**Jenkins参数化构建部署手册**

版本1.0

田政敏

河南双马纸品包装有限公司

[Pick the date]

文档信息

|  |  |  |  |
| --- | --- | --- | --- |
| 文档名称： | Jenkins参数化构建部署手册 | 文档版本号： | 1.0 |
| 文档作者： | Magic Star Trace | 生成日期： | 2018年2月7日 |
| 文档审核者： |  | 审核日期： |  |

文件修订记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **修订日期** | **版本号** | **修订说明** | **修订人** | **审核人** | **批准人** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

目录

[第一章 前言 4](#_Toc505776943)

[1.1 版权声明 4](#_Toc505776944)

[1.2 文档目的 4](#_Toc505776945)

[1.3 适用范围 4](#_Toc505776946)

[1.4 Jenkins 简介 4](#_Toc505776947)

[1.5 系统和工具版本介绍 5](#_Toc505776948)

[第二章 发布及回滚流程介绍 6](#_Toc505776949)

[2.1 发布(Update)流程图 6](#_Toc505776950)

[2.2 发布流程介绍 6](#_Toc505776951)

[2.3 回滚(Rollback)流程图 7](#_Toc505776952)

[2.4 回滚流程介绍 7](#_Toc505776953)

[第三章 Jenkins安装/初始化和插件安装 8](#_Toc505776954)

[3.1. Jenkins安装 8](#_Toc505776961)

[3.2. Jenkins初始化 9](#_Toc505776962)

[3.3. Jenkins插件安装 18](#_Toc505776963)

[第四章 全局工具和系统设置 21](#_Toc505776967)

[4.1 全局工具配置 21](#_Toc505776968)

[4.2 系统设置 23](#_Toc505776969)

[第五章 创建可以被maven参数化构建的项目 26](#_Toc505776970)

[6.4 新建任务 26](#_Toc505776971)

[6.5 参数化构建配置 26](#_Toc505776972)

[6.6 源码管理 32](#_Toc505776973)

[6.7 添加授权 34](#_Toc505776974)

[6.8 查看是否能获取到远程版本号 37](#_Toc505776975)

[6.9 添加配置文件仓库(Git仓库) 39](#_Toc505776976)

[6.10 Pre Steps 41](#_Toc505776977)

[6.11 设置maven Build参数 42](#_Toc505776978)

[6.12 升级和回滚配置(Post Steps) 42](#_Toc505776979)

[6.13 构建后步骤 47](#_Toc505776980)

[第六章 发布和回滚测试 49](#_Toc505776981)

[6.1 升级 49](#_Toc505776982)

[6.2 回滚 51](#_Toc505776983)

[第七章 其他 51](#_Toc505776984)

[7.1 思路拓展 51](#_Toc505776985)

[7.2 结束语 52](#_Toc505776986)

# 前言

## 版权声明

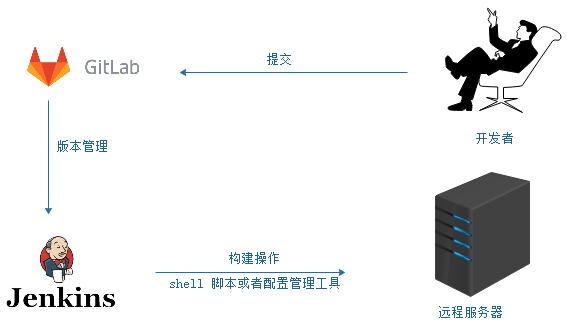
文档属”Magic Star Trace”所有，未经允许不得私自复制、转发或用于其他目的

## 文档目的

针对Jenkins参数化构建生产环境，总结Jenkins参数构建升级和回滚配置，初步制定惠尚科技有限公司Jenkins规范化使用标准

## Jenkins 简介

Jenkins是一个开源软件项目，是基于Java开发的一种持续集成工具，用于监控持续重复的工作，旨在提供一个开放易用的软件平台，使软件的持续集成变成可能



## 系统和工具版本介绍

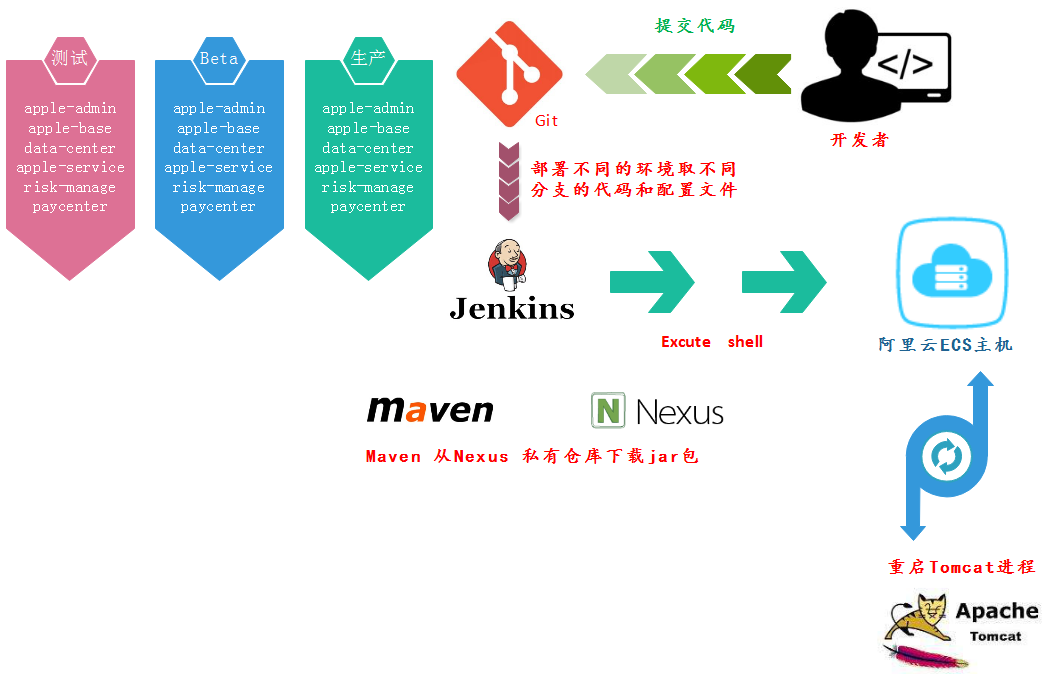
Ubuntu系统版本及Jenkins版本：

|  |  |
| --- | --- |
| OS版本 | Jenkins版本 |
| Ubuntu 14.04 Server | Jenkins Version 2.89.2 |

**目前该Jenkins主要用于构建JAVA项目**

# 发布及回滚流程介绍

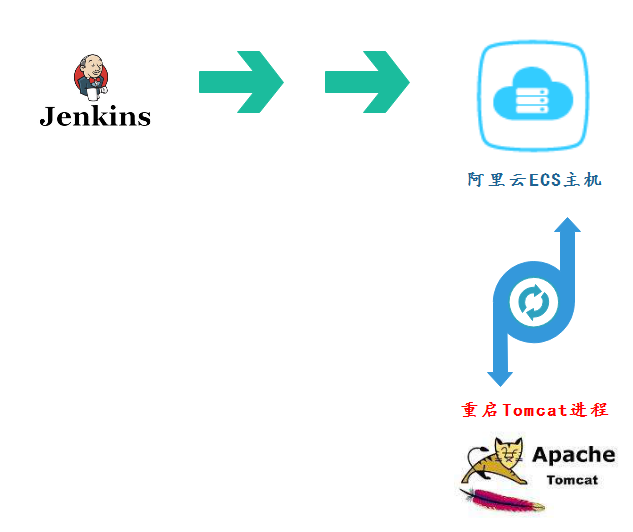
## 发布(Update)流程图



## 发布流程介绍

Jenkins接到参数化构建请求🡪检查插件是否完整性[能否支持构建时使用]🡪 Action选择Update🡪读取远程仓库上最新的版本🡪开始构建🡪下载源码🡪下载业务对应配置文件🡪替换配置文件🡪Maven Build[Maven Clean Install操作]🡪将生成出的war包传到远程服务器上🡪重启远程服务器上的Tomcat 实例🡪自动化测试介入并开始测试

## 回滚(Rollback)流程图



## 回滚流程介绍

Jenkins点击参数化构建🡪 Action选择rollback🡪选择历史备份文件[WAR包]🡪开始构建🡪检查插件是否完整性[能否支持构建时使用]🡪停止Tomcat进程🡪替换war包🡪 🡪重启远程服务器上的Tomcat 进程实例

# Jenkins安装/初始化和插件安装



## Jenkins安装

* 安装Jenkins

使用命令：wget -O /etc/yum.repos.d/jenkins.repo http://jenkins-ci.org/redhat/jenkins.repo

[root@jenkins ~]# wget -O /etc/yum.repos.d/jenkins.repo http://jenkins-ci.org/redhat/jenkins.repo

[root@jenkins ~]# yum install -y wget jenkins java

[root@jenkins ~]# service jenkins start

Starting Jenkins [确定]

[root@jenkins ~]#

[root@jenkins ~]#

[root@jenkins ~]# service jenkins status

jenkins (pid 2003) 正在运行...

root@ubuntu:~#

也可以使用Jenkins的war包配合Tomcat来部署，部署方式比较简单(略)

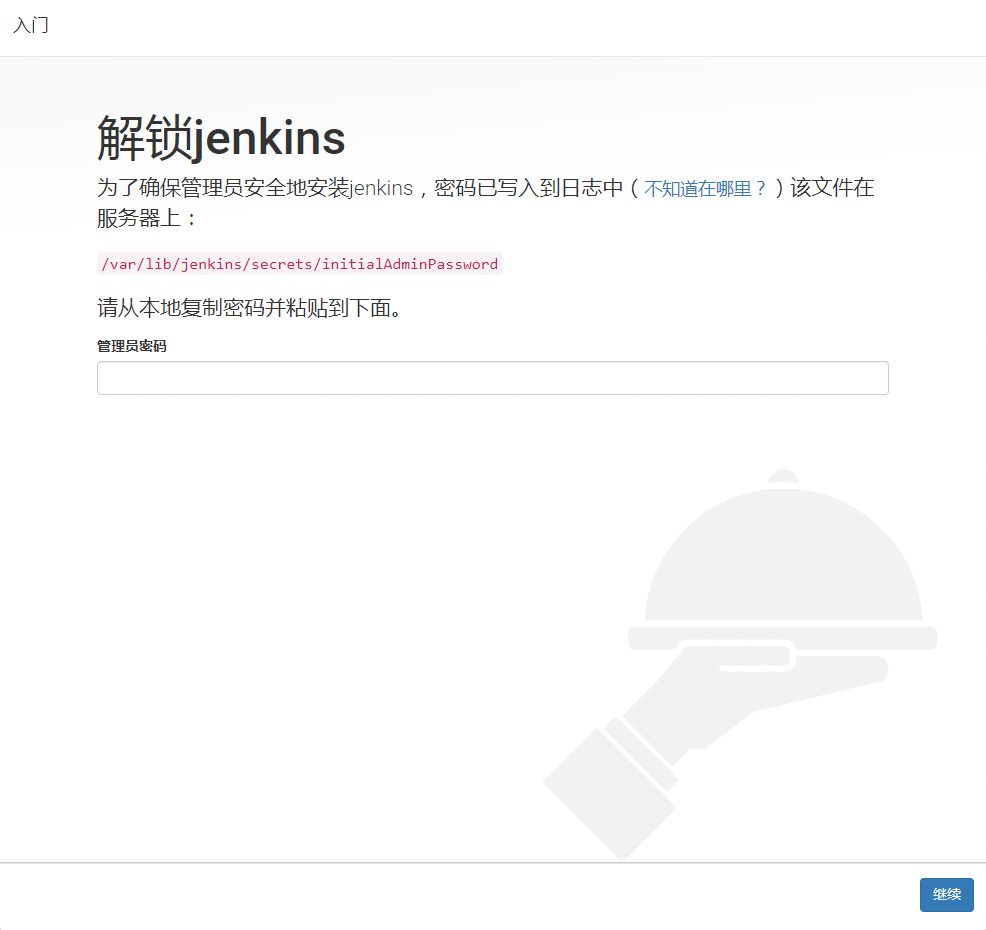
* 安装JDK

下载地址

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

wget http://download.oracle.com/otn-pub/java/jdk/8u161-b12/2f38c3b165be4555a1fa6e98c45e0808/jdk-8u161-linux-x64.tar.gz

## Jenkins初始化

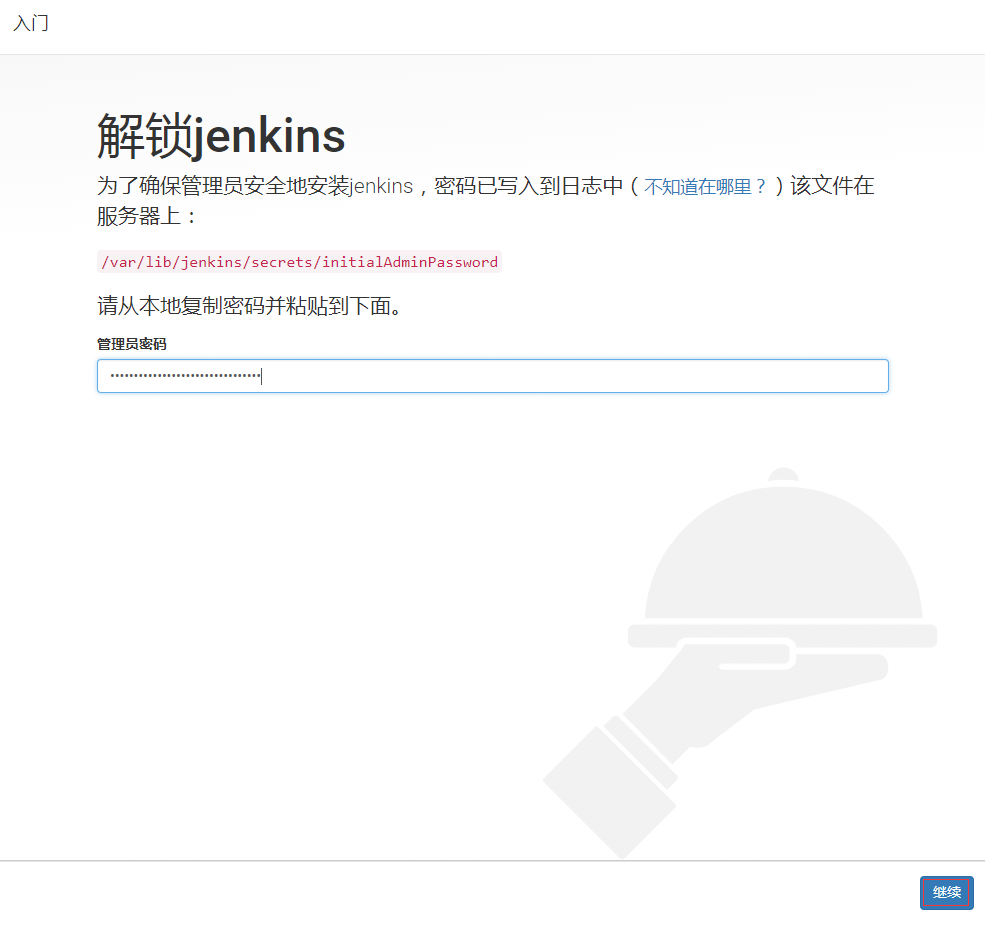


输入Jenkins的地址，如http://192.168.0.228:8080/jenkins

根据提示需要键入初始化的Jenkins的管理密码

回到Jenkins服务器节点上输命令：

[root@jenkins ~]# cat /var/lib/jenkins/secrets/initialAdminPassword



将得到的初始化密码粘贴到上管理员密码处并点击”继续”(如上图所示)



