#### Jenkins持续集成企业实战系列

#### 目 录

[1. 传统网站部署的流程 2](#_Toc466882026)

[2. 目前主流网站部署的流程 3](#_Toc466882027)

[3. Jenkins持续集成简介 4](#_Toc466882028)

[4. 持续集成意义 5](#_Toc466882029)

[5. 持续集成组件 5](#_Toc466882030)

[6. Jenkins安装部署 6](#_Toc466882031)

[7. Jenkins配置演示 7](#_Toc466882032)

[8. Jenkins手动构建 13](#_Toc466882033)

[9. Jenkins脚本自动化 15](#_Toc466882034)

[10. 自动部署脚本 16](#_Toc466882035)

[11. Jenkins插件安装 17](#_Toc466882036)

[12. Jenkins邮件设置 21](#_Toc466882037)

### 传统网站部署的流程

传统的网站部署，大家在运维过程中，网站部署是运维的工作之一，网站部署的流程大致分为：

需求分析—原型设计—开发代码—提交测试—内网部署—确认上线—备份数据—外网更新-最终测试，如果发现外网部署的代码有异常，需要及时回滚。（如下图所示）

#### 

#### 案例演示：部署LAMP网站，同时更新网页内容，检查效果！

#### 

#### 更新完后的网站内容:

#### 

### 目前主流网站部署的流程

#### 目前主流网站部署方法，通过Hudson/Jenkins工具平台实现全自动部署+测试，是一个可扩展的持续集成引擎，是一个开源软件项目，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。Jenkins非常易于安装和配置，简单易用。

#### 简单来说方便如下人员：

#### 开发人员：写好代码，不需要自己进行源码编译、打包等工作，直接将代码分支存放在SVN、GIT仓库即可。

#### 运维人员：减轻人工干预的错误率，同时解放运维人员繁杂的上传代码、手动备份、更新。

#### 测试人员：可以通过jenkins进行简单的代码及网站测试。

#### 

### Jenkins持续集成简介

#### 持续集成(Continuous Integration)是一种软件开发实践，对于提高软件开发效率并保障软件开发质量提供了理论基础。Jenkins 是一个开源软件项目，旨在提供一个开放易用的软件平台，使持续集成变成可能。本文正是从持续集成的基本概念入手，通过具体实例，介绍了如何基于 Jenkins 快速搭建持续集成环境。

Jenkins是一个可扩展的持续集成引擎，是一个开源软件项目，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。Jenkins非常易于安装和配置，简单易用。

### 持续集成意义

1. 持续集成中的任何一个环节都是自动完成的，无需太多的人工干预，有利于减少重复过程以节省时间、费用和工作量；
2. 持续集成保障了每个时间点上团队成员提交的代码是能成功集成的。换言之，任何时间点都能第一时间发现软件的集成问题，使任意时间发布可部署的软件成为了可能；
3. 持续集成还能利于软件本身的发展趋势，这点在需求不明确或是频繁性变更的情景中尤其重要，持续集成的质量能帮助团队进行有效决策，同时建立团队对开发产品的信心。

### 持续集成组件

1. **一个自动构建过程，包括自动编译、分发、部署和测试；**
2. **一个代码存储库**，即需要版本控制软件来保障代码的可维护性，同时作为构建过程的素材库，例如SVN、GIT代码库；
3. **一个jenkins持续集成服务器**(123.57.11.203服务器就是一个配置简单和使用方便的持续集成服务器)

### Jenkins安装部署

地址<http://mirrors.jenkins-ci.org/>下载适合的Jenkins版本。

Linux稳定版的Jenkins地址为：<http://ftp.yz.yamagata-u.ac.jp/pub/misc/jenkins/war-stable/1.651.2/jenkins.war>

由于Jenkins属于一个JAVA 代码，需要java容器才能运行jenkins，所以需要安装JDK+Tomcat，方法如下：

1. JAVA JDK安装

#下载相应的jdk软件包，然后解压安装，我这里包名称为：jdk-7u25-linux-x64.tar.gz

**tar -xzf jdk-7u25-linux-x64.tar.gz ;mkdir -p /usr/java/ ;mv jdk1.7.0\_25/ /usr/java/**

#然后配置环境变量，这样可以任何地方引用jdk，如下配置：

#**vi /etc/profile** 最后面加入以下语句：

**export JAVA\_HOME=/usr/java/jdk1.7.0\_25**

**export CLASSPATH=$CLASSPATH:$JAVA\_HOME/lib:$JAVA\_HOME/jre/lib**

**export PATH=$JAVA\_HOME/bin:$JAVA\_HOME/jre/bin:$PATH:$HOMR/bin**

#source /etc/profile #使环境变量马上生效

#java --version #查看java版本，看到jdk1.7.0\_25版本即代表java jdk安装成功。

1. Tomcat安装配置

#官方网站下载tomcat 6.0.30或者其他版本：

wget <http://mirrors.tuna.tsinghua.edu.cn/apache/tomcat/tomcat-6/v6.0.45/bin/apache-tomcat-6.0.45.tar.gz>

