OpenStack（计算+网络）

VirtualBox CentOS7.5 + OpenStack Rocky (All-in-One模式)

官方: https://docs.openstack.org/install-guide/

1. 控制节点的系统环境准备（4章）

1.0.系统环境

1）系统选择版本 CentOS Linux release 7.5

2）控制节点、计算节点 Controller： 192.168.56.126

1.1.配置域名解析 (4.4)

1）配置主机名 控制节点和计算节点配置相同，且都需要配置

设置主机名

```
hostnamectl set-hostname controller
```

添加主机映射

```
cat << EOF >> /etc/hosts
192.168.56.126 controller
EOF
```

配置主机名

1.2.关闭防火墙和selinux(4.3)

1）关闭iptables防火墙

```
systemctl stop firewalld.service
systemctl disable firewalld.service
systemctl status firewalld.service
```

2）关闭 selinux

```
setenforce 0
getenforce
sed -i 's#SELINUX=enforcing#SELINUX=disabled#g' /etc/sysconfig/selinux
grep SELINUX=disabled /etc/sysconfig/selinux
```

1.3.配置时间同步

1）在控制端配置时间同步服务

```
yum install chrony -y
```

2）编辑配置文件确认有以下配置

```
vi /etc/chrony.conf
--------------------------------
server ntp1.aliyun.com iburst
server ntp2.aliyun.com iburst
allow 192.168.0.0/16
```

3）重启ntp服务，并配置开机自启动

```
systemctl restart chronyd.service
systemctl status chronyd.service
systemctl enable chronyd.service
systemctl list-unit-files |grep chronyd.service
```

4）设置时区，同步时间

```
timedatectl set-timezone Asia/Shanghai
chronyc sources
timedatectl status
```

配置完成，进行同步测试

```
chronyc sources
例：
[root@controller ~]# chronyc sources
```

1.4.配置yum源（略）(4.7)

配置OpenStack的阿里云yum（略）

5）安装openstack客户端相关软件

```
yum install python-openstackclient openstack-selinux -y
```

1.5.安装数据库 (4.12)

可以修改系统内核更改最大连接数和文件句柄数

1.修改文件句柄数量

```
ulimit -SHn 65536
```

vi /etc/security/limits.conf 添加如下内容

```
* hard nofile 65536
* soft nofile 65536
```

为避免内存不足，增加交换空间

```
mkdir /data
dd if=/dev/zero of=/data/swap bs=1024 count=4096000
mkswap /data/swap
swapon /data/swap
```

```
vi /etc/fstab
-----------------添加如下内容-----------------
/data/swap     swap     swap    defaults   0 0
```

2.安装mariadb相关软件包 1）CentOS7.5默认数据库为maraidb

```
yum install mariadb mariadb-server MySQL-python python2-PyMySQL -y
```

2）创建openstack的数据库配置文件

```
vi /etc/my.cnf.d/mariadb_openstack.cnf
```

mysqld添加以下配置

```
[mysqld]
bind-address = 0.0.0.0
default-storage-engine = innodb
innodb_file_per_table = on
max_connections = 4096
collation-server = utf8_general_ci
character-set-server = utf8
init-connect = 'SET NAMES utf8'
```

3）启动数据库设置开机启动

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

4）初始化数据库并重新启动 设置密码，默认密码为空，然后输入密码123456，一路y回车

```
/usr/bin/mysql_secure_installation
```

```
systemctl restart mariadb.service
```

5）创建openstack相关数据库并授权 测试数据库

```
mysql -u root -p123456

flush privileges;
show databases;
select user,host from mysql.user;
exit
```

1.6.安装消息队列RABBITMQ (4.13)

1）安装rabbitmq-server

```
yum install rabbitmq-server -y
```

2）启动rabbitmq，并配置自启动

```
systemctl start  rabbitmq-server.service
systemctl status rabbitmq-server.service

systemctl enable rabbitmq-server.service
systemctl list-unit-files |grep rabbitmq-server.service
```

3）创建消息队列中openstack账号及密码（设置不成功，请重启系统）

```
rabbitmqctl add_user openstack openstack
rabbitmqctl set_permissions openstack ".*" ".*" ".*"
```

rabbitmq配置完毕

1.7.在控制节点上安装Memcached (4.14)

认证服务认证缓存使用Memcached缓存令牌。缓存服务memecached运行在控制节点。 在生产部署中，推荐联合启用防火墙、认证和加密保证它的安全。

1）安装Memcached用于缓存令牌

```
yum install memcached python-memcached -y
```

2）修改memcached配置文件

```
vi /etc/sysconfig/memcached
OPTIONS="-l 127.0.0.1,controller"
```

3）启动memcached并设置开机自启动

```
systemctl start memcached.service
systemctl status memcached.service
netstat -anptl | grep memcached

systemctl enable memcached.service
systemctl list-unit-files |grep memcached.service
```

1.8.在控制节点上安装Etcd服务

这个Etcd服务是新加入的，用于自动化配置

1）安装etcd服务

```
yum install etcd -y
```

2）修改etcd配置文件

```
vi /etc/etcd/etcd.conf
--------------------------------------------------
#[Member]
ETCD_DATA_DIR="/var/lib/etcd/default.etcd"
ETCD_LISTEN_PEER_URLS="http://192.168.56.126:2380"
ETCD_LISTEN_CLIENT_URLS="http://192.168.56.126:2379"
ETCD_NAME="controller"

#[Clustering]
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://192.168.56.126:2380"
ETCD_ADVERTISE_CLIENT_URLS="http://192.168.56.126:2379"
ETCD_INITIAL_CLUSTER="controller=http://192.168.56.126:2380"
ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster-01"
ETCD_INITIAL_CLUSTER_STATE="new"
```

3）启动etcd并设置开机自启动

```
systemctl start etcd.service
systemctl status etcd.service
netstat -anptl|grep etcd

systemctl enable etcd.service
systemctl list-unit-files |grep etcd.service
```

控制节点controller就完成基础环境的配置

2. 安装Keyston认证组件（5章）

2.1.在控制节点创建keystone相关数据库 (5.2)

1）创建keystone数据库并授权

```
mysql -p123456

CREATE DATABASE keystone;
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY 'keystone';
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY 'keystone';
flush privileges;
show databases;
select user,host from mysql.user;
exit
```

2.2.在控制节点安装keystone相关软件包 (5.3)

1）安装keystone相关软件包

配置Apache服务，使用带有“mod_wsgi”的HTTP服务器来相应认证服务请求

```
yum clean all

yum install openstack-keystone httpd mod_wsgi -y
yum install openstack-keystone python-keystoneclient openstack-utils -y
```

2）快速修改keystone配置 需要安装Openstack-utils

```
openstack-config --set /etc/keystone/keystone.conf database connection
mysql+pymysql://keystone:keystone@controller/keystone
openstack-config --set /etc/keystone/keystone.conf token provider fernet
```

查看生效的配置

```
egrep -v "^#|^$" /etc/keystone/keystone.conf
其他方式查看生效配置
grep '^[a-z]' /etc/keystone/keystone.conf
例：
[root@openstack01 tools]# grep '^[a-z]' /etc/keystone/keystone.conf
```

keystone不需要启动，通过http服务进行调用

2.3.初始化同步keystone数据库

1）同步keystone数据库（44张）

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

2）同步完成进行连接测试 保证所有需要的表已经建立，否则后面可能无法进行下去

```
mysql -h192.168.56.126 -ukeystone -pkeystone -e "use keystone;show tables;"
```