鼠标左键点按”安装”推荐的插件



Jenkins开始安装插件，等待完成



创建一个管理员账户以上所有信息均为必填项🡪点击保存并完成



Jenkins已就绪，Jenkins安装已完成点击开始使用Jenkins

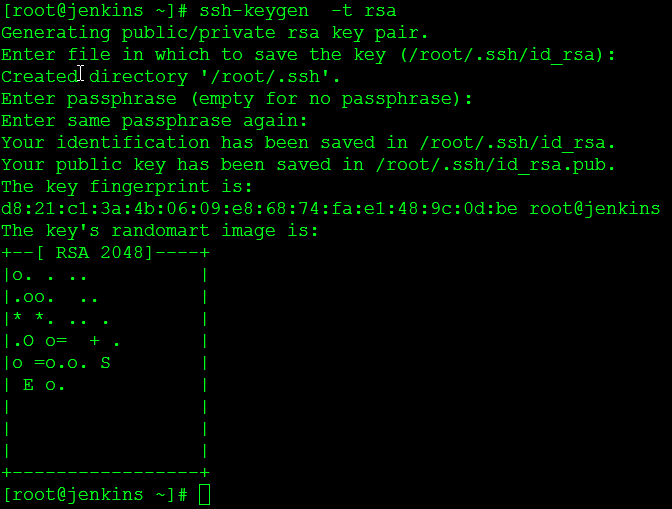


Jenkins安装完成

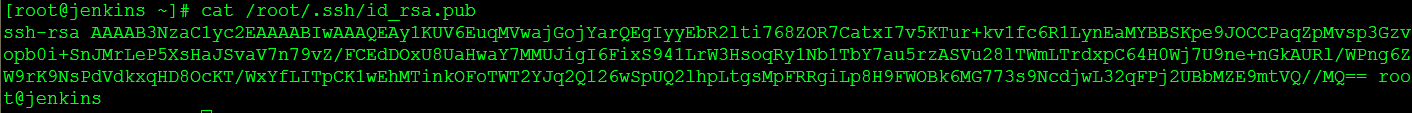
我们还需要一个key证书，用于Jenkins和服务器和GitLab之间建立信任关系(更新/克隆代码)

回到Jenkins服务器上执行命令： ssh-keygen -t rsa

[root@ubt-jenkins ~]# ssh-keygen -t rsa

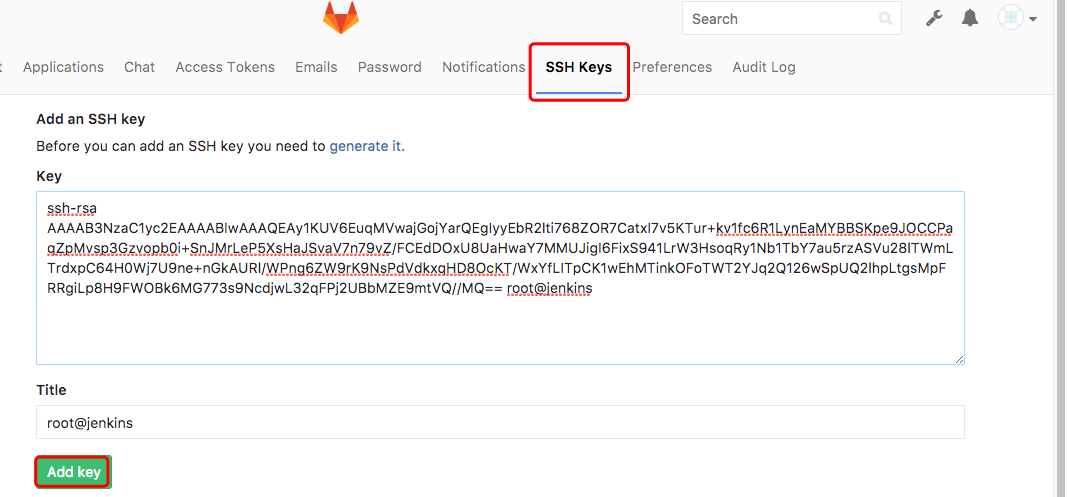


文件生成完成，cat /root/.ssh/id\_rsa.pub得到内容

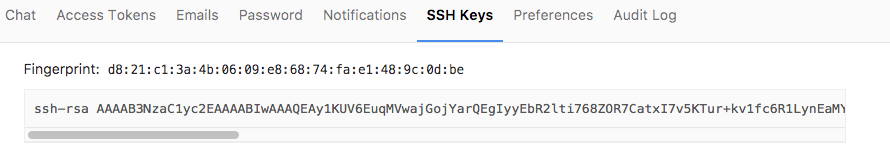


复制文件/root/.ssh/id\_rsa.pub内容，将其添加到Gitlab SSH Keys管理中

后续需要用于Jenkins从Gitlab 仓库拉取代码并同步的实际的业务服务器

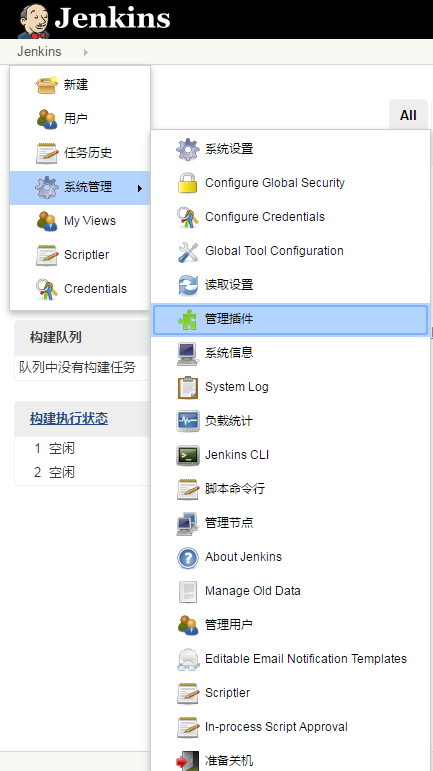


在用户设置找到SSH Keys如上图所示🡪然后添加/root/.ssh/id\_rsa.pub文件内容并点击Add key



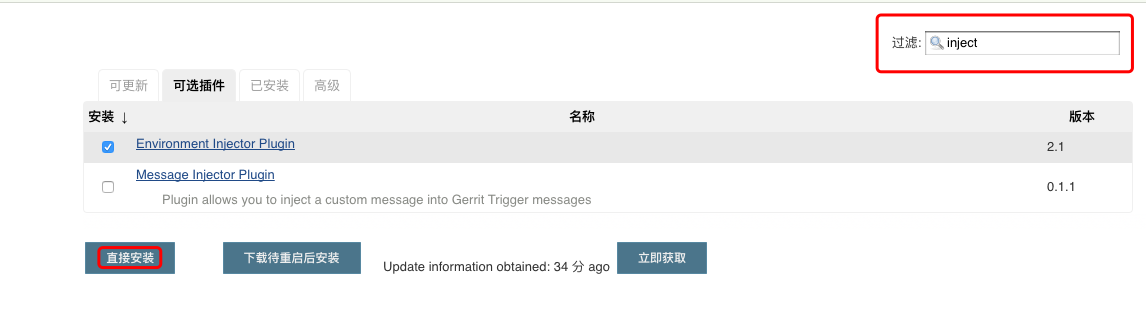
看到已经添加的SSH Keys

## Jenkins插件安装



依次点击Jenkins🡪系统管理🡪管理插件





在右上角过滤处输入插件名称🡪勾选插件🡪点击直接安装

我们需要安装插件来满足参数化构建

以下插件请自行按需安装：

*  Groovy
*  ThinBackup
*  Publish Over SSH
*  Git Parameter Plug-In
*  Email Extension Plugin
*  Multiple SCMs plugin
*  Active Choices Plug-in
*  Maven Integration plugin
*  Environment Injector Plugin
*  Extensible Choice Parameter plugin
*  Dynamic Extended Choice Parameter Plug-In

**插件功能解读：**

Groovy Jenkins执行Groovy代码的能力

我们将用该插件编写实现Groovy代码获取Git仓库的版本和历史备份记录

ThinBackup 备份全局配置(Jobs的任务配置，不包含插件和工具)

用于备份、恢复Jobs配置

Publish Over SSH Jenkins可直接在需要发布的节点上执行脚本

用于Jenkins支持在远程节点行执行

Git Parameter Plug-In 使得Jenkins可以读取GIt分支的版本号或Tag号

可用于获取Tag版本号

Multiple SCMs plugin Jenkins可克隆多个Git分支

下载配置文件和项目源码(git版本仓库)

Environment Injector Plugin 参数化构建核心插件

使jenkins支持参数化构建

Maven Integration plugin 使得Jenkins支持创建一个maven项目

使Jenkins支持插件maven项目

Dynamic Extended Choice Parameter Plug-In

Extensible Choice Parameter plugin

Environment Injector Plugin

用于根据选择来决定打印不同的命令结果

# 全局工具和系统设置

## 全局工具配置

我们还需要设置全局工具如果Groovy和Maven的安装或者配置工具的安装目录



以此点击Jenkins🡪系统管理🡪全局工具设置



Jenkins提示没有在Jenkins服务器上安装Git工具

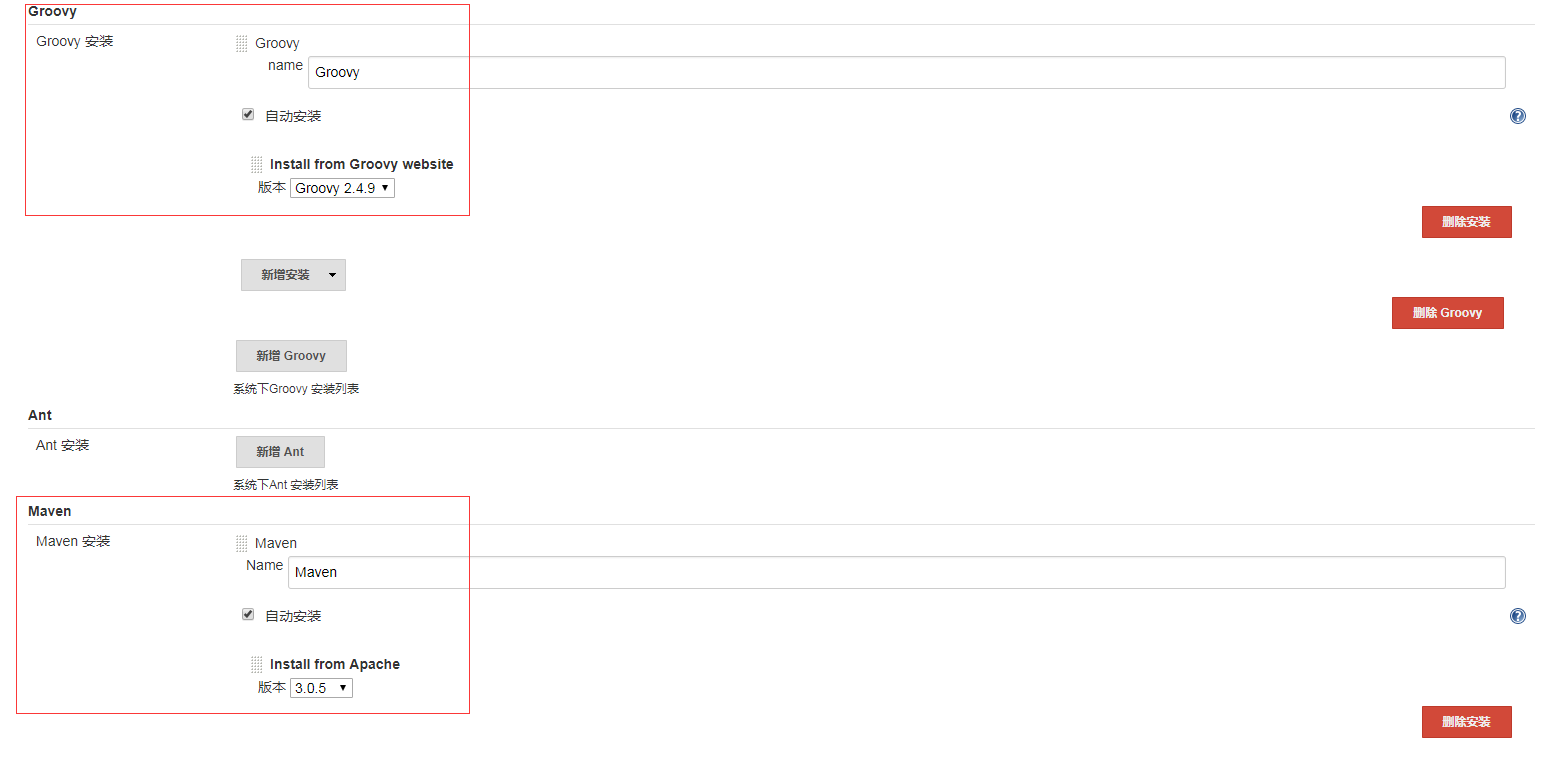
回到Jenkins服务器上执行命令：

[root@jenkins ~]# yum install -y git

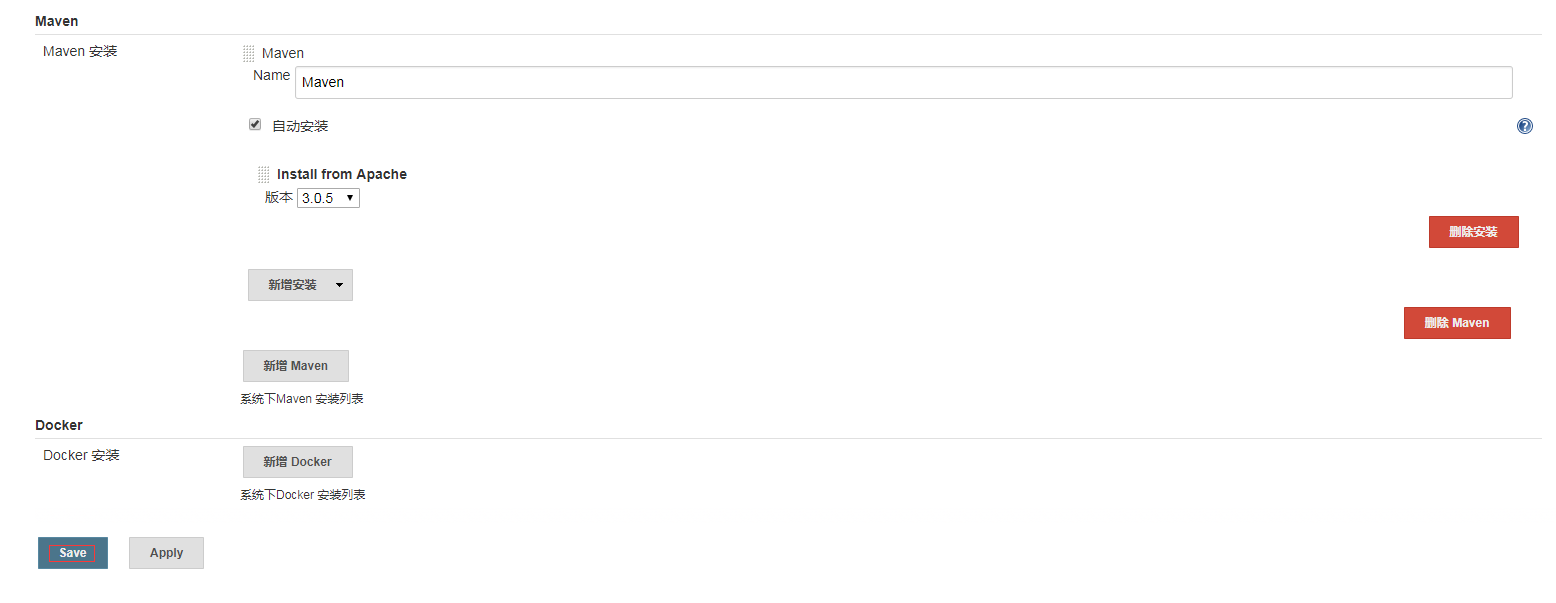
[root@jenkins ~]# git --version

git version 1.7.1

[root@jenkins ~]#



设置Groovy和Maven勾选上自动安装(如上图所示)



选择版本后🡪点击”Save”

