附录 A　综合实训——手动安装和部署 OpenStack 云平台

考虑到实际应用中大多需要手动部署 OpenStack，《OpenStack 云计算实战》一书相关章节中穿插讲解了 OpenStack 主要服务和组件的手动安装和配置的详细步骤。为帮助读者进一步巩固所学的 OpenStack 基础知识和操作技能，我们为该书增补一个综合实训篇，示范手动安装和部署 OpenStack 云平台的完整操作过程。综合实训建立的仍然是一个实验性质的 OpenStack 云平台，而不是生产环境的部署。注意《OpenStack 云计算实战》一书中的 OpenStack 版本为 Queens，目前 CentOS 7 操作系统能够安装的 OpenStack 最高版本是 Train，综合实训选择该 OpenStack 版本进行实验操作。

A.1　综合实训准备

首先进行综合实训准备，确定云部署目标，设计云部署架构，约定 OpenStack 账户密码。

A.1.1　确定云部署目标

本综合实训的目标是通过手动安装方式基于 CentOS 7 操作系统部署一个双节点的 OpenStack 云平台。计划这个整个云平台中部署 7 个基本服务，分别是 Keystone（身份管理服务）、Glance（镜像服务）、Placement（放置服务）、Nova（计算服务）、Neutron（网络服务）、Horizon（仪表板）和 Cinder（块存储服务）。其中前 5 个服务是 OpenStack Train 版本最小化部署所必需的，具体实施中需要按所列服务的顺序安装。

注意 OpenStack 从 Stein 版本开始将 Nova 项目的 nova-placement-api 服务作为一个独立的项目，专门提供放置服务，用于满足计算服务和其他任何服务的资源选择和使用的管理需求。

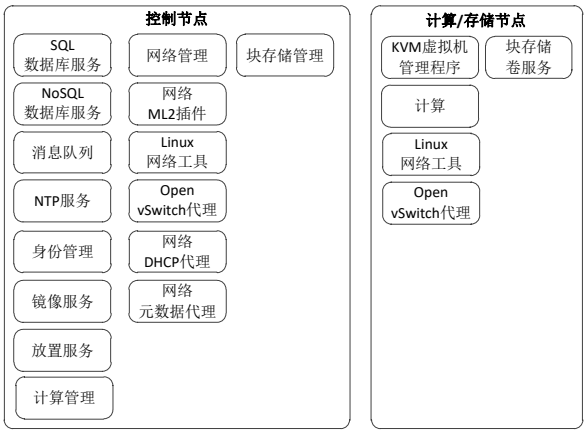

图 A-1　综合实训的服务布局

A.1.2 设计云部署架构

根据综合实训目标，在两个节点主机上进行云部署，以核心组件为主，虚拟网络方案选择自服务网络，L2 网络代理选择 Open vSwitch 代理，如图 A-1 所示。

A.1.3 OpenStack 账户密码约定

为方便实验，本综合实训中对于各个 OpenStack 服务，约定使用“SERVICE_PASS”格式的字符串来表示服务账户密码，使用“SERVICE_DBPASS”格式的字符串来表示数据库中相应用户的密码，其中“SERVICE”为不同服务项目代号的大写名称。例如，镜像服务的服务用户 glance 的密码为“GLANCE_PASS”，数据库用户 glance 的密码为“GLANCE_DBPASS”，计算服务的服务用户 nova 的密码为“NOVA_PASS”，数据库用户 nova 的密码为“NOVA_DBPASS”。当然读者可以使用自己的密码进行替换，只是注意一定要在服务配置文件中保持一致。

A.2 配置 OpenStack 云平台环境

环境配置是后续 OpenStack 各服务安装配置的前提条件。

A.2.1 准备两个节点主机

综合实训需要新建两台 CentOS 7 主机分别充当控制节点和计算节点，为便于实验，这里使用 VMware Workstation 创建虚拟机来实现，建议安装图形界面的 CentOS 7 操作系统。笔者实验环境的控制节点主机配置如下。

- 内存 4GB。
- CPU（处理器）双核。
- 硬盘 60GB。
- 网卡以桥接模式接入宿主机网络（可以连接 Intetnet）。

计算节点主机配置如下。

- 内存 8GB。
- CPU（处理器）双核。
- 硬盘两个，容量分别为 200GB 和 100GB。
- 网卡以桥接模式接入宿主机网络（可以连接 Intetnet）。

两个节点主机安装好 CentOS 7 操作系统之后，都需要进行以下配置以准备 OpenStack 安装环境。注意每个节点主机上的安装配置操作需要管理员权限，必须以 root 用户身份或者通过 sudo 命令来执行操作命令。为简单起见，后续过程中命令行操作均以 root 用户身份进行。

（1）禁用防火墙

```
systemctl disable firewalld
systemctl stop firewalld
```

（2）调整时区设置

安装 CentOS 7 英文版之后，执行以下命令将时区设置为上海。

```
timedatectl set-timezone "Asia/Shanghai"
```

A.2.2　配置节点主机网络

每个节点主机配置两个网卡，第 1 个设置为仅主机模式（用于内网通信），接入管理网络；第 2 个网卡设置为桥接模式（用于外网通信，可访问 Internet），接入提供者网络（外部网络），同时便于在线安装软件包。具体采用的网络拓扑如图 A-2 所示。读者实验时可以根据实际情况修改网络设置和 IP 地址范围。

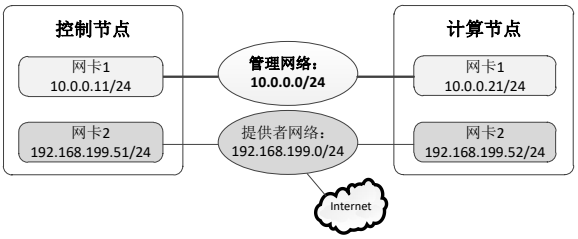

图 A-2　综合实训的网络拓扑

1．停用 NetworkManager 服务

systemctl disable NetworkManager

systemctl stop NetworkManager

systemctl enable network

systemctl start network

2．设置网卡

先在控制节点主机上设置。本例第 1 个网卡的名称为 ens33，通过 /etc/sysconfig/network-scripts/ifcfg-ens33 网卡配置文件进行设置，只需设置 IP 地址，不要设置默认网关（GATEWAY）和 DNS。该网卡的关键设置如下。

IPADDR=10.0.0.11

PREFIX=24

第 2 个网卡的名称为 ens37，通过/etc/sysconfig/network-scripts/ifcfg-ens37 网卡配置文件进行设置，需设置 IP 地址、默认网关和 DNS，便于接入 Internet。该网卡的关键设置如下。

IPADDR=192.168.199.51

PREFIX=24

GATEWAY=192.168.199.1

DNS1=114.114.114.114

设置完毕，执行 systemctl restart network 命令重启 network 服务使网卡设置更改生效。

计算节点主机参照控制节点修改相应的网卡配置，除了 IP 地址，配置基本相同，第 1 个网卡 IP 地址为 10.0.0.21，第 2 个网卡的 IP 地址为 192.168.199.52。

3．配置主机名解析

更改主机名，这里将控制节点主机名更改为 controller。

hostnamectl set-hostname controller

将计算节点主机名更改为 compute1。

一旦更改主机名，就必须将新的主机名追加到/etc/hosts 配置文件中。两个节点主机上的/etc/hosts 文件中都要包括以下配置。

```
# controller
10.0.0.11        controller
# compute1
10.0.0.21        compute1
```

4．测试连通性

完成上述配置之后，可以测试节点主机之间，节点主机与外网之间的连通性。从控制节点上测试到计算节点上管理网卡的连通性。

```
[root@controller ~]# ping -c 2 compute1
```

再从控制节点上测试到 Internet 的连通性。

```
[root@compute1 ~]# ping -c 2 www.163.com
```

再从计算节点上分别测试到控制节点和 Internet 的连通性。

A.2.3 设置时间同步

OpenStack 环境中所有节点的时间必须是同步的。在 CentOS 系统中一般使用时间同步软件 Chrony，如果没有安装，执行 yum install chrony 命令进行安装，本例默认已安装。

这里使用物理主机的 NTP 服务器，在/etc/chrony.conf 配置文件中加入以下语句（192.168.199.201 为 NTP 服务器地址）。

```
server 192.168.199.201 iburst
```

然后重启时间同步服务使设置生效。

```
systemctl restart chronyd.service
```

A.2.4 安装 OpenStack 软件包

两个节点主机都需要安装 OpenStack 软件包，各节点主机上分别进行下列操作。

（1）启用 OpenStack 软件库。

对于 Train 版本，执行以下命令安装 OpenStack 软件源。

```
yum install centos-release-openstack-train
```

（2）执行以下命令升级软件包。

```
yum -y upgrade
```

（3）执行以下命令安装 OpenStack 客户端软件。

```
yum -y install python-openstackclient
```

（4）CentOS 默认启用 SELinux，执行以下命令安装 openstack-selinux 软件包以自动管理 OpenStack 服务的安全策略。

```
yum -y install openstack-selinux
```

（5）完成上述操作之后，执行以下命令验证安装，输出版本号则表示成功安装。

```
[root@controller ~]# openstack --version
openstack 4.0.1
```

A.2.5　安装 SQL 数据库

大多数 OpenStack 服务需要使用 SQL 数据库来存储信息，数据库通常部署在控制节点上。在控制节点上执行下列操作来安装 MariaDB 数据库并进行初始化配置。

（1）执行以下命令安装相关的软件包。

```
[root@controller ~]# yum -y install mariadb mariadb-server python2-PyMySQL
```