例：

```
[root@openstack01 ~]# mysql -h192.168.56.126 -ukeystone -pkeystone -e "use keystone;show tables;"
[root@openstack01 ~]# mysql -h192.168.56.126 -ukeystone -pkeystone -e "use keystone;show tables;" | wc -l
```

2.4.初始化Fernet令牌库

以下命令无返回

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

2.5.配置启动Apache（httpd）

1）修改httpd主配置文件

```
vi /etc/httpd/conf/httpd.conf
```

修改如下：

```
ServerName controller
```

2）配置虚拟主机 创建keystone虚拟主机配置文件的快捷方式，也可直接复制cp

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

3）启动httpd并配置开机自启动

```
systemctl start httpd.service
systemctl status httpd.service
netstat -anptl|grep httpd

systemctl enable httpd.service
systemctl list-unit-files |grep httpd.service
```

若httpd启动失败，用以下方式重试：

```
yum remove apr httpd mod_wsgi
yum install apr httpd mod_wsgi
systemctl restart httpd.service
```

例：

```
[root@openstack01 ~]# systemctl start httpd.service
[root@openstack01 ~]# systemctl status httpd.service
[root@openstack01 ~]# netstat -anptl|grep httpd
[root@openstack01 ~]# systemctl enable httpd.service
[root@openstack01 ~]# systemctl list-unit-files |grep httpd.service
```

http服务配置完成

2.4.初始化keystone认证服务 (5.4)

1）创建 keystone 用户，初始化的服务实体和API端点

需要创建密码ADMIN__PASS作为openstack的管理员用户密码，这里创建为123456

```
keystone-manage bootstrap --bootstrap-password 123456 \
  --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
```

该命令会在keystone数据库执行如下操作： 1）在endpoint表增加3个服务实体的API端点 2）在local__user表中创建admin用户 3）在project表中创建admin和Default项目（默认域） 4）在role表创建3种角色，admin，member和reader 5）在service表中创建identity服务

export OS__PASSWORD要使用上面配置的ADMIN__PASS密码

编写环境变量

```
vi admin-openrc
-------------------------------------------
export OS_USERNAME=admin
export OS_PASSWORD=123456
export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_AUTH_URL=http://controller:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

查看声明的变量

```
source admin-openrc
env |grep OS_
```

例：

```
[root@openstack01 ~]# env|grep OS_
```

查看keystone实例相关信息

```
openstack endpoint list
openstack project list
openstack user list
```

2.5.创建keystone的一般实例

1）创建一个名为example的keystone域

```
openstack domain create --description "An Example Domain" example
```

例：

```
[root@openstack01 ~]# openstack domain create --description "An Example Domain" example
```

2）为keystone系统环境创建名为service的项目提供服务 用于常规（非管理）任务，需要使用无特权用户

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

例：
[root@openstack01 ~]# openstack project create --domain default --description "Service Project" service
```

3）创建myproject项目和对应的用户及角色 作为普通用户的项目

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

4）默认域创建myuser用户 使用--password选项为直接配置明文密码，使用--password-prompt选项为交互式输入密码 以下命令会在local_user表增加myuser用户

```
# 直接创建用户和密码
openstack user create --domain default  --password=myuser myuser
```

5）在role表创建myrole角色

```
openstack role create myrole
```

6）将myrole角色添加到myproject项目中和myuser用户组中

```
openstack role add --project myproject --user myuser myrole
```

2.8.验证操作keystone是否安装成功

1）作为管理员用户去请求一个认证的token 测试是否可以使用admin账户进行登陆认证，请求认证令牌

```
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
```

3）测试环境管理脚本 使用脚本加载相关客户端配置，以便快速使用特定租户和用户运行客户端

```
source admin-openrc
```

4）请求认证令牌

```
openstack token issue

```

keystone安装完毕

3.Glance镜像组件（6章）

3.1.在控制端安装镜像服务glance (6.2)

1）创建glance数据库

```
mysql -p123456

CREATE DATABASE glance;
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' IDENTIFIED BY 'glance';
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' IDENTIFIED BY 'glance';
flush privileges;
exit
```

3.2.在keystone上面注册glance (6.2)

1）在keystone上创建glance用户 以下命令在local_user表创建glance用户

```
source admin-openrc
openstack user create --domain default --password=glance glance
openstack user list
```

2）在keystone上将glance用户添加为service项目的admin角色(权限) 以下命令无输出

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

3）创建glance镜像服务的实体 以下命令在service表中增加glance项目

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

4）创建镜像服务的 API 端点（endpoint） 以下命令会在endpoint表增加3条项目

```
openstack endpoint create --region RegionOne image public http://192.168.56.126:9292
openstack endpoint create --region RegionOne image internal http://192.168.56.126:9292
openstack endpoint create --region RegionOne image admin http://192.168.56.126:9292
openstack endpoint list
```

Glance在keystone上面注册完成

3.3.glance相关软件 (6.3)

1）检查Python版本

```
[root@openstack01 tools]# python --version
```

2）安装glance软件

```
yum install openstack-glance python-glance python-glanceclient -y
```

3）执行以下命令快速配置glance-api.conf

```
openstack-config --set  /etc/glance/glance-api.conf database connection mysql+pymysql://glance:glance@controller/glance
openstack-config --set  /etc/glance/glance-api.conf keystone_authtoken www_authenticate_uri http://controller:5000
openstack-config --set  /etc/glance/glance-api.conf keystone_authtoken auth_url http://controller:5000
openstack-config --set  /etc/glance/glance-api.conf keystone_authtoken memcached_servers controller:11211
openstack-config --set  /etc/glance/glance-api.conf keystone_authtoken auth_type password
openstack-config --set  /etc/glance/glance-api.conf keystone_authtoken project_domain_name Default
```

```
openstack-config --set  /etc/glance/glance-api.conf keystone_authtoken user_domain_name Default
openstack-config --set  /etc/glance/glance-api.conf keystone_authtoken project_name service
openstack-config --set  /etc/glance/glance-api.conf keystone_authtoken username glance
openstack-config --set  /etc/glance/glance-api.conf keystone_authtoken password glance
openstack-config --set  /etc/glance/glance-api.conf paste_deploy flavor keystone
openstack-config --set  /etc/glance/glance-api.conf glance_store stores  file,http
openstack-config --set  /etc/glance/glance-api.conf glance_store default_store file
openstack-config --set  /etc/glance/glance-api.conf glance_store filesystem_store_datadir /var/lib/glance/images/
```

