**持续集成环境搭建及使用操作手册**

2017年08月01日

**文档信息及修订记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本** | **修订说明** | **日期** | **编写人** | **审核人** |
| 1.0 | 形成初稿 | 2017/08/01 |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

版权申明：

# 引言

## 编写目的

《持续集成环境搭建及使用操作手册》是持续集成系统搭建及使用书册，为系统实施阶段提供指导与参考。

预期读者：

系统运维人员

## 定义

列出本文件中用到的专门术语的定义和缩写词的原词组。

## 参考资料

<https://wiki.jenkins.io/display/JENKINS/Installing+Jenkins+on+Red+Hat+distributions>

[引言 3](#_Toc500243260)

[编写目的 3](#_Toc500243261)

[定义 3](#_Toc500243262)

[参考资料 3](#_Toc500243263)

[1.环境准备 4](#_Toc500243264)

[1.1 JDK安装 4](#_Toc500243265)

[1.2 git安装 5](#_Toc500243266)

[2.安装持续集成 5](#_Toc500243267)

[2.1 安装jenkins 5](#_Toc500243268)

[2.1.1 安装稳定版本 5](#_Toc500243269)

[2.1.2 启动与停止jenkins 6](#_Toc500243270)

[2.1.3 安装包做什么 6](#_Toc500243271)

[2.1.4 登录jenkins 6](#_Toc500243272)

[2.1.5 安装jenkins基础插件 11](#_Toc500243273)

[2.1.5.1 安装maven项目插件 13](#_Toc500243274)

[2.1.5.2安装git插件 17](#_Toc500243275)

[2.1.5.3安装ant插件 18](#_Toc500243276)

[2.1.5.4安装External Job插件 19](#_Toc500243277)

[2.1.5.5安装ssh发布插件 20](#_Toc500243278)

[2.1.5.6安装Copy Artifact Plugin 20](#_Toc500243279)

[2.1.5.7安装SSH plugin插件 21](#_Toc500243280)

[3. 配置项目 22](#_Toc500243281)

[3.1.创建Job 22](#_Toc500243282)

[3.1.1 顶级父项目 22](#_Toc500243283)

[3.1.2 plugins项目配置 23](#_Toc500243284)

[3.1.3 usercertification(实名服务) 25](#_Toc500243285)

[3.1.4 usercertification(实名服务)还原job 28](#_Toc500243286)

[3.1.5 bankcard(卡绑定服务) 32](#_Toc500243287)

[3.1.4 bankcard(卡绑定服务)还原job 34](#_Toc500243288)

[3.1.5 cashier(收银台) 36](#_Toc500243289)

[3.1.6 cashier(收银台)还原job 37](#_Toc500243290)

[3.2配置多服务器发布 38](#_Toc500243291)

[3.3 拆分打包与发布 40](#_Toc500243292)

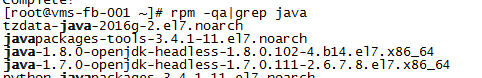
# 1.环境准备

## 1.1 JDK安装

为了运行jenkins,需要安装jdk：

先查看原本原机器是否安装了jdk,运行以下命令：

rpm -qa|grep java



运行卸载命令：

rpm -e --nodeps java-1.8.0-openjdk-headless-1.8.0.102-4.b14.el7.x86\_64

rpm -e --nodeps java-1.7.0-openjdk-headless-1.7.0.111-2.6.7.8.el7.x86\_64

再次运行rpm -qa|grep java，查看是否卸载成功

下载标准安装包：

http://10.15.166.49/SCFramework/deploy/blob/master/Ctrip-OpsApp-Tomcat7\_JDK1.7-7.021\_1.7.79\_p1-1.el6.x86\_64.rpm

cd 安装文件目录

rpm -ivh Ctrip-OpsApp-Tomcat7\_JDK1.7-7.021\_1.7.79\_p1-1.el6.x86\_64.rpm

配置环境变量：

vi /etc/profile

export JAVA\_HOME= /opt/app/jdk

export JRE\_HOME=$JAVA\_HOME/jre

export CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar:$JRE\_HOME/lib/rt.jar:$JRE\_HOME/lib

export PATH=$JAVA\_HOME/bin:$JRE\_HOME/bin:$PATH

查看 java –version

## 1.2 git安装

yum install git-core

# 2.安装持续集成

## 2.1 安装jenkins

### 2.1.1 安装稳定版本

运行如下命令：

* sudo wget -O /etc/yum.repos.d/jenkins.repo <http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo>
* sudo rpm --import [https://jenkins-ci.org/redhat/jenkins-ci.org.key](http://pkg.jenkins-ci.org/redhat/jenkins-ci.org.key)
* sudo yum install Jenkins

如果网络不通可以下载rpm安装包，地址如下：

<https://pkg.jenkins.io/redhat-stable/jenkins-2.60.2-1.1.noarch.rpm>

将安装包放在/opt/app目录下

执行如下命令：

cd /opt/app

rpm -ivh jenkins-2.60.2-1.1.noarch.rpm

### 2.1.2 启动与停止jenkins

* sudo service jenkins start/stop/restart
* sudo chkconfig jenkins on

**注意**：如果你得到了如下错误信息，确保java已经安装了：

Starting jenkins (via systemctl):  Job for jenkins.service failed. See 'systemctl status jenkins.service' and 'journalctl -xn' for details.

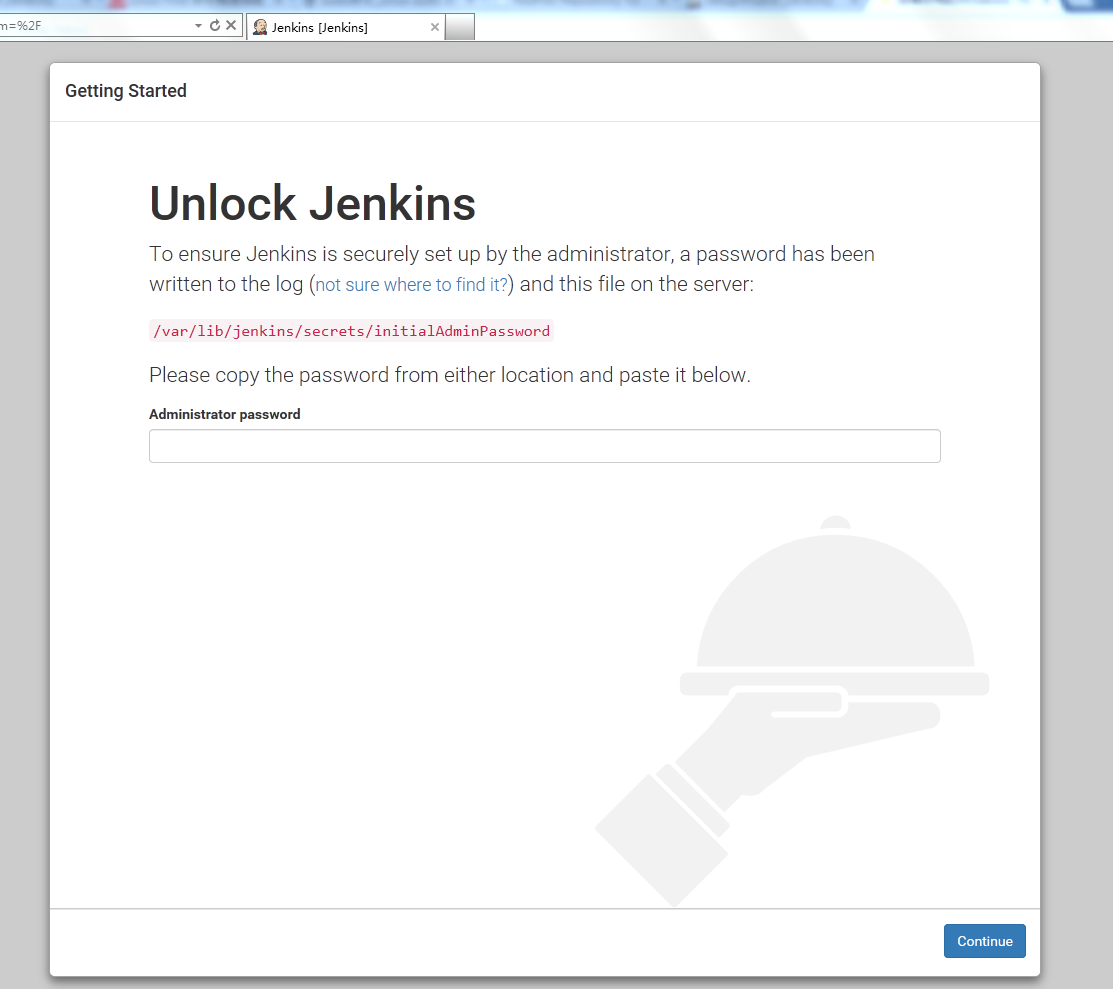
                                                           [FAILED]

### 2.1.3 安装包做什么

* Jenkins启动时以守护进程开始，更多细节查看/etc/init.d/jenkins
* .启动jenkins服务后，jenkins用户就被创建了，如果你想通过配置文件修改成另一个用户，你必须改变/var/log/jenkins，/var/lib/jenkins和/var/cache/Jenkins的所有者
* 日志文件放在/var/log/jenkins/jenkins.log。如果你遇到问题是检查这个文件。
* /etc/sysconfig/jenkins将在启动时捕获配置参数
* 默认的，jenkins监听8080端口。用你的浏览器访问这个端口开始配置。

### 2.1.4 登录jenkins

打开浏览器输入http://你的ip:8080/ 出现如下界面：

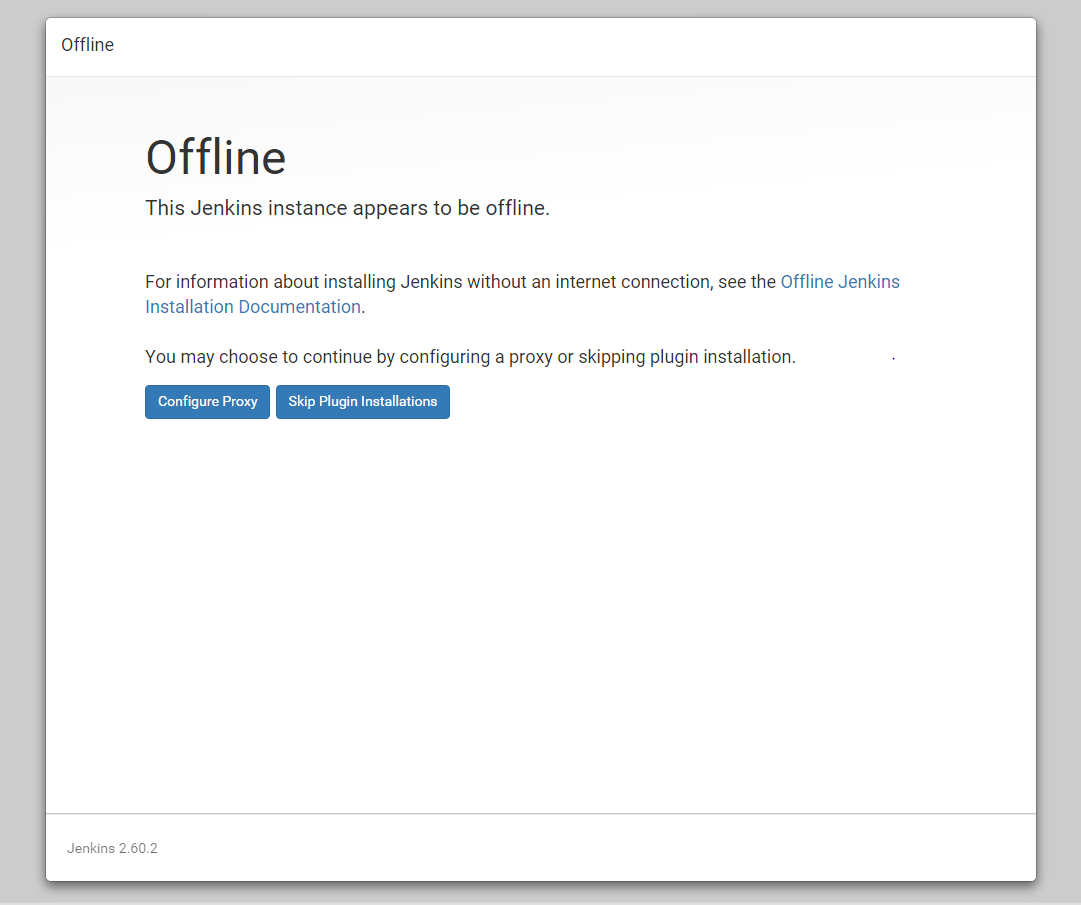


查看/var/lib/jenkins/secrets/initialAdminPassword内容：

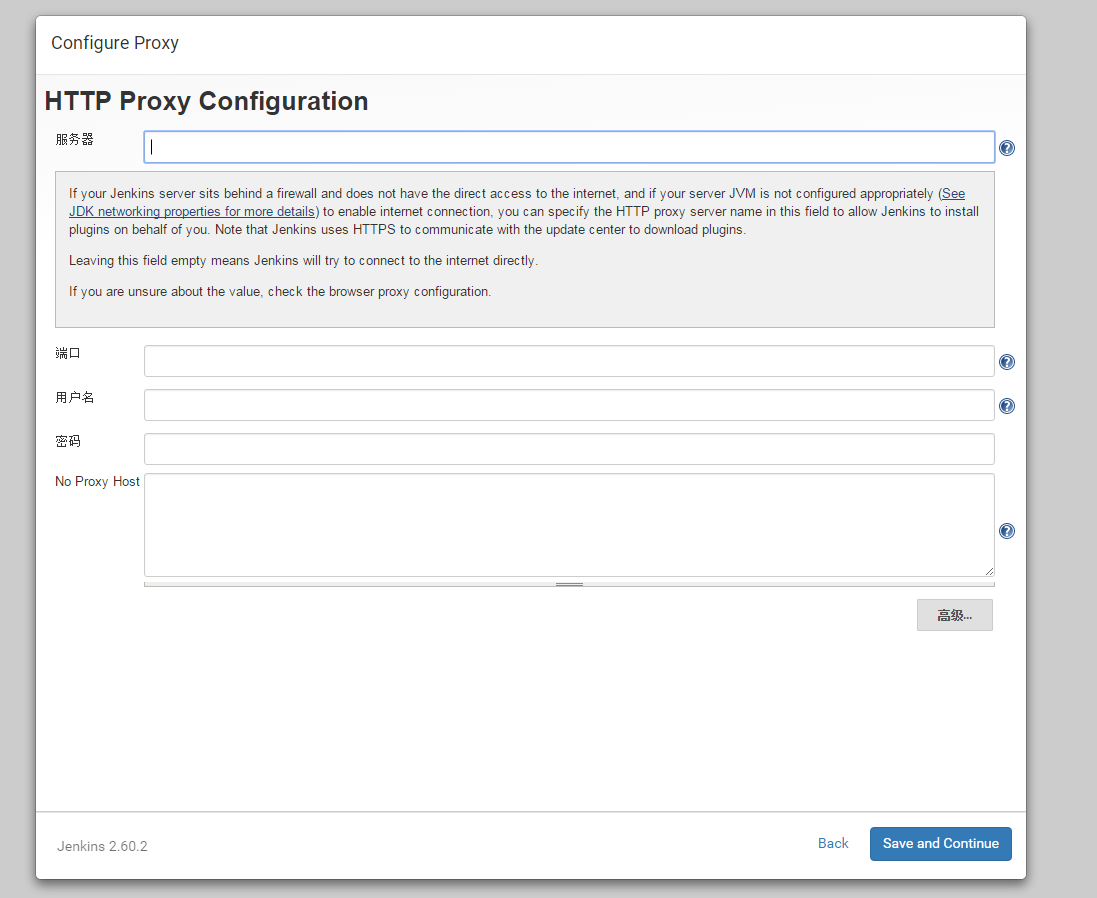
cat /var/lib/jenkins/secrets/initialAdminPassword

