Jenkins自动打包部署—Android

---20180907

环境准备： Linux+gradle+Jenkins+Windows

工程打包类型： 单一工程打包

说明： 本文档仅用作指导完成Jenkins自动打包的基本操作，如需深入了解可参考Jenkins官方网站的指导说明。

# Start

## 1准备：

惠普Windows电脑）：win10系统

Gradle 4.1

Jenkins 2.140

git 1.8.3.1

java 1.8.0\_101

SDK r24.4.1 另外SDK还需根据开发需要，使用命令分别进行下载对应的tool等工具

## 2安装java环境：

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>下载安装包，下载完成后，解压放置到意向目录下，配置java环境变量，即可使用。（如已有可跳过）

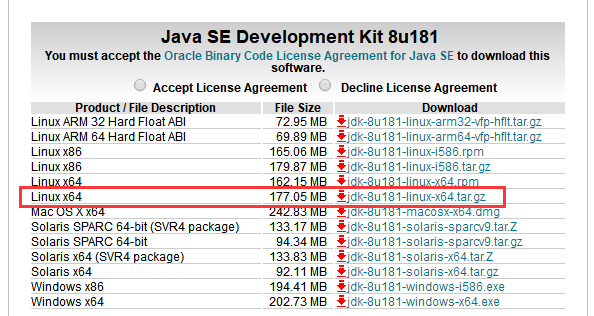
注：环境变量配置，前往/etc/profile,在文件末尾追加以下内容

export JAVA\_HOME=/usr/share/jdk1.6.0\_14

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

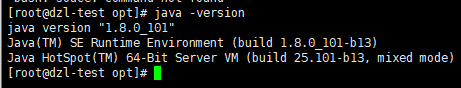
export CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

执行命令使环境变量立即生效source /etc/prof



验证java环境安装完成：

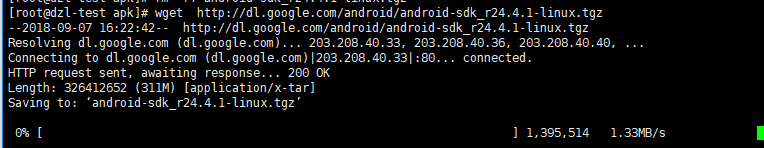
java –version



## 3安装SDK

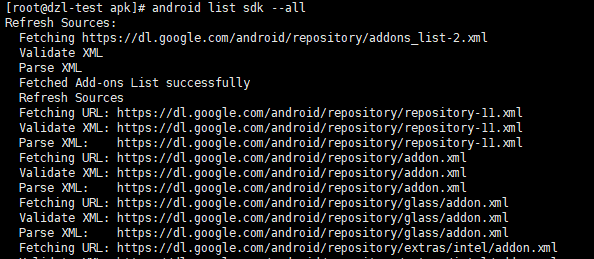
可直接用命令进行下载，和jdk一样，下载了解压既可使用。故cd到意向目录后，执行下面命令即可开始下载。

wget http://dl.google.com/android/android-sdk\_r24.4.1-linux.tgz



下载好后进行解压，后配置环境变量，方法参考jdk的配置。

配置好后，各工具包可用android list sdk –all查询可供下载的工具包，



使用以下命令进行下载安装对应序号下的工具

android update sdk -u --all –filter 工具包序号1，工具包序号2。。。

## 4安装Jenkins：

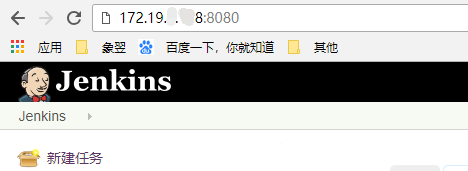
Windows登录Jenkins官网，下载Jenkins.War包，上传到服务器Tomcat根目录中（重启Tomcat即可启动Jenkins服务）（如没有也可放置任意路径中。需用命令进行启动 nohup java -jar /usr/local/disconf/apache-tomcat-9.0.7/webapps/jenkins.war --httpPort=8080 &）

## 5运行Jenkins

如上一章描述，根据不同情况来进行启动Jenkins，一是放在Tomcat下重启Tomcat即可启动，一是放在其他目录下则执行命令启动。如端口存在占用情况，可在启动命中更换端口。

