# Jenkins是什么？

Jenkins 是一个可扩展的持续集成引擎。

主要用于：

* l 持续、自动地构建/测试软件项目。
* l 监控一些定时执行的任务。

Jenkins拥有的特性包括：

* l 易于安装-只要把jenkins.war部署到servlet容器，不需要数据库支持。
* l 易于配置-所有配置都是通过其提供的web界面实现。
* l 集成RSS/E-mail通过RSS发布构建结果或当构建完成时通过e-mail通知。
* l 生成JUnit/TestNG测试报告。
* l 分布式构建支持Jenkins能够让多台计算机一起构建/测试。
* l 文件识别:Jenkins能够跟踪哪次构建生成哪些jar，哪次构建使用哪个版本的jar等。
* l 插件支持:支持扩展插件，你可以开发适合自己团队使用的工具。

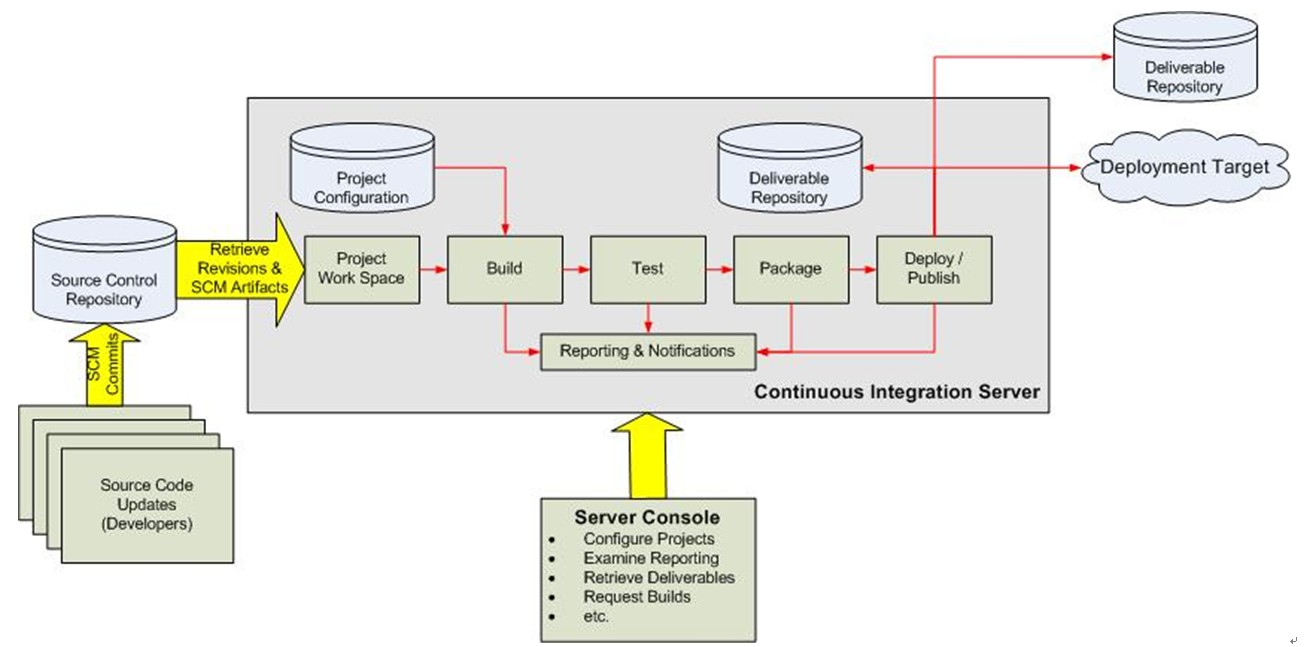
# 1 Jenkins的由来

目前持续集成(CI)已成为当前许多软件开发团队在整个软件开发生命周期内侧重于保证代码质量的常见做法。它是一种实践，旨在缓和和稳固软件的构建过程。并且能够帮助您的开发团队应对如下挑战：

* l 软件构建自动化 ：配置完成后，CI系统会依照预先制定的时间表，或者针对某一特定事件，对目标软件进行构建。
* l 构建可持续的自动化检查 ：CI系统能持续地获取新增或修改后签入的源代码，也就是说，当软件开发团队需要周期性的检查新增或修改后的代码时，CI系统会不断确认这些新代码是否破坏了原有软件的成功构建。这减少了开发者们在检查彼此相互依存的代码中变化情况需要花费的时间和精力(说直接一点也是钱啊，呵呵)。
* l 构建可持续的自动化测试 ：构建检查的扩展部分，构建后执行预先制定的一套测试规则，完成后触发通知(Email,RSS等等)给相关的当事人。
* l 生成后后续过程的自动化 :当自动化检查和测试成功完成，软件构建的周期中可能也需要一些额外的任务，诸如生成文档、打包软件、部署构件到一个运行环境或者软件仓库。这样，构件才能更迅速地提供给用户使用。

部署一个CI系统需要的最低要求是，一个可获取的源代码的仓库，一个包含构建脚本的项目。

下图概括了CI系统的基本结构：



该系统的各个组成部分是按如下顺序来发挥作用的：

1. 开发者检入代码到源代码仓库。

2. CI系统会为每一个项目创建了一个单独的工作区。当预设或请求一次新的构建时，它将把源代码仓库的源码存放到对应的工作区。

3. CI系统会在对应的工作区内执行构建过程。

4. （配置如果存在）构建完成后，CI系统会在一个新的构件中执行定义的一套测试。完成后触发通知(Email,RSS等等)给相关的当事人。

5. （配置如果存在）如果构建成功，这个构件会被打包并转移到一个部署目标(如应用服务器)或存储为软件仓库中的一个新版本。软件仓库可以是CI系统的一部分，也可以是一个外部的仓库，诸如一个文件服务器或者像Java.net、 SourceForge之类的网站。

6. CI系统通常会根据请求发起相应的操作，诸如即时构建、生成报告，或者检索一些构建好的构件。

Jenkins就是这么一个CI系统。之前叫做Hudson。

以下是使用Jenkins的一些理由：

* l 是所有CI产品中在安装和配置上最简单的。
* l 基于Web访问，用户界面非常友好、直观和灵活，在许多情况下，还提供了AJAX的即时反馈。
* l Jenkins是基于Java开发的(如果你是一个Java开发人员，这是非常有用的)，但它不仅限于构建基于Java的软件。
* l Jenkins拥有大量的插件。这些插件极大的扩展了Jenkins的功能；它们都是开源的，而且它们可以直接通过web界面来进行安装与管理。

## 1.1 Jenkins的目标

Jenkins的主要目标是监控软件开发流程，快速显示问题。所以能保证开发人员以及相关人员省时省力提高开发效率。

CI系统在整个开发过程中的主要作用是控制：当系统在代码存储库中探测到修改时，它将运行构建的任务委托给构建过程本身。如果构建失败了，那么CI系统将通知相关人员，然后继续监视存储库。它的角色看起来是被动的；但它确能快速反映问题。

特别是它具有以下优点：

* l Jenkins一切配置都可以在web界面上完成。有些配置如MAVEN\_HOME和Email，只需要配置一次，所有的项目就都能用。当然也可以通过修改XML进行配置。
* l 支持Maven的模块(Module)，Jenkins对Maven做了优化，因此它能自动识别Module，每个Module可以配置成一个job。相当灵活。
* l 测试报告聚合，所有模块的测试报告都被聚合在一起，结果一目了然，使用其他CI，这几乎是件不可能完成的任务。
* l 构件指纹(artifact fingerprint)，每次build的结果构件都被很好的自动管理，无需任何配置就可以方便的浏览下载。

# Jenkins安装与配置

# 2 Jenkins安装

在最简单的情况下，Jenkins 只需要两个步骤：

1.下载最新的版本（一个 WAR 文件）。Jenkins官方网址: [http://Jenkins-ci.org/](http://jenkins-ci.org/)

2.运行 java -jar jenkins.war

注意：Jenkins 需要运行 Java 5以及以上的版本。

还有一种安装方式就是将下载的war包文件部署到 servlet 容器，然后启动容器，在浏览器的URL地址栏中输入类似[http://localhost:8080/jenkins/](http://localhost:8088/hudson/)这样的地址即可。下图是安装成功后的界面（使用的是Linux+Tomcat6+Java6环境）：



图2 Jenkins主界面

# 3 Jenkins配置

写在配置前的一些话：Jenkins的配置不可能全部都说到的，大部分配置是有英文说明的，点击输入框后面的问号就可以看见了。英文不会用翻译工具，连蒙带猜的，多测试几次，你就懂了。

## 3.1 系统管理

在已运行的Jenkins主页中，点击左侧的系统管理进入如下界面：



图3 Jenkins系统管理

### 3.1.1 提示信息

Ps：版本不同提示的消息有可能不同

#### 3.1.1.1 Utf-8编码

Your container doesn't use UTF-8 to decode URLs. If you use non-ASCII characters as a job name etc, this will cause problems. See [Containers](http://wiki.jenkins-ci.org/display/JENKINS/Containers) and [Tomcat i18n](http://wiki.jenkins-ci.org/display/JENKINS/Tomcat#Tomcat-i18n) for more details.

Jenkins建议在tomcat中使用utf-8编码，配置tomcat下conf目录的server.xml文件

http://images.cnitblog.com/blog/529896/201308/12163628-5e18c80f1c44409fb584a8a618299e12.png

图4 URIEncoding编码设置

Ps：如果Job的控制台中文输出乱码，请将URIEncoding=”utf-8”更改为useBodyEncodingForURI="true"

#### 3.1.1.2 新的版本

New version of Jenkins (1.518.JENKINS-14362-jzlib) is available for download (changelog).

提示有新的版本可以下载了,喜欢更新的点击download去下载吧！

#### 3.1.1.3 安全设置

http://images.cnitblog.com/blog/529896/201308/12163634-a71e6cae19ce4f059fb19d845b7f5db7.png

图5 安全提示消息

詹金斯允许网络上的任何人代表您启动进程。考虑至少启用身份验证来阻止滥用。点击Dismiss忽略该消息,点击Setup Security进入设置界面.详细设置请参考 Configure Global Security(安全设置) 章节

### 3.1.2 系统设置

在已运行的Jenkins主页中，点击左侧的系统管理—>系统设置进入如下界面：

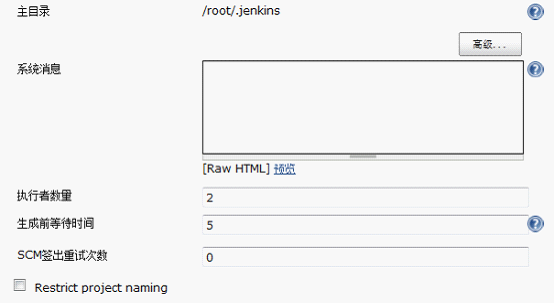


图6 系统设置界面

#### 3.1.2.1 JDK、Maven、Ant配置

配置一个JDK、Ant、Maven实例，请在每一节下面单击Add(新增) 按钮，这里将添加实例的名称和绝对地址。下图描述了这两个部分。



图7 JDK配置界面

JDK别名：给你看的，随便你自己，叫阿猫阿狗都可以

JAVA\_HOME：这个是本机JDK的安装路径（错误的路径会有红字提示你的）

自动安装：不推荐这个选项

后面Ant与Maven的配置是一样的，JDK去oracle官网下载，Ant与Maven去apache官网下载

Ps：每个文本框后面都有个问号，点击问号就会出现帮助信息

#### 3.1.2.2 邮件通知配置

##### 3.1.2.2.1 配置发件人地址



图8 发件人地址配置界面

System Admin e-mail address：Jenkins邮件发送地址，如果你这个没有配置，等着发邮件的时候报错吧，当时我也是这儿没有配置，郁闷了我一周的时间。⊙﹏⊙b汗

##### 3.1.2.2.2 配置邮件通知

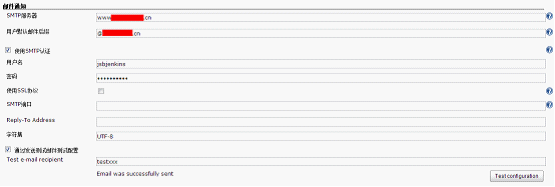


图9 邮件通知

这个就非常的简单了，根据的的邮箱提供者的参数配置就行了。

Ps：小技巧：用户默认邮件后缀配置了后，以后你填写邮件地址只需要@之前的就行了

#### 3.1.2.3 Subversion配置

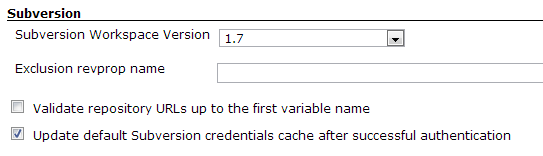


图10 Subversion配置

Subversion Workspace Version：Subversion 的版本号，选择你对应的版本号就行了

### 3.1.3 Configure Global Security(安全设置)

在已运行的Jenkins主页中，点击左侧的系统管理—>Configure Global Security进入如下界面：



图11 安全设置界面

设置如上图，保存后系统管理中就出现管理用户的选项。页面右上角也会出现登录/注册的选项。

### 3.1.4 管理用户设置

在右上角点击注册



图12 注册用户界面

点击sign up按钮，提示你现在已经登录.返回首页.

