Swift 单节点部署

```

# 在部署对象存储服务(swift)之前,你的环境必须包含身份验证服务(keystone);

# keystone需要MySQL数据库,Rabbitmq服务,Memcached服务;

# 内存：4G

# 系统：Ubuntu Server-14.04.5

# 安装方法：http://www.jianshu.com/p/9e77b3ad930a

# IP地址：192.168.10.55

# 主机名：object

```

---

## 基本环境配置

### 配置主机静态IP地址

```

vim /etc/network/interfaces

```

```

auto lo

iface lo inet loopback

auto eth0

# 将dhcp修改为static

iface eth0 inet static

# 静态IP地址

address 192.168.10.55

# 子网掩码

netmask 255.255.255.0

# 广播地址

broadcast 192.168.10.255

# 默认网关

gateway 192.168.10.2

# DNS服务器

## 谷歌DNS

dns-nameservers 8.8.8.8

## 阿里DNS

dns-nameservers 223.5.5.5

```

### 重启网卡

```

# 关闭网卡

ifdown eth0

# 开启网卡

ifup eth0

```

### 配置主机名

```

vim /etc/hostname

```

```

# 对于不同的节点，请做出相应的修改

# 清空文件内容

# 主机名

object

```

### 配置主机名解析

```

vim /etc/hosts

```

```

# 文件内容，请视实际情况做相应的修改

192.168.10.55 object

```

### 验证操作

```

ping -c 4 主机名

# 例如

ping -c 4 object

```

###配置Ubuntu更新源

```

vim /etc/apt/sources.list

```

```

# 请先把文件内容清空

# 中国科学技术大学源

deb http://mirrors.ustc.edu.cn/ubuntu/ trusty main restricted universe multiverse

deb http://mirrors.ustc.edu.cn/ubuntu/ trusty-security main restricted universe multiverse

deb http://mirrors.ustc.edu.cn/ubuntu/ trusty-updates main restricted universe multiverse

deb http://mirrors.ustc.edu.cn/ubuntu/ trusty-proposed main restricted universe multiverse

deb http://mirrors.ustc.edu.cn/ubuntu/ trusty-backports main restricted universe multiverse

deb-src http://mirrors.ustc.edu.cn/ubuntu/ trusty main restricted universe multiverse

deb-src http://mirrors.ustc.edu.cn/ubuntu/ trusty-security main restricted universe multiverse

deb-src http://mirrors.ustc.edu.cn/ubuntu/ trusty-updates main restricted universe multiverse

deb-src http://mirrors.ustc.edu.cn/ubuntu/ trusty-proposed main restricted universe multiverse

deb-src http://mirrors.ustc.edu.cn/ubuntu/ trusty-backports main restricted universe multiverse

```

###更新系统

```

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

```

###添加OpenStack库

```

apt-get install software-properties-common

# 此处命令行会停顿，请按Enter键继续

add-apt-repository cloud-archive:mitaka

```

### 安装OpenStack客户端

```

apt-get install python-openstackclient

```

###更新系统

```

# 此处为必需步骤

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

```

### 重启主机

```

shutdown -r now

# 重启电脑后,XShell要用新的IP地址连接虚拟机

# XShell的使用方法：http://www.jianshu.com/p/ada93cba0acd

```

---

## MySQL服务

### 安装软件包

```

# 此处会提示用户设置数据库密码

apt-get install mariadb-server python-pymysql

```

### 配置openstack.cnf

```

vim /etc/mysql/conf.d/openstack.cnf

```

```

[mysqld]

# controller的IP

bind-address = 192.168.10.55

default-storage-engine = innodb

innodb\_file\_per\_table

max\_connections = 4096

collation-server = utf8\_general\_ci

character-set-server = utf8

