# 基于jenkins构建接口自动化测试

为了将接口测试作为常规自动化执行任务，或者在无人值守的情况下定时执行并发布测试结果，故实现了基于jenkins构建接口自动化测试。具体流程如下：开发上传版本后，jenkins检查到github指定的仓库中有push动作，从github上拉取代码，打包，发布到tomcat，触发测试任务，自动从github拉取测试代码进行测试，将测试结果邮件发送到指定的邮箱。

服务器的代码由springboot实现，测试代码是python实现。整个流程的配置如下：

## 环境

centos7.6+ python3.6.5 + jenkins2.164.2+ JDK1.8+git

## python3.6.5安装和配置

linux下默认是带有python2.7版本的，我们的接口测试框架是基于Python3，所以需要重新安装。注意：一定不要卸载python2，因为centos7中的yum用的就是python2，卸载后会导致yum不能使用。步骤如下：

### 切换到安装目录

[root@test ~]# cd /usr/local/bin

### 下载压缩文件

[root@test bin]# wget [https://www.python.org/ftp/python/3.6.5/Python-3.6.5.tgz](https://www.python.org/ftp/python/3.6.5/Python-3.6.5.tgz" \t "https://www.jianshu.com/p/_blank)

### 解压缩该软件包

[root@test bin]# tar -zxvf Python-3.6.5.tgz

### 在解压后的目录安装

[root@test bin]# cd Python-3.6.5

[root@test Python-3.6.5]# ./configure --with-ssl

注意：在./configure过程中，如果没有加上–with-ssl参数时，默认安装的软件涉及到ssl的功能不可用，刚好pip3过程需要ssl模块

### 编译和安装

[root@test Python-3.6.5]# make

[root@test Python-3.6.5]# make install

### 验证当前系统下安装的python版本号

[root@test Python-3.6.5]# python -V

Python 3.6.5

### 备份原有配置

[root@test Python-3.6.5]# mv /usr/bin/python /usr/bin/python.bak

### 建立软链接

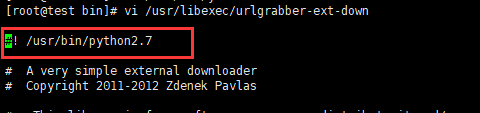
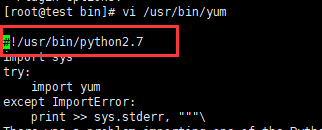
[root@test Python-3.6.5]ln -s /usr/local/bin/Python-3.6.5/python /usr/bin/python

[root@test Python-3.6.5]ln -s /usr/local/bin/Python-3.6.5/bin/pip3 /usr/bin/pip

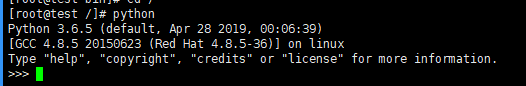
### 修改yum配置文件

为了使yum命令正常使用，修改以下两个配置文件将其配置的python依然指向2.x版本

/usr/bin/yum  
/usr/libexec/urlgrabber-ext-down  
将上面两个文件的头部文件修改为老版本即可  
!/usr/bin/python --> !/usr/bin/python2.7



### 切换到任意目录执行python验证软连接配置成功



### 安装依赖包

[root@test ~]# pip3 install openpyxl==2.5.4

[root@test ~]# pip3 install requests

## 安装git

yum -y install git

## 安装jdk1.8

### 检查是否安装wget

[root@test ~]# rpm -qa|grep wget

[root@test ~]# yum install wget（安装）

### 将window下载好的包放到linux下

安装lrzsz就可以将window下载好的文件放到linux下，要有xshell或者secureCRT(**先切换到要放置的路径,一般放到/opt/下)**：

[root@test opt]# rpm -qa|grep lrzsz（查看是否已安装lrzsz）

[root@test opt]# yum install lrzsz（安装lrzsz）

[root@test ~]# cd /opt/ (切换到/opt/下)

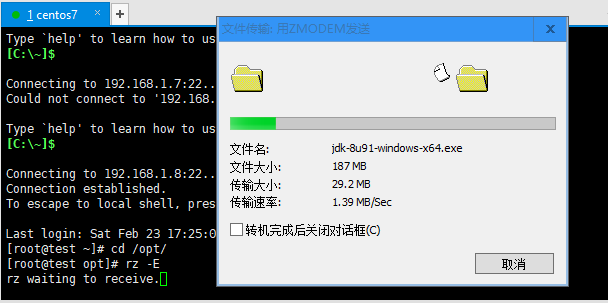
[root@test opt]# rz -E（拖拽）

rz waiting to receive.

[root@test opt]# ls

jdk-8u91-windows-x64.exe

安装成功后可以将文件拖拽到linux下



### tar命令将安装包解压

[root@test opt]# tar -zxvf jdk-8u11-linux-x64.tar.gz

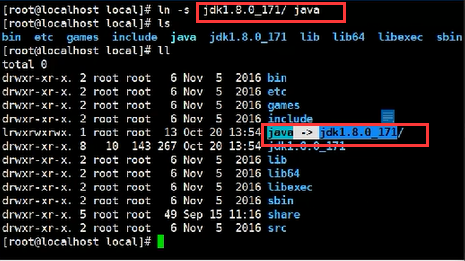
### 解压后放到 /user/local/,并做一个软连接

[root@test opt]#mv jdk1.8.0\_171/ /usr/local/

[root@test opt]#cd /usr/local/



新建软连接：ln -s jdk1.8.0\_171/ java



删除软连接一定一定不要加 -rf参数，有可能会把内容全部删掉

[root@test ~]# cd /opt/

[root@test opt]# ls

jdk1.8.0\_11 jdk-8u11-linux-x64.tar.gz jdk-8u91-windows-x64.exe

[root@test opt]# mv jdk1.8.0\_11/ /usr/local/

[root@test opt]# cd /usr/local/

[root@test local]# ls

bin etc games include jdk1.8.0\_11 lib lib64 libexec sbin share src

[root@test local]# ln -s jdk1.8.0\_11/ java

[root@test local]# ll

total 0

drwxr-xr-x. 2 root root 6 Apr 11 2018 bin

drwxr-xr-x. 2 root root 6 Apr 11 2018 etc

drwxr-xr-x. 2 root root 6 Apr 11 2018 games

drwxr-xr-x. 2 root root 6 Apr 11 2018 include

lrwxrwxrwx. 1 root root 12 Feb 23 21:01 java -> jdk1.8.0\_11/

drwxr-xr-x. 8 10 143 255 Jun 17 2014 jdk1.8.0\_11

drwxr-xr-x. 2 root root 6 Apr 11 2018 lib

drwxr-xr-x. 2 root root 6 Apr 11 2018 lib64

drwxr-xr-x. 2 root root 6 Apr 11 2018 libexec

drwxr-xr-x. 2 root root 6 Apr 11 2018 sbin

drwxr-xr-x. 5 root root 49 Feb 8 20:06 share

drwxr-xr-x. 2 root root 6 Apr 11 2018 src

### 配置java环境变量

在/etc/profile文件的最后添加java的环境变量，注意:一定要把之前的环境变量也要加上，否则就只剩java的环境变量了，其它命令必须用全路径才能执行：

PATH=/usr/local/java/bin:$PATH

[root@test opt]# vi /etc/profile(新增java环境变量)

[root@test opt]# source /etc/profile（使环境变量生效）

-bash: /usr/local/java/bin: No such file or directory

[root@test opt]# pwd

/opt

[root@test opt]# echo $PATH

### 查看java是否按照成功

[root@test local]# java -version

java version "1.8.0\_11"

Java(TM) SE Runtime Environment (build 1.8.0\_11-b12)

Java HotSpot(TM) 64-Bit Server VM (build 25.11-b03, mixed mode)

[root@test local]#

## 安装tomcat

同jdk流程基本一样，详细命令如下：

### 将window下载好的包通过xshell放到linux下

### tar命令将安装包解压

[root@test opt]# tar -zvxf apache-tomcat-8.5.38.tar.gz

### 解压后放到 /user/local/,并做一个软连接

[root@test opt]# mv apache-tomcat-8.5.38 /usr/local/

[root@test opt]# cd /usr/local/

[root@test local]# ln -s apache-tomcat-8.5.38/ tomcat

### 启动tomcat

[root@test tomcat]# cd /usr/local/tomcat/bin/

[root@test bin]# sh catalina.sh start

Using CATALINA\_BASE: /usr/local/tomcat

Using CATALINA\_HOME: /usr/local/tomcat

Using CATALINA\_TMPDIR: /usr/local/tomcat/temp

Using JRE\_HOME: /usr/local/java

Using CLASSPATH: /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tomcat-juli.jar

touch: cannot touch ‘/usr/local/tomcat/logs/catalina.out’: No such file or directory

catalina.sh: line 438: /usr/local/tomcat/logs/catalina.out: No such file or directory

[root@test bin]# mkdir /usr/local/tomcat/logs（出现以上错误执行此命令）

[root@test bin]# sh catalina.sh start

Using CATALINA\_BASE: /usr/local/tomcat

Using CATALINA\_HOME: /usr/local/tomcat

Using CATALINA\_TMPDIR: /usr/local/tomcat/temp

Using JRE\_HOME: /usr/local/java

Using CLASSPATH: /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tomcat-juli.jar

Tomcat started.