将文件的内容复制到Administrator password中，点击continue按钮

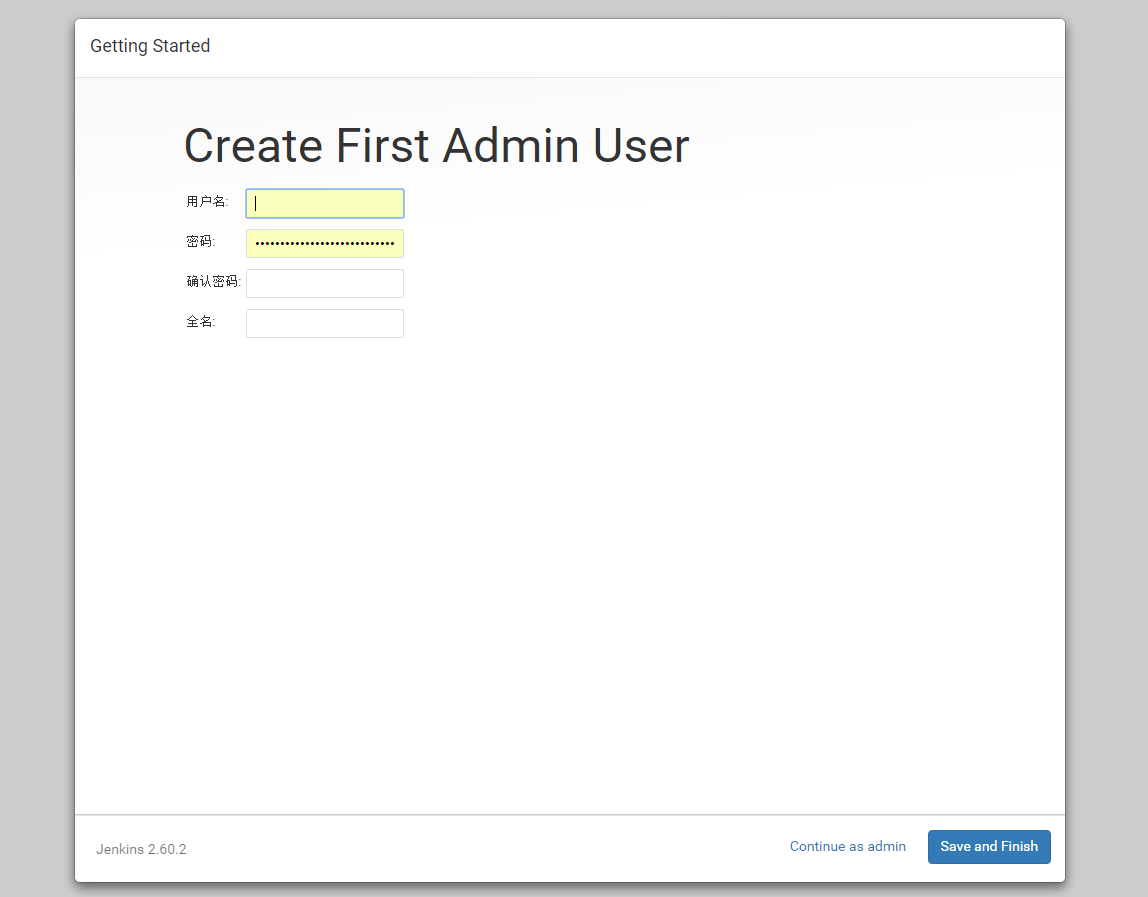
登陆后需要稍作等待，进入后出现如下界面：



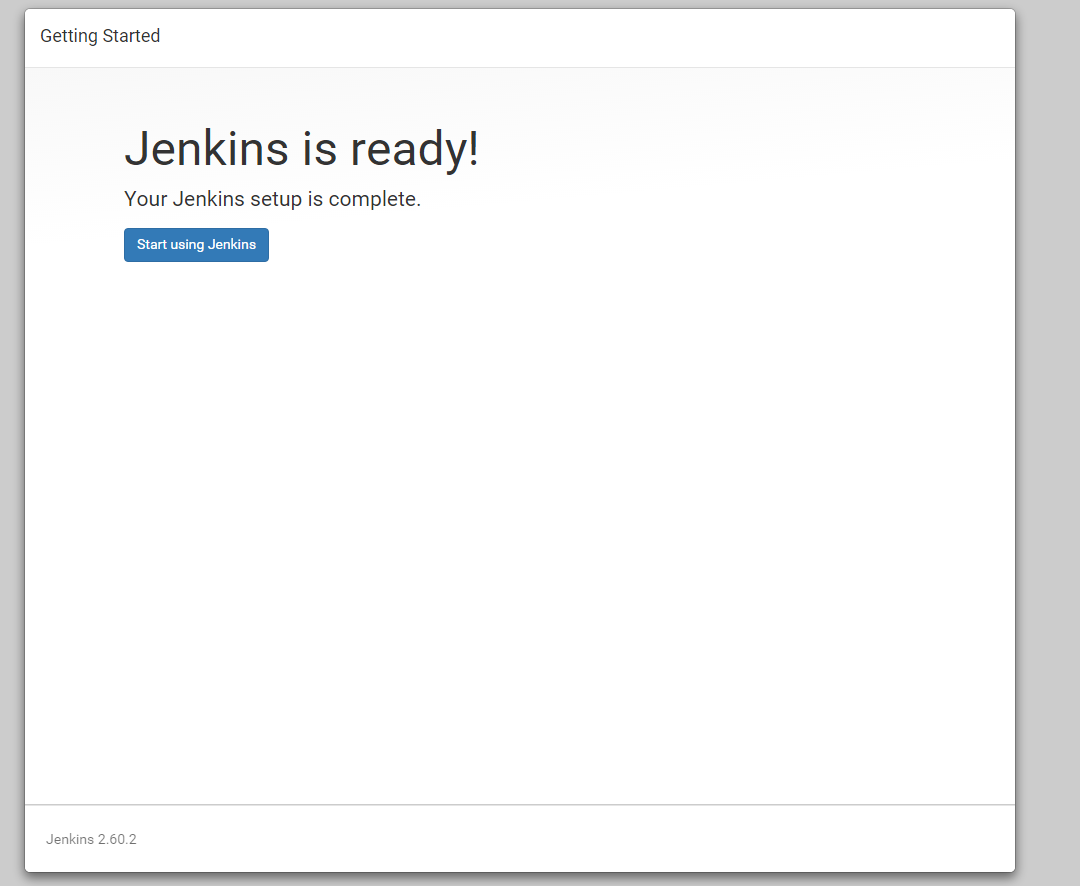
点击Configure Proxy按钮，跳转至如下代理配置界面：



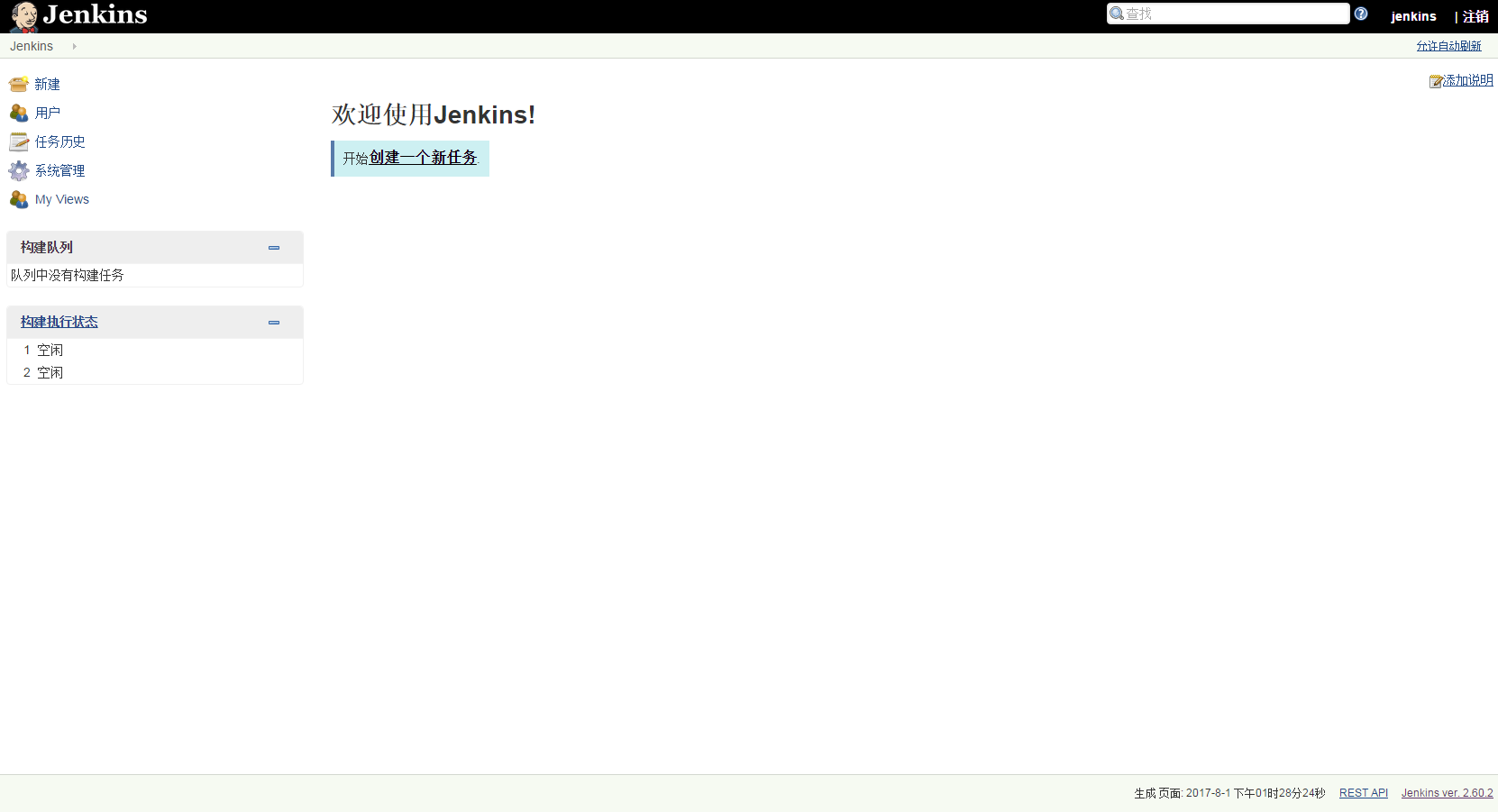
配置完成后，点击Save and Continue,之后会跳转回原来的Offline界面，再点击Skip Plugin Installations,跳转至创建用户界面：



