过数据备份和可扩展结构实现高容错性，不同于挂载目录的文件系统 |
| Block Storage | Cinder | 提供持久化块存储以运行实例，可插拔结构简化块存储设备的分配和管理 |
| Shared services | | |
| Identity service | Keystone | 为其他OpenStack服务提供授权和认证服务，提供所有服务的endpoint目录 |
| Image Service | Glance | 保存和回收虚拟机磁盘镜像，OpenStack计算节点在分配实例中用到该功能 |
| Telemetry | Ceilometer | 监控和计算OpenStack的收费，可扩展性，统计数据 |
| Higher-level services | | |
| Orchestration | Heat | 使用原生HOT模板或AWS云格式组织各云应用组件，通过OpenStack-native REST API或CloudFormation-compatible Query API |
| Database Service | Trove | 提供可扩展的稳定的云数据库 |



三节点架构：采用neutron配置网络



2节点架构：采用传统网络



### 环境变量

访问服务需提供认证信息，认证信息可通过命令行--os-xxx方式设置或环境变量设置

认证信息两种设置方式分别为token或user name、 password。

**问题：ERROR (CommandError): You must provide a username or user id via --os-username, --os-user-id, env[OS\_USERNAME] or env[OS\_USER\_ID]**

vi ~/.bash\_profile

export OS\_USERNAME=admin

export OS\_PASSWORD=ADMIN\_PASS

export OS\_TENANT\_NAME=admin

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

或者

$ export OS\_SERVICE\_TOKEN=ADMIN\_TOKEN

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

source ~/.bash\_profile

nova list

### openstack-config

设置配置文件内容

openstack-config –set 文件路径/文件名 组名 配置项 内容

openstack-config –set /etc/glance/glance-api.conf database connection mysql://glance:GLANCE\_DBPASS@controller/glance

### Identity Service

所有服务都必须在Identity Service注册

* keystone user-create：创建用户
* keystone user-list：显示所有用户
* keystone role-create：创建角色
* keystone tenant-create：创建租户
* keystone user-role-add：将用户赋予角色和租户
* keystone service-create：描述服务
* keystone service-list：查看所有注册服务
* keystone endpoint-create：API endpoints与service关联

keystone user-create --name=admin --pass=ADMIN\_PASS --email=ADMIN\_EMAIL

keystone role-create --name=admin

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

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

keystone service-create –name=keystone --type=identity --description=”xxxx”

keystone endpoint-create --service-id=$(keystone service-list | awk ‘/ identity / {print $2}’) --publicurl=http://controller:5000/v2.0

--internalurl=http://controller:5000/v2.0

--adminurl=http://controller:35357/v2.0

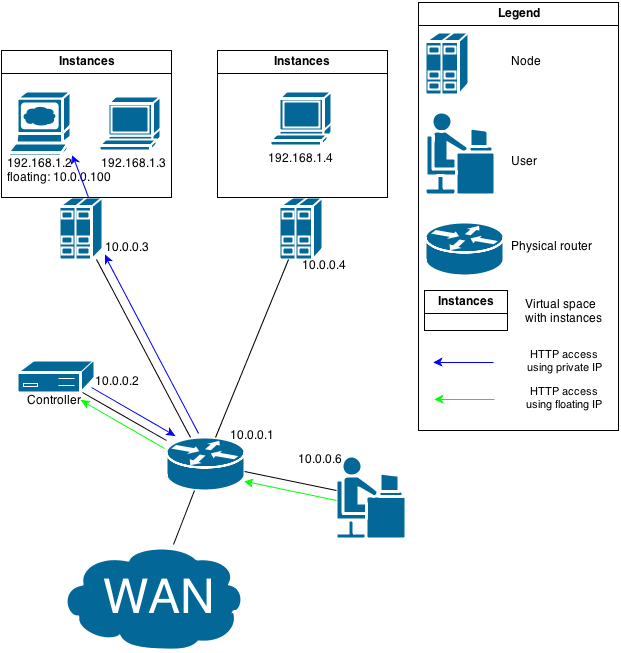
### Image Service

* glance image-create：创建镜像
* glance image-list：显示所有镜像

### Private IP vs Floating IP

Private IP通过DHCP分配至端口，通过ip a可查看，**私有网络中实例间通信**（通过虚拟交换机L2 agent在计算节点间）

Floating IP通过Neutron提供，Neutron L3 agent转发包到端口，**外部可通过floating ip访问实例**。

[](https://www.rdoproject.org/File:Neutron_private_floaring_ip.png)

## 安装

### devstack

**devstack快速安装openstack开发环境，重启后失效需重新安装。**

[root@localhost ~]$ systemctl stop firewalld.service 关闭防火墙

[root@localhost ~]$ systemctl disable firewalld.service

[root@localhost ~]$ yum install -y git

[root@localhost ~]$ adduser stack

[root@localhost ~]$ echo 'stack ALL=(ALL) NOPASSWD:ALL' >> /etc/sudoers

[root@localhost ~]$ su - stack

[stack @localhost ~]$ git clone <https://git.openstack.org/openstack-dev/devstack>

[stack@localhost ~]$ cd devstack

[stack @localhost devstack]$ ./stack.sh

过程中输入各种密码

登录浏览器输入<http://ip>，用户名是 admin，密码是刚才设定的密码，

unstack.sh：停止所有openstack服务，除了mysql和rabbitmq

clean.sh：停止服务并清除所有数据和文件。

可以mv ~/devstack/sample/local.conf ~/devstack

修改后执行./stack.sh

**问题：cp: cannot create regular file ‘/etc/glance/glance-registry-paste.ini’: Permission denied**

之前安装过devstack，用户变更导致，确保当前用户在/etc/sudoers中，且rm -fr /etc/glance

**问题：No valid host was found. There are not enough hosts available**

检查logs目录下的出错信息，原因是BIOS没有开启KVM，kvm模块未加载

<https://wiki.ubuntu.com/kvm>

http://docs.openstack.org/juno/config-reference/content/kvm.html

**问题：WARNING: this script has not been tested on trusty**

**[ERROR] ./stack.sh:137 If you wish to run this script anyway run with FORCE=yes**

FORCE=yes ./stack.sh

**问题：虚拟机显示的IP与实际分配IP不同，网络ping不通**

#### local.conf

local.conf安装前在devstack目录下创建，参考devstack/samples/local.conf，采用INI配置格式：

[[ <phase> | <config-file-name> ]]

phase是stack.sh执行的阶段名，config-file-name是配置文件路径。

phase包括：

* local：执行stackrc前从local.conf提取出localrc；
* pre-install：安装系统包之后，源代码之前；
* install：安装源代码后；
* post-config：配置layer 2 services之后启动之前；
* extra：启动服务之后，提取extra.d中的文件之前。

[[local|localrc]]

FLOATING\_RANGE=192.168.1.224/27

FIXED\_RANGE=10.11.12.0/24

FIXED\_NETWORK\_SIZE=256

FLAT\_INTERFACE=eth0

ADMIN\_PASSWORD=supersecret

MYSQL\_PASSWORD=iheartdatabases

RABBIT\_PASSWORD=flopsymopsy

SERVICE\_PASSWORD=iheartksl

SERVICE\_TOKEN=xyzpdqlazydog

         (1) FLOATING\_RANGE:  设置虚拟机自动分配IP的范围。例如 192.168.1.56/29，是指创建虚拟机时会分配（192.168.1.56-63，但只有57-62可用）内的IP。

         (2) FIXED\_RANGE 和 FIXED\_NETWORK\_SIZE:配置instances实例使用的内部地址空间。

         (3) FLAT\_INTERFACE: 设置连接到本地网络的以太网接口。

         (4) ADMIN\_PASSWORD: admin 和 demo 帐户用这个密码来登陆，建立OpenStack用户。

         (5) MYSQL\_PASSWORD: 设置mysql的密码，当你需要直接查看数据库时使用。

         (6) RABBIT\_PASSWORD:  设置RabbitMQ密码

         (7) SERVICE\_PASSWORD: openstack service(nova, glance,..) 用keystone认证使用

#### 问题

**问题1：xxx is not in the sudoers file. This incident will be reported.**

[root@localhost ~]$ echo "stack ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers

**问题2：服务器重启，如何恢复devstack安装的openstack？**

**先unstack.sh，重启后执行rejoin-stack.sh**

stack.sh运行时会创建stack-screenrc文件，保存所有服务启动的命令，重启后只要执行rejoin-stack.sh，它会读取stack-screenrc重启服务。

### packstack

**rdo快速安装openstack生产环境，只支持RHEL-based和x86\_64，如RedHat，CentOS。**

**建议采用全新的系统安装OpenStack**

[root@localhost ~]# yum update -y

配置网络后要重启

[root@localhost ~]# reboot

[root@localhost ~]# yum install -y <https://rdo.fedorapeople.org/rdo-release.rpm>

