

# #316 软件工程实验室 OpenStack-M 版本 部署手册

测试情况：

测试时间：始于 2017 年 05 月 20 日

作者：施凯，杨杰

时间：2017 年 05 月 20 日（第一版）

## 文档基本参数说明

配置名称:	OpenStack 的 M 版本部署手册
编写作者:	施凯, 杨杰
编写时间:	2017 年 05 月 20 日
配置说明:	完成在服务器端的集群配置安装
参考资料:	<a href="https://docs.openstack.org/mitaka/zh_CN/install-guide-ubuntu/">https://docs.openstack.org/mitaka/zh_CN/install-guide-ubuntu/</a>
命令速查:	<a href="https://docs.openstack.org/zh_CN/user-guide/cli-cheat-sheet.html">https://docs.openstack.org/zh_CN/user-guide/cli-cheat-sheet.html</a>
硬件环境:	Ubuntu 14.04 操作系统
所需软件:	Putty, OpenStack 的 M 版本在线安装源等
验证操作者:	施凯, 杨杰, 杨星光和史海洋

### 文档修订历史记录

日期	说明	版本号	修订者
2017 年 05 月 20 日	文档建立	V1.0.0	施凯，杨杰

# 1 硬件说明

◆ 请参考 L 版本文档说明

# 2 服务器 RAID 配置说明

◆ 请参考 L 版本文档说明

# 3 软件说明

◆ 请参考 L 版本文档说明

# 4 系统架构

◆ 请参考 L 版本文档说明

# 5 网络地址分配

◆ 请参考 L 版本文档说明

# 6 安装服务器 LVM 划分说明

◆ 请参考 L 版本文档说明

# 7 物理网络拓补图分析

◆ 请参考 L 版本文档说明

**特别注意：集群到底启用哪种网络结构，请具体查看附件相关文档。这里所指的网路包括：Linuxbridge+FLAT, Linuxbridge+VLAN,**

Linuxbridge+VXLAN, OpenSwith+VLAN, 或 OpenSwith+VXLAN 等, 我们已经做特殊的资料收集整理工作。

## 8 云平台 VLAN 的增强配置

◆ 请参考 L 版本文档说明

## 9 安装步骤

### 9.1 控制节点

控制节点是整个集群的控制中心，集群的所有命令转发，指令协调，外部用户访问，认证中心，集群所安装的各个服务，数据库等都需要在控制节点上指明，并且安装相应的组件。

◆ 其中核心包括：

- 消息队列组件：RabbitMQ
- 集群认证中心组件：KeyStone
- 外部访问：ApacheHTTP 服务器，项目、用户、角色、权限设定等
- 每个服务组件的实体创建，3 个不同作用域的 API 端点（endPoint）
- 构建 OpenStack 客户端脚本操作
- 所有服务组件的数据库，都部署在控制节点（统一管理）
- 核心部署-镜像存储 Glance 组件（管理部分）
- 核心部署-块存储 Cinder 组件（管理部分）
- 核心部署-计算 Nova 组件（管理部分）
- 核心部署-网络 Neturon 组件（管理部分）
- 控制节点部署 Dashborad 用于网页可视化管理

◆ 主要安装有：

KeyStone, nova-api, nova-cert, nova-consoleauth, nova-novncproxy,  
nova-scheduler, horizon  
Networking, NTP, MySql, RabbitMQ 等

◆ 物理主机部分：

- 配置管理网络-网卡信息-特别重要（走管理网络，主机名解析）
- 配置外部网络-网卡信息-特别重要，承担远程用户的集群连接和管理。

◆ 补充说明：

## 9.1.1 各主机基本网络配置

### ◆ 网络配置 (细节请查看网络地址分配)

配置 **/etc/network/interfaces**

IP 地址采用静态配置格式如下, 启用了多少个网卡, 就配置几个, 格式如下:

```
auto lo
```

```
iface lo inet loopback
```

```
auto eth1
```

```
iface eth1 inet static
```

```
address      IP 地址
```

```
netmask      掩码
```

```
gateway      网关地址
```

```
network      网络号 //非必要
```

```
broadcast    广播地址 //非必要
```

```
.....
```

主网络配网关, 其他不要配, 否则网络不能连接

### ◆ 修改主机名字 hostname

配置 **/etc/hostname**

```
hostname 主机名
```

//指明某台服务器的主机名字

### ◆ 修改域名与主机对应关系

配置 **/etc/hosts**

```
IP 地址 主机名
```

//将集群里面的所有机器 (一行一个主机), 按照这样的对应关系配置好, 非常重要需要主机名解析, 相当于 ping 主机名也是可以通的!!! (走管理网络-所以填写都是各自物理服务器管理网络网卡接口的 IP 地址)

### ◆ 设置 DNS

配置 **/etc/resolv.conf**

```
nameserver IP 地址
```

//后面的 IP 地址是域名服务器 (是正常的外网 DNS 地址, 比如学校是 202.120.111.3)

### ◆ 防止重启机器后 DNS 重置

配置 **/etc/resolvconf/resolv.conf.d/tail**

```
nameserver IP 地址
```

//后面的 IP 地址是域名服务器-同上

### ◆ 重启网络

**/etc/init.d/networking restart**

**检验标准：**

- 1、网络各个网卡已经工作，并且在 `ifconfig` 中看到配置已经启用。
- 2、网络能访问外网，例如可以 `ping` 通百度。
- 3、能进行域名解析，即 `ping` 集群中各个主机名也是可以通的（走管理网络）。
- 4、若配置都是正确的，网卡没有启动或者配置没有生效，请 `reboot` 服务器。
- 5、特别注意：网卡的名字，有些服务器命名为 `ethx`，有些服务器是 `emx`，其中 `x` 用阿拉伯数字表示 `0-n`。

## 9.1.2 在控制节点基本安装及配置消息 RabbitMQ

- ◆ NTP 同步时钟服务配置，集群时间全部同步控制节点服务器，目的是为了集群能够保持时钟同步。

- 安装软件

```
apt-get install chrony
```

- 注释或者删除所有 Server 开头的配置项(vi /etc/chrony/chrony.conf)，并且添加**控制节点服务器管理网络网卡**的 IP 地址，具体如上详细网络结构图  
`server 控制节点管理网络的网卡 IP 地址 iburst`

- 重启 NTP 服务

```
service chrony restart
```

- 验证 NTP 服务安装是否成功

```
chronyc sources
```

//显示如下表示安装成功！

```
lab316@controller:~$ chronyc sources
210 Number of sources = 1
MS Name/IP address         Stratum Poll Reach LastRx Last sample
=====
^? controller              0      6      0   10y   +0ns[  +0ns] +/-    0
ns
```

- ◆ OpenStack 核心库安装，包括其中的部分组件等。

```
apt-get install software-properties-common
```

```
add-apt-repository cloud-archive:mitaka
```

- ◆ 系统升级并重启服务器

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

//默认防火墙已经关闭，可以 `ufw disable` 但是不要重启电脑，接下来操作一气呵成！

- ◆ 安装 OpenStack 客户端

```
apt-get install python-openstackclient
```

大多数 OpenStack 服务使用 SQL 数据库来存储信息。典型地，数据库统一运行在控制节点上。其他节点（计算节点，存储节点，镜像节点，KeyStone 等认证服务）都通过远程访问形式进行数据库操作。目的是数据库安全，统一管理，不至于由于某个节点的故障导致集群失败。从而只需要单一的保证控制节点安全、稳定即可。

- ◆ 安装 MySQL 数据库并配置

```
apt-get install mariadb-server python-pymysql
```

**注意：为数据库用户 root 设置适当的密码。此密码用来每个节点服务器与数据库远程通讯读写作用（非常重要！不可忘记：默认我设为 123456）。**

- ◆ **创建**并编辑 `/etc/mysql/conf.d/openstack.cnf`，然后完成如下动作：  
在 `[mysqld]` 部分，设置 `bind-address` 值为**控制节点的管理网络 IP 地址**，使得其它节点可以通过**管理网络访问数据库(特别注意是：管理网络)**。并在 `[mysqld]` 部分，设置如下键值来启用一起有用的选项和 UTF-8 字符集：

`[mysqld]`

`bind-address = 10.0.0.11`(根据实际情况替换)

//此地址**控制节点管理网络 IP 地址**，其他节点访问此地址，即可找到数据库，具体如上详细网络结构图及地址分配。

`default-storage-engine = innodb`

`innodb_file_per_table`

`collation-server = utf8_general_ci`

`character-set-server = utf8`

`max_connections=4096`（防止后续服务器核数过多导致的数据库访问问题）

- ◆ 重启数据库服务：  
`service mysql restart`
- ◆ 执行 `mysql_secure_installation` 脚本来对数据库进行安全加固，根据提示选择 Y/N(重要)。

Telemetry 服务使用 NoSQL 数据库来存储信息（由于首次，我们没有安装流量计费服务，所以暂时这个数据库不安装，2.0 版本的时候引入）。

OpenStack 使用 **Message Queue** 协调操作和各服务的状态信息。消息队列服务一般运行在控制节点上。OpenStack 支持好几种消息队列服务包括 **RabbitMQ**，**Qpid** 和 **ZeroMQ**。不过，大多数发行版本的 OpenStack 包支持特定的消息队列服务。本指南安装 **RabbitMQ** 消息队列服务。

消息队列的原理与作用，入门可以参考 5 分钟玩转这本书，附录说明。

**此处我仅简述 RabbitMQ 消息队列服务在 Nova 中的一个命令中的流转及响应操作：**

- ◆ 安装 RabbitMQ 消息队列服务  
`apt-get install rabbitmq-server`

- ◆ 添加 **openstack** 用户

`rabbitmqctl add_user openstack (用户名) RABBIT_PASS (密码)`

**解释：**

此账户相当于是 RabbitMQ 的管理者，所以后面的密码需要记住。OpenStack 中每个组件或者服务都存在一个管理者，用于管理此服务或者组件。

- ◆ 给 **openstack** 用户配置写和读权限

```
rabbitmqctl set_permissions openstack (用户名) ".*" ".*" ".*"
```

- ◆ 看下监听端口，rabbitmq 使用的是 5672 端口

```
netstat -ntlp | grep 5672
```

- ◆ 安装 Memcached

```
apt-get install memcached python-memcache
```

- ◆ 编辑/etc/memcached.conf 文件和配置服务使用控制节点管理 IP 地址，以此使其它节点通过管理网访问控制节点。

```
-l 10.0.0.11
```

### 9.1.3 在控制节点安装和配置认证服务 Keystone

为 OpenStack 的各种服务提供认证和权限管理服务，简单的说 OpenStack 上的每一步操作都需要通过 Keystone 审核。

其中涉及到**认证**和**授权**两个操作。

**认证：**相当于用户、服务或者组件，向 Keystone 认证自己是否合法，然后返回一个 token。合法的 token 才能进行具体的命令操作，相当于会将 token 包装在命令数据包中进行实际的流转。只有通过认证的 token 才会被可以执行具体数据包中的其他命令。

**授权：**而 Keystone 会在返回 token 的时候，具体指明此认证的用户、服务或者组件所具有的权限，即你具体可以做什么操作。

**认证：解决你是谁的问题。授权：解决你能干什么的问题。**

◆ 根据如上所述的，在集群控制节点上，每次创建一个服务就要创建一个数据库、服务凭证和 API 端点（endPoint），详细说就是：

- 1、创建一个数据库
- 2、创建一个数据库管理员
- 3、获得自由管理员才能操作的凭证（上面的脚本）
- 4、创建一个服务实体，相当于 new 一个对象。
- 5、创建三个认证服务端点的变种，就是如上针对于 public, internal, admin

在具体节点上（控制节点）：

- 1、安装服务
- 2、配置服务，包括修改连接数据库地址文件，修改 Keystone 认证过程的配置文件等
- 3、重启服务

◆ 创建 keystone 数据库

```
mysql -u root -p //密码为上文安装 MySQL 数据库服务的时候的 root 密码  
CREATE DATABASE keystone; //创建一个 Keystone 数据库
```

◆ 对 keystone 数据库授予恰当的权限，如上一致，相当于创建管理员 Keystone，并且授予其最高权限，所谓的管理员。每个服务，组件等都会存在一个管理员。

```
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost'  
IDENTIFIED BY '输入 keystone 数据库的管理密码';
```

解释：

keystone.\*：表示对 keystone 数据库中的所有表（\*）

'keystone'@'localhost'：前面的 keystone 表示是一个用户，相当于对谁赋权限。

后面的 localhost 相当于指明用户的位置，localhost 表示在本机的 Keystone 用户对数据库访问。

后面输入的密码是：keystone 用户的密码

```
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY '输入 keystone 数据库的管理密码';
```

解释:

'keystone'@'%': 这里唯一的差别, 在于 KeyStone 用户可以处于任何主机。相当于用 '%' 表示从任何地址连接。

刷新数据库 flush privileges;

退出数据库 exit

- ◆ 生成一个随机值在初始的配置中作为管理员的令牌 (后面你将会知道, 其实我们是执行了一个文件, 以此获得管理员的操作权限, 但是此处由于我们连 keystone 都还没有安装, 恰恰在安装 keystone 所以临时生成令牌环文件)

```
openssl rand -hex 10
```

- ◆ 安装后禁止 KeyStone 服务自动启动  
echo "manual" > /etc/init/keystone.override

- ◆ 安装 KeyStone 服务软件  
sudo apt-get install keystone apache2 libapache2-mod-wsgi

- ◆ 进行配置 KeyStone 服务软件:

- 编辑文件 **/etc/keystone/keystone.conf**

在 **[DEFAULT]** 部分, 定义管理员 token 初始值及启用详细日志:

```
[DEFAULT]
```

```
...
```

```
admin_token = ADMIN_TOKEN
```

**//用你在前一步生成的随机数替换 ADMIN\_TOKEN**

```
verbose = True    //启用详细日志
```

- 在 **[database]** 部分, 配置数据库访问:

```
[database]
```

```
...
```

```
connection = mysql+pymysql://keystone:KEYSTONE_DBPASS@controller/keystone
```

解释:

将原有的 connection 删除或注释, 替换后面的 KeyStone 用户的管理员密码 (绿色), 由于默认设置 keyStone 数据库的管理员是 keystone 用户, 所以上面的用户名不需要修改。此相当于配置访问连接, KeyStone 服务访问 KeyStone 所属的数据库, 肯定是需要访问连接和用户名密码的, 所以我们将其填写在此。

这个用户名和密码必须是 KeyStone 数据库的管理员, 相当于 KeyStone 服务全权可以访问其自己的数据库。后面的 controller 表示连接的控制节点, **连接会走管理网络**, 由于我们配置了域名解析, 所以只需要填写主机名字就可。命令的完整意思是: 采用用户名 keystone 和密码 keyStone\_dbpass 去连接控制节点 controller 上的 keyStone 数据库。

这句命令也对应，本地对数据库的连接，我们之前也已经授权了本地的权限，如本节命令【对 KeyStone 数据库授予恰当的权限】。

- 在[token]部分，配置 Fernet UUID 令牌的提供者。

```
[token]
```

```
...
```

```
provider = fernet
```

- ◆ 初始化身份认证服务的数据库

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

- ◆ 初始化 Fernet keys

```
keystone-manage fernet_setup --keystone-user keystone --keystone-group  
keystone
```

---

排查方式：

需要先 source 权限脚本，再输入如下命令

列出所有用户：openstack user list

列出认证服务目录：openstack catalog list

### 9.1.3.1 配置 Apache HTTP 服务器

- ◆ 为什么安装此 Apache（仅我个人理解）：

**解释：**控制节点很多服务需要暴露出去，各服务及组件认证的时候通过 RestAPI 形式，其本质上是一个基于 https 形式的 URL。所以，需要有 ApacheHTTP 服务器去承载和接收请求。RestAPI 涉及到 endpoint 概念，我将在后面讲介绍。

- ◆ 编辑 **/etc/apache2/apache2.conf** 文件，配置 **ServerName** 选项为控制节点：

ServerName controller

直接在文件后面黏贴

**解释：**

直接在文件后面添加此语句，目的是指明服务器的节点名字，此名字已经在上文网络配置中设定，其他节点单独 ping controller 的时候也是可以通，说明 DNS 主机名解析是正常的！这点需要配合网络 DNS 原理中的“ping 主机名”和“ping IP”理解，**连接会走管理网络**。

- ◆ 使用下面内容创建 **/etc/apache2/sites-available/wsgi-keystone.conf** 文件

```
Listen 5000
Listen 35357

<VirtualHost *:5000>
    WSGIDaemonProcess keystone-public processes=5 threads=1 user=keystone
    group=keystone display-name=%{GROUP}
    WSGIProcessGroup keystone-public
    WSGIScriptAlias / /usr/bin/keystone-wsgi-public
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    ErrorLogFormat "%{cu}t %M"
    ErrorLog /var/log/apache2/keystone.log
    CustomLog /var/log/apache2/keystone_access.log combined

    <Directory /usr/bin>
        Require all granted
    </Directory>
</VirtualHost>

<VirtualHost *:35357>
    WSGIDaemonProcess keystone-admin processes=5 threads=1 user=keystone
    group=keystone display-name=%{GROUP}
    WSGIProcessGroup keystone-admin
    WSGIScriptAlias / /usr/bin/keystone-wsgi-admin
    WSGIApplicationGroup %{GLOBAL}
```

```
WSGIPassAuthorization On
ErrorLogFormat "%{cu}t %M"
ErrorLog /var/log/apache2/keystone.log
CustomLog /var/log/apache2/keystone_access.log combined

<Directory /usr/bin>
    Require all granted
</Directory>
</VirtualHost>
```

◆ **注意：**上面的用户名 keystone 就是我们安装 KeyStone 服务时管理员名字！并且指明了错误日志的保存地址！以上我用**红色标记**！

◆ 开启认证服务虚拟主机

```
ln -s /etc/apache2/sites-available/wsgi-keystone.conf /etc/apache2/sites-enabled
```

**解释：**  
将以上**新建的 wsgi-keystone.conf 文件**建立一个软连接，使配置的文件生效。

◆ 完成安装，并重新启动 Apache 服务器

```
service apache2 restart
```

◆ 默认情况下，Ubuntu 上的安装包会自动创建一个 SQLite 数据库。  
因为这里我们配置使用 MySQL 数据库服务器，所以删除 SQLite 服务库文件：

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

### 9.1.3.2 创建服务实体和 API 端点

身份认证服务提供**服务**的目录和他们的位置。

每个你添加到 OpenStack 环境中的服务在目录中需要一个 **service** 实体和一些 **API endpoints** (3 类, 分别对应 **public, internal, admin**)。

相当于: 每个添加到 OpenStack 中的服务都需要在认证中心 (KeyStone) 中登记注册, 否则认为非法, 这个很好理解, 反过来也说明了, 我们上文为啥需要安装 ApacheHTTP 服务器, 这就是为了认证通讯。而上文又说了 API endpoints, 其实这就是用来暴露给外部不同域连接使用, 所谓的外部可以是集群外部, 也可以是集群内部不同域, 用户可以通过远程调用的形式访问集群中的某些服务组件。集群中的某服务组件, 都已经被封装了接口, 对外暴露就是这个 endpoints 形式。我们远程调用的时候, 其实就是访问某个 endpoints, 采用的形式就是 http 的 post 形式! 具体请查询相关书籍。

#### **接下来进行注册各个服务及组件!**

身份认证服务提供服务的目录和他们的位置。每个你添加到 OpenStack 环境中的服务, **在目录中需要一个 service 实体和一些 API endpoints**。