将信息填全后点击Save and Finish,跳转至Getting Started界面：



点击Start using Jenkins



进入jenkins管理界面

### 2.1.5 安装jenkins基础插件

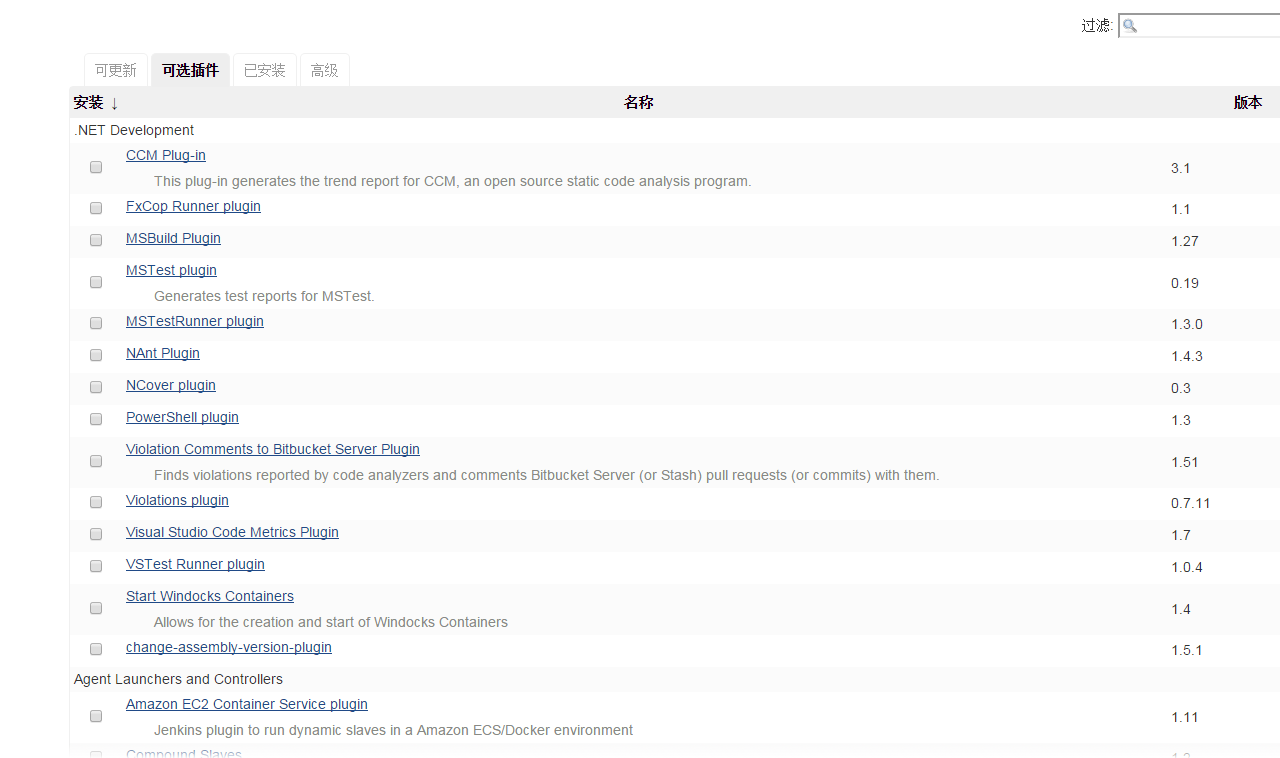
所有插件的管理通过点击管理界面上的系统管理》》管理插件



点击后，右侧会出现管理项。



点击管理插件后，选择可选插件选项卡，出现插件列表



要安装的插件可以通过过滤进行搜索。

#### 2.1.5.1 安装maven项目插件

在安装之前，点击管理界面的新建或创建一个新任务



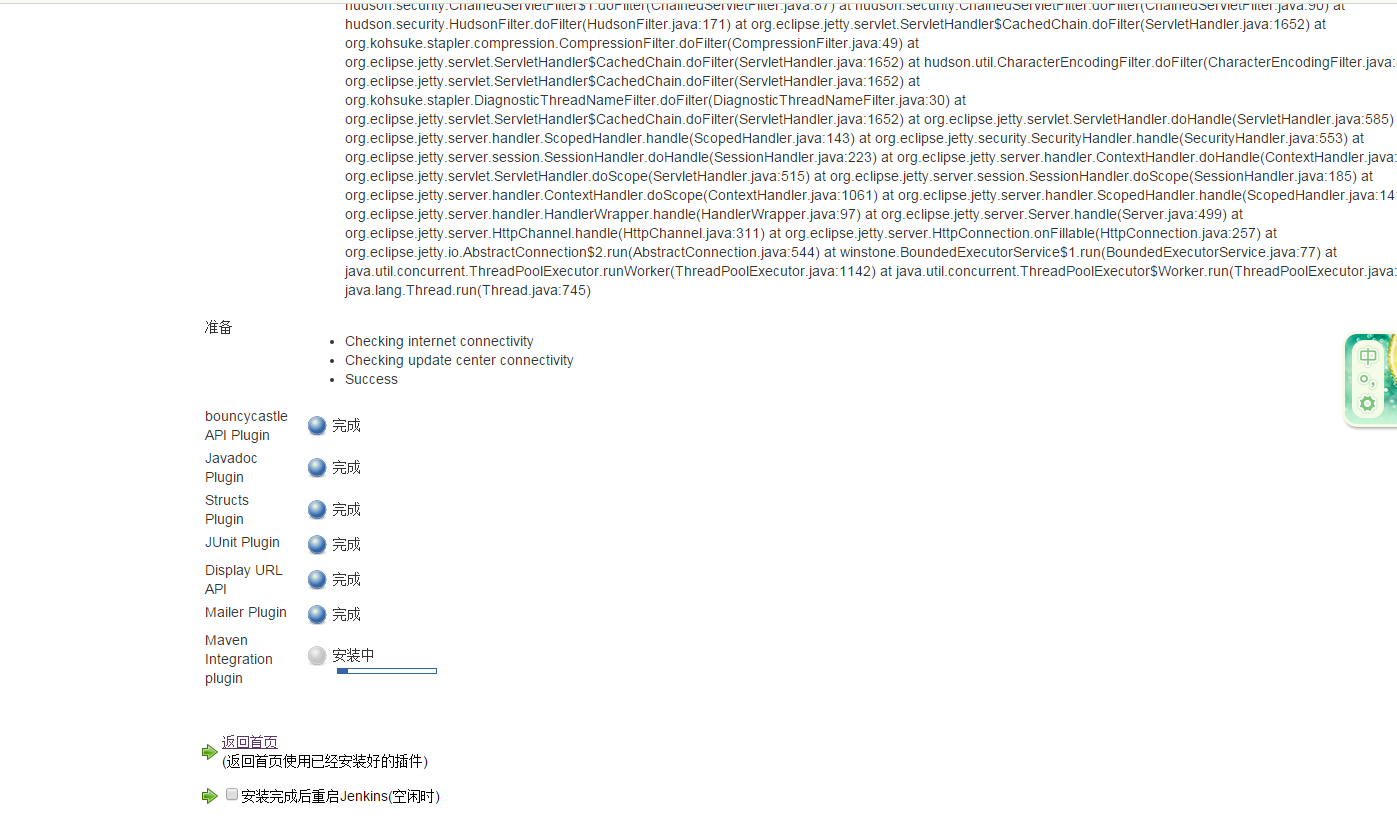
会出现如下界面：

此时只有一个构建一个自由风格的软件项目这一个选项。

我们回到maven插件管理界面

搜索[Maven Integration plugin](http://wiki.jenkins-ci.org/display/JENKINS/Maven+Project+Plugin)插件

在搜索结果中勾选后点击直接安装，安装过程如下：



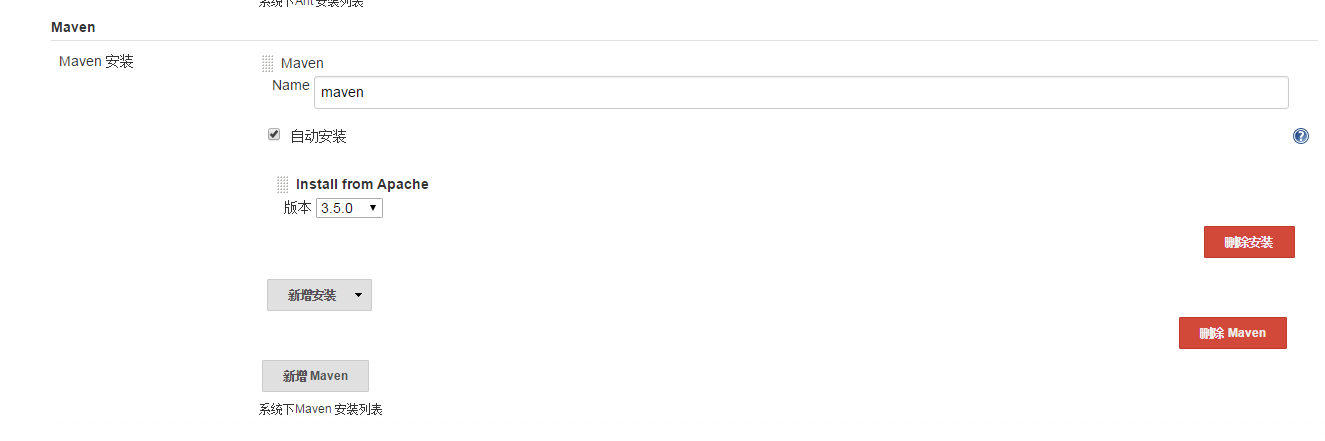
安装完成后，再新建一个任务，此时出现构建一个maven项目



接下来配置安装maven：

点击首页上系统管理》[Global Tool Configuration](http://10.32.109.70:8080/configureTools)

配置如下图



之后再第一次构建maven项目是会安装maven，安装完成后，在/var/lib/jenkins/.m2目录下增加settings.xml,文件内容如下：

<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0

https://maven.apache.org/xsd/settings-1.0.0.xsd">

<servers>

<server>

<id>scuser-snapshots</id>

<username>admin</username>

<password>sc12345678</password>

</server>

<server>

<id>scuser-releases</id>

<username>admin</username>

<password>sc12345678</password>

</server>

<server>

<id>snapshots</id>

<username>admin</username>

<password>sc12345678</password>

</server>

<server>

<id>releases</id>

<username>admin</username>

<password>sc12345678</password>

</server>

<server>

<id>nexus</id>

<username>admin</username>

<password>sc12345678</password>

</server>

<server>

<id>public</id>

<username>admin</username>

<password>sc12345678</password>

</server>

</servers>

<mirrors>

<mirror>

<id>nexus</id>

<mirrorOf>\*</mirrorOf>

<name>public</name>

<url>http://10.15.166.50:8081/nexus/content/groups/public/</url>

</mirror>

</mirrors>

<profiles>

<profile>

<id>SC</id>

<activation>

<activeByDefault>false</activeByDefault>

</activation>

<repositories>

<repository>

<id>public</id>

<name>all repoes</name>

<url>http://10.15.166.50:8081/nexus/content/groups/public</url>

<releases>

<enabled>true</enabled>

</releases>

<snapshots>

<enabled>true</enabled>

</snapshots>

</repository>

</repositories>

<pluginRepositories>

<pluginRepository>

<id>SC</id>

<url>http://10.15.166.50:8081/nexus/content/groups/public</url>

<releases>

<enabled>true</enabled>

</releases>

<snapshots>

<enabled>true</enabled>

</snapshots>

</pluginRepository>

</pluginRepositories>

</profile>

</profiles>

<activeProfiles>

<activeProfile>SC</activeProfile>

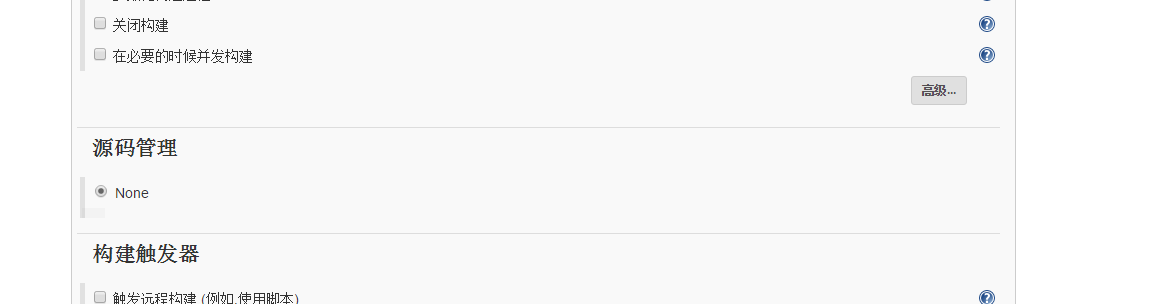
</activeProfiles>

</settings>

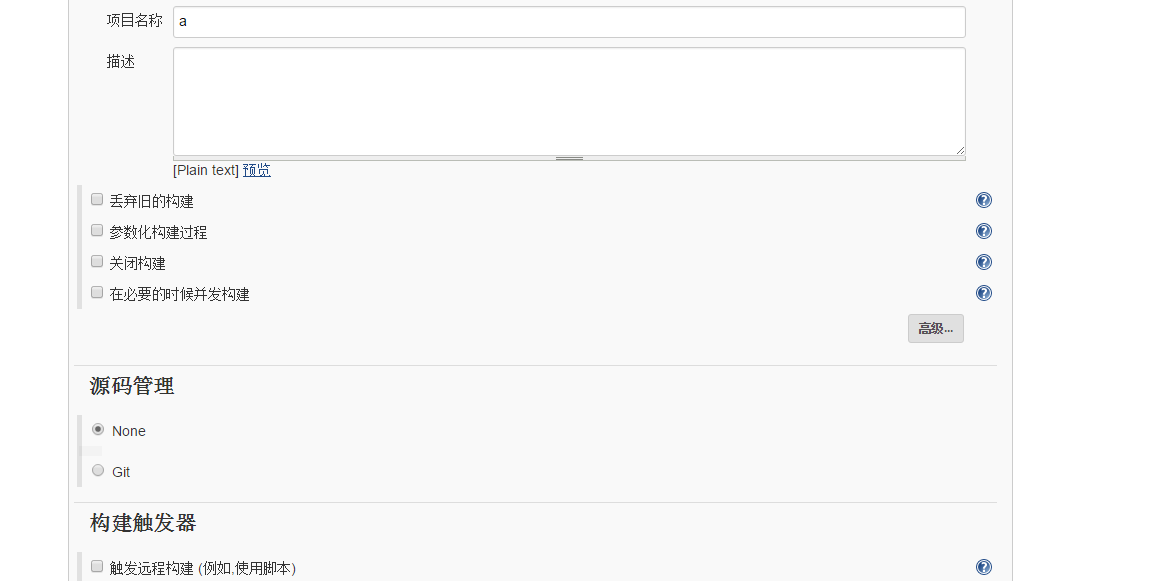
#### 2.1.5.2安装git插件

先随意创建一个项目（后面可以删除），点击配置

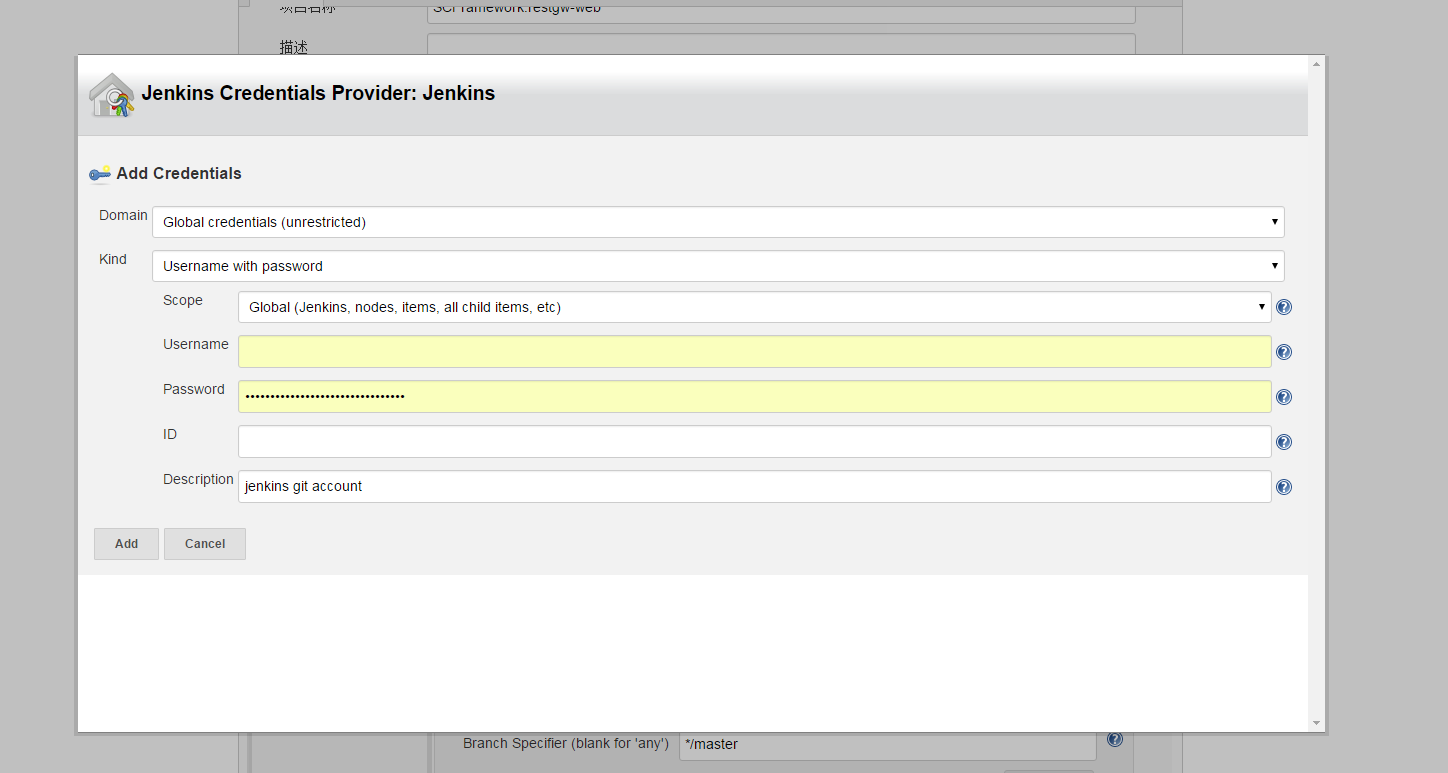




在源码管理中只有一个None的选项，此时进入插件管理界面，搜索[Git client plugin](https://plugins.jenkins.io/git-client)并点击直接安装，再搜索[Git plugin](https://plugins.jenkins.io/git)并点击直接安装，安装成功后，出现git选项