## 6进入Jenkins：

浏览器输入localhost：8080即可进入，或输入IP：端口 进入。



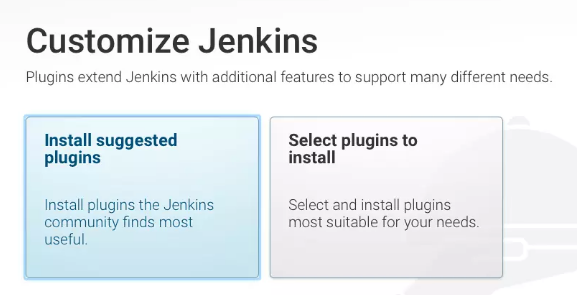
### 6.1登陆Jenkins：

进入到Jenkins首次登陆窗口后，需要填写秘钥登陆（保存在

/root/.jenkins/secrets/initialAdminPassword）如提示无权限，需要对文件/文件夹进行授权



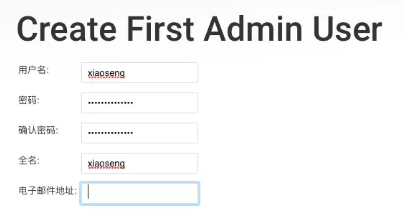
### 6.2安装默认插件



一段时间后刷新页面（安装完成后（均已有打勾或打×））

插件下载安装完后，页面不会自动刷新，需要手动进行刷新页面，可能存在一些插件没有安装成功，会打×，需要手动去进行安装，后面进行讲解（第7章节第1条既7.1）。

### 6.3设置用户账号：（此处采用默认admin账号）



此时点击右下方的跳过，使用admin账号进行

## 7 配置Jenkins

### 7.1插件补充

默认插件存在有失败或我们需要的插件不在默认里，则需要进行补充安装。（系统） 系统管理->管理插件

这里我们需要添加gradle plugin

GIT plugin

SSH Credentials Plugin

Git Changelog Plugin: 获取仓库提交的commit log

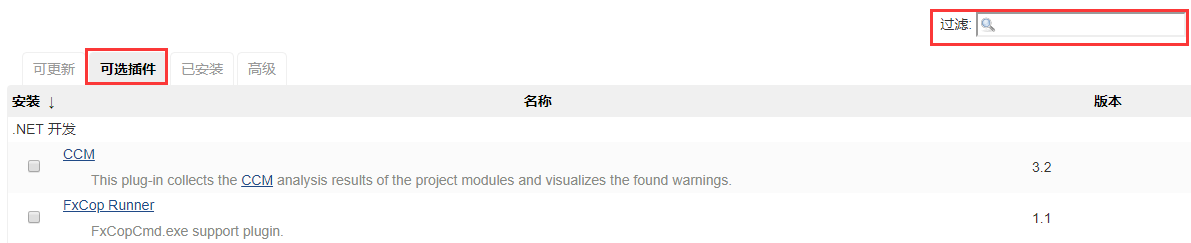
build-name-setter：用于修改Build名称

Post-Build Script Plug-in：在编译完成后通过执行脚本实现额外功能

Gitlab Hook Plugin



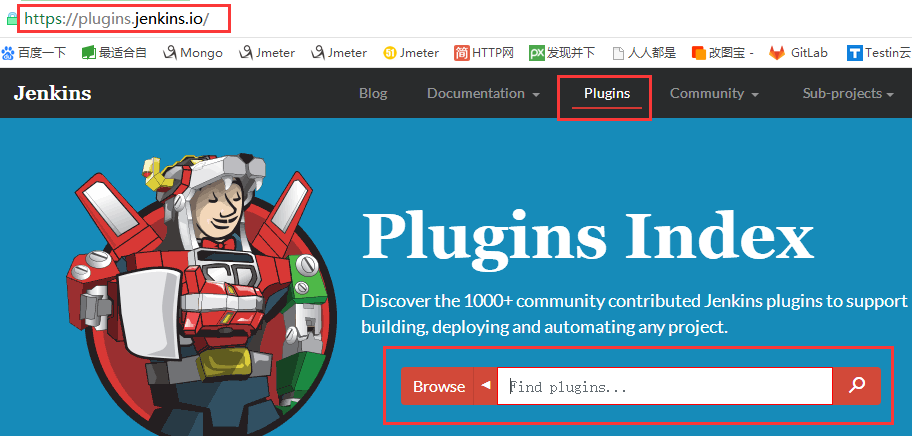
进入到管理插件界面后，点击进入可选插件



然后依次搜索插件进行直接安装，全部补充的插件安装好后，选择重启Jenkins。

### 7.2补充插件安装----上传安装

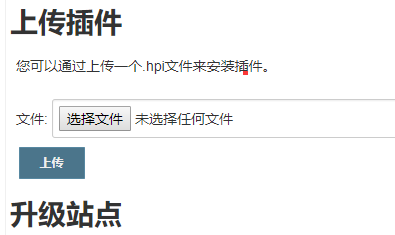
如遇网络问题，或没有自动查找到对应的插件，可到Jenkins.Plug（<https://plugins.jenkins.io/>）进行下载需要的插件。需下载.hpi类型的文件。



下载好插件后，进入 系统管理->管理插件 ，点击高级，拖动页面到“上传插件”

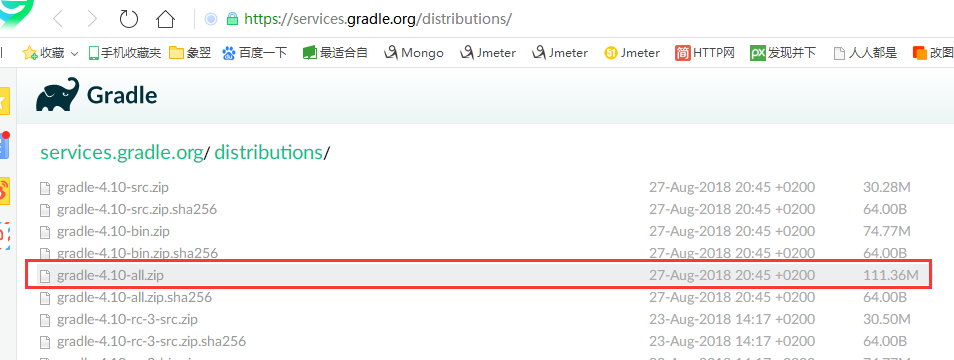


点击选择文件，在弹出的对话框中选择上面下载的插件文件，然后点击上传即可



### 7.3 Linux中的gradle工具

进入Windows，打开浏览器，前往gradle官网（<https://services.gradle.org/distributions/>）进行下载对应的gradle版本（和开发确认），这里使用的是gradle-4.1-all.zip的版本。



下载好后，上传到Linux服务器中，解压后进行配置环境信息，即可使用。

## 8创建工程任务

点击Jenkins首页，左侧显示选择列表，右侧显示现有工程任务，点击左侧的新建任务，进入新建任务对话框，输入任务名称，点选自由风格，点击页面底部“确定”即可。





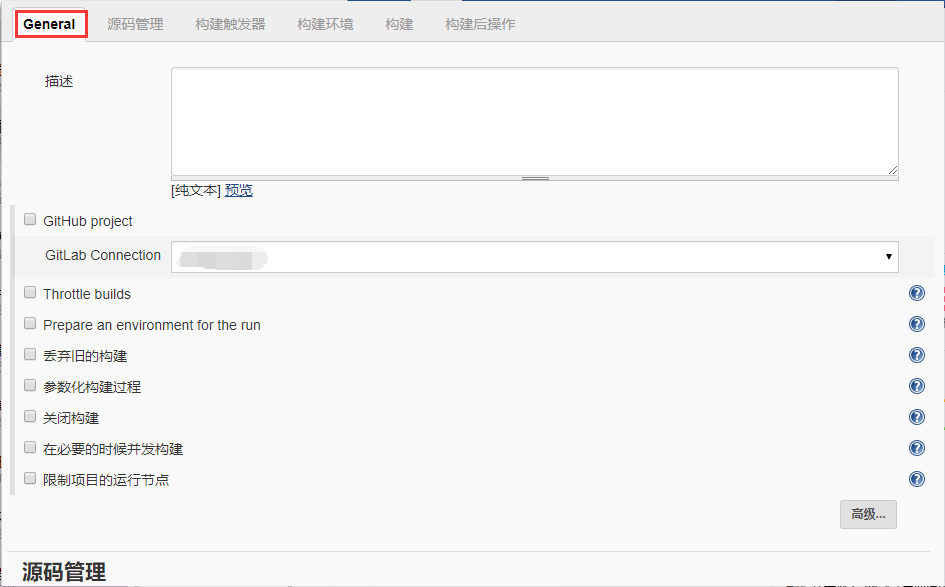
当然，如果我们已经创建了任务，并且其他配置都一样就改个证书什么的，可以下滑看到可以复制一份现有的任务，输入名字就可以选择，选择后点击页面底部“确定”即可。