创建不一样的角色权限, 并且赋予不同的用户, 其具有的操作权限是不一致的, 并且还需要考虑项目, 或者用户, 处于哪个域, 哪个项目等等一系列问题。

但是, 身份认证服务的数据库不包含支持传统认证和目录服务的信息。你必须使用身份认证服务创建的临时身**份验证令牌**用来初始化的服务实体和 API 端点。

你必须使用 `-os-token` 参数将认证令牌的值传递给 `command:openstack` 命令。

你必须使用 `-os-url` 参数将身份认证服务的 URL 传递给 `openstack` 命令或者设置 `OS_URL` 环境变量。

**解释:**

◆ 配置认证令牌(这个过程, 后续我们会采用文件形式保存并获得管理员权限, 为集群中其他服务、组件的安装, 提供管理员权限, 如下 `admin-openrc.sh`):  
`export OS_TOKEN=ADMIN_TOKEN`  
\\将 `ADMIN_TOKEN` 替换为你在 **keystone-install** 中生成的认证令牌  
比如: `export OS_TOKEN=294a4c8a8a475f9b9836` (其实也不一定要用随机生成, 上文, 只要集群中, 大家统一就可以了, 比如都用 `ADMIN`)  
配置端点 URL(同样采用主机名字表示**走管理网络**, 可以正常解析, 无需 IP):  
`export OS_URL=http://controller:35357/v3`  
配置认证 API 版本:  
`export OS_IDENTITY_API_VERSION=3`

**接下来标准动作: 一个对象, 三个服务点**

一个服务实体, 相当于 **new** 一个对象。

三个认证服务端点的变种, 就是如上针对于 **public, internal, admin**

◆ 创建服务实体和 API 端点

在你的 Openstack 环境中，认证服务管理服务目录。服务使用这个目录来决定您的环境中可用的服务。

为身份认证服务创建服务实体：（相当于 JAVA 概念中：为身份认证服务（类）new 一个对象，这过程就是为服务创建一个服务实体的过程！！）

```
openstack service create --name keystone --description "OpenStack Identity" identity
```

```
$ openstack service create \
  --name keystone --description "OpenStack Identity" identity
+-----+-----+
| Field | Value |
+-----+-----+
| description | OpenStack Identity |
| enabled | True |
| id | 4ddaae90388b4ebc9d252ec2252d8d10 |
| name | keystone |
| type | identity |
+-----+-----+
```

解释：

身份认证服务（keyStone 服务）管理了一个与你环境相关的 API 端点的目录（endpoint 目录）。服务使用这个目录来决定如何与你环境中的其他服务进行通信。

创建一个服务叫 keystone，描述信息为“OpenStack 认证服务”。

我自己的理解是：类似于 java 生成对象的思想，用类 keystone 创建一个对象 keystone 作为 identity 的服务对象。

◆ OpenStack 使用三个 API 端点的变种代表每种服务：admin、internal 和 public。默认情况下，

管理 API 端点：允许修改用户和租户而公共和内部 APIs 不允许这些操作。在生产环境中，出于安全原因，变种为了服务不同类型的用户可能驻留在单独的网络上（由于管理 API 端点权限太大，所以需要进行适当的权限划分，故出现了变种端点）。

公共 API 网络（public）为了让顾客管理他们自己的云在互联网上是可见的，对于外网访问集群。

管理 API 网络在管理云基础设施的组织中操作也是有所限制的。

内部 API 网络（internal）可能会限制在包含 OpenStack 服务的主机上。

此外，OpenStack 支持可伸缩性的多区域。

本指南，internal 和 admin 端点变种和默认 RegionOne 区域使用走管理网络，而 public 表示集群对外服务，所以需要设置为控制节点外网口的 IP 地址。

◆ 创建认证服务的 API 端点：

Public:

```
openstack endpoint create --region RegionOne identity public \
http://controller:5000/v3
```

解释：

endpoint 端点的创建，一个 region 区域，public 权限

访问 endpoint 点是：控制节点的 5000 端口。由于此为集群对外提供服务，所以将 controller 更换为**控制节点外网口的 IP 地址**。

Field	Value
enabled	True
id	30fff543e7dc4b7d9a0fb13791b78bf4
interface	public
region	RegionOne
region_id	RegionOne
service_id	8c8c0927262a45ad9066cfe70d46892c
service_name	keystone
service_type	identity
url	http://controller:5000/v2.0

Internal:

```
openstack endpoint create --region RegionOne identity internal \
http://controller:5000/v3
```

解释：

Internal 将走**管理网络**，在集群内部，所以填写主机名，这样其他节点也是可以通过**管理网络**进行正常的主机名解析的。

Field	Value
enabled	True
id	57cfa543e7dc4b712c0ab137911bc4fe
interface	internal
region	RegionOne
region_id	RegionOne
service_id	6f8de927262ac12f6066cfe70d99ac51
service_name	keystone
service_type	identity
url	http://controller:5000/v2.0

Admin:

```
openstack endpoint create --region RegionOne identity admin \
http://controller:35357/v3
```

解释：

admin 将走**管理网络**，在集群内部，所以填写主机名，这样其他节点也是可以通过**管理网络**进行正常的主机名解析的。

Field	Value
enabled	True
id	78c3dfa3e7dc44c98ab1b1379122ecb1
interface	admin
region	RegionOne
region_id	RegionOne
service_id	34ab3d27262ac449cba6cfe704dbc11f
service_name	keystone
service_type	identity
url	http://controller:35357/v2.0

每个添加到 OpenStack 环境中的服务要求一个或多个服务实体和三个认证服务中的 **API 端点的变种**。

一个服务实体，相当于 new 一个对象。

三个认证服务端点的变种，就是如上针对于 public, internal, admin

创建三个 endpoint 端点，其中**特别注意**地址后面的 controller，其实是指明控制节点的地址。由于三个服务对象的作用域不一样，所以在集群内部，internal

和 admin 集群将通过**走管理网络**，进行域名解析（全部采用域名解析的形式）。  
**而 public 是对外，相当于是对集群外部访问，所以我们将其更换为控制节点，外网卡的 IP 地址。**

换句话说，输入 controller 机器能解析到其 IP 地址。所以集群在配置的时候，必须都要很清晰的完成，在 ping 主机的时候，后面写 IP 地址和直接写主机名字都可。能正确的完成域名解析。好处就是：IP 地址的变动，不会影响集群的配置，所以集群内部，我们统一使用**走管理网络**，进行域名解析，填写主机名字是没错的。

### 9.1.3.3 创建项目、用户和角色

- ◆ 身份认证服务为每个 OpenStack 服务提供认证服务。认证服务是使用 domains, projects (tenants), users<user>和 roles<role>的组合。

解释:

相当于, 属于哪个域, 哪个项目, 哪个用户具有怎样的规则等等。

本文全局默认采用 default 域, 相当于一个域

- ◆ 创建域 default

```
openstack domain create --description "Default Domain" default
```

- ◆ 在你的环境中, 为进行管理操作, 创建管理的项目、用户和角色

```
openstack project create --domain default --description "Admin Project" admin
```

解释:

在域 default 中创建 admin 项目, 前面为 project create, 最后的 admin 表示项目名称, 此处仅表示一个名字, 并不是说是一个什么管理的项目等等, 这句话与后面的创建角色规则 admin 要区分开。

```
$ openstack project create --domain default \
--description "Admin Project" admin
+-----+-----+
| Field | Value |
+-----+-----+
| description | Admin Project |
| domain_id | default |
| enabled | True |
| id | 343d245e850143a096806dfaefa9afdc |
| is_domain | False |
| name | admin |
| parent_id | None |
+-----+-----+
```

同样的原理, 我们还需要创建服务 (后面安装 glance,nova 等等需要使用) 对象

```
openstack project create --domain default --description "Service Project" service
```

- ◆ 在域 default 中创建 admin 项目的 admin 管理用户

```
openstack user create --domain default --password-prompt admin
```

此处输入的密码, 请记住, 后面在重新生成 token 的时候会用到。

解释:

前面 user create 表示用户创建, 并且在域 default 中, 后面的 admin 表示创建的用户名。此处仅表示的一个用户名, 不是说我命名为 admin, 他就是一个 admin 管理员了, 还没有给他具体分配角色权限。

```
$ openstack user create --domain default \
--password-prompt admin
User Password:
Repeat User Password:
+-----+-----+
| Field | Value |
+-----+-----+
| domain_id | default |
| enabled | True |
| id | ac3377633149401296f6c0d92d79dc16 |
| name | admin |
+-----+-----+
```

◆ 创建一个 admin 角色

openstack role create admin

解释：

前面 role create 表示规则创建，创建 admin 规则。注意此处的 admin 表示创建的是 admin 规则，是最高管理员规则。但是这此仅仅创建了规则，还没有将规则赋予哪个用户所拥有。

```
$ openstack role create admin
+-----+-----+
| Field | Value |
+-----+-----+
| id | cd2cb9a39e874ea69e5d4b896eb16128 |
| name | admin |
+-----+-----+
```

◆ 添加 admin 角色到 admin 项目和用户上：

openstack role add --project admin --user admin admin

解释：

前面 role add 表示权限赋予，赋予 admin 项目的 admin 用户，最后面的 admin 表示角色规则，就是我们刚刚创建的。

**这样，我们就能把关系整理清楚，admin 项目中的 admin 用户，其实是具有 admin 角色权限的一个用户，即，此 admin 用户是一个管理员！**

重复此过程创建额外的项目和用户等等。

### 9.1.3.4 KeyStone 安装配置，验证操作

在安装其他服务之前确认身份认证服务的操作，因为安全性的原因，关闭临时认证令牌机制（还记得我们之前采用随机生成了一个临时令牌环吗？然后用这个临时的令牌环，我们创建并建立了 keystone，现在相当于我们将认证中心，keystone 安装好了，就必须关闭临时认证机制，之后所有的服务组件安装，都需要获得最高管理员权限，才可以往我们的集群中安装或者添加服务、组件等）。

- ◆ 编辑 `/etc/keystone/keystone-paste.ini` 文件，
  - 从 `[pipeline:public_api]`，`[pipeline:admin_api]` 和 `[pipeline:api_v3]` 部分删除 `admin_token_auth`。

- 重置 `OS_TOKEN` 和 `OS_URL` 环境变量：  
`unset OS_TOKEN OS_URL`

- ◆ 使用 admin 用户，重新请求认证令牌：  
`openstack --os-auth-url http://controller:35357/v3 \`  
`--os-project-domain-name default --os-user-domain-name default \`  
`--os-project-name admin --os-username admin token issue`

解释：

admin 用户就是上文创建的 keystone 管理员用户，相当于后继的操作我们都需要他来赋予凭证，然后进行集群中的组件、服务安装。

此处 `--os-auth-url` 指明 endpoint 的地址，向这个 endpoint 进行注册（我的理解）

`--os-project-domain-name` 指明项目域是哪个！

`--os-user-domain-name` 指明用户域是哪个！

`--os-project-name` 指明项目名称是哪个！

`--os-username` 指明用户是哪个！

（走管理网络）

```
$ openstack --os-auth-url http://controller:35357/v3 \
--os-project-domain-name default --os-user-domain-name default \
--os-project-name admin --os-username admin token issue
Password:
+-----+
| Field | Value |
+-----+
| expires | 2016-02-12T20:14:07.056119Z |
| id | gAAAAABWvi7_B8kKQD9wdXac8MoZiQldmJE0643d-e_j-XXq9AmIegIbA7UHGpV |
| | atnN21qtOMjCFWx7BReJEQnVOA3nc1RQgAYRsfSU_Mrsuwb4EDtnjU7HEpo8b4 |
| | o6ozsA_NmFWEpLeKy0uNn_WeKbAhYygrsmQGA49dc1HVnz-OMVLiyM9ws |
| project_id | 343d245e850143a096806dfeafa9afdc |
| user_id | ac3377633149401296f6c0d92d79dc16 |
+-----+
```

相当于刚刚都是采用临时令牌处理的，但是不安全，所以我们需要对用户 admin 进行重新的申请认证令牌，以后就使用这个新申请的认证令牌进行操作！这里我们对 admin 用户进行认证令牌重新申请，由于已经关联了 admin 用户具有 admin 的角色权限，所以换句话说，此处是对 **管理员** 用户进行认证令牌重新申请！（而一开始采用临时的令牌，因为我们连 keystone 都还没有安装，所有需

要用一个临时的进行授权，之后由于 keystone 已经安装好了，所有我们就需要使用真正的 token 了）

下面的这句话仅仅用于加深理解，无需配置！具体请参考：

[https://docs.openstack.org/liberty/zh\\_CN/install-guide-ubuntu/keystone-verify.html](https://docs.openstack.org/liberty/zh_CN/install-guide-ubuntu/keystone-verify.html)

[https://docs.openstack.org/liberty/zh\\_CN/install-guide-ubuntu/keystone-users.html](https://docs.openstack.org/liberty/zh_CN/install-guide-ubuntu/keystone-users.html)

如果：创建一个具有 user 角色权限，并且赋予某个域，某个项目的某一个用户，那么其仅仅具有非管理的权限。

```
$ openstack --os-auth-url http://controller:5000/v3 \
--os-project-domain-name default --os-user-domain-name default \
--os-project-name demo --os-username demo token issue
Password:
+-----+
| Field | Value |
+-----+
| expires | 2016-02-12T20:15:39.014479Z |
| id | gAAAAABWvi9bsh7vkiby5BpCCnc-JkbGhm9wH3fabS_cY7uab0ubesi-Me6IGWW |
| | yQqNegDDZ5jw7grI26vvgy1J5nCVwZ_zFRqPiz_qhbq29mgbQLglbkq6FQvzBRQ |
| | JcOzq3uwhzNxsZJWmzGC7rJE_H0A_a3UFhqv8M4zMRYSbS2YF0MyFmp_U |
| project_id | ed0b60bf607743088218b0a533d5943f |
| user_id | 58126687cbcc4888bfa9ab73a2256f27 |
+-----+
```

这个命令使用 demo 用户的密码和 API 端口 5000，这样只会允许对身份认证服务 API 的常规（非管理）访问。

### 9.1.3.5 创建 OpenStack 客户端环境脚本

前一节中使用环境变量和命令选项的组合通过 OpenStack 客户端与身份认证服务交互。为了提升客户端操作的效率，OpenStack 支持简单的 **客户端环境变量脚本即 OpenRC 文件**，因为对集群的操作（包括服务的注入，组件的安装），都需要向 keyStone 获得 admin 管理员的凭证才可，以此构建这份自动化文档。

**解释：**

相当于我们前期进行操作的时候，都是采用命令行一行行输入的，而假设做成脚本形式，在客户端直接注入配置的话，将大大提高效率，故我们构建了此脚本，我们发现此脚本中的参数数值与使用 **admin 用户，重新请求认证令牌** 内设置是一样的，仅仅此脚本是为了加快部署速度。

而 user 形式的生成，与上文重新 user 进行 token 生成是一样的道理，不在简述，具体如参考地址：

[https://docs.openstack.org/liberty/zh\\_CN/install-guide-ubuntu/keystone-verify.html](https://docs.openstack.org/liberty/zh_CN/install-guide-ubuntu/keystone-verify.html)

- ◆ 编辑创建文件 **admin-openrc** 并添加如下内容（**此处采用管理员权限**）

```
export OS_PROJECT_DOMAIN_NAME=default //指明项目处于什么域
export OS_USER_DOMAIN_NAME=default //指明用户处于什么域
export OS_PROJECT_NAME=admin //指明项目的名字
export OS_USERNAME=admin //指明用户名名字
export OS_PASSWORD=ADMIN_PASS //指明用户名的密码
export OS_AUTH_URL=http://controller:35357/v3 //认证的 URL，也说明了之前为啥会装个 apache 服务器，其实 endpoint 暴露出来之后，其认证的形式，都是采用 URL 进行的（controller 说明走管理网络）。
export OS_IDENTITY_API_VERSION=3 //API 版本号
export OS_IMAGE_API_VERSION=2 //后面镜像时候用
```

**注意：**将 **ADMIN\_PASS** 替换为你在认证服务中为 **admin** 用户选择的密码。

- ◆ 加载“**admin-openrc**”文件来身份认证服务的环境变量位置和“**admin**”项目和用户证书(相当于验证一次文件，此文件可以 Copy 到其他节点上使用):

. admin-openrc

- ◆ 请求认证令牌:

openstack token issue

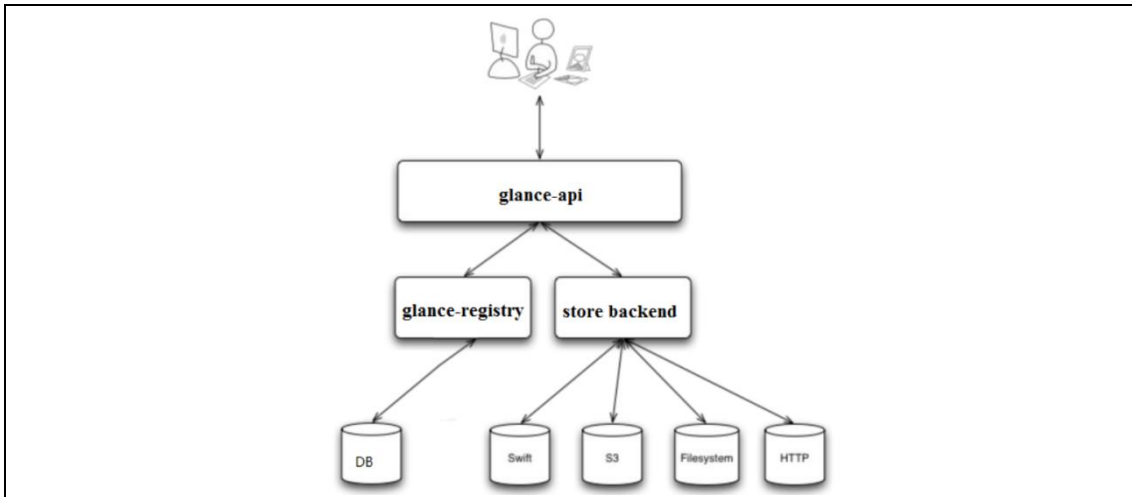
- ◆ 验证 admin 账户:

openstack user list

```
root@controller:~# openstack token issue
+-----+
| Field | Value |
+-----+
| expires | 2017-09-26T13:34:21.000000Z |
| id | gAAAAABZyXlnz-f0Da1Io0L0cGp_xt8fyk8fBvbPjpsLamyNz910P0/TtNI9tyxmc-ikwg6K1sTu0LlwSZ_YDXDnJxgnRMiKG4DluVnz3i7RfM2rgBpH1yeyBPH0S0TtBoby_thxb3Lekf307sy1f0-k0ghGEX-s8b8hpFhwkV8gYRmRrRVNQ |
| project_id | b9e63Ffd869942aaefdea46a5cfa805 |
| user_id | efdb6241159c46cdaeb604b443227f4 |
+-----+
root@controller:~# openstack user list
+-----+
| ID | Name |
+-----+
| efdb6241159c46cdaeb604b443227f4 | admin |
+-----+
root@controller:~#
```

## 9.1.4 在控制节点上安装和配置镜像服务 Glance

OpenStack 的镜像服务 (glance) 允许用户发现、注册和恢复虚拟机镜像。它提供了一个 REST API, 允许您查询虚拟机镜像的 metadata 并恢复一个实际的镜像。



1、Glance-api 是后台运行的服务进程, 对外提供 RestAPI, 响应 image 查询、获取和存储的调用。但是 glance-api 不会真正处理请求。

2、如果是与 image-metadata(元数据)相关操作, glance-api 会把请求转发给 glance-registry; 如果是与 image 自身存取想关的操作, glance-api 会把请求转发给 image 的 store-backend。

