目录

[一、swift源码多机部署 2](#_Toc324600335)

[1 、为每一台机器安装依赖包 2](#_Toc324600336)

[2. 为每一台机器创建swift用户 2](#_Toc324600337)

[3. 下载源码（在swift用户下操作） 3](#_Toc324600338)

[4. 安装swauth和swift （在swift用户下操作） 4](#_Toc324600339)

[5. Proxy节点的设置 4](#_Toc324600340)

[6. 配置Storage节点 7](#_Toc324600341)

[7、 测试swift(在Proxy主机上测试) 13](#_Toc324600342)

[二、Swift调试环境搭建：eclipse+pydev 15](#_Toc324600343)

[1、安装jdk 15](#_Toc324600344)

[2、安装eclipse 15](#_Toc324600345)

[3、配置PyDev 15](#_Toc324600346)

[4、导入swift项目 16](#_Toc324600347)

## 一、swift源码多机部署

所有操作都是在Ubuntu Server 11.10 32位系统上通过验证。本文参考了Openstack-swift的Instructions for a Multiple Server SwiftInstallation (Ubuntu)文档,具体信息如下：

Linux系统版本：Ubuntu Server 11.10 32-bit

Proxy Server IP： 192.168.1.129

Storage Server One: 192.168.1.130

Storage Server Two: 192.168.1.131

Storage Server Three: 192.168.1.132

官方文档： www.openstack.org

参考文档： http://swift.openstack.org/howto\_installmultinode.html

Swift版本： 1.4.8

Swauth版本： 1.0.4

### 1 、为每一台机器安装依赖包

sudo apt-get --option Dpkg::Options::=--force-confold --assume-yes updatesudo apt-get install pep8 pylint python-pip screen unzip wget psmisc git-core lsof openssh-server vim-nox locate python-virtualenv python-unittest2 iputils-ping wget curl tcpdump euca2ools python-setuptools python-dev python-lxml python-pastescript python-pastedeploy python-paste sqlite3 python-pysqlite2 python-sqlalchemy python-mysqldb python-webob python-greenlet python-routes libldap2-dev libsasl2-dev python-bcrypt curl gcc python-configobj python-coverage python-dev python-eventlet python-greenlet python-netifaces python-nose python-pastedeploy python-setuptools python-simplejson python-webob python-xattr sqlite3 xfsprogs

### 2. 为每一台机器创建swift用户

sudo useradd -mk /home/swift/ -s /bin/bash swift

sudo passwd swift #为swift用户添加密码，在此我将其设为了swift

编辑/etc/sudoer文件，在文件末尾添加如下代码

swift ALL=(ALL) NOPASSWD:ALL

### 3. 下载源码（在swift用户下操作）

在Proxy机器中下载swauth和swift源码

su swift #切换到swift用户

sudo mkdir /home/swift/openstack #创建一个目录来存放

cd /home/swift/openstack

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

cd swift

git checkout 1.4.8 #使用1.4.8版本，在swift目录下你可以

#使用git tag命令查看有多少个版本

git clone https://github.com/gholt/swauth.git #下载swauth

cd swauth

git checkout 1.0.4

在每一台Storage节点的机器中下载swift

su swift #切换到swift用户

sudo mkdir /home/swift/openstack #创建一个目录来存放

cd /home/swift/openstack

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

cd swift

git checkout 1.4.8 #使用1.4.8版本，在swift目录下你可以使用git tag命令查看有多少个版本

### 4. 安装swauth和swift （在swift用户下操作）

在Proxy机器中安装swauth和swift

cd /home/swift/openstack/swauth

sudo python setup.py install --record file.txt

cd /home/swift/openstack/swift #在Proxy节点中安装swift是为了使用swift的swift-proxy-server

sudo python setup.py install --record file.txt

注：file.txt文件记录了安装信息，要想删除所安装的软件通过命令sudo cat file.txt | xargs rm -rf即可删除，这一步可能会出现禁止删除问题，因此你需要使用root用户来操作。