### 8.1工程配置

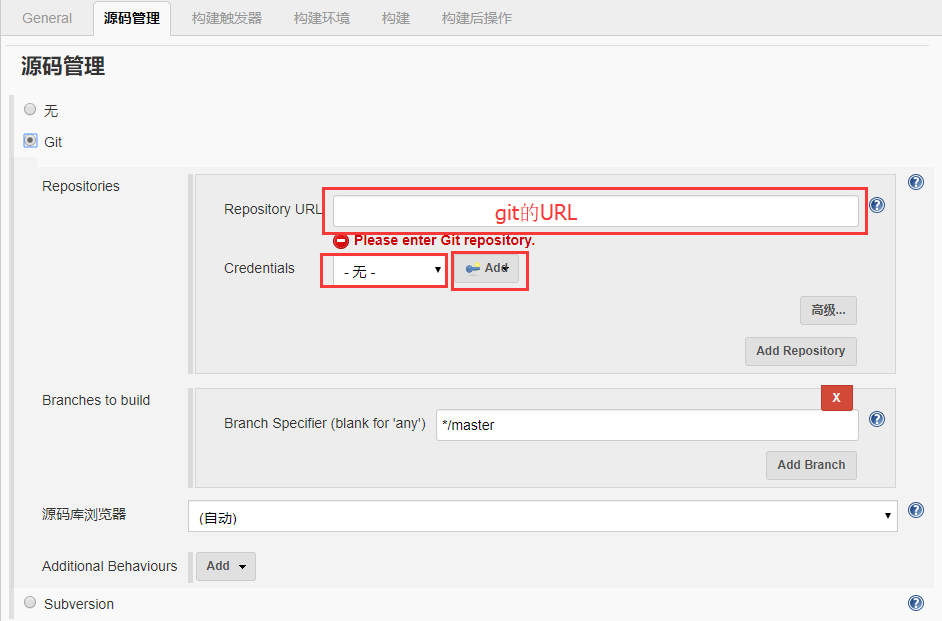
#### 8.1.1 General

上一步点击“确定”后，会直接进入到该工程任务的配置项页面中，在这里，我们可以再描述中输入对该工程任务的一个描述内容。及其他的一些需要的选项。



#### **8.1.2 源码管理**

当然，这个General标签下内容可以都不做选择。直接进入“源码管理”标签下的内容，并进行编辑。我们这里则跳过General，如有需要可根据相关内容进行选择。

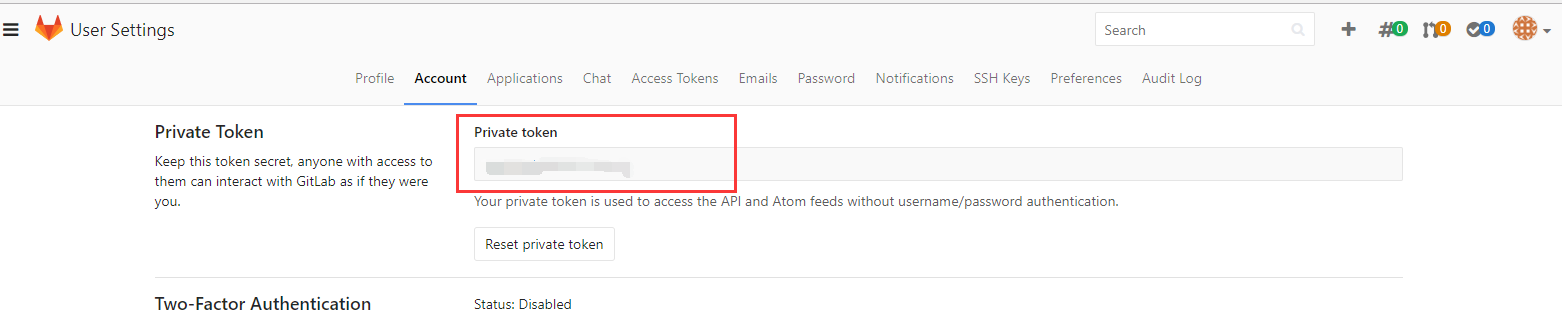


URL：填写gitlab的项目URL地址，此处需要注意，安装Jenkins的方式不同则此处选择的地址则不一样：如采用Jenkins官网下载的Jenkins.war包进行起的Jenkins服务，则此处需要使用ssh的URL地址，则需要进行秘钥对的匹配。如采用的是命令模式进行下载安装启动的服务（wget http://updates.jenkins-ci.org/download/war/2.109/jenkins.war），则可直接使用用户名和密码的形式进行。（究竟是否存在该限制不确定，但实践后，只有像上面说的操作，才得到了最后想要的结果，读者也可自行研看这里是哪里配置不正确。）