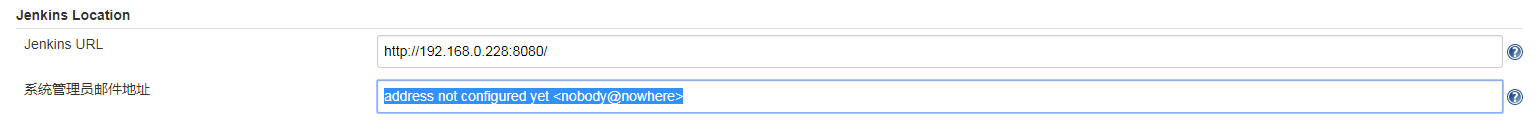
## 系统设置



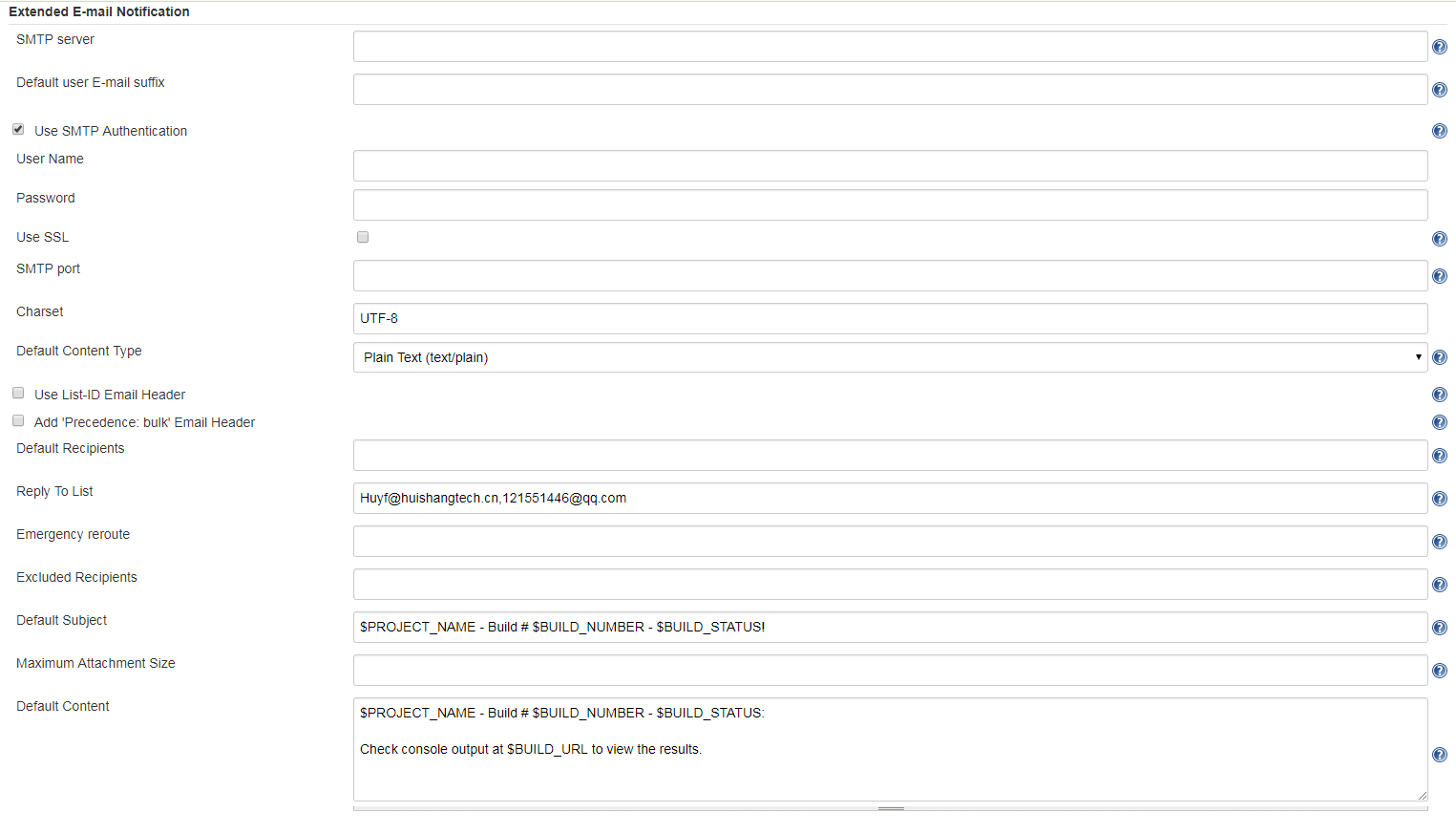
Jenkins🡪系统管理🡪系统设置



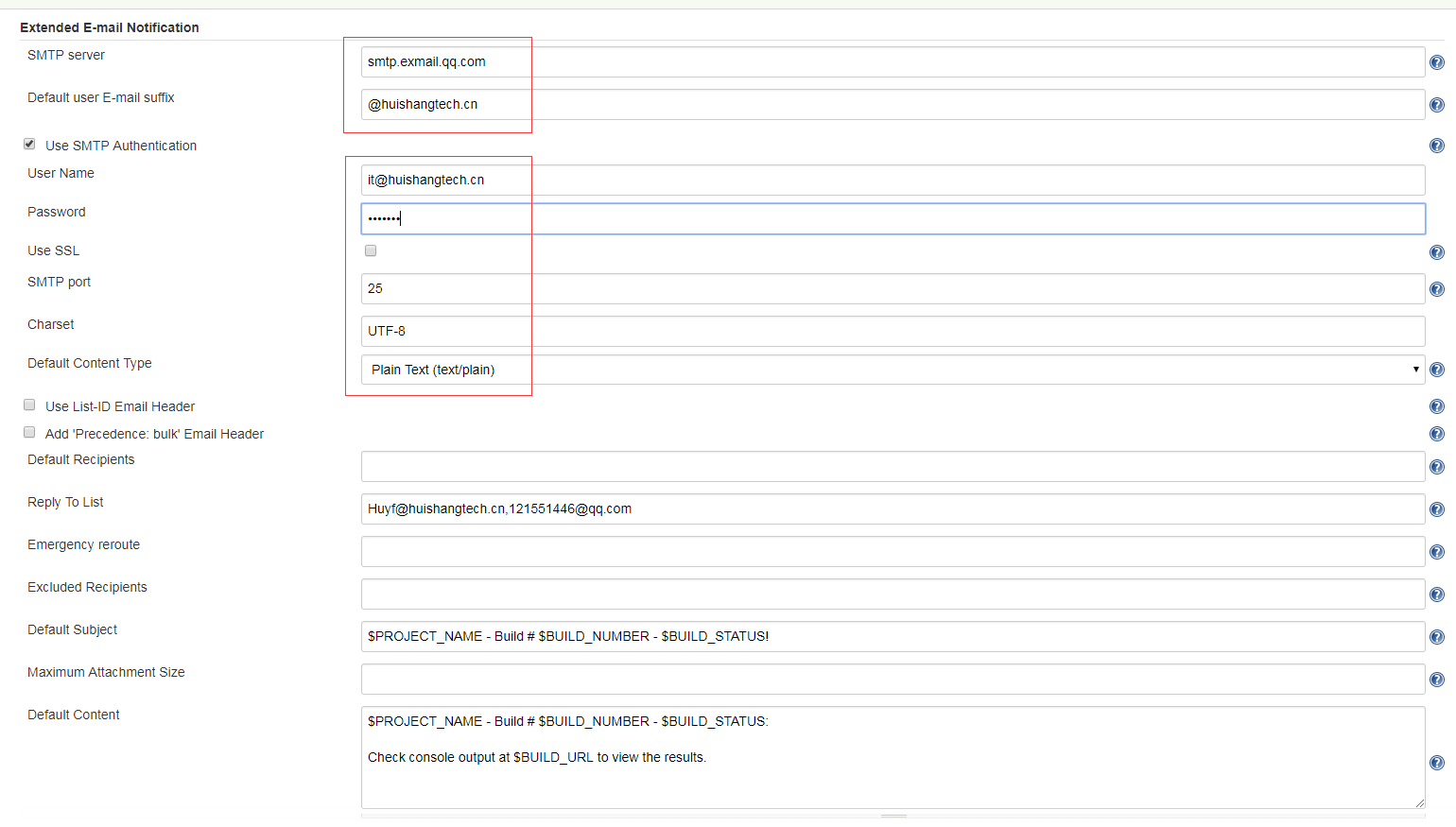
设置Maven的安装目录



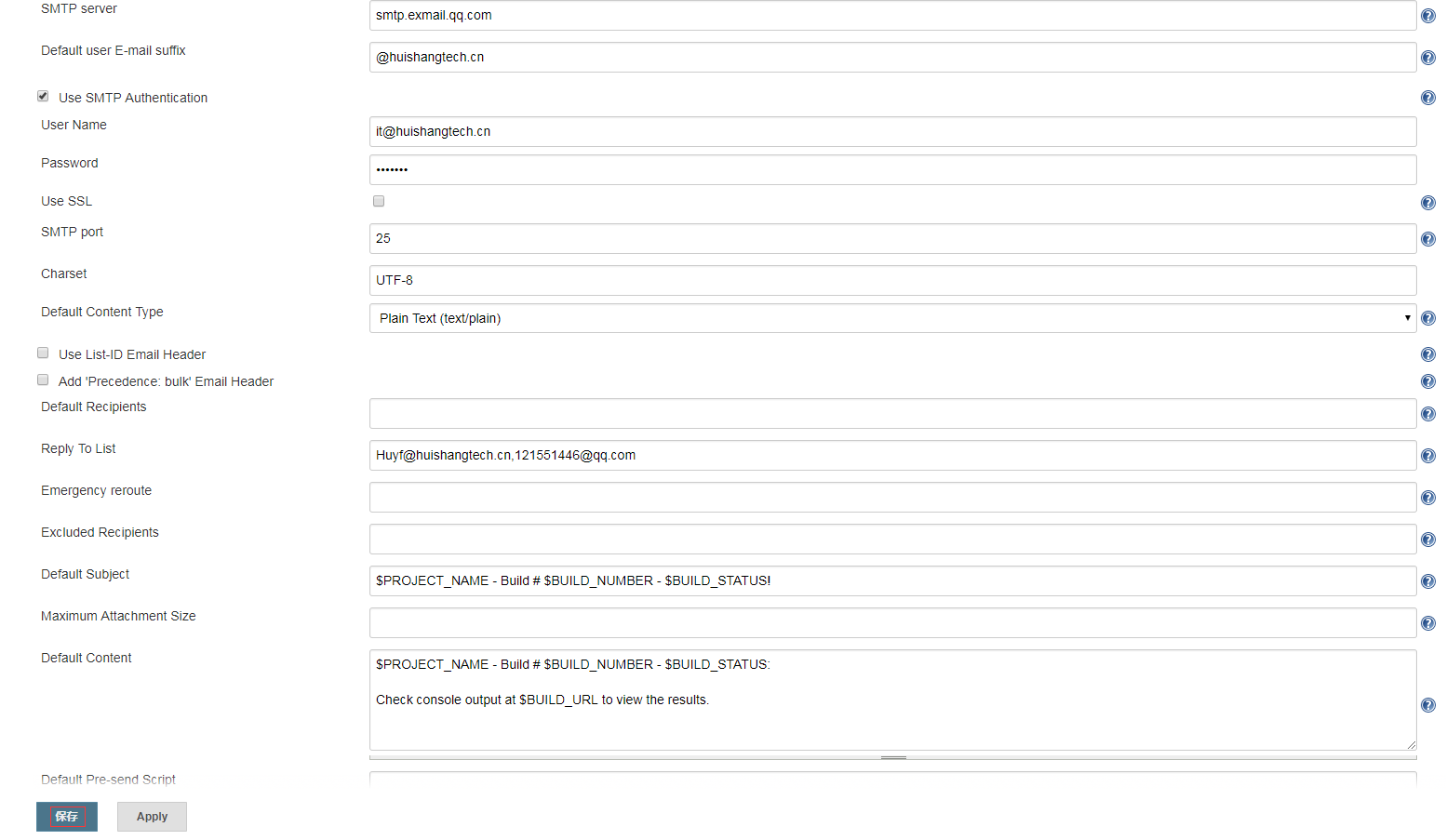
设置管理员邮箱，请自行填写



配置Extended E-mail Notification插件



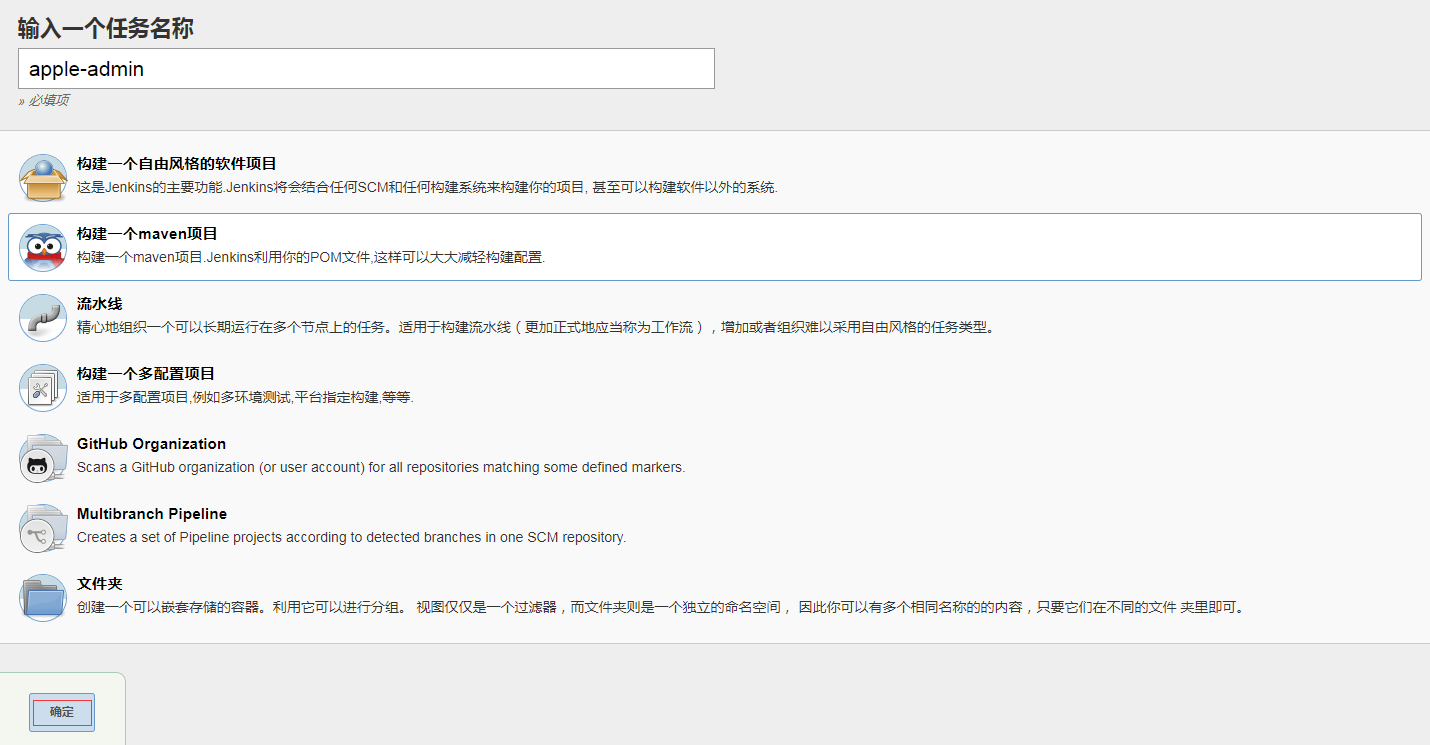
设置smtp服务器、权限、收件人等相关信息(如上图所示)



设置完成，点击”保存”

