# Swift all in one 搭建教程

**目录**

[Swift all in one 搭建教程 1](#_Toc530406251)

[1 搭建环境 1](#_Toc530406252)

[2 使用回环设备存储 2](#_Toc530406253)

[3 获取代码 3](#_Toc530406254)

[4 安装rsync 4](#_Toc530406255)

[5 启动memcached 5](#_Toc530406256)

[6 独立的日志，启动rsyslog 5](#_Toc530406257)

[7 配置每一个节点 7](#_Toc530406258)

[8 创建运行Swift的脚本 7](#_Toc530406259)

[9 启动Swift的主线程,分别包(proxy,  account, container, 和 object)这几个进程 11](#_Toc530406260)

[10获取X-Storage-Url和X-Auth-Token 11](#_Toc530406261)

[11检查能否获得account 12](#_Toc530406262)

[12检查python-swiftclient提供的swift命令是否可以用 12](#_Toc530406263)

[问题记录： 13](#_Toc530406264)

## 1 搭建环境

#### 硬件

本次搭建机器操作系统CentOS Linux release 7.2.1511 (Core)，内存4G，磁盘60G。

#### 软件

使用[Openstack Pike](https://mirrors.tuna.tsinghua.edu.cn/centos/7/cloud/x86_64/openstack-pike/)版本，配置可用的Openstack源，使用了清华开源镜像。配置服务器镜像：

1、cd /etc/yum.repos.d/  
2、vim CentOS-Base.repo

增加如下配置

[openstack]  
name=Openstack  
baseurl=https://mirrors.tuna.tsinghua.edu.cn/centos/7/cloud/x86\_64/openstack-pike/  
gpgcheck=0

使用YUM跟新库

yum update -y

#### 安装依赖

sudo yum update  
sudo yum install curl gcc memcached rsync sqlite xfsprogs git-core \  
 libffi-devel xinetd liberasurecode-devel \  
 openssl-devel python-setuptools \  
 python-coverage python-devel python-nose \  
 pyxattr python-eventlet \  
 python-greenlet python-paste-deploy \  
 python-netifaces python-pip python-dns \  
 python-mock

接下来选择使用[分区存储](https://blog.csdn.net/hubinqiang/article/details/57359565#partition-for-storage)(partition for storage)或使用[回环设备](https://blog.csdn.net/hubinqiang/article/details/57359565#loopback-device-for-storage)(loopback device for storage)进行存储，因为本次搭建是在一台机器中，也就是all in one模式，所以采用回环设备进行存储。

## 2 使用回环设备存储

注意：${USER}:${USER}一般是swift:swift

#### 1.使用回环设备作为存储

sudo cd /srv  
sudo truncate -s 10GB /srv/swift-disk  
sudo mkfs.xfs /srv/swift-disk

修改truncate命令中指定的大小以根据需要创建更大或更小的分区。

#### 2.编辑修改分区表 /etc/fstab 并添加

vim /etc/fstab

添加

/dev/sdb1 /mnt/sdb1 xfs noatime,nodiratime,nobarrier,logbufs=8 0 0

#### 3.创建装载点和链接

sudo mkdir /mnt/sdb1 #创建  
sudo mount /mnt/sdb1 #挂载  
sudo mkdir /mnt/sdb1/1 /mnt/sdb1/2 /mnt/sdb1/3 /mnt/sdb1/4 #创建4个目录

sudo chown ${USER}:${USER} /mnt/sdb1/\*

注释：修改所有组、所有者，一般是swift:swift

for x in {1..4}; do sudo ln -s /mnt/sdb1/$x /srv/$x; done #链接

sudo mkdir -p /srv/1/node/sdb1 /srv/1/node/sdb5 \  
 /srv/2/node/sdb2 /srv/2/node/sdb6 \  
 /srv/3/node/sdb3 /srv/3/node/sdb7 \  
 /srv/4/node/sdb4 /srv/4/node/sdb8 \  
 /var/run/swift

sudo chown -R ${USER}:${USER} /var/run/swift #修改

for x in {1..4}; do sudo chown -R ${USER}:${USER} /srv/$x/; done

备注：for x in {1..4}; do sudo chown -R ${USER}:${USER} /srv/$x/; done 一定要执行，否则下面权限报错。