登录后和匿名账号看到的首页有几点不同，如下图红框所示：



图13 用户登录界面

### 3.1.5 管理插件设置

建议先阅读[Jenkins插件](http://www.cnblogs.com/zz0412/p/jenkins02.html#_Jenkins插件)章节，在回来安装如下所示的插件。这个插件将生成的构件（war或者ear）部署到主流的服务器上。

插件名称：[Deploy Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Deploy+Plugin)

插件介绍：This plugin takes a war/ear file and deploys that to a running remote application server at the end of a build

## 3.2 项目构建设置

### 3.2.1 构建自由风格的Job

#### 3.2.1.1 新建自由风格构建任务

在已运行的Jenkins主页中，点击左侧的新建Job进入如下界面：

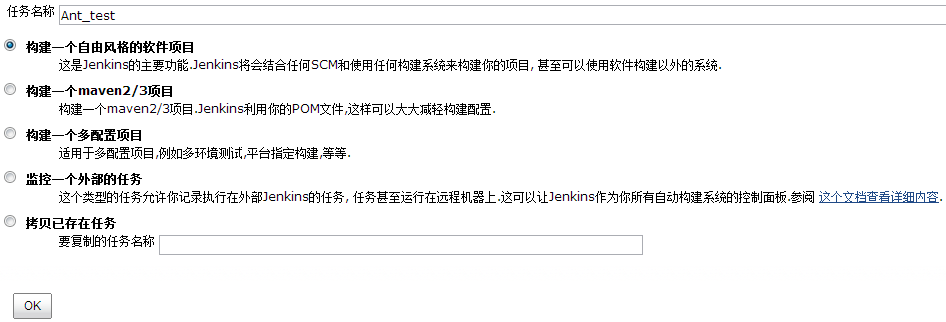


图14 新建Job界面

这时，需要为新的构建任务指定一个名称。（这里输入的任务名称为：Ant\_test）这里有几种的任务类型可供选择，鉴于初步介绍，先选择构建一个自由风格的软件项目。对于其他的类型,经常使用的是拷贝已存在任务;这主要为了能在现有的任务基础上新建任务。点击OK按钮，

#### 3.2.1.2 构建任务配置

##### 3.2.1.2.1 源码管理配置

演示是使用Subversion的链接，在Repository URL中输入你的项目链接，如果没有权限则会提示如下图：

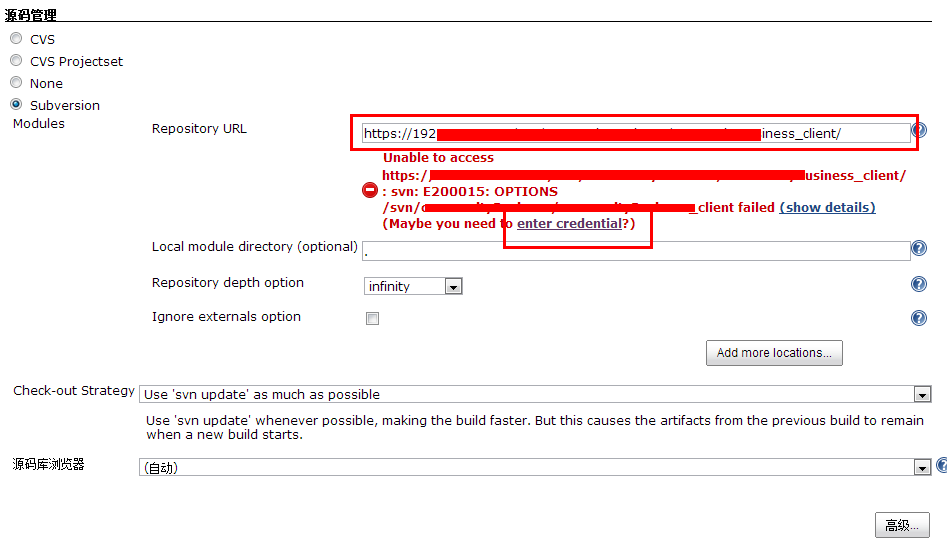


图15 Subversion配置界面

点击 enter credential 输入用户名和密码(我猜大家一般都是使用的用户名和密码登陆的)

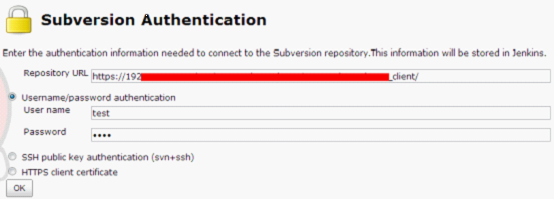


图16 Subversion权限认证界面

Ps：svn的用户名和密码设置了是没有办法在web界面修改的。如果要修改则先去Jenkins目录删除hudson.scm.SubversionSCM.xml文件（点到为止，毕竟这只是入门教程）

##### 3.2.1.2.2 构建触发器

在其他项目构建完成后才执行构建：指定的项目完成构建后，触发此项目的构建。

Poll SCM ：这是CI 系统中常见的选项。当您选择此选项，您可以指定一个定时作业表达式来定义Jenkins每隔多久检查一下您源代码仓库的变化。如果发现变化，就执行一次构建。例如，表达式中填写0,15,30,45 \* \* \* \*将使Jenkins每隔15分钟就检查一次您源码仓库的变化。

Build periodically ：此选项仅仅通知Jenkins按指定的频率对项目进行构建，而不管SCM是否有变化。如果想在这个Job中运行一些测试用例的话，它就很有帮助。

##### 3.2.1.2.3 Ant构建配置

因为我的项目是用ant脚本实现的编译和打包，所以我选择的是Invoke Ant，Ant Version选择你Ant配置的那个名字，注意不要选择default喔，那个选择了没有用。

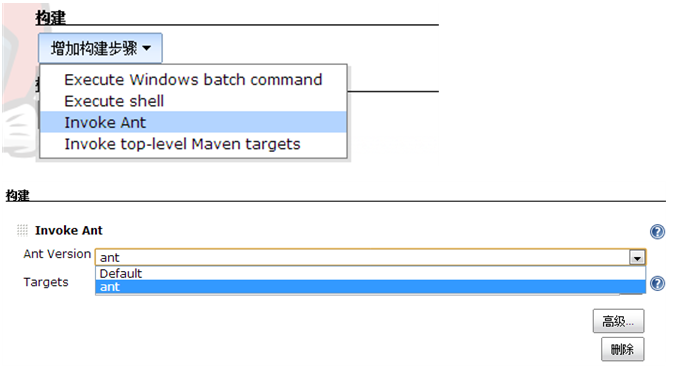


图17 Ant构建配置界面

如果你的构建脚本build.xml不在workspace根目录、或者说你的构建脚本不叫build.xml。那么需要在高级里设置Build File选项的路径，指明你的脚本。注意：是相对路径

部署请参考：[war文件部署](http://www.cnblogs.com/zz0412/p/jenkins02.html#_War文件部署)章节

### 3.2.2 构建Maven风格的Job

#### 3.2.2.1 新建Maven构建任务

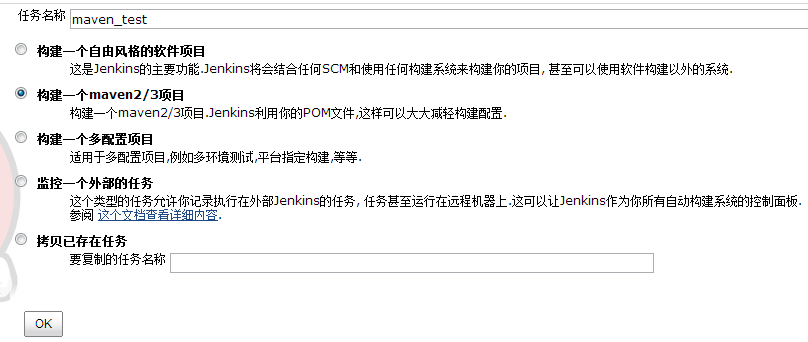


图18 新建Job界面

这时，需要为新的构建任务指定一个名称。（这里输入的任务名称为：maven\_test）这里有几种的任务类型可供选择，鉴于初步介绍，先选择构建一个maven2/3项目。对于其他的类型,经常使用的是拷贝已存在任务;这主要为了能在现有的任务基础上新建任务。点击OK按钮，

#### 3.2.2.2 构建任务配置



图19 构建任务配置界面

##### 3.2.2.2.1 源码管理配置

演示是使用Subversion的链接，在Repository URL中输入你的项目链接，如果没有权限则会提示如下图：

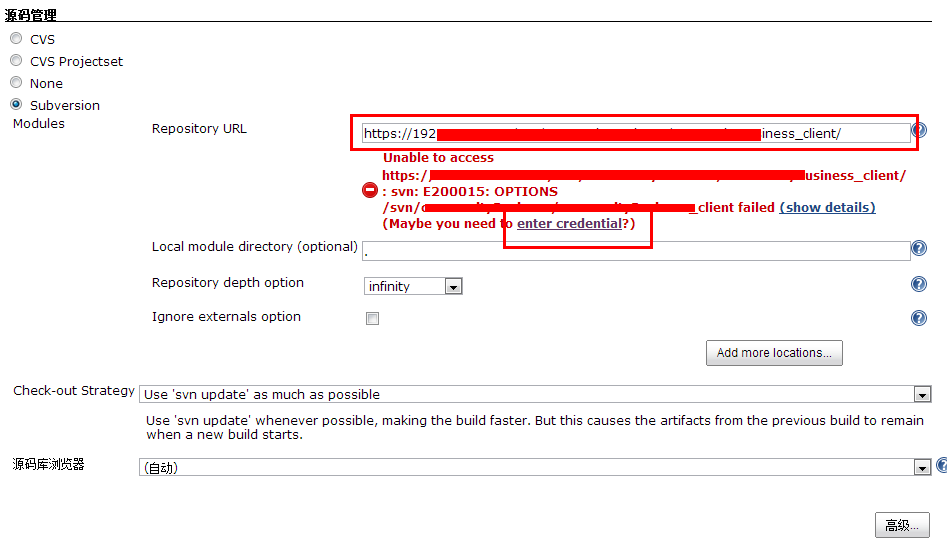


图20 Subversion配置界面

点击 enter credential 输入用户名和密码(我猜大家一般都是使用的用户名和密码登陆的)

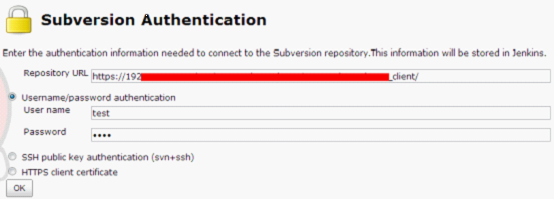


图21 Subversion权限认证界面

Ps：svn的用户名和密码设置了是没有办法在web界面修改的。如果要修改则先去Jenkins目录删除hudson.scm.SubversionSCM.xml文件（点到为止，毕竟这只是入门教程）