[root@localhost ~]# yum install git python-yaml -y

[root@localhost ~]# yum -y install python-setuptools

[root@localhost ~]# git clone https://github.com/stackforge/packstack.git

[root@localhost ~]# cd packstack

[root@localhost packstack]# python setup.py install\_puppet\_modules

[root@localhost packstack]# python setup.py install

[root@localhost packstack]# packstack --allinone

安装完成后，重启机器，访问http://ip/dashboard

用户名admin，密码

**问题1：**

**Preinstalling Puppet and discovering hosts' details[ ERROR ]**

**ERROR : Failed to run remote script, stdout: Loaded plugins: fastestmirror, langpacks, priorities**

**yum install puppet-server**

[root@localhost ~]# yum install puppet-server

**问题2：**

**Applying 192.168.1.4\_prescript.pp**

**192.168.1.4\_prescript.pp: [ ERROR ]**

**Applying Puppet manifests [ ERROR ]**

没有启动NetworkManager

### 普通安装

#### 准备工作

1. **基本硬件要求**

* Controller Node: 1 processor, 2 GB memory, and 5 GB storage
* Network Node: 1 processor, 512 MB memory, and 5 GB storage
* Compute Node: 1 processor, 2 GB memory, and 10 GB storage

计算节点安装64位系统，否则起64位image时会失败。

1. **networking**

关闭NetworkManager，启动network

# service NetworkManager stop

# service network start

# chkconfig NetworkManager off

# chkconfig network on

关闭iptables

# service iptables stop

# chkconfig iptables off

#vi /etc/sysconfig/network

HOSTNAME=controller

#vi /etc/hosts //确保能通过hostname相互访问

192.168.214.146 controller

192.168.214.145 compute

互ping一下

ping -c 4 compute

1. **NTP**

所有计算节点从controller上获取时间

# yum install ntp

controller# vi /etc/ntp.conf

restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

compute # vi /etc/ntp.conf

server 192.168.214.146 prefer

# service ntpd start

# chkconfig ntpd on

1. **Passwords**

|  |  |
| --- | --- |
| Password name | Description |
| Database password (no variable used) | Root password for the database |
| KEYSTONE\_DBPASS | Database password of Identity service |
| DEMO\_PASS | Password of user demo |
| ADMIN\_PASS | Password of user admin |
| GLANCE\_DBPASS | Database password for Image Service |
| GLANCE\_PASS | Password of Image Service user glance |
| NOVA\_DBPASS | Database password for Compute service |
| NOVA\_PASS | Password of Compute service user nova |
| DASH\_DBPASS | Database password for the dashboard |
| CINDER\_DBPASS | Database password for the Block Storage service |
| CINDER\_PASS | Password of Block Storage service user cinder |
| NEUTRON\_DBPASS | Database password for the Networking service |
| NEUTRON\_PASS | Password of Networking service user neutron |
| HEAT\_DBPASS | Database password for the Orchestration service |
| HEAT\_PASS | Password of Orchestration service user heat |
| CEILOMETER\_DBPASS | Database password for the Telemetry service |
| CEILOMETER\_PASS | Password of Telemetry service user ceilometer |
| TROVE\_DBPASS | Database password of Database service |
| TROVE\_PASS | Password of Database Service user trove |

1. **MySQL**

controller# yum install -y mysql mysql-server MySQL-python

controller# vi /etc/my.cnf

bind-address=0.0.0.0

default-storage-engine = innodb

collation-server = utf8\_general\_ci

init-connect = 'SET NAMES utf8'

character-set-server = utf8

controller# service mysqld start

controller# chkconfig mysqld on

删除MySQL的匿名用户

controller# mysql\_install\_db

controller# mysql\_secure\_installation

输入root密码时直接按enter，其余全部选Y

compute# yum install -y MySQL-python

1. **OpenStack packages**

# yum install <http://repos.fedorapeople.org/repos/openstack/openstack-icehouse/rdo-release-icehouse-3.noarch.rpm>

# yum install <http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm>

# yum install openstack-utils

# yum install openstack-selinux

# yum upgrade

1. **Messaging server**

controller# yum install qpid-cpp-server

controller # vi /etc/qpidd.conf

auth=no

controller # service qpidd start

controller # chkconfig qpidd on

#### Identity Service

##### 概述

Identity Service作用有两点：

* 用户管理。跟踪用户和权限。
* 服务目录。提供可用服务的API endpoints目录。

User：用户，Identity Service验证请求来源，User登录后得到tokens以访问资源，User可分配到Tenant中。

Credentials：证书，可以是user name和password，user name和API key，tokens。

Authentication：认证，Identity验证用户提供的Credential。用户通过user name和password，user name和API key获取Identity Service颁发的token，用于后续请求。

Token：用于访问资源的二进制文本，描述可访问资源的范围。可被随时回收或在一定期限内有效。

Tenant：容器用于分组或隔离资源，类似于组的概念。

Service：OpenStack服务。提供一个或多个endpoint，让用户可以访问资源执行操作。

Endpoint：URL格式的网络接入地址，用于访问服务。

Role：用户角色，允许用户执行特定操作，某角色下的用户将继承该角色的权限。

##### 安装

1. **安装Identity Service**

以下操作都在controller上执行

# yum install openstack-keystone python-keystoneclient

1. **配置MySQL**

#openstack-config --set /etc/keystone/keystone.conf database connection mysql://keystone:123456@controller/keystone

1. **创建数据库**

# mysql -u root –p

mysql> CREATE DATABASE keystone;

mysql> GRANT ALL PRIVILEGES ON keystone.\* TO 'keystone'@'localhost' \

IDENTIFIED BY 'KEYSTONE\_DBPASS';

mysql> GRANT ALL PRIVILEGES ON keystone.\* TO 'keystone'@'%' \

IDENTIFIED BY 'KEYSTONE\_DBPASS';

mysql> exit

1. **创建数据库表**

# su -s /bin/sh -c "keystone-manage db\_sync" keystone

随机生成token，注册该token作为identity service和其他服务交互的密码

# ADMIN\_TOKEN=$(openssl rand -hex 10)

# echo $ADMIN\_TOKEN

9142f6a44c94e351ba48 //记住该token

# openstack-config --set /etc/keystone/keystone.conf DEFAULT admin\_token $ADMIN\_TOKEN

默认情况下Keystone采用PKI token，生成注册key和认证，并限制数据访问权限

# keystone-manage pki\_setup --keystone-user keystone --keystone-group keystone

# chown -R keystone:keystone /etc/keystone/ssl

# chmod -R o-rwx /etc/keystone/ssl

# service openstack-keystone start

默认情况下，Identity Service会永久保存过期tokens，这回导致数据库变大，降低性能，应定期清理过期tokens

# (crontab -l 2>&1 | grep -q token\_flush) || echo '@hourly /usr/bin/keystone-manage token\_flush >/var/log/keystone/keystone-tokenflush.log 2>&1' >> /var/spool/cron/root

设置users，tenants，roles

由于一开始没有注册任何用户，因此必须使用上一步生成的tokens来生成第一个用户，可用命令加—os-token=ADMIN\_TOKEN或者设置环境变量

设置Identity Service所需的环境变量

# vi .bash\_profile

export OS\_SERVICE\_TOKEN=9142f6a44c94e351ba48

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

# source .bash\_profile

创建一个管理员admin 的user，role和tenant

Identity Service默认创建\_member\_角色，用于访问dashboard，需将该角色付给admin

# keystone user-create --name=admin --pass=123456 [--email=331274799@qq.com](mailto:--email=331274799@qq.com)

# keystone role-create --name=admin

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

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

# keystone user-role-add --user=admin --role=\_member\_ --tenant=admin

# keystone user-create --name=demo --pass=DEMO\_PASS --email=DEMO\_EMAIL

# keystone tenant-create --name=demo --description="Demo Tenant"

# keystone user-role-add --user=demo --role=\_member\_ --tenant=demo

新增一个名叫service的tenant，用于其他服务的注册

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

所有服务必须在Identity Service中通过keystone service-create和keystone endpoint-create注册，Identity Service自己也要注册，利用环境变量OS\_SERVICE\_TOKEN授权。

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

# keystone endpoint-create --service-id=$(keystone service-list | awk '/ identity / {print $2}')

--publicurl=http://controller:5000/v2.0 --internalurl=http://controller:5000/v2.0 --adminurl=http://controller:35357/v2.0

##### 验证

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

验证Identity Service在正确的endpoint上运行且user account根据预期的凭证成功建立

# keystone --os-username=admin --os-password=123456 --os-auth-url=http://controller:35357/v2.0 token-get

验证user account在指定的tenant上有明确定义的role

# keystone --os-username=admin --os-password=123456 --os-tenant-name=admin --os-auth-url=http://controller:35357/v2.0 token-get

通过设置环境变量简化命令行参数

# vi admin-openrc.sh

export OS\_USERNAME=admin

export OS\_PASSWORD=123456

export OS\_TENANT\_NAME=admin

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

# source admin-openrc.sh

# keystone token-get

查看注册用户

# keystone user-list

# keystone user-role-list --user admin --tenant admin

#### OpenStack client

以下操作都在客户端

* ceilometer - Telemetry API
* cinder - Block Storage API and extensions
* glance - Image Service API
* heat - Orchestration API
* keystone - Identity service API and extensions
* neutron - Networking API
* nova - Compute API and extensions
* swift - Object Storage API
* trove - Database Service API