4）执行以下命令可以快速配置glance-registry.conf

```
openstack-config --set  /etc/glance/glance-registry.conf database connection mysql+pymysql://glance:glance@controller/glance
openstack-config --set  /etc/glance/glance-registry.conf keystone_authtoken www_authenticate_uri http://controller:5000
openstack-config --set  /etc/glance/glance-registry.conf keystone_authtoken auth_url http://controller:5000
openstack-config --set  /etc/glance/glance-registry.conf keystone_authtoken memcached_servers controller:11211
openstack-config --set  /etc/glance/glance-registry.conf keystone_authtoken auth_type password
openstack-config --set  /etc/glance/glance-registry.conf keystone_authtoken project_domain_name Default
openstack-config --set  /etc/glance/glance-registry.conf keystone_authtoken user_domain_name Default
openstack-config --set  /etc/glance/glance-registry.conf keystone_authtoken project_name service
openstack-config --set  /etc/glance/glance-registry.conf keystone_authtoken username glance
openstack-config --set  /etc/glance/glance-registry.conf keystone_authtoken password glance
openstack-config --set  /etc/glance/glance-registry.conf paste_deploy flavor keystone
```

glance服务安装完毕

3.4.同步glance数据库

1）为glance镜像服务初始化同步数据库 生成的相关表（15张表）

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

2）同步完成进行连接测试 保证所有需要的表已经建立

```
mysql -h192.168.56.126 -uglance -pglance -e "use glance;show tables;"
```

3.5.启动glance镜像服务 1）启动glance镜像服务、并配置开机自启动

```
systemctl start openstack-glance-api.service openstack-glance-registry.service
systemctl status openstack-glance-api.service openstack-glance-registry.service

systemctl enable openstack-glance-api.service openstack-glance-registry.service
systemctl list-unit-files |grep openstack-glance*
```

3.5.检验glance (6.4)

可以下载小型的Linux镜像CirrOS用来进行部署测试。 下载地址：http://download.cirros-cloud.net/

1）下载镜像

```
yum install wget -y
wget http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img
```

2）获取管理员权限

```
source admin-openrc
```

3）上传镜像到glance 使用qcow2磁盘格式， bare容器格式上传镜像到镜像服务并设置公共可见

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

4）查看镜像

```
openstack image list
```

Glance镜像服务安装完成

4.安装Nova计算服务（7章）

4.1.在控制节点安装nova计算服务 (7.2)

1）创建nova相关数据库 nova服务在Rocky新增加了两个数据库

```
mysql -u root -p123456

CREATE DATABASE nova_api;
CREATE DATABASE nova;
CREATE DATABASE nova_cell0;
CREATE DATABASE placement;

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

GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' IDENTIFIED BY 'nova';
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' IDENTIFIED BY 'nova';

GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'localhost' IDENTIFIED BY 'nova';
GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'%' IDENTIFIED BY 'nova';

GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'localhost' IDENTIFIED BY 'placement';
GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'%' IDENTIFIED BY 'placement';

flush privileges;
show databases;
select user,host from mysql.user;
exit
```

4.2.在keystone上面注册nova服务

创建服务证书

1）在keystone上创建nova用户

```
source admin-openrc
openstack user create --domain default --password=nova nova
openstack user list
```

2）在keystone上将nova用户配置为admin角色并添加进service项目 以下命令无输出

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

3）创建nova计算服务的实体

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

4）创建计算服务的API端点（endpoint） 计算服务compute端点

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

5）Rocky版本的nova增加了placement项目 同样，创建并注册该项目的服务证书

```
openstack user create --domain default --password=placement placement
openstack role add --project service --user placement admin
openstack service create --name placement --description "Placement API" placement
```

创建placement项目的endpoint（API端口）

```
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
openstack endpoint list
```

4.3.在控制节点安装nova相关服务 (7.3)

1）安装nova相关软件包

```
yum install openstack-nova-api openstack-nova-conductor \
  openstack-nova-console openstack-nova-novncproxy \
  openstack-nova-scheduler openstack-nova-placement-api -y
```

2）快速修改nova配置

```
openstack-config --set  /etc/nova/nova.conf DEFAULT enabled_apis  osapi_compute,metadata
openstack-config --set  /etc/nova/nova.conf DEFAULT my_ip 192.168.56.126
openstack-config --set  /etc/nova/nova.conf DEFAULT use_neutron  true
openstack-config --set  /etc/nova/nova.conf DEFAULT firewall_driver
nova.virt.firewall.NoopFirewallDriver
openstack-config --set  /etc/nova/nova.conf DEFAULT transport_url
rabbit://openstack:openstack@controller
openstack-config --set  /etc/nova/nova.conf api_database connection
mysql+pymysql://nova:nova@controller/nova_api
openstack-config --set  /etc/nova/nova.conf database connection
mysql+pymysql://nova:nova@controller/nova
```

```
openstack-config --set  /etc/nova/nova.conf placement_database connection
mysql+pymysql://placement:placement@controller/placement
openstack-config --set  /etc/nova/nova.conf api auth_strategy  keystone
openstack-config --set  /etc/nova/nova.conf keystone_authtoken auth_url
http://controller:5000/v3
openstack-config --set  /etc/nova/nova.conf keystone_authtoken memcached_servers
controller:11211
openstack-config --set  /etc/nova/nova.conf keystone_authtoken auth_type  password
openstack-config --set  /etc/nova/nova.conf keystone_authtoken project_domain_name  default
openstack-config --set  /etc/nova/nova.conf keystone_authtoken user_domain_name  default
openstack-config --set  /etc/nova/nova.conf keystone_authtoken project_name  service
openstack-config --set  /etc/nova/nova.conf keystone_authtoken username  nova
openstack-config --set  /etc/nova/nova.conf keystone_authtoken password  nova
openstack-config --set  /etc/nova/nova.conf vnc enabled true
openstack-config --set  /etc/nova/nova.conf vnc server_listen '$my_ip'
openstack-config --set  /etc/nova/nova.conf vnc server_proxyclient_address '$my_ip'
openstack-config --set  /etc/nova/nova.conf glance api_servers  http://controller:9292
openstack-config --set  /etc/nova/nova.conf oslo_concurrency lock_path  /var/lib/nova/tmp
openstack-config --set  /etc/nova/nova.conf placement region_name RegionOne
openstack-config --set  /etc/nova/nova.conf placement project_domain_name Default
openstack-config --set  /etc/nova/nova.conf placement project_name service
openstack-config --set  /etc/nova/nova.conf placement auth_type password
openstack-config --set  /etc/nova/nova.conf placement user_domain_name Default
openstack-config --set  /etc/nova/nova.conf placement auth_url http://controller:5000/v3
openstack-config --set  /etc/nova/nova.conf placement username placement
openstack-config --set  /etc/nova/nova.conf placement password placement
openstack-config --set  /etc/nova/nova.conf scheduler discover_hosts_in_cells_interval 300
```

检查生效的nova配置

```
egrep -v "^#|^$" /etc/nova/nova.conf
```

3）修改nova的虚拟主机配置文件 需要修改nova虚拟主机配置文件，增加内容如下：

```
vi /etc/httpd/conf.d/00-nova-placement-api.conf

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

修改完毕重启httpd服务

```
systemctl restart httpd
systemctl status httpd
```

nova计算服务的软件包安装完成

4.4.同步nova数据

nova_api有32张表，placement有32张表，nova_cell0有110张表，nova也有110张表

1）初始化nova-api和placement数据库

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

验证数据库

```
mysql -h192.168.56.126 -unova -pnova -e "use nova_api;show tables;"
mysql -h192.168.56.126 -uplacement -pplacement -e "use placement;show tables;"
```

2）初始化nova__cell0和nova数据库 注册cell0数据库

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

创建cell1单元

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

初始化nova数据库

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

检查确认cell0和cell1注册成功