此处我们使用的是秘钥对模式，步骤如下：1、登录Gitlab，点击左上角“狐狸头”，后点击右上角的按钮，选择进入setting选项；



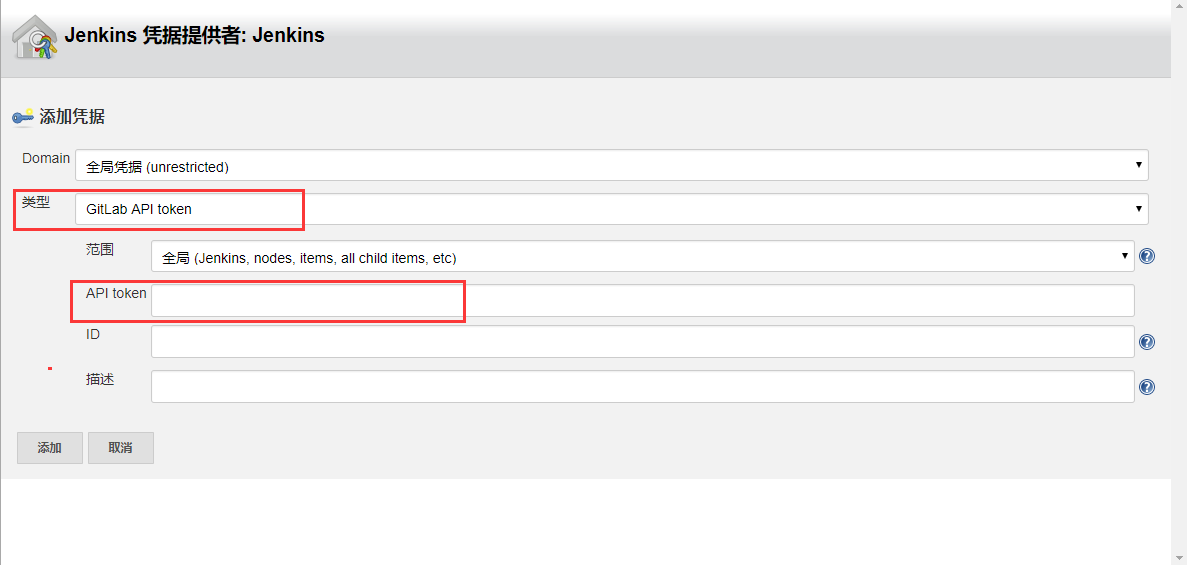
2、点击account并复制token；



3、进入Jenkins，进入系统设置，找到Gitlab选项，并填写信息，点击add进入添加；



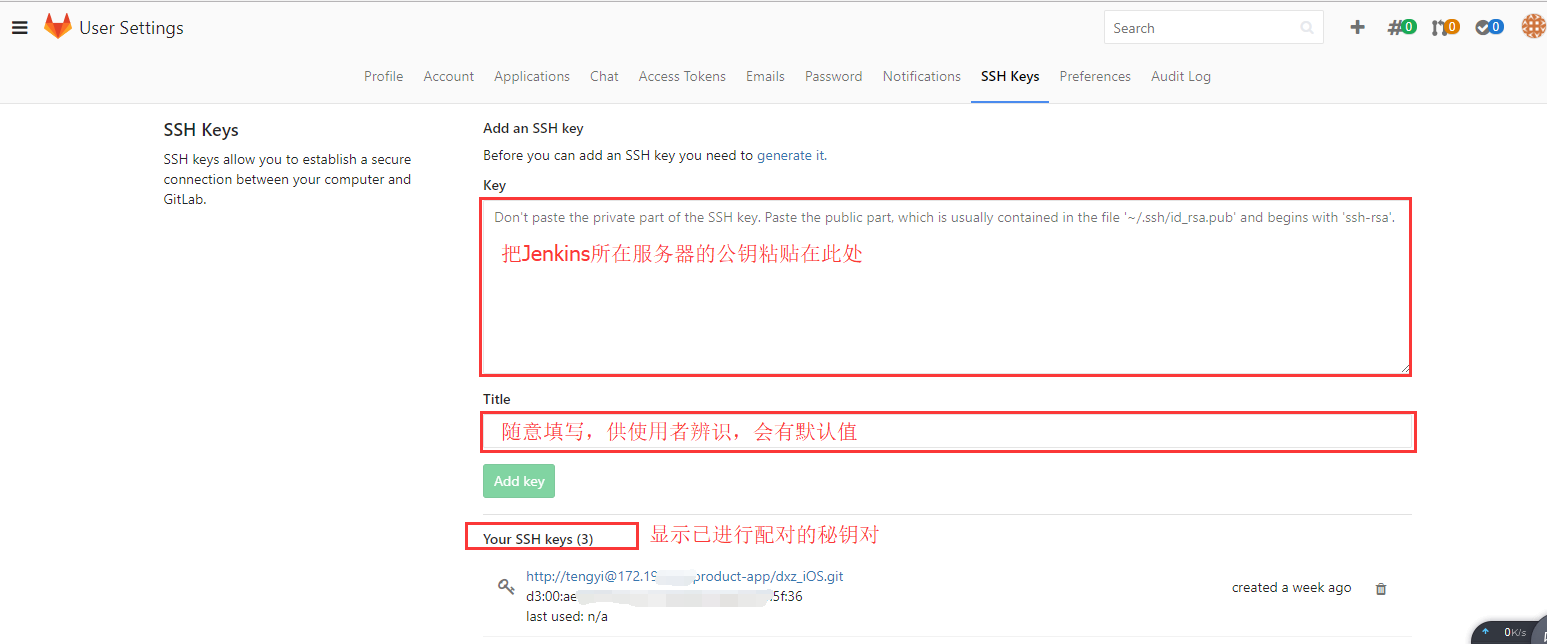
把Gitlab上的token粘贴到api token处，添加完回到上个界面点击test测试下是否可用，可用会有success成功的提示；



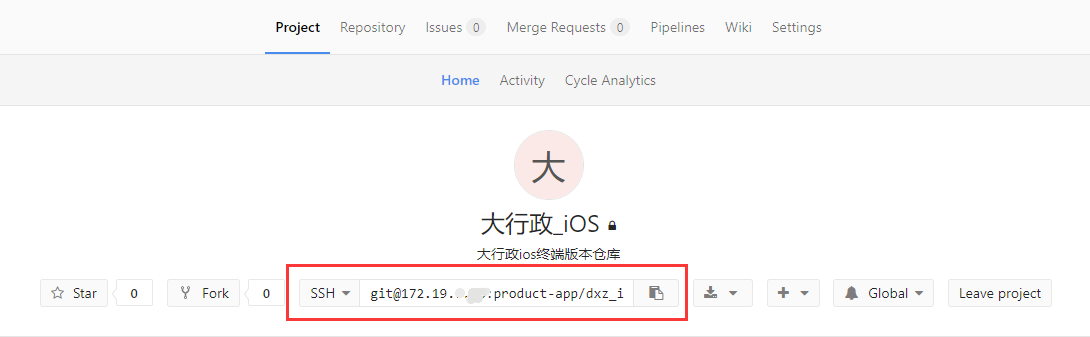
4、填写git连接及连接方式：URL处选择ssh的连接，连接可从Gitlab上的项目中获取



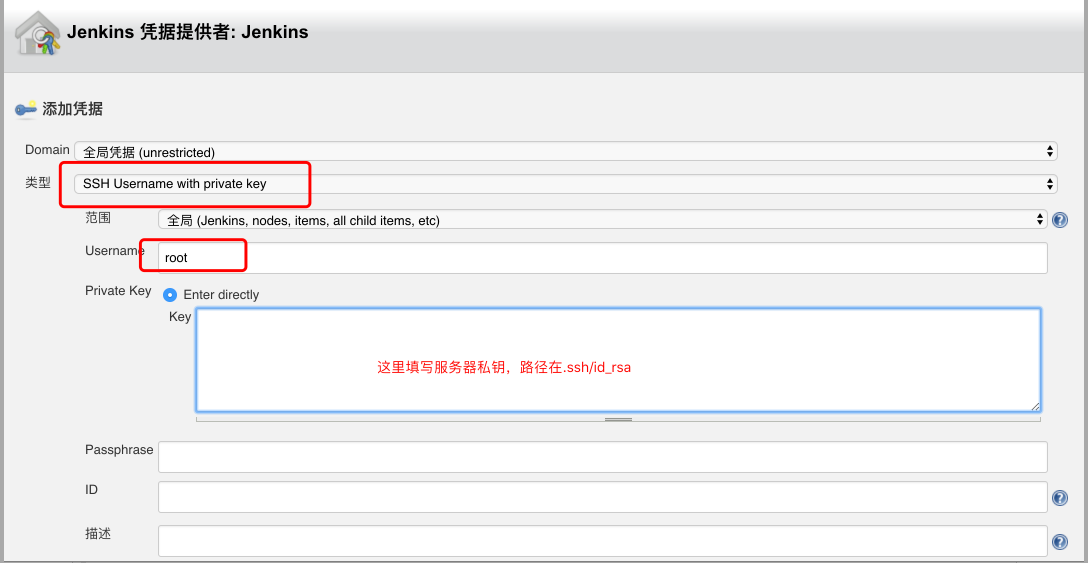
把Jenkins所在服务器的公钥放置到此处（进入Jenkins所在的Linux服务器，生成随机秘钥对（命令openssl rand -hex 12）（如已存在，可进入~/.ssh，下面有公钥id\_rsa.pub和私钥id\_rsa））