```

### 重启mysql服务

```

service mysql restart

```

### mysql安全初始化

```

# 提示输入密码，问题推荐输入n、y、y、y、y

mysql\_secure\_installation

```

---

## Rabbitmq服务

### 安装软件包

```

apt-get install rabbitmq-server

```

### 添加OpenStack用户

```

# 此处密码为0901

rabbitmqctl add\_user openstack 0901

```

### 为OpenStack用户添加读、写及访问权限

```

rabbitmqctl set\_permissions openstack ".\*" ".\*" ".\*"

```

---

## Memcached服务

### 安装软件包

```

apt-get install memcached python-memcache

```

### 配置memcached.conf

```

vim /etc/memcached.conf

```

```

# controller的IP地址

-l 192.168.10.55

```

### 重启服务

```

service memcached restart

```

---

## keystone的安装

### 进入数据库

```

# 提示输入数据库密码

mysql -u root -p

```

### 创建keystone数据库

```

CREATE DATABASE keystone;

```

### 赋予keystone相关权限

```

# 根据实际情况修改密码

GRANT ALL PRIVILEGES ON keystone.\* TO 'keystone'@'localhost' IDENTIFIED BY '0901';

GRANT ALL PRIVILEGES ON keystone.\* TO 'keystone'@'%' IDENTIFIED BY '0901';

```

### 退出数据库

```

exit

```

### 生成随机值作为临时令牌(token)

```

# token：0c7030a400cf77890c75

# token值要与后文统一

openssl rand -hex 10 自己的：28bd28393b478ed5331d

```

### 禁用keystone在安装完成后自启

```

echo "manual" > /etc/init/keystone.override

```

### 安装软件包

```

apt-get install keystone apache2 libapache2-mod-wsgi

```

### 配置keystone.conf

```

vim /etc/keystone/keystone.conf

```

```

[DEFAULT]

# token：0c7030a400cf77890c75

# token值要与后文统一

admin\_token = 0c7030a400cf77890c75

[database]

# 注释掉原connection

# 根据实际情况修改密码

connection = mysql+pymysql://keystone:0901@object/keystone

# 在第1987行

[token]

provider = fernet

```

### 同步keystone数据库

```

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

```

### 初始化Fernet键

```

keystone-manage fernet\_setup --keystone-user keystone --keystone-group keystone

```

### 配置apache2.conf

```

vim /etc/apache2/apache2.conf

```

```

# 添加该项

ServerName object

```

### 新建并配置wsgi-keystone.conf

```

vim /etc/apache2/sites-available/wsgi-keystone.conf

```

```

Listen 5000

Listen 35357

<VirtualHost \*:5000>

WSGIDaemonProcess keystone-public processes=5 threads=1 user=keystone group=keystone display-name=%{GROUP}

WSGIProcessGroup keystone-public

WSGIScriptAlias / /usr/bin/keystone-wsgi-public

WSGIApplicationGroup %{GLOBAL}

WSGIPassAuthorization On

ErrorLogFormat "%{cu}t %M"

ErrorLog /var/log/apache2/keystone.log

CustomLog /var/log/apache2/keystone\_access.log combined

<Directory /usr/bin>

Require all granted

</Directory>

</VirtualHost>

<VirtualHost \*:35357>

WSGIDaemonProcess keystone-admin processes=5 threads=1 user=keystone group=keystone display-name=%{GROUP}

WSGIProcessGroup keystone-admin

WSGIScriptAlias / /usr/bin/keystone-wsgi-admin

WSGIApplicationGroup %{GLOBAL}

WSGIPassAuthorization On

ErrorLogFormat "%{cu}t %M"

ErrorLog /var/log/apache2/keystone.log

CustomLog /var/log/apache2/keystone\_access.log combined

<Directory /usr/bin>

Require all granted

</Directory>

</VirtualHost>

```

### 使apache支持虚拟机的身份认证服务

```

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

```

### 重启appache服务

```

service apache2 restart

```

### 删除SQLite数据库文件

```

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

```

### 配置身份验证令牌

```

# token值要与前文统一

export OS\_TOKEN=0c7030a400cf77890c75

```

### 配置Endpoint的URL

```

export OS\_URL=http://object:35357/v3

```

### 配置API版本

```

export OS\_IDENTITY\_API\_VERSION=3

```

### 创建identity服务实体

```

# 执行结果为表格

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

```

### 创建identity服务endpoint

```

# 执行结果为表格

openstack endpoint create --region RegionOne identity public http://object:5000/v3

openstack endpoint create --region RegionOne identity internal http://object:5000/v3

openstack endpoint create --region RegionOne identity admin http://object:35357/v3