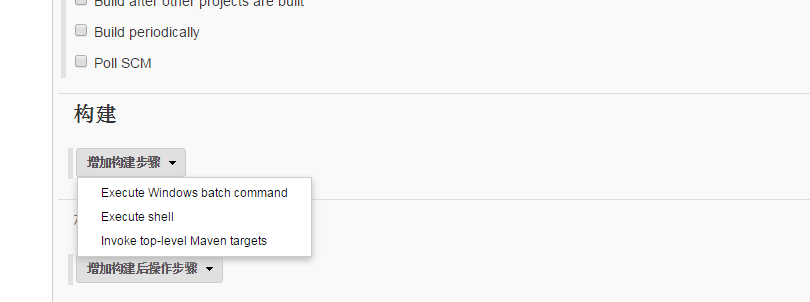


选中git选项，点击Add Credentials,配置git账户

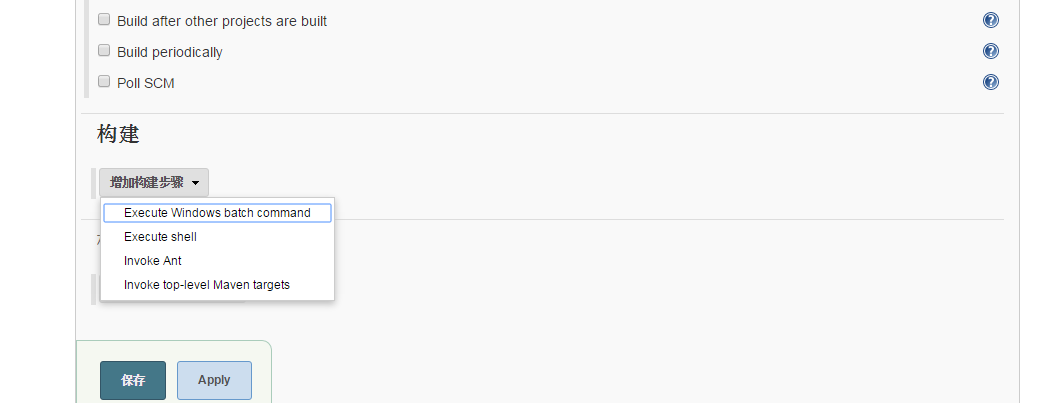


#### 2.1.5.3安装ant插件

在项目配置的构建这一步增加构建步骤没有ant构建这一项：

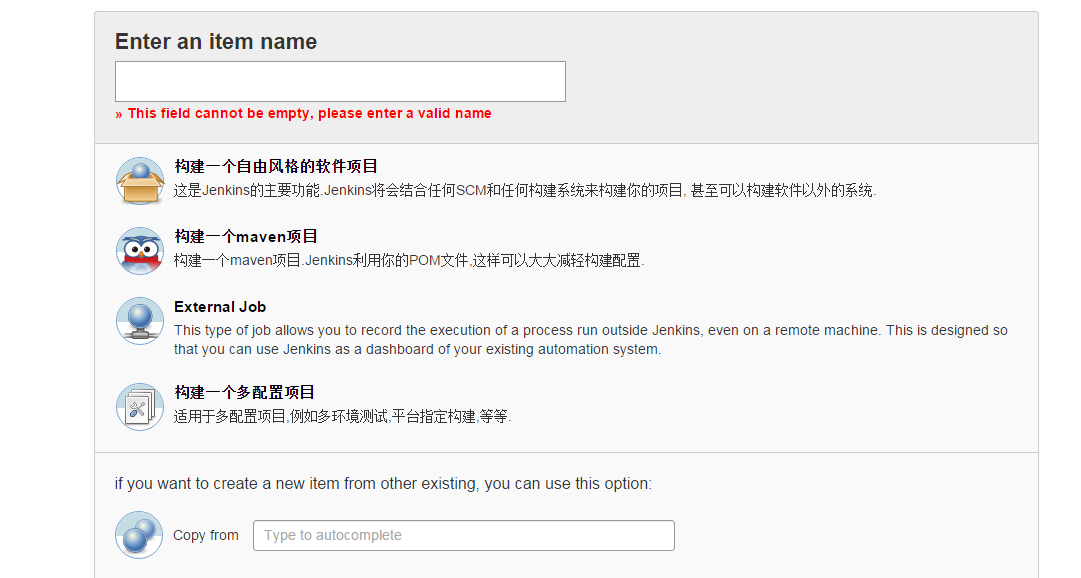


在插件管理界面中，搜索[Ant Plugin](https://plugins.jenkins.io/ant)，勾选插件并点击直接安装，完成后

多了一项Invoke Ant.

#### 2.1.5.4安装External Job插件

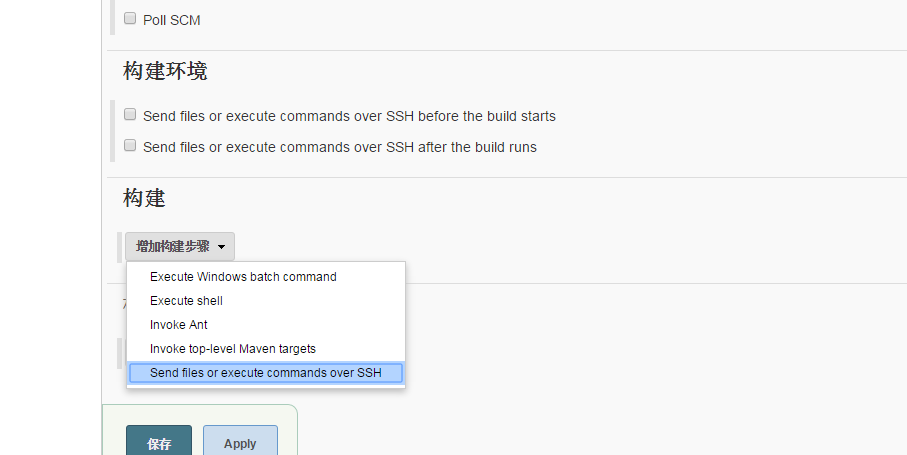
目前的新建的项目类型只有两个，在插件管理界面中，搜索[External Monitor Job Type Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Monitoring+external+jobs)，并点击直接安装



安装完成后，多出了两个Job类型，External Job和构建一个多配制项目，并且可以从已有项目中直接复制配置，见上图最有一项，水平线的下方。

#### 2.1.5.5安装ssh发布插件

搜索[Publish Over SSH](https://plugins.jenkins.io/publish-over-ssh)，并点击直接安装



构建步骤中多了一项Send files or execute commands over SSH.

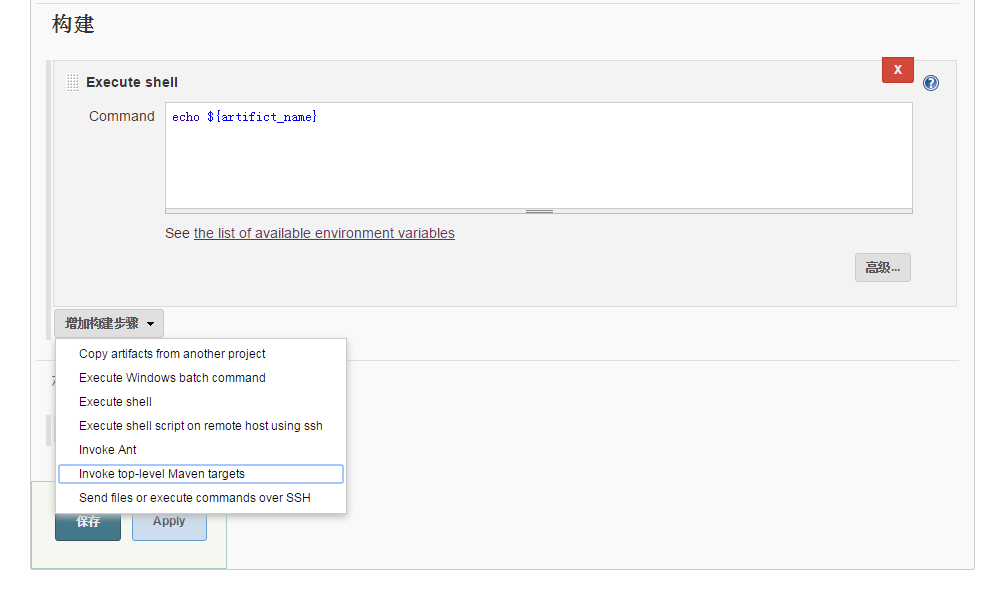
#### 2.1.5.6安装[Copy Artifact Plugin](http://wiki.jenkins-ci.org/display/JENKINS/Copy+Artifact+Plugin)



搜索[Copy Artifact Plugin](http://wiki.jenkins-ci.org/display/JENKINS/Copy+Artifact+Plugin)，并点击直接安装

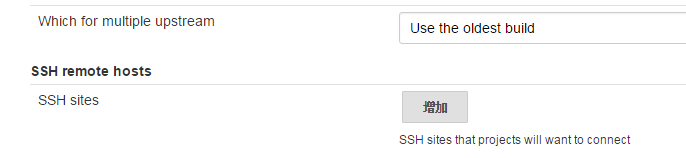
#### 2.1.5.7安装[SSH plugin](https://plugins.jenkins.io/ssh)插件