3、glance-registry 是系统后台运行的服务进程, 复杂处理和存储 image 的 metadata, 例如 image 的大小和类型。

4、image 的 metadata 会保存到 database 中, 默认 mysql 数据库。

5、glance 自身并不存 image, 真正的 image 是存放在 backend 中的:

A directory on a local file system

Gridfs

Ceph rbd (V2.0 的时候统一存储使用)

Amazon s3

Sheepdog

Openstack block storage(cinder)

Openstack object storage(swift)

Vmware esx

◆ 根据如上所述的, 在控制节点上, 每次创建一个服务就要创建一个数据库、服务凭证和 API 端点 (endpoint), 详细说就是:

1、创建一个数据库

2、创建一个数据库管理员

3、获得自由管理员才能操作的凭证 (上面的脚本)

4、创建一个服务实体, 相当于 new 一个对象。

5、创建三个认证服务端点的变种, 就是如上针对于 public, internal, admin

在具体节点上（控制、计算，存储节点）：

- 1、安装服务
- 2、配置服务，包括修改连接数据库地址文件，修改 keystone 认证过程的配置文件等
- 3、重启服务

具体控制管理是安装在控制节点上（服务注册等），而具体负责存储等是在存储节点上。在控制节点仅仅安装，服务的访问点，数据库管理等等。存储节点都是通过远程连接形式访问控制节点的相应数据库。

◆ 创建 Glance 数据库

```
mysql -u root -p          //连接数据库
CREATE DATABASE glance;    //创建一个 Glance 数据库
```

◆ 对 Glance 数据库赋予权限：

```
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' IDENTIFIED
BY '输入 glance 用户的密码 GLANCE_DBPASS';
```

解释：

`glance.*`：表示对 glance 数据库中的所有表（\*）

`'glance'@'localhost'`：前面的 glance 表示是一个用户，相当于对谁赋权限。

后面的 localhost 相当于指明用户的位置，localhost 表示在本机 glance 用户对数据库访问。后面输入的密码是：glance 用户的密码，相当于创建一个对 glance 数据库的管理员，被赋予权限。

```
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' IDENTIFIED BY '输
入 glance 用户的密码 GLANCE_DBPASS';
```

解释：

`'glance'@'%'`：这里唯一的差别，在于 glance 用户可以处于任何主机。相当于用 '%' 表示从任何地址连接。

刷新数据库 `flush privileges;`

退出数据库 `exit`

◆ 以下的操作需要最高级管理员才可以操作，比如往集群里面注册一个服务什么的，所以需要获得管理员凭证，故：**获得 admin 凭证来获取只有管理员能执行命令的访问权限**

```
. admin-openrc
```

//此文件内容，我已经在上部解释，此文件及其内容可以先前建立好。

◆ 创建 Glance 用户

```
openstack user create --domain default --password-prompt glance
```

解释：

在缺省域，创建一个用户 glance。其实就是创建一个 Glance 镜像服务的管理者，名字为 glance。但是注意，此处我们仅仅是创建了一个用户，并没给其赋予权限，具体如下所述。（**命令执行的时，需要执行设定密码-请记住！**）

- ◆ 添加 **admin** 角色到 **glance** 用户和 **service** 项目上  
`openstack role add --project service --user glance admin`

解释:

将 **admin** 角色权限赋予给 **glance** 镜像服务的管理者 **glance**。这就相当于用户 **glance** 具有了管理员的权限。`--user glance` 就是指明那个用户，最后面的 **admin** 表示角色权限（我们之前设定的）。

接下来标准动作：一个对象，三个服务点

一个服务实体，相当于 **new** 一个对象。

三个认证服务端点的变种，就是如上针对于 **public, internal, admin**

- ◆ 创建一个服务实体

```
openstack service create --name glance --description "OpenStack Image service"
image
```

解释:

创建一个服务，名字叫 **glance**，描述信息为“OpenStack 镜像服务”。

我自己的理解是：类似于 **java** 生成对象的思想，用类 **Glance** 创建一个对象 **image** 作为 **glance** 的服务对象。

- ◆ 创建三类 **endpoint** 端点，分别针对于 **public, internal, admin**

```
openstack endpoint create --region RegionOne image \
public http://controller:9292
```

//controller 换成 **glance** 节点外网口的 IP 地址（因为具体存储服务我们安装在 **glance** 节点上）

```
openstack endpoint create --region RegionOne image \
internal http://glance:9292
```

//走管理网络

```
openstack endpoint create --region RegionOne image \
admin http://glance:9292
```

//走管理网络

解释:

创建三个 **endpoint** 端点，其中特别注意地址后面的 **controller**，换成 **glance** 节点外网口的 IP 地址。由于三个服务对象的作用域不一样，所以在集群内部，**internal** 和 **admin** 集群将通过走管理网络，进行域名解析（全部采用域名解析的形式）。而 **public** 是对外，相当于是对集群外部访问，所以我们将其更换为 **glance** 节点，外网卡的 IP 地址。

换句话说，输入 **glance** 机器能解析到其 IP 地址。所以集群在配置的时，必须都要很清晰的完成。在 **ping** 主机的时候，后面写 IP 地址和直接写主机名字都可。能正确的完成域名解析。好处就是：IP 地址的变动，不会影响集群的配置，所以集群内部，我们统一使用走管理网络进行域名解析，填写主机名字是没错的。

特别注意：

此处为什么我仅仅写了 **glance** 的地址，具体请看后文手写版本说明。

**Glance** 没有类似于 **nova**, **cinder** 这类的调度器，后面还带几个具体的组件。**Glance** 就只有一个。

所以，我们若单节点部署，那么其在 **keystone** 注册里面的服务暴露点，就必须填写具体安装 **glance** 节点的地址信息。也就是为什么，我们需要填写 **glance** 管理网络地址，及 **glance** 外网地址。

查看镜像列表：

```
. admin-openrc  
glance image-list
```

查询 endpoint 用  
openstack endpoint list

```
lab316@controller:~$ source admin-openrc.sh  
lab316@controller:~$ glance image-list  
+-----+-----+  
| ID | Name |  
+-----+-----+  
| 36209e51-ddef-4768-8c44-130cc466fa1d | ubuntu14-image |  
| 27eba2af-f1a0-42df-a69b-3eee6403e52b | windows7-64 |  
| 8a967c3d-4b0f-4254-9a71-04544fc4a4d4 | winServer08R2 |  
+-----+-----+
```

## 9.1.5 在控制节点上安装和配置块存储服务 Cinder

- ◆ 根据如上所述的，在控制节点上，每次创建一个服务就要创建一个数据库、服务凭证和 API 端点（endpoint），详细说就是：

- 1、创建一个数据库
- 2、创建一个数据库管理员
- 3、获得自由管理员才能操作的凭证（上面的脚本）
- 4、创建一个服务实体，相当于 new 一个对象。
- 5、创建三个认证服务端点的变种，就是如上针对于 public, internal, admin

在具体节点上（控制、计算，存储节点）：

- 1、安装服务
- 2、配置服务，包括修改连接数据库地址文件，修改 keystone 认证过程的配置文件等
- 3、重启服务

- ◆ 创建 Cinder 数据库

```
mysql -u root -p //连接数据库
CREATE DATABASE cinder; //创建一个 Cinder 数据库
```

- ◆ 对 Cinder 数据库赋予权限：

```
GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' IDENTIFIED
BY '输入 cinder 用户的密码 CINDER_DBPASS';
```

解释：

`cinder.*`：表示对 glance 数据库中的所有表（\*）

`'cinder'@'localhost'`：前面的 cinder 表示是一个用户，相当于对谁赋权限。

后面的 localhost 相当于指明用户的位置，localhost 表示在本机 cinder 用户对数据库访问。后面输入的密码是：cinder 用户的密码，相当于创建一个对 cinder 数据库的管理员，被赋予权限。

```
GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' IDENTIFIED BY '输入
cinder 用户的密码 CINDER_DBPASS';
```

解释：

`'cinder'@'%'`：这里唯一的差别，在于 cinder 用户可以处于任何主机。相当于用 '%' 表示从任何地址连接。

刷新数据库 `flush privileges`;

退出数据库 `exit`

- ◆ 以下的操作需要最高级管理员才可以操作，比如往集群里面注册一个服务什么的，所以需要获得管理员凭证，故：**获得 admin 凭证来获取只有管理员能执行命令的访问权限**

```
. admin-openrc
```

//此文件内容，我已经在上部解释，此文件及其内容可以先前建立好。

◆ 创建 cinder 用户

```
openstack user create --domain default --password-prompt cinder
```

解释:

在缺省域，创建一个用户 cinder。其实就是创建一个 cinder 块存储服务的管理者，名字为 cinder。但是注意，此处我们仅仅是创建了一个用户，并没给其赋予权限，具体如下所述。*(命令执行的时，需要执行设定密码-请记住!)*

◆ 添加 admin 角色到 cinder 用户和 service 项目上

```
openstack role add --project service --user cinder admin
```

解释:

将 admin 角色权限赋予给 cinder 块存储服务的管理者 cinder。这就相当于用户 cinder 具有了管理员的权限。--user cinder 就是指明那个用户，最后面的 admin 表示角色权限（我们之前设定的）。

接下来标准动作：一个对象，三个服务点

一个服务实体，相当于 new 一个对象。

三个认证服务端点的变种，就是如上针对于 public, internal, admin

◆ 块存储设备需要两个实体（与其他有区别）

创建 cinder 和 cinderv2 服务实体

```
openstack service create --name cinder --description "OpenStack Block Storage" volume
```

```
openstack service create --name cinderv2 --description "OpenStack Block Storage" volumev2
```

解释:

创建两个服务，名字叫 cinder 和 cinderv2，描述信息为“OpenStack 块存储服务”。我自己的理解是：类似于 java 生成对象的思想，用类 cinder 创建一个对象 volume 作为 cinder 的服务对象。

◆ 创建三类 endpoint 端点，分别针对于 public, internal, admin

```
openstack endpoint create --region RegionOne \
volume public http://controller:8776/v1/%(tenant_id)s
//controller 换成控制节点外网口的 IP 地址
```

```
openstack endpoint create --region RegionOne volume \
internal http://controller:8776/v1/%(tenant_id)s
//走管理网络
```

```
openstack endpoint create --region RegionOne volume \
admin http://controller:8776/v1/%(tenant_id)s
//走管理网络
```

```
openstack endpoint create --region RegionOne volumev2 \
public http://controller:8776/v2/%(tenant_id)s
//controller 换成控制节点外网口的 IP 地址
```

```
openstack endpoint create --region RegionOne volumev2 \
internal http://controller:8776/v2/%\tenant_id\
//走管理网络
openstack endpoint create --region RegionOne volumev2 \
admin http://controller:8776/v2/%\tenant_id\
//走管理网络
```

**解释：**

创建三个 endpoint 端点，其中特别注意地址后面的 controller，其实是指明控制节点的地址。由于三个服务对象的作用域不一样，所以在集群内部，internal 和 admin 集群将通过走管理网络，进行域名解析（全部采用域名解析的形式）。而 public 是对外，相当于是对集群外部访问，所以我们将其更换为控制节点，外网卡的 IP 地址。

换句话说，输入 controller 机器能解析到其 IP 地址。所以集群在配置的时候，必须都要很清晰的完成，在 ping 主机的时候，后面写 IP 地址和直接写主机名字都可。能正确的完成域名解析。好处就是：IP 地址的变动，不会影响集群的配置，所以集群内部，我们统一使用走管理网络，进行域名解析，填写主机名字是没错的。

#### ◆ 安装 Cinder 包

```
apt-get install cinder-api cinder-scheduler
```

#### ◆ 编辑 /etc/cinder/cinder.conf，同时完成如下操作：

- 在[**DEFAULT**]，配置 RabbitMQ 消息队列，认证服务，启用网络服务支持，启用详细日志，启用后端 LVM 等

```
[DEFAULT]
...
rpc_backend = rabbit           //配置采用 RabbitMQ 作为消息队列
auth_strategy = keystone       //指明，认证采用 KeyStone 服务
my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
//替换为控制节点上的管理网络接口的 IP 地址
verbose = True
```

- 在 [**database**] 部分，配置数据库访问：

```
[database]
...
connection = mysql+pymysql://cinder:CINDER_DBPASS@controller/cinder
```

**解释：**

前面的 cinder 表示 cinder 的 mysql 数据库的管理员。后面的 CINDER\_DBPASS 表示这个 cinder 数据库管理员的密码，而后面的 controller 表示连接的控制节点，连接会走管理网络，由于我们配置了域名解析，所以只需要填写主机名字就可。命令的完整意思是：采用用户名 cinder 和密码 cinder\_dbpass 去连接控制节点 controller 上的 cinder 数据库。

这句命令也对应，本地对数据库的连接，我们之前也已经授权了本地的权限，如本节命令【对 cinder 数据库赋予相应的权限】。

- 在[oslo\_messaging\_rabbit]部分，配置 RabbitMQ 消息队列访问：

```
[oslo_messaging_rabbit]
```

```
...
```

```
rabbit_host = controller //走管理网络，能找到这台主机（能域名解析）
```

```
rabbit_userid = openstack//这个其实是 RabbitMQ 的管理员用户名
```

```
rabbit_password = RABBIT_PASS//所对应的管理员密码
```

- 在[keystone\_authtoken]部分，配置认证服务访问：

```
[keystone_authtoken]
```

```
...
```

```
auth_uri = http://controller:5000
```

```
auth_url = http://controller:35357
```

```
//走管理网络，可以找到控制节点，指明认证的 URL 地址
```

```
memcached_servers = controller:11211
```

```
auth_plugin = password
```

```
//认证采用的形式为密码
```

```
project_domain_name = default
```

```
user_domain_name = default
```

```
project_name = service
```

```
//注册服务实体的名字，注册的是一个服务
```

```
username = cinder
```

```
password = CINDER_PASS
```

```
//此处为在 keystone 认证服务中，注册的 cinder 用户名及其密码。
```

在 [keystone\_authtoken] 中注释或者删除其他选项。

- 在 [oslo\_concurrency] 部分，配置锁路径：

```
[oslo_concurrency]
```

```
...
```

```
lock_path = /var/lib/cinder/tmp
```

- ◆ 初始化块设备服务的数据库

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

- ◆ 配置计算节点以使用块设备存储（等安装完 nova 组件后返过来配置）

- 编辑文件 /etc/nova/nova.conf 并添加如下到其中：

```
[cinder]
```

```
os_region_name = RegionOne
```

再次执行：su -s /bin/sh -c "cinder-manage db sync" cinder

- ◆ 完成安装

- 重启计算 API 服务：  
service nova-api restart
- 重启块设备存储服务：  
service cinder-scheduler restart

```
service cinder-api restart
```

- ◆ 默认情况下，Ubuntu 上的安装包会自动创建一个 SQLite 数据库。因为这里配置使用 SQL 数据库服务器，所以你可以 SQLite 服务库文件：

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

- ◆ 验证是否安装成功

```
. admin-openrc
```

```
cinder service-list
```

```
root@controller:~# cinder service-list
+-----+-----+-----+-----+-----+-----+-----+-----+
| Binary | Host | Zone | Status | State | Updated_at | Disabled Reason |
+-----+-----+-----+-----+-----+-----+-----+
| cinder-scheduler | controller | nova | enabled | up | 2017-06-15T10:18:45.000000 | - |
| cinder-volume | cinder@lvm | nova | enabled | up | 2017-06-15T10:18:43.000000 | - |
+-----+-----+-----+-----+-----+-----+-----+
```

## 9.1.6 在控制节点上安装并配置计算服务 Nova

在控制节点上，安装 nova 的控制服务，具体的计算由计算节点负责，但是在控制节点上，需要部署数据库、管理服务等等。外部用户的入口是控制节点，由暴露的 endpoint 提供访问，并且由计算节点进行具体的实施。

**简单的具体工作流程为：**

- 1、客户（可以理解为 OpenStack 最终用户，也可以是其他的程序）向 API（nova-api）发送请求“帮我创建一个虚拟机”
- 2、API 对请求做一些必要的处理之后，想 Messaging(RabbitMQ) 发送一条消息“让 Scheduler 创建一个虚拟机”。
- 3、Scheduler（nova-scheduler）从 Messaging 获取到 API 发给他的消息，然后执行调度算法，从若干个计算节点中选出节点 A。
- 4、Scheduler 想 Messaging 发送一条消息“在计算节点 A 上创建一个虚拟机”。
- 5、计算节点 A 的 Compute(nova-computing) 从 Messaging 中获取到 Scheduler 发送给他的消息，然后在本节点上的 hypervisor 上创建并启动虚拟机。
- 6、在虚拟机创建的过程中，Compute 如果需要查询或者更新数据库信息，会通过 Messaging 向 Conductor(nova-conductor) 发送消息，Conductor 负责数据库访问。

◆ 根据如上所述的，在控制节点上，每次创建一个服务就要创建一个数据库、服务凭证和 API 端点（endpoint），详细说就是：

- 1、创建一个数据库
- 2、创建一个数据库管理员
- 3、获得只有管理员才能操作的凭证（上面的脚本）
- 4、创建一个服务实体，相当于 new 一个对象。
- 5、创建三个认证服务端点的变种，就是如上针对于 public, internal, admin

在具体节点上（控制、计算，存储节点）：

- 1、安装服务
- 2、配置服务，包括修改连接数据库地址文件，修改 keystone 认证过程的配置文件等
- 3、重启服务

◆ 创建 nova 数据库

```
mysql -u root -p          //连接数据库
CREATE DATABASE nova;     //创建一个 nova 数据库
CREATE DATABASE nova_api;
```

◆ 对 nova 数据库赋予相应的权限

```
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' IDENTIFIED BY
输入 nova 用户的密码 NOVA_DBPASS';
```

**解释：**

**nova.\***：表示对 nova 数据库中的所有表 (\*)

`'nova'@'localhost'`: 前面的 nova 表示是一个用户, 相当于对谁赋权限。后面的 localhost 相当于指明用户的位置, localhost 表示在 localhost 的 nova 用户对数据库访问。后面输入的密码是: `nova` 用户的密码, 相当于创建一个对 nova 数据库的管理员, 被赋予权限。

```
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' IDENTIFIED BY '输入  
nova 用户的密码 NOVA_DBPASS';
```

解释:

`'nova'@'%'`: 这里唯一的差别, 在于 nova 用户可以处于任何主机。相当于用 '%' 表示从任何地址连接。

```
GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' \  
IDENTIFIED BY 'NOVA_DBPASS';  
GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' \  
IDENTIFIED BY 'NOVA_DBPASS';
```

刷新数据库 flush privileges;

退出数据库 exit

◆ 以下的操作需要最高级管理员才可以操作, 比如往集群里面注册一个服务等, 所以需要获得管理员凭证, 故: **获得 admin 凭证来获取只有管理员能执行命令的访问权限**

```
. admin-openrc
```

//此文件内容, 我已经在上部解释, 此文件及其内容可以先前建立好。

◆ 创建 nova 用户

```
openstack user create --domain default --password-prompt nova
```

解释:

在缺省域, 创建一个用户 nova。其实就是创建一个 nova 计算服务的管理者, 名字为 nova。但是注意, 此处我们仅仅是创建了一个用户, 并没给其赋予权限, 具体如下所述。(命令执行的时, 需要执行设定密码-请记住!)