```
su -s /bin/sh -c "nova-manage cell_v2 list_cells" nova
```

验证数据库

```
mysql -h192.168.56.126 -unova -pnova -e "use nova_cell0;show tables;"
mysql -h192.168.56.126 -unova -pnova -e "use nova;show tables;"
```

5）检查确认cell0和cell1注册成功

```
su -s /bin/sh -c "nova-manage cell_v2 list_cells" nova

```

返回的数据存储在nova__api数据库的cell__mappings表中

4.5.启动nova服务

1）启动nova服务并设置为开机自启动 需要启动5个服务

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

systemctl status openstack-nova-api.service openstack-nova-consoleauth.service \
  openstack-nova-scheduler.service openstack-nova-conductor.service \
  openstack-nova-novncproxy.service

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

systemctl list-unit-files |grep openstack-nova* |grep enabled
```

控制节点安装nova计算服务完毕

5.安装Nova计算节点（7章）

5.1.配置基本环境（All-in-One模式忽略）

...

5.2（All-in-One模式忽略）

5.3（All-in-One模式忽略）

5.4（All-in-One模式忽略）

5.5.安装nova计算节点相关软件包 (7.4)

1）计算节点安装nova软件包

```
yum install openstack-nova-compute python-openstackclient openstack-utils -y
```

2）快速修改配置文件（/etc/nova/nova.conf）

```
openstack-config --set  /etc/nova/nova.conf DEFAULT my_ip 192.168.56.126
openstack-config --set  /etc/nova/nova.conf DEFAULT use_neutron True
openstack-config --set  /etc/nova/nova.conf DEFAULT firewall_driver nova.virt.firewall.NoopFirewallDriver
openstack-config --set  /etc/nova/nova.conf DEFAULT enabled_apis  osapi_compute,metadata
openstack-config --set  /etc/nova/nova.conf DEFAULT transport_url rabbit://openstack:openstack@controller
openstack-config --set  /etc/nova/nova.conf api auth_strategy  keystone
openstack-config --set  /etc/nova/nova.conf keystone_authtoken auth_url http://controller:5000/v3
openstack-config --set  /etc/nova/nova.conf keystone_authtoken memcached_servers controller:11211
openstack-config --set  /etc/nova/nova.conf keystone_authtoken auth_type password
openstack-config --set  /etc/nova/nova.conf keystone_authtoken project_domain_name default
openstack-config --set  /etc/nova/nova.conf keystone_authtoken user_domain_name default
openstack-config --set  /etc/nova/nova.conf keystone_authtoken project_name  service
openstack-config --set  /etc/nova/nova.conf keystone_authtoken username nova
openstack-config --set  /etc/nova/nova.conf keystone_authtoken password nova
openstack-config --set  /etc/nova/nova.conf vnc enabled True
openstack-config --set  /etc/nova/nova.conf vnc server_listen 0.0.0.0
openstack-config --set  /etc/nova/nova.conf vnc server_proxyclient_address  '$my_ip'
openstack-config --set  /etc/nova/nova.conf vnc novncproxy_base_url http://controller:6080/vnc_auto.html
openstack-config --set  /etc/nova/nova.conf glance api_servers http://controller:9292
openstack-config --set  /etc/nova/nova.conf oslo_concurrency lock_path /var/lib/nova/tmp
openstack-config --set  /etc/nova/nova.conf placement region_name RegionOne
openstack-config --set  /etc/nova/nova.conf placement project_domain_name Default
openstack-config --set  /etc/nova/nova.conf placement project_name service
openstack-config --set  /etc/nova/nova.conf placement auth_type password
openstack-config --set  /etc/nova/nova.conf placement user_domain_name Default
openstack-config --set  /etc/nova/nova.conf placement auth_url http://controller:5000/v3
openstack-config --set  /etc/nova/nova.conf placement username placement
openstack-config --set  /etc/nova/nova.conf placement password placement
```

查看生效的配置：

```
egrep -v "^#|^$" /etc/nova/nova.conf
```

3）配置虚拟机的硬件加速 首先确定计算节点是否支持虚拟机的硬件加速。

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

如果返回位0，表示计算节点不支持硬件加速，需要配置libvirt使用QEMU方式管理虚拟机，使用以下命令：

```
openstack-config --set  /etc/nova/nova.conf libvirt virt_type  qemu
```

4）启动nova相关服务，并配置为开机自启动

```
systemctl start libvirtd.service openstack-nova-compute.service
systemctl status libvirtd.service openstack-nova-compute.service
systemctl enable libvirtd.service openstack-nova-compute.service
systemctl list-unit-files |grep libvirtd.service
systemctl list-unit-files |grep openstack-nova-compute.service
```

5）将计算节点增加到cell数据库 以下命令在控制节点操作

```
source admin-openrc
```

检查确认数据库有新的计算节点

```
openstack compute service list --service nova-compute
openstack compute service list
```

手动将新的计算节点添加到openstack集群

```
su -s /bin/sh -c "nova-manage cell_v2 discover_hosts --verbose" nova

```

设置新创建节点自动注册的任务

```
[scheduler]
discover_hosts_in_cells_interval = 300
```

计算节点安装完毕

5.6.在控制节点进行验证

1）应用管理员环境变量脚本

```
source admin-openrc
```

2）列表查看安装的nova服务组件 验证是否成功注册并启动

```
openstack compute service list

```

3）在身份认证服务中列出API端点以验证其连接性

```
openstack catalog list

```

4）在镜像服务中列出已有镜像已检查镜像服务的连接性

```
openstack image list

```

5）检查nova各组件的状态 检查placement API和cell服务是否正常工作

```
nova-status upgrade check

```

nova计算节点安装完毕

6.安装Neutron网络服务（8章）

6.1.主机网络配置及测试(略)

1）控制节点配置

```
vi /etc/hosts
-------------------------------------------
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.56.126     controller
```

2）计算节点配置

```
vi /etc/hosts
-------------------------------------------
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.56.126     controller
```

3）块存储节点配置

```
vi /etc/hosts
-------------------------------------------
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.56.126     controller
```

以上节点的hosts文件配置相同，其他类型节点也照此配置即可

4）检测各节点到控制节点和公网的联通性 控制节点 ping -c 4 controller

6.2.在keystone数据库中注册neutron相关服务 (8.2)

1）创建neutron数据库，并授予访问权限

```
mysql -p123456

CREATE DATABASE neutron;
GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' IDENTIFIED BY 'neutron';
GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' IDENTIFIED BY 'neutron';
exit
```

2）在keystone上创建neutron用户

```
source admin-openrc
openstack user create --domain default --password=neutron neutron
openstack user list
```

3）将neutron添加到service项目并授予admin角色 以下命令无输出

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

4）创建neutron服务实体

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

5）创建neutron网络服务的API端点（endpoint）

```
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
openstack endpoint list
```

6.3.控制节点安装neutron组件 (8.3)

以下为第一种Networking Option 1: Provider networks 1）安装neutron软件包

```
yum install openstack-neutron openstack-neutron-ml2 openstack-neutron-linuxbridge ebtables -y
```