用上面的项目名称小写代替PROJECT

# yum install -y python-PROJECTclient 执行9次

# vi admin-openrc.sh

export OS\_USERNAME=admin

export OS\_PASSWORD=123456

export OS\_TENANT\_NAME=admin

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

# source admin-openrc.sh

#### Image Service

##### 概述

* glance-api：接收API请求，以发现、回收、存储镜像
* glance-registry：存储、处理、回收镜像元数据，元数据包括大小，类型等信息
* database：存储镜像元数据
* storage repository for image files：支持多种类型的存储仓库，比如普通文件系统、Object Storage、RADOS块设备、HTTP、Amazon S3.其中一些只支持只读。

##### 安装

1. **安装Image Service**

# yum install -y openstack-glance python-glanceclient

1. **配置MySQL**

# openstack-config --set /etc/glance/glance-api.conf database connection mysql://glance:123456@controller/glance

# openstack-config --set /etc/glance/glance-registry.conf database connection mysql://glance:123456@controller/glance

1. **配置Message broker**

# openstack-config --set /etc/glance/glance-api.conf DEFAULT rpc\_backend qpid

# openstack-config --set /etc/glance/glance-api.conf DEFAULT qpid\_hostname controller

1. **创建数据库**

# mysql -u root –p

mysql> CREATE DATABASE glance;

mysql> GRANT ALL PRIVILEGES ON glance.\* TO 'glance'@'localhost' IDENTIFIED BY '123456';

mysql> GRANT ALL PRIVILEGES ON glance.\* TO 'glance'@'%' IDENTIFIED BY '123456';

mysql> exit

1. **创建数据库表**

# su -s /bin/sh -c "glance-manage db\_sync" glance

1. **在Identity Service中注册用户**

# keystone user-create --name=glance --pass=123456 [--email=331274799@qq.com](mailto:--email=331274799@qq.com)

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

1. **在Image Service中注册Identity Service**

# openstack-config --set /etc/glance/glance-api.conf keystone\_authtoken auth\_uri <http://controller:5000>

# openstack-config --set /etc/glance/glance-api.conf keystone\_authtoken auth\_host controller

# openstack-config --set /etc/glance/glance-api.conf keystone\_authtoken auth\_port 35357

# openstack-config --set /etc/glance/glance-api.conf keystone\_authtoken auth\_protocol http

# 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\_user glance

# openstack-config --set /etc/glance/glance-api.conf keystone\_authtoken admin\_password 123456

# openstack-config --set /etc/glance/glance-api.conf paste\_deploy flavor keystone

# openstack-config --set /etc/glance/glance-registry.conf keystone\_authtoken auth\_uri <http://controller:5000>

# openstack-config --set /etc/glance/glance-registry.conf keystone\_authtoken auth\_host controller

# openstack-config --set /etc/glance/glance-registry.conf keystone\_authtoken auth\_port 35357

# openstack-config --set /etc/glance/glance-registry.conf keystone\_authtoken auth\_protocol http

# 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\_user glance

# openstack-config --set /etc/glance/glance-registry.conf keystone\_authtoken admin\_password 123456

# openstack-config --set /etc/glance/glance-registry.conf paste\_deploy flavor keystone

1. **在Identity Service中注册Image Service**

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

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

1. **启动Image Service**

# service openstack-glance-api start

# service openstack-glance-registry start

# chkconfig openstack-glance-api on

# chkconfig openstack-glance-registry on

##### 验证

# source admin-openrc.sh

# mkdir /tmp/images

# cd /tmp/images/

# wget <http://cdn.download.cirros-cloud.net/0.3.2/cirros-0.3.2-x86_64-disk.img>

将其上传到Image Service中

glance image-create --name=IMAGELABEL --disk-format=FILEFORMAT \

--container-format=CONTAINERFORMAT --is-public=ACCESSVALUE < IMAGEFILE

IMAGELABE：image别名

FIREFORMAT：image文件类型，有qcow2,raw,vhd,vmdk,vdi,iso.aki.ari.ami

不知道可通过file查看

CONTAINERFORMAT：容器格式，有bare,ovf,aki,ari,ami

bare是指不包含元数据

ACCESSVALUE：是否可访问，true：所有用户可查看和使用该image false：只有管理员可查可和使用

IMAGEFILE：image名称

# glance image-create --name "cirros-0.3.2-x86\_64" --disk-format qcow2 --container-format bare --is-public True --progress < cirros-0.3.2-x86\_64-disk.img

# glance image-list

# rm -rf /tmp/images //已经上传到Image Service中，可删除本地image

也可使用--copy-from直接从网络上传image

# glance image-create --name="cirros-0.3.2-x86\_64" --disk-format=qcow2 --container-format=bare --is-public=true --copy-from http://cdn.download.cirros-cloud.net/0.3.2/cirros-0.3.2-x86\_64-disk.img

#### Compute Service

##### 概述

Compute Service是IaaS的重要组成部分，用于管理云计算平台，用Python实现，需要与Identity Service进行验证，Image Service请求镜像，通过Dashboard与用户和管理员交互。

* nova-api：接受与反馈用户请求，支持OpenStack Compute API，Amazon EC2 API。初始化实例.
* nova-api-metadata：接受实例的元数据请求，只在multi-host模式的nova-network安装时使用。
* nova-compute：通过hypervisor API创建和终结虚拟机实例，例如XenAPI（XenServer/XCP），libvirt（KVM或QEMU），VMWareAPI（VMWare）。从队列中获取请求，执行一系列系统命令，例如启动KVM实例，更新数据库状态等。
* nova-scheduler：从队列中获取虚拟机实例请求，决定哪个计算节点运行该实例。
* nova-conductor：
* nova-network：
* nova-dhcpbridge：
* nova-consoleauth：
* nova-novncproxy：通过VNC进入运行中的虚拟机接口，支持浏览器客户端。
* nova-xvpnvncproxy：通过VNC进入运行中的虚拟机接口，支持Java客户端。
* nova-cert：管理x509认证。
* nova-objectstore：提供S3接口注册镜像，在支持euca2ools安装中需要，euca2ools采用S3通信，nova-objectstore将S3转换成Image Service的请求。
* euca2ools：管理云资源的命令行解析器。
* nova client：使用户作为tenant或end user提交命令。
* nova-manage client：让管理员提交命令。

##### 安装

**以下操作在controller中执行**

1. **安装Compute Service**

# yum install openstack-nova-api openstack-nova-cert openstack-nova-conductor openstack-nova-console openstack-nova-novncproxy openstack-nova-scheduler python-novaclient

1. **配置MySQL**

# openstack-config --set /etc/nova/nova.conf database connection mysql://nova:123456@controller/nova

1. **配置Message broker**

# openstack-config --set /etc/nova/nova.conf DEFAULT rpc\_backend qpid

# openstack-config --set /etc/nova/nova.conf DEFAULT qpid\_hostname controller

1. **配置VNC**

# openstack-config --set /etc/nova/nova.conf DEFAULT my\_ip 192.168.214.146

# openstack-config --set /etc/nova/nova.conf DEFAULT vncserver\_listen 192.168.214.146

# openstack-config --set /etc/nova/nova.conf DEFAULT vncserver\_proxyclient\_address 192.168.214.146

1. **创建数据库**

# mysql -u root -p

mysql> CREATE DATABASE nova;

mysql> GRANT ALL PRIVILEGES ON nova.\* TO 'nova'@'localhost' IDENTIFIED BY '123456';

mysql> GRANT ALL PRIVILEGES ON nova.\* TO 'nova'@'%' IDENTIFIED BY '123456';

mysql> exit

1. **创建数据库表**

# su -s /bin/sh -c "nova-manage db sync" nova

1. **在Identity Service中注册用户**

# keystone user-create --name=nova --pass=123456 --email=331274799@qq.com

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

1. **在Compute Service中注册Identity Service**

# openstack-config --set /etc/nova/nova.conf DEFAULT auth\_strategy keystone

# openstack-config --set /etc/nova/nova.conf keystone\_authtoken auth\_uri http://controller:5000

# 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 123456

1. **在Identity Service中注册Compute Service**

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

# keystone endpoint-create --service-id=$(keystone service-list | awk '/ compute / {print $2}') --publicurl=http://controller:8774/v2/%\(tenant\_id\)s --internalurl=http://controller:8774/v2/%\(tenant\_id\)s --adminurl=http://controller:8774/v2/%\(tenant\_id\)s