##### 3.2.2.2.2 构建触发器

在其他项目构建完成后才执行构建：指定的项目完成构建后，触发此项目的构建。

Poll SCM ：这是CI 系统中常见的选项。当您选择此选项，您可以指定一个定时作业表达式来定义Jenkins每隔多久检查一下您源代码仓库的变化。如果发现变化，就执行一次构建。例如，表达式中填写0,15,30,45 \* \* \* \*将使Jenkins每隔15分钟就检查一次您源码仓库的变化。

Build periodically ：此选项仅仅通知Jenkins按指定的频率对项目进行构建，而不管SCM是否有变化。如果想在这个Job中运行一些测试用例的话，它就很有帮助。

##### 3.2.2.2.3 Maven构建设置

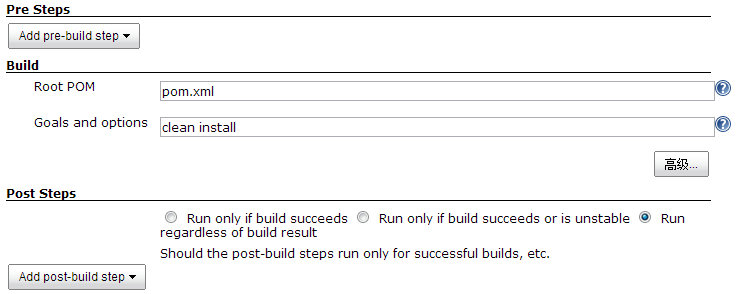


图22 Maven构建配置界面

2013-08-22补充Goals and options ：clean install -Dmaven.test.skip=true　　#加入了跳过测试的代码

Root POM:填写你项目的pom.xml文件的位置，注意：是相对位置，如果该文件不存在，会有红色字提示。

部署请参考：[war文件部署](http://www.cnblogs.com/zz0412/p/jenkins02.html#_War文件部署)章节

##### 3.2.2.2.4 构建maven项目的心得

使用Jenkins构建maven项目的一点小心得：

maven项目的构建是比较麻烦的，如果你的项目是下图这种结构。那么恭喜你！你新建一个job就可以了，因为只有一个根。如果你的svn地址是：https://192.xxx/Pe\_Project/root-pom，那么Root POM只需要保持默认就行了，因为Jenkins可以再workspace目录下面找到pom.xml文件

如果你的svn地址是：https://192.xxx/Pe\_Project，那么Root POM需要指定为root-pom/pom.xml，因为Jenkins可以再workspace/root-pom目录下面找到pom.xml文件

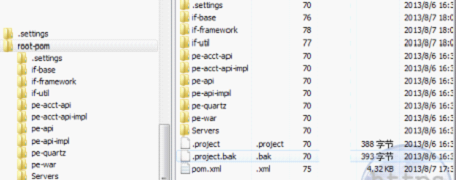


图23 Maven项目结构界面1

上面这种方法打包的时候非常简单，但是用eclipse开发的时候你就不右键run as —>tomca启动了，如果你想使用这种方式，将tomcat换成jetty即可。

如果你的项目是下图这种结构，那么非常悲剧的告诉你，你要建立好几个job来构建这一个项目，因为这个项目有4个根。

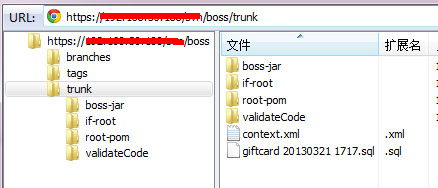


图24 Maven项目结构界面2

上面这种方法打包的时候比较麻烦，但是用eclipse开发的时候你就可以使用右键run as —>tomca启动了

### 3.2.3 邮件通知设置

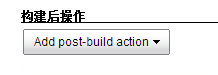


图25 构建后操作界面

选择Add post-build action，然后选择E-mail Notification，如下图：

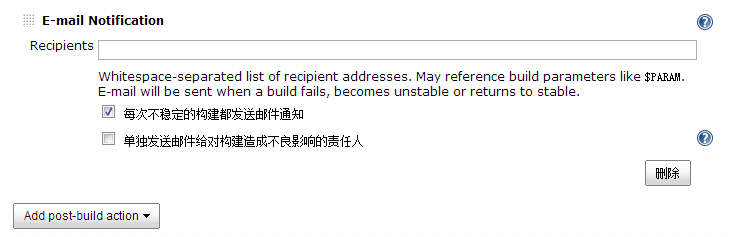


图26 收件人列表界面

在Recipients中输入收件人邮件地址，如果用多个收件人用“,”英文逗号隔开

### 3.2.4 War文件部署设置

首先你必须安装好[Deploy Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Deploy+Plugin)插件，然后在tomcat的conf目录配置tomcat-users.xml文件，在<tomcat-users>节点里添加如下内容：

<role rolename="manager-gui"/>

<role rolename="manager-script"/>

<role rolename="manager-jmx"/>

<role rolename="manager-status"/>

<user username="username" password="password" roles="manager-gui,manager-script,manager-jmx,manager-status"/>

引号里的username和password可以随便替换，待会要用的。

好了，回到Jenkins项目配置页面：

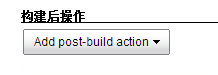


图27 构建后操作界面

选择Add post-build action，然后选择Deploy war/ear to a container，如下图：

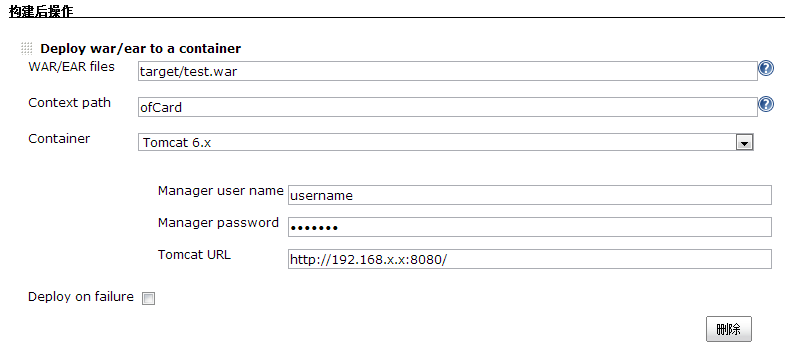


图28 远程部署配置界面

WAR/EAR files：war文件的存放位置，如：target/test.war 注意：相对路径，target前是没有/的。

Context path：访问时需要输入的内容，如ofCard访问时如下：http://192.168.x.x:8080/ofCard/如果为空，默认是war包的名字。

Container：选择你的web容器，如tomca 6.x

Manager user name：填入tomcat-users.xml配置的username内容

Manager password：填入tomcat-users.xml配置的password内容

Tomcat URL：填入http://192.168.x.x:8080/

Deploy on failure：构建失败依然部署，一般不选择

注意：虽然这种部署方法可能会导致tomcat加载时出现卡死的现象。但是也是最简单的部署方式。如果卡死了重启下就好了，将tomcat的java内存参数调高可以解决这个问题。

最后不要忘记点击保存喔。

好了！到此一个项目的获取源码，打包，远程部署，邮件通知就完成了。

## 3.3 监控

当任务一旦运行，您将会看到这个任务正在队列中的仪表板和当前工作主页上运行。这两种显示如下。

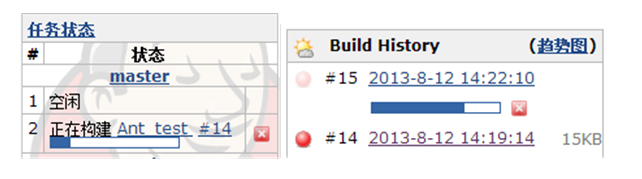


图29 主页监控（左），项目监控（右）

一旦构建完成后，完成后的任务将会有三个地方进行显示。

你可以在Jenkins的控制面板上看到它，如下图。

http://images.cnitblog.com/blog/529896/201308/12164320-7fa46061ee2147d3a66655ce878609ed.png

图30 主页项目界面

在上面展示的截图中，您将注意到有两个图标描述当前作业的状态。S栏目代表着“最新构建状态”，W栏目代表着“构建稳定性”。Jenkins使用这两个概念来介绍一个作业的总体状况：

构建状态:下图中分级符号概述了一个Job新近一次构建会产生的四种可能的状态：

Successful:完成构建，且被认为是稳定的。

Unstable:完成构建，但被认为不稳定。

Failed:构建失败。

Disabled:构建已禁用。

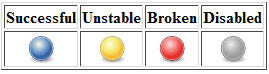


图31 构建状态界面

构建稳定性: 当一个Job中构建已完成并生成了一个未发布的目标构件，如果您准备评估此次构建的稳定性，Jenkins会基于一些后处理器任务为构建发布一个稳健指数 (从0-100 )，这些任务一般以插件的方式实现。它们可能包括单元测试(JUnit)、覆盖率(Cobertura )和静态代码分析(FindBugs)。分数越高，表明构建越稳定。下图中分级符号概述了稳定性的评分范围。任何构建作业的状态(总分100)低于80分就是不稳定的。

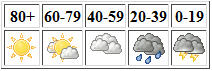


图32 构建稳定性界面

你也可以在当前Job主界面上看到它，如下图左下部分



图33 项目主界面

当前作业主页上还包含了一些有趣的条目。左侧栏的链接主要控制Job的配置、删除作业、构建作业。右边部分的链接指向最新的项目报告和构件。

通过点击构建历史（Build History）中某个具体的构建链接，您就能跳转到Jenkins为这个构建实例而创建的构建主页上。如下图



图34 构建历史界面

如果你想通过视图输出界面来监控当前任务的进展情况。你可以单击Console Output（控制台输出）。如果工作已完成，这将显示构建脚本产生的静态输出；如果作业仍然在运行中，Jenkins将不断刷新网页的内容，以便您可以看到它运行时的输出。如下图：



图35 控制台输出界面

# 4 Jenkins插件

从Jenkins现有的功能扩展或开发者们为Jenkins提供的新功能都可以称之为Jenkins插件。有些插件可以无缝添加到您的构建过程，而其它，诸如除CVS和Subversion的SCM插件则需要源代码控制系统的支持。

## 4.1 Jenkins插件安装

Jenkins 插件管理器允许您安装新的插件，和更新您Jenkins服务器上的插件。管理者将连接到联机资料库，检索可用的和已更新的插件。如果您的Jenkins服务器 无法直接连接到外部资源，您可以从Jenkins网站上下载。

