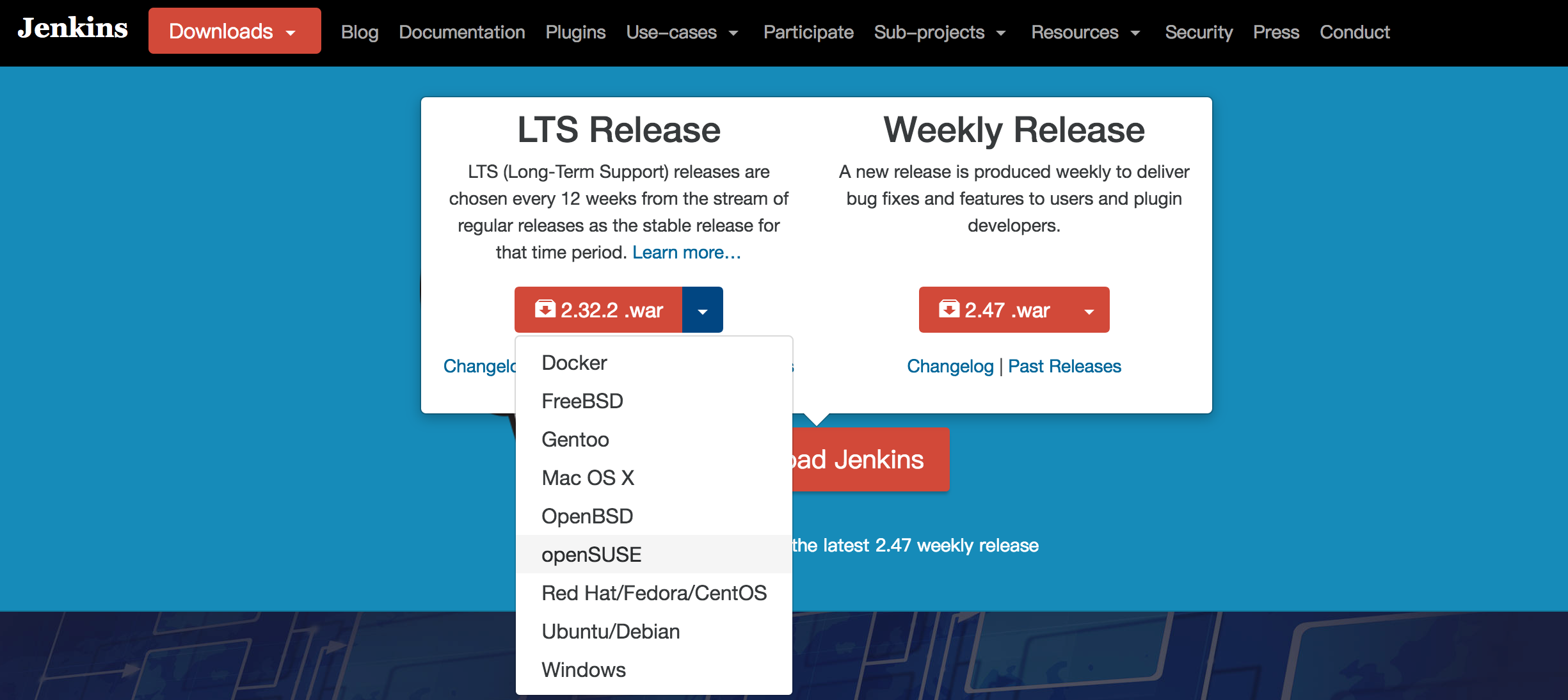
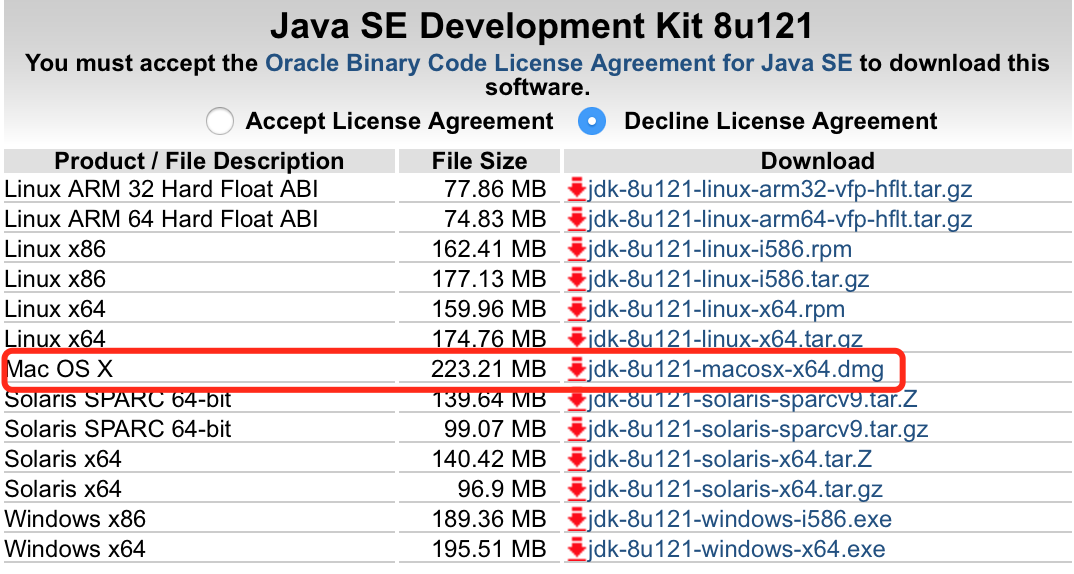
**Jenkins for iOS with fastlane and svn**

一、安装

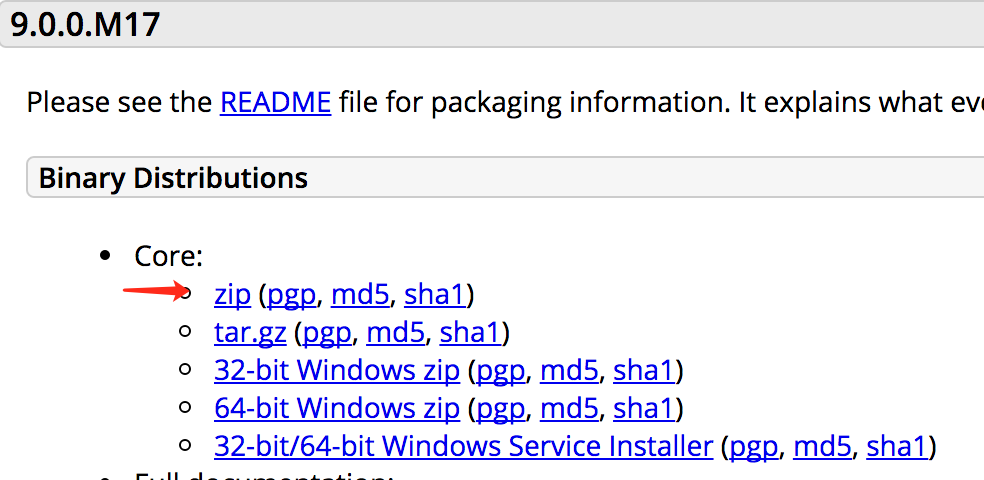


Jenkins的安装需要JDK环境，JDK安装方法自行参考网络。Jenkins的安装有两种方式，一种是java包安装，另一种是pkg可执行程序（两种安装后的配置一样，pkg安装会在电脑上多出一个用户）。本文采用war包 + tomcat安装方法。

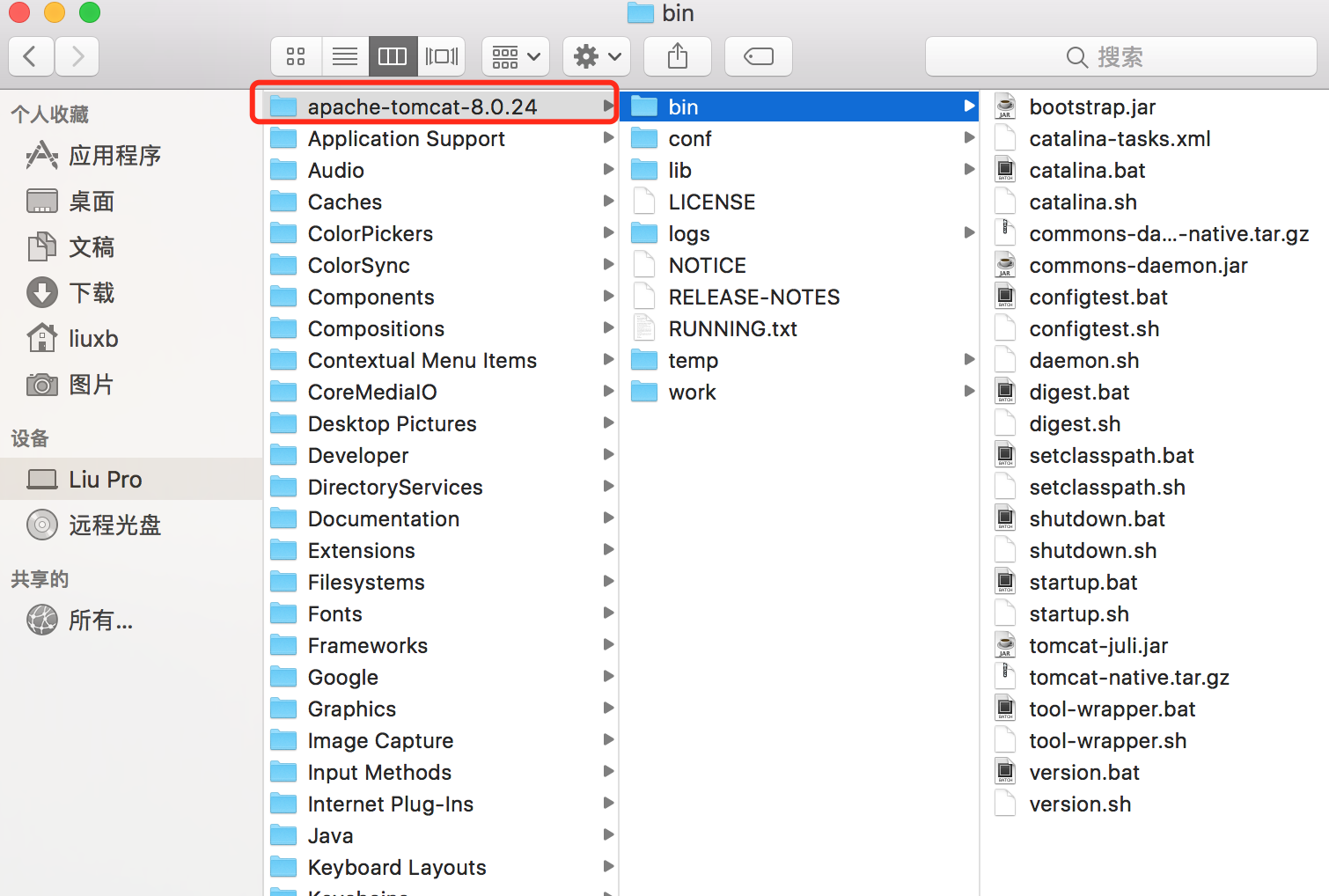
**第一步：安装Java环境，这里不再赘述。请自行下载安装**[**Java**](http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html)**，如下图所示**