添加以下几行到/etc/rc.local中(在exit 0之前)

mkdir -p /var/cache/swift1 /var/cache/swift2 /var/cache/swift3 /var/cache/swift4  
chown swift:swift /var/cache/swift\*  
mkdir -p /var/run/swift  
chown swift:swift /var/run/swift

## 3 获取代码

#### 1.下载python-swiftclient代码

cd $HOME; git clone https://github.com/openstack/python-swiftclient.git

#### 2.编译并安装python-swiftclient

git clone https://github.com/openstack/swift.git

#### 3.下载swift的代码

git clone https://github.com/openstack/swift.git

#### 4.编译并安装swift

cd $HOME/swift; sudo pip install -r requirements.txt; sudo python setup.py develop; cd -

#### 5.安装swift测试相关的依赖

cd $HOME/swift; sudo pip install -r test-requirements.txt

## 4 安装rsync

#### 创建/etc/rsyncd.conf:

cp $HOME/swift/doc/saio/rsyncd.conf /etc/  
sudo sed -i "s/<your-user-name>/swift/" /etc/rsyncd.conf

**修改rsyncd.conf文件**

uid = <your-user-name>  
gid = <your-user-name>  
更改为  
uid = swift  
gid = swift

#### 在系统的SELinux是Enforcing模式设置为Permissive

sudo setenforce Permissive

#### 启动rsync

systemctl start rsyncd.service

#### 验证rsync是不是接受了所有的连接

rsync rsync://pub@localhost/

如果成功会看到

account6012   
account6022   
account6032   
account6042   
container6011   
container6021   
container6031   
container6041   
object6010   
object6020   
object6030   
object6040

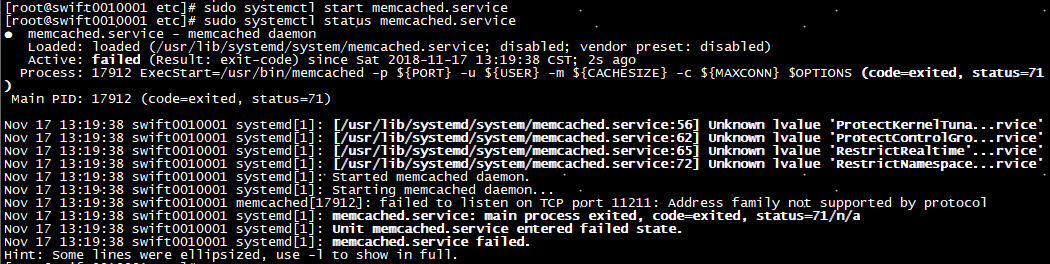
## 5 启动memcached

systemctl start memcached.service

查看服务状态  
systemctl status memcached.service

发现报错了，如下：

failed to listen on TCP port 11211: Address family not supported by protocol

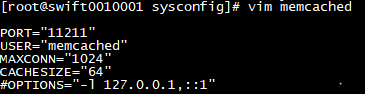


具体原因可以查看：

<https://serverfault.com/questions/286381/memcached-problems-failed-to-listen-on-tcp-port-11211>

<http://www.bubuko.com/infodetail-2039138.html>

解决办法是：删除/etc/sysconfig/memcached里面option，保存运行



再次启动查看，服务正常启动

systemctl start memcached.service  
systemctl status memcached.service

## 独立的日志，启动rsyslog

#### 安装swift rsyslogd的配置

cp $HOME/swift/doc/saio/rsyslog.d/10-swift.conf /etc/rsyslog.d/

查看该conf文件以确定是否要将日志独立出来，以及是否需要每小时处理统计信息的日志。 为方便起见，我们提供以下默认内容：在/etc/rsyslog.d/ 10-swift.conf可以查看，默认如下