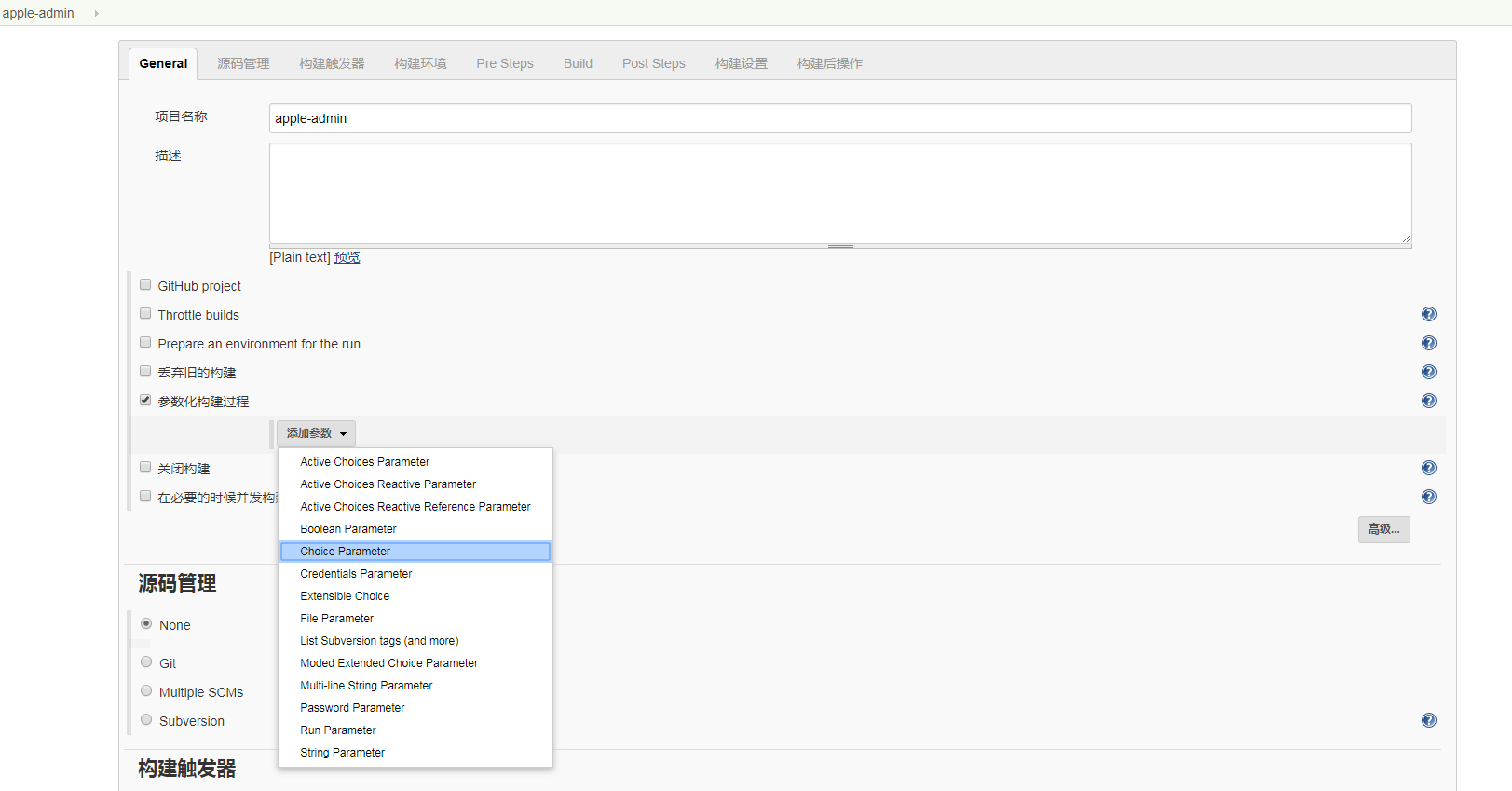
# 创建可以被maven参数化构建的项目

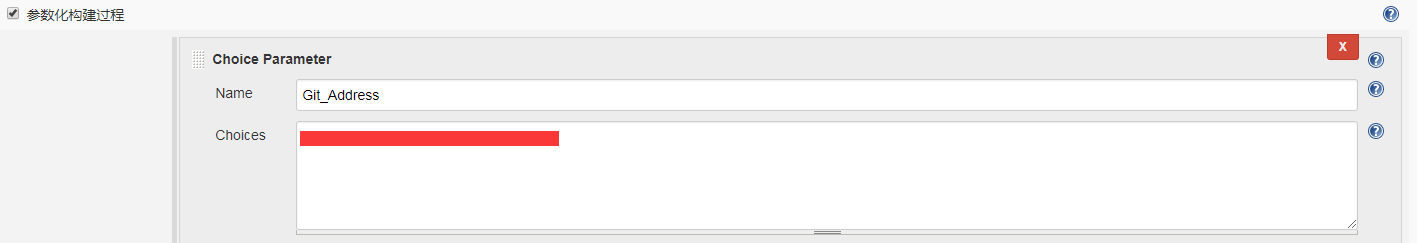
## 新建任务



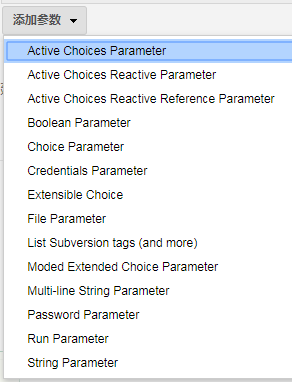
输入任务名字🡪选择构建一个maven项目🡪点击”确定”

## 参数化构建配置

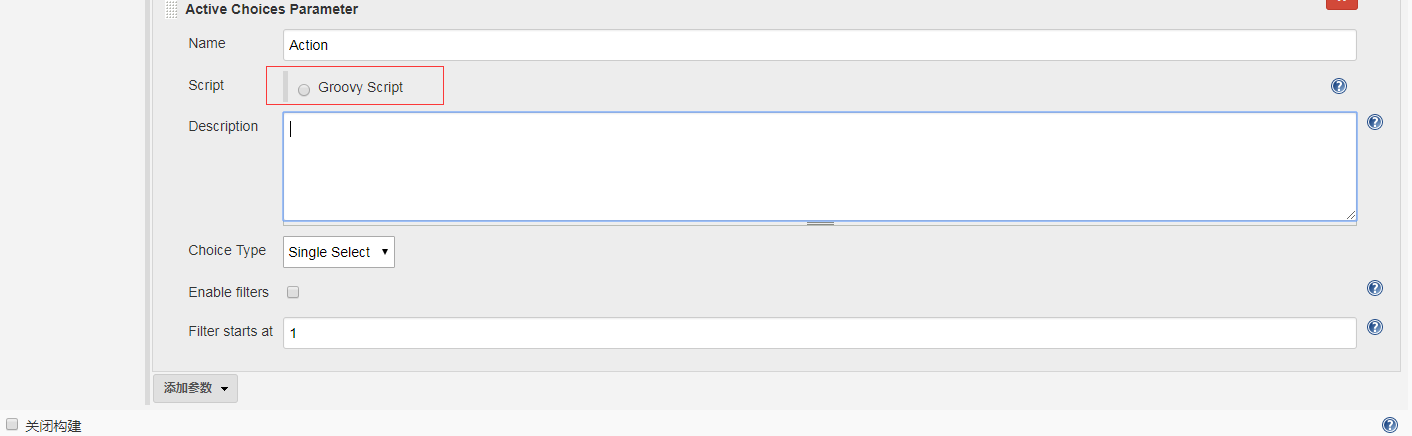


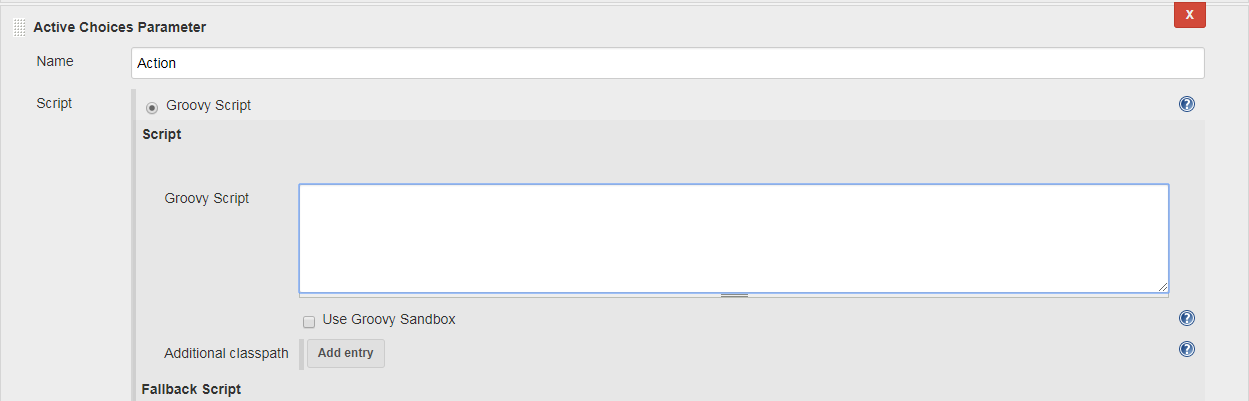
勾选”参数化构建过程”🡪选择”Choice Parameter”

设置一个变量名如果”Git\_Address”，后面会在构建脚本及Groovy脚本中调用变量

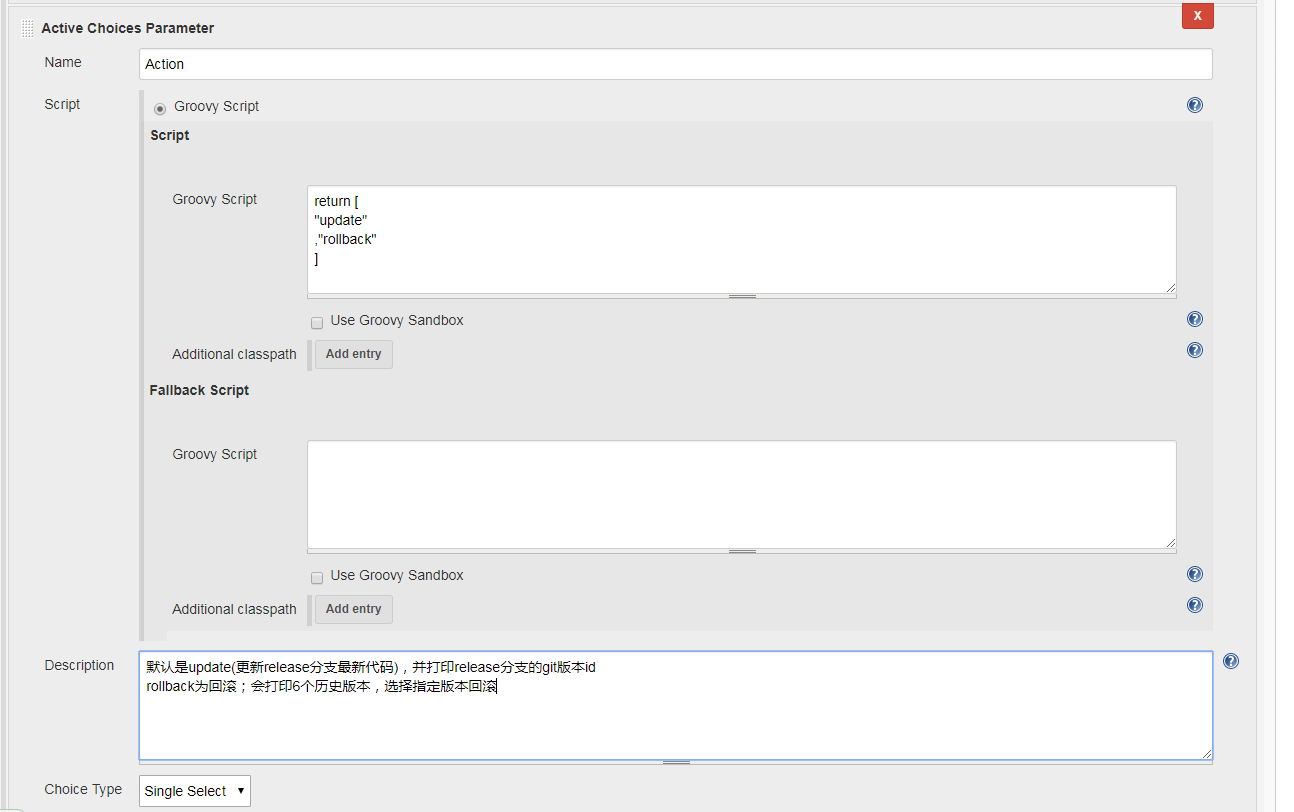


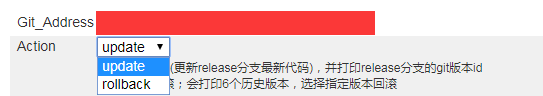
继续添加参数” Active Choice Parameter”



填入变量名并点击Goroov Script(如上图)

我们可在Groovy Script选项框中填写Groovy脚本





最终实现可选择的Action视图(如上图所示)

return [

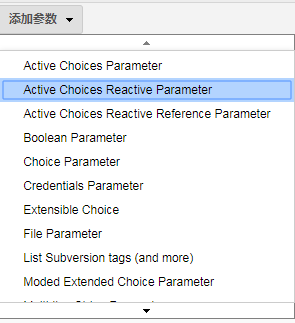
"update"

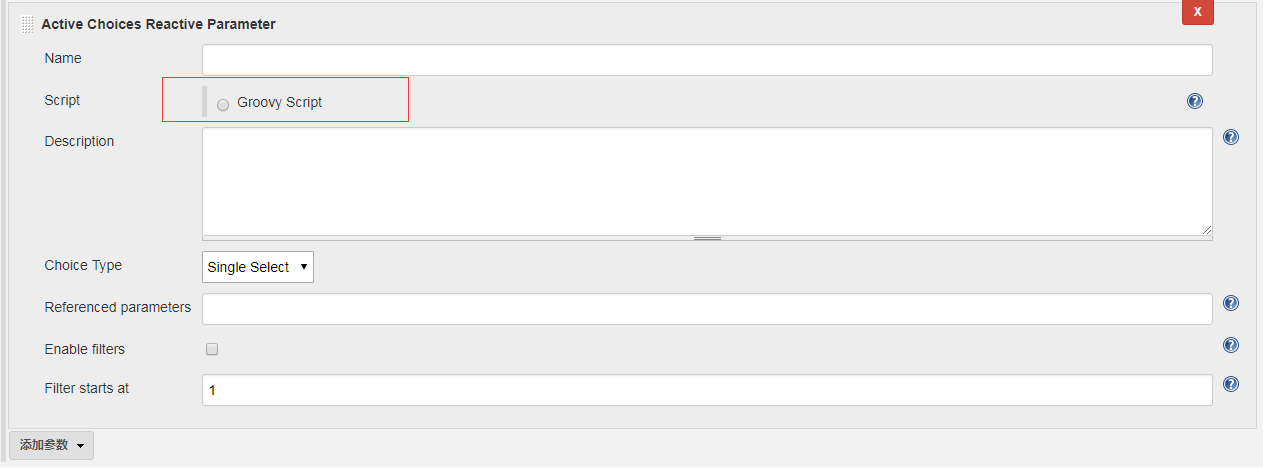
,"rollback"

]