### 关闭防火墙开机自启动

systemctl status firewalld.service

systemctl disable firewalld.service

[root@test opt]# systemctl stop firewalld.service

[root@test opt]# systemctl disable firewalld.service

Removed symlink /etc/systemd/system/multi-user.target.wants/firewalld.service.

Removed symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.

[root@test opt]#

### 在windowx下用浏览器访问

浏览器输入：192.168.1.9:8080（将ip修改为服务器对应的值）

### 修改监听端口(可选)

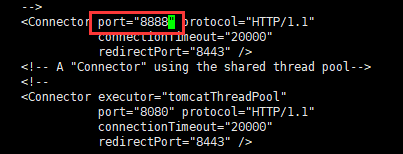
sprintboot内嵌tomcat默认的监听的端口也是8080，和tomcat是冲突的，将tomcat中server.xml修改端口为8888

[root@test ~]# cd /usr/local/tomcat/conf

/usr/local/tomcat/conf

[root@test conf]# vi server.xml

[root@test conf]#



### 查看tomcat执行日志

在tomcat运行时，输入 tail -f catalina.out 便可以查看tomcat的实时运行状态

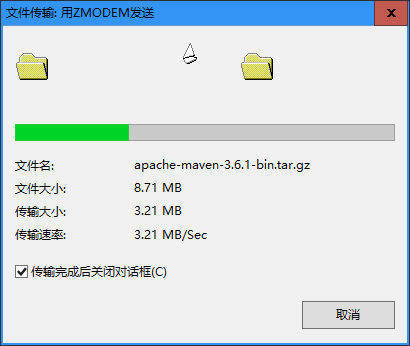
## 安装maven

<https://blog.csdn.net/zwjzqqb/article/details/82378528>

### 切换到安装目录

[root@test ~]#cd /usr/local

### 将下载好的文件拖拽到linux下



### 解压缩

[root@test local]# tar -zxvf apache-maven-3.6.1-bin.tar.gz

### 删除压缩包

[root@test local]# rm -rf apache-maven-3.6.1-bin.tar.gz

### 验证安装成功

ll apache-maven-3.6.1/

apache-maven-3.6.1/bin/mvn --version

[root@test local]# ll apache-maven-3.6.1/

[root@test local]# apache-maven-3.6.1/bin/mvn --version

Apache Maven 3.6.1 (d66c9c0b3152b2e69ee9bac180bb8fcc8e6af555; 2019-04-05T03:00:29+08:00)

Maven home: /usr/local/apache-maven-3.6.1

Java version: 1.8.0\_11, vendor: Oracle Corporation, runtime: /usr/local/jdk1.8.0\_11/jre

Default locale: en\_US, platform encoding: UTF-8

OS name: "linux", version: "3.10.0-957.el7.x86\_64", arch: "amd64", family: "unix"

### 配置maven的环境变量

编辑/etc/profile，在文档的最后一行后面加上以下代码，并重置配置文件，使配置生效

[root@test ~]# vi /etc/profile

MAVEN\_HOME=/usr/local/apache-maven-3.6.1

export PATH=${MAVEN\_HOME}/bin:${PATH}

[root@test ~]#source /etc/profile//重置配置文件，使刚才配置生效

### 验证配置成功

使用mvn-v验证是否配置成功，如果出现以下内容即表示配置成功

[root@test local]# cd /

[root@test /]# mvn -v

Apache Maven 3.6.1 (d66c9c0b3152b2e69ee9bac180bb8fcc8e6af555; 2019-04-05T03:00:29+08:00)

Maven home: /usr/local/apache-maven-3.6.1

Java version: 1.8.0\_11, vendor: Oracle Corporation, runtime: /usr/local/jdk1.8.0\_11/jre

Default locale: en\_US, platform encoding: UTF-8

OS name: "linux", version: "3.10.0-957.el7.x86\_64", arch: "amd64", family: "unix"

## jenkins安装和配置

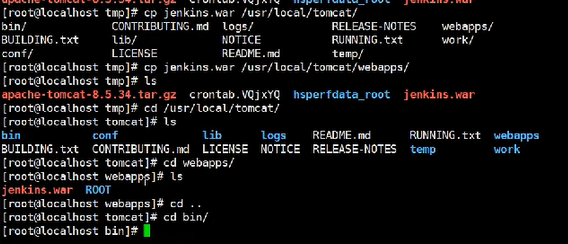
jenkins可以yum安装，也可以war包方式安装，我们以war包安装方式为例进行说明。

### wget下载Jenkins

[root@test tmp]# wget <http://mirrors.jenkins.io/war-stable/latest/jenkins.war>

[root@test tmp]# ls

frist.sh hsperfdata\_root jenkins.war script



### 将war包放到tomcat的webapps下

[root@test tmp]# cp jenkins.war /usr/local/tomcat/webapps/

### 启动jenkins:

方式一：（推荐）

上一步已经把war包放到了tomcat的webapps目录下，启动tomcat后就可以直接在浏览器中访问了

方式二：java -jar方式启动

java -jar /usr/local/apache-tomcat-8.5.38/webapps/jenkins.war --httpPort=8181

### 在windowx下用浏览器访问

方式一：对应上一步的方式一

<http://192.168.1.8:8888/jenkins>

方式二：对应上一步的方式二

[http://192.168.1.8:8181](http://172.20.10.4:8080/)/(红色表示虚拟机的IP，根据自己的真实地址修改即可)，注意：一定不要加"/jenkins"这个路径

### yum方式安装jenkins

# cd /etc/yum.repos.d/

# wget http://pkg.jenkins.io/redhat/jenkins.repo

# rpm --import http://pkg.jenkins.io/redhat/jenkins.io.key

# yum install -y jenkins

# systemctl start jenkins

## jenkins网页配置和插件安装

jenkins启动后，打开浏览器输入ip+启动时指定的端口，比如192.168.1.9:8181，对jenkins进行网页配置

### 根据提示打开指定的文件，输入密钥



[root@test ~]# cat /root/.jenkins/secrets/initialAdminPassword

2fca082cbb7e496cb23998d961dee2c6

### 安装插件

输入密码点下确认之后选择插件，选择默认安装插件即可，此过程需要一段时间，而且根据网络不同有些插件安装不成功，但是之后可以自己再安装即可

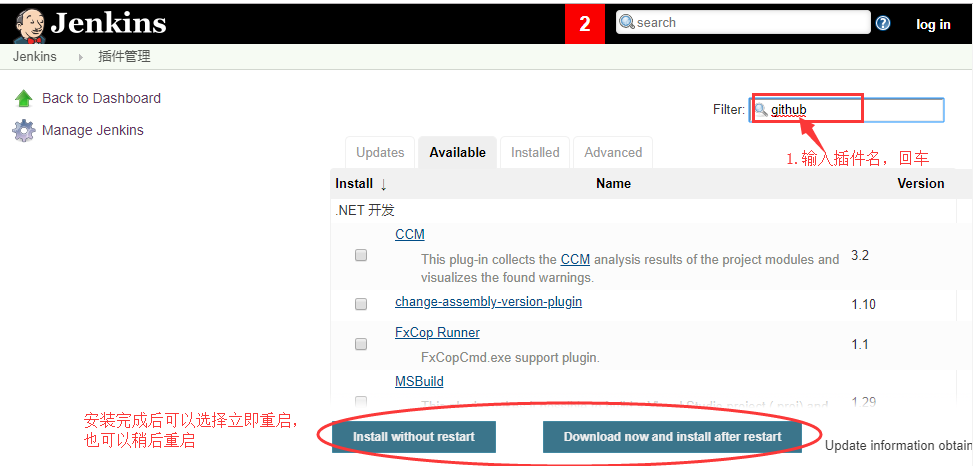


可以直接安装推荐的插件，也可以后续根据需要自行安装

### 创建管理员账号



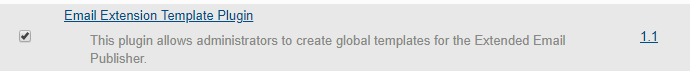
### 自行安装指定插件



已经安装过了，所以显示不出来gtihub的插件

其它插件安装方法类型，接下来我们需要用到的插件如下，按以上方法安装即可：

[Email Extension Template Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Email-ext+Template+Plugin)：扩展的邮件插件，功能更强大，可以设置构建成功发给哪些人，构建失败发送给哪些人等场景



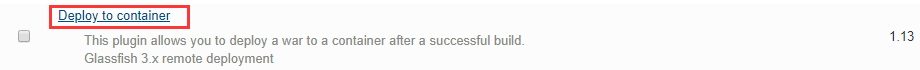
[GitHub Branch Source](https://wiki.jenkins-ci.org/display/JENKINS/GitHub+Branch+Source+Plugin)：github插件



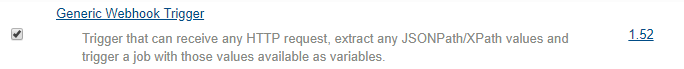
[Maven Integration plugin](https://wiki.jenkins.io/display/JENKINS/Maven+Project+Plugin)：用来创建Maven项目，不安装的话是没有的



[Deploy to container Plugin](http://wiki.jenkins-ci.org/display/JENKINS/Deploy+Plugin)：用来将war包部署到tomcat服务器上