◆ 添加 admin 角色到 nova 用户和 service 项目上

```
openstack role add --project service --user nova admin
```

解释:

将 admin 角色权限赋予给 nova 计算服务的管理者 nova。这就相当于用户 nova 具有了管理员的权限。--user nova 就是指明哪个用户, 最后面的 admin 表示角色权限(我们之前设定的)。

接下来标准动作: 一个对象, 三个服务点

一个服务实体, 相当于 new 一个对象。

三个认证服务端点的变种, 就是如上针对于 public, internal, admin

◆ 创建一个服务实体

```
openstack service create --name nova --description \  
"OpenStack Compute" compute
```

解释:

创建一个服务叫 nova，描述信息为“OpenStack 计算服务”。

我自己的理解是：类似于 java 生成对象的思想，用类 nova 创建一个对象 compute 作为 nova 的服务对象。

◆ 创建三类 endpoint 端点，分别针对于 **public, internal, admin** 权限域

```
openstack endpoint create --region RegionOne \
compute public http://controller:8774/v2.1/%(tenant_id)s
//controller 换成控制节点外网口的 IP 地址
```

```
openstack endpoint create --region RegionOne \
compute internal http://controller:8774/v2.1/%(tenant_id)s
//走管理网络
```

```
openstack endpoint create --region RegionOne \
compute admin http://controller:8774/v2.1/%(tenant_id)s
//走管理网络
```

解释：

创建三个 endpoint 端点，其中特别注意地址后面的 controller，其实是指明控制节点的地址。由于三个服务对象的作用域不一样，所以在集群内部，internal 和 admin 集群将通过走管理网络，进行域名解析（全部采用域名解析的形式）。而 public 是对外，相当于是对集群外部访问，所以我们将其更换为控制节点，外网卡的 IP 地址。

换句话说，输入 controller 机器能解析到其 IP 地址。所以集群在配置的时候，必须都要很清晰的完成，在 ping 主机的时候，后面写 IP 地址和直接写主机名字都可。能正确的完成域名解析。好处就是：IP 地址的变动，不会影响集群的配置，所以集群内部，我们统一使用走管理网络，进行域名解析，填写主机名字是没错的。

◆ 安装 nova 软件包

```
apt-get install nova-api nova-conductor nova-consoleauth \
nova-novncproxy nova-scheduler
```

◆ 配置文件/etc/nova/nova.conf 修改

- 在 [api\_database] 部分，配置数据库访问：

```
[api_database]
...
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova_api
```

- 在 [database] 部分，配置数据库访问：

```
[database]
...
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova
```

解释：

前面的 nova 表示 nova 的 mysql 数据库的管理员。后面的 NOVA\_DBPASS

表示这个 nova 数据库管理员的密码，而后面的 controller 表示连接的控制节点，**连接会走管理网络**，由于我们配置了域名解析，所以只需要填写主机名字就可。命令的完整意思是：采用用户名 nova 和密码 nova\_dbpass 去连接控制节点 controller 上的 nova 数据库。

这句命令也对应，本地对数据库的连接，我们之前也已经授权了本地的权限，如本节命令【对 nova 数据库赋予相应的权限】。

- 在“**[DEFAULT]**”部分，配置 RabbitMQ 消息队列，认证服务访问，配置 my\_ip 来使用控制节点的管理接口的 IP 地址，启用网络服务支持，禁用 EC2 API 和启用日志功能。

**[DEFAULT]**

...

rpc\_backend = rabbit //指明 RabbitMQ

//认证形式，采用 keystone

auth\_strategy = keystone

//配置 my\_ip 来使用控制节点的管理接口的 IP 地址

my\_ip = 10.0.0.11 //管理网络，网卡接口的 IP 地址(具体如网络图)

use\_neutron = True

firewall\_driver = nova.virt.firewall.NoopFirewallDriver

//默认情况下，计算机使用内部防火墙服务。由于网络包括防火墙服务，你必须使用 nova.virt.firewall.NoopFirewallDriver 驱动程序禁用计算机防火墙服务。

//禁用 EC2 API

enabled\_apis=osapi\_compute,metadata

verbose = True //启用详细日志

- 在“**[oslo\_messaging\_rabbit]**”部分，配置“RabbitMQ”消息队列访问：

**[oslo\_messaging\_rabbit]**

...

rabbit\_host = controller

//指明 RabbitMQ 是安装在控制节点上的，注意此处也是主机名表示，无需采用 IP 地址形式，因为可以域名解析了，**走管理网络**。

rabbit\_userid = openstack //指明 RabbitMQ 管理员的用户名

rabbit\_password = RABBIT\_PASS //指明 RabbitMQ 管理员用户名的密码

- 在“**[keystone\_authtoken]**”部分，配置认证服务访问：

**[keystone\_authtoken]**

...

auth\_uri = http://controller:5000

auth\_url = http://controller:35357

//指明 keystone 认证的地址，走管理网络。

memcached\_servers = controller:11211

auth\_plugin = password

```
//认证形式是采用密码形式
project_domain_name = default
user_domain_name = default
project_name = service
//注册服务实体的名字，注册的是一个服务
username = nova
password = NOVA_PASS
```

//此处为在 keystone 认证服务中，注册的 nova 用户名及其密码。  
在 **[keystone\_authtoken]** 中注释或者删除其他选项。

- 在**[vnc]**部分，配置 VNC 代理使用控制节点的管理 IP 地址

```
[vnc]
...
vncserver_listen = $my_ip
vncserver_proxyclient_address = $my_ip
//使用上面的配置 IP 地址
```

- 在**[glance]**部分，配置镜像服务的位置：

```
[glance]
...
api_servers = http://controller:9292
( glance 镜像服务器-存储网络-网卡的 IP 地址 )
```

//指明 glance 镜像存储的位置，由于我们采用了存储网络和管理网络分离，所以此处输入的是 **glance 镜像服务器-存储网络-网卡的 IP 地址**。

或自己写 glance,由于走管理网络就能找到，这里的目的是指明 glance 主机的位置，计算节点需要知道 glance 在哪儿。**我们建议写上面的的 IP 地址。**

- 在 **[oslo\_concurrency]** 部分，配置锁路径：

```
[oslo_concurrency]
...
lock_path = /var/lib/nova/tmp
```

**由于一个打包的 bug ，必须从 [DEFAULT] 区域去除 logdir 选项。**

- ◆ 同步 Compute 服务节点数据库：
 

```
su -s /bin/sh -c "nova-manage api_db sync" nova
su -s /bin/sh -c "nova-manage db sync" nova
```
- ◆ 重启计算服务=安装完成
 

```
service nova-api restart
service nova-consoleauth restart
service nova-scheduler restart
service nova-conductor restart
service nova-novncproxy restart
```
- ◆ 默认情况下，Ubuntu 上的安装包会自动创建一个 SQLite 数据库。 因为这里配置使用 SQL 数据库服务器，所以你可以删除 SQLite 服务库文件：

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

#### ◆ 验证方式

```
. admin-openrc
```

```
nova service-list
```

```
openstack endpoint list
```

```
root@controller:/# source admin-openrc.sh
root@controller:/# nova service-list
```

Id	Binary	Host	Zone	Status	State	Updated_at	Disabled Reason
1	nova-cert	controller	internal	enabled	up	2017-05-21T03:13:20.000000	-
2	nova-consoleauth	controller	internal	enabled	up	2017-05-21T03:13:15.000000	-
3	nova-scheduler	controller	internal	enabled	up	2017-05-21T03:13:14.000000	-
4	nova-conductor	controller	internal	enabled	up	2017-05-21T03:13:11.000000	-
5	nova-compute	computer1	nova	enabled	up	2017-05-21T03:13:18.000000	-

## 9.1.7 在控制节点上安装并配置网络服务 Neturon

OpenStack 网络比较复杂，涉及到 SDN 的知识，包括：Linux 桥，虚拟交换机（二层交换 Switching），虚拟路由器（三层路由 Routing），负载均衡 Load Balancing，防火墙 Firewalling，及五类网络类型，包括 *local*，*flat*，*vlan*，*vxlan*，*gre*。具体请查询相关书籍（*深入浅出 Neutron:OpenStack 网络技术及官方文件-附录注释*）。

### 本手册采用自服务网络类型！！

- ◆ 根据如上所述的，在控制节点上，每次创建一个服务就要创建一个数据库、服务凭证和 API 端点（endpoint），详细说就是：

- 1、创建一个数据库
- 2、创建一个数据库管理员
- 3、获得自由管理员才能操作的凭证（上面的脚本）
- 4、创建一个服务实体，相当于 new 一个对象。
- 5、创建三个认证服务端点的变种，就是如上针对于 public, internal, admin

在具体节点上（控制、计算，存储节点）：

- 1、安装服务
- 2、配置服务，包括修改连接数据库地址文件，修改 keystone 认证过程的配置文件等
- 3、重启服务

- ◆ 创建 neturon 数据库

```
mysql -u root -p          //连接数据库
CREATE DATABASE neutron;  //创建一个 neturon 数据库
```

- ◆ 对 neturon 数据库赋予相应的权限

```
GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost'
IDENTIFIED BY 'NEUTRON_DBPASS';
```

解释：

`neutron.*`：表示对 neturon 数据库中的所有表（\*）

`'neutron'@'localhost'`：前面的 neturon 表示是一个用户，相当于对谁赋权限。后面的 localhost 相当于指明用户的位置，localhost 表示在 localhost 的 neutron 用户对数据库访问。后面输入的密码是：`neutron` 用户的密码，相当于创建一个对 neturon 数据库的管理员，被赋予权限。

```
GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' IDENTIFIED BY
NEUTRON_DBPASS';
```

解释：

`'neutron'@'%'`：这里唯一的差别，在于 neutron 用户可以处于任何主机。相当于用 '%' 表示从任何地址连接。

刷新数据库 `flush privileges;` 退出数据库 `exit`

- ◆ 以下的操作需要最高级管理员才可以操作，比如往集群里面注册一个服务什么的，所以需要获得管理员凭证，故：**获得 admin 凭证来获取只有管理员能**

## 执行命令的访问权限

`. admin-openrc`

//此文件内容，我已经在上部解释，此文件及其内容可以先前建立好。

### ◆ 创建 neutron 用户

```
openstack user create --domain default --password-prompt neutron
```

解释：

在缺省域，创建一个用户 `neturon`。其实就是创建一个 `neturon` 网络服务的管理者，名字为 `neturon`。但是注意，此处我们仅仅是创建了一个用户，并没给其赋予权限，具体如下所述。**(命令执行的时，需要执行设定密码-请记住！)**

### ◆ 添加 `admin` 角色到 `neturon` 用户和 `service` 项目上

```
openstack role add --project service --user neutron admin
```

解释：

将 `admin` 角色权限赋予给 `neturon` 网络服务的管理者 `neturon`。这就相当于用户 `neturon` 具有了管理员的权限。`--user neutron` 就是指明哪个用户，最后面的 `admin` 表示角色权限（我们之前设定的）。

接下来标准动作：一个对象，三个服务点

一个服务实体，相当于 `new` 一个对象。

三个认证服务端点的变种，就是如上针对于 `public`, `internal`, `admin`

### ◆ 创建一个服务实体

```
openstack service create --name neutron --description "OpenStack Networking" network
```

解释：

创建一个服务叫 `neturon`，描述信息为“OpenStack 网络服务”。

我自己的理解是：类似于 java 生成对象的思想，用类 `neturon` 创建一个对象 `network` 作为 `neturon` 的服务对象。

### ◆ 创建三类 endpoint 端点，分别针对于 `public`, `internal`, `admin` 权限域

```
openstack endpoint create --region RegionOne network public \
http://controller:9696
```

//controller 换成控制节点外网口的 IP 地址

```
openstack endpoint create --region RegionOne network internal \
http://controller:9696
```

//走管理网络

```
openstack endpoint create --region RegionOne network admin \
http://controller:9696
```

//走管理网络

解释：

创建三个 `endpoint` 端点，其中特别注意地址后面的 `controller`，其实是指明控制节点的地址。由于三个服务对象的作用域不一样，所以在集群内部，`internal`

和 admin 集群将通过走管理网络，进行域名解析（全部采用域名解析的形式）。而 **public** 是对外，相当于是对集群外部访问，所以我们将其更换为控制节点，外网卡的 IP 地址。

换句话说，输入 controller 机器能解析到其 IP 地址。所以集群在配置的时候，必须都要很清晰的完成，在 ping 主机的时候，后面写 IP 地址和直接写主机名字都可。能正确的完成域名解析。好处就是：IP 地址的变动，不会影响集群的配置，所以集群内部，**我们统一使用走管理网络**，进行域名解析，填写主机名字是没错的。

◆ 安装 neturon 软件包(构建带 L3 的自服务网络插件)

```
apt-get install neutron-server neutron-plugin-ml2 \
neutron-linuxbridge-agent neutron-l3-agent neutron-dhcp-agent \
neutron-metadata-agent
```

◆ 配置文件 **/etc/neutron/neutron.conf** 的修改：

- 在 **[database]** 部分，配置数据库访问：

```
[database]
...
connection=
mysql+pymysql://neutron:NEUTRON_DBPASS@controller/neutron
```

解释：

前面的 **neutron** 表示 neutron 的 mysql 数据库的管理员。后面的 **NEUTRON\_DBPASS** 表示这个 neutron 数据库管理员的密码，而后面的 controller 表示连接的控制节点，**连接会走管理网络**，由于我们配置了域名解析，所以只需要填写主机名字就可。

命令的完整意思是：采用用户名 neutron 和密码 neutron\_dbpass 去连接控制节点 controller 上的 neutron 数据库。

这句命令也对应，本地对数据库的连接，我们之前也已经授权了本地的权限，如本节命令【对 neturon 数据库赋予相应的权限】。

- 在 **[DEFAULT]** 部分，启用 Layer 2 (ML2) 插件模块，路由服务和重叠的 IP 地址及**相关配置技术**：

```
[DEFAULT]
...
core_plugin = ml2           //核心是 2 层交换技术（引入二层插件）
service_plugins = router    //启用虚拟路由技术
allow_overlapping_ips = True //浮动 IP 技术（VM 能远程访问-当处于虚拟路由器下面的子网中）
```

```
rpc_backend = rabbit        //启用消息队列
auth_strategy = keystone    //启用认证服务
```

```
//配置网络以能够反映计算网络拓扑变化
notify_nova_on_port_status_changes = True
```

```
notify_nova_on_port_data_changes = True
```

```
//启动详细日志
```

```
verbose = True
```

- 由于启用 RabbitMQ，在[oslo\_messaging\_rabbit]部分，配置 RabbitMQ 消息队列访问

```
[oslo_messaging_rabbit]
```

```
...
```

```
rabbit_host = controller //指明 rabbit 所处的位置
```

```
rabbit_userid = openstack //指明 RabbitMQ 管理员的用户名
```

```
rabbit_password = RABBIT_PASS //指明 RabbitMQ 管理员的用户名密码
```

//指明 RabbitMQ 是安装在控制节点上的，注意此处也是主机名表示，无需采用 IP 地址形式，因为可以域名解析，走管理网络。

- 由于启动了 Keystone 认证，在[keystone\_authtoken]部分，配置认证服务访问

```
[keystone_authtoken]
```

```
...
```

```
//指明 keystone 认证的地址，走管理网络
```

```
auth_uri = http://controller:5000
```

```
auth_url = http://controller:35357
```

```
memcached_servers = controller:11211
```

```
//认证形式是采用密码形式
```

```
auth_plugin = password
```

```
//注册服务实体的名字，注册的是一个服务
```

```
project_domain_name = default
```

```
user_domain_name = default
```

```
project_name = service
```

```
username = neutron
```

```
password = NEUTRON_PASS
```

```
//此处为在 keystone 认证服务中，注册的 neutron 用户名及其密码。
```

**在 [keystone\_authtoken] 中注释或者删除其他选项。**

- 必须配置[nova]部分，配置网络以能够反映计算网络拓扑变化

```
[nova]
```

```
...
```

```
//指明 keystone 认证的地址，走管理网络
```

```
auth_url = http://controller:35357
```

```
//认证形式是采用密码形式(解释同如上)
```

```
auth_plugin = password
```

```
project_domain_name = default
```

```
user_domain_name = default
```

```
region_name = RegionOne
```

```
project_name = service
username = nova
password = NOVA_PASS
```

//使用在 keystone 中注册的 nova 服务用户名及其密码。

- ◆ 配置 Modular Layer 2 (ML2) 插件，ML2 插件使用 **Linux 桥接机制**为实例创建 layer-2 （桥接/交换）虚拟网络基础设施

编辑 **/etc/neutron/plugins/ml2/ml2\_conf.ini**

- 启用二层的标准网络类型，在[ml2]部分，启用 flat，VLAN 和 VXLAN 网络

```
[ml2]
```

```
...
```

//配置 ML2 插件后，删除可能导致数据库不一致的 type\_drivers 项的值

//指明二层所支持的网络类型

```
type_drivers = flat, vlan, vxlan
```

//启用 VXLAN 项目（私有）网络

```
tenant_network_types = flat
```

//启用 Linux 桥接和 layer-2 population mechanisms

```
mechanism_drivers = linuxbridge, l2population
```

//启用端口安全扩展驱动

```
extension_drivers = port_security
```

- 由于启用了 flat 扁平网络，所以配置其类型，在[ml2\_type\_flat]部分，配置公共 flat 提供网络

```
[ml2_type_flat]
```

```
...
```

```
flat_networks = provider
```

- 由于启用了 VXLAN 网络，在[ml2\_type\_vxlan]部分，配置 VXLAN 网络标识范围与私有网络不同：相当于 VLAN 号的范围

```
[ml2_type_vxlan]
```

```
...
```

```
vni_ranges = 1:1000
```

- 在 openstack 网络中，支持安全组功能，所以在[securitygroup]部分，启用 ipset 增加安全组的方便性

```
[securitygroup]
```

```
...
```

```
enable_ipset = True
```

- ◆ 配置 Linux 桥接代理，Linux 桥接代理为实例创建包括私有网络的 VXLAN 隧道和处理安全组的 layer-2（桥接/交换）虚拟网络设施。

编辑 **/etc/neutron/plugins/ml2/linuxbridge\_agent.ini** 文件并完成下面的操作：

- 在[linux\_bridge]部分，映射公共虚拟网络到公共物理网络接口指定：

[linux\_bridge]

physical\_interface\_mappings = provider:PUBLIC\_INTERFACE\_NAME

//因为上文配置 flat\_networks = public，所以 FLAT 网络需要占据一块物理网卡，而这物理网卡是可以上外网的网卡接口。需要将 PUBLIC\_INTERFACE\_NAME 替换为，控制节点服务器上某个物理网卡端口（连接外网的），因为处于 FLAT 下的虚拟机都会都过这块网卡上外网，并且被分配到一个外网的 IP。

//而所谓浮动 IP 技术，是当集群中启用了虚拟机路由器之后，在其下面的虚拟机都已经处于不同的子网了，虽然虚拟机可以上网，但是不能被远程连接，因为被虚拟路由器隔离了，所以我们需要在虚拟路由器上启用浮动 IP 技术，将内网地址与外网 IP 进行一对一的映射，这样远程用户就可以连接虚拟机了（**特别注意：浮动 IP 技术是用在虚拟路由器上的**）。

因为浮动 IP 技术将虚拟机的网络通过此物理服务器端口进行转发出去，将数据包转到公网，浮动 IP 技术相当于是 1-1 的 NAT 映射（具体原理，请具体理解浮动 IP 技术）。

- 在[vxlan]部分，启用 VXLAN 覆盖网络，配置处理覆盖网络和启用 layer-2 的物理网络接口的 IP 地址，VXLAN 相当于是 VLAN 中的 VLAN 技术。

[vxlan]

enable\_vxlan = True

local\_ip = OVERLAY\_INTERFACE\_IP\_ADDRESS

l2\_population = True

//此 IP 需要的是虚拟机网络挂接在服务器物理网卡的端口的 IP 地址（，一般请设置为控制节点管理网络 IP，请查看本文件网络图具体说明）。

- 在[securitygroup]部分，启用安全组并配置 Linux 桥接 iptables 防火墙驱动：

[securitygroup]

...