（2）编辑/etc/my.cnf.d/openstack.cnf 配置文件，[mysqld]节的设置如下。

```
[mysqld]
bind-address = 10.0.0.11
default-storage-engine = innodb
innodb_file_per_table = on
max_connections = 4096
collation-server = utf8_general_ci
character-set-server = utf8
```

其中 bind-address 选项设置为控制节点的管理 IP 地址，以允许其他节点通过管理网络进行访问，其他选项则用于启用有用的选项和 UTF-8 字符集。

（3）执行以下命令将 MariaDB 设置为开机自动启动，并启动该数据库服务。

```
systemctl enable mariadb.service
systemctl start mariadb.service
```

（4）通过运行 mysql_secure_installation 脚本启动安全配置向导来提高数据库的安全性。

```
[root@controller ~]# mysql_secure_installation
……
Enter current password for root (enter for none):        #默认 root 密码为空，初次运行时直接回车
OK, successfully used password, moving on...
……
Set root password? [Y/n] y          # 是否设置 root 用户密码，输入 y 并回车或直接回车
New password:                       # 输入拟设置的 root 用户密码
Re-enter new password:              # 再次输入相同的密码
Password updated successfully!
Reloading privilege tables..
……
Remove anonymous users? [Y/n] y     #是否删除匿名用户，直接回车以删除
……
Disallow root login remotely? [Y/n] y       #是否禁止 root 远程登录，直接回车以禁止
……
Remove test database and access to it? [Y/n] y     # 是否删除 test 数据库，直接回车
……
Reload privilege tables now? [Y/n] y                # 是否重新加载权限表使修改生效，直接回车
……
installation should now be secure.
```

A.2.6 安装消息队列服务

消息队列服务通常在控制节点上运行。这里选择大多数发行版都支持的 RabbitMQ 消息队列服务进行部署，在控制节点上完成下列操作。

（1）执行以下命令安装相应的软件包。

yum -y install rabbitmq-server

（2）将 RabbitMQ 服务设置为开机自动启动，并启动该消息队列服务。

systemctl enable rabbitmq-server.service

systemctl start rabbitmq-server.service

（3）添加一个名为 openstack 的用户账户。

[root@controller ~]# rabbitmqctl add_user openstack RABBIT_PASS

Creating user "openstack"

这里将其密码设置为 RABBIT_PASS，可用自己的密码进行替换。

（4）执行以下命令为 openstack 用户配置写入和读取访问权限。

[root@controller ~]# rabbitmqctl set_permissions openstack ".*" ".*" ".*"

Setting permissions for user "openstack" in vhost "/"

A.2.7 安装 Memcached 服务

OpenStack 服务的身份管理机制使用 Memcached 来缓存令牌。Memcached 服务通常在控制节点上运行。在控制节点上完成下列操作来安装 Memcached 服务。

（1）安装相应的软件包。

yum -y install memcached python-memcached

（2）编辑/etc/sysconfig/memcached 配置文件，在 OPTIONS 选项值中加入控制节点 controller。

OPTIONS="-l 127.0.0.1,::1,controller"

这使得 Memcached 服务能够使用控制节点的管理 IP 地址，以允许其他节点通过管理网络访问该服务。

（3）将 Memcached 设置为开机自动并启动该服务。

systemctl enable memcached.service

systemctl start memcached.service

A.2.8 安装 Etcd

OpenStack 服务可以使用 Etcd（分布式可靠键值存储）来进行分布式键锁定，存储配置，跟踪服务活动性。Etcd 服务在控制节点上运行，下面在控制节点上安装该服务。

（1）安装软件包。

yum -y install etcd

（2）编辑/etc/etcd/etcd.conf 配置文件，将 ETCD_INITIAL_CLUSTER、ETCD_INITIAL_ADVERTISE_PEER_URLS、ETCD_ADVERTISE_CLIENT_URLS 和 ETCD_LISTEN_CLIENT_URLS 等选项的值设置为控制节点的管理 IP 地址，以允许其他节点通过管理网络访问 Etcd 服务。

#[Member]

```
ETCD_DATA_DIR="/var/lib/etcd/default.etcd"
ETCD_LISTEN_PEER_URLS="http://10.0.0.11:2380"
ETCD_LISTEN_CLIENT_URLS="http://10.0.0.11:2379"
ETCD_NAME="controller"
#[Clustering]
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://10.0.0.11:2380"
ETCD_ADVERTISE_CLIENT_URLS="http://10.0.0.11:2379"
ETCD_INITIAL_CLUSTER="controller=http://10.0.0.11:2380"
ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster-01"
ETCD_INITIAL_CLUSTER_STATE="new"
```

（3）将 Etcd 服务设置为开机自动启动，并启动该服务。

```
systemctl enable etcd
systemctl start etcd
```

A.3　安装和部署 Keystone

除了基础环境之外，Keystone 身份管理服务是需要第一个安装的 OpenStack 服务。该服务通常单独安装在控制节点上。考虑到扩展性，需要部署 Fernet 令牌和 Apache HTTP 服务器来处理认证请求。下面的操作都是在控制节点上进行的。

A.3.1　创建 Keystone 数据库

安装和配置身份服务之前，必须创建一个数据库。每个组件 OpenStack 都要有一个自己的数据库，Keystone 也不例外，需要在后端安装一个数据库用来存放用户的相关数据。

（1）使用数据库访问客户端以 root 用户身份连接到数据库服务器，输入正确的 root 密码后进入到 MariaDB 客户端交互操作界面。

（2）执行以下命令创建 Keystone 数据库（名称为 keystone）。

```
MariaDB [(none)]> CREATE DATABASE keystone;
Query OK, 1 row affected (0.000 sec)
```

（3）再依次执行以下两条命令对 Keystone 数据库授予合适的账户访问权限（本例 keystone 账户的数据库访问密码设置为 KEYSTONE_DBPASS）。

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' \
IDENTIFIED BY 'KEYSTONE_DBPASS';        # 授予来自本地的 keystone 账户全部权限
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' \
IDENTIFIED BY 'KEYSTONE_DBPASS';      #授予来自任何地址的 keystone 账户全部权限
```

（4）执行以下命令退出数据库访问客户端。

```
MariaDB [(none)]> exit
Bye
```

A.3.2 安装和配置 Keystone 及相关组件

1．执行以下命令安装所需的软件包。

yum -y install openstack-keystone httpd mod_wsgi

其中 openstack-keystone 是 Keystone 软件包名。Keystone 是基于 WSGI 的 Web 应用程序，而 httpd 是一个兼容 WSGI 的 Web 服务器，因此还需安装 httpd 及其 mod_wsgi 模块。

2．编辑/etc/keystone/keystone.conf 配置文件，完成下列配置任务

（1）在[database]节中配置数据库访问。

[database]

...

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

设置该选项的目的是让 Keystone 服务能知道如何连接到后端的数据库 keystone。其中 pymysql 是一个可以操作 mysql 的 python 库。双斜杠后面的格式为：用户名:密码@mysql 服务器地址/数据库。还应注意将[database]节中的其他连接配置注释掉，或直接删除。

（2）在[token]节中配置 Fernet 令牌提供者。

[token]

...

provider = fernet

该选项默认被注释掉，其中 fernet 是一种生成令牌的方式，还有一种方式是 pki。

3．初始化 Keystone 数据库

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

Python 的对象关系映射（ORM）需要初始化，以生成数据库表结构。

4．初始化 Fernet 密钥库以生成令牌

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

keystone-manage credential_setup --keystone-user keystone --keystone-group keystone

这两个命令实际上完成了 Keystone 对自己授权的一个过程，创建了一个 keystone 用户与一个 keystone 组，并对这个用户和组授权。因为 keystone 是对其他组件提供认证的服务，所以它先要对自己进行一下认证。

5．对 Keystone 服务执行初始化操作

```
keystone-manage bootstrap --bootstrap-password ADMIN_PASS \
  --bootstrap-admin-url http://controller:5000/v3/ \
  --bootstrap-internal-url http://controller:5000/v3/ \
  --bootstrap-public-url http://controller:5000/v3/ \
  --bootstrap-region-id RegionOne
```

该命令实际上是执行基本的初始化过程，为 Keystone 服务设置管理员密码，并创建 API 端点。在 OpenStack 的 Queens 版本发布之前，Keystone 需要在两个分开的端口上运行，

以适应 Identity v2 API（它在 35357 端口上运行单独的管理服务）。

A.3.3　配置 Apache HTTP 服务器

必须将 Web 服务器与 Keystone 进行整合，这就涉及了 WSGI。

（1）编辑/etc/httpd/conf/httpd.conf 文件，配置 ServerName 选项，使其指向控制节点。

ServerName controller

（2）创建一个到/usr/share/keystone/wsgi-keystone.conf 文件的链接文件。

ln -s /usr/share/keystone/wsgi-keystone.conf /etc/httpd/conf.d/

这实际上是为 mod_wsgi 模块添加配置文件，除了做软链接，还可以直接复制该文件。

A.3.4　完成 Keystone 安装

（1）启动 Apache HTTP 服务并将其配置开机自动启动。

systemctl enable httpd.service

systemctl start httpd.service

（2）通过导出环境变量来配置管理员账户。