**tar xzf apache-tomcat-6.0.45.tar.gz**

#直接解压就可以使用，解压完成执行：

**mv apache-tomcat-6.0.45 /usr/local/tomcat**

把得到的war包直接扔到tomcat下webapps/ROOT/下，启动tomcat，Jenkins就安装完毕！

**rm -rf /usr/local/tomcat/webapps/\* ;**

**mkdir -p /usr/local/tomcat/webapps/ROOT/;**

**mv jenkins.war /usr/local/tomcat/webapps/ROOT/;**



### Jenkins配置演示

在学习Jenkins之前我们先来了解几个常用的开发代码、构建代码的几个名词：MAKE、ANT、MAVEN、Eclipse、Jenkins等

1. Make工具

Linux或者Windows最原始的编译工具，在linux下编译程序常用make（Windows下对应的工具为nmake)。它负责组织构建的过程，负责指挥编译器如何编译，连接器如何连接，最后生成一个可用的文件。

1. Ant工具

Make编译一些比较复杂的工具使用起来不灵活，语法很难理解，延伸出ant工具。

1. Maven工具

Maven工具是对ant工具的进一步改进，在make工具中，如果我们要编译某些源文件，首先要安装编译器等工具。有时候需要不同版本的编译器，在java的编译器需要不同的各种包的支持，如果把每个包都下载下来，在makefile中进行配置制定，当需要的包非常多时，很难管理。

Maven像make一样，是个构建(build)工具，它如何调用各种不同的编译器连接器等呢？使用Maven plugin (maven 插件）。

1. Jenkins工具

Maven已经很强大了，那Jenkins是个什么东西？

其实Maven还是不够强大。Maven可以控制编译，控制连接，可以生成各种报告，可以进行代码测试。

可是如何控制这个流程呢？ 编译还是先连接？先进行代码测试，还是先生成报告？可以使用脚本来对maven进行控制，实现这些流程的控制。

1. Eclipse工具

Eclipse 是一个开放源代码的、基于Java的可扩展开发平台。就其本身而言，它只是一个框架和一组服务，用于通过插件组件构建开发环境。幸运的是，Eclipse 附带了一个标准的插件集，包括Java开发工具（Java Development Kit，JDK）。

1. [Maven](http://baike.baidu.com/view/336103.htm)项目对象模型POM (Project Object Model)，可以通过一小段描述信息来管理项目的构建，报告和文档的软件[项目管理工具](http://baike.baidu.com/view/3169783.htm" \t "_blank)。Maven 除了以程序构建能力为特色之外，还提供高级项目管理工具。
2. POM是Maven项目中的文件，使用XML表示，名称叫做pom.xml。在Maven中，当谈到Project的时候，不仅仅是一堆包含代码的文件。一个Project往往包含一个配置文件，包括了与开发者有关的，缺陷跟踪系统，组织与许可，项目的URL，项目依赖，以及其他。它包含了所有与这个项目相关的东西。事实上，在Maven世界中，project可以什么都没有，甚至没有代码，但是必须包含pom.xml文件。
3. 由于 Maven 的缺省构建规则有较高的可重用性，所以常常用两三行 Maven 构建脚本就可以构建简单的项目。由于 Maven 的面向项目的方法，许多 Apache Jakarta 项目发文时使用 Maven，而且公司项目采用 Maven 的比例在持续增长。

安装JDK及MAVEN工具,并在Jenkins平台进行设置,如下步骤

**wget** [**http://mirrors.tuna.tsinghua.edu.cn/apache/maven/maven-3/3.3.9/binaries/apache-maven-3.3.9-bin.tar.gz**](http://mirrors.tuna.tsinghua.edu.cn/apache/maven/maven-3/3.3.9/binaries/apache-maven-3.3.9-bin.tar.gz)

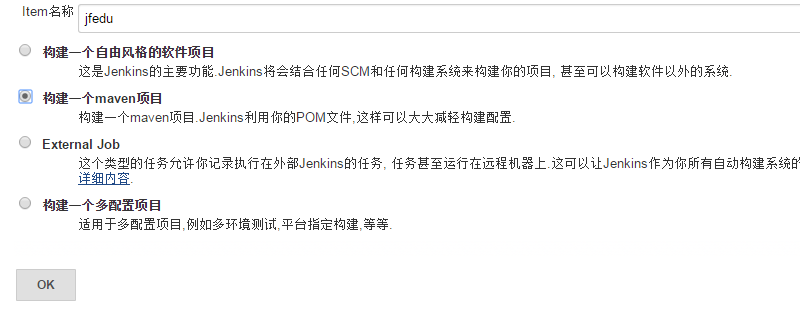
**tar -xzf apache-maven-3.3.9-bin.tar.gz ;mv apache-maven-3.3.9 /usr/maven/**