enable\_security\_group = True

firewall\_driver= neutron.agent.linux.iptables\_firewall.IptablesFirewallDriver

#### ◆ 配置 layer-3 代理

编辑 /etc/neutron/l3\_agent.ini 文件并完成下面操作：

- 在 [DEFAULT]部分，配置 Linux 桥接网络驱动和外部网络桥接：

[DEFAULT]

...

interface\_driver = neutron.agent.linux.interface.BridgeInterfaceDriver

external\_network\_bridge =

//故意缺少值，这样就可以在一个代理上启用多个外部网络

verbose = True

//启用详细日志

#### ◆ 配置 DHCP 代理

编辑`/etc/neutron/dhcp_agent.ini` 文件并完成下面的操作：

- 在`[DEFAULT]`部分，配置 Linux 桥接网卡驱动，Dnsmasq DHCP 驱动并启用隔离元数据，这样在公共网络上的实例就可以通过网络访问元数据：

```
[DEFAULT]
...
interface_driver = neutron.agent.linux.interface.BridgeInterfaceDriver
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
//启用详细日志
verbose = True
enable_isolated_metadata = True
```

叠加网络比如 VXLAN 包含额外的数据包头，这些数据包头增加了开销，减少了有效内容或用户数据的可用空间。在不了解虚拟网络架构的情况下，实例尝试用缺省的以太网 最大传输单元 (MTU) 1500 字节发送数据包。以太网协议(IP)网络使用发现 MTU 路径 (PMTUD) 机制检测端到端 MTU，相应的调整包大小。但是，有些操作系统和网络阻塞、缺省 PMTUD 支持导致性能下降或者连接失败。

理想情况，通过在包含您租户虚拟网络的物理网络上启用 jumbo frames，您可以避免这些问题。Jumbo 帧支持 MTU 接近 9000 字节，取消虚拟网上 VXLAN 过量的影响。然而，许多网络设备缺少像 jumbo 帧的支持，OpenStack 管理员经常不能控制网络基础设施。考虑到以后的并发症，您可以通过减少实例 MTU 来取消 VXLAN 过量的方法避免 MTU 问题。选择合适的 MTU 值经常需要测试，但是 1450 字节在大数环境下可以正常工作。您可以配置 DHCP 服务器也调整 MTU 值，DHCP 服务器给实例分配 IP 地址。

#### ◆ 配置元数据代理

编辑`/etc/neutron/metadata_agent.ini` 文件并完成下面的操作：

- 在`[DEFAULT]`部分，配置访问参数：

```
[DEFAULT]
...
nova_metadata_ip = controller //配置元数据主机
//配置元数据代理共享密码
metadata_proxy_shared_secret = METADATA_SECRET (只要统一，都可)
verbose = True //启用详细日志
```

#### ◆ 配置计算使用网络

编辑`/etc/nova/nova.conf` 文件并完成下面操作

在`[neutron]`部分，配置访问参数，启用元数据代理和配置 secret：

```
[neutron]
...
//控制节点的认证地址及端口
url = http://controller:9696
```

```

auth_url = http://controller:35357
//认证形式为密码
auth_plugin = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service

username = neutron
password = NEUTRON_PASS
//此处为在 keystone 认证服务中，注册的 neutron 用户名及其密码。
service_metadata_proxy = True
metadata_proxy_shared_secret = METADATA_SECRET （只要统一，都可）
// METADATA_SECRET 红色标记只要是相同的密钥即可。

```

◆ 同步数据库

```

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

```

◆ 重启网络服务=安装完成

```

service nova-api restart

```

```

service neutron-server restart
service neutron-linuxbridge-agent restart
service neutron-dhcp-agent restart
service neutron-metadata-agent restart

```

◆ 同样重启 layer-3 服务

```

service neutron-l3-agent restart

```

◆ 默认情况下，Ubuntu 上的安装包会自动创建一个 SQLite 数据库。  
因为这里配置使用 SQL 数据库服务器，所以你可以 SQLite 服务库文件：

```

rm -f /var/lib/neutron/neutron.sqlite

```

◆ 验证安装成功：

```

. admin-openrc
neutron ext-list
neutron agent-list

```

```

root@controller:/# source admin-openrc.sh
root@controller:/# neutron ext-list

```

alias	name
dns-integration	DNS Integration
ext-gw-mode	Neutron L3 Configurable external gateway mode
binding	Port Binding
agent	agent
subnet_allocation	Subnet Allocation
l3_agent_scheduler	L3 Agent Scheduler
external-net	Neutron external network
flavors	Neutron Service Flavors
net-mtu	Network MTU
quotas	Quota management support
l3-ha	HA Router extension
provider	Provider Network
multi-provider	Multi Provider Network
extraroute	Neutron Extra Route
router	Neutron L3 Router
extra_dhcp_opt	Neutron Extra DHCP opts
security-group	security-group
dhcp_agent_scheduler	DHCP Agent Scheduler
rbac-policies	RBAC Policies
port-security	Port Security
allowed-address-pairs	Allowed Address Pairs
dvr	Distributed Virtual Router

```

root@controller:~# source admin-openrc.sh
root@controller:~# neutron agent-list

```

id	agent_type	host	alive	admin_state_up	binary
06f8b20d-e9b4-44e4-a206-47152577cb1e	Linux bridge agent	compute1	:-)	True	neutron-linuxb
0ce5085b-4472-4d1e-9378-2170355ba64f	Linux bridge agent	controller	:-)	True	neutron-linuxb
0ea0dda4-edca-4f7e-97d6-bbe931c6de6f	DHCP agent	controller	:-)	True	neutron-dhcp-a
8c051c15-0141-44fc-9e29-1d2db77e7d82	Metadata agent	controller	:-)	True	neutron-metada
eab305be-bfbf-4298-a7ee-bfc9ca952598	Linux bridge agent	compute2	:-)	True	neutron-linuxb
fdaa1399-e51d-4063-afad-a53623b35eaa	L3 agent	controller	:-)	True	neutron-l3-age

## 9.1.8 在控制节点创建仪表盘 Dashboard

Dashboard 相当于是一个 OpenStack 集群管理的可视化界面，在里面很多命令的操作都将转化为图形化操作，并且能在其中反映集群的性能情况等。Openstack 的 Dashboard，是基于 OpenStack 各个组件开发的 web 管理后台，项目名字是 Horizon。目前 Dashboard 并没有实现全部的 API 功能，很多功能可能是 API 提供，但是 Dashboard 没有去实现。同时 Dashboard 还欠缺不少功能，等待大家一起完善，具体内容可查看附录的相关书籍或者视频教程。

### ◆ 安装软件包

```
apt-get install openstack-dashboard
```

### ◆ 配置文件修改

编辑文件 `/etc/openstack-dashboard/local_settings.py` 并完成如下动作：

- 在 **controller** 节点上配置仪表盘以使用 OpenStack 服务

```
OPENSTACK_HOST = "controller"
//指明主机名字（horizon 的位置）
```

- 允许所有主机访问仪表板：

```
ALLOWED_HOSTS = ['*',]
```

- 配置 **memcached** 会话存储服务(将其他的会话存储服务配置注释)：

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

- 选择性地配置时区（修改之前配置项）：

```
TIME_ZONE = "TIME_ZONE" //改为 Asia/Shanghai
```

- 为通过仪表盘 **创建的用户** 配置默认的用户角色（修改之前配置项）：

```
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "default"
OPENSTACK_KEYSTONE_URL =
"http://%s:5000/v3" % OPENSTACK_HOST
```

//特别注意，此处配置的用户角色和 KeyStone 是一一对应的，可如上检查 keystone 里面 **user 角色** 是否已经配置并且启用。

- 启用 multi-domain model:

```
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
```

//True 请采用大写，否则出错

- 配置服务 API 版本，通过 Keystone V3 API 登录 dashboard（直接添加）：

```
OPENSTACK_API_VERSIONS = {
```

```
    "identity": 3,
```

```
    "image": 2,
```

```
    "volume": 2,
```

```
}
```

- ◆ 重新启动服务

```
service apache2 reload
```

- ◆ 并重新启动 Apache 服务器

```
service apache2 restart
```

- ◆ 访问网页地址为：**http://控制节点的服务器对外网卡的 IP/horizon**

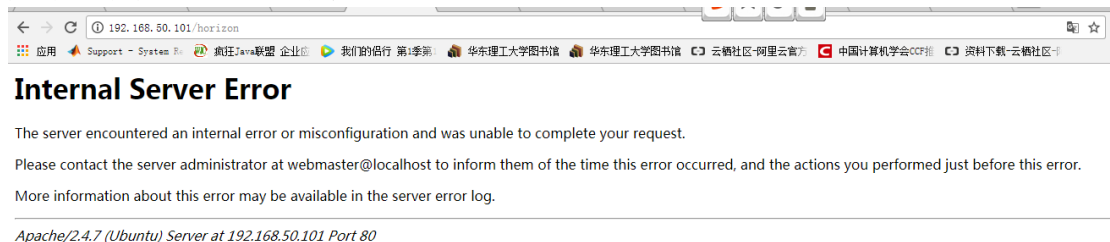
用户名：admin

密码：ADMIN

特别注意：admin 这个用户，就是 KeyStone 里面的管理员用户，也验证了 OpenStack 中所有的认证都需要 KeyStone 来完成。

我们本集群设置的 keyStone 的管理员为 admin 密码是大写的 ADMIN，所有从网页版登录的用户也就是这个。

如果出现如下错误，请做相应的排查



```
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
```

//True 请采用大写，否则出错，如上所述。

## 9.2 计算节点

计算节点是整个集群的计算中心，集群中运行的虚拟机都会在计算节点上，而具体的虚拟机-虚拟磁盘或者其存储都会在共享存储，计算节点仅提供 CPU 和内存等计算资源，其都需要在 Nova 调度器的作用下工作，而调度器是安装在控制节点的，所以计算机节点，某些数据的传输，及命令的操作，都需要与控制节点进行通讯。

◆ 其中核心包括：

- 统一时钟：NTP
- 指明控制节点和存储节点的位置等配置
- 身份认证等需要远程连接的信息配置
- 配置网络服务等相关配置
- 核心部署-计算 Nova 组件（计算部分）
- 核心部署-网络 Neturon 组件（操作部分）

◆ 主要安装有：

nova-compute, nova-network, networking, NTP, MySql Client 等

◆ 物理主机部分：

- 配置管理网络-网卡信息-特别重要（走管理网络，主机名解析）
- 配置外部网络-网卡信息-特别重要，承担虚拟机的外网服务和对其远程连接（也相当于是承担 VM 网络）。
- 配置存储网络-网卡信息-特别重要（走存储网络，数据、镜像等传输）

◆ 补充说明：

## 9.2.1 各主机基本网络配置

- ◆ 网络配置 (细节请查看网络地址分配)

配置 **/etc/network/interfaces**

IP 地址采用静态配置格式如下, 启用了多少个网卡, 就配置几个, 格式如下:

```
auto lo
```

```
iface lo inet loopback
```

```
auto eth1
```

```
iface eth1 inet static
```

```
address      IP 地址
```

```
netmask      掩码
```

```
gateway      网关地址
```

```
network      网络号    //非必要
```

```
broadcast    广播地址  //非必要
```

.....

主网络配网关, 其他不要配, 否则网络不能连接

- ◆ 修改主机名字 hostname

配置 **/etc/hostname**

```
hostname 主机名
```

//指明某台服务器的主机名字

- ◆ 修改域名与主机对应关系

配置 **/etc/hosts**

```
IP 地址 主机名
```

//将集群里面的所有机器 (一行一个主机), 按照这样的对应关系配置好, 非常重要需要主机名解析, 相当于 ping 主机名也是可以通的!!! (走管理网络-所以填写都是各自物理服务器管理网络网卡接口的 IP 地址)

- ◆ 设置 DNS

配置 **/etc/resolv.conf**

```
nameserver IP 地址
```

//后面的 IP 地址是域名服务器 (是正常的外网 DNS 地址, 比如学校是 202.120.111.3)

- ◆ 防止重启机器后 DNS 重置

配置 **/etc/resolvconf/resolv.conf.d/tail**

```
nameserver IP 地址
```

//后面的 IP 地址是域名服务器-同上

- ◆ 重启网络

**/etc/init.d/networking restart**

**检验标准：**

- 1、网络各个网卡已经工作，并且在 `ifconfig` 中看到配置已经启用。
- 2、网络能访问外网，例如可以 `ping` 通百度。
- 3、能进行域名解析，即 `ping` 集群中各个主机名也是可以通的（走管理网络）。
- 4、若配置都是正确的，网卡没有启动或者配置没有生效，请 `reboot` 服务器。
- 5、特别注意：网卡的名字，有些服务器命名为 `ethx`，有些服务器是 `emx`，其中 `x` 用阿拉伯数字表示 `0-n`。

## 9.2.2 在计算节点部署 Nova 组件及配置服务

### ◆ 安装 NTP 服务

```
sudo apt-get install chrony
```

- 修改配置文件 `/etc/chrony/chrony.conf`，使得其指向控制节点，并将控制节点作为时钟基准源

```
server controller iburst
```

#注释掉所有以 `server` 开头的配置项，并在末尾添加此。

解释：

此处的计算节点会走管理网络，并找到控制节点，并且以控制节点作为主时钟基准源，那么此处又要明确，`controller` 和 IP 是一一对应的，即可以 ping 主机名字，要也可以 ping ip 地址，此处走的是管理网络（从另外一方面讲，只有管理网络才是真正全部连接，控制节点，计算节点和存储节点等所有节点）。

- 重启 NTP 服务

```
service chrony restart
```

- 验证 NTP 服务的安装

```
chronyc sources
```

### ◆ 安装 OpenStack-Liberty Package 包

```
apt-get install software-properties-common
```

```
add-apt-repository cloud-archive:mitaka
```

### ◆ 安装 OpenStack-Liberty 客户端

```
apt-get install python-openstackclient
```

### ◆ 升级并重启系统

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

```
reboot
```

根据之前的分析，任何在 OpenStack 执行服务安装，组件部署的时候，都需要向 KeyStone 获得一个管理员的 Token，其实就是相当于获得一个管理员的令牌权限才能操作。所以，我们需要之前的定义 **声明一下接下来的操作是管理员在做**，执行 `source admin-openrc.sh`。具体的 `admin-openrc.sh` 内容如下，原理解释请查看上文控制节点安装 keystone-创建 OpenStack 客户端环境脚本，7.1.3 节。

```
export OS_PROJECT_DOMAIN_NAME=default //指明项目处于什么域
export OS_USER_DOMAIN_NAME=default //指明用户处于什么域
export OS_PROJECT_NAME=admin //指明项目的名字
export OS_USERNAME=admin //指明用户名名字
export OS_PASSWORD=ADMIN_PASS //指明用户名的密码
export OS_AUTH_URL=http://controller:35357/v3 //认证的 URL，也说明了之前为啥会装个 apache 服务器，其实 endpoint 暴露出来之后，其认证的形式，
```

都是采用 URL 进行的（controller 说明走管理网络）。

```
export OS_IDENTITY_API_VERSION=3 //API 版本号
```

```
export OS_IMAGE_API_VERSION=2 //后面镜像时候用
```

注意：将 **ADMIN\_PASS** 替换为你在认证服务中为 **admin** 用户选择的密码。

```
. admin-openrc //别忘了执行(先建立上面的脚本，再执行脚本)
```

#### ◆ 安装软件包

```
apt-get install nova-compute
```

#### ◆ 配置文件修改，编辑 **/etc/nova/nova.conf**

- 在**[DEFAULT]**，配置 RabbitMQ 消息队列，认证服务，启用网络服务支持，启用详细日志等

**[DEFAULT]**

```
rpc_backend = rabbit //配置采用 RabbitMQ 作为消息队列
```

```
auth_strategy = keystone //指明，认证采用 KeyStone 服务
```

```
my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
```

//替换为计算节点上的管理网络接口的 IP 地址

//启用网络服务支持

```
use_neutron = True
```

```
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

```
verbose = True //打开日志调试信息
```

- 在**[oslo\_messaging\_rabbit]**部分配置 RabbitMQ 消息队列详细信息

**[oslo\_messaging\_rabbit]**

...

```
rabbit_host = controller //走管理网络，能找到这台主机（能域名解析）
```

```
rabbit_userid = openstack//这个其实是 RabbitMQ 的管理员用户名
```

```
rabbit_password = RABBIT_PASS //所对应的管理员密码
```

- 在**[keystone\_authtoken]**部分配置认证服务详细信息

**[keystone\_authtoken]**

...

```
auth_uri = http://controller:5000
```

```
auth_url = http://controller:35357
```

```
memcached_servers = controller:11211
```

//走管理网络，可以找到控制节点，指明认证的 URL 地址

```
auth_plugin = password
```

//认证采用的形式为密码

```
project_domain_name = default
```

```
user_domain_name = default
```

```
project_name = service
```

//注册服务实体的名字，注册的是一个服务

username = nova

password = NOVA\_PASS

//此处为在 keystone 认证服务中，注册的 nova 用户名及其密码。

在 **[keystone\_authtoken]** 中注释或者删除其他选项。

- 在**[vnc]**启用并配置远程控制台访问

**[vnc]**

...

enabled = True

vncserver\_listen = 0.0.0.0

vncserver\_proxyclient\_address = **\$my\_ip**

//替换为计算节点上的管理网络接口的 IP 地址

解释:

服务器组件监听所有的 IP 地址，而代理组件仅仅监听计算节点管理网络接口的 IP 地址。基本的 URL 指示可以使用 web 浏览器访问位于该计算节点上实例的远程控制台的位置。

novncproxy\_base\_url = http://**controller**:6080/vnc\_auto.html

若在浏览器中不能解析 controller，请将其更换为控制节点的 IP 地址（外网口，我们建议这么做），但是对于集群来说，集群是识别这个 controller 主机的，能进行正常的域名解析（通过管理网络）。

- 在**[glance]**部分，配置镜像服务的位置

**[glance]**

...

~~host = controller~~ //更换为镜像服务器的位置

//一般填写存储服务器其存储网络的 IP 地址（对方的）

api\_servers = http://controller:9292

- 在 **[oslo\_concurrency]** 部分，配置锁路径:

**[oslo\_concurrency]**

...

lock\_path = /var/lib/nova/tmp

由于一个打包的 bug，必须从 **[DEFAULT]** 区域去除 logdir 选项。

- ◆ 检测计算节点服务器是否支持虚拟机硬件加速

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

解释:

如果这个命令返回 1 或者更大的值，说明您的计算节点支持硬件加速，一般不需要进行额外的配置。如果这个命令返回 0，你的计算节点不支持硬件加速，你必须配置 libvirt 使用 QEMU 而不是使用 KVM。

在 **/etc/nova/nova-compute.conf** 文件中像下面这样编辑 **[libvirt]** 部分

**[libvirt]**

...