1. **启动Compute Service**

# 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

# nova image-list

**以下操作在compute中执行**

1. **安装Compute Service**

# yum install openstack-nova-compute

1. **配置MySQL**

# openstack-config --set /etc/nova/nova.conf database connection mysql://nova:123456@controller/nova

1. **在Compute Service中配置Identity Service**

# openstack-config --set /etc/nova/nova.conf DEFAULT auth\_strategy keystone

# openstack-config --set /etc/nova/nova.conf keystone\_authtoken auth\_uri http://controller:5000

# 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 123456

1. **配置Message broker**

# openstack-config --set /etc/nova/nova.conf DEFAULT rpc\_backend qpid

# openstack-config --set /etc/nova/nova.conf DEFAULT qpid\_hostname controller

1. **配置远程访问实例**

# openstack-config --set /etc/nova/nova.conf DEFAULT my\_ip 192.168.214.145

# 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 192.168.214.145

# openstack-config --set /etc/nova/nova.conf DEFAULT novncproxy\_base\_url http://controller:6080/vnc\_auto.html

# openstack-config --set /etc/nova/nova.conf DEFAULT glance\_host controller

1. **配置虚拟化技术**

检测硬件或hypervisor支持虚拟机硬件加速

# egrep -c '(vmx|svm)' /proc/cpuinfo

>0：支持，采用KVM

=0：不支持，需要配置libvirt，采用QEMU，执行

# openstack-config --set /etc/nova/nova.conf libvirt virt\_type qemu

1. **启动Compute Service**

# service libvirtd start

# service messagebus start

# chkconfig libvirtd on

# chkconfig messagebus on

# service openstack-nova-compute start

# chkconfig openstack-nova-compute on

#### Networking

两种搭建方式，一种是neutron，一种是nova-network

controller：

# openstack-config --set /etc/nova/nova.conf DEFAULT network\_api\_class nova.network.api.API

# openstack-config --set /etc/nova/nova.conf DEFAULT security\_group\_api nova

# service openstack-nova-api restart

# service openstack-nova-scheduler restart

# service openstack-nova-conductor restart

compute：

# yum install openstack-nova-network openstack-nova-api

# openstack-config --set /etc/nova/nova.conf DEFAULT network\_api\_class nova.network.api.API

# openstack-config --set /etc/nova/nova.conf DEFAULT security\_group\_api nova

# 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\_network\_bridge br100

# openstack-config --set /etc/nova/nova.conf DEFAULT flat\_interface eth0

# openstack-config --set /etc/nova/nova.conf DEFAULT public\_interface eth0

# service openstack-nova-network start

# service openstack-nova-metadata-api start

# chkconfig openstack-nova-network on

# chkconfig openstack-nova-metadata-api on

# source admin-openrc.sh

# nova network-create demo-net --bridge br100 --multi-host T --fixed-range-v4 192.168.214.144/28

# nova net-list

#### Dashboard

1. **安装Dashboard**

# yum install memcached python-memcached mod\_wsgi openstack-dashboard

1. **修改配置文件**

LOCATION和/etc/sysconfig/memcached保持一致

# vi /etc/openstack-dashboard/local\_settings

CACHES = {

'default': {

'BACKEND' : 'django.core.cache.backends.memcached.MemcachedCache',

'LOCATION' : '127.0.0.1:11211'

}

}

ALLOWED\_HOSTS = ['\*', 'localhost']

TIME\_ZONE=’’UTC

OPENSTACK\_HOST = "controller"

# setsebool -P httpd\_can\_network\_connect on

1. **启动服务**

# service httpd start

# service memcached start

# chkconfig httpd on

# chkconfig memcached on

<http://controller/dashboard>登录

#### Launch Instance

# source demo-openrc.sh

# ssh-keygen

# nova keypair-add --pub-key ~/.ssh/id\_rsa.pub demo-key

# nova keypair-list

创建一个虚拟机需要flavor，image名称，网络，security group，key，instance name。

flavor是指虚拟资源描述，包括处理器、内存、硬盘。

# nova flavor-list //查看可用flavor

# nova image-list //查看可用image

# neutron net-list //查看可用网络，这步之前需要# source admin-openrc.sh，其他都是source demo-openrc.sh

# nova secgroup-list //查看可用安全组，默认安全组是建立一个防火墙阻止外部进入实例

# nova boot --flavor m1.tiny --image cirros-0.3.2-x86\_64 --nic net-id=98e73831-daeb-4e31-af2c-52e65cecf017 --security-group default --key-name demo-key demo-instance1

#### Block Storage

##### 概述

Block Storage Service向compute为实例请求逻辑卷

cinder-api：接收API请求，将其导向cinder-volume。

cinder-volume：根据请求，读写块存储数据库，通过消息队列与其他进程交互（如cinder-scheduler），可通过驱动与多种存储设备交互。

cinder-scheduler：与nova-scheduler类似，获取可用的块存储节点，生成逻辑卷。

Messaging queue：进程交互路由。

##### 安装

1. **安装Block Storage**

# yum install openstack-cinder

1. **配置MySQL**

# openstack-config --set /etc/cinder/cinder.conf database connection mysql://cinder:123456@controller/cinder

1. **创建数据库**

# mysql -u root -p

mysql> GRANT ALL PRIVILEGES ON cinder.\* TO 'cinder'@'localhost' IDENTIFIED BY '123456';

mysql> GRANT ALL PRIVILEGES ON cinder.\* TO 'cinder'@'%' IDENTIFIED BY '123456';

1. **创建数据库表**

# su -s /bin/sh -c "cinder-manage db sync" cinder

1. **在Identity Service中注册用户**

# keystone user-create --name=cinder --pass=123456 --email=331274799@qq.com

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

1. **在Block Storage中注册Identity Service**

# openstack-config --set /etc/cinder/cinder.conf DEFAULT auth\_strategy keystone

# openstack-config --set /etc/cinder/cinder.conf keystone\_authtoken auth\_uri http://controller:5000

# openstack-config --set /etc/cinder/cinder.conf keystone\_authtoken auth\_host controller

# openstack-config --set /etc/cinder/cinder.conf keystone\_authtoken auth\_protocol http

# openstack-config --set /etc/cinder/cinder.conf keystone\_authtoken auth\_port 35357

# openstack-config --set /etc/cinder/cinder.conf keystone\_authtoken admin\_user cinder

# openstack-config --set /etc/cinder/cinder.conf keystone\_authtoken admin\_tenant\_name service

# openstack-config --set /etc/cinder/cinder.conf keystone\_authtoken admin\_password 123456

1. **配置Message blocker**

# 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

1. **在Identity Service中注册Block Storage**

# keystone service-create --name=cinder --type=volume --description="OpenStack Block Storage"

# keystone endpoint-create --service-id=$(keystone service-list | awk '/ volume / {print $2}') --publicurl=http://controller:8776/v1/%\(tenant\_id\)s --internalurl=http://controller:8776/v1/%\(tenant\_id\)s --adminurl=http://controller:8776/v1/%\(tenant\_id\)s

# keystone service-create --name=cinderv2 --type=volumev2 --description="OpenStack Block Storage v2"

# keystone endpoint-create --service-id=$(keystone service-list | awk '/ volumev2 / {print $2}') --publicurl=http://controller:8776/v2/%\(tenant\_id\)s --internalurl=http://controller:8776/v2/%\(tenant\_id\)s --adminurl=http://controller:8776/v2/%\(tenant\_id\)s

1. **启动Block Storage**

# service openstack-cinder-api start

# service openstack-cinder-scheduler start

# chkconfig openstack-cinder-api on

# chkconfig openstack-cinder-scheduler on

为虚拟机新增一块硬盘

虚拟机->设置->添加->硬盘…..

#fdisk –l 看到新增硬盘为/dev/sdb

# pvcreate /dev/sdb

# vgcreate cinder-volumes /dev/sdb

# vi /etc/lvm/lvm.conf //不需要修改该文件

devices {

filter = ["a/.\*/" ]

}

# pvdisplay //可看到/dev/sdb

1. **安装Block Storage**

# yum install openstack-cinder scsi-target-utils

1. **在Block Storage中配置Identity Service**

# openstack-config --set /etc/cinder/cinder.conf DEFAULT auth\_strategy keystone

# openstack-config --set /etc/cinder/cinder.conf keystone\_authtoken auth\_uri <http://controller:5000>

# openstack-config --set /etc/cinder/cinder.conf keystone\_authtoken auth\_host controller

# openstack-config --set /etc/cinder/cinder.conf keystone\_authtoken auth\_protocol http

# openstack-config --set /etc/cinder/cinder.conf keystone\_authtoken auth\_port 35357

# openstack-config --set /etc/cinder/cinder.conf keystone\_authtoken admin\_user cinder

# openstack-config --set /etc/cinder/cinder.conf keystone\_authtoken admin\_tenant\_name service

# openstack-config --set /etc/cinder/cinder.conf keystone\_authtoken admin\_password 123456

1. **配置Message broker**

# 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

1. **配置MySQL**

# openstack-config --set /etc/cinder/cinder.conf database connection mysql://cinder:123456@controller/cinder

# openstack-config --set /etc/cinder/cinder.conf DEFAULT glance\_host controller

# vi /etc/tgt/targets.conf

include /etc/cinder/volumes/\*

1. **启动Block Storage**

# service openstack-cinder-volume start

# service tgtd start

# chkconfig openstack-cinder-volume on

# chkconfig tgtd on

##### 验证

# source demo-openrc.sh

# cinder create --display-name myVolume 1

# cinder list

#### Object Storage

##### 概述

* swift-proxy-server：接收Object Storage API和HTTP请求寻找相应的account，container，object的位置，正确引导请求。Proxy server可使用cache如memcache以提高效率。
* swift-account-server：管理Object Storage Service定义的account。
* swift-container-server：管理Object Storage Service中containers，folders的映射。
* swift-object-server：管理存储节点上的objects，比如文件。
* 周期进程：执行大型数据存储操作。复制服务保证集群一致性。其他进程包括更新

##### 安装

# keystone user-create --name=swift --pass=123456 [--email=331274799@qq.com](mailto:--email=331274799@qq.com)

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

# keystone service-create --name=swift --type=object-store --description="OpenStack Object Storage"

# keystone endpoint-create --service-id=$(keystone service-list | awk '/ object-store / {print $2}') --publicurl='http://controller:8080/v1/AUTH\_%(tenant\_id)s' --internalurl='http://controller:8080/v1/AUTH\_%(tenant\_id)s' --adminurl=http://controller:8080