复制目标项目的ssh地址



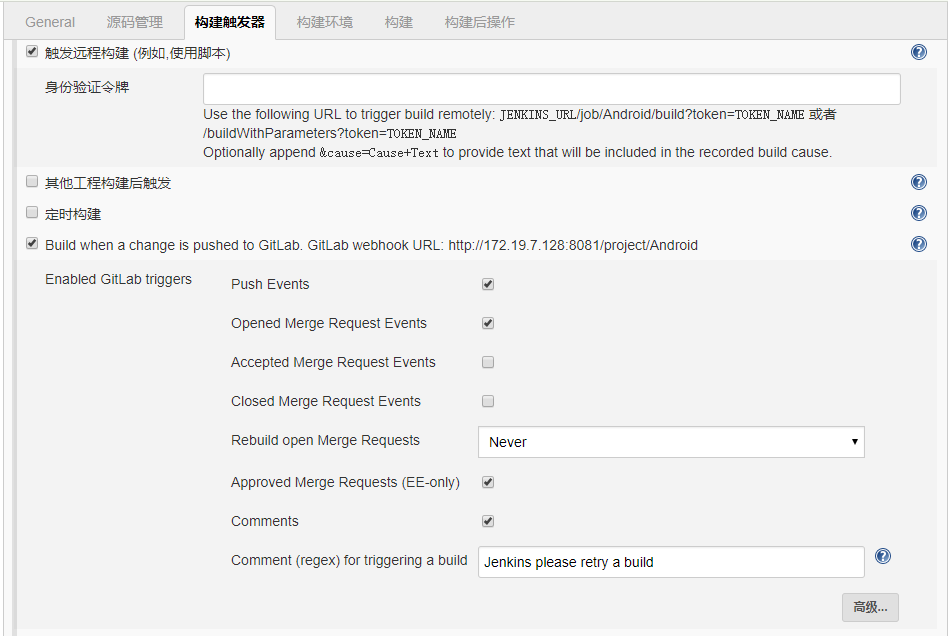
其中在选择连接账户时，需要点击add进行增加ssh的秘钥（如同样的服务器和同样的Gitlab已做过增添，则可跳过）；



返回选择Credentials里刚刚添加的账号，然后选择要进行打包作业的代码的分支，注意前面要有\*/，然后就完成了“源码管理”的内容填写，

#### 8.1.3 构建触发器

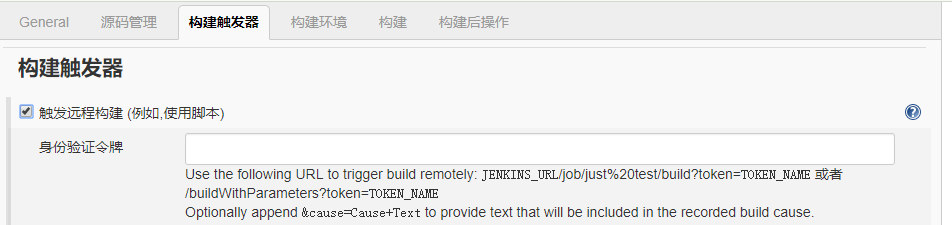
完成“源码管理”的内容填写后，进入“构建触发器”，打包作业，称之为构建，该标签的意思则是要让Jenkins如何在什么条件下开始进行构建打包作业，对常用几项进行区别讲解：





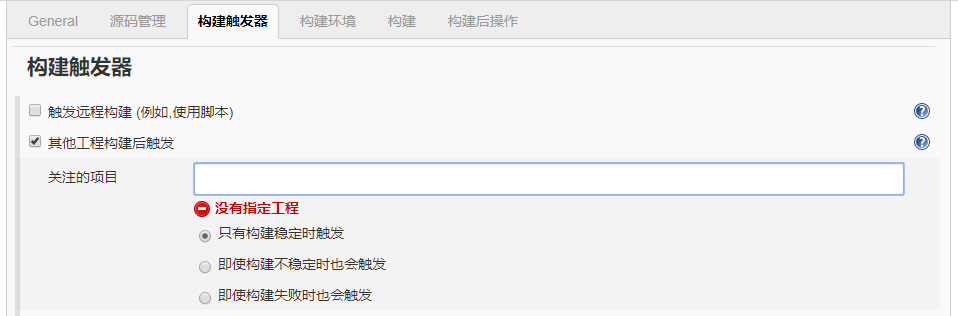
触发远程构建：

勾选此项后，选项下端会有相关提示，通过复制这个URL地址，在别的机器上打开这个URL就相当于给这个jenkins服务器发送了一个构建请求。只不过这个请求是在其他人电脑上发出的。这里重点来看看URL的组成，第一个参数JENKINS\_URL,这里我们写IP地址或者机器hostname，第二个参数TOKEN\_NAME就是你在身份验证令牌文本输入框输入的值。这里我们把令牌设置成123456，然后我就在我机器的另外一个浏览器来模拟远程构建，这个时候我的远程构建地址就是这样的：localhost:8080/job/Test1/build?token=123456，此时在他人的浏览器中输入该远程构建地址，即可开始构建该任务。也可在最后加上“&cause=书写构建原因 ”。