搜索[SSH plugin](https://plugins.jenkins.io/ssh)插件，并点击直接安装

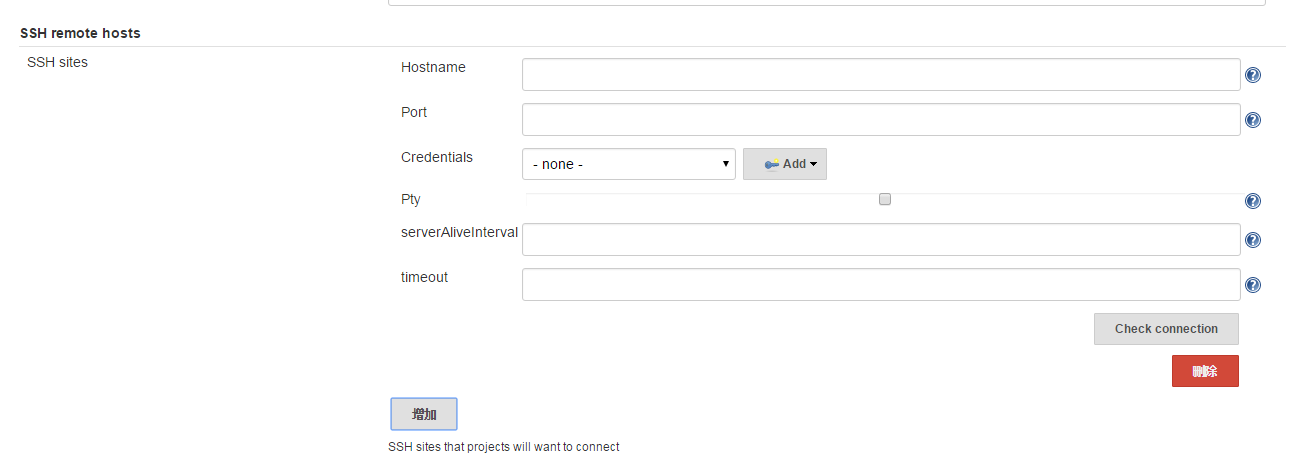


构建步骤中出现Execute shell script on remote host using ssh

到系统管理中配置远程服务器



点击增加按钮



填写完服务器信息后点击保存

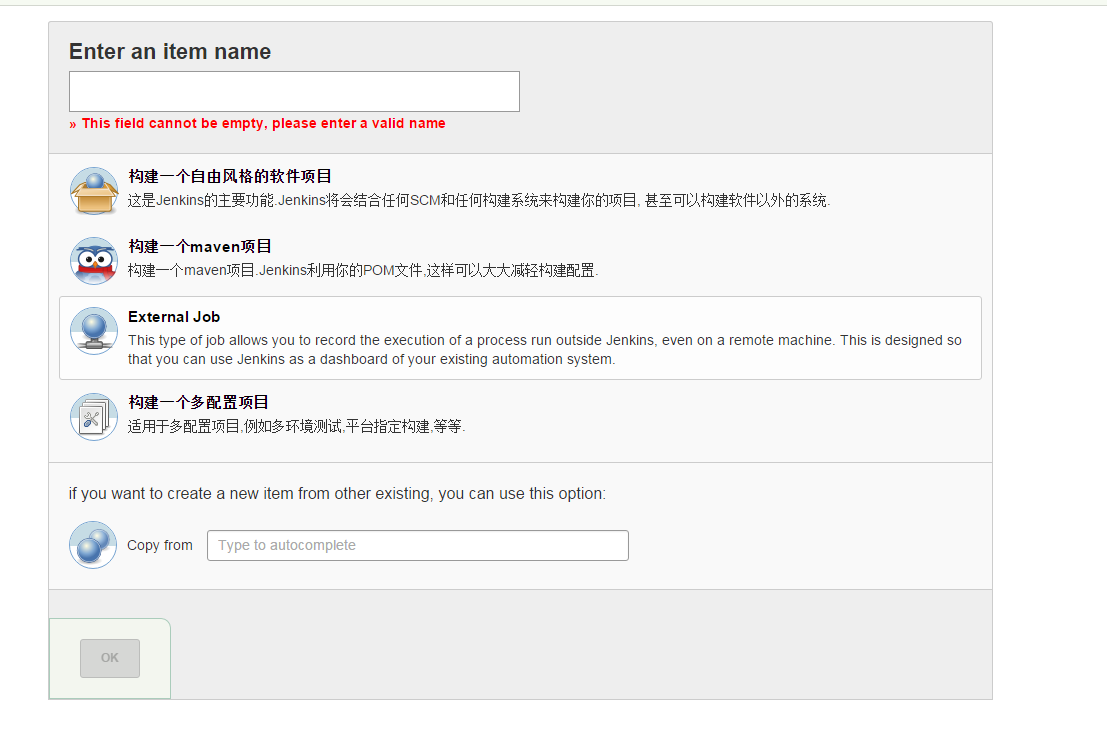
# 3. 配置项目

## 3.1.创建Job

### 3.1.1 顶级父项目

这个项目是所有项目的依赖父项目，需要先构建。

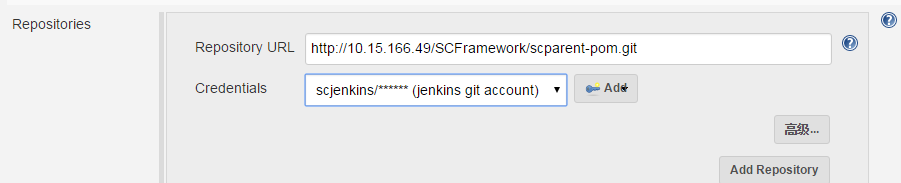
点击新建，选择创建一个maven项目



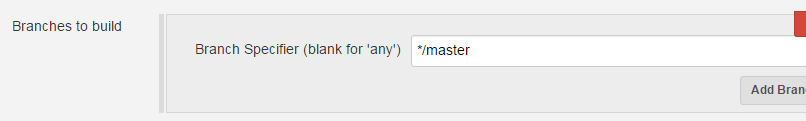
输入名字：scparent-pom（顶级项目）

配置git地址：<http://10.15.166.49/SCFramework/scparent-pom.git>

配置git用户：选择scjenkins



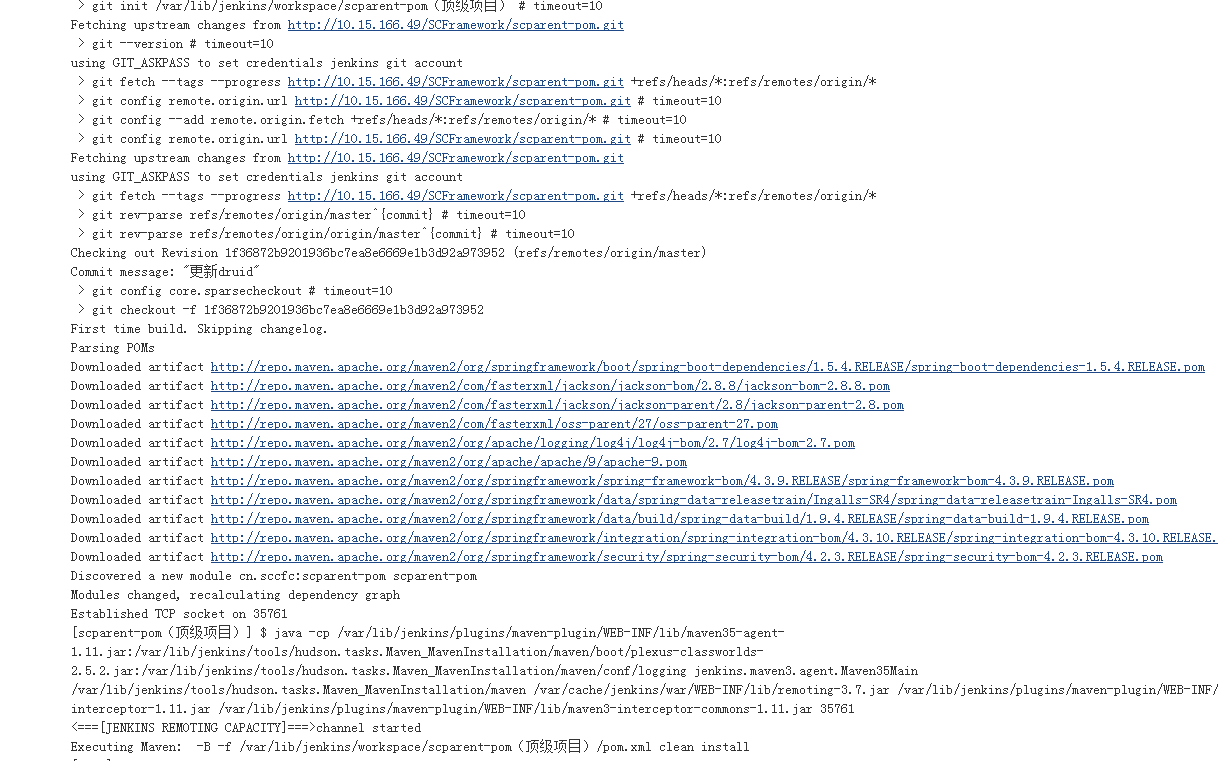
配置获取的分支：默认master，不用改



配置maven的构建目标和选项



点击保存，保存成功后点击立即构建。点击构建进度条可以看到如下日志：



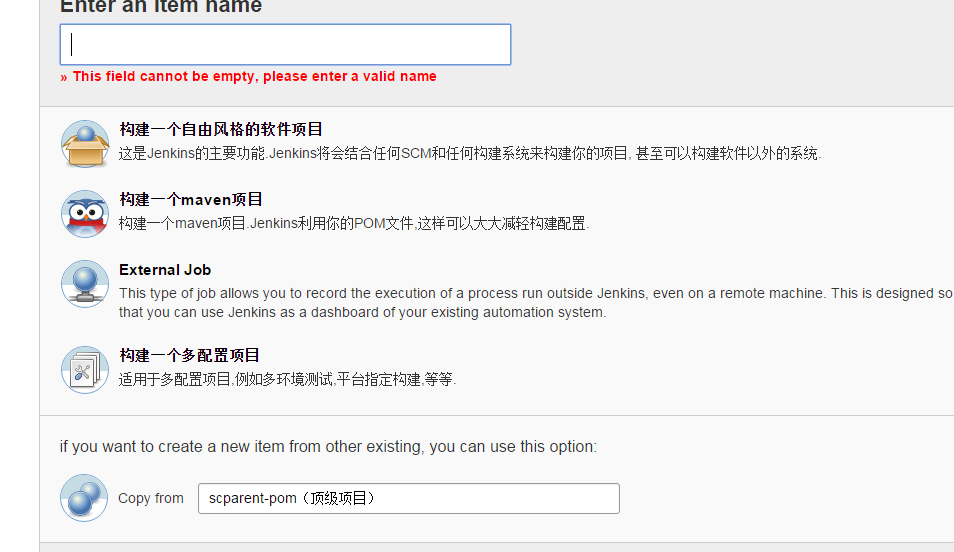
构建成功后，对应的包就存入了本地maven库。

### 3.1.2 plugins项目配置

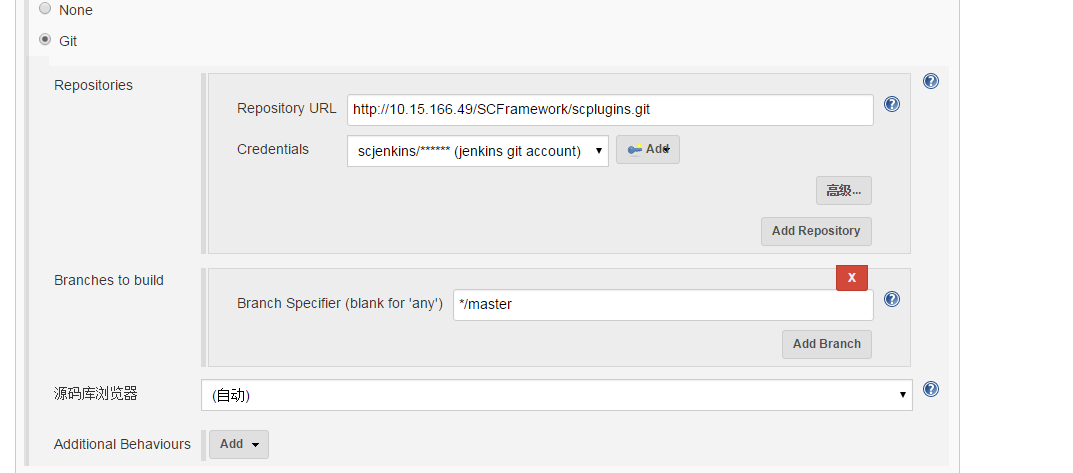
这个项目是所有项目的依赖的基础项目，仅次于顶级父项目构建。

可以从刚才的父项目复制，点击新建，在Copy From 里输入顶级父项目的名字，如下图，

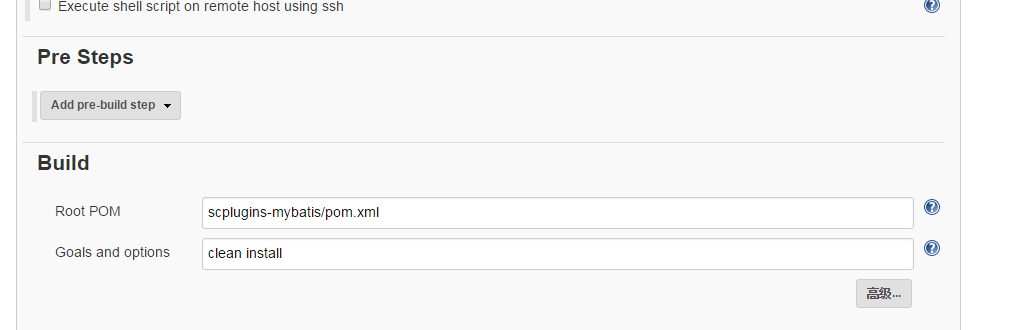
并输入该项目自己的名字



需要修改下git地址，如下



需要修改下maven运行的命令配置：

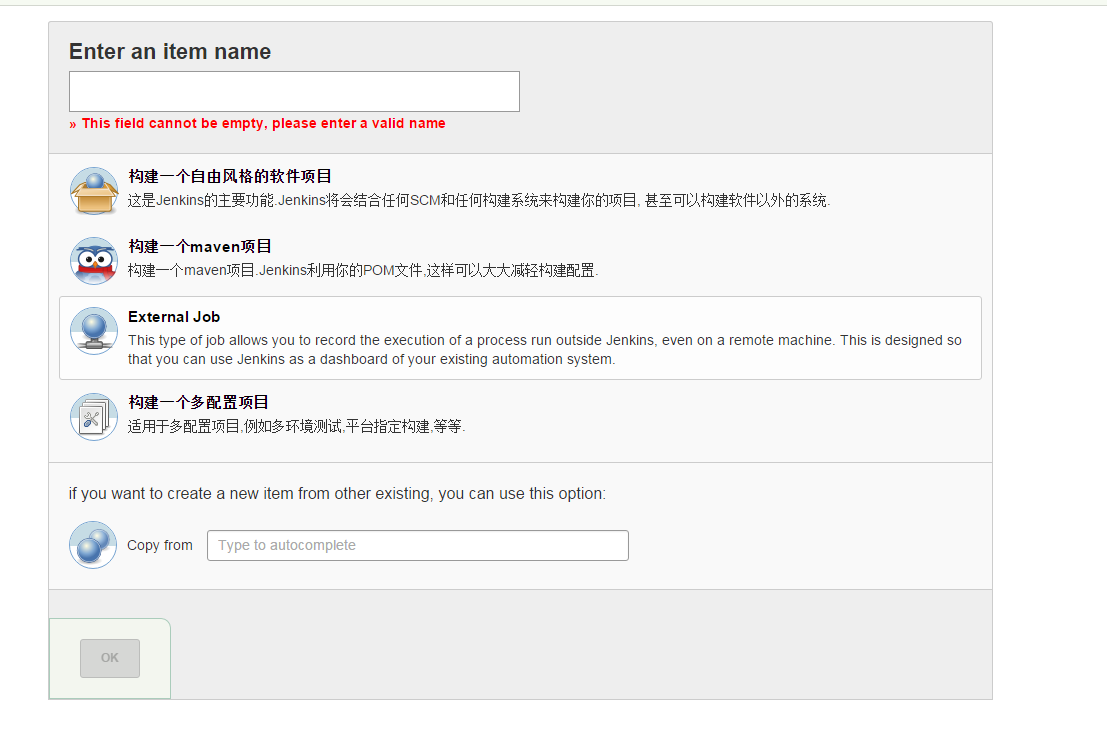


点击保存，保存成功后点击立即构建。构建成功后，对应的包就存入了本地maven库。

同样的也配置cat插件的job

### 3.1.3 usercertification(实名服务)

点击新建，选择创建一个maven项目

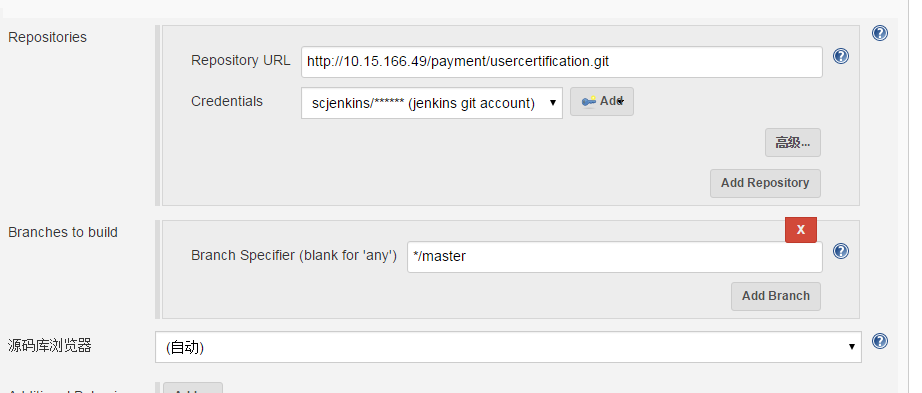


输入名字：usercertification (实名服务)