**第二步：安装Tomcat，打开Tomcat官网，如下图所示。**



**1、将下载的zip包解压（可以重命名），把解压后的文件夹放到 /Library下。**



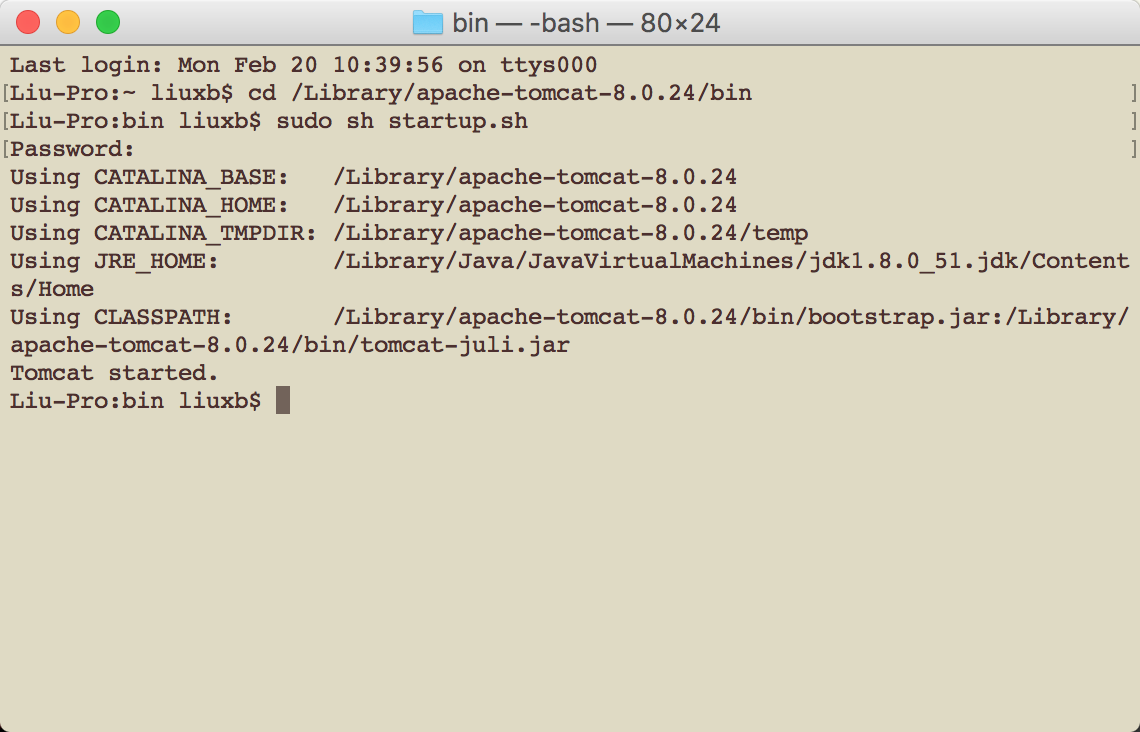
**2、在终端启动Tomcat服务器，这里首先cd到Tomcat的bin目录：**