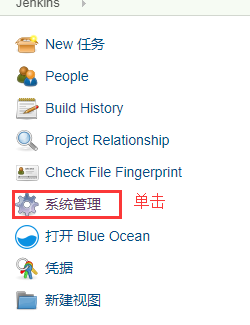


[Generic Webhook Trigger](https://wiki.jenkins-ci.org/display/JENKINS/Generic+Webhook+Trigger+Plugin)：用来创建webhook，获取github的改动信息，自动触发部署

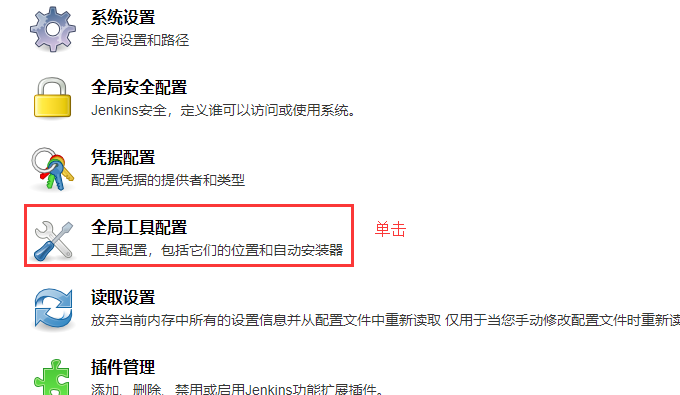


### 系统管理-->全局工具配置

系统管理：



选择全局工具配置：



JDK:



Git:



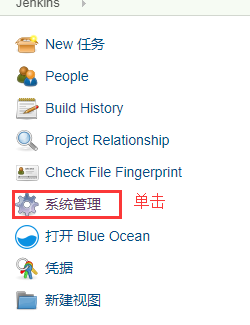
Maven:



以上工具都有自动安装的选项，不过建议手动安装。

### 系统管理-->系统设置

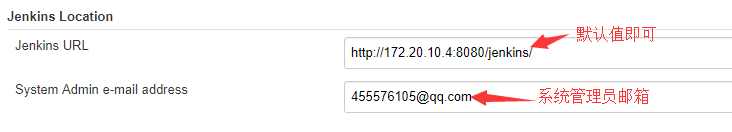
系统管理：



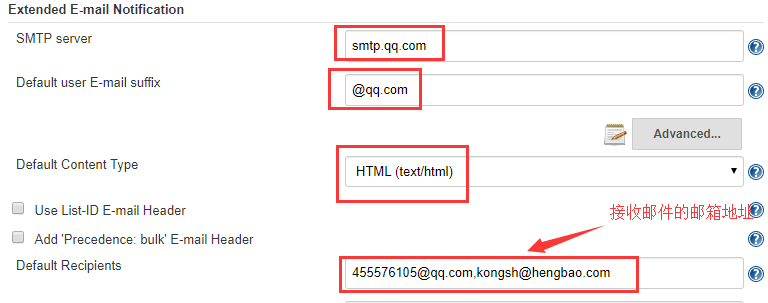
系统设置：



Jenkins Location：



扩展邮件配置：(可选)



1. mail Notification:



配置完成后，单击页面最下方的保存按钮

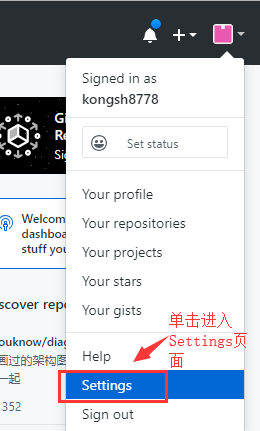


## github和jenkins同步配置

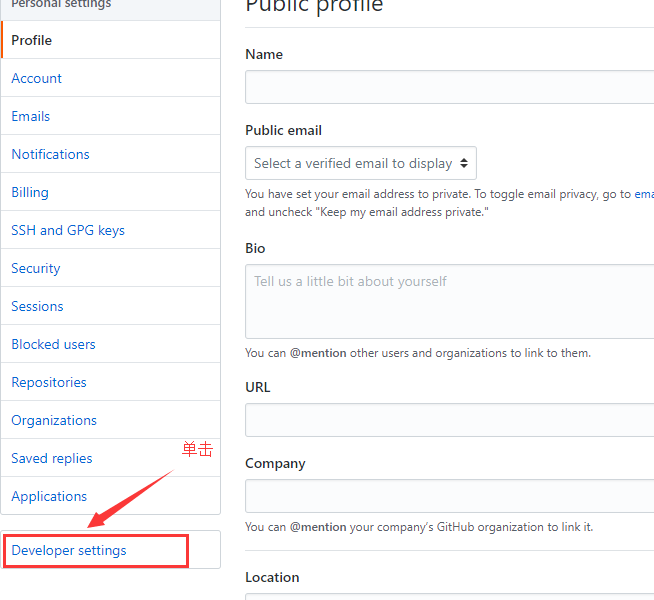
### 创建access token

Jenkins做一些需要权限的操作的时候就用这个access token去鉴权，设置流程如下：

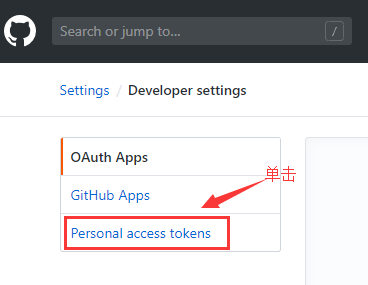
* 登录github，进入Settings页面



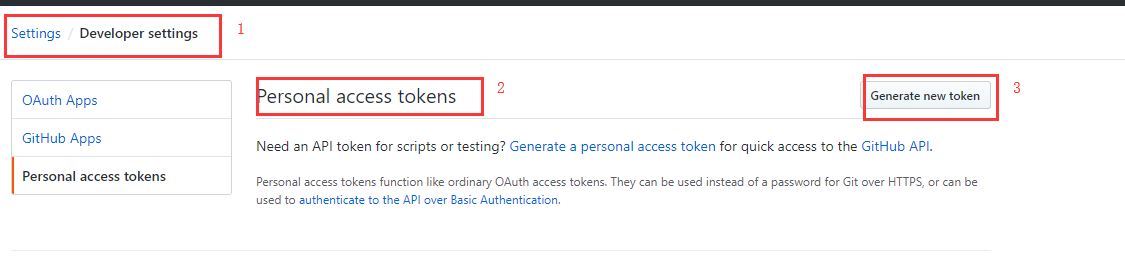
* 单击左下角的Developer settings



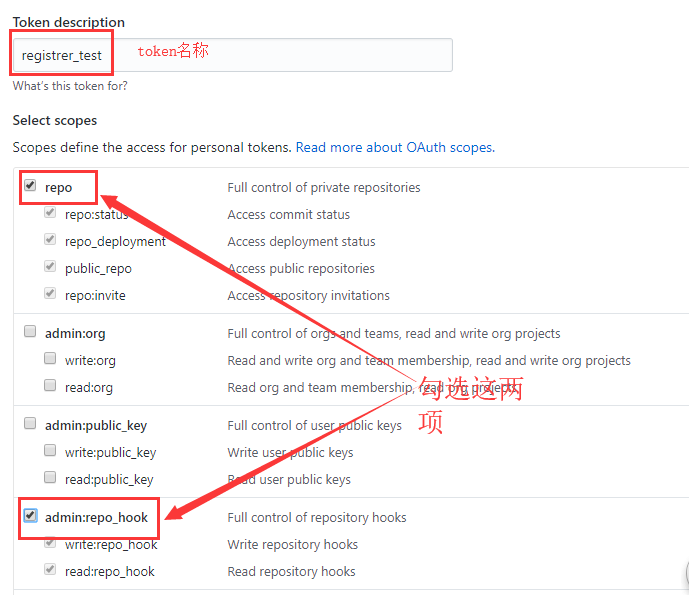
* 跳转到Developer settings页面，单击左下角的Personal access tokens



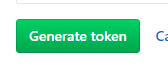
* 跳转到”Personal access tokens”页面后，点击右上角的Generate new token



* 跳转到创建Token的页面，按下图配置即可

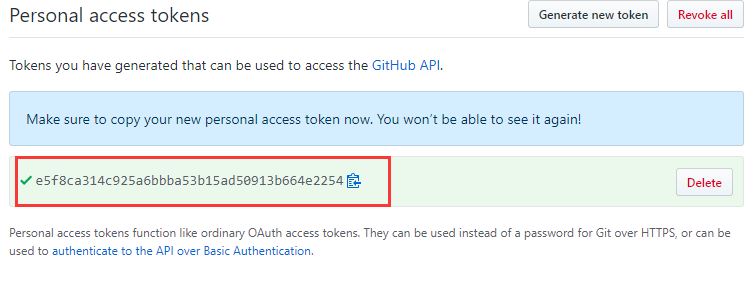


* 单击底部的Generate token按钮

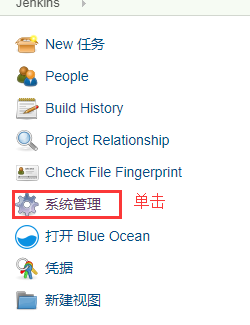


生成的token:

e5f8ca314c925a6bbba53b15ad50913b664e2254



* 进入到jenkins，系统管理-->系统设置



系统设置：



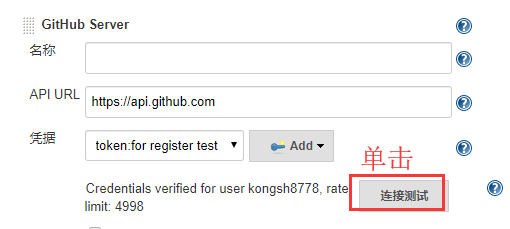
* 找到GitHub，添加一个GitHub 服务如下：



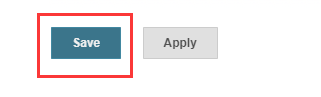
* 弹出的页面中，类型选择Secret text，”Secret”填入前面在GitHub上生成的Personal access tokens，Description写一些描述信息，如下图：



