# 在[CentOS](https://www.linuxidc.com/topicnews.aspx?tid=14" \t "_blank" \o "CentOS) 7上安装Jenkins

### 安装

添加yum repos，然后安装

sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/[RedHat](https://www.linuxidc.com/topicnews.aspx?tid=10)/jenkins.repo

sudo rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key

sudo yum install jenkins

### 如果没有java的话要安装java

sudo yum install java

### 启动和停止

sudo service jenkins start/stop/restart

sudo chkconfig jenkins on

### jenkins的默认设置

* Jenkins会随系统启动而启动。详情参照/etc/init.d/jenkins
* Jenkins会创建一个用户叫做jenkins, 如果你修改了user，则要修修改所属者：/var/log/jenkins,/var/lib/jenkins,/var/cache/jenkins
* 如果遇到问题，查看日志/var/log/jenkins/jenkins.log
* 配置文件/etc/sysconfig/jenkins
* 默认启用8080

### 打开和关闭防火墙

firewall-cmd --zone=public --add-port=8080/tcp --permanent

firewall-cmd --zone=public --add-service=http --permanent

firewall-cmd --reload

firewall-cmd --list-all

### 关于centos上的java

Jenkins不支持在centos的默认的jdk上工作。如果如下所示，则需要remove：

java -version

java version "1.5.0"

gij (GNU libgcj) version 4.4.6 20110731 (Red Hat 4.4.6-3)

为了正确使用Jenkins：

yum remove java

然后可以安装openjdk

yum install java-1.7.0-openjdk

正确如下：

java -version

java version "1.7.0\_79"

OpenJDK Runtime Environment (rhel-2.5.5.1.el6\_6-x86\_64 u79-b14)

OpenJDK 64-Bit Server VM (build 24.79-b02, mixed mode)

### 配置端口

修改/etc/sysconfig/jenkins:

JENKINS\_PORT="8080"

### 配置java路径

直接启动：

sudo service jenkins start

但是发现启动失败，于是需要配置java位置：

vi /etc/init.d/jenkins

在启动加入本机的java：

[root@iZ94xvn71eeZ ~]# vim /etc/init.d/jenkins

# Set up environment accordingly to the configuration settings

[ -n "$JENKINS\_HOME" ] || { echo "JENKINS\_HOME not configured in $JENKINS\_CONFIG";

if [ "$1" = "stop" ]; then exit 0;

else exit 6; fi; }

[ -d "$JENKINS\_HOME" ] || { echo "JENKINS\_HOME directory does not exist: $JENKINS\_HOME";

if [ "$1" = "stop" ]; then exit 0;

else exit 1; fi; }

# Search usable Java as /usr/bin/java might not point to minimal version required by Jenkins.

# see http://www.nabble.com/guinea-pigs-wanted-----Hudson-RPM-for-RedHat-Linux-td25673707.html

candidates="

/etc/alternatives/java

/usr/lib/jvm/java-1.8.0/bin/java

/usr/lib/jvm/jre-1.8.0/bin/java

/usr/lib/jvm/java-1.7.0/bin/java

/usr/lib/jvm/jre-1.7.0/bin/java

/usr/bin/java

/usr/java/jdk1.7.0\_80/bin/java

"