2）快速配置/etc/neutron/neutron.conf

```
openstack-config --set  /etc/neutron/neutron.conf database connection mysql+pymysql://neutron:neutron@controller/neutron
openstack-config --set  /etc/neutron/neutron.conf DEFAULT core_plugin  ml2
openstack-config --set  /etc/neutron/neutron.conf DEFAULT service_plugins
openstack-config --set  /etc/neutron/neutron.conf DEFAULT transport_url rabbit://openstack:openstack@controller
openstack-config --set  /etc/neutron/neutron.conf DEFAULT auth_strategy  keystone
openstack-config --set  /etc/neutron/neutron.conf keystone_authtoken www_authenticate_uri http://controller:5000
openstack-config --set  /etc/neutron/neutron.conf keystone_authtoken auth_url http://controller:5000
openstack-config --set  /etc/neutron/neutron.conf keystone_authtoken memcached_servers controller:11211
openstack-config --set  /etc/neutron/neutron.conf keystone_authtoken auth_type  password
openstack-config --set  /etc/neutron/neutron.conf keystone_authtoken project_domain_name default
openstack-config --set  /etc/neutron/neutron.conf keystone_authtoken user_domain_name default
openstack-config --set  /etc/neutron/neutron.conf keystone_authtoken project_name  service
openstack-config --set  /etc/neutron/neutron.conf keystone_authtoken username  neutron
openstack-config --set  /etc/neutron/neutron.conf keystone_authtoken password  neutron
openstack-config --set  /etc/neutron/neutron.conf DEFAULT notify_nova_on_port_status_changes True
openstack-config --set  /etc/neutron/neutron.conf DEFAULT notify_nova_on_port_data_changes True
openstack-config --set  /etc/neutron/neutron.conf nova auth_url  http://controller:5000
openstack-config --set  /etc/neutron/neutron.conf nova auth_type  password
openstack-config --set  /etc/neutron/neutron.conf nova project_domain_name  default
openstack-config --set  /etc/neutron/neutron.conf nova user_domain_name  default
openstack-config --set  /etc/neutron/neutron.conf nova region_name  RegionOne
```

```
openstack-config --set  /etc/neutron/neutron.conf nova project_name  service
openstack-config --set  /etc/neutron/neutron.conf nova username  nova
openstack-config --set  /etc/neutron/neutron.conf nova password  nova
openstack-config --set  /etc/neutron/neutron.conf oslo_concurrency lock_path
/var/lib/neutron/tmp
```

查看生效的配置

```
egrep -v '(^$|^#)' /etc/neutron/neutron.conf

```

3）快速配置/etc/neutron/plugins/ml2/ml2_conf.ini

```
openstack-config --set  /etc/neutron/plugins/ml2/ml2_conf.ini ml2 type_drivers  flat,vlan
openstack-config --set  /etc/neutron/plugins/ml2/ml2_conf.ini ml2 tenant_network_types
openstack-config --set  /etc/neutron/plugins/ml2/ml2_conf.ini ml2 mechanism_drivers
linuxbridge
openstack-config --set  /etc/neutron/plugins/ml2/ml2_conf.ini ml2 extension_drivers
port_security
openstack-config --set  /etc/neutron/plugins/ml2/ml2_conf.ini ml2_type_flat flat_networks
provider
openstack-config --set  /etc/neutron/plugins/ml2/ml2_conf.ini securitygroup enable_ipset
True
```

查看生效的配置

```
egrep -v '(^$|^#)' /etc/neutron/plugins/ml2/ml2_conf.ini
```

4）快速配置/etc/neutron/plugins/ml2/linuxbridge_agent.ini

```
openstack-config --set   /etc/neutron/plugins/ml2/linuxbridge_agent.ini linux_bridge
physical_interface_mappings  provider:enp0s8
openstack-config --set   /etc/neutron/plugins/ml2/linuxbridge_agent.ini vxlan  enable_vxlan
False
openstack-config --set   /etc/neutron/plugins/ml2/linuxbridge_agent.ini securitygroup
enable_security_group  True
openstack-config --set   /etc/neutron/plugins/ml2/linuxbridge_agent.ini securitygroup
firewall_driver neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

查看生效的配置

```
egrep -v '(^$|^#)' /etc/neutron/plugins/ml2/linuxbridge_agent.ini
```

以下参数在启动neutron-linuxbridge-agent.service的时候会自动设置为1

```
有误（略）
sysctl net.bridge.bridge-nf-call-iptables
sysctl net.bridge.bridge-nf-call-ip6tables
```

5）快速配置/etc/neutron/dhcp_agent.ini

```
openstack-config --set   /etc/neutron/dhcp_agent.ini DEFAULT  interface_driver  linuxbridge
openstack-config --set   /etc/neutron/dhcp_agent.ini DEFAULT  dhcp_driver
neutron.agent.linux.dhcp.Dnsmasq
openstack-config --set   /etc/neutron/dhcp_agent.ini DEFAULT  enable_isolated_metadata  True
```

查看生效的配置

```
egrep -v '(^$|^#)' /etc/neutron/dhcp_agent.ini
```

至此，方式1的配置文件修改完毕

6）快速配置/etc/neutron/metadata_agent.ini

```
openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT nova_metadata_host controller
openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT metadata_proxy_shared_secret neutron
```

查看生效的配置

```
egrep -v '(^$|^#)' /etc/neutron/metadata_agent.ini
```

shared_secret选项是元数据代理，需要设置一个合适的密码这里设置为neutron

7）配置计算服务使用网络服务 快速配置/etc/nova/nova.conf，将neutron添加到计算节点中

```
openstack-config --set  /etc/nova/nova.conf  neutron url http://controller:9696
openstack-config --set  /etc/nova/nova.conf  neutron auth_url http://controller:5000
openstack-config --set  /etc/nova/nova.conf  neutron auth_type password
openstack-config --set  /etc/nova/nova.conf  neutron project_domain_name default
openstack-config --set  /etc/nova/nova.conf  neutron user_domain_name default
openstack-config --set  /etc/nova/nova.conf  neutron region_name RegionOne
openstack-config --set  /etc/nova/nova.conf  neutron project_name service
openstack-config --set  /etc/nova/nova.conf  neutron username neutron
openstack-config --set  /etc/nova/nova.conf  neutron password neutron
openstack-config --set  /etc/nova/nova.conf  neutron service_metadata_proxy true
openstack-config --set  /etc/nova/nova.conf  neutron metadata_proxy_shared_secret neutron
```

查看生效的配置

```
egrep -v '(^$|^#)' /etc/nova/nova.conf
```

8）初始化网络插件 创建网络插件的链接，初始化网络的脚本插件会用到/etc/neutron/plugin.ini，需要使用ML2的插件进行提供

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

9）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
```

10）重启nova_api服务

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

11）启动neutron服务并设置开机启动 需要启动4个服务

```
systemctl restart neutron-server.service neutron-linuxbridge-agent.service neutron-dhcp-agent.service neutron-metadata-agent.service
systemctl status neutron-server.service neutron-linuxbridge-agent.service neutron-dhcp-agent.service neutron-metadata-agent.service
systemctl enable neutron-server.service neutron-linuxbridge-agent.service neutron-dhcp-agent.service neutron-metadata-agent.service
systemctl list-unit-files |grep neutron* |grep enabled
```