在已运行的Jenkins主页中，点击左侧的系统管理—>管理插件进入如下界面：

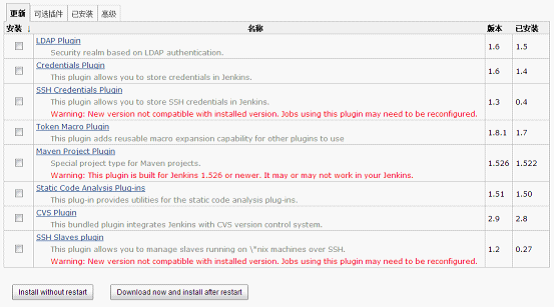


图36 插件管理界面

它包含四个标签：

更新:清单中列示了Jenkins为某些插件搜索到了可用的更新。列出的每个插件可以被选择并应用更新。

可选安装:清单中列示了可用于安装（而不是目前已安装的）的所有插件。列出的每个插件都可以被选择并安装。

已安装:清单中列示了已经安装的插件。

高级:允许您通过设定HTTP代理的方式使Jenkins与在线插件库建立连接。此外，还提供了一个上传设备，可以安装你在Jenkins以外已下载的那些插件。

由上图可知，Jenkins缺省集成了maven2插件，并且一旦插件有新版本，会提示更新新版本插件。

如果想安装新的插件，可以点击tab分页中的可选插件。如下图：

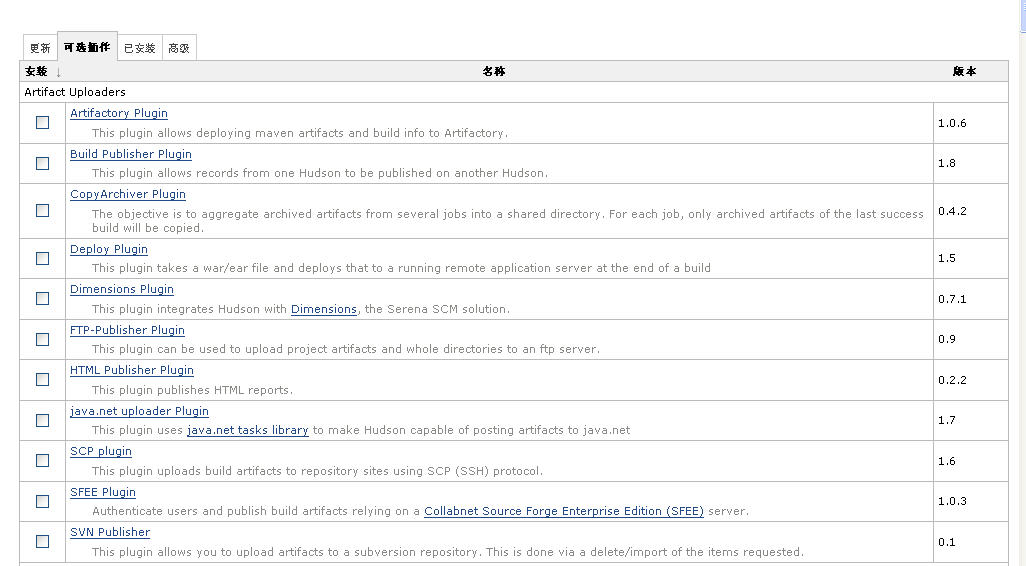


图37 可选插件界面

从图可知，各种Jenkins插件根据之前所记述的类型进行分门别类。可勾选任意想安装的Jenkins插件，点击Install without restart按钮进行安装。安装后，所有插件以hpi作为后缀名放置在plugins文件夹下。如果是高级用户还可以自行开发插件方便具体项目使用。

注意：安装完成后需要重启Jenkins部署的容器。这样才能使用新装的插件。

## 4.2 Jenkins插件安装示例

Jenkins运行自动部署war包到servlet容器内，要实现这个功能必须安装一个插件。



图38 安装插件界面



图39 插件安装界面

好了，到此[Deploy Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Deploy+Plugin)插件安装完成！

最后，如果感觉我太啰嗦了去看这篇吧：

**使用email-ext替换Jenkins的默认邮件通知**

# 1 简述

众所周知，Jenkins默认提供了一个邮件通知，能在构建失败、构建不稳定等状态后发送邮件。但是它本身有很多局限性，比如它的邮件通知无法提供详细的邮件内容、无法定义发送邮件的格式、无法定义灵活的邮件接收配置等等。在这样的情况下，我们找到了[Jenkins Email Extension Plugin](http://wiki.jenkins-ci.org/display/JENKINS/Email-ext+plugin)。该插件能允许你自定义邮件通知的方方面面，比如在发送邮件时你可以自定义发送给谁，发送具体什么内容等等。本文不会告诉你如何安装该插件，关于插件的安装请参考[这里](http://www.cnblogs.com/zz0412/p/jenkins02.html)。

# 2 配置

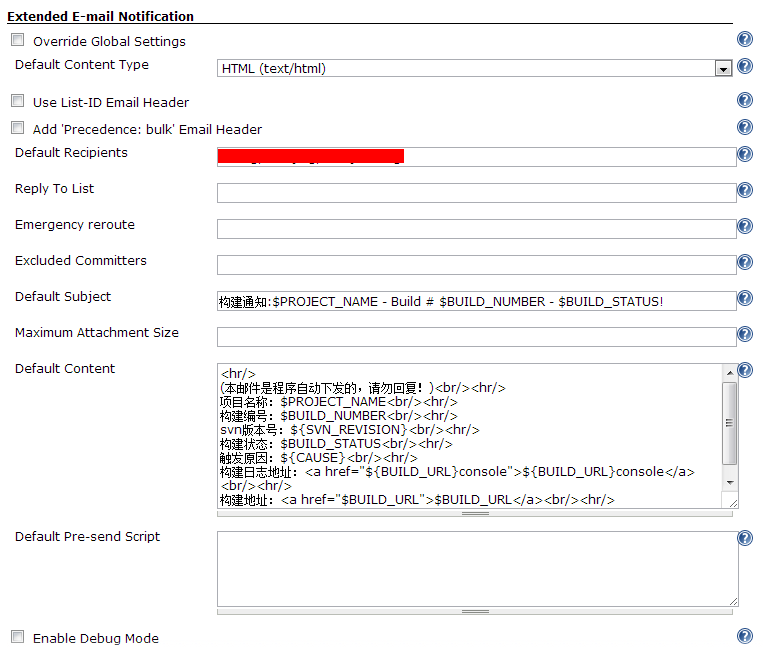
它主要包含两个部分：全局配置和项目配置。

## 2.1 全局配置

当然，在一个项目中应用email-ext插件之前，您必须做一些全局的配置。现在先跳转到Jenkins的“系统设置”页面，如下图：



找到标题为“Extended E-mail Notification”的片段，你就能配置一些全局的email-ext属性。这些属性必须匹配你SMTP邮件服务器的设置。这一节不仅能配置成Jenkins原有邮件通知的镜像(虽然有很多配置是一样的，但这是个不同的扩展点)，而且还增加了一些额外的功能。输入框中名为 Default Subject 和 Default Content 的项允许你在全局级别配置邮件的内容。这样做的话，可以使您为所有的项目按您的需求做更好的、更简单的配置。如下图。



释放个我的配置：

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

Default Content：

<hr/>

(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/>

构建编号：$BUILD\_NUMBER<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/>

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

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

下面解释一下常用的属性。

### 2.1.1 全局属性详解

1. Override Global Settings：如果不选，该插件将使用默认的E-mail Notification通知选项。反之，您可以通过指定不同于( 默认选项)的设置来进行覆盖。

2. Default Content Type：指定构建后发送邮件内容的类型，有Text和HTML两种.

3. Use List-ID Email Header：为所有的邮件设置一个List-ID的邮件信头，这样你就可以在邮件客户端使用过滤。它也能阻止邮件发件人大部分的自动回复(诸如离开办公室、休假等等)。你可以使用你习惯的任何名称或者ID号，但是他们必须符合如下其中一种格式(真实的ID必须要包含在<和>标记里)：  
<ci-notifications.company.org>  
Build Notifications <ci-notifications.company.org>  
“Build Notifications” <ci-notifications.company.org>  
关于更详细的List-ID说明请参阅[RFC-2919](http://www.apps.ietf.org/rfc/rfc2919.html).

4. Add 'Precedence: bulk' Email Header：设置优先级,更详细说明请参阅[RFC-3834](http://tools.ietf.org/search/rfc3834).

5. Default Recipients：自定义默认电子邮件收件人列表。如果没有被项目配置覆盖,该插件会使用这个列表。您可以在项目配置使用$ DEFAULT\_RECIPIENTS参数包括此默认列表，以及添加新的地址在项目级别。添加抄送：cc:电子邮件地址例如,CC:someone@somewhere.com

6. Reply To List：回复列表, A comma separated list of e-mail addresses to use in the Reply-To header of the email. This value will be available as $DEFAULT\_REPLYTO in the project configuration.

7. Emergency reroute：如果这个字段不为空，所有的电子邮件将被单独发送到该地址（或地址列表）。

8. Excluded Committers：防止邮件被邮件系统认为是垃圾邮件,邮件列表应该没有扩展的账户名(如:@domain.com),并且使用逗号分隔

9. Default Subject：自定义邮件通知的默认主题名称。该选项能在邮件的主题字段中替换一些参数，这样你就可以在构建中包含指定的输出信息。

10. Maximum Attachment Size：邮件最大附件大小。

11. Default Content：自定义邮件通知的默认内容主体。该选项能在邮件的内容中替换一些参数，这样你就可以在构建中包含指定的输出信息。

12. Default Pre-send Script：默认发送前执行的脚本（注：grooy脚本，这是我在某篇文章上看到的，不一定准确）。

13. Enable Debug Mode：启用插件的调试模式。这将增加额外的日志输出，构建日志以及Jenkins的日志。在调试时是有用的，但不能用于生产。

14. Enable Security：启用时，会禁用发送脚本的能力，直接进入Jenkins实例。如果用户试图访问Jenkins管理对象实例，将抛出一个安全异常。

15. Content Token Reference：邮件中可以使用的变量，所有的变量都是可选的。具体介绍请查看[全局邮件变量](http://www.cnblogs.com/zz0412/p/jenkins_jj_01.html#_全局邮件变量)章节。

### 2.1.2 全局邮件变量

Ps：看着感觉有点晕头，对比着Jenkins页面看要好些。

email-ext插件允许使用变量来动态插入数据到邮件的主题和内容主体中。变量是一个以$(美元符号)开始，并以空格结束的字符串。当一个邮件触发时，主题和内容主体字段的所有变量都会通过真实的值动态地替换。同样，变量中的“值”能包含其它的变量，都将被替换成真实的内容。

比如，项目配置页的默认主题和内容分别对应的是全局配置页面的DEFAULT\_SUBJECT和DEFAULT\_CONTENT，因此它会自动地使用全局的配置。同理，触发器中的Subject和Content分别对应的是项目配置页面的DEFAULT\_SUBJECT和DEFAULT\_CONTENT，所以它也会自动地使用项目的配置。由于变量中的“值”能包含其它的变量，所以就能为变量快速地创建不同的切入点：全局级别(所有项目)，专属级别(单一项目)，触发器级别(构建结果)。

如果你要查看所有可用的变量，你可以点击配置页的Content Token Reference的问号获取详细的信息。

所有的变量都是可选的，每个变量可以如下表示，字符串类型使用name=“value”，而布尔型和数字型使用name=value。如果{和}标记里面没有变量，则不会被解析。示例：$TOKEN,${TOKEN},${TOKEN,count=100},${ENV,var=”PATH”}