在每一台Storage节点的机器中安装swift

cd /home/swift/openstack/swift

sudo python setup.py install --record file.txt

### 5. Proxy节点的设置

1.sudo apt-get install memcached #安装缓存服务器

修改/etc/ memcached.conf文件，将-l 127.0.0.1改为-l 192.168.1.129（这里我是根据我自己的情况设定的，具体原因见第一点中的图）

sudo service memcached restart

sudo mkdir /etc/swift

cd /etc/swift

sudo chown -R swift:swift /etc/swift

cp /home/swift/openstack/swift/etc/proxy-server.conf /etc/swift/

cp /home/swift/openstack/swift/etc/swift.conf /etc/swift/

2.修改/etc/swift/proxy-server.conf文件，具体内容如下，原文件中没有的项需要自行增加

[DEFAULT]

bind\_port = 8080

user = swift

log\_facility = LOG\_LOCAL0

[pipeline:main]

pipeline = catch\_errors healthcheck cache ratelimit swauth proxy-server

[app:proxy-server]

use = egg:swift#proxy

allow\_account\_management = true

account\_autocreate = true

[filter:swauth]

use = egg:swauth#swauth

# Highly recommended to change this.

super\_admin\_key = swauthkey

default\_swift\_cluster = local#http://192.168.1.129:8080/auth/v1.0

[filter:healthcheck]

use = egg:swift#healthcheck

[filter:cache]

use = egg:swift#memcache

3.修改/etc/swift/swift.conf，红色部分是随意更改的，你可以根据自己的需要更改

[swift-hash]

swift\_hash\_path\_suffix = tuletech

4.生成相关ring以及builder文件，使用如下命令生成，红色部分是根据具体情况而更改的，具体原因见摘要说明里的图。每一台机器使用一个域（z1, z2, z3…依次递增）

sudo chown -R swift:swift /etc/swift/\*

cd /etc/swift

swift-ring-builder object.builder create 18 3 1

swift-ring-builder container.builder create 18 3 1

swift-ring-builder account.builder create 18 3 1

export HOST\_IP=192.168.1.130

swift-ring-builder object.builder add z1-${HOST\_IP}:6010/sdb1 100

swift-ring-builder container.builder add z1-${HOST\_IP}:6011/sdb1 100

swift-ring-builder account.builder add z1-${HOST\_IP}:6012/sdb1 100

export HOST\_IP=192.168.1.131

swift-ring-builder object.builder add z2-${HOST\_IP}:6010/sdb1 100

swift-ring-builder container.builder add z2-${HOST\_IP}:6011/sdb1 100

swift-ring-builder account.builder add z2-${HOST\_IP}:6012/sdb1 100

export HOST\_IP=192.168.1.132

swift-ring-builder object.builder add z3-${HOST\_IP}:6010/sdb1 100

swift-ring-builder container.builder add z3-${HOST\_IP}:6011/sdb1 100

swift-ring-builder account.builder add z3-${HOST\_IP}:6012/sdb1 100

swift-ring-builder object.builder rebalance

swift-ring-builder container.builder rebalance

swift-ring-builder account.builder rebalance

5.启动proxy服务

swift-init proxy start

### 6. 配置Storage节点

因为每个Storage节点的设置基本上是相似的，所以在这里只拿其中一个节点做示例（192.168.1.130），其他节点只需要重复一下几步操作就可以了

1.创建/etc/swift目录

sudo mkdir /etc/swift

sudo chown -R swift:swift /etc/swift/\*

2.将Proxy节点上/etc/swift/中的account.ring.gz container.ring.gz object.ring.gz swift.conf拷贝到当前存储节点（192.168.1.130）/etc/swift目录中，可使用如下命令

scp swift@192.168.1.129:/etc/swift/\*.ring.gz /etc/swift/

scp swift@192.168.1.129:/etc/swift/swift.conf /etc/swift/

sudo chown -R swift:swift /etc/swift/\*

3.更改/etc/rsyncd.conf文件，如果该文件不存在则需要自行创建，内容如下

uid = swift

gid = swift

log file = /var/log/rsyncd.log

pid file = /var/run/rsyncd.pid

address = 192.168.1.130 #这里也可以改为127.0.0.1

[account]

max connections = 2

path = /srv/node/

read only = false

lock file = /var/lock/account.lock

[conainer]

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

编辑/etc/default/rsync:

将RSYNC\_ENABLE设置为true

更改好之后，重启该服务

sudo service rsync restart

4.存储点的设置

这里有分两种情况来设置存储点

1.假设你的系统里有一个单独分区，使用此分区来做存储点，在这里假设系统中有/dev/sdb1（注：这里根据你自己系统的情况而定）这个分区未被使用，我们用它来做存储点。

sudo mkdir -p /srv/node/sdb1

sudo mkfs.xfs -i size=1024 /dev/sdb1 #以xfs方式格式化分区

sudo chmod a+w /etc/fstab

sudo echo “/dev/sdb1 /srv/node/sdb1 xfs noatime,nodiratime

nobarrier,logbufs=8 0 0” >> /etc/fstab #系统启动时自动挂载，这里的sdb1是一定不能改的，因为在做Proxy节点生成相应的ring文件时使用了sdb1 (swift-ring-builder object.builder add z1-${HOST\_IP}:6010/sdb1 100)的时候，加入需要更改则两个地方都需要改

sudo mount /srv/node/sdb1

sudo chown -R swift:swift /srv/node/sdb1

sudo chmod a+w -R /srv/node/sdb1

2.如果系统里没有单独的分区来做存储点，则需要创建一个临时分区来做存储点

sudo mkdir -p /srv/node/sdb1

sudo dd if=/dev/zero of=/srv/swift-disk bs=1024 count=0 seek=1000000 #这个命令是在/srv/下创建一个名为swift-disk的存储区，你可以改变seek的大小来改变swift-disk的大小

sudo mkfs.xfs -i size=1024 /srv/swift-disk

sudo chmod a+w /etc/fstab

sudo echo “/srv/swift-disk /srv/node/sdb1 xfs loop,noatime,nodiratime,nobarrier,logbufs=8 0 0” >> /etc/fstab #系统启动时自动挂载

sudo mount /srv/node/sdb1

sudo chown -R swift:swift /srv/node/sdb1

sudo chmod a+w -R /srv/node/sdb1

sudo chmod a+w /srv/swift-disk

创建相关的目录

sudo mkdir /var/run/swift

sudo chown swift:swift /var/run/swift

sudo chmod a+w /var/run/swift

在/etc/rc.local的exit 0之前加入下列三行

mkdir /var/run/swift

chown swift:swift /var/run/swift

chmod a+w /var/run/swift

5.Swift文件配置

创建/etc/swift/account-server.conf文件，并加入如下配置

[DEFAULT]

devices = /srv/node

mount\_check = false

bind\_port = 6012

user = swift

bind\_ip = 0.0.0.0

workers = 2

[pipeline:main]

pipeline = account-server

[app:account-server]

use = egg:swift#account

[account-replicator]

[account-auditor]

[account-reaper]

创建/etc/swift/object-server.conf文件

[DEFAULT]

devices = /srv/node

mount\_check = false

bind\_port = 6010

user = swift

bind\_ip = 0.0.0.0

workers = 2

[pipeline:main]

pipeline = object-server

[app:object-server]

use = egg:swift#object

[object-replicator]

[object-updater]

[object-auditor]

创建/etc/swift/ container-server.conf文件

[DEFAULT]

devices = /srv/node

mount\_check = false

bind\_port = 6011

user = swift

bind\_ip = 0.0.0.0

workers = 2

[pipeline:main]

pipeline = container-server

[app:container-server]

use = egg:swift#container

[container-replicator]

[container-updater]

[container-auditor]

[container-sync]

6.启动swift服务

sudo chown -R swift:swift /etc/swift/\*

swift-init all start

#当启动的时候可能会报WARNING: Unable to increase file descriptor limit. Running as non-root? 这是正常情况

### 7、 测试swift(在Proxy主机上测试)

1. 初始化swauth

swauth-prep -K swauthkey #执行成功的话则不会有任何信息显示

#在这一步经常会出现Auth subsystem prep failed: 500 ServerError这种错误，发生错误的原因可能是你的/srv下的目录的拥有者不是属于swift用户。

2. 添加用户

swauth-add-user -A http://127.0.0.1:8080/auth/ -K swauthkey -a test tester testing #在tenant test中添加tester用户并设置密码为testing

3. 检测swift运行状态

swift -A http://127.0.0.1:8080/auth/v1.0 -U test:tester -K testing stat -v #运行成功则会返回类似如下的信息