**然后设置如下：**



1. 返回首页，第一次进入里面没有数据，我们需要创建项目工程job。
2. 点击左上角的new job，在new job页面需要选择job的类型，Jenkins支持几种类型，我们选择“构建一个maven项目”，Job名为：jfedu，如图所示：

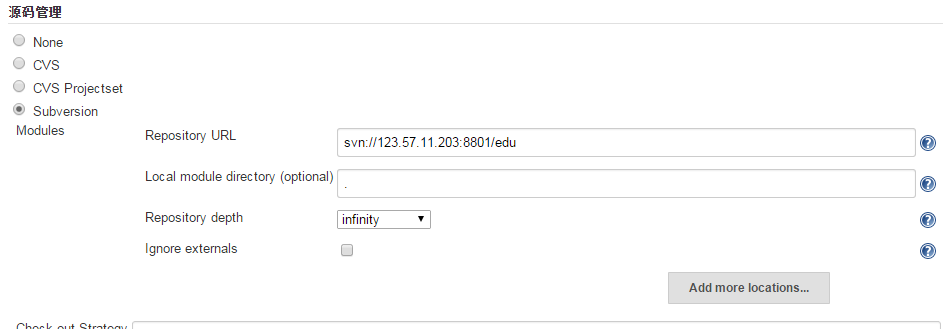


1. 点击OK按钮后，进会进入详细配置界面，详细配置界面的配置项很多，不过不用怕，大部分使用默认配置就可以了，下面就说说我们需要修改的几个地方：
2. Source Code Management

因为我们使用SVN管理源码，所以这里选择Subversion，并在Repository URL中输入我们的SVN地址：

[svn://123.57.11.203:8801/edu/](http://123.57.11.203:8801/edu/)

输入SVN库的URL地址后，Jenkins会自动验证地址，并给予提示。

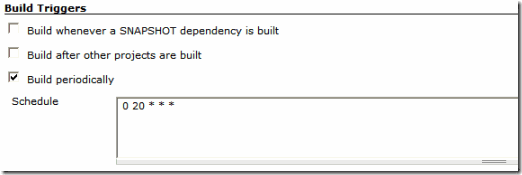


如果需要SVN用户名和密码：

用户名:wuguangke

密码：shareku&1118

1. 配置自动构建的计划，假设我们想让项目晚上8点自动构建一次，只需要在Build Triggers中选择Build periodically，并在Schedule中输入 0 20 \* \* \*，我配置的是每晚8点自动构建：

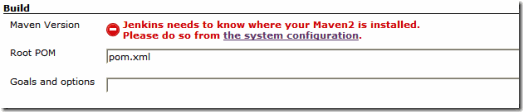
[](http://images.cnitblog.com/blog/330894/201301/30164334-db7467f4e2314133b2cc03420d9310a9.gif)

注：Schedule的配置规则是有5个空格隔开的字符组成，从左到右分别代表：分 时 天 月 年。\*代表所有，0 20 \* \* \* 表示“在任何年任何月任何天20点0分”进行构建。

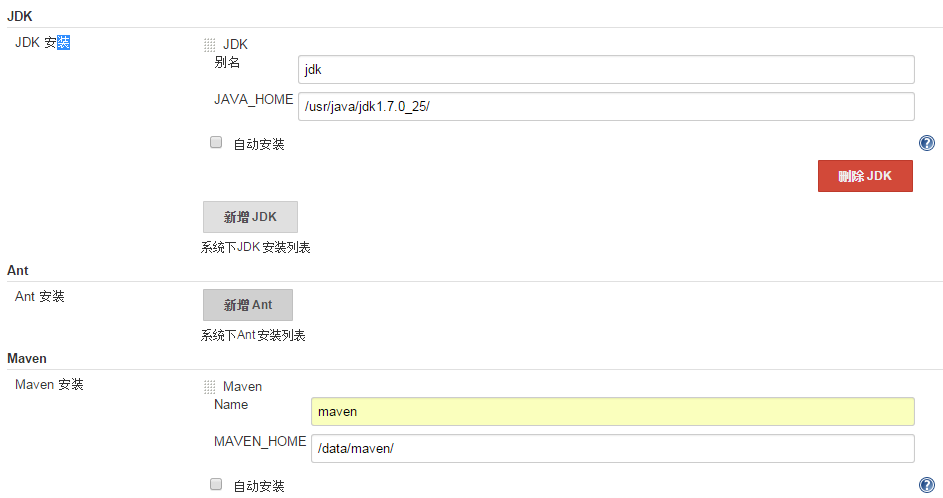
1. 配置到这里，可能有人发现在Build配置节点，有红色错误信息，提示

**Jenkins needs to know where your Maven2 is installed.   
Please do so from the system configuration.**

如图所示：

[](http://images.cnitblog.com/blog/330894/201301/30164335-d664d1a41920431ca11e5ef57a397d6c.gif)