```
export OS_USERNAME=admin                          # 管理员账户
export OS_PASSWORD=ADMIN_PASS                     # 密码
export OS_PROJECT_NAME=admin                      # 项目名
export OS_USER_DOMAIN_NAME=Default                # 域名
export OS_PROJECT_DOMAIN_NAME=Default
export OS_AUTH_URL=http://controller:5000/v3      # 认证 URL
export OS_IDENTITY_API_VERSION=3                  # 指定版本信息
```

这些管理员账户配置是由上述 keystone-manage bootstrap 命令产生的。如果想让用户获取权限必须要指定用户所在的项目。

（3）验证 Keystone 安装。执行 openstack service list 命令查看服务列表，可发现已包括 Keystone 服务。

可以直接在控制节点上访问 http://controller:5000/v3 网址来验证是否能够访问 Keystone 服务，结果表明能够成功访问。

[root@controller ~]# curl http:// controller:5000/v3

{"version": {"status": "stable", "updated": "2019-07-19T00:00:00Z", "media-types": [{"base": "application/json", "type": "application/vnd.openstack.identity-v3+json"}], "id": "v3.13", "links": [{"href": "http://192.168.199.51:5000/v3/", "rel": "self"}]}}

A.3.5　创建域、项目、用户和角色

Keystone 服务为每个 OpenStack 服务提供认证服务。认证服务需要使用域、项目、用户和角色的组合。注意在控制节点上执行下面的操作之前，确认已经在当前命令行中导出了管理员账户的环境变量（参见 A.3.4 节）。

1. 创建域

Keystone 安装过程中默认已经创建了一个默认域，执行 openstack domain list 命令可以

查看。可以发现，该默认域的 ID 为 default，名称为 Default。可根据需要再创建自己的域。

2．创建项目

执行 openstack project list 命令可以查看默认已经创建的项目。可以发现，默认仅创建一个名为 admin 的项目供管理员使用。其他 OpenStack 服务要通过 Keystone 进行集中统一认证，必须进行注册，本综合实训云部署所有的 OpenStack 服务共用一个名为 service 的项目，其中包含添加到环境中每个服务的一个唯一用户。执行以下命令创建该项目。

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

3．创建用户

执行 openstack user list 命令可以查看默认已经创建的用户。常规任务应使用无特权的项目和用户。本综合实训中需要创建 demo 项目和 demo 用户用于测试。创建 demo 项目的命令如下。

```
openstack project create --domain default    --description "Demo Project"    demo
```

接下来再创建 demo 用户，将其密码设置为 DEMO_PASS。

```
[root@controller ~]# openstack user create --domain default    --password-prompt demo
User Password:
Repeat User Password:
```

4．创建角色

默认 Keystone 提供 3 个角色：admin、member 和 reader，执行以下命令可以查看。

```
[root@controller ~]# openstack role list
+----------------------------------+--------+
| ID                               | Name   |
+----------------------------------+--------+
| 03663721ff444d828d26858ab66ecb3f | admin  |
| a3a328fa9f47463ea273cba681d428cd | reader |
| a77aac520661470db5d0ff8fb3f56118 | member |
```

这里再创建一个名为 demo 的角色。

```
openstack role create demo
```

管理员可以将这些角色分配给某个项目、某个域或整个系统的用户或组。本综合实训中需要将 member 角色添加到 demo 项目和 demo 用户，执行以下命令即可。

```
openstack role add --project demo --user demo member
```

A.3.6　验证 Keystone 服务的安装

在安装其他服务之前，在控制节点上验证 Keystone 身份管理服务的操作。

（1）如果当前已经设置了 admin 用户的环境变量，则执行以下命令取消临时设置的 OS_AUTH_URL 和 OS_PASSWORD 环境变量。

```
unset OS_AUTH_URL OS_PASSWORD
```

（2）如图 A-3 所示，执行命令以 admin 用户身份请求认证令牌，这里 admin 的密码是

ADMIN_PASS。

```
[root@controller ~]# openstack --os-auth-url http://controller:5000/v3   --os-project-domain-name Default
--os-user-domain-name Default   --os-project-name admin --os-username admin token issue
Password:
Password:
+------------+-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------+
| Field      | Value
                                                                                                |
+------------+-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------+
| expires    | 2020-10-19T13:45:53+0000
                                                                                                |
| id         | gAAAAABfjYqBWKU3puH60ER5l92gHQVJzLXiyw5vstSYw-9bpy1lwck2LdCST-W9wtUpodGyBpiX1qE_LfvZ39n3eJa
pBfzJyTp378XWNDtk0RO1nZttb6G3Xj2JkCcbsVWkAFwXQu3NnvQQ-6yrD3LHuTg9XGZ5mJSRTtrogqTM_6uC6wJ59pk |
| project_id | 29981a2ab4184a6c9bb6275b7be17f02
                                                                                                |
| user_id    | c4c59774db88403182105adaca4ca11a
                                                                                                |
```

图 A-3　以 admin 身份请求认证令牌

A.3.7　创建 OpenStack 客户端环境脚本

前面使用环境变量和命令选项的组合来让 OpenStack 客户端与身份管理服务进行交互。为提高客户端操作的效率，OpenStack 支持使用简单的客户端环境脚本，该脚本也被称为 OpenRC 文件。这些脚本通常包含适合所有客户端的通用选项，但也支持独特的选择。

1．创建脚本

这里为 admin 和 demo 项目以及用户创建客户端环境脚本。综合实训的后续部分将参考这些脚本来加载用于客户端操作的相应凭据。客户端环境脚本文件的存放路径不受限制，可以根据需要存放到任何位置，但要确保其安全。为方便实验操作，这里将其存放在/root 目录，即 root 用户的主目录。

在控制节点上的/root 目录中创建一个名为 admin-openrc 的文件作为 admin 管理员的客户端环境脚本，并加入以下内容。

```
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_AUTH_URL=http://controller:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

接着再为 demo 用户创建一个名为 demo-openrc 的客户端环境脚本文件，并添加以下内容。

```
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=demo
export OS_USERNAME=demo
export OS_PASSWORD=DEMO_PASS
export OS_AUTH_URL=http://controller:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

2. 使用脚本

要以指定的项目和用户身份运行客户端（执行 openstack 等客户端命令），可以在运行这些命令之前简单地加载相应的客户端环境脚本。下面给出一个例子。

执行以下命令加载 admin-openrc 文件，通过身份管理服务的位置、admin 项目和用户凭据来设置环境变量，然后向身份管理服务请求一个认证令牌，如图 A-4 所示。使用 source 命令导出脚本时往往使用“.” 符号来替代该命令。

```
[root@controller ~]# . admin-openrc
[root@controller ~]# openstack token issue
+------------+----------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------+
| Field      | Value
                                                                                        |
+------------+----------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------+
| expires    | 2020-10-19T13:49:54+0000
                                                                                        |
| id         | gAAAAABfjYtyBVwQ2B5JCRwvzicTIUM66rSugAq_IH6nVbkqMXrbGzsMNxvzx9rVxMG8macnEkicIPR4EDdgDH9uXh_
Su9rtuC5niUJ1aFkOvjfhPLzMDvCWCvWHKN9J6TkPhhxNrUgTJsSSS6qOkMvA1HpyLnqze6ONY7DxjIyY4p-ZkYD76pA |
| project_id | 29981a2ab4184a6c9bb6275b7be17f02
                                                                                        |
| user_id    | c4c59774db88403182105adaca4ca11a
```

图 A-4　加载客户端环境脚本并请求认证令牌

A.4　安装和部署 Glance

为简单起见，综合实训中以文件系统作为镜像存储后端。在控制节点上部署 Glance 镜像服务，下面的操作都是在控制节点上进行的。

A.4.1　完成 Glance 的安装准备

安装和配置 Glance 镜像服务之前，必须创建数据库、服务凭据和 API 端点。

1. 创建 Glance 数据库

（1）以 root 用户身份使用数据库访问客户端连接到数据库服务器。

mysql -u root -p

（2）创建 Glance 数据库（名称为 glance）。

MariaDB [(none)]> CREATE DATABASE glance;

（3）对 Glance 数据库 glance 用户授予访问权限（这里该账户密码为 GLANCE_DBPASS）。

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \
  IDENTIFIED BY 'GLANCE_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' \
  IDENTIFIED BY 'GLANCE_DBPASS';
```

（4）退出数据库访问客户端。

2. 获取管理员凭据

后续命令行操作需要管理员身份，首先要加载 admin 的客户端环境脚本，以获得只有管理员能执行命令的访问权限。

. admin-openrc

3．创建 Glance 服务凭据

（1）创建 Glance 用户（命名为 glance，这里密码设为 GLANCE_PASS）

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

（2）将 admin 角色授予 glance 用户和 service 项目。

openstack role add --project service --user glance admin

（3）在服务目录中创建镜像服务的服务实体（名为 glance）。

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

4．创建镜像服务的 API 端点

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

openstack endpoint create --region RegionOne　 image internal http://controller:9292

openstack endpoint create --region RegionOne　image admin http://controller:9292

A.4.2　安装和配置 Glance 组件

1．安装 Glance 软件包

执行以下命令安装所需的软件包。

yum -y install openstack-glance

2．编辑/etc/glance/glance-api.conf 配置文件

（1）在[database]节中配置数据库访问。

[database]

...