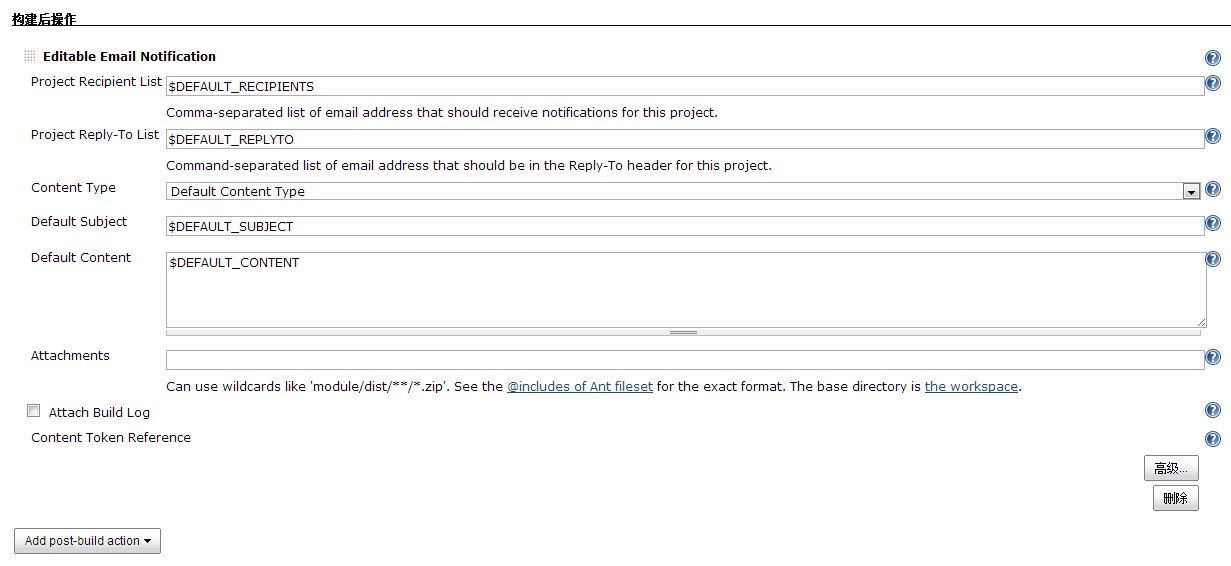
提示：用英文逗号分隔变量的参数。

下面我解释一下常用的属性。

* ${FILE,path="PATH"} 包括指定文件（路径）的含量相对于工作空间根目录。
  + path文件路径，注意：是工作区目录的相对路径。
* ${BUILD\_NUMBER} 显示当前构建的编号。
* ${JOB\_DESCRIPTION} 显示项目描述。
* ${SVN\_REVISION} 显示svn版本号。还支持Subversion插件出口的SVN\_REVISION\_n版本。
* ${CAUSE} 显示谁、通过什么渠道触发这次构建。
* ${CHANGES } -显示上一次构建之后的变化。
  + showPaths 如果为 true,显示提交修改后的地址。默认false。
  + showDependencies 如果为true，显示项目构建依赖。默认为false
  + format 遍历提交信息，一个包含%X的字符串，其中%a表示作者，%d表示日期，%m表示消息，%p表示路径，%r表示版本。注意，并不是所有的版本系统都支持%d和%r。如果指定showPaths将被忽略。默认“[%a] %m\\n”。
  + pathFormat 一个包含“%p”的字符串，用来标示怎么打印路径。
* ${BUILD\_ID}显示当前构建生成的ID。
* ${PROJECT\_NAME} 显示项目的全名。（见AbstractProject.getFullDisplayName）
* ${PROJECT\_DISPLAY\_NAME} 显示项目的显示名称。（见AbstractProject.getDisplayName）
* ${SCRIPT} 从一个脚本生成自定义消息内容。自定义脚本应该放在"$JENKINS\_HOME/email-templates"。当使用自定义脚本时会默认搜索$JENKINS\_HOME/email-templatesdirectory目录。其他的目录将不会被搜索。
  + script 当其使用的时候，仅仅只有最后一个值会被脚本使用（不能同时使用script和template）。
  + template常规的simpletemplateengine格式模板。
* ${JENKINS\_URL} 显示Jenkins服务器的url地址（你可以再系统配置页更改）。
* ${BUILD\_LOG\_MULTILINE\_REGEX}按正则表达式匹配并显示构建日志。
  + regex [java.util.regex.Pattern](http://download.oracle.com/javase/6/docs/api/java/util/regex/Pattern.html) 生成正则表达式匹配的构建日志。无默认值，可为空。
  + maxMatches 匹配的最大数量。如果为0，将匹配所有。默认为0。
  + showTruncatedLines 如果为true，包含[...truncated ### lines...]行。默认为true。
  + substText 如果非空，就把这部分文字（而不是整行）插入该邮件。默认为空。
  + escapeHtml 如果为true，格式化HTML。默认为false。
  + matchedSegmentHtmlStyle 如果非空，输出HTML。匹配的行数将变为<b style=”your-style-value”> html escaped matched line </b>格式。默认为空。
* ${BUILD\_LOG} 显示最终构建日志。
  + maxLines 日志最多显示的行数，默认250行。
  + escapeHtml 如果为true，格式化HTML。默认false。
* ${PROJECT\_URL} 显示项目的URL地址。
* ${BUILD\_STATUS} -显示当前构建的状态(失败、成功等等)
* ${BUILD\_URL} -显示当前构建的URL地址。
* ${CHANGES\_SINCE\_LAST\_SUCCESS} -显示上一次成功构建之后的变化。
  + reverse在顶部标示新近的构建。默认false。
  + format遍历构建信息，一个包含%X的字符串，其中%c为所有的改变，%n为构建编号。默认”Changes for Build #%n\n%c\n”。
  + showPaths,changesFormat,pathFormat分别定义如${CHANGES}的showPaths、format和pathFormat参数。
* ${CHANGES\_SINCE\_LAST\_UNSTABLE} -显示显示上一次不稳固或者成功的构建之后的变化。
  + reverse在顶部标示新近的构建。默认false。
  + format遍历构建信息，一个包含%X的字符串，其中%c为所有的改变，%n为构建编号。默认”Changes for Build #%n\n%c\n”。
  + showPaths,changesFormat,pathFormat分别定义如${CHANGES}的showPaths、format和pathFormat参数。
* ${ENV} –显示一个环境变量。
  + var– 显示该环境变量的名称。如果为空，显示所有，默认为空。
* ${FAILED\_TESTS} -如果有失败的测试，显示这些失败的单元测试信息。
* ${JENKINS\_URL} -显示Jenkins服务器的地址。(你能在“系统配置”页改变它)。
* ${HUDSON\_URL} -不推荐，请使用$JENKINS\_URL
* ${PROJECT\_URL} -显示项目的URL。
* ${SVN\_REVISION} -显示SVN的版本号。
* ${JELLY\_SCRIPT} -从一个Jelly脚本模板中自定义消息内容。有两种模板可供配置：HTML和TEXT。你可以在$JENKINS\_HOME/email-templates下自定义替换它。当使用自动义模板时，”template”参数的名称不包含“.jelly”。
  + template模板名称，默认”html”。
* ${TEST\_COUNTS} -显示测试的数量。
  + var– 默认“total”。
    - total -所有测试的数量。
    - fail -失败测试的数量。
    - skip -跳过测试的数量。

## 2.2 项目配置

要想在一个项目中使用email-ext插件，你首先必须在项目配置页激活它。在构建后操作——”Add Post-build Actions”选项中勾选”Editable Email Notification”标签。



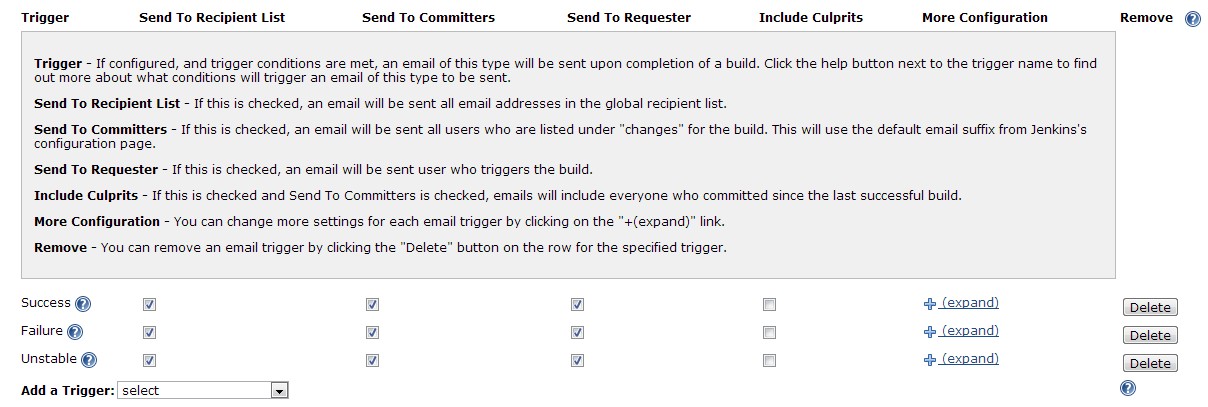
### 2.2.1 项目基本配置

当插件激活后你就能编辑如下字段（只列出常用的字段）：

* Project Recipient List：这是一个以逗号(或者空格)分隔的收件人邮件的邮箱地址列表。允许您为每封邮件指定单独的列表。Ps：如果你想在默认收件人的基础上添加收件人：$DEFAULT\_RECIPIENTS,<新的收件人>
* Default Subject：允许你配置此项目邮件的主题。
* Default Content：跟Default Subject的作用一样，但是是替换邮件内容。
* Attach Build Log：附件构建日志。
  + Compress Build Log before sending：发送前压缩生成日志（zip格式）。

#### 2.2.1.1 项目高级配置

要查看插件的高级配置，请点击”高级”按钮。该选项允许您各种类型的邮件触发器指定接收者。默认情况下，是没有配置的触发器，所以默认情况下不会发送邮件。要增加更多的触发器，选择“Add a Trigger”旁边下拉列表中的类型，它会增加到控件上面的列表中。一旦你增加了一个触发器，你就可以对它做一些选择。如果你点击一个触发器旁边的”?”号，它将告诉你在什么条件下会触发邮件发送。如下图。



* Send to Recipient List：如果勾选，邮件将发送到”Project Recipient List”中的所有邮件地址。
* Send to Committers：该邮件会发给上次构建时检查过代码的人员，该插件会基于提交者的ID和追加Jenkins配置页面的(default email suffix)默认邮件后缀来生成一个邮件地址。譬如，上次提交代码的人是”first.last”， 默认的电子邮件后缀为“@somewhere.com”，那么电子邮件将被发送到“first.last@ somewhere.com”。
* Send To Requester：如果勾选，邮件将发送给构建触发者。
* Include Culprits：如果勾选，而且 “Send To Committers”勾选，邮件将包含最后成功构建的提交者。
* More Configuration：通过单击”+(expand)”链接您能为每个邮件触发器作更多单独的设置。
  + Recipient List：这是一个以逗号(或者空格)分隔的可接受邮件的邮箱地址列表。如果触发就发送邮件到该列表。该列表会追加在”Global Recipient List”里。
  + Subject：指定选择邮件的主题。注意：高级选项中的邮件触发器类型可覆盖对它的配置。
  + Content：指定选择邮件的内容主体。注意：高级选项中的邮件触发器类型可覆盖对它的配置。
* Remove通过单击指定触发器当前行的”Delete”按钮，你可以删除该触发器。

#### 2.2.1.2 触发器类型

注意：所有的触发器都只能配置一次。

* Failure：即时发送构建失败的邮件。如果”Still Failing”触发器已配置，而上一次构建的状态是”Failure”，那么”Still Failing”触发器将发送一封邮件来替代(它)。
* Unstable：即时发送构建不稳固的邮件。如果”Still Unstable”触发器已配置，而上一次构建的状态是”Unstable”，那么”Still Unstable”触发器将发送一封邮件来替代(它)。
* Still Failing：如果两次或两次以上连续构建的状态为”Failure”，发送该邮件。
* Success：如果构建的状态为”Successful”发送邮件。如果”Fixed”已配置，而上次构建的状态为“Failure”或“Unstable”，那么”Fixed”触发器将发送一封邮件来替代(它)。
* Fixed：当构建状态从“Failure”或“Unstable”变为”Successful”时发送邮件。
* Still Unstable：如果两次或两次以上连续构建的状态为” Unstable “，发送该邮件。
* Before Build：当构建开始时发送邮件。

#### 2.2.1.3 项目邮件变量

注意：这里只解释全局配置页面中缺少的变量。

* ${DEFAULT\_SUBJECT}：这是Jenkins系统配置页面默认配置的邮件主题
* ${DEFAULT\_CONTENT}：这是Jenkins系统配置页面默认配置的邮件内容主体
* ${PROJECT\_DEFAULT\_SUBJECT}：这是项目的默认邮件主题。高级配置中使用该令牌的结果要优先于Default Subject字段。警告：不要在Default Subject 或者Default Content中使用该令牌，它会产生一个未知的结果。
* ${PROJECT\_DEFAULT\_CONTENT}：这是项目的默认邮件内容主体。高级配置中使用该令牌的结果要优先于Default Content字段。警告：不要在Default Subject 或者Default Content中使用该令牌，它会产生一个未知的结果。

# 3 Jelly脚本

从Jenkins 2.9版本开始我们可以使用Jelly脚本。Jelly脚本跟Hudson的API挂钩，能获得你想要的任何信息，所以它很强大。插件有两个打包后的Jelly脚本，当然你也可以自定义(脚本)。

