Centos 安装Tsung

# 环境信息

| **软件** | **版本** |
| --- | --- |
| os | Centos 8.3 2011 |
| Tsung | 1.7.0 |
|  |  |

# 配置代理

## 2.1 yum代理配置

curl -o /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-8.repo

# 配置代理  
sed -i -e "s|mirrors.cloud.aliyuncs.com|mirrors.aliyun.com|g " /etc/yum.repos.d/CentOS-\*

sed -i -e "s|releasever|releasever-stream|g" /etc/yum.repos.d/CentOS-\*

yum clean all && yum makecache  
sudo dnf install centos-release-stream -y  
sudo dnf swap centos-{linux,stream}-repos -y  
sudo dnf distro-sync -y  
​  
# 同步时间

rpm -ivh http://mirrors.wlnmp.com/centos/wlnmp-release-centos.noarch.rpm  
yum install wntp -y  
ntpdate ntp1.aliyun.com  
# 安装gcc 编译工具  
dnf group install "Development Tools" -y  
dnf install man-pages -y  
gcc --version  
yum install make -y  
​

## 2.2 google 代理安装

安装 Clash

1. 下载当前操作系统与 CPU 架构对应的包文件，我这儿是 X86\_64 平台下 CentOS8 所以对应的使用 [clash-linux-amd64-v1.6.5.gz](https://github.com/Dreamacro/clash/releases/download/v1.6.5/clash-linux-amd64-v1.6.5.gz) 包即可

Shell

wget -O clash.gz https://github.com/Dreamacro/clash/releases/download/v1.6.5/clash-linux-amd64-v1.6.5.gz

1. 下载好后解压安装包中 clash 到 /usr/local/bin/ 目录下，并删除压缩包文件

shell  
​  
gzip -dc clash.gz > /usr/local/bin/clash  
chmod +x /usr/local/bin/clash  
rm -f clash.gz

1. 创建配置文件目录，并下载 MMDB 文件

shell  
​  
mkdir /etc/clash  
wget -O /etc/clash/Country.mmdb https://www.sub-speeder.com/client-download/Country.mmdb

1. 创建 systemd 脚本，脚本文件路径为 /etc/systemd/system/clash.service，内容如下：

/etc/systemd/system/clash.service  
​  
[Unit]  
Description=clash daemon  
​  
[Service]  
Type=simple  
User=root  
ExecStart=/usr/local/bin/clash -d /etc/clash/  
Restart=on-failure  
​  
[Install]  
WantedBy=multi-user.target

1. 重载 systemctl daemon

shell  
​  
systemctl daemon-reload

配置代理上网

1. 导入已有的科学上网订阅链接，如果没有机场的话推荐使用[心阶(www.xinjiecloud.online)](https://www.xinjiecloud.online/auth/register?code=EzE2)实测用了一年稳的一逼

shell  
​  
wget -O /etc/clash/config.yaml [你的订阅链接]

1. 设置系统代理，添加配置文件 /etc/profile.d/proxy.sh 并在其中写入如下内容：

/etc/profile.d/proxy.sh  
​  
export http\_proxy="127.0.0.1:7890"  
export https\_proxy="127.0.0.1:7890"  
export no\_proxy="localhost, 127.0.0.1"

1. 重载 /etc/profile 配置

shell  
​  
source /etc/profile

1. 启动 clash 服务，并设置为开机自动启

shell  
​  
systemctl start clash  
systemctl enable clash

1. 测试 goolge.com 访问

shell  
​  
# curl google.com  
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">  
<TITLE>301 Moved</TITLE></HEAD><BODY>  
<H1>301 Moved</H1>  
The document has moved  
<A HREF="http://www.google.com/">here</A>.  
</BODY></HTML>

配置 web-ui

1. 克隆 [clash-dashboard](https://github.com/Dreamacro/clash-dashboard) 项目到本地

shell  
​  
git clone -b gh-pages --depth 1 https://github.com/Dreamacro/clash-dashboard /opt/clash-dashboard