* 设置完成后单击连接测试，提示Credentials verified for user UUserName, rate limit: xxx,则表明有效。



* 单击页面最底部的save按钮

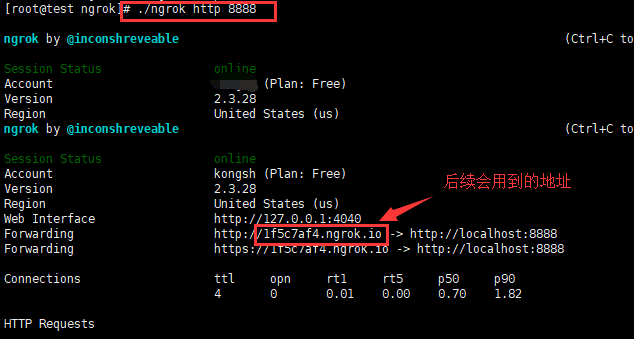
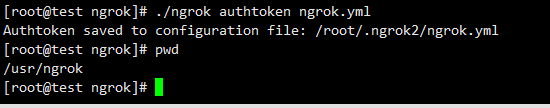
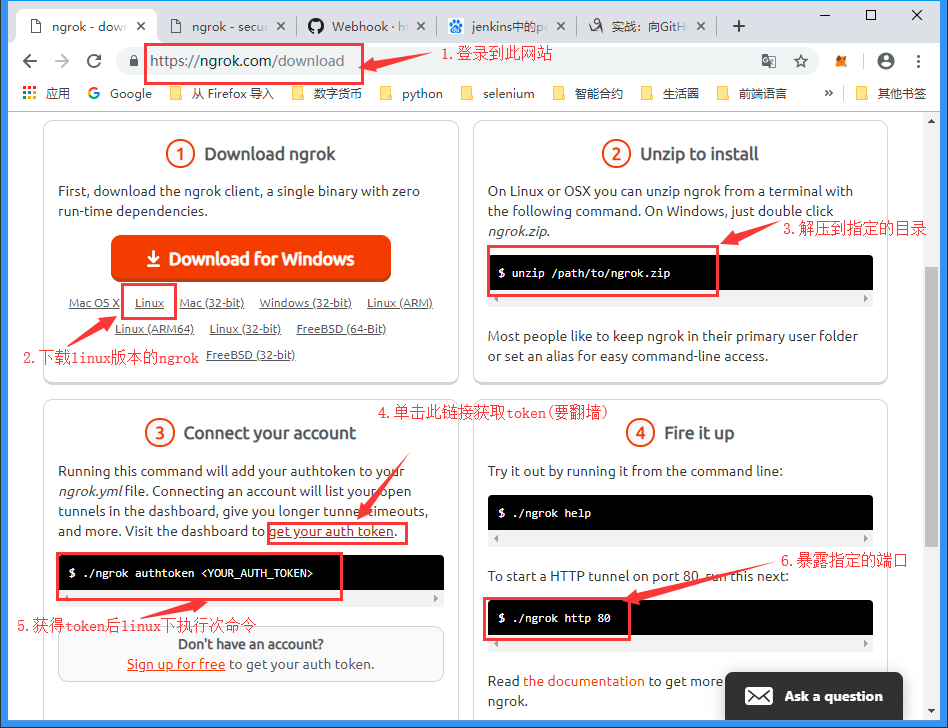


### 下载ngrok，将服务器IP暴露到网络上

GitHub收到提交的代码后要主动通知Jenkins，所以Jenkins所在服务器一定要有外网IP，否则GitHub无法访问，解决方法：下载ngrok，将IP暴露到网络上。

* 登录到<https://ngrok.com/download>
* 下载并配置ngrok

注意：第4步执行时需要翻墙，不然是看不到token的



注意：关机重启后就要重新执行以下命令生成新的IP，同时jenkins和github中的webhook对应的ip也要修改

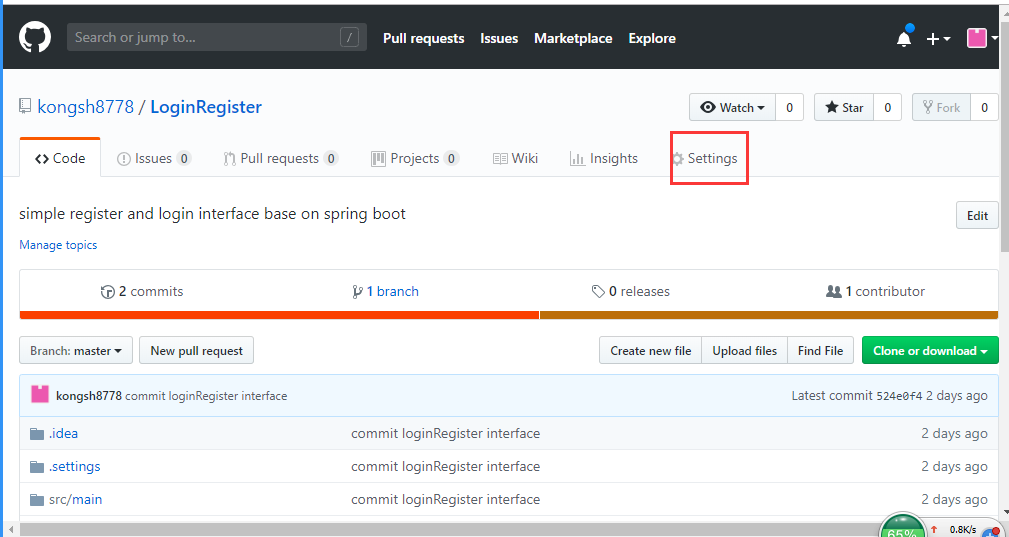
[root@test ~]# cd /usr/ngrok/

[root@test ngrok]# ./ngrok http 8888

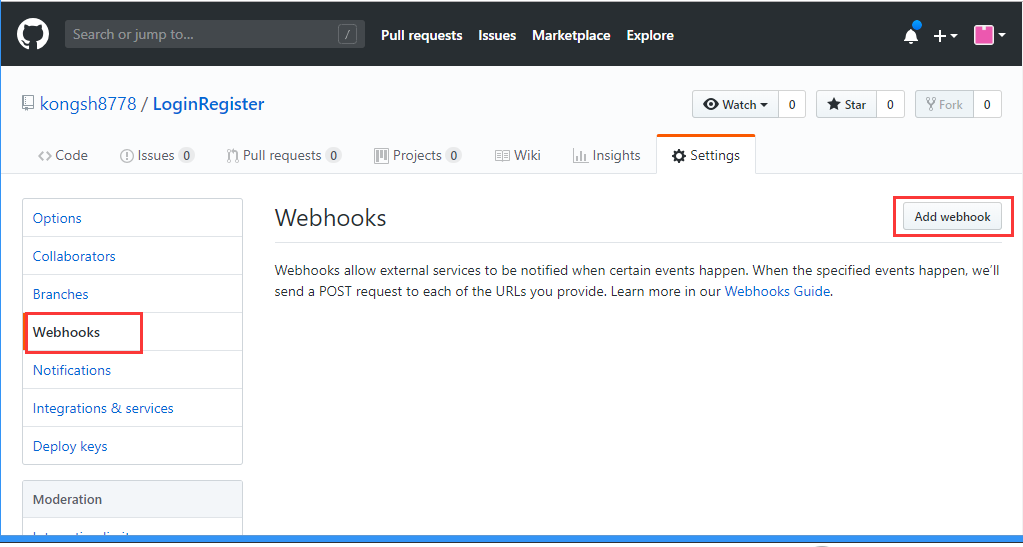
### 配置Jenkins的webhook地址

在整个流程中关键的一步就是监控到github中指定的仓库有push事件时，就启动任务的自动构建，这就需要用到webhook。配置流程如下：

* 登录到github，进入到需要监控的仓库，点击settings:



* 点击Webhooks-->Add webhooks

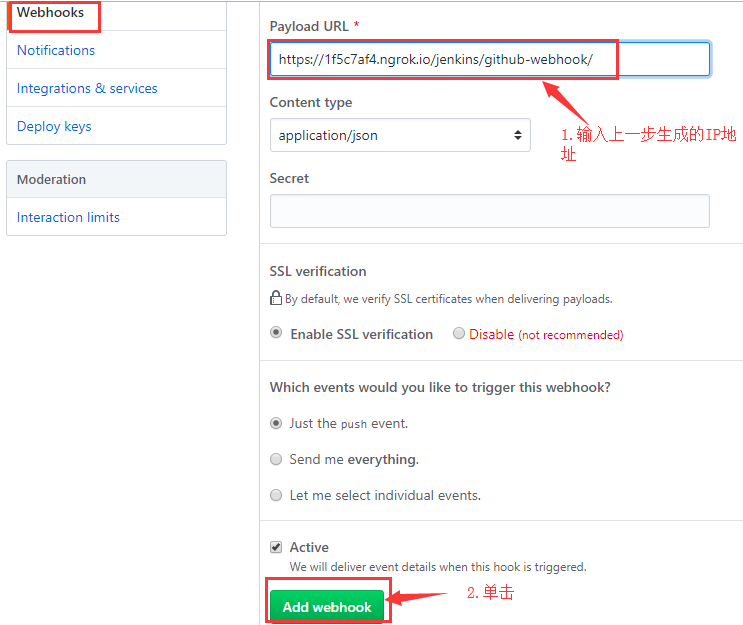


* Payload URL

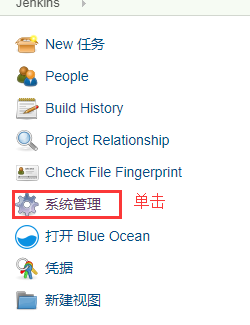
填写jenkins的项目地址+github-webhook/，如：

<https://1f5c7af4.ngrok.io/jenkins/github-webhook/>

标黄的部分就是9.1.1节用ngrok生成的IP地址，其它值默认即可。



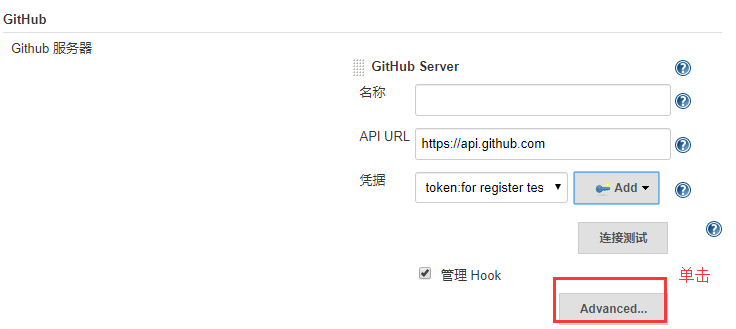
* 进入到jenkins，系统管理-->系统设置



系统设置：



* 找到GitHub，单击Advanced...

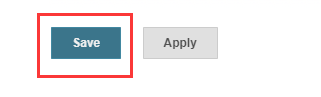


* 配置webhook

注意：post地址一定是[http://your\_jenkins\_url/github-webhook/](https://link.jianshu.com/?t=https://github.com/Beyyes/jenkins_test/settings/hooks/15844304" \t "https://www.jianshu.com/p/_blank)，不能去掉最后的“/”



* 单击页面最底部的save按钮



## 构建自动触发执行的项目

### 构建测试任务：

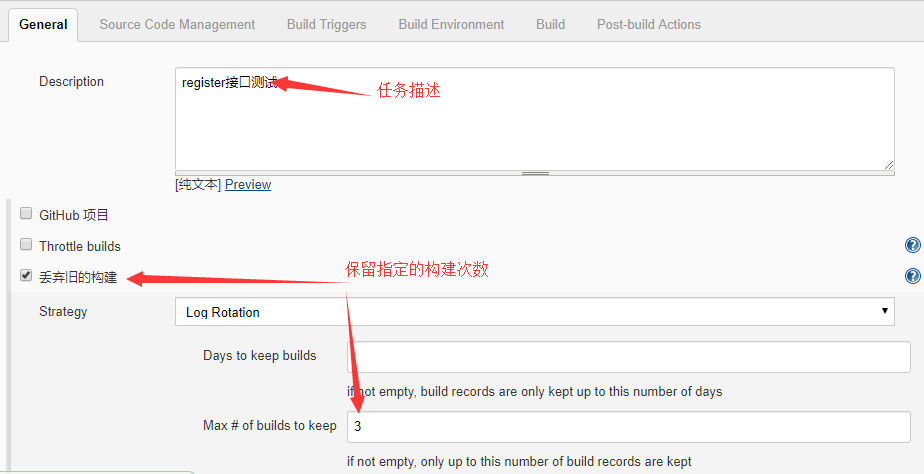
* 单击new 任务:



* 选择自由风格的项目：



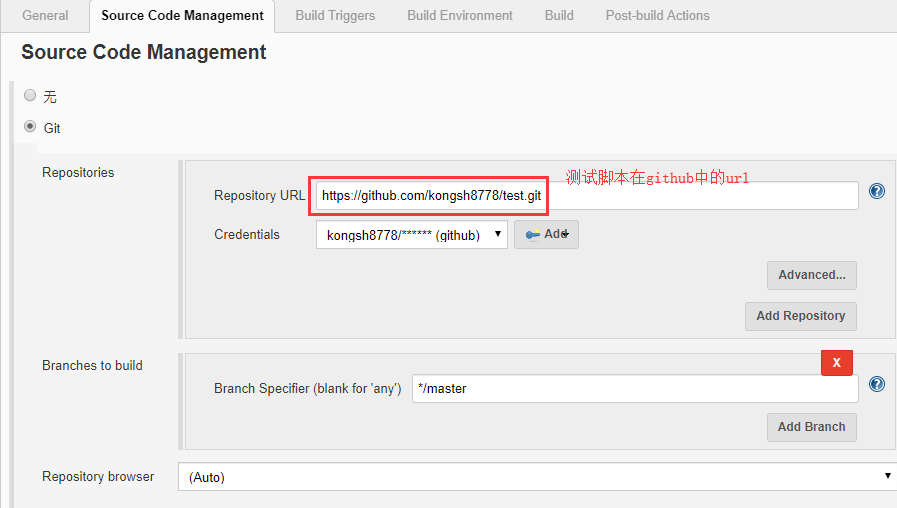
* General:



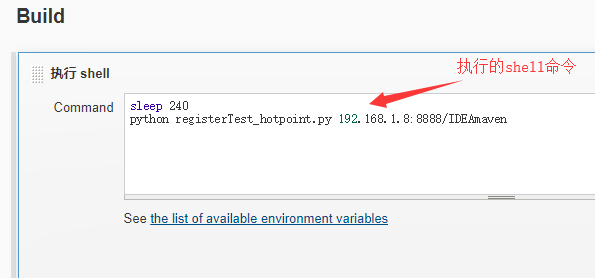
说明：

丢弃旧的构建选项：用来配置保留最近几次构建的结果，以此来节省空间。Max # builds to keep表示最大次数，构建次数超过设定的值后，只保留最新的几次。

* Source Code Management：



* Build：



shell脚本说明：

# 等待4分钟，待服务启动完毕，可以自行设置

sleep 240

# 执行registerTest\_hotpoint.py 脚本

# 参数192.168.1.8:8888/IDEAmaven，表示访问的网址，测试脚本会拼接为完整的路径

python registerTest\_hotpoint.py 192.168.1.8:8888/IDEAmaven

* 保存后，测试任务配置完毕

### 构建maven项目：

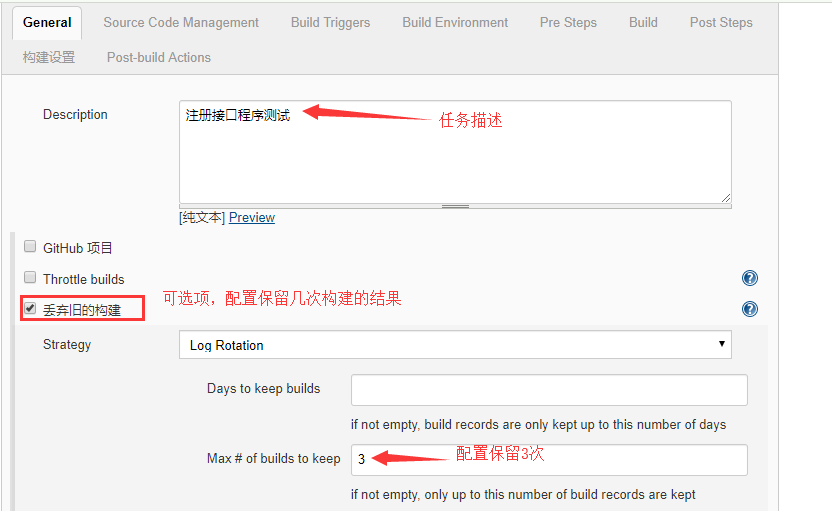
单击new 任务:



输入任务名称：比如interface

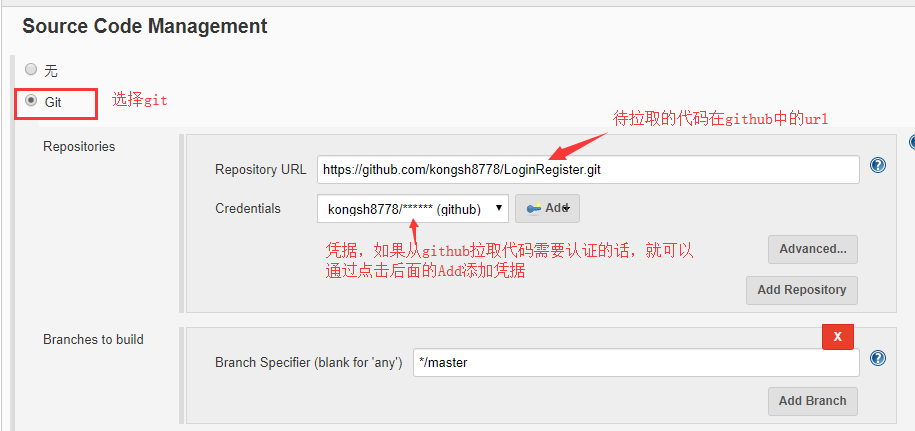


**General:**



**Source Code Management:**

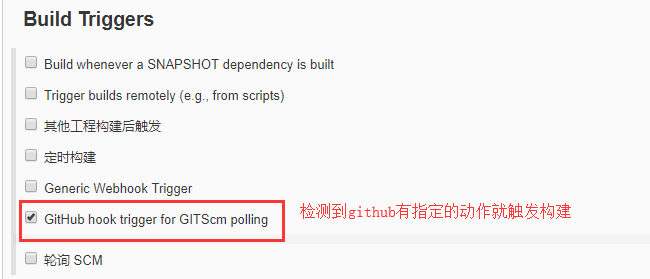
<https://github.com/kongsh8778/LoginRegister.git>