关于插件中默认的两个Jelly脚本：一个用来设计HTML格式邮件，另一个则是定义TEXT格式邮件。你能通过使用模板参数指定插件调用哪一个脚本。它们的使用方法如下：

文本格式：${JELLY\_SCRIPT,template=”text”}

HTML格式:${JELLY\_SCRIPT,template=”html”}

你也能编写属于自己的Jelly脚本。Jelly脚本能跟Jenkins 的API(包括[hudson.model.AbstractBuild](http://jenkins-ci.org/javadoc/hudson/model/AbstractBuild.html)和[hudson.model.AbstractProject](http://jenkins-ci.org/javadoc/hudson/model/AbstractProject.html))挂钩，因而特别强大。如果你打算这么做，你可以先参考现有的[html](http://fisheye.jenkins-ci.org/browse/Hudson/trunk/hudson/plugins/email-ext/src/main/resources/hudson/plugins/emailext/templates/html.jelly?r=HEAD)和[text](http://fisheye.jenkins-ci.org/browse/Hudson/trunk/hudson/plugins/email-ext/src/main/resources/hudson/plugins/emailext/templates/text.jelly?r=HEAD)脚本一探究竟。

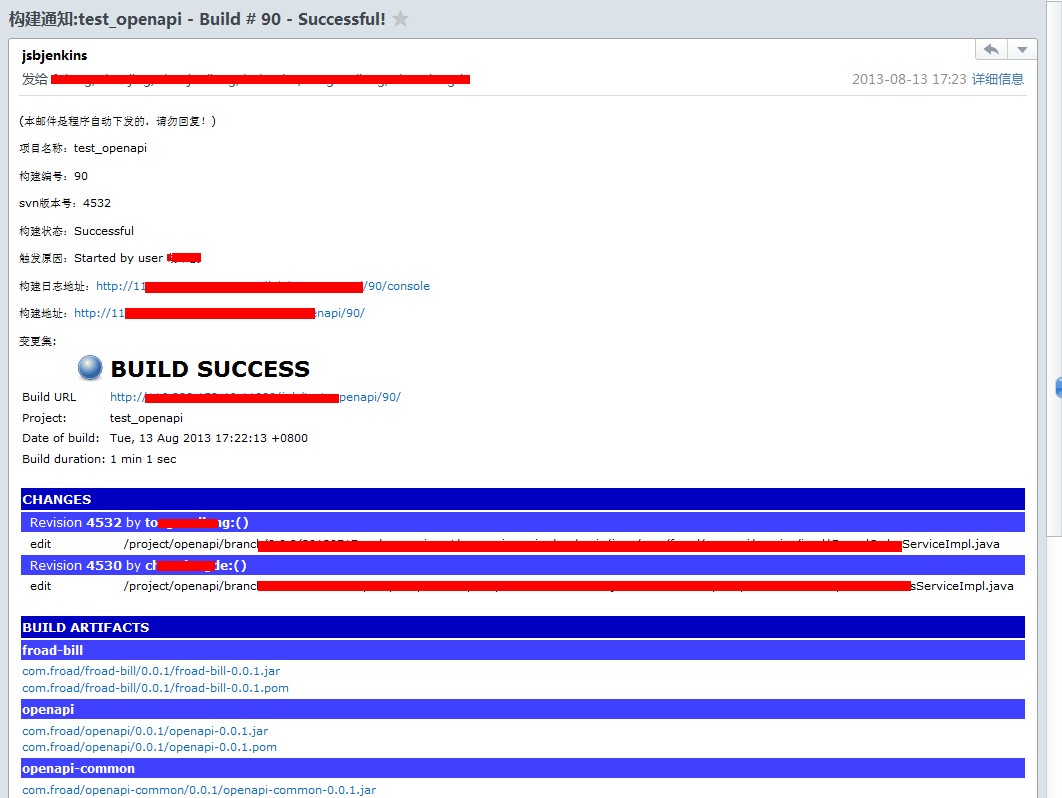
值得注意的是，拥有Hudson管理员权限是使用自定义Jelly脚本(该脚本没有跟email-ext打包)的前提。脚本的生成步骤本身其实相对简单：

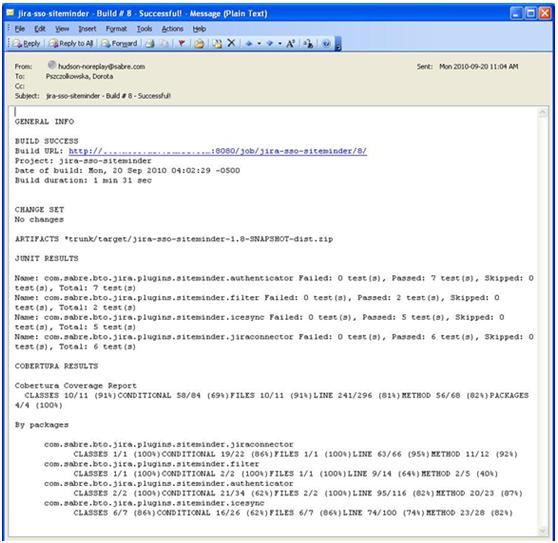
1. 创建Jelly脚本。脚本的名称应该是<名称>.jelly。名称以.jelly结尾是很重要的。

2. 把脚本存放在JENKINS\_HOME\email-templates文件夹里。

3. 使用Jelly变量，让template匹配你的脚本名称(不要包含后缀)。比如，脚本的名称为foobar.jelly，则邮件内容中应该是${JELLY\_SCRIPT,template=”foobar”}。

下面两个图就是就是使用Jelly脚本生成的邮件（最新版Email-ext新增html\_gamil模板，它跟html模板类似，所以这里不再显示它的截图）：





# 4 总结

以上就是我介绍的Email-ext插件，由于自己的局限，对于它的使用没有更深的了解。参考资料[2]中还有关于它的扩展，你也可以自行扩充它的功能。文章部分内容来源于参考资料[3]。如果您有关于该插件以及Jenkins使用的更多更好的感受，我期待与您一起分享。

**02email-ext邮件通知模板**

发现一个很好的邮件通知模板，根据我的需求定制了一些。分享一下。

Default Subject：

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

Default Content：

复制代码

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>${ENV, var="JOB\_NAME"}-第${BUILD\_NUMBER}次构建日志</title>

</head>

<body leftmargin="8" marginwidth="0" topmargin="8" marginheight="4"

offset="0">

<table width="95%" cellpadding="0" cellspacing="0"

style="font-size: 11pt; font-family: Tahoma, Arial, Helvetica, sans-serif">

<tr>

<td>(本邮件是程序自动下发的，请勿回复！)</td>

</tr>

<tr>

<td><h2>

<font color="#0000FF">构建结果 - ${BUILD\_STATUS}</font>

</h2></td>

</tr>

<tr>

<td><br />

<b><font color="#0B610B">构建信息</font></b>

<hr size="2" width="100%" align="center" /></td>

</tr>

<tr>

<td>

<ul>

<li>项目名称&nbsp;：&nbsp;${PROJECT\_NAME}</li>

<li>构建编号&nbsp;：&nbsp;第${BUILD\_NUMBER}次构建</li>

<li>SVN&nbsp;版本：&nbsp;${SVN\_REVISION}</li>

<li>触发原因：&nbsp;${CAUSE}</li>

<li>构建日志：&nbsp;<a href="${BUILD\_URL}console">${BUILD\_URL}console</a></li>

<li>构建&nbsp;&nbsp;Url&nbsp;：&nbsp;<a href="${BUILD\_URL}">${BUILD\_URL}</a></li>

<li>工作目录&nbsp;：&nbsp;<a href="${PROJECT\_URL}ws">${PROJECT\_URL}ws</a></li>

<li>项目&nbsp;&nbsp;Url&nbsp;：&nbsp;<a href="${PROJECT\_URL}">${PROJECT\_URL}</a></li>

</ul>

</td>

</tr>

<tr>

<td><b><font color="#0B610B">Changes Since Last

Successful Build:</font></b>

<hr size="2" width="100%" align="center" /></td>

</tr>

<tr>

<td>

<ul>

<li>历史变更记录 : <a href="${PROJECT\_URL}changes">${PROJECT\_URL}changes</a></li>

</ul> ${CHANGES\_SINCE\_LAST\_SUCCESS,reverse=true, format="Changes for Build #%n:<br />%c<br />",showPaths=true,changesFormat="<pre>[%a]<br />%m</pre>",pathFormat="&nbsp;&nbsp;&nbsp;&nbsp;%p"}

</td>

</tr>

<tr>

<td><b>Failed Test Results</b>

<hr size="2" width="100%" align="center" /></td>

</tr>

<tr>

<td><pre

style="font-size: 11pt; font-family: Tahoma, Arial, Helvetica, sans-serif">$FAILED\_TESTS</pre>

<br /></td>

</tr>

<tr>

<td><b><font color="#0B610B">构建日志 (最后 100行):</font></b>

<hr size="2" width="100%" align="center" /></td>

</tr>

<!-- <tr>

<td>Test Logs (if test has ran): <a

href="${PROJECT\_URL}ws/TestResult/archive\_logs/Log-Build-${BUILD\_NUMBER}.zip">${PROJECT\_URL}/ws/TestResult/archive\_logs/Log-Build-${BUILD\_NUMBER}.zip</a>

<br />

<br />

</td>

</tr> -->

<tr>

<td><textarea cols="80" rows="30" readonly="readonly"

style="font-family: Courier New">${BUILD\_LOG, maxLines=100}</textarea>

</td>

</tr>

</table>

</body>

</html>

复制代码

**03parameterized-trigger插件**

说明：这个插件可以根据已经完成构建的结果，触发新Job或者传递参数。

官方说明：[Parameterized Trigger Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Parameterized+Trigger+Plugin)

安装步骤：

系统管理→管理插件→可选插件→Build Triggers→[Parameterized Trigger Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Parameterized+Trigger+Plugin)

安装完成后到项目（job）配置页面：

构建后操作→Add post-build action→Trigger parameterized build on other projects

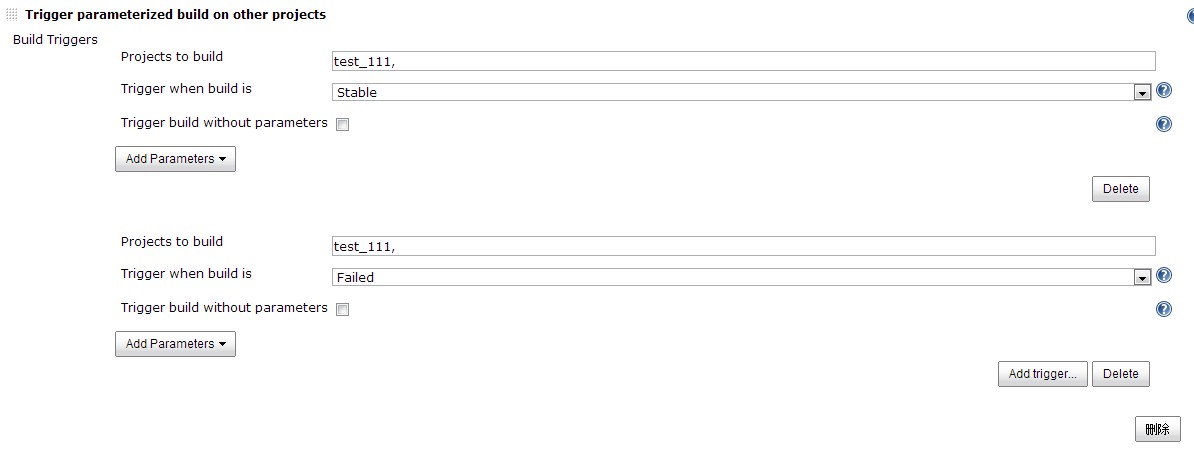


图1 插件配置示例界面

Add trigger ：添加触发

Projects to build：需要构建的项目（job）名,如果有多个项目用 , 分隔。

Trigger when build is：构建触发的条件

* Stable 成功（稳定）（等价于successful）
* Unstable 不稳定
* Stable or unstable but not failed 成功（稳定）或者不稳定，但是没有失败
* Unstable or Failed but not stable 不稳定或者失败，但是没有成功（稳定）
* Failed 失败
* Complete (always trigger) 完成（总是触发）