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

设置这个参数的目的是让 Keystone 能知道如何连接到后端的 Glance 数据库。其中 pymysql 是一个可以操作 mysql 的 python 库。双斜杠后面的格式为：用户名:密码@mysql 服务器地址/数据库。还应注意将[database]节中的其他连接配置注释掉，或直接删除。

（2）在[keystone_authtoken]和[paste_deploy]节中配置身份管理服务访问。注意将[keystone_authtoken]节的其他选项注释掉或直接删除。

...

www_authenticate_uri　= http://controller:5000

auth_url = http://controller:5000

memcached_servers = controller:11211

auth_type = password

project_domain_name = Default

user_domain_name = Default

project_name = service

username = glance

password = GLANCE_PASS

[paste_deploy]

...

flavor = keystone

（3）在[glance_store]节中配置镜像存储。

[glance_store]

...

stores = file,http

default_store = file

filesystem_store_datadir = /var/lib/glance/images/

这里定义本地文件系统以及存储路径。

3．初始化镜像服务数据库

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

Python 的对象关系映射需要初始化来生成数据库表结构。

A.4.3 完成 Glance 服务的安装

将 Glance 镜像服务配置为开机自动启动，并启动镜像服务。

systemctl enable openstack-glance-api.service

systemctl start openstack-glance-api.service

A.4.4 验证 Glance 镜像操作

使用 CirrOS 镜像来验证镜像服务操作，测试 OpenStack 部署。

（1）执行以下命令下载 CirrOS 源镜像。

wget http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img

（2）加载 admin 用户的客户端环境脚本，获得只有管理员能执行命令的访问权限。

. admin-openrc

（3）执行以下命令，以 qcow2 磁盘格式和 bare 容器格式将镜像上传到 Glance 镜像服务，并将其设置为公共可见以让所有的项目都可以访问它。

openstack image create "cirros" --file cirros-0.4.0-x86_64-disk.img --disk-format qcow2 --container-format bare --public

（4）执行 openstack image list 命令查看当前的镜像列表，确认镜像的上传，并查看其状态。

A.5 安装和部署 Placement

Nova 计算服务需要 Placement 来支持，因此应该在 Keystone 身份管理服务安装之后、其他服务安装之前安装 Placement。在控制节点上部署放置服务，下面的操作都是在控制节点上进行的。

A.5.1　完成放置服务安装的前期准备

在安装和配置放置服务之前，必须创建数据库、服务凭据和 API 端点。

1．创建 Placement 数据库

（1）使用数据库访问客户端以 root 用户身份连接到数据库服务器。

mysql -u root -p

（2）创建名为 placement 的 Placement 数据库。

MariaDB [(none)]> CREATE DATABASE placement;

（3）授予 placement 用户对数据库的访问权限，该用户密码为 PLACEMENT_DBPASS。

MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'localhost' \
  IDENTIFIED BY 'PLACEMENT_DBPASS';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'%' \
  IDENTIFIED BY 'PLACEMENT_DBPASS';

（4）退出数据库访问客户端。

2．创建用户和端点

（1）加载 admin 用户的客户端环境脚本，获得只有管理员能执行命令的访问权限。

. admin-openrc

（2）创建名为 placement 的放置服务用户，密码使用 PLACEMENT_PASS。

openstack user create --domain default --password-prompt placement

（3）将 admin 角色授予 glance 用户和 service 项目。

openstack role add --project service --user placement admin

（4）在服务目录中创建 Placement 服务实体（名为 placement 的服务）。

openstack service create --name placement --description "Placement API" placement

（5）创建 Placement 服务端点。

openstack endpoint create --region RegionOne placement public http://controller:8778

openstack endpoint create --region RegionOne placement internal http://controller:8778

openstack endpoint create --region RegionOne placement admin http://controller:8778

根据具体环境，端点的 URL 将根据端口（可能是 8780 而不是 8778，或者根本没有端口）和主机名而有所不同，需要确定正确的 URL。

A.5.2　安装和配置放置服务组件

1．安装软件包

yum -y install openstack-placement-api

2．编辑/etc/placement/placement.conf 文件并完成以下操作

（1）在[placement_database]节中配置数据库访问。

[placement_database]

```
# ...
connection = mysql+pymysql://placement:PLACEMENT_DBPASS@controller/placement
```

（2）在[api]和[keystone_authtoken]节中配置身份管理服务访问。

```
[api]
# ...
auth_strategy = keystone

[keystone_authtoken]
# ...
auth_url = http://controller:5000/v3
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = placement
password = PLACEMENT_PASS
```

3．初始化名为 placement 的数据库

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

A.5.3　完成放置服务安装并进行验证

1．重新启动 httpd 服务

```
systemctl restart httpd
```

2．执行状态检查以确保一切正常

```
[root@controller ~]#. admin-openrc
[root@controller ~]# placement-status upgrade check
+----------------------------------+
| Upgrade Check Results                |
+----------------------------------+
| Check: Missing Root Provider IDs     |
| Result: Success                      |
| Details: None                        |
+----------------------------------+
| Check: Incomplete Consumers          |
| Result: Success                      |
| Details: None                        |
```

3．针对放置服务运行一些命令进行测试

（1）安装 osc-placement 插件

该插件需要通过 pip 工具安装，由于目前系统中未安装 pip，需要先安装 pip，再安装该插件。

[root@controller ~]# yum -y install epel-release

[root@controller ~]# yum -y install python-pip

[root@controller ~]# pip install osc-placement

（2）列出可用的资源类和特性

执行以下命令列出可用的资源类时报错。

[root@controller ~]# openstack --os-placement-api-version 1.2 resource class list --sort-column name

Expecting value: line 1 column 1 (char 0)

这是软件包缺陷所造成的，必须将以下配置添加到 /etc/httpd/conf.d/00-nova-placement-api.conf 配置文件中，让 Placement API 可以被访问。

```
<Directory /usr/bin>
    <IfVersion >= 2.4>
        Require all granted
    </IfVersion>
    <IfVersion < 2.4>
        Order allow,deny
        Allow from all
    </IfVersion>
</Directory>
```

保存该配置文件，执行 systemctl restart httpd 命令重启 HTTP 服务使上述设置生效。

再次执行以下命令列出可用的资源类时，返回的结果正常。

openstack --os-placement-api-version 1.2 resource class list --sort-column name

执行以下命令列出可用的特性时，返回的结果同样正常。

openstack --os-placement-api-version 1.6 trait list --sort-column name

至此，说明放置服务已经成功安装。

A.6　安装和部署 Nova

Nova 由多个组件和服务组成，可以部署在计算节点和控制节点这两类节点上。Nova 部署必须初始化 Cell 架构。控制节点如果不同时作为计算节点，无须安装 nova-compute，但要安装其他 Nova 组件和服务。计算服务支持多种虚拟机管理器来部署实例或虚拟机，综合实训中在计算节点上使用带 KVM 扩展的 QEMU 来支持虚拟机的硬件加速。

A.6.1　在控制节点上完成 Nova 的安装准备

安装和配置 Nova 计算服务之前，必须创建数据库、服务凭据和 API 端点。

1. 创建 Nova 数据库

（1）以 root 用户身份登录连接到数据库服务器。

```
mysql -u root -p
```

（2）分别创建名为 nova_api、nova 和 nova_cell0 的 3 个数据库。

```
MariaDB [(none)]> CREATE DATABASE nova_api;
MariaDB [(none)]> CREATE DATABASE nova;
MariaDB [(none)]> CREATE DATABASE nova_cell0;
```

（3）对上述数据库 nova 用户授予访问权限。这里将该用户的密码设为 NOVA_DBPASS。

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';
```

（4）退出数据库访问客户端。

2. 获取管理员凭据

后续命令行操作需要管理员身份，需要加载 admin 用户的客户端环境脚本。

```
. admin-openrc
```

3. 创建计算服务凭据

（1）创建 nova 用户，将其密码设为 NOVA_PASS。

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

（2）将 admin 角色授予 nova 用户和 service 项目。

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

（3）创建 Nova 的计算服务实体（名为 nova 的服务）。

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

（4）创建计算服务的 API 端点

```
openstack endpoint create --region RegionOne    compute public http://controller:8774/v2.1
openstack endpoint create --region RegionOne    compute admin http://controller:8774/v2.1
openstack endpoint create --region RegionOne    compute admin http://controller:8774/v2.1
```

A.6.2　在控制节点上安装和配置 Nova 组件

1．安装软件包

yum -y install openstack-nova-api openstack-nova-conductor openstack-nova-novncproxy openstack-nova-scheduler

2．编辑/etc/nova/nova.conf 配置文件

（1）在[DEFAULT]节中仅启用 compute 和 metadata API。

```
[DEFAULT]
# ...
enabled_apis = osapi_compute,metadata
```

（2）在[api_database]和[database]节中配置数据库访问。

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

（3）在[DEFAULT]节中配置 RabbitMQ 消息队列访问。

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller:5672/
```

（4）在[api]和[keystone_authtoken]节中配置身份管理服务访问。将[keystone_authtoken]节中的其他选项注释掉或直接删除。

```
[api]
# ...
auth_strategy = keystone
[keystone_authtoken]
# ...
www_authenticate_uri = http://controller:5000/
auth_url = http://controller:5000/
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = NOVA_PASS
```

（5）在[DEFAULT]节中使用 my_ip 选项配置控制节点的管理网络接口 IP 地址。

```
[DEFAULT]
```

```
# ...
my_ip = 10.0.0.11
```

（6）在[DEFAULT]节中启用对网络服务的支持。

```
[DEFAULT]
# ...
use_neutron = true
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

注意在默认情况下，计算服务使用自己的防火墙驱动。而网络服务也包括一个防火墙驱动，因此必须使用 nova.virt.firewall.NoopFirewallDriver 来禁用计算服务的防火墙驱动。

要使计算服务能利用网络服务，还需在[neutron]节中进行适当的配置，具体请参见 A.7 节。

（7）在[vnc]节中配置 VNC 代理使用控制节点的管理网络接口 IP 地址。

```
[vnc]
enabled = true
# ...
server_listen = $my_ip
server_proxyclient_address = $my_ip
```

（8）在[glance]节中配置镜像服务 API 的位置。

```
[glance]
# ...
api_servers = http://controller:9292
```

（9）在[oslo_concurrency]节中配置锁定路径（Lock Path）。

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

（10）在[placement]节中配置放置服务 API。

```
[placement]
# ...
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:5000/v3
username = placement
password = PLACEMENT_PASS
```

3．初始化 nova-api 数据库