这是因为Jenkins找不到maven的原因，点击"system configuration"，是system configuration的maven配置中添加maven目录就OK，如下图：



我设置了JRE 7和MAVEN的安装目录。

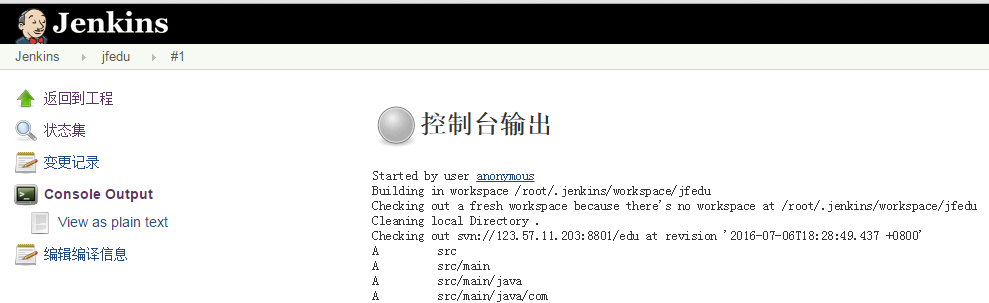
点击左下角的SAVE按钮，保存设置，保存好所有配置后，我们第1个job就算是完成了。

### Jenkins手动构建

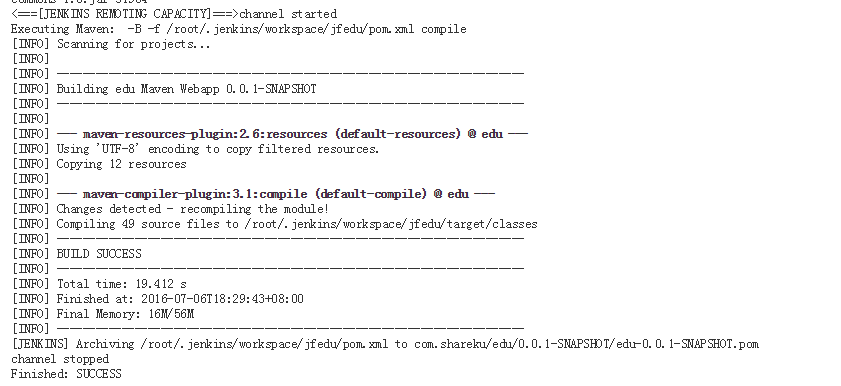
在经过上面的配置后，回到Jenkins首页，在首页可以看到刚才添加的1个job，如下图：



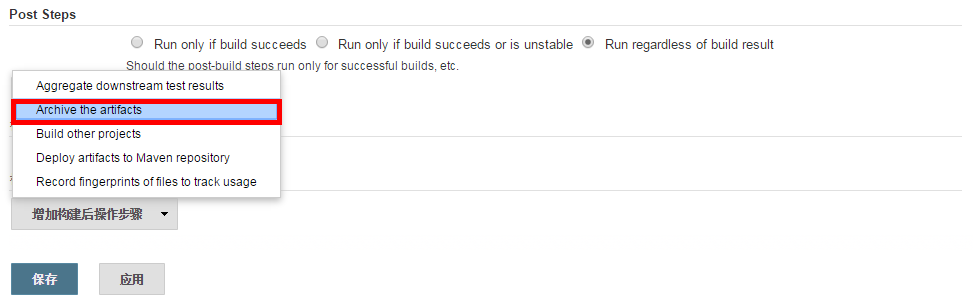
点击某1个job后后面的"Schedule a build"图片手动构建，点击完后，会在左边的Build Queue或者[Build Executor Status](http://localhost:8080/hudson/computer/" \t "_blank) 显示正在构建的任务，在自动构建完后，刷新页面，就可以看到构建结果了，如何某个项目构建失败，点击项目后面的构建数字(从1开始递增)进入项目的"[Console Output](http://localhost:8080/hudson/job/Entities/lastSuccessfulBuild/console) "可以查看项目构建失败的原因。当然我们也可以配置把构建失败的结果发到邮箱。