控制端的neutron网络服务就安装完成，之后需要在计算节点安装网络服务组件

6.4.在计算节点安装neutron网络组件 (8.4)

1）安装neutron组件

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

2）快速配置/etc/neutron/neutron.conf

```
openstack-config --set /etc/neutron/neutron.conf DEFAULT transport_url rabbit://openstack:openstack@controller
openstack-config --set /etc/neutron/neutron.conf DEFAULT auth_strategy keystone
openstack-config --set /etc/neutron/neutron.conf keystone_authtoken www_authenticate_uri http://controller:5000
openstack-config --set /etc/neutron/neutron.conf keystone_authtoken auth_url http://controller:5000
openstack-config --set /etc/neutron/neutron.conf keystone_authtoken memcached_servers controller:11211
openstack-config --set /etc/neutron/neutron.conf keystone_authtoken auth_type password
openstack-config --set /etc/neutron/neutron.conf keystone_authtoken project_domain_name default
openstack-config --set /etc/neutron/neutron.conf keystone_authtoken user_domain_name default
openstack-config --set /etc/neutron/neutron.conf keystone_authtoken project_name service
openstack-config --set /etc/neutron/neutron.conf keystone_authtoken username neutron
openstack-config --set /etc/neutron/neutron.conf keystone_authtoken password neutron
openstack-config --set /etc/neutron/neutron.conf oslo_concurrency lock_path /var/lib/neutron/tmp
```

查看生效的配置

```
egrep -v '(^$|^#)' /etc/neutron/neutron.conf
```

3）快速配置/etc/neutron/plugins/ml2/linuxbridge_agent.ini

```
openstack-config --set /etc/neutron/plugins/ml2/linuxbridge_agent.ini linux_bridge physical_interface_mappings  provider:enp0s8
openstack-config --set /etc/neutron/plugins/ml2/linuxbridge_agent.ini vxlan enable_vxlan false
openstack-config --set /etc/neutron/plugins/ml2/linuxbridge_agent.ini securitygroup enable_security_group true
openstack-config --set /etc/neutron/plugins/ml2/linuxbridge_agent.ini securitygroup firewall_driver neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

注意：第一个选项physical_interface_mappings选项要配置计算节点自身的网卡名称，如provider:enp0s8

查看生效的配置

```
egrep -v '(^$|^#)' /etc/neutron/plugins/ml2/linuxbridge_agent.ini
```

4）配置nova计算服务与neutron网络服务协同工作 快速配置/etc/nova/nova.conf

```
openstack-config --set /etc/nova/nova.conf neutron url http://controller:9696
openstack-config --set /etc/nova/nova.conf neutron auth_url http://controller:5000
openstack-config --set /etc/nova/nova.conf neutron auth_type password
openstack-config --set /etc/nova/nova.conf neutron project_domain_name default
openstack-config --set /etc/nova/nova.conf neutron user_domain_name default
openstack-config --set /etc/nova/nova.conf neutron region_name RegionOne
openstack-config --set /etc/nova/nova.conf neutron project_name service
openstack-config --set /etc/nova/nova.conf neutron username neutron
openstack-config --set /etc/nova/nova.conf neutron password neutron
```

查看生效的配置

```
egrep -v '(^$|^#)' /etc/nova/nova.conf
```

5）重启计算节点

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

6）启动neutron网络组件，并配置开机自启动 需要启动1个服务，网桥代理

```
systemctl restart neutron-linuxbridge-agent.service
systemctl status neutron-linuxbridge-agent.service

systemctl enable neutron-linuxbridge-agent.service
systemctl list-unit-files |grep neutron* |grep enabled
```

计算节点的网络配置完成，转回到控制节点进行验证操作

6.5.在控制节点检查确认neutron服务安装成功

Verify operation： https://docs.openstack.org/neutron/rocky/install/verify.html

以下命令在控制节点执行

1）获取管理权限

```
source admin-openrc
```

2）列表查看加载的网络插件

```
openstack extension list --network
```

3）查看网络代理列表

```
openstack network agent list
```

控制节点有3个服务，计算节点有1个服务

7.Horizon服务组件（9章）

7.0.horizon（dashboard）

Rocky-官方 https://docs.openstack.org/install-guide/openstack-services.html#minimal-deployment-for-rocky

7.1.安装dashboard 控台 (9.2)

1）安装dashboard软件包

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

2）修改配置文件/etc/openstack-dashboard/local_settings 检查确认有以下配置

```
vi /etc/openstack-dashboard/local_settings
----------------------------------------------------------
ALLOWED_HOSTS = ['*', ]
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
OPENSTACK_API_VERSIONS = {
    "identity": 3,
    "image": 2,
    "volume": 2,
}
OPENSTACK_HOST = "controller"
OPENSTACK_KEYSTONE_URL = "http://%s:5000/v3" % OPENSTACK_HOST
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "default"
CACHES = {
    'default': {
         'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
         'LOCATION': 'controller:11211',
    }
}
OPENSTACK_NEUTRON_NETWORK = {
    'enable_router': False,
    'enable_quotas': False,
    'enable_distributed_router': False,
    'enable_ha_router': False,
    'enable_fip_topology_check': False,
    'enable_lb': False,
    'enable_firewall': False,
    'enable_vpn': False,
}
TIME_ZONE = "Asia/Shanghai"
```

3）修改/etc/httpd/conf.d/openstack-dashboard.conf 增加以下内容

```
vi /etc/httpd/conf.d/openstack-dashboard.conf
---------------------------------------------
WSGIApplicationGroup %{GLOBAL}
```

4）重启web服务器以及会话存储服务

```
systemctl restart httpd.service memcached.service
systemctl status httpd.service memcached.service
```

5）检查dashboard是否可用 在浏览器中输入下面的地址：用户名和密码，域名用default http://controller:80/dashboard

6）其他可选的dashboard配置 https://docs.openstack.org/horizon/rocky/install/next-steps.html

8. 启动实例 Instances（15章）

8.0.neutron的两种虚拟网络

1）Provider network（提供者网络）

集群中的各个节点通过物理网络连接，节点内部通过（provider网桥/交换机）与物理网络进行连接。集群中的实例（虚拟机）通过Provider网络为其分配映射的tap端口与桥接网卡传输数据从而进行内外部通信。

2）Self-service network（自服务网络）

在provider网络上的扩展，通过self-service网桥使用vxlan技术创建一个独立的网络，这个独立的网络也可以通过vxlan tunnels连接到物理网络进行数据传输。

8.1.创建provider提供者网络 (15.1)

1）在控制节点上，创建网络接口 加载 admin 凭证

```
source admin-openrc
openstack network create --share --external --provider-physical-network provider  --provider-network-type flat provider
openstack network list
```

2）检查网络配置 确认ml2_conf.ini以下配置选项 上面的命令--provider-network-type flat网络名称provider与此对应

```
cat /etc/neutron/plugins/ml2/ml2_conf.ini
--------------------------------------------------
[ml2_type_flat]
flat_networks = provider
```

确认linuxbridge_agent.ini以下配置选项 --provider-physical-network provider与此对应，控制节点的网卡名称