```
virt_type = qemu
```

- ◆ 重启网络服务=安装完成

```
service nova-compute restart
```

- ◆ 默认情况下，Ubuntu 上的安装包会自动创建一个 SQLite 数据库。  
因为这里配置使用 SQL 数据库服务器，所以你可以 SQLite 服务库文件：

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

- ◆ 验证 nova 是否安装成功

```
. admin-openrc
```

```
nova service-list
```

```
nova endpoints
```

```
nova image-list
```

//上述三个命令均有输出

```
root@computer1:/# nova service-list
+-----+-----+-----+-----+-----+-----+-----+-----+
| Id | Binary | Host | Zone | Status | State | Updated_at | Disabled Reason |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | nova-cert | controller | internal | enabled | up | 2017-05-20T15:22:50.000000 | - |
| 2 | nova-consoleauth | controller | internal | enabled | up | 2017-05-20T15:22:51.000000 | - |
| 3 | nova-scheduler | controller | internal | enabled | up | 2017-05-20T15:22:46.000000 | - |
| 4 | nova-conductor | controller | internal | enabled | up | 2017-05-20T15:22:46.000000 | - |
| 5 | nova-compute | computer1 | nova | enabled | up | 2017-05-20T15:22:55.000000 | - |
+-----+-----+-----+-----+-----+-----+-----+-----+
root@computer1:/#
```

### 9.2.3 在计算节点部署 Neturon 组件及配置服务

#### ◆ 安装软件

```
apt-get install neutron-linuxbridge-agent
```

#### ◆ 编辑 `/etc/neutron/neutron.conf` 文件并完成如下配置：

- 在 `[database]` 部分，注释所有 `connection` 项，因为计算节点不直接访问数据库。

- 在 `[DEFAULT]` 配置 RabbitMQ 消息队列，认证服务，启用详细日志

```
[DEFAULT]
```

```
...
```

```
rpc_backend = rabbit           //配置采用 RabbitMQ 作为消息队列
```

```
auth_strategy = keystone       //指明，认证采用 KeyStone 服务
```

```
verbose = true                 //打开日志调试信息
```

- 在 `[oslo_messaging_rabbit]` 配置 RabbitMQ 消息队列：

```
[oslo_messaging_rabbit]
```

```
...
```

```
rabbit_host = controller       //走管理网络，能找到这台主机（能域名解析）
```

```
rabbit_userid = openstack      //这个其实是 RabbitMQ 的管理员用户名
```

```
rabbit_password = RABBIT_PASS //所对应的管理员密码
```

- 在 `[keystone_authtoken]` 部分，配置认证服务  
（注释掉已有的相关配置行，添加如下内容）：

```
[keystone_authtoken]
```

```
...
```

```
//走管理网络，指明认证服务的地址及端口号（控制节点）
```

```
auth_uri = http://controller:5000
```

```
auth_url = http://controller:35357
```

```
memcached_servers = controller:11211
```

```
auth_plugin = password
```

```
//认证形式为密码
```

```
project_domain_name = default
```

```
user_domain_name = default
```

```
project_name = service
```

```
//注册服务实体的名字，注册的是一个服务
```

```
username = neutron
```

```
password = NEUTRON_PASS
```

```
//此处为在 keystone 认证服务中，注册的 neutron 用户名及其密码。
```

#### ◆ 配置自服务网络结构，Linux 桥代理形式

Linux 桥接代理为实例创建包括私有网络的 VXLAN 隧道和处理安全组的 layer-2（桥接/交换）虚拟网络设施。

- 编辑 `/etc/neutron/plugins/ml2/linuxbridge_agent.ini` 文件

在[linux\_bridge]部分，映射公共虚拟网络到公共物理网络接口

[linux\_bridge]

physical\_interface\_mappings = provider:PUBLIC\_INTERFACE\_NAME

//蓝色部分替换为计算节点物理网卡名字（连接外网的），比如 eth1 或者 em1 等

解释：

- 在[vxlan]部分，启用 VXLAN 覆盖网络，配置处理覆盖网络和启用 layer-2 的物理网络接口的 IP 地址

[vxlan]

enable\_vxlan = True

local\_ip = OVERLAY\_INTERFACE\_IP\_ADDRESS

l2\_population = True

//替换为 IP 地址为计算节点管理网络 IP 地址

解释：

- 在[securitygroup]部分，启用安全组并配置 Linux 桥接 iptables 防火墙驱动

[securitygroup]

...

enable\_security\_group = True

firewall\_driver= neutron.agent.linux.iptables\_firewall.IptablesFirewallDriver

- 配置计算使用 neutron 网络，修改文件/etc/nova/nova.conf  
在[neutron]部分，配置访问参数：

[neutron]

...

//走管理网络，指明服务的地址及端口号（控制节点）

url = http://controller:9696

auth\_url = http://controller:35357

//认证形式采用密码

auth\_plugin = password

project\_domain\_name = default

user\_domain\_name = default

region\_name = RegionOne

project\_name = service

username = neutron

password = NEUTRON\_PASS

//此处为在 keystone 认证服务中，注册的 neutron 用户名及其密码。

- 配置计算使用 cinder 存储，修改文件/etc/nova/nova.conf

[cinder]

os\_region\_name = RegionOne

- ◆ 完成安装，重启计算服务和 Linux 桥代理

```
service nova-compute restart
```

```
service neutron-linuxbridge-agent restart
```

- ◆ 验证 Neutron 服务

```
. admin-openrc
```

```
neutron agent-list
```

```
neutron ext-list
```

```
root@computer1:/# source admin-openrc.sh
root@computer1:/# neutron agent-list
```

id	agent_type	host	alive	admin_state_up	binary
6b465de9-94ae-44ba-bc7e-44e28ed8c50b	DHCP agent	controller	:-)	True	neutron-dhcp-agent
a57f804a-ca06-4672-b00a-acf256641030	L3 agent	controller	:-)	True	neutron-l3-agent
adab194f-b392-43f5-aeba-3d37df397e69	Linux bridge agent	controller	:-)	True	neutron-linuxbridge-agent
b5550ab5-ad28-462d-843d-5d7385f5901c	Metadata agent	controller	:-)	True	neutron-metadata-agent
d4d989bb-3a9a-4fb2-8bb8-f6f7a6c0eed0	Linux bridge agent	computer1	:-)	True	neutron-linuxbridge-agent

## 9.3 镜像存储节点

Glance 自己并不存储 image，真正的 image 是存放在 backend 中：

A directory on a local file system

Gridfs

Ceph rbd（实验室 V2.0 的时候统一存储也会构建并使用）

Amazon s3

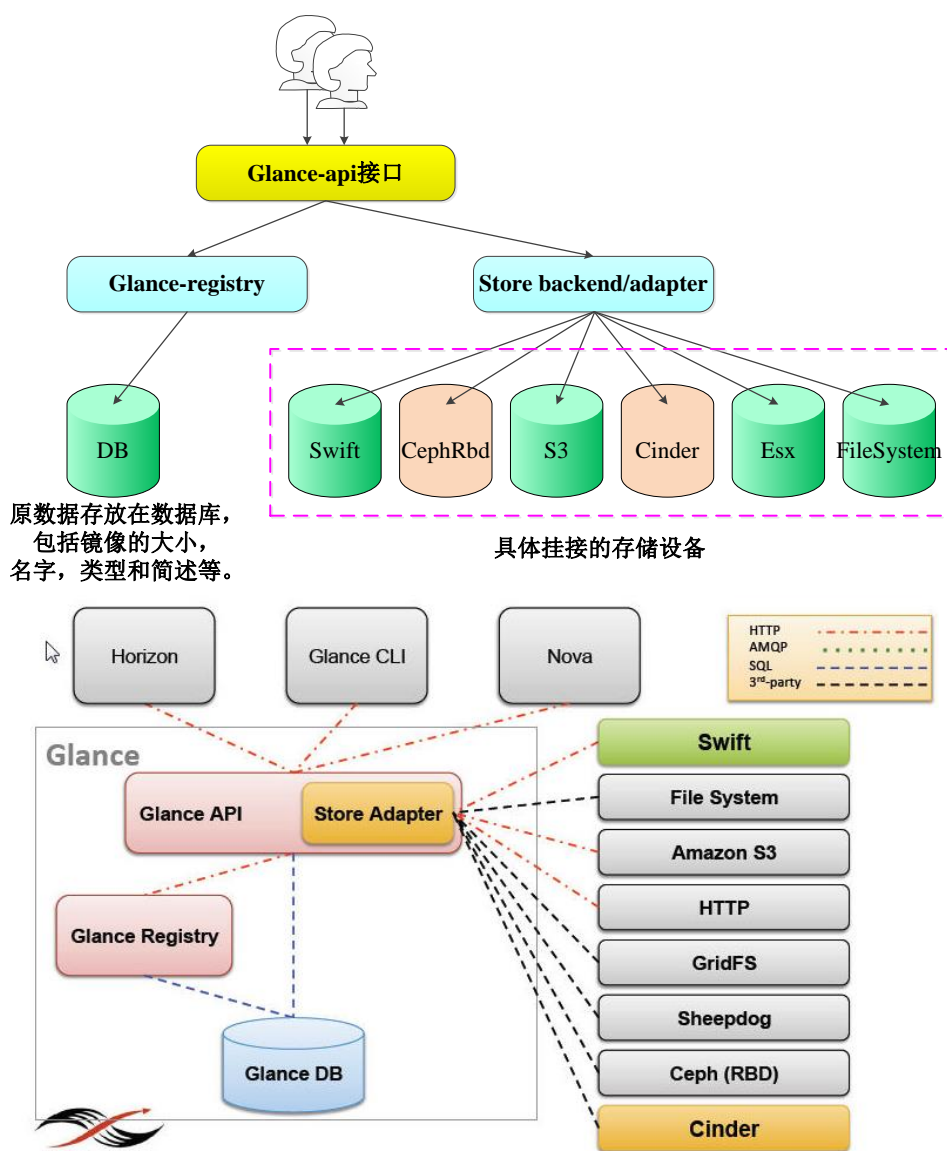
Sheepdog

Openstack block storage (Cinder)

Openstack object storage (Swift)

Vmware esx

具体使用哪种 backend，是在 `/etc/glance/glance-api.conf` 中配置的。



安装镜像存储的服务组件：

Glance 提供 restful API 可以查询虚拟机镜像的 metadata，Glance 目前提供的

参考实现中 Registry Server 仅是使用 Sql 数据库存储 metadata，由此获得镜像的存储位置及基本信息。

通过 Glance 虚拟机镜像可以被存储到多种存储上，比如简单的文件存储或者对象存储（如 OpenStack 中 Swift 项目），目前主流支持 S3, Swift, SheepDog, Ceph(分布式存储系统，后期我们实验室也会构建)，简单的文件存储及只读的 HTTPS 存储。其实相当于一个接口，后端可以挂接各种具体的存储服务设备（结构图所示），前端通过 API Server 向多个 Client 提供服务。

### 9.3.1 各主机基本网络配置

- ◆ 网络配置 (细节请查看网络地址分配)

配置 **/etc/network/interfaces**

IP 地址采用静态配置格式如下, 启用了多少个网卡, 就配置几个, 格式如下:

```
auto lo
iface lo inet loopback
```

```
auto eth1
iface eth1 inet static
address    IP 地址
netmask    掩码
gateway    网关地址
network    网络号 //非必要
broadcast  广播地址 //非必要
```

.....

主网络配网关, 其他不要配, 否则网络不能连接

- ◆ 修改主机名字 hostname

配置 **/etc/hostname**

```
hostname 主机名
```

//指明某台服务器的主机名字

- ◆ 修改域名与主机对应关系

配置 **/etc/hosts**

```
IP 地址 主机名
```

//将集群里面的所有机器 (一行一个主机), 按照这样的对应关系配置好, 非常重要需要主机名解析, 相当于 ping 主机名也是可以通的!!! (走管理网络-所以填写都是各自物理服务器管理网络网卡接口的 IP 地址)

- ◆ 设置 DNS

配置 **/etc/resolv.conf**

```
nameserver IP 地址
```

//后面的 IP 地址是域名服务器 (是正常的外网 DNS 地址, 比如学校是 202.120.111.3)

- ◆ 防止重启机器后 DNS 重置

配置 **/etc/resolvconf/resolv.conf.d/tail**

```
nameserver IP 地址
```

//后面的 IP 地址是域名服务器-同上

- ◆ 重启网络

**/etc/init.d/networking restart**

**检验标准：**

- 1、网络各个网卡已经工作，并且在 `ifconfig` 中看到配置已经启用。
- 2、网络能访问外网，例如可以 `ping` 通百度。
- 3、能进行域名解析，即 `ping` 集群中各个主机名也是可以通的（走管理网络）。
- 4、若配置都是正确的，网卡没有启动或者配置没有生效，请 `reboot` 服务器。
- 5、特别注意：网卡的名字，有些服务器命名为 `ethx`，有些服务器是 `emx`，其中 `x` 用阿拉伯数字表示 `0-n`。

### 9.3.2 在存储节点部署 Glance 组件及配置服务

#### ◆ 安装 NTP 服务

```
sudo apt-get install chrony
```

- 修改配置文件 **/etc/chrony/chrony.conf**，使得其指向控制节点，并将控制节点作为时钟基准源

```
server controller iburst
```

#注释掉所有以 server 开头的配置项，并在末尾添加此。

**解释：**

此处的计算节点会**走管理网络**，并找到控制节点，并且以控制节点作为主时钟基准源，那么此处又要明确，controller 和 IP 是一一对应的，即可以 ping 主机名字，要也可以 ping ip 地址，此处走的是管理网络（从另外一方面讲，只有管理网络才是真正全部连接，控制节点，计算节点和存储节点等所有节点）。

- 重启 NTP 服务

```
service chrony restart
```

- 验证 NTP 服务的安装

```
chronyc sources
```

#### ◆ 安装 OpenStack-Liberty Package 包

```
apt-get install software-properties-common
```

```
add-apt-repository cloud-archive:mitaka
```

#### ◆ 安装 OpenStack-Liberty 客户端

```
apt-get install python-openstackclient
```

#### ◆ 升级并重启系统

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

```
reboot
```

根据之前的分析，任何在 OpenStack 执行服务安装，组件部署的时候，都需要向 KeyStone 获得一个管理员的 Token，其实就是相当于获得一个管理员的令牌权限才能操作。所以，我们需要之前的定义**声明一下接下来的操作是管理员在做**，执行 source admin-openrc.sh。具体的 admin-openrc.sh 内容如下，原理解释请查看上文控制节点安装 keystone-创建 OpenStack 客户端环境脚本，7.1.3 节。

```
export OS_PROJECT_DOMAIN_NAME=default //指明项目处于什么域
export OS_USER_DOMAIN_NAME=default //指明用户处于什么域
export OS_PROJECT_NAME=admin //指明项目的名字
export OS_USERNAME=admin //指明用户名名字
export OS_PASSWORD=ADMIN_PASS //指明用户名的密码
export OS_AUTH_URL=http://controller:35357/v3 //认证的 URL，也说明了之前为啥会装个 apache 服务器，其实 endpoint 暴露出来之后，其认证的形式，
```

都是采用 URL 进行的（走管理网络-域名解析）。

```
export OS_IDENTITY_API_VERSION=3 //API 版本号
```

```
export OS_IMAGE_API_VERSION=2 //后面镜像时候用
```

注意：将 **ADMIN\_PASS** 替换为你在认证服务中为 **admin** 用户选择的密码。

```
. admin-openrc //别忘了执行(先建立上面的脚本，再执行脚本)
```

◆ 安装工具支持包软件

```
apt-get install glance
```

◆ 编辑文件 **/etc/glance/glance-api.conf** 并完成如下配置：

- 在 **[DEFAULT]** 部分，配置 **noop** 禁用通知，因为他们只适合与可选的 Telemetry 服务，并且启用详细日志：

```
[DEFAULT]
```

```
...
```

```
verbose = True
```

- 在 **[database]** 部分，配置数据库访问：

```
[database]
```

```
...
```

```
connection=mysql+pymysql://glance:GLANCE_DBPASS@controller/glance
```

解释：

前面的 **glance** 表示 glance 的 mysql 数据库的管理员。后面的 **GLANCE\_DBPASS** 表示这个 cinder 数据库管理员的密码，而后面的 **controller** 表示连接的控制节点，**连接会走管理网络**，由于我们配置了域名解析，所以只需要填写主机名字就可。命令的完整意思是：采用用户名 **glance** 和密码 **cinder\_dbpass** 去连接控制节点 **controller** 上的 **glance** 数据库。

这句命令也对应，本地对数据库的连接，我们之前也已经授权了本地的权限，如本节命令【对 glance 数据库赋予相应的权限】。

- 在 **[keystone\_authtoken]** 和 **[paste\_deploy]** 部分，配置认证服务访问：

```
[keystone_authtoken]
```

```
...
```

```
auth_uri = http://controller:5000
```

```
auth_url = http://controller:35357
```

```
memcached_servers = controller:11211
```

```
auth_type = password
```

```
//走管理网络，可以找到控制节点，指明认证的 URL 地址
```

```
auth_plugin = password
```

```
//认证采用的形式为密码
```

```
project_domain_name = default
```

```
user_domain_name = default
```

```
project_name = service
```

```
username = glance
password = GLANCE_PASS
```

//此处为在 keystone 认证服务中，注册的 glance 用户名及其密码。

在 [keystone\_authtoken] 中注释或者删除其他选项。

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

- 在 [glance\_store]部分，配置本地文件系统存储和镜像文件位置（后继修改为 ceph 对接存储）：

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

解释：/glanceStore

指明镜像存储采用的形式，此处配置为采用文件存储来存储镜像。

真实生产环境，我们采用统一后端存储。

- ◆ 编辑文件 /etc/glance/glance-registry.conf 并完成如下动作：

- 在 [DEFAULT]部分，配置 noop 禁用通知，因为他们只适合与可选的 Telemetry 服务，并且启用详细日志：

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

- 在 [database] 部分，配置数据库访问：

```
[database]
...
connection=mysql+pymysql://glance:GLANCE_DBPASS@controller/glance
```

解释：

前面的 glance 表示 glance 的 mysql 数据库的管理员。后面的 GLANCE\_DBPASS 表示这个 cinder 数据库管理员的密码，而后面的 controller 表示连接的控制节点，**连接会走管理网络**，由于我们配置了域名解析，所以只需要填写主机名字就可。命令的完整意思是：采用用户名 glance 和密码 cinder\_dbpass 去连接控制节点 controller 上的 glance 数据库。

这句命令也对应，本地对数据库的连接，我们之前也已经授权了本地的权限，如本节命令【对 glance 数据库赋予相应的权限】。

- 在 [keystone\_authtoken] 和 [paste\_deploy] 部分，配置认证服务访问：

```
[keystone_authtoken]
...
auth_uri = http://controller:5000
```

```
auth_url = http://controller:35357
//走管理网络，可以找到控制节点，指明认证的 URL 地址
auth_plugin = password
//认证采用的形式为密码
project_domain_name = default
user_domain_name = default
project_name = service
username = glance
password = GLANCE_PASS
//此处为在 keystone 认证服务中，注册的 glance 用户名及其密码。
在 [keystone_authtoken] 中注释或者删除其他选项。
```

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

- 写入镜像服务数据库  

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

◆ 启动服务，确认安装完成

```
service glance-registry restart
service glance-api restart
```

◆ 默认情况下，Ubuntu 上的安装包会自动创建一个 SQLite 数据库。  
 因为这里配置使用 SQL 数据库服务器，所以你可以 SQLite 服务库文件：

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

◆ 验证 Glance 服务

- 配置 glance api 版本为 v2(M 版本不需要已经添加到脚本中)  

```
echo "export OS_IMAGE_API_VERSION=2"|tee -a admin-openrc.sh \
demo-openrc.sh
```
- 生效环境变量  

```
. admin-openrc
```
- 下载一个测试镜像（需要连接外网）  

```
wget http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img
```
- 使用 Glance 服务 upload 镜像到存储  

```
glance image-create --name "cirros" --file cirros-0.3.4-x86_64-disk.img \
--disk-format qcow2 --container-format bare --visibility public --progress
```