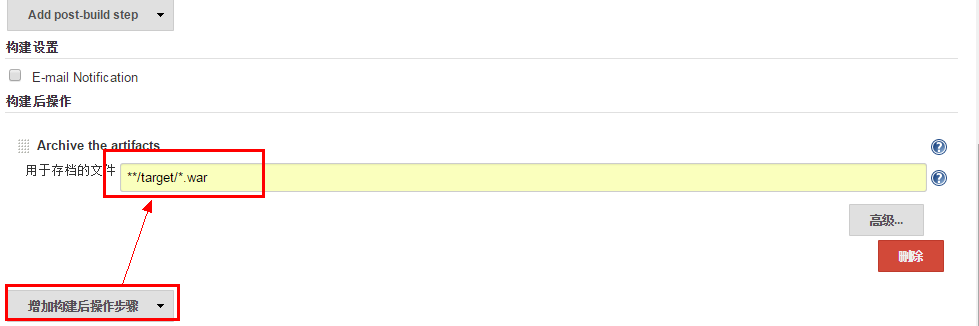


构建完成

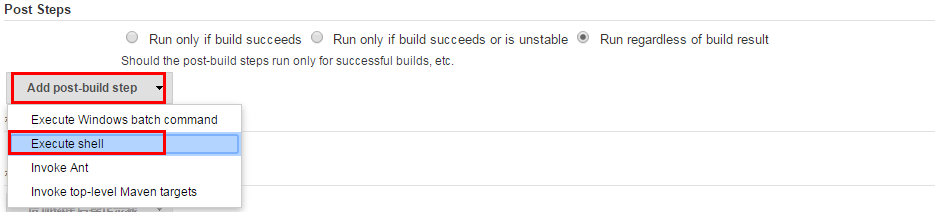


### Jenkins脚本自动化

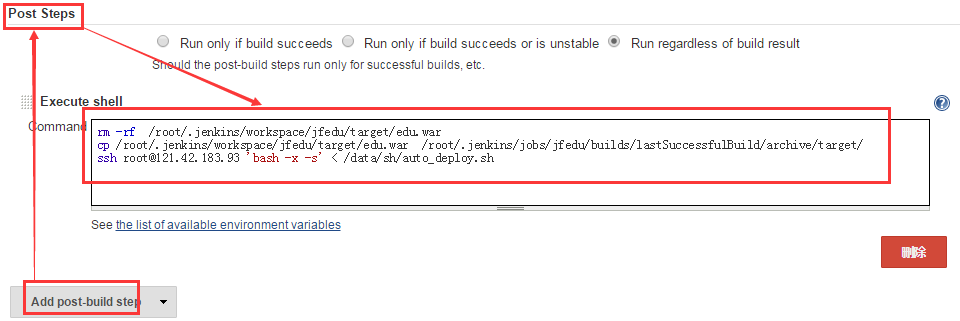
1. 点击工程名称-配置-构建后操作-设置Jenkins编译打包完后，war包存放的位置，如下图：
2. 然后输入如下配置：\*\*/target/\*.war



1. 如上-构建触发器-一个构建步骤，选择Add-post-build step-Excute shell



1. 然后把脚本内容写入即可



代码如下：

**rm -rf /root/.jenkins/workspace/jfedu/target/edu.war**

**cp /root/.jenkins/workspace/jfedu/target/edu.war /root/.jenkins/jobs/jfedu/builds/lastSuccessfulBuild/archive/target/**

**ssh [root@121.42.183.93](mailto:root@121.42.183.93) 'bash -x -s' < /data/sh/auto\_deploy.sh**

### 自动部署脚本

#!/bin/bash

#Auto deploy Tomcat for jenkins perf

#by author wugk 2016-07-06

export JAVA\_HOME=/usr/java/jdk1.6.0\_25

TOMCAT\_PID=`/usr/sbin/lsof -n -P -t -i :10013`

TOMCAT\_DIR="/usr/local/tomcat/"

FILES="jfedu.war"

DES\_DIR="/export/data/tomcatRoot/ptest.jd.com/"

DES\_URL="http://123.57.11.203:8080/job/Ptest\_Online/lastSuccessfulBuild/target"

BAK\_DIR="/export/backup/`date +%Y%m%d-%H%M`"