上述最后一行服务器java所在目录。

重启，可以正常运行了。

### 启动方式

### 服务启动

service jenkins start

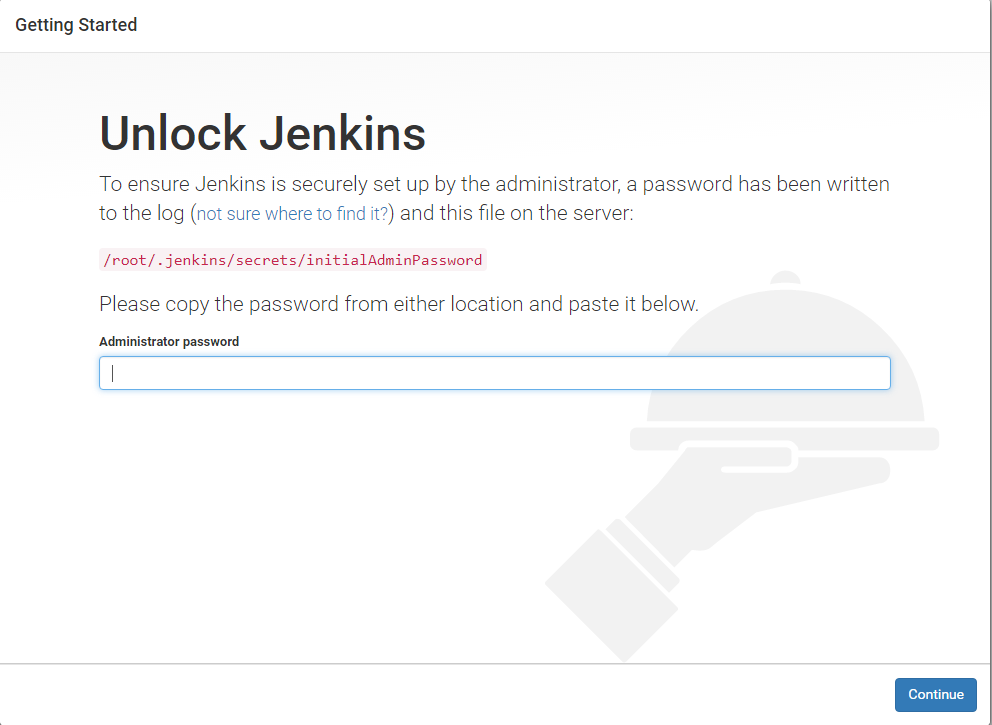
### 部署到tomcat

将/usr/lib/jenkins/jenkins.war部署到tomcat中

# Jenkins安装

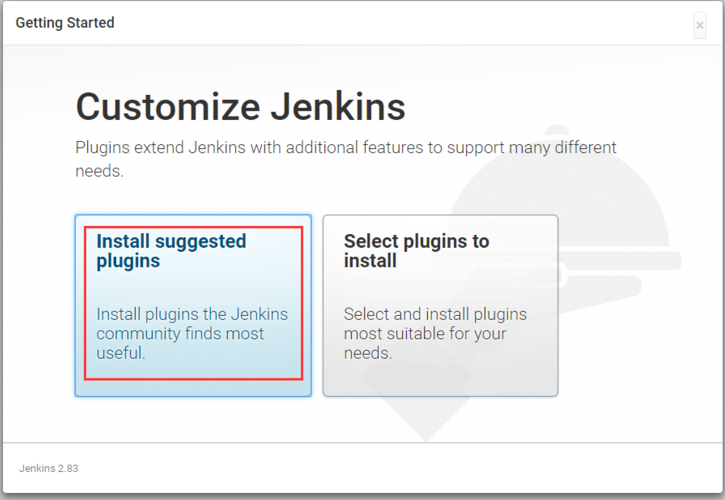
### 安装

服务启动后，访问IP:上边配置的端口，根据提示找到超级密码

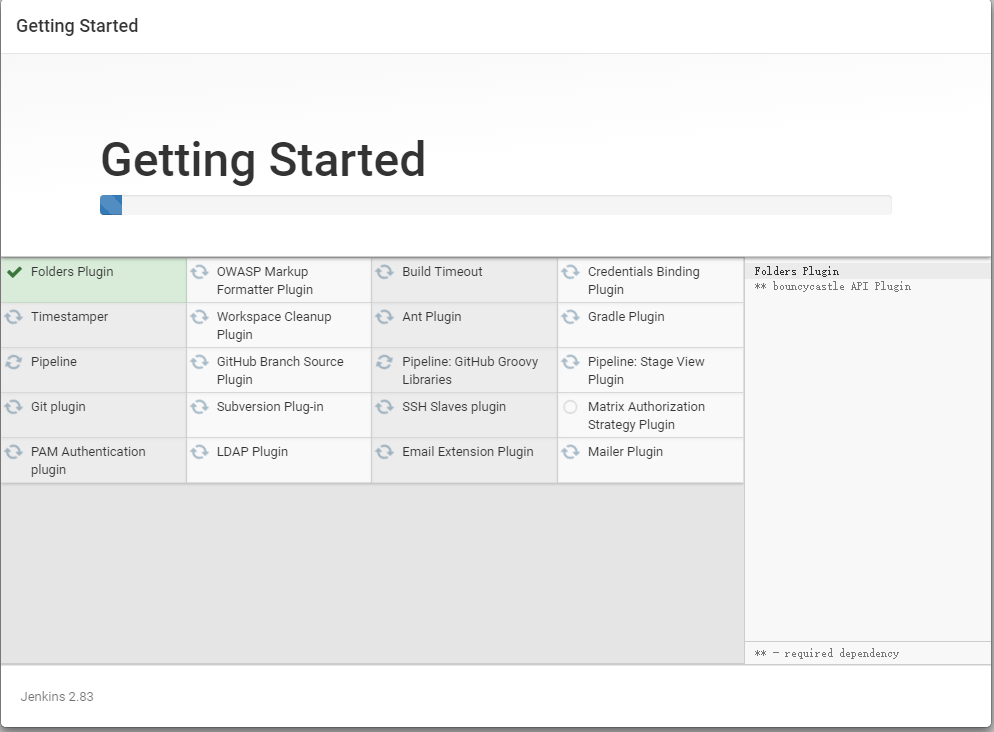


在/root/.jenkins/secrets/initialAdminPassword文件中找到超级密码，输入进行下一步。

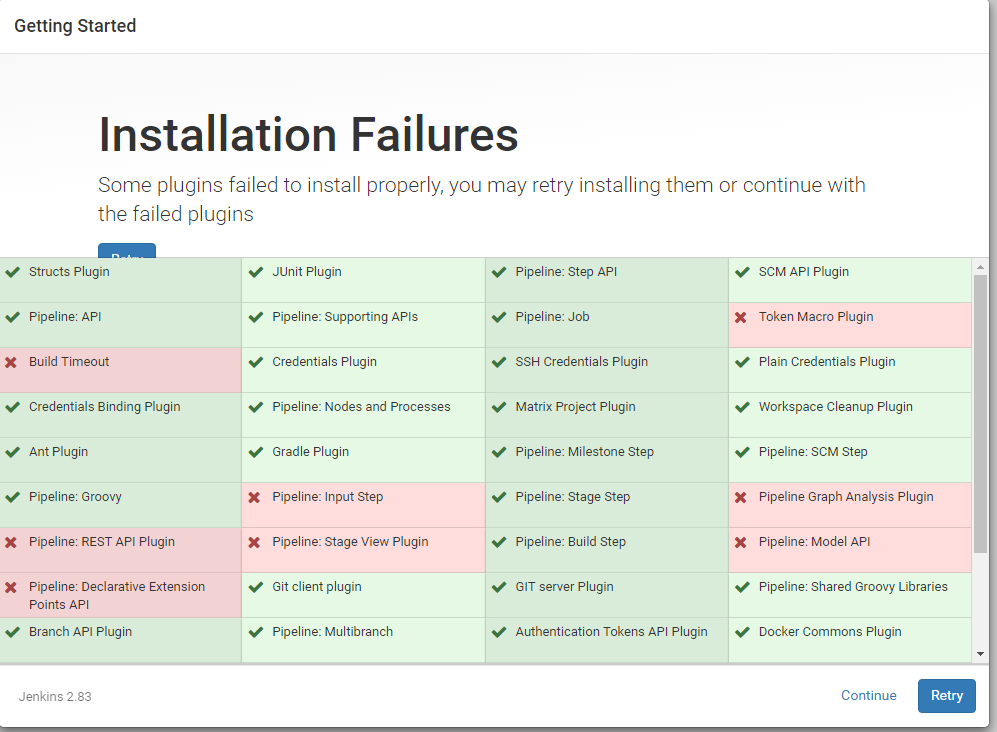
进入用户自定义插件界面，建议选择安装官方推荐插件，因为安装后自己也得安装:



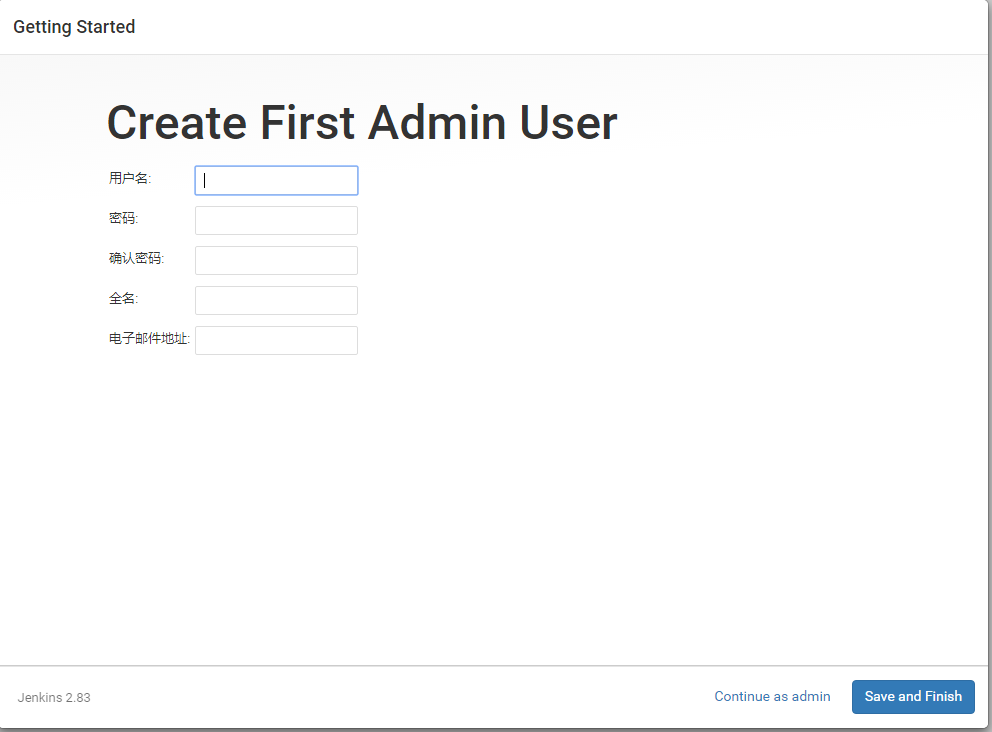
接下来是进入插件安装进度界面:



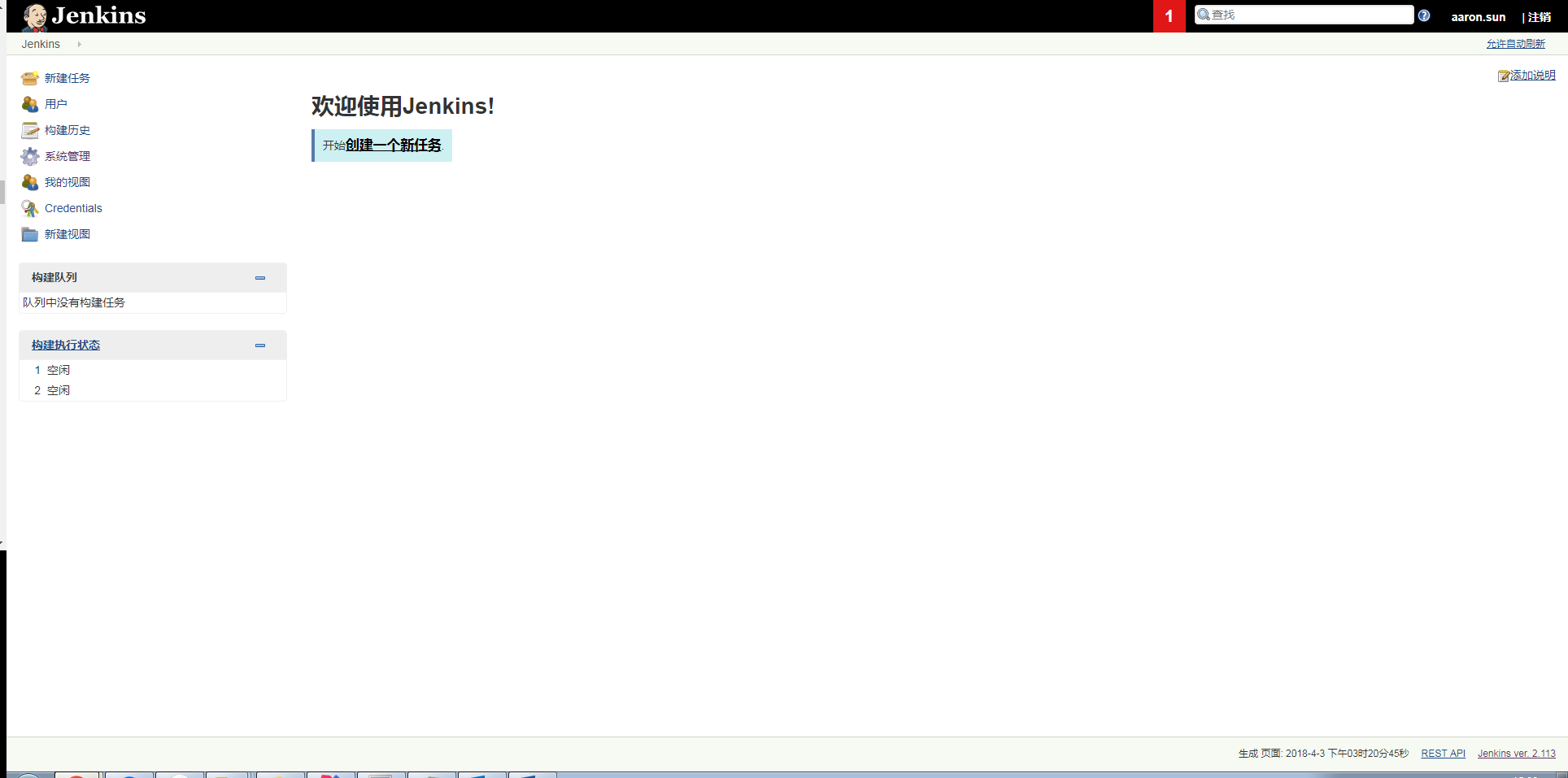
插件一次可能不会完全安装成功，可以点击Retry再次安装。直到全部安装成功



等待一段时间之后，插件安装完成，配置用户名密码:



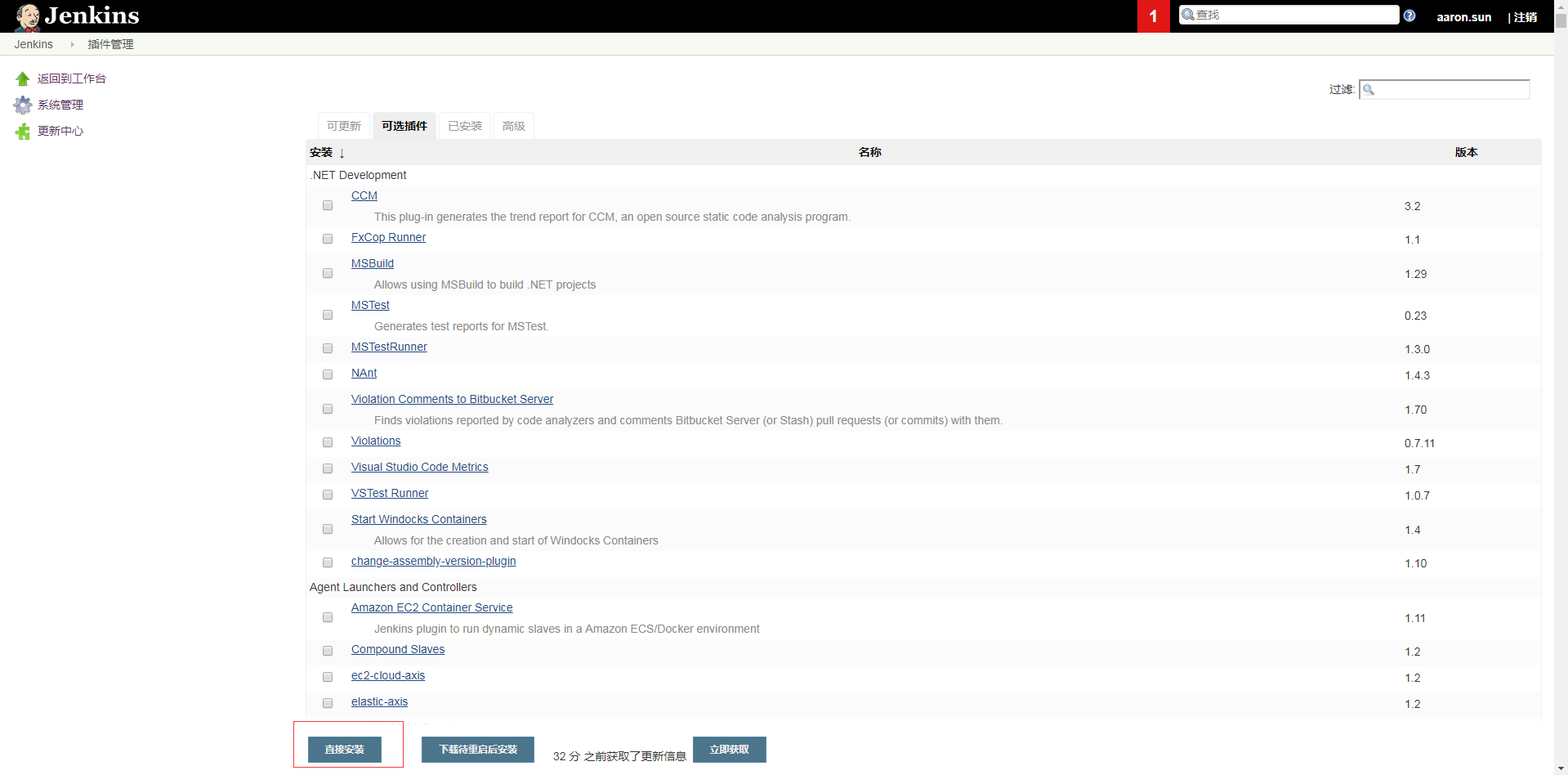
进入首页，Jenkins安装完成



# Jenkins插件安装和配置

有很多插件都是选择的默认的安装的，所以现在需要我们安装的插件不多，Git plugin和Maven Integration plugin，publish over SSH、SSH插件。