sudo chmod 755 \*.sh

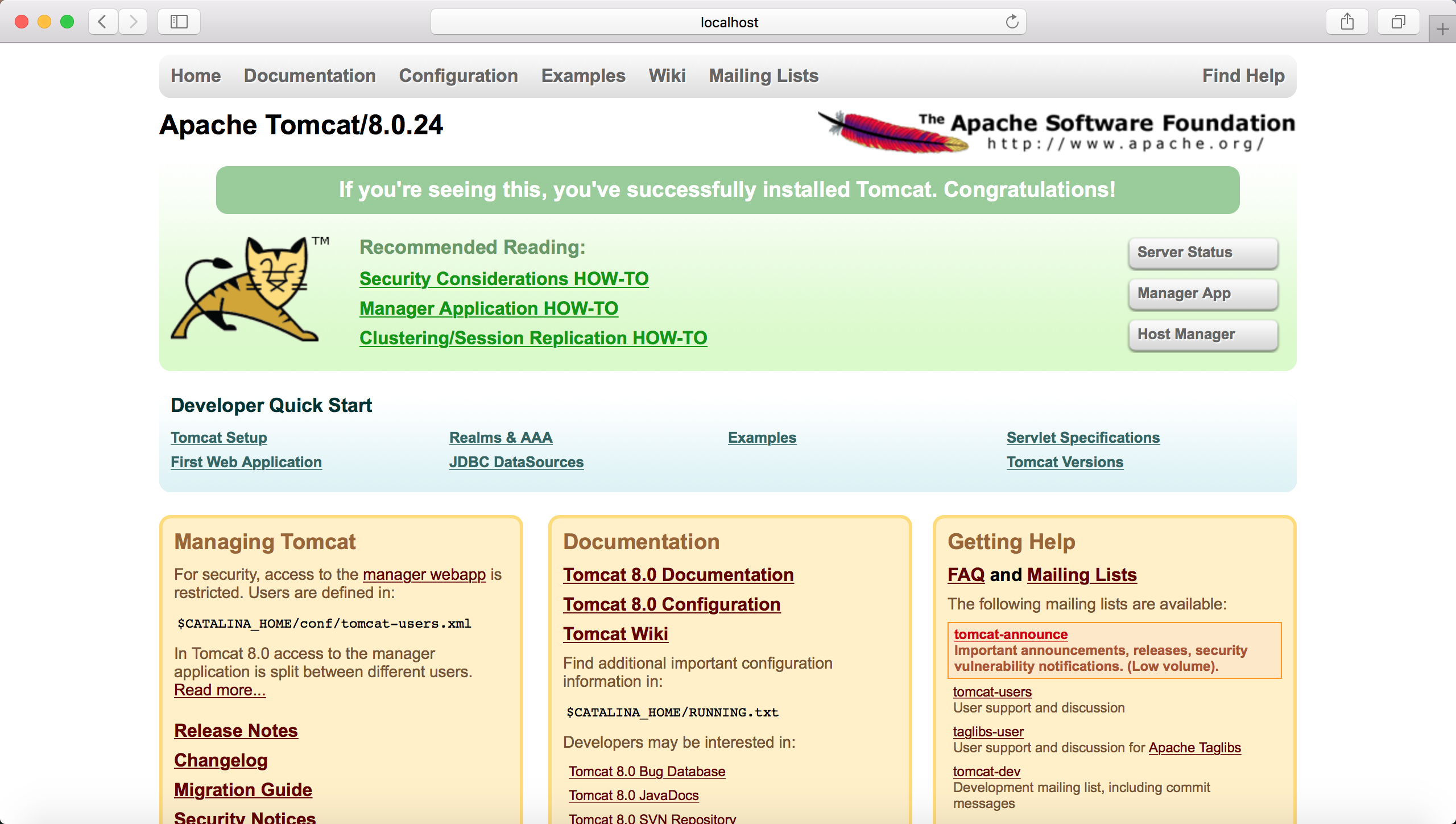
**按回车键之后会提示输入密码，请输入管理员密码。之后输入并回车：**

sudo sh startup.sh

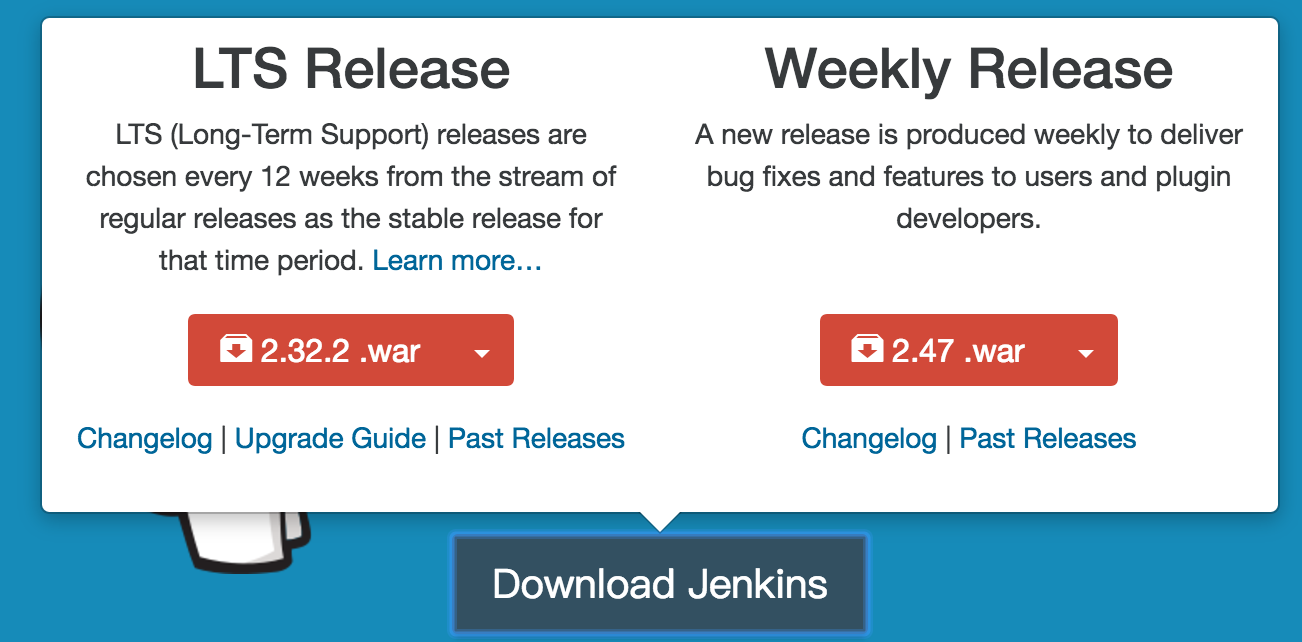
**执行完startup.sh的结果如下：**



然后在浏览器里输入：localhost:8080就OK了。如下图所示：

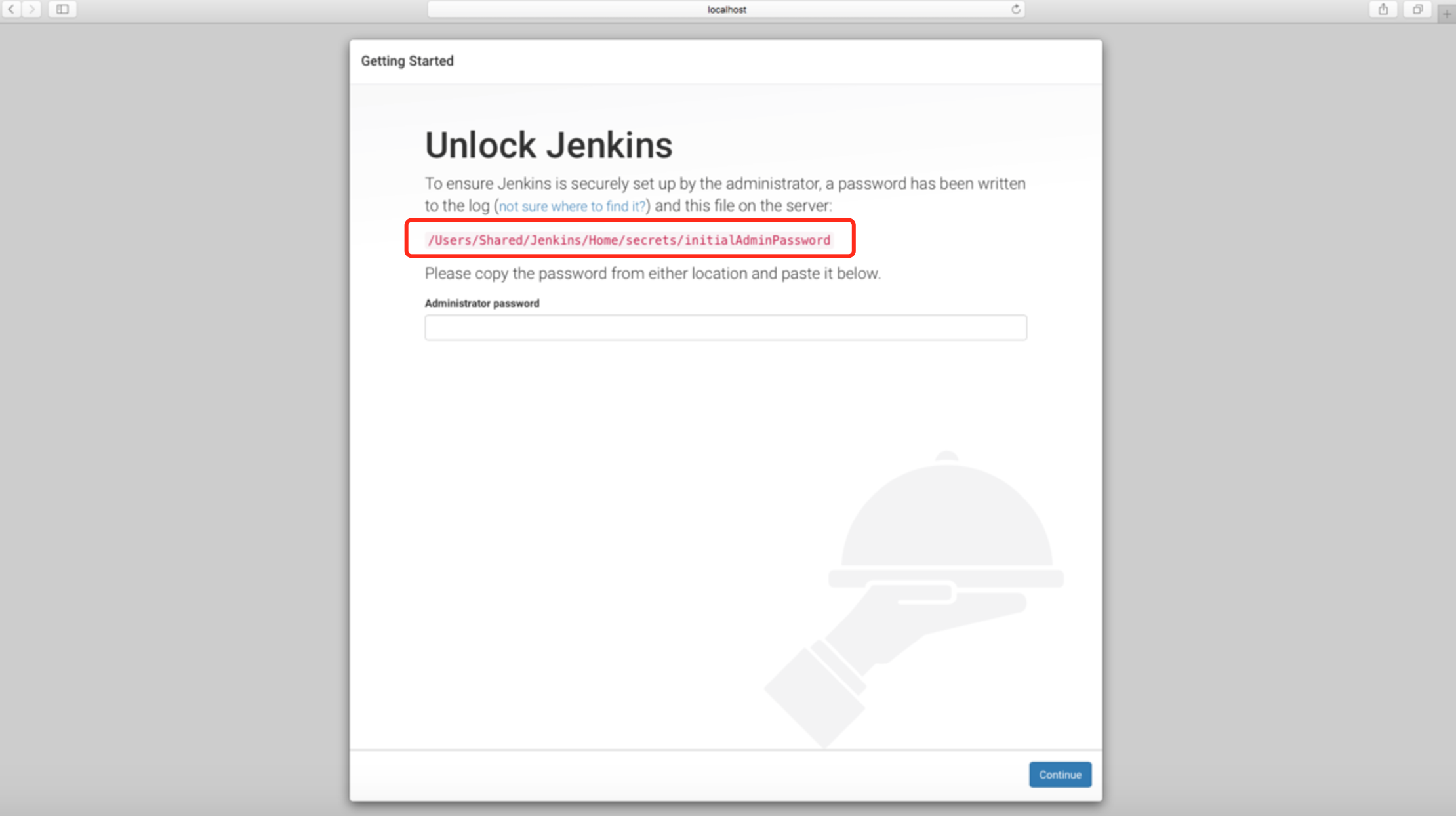


**第二步：安装Jenkins，打开**[**Jenkins官网**](https://jenkins.io/index.html)**，如下图所示。下载Jenkins.war包。**



**在tomcat的安装目录下，找到webapps，然后将下载的war包放到该文件夹下即可。**

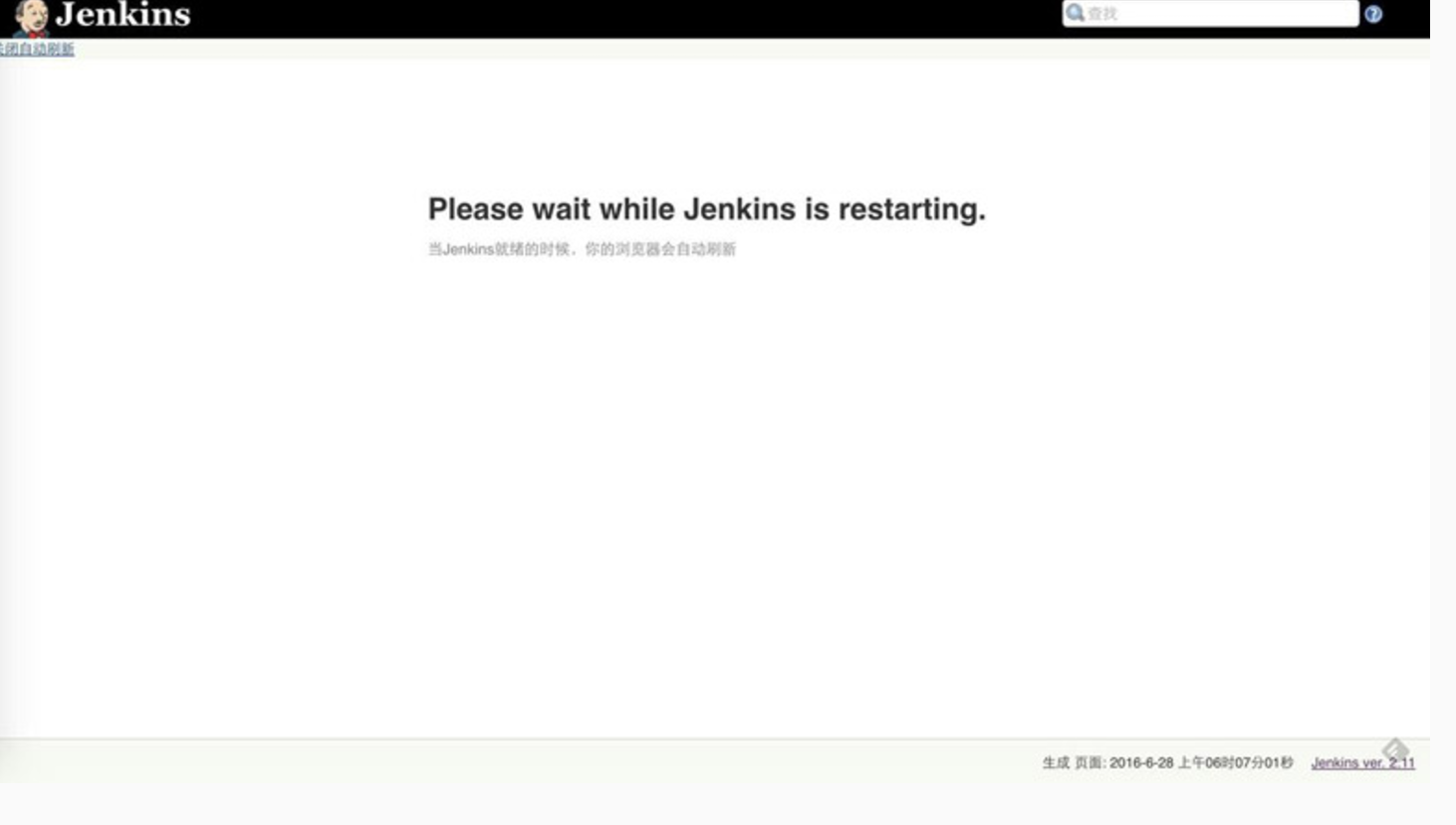
**在浏览器里输入：localhost:8080/Jenkins/，几分钟后即可以看到如下界面：**

****

**根据自己安装的红色提示，前往该文件打开，找到初始密码。（pkg安装模式，会提示没有打开文件夹权限，则需要你手动获取对应的读写权限）。接下来则是傻瓜式操作。如下图所示：**

****

****

****

****

**设置用户名密码邮件等，最后Save and Finish，OK，到此Jenkins初始化安装完成。**

二、安装插件

**在“系统管理->管理插件->可选插件”中，选择下载必要的插件。**

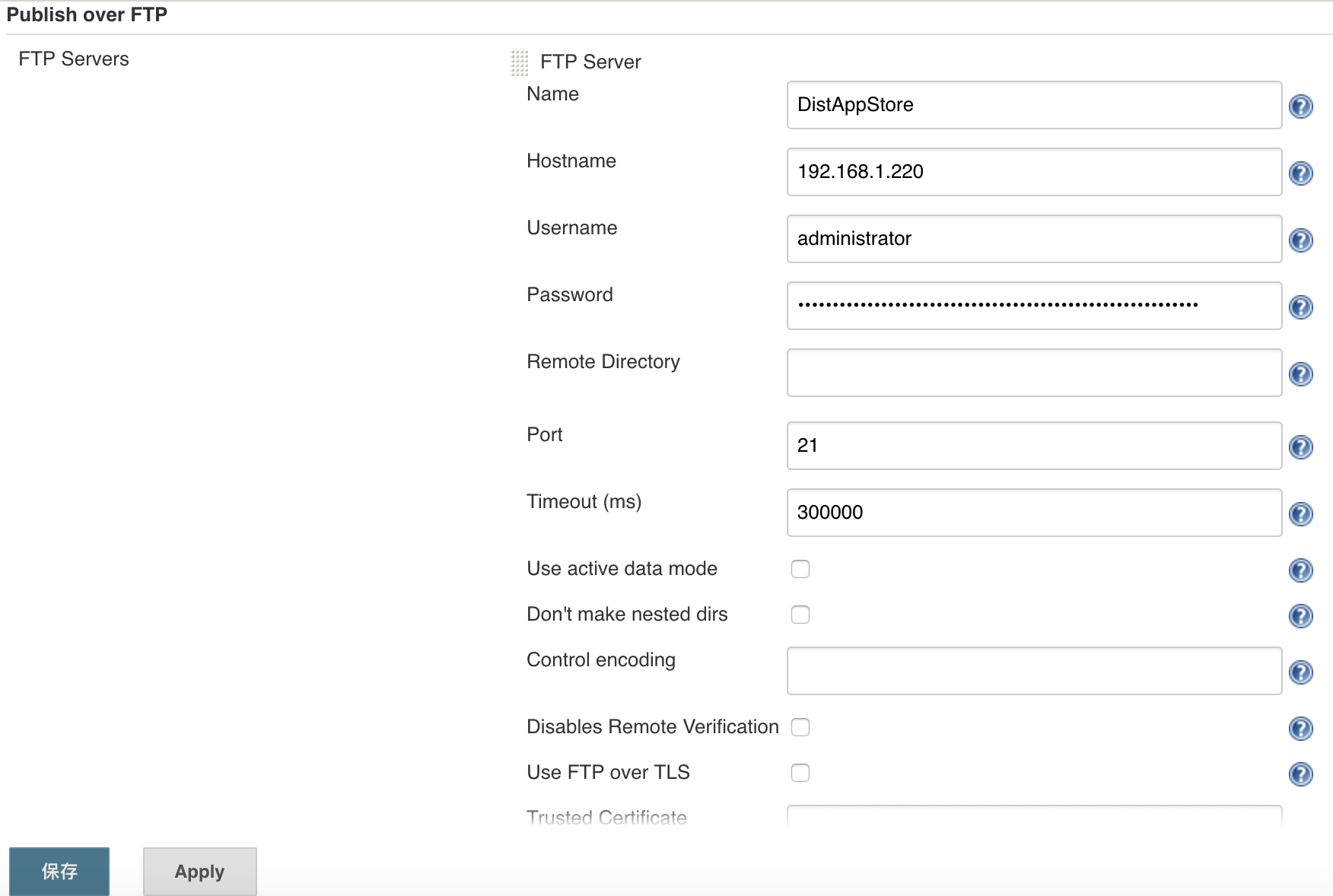
1. **Publish Over FTP Plugin**
2. **Email Extension Plugin**