其他工程构建后触发：

选择一个其他构建任务，在这个构建任务完成构建后，当前任务会紧接着进行构建，多用于多部件需要分开启动的、带有联动关系的项目。



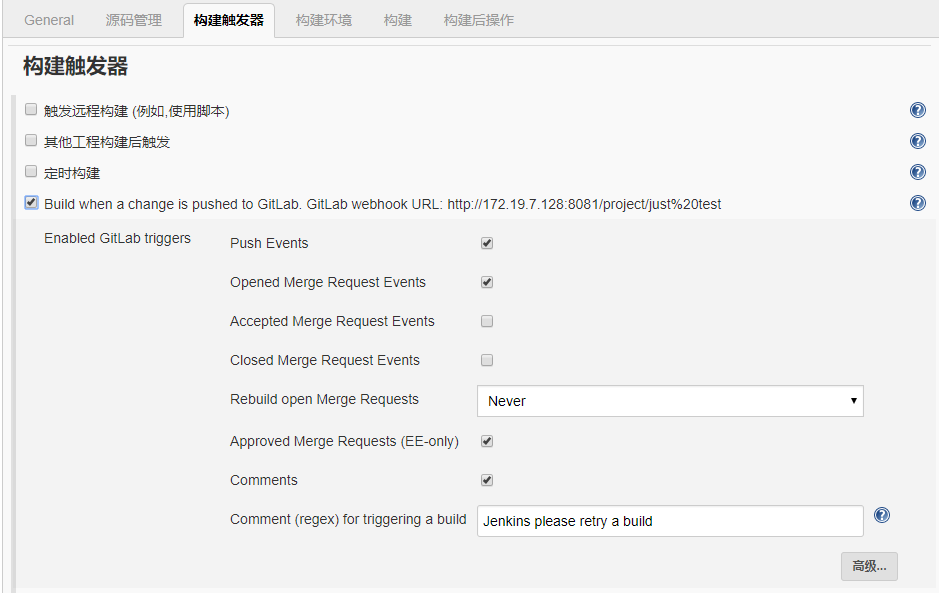
定时构建：

顾名思义，既在输入框内输入指定的时间，到了该时间则自动开始构建，时间的填写有一定的要求规则，详细可网上搜索Jenkins定时构建，既有详细说明，这里不做赘述。



Build when a change is pushed to GitLab.：

既当Gitlab上对应的分支代码上有更改时，Jenkins会立即开始构建该任务。



轮询 SCM：

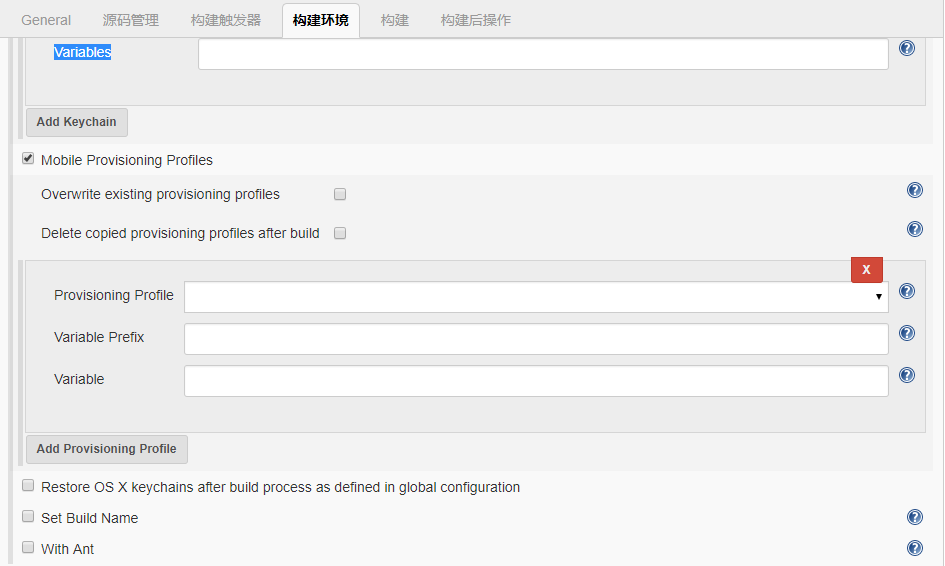
既在指定的时间内，Jenkins会随机选定其中的某个时间进行查询Gitlab对应分支是否有代码改动，如有，则进行构建任务，过了时间段后，再有代码修改，则不会触发构建。



在这里，我们使用的是“持续集成”自动打包部署的策略，因此不做轻量级gitlab的webhooksd 配置了。

#### 8.1.4 构建环境

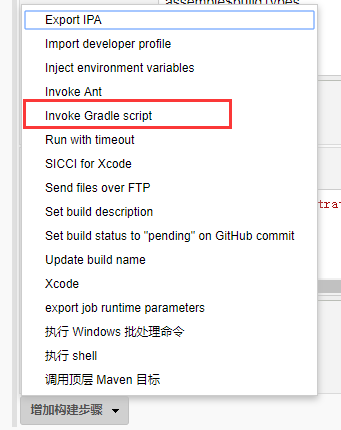
进行编辑构建任务所处的环境，这个需要看项目的需要，一般不需要进行选择，这里不做选择，如是做iOS的项目自动打包部署作业，则需要进行选择xcode的选择。



#### 8.1.5 构建

此处进行选择搭建构建的方式、构建步骤。

构建的方式多种多样，组合方式随意，因此此处不做赘述，仅对此次任务使用的进行说明。



这里选择Invoke Gradle script

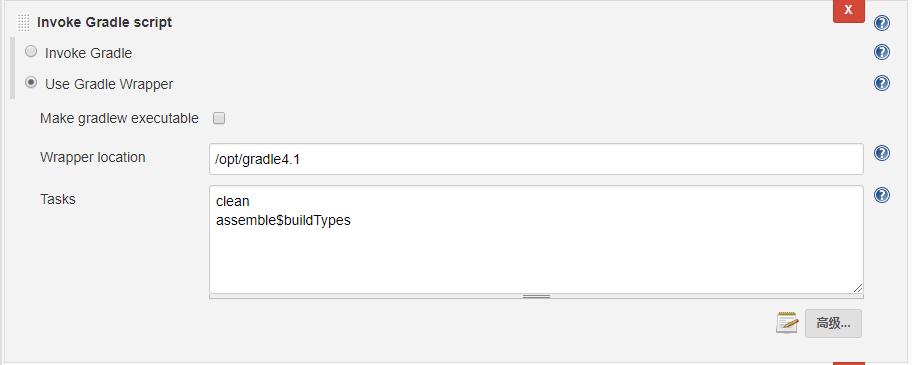
后进行参数化打包，该参数需在General中进行设立参数化构建过程，选择“选项参数”/“文本参数”然后输入要进行参数化的参数名即可引用。



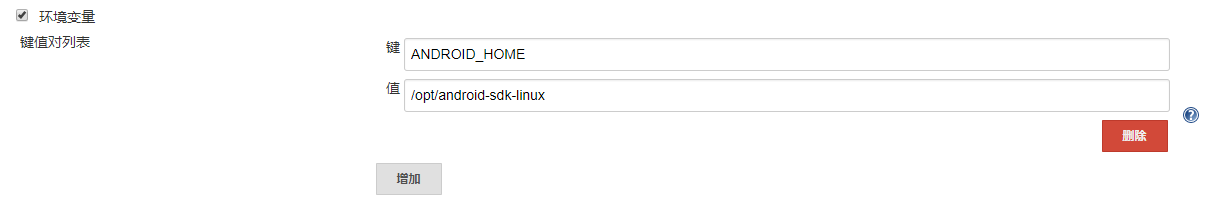
Invoke Gradle

选择所需的gradle版本，然后在打包脚本中输入欲进行的操作，这里先做就构建的清除clean，然后进行参数化打包，输出路径可在代码中指定，如无则默认在Jenkins的build文件夹里，

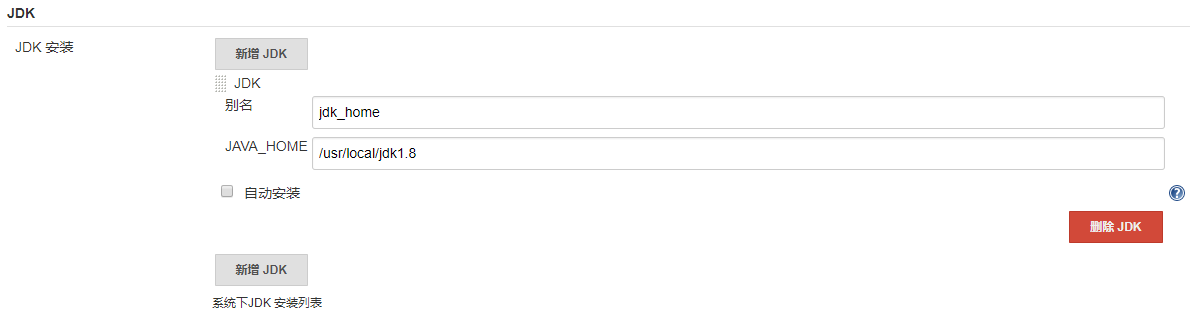
注意：填写好Use Gradle Wrapper记得勾选回Invoke Gradle，这样才能正常工作