插件安装：系统管理 > 插件管理 > 可选插件,勾选需要安装的插件，点击直接安装或者下载重启后安装

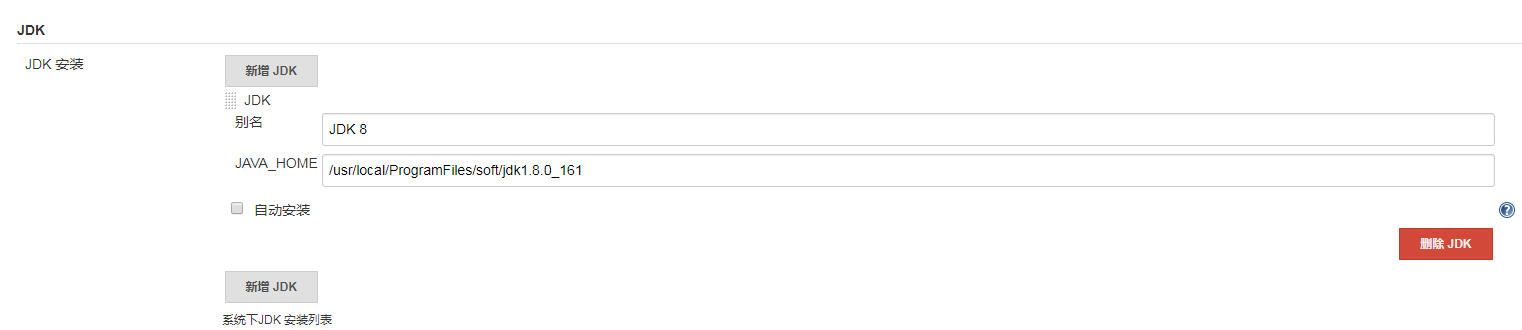


### 全局变量配置

系统管理 > 全局工具配置

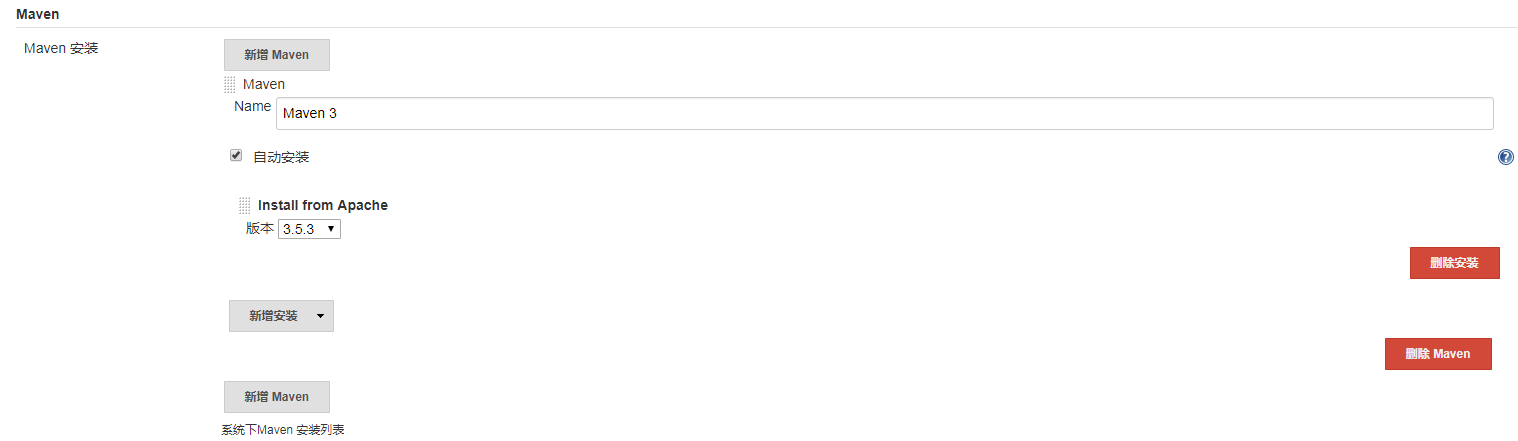
**JDK**

配置本地JDK的路径，去掉勾选自动安装



**Maven**

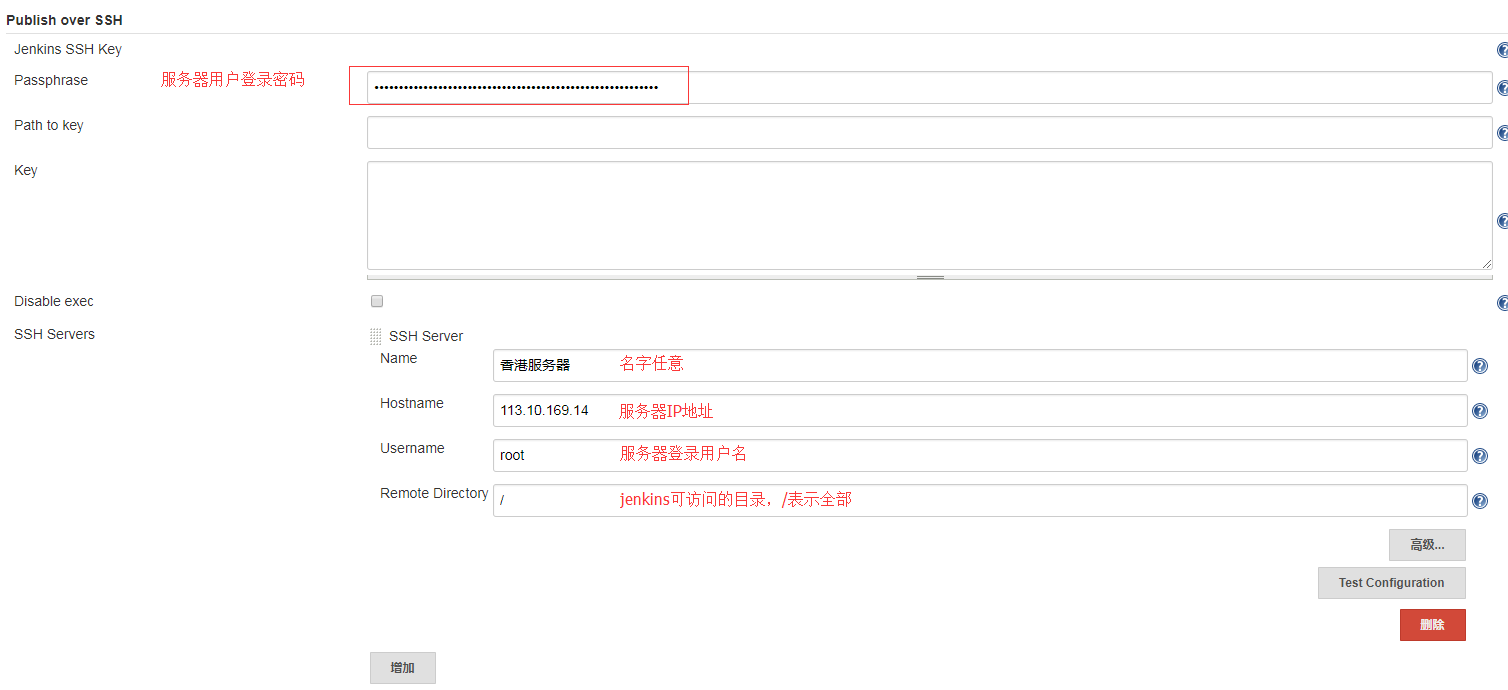