三、系统配置

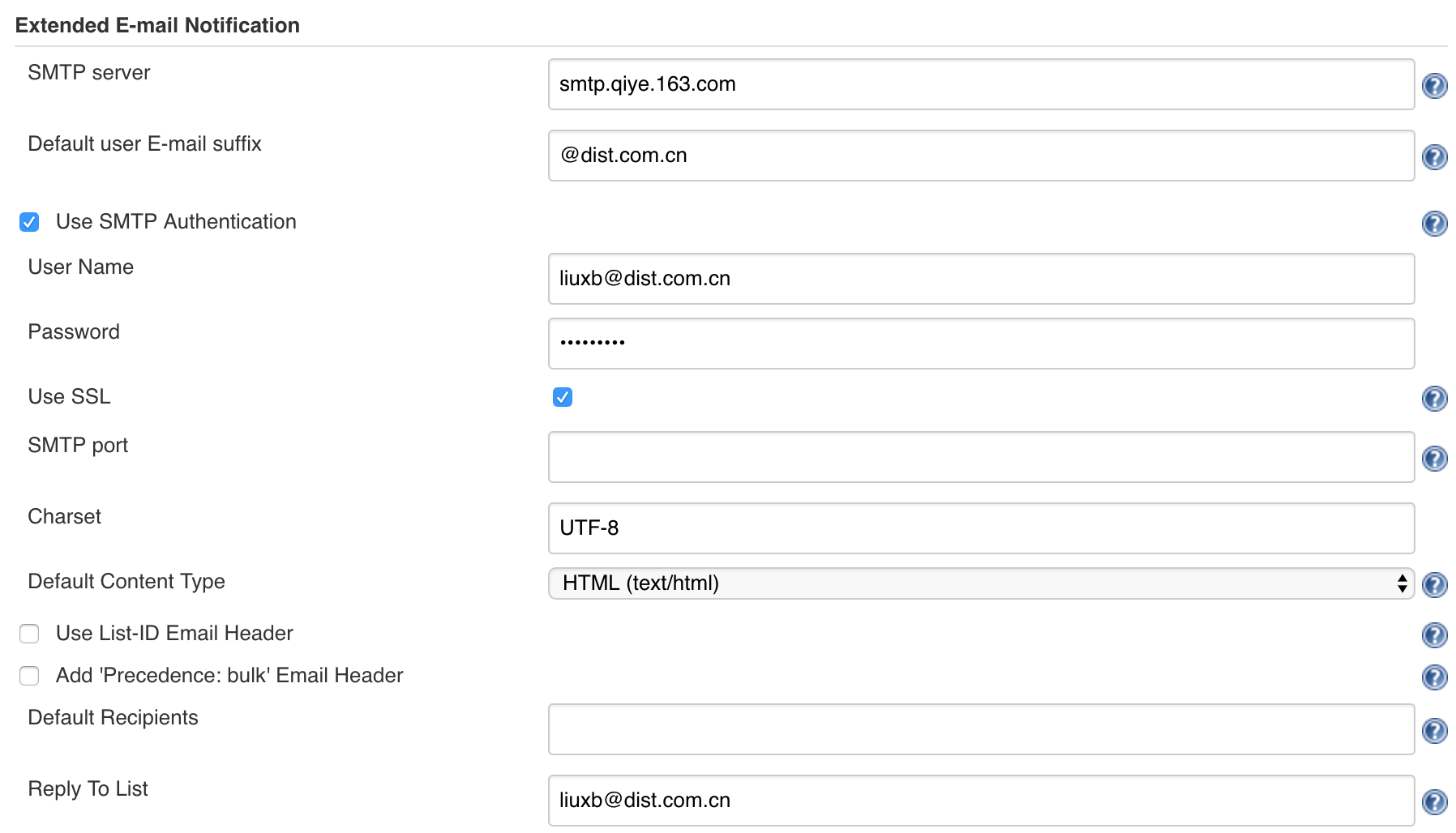
**1、在“系统管理->系统设置”中找到“Jenkins Location”配置，配置如下图：**



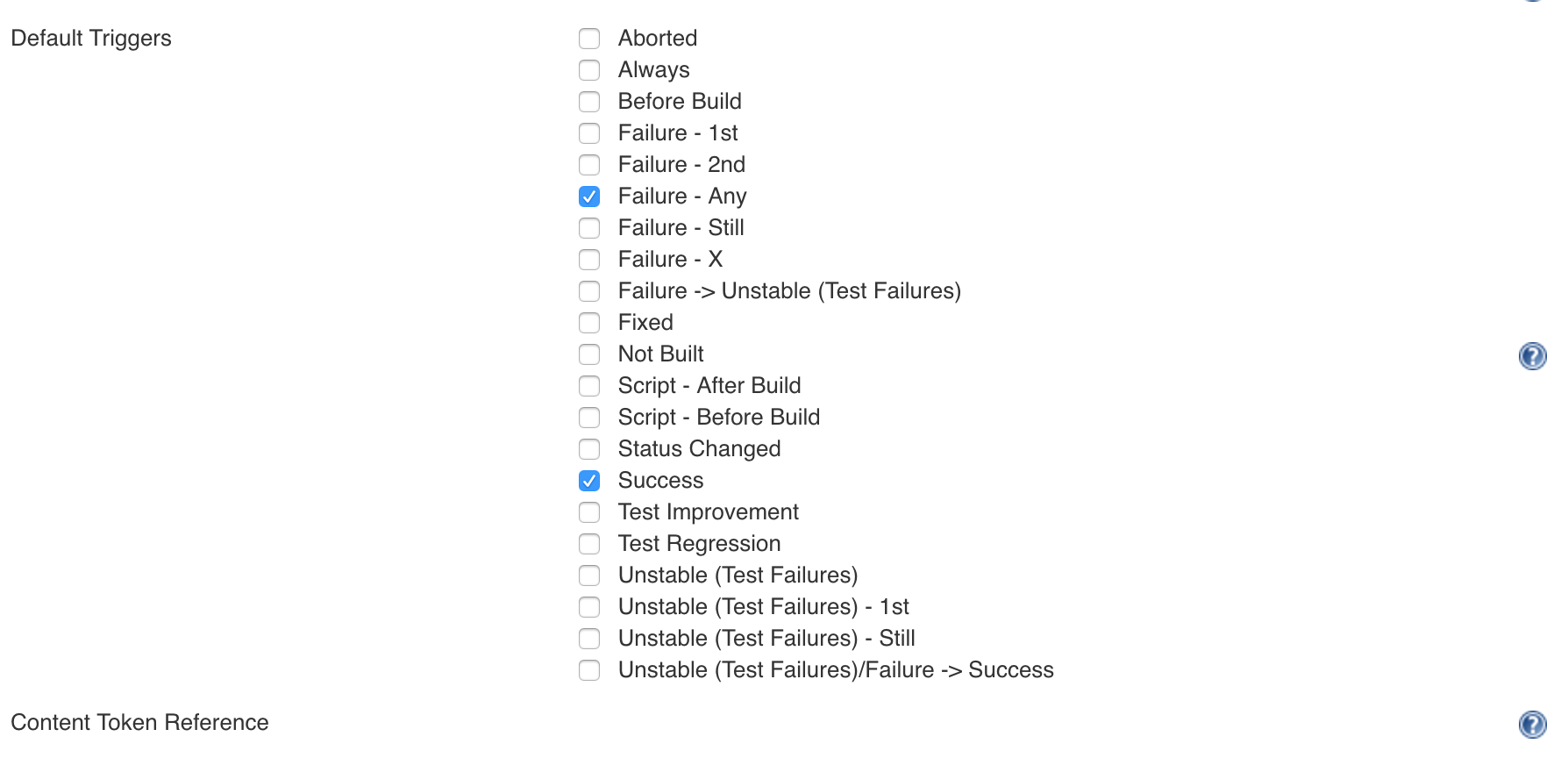
**1、在“系统管理->系统设置”中找到“Publish over FTP”配置，配置如下图：**

****

**2、在“系统管理->系统设置”中找到“Extended E-mail Notification”配置，配置如下图：**

****

****

****

**Default Content样例：**

(本邮件是程序自动下发的，请勿回复！)**<br/><hr/>**

项目名称：$PROJECT\_NAME**<br/><hr/>**

版本号：${FILE,path="version.txt"}**<br/><hr/>**

svn版本号：${SVN\_REVISION}**<br/><hr/>**

构建状态：$BUILD\_STATUS**<br/><hr/>**

触发原因：${CAUSE}**<br/><hr/>**

构建日志地址：**<a** href="${BUILD\_URL}console"**>**${BUILD\_URL}console**</a><br/><hr/>**