[ -n "$TOMCAT\_PID" ] && kill -9 $TOMCAT\_PID

cd $DES\_DIR

rm -rf $FILES

mkdir -p $BAK\_DIR;\cp -a $DES\_DIR/\* $BAK\_DIR/

rm -rf $DES\_DIR/\*

wget $DES\_URL/$FILES

/usr/java/jdk1.6.0\_25/bin/jar -xvf $FILES

####################

cd $TOMCAT\_DIR;rm -rf work

/bin/sh $TOMCAT\_DIR/bin/start.sh

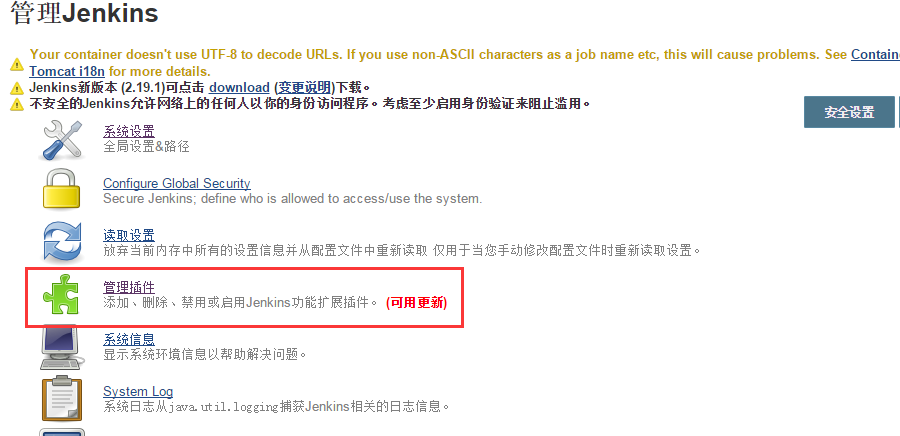
sleep 10

tail -n 50 $TOMCAT\_DIR/logs/catalina.out

提示：如果有多台，怎么去部署，可以发散思维哟！！！可以考虑结合底层saltstack、Ansible工具去实现。

### Jenkins插件安装

在jenkins页面“系统管理”-->“管理插件”-->"可选插件"中搜索email-ext-plugin插件选择并安装，也可以点击“高级”-然后手动上传安装。



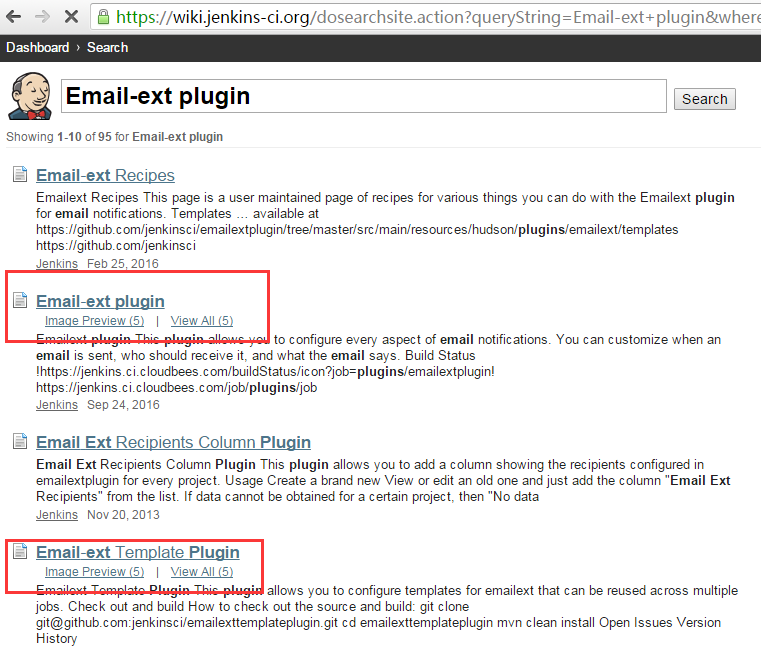
注：也可以手动下载插件安装，到官网下载插件，然后将下载的插件传到服务器jenkins根目录下的plugins目录，即~/.jenkins/plugins目录，重启jenkins即可

官网插件下载地址<https://wiki.jenkins-ci.org/display/JENKINS/Plugins>

如下我们来安装Email-ext-Plugin邮件插件包，安装方法如下：

首先去Jenkins插件官网下载email-ext和token-macro、email-template

可以搜索某个插件，输入插件名称即可：



分别从如下地址下载三个插件，然后上传到Jenkins，实现安装：

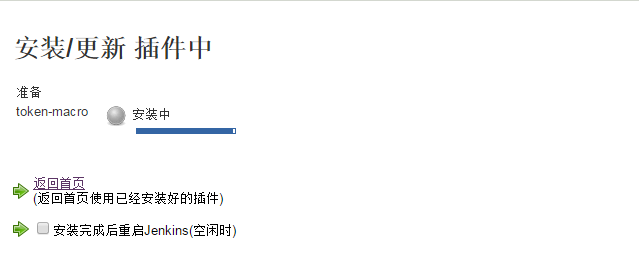
<https://wiki.jenkins-ci.org/display/JENKINS/Email-ext+plugin>