```

### 创建一个默认的domain

```

# 执行结果为表格

openstack domain create --description "Default Domain" default

```

### 创建一个admin project

```

# 执行结果为表格

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

```

### 创建一个admin user

```

# 此处会提示用户设置用户密码

# 执行结果为表格

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

```

### 创建一个admin role

```

# 执行结果为表格

openstack role create admin

```

### 将role添加到admin project和admin user里面去

```

# 此处无输出则执行正确

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

```

### 创建一个service project

```

# 执行结果为表格

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

```

### 配置keystone-paste.ini

```

vim /etc/keystone/keystone-paste.ini

```

```

# 分别从[pipeline:public\_api]、[pipeline:admin\_api] and [pipeline:api\_v3] 移除 admin\_token\_auth

```

### 移除临时token

```

unset OS\_TOKEN OS\_URL

```

### 作为admin管理员请求一个身份验证令牌

```

# 提示输入admin的密码

# 执行结果为表格

openstack --os-auth-url http://object:35357/v3 --os-project-domain-name default --os-user-domain-name default --os-project-name admin --os-username admin token issue

```

### 简化操作

```

# 将环境变量写入配置文件

# 简化每次重启主机后需加载脚本的操作

# 直接在命令行执行以下命令,再遇到需要加载脚本时就不需要执行了

# 0901为密码

echo "export OS\_PROJECT\_DOMAIN\_NAME=default" >> /etc/profile

echo "export OS\_USER\_DOMAIN\_NAME=default" >> /etc/profile

echo "export OS\_PROJECT\_NAME=admin" >> /etc/profile

echo "export OS\_USERNAME=admin" >> /etc/profile

echo "export OS\_PASSWORD=0901" >> /etc/profile #此处为admin密码

echo "export OS\_AUTH\_URL=http://object:35357/v3" >> /etc/profile

echo "export OS\_IDENTITY\_API\_VERSION=3" >> /etc/profile

echo "export OS\_IMAGE\_API\_VERSION=2" >> /etc/profile

```

### 重新加载配置文件

```

source /etc/profile

```

### 请求获取令牌

```

openstack token issue

```

---

## Swift单节点安装

### 创建swift用户

```

# 此处会提示用户设置用户密码

# 执行结果为表格

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

```

### 将admin role添加到swift user

```

# 此处无输出则正确

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

```

### 创建Object Storage服务实体

```

# 执行结果为表格

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

```

### 创建Object Storage服务endpoint

```

openstack endpoint create --region RegionOne object-store public http://object:8080/v1/AUTH\_%\(tenant\_id\)s

openstack endpoint create --region RegionOne object-store internal http://object:8080/v1/AUTH\_%\(tenant\_id\)s

openstack endpoint create --region RegionOne object-store admin http://object:8080/v1

```

### 安装软件包

```

apt-get install swift swift-proxy python-swiftclient python-keystoneclient python-keystonemiddleware memcached

```

### 创建swift目录

```

mkdir -p /etc/swift

```

### 从对象存储源仓库中获取代理服务配置文件

```

# 耐心等待,可能获取失败

curl -o /etc/swift/proxy-server.conf https://git.openstack.org/cgit/openstack/swift/plain/etc/proxy-server.conf-sample?h=mitaka-eol

```

### 配置proxy-server.conf

```

vim /etc/swift/proxy-server.conf

```

```

[DEFAULT]

bind\_port = 8080

user = swift

swift\_dir = /etc/swift

# 从[pipeline:main]中移除tempurl和tempauth,添加authtoken和keystoneauth,请不要改变模块的顺序;

[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

[app:proxy-server]

use = egg:swift#proxy

account\_autocreate = True

# 配置文件中有,但被注释掉了,直接添加即可

[filter:keystoneauth]

use = egg:swift#keystoneauth

operator\_roles = admin,user

# 配置文件中有,但被注释掉了,直接添加即可

[filter:authtoken]

paste.filter\_factory = keystonemiddleware.auth\_token:filter\_factory

auth\_uri = http://object:5000

auth\_url = http://object:35357

memcached\_servers = object:11211

auth\_type = password

project\_domain\_name = default

user\_domain\_name = default

project\_name = service

username = swift

password = 0901

delay\_auth\_decision = True

[filter:cache]

use = egg:swift#memcache

memcache\_servers = object:11211