```
su -s /bin/sh -c "nova-manage api_db sync" nova
```

4．注册 cell0 数据库

su -s /bin/sh -c "nova-manage cell_v2 map_cell0" nova

5．创建 cell1 单元

su -s /bin/sh -c "nova-manage cell_v2 create_cell --name=cell1 --verbose" nova

6．初始化 nova 数据库

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

7．验证 nova 的 cell0 和 cell1 已正确注册

验证结果表明正确注册，如图 A-5 所示。

```
[root@controller ~]# su -s /bin/sh -c "nova-manage cell_v2 list_cells" nova
+-------+--------------------------------------+----------------------------------------+-------------------------------------------+----------+
|  Name |                 UUID                 |              Transport URL             |               Database Connection         | Disabled |
+-------+--------------------------------------+----------------------------------------+-------------------------------------------+----------+
| cell0 | 00000000-0000-0000-0000-000000000000 |                 none:/                 | mysql+pymysql://nova:****@controller/nova_cell0 |  False   |
| cell1 | eda070ca-e035-4f63-b5dc-29ecd6b5eb83 | rabbit://openstack:****@controller:5672/ |    mysql+pymysql://nova:****@controller/nova    |  False   |
+-------+--------------------------------------+----------------------------------------+-------------------------------------------+----------+
```

图 A-5　cell0 和 cell1 已正确注册

A.6.3　在控制节点上完成 Nova 安装

执行以下命令将计算服务配置为开机自动启动，并启动计算服务。

```
systemctl enable openstack-nova-api.service openstack-nova-scheduler.service \
    openstack-nova-conductor.service openstack-nova-novncproxy.service
systemctl start openstack-nova-api.service openstack-nova-scheduler.service \
    openstack-nova-conductor.service openstack-nova-novncproxy.service
```

A.6.4　在计算节点上安装和配置 Nova 组件

1．安装软件包

yum install openstack-nova-compute

2．编辑/etc/nova/nova.conf 配置文件

（1）在[DEFAULT]节中仅启用 compute 和 metadata API。

```
[DEFAULT]
# ...
enabled_apis = osapi_compute,metadata
```

（2）在[DEFAULT]节中配置 RabbitMQ 消息队列访问。

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

（3）在[api]和[keystone_authtoken]节中配置身份认证服务访问。将[keystone_authtoken]节中的其他选项注释掉或直接删除。

```
[api]
```

```
# ...
auth_strategy = keystone

[keystone_authtoken]
# ...
www_authenticate_uri = http://controller:5000/
auth_url = http://controller:5000/
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = NOVA_PASS
```

（4）在[DEFAULT]节中使用 my_ip 参数配置计算节点的管理网络接口 IP 地址。

```
[DEFAULT]
# ...
my_ip = 10.0.0.21
```

（5）在[DEFAULT]节中启用对网络服务的支持。

```
[DEFAULT]
# ...
use_neutron = true
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

要使计算服务能够利用网络服务，还需要在[neutron]节中进行适当的配置，具体请参见 A.7 节。

（6）在[vnc]节中启用和配置远程控制台访问。

```
[vnc]
# ...
enabled = true
server_listen = 0.0.0.0
server_proxyclient_address = $my_ip
novncproxy_base_url = http://controller:6080/vnc_auto.html
```

注意此处设置与控制节点不同。服务器组件在所有的 IP 地址上侦听，而代理组件仅在计算节点上的管理网络接口 IP 地址上侦听。novncproxy_base_url 指定要使用浏览器访问该计算节点上虚拟机实例的远程控制台的 URL 地址。如果控制节点主机名不能被解析，则需要使用控制节点的 IP 地址来代替。

（7）在[glance]节中配置镜像服务 API 的位置。

```
[glance]
# ...
api_servers = http://controller:9292
```

（8）在[oslo_concurrency]节中配置锁定路径（lock path）。

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

（9）在[placement]节中配置放置服务 API。

```
[placement]
# ...
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:5000/v3
username = placement
password = PLACEMENT_PASS
```

A.6.5　在计算节点上完成 Nova 安装

1．确定计算节点是否支持虚拟机的硬件加速

```
[root@compute1 ~]# egrep -c '(vmx|svm)' /proc/cpuinfo
4
```

如果返回值等于或大于 1，则说明支持硬件加速，不必进行其他配置。

如果返回值为 0，则说明计算节点不支持硬件加速，必须配置 libvirt 使用 QEMU 而不是 KVM。具体方法是在/etc/nova/nova.conf 文件的[libvirt]节中配置如下。

```
[libvirt]
# ...
virt_type = qemu
```

2．启动计算服务及其依赖，并将其配置开机自动启动

```
systemctl enable libvirtd.service openstack-nova-compute.service
systemctl start libvirtd.service openstack-nova-compute.service
```

如果 nova-compute 服务启动失败，可检查/var/log/nova/nova-compute.log 日志文件。“AMQP server on controller:5672 is unreachable likely”这样的错误消息说明控制节点上的防火墙阻止 5672 端口的访问，这就需要开放该端口。

A.6.6　将计算节点添加到 cell 数据库

当添加新的计算节点时，必须在控制节点上运行 nova-manage cell_v2 discover_hosts 命令来注册这些新的计算节点。转到控制节点上进行下列操作。

（1）操作需要管理员身份，首先要加载 admin 凭据的环境脚本，然后确认数据库中有哪些计算节点主机。

```
[root@controller ~]# . admin-openrc
```

```
[root@controller ~]# openstack compute service list --service nova-compute
+---+------------+--------+-----+-------+----+-------------------------+
| ID  | Binary          | Host          | Zone  | Status  | State | Updated At                  |
+---+------------+--------+-----+-------+----+-------------------------+
|   6  | nova-compute | compute1 | nova  | enabled | up    | 2020-10-14T03:08:04.000000 |
```

这里发现有计算节点主机 compute1。

（2）注册计算节点主机。

```
[root@controller ~]# su -s /bin/sh -c "nova-manage cell_v2 discover_hosts --verbose" nova
```

A.6.7　验证 Nova 计算服务的安装

在控制节点上进行下列操作来验证验证 Nova 计算服务的安装。

加载 admin 凭据的环境脚本，然后列出计算服务组件以验证每个进程是否成功启动和注册。

```
[root@controller ~]# . admin-openrc
[root@controller ~]# openstack compute service list
+--+------------+--------+------+------+-----+-----------------------+
| ID | Binary          | Host          | Zone  | Status  | State | Updated At                   |
+--+------------+--------+------+------+-----+-----------------------+
| 1  | nova-conductor | controller  | internal | enabled | up    | 2020-10-14T03:15:07.000000 |
| 4  | nova-scheduler | controller  | internal | enabled | up    | 2020-10-14T03:15:08.000000 |
| 6  | nova-compute  | compute1 | nova    | enabled | up    | 2020-10-14T03:15:04.000000 |
```

再执行以下命令查看计算节点，显示计算节点 compute1 的虚拟机管理器类型为 QEMU。

```
[root@controller ~]# openstack hypervisor list
+----+--------------------+-----------------+-----------+-------+
| ID    | Hypervisor Hostname      | Hypervisor Type      | Host IP          | State     |
+----+--------------------+-----------------+-----------+-------+
|   1   | compute1                    | QEMU                 | 10.0.0.21      | up          |
```

A.7　安装和部署 Neutron

这里部署的是自服务网络，Neutron 的的 L2 代理选择是 OVS（Open vSwitch）代理，由 OVS 交换机作为网络提供者。如果规模不大，无须部署专用的网络节点，只需在控制节点和计算节点上部署所需的 Neutron 服务组件即可。这里控制节点兼作网络节点。控制节点和计算节点都要安装 Neutron 的 OVS 代理组件。

A.7.1　在控制节点上完成网络服务的安装准备

安装和配置网络服务之前，必须创建数据库、服务凭据和 API 端点。

（1）创建 Neutron 数据库

以 root 用户身份使用数据库访问客户端连接到数据库服务器。

mysql -u root -p

然后依次执行以下命令创建数据库并设置访问权限，完成之后退出数据库访问客户端。这里 neutron 账户的密码设为 NEUTRON_DBPASS。

```
MariaDB [(none)]>  CREATE DATABASE neutron;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' \
  IDENTIFIED BY 'NEUTRON_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' \
  IDENTIFIED BY 'NEUTRON_DBPASS';
```

（2）加载 admin 用户的环境脚本

```
. admin-openrc
```

（3）创建 Neutron 服务凭据

依次执行以下命令创建 neutron 用户（密码设为 NEUTRON_PASS），将 admin 角色授予该用户，并创建 Neutron 的服务实体（名为 neutron 的服务）。

```
openstack user create --domain default --password-prompt neutron
openstack role add --project service --user neutron admin
openstack service create --name neutron  --description "OpenStack Networking" network
```

（4）创建 Neutron 服务的 API 端点

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

A.7.2　在控制节点上配置网络选项

根据要部署的虚拟网络类型来配置网络选项，这里选择自服务网络。通过适当的配置，自服务网络也可以支持将虚拟机实例连接到提供者网络。

1. 安装网络组件

```
# yum -y install openstack-neutron openstack-neutron-ml2 openstack-neutron-openvswitch ebtables
```

其中包括 OVS 代理。

2. 安装 OVS

CentOS 7 中默认已安装，可检查 OVS 服务的状态。

```
[root@controller ~]# systemctl status openvswitch
```

● openvswitch.service - Open vSwitch

Loaded: loaded (/usr/lib/systemd/system/openvswitch.service; enabled; vendor preset: disabled)