# Uncomment the following to have a log containing all logs together  
#local1,local2,local3,local4,local5.\* /var/log/swift/all.log  
  
# Uncomment the following to have hourly proxy logs for stats processing  
#$template HourlyProxyLog,"/var/log/swift/hourly/%$YEAR%%$MONTH%%$DAY%%$HOUR%"  
#local1.\*;local1.!notice ?HourlyProxyLog  
  
local1.\*;local1.!notice /var/log/swift/proxy.log  
local1.notice /var/log/swift/proxy.error  
local1.\* ~  
  
local2.\*;local2.!notice /var/log/swift/storage1.log  
local2.notice /var/log/swift/storage1.error  
local2.\* ~  
  
local3.\*;local3.!notice /var/log/swift/storage2.log  
local3.notice /var/log/swift/storage2.error  
local3.\* ~  
  
local4.\*;local4.!notice /var/log/swift/storage3.log  
local4.notice /var/log/swift/storage3.error  
local4.\* ~  
  
local5.\*;local5.!notice /var/log/swift/storage4.log  
local5.notice /var/log/swift/storage4.error  
local5.\* ~  
  
local6.\*;local6.!notice /var/log/swift/expirer.log  
local6.notice /var/log/swift/expirer.error  
local6.\* ~

#### 编辑 /etc/rsyslog.conf 并进行以下修改

在“GLOBAL DIRECTIVES” 栏下添加

$PrivDropToGroup adm

#### 如果使用的是hourly logs则执行下面的

sudo mkdir -p /var/log/swift/hourly

否则执行：

sudo mkdir -p /var/log/swift

#### 创建日志目录，启动syslog

sudo chown -R root:adm /var/log/swift  
sudo chmod -R g+w /var/log/swift  
sudo systemctl restart rsyslog.service #重启

## 配置每一个节点

#### 1. 删除已经存在的swift目录(可选)

rm -rf /etc/swift

#### 2. 往/etc/swift中加配置文件

cd $HOME/swift/doc; sudo cp -r saio/swift /etc/swift; cd -  
sudo chown -R ${USER}:${USER} /etc/swift

具体修改后配置文件结果可以在/etc/swift目录下查看

#### 3. 在Swift配置文件中更新<your-user-name>引用

find /etc/swift/ -name \\*.conf | xargs sudo sed -i "s/<your-user-name>/${USER}/"

注意这里面<your-user-name>表示你的用户名，${USER}用swift代替

## 创建运行Swift的脚本

#### 复制SAIO脚本，重新设置环境

mkdir -p $HOME/bin  
cd $HOME/swift/doc; cp saio/bin/\* $HOME/bin; cd -  
chmod +x $HOME/bin/\*

#### 编辑$HOME/bin/resetswift脚本

cd $HOME/bin/resetswift

模板resetswift脚本如下所示：

#!/bin/bash  
  
set -e  
  
swift-init all kill  
# Remove the following line if you did not set up rsyslog for individual logging:  
sudo find /var/log/swift -type f -exec rm -f {} \;  
if cut -d' ' -f2 /proc/mounts | grep -q /mnt/sdb1 ; then  
 sudo umount /mnt/sdb1  
fi  
# If you are using a loopback device set SAIO\_BLOCK\_DEVICE to "/srv/swift-disk"  
sudo mkfs.xfs -f ${SAIO\_BLOCK\_DEVICE:-/dev/sdb1}  
sudo mount /mnt/sdb1  
sudo mkdir /mnt/sdb1/1 /mnt/sdb1/2 /mnt/sdb1/3 /mnt/sdb1/4  
sudo chown ${USER}:${USER} /mnt/sdb1/\*  
mkdir -p /srv/1/node/sdb1 /srv/1/node/sdb5 \  
 /srv/2/node/sdb2 /srv/2/node/sdb6 \  
 /srv/3/node/sdb3 /srv/3/node/sdb7 \  
 /srv/4/node/sdb4 /srv/4/node/sdb8  
sudo rm -f /var/log/debug /var/log/messages /var/log/rsyncd.log /var/log/syslog  
find /var/cache/swift\* -type f -name \*.recon -exec rm -f {} \;  
if [ "`type -t systemctl`" == "file" ]; then  
 sudo systemctl restart rsyslog  
 sudo systemctl restart memcached  
else  
 sudo service rsyslog restart  
 sudo service memcached restart  
fi