此处可自行安装，小编使用的自动安装



### Push SSH

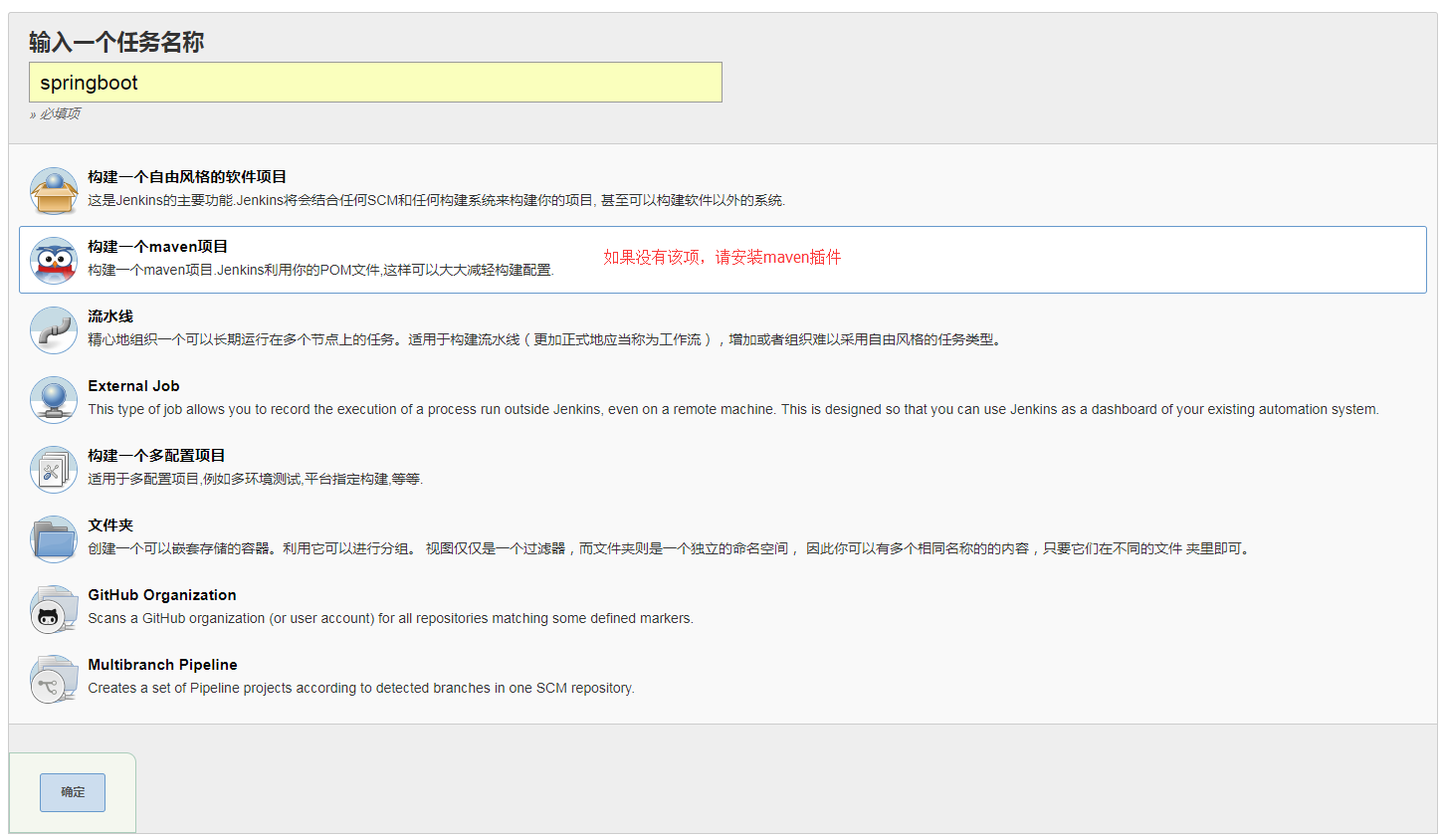
系统管理 > 系统设置

选择 Publish over SSH

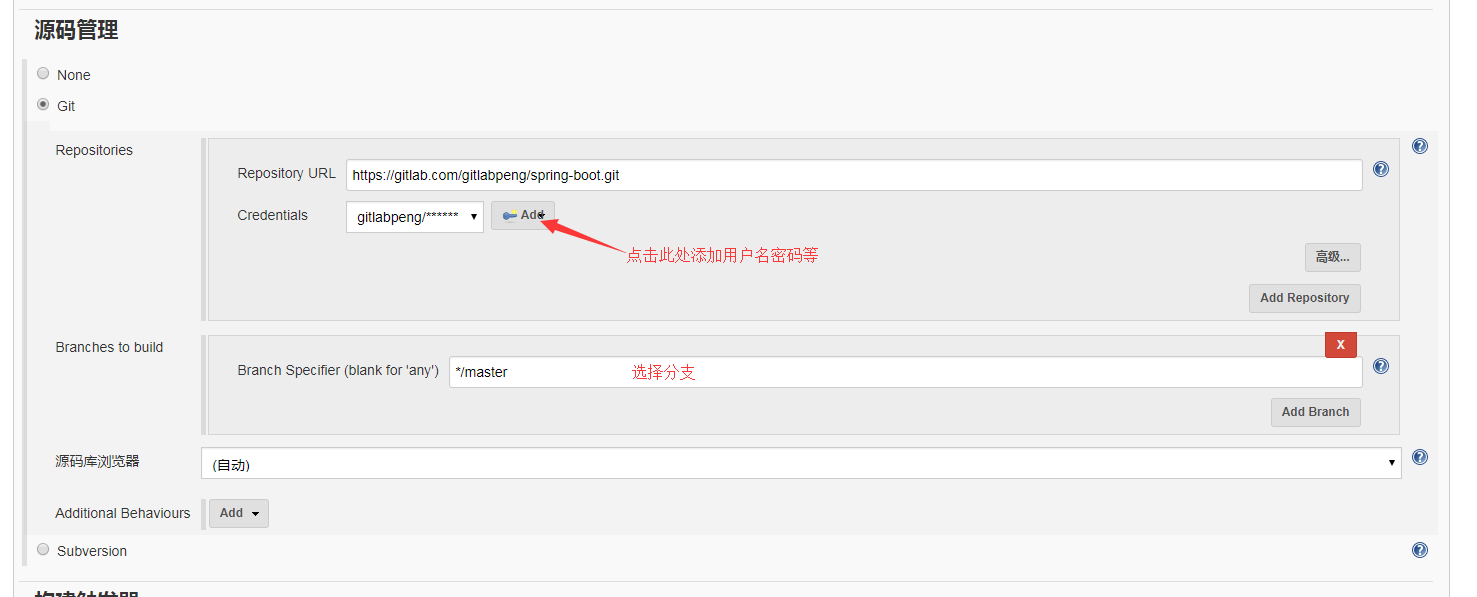


# Jenkins添加gitlab上的spring boot项目

### 新建Maven项目



### 源码管理



### 