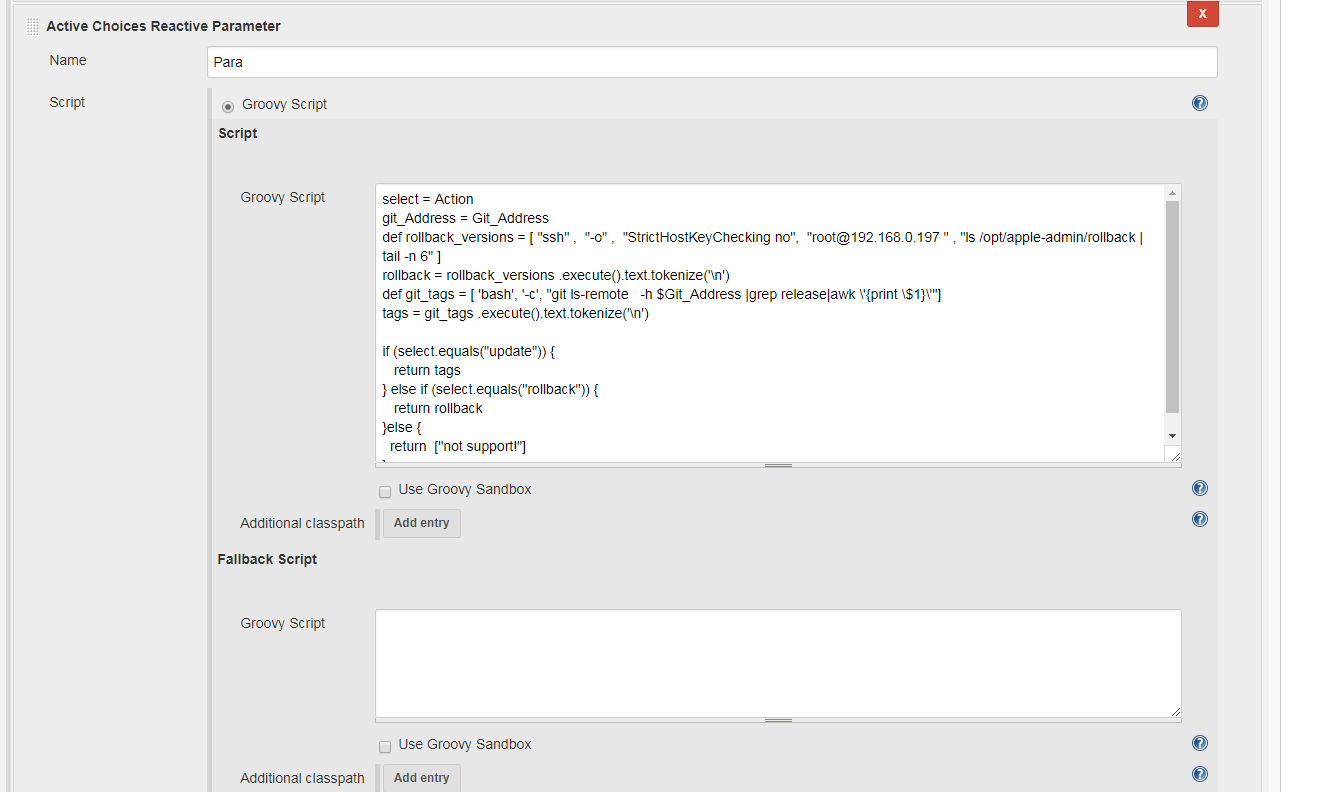
将以上内容粘贴进Groov Script并填写说明

[根据实际情景和环境自行修改Action其他操作如备份等]

  
继续添加参数” Active Choice Reactive Parameter”



[同样在点击Groovy Script选项框内启用Groovy Script支持]



[根据实际场景来修改Groovy Script中的Groovy Script脚本]

select = Action

git\_Address = Git\_Address

def rollback\_versions = [ "ssh" , "-o" , "StrictHostKeyChecking no", "root@192.168.0.197 " , "ls /opt/apple-admin/rollback | tail -n 6" ]

rollback = rollback\_versions .execute().text.tokenize('\n')

def git\_tags = [ 'bash', '-c', "git ls-remote -h $Git\_Address |grep release|awk \'{print \$1}\'"]

tags = git\_tags .execute().text.tokenize('\n')

if (select.equals("update")) {

return tags

} else if (select.equals("rollback")) {

return rollback

}else {

return ["not support!"]

}

* **Groovy Script解读**

select = Action # 变量名为Action

git\_Address = Git\_Address # 调用了之前创建的变量(Git地址)

def rollback\_versions = [脚本逻辑：是通过ssh在远程执行命令来获取发布节点上备份的历史文件并打印列表赋给Para变量，用于回滚时调用]

def git\_tags = [通过git命令获取远程git仓库分支最新的版本号，赋给Para变量用于发布时调用]

if (select.equals("update")) {

return tags

} else if (select.equals("rollback")) {

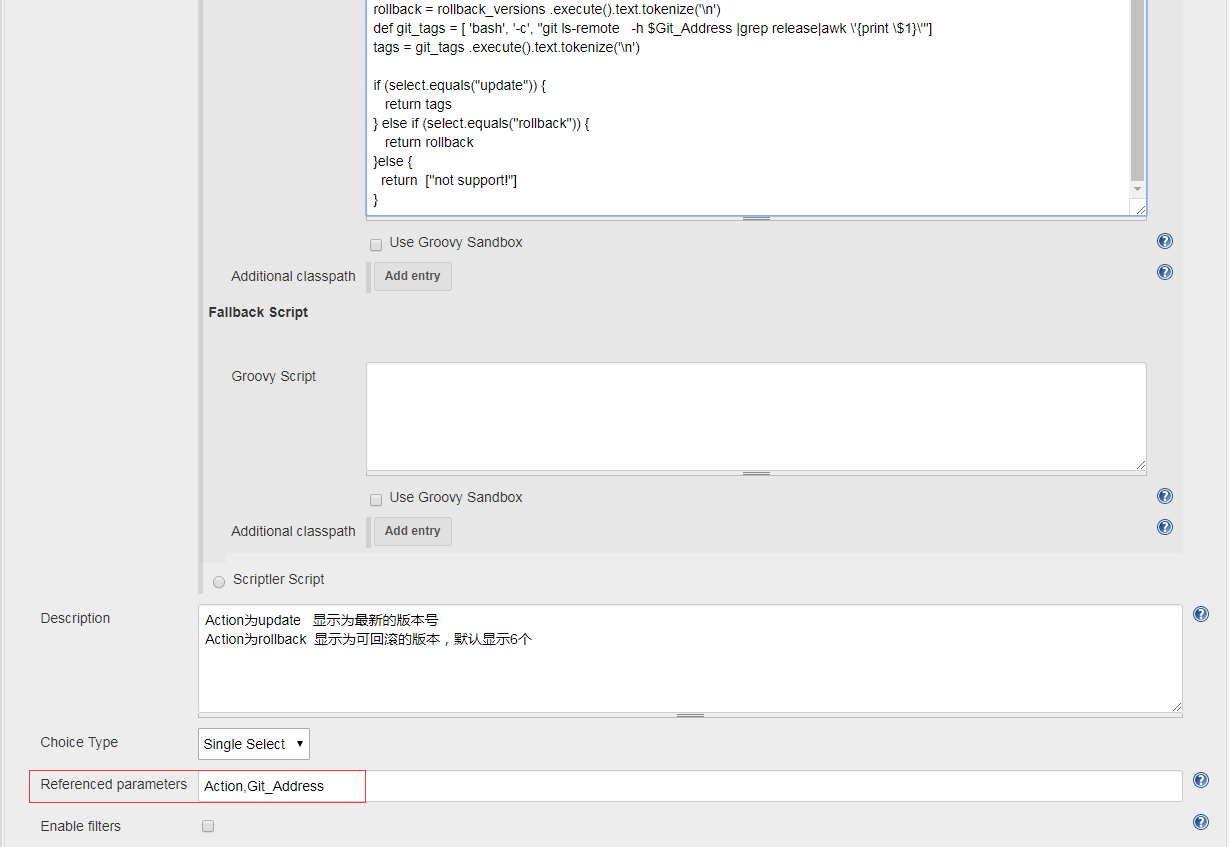
return rollback

}else {

return ["not support!"]

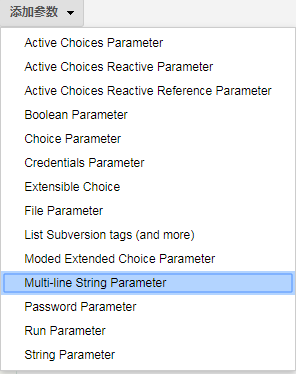
}

Action选项判断，根据选择了不同的Action选项来调用不同的命令参数

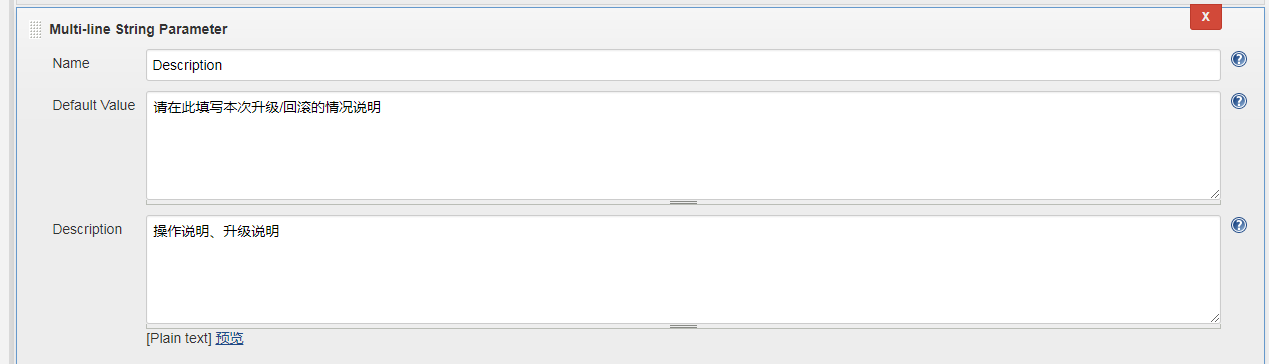
[根据实际的场景、环境自行修改脚本内容]

添加说明/描述

并在”Referenced parameters”选项添加”Action,Git\_Address”(如上图所示)

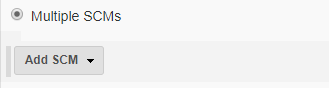


继续添加参数” Multi-line String Parameter”

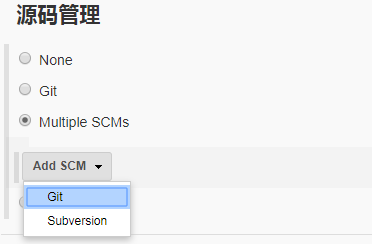


升级时应添加升级/回滚说明”包括但不限于升级内容、回滚回滚、业务升级说明/回滚说明”

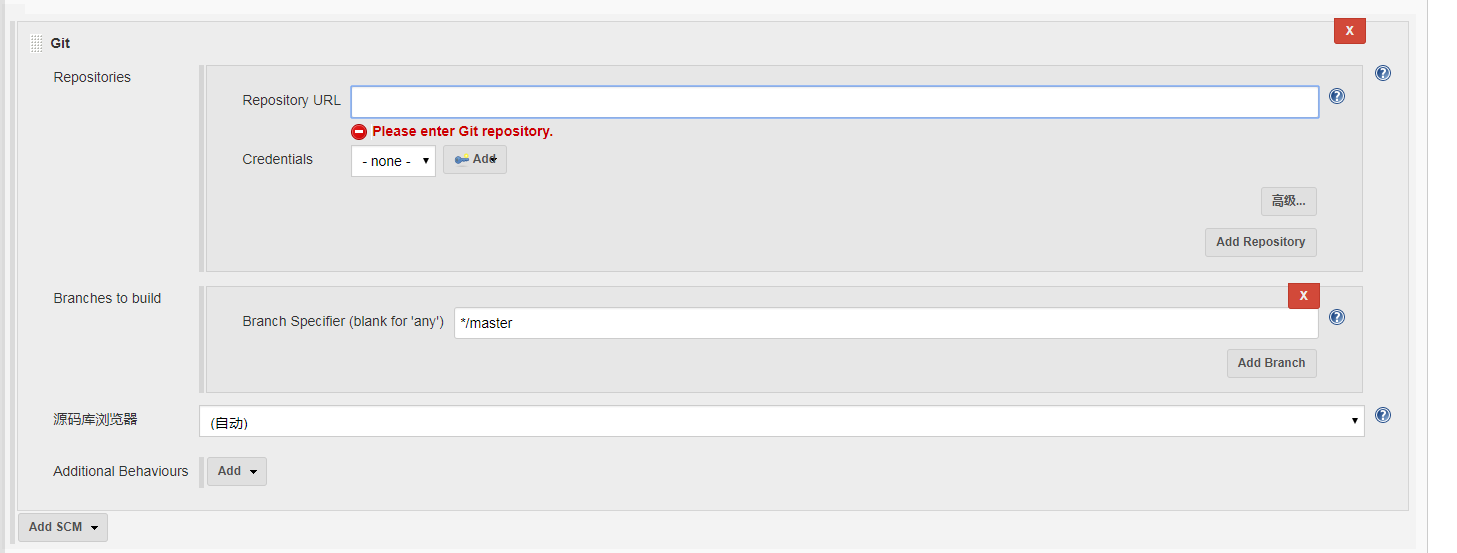
## 源码管理



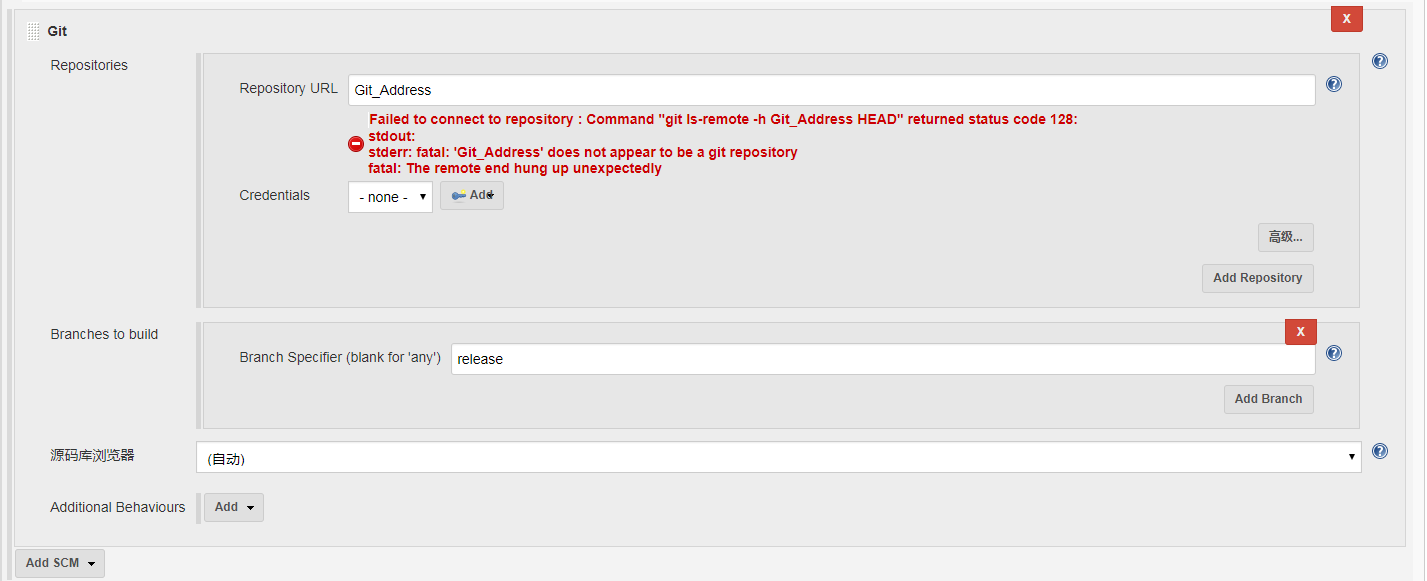
选择Multiple SCMs



支持Git和SVN(根据实际的工具选择)