如果使用回环设备，请使用/srv/swift-disk添加环境变量var替换/dev/sdb1：

echo "export SAIO\_BLOCK\_DEVICE=/srv/swift-disk" >> $HOME/.bashrc

如果没有为单个日志记录设置rsyslog，请删除find /var/log/swift ...行

sed -i "/find \/var\/log\/swift/d" $HOME/bin/resetswift

#### 安装运行测试的样本配置文件

cp $HOME/swift/test/sample.conf /etc/swift/test.conf

#### 添加运行测试用的环境变量

echo "export SWIFT\_TEST\_CONFIG\_FILE=/etc/swift/test.conf" >> $HOME/.bashrc

#### 确保把bin目录设置到环境变量中去

echo "export PATH=${PATH}:$HOME/bin" >> $HOME/.bashrc

#### source 刚刚设置环境变量，使其生效

. $HOME/.bashrc

#### 使用提供的脚本初始化rings

cd $HOME/bin/下看淡remarkings脚本，执行该脚本 **. remarkings**

#!/bin/bash  
set -e

cd /etc/swift  
rm -f \*.builder \*.ring.gz backups/\*.builder backups/\*.ring.gz  
swift-ring-builder object.builder create 10 3 1  
swift-ring-builder object.builder add r1z1-127.0.0.1:6010/sdb1 1  
swift-ring-builder object.builder add r1z2-127.0.0.1:6020/sdb2 1  
swift-ring-builder object.builder add r1z3-127.0.0.1:6030/sdb3 1  
swift-ring-builder object.builder add r1z4-127.0.0.1:6040/sdb4 1  
swift-ring-builder object.builder rebalance  
swift-ring-builder object-1.builder create 10 2 1  
swift-ring-builder object-1.builder add r1z1-127.0.0.1:6010/sdb1 1  
swift-ring-builder object-1.builder add r1z2-127.0.0.1:6020/sdb2 1  
swift-ring-builder object-1.builder add r1z3-127.0.0.1:6030/sdb3 1  
swift-ring-builder object-1.builder add r1z4-127.0.0.1:6040/sdb4 1  
swift-ring-builder object-1.builder rebalance  
swift-ring-builder object-2.builder create 10 6 1  
swift-ring-builder object-2.builder add r1z1-127.0.0.1:6010/sdb1 1  
swift-ring-builder object-2.builder add r1z1-127.0.0.1:6010/sdb5 1  
swift-ring-builder object-2.builder add r1z2-127.0.0.1:6020/sdb2 1  
swift-ring-builder object-2.builder add r1z2-127.0.0.1:6020/sdb6 1  
swift-ring-builder object-2.builder add r1z3-127.0.0.1:6030/sdb3 1  
swift-ring-builder object-2.builder add r1z3-127.0.0.1:6030/sdb7 1  
swift-ring-builder object-2.builder add r1z4-127.0.0.1:6040/sdb4 1  
swift-ring-builder object-2.builder add r1z4-127.0.0.1:6040/sdb8 1  
swift-ring-builder object-2.builder rebalance  
swift-ring-builder container.builder create 10 3 1  
swift-ring-builder container.builder add r1z1-127.0.0.1:6011/sdb1 1  
swift-ring-builder container.builder add r1z2-127.0.0.1:6021/sdb2 1  
swift-ring-builder container.builder add r1z3-127.0.0.1:6031/sdb3 1  
swift-ring-builder container.builder add r1z4-127.0.0.1:6041/sdb4 1  
swift-ring-builder container.builder rebalance  
swift-ring-builder account.builder create 10 3 1  
swift-ring-builder account.builder add r1z1-127.0.0.1:6012/sdb1 1  
swift-ring-builder account.builder add r1z2-127.0.0.1:6022/sdb2 1  
swift-ring-builder account.builder add r1z3-127.0.0.1:6032/sdb3 1  
swift-ring-builder account.builder add r1z4-127.0.0.1:6042/sdb4 1  
swift-ring-builder account.builder rebalance