```

### 磁盘模拟存储节点

```

# 模拟两个存储节点,每个节点2个空磁盘

# 关闭虚拟机，为我们的虚拟机添加4个10G的空磁盘;

# 虚拟机磁盘名称：sda(系统区)、sdb、sdc、sdd、sde;

# 验证检查,查看是否有以上磁盘;

ls /dev/sd\*

```

![步骤1](http://upload-images.jianshu.io/upload\_images/1152061-93a5993cb0be3eed.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240)

![步骤2](http://upload-images.jianshu.io/upload\_images/1152061-fa072d5d32810f55.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240)

![步骤3](http://upload-images.jianshu.io/upload\_images/1152061-dcbb804a8f43d6b7.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240)

![步骤4](http://upload-images.jianshu.io/upload\_images/1152061-00d315392cf298e5.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240)

![步骤5](http://upload-images.jianshu.io/upload\_images/1152061-bc61ee842e645eb1.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240)

![步骤6](http://upload-images.jianshu.io/upload\_images/1152061-e87f4bdf493a8052.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240)

![步骤7](http://upload-images.jianshu.io/upload\_images/1152061-cef6048d19cbe2f3.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240)

### 安装软件包

```

apt-get install xfsprogs rsync

```

### 格式化空磁盘

```

mkfs.xfs /dev/sdb

mkfs.xfs /dev/sdc

mkfs.xfs /dev/sdd

mkfs.xfs /dev/sde

```

### 创建挂载点目录结构

```

mkdir -p /srv/node/sdb

mkdir -p /srv/node/sdc

mkdir -p /srv/node/sdd

mkdir -p /srv/node/sde

```

### 配置fstab(自动挂载)

```

vim /etc/fstab

```

```

# 以下内容追加到配置文件

/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

/dev/sdd /srv/node/sdd xfs noatime,nodiratime,nobarrier,logbufs=8 0 2

/dev/sde /srv/node/sde xfs noatime,nodiratime,nobarrier,logbufs=8 0 2

```

### 挂载设备

```

mount /srv/node/sdb

mount /srv/node/sdc

mount /srv/node/sdd

mount /srv/node/sde

```

### 配置rsyncd.conf

```

vim /etc/rsyncd.conf

```

```

uid = swift

gid = swift

log file = /var/log/rsyncd.log

pid file = /var/run/rsyncd.pid

# 本机 IP 地址

address = 192.168.10.55

[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

```

### 配置开启rsync服务

```

vim /etc/default/rsync

```

```

RSYNC\_ENABLE=true

```

### 启动rsyns服务

```

service rsync start

```

### 安装软件包

```

apt-get install swift swift-account swift-container swift-object

```

### 获取配置文件

```

# 耐心等待,可能获取失败

curl -o /etc/swift/account-server.conf https://git.openstack.org/cgit/openstack/swift/plain/etc/account-server.conf-sample?h=mitaka-eol

curl -o /etc/swift/container-server.conf https://git.openstack.org/cgit/openstack/swift/plain/etc/container-server.conf-sample?h=mitaka-eol

curl -o /etc/swift/object-server.conf https://git.openstack.org/cgit/openstack/swift/plain/etc/object-server.conf-sample?h=mitaka-eol

```

### 配置account-server.conf

```

vim /etc/swift/account-server.conf

```

```

[DEFAULT]

# 本机 IP 地址

bind\_ip = 192.168.10.55

bind\_port = 6002

user = swift

swift\_dir = /etc/swift

devices = /srv/node

mount\_check = True

[pipeline:main]

pipeline = healthcheck recon account-server

[filter:recon]

use = egg:swift#recon

recon\_cache\_path = /var/cache/swift

```

### 配置container-server.conf

```

vim /etc/swift/container-server.conf

```

```

[DEFAULT]

# 本机 IP 地址

bind\_ip = 192.168.10.55

bind\_port = 6001

user = swift

swift\_dir = /etc/swift

devices = /srv/node

mount\_check = True

[pipeline:main]

pipeline = healthcheck recon container-server

[filter:recon]

use = egg:swift#recon

recon\_cache\_path = /var/cache/swift

```

### 配置object-server.conf

```

vim /etc/swift/object-server.conf

```

```

[DEFAULT]

# 本机 IP 地址

bind\_ip = 192.168.10.55

bind\_port = 6000

user = swift

swift\_dir = /etc/swift

devices = /srv/node

mount\_check = True

[pipeline:main]

pipeline = healthcheck recon object-server

[filter:recon]

use = egg:swift#recon

recon\_cache\_path = /var/cache/swift

recon\_lock\_path = /var/lock