变更集:${JELLY\_SCRIPT,template="html"}**<br/><hr/>**

三、**fastlane**安装

**1、安装ruby版本>=2.2**

**1.1 安装rvm版本管理器**

$ curl -L https://get.rvm.io | bash -s stable

**1.2 等待一段时间后， 使用一下命令进行验证**

$ source ~/.bashrc

$ source ~/.bash\_profile

**1.3 测试是否安装正常**

$ rvm -v

如果出现rvm（版本号）.....基本就算是安装RVM成功了。

补充一些常用命令：

rvm list 查看已安装ruby

rvm list known 列出ruby可安装版本信息

rvm remove 2.2.2 卸载一个已安装的ruby版本

gem source 查看已有源

gem sources -a [http://ruby.taobao.org](http://ruby.taobao.org/)把源切换至淘宝镜像服务器

**1.4 安装ruby**

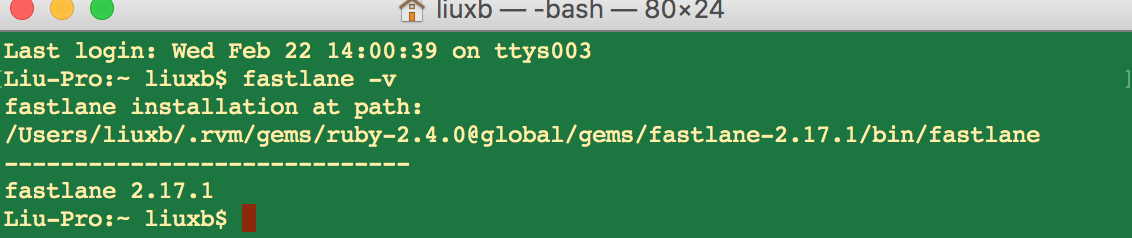
$ rvm install 2.4

**2、安装fastlane，详细资料请看**[**GitHub**](https://github.com/fastlane/fastlane)

$ sudo gem install fastlane

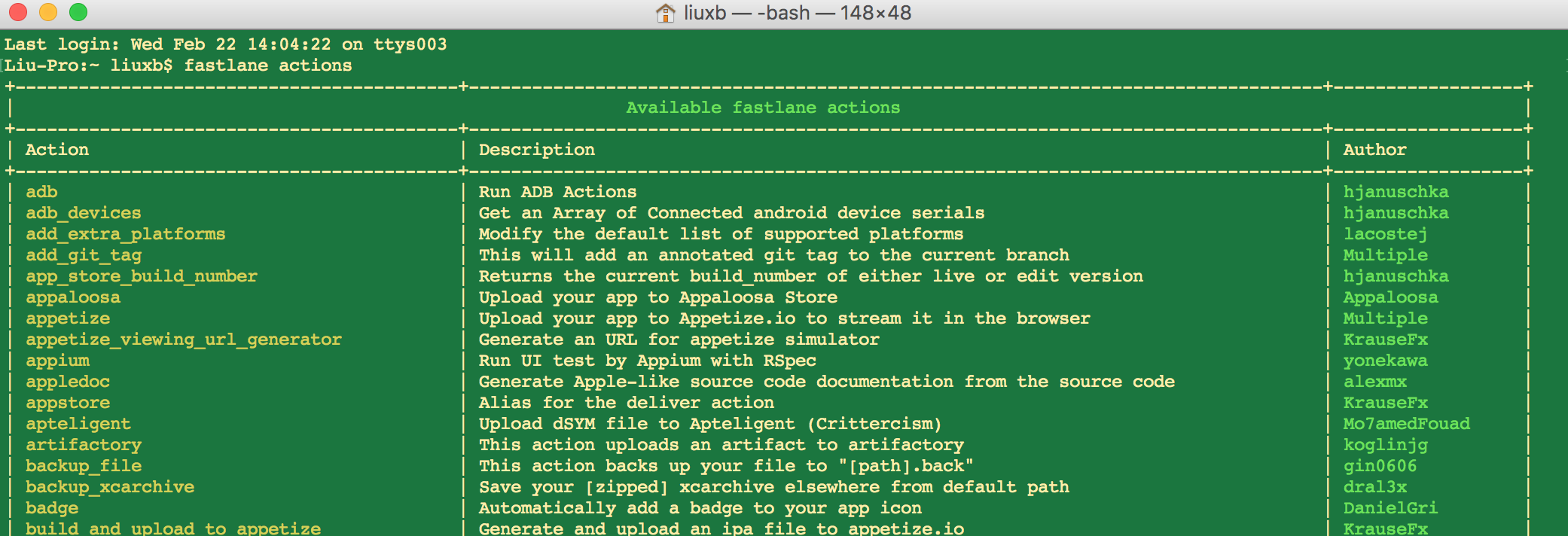
**2.1 查看版本：**

$ fastlane –v



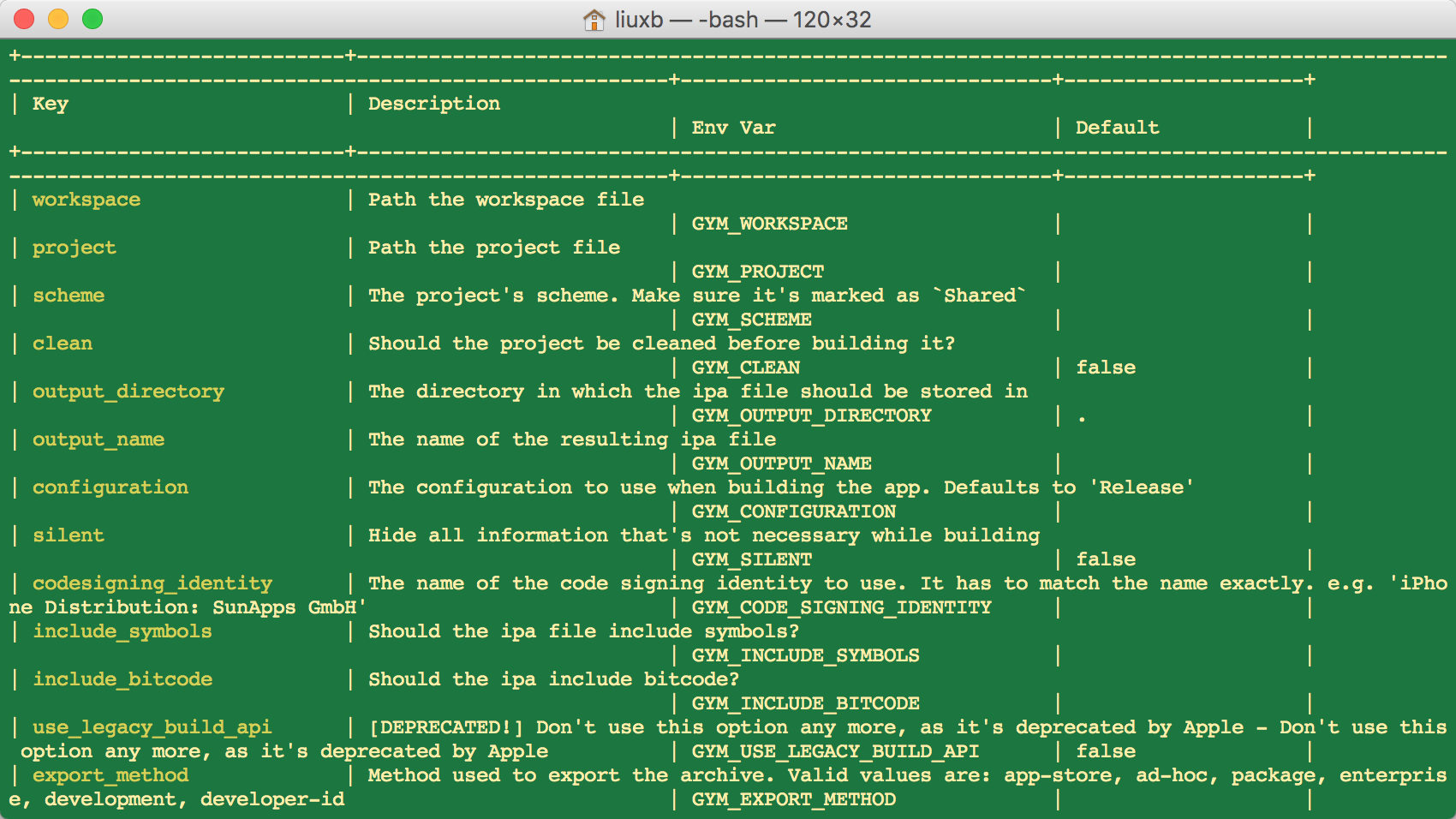
**2.2 查看命令方法**

$ fastlane actions

****

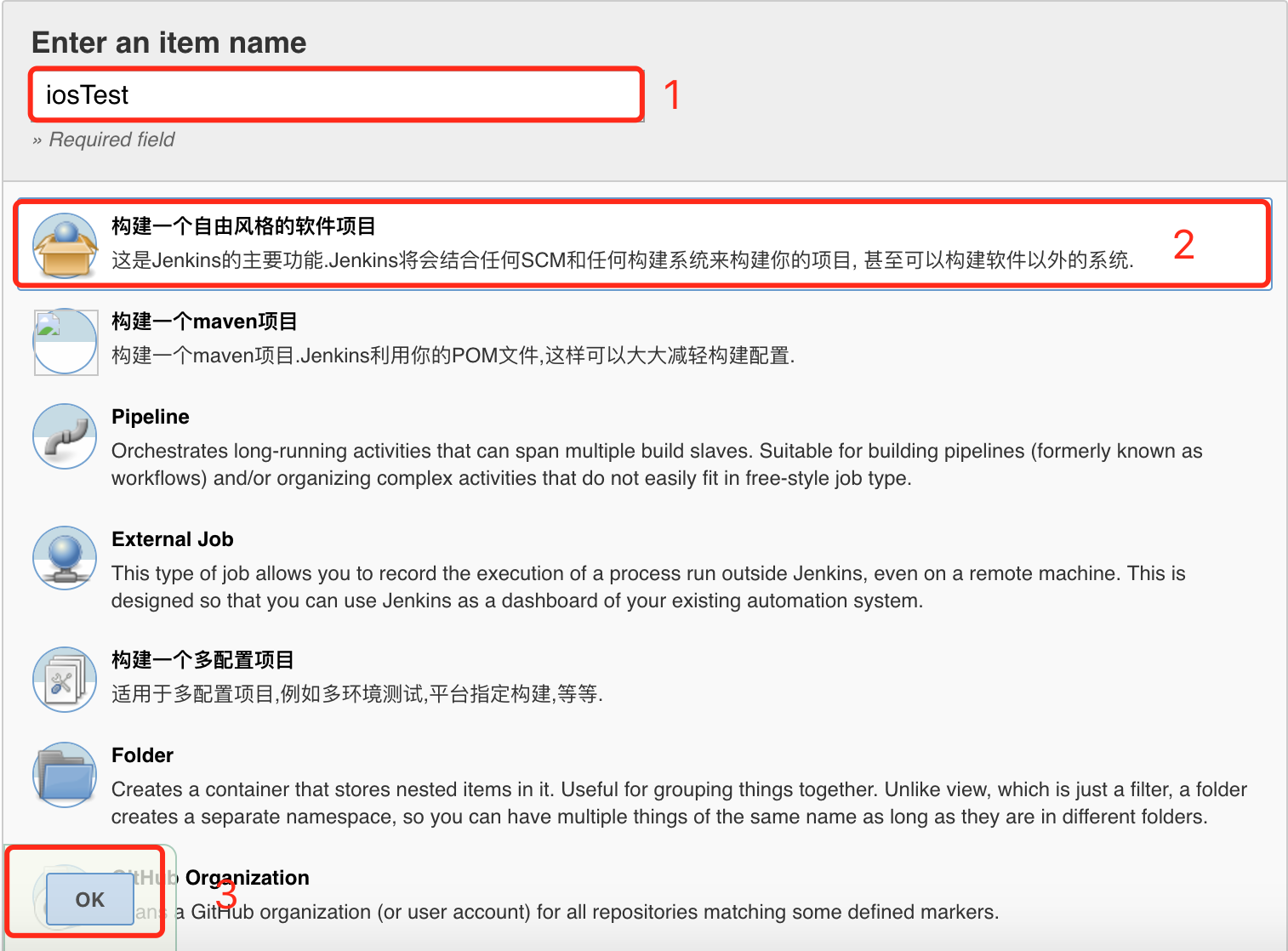
**2.2 查看指定方法**

$ fastlane actions gym

****

四、**Jenkins**新建**Job**

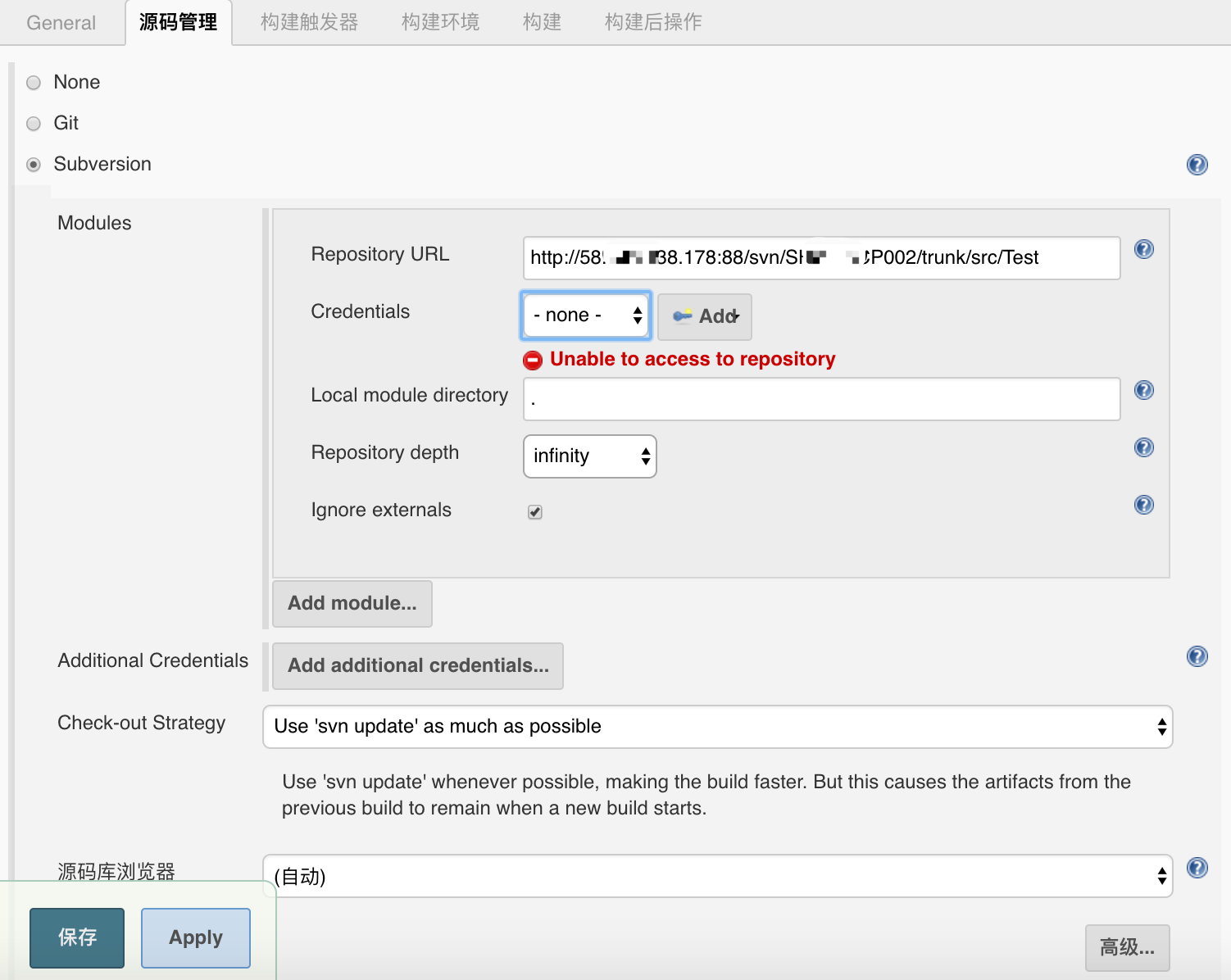
**1、新建一个item，选择自由风格的项目**



**2、输入项目名称，描述等基本信息。**



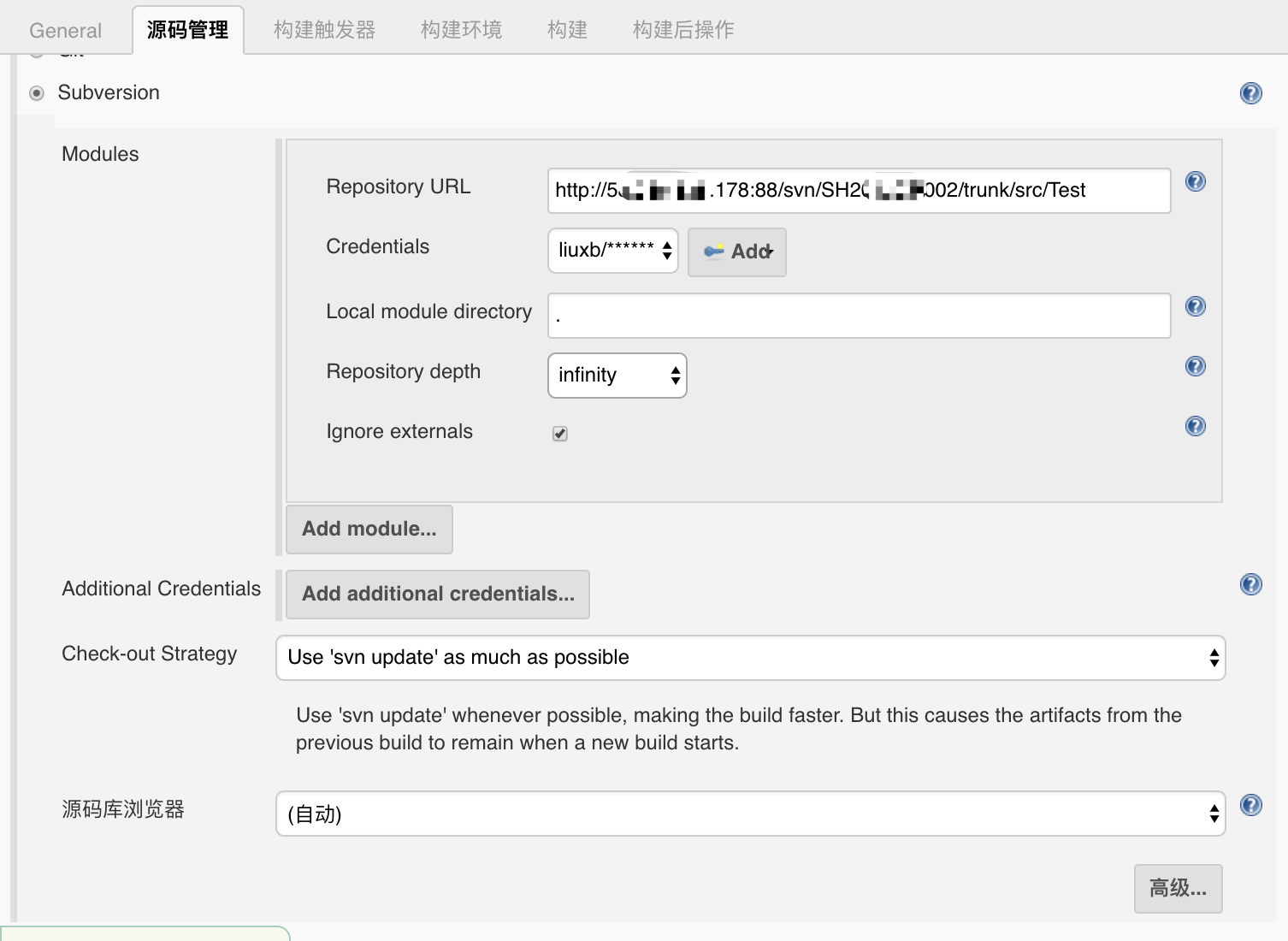