Active: active (exited) since Tue 2020-10-20 15:37:50 CST; 4min 42s ago

……

3．配置 Neutron 服务器组件

编辑/etc/neutron/neutron.conf 文件，完成下列设置。

（1）在[database]节中配置数据库访问。将该节中其他 connection 选项注释掉或删除。

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

（2）在[DEFAULT]节中启用 ML2 插件、路由服务和重叠 IP 地址。

```
[DEFAULT]
# ...
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = true
```

（3）在[DEFAULT]节中配置 RabbitMQ 消息队列访问。

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

（4）在[DEFAULT]和[keystone_authtoken]节中配置身份管理服务访问。将[keystone_authtoken]节中的其他选项注释掉或直接删除。

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

[keystone_authtoken]
# ...
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = neutron
password = NEUTRON_PASS
```

（5）在[DEFAULT]和[nova]节中配置网络拓扑的变动时，网络服务能够通知计算服务。

```
[DEFAULT]
# ...
notify_nova_on_port_status_changes = true
notify_nova_on_port_data_changes = true
```

```
[nova]
# ...
auth_url = http://controller:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = nova
password = NOVA_PASS
```

（6）在[oslo_concurrency]节中配置锁定路径。

```
[oslo_concurrency]
# ...
lock_path = /var/lib/neutron/tmp
```

4．配置 ML2 插件

编辑/etc/neutron/plugins/ml2/ml2_conf.ini 文件，完成下列操作。

（1）在[ml2]节中启用 flat、VLAN 和 VXLAN 网络。

```
[ml2]
# ...
type_drivers = flat,vlan,vxlan
```

（2）在[ml2]节中启用 VXLAN 自服务网络。

```
[ml2]
# ...
tenant_network_types = vxlan
```

（3）在[ml2]节中启用 Open vSwitch 代理和 L2 population 机制。

```
[ml2]
# ...
mechanism_drivers = openvswitch,l2population
```

配置 ML2 插件之后，删除 type_drivers 选项中的值会导致数据库不一致。

（4）在[ml2]节中启用端口安全扩展驱动。

```
[ml2]
# ...
extension_drivers = port_security
```

（5）在[ml2_type_flat]节中列出可以创建 Flat 类型提供者网络的物理网络名称。

```
[ml2_type_flat]
# ...
flat_networks = *
```

这里的物理网络名称是 Flat 网络的标记，在创建 Flat 类型提供者网络时需要指定它。默认的“*”值表示可以允许使用任意字符串的物理网络名称。如果使用空值则禁用 Flat

类型网络。

（6）在[ml2_type_vxlan]节中配置自服务网络的 VXLAN 网络 ID 范围。

```
[ml2_type_vxlan]
# ...
vni_ranges = 10:1000
```

（7）在[securitygroup]节中，启用 ipset 以提高安全组规则的效率。

```
[securitygroup]
# ...
enable_ipset = true
```

5. 配置 OVS 代理

（1）创建 OVS 提供者网桥

首先创建一个 OVS 提供者网桥（外部网桥），并将提供者网络的网卡添加到该网桥的一个端口。可以使用 ovs-vsctl add-br 和 ovs-vsctl add-port 命令来实现，但这种方式实现的配置在开机后会丢失，这里改用网卡配置文件来实现。这里将 OVS 外部网桥命名为 br-ex。将提供者网络的网卡 ens37 配置文件/etc/sysconfig/network-scripts/ifcfg-ens37 的配置内容更改如下。

```
NAME=ens37
DEVICE=ens37
TYPE=OVSPort
DEVICETYPE=ovs
OVS_BRIDGE=br-ex
ONBOOT=yes
```

再创建/etc/sysconfig/network-scripts/ifcfg-br-ex 网卡配置文件用于 OVS 外部网桥（网桥名称为 br-ex），该文件的配置内容如下。

```
NAME=br-ex
DEVICE=br-ex
DEVICETYPE=ovs
TYPE=OVSBridge
BOOTPROTO=static
IPADDR=192.168.199.51
PREFIX=24
GATEWAY=192.168.199.1
DNS1=114.114.114.114
ONBOOT=yes
```

执行 systemctl restart network 命令重启 network 服务使上述修改生效，然后验证网桥的设置。执行以下命令列出当前 OVS 网桥列表，可发现已经创建了 br-ex 网桥。

```
[root@controller ~]# ovs-vsctl list-br
br-ex
br-int
```

br-tun

再执行以下命令列出 br-ex 网桥端口列表，可发现该网桥已有 ens37 端口。

```
[root@controller ~]# ovs-vsctl list-ports br-ex
ens37
phy-br-ex
[root@controller ~]#
```

（2）配置 OVS 代理

接下来配置 OVS 代理，主要是编辑/etc/neutron/plugins/ml2/openvswitch_agent.ini 文件，主要配置如下。

```
[ovs]
bridge_mappings = extnet:br-ex
local_ip = 10.0.0.11

[agent]
tunnel_types = vxlan
l2_population = True

[securitygroup]
firewall_driver = iptables_hybrid
```

其中 bridge_mappings 选项定义物理网络名称到本节点上的 OVS 网桥名称的映射，主要用于 Flat 或 VLAN 类型的物理网络。可以以列表方式定义多个映射，OVS 网桥名称不能超过 11 个字符，物理网络名称应在 ml2_conf.ini 文件中列出的一致，比如 flat_networks 选项的值。每个网桥必须存在，且应有一个配置为端口的物理网络接口。配置的所有物理网络应当映射到每个代理节点上相应的网桥。注意，如果从此处映射中删除一个 OVS 网桥，确保从 OVS 集成网桥断开该网桥的连接，因为该网桥不再由代理管理。

这里的 local_ip 选项定义本地覆盖网络端点的 IP 地址，本例中为当前节点管理网络接口 IP 地址。各个节点上都要定义。

tunnel_types 选项指定隧道类型。

firewall_driver 选项用于安全组防火墙。

6．配置 L3 代理

L3 代理为自服务虚拟网络提供路由和 NAT 服务。编辑/etc/neutron/l3_agent.ini 文件，在 [DEFAULT]节中配置 OVS 接口驱动。

```
[DEFAULT]
# ...
interface_driver = openvswitch
```

7．配置 DHCP 代理

DHCP 代理为虚拟网络提供 DHCP 服务。编辑/etc/neutron/dhcp_agent.ini 文件，在 [DEFAULT]节中配置 OVS 接口驱动，启用元数据隔离（让提供者网络上的实例能够通过

网络访问元数据）。

```
[DEFAULT]
interface_driver = openvswitch
enable_isolated_metadata = True
force_metadata = True
```

A.7.3 在控制节点上配置元数据代理

元数据代理为实例提供像安全凭据这样的配置信息。编辑/etc/neutron/metadata_agent.ini 文件，在[DEFAULT]节中配置元数据主机和共享密码（可根据需要替换 METADATA_SECRET 密码）。

```
[DEFAULT]
# ...
nova_metadata_host = controller
metadata_proxy_shared_secret = METADATA_SECRET
```

A.7.4 在控制节点上配置计算服务使用网络服务

编辑/etc/nova/nova.conf 文件，在[neutron]节中设置访问参数，启用元数据代理，并配置密码。

```
[neutron]
# ...
auth_url = http://controller:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
service_metadata_proxy = true
metadata_proxy_shared_secret = METADATA_SECRET
```

A.7.5 在控制节点上完成网络服务安装

（1）网络服务初始化脚本需要一个指向 ML2 插件配置文件/etc/neutron/plugins/ml2/ml2_conf.ini 的符号连接/etc/neutron/plugin.ini。如果该符号连接未创建，执行以下命令创建。

```
ln -s /etc/neutron/plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
```

（2）初始化数据库。

```
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
```

（3）重启计算 API 服务。

```
systemctl restart openstack-nova-api.service
```

（4）启动网络服务并将其配置为开机自动启动。

```
systemctl enable neutron-server.service \
  neutron-openvswitch-agent.service neutron-dhcp-agent.service \
  neutron-metadata-agent.service neutron-l3-agent.service
systemctl start neutron-server.service \
  neutron-openvswitch-agent.service neutron-dhcp-agent.service \
  neutron-metadata-agent.service neutron-l3-agent.service
```

A.7.6　在计算节点上安装 Neutron 组件

计算节点负责实例的连接和安全组，需要安装 Neutron 的 OVS 代理组件。执行以下命令安装所需的软件包。

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

A.7.7　在计算节点上配置网络通用组件

网络通用组件配置包括认证机制、消息队列和插件。编辑/etc/neutron/neutron.conf 配置文件，设置以下选项。

（1）在[database]节中将连接设置语句注释掉，因为计算节点不访问 neutron 数据库。

（2）在[DEFAULT]节中配置 RabbitMQ 消息队列访问。

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

（3）在[DEFAULT]和[keystone_authtoken]节中配置身份管理服务访问。将[keystone_authtoken]节中的其他选项注释掉或直接删除。

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

[keystone_authtoken]
# ...
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = neutron
password = NEUTRON_PASS
```

（4）在[oslo_concurrency]节中配置锁定路径。

```
[oslo_concurrency]
# ...
lock_path = /var/lib/neutron/tmp
```

A.7.8 在计算节点上配置网络选项

选择与控制节点相同的网络选项，即自服务网络。这里计算节点仅安装有 OVS 代理，通过编辑/etc/neutron/plugins/ml2/openvswitch_agent.ini 文件进行配置，具体配置如下。

```
[ovs]
local_ip = 10.0.0.21

[agent]
tunnel_types = vxlan
l2_population = True

[securitygroup]
firewall_driver = iptables_hybrid
```

这里 local_ip 选项设置为计算节点管理网络接口 IP 地址。在[vxlan]节中启用 VXLAN 覆盖网络和 L2 population。在[securitygroup]节中配置安全组混合 iptables 防火墙驱动。