构建环境

构建环境中勾选“Add timestamps to the Console Output”，代码构建的过程中会将日志打印出来

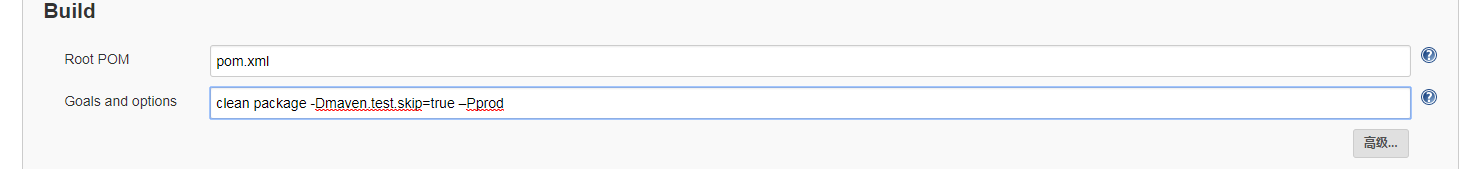


### Build

在Build中输入打包前的mvn命令，如：

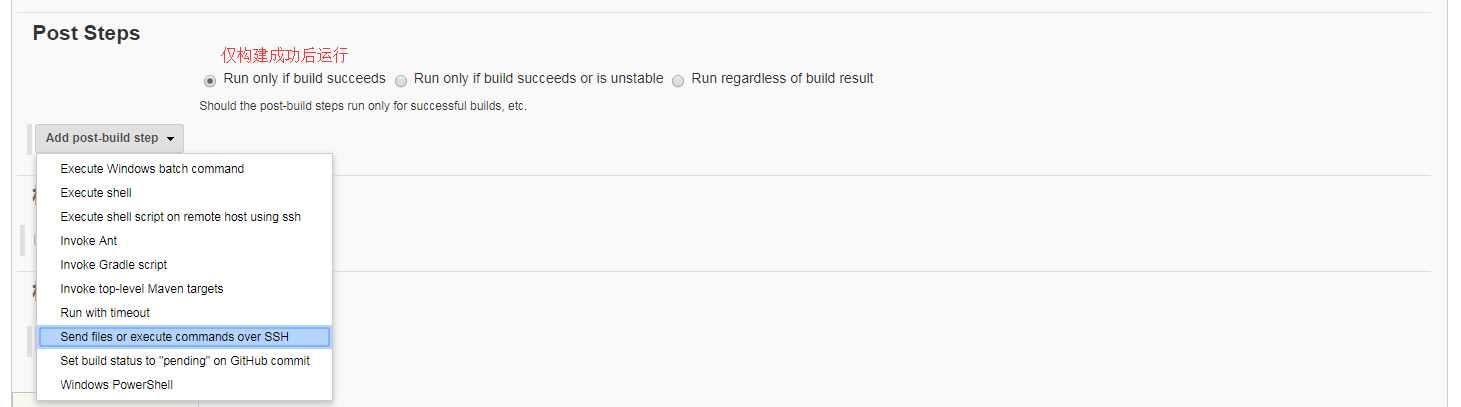
clean package -Dmaven.test.skip=true –Pprod