添加上面的Git\_Address变量(里面包括了Git地址)及分支



提示无法连接到存储库，需要需要添加授权

## 添加授权

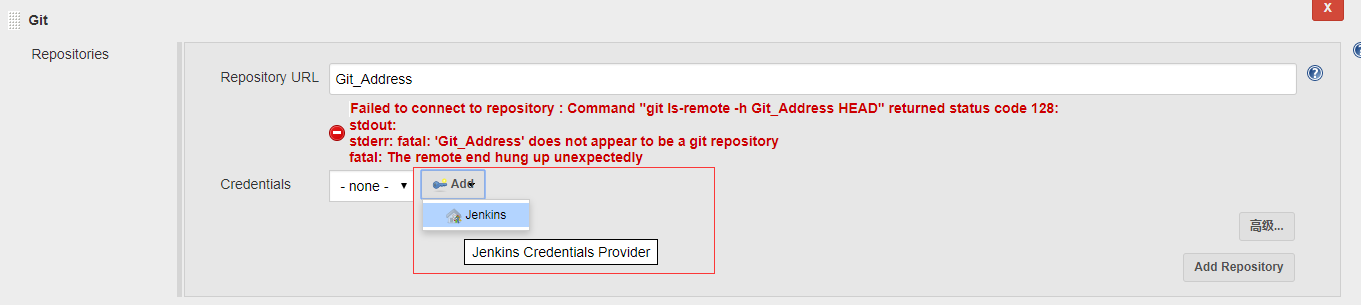
Jenkins需要从Git上下载代码，Git需要配置认证

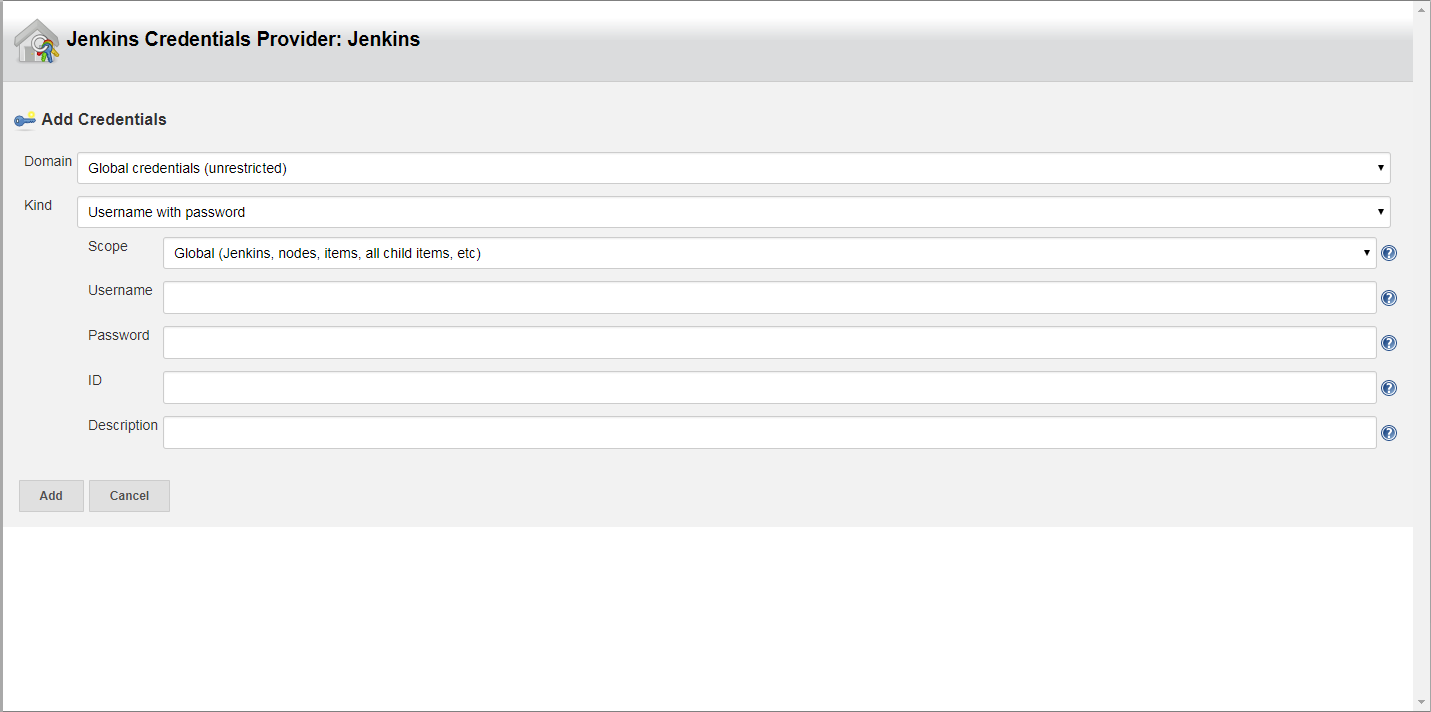
回到Jenkins服务器上执行命令

[root@jenkins ~]# cat /root/.ssh/id\_rsa

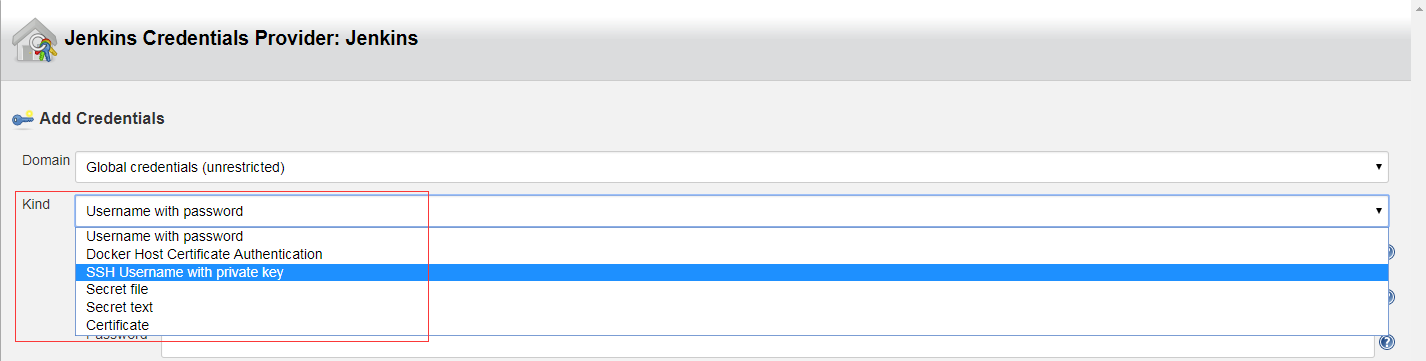


复制以上cat命令输出得到的内容

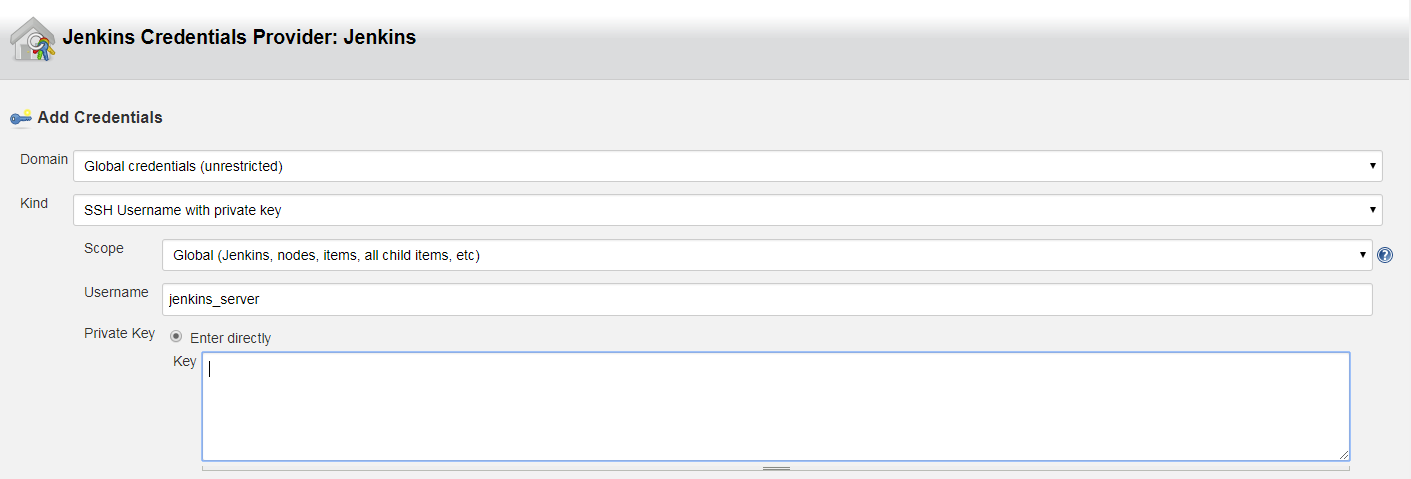


回到Jenkins项目添加页，点击Add🡪jenkins

进入添加Jenkins证书添加页面



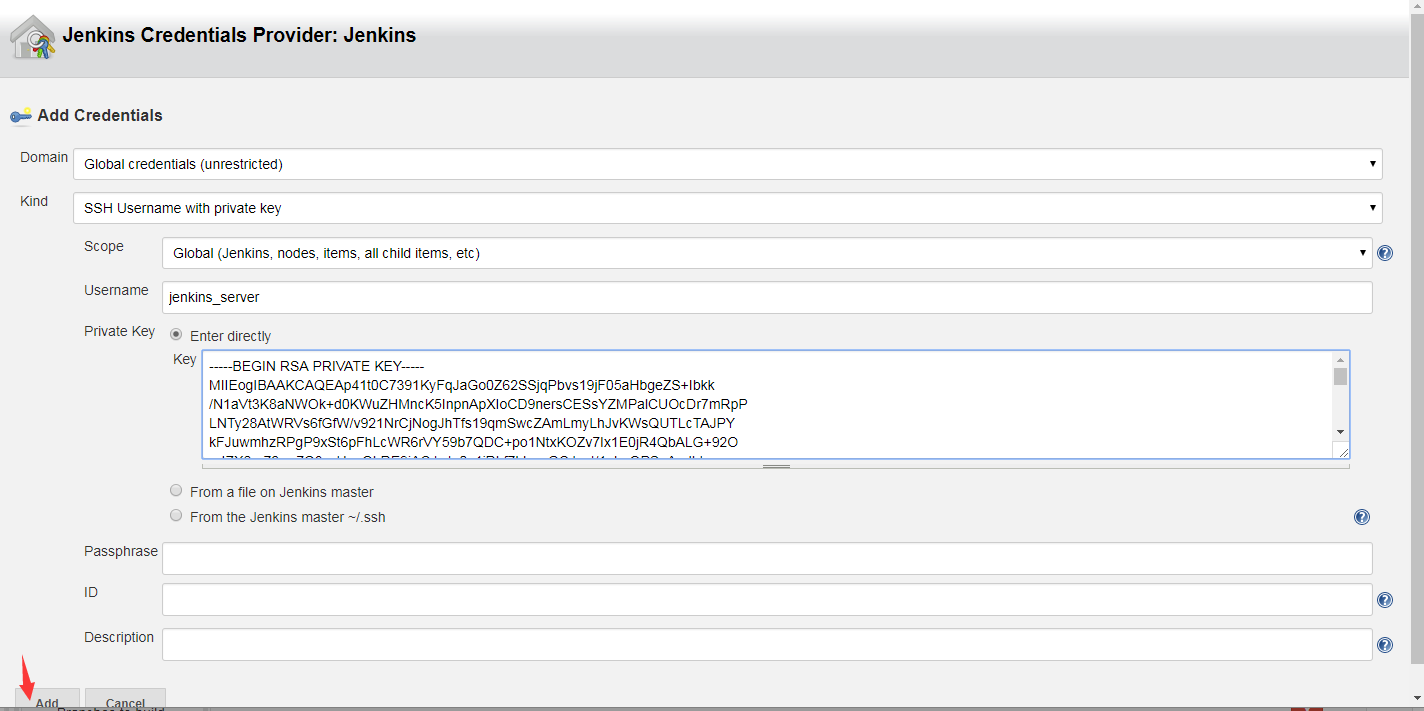
点击Kind，选择” SSH Username with private key”

s

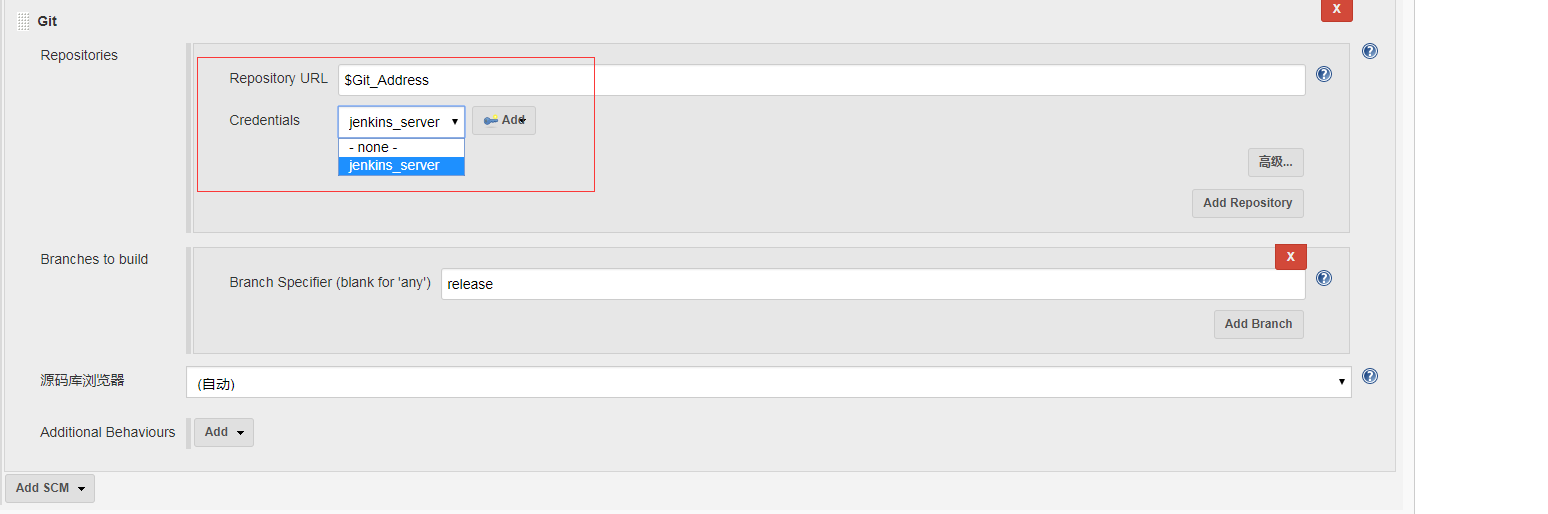
填写username(非系统用户，理论上随便写都可以)点击”Enter directly”



将之前在Jenkins节点上复制的/root/.ssh/id\_rsa(上图内容)粘贴到”Key”框中



粘贴完成后点击”ADD”完成添加(如上图所示)



在Credentials选项中，选择刚才新建的用户(Jenkins\_server)

注意[据实际创建来填写分支名称]

## 查看是否能获取到远程版本号



我们点击保存看下是否能够正常打印远程仓库的版本号



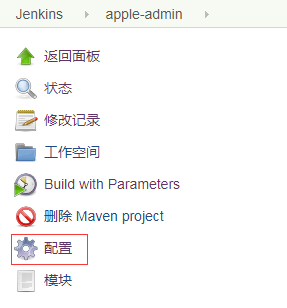
选择参数化构建(如上图所示)