Trigger build without parameters：构建的job不带参数(选择后下面的参数无效)

没有特别的需求，参数可以忽略。

Add Parameters：添加参数

* Boolean parameters 布尔参数(键值对)

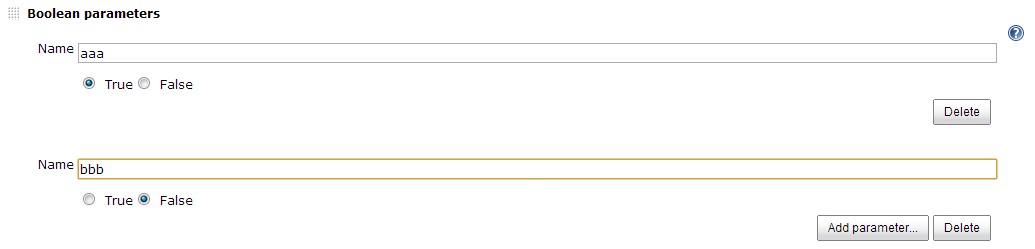


图2 Boolean parameters 界面

* Build on the same node 在同一个节点上构建

（没有什么可以填写的，选择就可以了）



图3 Build on the same node界面

* Current build parameters 当前构建参数

（没有什么可以填写的，选择就可以了）

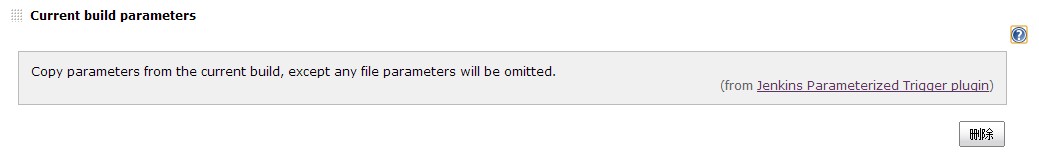


图4 Current build parameters 界面

* Parameters from properties file 从属性文件获取参数

相对于工作区的路径，用逗号分隔文件列表。文件类型：\*.properties 。如果你不知道怎么写属性文件，建议你询问java开发人员。

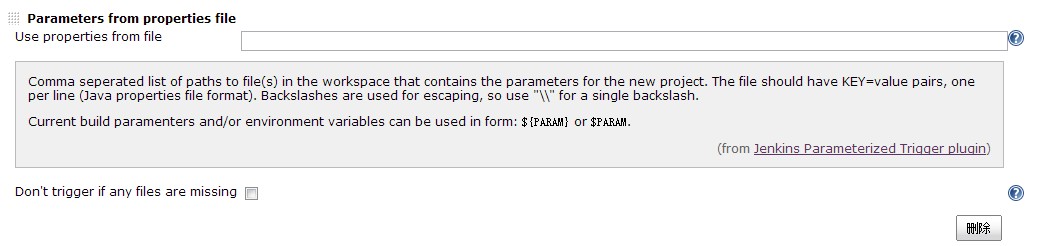


图5 Parameters from properties file界面

* Predefined parameters 预定义参数

和properties文件的语法相同 。如果你不知道怎么写属性文件，建议你询问java开发人员。

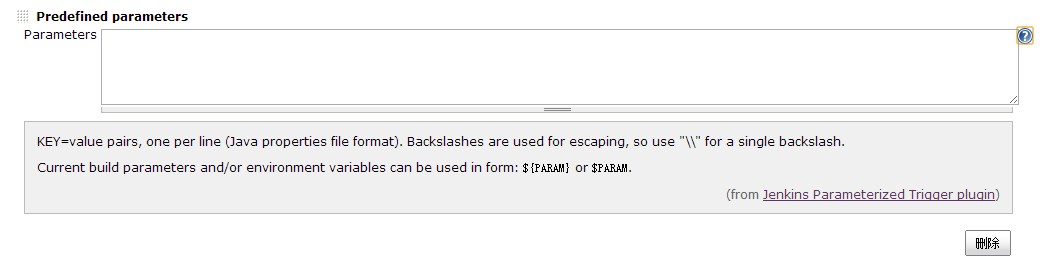


图6 Predefined parameters界面

* Restrict matrix execution to a subset 限制矩阵执行的一个子集

指定Groovy的过滤器表达式限制将运行的下游项目组合子集。

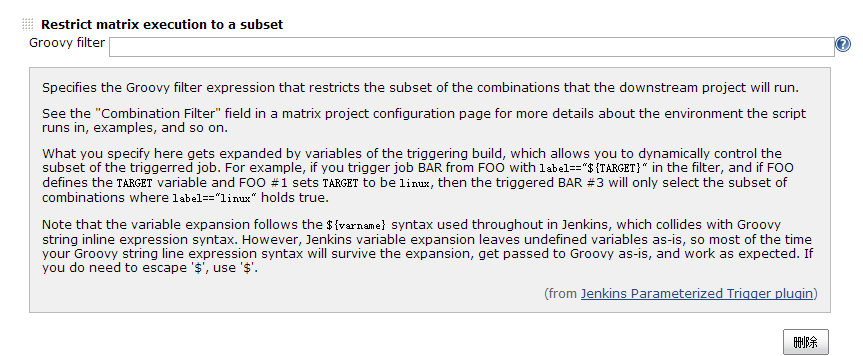


图7 Restrict matrix execution to a subset 界面

* Subversion revision svn版本

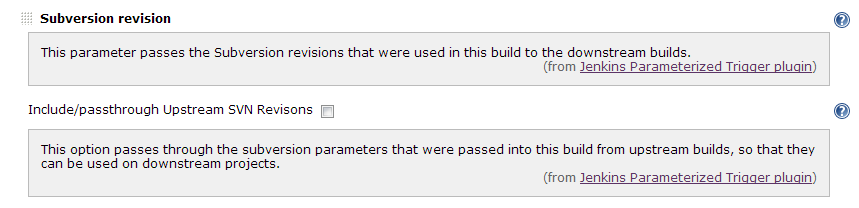


图8 Subversion revision 界面

**04Publish Over FTP Plugin插件**

说明：这个插件可以将构建的产物（例如：Jar）发布到FTP中去。

官方说明：[Publish Over FTP Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Publish+Over+FTP+Plugin)

安装步骤：

系统管理→管理插件→可选插件→Artifact Uploaders→[Publish Over FTP Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Publish+Over+FTP+Plugin)

# 系统设置

FTP Server Name：给你自己看的名字，爱叫什么叫什么

Hostname：主机IP或者域名

Username：ftp登陆用户名

Password：ftp密码

Remote Directory：远程根目录（建议设置为：/）

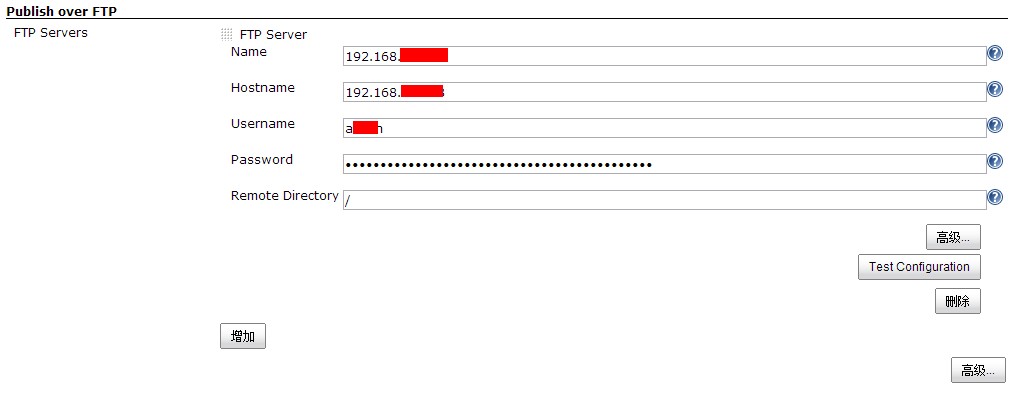


图1 系统设置基本界面

点击Test Configuration上面的高级，如下图：

Port：端口（不知道问管理员去）

Timeout (ms)：超时时间（毫秒）

Use active data mode： （未选中）默认选项使用PASV（被动模式），选中使用PORT （主动模式）

Don't make nested dirs：不创建下级目录（具体的自己看帮助）

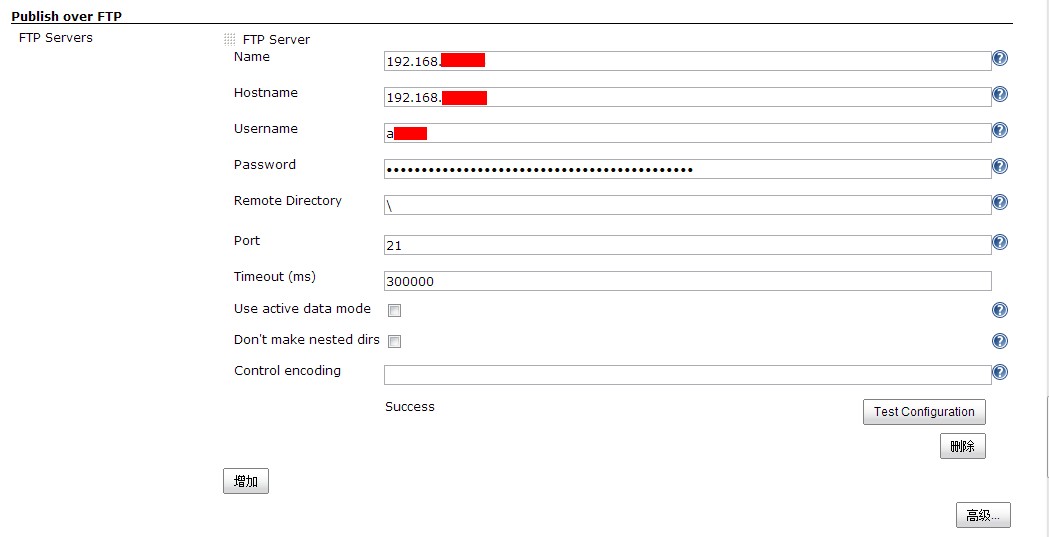


图2 系统设置高级界面

上图右下角还有一个高级选项。有兴趣的自己可以折腾下。提示：你需要先保存后，刷新页面，在配置。不然有个选项无法选择的。

# 项目配置

启用步骤：

构建后操作→Add post-build action→Send build artifacts over FTP

FTP Server Name：选个一个你在系统设置里配置的配置的名字

Transfer Set Source files：需要上传的文件（注意：相对于工作区的路径,可以是单个文件也可以是目录）

Remove prefix：移除目录（只能指定Transfer Set Source files中的目录）

Remote directory：远程目录（根据你的需求填写吧，因为我这儿是测试,所以偷懒直接用/）



图3 项目设置基本界面

点击Remote directory后面的高级，如下图

Exclude files：排除的文件（在你传输目录的时候很有用，使用通配符，例如：\*\*/\*.log,\*\*/\*.tmp,.git/）

Pattern separator：分隔符（配置Exclude files和Source files的分隔符。如果你这儿更改了，上面的内容也需要更改）

No default excludes：禁止默认的排除规则（具体的自己看帮助）

Make empty dirs：此选项会更改插件的默认行为。默认行为是匹配该文件是否存在，如果存在则创建目录存放。选中此选项会直接创建一个目录存放文件，即使是空目录。（个人理解）