其中：-Dmaven.test.skip=true 是跳过测试，-Pprod表示打包时使用prod后缀的资源文件。

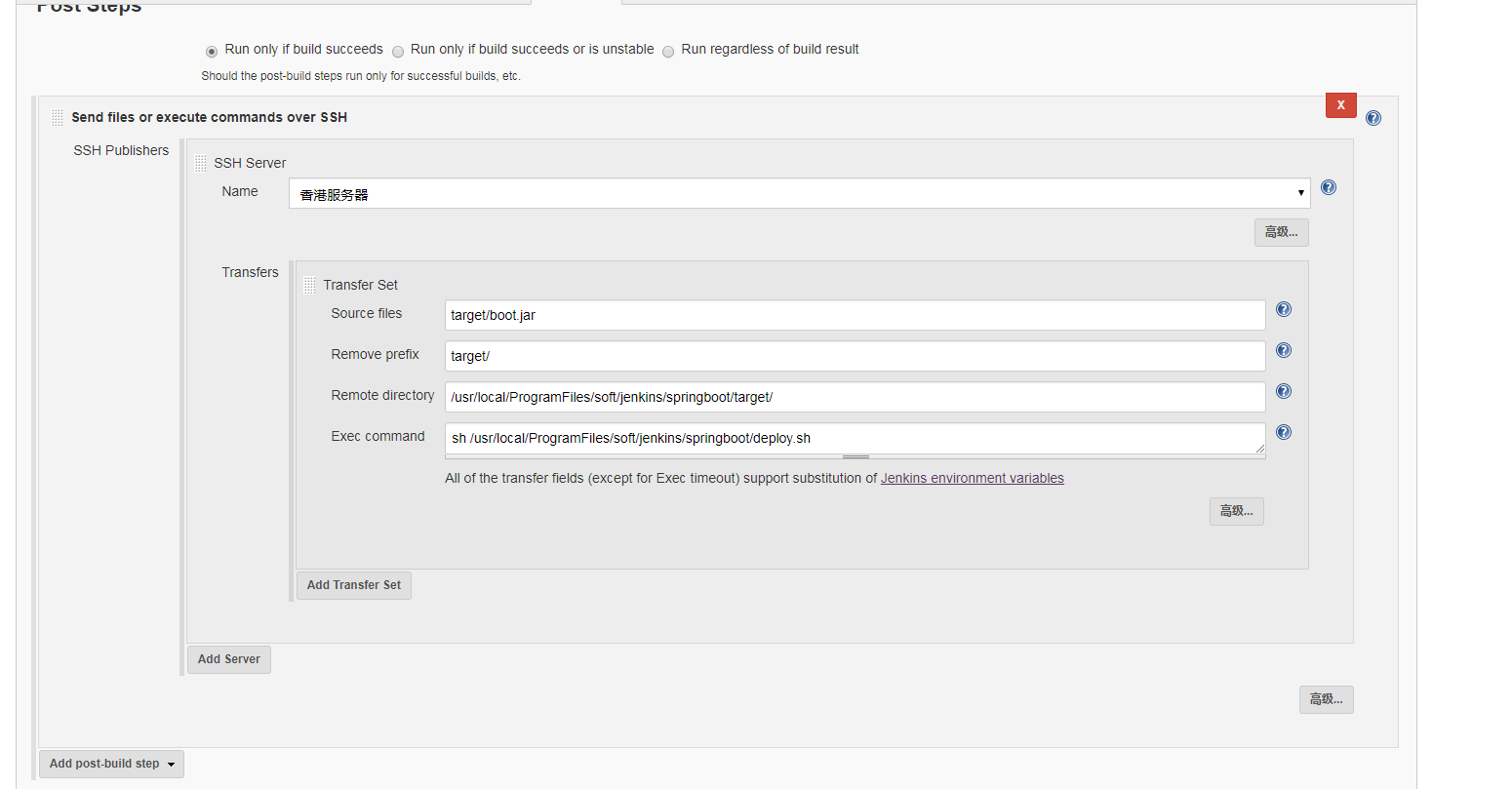


### Post Step

点击Add post-build step，选择 Send files or execute commands over SSH



Name选择之前配置过的Push SSH



Source files配置: target/xxx.jar 项目jar包名  
Remove prefix: target/  
Remote directory: 代码应用服务器的目录地址，  
Exec command：应用服务器对应的脚本。

xxx.sh脚本内容

DATE=$(date +%Y%m%d)

export JAVA\_HOME PATH CLASSPATH

JAVA\_HOME=/usr/local/ProgramFiles/soft/jdk1.8.0\_161

PATH=$JAVA\_HOME/bin:$JAVA\_HOME/jre/bin:$PATH

CLASSPATH=.:$JAVA\_HOME/lib:$JAVA\_HOME/jre/lib:$CLASSPATH

DIR=/usr/local/ProgramFiles/soft/jenkins/springboot

TARGET\_DIR=$DIR/target

JARFILE=boot.jar

PID=`ps -ef | grep $JARFILE | grep -v grep | awk '{print $2}'`

if [ ! -d $DIR/backup ];then

mkdir -p $DIR/backup

fi

cd $DIR

echo "PID:$PID 开始关闭项目"

kill -9 $PID

echo "项目已关闭"

echo "备份$JARFILE"

if [ ! -f "$JARFILE" ]; then

echo "$JARFILE 文件不存在"

else

mv $JARFILE backup/$DATE$JARFILE

echo "备份完成，备份路径：$DIR/backup"

fi

echo "替换$JARFILE"

mv -f $TARGET\_DIR/$JARFILE $DIR

echo "替换完成，正在启动，请稍后..."

java -jar $JARFILE > out.log &

if [ $? = 0 ];then

sleep 30

tail -n 50 out.log

fi

cd backup/

ls -lt|awk 'NR>5{print $NF}'|xargs rm –rf

这段脚本的意思，就是kill旧项目，删除旧项目，启动新项目，备份老项目。