可选配置：

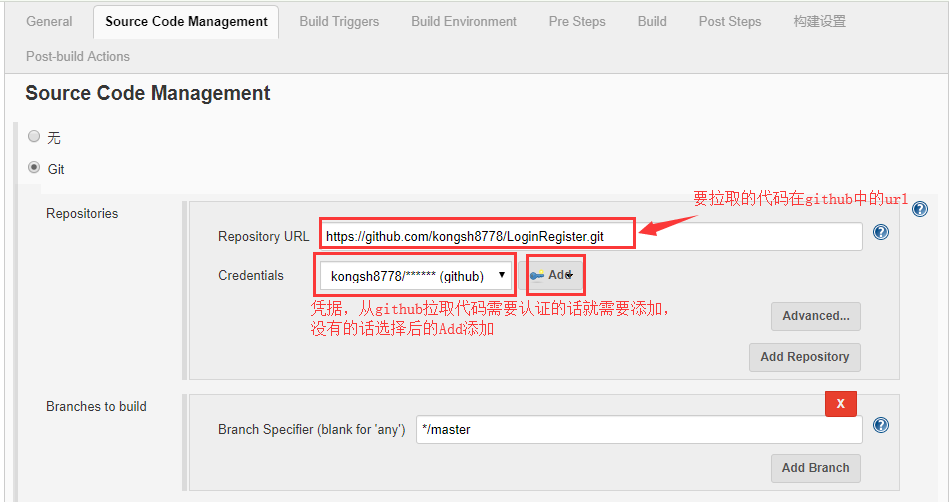
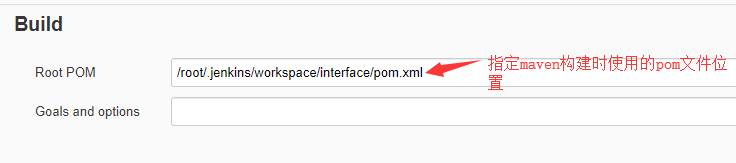


**Build Triggers:**

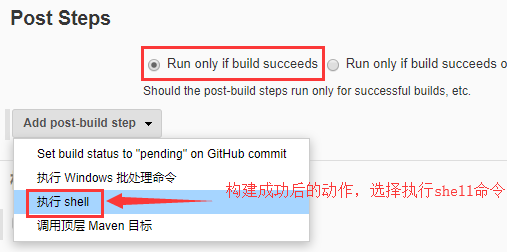


**Build:**

/root/.jenkins/workspace/interface/pom.xml



**Post Steps:**



输入以下命令：

#!/bin/bash

#copy file and restart tomcat

export BUILD\_ID=dontKillMe

#我的tomcat路径

tomcat\_path=/usr/local/tomcat

#我的war包名称

war\_name=IDEAmaven.war

#jenkins默认工作空间，构建的时候jenkins会从git服务器上把项目源码下载到workspace路径下

file\_path=/root/.jenkins/workspace/interface\_copy/target

tomcat\_name='apache-tomcat-8.5.38' #tomcat 完整路劲 或 最后级路劲文件夹都可以

#获取 tomcat\_path 所知tomcat 的进程ID

TomcatID=$(ps -ef |grep tomcat |grep -w $tomcat\_name|grep -v 'grep'|awk '{print $2}')

echo $TomcatID $tomcat\_name

#判断进程是否存在

if [ $TomcatID ];

then

echo "$TomcatID tomcat is starting ................."

#关闭tomcat

$tomcat\_path/bin/shutdown.sh

echo "tomcat stop..................................."

else

echo "tomcat not start ............................."

fi

#休眠3秒

sleep 3s

echo "rm -rf ${tomcat\_path}/webapps/${war\_name}"

#删除tomcat目录下老的war包和解压后的项目目录

rm -rf ${tomcat\_path}/webapps/${war\_name}\*

cd $file\_path

#复制jenkins工作空间中生成的war包到tomcat的webapps目录下

cp ${war\_name} ${tomcat\_path}/webapps/

sleep 5s

#$tomcat\_path/bin/catalina.sh

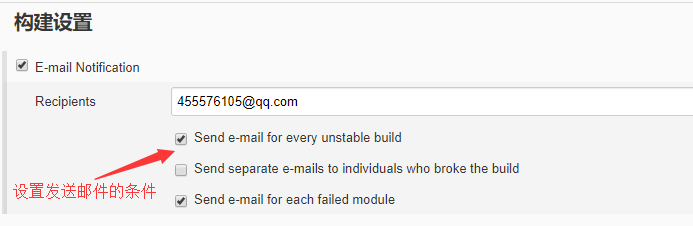
#重启tomcat

$tomcat\_path/bin/catalina.sh

echo "tomcat server restarted"



**构建设置：**



**Post-build Actions：**

选择构建其它工程，当本任务构建完成后自动出发触发指定测试任务的执行：



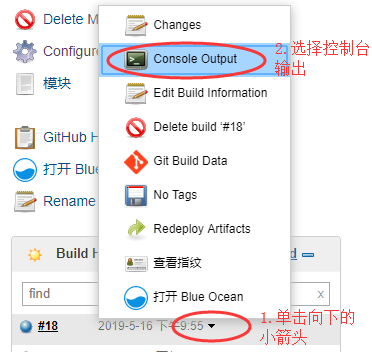
在jenkins首页可以看到所有构建的项目：



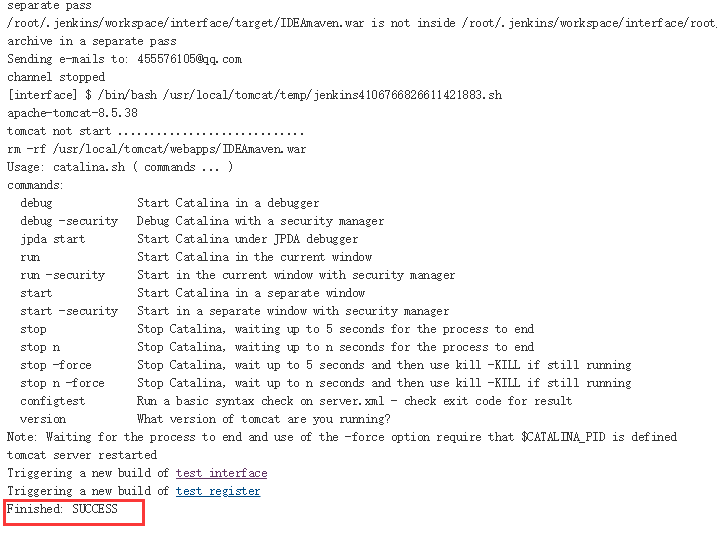
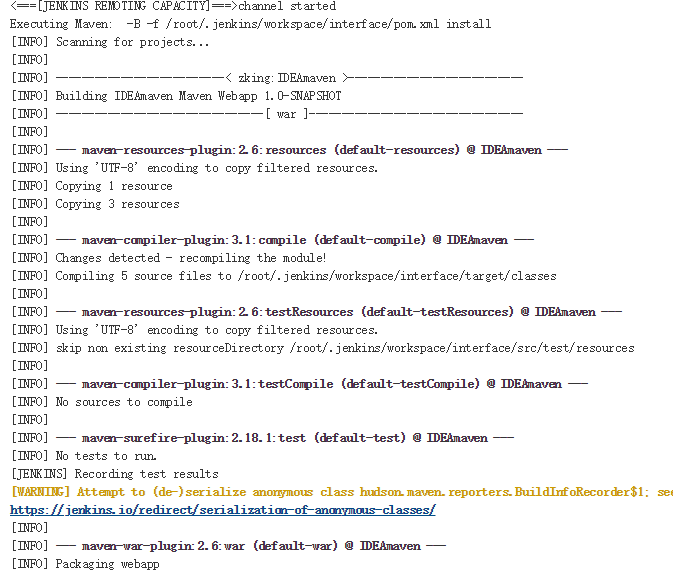
红色表示构建失败；蓝色表示构建成功