**3、源代码管理，因为使用的是SVN，所以选择Subversion。具体操作如下图所示：**



点击Add配置SVN用户信息。如下图所示：



成功配置如下图所示：



1. **配置构建触发器**

**3.1触发器支持多种类型，常用的有：**

A 定期进行构建（Build periodically）

B 根据提交进行构建（Build when a change is pushed to GitHub）

C 定期检测代码更新，如有更新则进行构建（Poll SCM）

构建触发器的选择为复合选项，若选择多种类型，则任一类型满足构建条件时就会执行构建工作。

关于定时器（Schedule）的格式，简述如下：MINUTE HOUR DOM MONTH DOW

* MINUTE: Minutes within the hour (0-59)
* HOUR: The hour of the day (0-23)
* DOM: The day of the month (1-31)
* MONTH: The month (1-12)
* DOW: The day of the week (0-7) where 0 and 7 are Sunday.

通常情况下需要指定多个值，这时可以采用如下operator（优先级从上到下）：

* \*适配所有有效的值，若不指定某一项，则以\*占位；
* M-N适配值域范围，例如7-9代表7/8/9均满足；
* M-N/X或\*/X：以X作为间隔；
* A,B,C：枚举多个值。

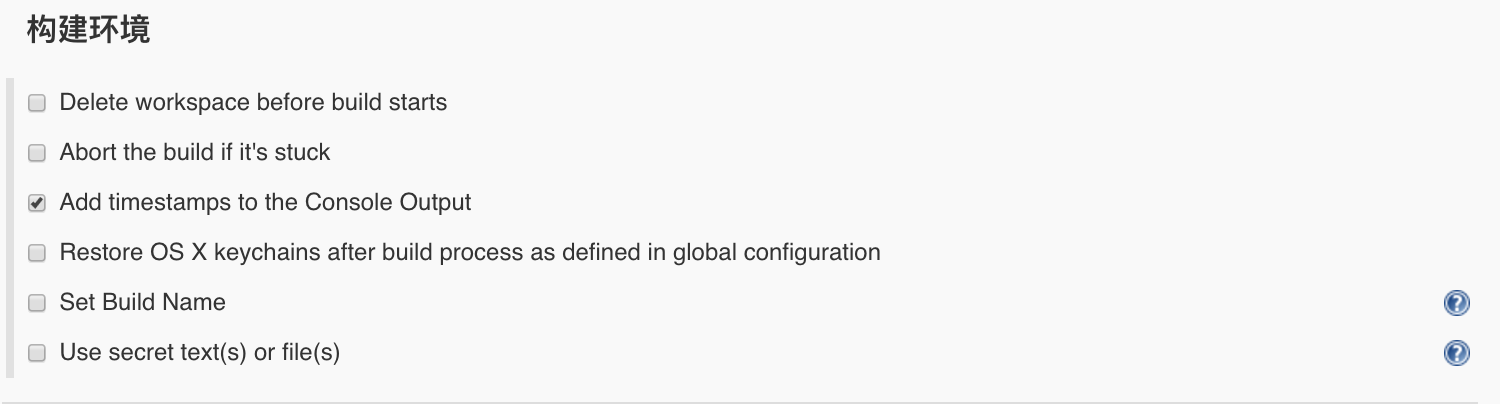
另外，为了避免多个任务在同一时刻同时触发构建，在指定时间段时可以配合使用H字符。添加H字符后，Jenkins会在指定时间段内随机选择一个时间点作为起始时刻，然后加上设定的时间间隔，计算得到后续的时间点。直到下一个周期时，Jenkins又会重新随机选择一个时间点作为起始时刻，依次类推。

为了便于理解，列举几个示例：

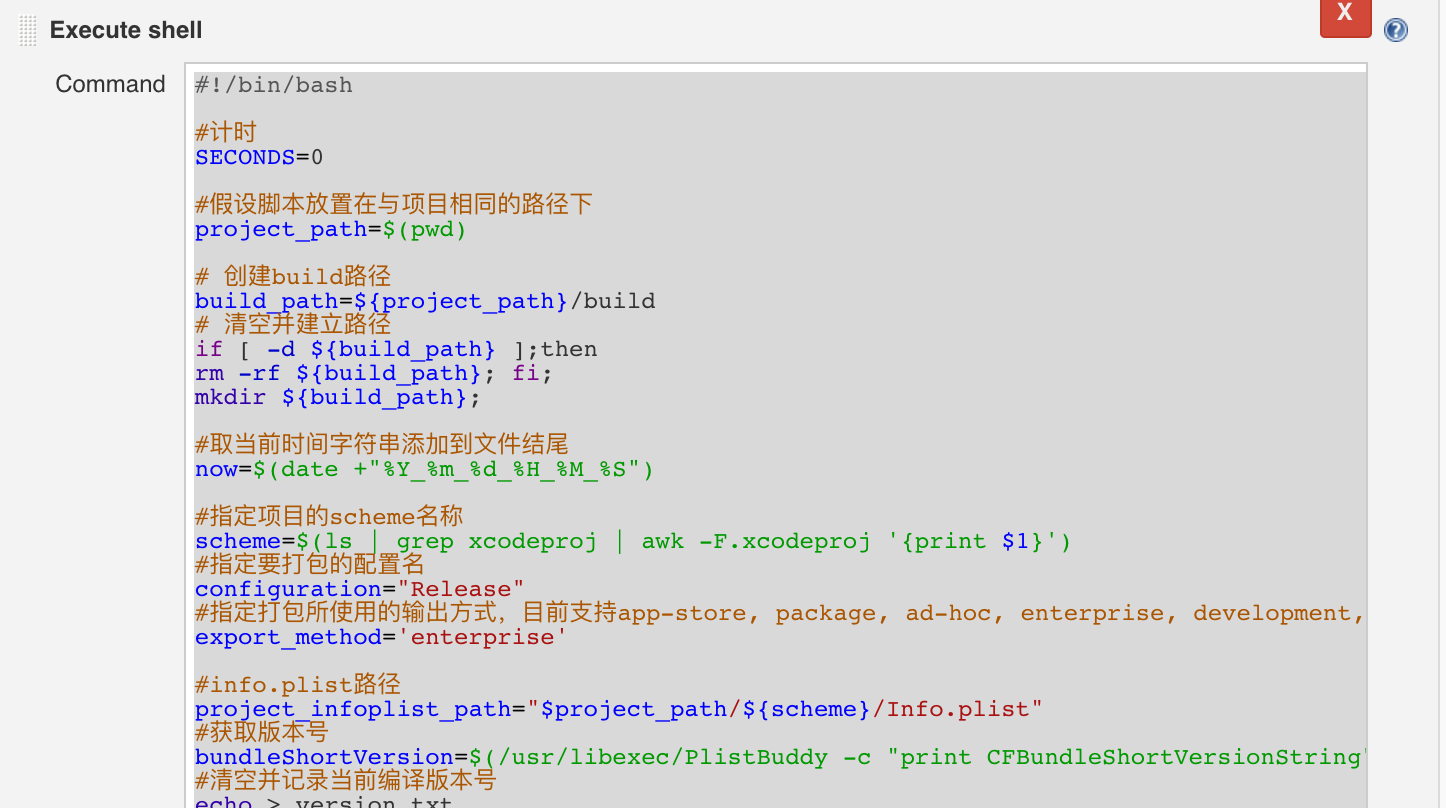
* H/15 \* \* \* \*：代表每隔15分钟，并且开始时间不确定，这个小时可能是:07,:22,:37,:52，下一个小时就可能是:03,:18,:33,:48；
* H(0-29)/10 \* \* \* \*：代表前半小时内每隔10分钟，并且开始时间不确定，这个小时可能是:04,:14,:24，下一个小时就可能是:09,:19,:29；
* H 23 \* \* 1-5：工作日每晚23:00至23:59之间的某一时刻；