A.7.9 在计算节点上配置计算服务使用网络服务

编辑/etc/nova/nova.conf file 文件，在[neutron]节中设置访问选项。

```
[neutron]
# ...
auth_url = http://controller:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
```

其中 password 选项设置身份管理服务中的 neutron 用户的密码。

A.7.10 在计算节点上完成网络服务安装

执行以下命令重启计算服务。

```
systemctl restart openstack-nova-compute.service
```

执行以下命令启动 OVS 代理服务并将其配置为开机自动启动。

```
systemctl enable neutron-openvswitch-agent.service
systemctl start neutron-openvswitch-agent.service
```

A.7.11　验证网络服务运行

在控制节点上加载 admin 用户的环境脚本。

source admin-openrc

查看网络代理列表，结果如图 A-6 所示，表明控制节点和计算节点上的网络代理组件正常运行。

```
[root@controller ~]# openstack network agent list
+--------------------------------------+--------------------+------------+-------------------+-------+-------+---------------------------+
| ID                                   | Agent Type         | Host       | Availability Zone | Alive | State | Binary                    |
+--------------------------------------+--------------------+------------+-------------------+-------+-------+---------------------------+
| 30a61436-6ef7-4a71-a1d3-0c4871d20225 | Metadata agent     | controller | None              | :-)   | UP    | neutron-metadata-agent    |
| b0381c0d-af7d-4748-a01f-11a5a7977b87 | DHCP agent         | controller | nova              | :-)   | UP    | neutron-dhcp-agent        |
| d78d7f6f-6775-43b5-955b-03119817151e | Open vSwitch agent | compute1   | None              | :-)   | UP    | neutron-openvswitch-agent |
| fbf01a1e-7a3e-49d7-813c-cd405dfb9262 | L3 agent           | controller | nova              | :-)   | UP    | neutron-l3-agent          |
| fd5d3dbd-13c3-4c19-ab85-2e57b5a54d5e | Open vSwitch agent | controller | None              | :-)   | UP    | neutron-openvswitch-agent |
```

图 A-6　网络代理列表

A.7.12　创建初始网络

当前配置支持多个 VXLAN 自服务网络。为简化实验，这里创建一个 Flat 类型的提供者网络、一个自服务网络和一个路由器，路由器的外部网关位于提供者网络。该路由器使用 NAT 来转发 IPv4 网络流量。在控制节点上执行下列操作。

（1）加载 admin 用户的环境脚本。

source admin-openrc

（2）创建一个名为 public1 的提供者网络。

openstack network create --project admin --share --external \

--availability-zone-hint nova --provider-physical-network extnet \

--provider-network-type flat public1

其中--share 选项表示任何项目都可使用该网络，--external 选项表示是外部网络，--provider-physical-network 选项指定物理网络，--provider-network-type 选项指定物理网络类型。

（3）在上述提供者网络基础上创建一个名为 public1_subnet 的 IPv4 子网。

openstack subnet create --network public1 \

  --allocation-pool start=192.168.199.61,end=192.168.199.90 \

  --dns-nameserver 114.114.114.114 --gateway 192.168.199.1 \

  --subnet-range 192.168.199.0/24 public1_subnet

（4）加载普通用户 demo 的环境脚本。

source demo-openrc

（5）执行以下命令创建一个名为 private1 自服务网络。

openstack network create private1

（6）基于该自服务网络创建一个名为 private1_subnet 的 IPv4 子网。

openstack subnet create --subnet-range 172.16.1.0/24 \

--network private 1 --dns-nameserver 114.114.114.114　private1_subnet

（7）创建一个名为 router1 的路由器。

openstack router create router1

（8）添加 private1_subnet 子网作为该路由器的接口。

openstack router add subnet router1 private1_subnet

（9）添加上述提供者网络作该路由器的网关。

openstack router set --external-gateway public1 router1

A.7.13 验证网络操作

（1）在网络节点（由控制节点充当）上验证 qrouter 名称空间的创建。

[root@controller ~]# ip netns

qrouter-2f05c572-6bf1-40bc-b0d1-2c01afc65560 (id: 1)

qdhcp-1713f277-843b-4653-b9f5-91d69d602b72 (id: 0)

（2）加载 admin 的环境脚本，然后创建一个实例类型。

[root@controller ~]# source demo-openrc

[root@controller ~]# openstack flavor create --public m1.tiny --id 1 --ram 512 --disk 1 --vcpus 1 --rxtx-factor 1

（3）加载普通用户 demo 的环境脚本。

source demo-openrc

（4）创建安全组规则以允许通过网络 ping 和 SSH 访问虚拟机实例。

[root@controller ~]# openstack security group rule create --proto icmp default

[root@controller ~]# openstack security group rule create --proto tcp --dst-port 22 default

（5）基于上述自服务网络创建一个虚拟机实例，使用的是 CirrOS 镜像，选择的是刚创建实例类型。

[root@controller ~]# openstack server create --flavor 1 --image cirros --network private1 testVM1

（6）执行 openstack server list 命令查看实例列表，可以发现该实例的 IP 地址来自 private1 自服务网络。

（7）为虚拟机实例分配浮动 IP 地址以解决外部网络访问问题。

在提供者网络中创建一个浮动 IP 地址。

[root@controller ~]# openstack floating ip create public1

```
+-------------------+--------------------+
| Field                                  | Value                                  |
+-------------------+--------------------+
| created_at                        | 2020-10-17T12:12:07Z   |
| description                        |                                        |
| dns_domain                        | None                                |
| dns_name                          | None                                 |
| fixed_ip_address              | None                                 |
| floating_ip_address         | 192.168.199.63                |
```

这里创建的浮动 IP 地址为 192.168.199.63。

为该实例分配该 IP 地址。

[root@controller ~]# openstack server add floating ip testVM1 192.168.199.63

在控制节点上 ping 该实例的浮动 IP，测试到该实例的通信，结果正常。

```
[root@controller ~]# ping -c 2 192.168.199.63
PING 192.168.199.63 (192.168.199.63) 56(84) bytes of data.
64 bytes from 192.168.199.63: icmp_seq=1 ttl=63 time=2.56 ms
64 bytes from 192.168.199.63: icmp_seq=2 ttl=63 time=1.32 ms
```

（8）根据需要登录到该实例，测试到 Internet 或外部网络的通信。

A.8　安装和部署 Horizon

在控制节点上安装 Horizon。确认使用 Apache HTTP 服务器和 Memcached 服务的 Keystone 身份管理服务已经安装和配置好，并在正常运行。

A.8.1　安装和配置 Horizon 组件

1. 安装软件包

```
yum -y install openstack-dashboard
```

2. 编辑/etc/openstack-dashboard/local_settings 配置文件

（1）配置仪表板使用控制节点上的 OpenStack 服务。

```
OPENSTACK_HOST = "controller"
```

（2）设置允许访问 Dashboard 的主机。

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

这里使用通配符“*”来代表所有的主机。

（3）配置 Memcached 会话存储服务。注意将除以下选项的其他会话存储配置选项全部注释掉。

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

（4）启用 Identity API v3 支持，此为默认设置。

```
OPENSTACK_KEYSTONE_URL = "http://%s:5000/v3" % OPENSTACK_HOST
```

（5）启用对多个域的支持。

```
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
```

（6）配置 API 版本。

```
OPENSTACK_API_VERSIONS = {
    "identity": 3,
    "image": 2,
```

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

（7）配置通过仪表板创建的用户的默认域。

OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "Default"

（8）配置过仪表板创建的用户的默认角色。

OPENSTACK_KEYSTONE_DEFAULT_ROLE = "reader"

（9）根据需要配置时区，这里将时区改为上海。

TIME_ZONE = "Asia/Shanghai"

（10）添加以下设置。

WEBROOT 定义访问仪表板访问路径（相对于 Web 服务器根目录），注意路径末尾要加斜杠

WEBROOT = '/dashboard/'

以下两个选项分别定义登录和退出登录（注销）的路径

LOGIN_URL = '/dashboard/auth/login/'

LOGOUT_URL = '/dashboard/auth/logout/'

LOGIN_REDIRECT_URL 选项定义登录重定向路径

LOGIN_REDIRECT_URL = '/dashboard/'

3．调整其他配置

如果/etc/httpd/conf.d/openstack-dashboard.conf 文件中没有包含以下定义，将该定义语句添加到该文件中。

WSGIApplicationGroup %{GLOBAL}

A.8.2 完成 Horizon 安装并进行验证

重启 Web 服务和会话存储服务。

systemctl restart httpd.service memcached.service

在控制节点上通过浏览器访问 http://controller/dashboard 网址，出现正常的登录界面，输入 admin 账户和密码，如图 A-7 所示。

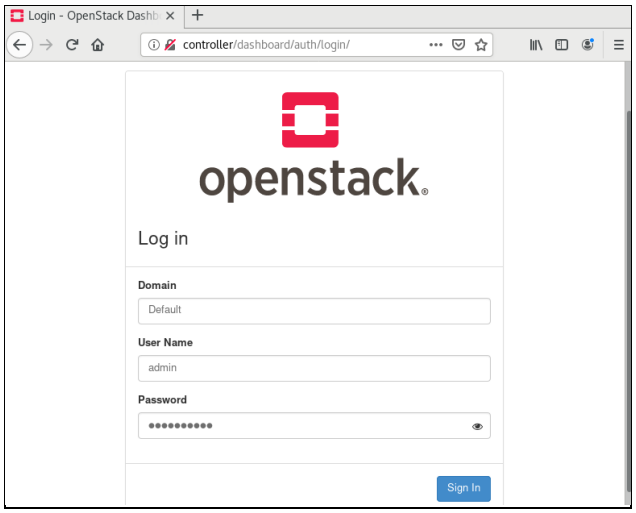

图 A-7　仪表板登录界面

A.9　安装和部署 Cinder

块存储 API（cinder-api）和调度服务（cinder-scheduler）通常部署在控制节点上。根据所用的存储驱动，卷服务（cinder-volume）可以部署在控制节点、计算节点或者专门的存储节点上。综合实训的双节点系统中由计算节点兼作存储节点，为实例提供卷服务。