StorageURL:http://127.0.0.1:8080/v1/AUTH\_84841e28-93a8-4e9c-a295-f9734f25da01

AuthToken: AUTH\_tkfacf171def344d03bf01519e93327f3c

Account:AUTH\_ 84841e28-93a8-4e9c-a295-f9734f25da01

Containers:0

Objects:0

Bytes:0

Accept-Ranges:bytes

X-Trans-Id:txa0c57ad0fd6d435b8287ca111ca0269a

注:红色字体部分是根据具体情况而定的http://www.msdnproject.com/，并不是每个人系统返回都是一样的，假如出现了错误你可以在/var/log/swift/目录下查询到相应的信息

9. 上传和下载文件(在Proxy主机上测试)

1. 获取X-Storage-Token和X-Storage-Url

curl -v -H 'X-Storage-User: test:tester' -H 'X-Storage-Pass: testing' http://127.0.0.1:8080/auth/v1.0 #执行完该条语句会返回相应的信息，例如（返回信息的一小部分）：

HTTP/1.1200 OK

X-Storage-Url:http://127.0.0.1:8080/v1/AUTH\_84841e28-93a8-4e9c-a295-f9734f25da01

X-Storage-Token:AUTH\_tkfacf171def344d03bf01519e93327f3c

X-Auth-Token:AUTH\_tkfacf171def344d03bf01519e93327f3c

2. 创建container和列出容器

curl -X PUT -D - -H "X-Auth\_Token:AUTH\_tkfacf171def344d03bf01519e93327f3c"http://127.0.0.1:8080/v1/AUTH\_84841e28-93a8-4e9c-a295-f9734f25da01/cynric #创建tule容器，红色部分是上一步返回的信息

swift -A http://127.0.0.1:8080/auth/v1.0 -U test:tester -K testing list #查看test用户的container列表

3. 上传和列出文件

swift -A http://127.0.0.1:8080/auth/v1.0 -U test:tester -K testing upload cynric/etc/swift/proxy-server.conf #将/etc/swift/proxy-server.conf文件上传到tule容器中

swift -A http://127.0.0.1:8080/auth/v1.0 -U test:tester -K testing list tule #列出tule容器中的文件

4. 下载文件

swift -A http://127.0.0.1:8080/auth/v1.0 -U test:tester -K testing download tule #将tule容器下的所有文件下载到本地

注意：如果上传文件的时候，有嵌套目录，则上传到swift上后，也是以嵌套目录的形式存在，下载后，会在当前目录创建一样的嵌套目录。

## 二、Swift调试环境搭建：eclipse+pydev

Eclipse是由java开发的，所以要想让Eclipse跑起来就得安装jdk，在命令行里输入javac就能查看本机是否已经安装好了jdk了。

### 1、安装jdk

先从oracle官网上下载jdk，放到/usr/local/src/文件中，由于找不到linux的下载连接，所以只能在界面中用FF直接下载了。

1、cd /usr/local/src/ #进入所放下载文件的目录  
2、chmod u+x jdk-6u22-linux-i586-rpm.bin #由于下载的安装文件没有x权限，所以给root用户添加x权限  
3、./jdk-6u22-linux-i586-rpm.bin #安装完成

此时在命令行里输入java -version已经能看到java的安装信息了。

### 2、安装eclipse

到官eclipse官网上下载你自己想要的linux下的eclipse版本，保存到/usr/local/src/目录中，我这里下的是eclipse-jee-indigo-SR2-linux-gtk.tar.gz，是个压缩包，eclipse是免安装的，直接解压缩就可以使用了。

1、cd /usr/local/src/ #进入下载的文件目录中  
2、tar -zxv -f eclipse-cpp-helios-SR1-linux-gtk.tar.gz -C /usr/local/ #-C解压到的目录，自动生成一下eclipse的目录

启动eclipse只能在图形界面中启动，不能在命令行模式下启动。可以在图形界面中像windows一样找到该解压目录/usr/local/eclipse/，双击eclipse文件就能启动了，也可以在终端输入/usr/local/eclipse/eclipse来启动。

### 3、配置PyDev

(1)启动eclipse，选择Help—>Install New Software...