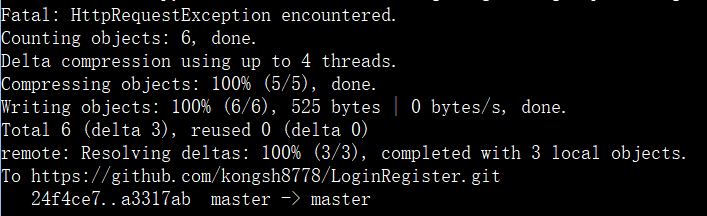
查看详细的构建过程：



看到构建过程中的信息：



### 修改web工程代码并添加到Github

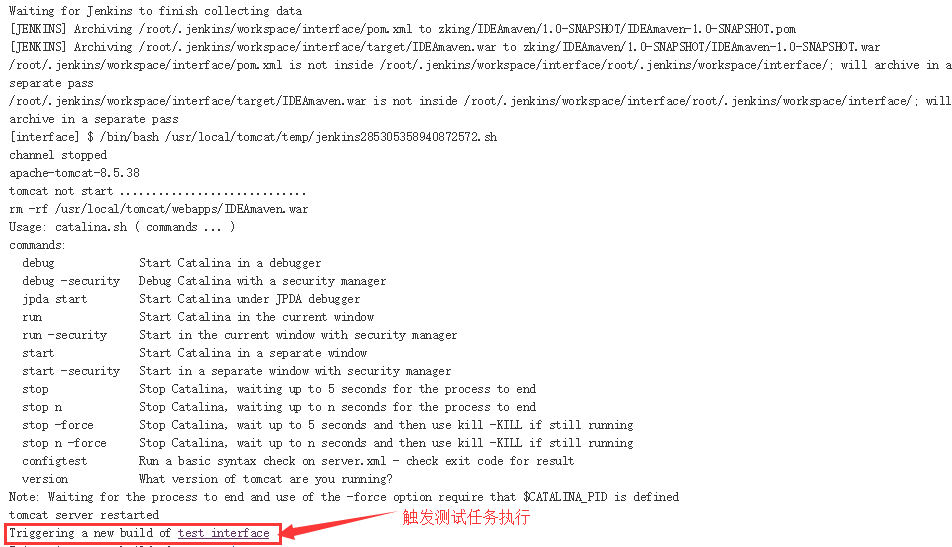
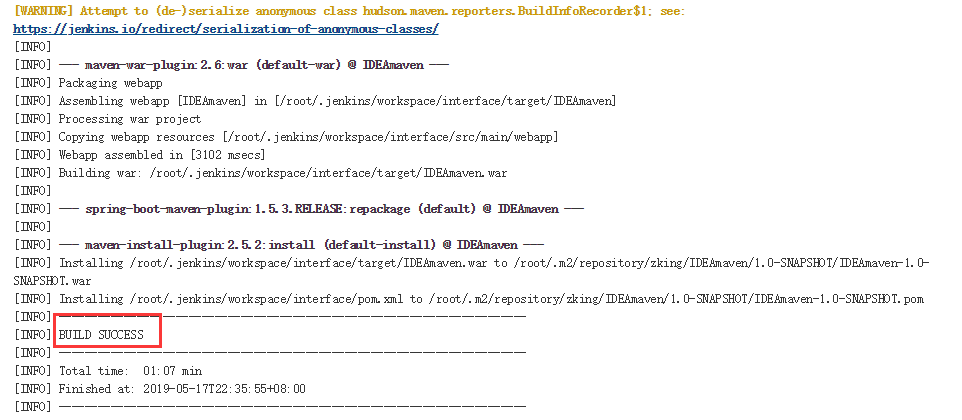
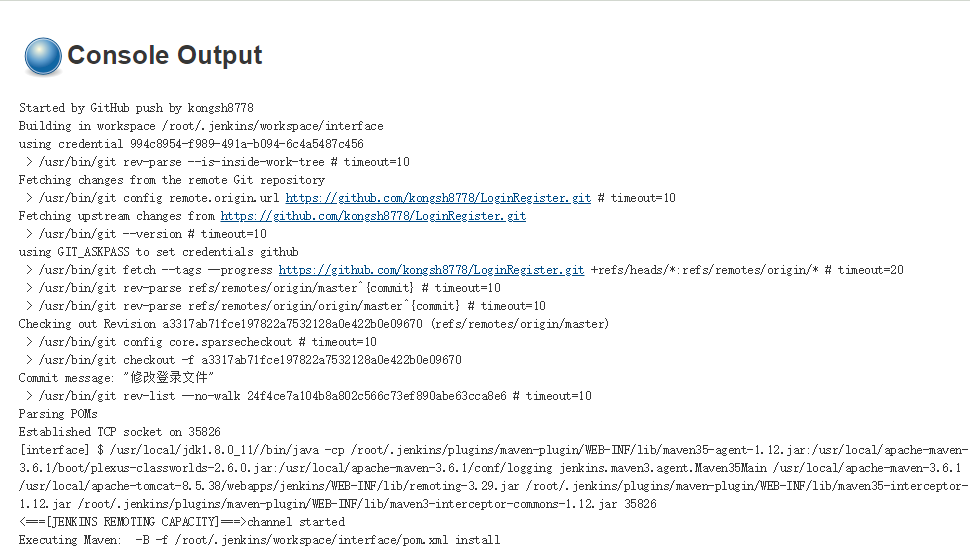


### jenkins自动触发maven构建：

触发10.1.2建立的任务：



执行日志如下：



### 触发测试脚本执行

触发的是10.1.1建立的测试任务



执行日志如下：

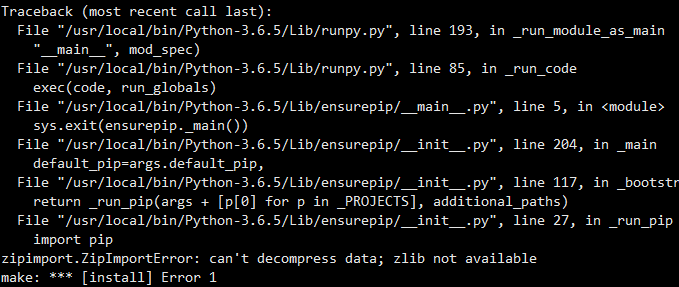


### 邮件发送的构建日志



## 问题汇总

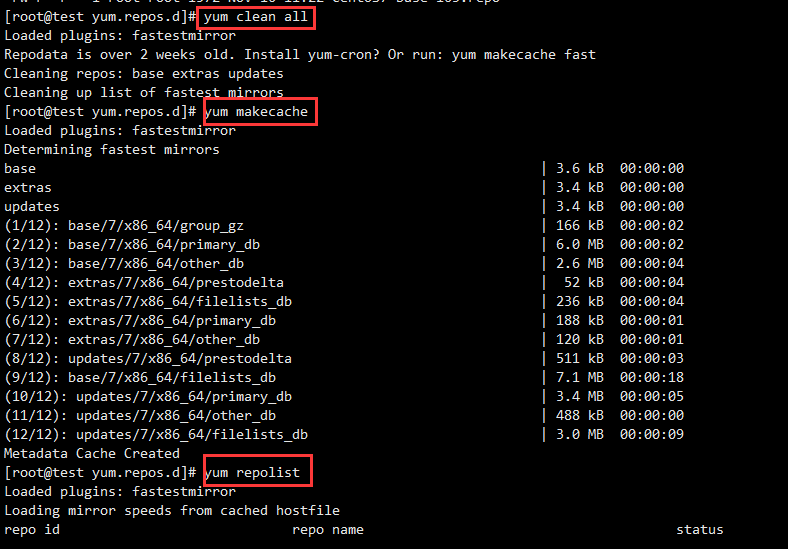
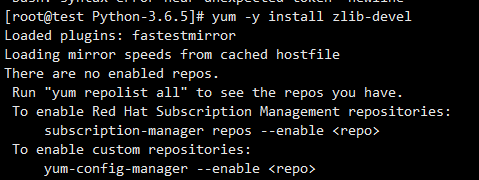
### 编译python出现"zipimport.ZipImportError"错误问题



这个问题主要就是包还未安装，解决方法：yum -y install zlib\*

### 重装yum后不可用

解决方法参考：<https://blog.csdn.net/qq_39720249/article/details/84104646>



### 误删了Python2导致yum不能使用

参考链接：

https://www.jianshu.com/p/89df82a5d74b

https://blog.csdn.net/huanyingfengxing/article/details/79834960

mkdir /usr/local/src/python

cd /usr/local/src/python

centos7需要下载的rpm文件列表

wget http://mirrors.163.com/centos/7.6.1810/os/x86\_64/Packages/python-libs-2.7.5-76.el7.x86\_64.rpm

wget http://mirrors.163.com/centos/7.6.1810/os/x86\_64/Packages/python-2.7.5-76.el7.x86\_64.rpm

wget http://mirrors.163.com/centos/7.6.1810/os/x86\_64/Packages/python-iniparse-0.4-9.el7.noarch.rpm

wget http://mirrors.163.com/centos/7.6.1810/os/x86\_64/Packages/python-pycurl-7.19.0-19.el7.x86\_64.rpm

wget http://mirrors.163.com/centos/7.6.1810/os/x86\_64/Packages/python-urlgrabber-3.10-9.el7.noarch.rpm

wget http://mirrors.163.com/centos/7.6.1810/os/x86\_64/Packages/rpm-python-4.11.3-35.el7.x86\_64.rpm

安装以上所有的包：rpm -ivh xxx.rpm，注意：python-libs-2.7.5-76.el7.x86\_64.rpm一定要先安装

centos7需要下载yum安装包：

就是它依赖了上面的python库

wget http://mirrors.163.com/centos/7.6.1810/os/x86\_64/Packages/yum-3.4.3-161.el7.centos.noarch.rpm

wget http://mirrors.163.com/centos/7.6.1810/os/x86\_64/Packages/yum-metadata-parser-1.1.4-10.el7.x86\_64.rpm

wget http://mirrors.163.com/centos/7.6.1810/os/x86\_64/Packages/yum-plugin-fastestmirror-1.1.31-50.el7.noarch.rpm