# mkdir -p /etc/swift

# vi /etc/swift/swift.conf

[swift-hash]

# random unique string that can never change (DO NOT LOSE)

swift\_hash\_path\_suffix = fLIbertYgibbitZ

# yum install openstack-swift-account openstack-swift-container openstack-swift-object xfsprogs xinetd

在加一块硬盘，需重启后，命名为/dev/sdc

# fdisk /dev/sdc

# mkfs.xfs /dev/sdc1

# echo "/dev/sdc1 /srv/node/sdc1 xfs noatime,nodiratime,nobarrier,logbufs=8 0 0" >> /etc/fstab

# mkdir -p /srv/node/sdc1

# mount /srv/node/sdc1

# chown -R swift:swift /srv/node

# vi /etc/rsync.conf

uid = swift

gid = swift

log file = /var/log/rsyncd.log

pid file = /var/run/rsyncd.pid

address = 192.168.214.0

[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

# vi /etc/xinetd.d/rsync

disable = false

# service xinetd start

# mkdir -p /var/swift/recon

# chown -R swift:swift /var/swift/recon

# yum install openstack-swift-proxy memcached python-swiftclient python-keystone-auth-token

# vi /etc/sysconfig/memcached

OPTIONS="-l 192.168.214.0"

# service memcached start

# chkconfig memcached

# 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

#### Orchestration Service

##### 概述

提供模板以描述云应用，通过OpenStack API请求运行云应用。

* heat：命令行客户端，与heat-api通信以运行AWS CloudFormation API。终端开发者可直接使用REST API。
* heat-api：提供OpenStack原生REST API，处理API请求，发送到heat-engine。
* heat-api-cfn：提供AWS查询API，处理API请求，发送到heat-engine。
* heat-engine：启动模版，返回事件给API请求者。

##### 安装

1. **安装Orchestration Service**

# yum install openstack-heat-api openstack-heat-engine openstack-heat-api-cfn

1. **配置MySQL**

# openstack-config --set /etc/heat/heat.conf database connection mysql://heat:123456@controller/heat

1. **配置数据库**

$ mysql -u root -p

mysql> CREATE DATABASE heat;

mysql> GRANT ALL PRIVILEGES ON heat.\* TO 'heat'@'localhost' IDENTIFIED BY '123456';

mysql> GRANT ALL PRIVILEGES ON heat.\* TO 'heat'@'%' IDENTIFIED BY '123456';

1. **配置数据库表**

# su -s /bin/sh -c "heat-manage db\_sync" heat

1. **在Identity Service中注册用户**

# keystone user-create --name=heat --pass=123456 --email=331274799@qq.com

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

1. **修改配置文件**

# vi /etc/heat/heat.conf

[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 = 123456

[ec2authtoken]

auth\_uri = http://controller:5000/v2.0

1. **在Identity Service中注册Orchestration Service**

# keystone service-create --name=heat --type=orchestration --description="Orchestration"

# keystone endpoint-create --service-id=$(keystone service-list | awk '/ orchestration / {print $2}') --publicurl=http://controller:8004/v1/%\(tenant\_id\)s --internalurl=http://controller:8004/v1/%\(tenant\_id\)s --adminurl=http://controller:8004/v1/%\(tenant\_id\)s

# keystone service-create --name=heat-cfn --type=cloudformation --description="Orchestration CloudFormation"

# keystone endpoint-create --service-id=$(keystone service-list | awk '/ cloudformation / {print $2}') --publicurl=http://controller:8000/v1 --internalurl=http://controller:8000/v1 --adminurl=http://controller:8000/v1

1. **创建角色**

# keystone role-create --name heat\_stack\_user

1. **配置waitcondition server**

# openstack-config --set /etc/heat/heat.conf DEFAULT heat\_metadata\_server\_url <http://controller:8000>

# openstack-config --set /etc/heat/heat.conf DEFAULT heat\_waitcondition\_server\_url <http://controller:8000/v1/waitcondition>

1. **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

# vi test-stack.yml

heat\_template\_version: 2013-05-23

description: Test Template

parameters:

ImageID:

type: string

description: Image use to boot a server

NetID:

type: string

description: Network ID for the server

resources:

server1:

type: OS::Nova::Server

properties:

name: "Test server"

image: { get\_param: ImageID }

flavor: "m1.tiny"

networks:

- network: { get\_param: NetID }

outputs:

server1\_private\_ip:

description: IP address of the server in the private network

value: { get\_attr: [ server1, first\_address ] }

# NET\_ID=$(nova net-list | awk '/ demo-net / { print $2 }')

# echo $NET\_ID

# heat stack-create -f test-stack.yml -P "ImageID=cirros-0.3.2-x86\_64;NetID=$NET\_ID" testStack

# heat stack-list

#### Telemetry Module

##### 概述

* ceilometer-agent-compute：在各计算节点上收集资源使用数据，
* ceilometer-agent-central：在中心管理服务器上收集资源使用数据。
* ceilometer-collector：在一个或多个中心管理服务器上监控消息队列。
* ceilometer-alarm-notifier：在一个或多个中心管理服务器上允许设置警告，如果超过阈值。
* data store：支持同步写和读。
* ceilometer-api：在一个或多个中心管理服务器上提供访问数据。

##### 安装

以下操作在controller执行

1. **安装Telemetry Service**

# yum install openstack-ceilometer-api openstack-ceilometer-collector openstack-ceilometer-notification openstack-ceilometer-central openstack-ceilometer-alarm python-ceilometerclient

1. **安装MongoDB**

# yum install mongodb-server mongodb

# service mongod start

# chkconfig mongod on

# mongo

> db=db.getSiblingDB('ceilometer');

> db.addUser({user:"ceilometer", pwd:"123456",roles:["readWrite","dbAdmin"]});

# openstack-config --set /etc/ceilometer/ceilometer.conf database connection mongodb://ceilometer:123456@controller:27017/ceilometer

# CEILOMETER\_TOKEN=$(openssl rand -hex 10)

# echo $CEILOMETER\_TOKEN

# openstack-config --set /etc/ceilometer/ceilometer.conf publisher metering\_secret $CEILOMETER\_TOKEN

# openstack-config --set /etc/ceilometer/ceilometer.conf DEFAULT rpc\_backend ceilometer.openstack.common.rpc.impl\_qpid

# keystone user-create --name=ceilometer --pass=123456 --email=331274799@qq.com

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

# openstack-config --set /etc/ceilometer/ceilometer.conf DEFAULT auth\_strategy keystone

# 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 auth\_uri http://controller:5000

# openstack-config --set /etc/ceilometer/ceilometer.conf keystone\_authtoken admin\_password 123456

# openstack-config --set /etc/ceilometer/ceilometer.conf service\_credentials os\_auth\_url http://controller:5000/v2.0

# openstack-config --set /etc/ceilometer/ceilometer.conf service\_credentials os\_username ceilometer

# openstack-config --set /etc/ceilometer/ceilometer.conf service\_credentials os\_tenant\_name service

# openstack-config --set /etc/ceilometer/ceilometer.conf service\_credentials os\_password 123456

# keystone service-create --name=ceilometer --type=metering --description="Telemetry"

# keystone endpoint-create --service-id=$(keystone service-list | awk '/ metering / {print $2}') --publicurl=http://controller:8777 --internalurl=http://controller:8777 --adminurl=http://controller:8777

# service openstack-ceilometer-api start

# service openstack-ceilometer-notification start

# service openstack-ceilometer-central start

# service openstack-ceilometer-collector start

# service openstack-ceilometer-alarm-evaluator start

# service openstack-ceilometer-alarm-notifier start

# chkconfig openstack-ceilometer-api on

# chkconfig openstack-ceilometer-notification on

# chkconfig openstack-ceilometer-central on

# chkconfig openstack-ceilometer-collector on

# chkconfig openstack-ceilometer-alarm-evaluator on

# chkconfig openstack-ceilometer-alarm-notifier on

以下操作在compute执行

# yum install openstack-ceilometer-compute python-ceilometerclient python-pecan

# 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

# vi /etc/nova/nova.conf

notification\_driver = nova.openstack.common.notifier.rpc\_notifier

notification\_driver = ceilometer.compute.nova\_notifier

# service openstack-nova-compute restart

# openstack-config --set /etc/ceilometer/ceilometer.conf publisher metering\_secret CEILOMETER\_TOKEN

# openstack-config --set /etc/ceilometer/ceilometer.conf DEFAULT qpid\_hostname controller

# 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 123456

# openstack-config --set /etc/ceilometer/ceilometer.conf service\_credentials os\_username ceilometer

# openstack-config --set /etc/ceilometer/ceilometer.conf service\_credentials os\_tenant\_name service

# openstack-config --set /etc/ceilometer/ceilometer.conf service\_credentials os\_password 123456

# openstack-config --set /etc/ceilometer/ceilometer.conf service\_credentials os\_auth\_url http://controller:5000/v2.0

# service openstack-ceilometer-compute start

# chkconfig openstack-ceilometer-compute on

以下在controller中执行

# openstack-config --set /etc/glance/glance-api.conf DEFAULT notification\_driver messaging

# openstack-config --set /etc/glance/glance-api.conf DEFAULT rpc\_backend qpid

# service openstack-glance-api restart

# service openstack-glance-registry restart

# 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

# service openstack-cinder-api restart

# service openstack-cinder-scheduler restart

# service openstack-cinder-volume restart

# keystone role-create --name=ResellerAdmin

# keystone user-role-add --tenant service --user ceilometer --role ff9dfbdc09a4457cbec53ab4e204298a

## 问题

**问题：Bypassing authentication using a token & endpoint**

.bash\_profile中设置OS\_SERVICE\_TOKEN和OS\_SERVICE\_ENDPOINT

unset OS\_SERVICE\_TOKEN OS\_SERVICE\_ENDPOINT

admin\_openrc.sh中也设置了用户名密码等信息，导致冲突

**问题：ERROR: Malformed request URL: URL's project\_id '199774f96ea343fdb50d0c024f0e548a' doesn't match Context's project\_id**