执行后输出：

Device d0r1z1-127.0.0.1:6010R127.0.0.1:6010/sdb1\_"" with 1.0 weight got id 0  
Device d1r1z2-127.0.0.2:6020R127.0.0.2:6020/sdb2\_"" with 1.0 weight got id 1  
Device d2r1z3-127.0.0.3:6030R127.0.0.3:6030/sdb3\_"" with 1.0 weight got id 2  
Device d3r1z4-127.0.0.4:6040R127.0.0.4:6040/sdb4\_"" with 1.0 weight got id 3  
Reassigned 3072 (300.00%) partitions. Balance is now 0.00. Dispersion is now 0.00  
Device d0r1z1-127.0.0.1:6010R127.0.0.1:6010/sdb1\_"" with 1.0 weight got id 0  
Device d1r1z2-127.0.0.2:6020R127.0.0.2:6020/sdb2\_"" with 1.0 weight got id 1  
Device d2r1z3-127.0.0.3:6030R127.0.0.3:6030/sdb3\_"" with 1.0 weight got id 2  
Device d3r1z4-127.0.0.4:6040R127.0.0.4:6040/sdb4\_"" with 1.0 weight got id 3  
Reassigned 2048 (200.00%) partitions. Balance is now 0.00. Dispersion is now 0.00  
Device d0r1z1-127.0.0.1:6010R127.0.0.1:6010/sdb1\_"" with 1.0 weight got id 0  
Device d1r1z1-127.0.0.1:6010R127.0.0.1:6010/sdb5\_"" with 1.0 weight got id 1  
Device d2r1z2-127.0.0.2:6020R127.0.0.2:6020/sdb2\_"" with 1.0 weight got id 2  
Device d3r1z2-127.0.0.2:6020R127.0.0.2:6020/sdb6\_"" with 1.0 weight got id 3  
Device d4r1z3-127.0.0.3:6030R127.0.0.3:6030/sdb3\_"" with 1.0 weight got id 4  
Device d5r1z3-127.0.0.3:6030R127.0.0.3:6030/sdb7\_"" with 1.0 weight got id 5  
Device d6r1z4-127.0.0.4:6040R127.0.0.4:6040/sdb4\_"" with 1.0 weight got id 6  
Device d7r1z4-127.0.0.4:6040R127.0.0.4:6040/sdb8\_"" with 1.0 weight got id 7  
Reassigned 6144 (600.00%) partitions. Balance is now 0.00. Dispersion is now 0.00  
Device d0r1z1-127.0.0.1:6011R127.0.0.1:6011/sdb1\_"" with 1.0 weight got id 0  
Device d1r1z2-127.0.0.2:6021R127.0.0.2:6021/sdb2\_"" with 1.0 weight got id 1  
Device d2r1z3-127.0.0.3:6031R127.0.0.3:6031/sdb3\_"" with 1.0 weight got id 2  
Device d3r1z4-127.0.0.4:6041R127.0.0.4:6041/sdb4\_"" with 1.0 weight got id 3  
Reassigned 3072 (300.00%) partitions. Balance is now 0.00. Dispersion is now 0.00  
Device d0r1z1-127.0.0.1:6012R127.0.0.1:6012/sdb1\_"" with 1.0 weight got id 0  
Device d1r1z2-127.0.0.2:6022R127.0.0.2:6022/sdb2\_"" with 1.0 weight got id 1  
Device d2r1z3-127.0.0.3:6032R127.0.0.3:6032/sdb3\_"" with 1.0 weight got id 2  
Device d3r1z4-127.0.0.4:6042R127.0.0.4:6042/sdb4\_"" with 1.0 weight got id 3  
Reassigned 3072 (300.00%) partitions. Balance is now 0.00. Dispersion is now 0.00