安装以上所有的包：rpm -ivh xxx.rpm，注意：第1个包和第3个包一定要一起装

[root@test python]# rpm -ivh yum-3.4.3-161.el7.centos.noarch.rpm yum-plugin-fastestmirror-1.1.31-50.el7.noarch.rpm

不然提示以下错误



查看yum安装成功：yum

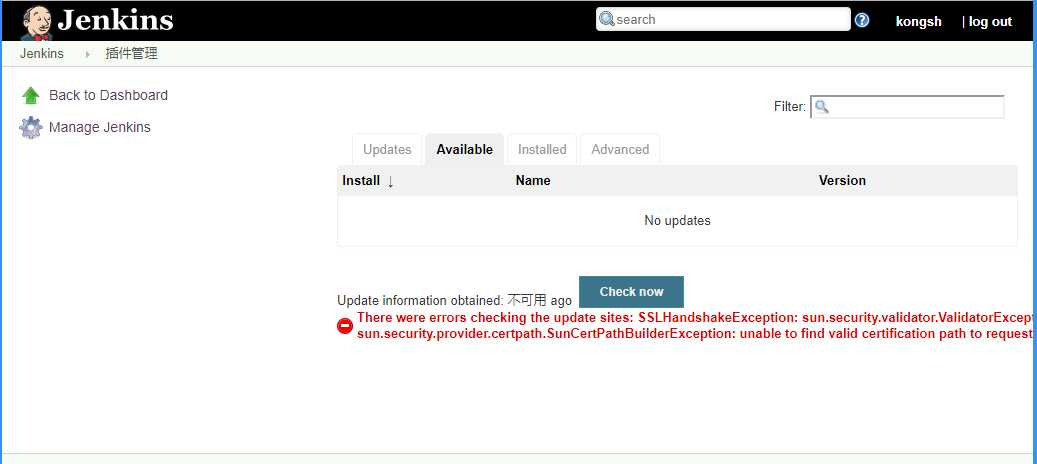
查看python版本：python

### pip is configured with locations that require TLS/SSL, however the ssl module in Python is not available.

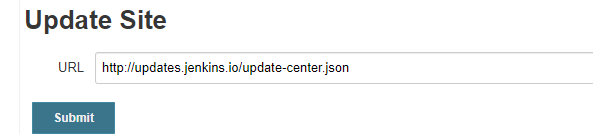
<https://jingyan.baidu.com/article/cbf0e500475c042eab289362.html>

### jenkins插件不可以用

<https://www.jianshu.com/p/ca85e847fc34>



将https改为http，如下：



**发送邮件需要先安装Email Extension Plugin插件**

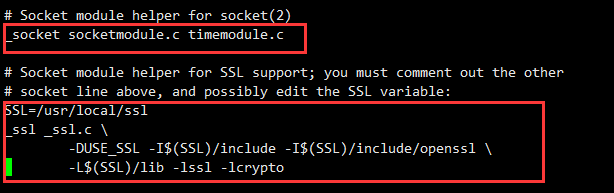
### 解决jenkins git timeout的问题

<https://www.jianshu.com/p/264772bb9264>

[root@test Modules]# pwd

/usr/local/bin/Python-3.6.5/Modules

[root@test Modules]# vi Setup.dist



### jenkis构建[“ERROR: Error fetching remote repo 'origin'”问题](https://www.cnblogs.com/feng0520/p/9070071.html)



点击清理工作空间

这个问题被坑了很久，原因是github文件太大，导致clone的时间非常长，设置超时时间为40分钟也不能成功。解决方法就是命令行后面加--depth=1，就会只clone最后一次commit的内容：

### pom.xml不可用

写成绝对路径

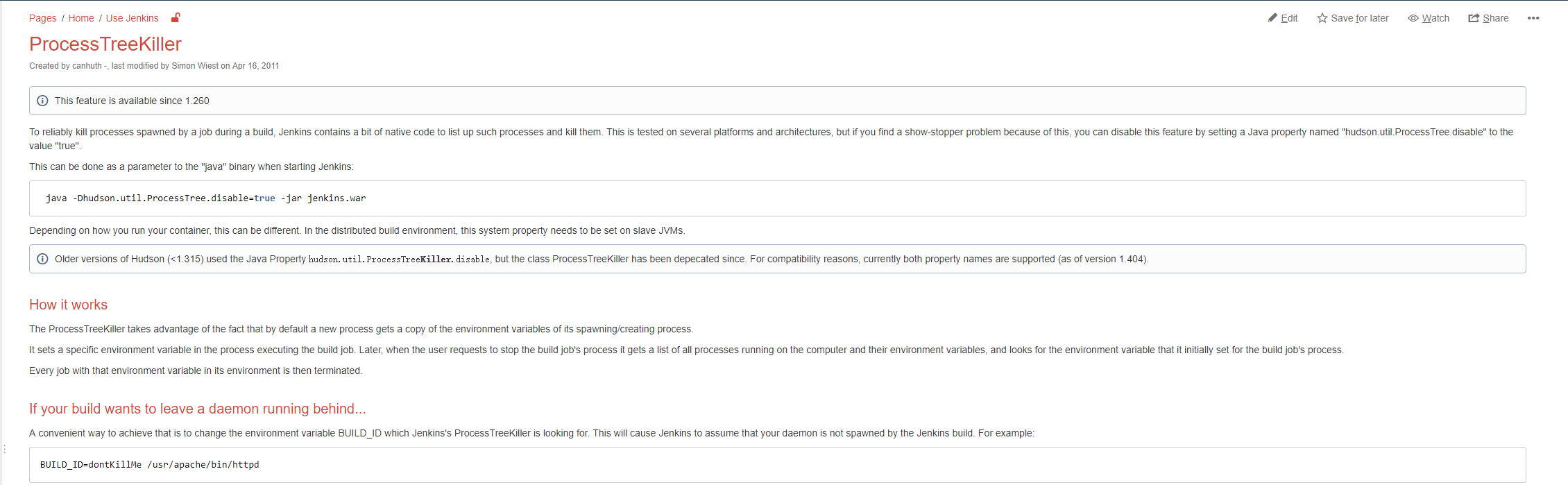
### linux下启动war包后，window下浏览器不能访问

可能的原因：

1. linux防火墙要关闭
2. 使用的是tomcat src的包，需要使用core

### jenkins杀死启动的tomcat进程

<https://blog.csdn.net/u011781521/article/details/80210985>



jenkins构建job成功后，ps -ef查不到相关的进程，导致服务不能访问，原因是jenkins构建成功后会将所有的进程杀掉，解决方法：

启动jenkins是加上java -Dhudson.util.ProcessTree.disable=true -jar

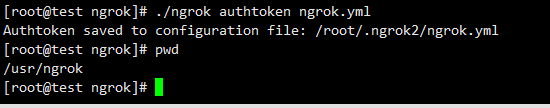
### could not conntact to server

**If you want to try to run Jenkins on localhost, the other way around is that, install ngrok:**

<https://www.cnblogs.com/Javame/p/9880888.html>

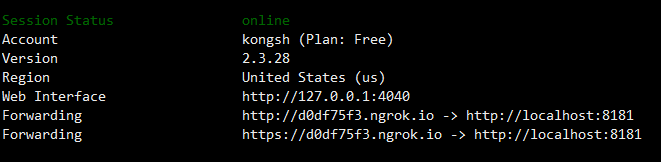
登录到<https://ngrok.com/download>，下载ngrok，将IP暴露到网络上才能发送通知

./ngrok authtoken 3LoDHiNZAGSc3Cdi9ThEW\_75ZCKKBh73UCHDc4WdAy



cd /usr/ngrok/

./ngrok http 8181

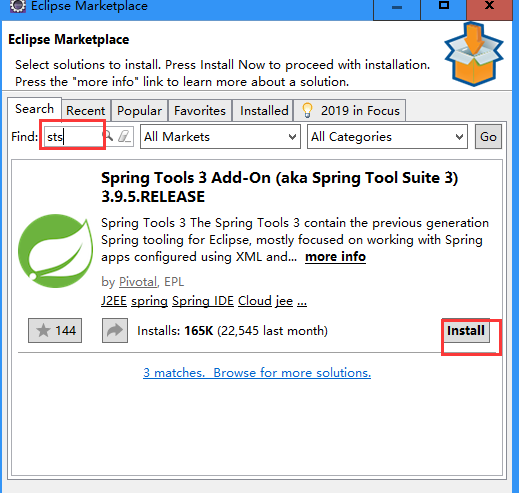


### 启动程序没有报错，但是ps -ef看不到进程

解决方法：执行./catalina.sh run在当前控制台打印出日志信息，查看报错信息

### eclipse配置maven

<https://blog.csdn.net/qq_42670048/article/details/83155061>



### eclipse创建springboot项目

安装插件导向窗口完成后，在eclipse右下角将会出现安装插件的进度，等插件安装完成后重启eclipse生效

@SpringBootApplication无法被解析引入。

<https://blog.csdn.net/xujian_2001/article/details/78894833>

### eclipse 忽略html报错

<https://blog.csdn.net/qq_27833059/article/details/72953820>

<http://127.0.0.1:8080/front/register>