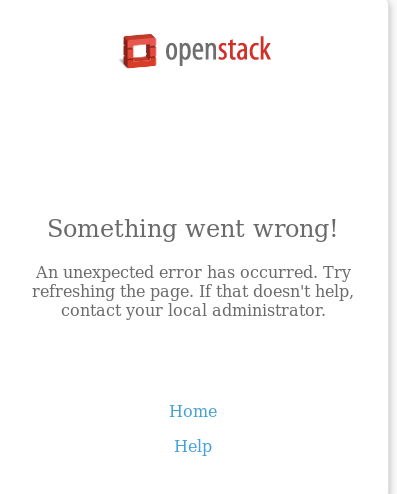
缺少

# openstack-config --set /etc/nova/nova.conf DEFAULT auth\_strategy keystone

# openstack-config --set /etc/nova/nova.conf keystone\_authtoken auth\_uri http://controller:5000

记得重启让配置生效

**Dashboard无法启动**



Something went wrong!

An unexpected error has occurred. Try refreshing the page. If that doesn't help, contact your local administrator.

查看/var/log/httpd/error\_log

**SuspiciousOperation: Invalid HTTP\_HOST header (you may need to set ALLOWED\_HOSTS): controller**

vi /etc/openstack-dashboard/local\_settings

ALLOWED\_HOSTS = ['\*', 'localhost']

service httpd restart

**Authentication backends without a `supports\_inactive\_user` attribute are deprecated. Please define it in <class 'openstack\_auth.backend.KeystoneBackend'>.Login successful for user "admin".**

# vi /etc/selinux/config

#SELINUX=enforcing

SELINUX=disabled

**No handlers could be found for logger "heat.common.config"**

忽略该问题

# Orchestration

定义人可读、机器可执行的服务以管理OpenStack基础设施和应用的整个生命周期。

## heat 介绍

Heat是OpenStack Orchestration的主要组成部分，通过模板实现资源的自动化部署。

Heat提供规定格式的任务模板，允许开发者定义部署系统时的一连串任务，然后由heat调用heat-engine按顺序执行模版中定义的任务，以生成最终的应用系统。

Heat提供两种模板：符合AWS CloudFormation格式要求的CFN模板（面向CloudFormation-compatible Query API）和HOT模板（Heat Orchestration Template，面向OpenStack-native Rest API）

## heat 功能

1. Heat template以文本形式描述云应用的基础设施，人可读且可通过版本控制；
2. Heat template支持的基础设施资源包括：servers，floating ips，volumes，security groups，users等；
3. Heat可与其他服务如Ceilometer集成，以使用高级功能，如instance autoscaling；
4. 模版指明资源间依赖关系，以使heat能够按顺序调用OpenStack API生成各组件；
5. Heat管理应用的整个生命周期，当需要改变配置时，只需修改模板并利用它更新stack，heat知道该如何正确地实现变动；
6. Heat template能够与其他配置管理软件，如Puppet，Chef集成。

## heat 组成

* heat：CLI，与heat-api通信以执行AWS CloudFormation APIs。
* heat-api：提供OpenStack原生 Rest API，通过RPC向heat-engine发送API请求。
* heat-api-cfn：提供AWS-style Query API
* heat-engine：完成编排的主要工作，返回结果给API调用者。

## heat 术语

* Stack，是CloudFormation中管理一组资源的基本单位。一个栈往往对应与一个应用程序。Stack就是Heat管理应用程序的逻辑单元，可认为是模板的实例。
* Resources，一个栈可以拥有很多资源，资源是底层服务的抽象。CPU，memory，disk，网络等都可以看作是资源。资源和资源之间会存在依赖关系。Heat在创建栈的时候会自动解析依赖关系，按顺序创建资源。从Havana开始Heat能并行的创建没有依赖关系的资源。

## heat 使用

采用devstack安装时，修改localrc文件

IMAGE\_URLS+=",[http://cloud.fedoraproject.org/fedora-20.x86\_64.qcow2](http://cloud.fedoraproject.org/fedora-20.x86_64.qcow2" \t "_blank)"

yum update

./stack.sh 部署openstack环境

source openrc 改为管理员权限

nova flavor-list 查看虚拟机类型

nova image-list 查看镜像

nova keypair-add heat\_key 生成用于访问虚拟机的key

创建堆栈，-f指明heat模板，-P用于设置模板定义参数

heat stack-create teststack -f ./WordPress\_Native.yaml -P "key\_name=heat\_key;instance\_type=m1.tiny;image\_name=cirros-0.3.0-i386-uec"

heat stack-list 查看生成的堆栈

heat event-list teststack 查看堆栈事件

heat stack-show teststack 查看堆栈详情

heat stack-delete teststack 删除堆栈，回收资源

## heat 模板

 HOT作为CFN（Heat CloudFormation-compatible format）的取代模板，是Heat未来支持的原生格式，目前正在开发中，未来会提供CFN模板的所有功能。

**HOT template 结构：**

heat-template-version: 2013-05-23，YAML文档

description：可选，模版描述

parameter-groups：可选，声明输入参数的组和顺序

parameters：可选，用于自定义模板的一组输入值

resources：必填，定义一组资源及其相互关系，且必须至少包含一个resource定义

outputs：可选，栈创建者可以看到的一组值

**heat template示例：**

https://github.com/openstack/heat-templates

分为与AWS CloudFormation兼容的cfn目录下的CFN模板和hot目录下的HOT模板。

1. **示例1**

heat\_template\_version: 2013-05-23 //必填，指明当前HOT的版本号

description: Simple template to deploy a single compute instance

//description可选，描述该文档的功能—部署单个计算实例，如果多行文本则采用以下格式

description: >

This is how you can provide a longer description

of your template that goes over several lines.

//resources必填，且至少包含一个resource定义，该例子中，计算节点包含key\_name,image,flavor的三个参数，其值必须已在openstack中定义

resources:

my\_instance:

type: OS::Nova::Server

properties:

key\_name: my\_key

image: F18-x86\_64-cfntools

flavor: m1.small

1. **示例2**

heat\_template\_version: 2013-05-23

description: Simple template to deploy a single compute instance

//为避免硬编码，parameters 用于预定义参数，允许用户在部署时定制模板，以下就是让用户在部署时自行填写key\_name,image\_id等参数

parameters:

key\_name:

type: string

label: Key Name

description: Name of key-pair to be used for compute instance

image\_id:

type: string

label: Image ID

description: Image to be used for compute instance

instance\_type:

type: string

label: Instance Type

description: Type of instance (flavor) to be used

default: m1.small //default如果部署时没有提供相应的参数，则采用该默认值

constraints: //constraints限制可选值

- allow\_values: [ m1.medium, m1.large, m1.xlarge ]

description: Value must be one of m1.medium, m1.large or m1.xlarge.

database\_password:

type: string

label: Database Password

description: Password to be used for database

hidden: true //hidden部署时隐藏用户的输入

constraints:

- length: { min: 6, max: 8 }

description: Password length must be between 6 and 8 characters.

- allowed\_pattern: "[a-zA-Z0-9]+"

description: Password must consist of characters and numbers only.

- allowed\_pattern: "[A-Z]+[a-zA-Z0-9]\*"

description: Password must start with an uppercase character.