1. **构建环境（没有用到，暂未深入研究）**



1. **构建选择Execute shell**



执行脚本：

*#!/bin/bash*

*#计时*

SECONDS=0

*#假设脚本放置在与项目相同的路径下*

project\_path=**$(**pwd**)**

*# 创建build路径*

build\_path=**${**project\_path**}**/build

*# 清空并建立路径*

**if** [ -d **${**build\_path**}** ];**then**

rm -rf **${**build\_path**}**; **fi**;

mkdir **${**build\_path**}**;

*#取当前时间字符串添加到文件结尾*

now=**$(**date +"%Y\_%m\_%d\_%H\_%M\_%S"**)**

*#指定项目的scheme名称*

scheme=**$(**ls | grep xcodeproj | awk -F.xcodeproj '{print $1}'**)**

*#指定要打包的配置名*

configuration="Release"

*#指定打包所使用的输出方式，目前支持app-store, package, ad-hoc, enterprise, development, 和developer-id，即xcodebuild的method参数*

export\_method='enterprise'

*#info.plist路径*

project\_infoplist\_path="$project\_path/${scheme}/Info.plist"

*#获取版本号*

bundleShortVersion=**$(**/usr/libexec/PlistBuddy -c "print CFBundleShortVersionString" "${project\_infoplist\_path}"**)**

*#清空并记录当前编译版本号*

echo > version.txt

echo "${bundleShortVersion}">> version.txt

*#指定项目地址*

workspace\_path="$project\_path/${scheme}.xcworkspace"

*#指定输出路径*

output\_path=**${**build\_path**}**

*#指定输出归档文件地址*

archive\_path="$output\_path/${scheme}\_${now}.xcarchive"

*#指定输出ipa地址*

ipa\_path="$output\_path/${scheme}\_${now}.ipa"

*#指定输出ipa名称*

ipa\_name="${scheme}.ipa"

*#获取执行命令时的commit message*

commit\_msg="$1"

*#输出设定的变量值*

echo "===workspace path: ${workspace\_path}==="

echo "===archive path: ${archive\_path}==="

echo "===ipa path: ${ipa\_path}==="

echo "===export method: ${export\_method}==="

echo "===commit msg: $1==="

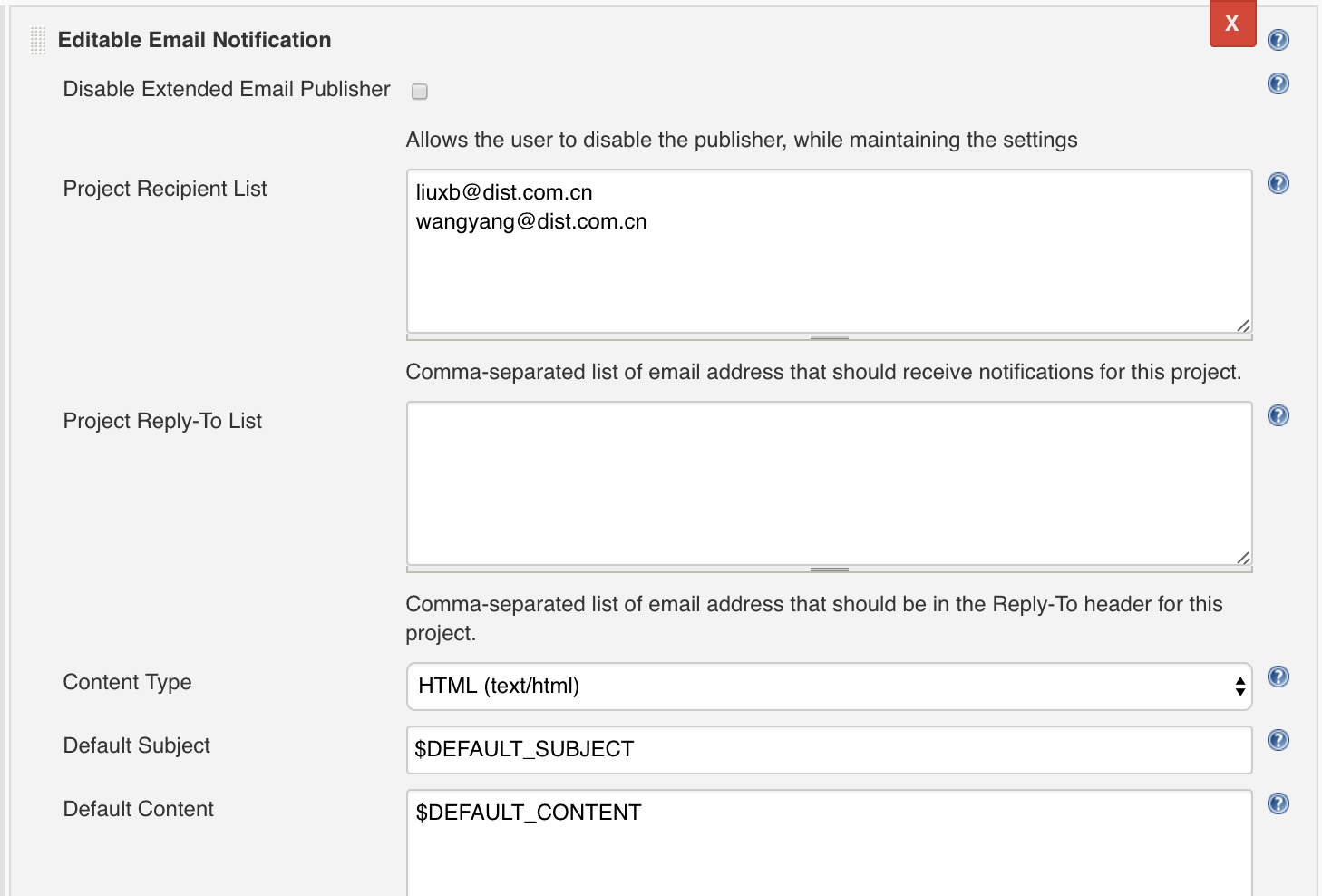
*#先清空前一次build*

fastlane gym --workspace **${**workspace\_path**}** --scheme **${**scheme**}** --clean --configuration **${**configuration**}** --archive\_path **${**archive\_path**}** --export\_method **${**export\_method**}** --output\_directory **${**output\_path**}** --output\_name **${**ipa\_name**}**