1. 修改 clash 配置文件中 external-ui 的值为 /opt/clash-dashboard

sehll  
​  
sed -i "s/^#\{0,1\} \{0,1\}external-ui.\*/external-ui: \/opt\/clash-dashboard/" /etc/clash/config.yaml

1. 重启 clash 服务

shell  
​  
systemctl restart clash

1. 通过浏览器访问 localhost:9090/ui，其中 localhost 替换为 clash 部署服务器的 IP

配置定时更新订阅

使用如下脚本填写相关配置项目并放入 /etc/cron.weekly 目录下，每周自动更新订阅配置文件即可

/etc/cron.weekly/clash.sh  
​  
#!/usr/bin/env bash  
​  
# 订阅链接地址  
SUBSCRIBE=""  
# web-ui存放目录，留空则保持默认不修改  
WEB\_UI=""  
# API 端口，留空则保持默认不修改  
CONTROLLER\_API\_PROT=""  
# API 口令，留空则保持默认不修改  
SECRET=""  
​  
CLASH\_CONFIG="/etc/clash/config.yaml"  
​  
​  
if [ -z "${SUBSCRIBE}" ]; then  
  echo "Subscription address cannot be empty"  
  exit 1  
fi  
​  
systemctl stop clash  
​  
wget --no-proxy -O ${CLASH\_CONFIG} ${SUBSCRIBE}  
​  
if [ -n "${WEB\_UI}" ]; then  
sed -i "s?^#\{0,1\} \{0,1\}external-ui.\*?external-ui: ${WEB\_UI}?" ${CLASH\_CONFIG}  
fi  
​  
if [ -n "${CONTROLLER\_API\_PROT}" ]; then  
sed -i "s?^external-controller.\*?external-controller: '0.0.0.0:${CONTROLLER\_API\_PROT}'?" ${CLASH\_CONFIG}  
fi  
​  
if [ -n "${SECRET}" ]; then  
sed -i "s?^secret.\*?secret: '${SECRET}'?" ${CLASH\_CONFIG}  
fi  
​  
systemctl start clash

上述脚本写入 /etc/cron.weekly/clash.sh 并配置好相关变量后，保存退出并赋予可执行权限

shell  
​  
chmod 0755 /etc/cron.weekly/clash.sh

至此，Linux 下 clash 配置完成，可以愉快的上外网拉包了

参考：https://little-star.love/posts/f2114751/

# 3， 安装Tsung

## 3.1 安装erlang

Mac OS X通过 Homebrew 即可安装 tsung，命令：  
​  
brew install tsung  
Ubuntu 安装也很简单，命令：  
​  
apt-get install tsung  
其他系统需要先下载源码包再安装，源码包下载地址：  
​  
http://tsung.erlang-projects.org/dist/  
以tsung 1.7.0为例，下载安装命令：  
​  
wget http://tsung.erlang-projects.org/dist/tsung-1.7.0.tar.gz # 以1.7.0版本为例  
tar -zxf tsung-1.6.0.tar.gz   # 解压  
cd tsung-1.7.0               # 进入目录  
./configure                   # 配置，生成 Makefile 文件  
make                         # 编译  
make install                 # 安装  
configure 的时候，如果没有安装 tsung 依赖的 Erlang 语言库，会提示下面的错误：  
​  
...  
checking for Erlang/OTP root directory... configure: error: in `/root/tsung-1.7.0':  
configure: error: test Erlang program execution failed  
...  
可以用下面的命令先安装erlang，再 configure。  
​  
brew install erlang       # OS X Homebrew  
port install erlang       # OS X MacPorts  
apt-get install erlang     # Ubuntu 和 Debian  
yum install erlang         # Fedora  
pkg install erlang         # FreeBSD  
其他系统需要从源码安装 Erlang，稍微复杂一些，可参考：https://github.com/erlang/otp/blob/maint/HOWTO/INSTALL.md。

编译erlang前 有很多前提条件。

其中 需要编译wxWidgets

过程：

yum install gtk2-devel -y  
git clone --recurse-submodules https://github.com/wxWidgets/wxWidgets.git #依赖梯子  
cd wxWidgets/  
./configure --with-gtk  
make install