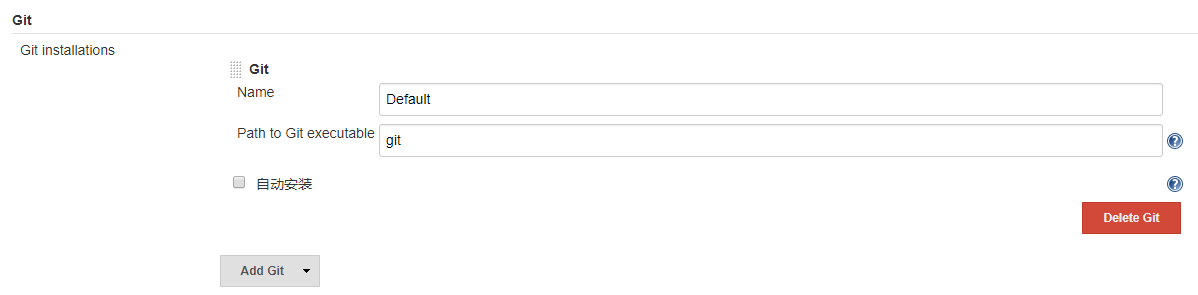


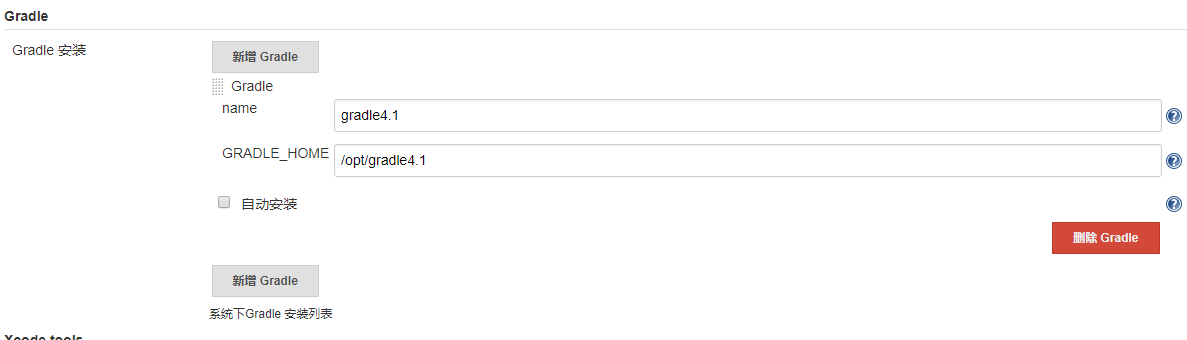
其中，选择gradle的版本时，需要先在Jenkins的系统管理—>系统设置，进行环境变量的配置：这里我们是Android类的打包，需要配置SDK的环境变量



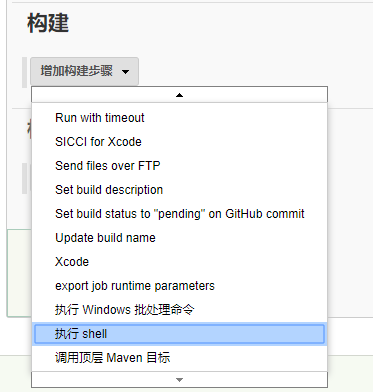
然后进行配置全局工具的配置：系统管理—>全局工具配置，根据需要进行配置，（注意安装了jre也会显示jdk已安装，但必须安装jdk的包，所以需要卸载jre后进行安装jdk的包）







同时，通过shell脚本的调用，来进行分支的合并操作，以及打包好后的上传蒲公英的作业。



上传蒲公英的命令为

curl -F "file=@/要上传的文件的路径" -F "uKey=蒲公英上用户秘钥" -F "\_api\_key=蒲公英上应用的接口秘钥" https://www.pgyer.com/apiv1/app/upload