resources:

my\_instance:

type: OS::Nova::Server //指明资源类型，如OS::Nova::Server

properties: //指明资源参数，可硬编码，也可通过get\_param方法引用前面预定义参数

key\_name: { get\_param: key\_name }

image: { get\_param: image\_id }

flavor: { get\_param: instance\_type }

//outputs输出结果，例子中将计算节点的IP输出

outputs:

instance\_ip:

description: The IP address of the deployed instance

value: { get\_attr: [my\_instance, first\_address] }

## heat 典型应用

部署一个WordPress应用，只包含单个服务器，上面运行WordPress和MySQL。

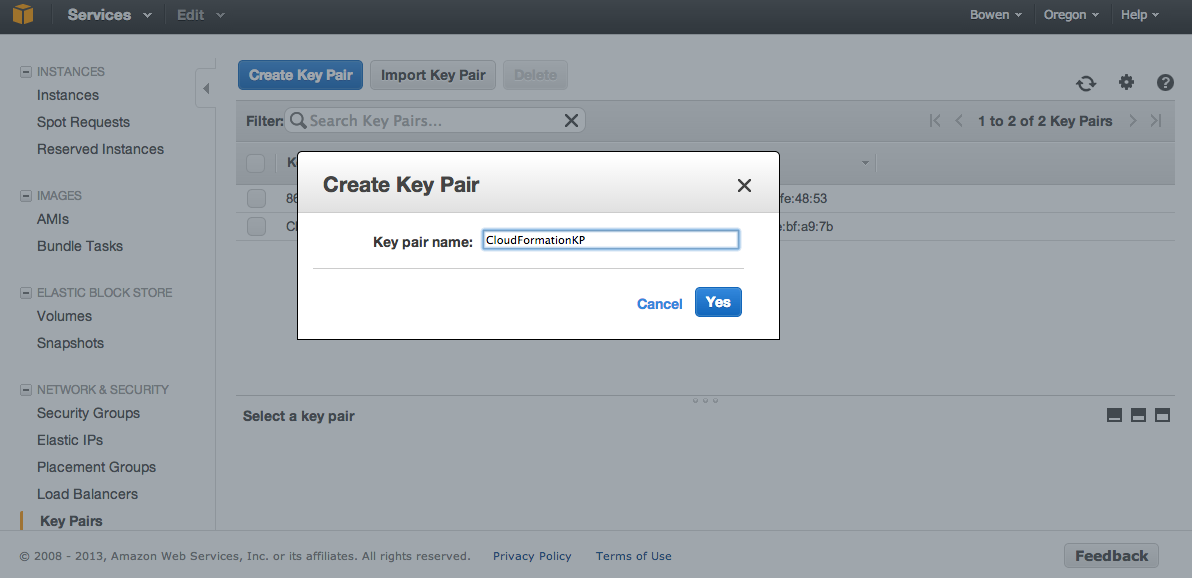
https://github.com/openstack/heat-templates/blob/master/hot/F20/WordPress\_Native.yaml

部署一个WordPress应用，该应用包含2台服务器，一台运行WordPress的Web服务器，一台保存数据的MySQL服务器。

<https://github.com/openstack/heat-templates/blob/master/hot/F20/WordPress_2_Instances.yaml>

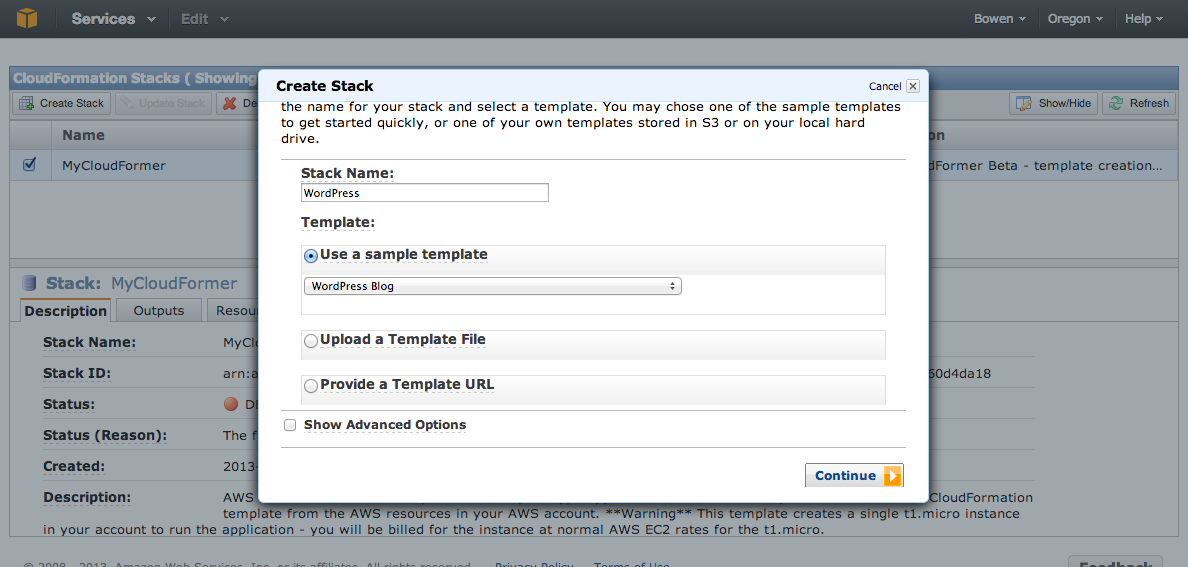
通过AWS Management Console，使用CloudFormation来创建一个部署在EC2上、使用RDS作为数据库的WordPress网站。

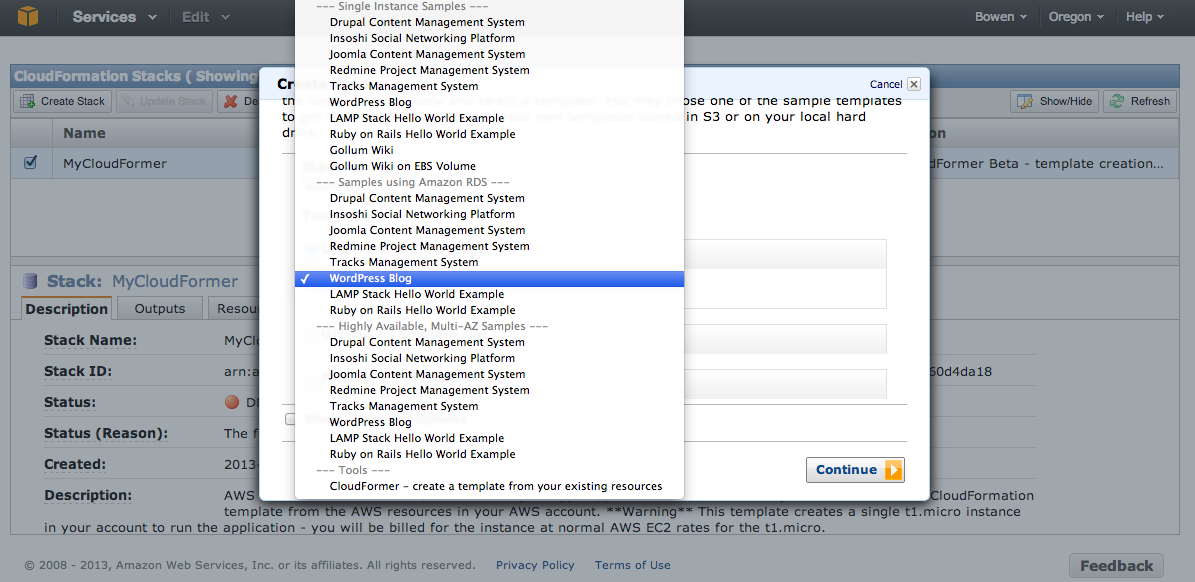
首先，需要登录到AWS Management Console，选择EC2服务，点击左侧菜单的Key Pair,创建一个Key Pair。这个Key Pair将在接下来被使用，主要使AWS能够ssh到创建的EC2机器上。当然你也可以使用已有的Key Pair。



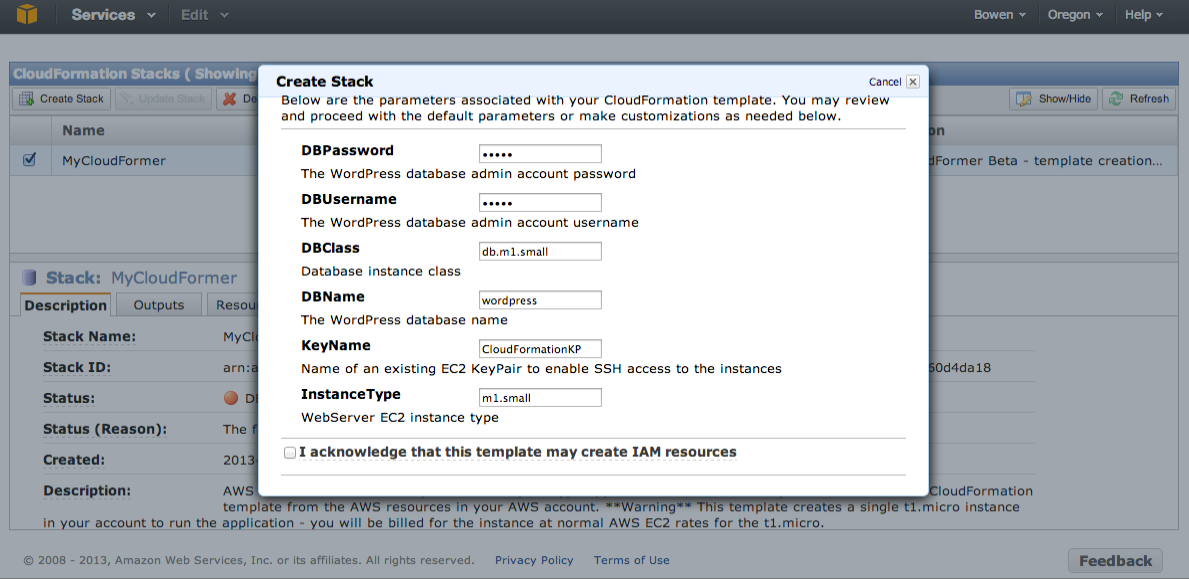
然后选择CloudFormation服务，点击Create Stack按钮。

然后输入Stack名称，并选择一个Template。由于我们要创建一个WordPress的站点，可以选择use a sample temple,并选择WordPress这个模板。