## 启动Swift的主线程,分别包(proxy,  account, container, 和 object)这几个进程

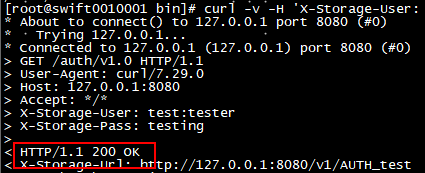
cd /root/bin

执行startmain脚本 startmain

## 10 获取X-Storage-Url和X-Auth-Token

curl -v -H 'X-Storage-User: test:tester' -H 'X-Storage-Pass: testing' http://127.0.0.1:8080/auth/v1.0

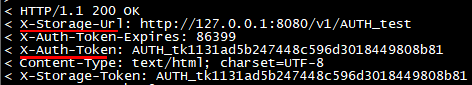
如果显示如下200 ok表示成功



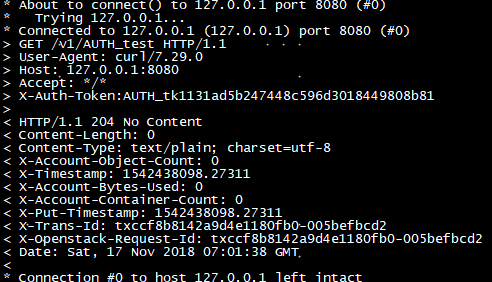
## 检查能否获得account

curl -v -H 'X-Auth-Token: <token-from-x-auth-token-above>' < X-Auth-Token -from-x-storage-url-above>

其中X-Auth-Token和url，由上面步骤10中查看如下：

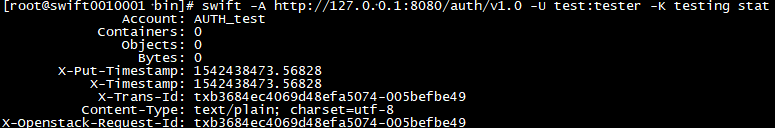


结果如下



## 12 检查python-swiftclient提供的swift命令是否可以用

swift -A http://127.0.0.1:8080/auth/v1.0 -U test:tester -K testing stat



表示正常启动

## 问题记录：

1、在创建容器时候报错，查看日志/var/log/swift/proxy.error有以下错误



解决办法在、etc/swift/proxy-server.conf的[DEFAULT]下面添加：

allow\_account\_management = true

account\_autocreate = true

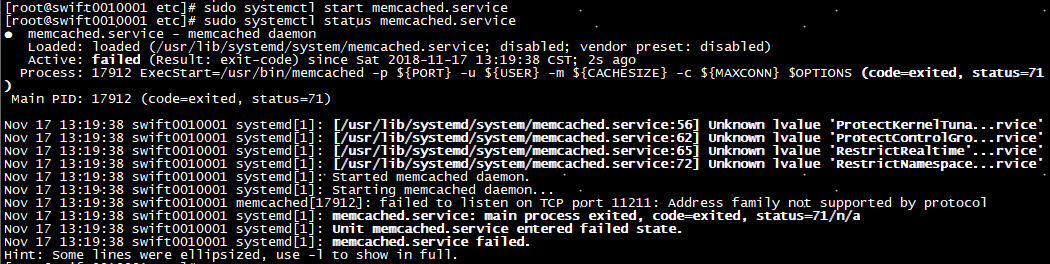
2 启动memcached

systemctl start memcached.service

查看服务状态  
systemctl status memcached.service

发现报错了，如下：

failed to listen on TCP port 11211: Address family not supported by protocol

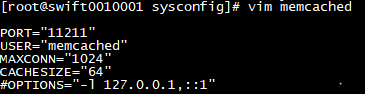


具体原因可以查看：

<https://serverfault.com/questions/286381/memcached-problems-failed-to-listen-on-tcp-port-11211>

<http://www.bubuko.com/infodetail-2039138.html>

解决办法是：删除/etc/sysconfig/memcached里面option，保存运行



再次启动查看，服务正常启动

systemctl start memcached.service  
systemctl status memcached.service