(2)在workwith里键入<http://www.fabioz.com/pydev/updates> ，单击Add，eclipse自动搜索到PyDev的插件，选择之，一步步Next进行安装

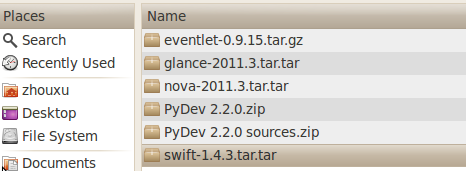
(3)查看PyDev是否安装成功，选择Help—>About Eclipst—>Installation Details，即可到具体安装的插件

(4)配置PyDev解释器，选择Windows—>Preferences—>PyDev—>Interpreter-python，点击New，找到自己python的安装路径，选择python.exe，再选择具体的Libraries.最后OK

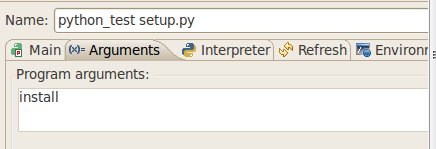
(5)最后就是使用PyDev环境了，在eclipse的最右上角显示着当前正在使用的是哪个开发环境，选择Open Perspective—>Other—>PyDev—>OK，即可切换到PyDev开发环境

### 4、导入swift项目

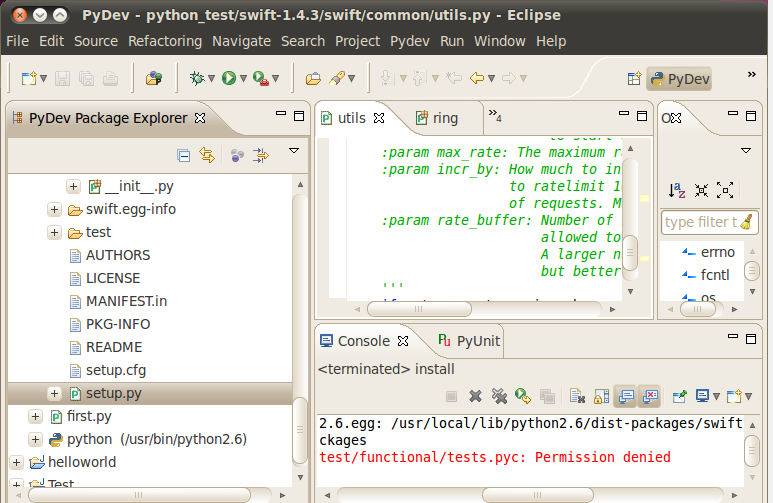
工程—>import->🡪General->🡪Archive File



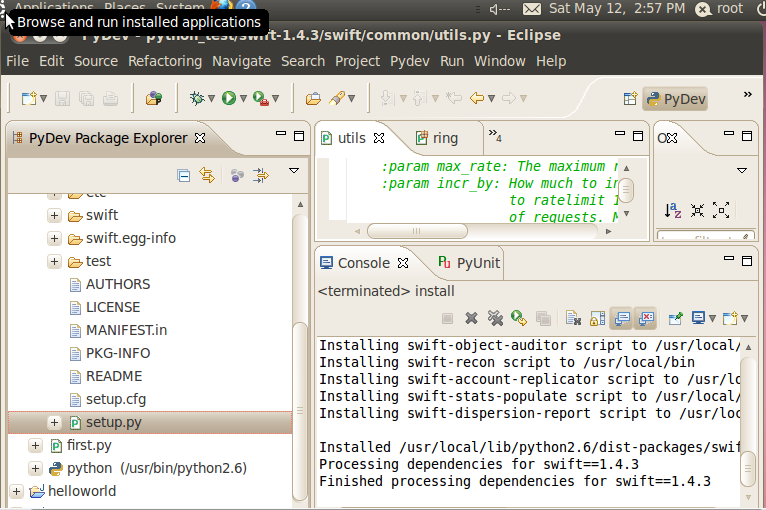
### 5、运行swift

运行swift目录下面的setup.py,选择Run Configurations在新窗口中Arguments下的Programe arguments填入install 

如果在非特权用户下运行，最后会报错：



切换到root账户下，重启eclipse然后就可以运行成功了。



要在eclipse集成环境下运行swift,首先要把相关依赖包安装好，就是前面部署swift的依赖包。