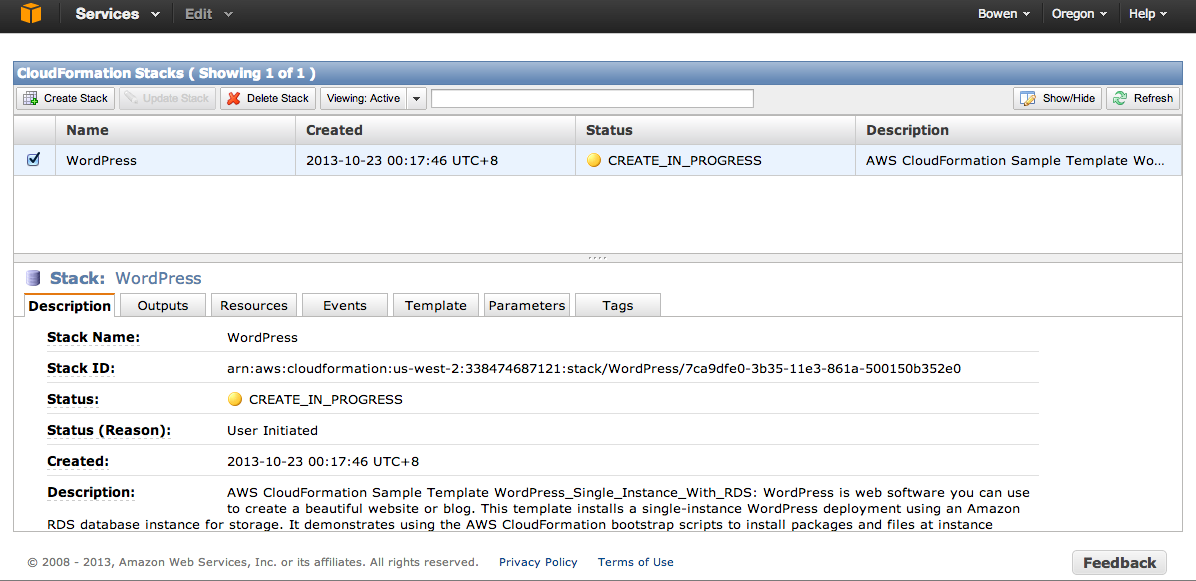




然后点击continue按钮，配置相关的参数。注意在KeyName一项中输入我们第一步创建的Key Pair。

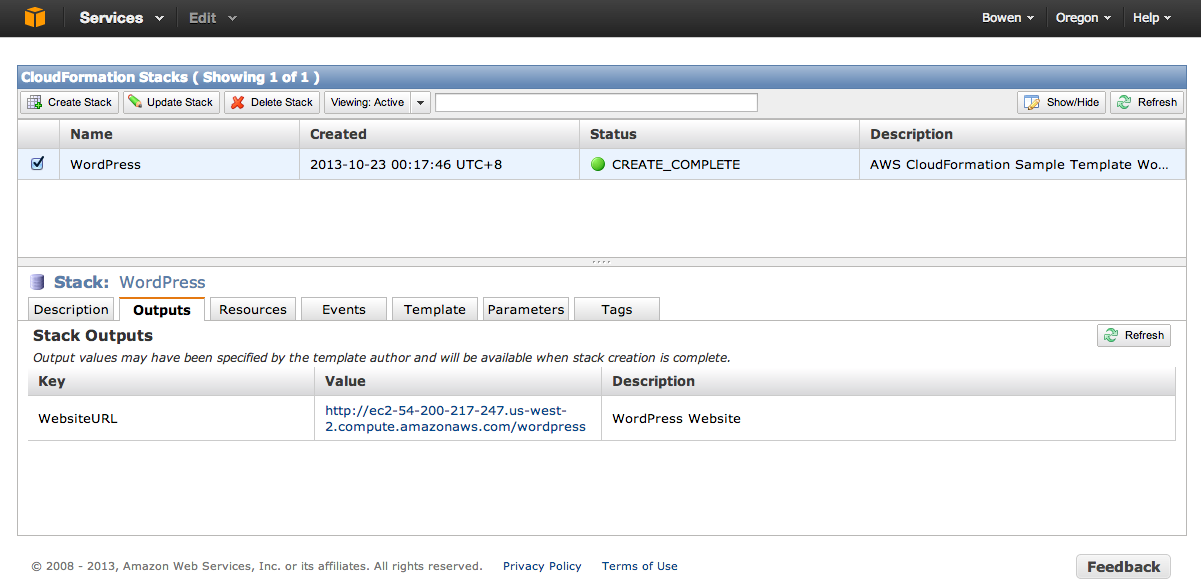


然后点击continue按钮，配置此Stack的标签，这个是可选项，可以跳过。再点一下continue，将会再次确认想要的创建的资源信息，继续后就可以看到资源正在创建了。



页面下半部分有很多标签，你可以随时查看该Stack的描述、资源、事件、参数、输出等各项信息。

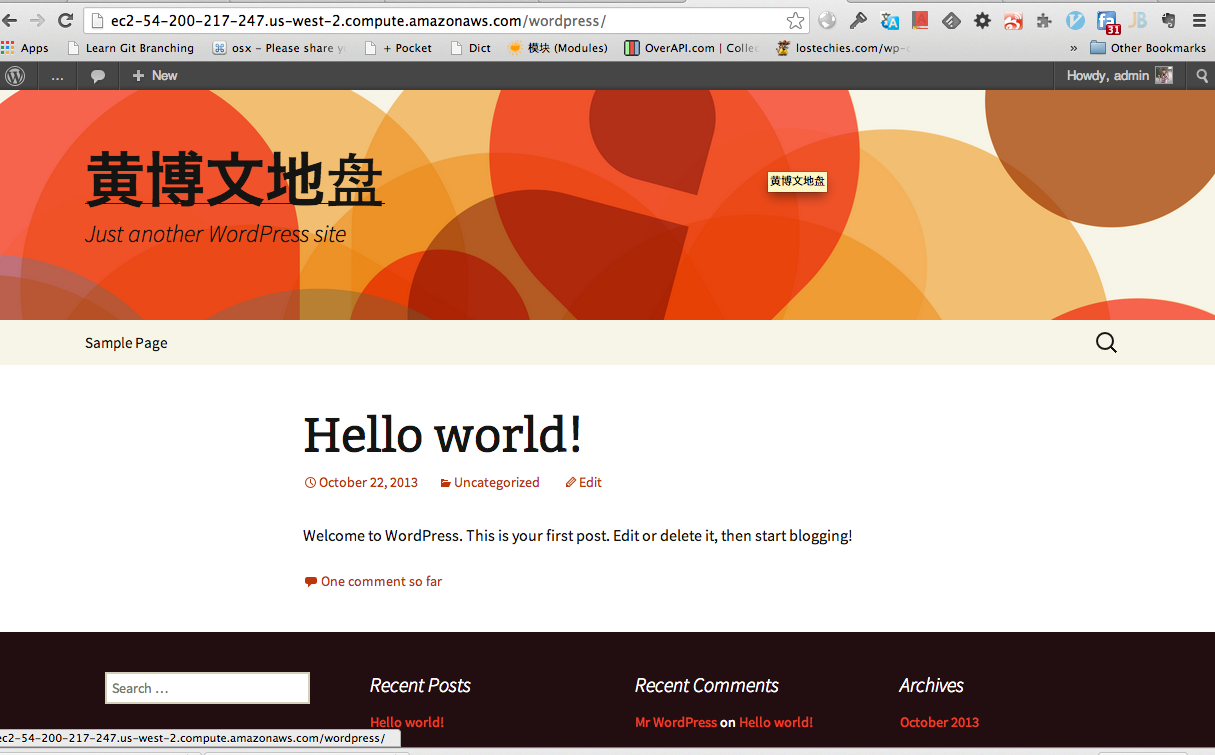
等待大约20分钟，Stack就会执行完毕，在Outputs标签中你会看到有一个url。



这个就是我们创建的WordPress的入口地址，访问该链接会进入WordPress的初始化设置页面。

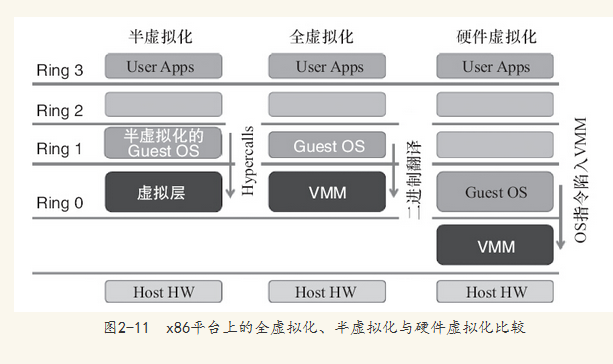


配置完以后，一个新的WordPress就诞生了，重新访问URL，你会看到你的博客首页。



# KVM

# XEN



全虚拟化：Guest OS的特权级别在Ring 1，对特权指令采用动态二进制翻译方式，无需对Guest OS进行修改，ESX。

半虚拟化：Guest OS的特权级别在Ring 1，所有特权指令都以Hypercall方式调用，需要对Guest OS进行修改，XEN。

硬件虚拟化：Guest OS的特权级别在Ring 0，增加根模式支持，特权指令自动陷入根模式VMM中。无需对Guest OS进行修改，VT-x和AMD-x，KVM。