<https://wiki.jenkins-ci.org/display/JENKINS/Token+Macro+Plugin>

<https://wiki.jenkins-ci.org/display/JENKINS/Email-ext+Template+Plugin>

如下图，先安装token插件：

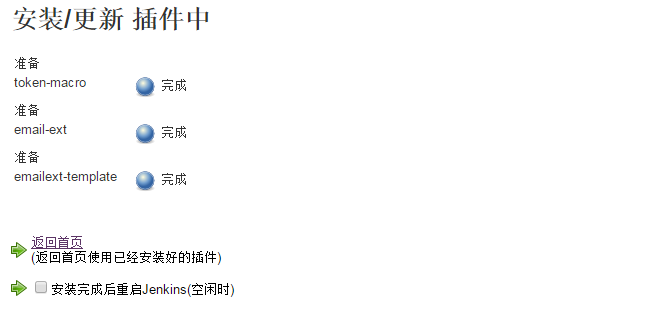




然后安装email插件

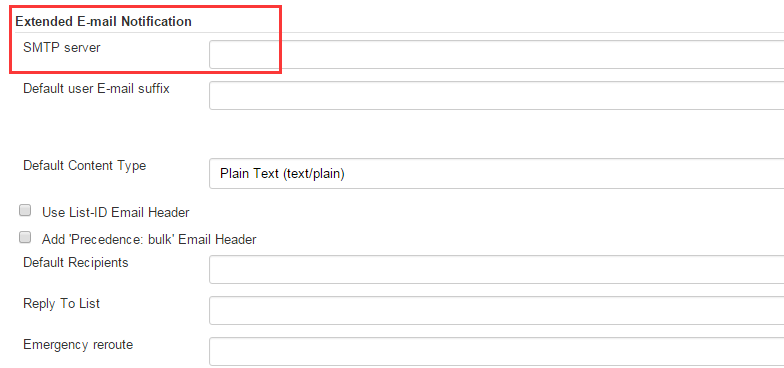


最终如下三个插件安装成功：



安装完成后，刷新Jenkins WEB界面即可。

**安装完email插件，打开系统管理-我们可以看到多了一个extended-E-mail-Notification菜单选项**



如上是安装Jenkins插件的方法，其他GIT插件、Publish Over插件安装方法一样。

### Jenkins邮件设置

Jenkins持续集成,当我们自动打包部署完,我们可以发送一封邮件给相关的负责人,接下来带大家一起来看一下Jenkins持续集成邮件设置,邮件设置分为如下几个步骤:

构建完成后，告知构建信息及结果。主要使用到两个插件：**Email Extension Plugin、Email Extension Template Plugin（上一章，我们已经安装完这3个插件，这里可以忽略不需要再次安装）**

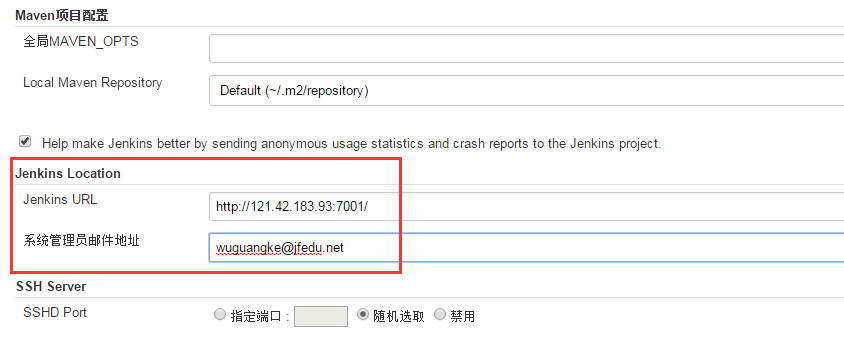
**常见参数如下：**

**SMTP server：设置邮件提供商的SMTP地址，mail.jfedu.net  
Default Content Type：内容展现的格式，一般选择HTML  
Default Recipients：默认收件人  
Use SMTP Authentication：使用SMTP身份验证，点开Default user E-mail suffix下面的高级按钮才能看到，选中。  
User Name：邮件发送账户的用户名。  
Password：邮件发送账户的密码。  
SMTP port：SMTP端口。**