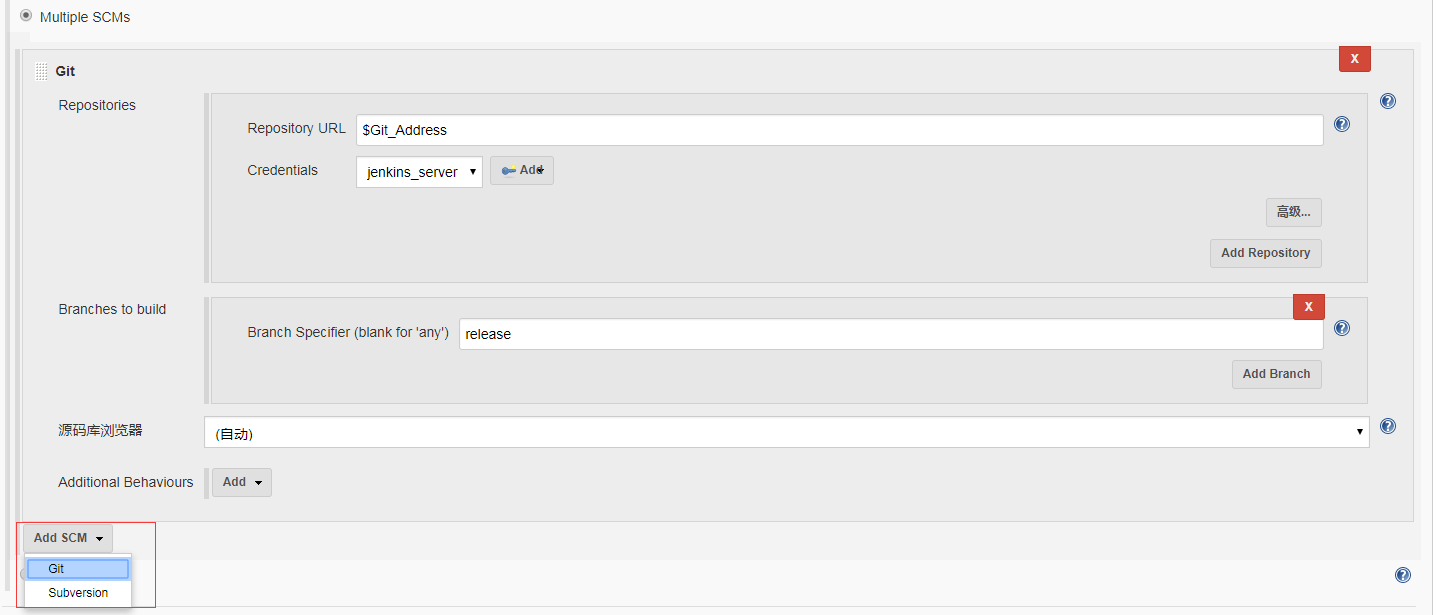
如能正常打印出版本号，说明脚本和参数配置都是正确的

如果不能正常打印请检查(版本地址，插件及Groovy脚本的是否准确)

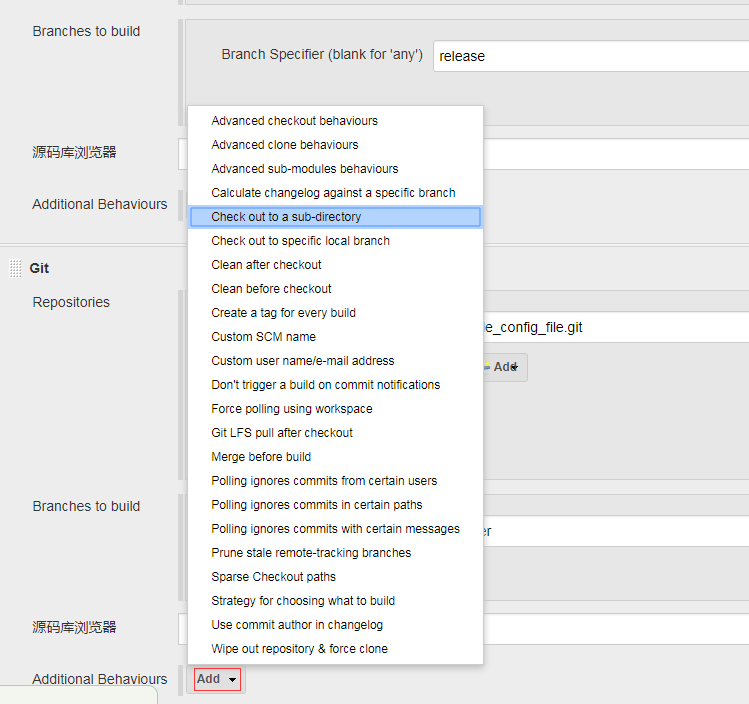
## 添加配置文件仓库(Git仓库)



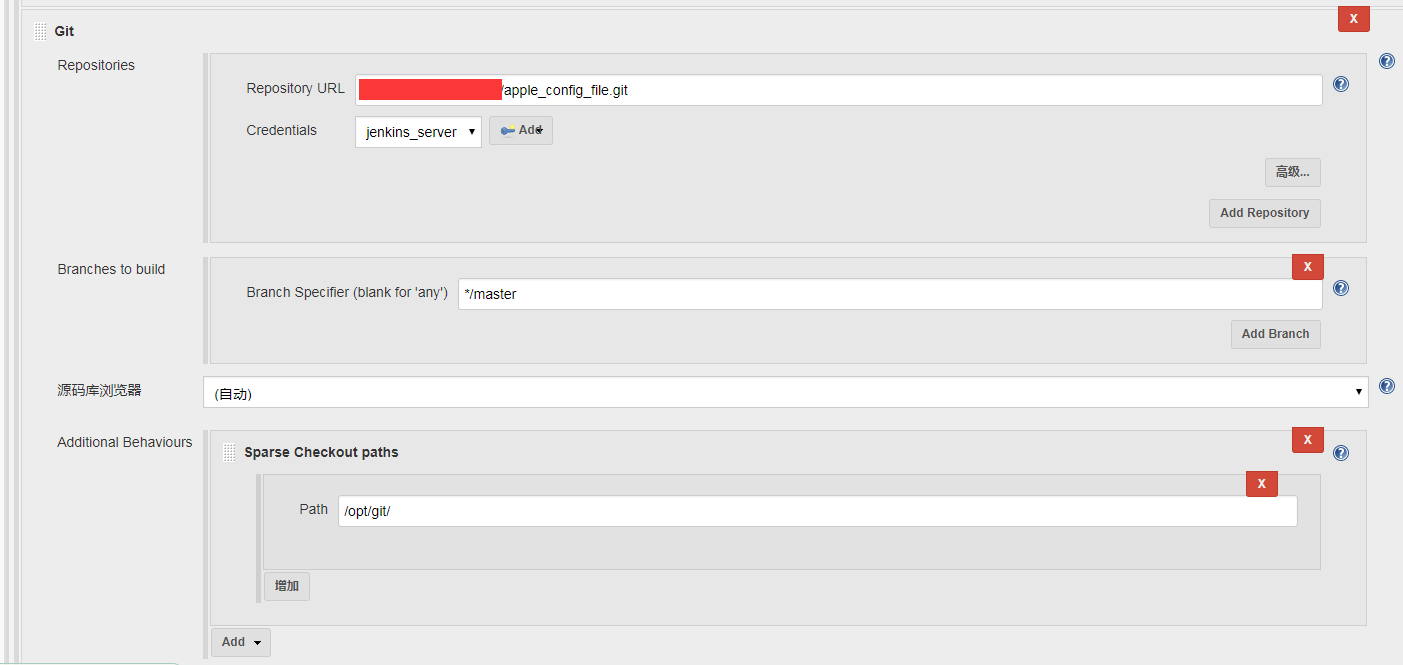
回到项目点击🡪配置



业务配置文件在Git仓库上托管需要配合结合项目一起克隆(Git Clone)

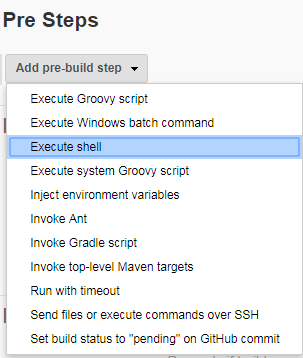


在”Additional Behaviours”处选择” Check out to a sub-directory”



设置Git项目存放目录/opt/git/ [Jenkins能够自动创建]

## Pre Steps



[配置jenkins在构建前需要的操作]

点击Add pre-build step🡪Execute shell



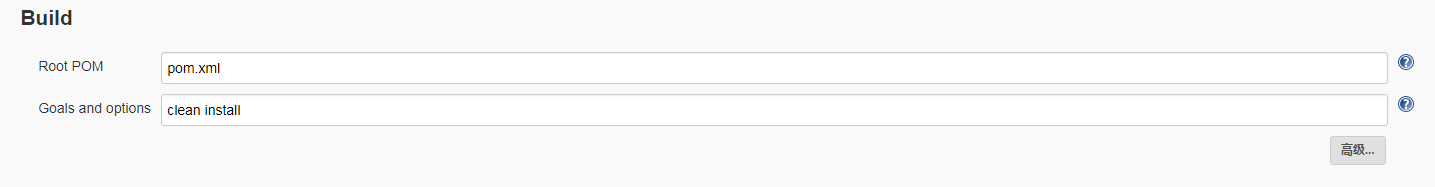
拷贝配置文件到指定项目(源码目录)

* **解读Command脚本/命令：**

cp -r /opt/git/apple\_config\_file/admin/\* $WORKSPACE/src/main/resources/

拷贝该项目的配置文件到本项目在Jenkins上的临时目录

## 设置maven Build参数



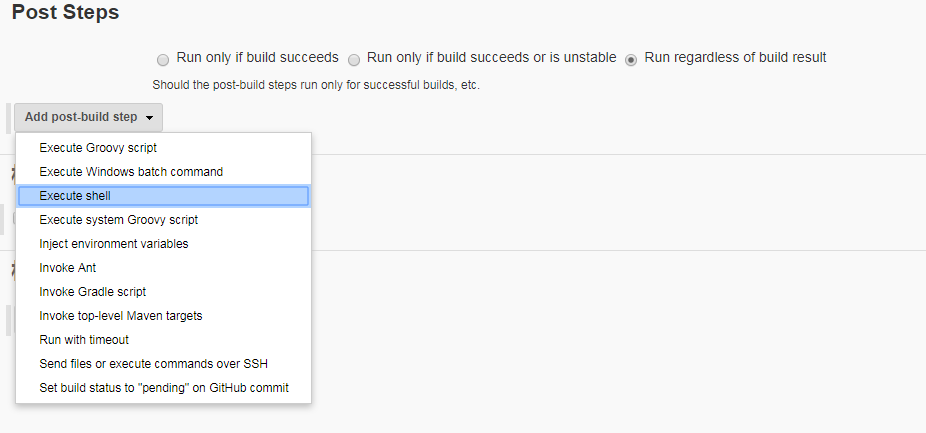
* **Maven常用参数：**

clean install

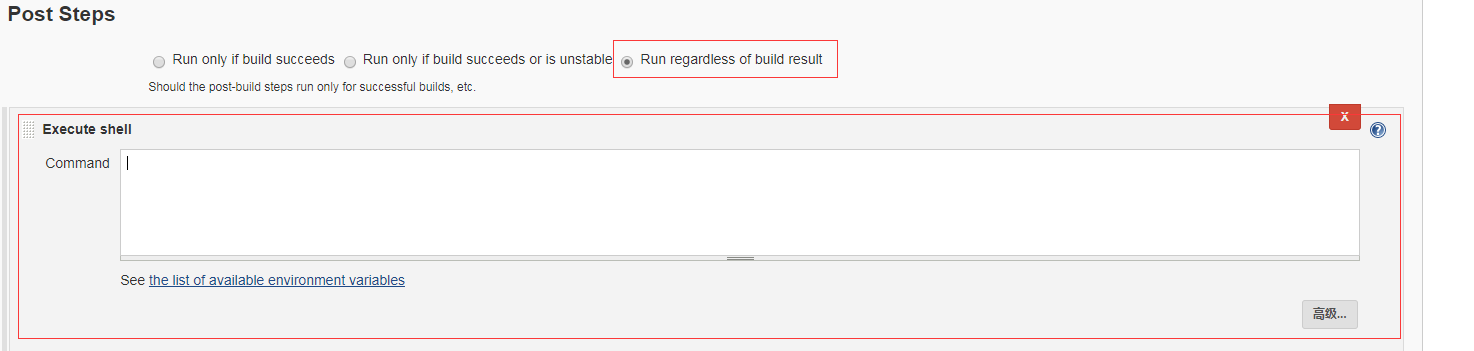
clean install -Dmaven.test.skip=true #跳过测试构建

## 升级和回滚配置(Post Steps)

* 升级配置



添加构建后操作🡪Execute shell



在上图Command中添加构建命令



[利用脚本配合工具协助来完成升级操作]

if [ "$Action" = "update" ] ; then

#scp /var/lib/jenkins/.m2/repository/com/monghoo/apple\_admin/1.0-SNAPSHOT/apple\_admin-SNAPSHOT.war root@192.168.0.197:/root/apple-admin.war

ssh root@192.168.0.197 "bash -s -x" < /var/lib/jenkins/apple\_update.sh apple-admin

fi

* **Command解读：**

1.判断如果Action为update(升级操作)，执行正常的构建操作并将生成好的scp到远程的服务器上

2.将Jenkins在本地的脚本能在远程服务器上执行bash -s -x参数，来完成升级操作

* **apple\_update.sh脚本：**

#!/bin/sh

DATE=`date "+%F-%H:%M"`

NAME=$1

case $NAME in

"apple-trade-base")

UPFOLDER=/home/apple-trade/apple-trade-base

;;

"apple-trade-service")

UPFOLDER=/home/apple-trade/apple-trade-service

;;

"apple-data-center")

UPFOLDER=/home/apple-trade/apple-data-center

;;

"apple-admin")

UPFOLDER=/home/apple-trade/apple-admin/

;;

"apple-risk-manage")

UPFOLDER=/home/apple-trade/apple-risk-manage

;;

"paycenter")

UPFOLDER=/home/apple-trade/paycenter/

;;

esac

DST=${UPFOLDER}/webapps/

START=${UPFOLDER}/bin/startup.sh

STOP=${UPFOLDER}/bin/shutdown.sh

/bin/cp /root/${NAME}.war ${DST}

if [ -f ${UPFOLDER}/bin/CATALINA\_PID ] ; then

/bin/bash $STOP

sleep 20

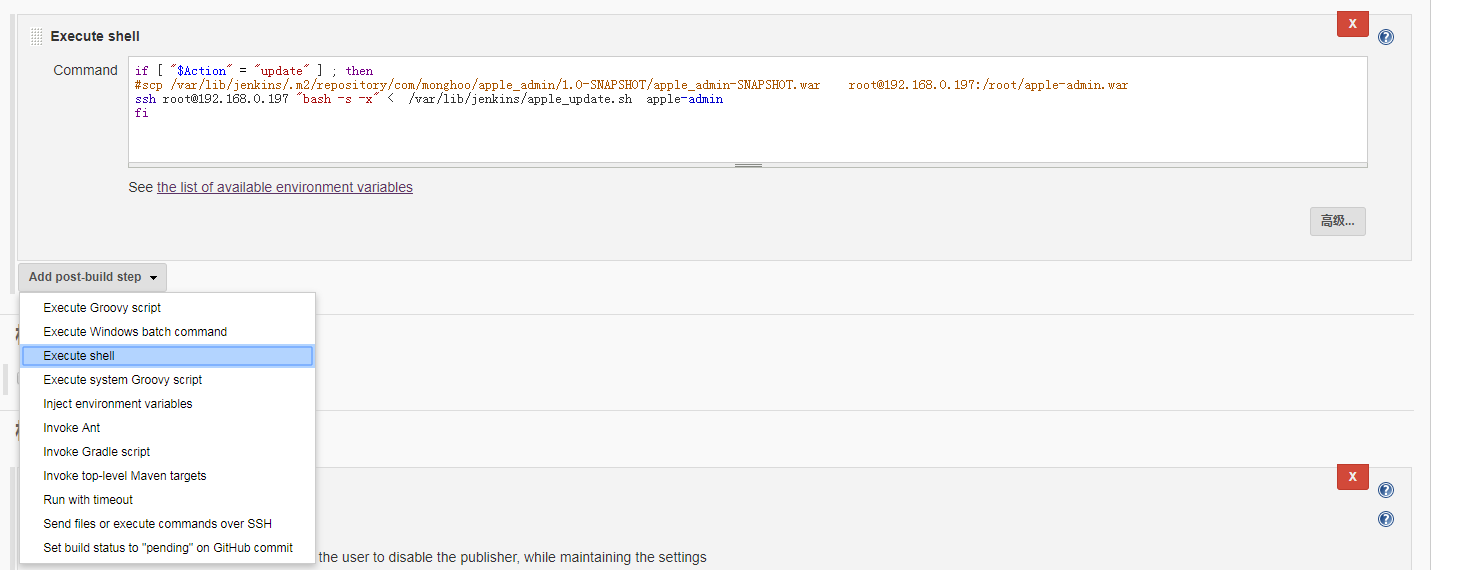
fi

rm -rf ${UPFOLDER}/webapps/${NAME}/

/bin/bash $START

cp ${DST}/${NAME}.war /opt/apple-trade/${NAME}/rollback/${NAME}\_${DATE}.war ||exit 1