验证erlang

# erl  
# ssl:versions().  
# halt().

## 3.2 安装Tsung

tar -xvf tsung-1.7.0.tar.gz  
./configure   
make -j2  
make install  
tsung  
tsung-recorder  
​  
#安装依赖  
yum -y install gnuplot perl5 perl-ExtUtils-CBuilder perl-ExtUtils-MakeMaker   # tsung\_stats.pl 需要用到的 gnuplot

## 3.3 安装perl Template,用于生成报告模版

官网下载：[http://www.template-toolkit.org/](https://www.oschina.net/action/GoToLink?url=http%3A%2F%2Fwww.template-toolkit.org%2F)

cd /usr/local  
wget http://www.cpan.org/modules/by-module/Template/Template-Toolkit-3.010.tar.gz  
tar -zxvf Template-Toolkit-3.010.tar.gz  
cd Template-Toolkit-3.010  
perl Makefile.PL    
make    
make test    
make install

# 使用Tsung

## 4,1 启动

**tsung**安装完成后，会生成两个命令文件：

1. tsung
2. tsung-recorder

默认生成在 /usr/local/bin 目录下，使用-h参数可以查到它们所有的参数：

tsung -h

在启动 tsung 之前，我们需要一个 XML 格式的配置文件，tsung会根据该配置文件进行测试。

/usr/share/doc/tsung/examples 目录下有一些 XML 配置文件范例，

将其中HTTP测试配置文件 http\_simple.xml 拷贝到当前目录，

再执行，命令如下：

cp /usr/local/share/doc/tsung/examples/http\_simple.xml ./   # 拷贝范例配置文件  
tsung -f http\_simple.xml start                             # 指定xml文件并开始。如果不指定，则默认使用 ~/.tsung/tsung.xml

## 4.2 结果

**tsung** 测试结束后，测试结果 Log 保存在目录 ~/.tsung/log/ 下。

当启动一个新的测试时，会在这个目录下面创建一个新的子目录，用以保存测试的数据。

Log 格式为当前日期和时间的组合，例如：~/.tsung/log/20220316-0307。

默认情况下，控制节点会启动一个嵌入的网站服务器，侦听8091端口（可以用 -n 选项禁用）。

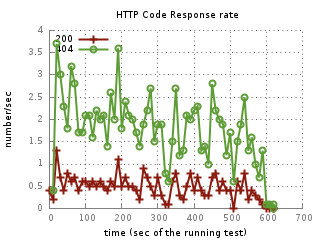
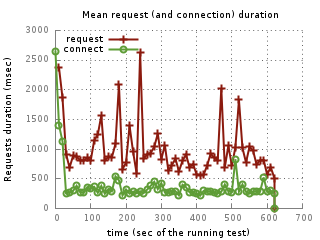
测试过程也可以用tail命令查看实时记录：

tail -f ~/.tsung/log/20220316-0307/tsung.log

测试结束之后，再用tsung\_stats.pl生成报表。

yum install gnuplot       # tsung\_stats.pl 需要用到的 gnuplot  
mkdir http\_simple         # 创建用以保存报表的目录  
cd http\_simple           # 进入目录  
/usr/local/lib/tsung/bin/tsung\_stats.pl --stats ~/.tsung/log/20220316-0307/tsung.log # 生成报表

之后，会在 http\_simple 目录下生成3个目录和1个 log 文件，其中 images 目录下就是报表图片，类似如下。

以上仅是测试，没有实际用处，因为没有配置要测试的网站，所以所得到的结果没有实际意义。

需要修改 XML 配置文件后再测试，才会得到有意义的结果。

# 配置说明

打开 http\_simple.xml ，下面来讲几个关键的配置。

## 4.1 clients

用户产生的方式：

<clients>  
<client host="localhost" use\_controller\_vm="true" maxusers="30000"/>  
</clients>