```
cat /etc/neutron/plugins/ml2/linuxbridge_agent.ini
--------------------------------------------------
[linux_bridge]
physical_interface_mappings = provider:enp0s8
```

3）创建provider子网

```
openstack subnet create --network provider --allocation-pool start=192.168.56.210,end=192.168.56.220 --dns-nameserver 4.4.4.4 --gateway 192.168.56.1 --subnet-range 192.168.56.0/24 provider-subnet01
openstack subnet list
```

provider网络创建完成，可以创建虚拟机

8.2.创建实例

1）创建私有网络接口 创建flavor

```
openstack flavor create --id 0 --vcpus 1 --ram 64 --disk 1 m1.nano
```

创建一个Self-service网络的虚拟机 这里的net-id是

```
openstack network list

将网络ID修改为你本机的ID
openstack server create --flavor m1.nano --image cirros --nic net-id=e107b690-9daa-42f2-8cc0-379b6f09cd5e vm1
```

查看是否创建成功

```
openstack server list
```

安装完毕，可通过 http://192.168.56.126/dashboard 查看和操作。

OpenStack（存储）

1. Swift安装准备（12.2）

OpenStack中Swift 服务依赖于身份验证Keystone;

同时Swift 可作为独立存储组件，在没有任何OpenStack服务的情况下运行;

配置对象存储服务之前，须创建服务凭证和API端点。

1.admin凭据

```
source admin-openrc
```

2.要创建身份服务凭据，请完成以下步骤 A.创建swift用户

```
openstack user create --domain default --password=swift swift
```

B.将admin角色添加至swift

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

C.创建swift服务实体

```
openstack service create --name swift \
  --description "OpenStack Object Storage" object-store
```

3.创建对象存储服务API端点

```
openstack endpoint create --region RegionOne \
  object-store public http://controller:8080/v1/AUTH_%\(tenant_id\)s
openstack endpoint create --region RegionOne \
  object-store internal http://controller:8080/v1/AUTH_%\(tenant_id\)s
openstack endpoint create --region RegionOne \
  object-store admin http://controller:8080/v1
```

2.安装和配置的部件（12.3）

1、安装服务组件

```
yum install -y openstack-swift-proxy python-swiftclient \
  python-keystoneclient python-keystonemiddleware \
  memcached
```

2、从Object Storage源存储库获取代理服务配置文件

若下载文件失败，可直接拷贝课件“配置文件”中相应的文件至/etc/swift目录下

```
curl -o /etc/swift/proxy-server.conf
https://opendev.org/openstack/swift/raw/branch/master/etc/proxy-server.conf-sample
```

3、 编辑/etc/swift/proxy-server.conf文件并完成以下操作

A.在该[DEFAULT]部分中，配置绑定端口，用户和配置目录：

```
vi /etc/swift/proxy-server.conf
```

```
[DEFAULT]
...
bind_port = 8080
user = swift
swift_dir = /etc/swift
```

B.在[pipeline:main]部分中，

**删除tempurl和 tempauth模块并添加authtoken和keystoneauth 模块

```
[pipeline:main]
pipeline = catch_errors gatekeeper healthcheck proxy-logging cache container_sync bulk
ratelimit authtoken keystoneauth container-quotas account-quotas slo dlo versioned_writes
proxy-logging proxy-server
```

C.在[app:proxy-server]部分中，启用自动帐户创建

```
[app:proxy-server]
use = egg:swift#proxy
...
account_autocreate = True

```

D.在[filter:keystoneauth]部分中，配置操作员角色

```
[filter:keystoneauth]
use = egg:swift#keystoneauth
...
operator_roles = admin,user
```

E.在[filter:authtoken]中，配置身份服务访问

```
[filter:authtoken]
paste.filter_factory = keystonemiddleware.auth_token:filter_factory
...
auth_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 = swift
password = swift
delay_auth_decision = True
```

F.在[filter:cache]部分中，配置memcached位置

```
[filter:cache]
use = egg:swift#memcache
...
memcache_servers = controller:11211
```

3. 安装和配置存储节点（12.4）

若此处存储节点为计算节点和网络节点，应分别在两个节点都完成以下步骤;

注意本实验是All-In-One模式，所有操作在一台VirtualBox CentOS虚拟机中完成。

1. 先决条件

安装和配置存储节点上的对象存储服务之前，须预先准备存储设备（添加两块硬盘，大小为1G；VirtualBox Centos实验环境中已经预备）

1、 安装支持实用程序包

```
yum install -y xfsprogs rsync
```

2、 格式/dev/sdb和/dev/sdc设备XFS

```
mkfs.xfs /dev/sdb
mkfs.xfs /dev/sdc
```

3、 创建挂载点目录结构

```
mkdir -p /srv/node/sdb
mkdir -p /srv/node/sdc
```

4、 编辑/etc/fstab文件，添加以下

```
vi /etc/fstab
--------------------------------------------------------------------
/dev/sdb /srv/node/sdb xfs noatime,nodiratime,nobarrier,logbufs=8 0 2
/dev/sdc /srv/node/sdc xfs noatime,nodiratime,nobarrier,logbufs=8 0 2
```

5、 安装设备

```
mount /srv/node/sdb
mount /srv/node/sdc
```

6、 创建或编辑/etc/rsyncd.conf文件包含以下

```
uid = swift
gid = swift
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
address =192.168.56.126
[account]
max connections = 2
path = /srv/node/
read only = False
lock file = /var/lock/account.lock
[container]
max connections = 2
path = /srv/node/
read only = False
lock file = /var/lock/container.lock
[object]
max connections = 2
path = /srv/node/
read only = False
lock file = /var/lock/object.lock
```

7、启动rsyncd服务，配置为系统启动时自启动

```
systemctl enable rsyncd.service
systemctl start rsyncd.service
```

2. 安装和配置组件（12.4.2）

1.安装服务组件

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

2.从对象存储源存储库获取accounting, container, object服务配置文件

若下载文件失败，可直接拷贝课件“配置文件”中相应的文件至/etc/swift目录下

```
curl -o /etc/swift/account-server.conf
https://opendev.org/openstack/swift/raw/branch/master/etc/account-server.conf-sample
curl -o /etc/swift/container-server.conf
https://opendev.org/openstack/swift/raw/branch/master/etc/container-server.conf-sample
curl -o /etc/swift/object-server.conf
https://opendev.org/openstack/swift/raw/branch/master/etc/object-server.conf-sample
```

3、 编辑/etc/swift/account-server.conf文件并完成以下操作

A.在[DEFAULT]部分中，配置绑定IP地址、绑定端口、用户、配置目录和挂载点目录

```
[DEFAULT]
...
bind_ip = 192.168.56.126
bind_port = 6202
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = True
```

B. 在[pipeline:main]部分，启用适当的模块

```
[pipeline:main]
pipeline = healthcheck recon account-server
```

C. 在[filter:recon]部分，配置 recon (meters)缓存目录

```
[filter:recon]
use = egg:swift#recon
...
recon_cache_path = /var/cache/swift
```

4、 编辑/etc/swift/container-server.conf文件并完成以下操作

A.在[DEFAULT]部分中，配置绑定IP地址、绑定端口、用户、配置目录和挂载点目录

```
[DEFAULT]
...
bind_ip = 192.168.56.126
bind_port = 6201
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = True
```

B. 在[pipeline:main]部分，启用相应的模块