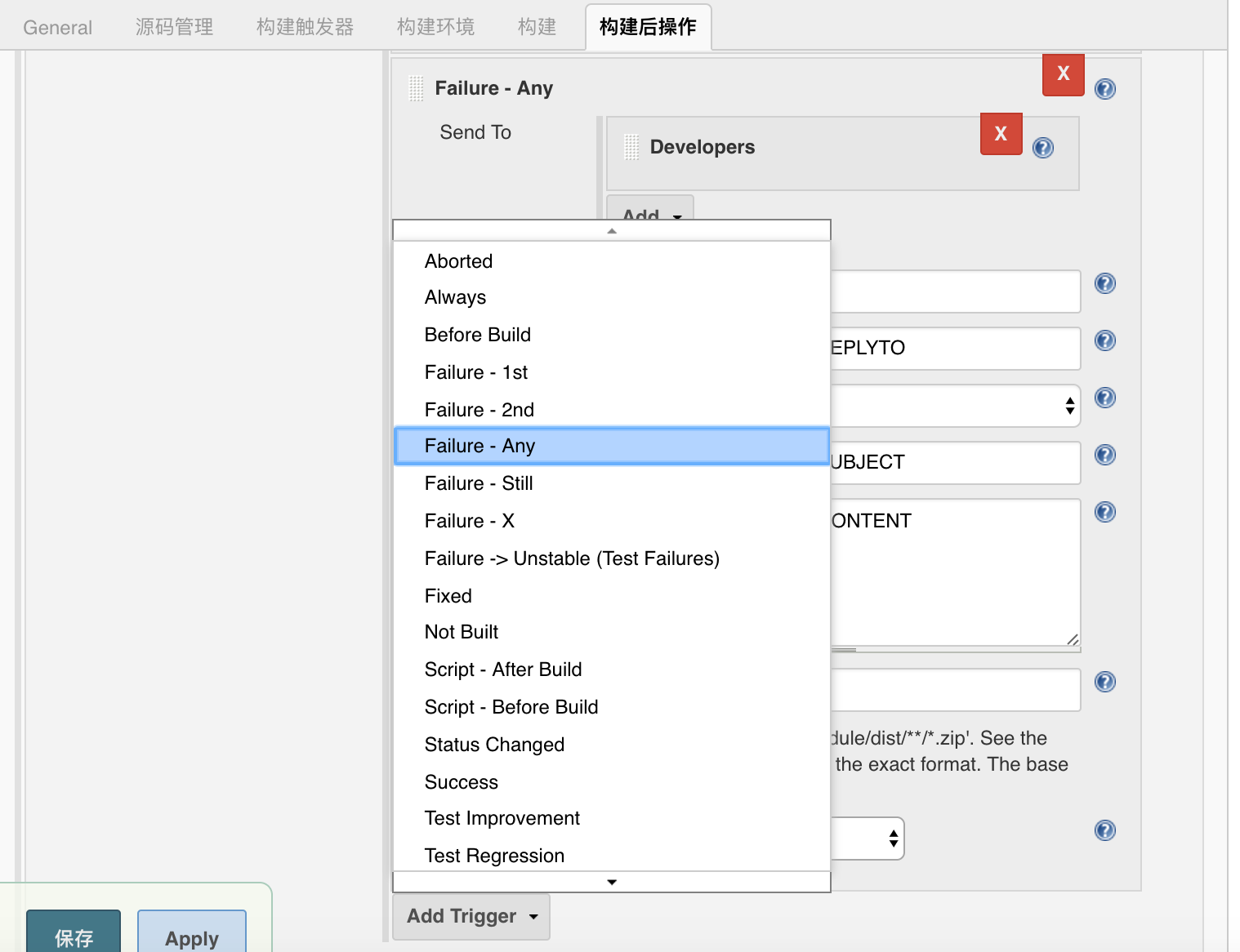
*#输出总用时*

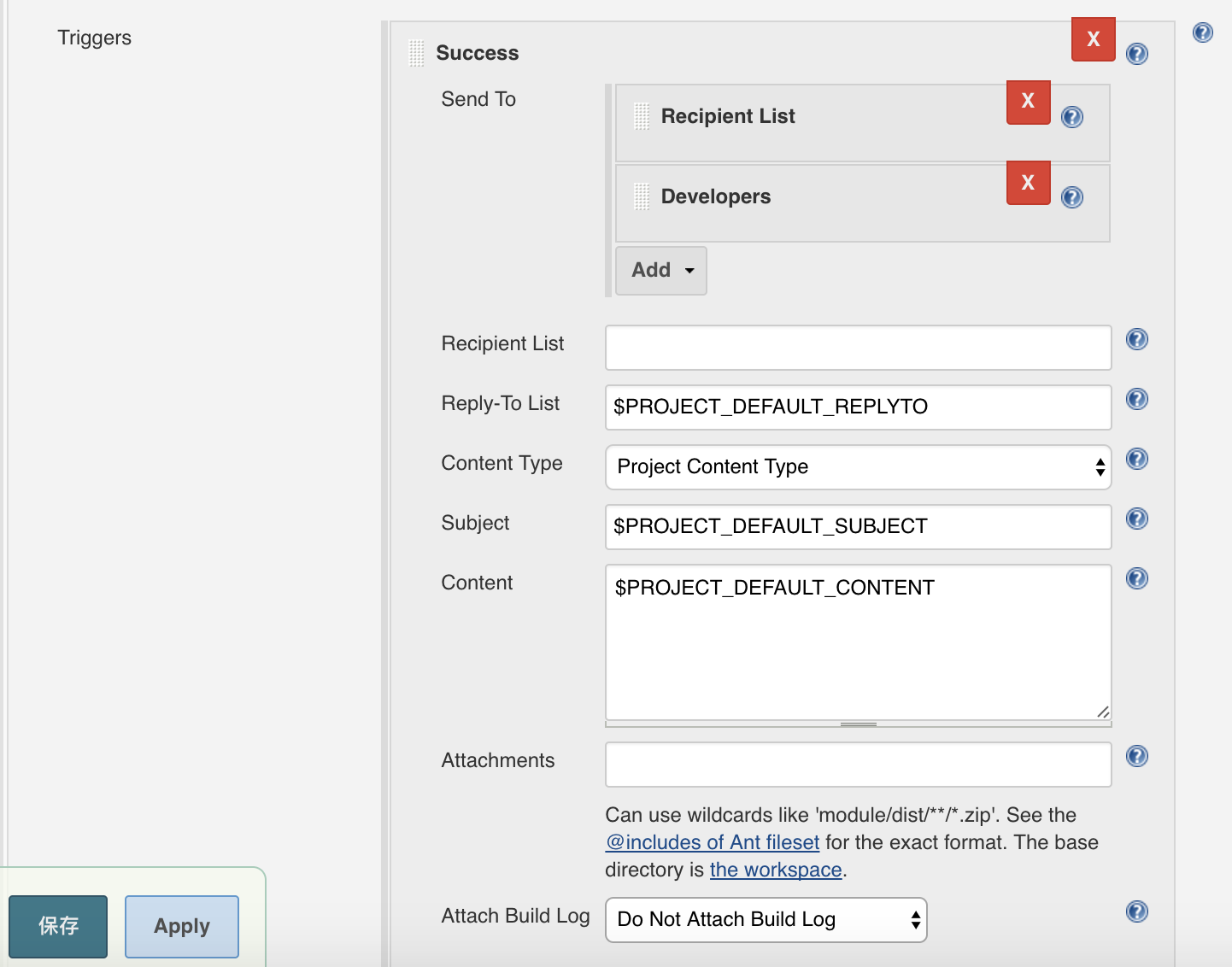
echo "===Finished. Total time: ${SECONDS}s==="

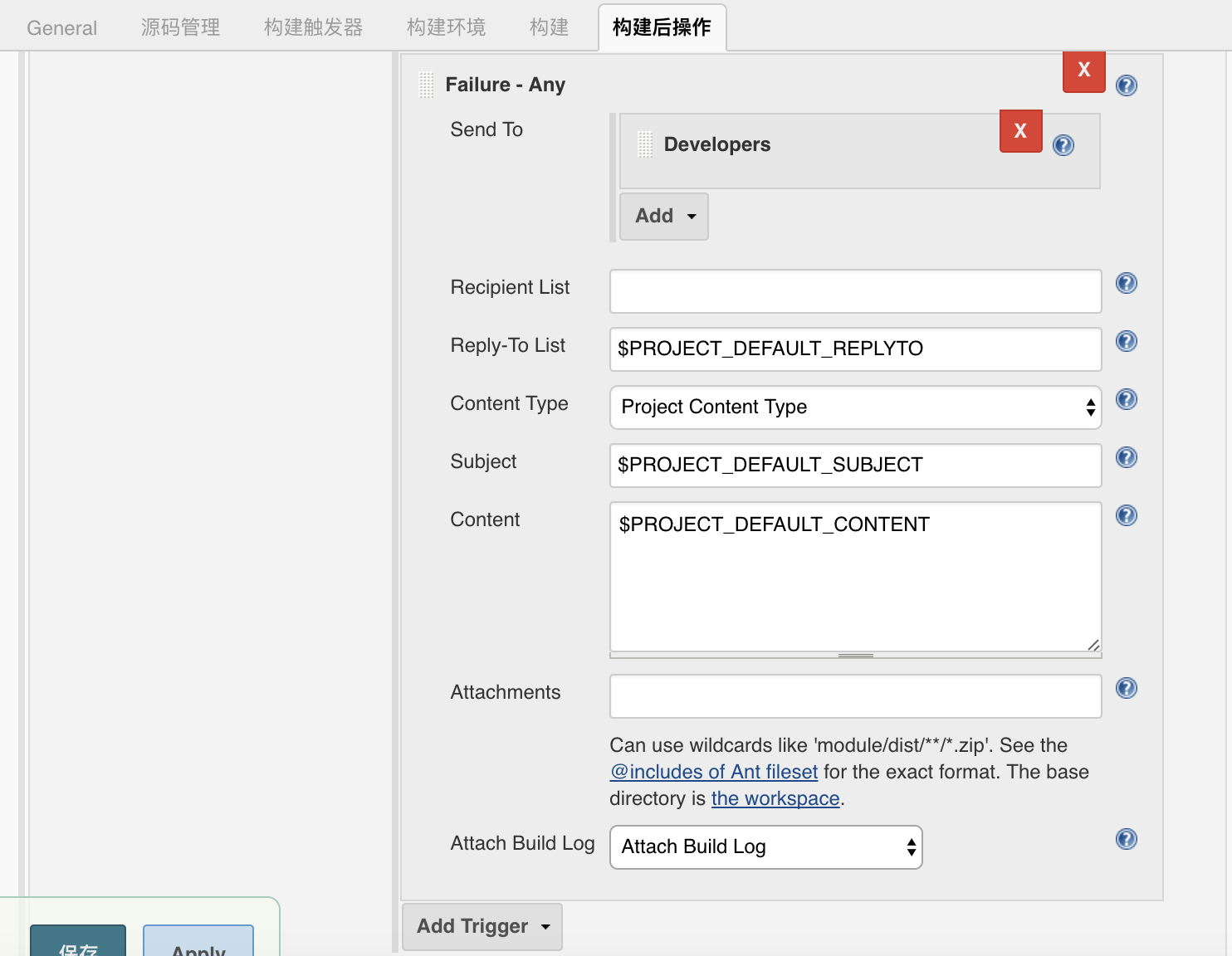
1. **构建后操作增加邮件提醒，选择Editable Email Notification**



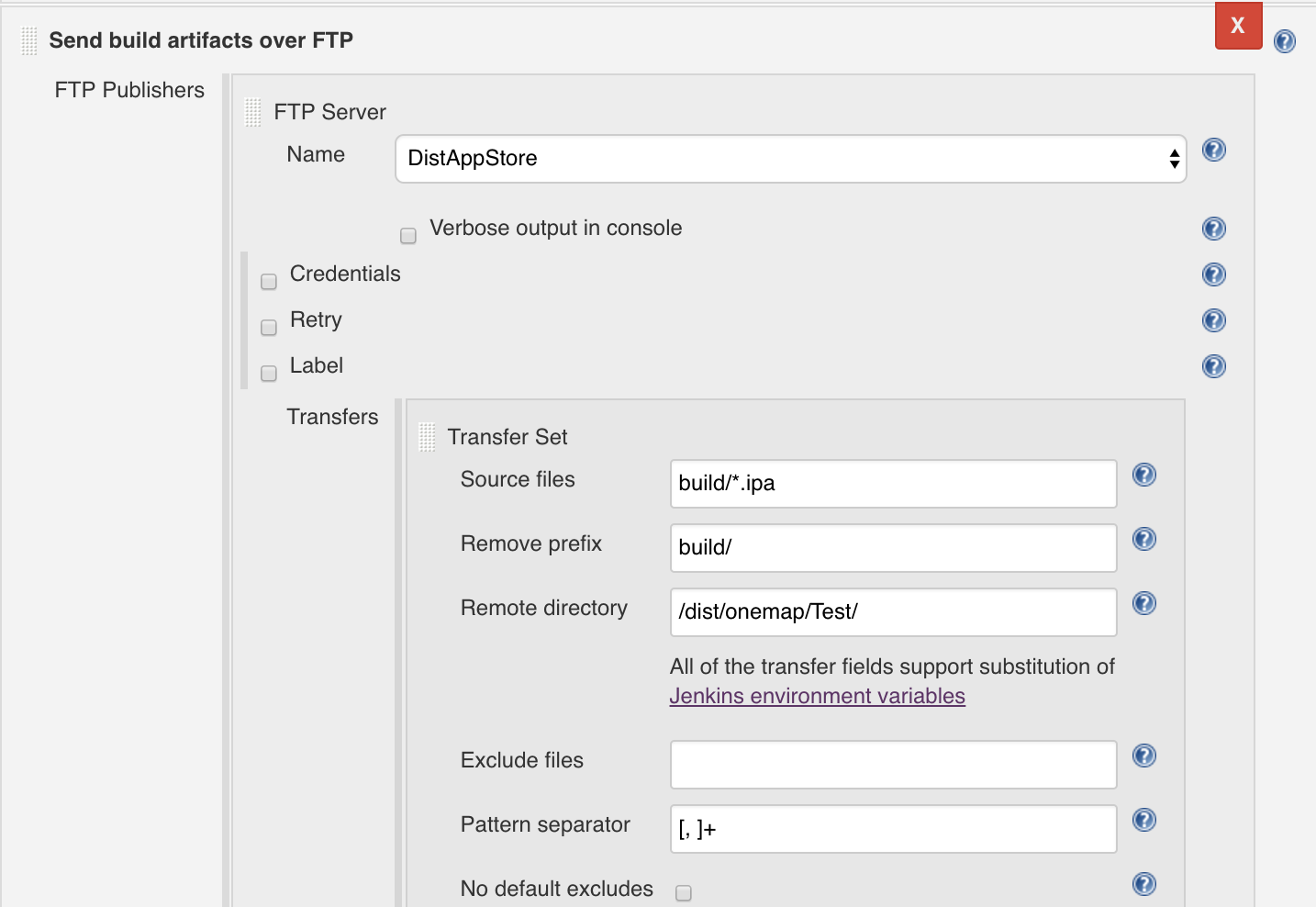
配置触发操作：







1. **构建后操作增加文件上传，选择Send build artifacts over FTP**



1. **构建完毕，执行构建。查看结果吧。**