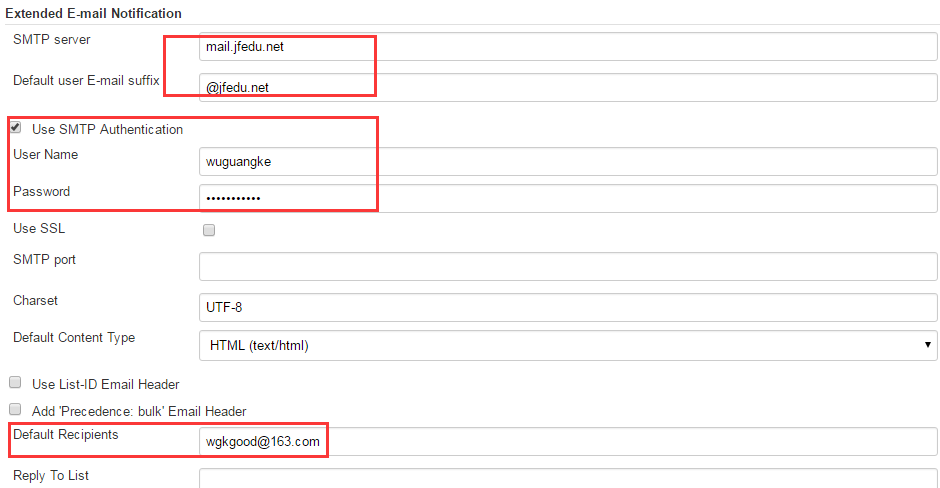
1. **首先要设置Jenkins发送邮件的发送者**



1. **然后选择Jenkins Location设置如下：（添加系统管理员邮件）**



1. **设置发送邮件的SMTP、邮箱后缀，发送类型html、接收者或者抄送者**



1. **设置邮件的标题**

Default Subject内容如下：

构建通知：$PROJECT\_NAME - Build # $BUILD\_NUMBER - $BUILD\_STATUS

1. **设置邮件的内容**

Default Content内容如下：

<hr/>

<h3>(本邮件是程序自动下发的，请勿回复！)</h3><hr/>

项目名称：$PROJECT\_NAME<br/><hr/>

构建编号：$BUILD\_NUMBER<br/><hr/>

构建状态：$BUILD\_STATUS<br/><hr/>

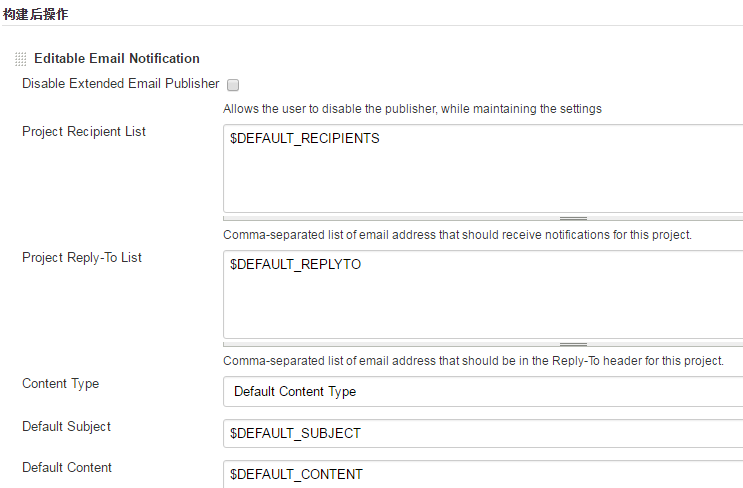
触发原因：${CAUSE}<br/><hr/>

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

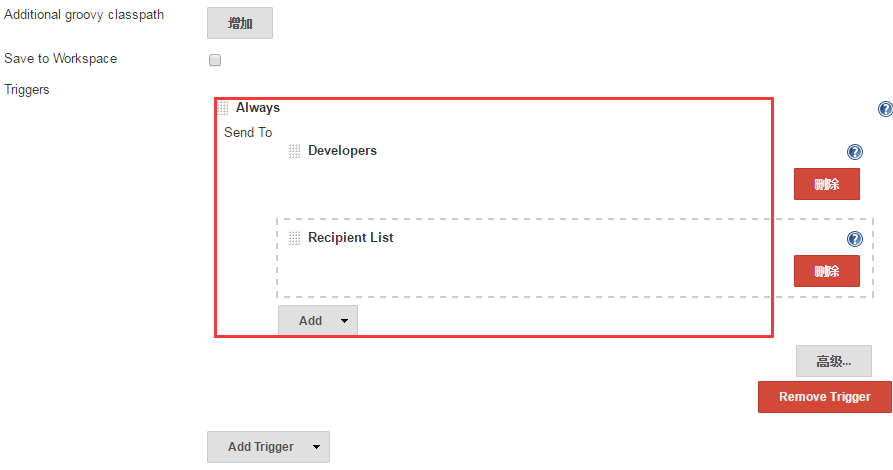
构建地址：<a href="$BUILD\_URL">$BUILD\_URL</a><br/><hr/>

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

1. **工程项目邮件设置**



选择trigger阀值



1. **构建邮件验证**