#注意！脚本含自动备份操作，请结合业务自行修改脚本、创建对应目录





在上图Command中添加构建命令



[利用脚本、工具等协助来完成回滚操作]

[结合实际业务场景来修改/创建升级/回滚或备份操作]

if [ "$Action" = "rollback" ];then

ssh root@192.168.0.197 "bash -s -x" < /var/lib/jenkins/apple/rollback.sh apple-admin $Para

fi

* **Command解读：**

1.判断如果Action为rollback(回滚操作)，从Para变量读取历史文件名

2.通过linux mv或者ln命令来替换项目根目录来实现回滚操作

* **apple\_rollback脚本：**

#!/bin/sh

NAME=$1

Para=$2

case $1 in

"apple-trade-base")

UPFOLDER=/home/apple-trade/apple-trade-base

;;

"apple-trade-service")

UPFOLDER=/home/apple-trade/apple-trade-service

;;

"apple-data-center")

UPFOLDER=/home/apple-trade/apple-data-center

;;

"apple-admin")

UPFOLDER=/home/apple-trade/apple-admin/

;;

"apple-risk-manage")

UPFOLDER=/home/apple-trade/apple-risk-manage

;;

"paycenter")

UPFOLDER=/home/apple-trade/paycenter/

;;

esac

DST=${UPFOLDER}/webapps

START=${UPFOLDER}/bin/startup.sh

STOP=${UPFOLDER}/bin/shutdown.sh

/bin/cp /opt/apple-trade/${NAME}/rollback/${Para} $DST/${NAME}.war

if [ -f ${UPFOLDER}/bin/CATALINA\_PID ] ; then

/bin/bash $STOP

sleep 20

fi

rm -rf ${UPFOLDER}/webapps/${NAME}

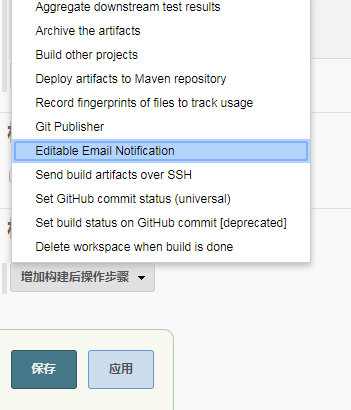
/bin/bash $START

* 回滚配置



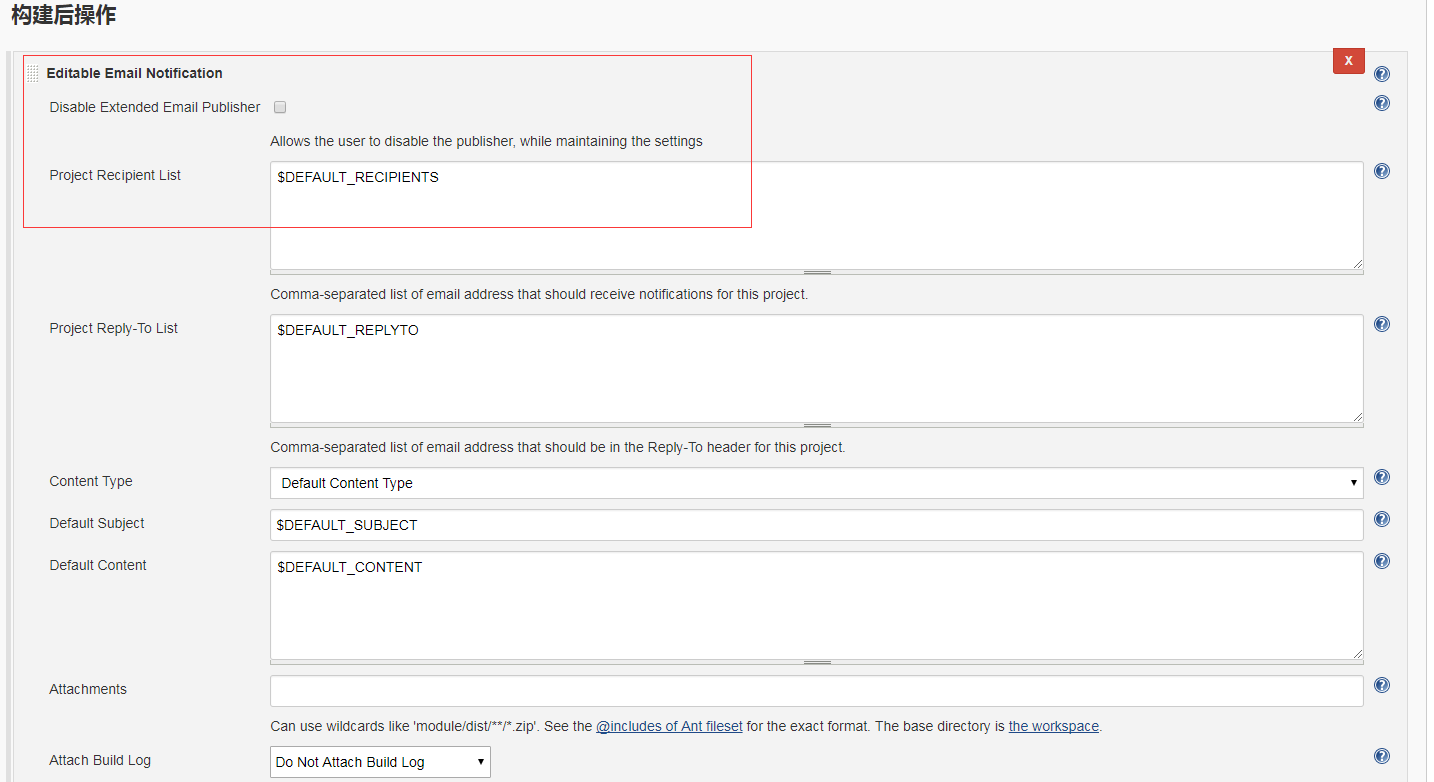
参数化构建时Action选择rollback可以看到历史版本记录

## 构建后步骤

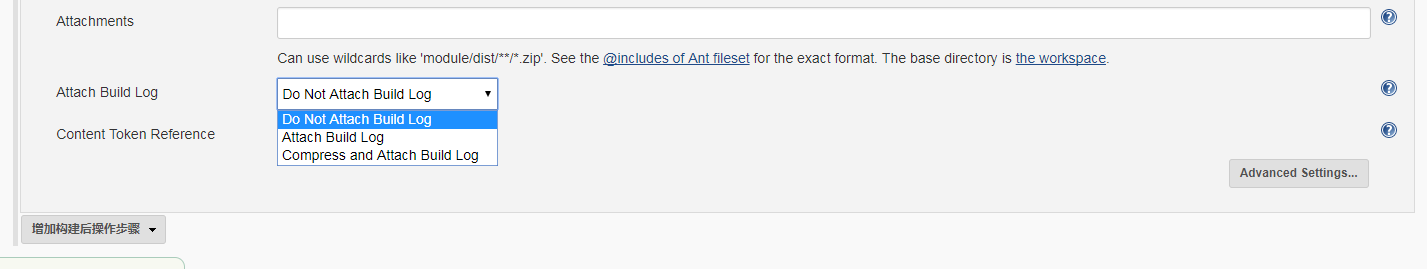


增加构建后操作步骤🡪”Edisable ExtendeD Email Publisher”

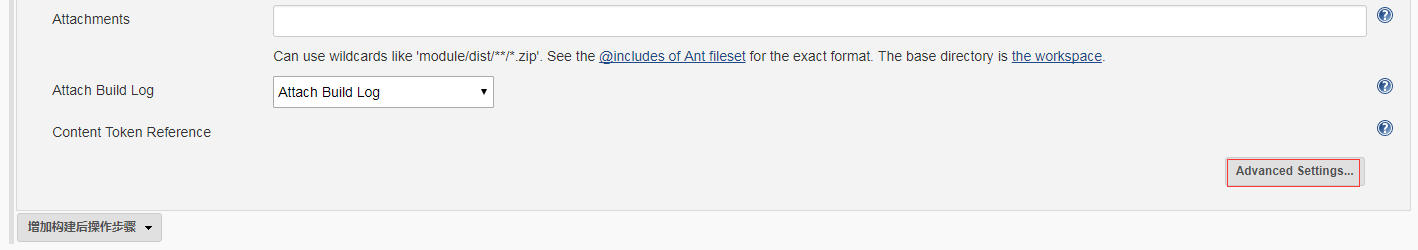
[配置构建后发送邮件通知管理员]



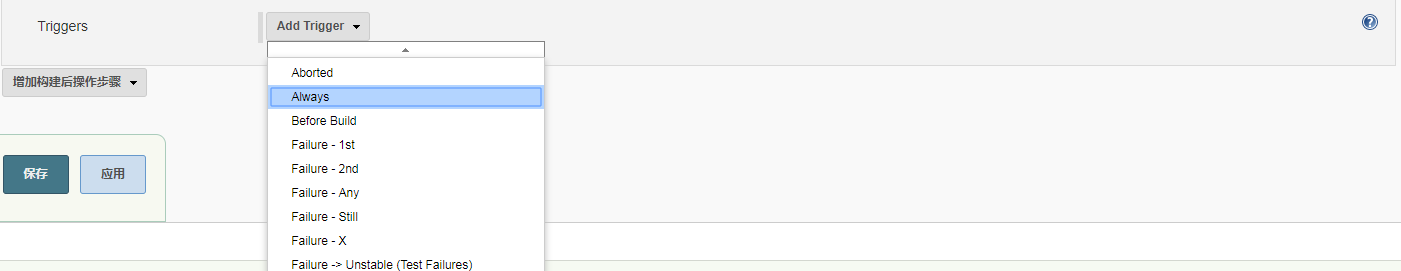
我们还可以在项目上添加新的邮件收件人



也可以配置”将构建过程添加在邮件的附件中”



点击”Advanced Settings..”



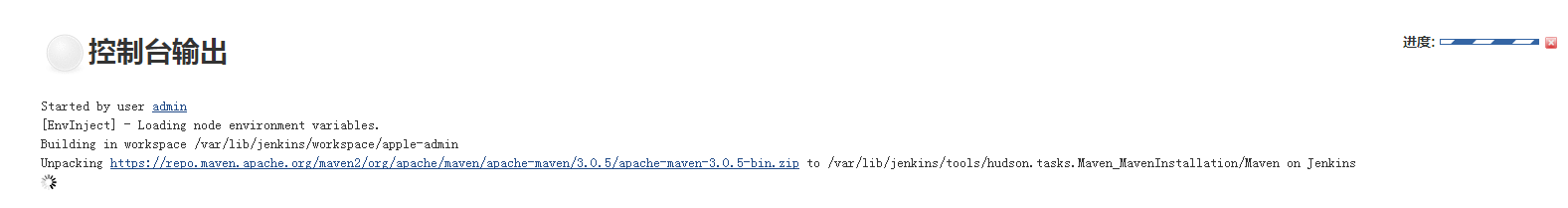
可以选择触发条件(如构建成功/失败)来发送邮件通知

# 发布和回滚测试

## 升级



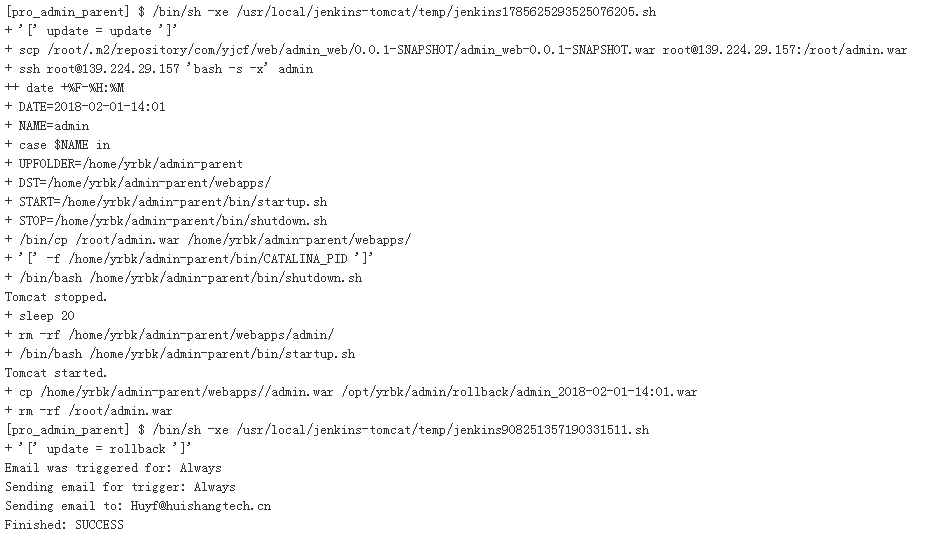
Action默认就是Update，不需选择直接点击构建构建



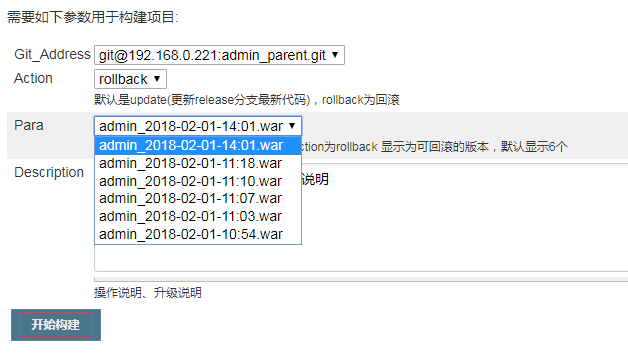
开始自动下载maven和Groovy工具



开始下载源码和配置文件并使用maven构建项目

构建成功，重启Tomcat

## 回滚

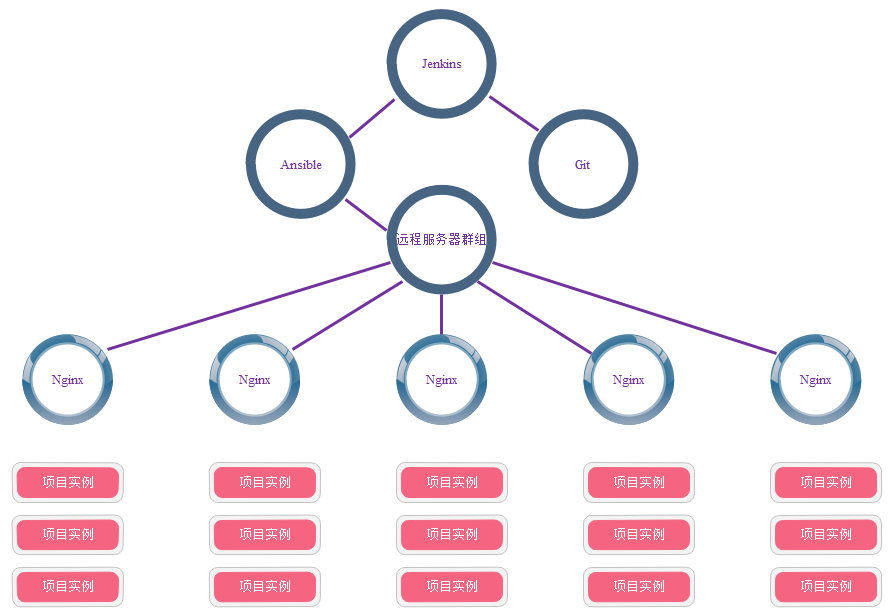


# 其他

## 思路拓展

Jenkins还可用于构建其他编程语言的项目[如PHP、HTML语言等]

PHP、HTML项目我们可用配合Ansible[配置管理工具]来实现代码的推送、回滚时版本的控制及多台同时批量部署可节省人工运维成本大幅提高部署效率



## 结束语

感谢耐心阅读；至此，Jenkins参数化构建部署已分享完成



如有疑问请打开微信扫描上方二维码联系撰写作者



**、**