A.9.1　在控制节点上完成 Cinder 的安装准备

安装和配置块存储服务之前，必须创建数据库、服务凭据和 API 端点。

（1）创建 Cinder 数据库

使用数据库访问客户端，以 root 用户身份连接到数据库服务器。

```
mysql -u root -p
```

依次执行以下命令创建数据库并设置访问权限，完成之后退出数据库访问客户端。这里 cinder 用户的密码设为 CINDER_DBPASS。

```
MariaDB [(none)]> CREATE DATABASE cinder;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' \
  IDENTIFIED BY 'CINDER_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' \
  IDENTIFIED BY 'CINDER_DBPASS';
MariaDB [(none)]> exit;
```

（2）加载 admin 的环境脚本。后续命令行操作需要云管理员身份。

```
. admin-openrc
```

（3）创建 cinder 服务凭据。依次执行以下命令创建 cinder 用户，将 admin 角色授予该用户，并创建 cinderv2 和 cinderv3 的服务实体。

```
openstack user create --domain default --password-prompt cinder
openstack role add --project service --user cinder admin
openstack service create --name cinderv2  --description "OpenStack Block Storage" volumev2
openstack service create --name cinderv3  --description "OpenStack Block Storage" volumev3
```

（4）创建块存储服务的 API 端点（应为每个服务实体创建端点）。

```
openstack endpoint create --region RegionOne
  volumev2 public http://controller:8776/v2/%\(project_id\)s
openstack endpoint create --region RegionOne \
  volumev2 internal http://controller:8776/v2/%\(project_id\)s
openstack endpoint create --region RegionOne \
  volumev2 admin http://controller:8776/v2/%\(project_id\)s
openstack endpoint create --region RegionOne \
  volumev3 public http://controller:8776/v3/%\(project_id\)s
openstack endpoint create --region RegionOne \
  volumev3 internal http://controller:8776/v3/%\(project_id\)s
```

```
openstack endpoint create --region RegionOne \
  volumev3 admin http://controller:8776/v3/%\(project_id\)s
```

A.9.2 在控制节点上安装和配置 Cinder 组件

1. 安装软件包

```
yum -y install openstack-cinder
```

2. 编辑/etc/cinder/cinder.conf 文件以完成相关设置

（1）在[database]节中配置数据库访问。

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

（2）在[DEFAULT]节中配置 RabbitMQ 消息队列访问。

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

（3）在[DEFAULT]和[keystone_authtoken]节中配置身份管理服务访问。注意将[keystone_authtoken]节中的其他选项注释掉或直接删除。

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

[keystone_authtoken]
# ...
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = cinder
password = CINDER_PASS
```

（4）在[DEFAULT]节中配置 my_ip 选项（其值为控制节点上管理网络接口的 IP 地址）。

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

（5）在[oslo_concurrency]节中配置锁定路径。

```
[oslo_concurrency]
```

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

3．初始化块存储数据库

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

A.9.3　在控制节点上配置计算服务使用块存储服务

编辑/etc/nova/nova.conf 配置文件，在[cinder]节中添加以下设置：

```
[cinder]
os_region_name = RegionOne
```

A.9.4　在控制节点上完成 Cinder 安装

（1）重启计算 API 服务

```
systemctl restart openstack-nova-api.service
```

（2）启动块存储服务并将其配置为开机自动启动

```
systemctl enable openstack-cinder-api.service openstack-cinder-scheduler.service
systemctl start openstack-cinder-api.service openstack-cinder-scheduler.service
```

A.9.5　在存储节点上完成 Cinder 的安装准备

安装和配置块存储服务之前，必须准备好存储设备。

（1）在存储节点主机上增加一块硬盘

为便于实验，本例通过为 CentOS 7 虚拟机增加一块硬盘。可查看当前的磁盘设备列表。

```
[root@compute1 ~]# fdisk -l
Disk /dev/sda: 214.7 GB, 214748364800 bytes, 419430400 sectors
……
   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048     2099199     1048576   83  Linux
/dev/sda2         2099200   419430399   208665600   8e  Linux LVM
```

（2）安装支持工具包

CentOS 7 操作系统默认已安装 LVM 包。如果没有安装，执行以下命令安装 LVM 包，启动 LVM 元数据服务并将其配置为开机自动启动。

```
yum -y install lvm2 device-mapper-persistent-data
systemctl enable lvm2-lvmetad.service
systemctl start lvm2-lvmetad.service
```

（3）创建 LVM 物理卷/dev/sdb

```
[root@compute1 ~]# pvcreate /dev/sdb
  Physical volume "/dev/sdb" successfully created.
```

（4）基于该物理卷创建 LVM 卷组 cinder-volumes

```
[root@compute1 ~]# vgcreate cinder-volumes /dev/sdb
  Volume group "cinder-volumes" successfully created
```

vgcreate cinder-volumes /dev/sdb

Cinder 块存储服务在这个卷组中创建逻辑卷。

（5）编辑/etc/lvm/lvm.conf 文件

在“devices”节中添加一个过滤器来接受/dev/sdb 设备并拒绝所有其他设备。

devices {

...

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

只有实例能够访问块存储卷。但是，底层的操作系统管理与卷关联的设备。默认情况下，LVM 卷扫描工具扫描/dev 目录以获取那些包括卷的块存储设备。如果项目在其卷上使用 LVM，扫描工具探测这些卷并试图缓存它们，这会导致底层操作系统和项目卷的多种问题。因此，必须重新配置 LVM 使得仅扫描包括卷组的设备。

过滤器中的每项以 a 表示接受，以 r 开头表示拒绝，设备名使用正则表达式。数组必须以 r/.*/结尾表示拒绝任何其余的设备。本例中/dev/sda 设备包含操作系统，必须将该设备也加入到过滤器中。

A.9.6 在存储节点上安装和配置 Cinder 组件

首先安装软件包。

yum -y install openstack-cinder targetcli python-keystone

然后编辑/etc/cinder/cinder.conf 文件并完成相应设置

（1）在[database]节中配置数据库访问。

[database]

...

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

（2）在[DEFAULT]节中配置 RabbitMQ 消息队列访问。

[DEFAULT]

...

transport_url = rabbit://openstack:RABBIT_PASS@controller

（3）在[DEFAULT]和[keystone_authtoken]节中配置身份管理服务访问。注意将[keystone_authtoken]节中的其他选项注释掉或直接删除。

[DEFAULT]

...

auth_strategy = keystone

[keystone_authtoken]

...

www_authenticate_uri = http://controller:5000

auth_url = http://controller:5000

memcached_servers = controller:11211

auth_type = password

project_domain_name = default

user_domain_name = default

project_name = service

username = cinder

password = CINDER_PASS

（4）在[DEFAULT]节中配置 my_ip 选项（其值为存储节点上管理网络接口的 IP 地址）。

[DEFAULT]

...

my_ip = 10.0.0.21

（5）在[lvm]节中配置 LVM 后端，包括 LVM 驱动、cinder-volumes 卷组、iSCSI 协议和适当的 iSCSI 服务。如果[lvm]节不存在，需要添加该节。

[lvm]

volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver

volume_group = cinder-volumes

target_protocol = iscsi

target_helper = lioadm

（6）在[DEFAULT]节中启用 LVM 后端。

[DEFAULT]

...

enabled_backends = lvm

后端可随意命名，此例中使用驱动名称作为后端的名称。

（7）在[DEFAULT]节中配置镜像服务 API 的位置。

[DEFAULT]

...

glance_api_servers = http://controller:9292

（8）在[oslo_concurrency]节中配置锁定路径。

[oslo_concurrency]

...

lock_path = /var/lib/cinder/tmp

A.9.7　在存储节点上完成 Cinder 安装

启动块存储卷服务及其依赖组件，并配置它们开机自动启动。

systemctl enable openstack-cinder-volume.service target.service

systemctl start openstack-cinder-volume.service target.service

A.9.8　验证 Cinder 服务操作

（1）检查 Cinder 服务运行情况

在控制节点上加载 admin 凭据，以获取管理员权限。

. admin-openrc

执行以下命令列出 Cinder 块存储服务组件。

[root@controller ~]# openstack volume service list

```
+--------------+------------+-----+-------+-----+------------------------+
| Binary       | Host       | Zone | Status | State | Updated At             |
+--------------+------------+-----+-------+-----+------------------------+
| cinder-scheduler | controller | nova | enabled | up | 2020-10-18T02:49:09.000000 |
| cinder-volume | compute1@lvm | nova | enabled | up | 2020-10-18T02:49:08.000000 |
```

结果表明 Cinder 块存储服务正常运行，控制节点上运行 cinder-scheduler 子服务，存储节点（这里由计算节点主机兼任）上运行 cinder-volume 子服务。

（2）创建卷进行要测试

在控制节点上创建一个卷，并指定可用域、卷大小。

```
[root@controller ~]# openstack volume create  --size 5 --availability-zone nova testVol
```

创建完毕，查看卷列表，可发现该卷已成功创建。

```
[root@controller ~]# openstack volume list
+--------------------------------------+-------+-------+----+-------------+
| ID                                   | Name  | Status | Size | Attached to |
+--------------------------------------+-------+-------+----+-------------+
| e1c25033-c3d4-49e2-8e7d-bc19ab4e768b | testVol | available |  5 |             |
```