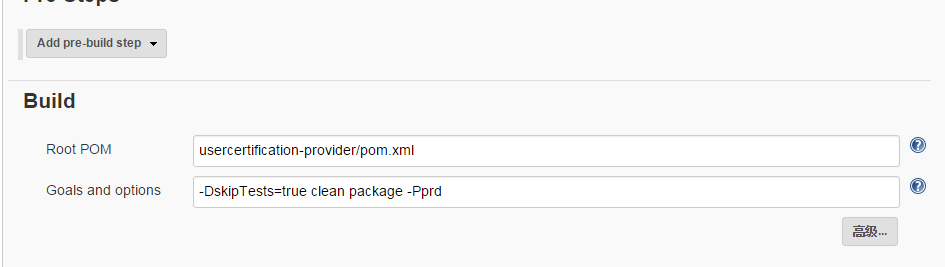
配置git地址：http://10.15.166.49/payment/usercertification.git

配置git用户：选择scjenkins

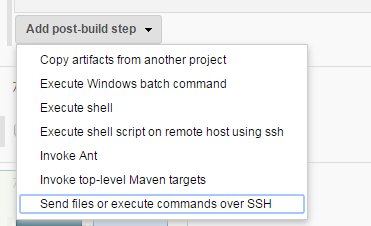
配置获取的分支：默认master，不用改



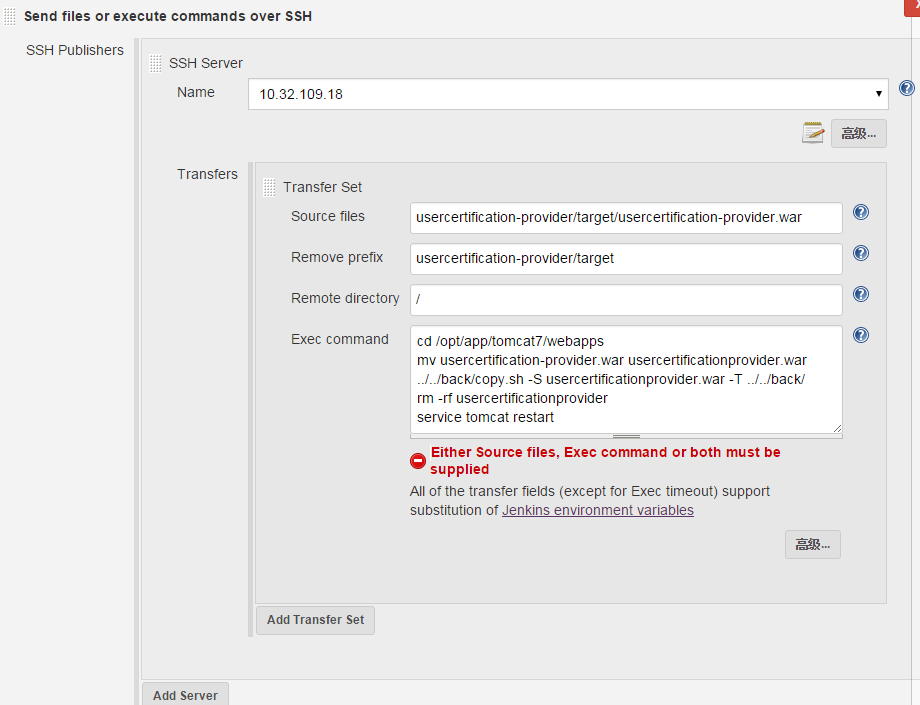
配置maven的构建目标和选项



配置发布包发布，点击Add post-build step,选择Send files or execute commands over SSH：



配置SSH server



Source files为需要发布的包路径，从工作空间根目录开始的相对路径

Remove prefix 为需要去除的前缀（为Source files文件名称的前面部分）

Remote directory 为发布到服务器后的文件所在目录，对应插件配置的文件目录作为根目录。

Exec command 为当文件发布完成后需要执行的命令

执行的命令中包含备份的脚本copy.sh,需要放置在/opt/app/back目录下

脚本如下：

#!/bin/sh

while [ -n "$1" ]

do

case "$1" in

-S) shift

source="$1"

shift

;;

-T) shift

target="$1"

shift

;;

esac

done

function filename\_withdate()

{

TIME=`date +"%Y-%m-%d"`

file\_name=$1

prename=${file\_name%.\*}

extension=${file\_name##\*.}

file\_name=${prename}${TIME}.${extension}

}

function path\_filename()

{

path\_filename=$1

file\_name=${path\_filename##\*/}

}

path\_filename "$source"

echo "copying file ${file\_name}"

filename\_withdate $file\_name

if [ -e "${target}${file\_name}" ]

then

while [ -e "${target}${file\_name}" ]

do

number\_withbracket=`echo $file\_name|grep -o -Eh '[(][0-9]+[)]'`

if [ -n "$number\_withbracket" ]

then

number=`echo $number\_withbracket|grep -o -Eh '[0-9]+'`

number\_add=`expr $number + 1`

file\_name=`echo $file\_name|sed "s/$number\_withbracket/($number\_add)/"`

else

file\_name="${file\_name%.\*}(1).${file\_name##\*.}"

fi

done

fi

cp $source "${target}${file\_name}"

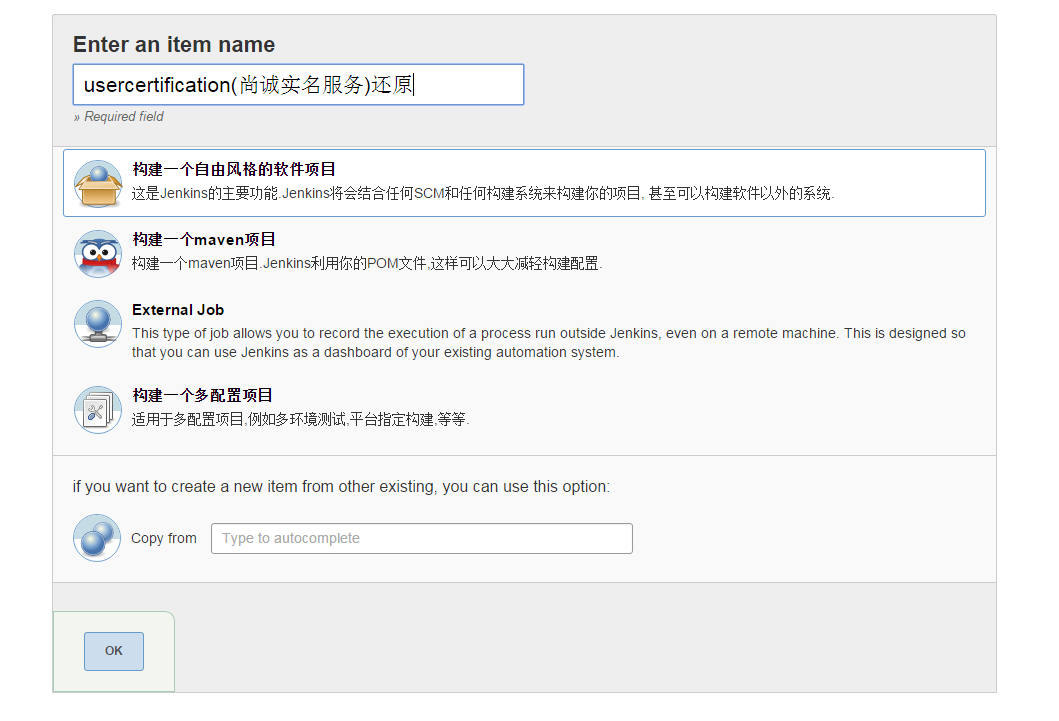
echo "copyed to ${target}${file\_name}"

点击保存，保存成功后点击立即构建。构建成功后可在/opt/app/back目录中看到备份的文件：

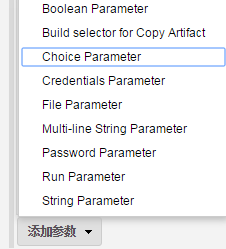


### 3.1.4 usercertification(实名服务)还原job

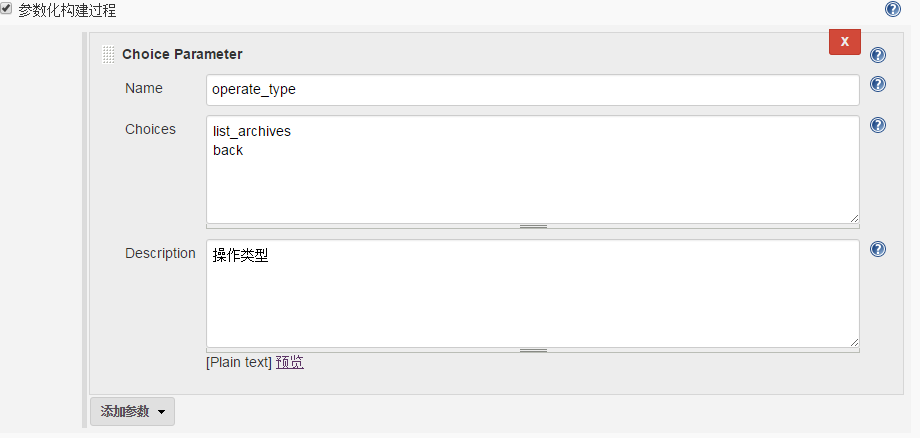
点击新建，选择构建一个自由风格的软件项目



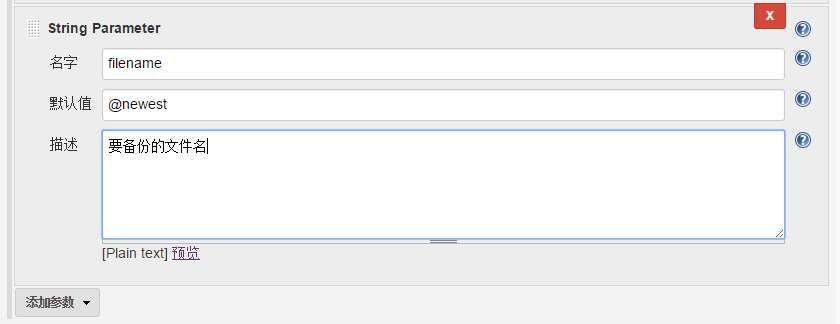
勾选参数化构建过程，点击添加参数，选择Choice Parameter



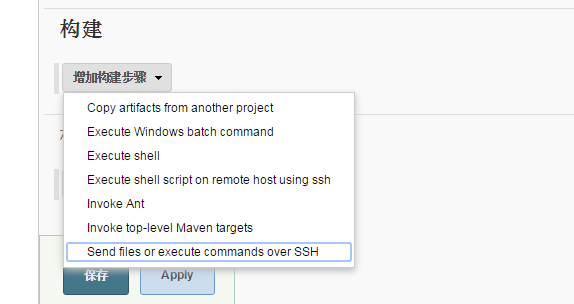
可以按下图进行配置



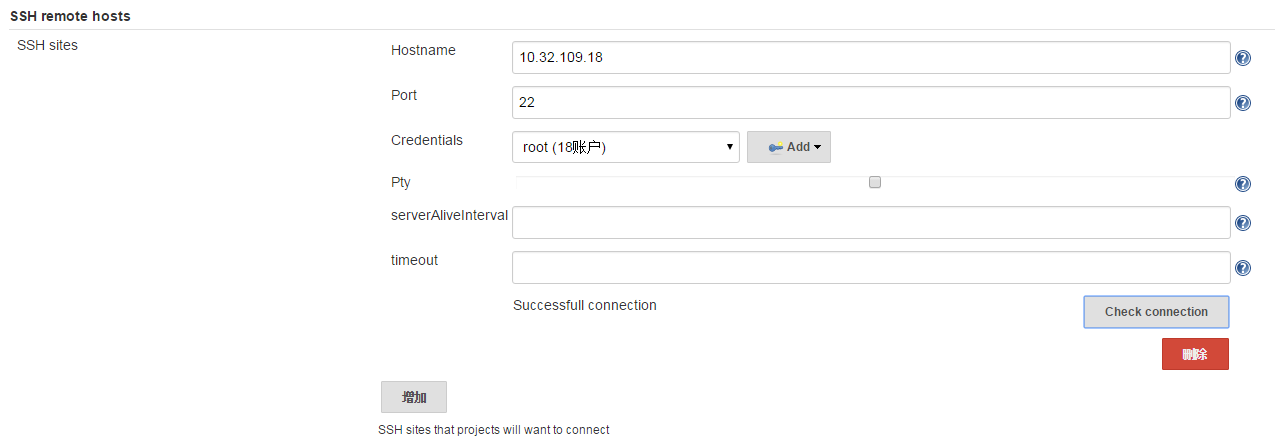
再点击添加参数，选择String Parameter



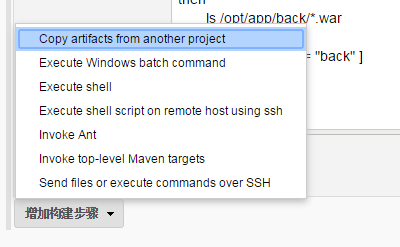
添加构建步骤选择Send files or execute commands over SSH