```

root@controller:/# source admin-openrc.sh
root@controller:/# glance image-create --name "cirros" --file cirros-0.3.4-x86_64-disk.img \
> --disk-format qcow2 --container-format bare --visibility public --progress
[=====] 100%
+-----+-----+
| Property | Value |
+-----+-----+
| checksum | eeleca47dc88f4879d8a229cc70a07c6 |
| container_format | bare |
| created_at | 2017-05-21T02:33:21Z |
| disk_format | qcow2 |
| id | 777e8795-5e55-48fe-9e86-81874cfd215d |
| min_disk | 0 |
| min_ram | 0 |
| name | "cirros" |
| owner | 08b5f326225e4cf8bb80e71e9d95bf96 |
| protected | False |
| size | 13287936 |
| status | active |
| tags | [] |
| updated_at | 2017-05-21T02:33:21Z |
| virtual_size | None |
| visibility | public |
+-----+-----+

```

## 9.4块存储节点

块存储节点是整个集群的计算中心，集群中运行的虚拟机都会在计算节点上，而具体的虚拟机-虚拟磁盘或者其存储都会在共享存储，至于存在那个地址，什么形式（比如多台存储，统一存储 ceph 等），而其调度器是安装在控制节点的，所以存储节点，某些数据的传输，及命令的操作，都需要与控制节点进行通讯。

◆ 其中核心包括：

- 统一时钟：NTP
- 指明控制节点和存储节点的位置等配置
- 身份认证等需要远程连接的信息配置
- 配置网络服务等相关配置
- 配置卷组 LVM，后端以 LVM 驱动，卷组 cinder-volumes，iSCSI 协议和正确的 iSCSI 服务
- 核心部署-块存储 Cinder 组件（存储部分）

◆ 主要安装有：

lvm2, cinder-volume, python-mysqldb, networking, NTP, MySql Client 等

◆ 物理主机部分：

- 配置管理网络-网卡信息-特别重要（走管理网络，主机名解析）
- 配置外部网络-网卡信息-特别重要，承担虚拟机的外网服务和对其远程连接（也相当于是承担 VM 网络）。
- 配置存储网络-网卡信息-特别重要（走存储网络，数据、镜像等传输）

◆ 补充说明：

## 9.4.1 各主机基本网络配置

- ◆ 网络配置 (细节请查看网络地址分配)

配置 **/etc/network/interfaces**

IP 地址采用静态配置格式如下, 启用了多少个网卡, 就配置几个, 格式如下:

```
auto lo
iface lo inet loopback
```

```
auto eth1
iface eth1 inet static
address    IP 地址
netmask    掩码
gateway    网关地址
network    网络号 //非必要
broadcast  广播地址 //非必要
```

.....

主网络配网关, 其他不要配, 否则网络不能连接

- ◆ 修改主机名字 hostname

配置 **/etc/hostname**

```
hostname 主机名
```

//指明某台服务器的主机名字

- ◆ 修改域名与主机对应关系

配置 **/etc/hosts**

```
IP 地址 主机名
```

//将集群里面的所有机器 (一行一个主机), 按照这样的对应关系配置好, 非常重要需要主机名解析, 相当于 ping 主机名也是可以通的!!! (走管理网络-所以填写都是各自物理服务器管理网络网卡接口的 IP 地址)

- ◆ 设置 DNS

配置 **/etc/resolv.conf**

```
nameserver IP 地址
```

//后面的 IP 地址是域名服务器 (是正常的外网 DNS 地址, 比如学校是 202.120.111.3)

- ◆ 防止重启机器后 DNS 重置

配置 **/etc/resolvconf/resolv.conf.d/tail**

```
nameserver IP 地址
```

//后面的 IP 地址是域名服务器-同上

- ◆ 重启网络

**/etc/init.d/networking restart**

**检验标准：**

- 1、网络各个网卡已经工作，并且在 `ifconfig` 中看到配置已经启用。
- 2、网络能访问外网，例如可以 `ping` 通百度。
- 3、能进行域名解析，即 `ping` 集群中各个主机名也是可以通的（走管理网络）。
- 4、若配置都是正确的，网卡没有启动或者配置没有生效，请 `reboot` 服务器。
- 5、特别注意：网卡的名字，有些服务器命名为 `ethx`，有些服务器是 `emx`，其中 `x` 用阿拉伯数字表示 `0-n`。

## 9.4.2 在块存储节点部署 Cinder 组件及配置服务

### ◆ 安装 NTP 服务

```
sudo apt-get chrony
```

- 修改配置文件 `/etc/chrony/chrony.conf`，使得其指向控制节点，并将控制节点作为时钟基准源

```
server controller iburst
```

#注释掉所有以 server 开头的配置项，并在末尾添加此。

解释：

此处的计算节点会 **走管理网络**，并找到控制节点，并且以控制节点作为主时钟基准源，那么此处又要明确，controller 和 IP 是一一对应的，即可以 ping 主机名字，要也可以 ping ip 地址，此处走的是管理网络（从另外一方面讲，只有管理网络才是真正全部连接，控制节点，计算节点和存储节点等所有节点）。

- 重启 NTP 服务

```
service chrony restart
```

- 验证 NTP 服务的安装

```
chronyc sources
```

### ◆ 安装 OpenStack-Liberty Package 包

```
apt-get install software-properties-common
```

```
add-apt-repository cloud-archive:mitaka
```

### ◆ 安装 OpenStack-Liberty 客户端

```
apt-get install python-openstackclient
```

### ◆ 升级并重启系统

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

```
reboot
```

根据之前的分析，任何在 OpenStack 执行服务安装，组件部署的时候，都需要向 KeyStone 获得一个管理员的 Token，其实就是相当于获得一个管理员的令牌权限才能操作。所以，我们需要之前的定义 **声明一下接下来的操作是管理员在做**，执行 `source admin-openrc.sh`。具体的 `admin-openrc.sh` 内容如下，原理解释请查看上文控制节点安装 keystone-创建 OpenStack 客户端环境脚本，7.1.3 节。

```
export OS_PROJECT_DOMAIN_NAME=default //指明项目处于什么域
export OS_USER_DOMAIN_NAME=default //指明用户处于什么域
export OS_PROJECT_NAME=admin //指明项目的名字
export OS_USERNAME=admin //指明用户名名字
export OS_PASSWORD=ADMIN_PASS //指明用户名的密码
export OS_AUTH_URL=http://controller:35357/v3 //认证的 URL，也说明了之前为啥会装个 apache 服务器，其实 endpoint 暴露出来之后，其认证的形式，
```

都是采用 URL 进行的（走管理网络-域名解析）。

```
export OS_IDENTITY_API_VERSION=3 //API 版本号
```

```
export OS_IMAGE_API_VERSION=2 //后面镜像时候用
```

注意：将 **ADMIN\_PASS** 替换为你在认证服务中为 **admin** 用户选择的密码。

```
. admin-openrc //别忘了执行
```

如果启动 Ceph 作为后端存储，那么无需配置 lvm

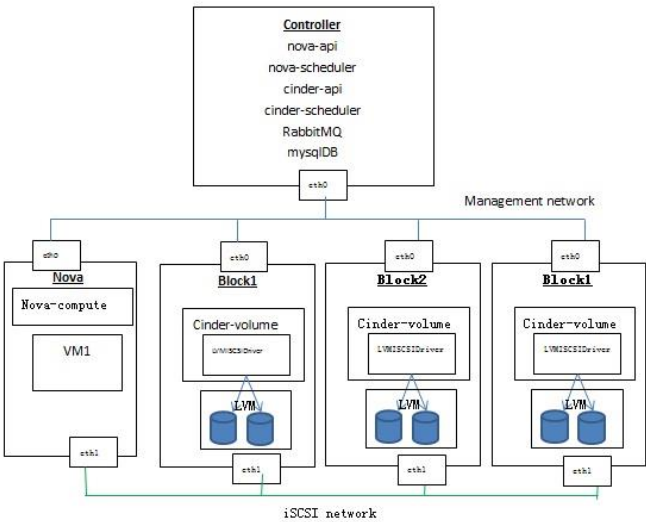
◆ 安装工具支持包软件

```
apt-get install lvm2
```

需要先安装 LVM2 的工具支持包

具体配置请查看官方文档(一般服务器在安装系统的时候，都已经使用 LVM)

[https://docs.openstack.org/liberty/zh\\_CN/install-guide-ubuntu/cinder-storage-install.html](https://docs.openstack.org/liberty/zh_CN/install-guide-ubuntu/cinder-storage-install.html)



在本实验，我们采用 LVM 形式进行提供块存储。LVM 表示逻辑卷管理，当服务器存在很多磁盘的时候，在这些磁盘上建立一层逻辑层，屏蔽底下的具体硬盘，对上提供的就是一个大硬盘（由底下的这些小硬盘所组成）。类似于磁盘阵列，我们实验中，将镜像节点和 cinder 节点是联合在一起的。我们划分磁盘阵列其中 1T 的空间装操作系统，其他全部划分为另外一个分区（采用 fdisk 操作）。真实生产环境，可以单独使用 LVM，整个 sdb 空间，或采用 Ceph 作后端存储。

操作系统/dev/sda	另外一个大分区/dev/sdb	
	分区划分为 glance 存储 /dev/sdb1	分区划分为 cinder 存储 /dev/sdb2

创建物理卷

```
pvcreate /dev/sdb2
```

在创建的物理卷上创建卷组（特别注意，卷组名称在后文的 LVM 中对应）

```
vgcreate cinder-volumes /dev/sdb2
```

官方原话：

只有实例可以访问块存储卷组。不过，底层的操作系统管理这些设备并将其与卷关联。默认情况下，LVM 卷扫描工具会扫描``/dev`` 目录，查找包含卷的块存储设备。如果项目在他们的卷上使用 LVM，扫描工具检测到这些卷时会尝试缓存它们，可能会在底层操作系统和项目卷上产生各种问题。您必须重新配置 LVM，让它只扫描包含``cinder-volume``卷组的设备。编辑``/etc/lvm/lvm.conf``文件并完成下面的操作：

在``devices``部分，添加一个过滤器，只接受``/dev/sdb``设备，拒绝其他所有

设备：

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

每个过滤器组中的元素都以 a(accept) 开头，或 r(reject)开头，并且包括一个设备名称的正则表达式规则。过滤器组必须以“r/.\*/”结束，过滤所有保留设备。

#### ⚠ 警告

如果您的存储节点在操作系统磁盘上使用了 LVM，您还必需添加相关的设备到过滤器中。例如，如果 /dev/sda 设备包含操作系统：

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

这里是不是少了上面这个图？如果操作系统磁盘使用了 LVM，也需要把它加入过滤器。

---

注：如果 **Compute** 节点操作系统磁盘（如 /dev/sda）也使用 LVM，则必须在 **Compute** 节点上修改配置文件 `sudo vi /etc/lvm/lvm.conf`，将操作系统磁盘添加到过滤器允许访问列表。如下：

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

◆ 安装软件

```
apt-get install cinder-volume
```

◆ 编辑 `/etc/cinder/cinder.conf`，同时完成如下操作：

- 在`[DEFAULT]`，配置 RabbitMQ 消息队列，认证服务，启用网络服务支持，启用详细日志，启用后端 LVM 等

```
[DEFAULT]
```

```
...
```

```
rpc_backend = rabbit           //配置采用 RabbitMQ 作为消息队列
```

```
auth_strategy = keystone       //指明，认证采用 KeyStone 服务
```

```
my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
```

```
//替换为块存储节点上的管理网络接口的 IP 地址
```

```
-----  
enabled_backends = lvm         //采用 LVM 形式作为后端存储(上文已介绍)
```

//假如配置成 ceph 后端，那么需要修改 lvm，当前上文的 lvm 也就无需安装了，后面会对接 ceph 存储的。

```
-----  
glance_api_servers = http://controller:9292
```

```
//指明 Glance 节点的位置，指明主机名字，通过管理网络
```

```
verbose = True
```

- 在 `[database]` 部分，配置数据库访问：

```
[database]
```

```
...
```

```
connection = mysql+pymysql://cinder:CINDER_DBPASS@controller/cinder
```

解释：

前面的 `cinder` 表示 cinder 是 mysql 数据库管理员。后面的 `CINDER_DBPASS` 表示这个 cinder 数据库管理员的密码，而后面的 `controller` 表示连接的控制节点，**连接会走管理网络**，由于我们配置了域名解析，所以只需要填写主机名字就可。命令的完整意思是：采用用户名 `cinder` 和密码 `cinder_dbpass` 去连接控制节点 `controller` 上的 `cinder` 数据库。

这句命令也对应，本地对数据库的连接，我们之前也已经授权了本地的权限，如本节命令【对 cinder 数据库赋予相应的权限】。

- 在`[oslo_messaging_rabbit]`部分，配置 RabbitMQ 消息队列访问：

```
[oslo_messaging_rabbit]
```

```
...
```

```
rabbit_host = controller      //走管理网络，能找到这台主机（能域名解析）
```

```
rabbit_userid = openstack     //这个其实是 RabbitMQ 的管理员用户名
```

```
rabbit_password = RABBIT_PASS//所对应的管理员密码
```

- 在[keystone\_authtoken]部分，配置认证服务访问：

[keystone\_authtoken]

...

auth\_uri = http://controller:5000

auth\_url = http://controller:35357

memcached\_servers = controller:11211

//走管理网络，可以找到控制节点，指明认证的 URL 地址

auth\_plugin = password

//认证采用的形式为密码

project\_domain\_name = default

user\_domain\_name = default

project\_name = service

//注册服务实体的名字，注册的是一个服务

username = cinder

password = CINDER\_PASS

//此处为在 keystone 认证服务中，注册的 cinder 用户名及其密码。

在 [keystone\_authtoken] 中注释或者删除其他选项。

-----  
//假如配置成 ceph 后端，那么需要修改 lvm，当前上文的 lvm 也就无需安装了，后面会对接 ceph 存储的。

- 在[lvm]部分，配置 LVM 后端采用 LVM 存储形式，卷组名 cinder-volumes，iSCSI 访问协议：

[lvm]

...

volume\_driver = cinder.volume.drivers.lvm.LVMVolumeDriver

volume\_group = cinder-volumes //卷组名称与建立的卷组一致

iscsi\_protocol = iscsi （有兴趣的同学可以了解 iscsi，此处我不再简述）

iscsi\_helper = tgtadm

- 在 [oslo\_concurrency] 部分，配置锁路径：

[oslo\_concurrency]

...

lock\_path = /var/lib/cinder/tmp

◆ 启动服务，确认安装完成

service tgt restart

service cinder-volume restart

◆ 默认情况下，Ubuntu 上的安装包会自动创建一个 SQLite 数据库。  
因为这里配置使用 SQL 数据库服务器，所以你可以 SQLite 服务库文件：

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

◆ 验证是否安装成功