**tsung**运行时可以由很多的虚拟机（VM）组成，client配置指明这个client机器上最多生成的用户数，

如果use\_controller\_vm为true的话，那么当用户数达到maxusers，tsung会自动生成新的虚拟机。

## 4.2 servers

<servers>  
<server host="garden.blue.jude.poppen.lab" port="80" type="tcp"></server>  
</servers>

server段可以配置**被测服务器**的相关信息，也可以配置成集群，如下

<servers>  
<server host="server1" port="80" type="tcp" weight="4"></server>  
<server host="server2" port="80" type="tcp" weight="1"></server>  
</servers>

**tsung**会根据weight值来选择发起请求的server。

## 4.3 monitoring

系统监控服务，配置完后可获取被测server的CPU、内存、负载、数据库等相关信息。

可以配置成erlang的监控服务和snmp的监控服务。

<monitoring>  
<monitor host="garden" type="erlang">  
  <mysqladmin port="3306" username="root" />  
</monitor>  
</monitoring>

## 4.4 load

<load>  
<arrivalphase phase="1" duration="3" unit="minute">  
  <users maxnumber="100" interarrival="0.02" unit="second" ></users>  
</arrivalphase>  
</load>

load段可配置访问的负载，访问可以配成多个阶段，由phase值指定。

duration是测试持续时间，unit是单位。

users段的maxnumber限制了生成的最大用户数，

interarrival="0.02"表示0.02秒产生一个新用户，用户按照session的配置顺序执行session中的request。

## 4.5 options

<options>  
<option type="ts\_http" name="user\_agent">  
<user\_agent probability="80">Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.7.8) Gecko/20050513 Galeon/1.3.21</user\_agent>  
<user\_agent probability="20">Mozilla/5.0 (Windows; U; Windows NT 5.2; fr-FR; rv:1.7.8) Gecko/20050511 Firefox/1.0.4</user\_agent>  
</option>  
</options>

options段可配置一些请求的信息，如agent信息。

## 4.6 sessions

<sessions>  
<session name="http-example" probability="70" type="ts\_http">  
  <setdynvars sourcetype="random\_number" start="1" end ="100">  
    <var name="itemid" />  
  </setdynvars>  
  <transaction name='getlist'>  
    <request subst="true">  
      <http url="/comment/getList" method="POST" contents = "item\_type=image&amp;item\_id=%%\_itemid%%"></http>  
    </request>  
  </transaction>  
</session>  
<session name="http-example" probability="30" type="ts\_http">  
  <setdynvars sourcetype="random\_number" start="1" end="100">  
    <var name="itemid" />  
  </setdynvars>  
  <setdynvars sourcetype="random\_number" start="20" end="5000000">  
    <var name="content" />  
  </setdynvars>  
  <transaction name='getlist'>  
    <request subst="true">  
        <http url="/comment/addComment" method="POST" contents = "item\_type=image&amp;item\_id=%%\_itemid%%&amp;content=%%\_content%%"></http>  
    </request>  
  </transaction>  
</session>  
</sessions>

可配置多个子session，进而可测试多个api，

可以设置请求概率，在probability里定义，

要求每个session的probability之和是100，类型是http。

sessions里可用for来设定请求次数，如下

<for from="1" to="@loop" incr="1" var="counter">

在里面可以设置请求的具体信息。

在请求参数里可以带上随机数。

随机数和随机字符串的定义如下：

<setdynvars sourcetype="random\_number" start="20" end="5000000">  
<var name="xxx" />  
</setdynvars>  
<setdynvars sourcetype="random\_string" length="10">  
<var name="xxx" />  
</setdynvars>

以%%\_xxx%%的形式来调用。

这里必须注意的是，要使用随机数，request必须加上subst="true"参数，不然随机数无法被引用成功。

随机数也可从文件读取，如csv。

http内部可定义header参数：

<http\_header name="Authorization" value="111"/>  
<http\_header name="Cookie" value="authToken=%%\_auth\_token%%; Path=/"/>  
<!-- content-Type：POST请求参数的格式，如果是json格式可以这样写 -->  
<http\_header name="Content-Type" value="application/json"/>

thinktime可用于定义两个请求的间隔时间

<thinktime value="1"/>

另外可定义不同的transaction ，这样子结果里就会显示不同transaction的具体信息。