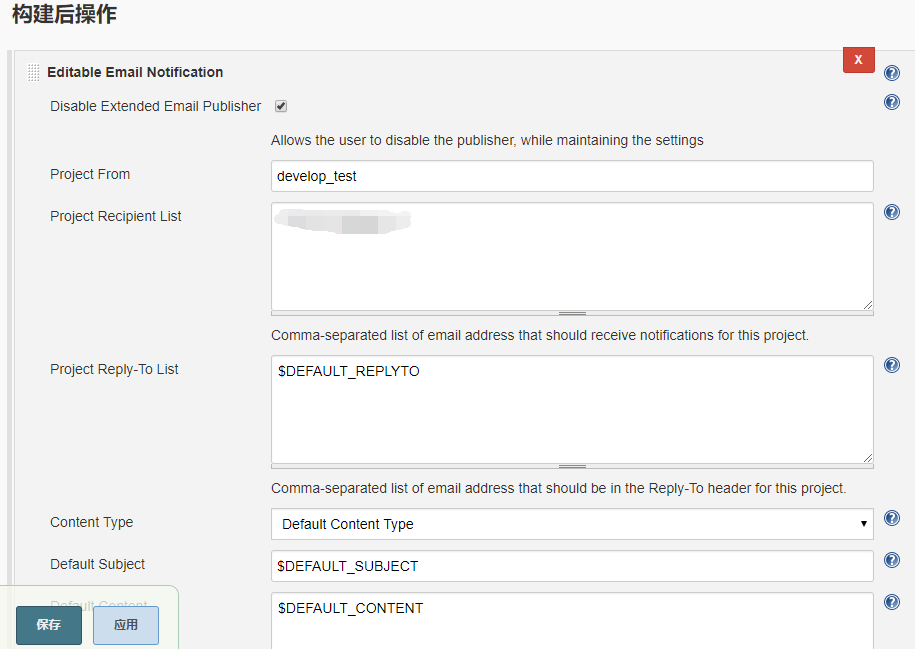
#### 8.1.6构建后操作

构建完成后进行的一些操作，可不选择，这里我选用的**Editable Email Notification**

**勾选**Disable Extended Email Publisher后，构建成功的则不发送邮件提醒。

Project From ：填写项目名称，可随意填写，用于自己识别是哪个构建任务就可以

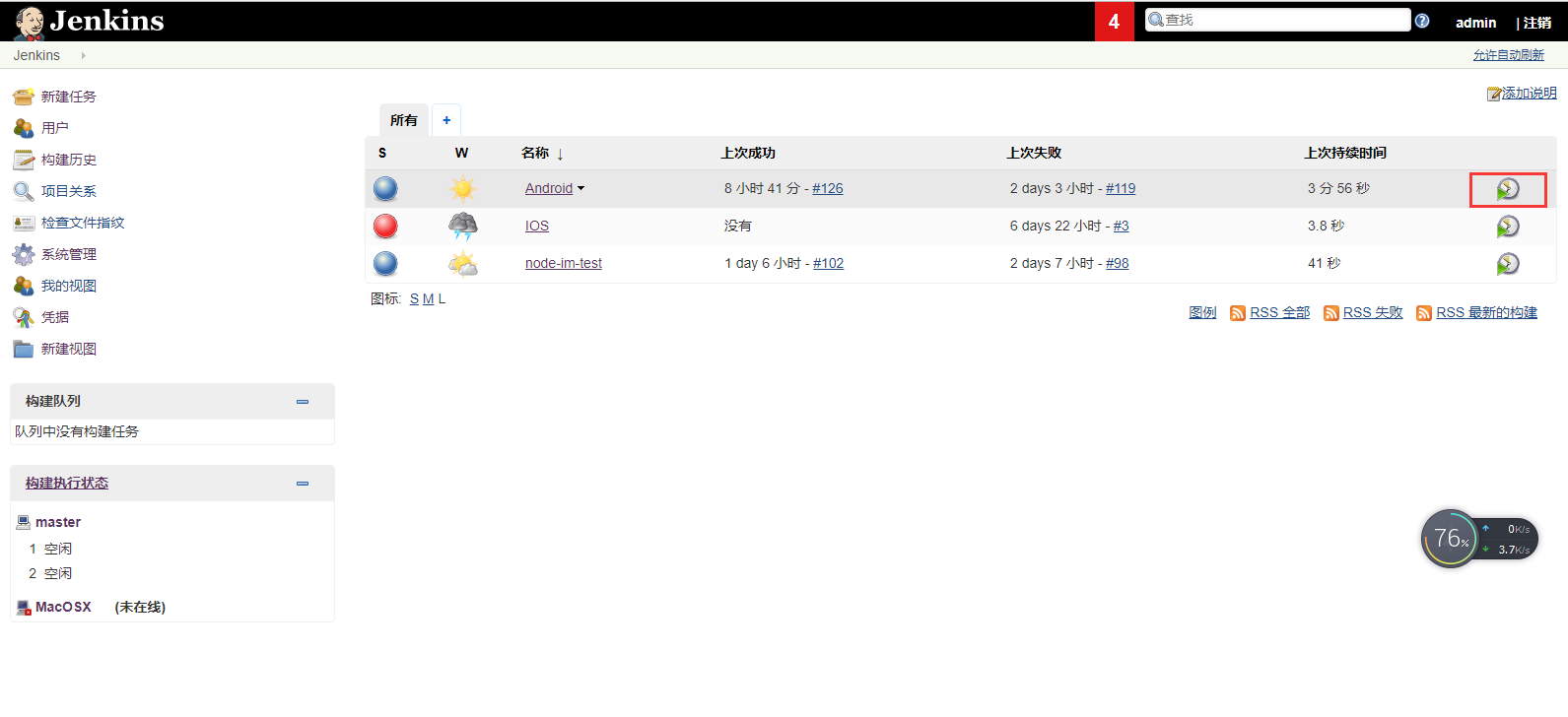
Project Recipient List：填写接收提醒的地址，这里使用一个QQ邮箱做接收



完成后，即可点击保存，则完成了全部的工程任务的配置。

## 9 发起构建

配置好任务后，可进行手动操作进入构建状态，有三个地方可触发，分别为：首页的工程后面的三角符号；右击工程名右边的下拉三角，点击立即构建；点击工程名，进入工程，选择右侧目录的“立即构建”。





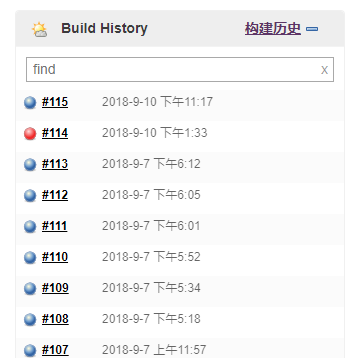


**9.1构建历史查看**

首先是在首页中可以快速查看上一次构建信息：



其次是点击项目名称进入项目后，在左下部分区域将显示历史记录



点击单次构建历史，可进行查看本次构建的详细信息系统：



同时点击上面页面的左侧功能按钮，进入对应的功能

其中：

“变更记录”可进行查看本次构建详细的变更信息

“控制台输出”可查看本次构建产生的日志

“轮询日志”可查看本次构建进行轮询的信息

## 10 补充终端的操作

### 10.1Jenkins工作路径

Jenkins会在终端中生成一个工作空间，用于保存项目任务的的构建，默认路径为/root/.jenkins/workspace/构建任务的名称

打成的包默认保存在/root/.jenkins/workspace/构建任务的名称/build

里的应用名命名的文件夹里（如有做输出路径的限制，则在限制的路径中）。

### 10.2 Git for linux

#### 10.2.1安装

（1）首先先更新系统

sudo yum update

（2）安装依赖的包

sudo yum install curl-devel expat-devel gettext-devel openssl-devel zlib-devel gcc perl-ExtUtils-MakeMaker

（3）下载git源码并解压缩

$ wget https://github.com/git/git/archive/v2.3.0.zip

$ unzip v2.3.0.zip

$ cd git-2.3.0

（4）编译安装

将其安装在“/usr/local/git”（意向）目录下，执行以下命令：

make prefix=/usr/local/git all

sudo make prefix=/usr/local/git install

（5）此时你如果使用git --version 查看git版本的话，发现git仍然是1.7.1版本。这是因为它默认使用了"/usr/bin"下的git。

你可以用下面的命令查看git所在的路径：

$ whereis git

git: /usr/bin/git /usr/local/git /usr/share/man/man1/git.1.gz

（6）我们要把编译安装的git路径放到环境变量里，让它替换"/usr/bin"下的git。为此我们可以修改“/etc/profile”文件（或者/etc/bashrc文件）。

sudo vim /etc/profile

然后在文件的最后一行，添加下面的内容，然后保存退出。

export PATH=/usr/local/git/bin:$PATH

（7）使用source命令应用修改。

source /etc/profile

（8）然后再次使用git --version 查看git版本，发现输出2.3.0，表明安装成功。

#### 10.2.2设置Git

（1）设置用户名和email。

安装好后需要做初始化（your\_name,your\_email,gmail均和自己的匹配即可）

git config --global user.name "your\_name"

git config --global user.email "your\_email@gmail.com"

此时，Home目录下会新建一个.gitconfig文件

#### 10.2.3为GitHub账号添加SSH Keys

以公钥认证方式访问SSH协议的Git服务器时无需输入口令，而且更安全。（访问HTTP协议的Git服务器时，比如提交修改，每次都需要输入口令。）

（1）创建SSH key

$ ssh-keygen -t rsa -C "youremail@163.com"

系统会提示key的保存位置（一般是~/.ssh目录）和指定口令，保持默认，连续三次回车即可。

（2）Copy SSH Key

然后用vim打开该文件，id\_rsa.pub文件内的内容，粘帖到github帐号管理的添加SSH key界面中。

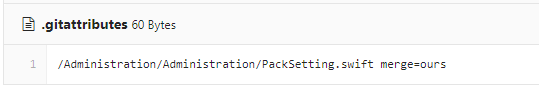
vim ~/.ssh/id\_rsa.pub

（3）添加到GitHub

登录github-> Accounting settings图标-> SSH key-> Add SSH key-> 填写SSH key的名称（可以起一个自己容易区分的），然后将拷贝的~/.ssh/id\_rsa.pub文件内容粘帖-> add key”按钮添加。

## 11关于忽略文件

在进行代码同步时，有一些文件的内容，存在一定的个性化设置或本地化生产的内容，这时候我们需要把我们不需要或者说不能要的文件做忽略处理，这里采用的是在工程根目录下生成忽略文件 **.gitattributes** 来进行忽略不需要同步的文件，其中路径建议采用绝对路径来进行编写，格式为：路径 merge=ours。



添加了文件后，需在工程根目录下执行git config --global merge.ours.driver true以使上述文件生效。

## 12更多关于

Gitlab的操作：

Linux Git操作：

Shell脚本的编写：

Gradle的常用命令：

Git常用命令：

参考文献：

http://debugtalk.com/post/iOS-Android-Packing-with-Jenkins/