```
. admin-openrc
```

```
cinder service-list
```

```
root@cinder:~# source admin-openrc.sh
root@cinder:~# cinder service-list
+-----+-----+-----+-----+-----+-----+
| Binary | Host | Zone | Status | State | Updated_at |
| Disabled Reason |
+-----+-----+-----+-----+-----+-----+
| cinder-scheduler | controller | nova | enabled | up | 2017-06-15T09:48:45.000000 |
| - | |
| cinder-volume | cinder@lvm | nova | enabled | up | 2017-06-15T09:48:52.000000 |
| - | |
+-----+-----+-----+-----+-----+-----+
```

## 10 调试日志地址说明

◆ 对于分布式安装的 OpenStack，日志一般存放在 `/var/log/xxx` 目录下。  
比如 nova 放在 `/var/log/nova` 下，Glance 放在 `/var/log/glance` 下等，每个子服务的日志文件是单独保存，比如 nova-api 的日志一般就命名为 `/var/log/nova/api.log`，其他类似。

◆ 日志的格式

**<时间戳><日志等级> <Request ID><日志内容><源代码位置>**

- **时间戳：**日志记录的时间，包括年，月，日，时，分，秒。
- **日志等级：**有 INFO,WARNING,ERROR,DEBUG 等
- **代码模块：**当前运行的代码模块
- **Request ID：**日志会记录连续不同的操作，每个操作都被分配唯一的 Request ID，便于查找。
- **日志内容：**这是日志的主体，记录当前正在执行的操作和结果等重要信息。
- **源代码位置：**日志代码的位置，包括方法名称，源代码文件的目录位置和行号，但是这一些不是所有日志都有的。

补充说明：

## 11 原理介绍和关键词解释(未完成,具体参考 L 版本)

### 11.1.1 Endpoint 原理

#### Endpoint:

一个可以通过网络来访问和定位某个 Openstack service 的地址，通常是一个 URL。比如，当 Nova 需要访问 Glance 服务去获取 image 时，Nova 通过访问 Keystone 拿到 Glance 的 endpoint，然后通过访问该 endpoint 去获取 Glance 服务。我们可以通过 Endpoint 的 region 属性去定义多个 region。

#### Endpoint 该使用对象分为三类:

admin url -> 给 admin 用户使用，Post: 35357

internal url -> OpenStack 内部服务使用来跟别的服务通信，Port: 5000

public url -> 其它用户可以访问的地址，Post: 5000

创建完 service 后创建 API EndPoint. 在 openstack 中，每一个 service 都有三种 end points. Admin, public, internal。Admin 是用作管理用途的，如它能够修改 user/tenant(project)。public 是让客户调用的，比如可以部署在外网上让客户可以管理自己的云。internal 是 openstack 内部调用的。三种 endpoints 在网络上开放的权限一般也不同。Admin 通常只能对内网开放，public 通常可以对外网开放 internal 通常只能对安装有 openstack 对服务的机器开放。

## 11.1.2 Keystone 认证及访问过程分析

结合手写稿扫描件（另附），详细理解 Keystone 的认证过程，及各个服务组件之间，是如何向 keyStone 申请 endpoint-服务断电，又是怎么向 keystone 申请 token 的。

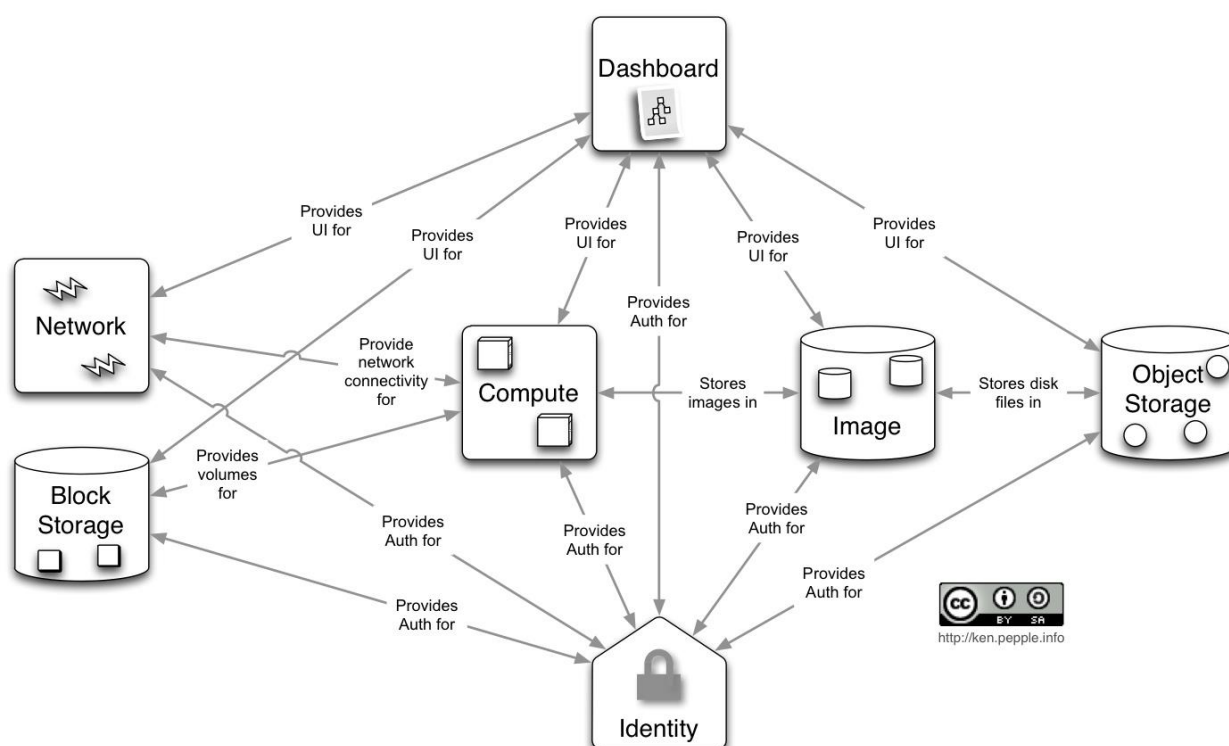
特别了解：keystone 5000 端口是干什么用的。

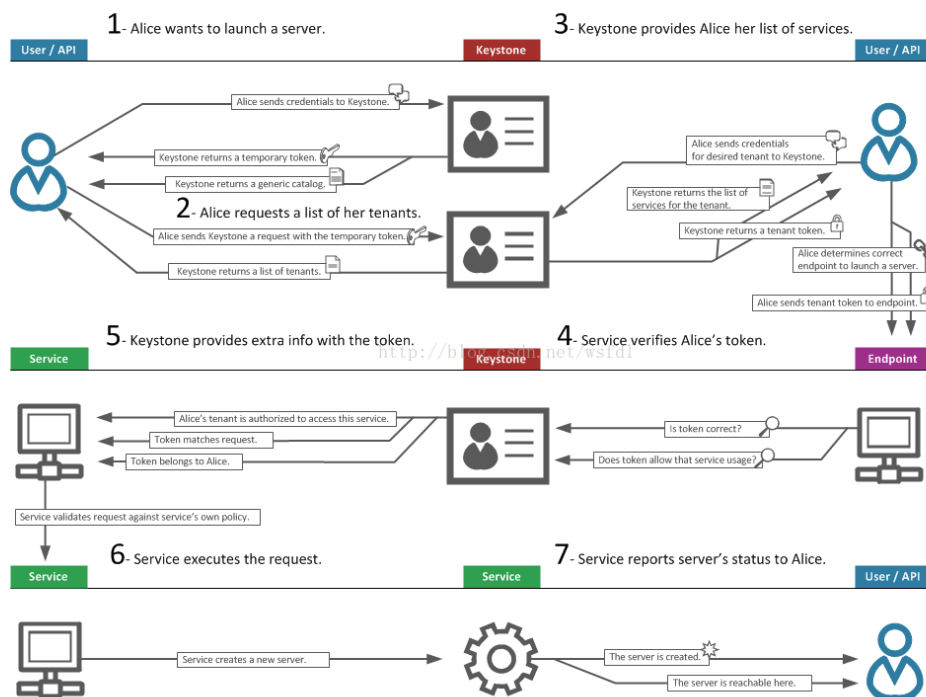
明确，例如 nova 要使用 glance，它是如何用配置的认证信息去申请 token，然后向 5000 端口发起查询，glance 的服务端点 endpoint 的。

由于 nova, cinder 等都是有调度器，并且具体操作都是在调度器的作用下，进行工作的，所以，我们多节点部署的时候，都是存在在控制节点上安装调度器组件，而在具体节点上安装，其他组件。

也说明了，为什么我们在服务端点的时候，会填写控制节点的外网 IP 及控制节点的管理端口 IP，而 Glance 是不一样的写法！！

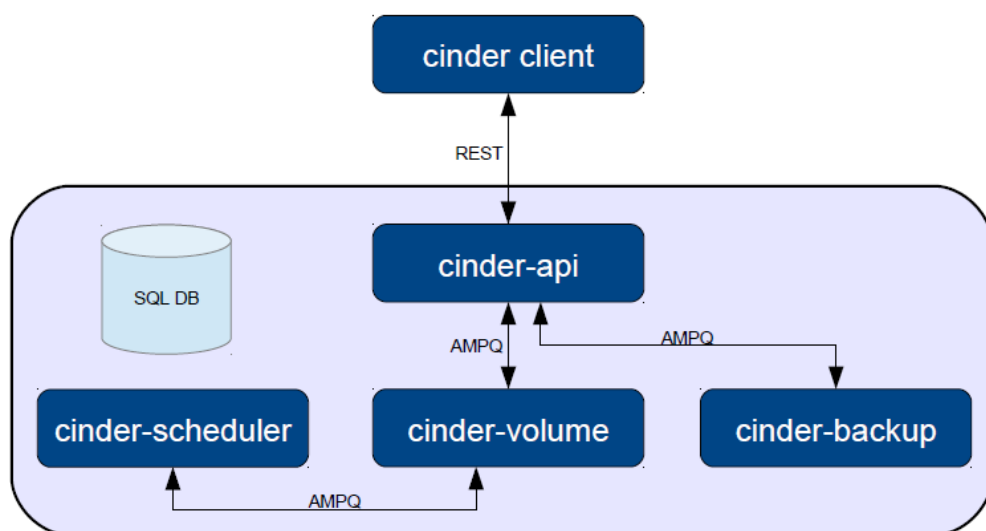
**必须非常详细了解！！**





### 11.1.3 Nova 调度原理

#### 11.1.4 Cinder 调度原理



## 11.1.5 OpenStack 项目与组件介绍

### 核心项目 3 个

#### 控制台

服务名: Dashboard

项目名: Horizon

功能: web 方式管理云平台, 建云主机, 分配网络, 配安全组, 加云盘

#### 计算

服务名: 计算

项目名: Nova

功能: 负责响应虚拟机创建请求、调度、销毁云主机

#### 网络

服务名: 网络

项目名: Neutron

功能: 实现 SDN (软件定义网络), 提供一整套 API, 用户可以基于该 API 实现自己定义专属网络, 不同厂商可以基于此 API 提供自己的产品实现

### 存储项目 2 个

#### 对象存储

服务名: 对象存储

项目名: Swift

功能: REST 风格的接口和扁平的数据组织结构。RESTFUL HTTP API 来保存和访问任意非结构化数据, ring 环的方式实现数据自动复制和高度可以扩展架构, 保证数据的高度容错和可靠性

#### 块存储

服务名: 块存储

项目名: Cinder

功能: 提供持久化块存储, 即为云主机提供附加云盘。

### 共享服务项目 3 个

#### 认证服务

服务名: 认证服务

项目名: Keystone

功能: 为访问 openstack 各组件提供认证和授权功能, 认证通过后, 提供一个服务列表 (存放你有权访问的服务), 可以通过该列表访问各个组件。

#### 镜像服务

服务名: 镜像服务

项目名: Glance

功能: 为云主机安装操作系统提供不同的镜像选择

## 计费服务

服务名：计费服务

项目名：Ceilometer

功能：收集云平台资源使用数据，用来计费或者性能监控

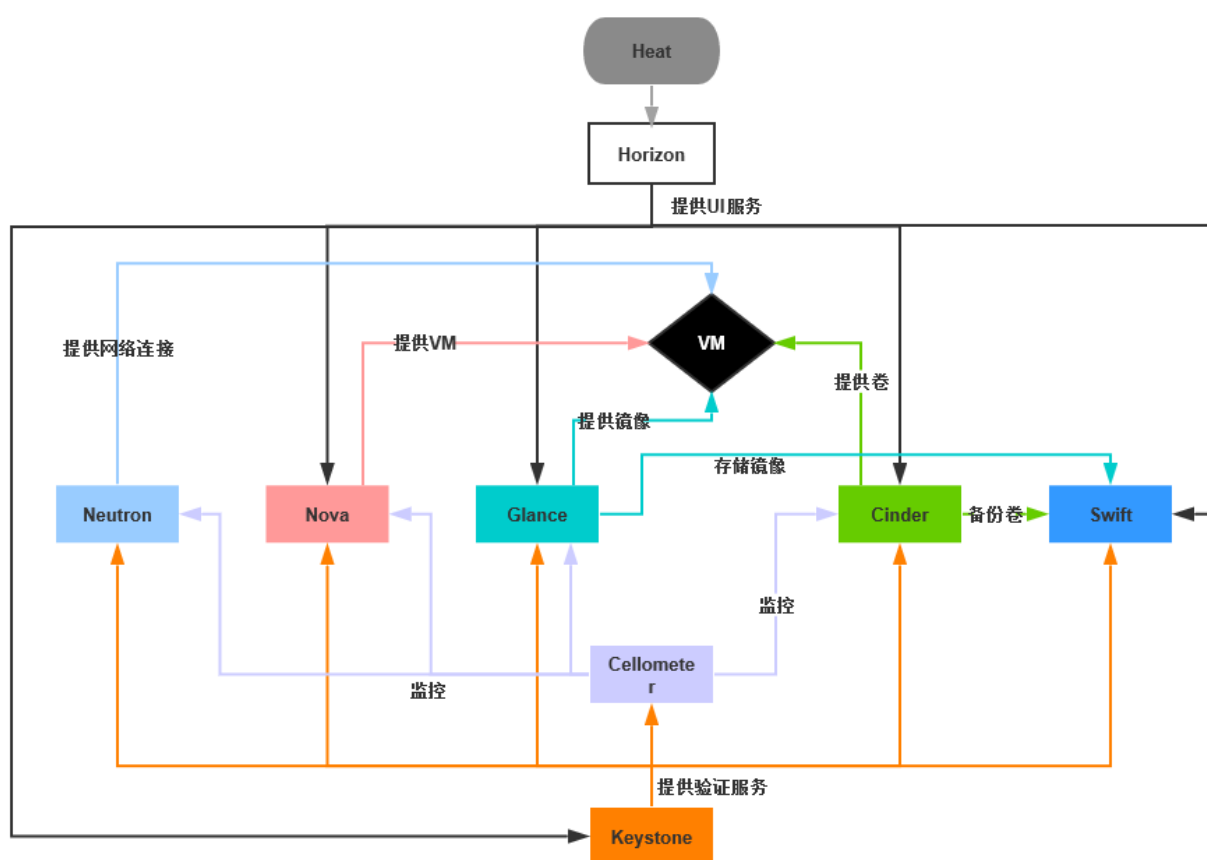
## 高层服务项目 1 个

### 编排服务

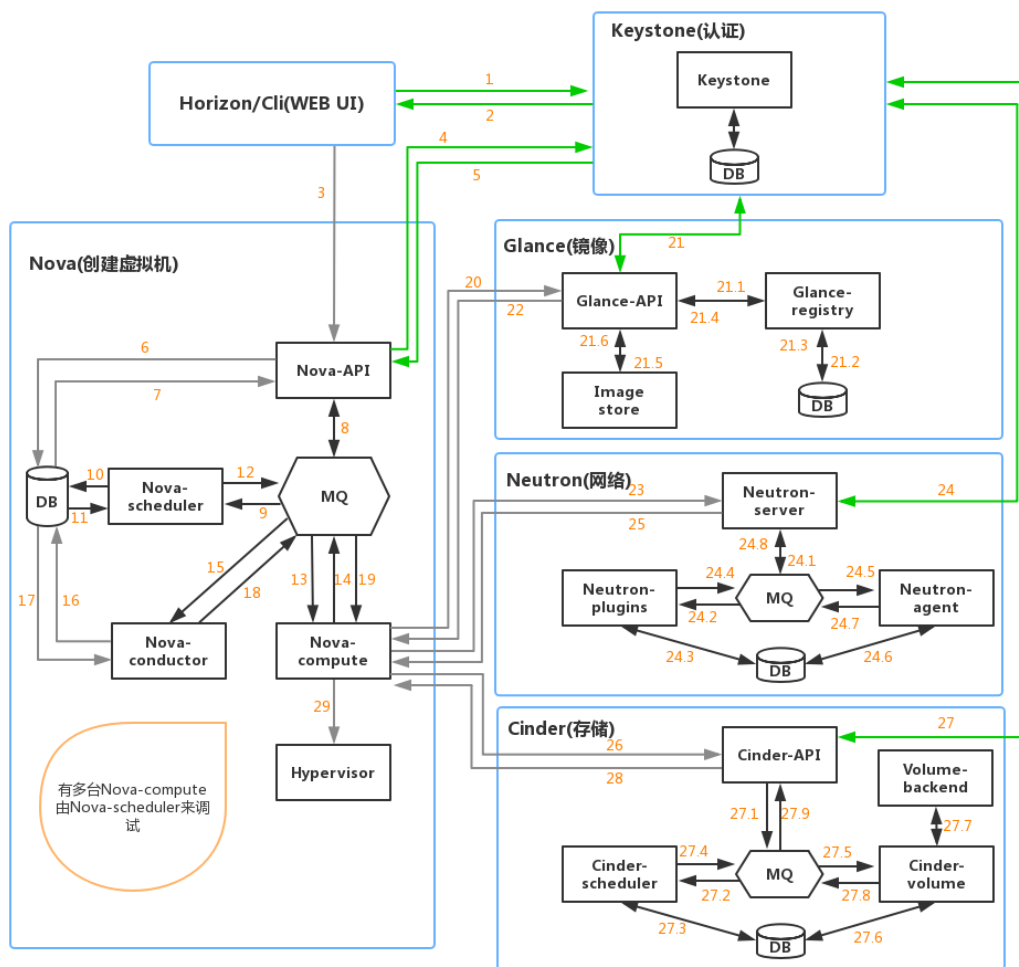
服务名：编排服务

项目名：Heat

功能：自动化部署应用，自动化管理应用整个生命周期. 主要用于 Paas



Openstack各组件逻辑关系图



Openstack新建云主机流程图

- 界面或命令行通过 RESTful API 向 keystone 获取认证信息。
- keystone 通过用户请求认证信息，并生成 auth-token 返回给对应的认证请求。
- 界面或命令行通过 RESTful API 向 nova-api 发送一个 boot instance 的请求(携带 auth-token)，包含需要创建的虚拟机信息，如 CPU、内存、硬盘、网络等。
- nova-api 接受请求后向 keystone 发送认证请求，查看 token 是否为有效用户和 token。
- keystone 验证 token 是否有效，如有效则返回有效的认证和对应的角色（注：有些操作需要有角色权限才能操作）。
- 通过认证后 nova-api 和数据库通讯。
- 初始化新建虚拟机的数据库记录。
- nova-api 通过 rpc.call 向 nova-scheduler 请求是否有创建虚拟机的资源(Host ID)。
- nova-scheduler 进程侦听消息队列，获取 nova-api 的请求。
- nova-scheduler 通过查询 nova 数据库中计算资源的情况，并通过调度算法计

算符合虚拟机创建需要的主机。

- 对于有符合虚拟机创建的主机，nova-scheduler 更新数据库中虚拟机对应的物理主机信息。
- nova-scheduler 通过 rpc.cast 向 nova-compute 发送对应的创建虚拟机请求的消息（调度到选定的 nova-compute 上创建 VM）。
- nova-compute 会从对应的消息队列中获取创建虚拟机请求的消息（新版的 openstack，不允许 Nova-compute 直接连接数据库，只能通过 nova-conductor 去调用），有多个 nova-compute 节点）
- nova-compute 通过 rpc.call 向 nova-conductor 请求获取虚拟机消息。（Flavor）
- nova-conductor 从消息队队列中拿到 nova-compute 请求消息。
- nova-conductor 根据消息查询虚拟机对应的信息。
- nova-conductor 从数据库中获得虚拟机对应信息。
- nova-conductor 把虚拟机信息通过消息的方式发送到消息队列中。
- nova-compute 从对应的消息队列中获取虚拟机信息消息。
- nova-compute 通过 keystone 的 RESTfull API 拿到认证的 token，并通过 HTTP 请求 glance-api 获取创建虚拟机所需要镜像。
- glance-api 向 keystone 认证 token 是否有效，并返回验证结果。
- token 验证通过，nova-compute 获得虚拟机镜像信息(URL)。
- nova-compute 通过 keystone 的 RESTfull API 拿到认证 k 的 token，并通过 HTTP 请求 neutron-server 获取创建虚拟机所需要的网络信息。
- neutron-server 向 keystone 认证 token 是否有效，并返回验证结果。
- token 验证通过，nova-compute 获得虚拟机网络信息。
- nova-compute 通过 keystone 的 RESTfull API 拿到认证的 token，并通过 HTTP 请求 cinder-api 获取创建虚拟机所需要的持久化存储信息。
- cinder-api 向 keystone 认证 token 是否有效，并返回验证结果。
- token 验证通过，nova-compute 获得虚拟机持久化存储信息。
- nova-compute 根据 instance 的信息调用配置的虚拟化驱动来创建虚拟机。

## 12 几类密码汇总说明（需要修改）

- ◆ **KeyStone 组件及对应数据库：**  
**用户名：**统一设定为 keystone  
**密码：**用户自行设定密码  
**作用：**每安装一个核心服务组件，都需要设定一个数据库，由此设定对应数据库管理员账户及密码。服务组件也将通过此账户密码与数据库交互，故此账户密码即是一个服务组件的管理员账户，又是此数据库的管理员账户。
- ◆ **RabbitMQ 组件：**  
**用户名：**设定为 openstack  
**密码：**用户自行设定密码  
**作用：**整个集群命令，消息的流转都需要通过 RabbitMQ 消息组件，那么集群安装此服务后，需要有个管理员管理这个消息组件，由此需要设定一个管理员，其他服务组件发送命令，命令流转都到采用这个管理员用户才可以，相当于只有此管理员才有权限操纵 RabbitMQ。
- ◆ **Nova 组件及对应数据库：**  
**用户名：**统一设定为 nova  
**密码：**用户自行设定密码  
**作用：**理由同 KeyStone
- ◆ **Glance 组件及对应数据库：**  
**用户名：**统一设定为 glance  
**密码：**用户自行设定密码  
**作用：**理由同 KeyStone
- ◆ **Neturon 组件：**  
**用户名：**统一设定为 neutron  
**密码：**用户自行设定密码  
**作用：**理由同 KeyStone
- ◆ **数据库 MySql 服务组件：**  
**用户名：**root  
**密码：**用户自行设定密码  
**作用：**整个集群采用 Mysql 数据作为存储数据库，故安装 Mysql 数据库需要一个管理员，此管理员具有对数据库操作的最高权限，其他服务组件操作数据库都需要用此账号和密码访问。
- ◆ **建立项目，用户和角色**  
建立用户和角色比较重要，集群中每建立一个用户，都需要给其设定角色等级（管理员还是普通用户），故前提条件需要先设定有哪些角色。  
具体如下：

- ◆ 数据库有个总的管理员 **root**，那么其也对应一个总数据库密码。  
在数据库中每建立一个新的服务数据库，那么需要授权一个用户及密码，所以我们每次建立一个服务，就将其名字作为授权的用户及密码，例如 **glance**。

## 13 参考书目及资料

### ◆ 参考官方手册地址：

- <https://docs.openstack.org/>
- [https://docs.openstack.org/zh\\_CN/](https://docs.openstack.org/zh_CN/)
- [https://docs.openstack.org/mitaka/zh\\_CN/install-guide-ubuntu/](https://docs.openstack.org/mitaka/zh_CN/install-guide-ubuntu/) M 版本
- [https://docs.openstack.org/liberty/zh\\_CN/install-guide-ubuntu/](https://docs.openstack.org/liberty/zh_CN/install-guide-ubuntu/) L 版本

### ◆ 强烈推荐参考书：

- 每天 5 分钟玩转 OpenStack （作用：入门级的概念知识点梳理）
- Open Stack 设计与实现（第二版）（作用：原理性介绍 OpenStack，英特尔开源技术中心编著）
- 深入浅出 Neutron:OpenStack 网络技术 （作用：重点介绍 openstack 的虚拟网络，很重要）
- OpenStack 运维指南
- OpenStack 开源云王者归来:云计算、虚拟化、Nova、Swift、Quantum 与 Hadoop