```
[pipeline:main]
pipeline = healthcheck recon container-server
```

C. 在[filter:recon]部分，配置 recon (meters)缓存目录

```
[filter:recon]
use = egg:swift#recon
...
recon_cache_path = /var/cache/swift
```

5、编辑/etc/swift/object-server.conf文件并完成以下操作

A.在[DEFAULT]部分中，配置绑定IP地址、绑定端口、用户、配置目录和挂载点目录

```
[DEFAULT]
...
bind_ip = 192.168.56.126
bind_port = 6200
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = True
```

B. 在[pipeline:main]部分，启用适当的模块

```
[pipeline:main]
pipeline = healthcheck recon object-server
```

C. 在[filter:recon]部分，配置 recon (meters)缓存目录

```
[filter:recon]
use = egg:swift#recon
...
recon_cache_path = /var/cache/swift
recon_lock_path = /var/lock
```

6、确保挂载点目录结构的所有者

```
chown -R swift:swift /srv/node
```

7、创建recon目录并确保它的用户和组是Swift

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

3. 创建并分发ring（12.5）

在启动对象存储服务之前，须创建初始帐户、容器和对象环。

3.1 创建账户ring

帐户服务器使用帐户环来维护容器列表。

1.转到/etc/swift目录。

```
cd /etc/swift
```

2.创建基本account.builder文件：

```
swift-ring-builder account.builder create 10 2 1
```

3.将每个存储节点添加到环中

```
swift-ring-builder account.builder add \
  --region 1 --zone 1 --ip 192.168.56.126 --port 6202 --device sdb --weight 100
swift-ring-builder account.builder add \
  --region 1 --zone 1 --ip 192.168.56.126 --port 6202 --device sdc --weight 100

```

4.验证 ring 的内容

```
swift-ring-builder account.builder
```

5.平衡 ring

```
swift-ring-builder account.builder rebalance
```

3.2 创建容器ring

帐户服务器使用帐户 ring 来维护一个容器的列表。 1、切换到 /etc/swift目录。

2、创建基本container.builder文件

```
swift-ring-builder container.builder create 10 2 1
```

3、添加每个节点到 ring 中

```
swift-ring-builder container.builder add \
  --region 1 --zone 1 --ip 192.168.56.126 --port 6201 --device sdb --weight 100
swift-ring-builder container.builder add \
  --region 1 --zone 1 --ip 192.168.56.126 --port 6201 --device sdc --weight 100

```

4、验证 ring 的内容

```
swift-ring-builder container.builder
```

5、平衡 ring

```
swift-ring-builder container.builder rebalance
```

3.3 创建对象ring

对象服务器使用对象环来维护对象在本地设备上的位置列表。 1、切换到 /etc/swift目录。

2、创建基本object.builder文件

```
swift-ring-builder object.builder create 10 2 1
```

3、添加每个节点到 ring 中

```
swift-ring-builder object.builder add \
  --region 1 --zone 1 --ip 192.168.56.126 --port 6200 --device sdb --weight 100
swift-ring-builder object.builder add \
  --region 1 --zone 1 --ip 192.168.56.126 --port 6200 --device sdc --weight 100

```

4、验证 ring 的内容:

```
swift-ring-builder object.builder
```

5、平衡 ring:

```
swift-ring-builder object.builder rebalance
```

分发环配置文件

复制account.ring.gz，container.ring.gz和object.ring.gz 文件到每个存储节点和其他运行了代理服务的额外节点的 /etc/swift 目录。 All-in-One模式下忽略此操作。

4. 完成安装（ 12.6 ）

1、获取/etc/swift/swift.conf从源存储库中获取文件

```
curl -o /etc/swift/swift.conf \
  https://opendev.org/openstack/swift/raw/branch/stable/rocky/etc/swift.conf-sample
```

2、编辑/etc/swift/swift.conf文件并完成以下操作:

A.在该[swift-hash]部分中，配置散列路径前缀和后缀。

```
[swift-hash]
...
swift_hash_path_suffix = hsystsy.cy
swift_hash_path_prefix = hsystsy.cy
```

B.在该[storage-policy:0]部分中，配置默认存储策略：

```
[storage-policy:0]
...
name = Policy-0
default = yes
```

3、将swift.conf文件复制到/etc/swift每个存储节点上的目录以及任何运行代理服务的其他节点。 4、在所有节点上，确保配置目录的所有权

```
chown -R root:swift /etc/swift
```

5.在控制器节点和运行代理服务的任何其他节点上，启动Object Storage代理服务（包括其依赖关系），并将其配置为在系统引导时启动

```
systemctl enable openstack-swift-proxy.service memcached.service
systemctl restart openstack-swift-proxy.service memcached.service
```

在存储节点上，启动对象存储服务，并配置它们在系统启动时启动

```
systemctl enable openstack-swift-account.service openstack-swift-account-auditor.service \
  openstack-swift-account-reaper.service openstack-swift-account-replicator.service
systemctl start openstack-swift-account.service openstack-swift-account-auditor.service \
  openstack-swift-account-reaper.service openstack-swift-account-replicator.service
systemctl enable openstack-swift-container.service \
  openstack-swift-container-auditor.service openstack-swift-container-replicator.service \
  openstack-swift-container-updater.service
systemctl start openstack-swift-container.service \
  openstack-swift-container-auditor.service openstack-swift-container-replicator.service \
  openstack-swift-container-updater.service
systemctl enable openstack-swift-object.service openstack-swift-object-auditor.service \
  openstack-swift-object-replicator.service openstack-swift-object-updater.service
systemctl start openstack-swift-object.service openstack-swift-object-auditor.service \
  openstack-swift-object-replicator.service openstack-swift-object-updater.service
```

5. 验证操作（ 12.7 ）

验证对象存储服务 若使用Linux 7 or CentOS 7且尚未关闭SELinux，需要修改目录的安全上下文，更改为swift_data_t类型，object_r 角色和system_u用户的最低安全级别（s0）。若已经关闭SElinux，可忽略以下操作。

```
chcon -R system_u:object_r:swift_data_t:s0 /srv/node
```

1、加载证书信息

```
source admin-openrc
```

2、显示Swift服务状态

```
swift stat
```

```
[root@controller swift]# swift stat
```

3、创建容器container

```
openstack container create cont1
```

```
[root@controller ~]# openstack container create cont1
```

4、将测试文件上传到cont1容器

```
cd /root
cp admin-openrc testfile1
openstack object create cont1 testfile1
```

```
[root@controller ~]# openstack object create cont1 testfile1
```

5、在cont1容器中列出文件

```
openstack object list cont1
```

```
[root@controller ~]# openstack object list cont1
```

6、从container1容器中下载一个测试文件

先删除本地testfile1文件

```
[root@controller ~]# rm testfile1
```

```
openstack object save cont1 testfile1
```

从容器下载testfile1文件至本地

```
[root@controller ~]# openstack object save cont1 testfile1
```

查看testfile1

```
[root@controller ~]# more testfile1
export OS_USERNAME=admin
export OS_PASSWORD=123456
export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_AUTH_URL=http://controller:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

实验完毕，可通过 http://192.168.56.126/dashboard 查看和操作容器。