```

### 修改挂载点的权限

```

chown -R swift:swift /srv/node

```

### 创建recon目录并设置权限

```

mkdir -p /var/cache/swift

chown -R root:swift /var/cache/swift

chmod -R 775 /var/cache/swift

```

---

## 创建并分配初始化环(rings)

### 切换到swift目录

```

cd /etc/swift

```

### 创建account.builder文件

```

# 此处无输出则正确

swift-ring-builder account.builder create 10 3 1

```

### 将每个存储节点添加到环(ring)中

```

swift-ring-builder account.builder add --region 1 --zone 1 --ip 192.168.10.55 --port 6002 --device sdb --weight 100

swift-ring-builder account.builder add --region 1 --zone 1 --ip 192.168.10.55 --port 6002 --device sdc --weight 100

swift-ring-builder account.builder add --region 1 --zone 2 --ip 192.168.10.55 --port 6002 --device sdd --weight 100

swift-ring-builder account.builder add --region 1 --zone 2 --ip 192.168.10.55 --port 6002 --device sde --weight 100

```

### 验证操作

```

swift-ring-builder account.builder

```

### 平衡环

```

swift-ring-builder account.builder rebalance

```

### 切换到swift目录

```

cd /etc/swift

```

### 创建container.builder文件

```

# 此处无输出则正确

swift-ring-builder container.builder create 10 3 1

```

### 将每个存储节点添加到环(ring)中

```

swift-ring-builder container.builder add --region 1 --zone 1 --ip 192.168.10.55 --port 6001 --device sdb --weight 100

swift-ring-builder container.builder add --region 1 --zone 1 --ip 192.168.10.55 --port 6001 --device sdc --weight 100

swift-ring-builder container.builder add --region 1 --zone 2 --ip 192.168.10.55 --port 6001 --device sdd --weight 100

swift-ring-builder container.builder add --region 1 --zone 2 --ip 192.168.10.55 --port 6001 --device sde --weight 100

```

### 验证操作

```

swift-ring-builder container.builder

```

### 平衡环

```

swift-ring-builder container.builder rebalance

```

### 切换到swift目录

```

cd /etc/swift

```

### 创建object.builder文件

```

# 此处无输出则正确

swift-ring-builder object.builder create 10 3 1

```

### 将每个存储节点添加到环(ring)中

```

swift-ring-builder object.builder add --region 1 --zone 1 --ip 192.168.10.55 --port 6000 --device sdb --weight 100

swift-ring-builder object.builder add --region 1 --zone 1 --ip 192.168.10.55 --port 6000 --device sdc --weight 100

swift-ring-builder object.builder add --region 1 --zone 2 --ip 192.168.10.55 --port 6000 --device sdd --weight 100

swift-ring-builder object.builder add --region 1 --zone 2 --ip 192.168.10.55 --port 6000 --device sde --weight 100

```

### 验证操作

```

swift-ring-builder object.builder

```

### 平衡环

```

swift-ring-builder object.builder rebalance

```

### 从源仓库获取swift.conf

```

# 耐心等待,可能获取失败

curl -o /etc/swift/swift.conf https://git.openstack.org/cgit/openstack/swift/plain/etc/swift.conf-sample ?h=mitaka-eol

```

### 配置swift.conf

```

vim /etc/swift/swift.conf

```

```

[swift-hash]

# suffix与prefix自定义

swift\_hash\_path\_suffix = Ben

swift\_hash\_path\_prefix = Ben

[storage-policy:0]

name = Policy-0

default = yes

```

### 设置权限

```

chown -R root:swift /etc/swift

```

### 重启服务

```

service memcached restart

service swift-proxy restart

swift-init all start

```

### 查看swift状态

```

swift stat

```

### 创建容器Ben

```

openstack container create Ben

```

### 上传测试文件到容器Ben

```

# 文件需要我们自行去创建

# 注意 FILENAME 的修改

openstack object create Ben FILENAME

```

### 列出容器 Ben 存储的FILES

```

openstack object list Ben

```

### 下载容器Ben存储的FILENAME

```

# 此处无输出则正确

openstack object save Ben FILENAME

```