如果还未创建过对应的ssh服务器连接配置，先创建ssh服务器连接配置：



增加构建步骤：Copy artifacts from another project



添加如下命令：

if [ "$operate\_type" = "list\_archives" ]

then

ls /opt/app/back/\*.war

fi

if [ "$operate\_type" = "back" ]

then

if [ "$filename" = "@newest" ]

then

cd /opt/app/back

./back.sh -T /opt/app/tomcat7/webapps

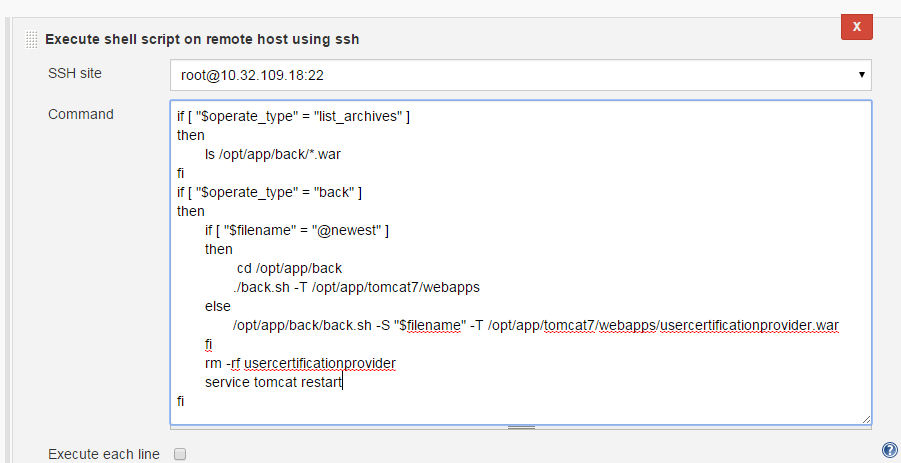
else

/opt/app/back/back.sh -S "$filename" -T /opt/app/tomcat7/webapps/usercertificationprovider.war

fi

rm -rf usercertificationprovider

service tomcat restart

fi

其中back.sh的脚本如下：

#!/bin/sh

default\_source\_path="/opt/app/tomcat7/webapps"

while [ -n "$1" ]

do

case "$1" in

-S) shift

source="$1"

shift

;;

-T) shift

target="$1"

shift

;;

esac

done

if [ ! -n "$source" ]

then

source=`ls /opt/app/back/\*.war|head -n 1`

source="$source"

fi

if [ ! -n "$target" ]

then

target="/opt/app/tomcat7/webapps/example.war"

fi

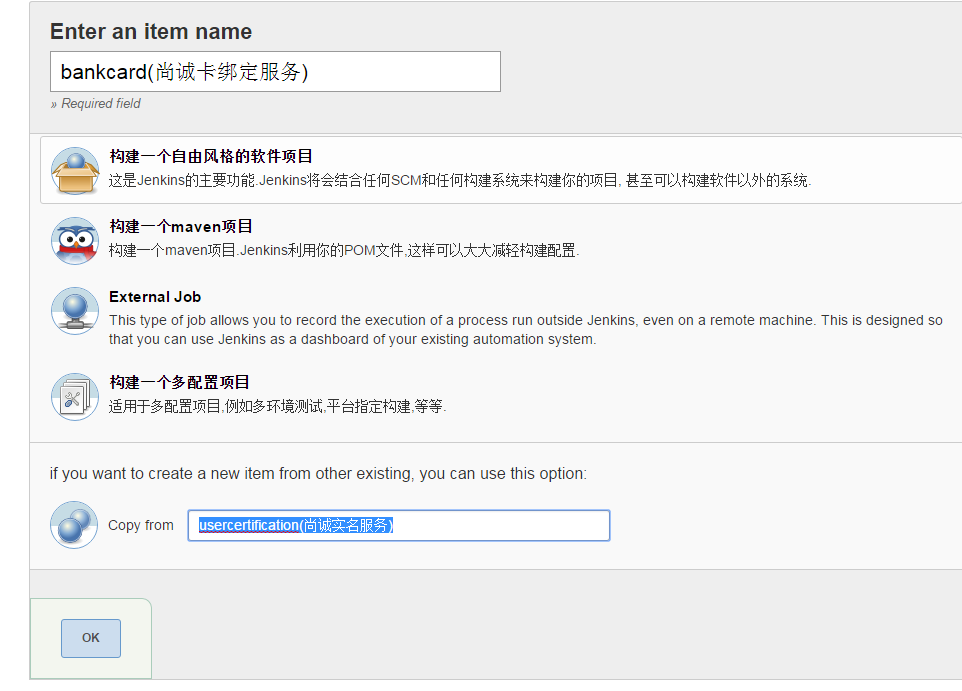
cp $source $target

echo "copyed to ${target}"

其中example.war可以改为自己想要的包的名称

### 3.1.5 bankcard(卡绑定服务)

从usercertification(实名服务)复制job配置：

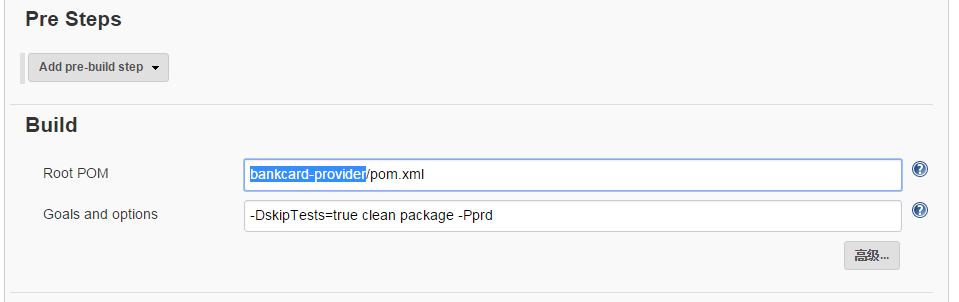


点击ok.

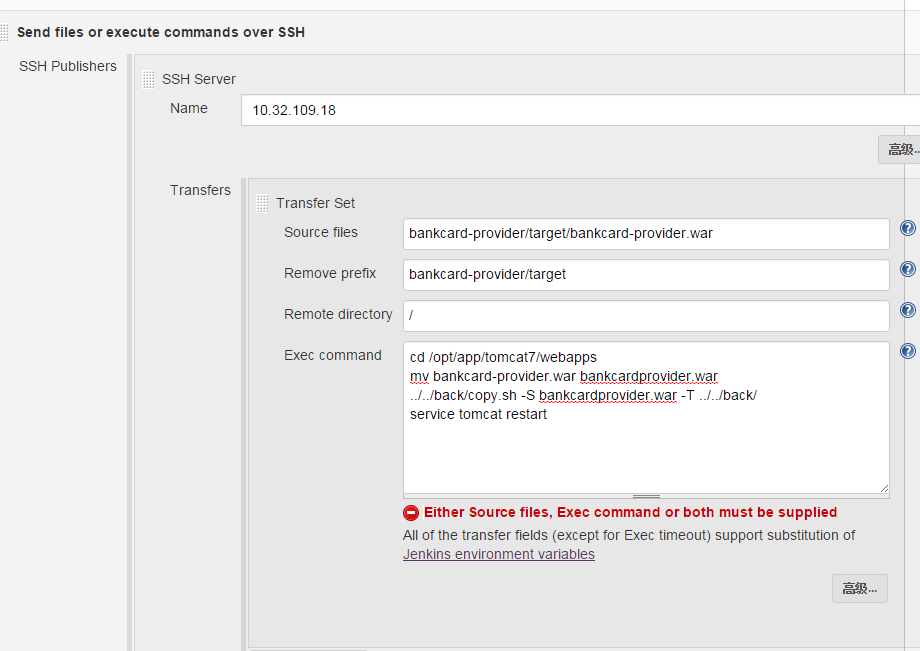
修改git配置：



修改maven 命令：



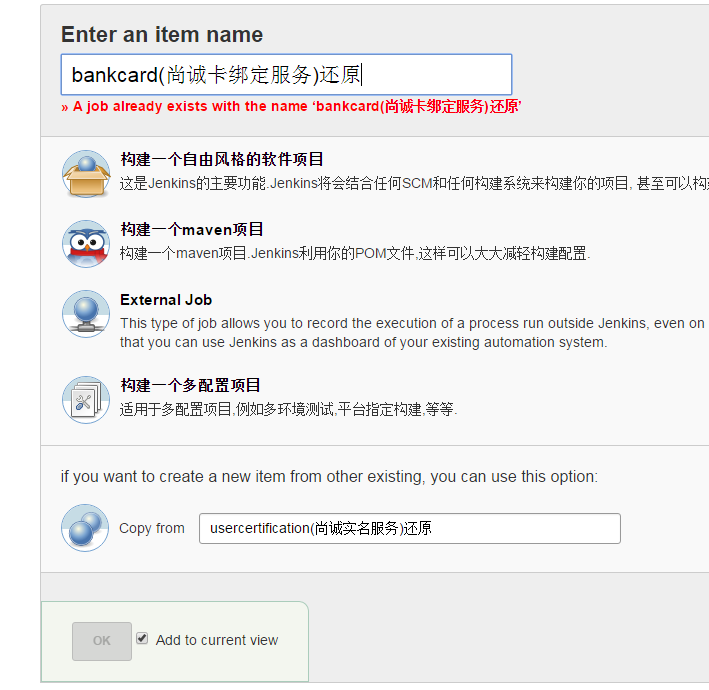
修改发布配置：



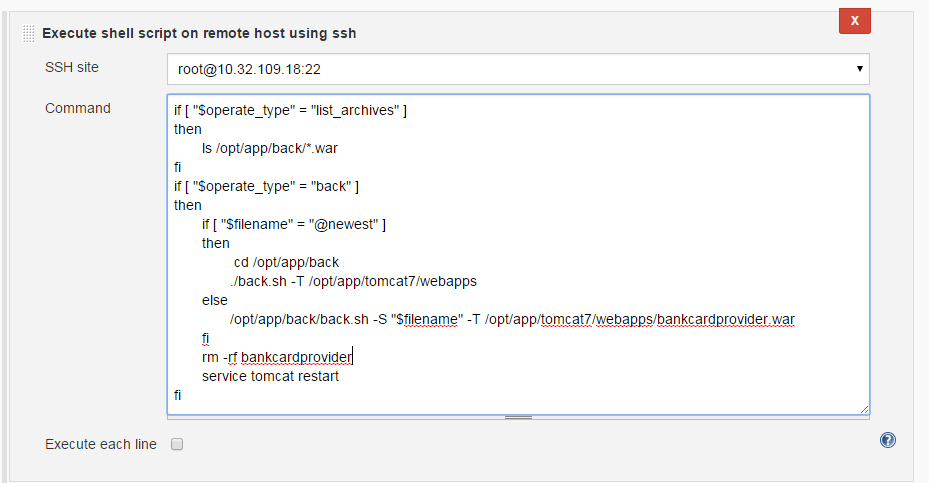
### 3.1.4 bankcard(卡绑定服务)还原job

复制usercertification(实名服务)还原job的配置：

注意项：目前还原的脚本考虑的是一个服务器部署一个应用，所以选择还原到最后一次构建发布的包时（也就是构建时，参数filename填写了@newest），会选择日期最新的一个包，多个应用时并不会分辨是哪一个应用。

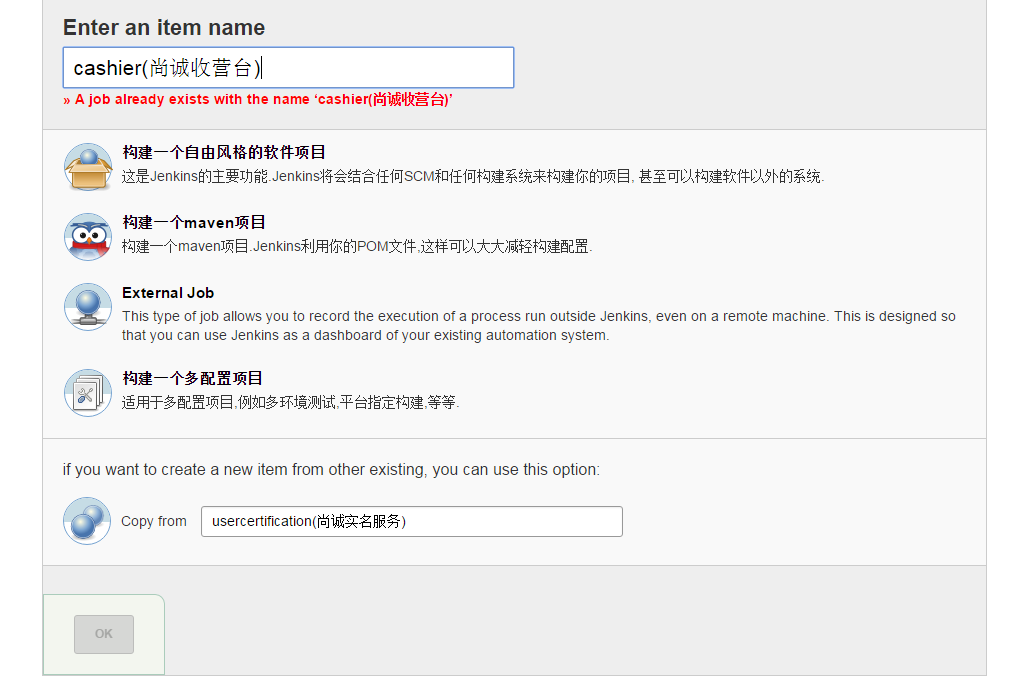


修改执行的脚本，修改脚本中的war文件名为bankcardprovider.war：



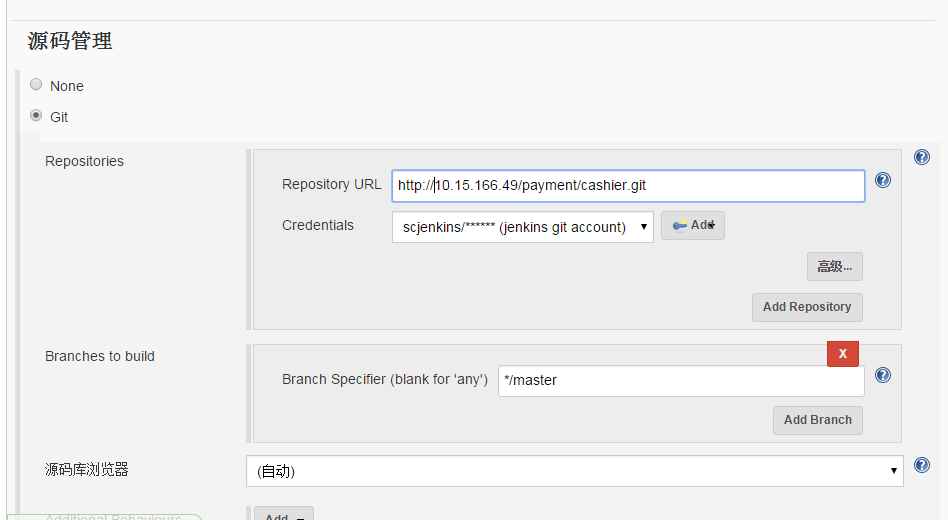
### 3.1.5 cashier(收银台)

从usercertification(实名服务)复制job配置：

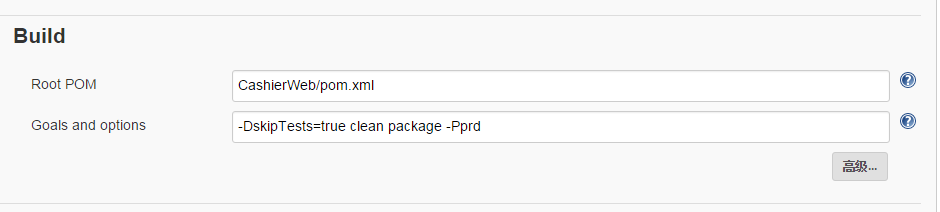


点击ok.

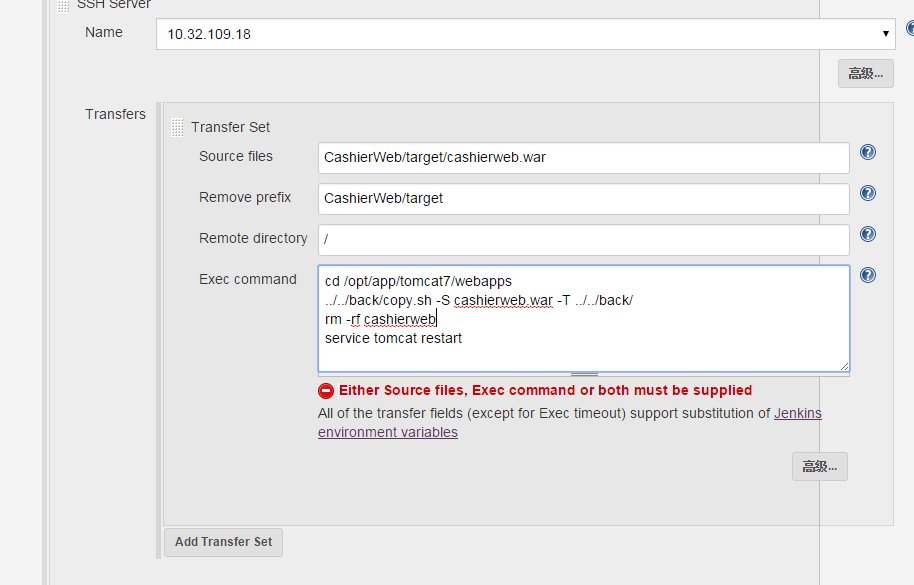
修改git配置：



修改maven 命令：



修改发布配置：



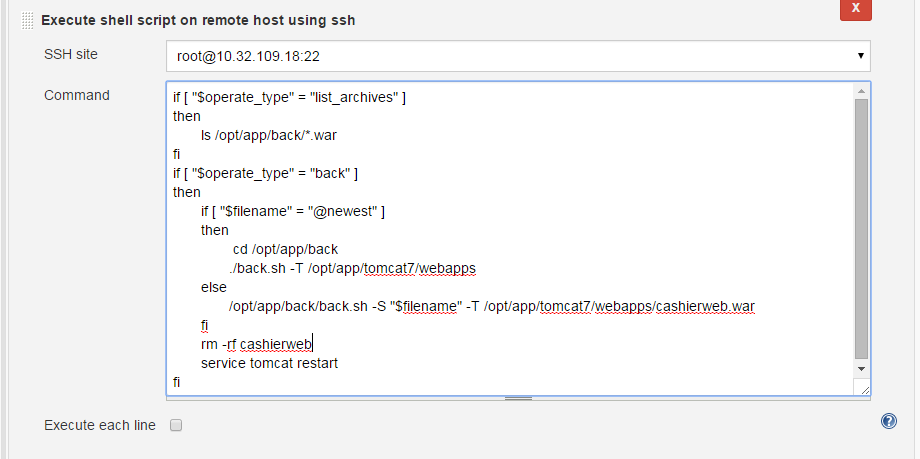
### 3.1.6 cashier(收银台)还原job

复制usercertification(实名服务)还原job的配置：

注意项：目前还原的脚本考虑的是一个服务器部署一个应用，所以选择还原到最后一次构建发布的包时（也就是构建时，参数filename填写了@newest），会选择日期最新的一个包，多个应用时并不会分辨是哪一个应用。

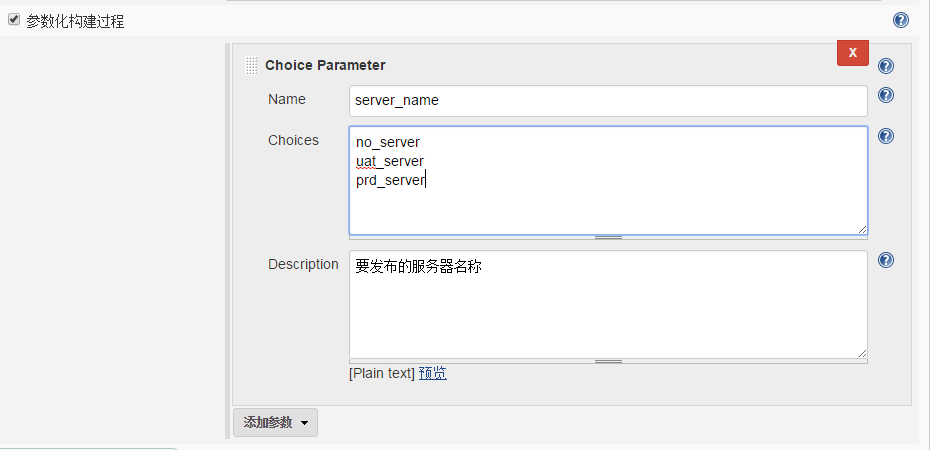


修改执行的脚本，修改脚本中的war文件名为cashierweb.war：

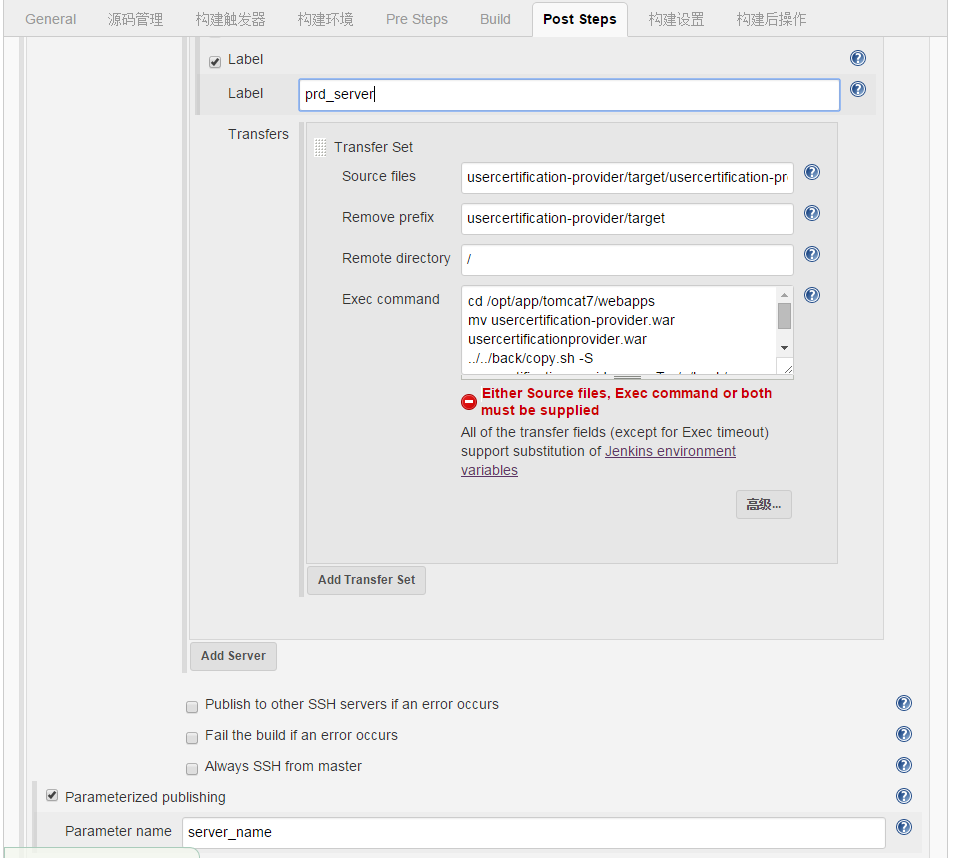


## 3.2配置多服务器发布

点击usercertification(实名服务)的配置，勾选参数化构建过程，点击添加参数，选择Choice Parameter



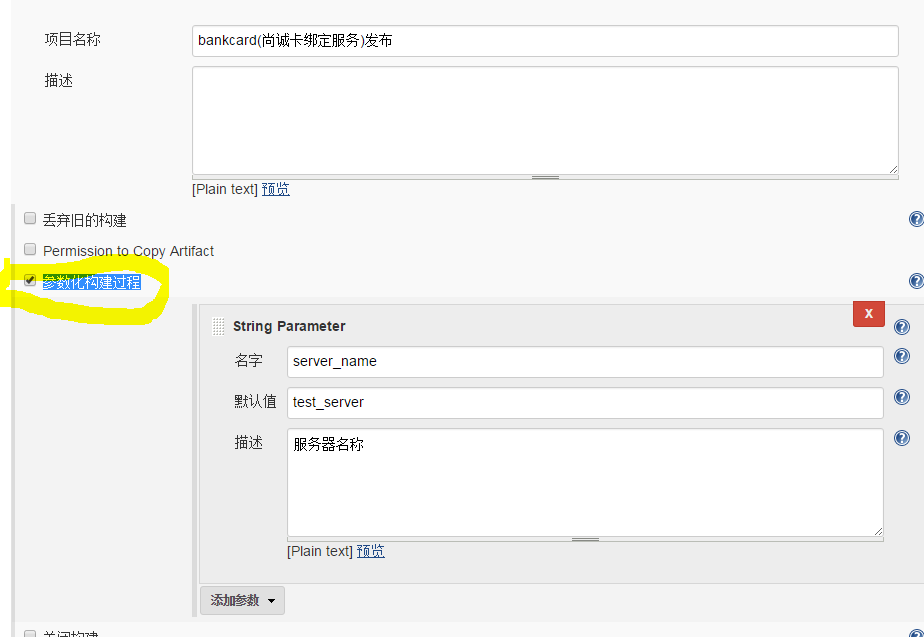
在Post Steps里的Send files or excute commands over SSH(注意不是里面SSH server里的高级)，勾选Parameterized publishing，Parameter name填写上一步配置的选择参数的名称，server\_name.点击SSH server的高级，勾选label，填入prd\_server（表示生产环境），对应值根据服务器来填写。



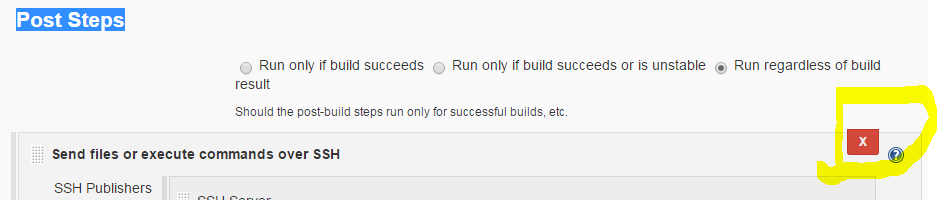
此时当构建时，选择prd\_server就会发布到这台服务器上，如果一个都对应不上，就不发布。

## 3.3 拆分打包与发布

将job bankcard(卡绑定服务)改为bankcard(卡绑定服务)打包,并取消参数化构建过程

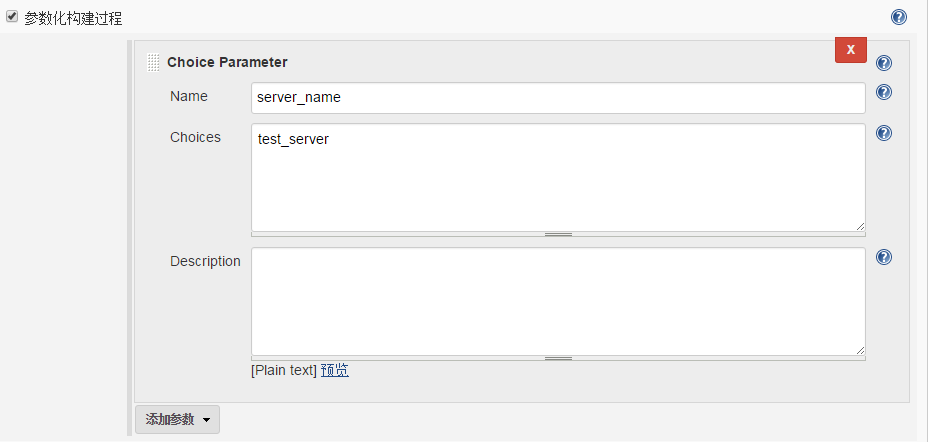


移除bankcard(卡绑定服务)打包中Post Steps

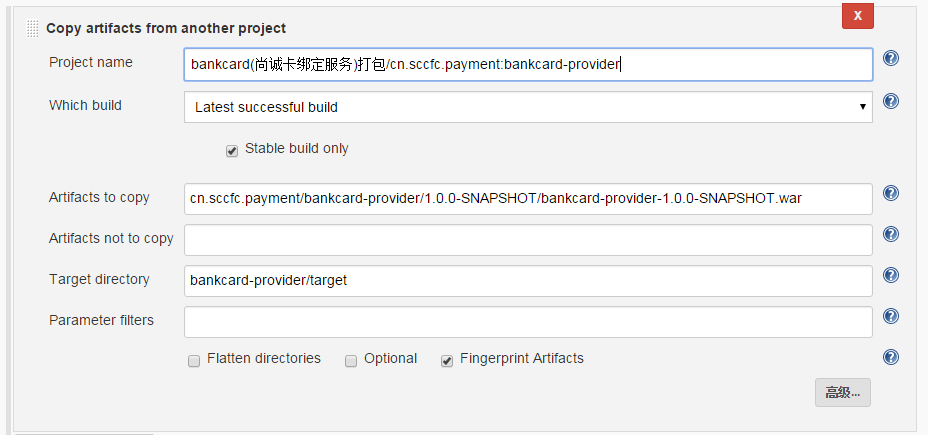


点击保存

新建一个自由风格的项目bankcard(卡绑定服务)发布，勾选参数化构建过程，选择choice parameter



点击增加构建步骤选择Copy artifacts from anther project



点击增加构建步骤Send files or execute commands over SSH