Flatten files：只在ftp上建立文件，不创建目录（除了远程目录）

Remote directory is a date format:远程目录建立带日期的文件夹（需要在Remote directory中配置日期格式），具体格式参考下表：

|  |  |
| --- | --- |
| **Remote directory** | **Directories created** |
| 'qa-approved/'yyyyMMddHHmmss | qa-approved/20101107154555 |
| 'builds/'yyyy/MM/dd/'build-${BUILD\_NUMBER}' | builds/2010/11/07/build-456 (if the build was number 456) |
| yyyy\_MM/'build'-EEE-d-HHmmss | 2010\_11/build-Sun-7-154555 |
| yyyy-MM-dd\_HH-mm-ss | 2010-11-07\_15-45-55 |

Clean remote:上传前会删除远程目录中的所有的文件（血的教训啊，测试的时候用的是运营小组的ftp，然后一不小心就把他们的数据删除了，害的我去做数据恢复。）

ASCII mode：文件传输的方式，一般默认不选。



图4 项目设置高级界面

好了，常用的选项就这些，其他的自己折腾吧。这个插件适合动手能力强的人。

**05FTP publisher plugin插件**

说明：这个插件可以将构建的产物（例如：Jar）发布到FTP中去。

官方说明：[FTP publisher plugin](http://wiki.hudson-ci.org/display/HUDSON/FTP-Publisher+Plugin)

安装步骤：

系统管理→管理插件→可选插件→Artifact Uploaders→[FTP publisher plugin](http://wiki.hudson-ci.org/display/HUDSON/FTP-Publisher+Plugin)

# 系统设置

Profile Name：给你自己看的名字，爱叫什么叫什么

hostname：主机IP或者域名

Port：端口（不知道问管理员去）

TimeOut：超时时间（毫秒）

Root Repository Path：远程根目录（建议设置为：/）

User Name：ftp登陆用户名

Password：ftp密码

增加：增加一个ftp服务端配置

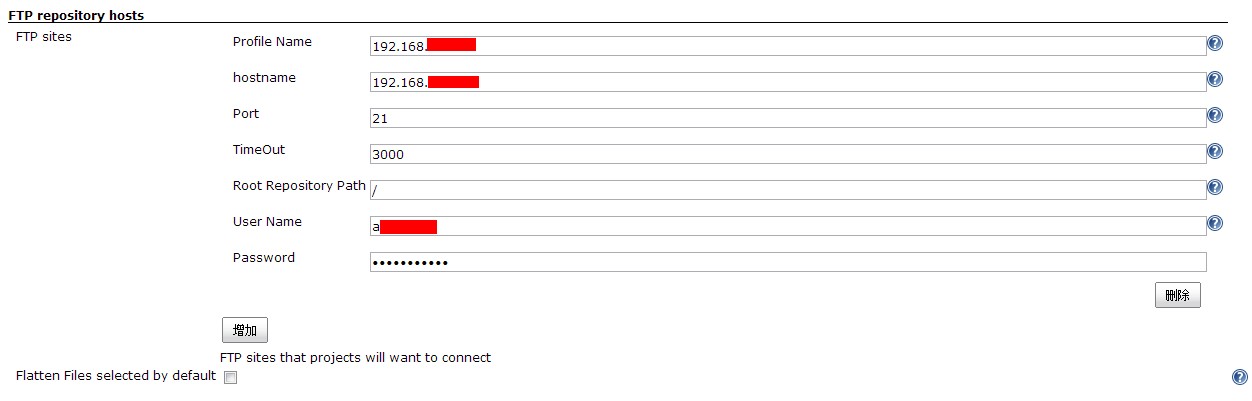


图1 系统设置界面

是不是比Publish Over FTP Plugin的配置简单多了。

# 项目配置

启用步骤：

构建后操作→Add post-build action→Publish artifacts to FTP

FTP site：选个一个你在系统设置里配置的ftp名字

Files to upload（点击后面的增加）

　　Source：需要上传的文件（注意：相对于工作区的路径,只能是单个文件）

　　Destination：目标文件夹（配置该选项后，如果不存在会创建一个目录）

增加：再增加一个上传选项

Use timestamps：配置该选项后，每次上传会创建一个时间目录。如：2013-09-09\_17-23-39

Flatten files：只在ftp上建立文件，不创建目录（除了远程目录）【Source中存在的目录不会在ftp上被创建，如：aaa目录】

Skip publishing (e.g. temporarily)：选择该选项后不会发布构建到ftp。测试日志：Publish artifacts to FTP - Skipping...



好了，ftp上传教程到此结束。这个插件很简单。

**07更改Jenkins的主目录**

Jenkins默认会存放在用户主目录下的.jenkins文件夹中

如：Linux root用户：/root/.jenkins

注意：这是linux版本的。windows系统请自行更改。这个值在Jenkins运行时是不能更改的. 请先将Jenkins停止运行。

如果查看当前Jenkins目录：系统管理→系统设置→主目录→/root/.jenkins（注意：用户不同，路径也不同）

Jenkins储存所有的数据文件在这个目录下. 你可以通过以下几种方式更改：

1. 使用你Web容器的管理工具设置JENKINS\_HOME环境参数.
2. 打开tomcat的bin目录，编辑catalina.sh文件。  
   在# OS specific support. $var \_must\_ be set to either true or false.上面添加：export JENKINS\_HOME=""

在引号中填入你的路径。

1. 在启动Web容器之前设置JENKINS\_HOME环境变量.
2. 用root用户登录
3. 编辑profile文件：vi /etc/profile
4. 在最后加入：export JENKINS\_HOME=xxxx
5. 保存，退出后执行：source /etc/profile

让配置生效

1. (不推荐)更改Jenkins.war(或者在展开的Web容器)内的web.xml配置文件.

复制代码

<!-- if specified, this value is used as the Hudson home directory -->

<env-entry>

<env-entry-name>HUDSON\_HOME</env-entry-name>

<env-entry-type>java.lang.String</env-entry-type>

<env-entry-value></env-entry-value>

</env-entry>

<!-- 在<env-entry-value>节点中填入路径，windows系统建议使用/分隔路径 -->

复制代码

好了，现在运行Jenkins看看效果吧。

**08Jenkins纳入版本控制**

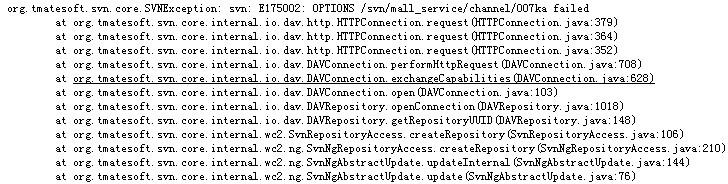
是不是有过这种经历：某天手贱，把某一个配置更改了。然后只能在那儿苦逼的看着，然后的然后把以前的配置重新配一次。坑爹的是以前配置的什么都忘记了。。。尼玛，坑啊( ⊙ o ⊙ )！

某天在群里和一群人吹牛逼，突然冒出一个人说手贱把某个job删除了，有没有办法恢复。然后我们果断的告诉了ta实情。然后这聊天的主题就跑到把jenkins加入到版本控制中。我也曾经手贱过。运气好，前几天要测试一些东西，在另一台机器上面还有备份。所以折腾折腾，就写了这篇文章。

系统：CentOS 5.8

版本控制软件：Subversion 1.7+

思路：写一个脚本（shell）定时去跑一次add和commit。

1. 建立普通用户  
   用一个用户会出问题，jenkins构建的时候会直接报错。貌似是因为https证书的问题。如果你是用的http协议可以尝试一个用户搞定。

这个问题的解决方法：将用户目录下.subversion的auth目录删除就可以了

useradd froad #建立普通用户跑Jenkins，root用户用了跑shell脚本  
#启动Jenkins略过  
#因为我是用root用户登录，然后 su froad切换的，所以就没有设置密码。

1. 安装Subversion1.7以上版本（当做客户端给root用户使用）  
   。其实有用的就是bin/svn这个程序而已。如果你安装了Subversion edge 直接把bin和lib目录拷贝过来就行了。为什么要用1.7以上的？因为1.7后.svn目录只有一个了。我嫌以前的版本.svn目录太多了，烦！提供个解压即可用的svn(从subversion edge中拷贝出来的) [点我去下载](http://pan.baidu.com/s/17Zwse)
2. 远程Subversion的一些准备工作  
   如果你喜欢可以创建一个新的库和用户，如果你不like，随便你！
3. 检出svn目录

复制代码

[root@localhost ~]# /home/froad/svn/bin/svn co https://192.168.xxx.xxx/svn/Jenkins/trunk /home/froad/.jenkins/ --username jenkins

#说明：/home/froad/.jenkins这个参数可以直接把检出的.svn目录放到.jenkins目录，我是懒人不想在mv一次。--username是svn的用户

#svn语法：svn co url path --username xxx

Error validating server certificate for **'https://192.168.xxx.xxx:443'**:

- The certificate is not issued by a trusted authority. Use the

fingerprint to validate the certificate manually!

- The certificate hostname does not match.

Certificate information:

- Hostname: froad-jskfb

- Valid: from Apr 16 03:04:08 2013 GMT until Apr 14 03:04:08 2023 GMT

- Issuer: (null), (null), (null), (null), (null) ((null))

- Fingerprint: D6:EB:58:xxxxxxxxxxxxxxxFC:7D:04:64

(R)eject, accept (t)emporarily or accept (p)ermanently? p #说明：R 拒绝 t 暂时接受 p 永久接受

Authentication realm: <https://192.168.xxx.xxx:443> VisualSVN Server

Password for **'jenkins'**: \*\*\*\*\*\*\*\*\*\*\*\*\* #说明：输入你的用户密码

-----------------------------------------------------------------------

ATTENTION! Your password for authentication realm:

<https://192.168.xxx.xxx:443> VisualSVN Server

can only be stored to disk unencrypted! You are advised to configure

your system so that Subversion can store passwords encrypted, if

possible. See the documentation for details.

You can avoid future appearances of this warning by setting the value

of the **'store-plaintext-passwords'** option to either **'yes'** or **'no'** in

**'/root/.subversion/servers'**.

-----------------------------------------------------------------------

Store password unencrypted (yes/no)? yes #说明：存储未加密的密码

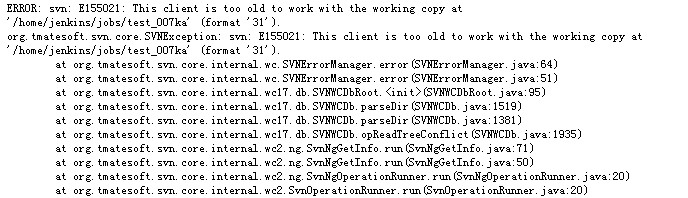
Checked out revision 1.

#现在版本库都检出了，下面的就简单了。add一下，然后commit一下。打完收工。

#如果你是这么想的，那么结果会郁闷死你。接着往下看吧。

复制代码

1. 更改Jenkins的workspace目录

为什么要更改workspace目录呢？因为里面有个捣蛋的目录.svn。这个目录是个深坑，开始我花了大力气去忽略这个目录。坑爹的是如果我把工作目录清空了，然后在构建就会一直报错。因为SVNkit跑去找JENKINS\_HOME下的.svn目录（这个目录就是我们上面检出的）。  


测试的时候是直接把.svn目录检出到test\_007ka目录。大家将就看。除了目录不一样，其他没有区别。

SVNkit会先去找.svn目录，如果到顶层目录还是没有找到.svn目录，就会在workspace目录检出。这是svn和git的一个特性。哪儿看到的？那个《git权威指南》。  
系统管理→系统设置→主目录（的右边问号下面）→高级（是不是忽略了啊\(^o^)/~）→工作空间根目录

点开后面的问号可以看见3个参数（配置路径需要的）：

* + - ${JENKINS\_HOME} — Jenkins home directory.#JENKINS\_HOME这个参数不用说了
    - ${ITEM\_ROOTDIR} — Root directory of a job for which the default workspace is allocated.#ITEM\_ROOTDIR：默认的工作空间目录。完整的路径就是JENKINS\_HOME/jobs/xxxx/workspace
    - ${ITEM\_FULL\_NAME} — '/'-separated job name, like "foo/bar".#ITEM\_FULL\_NAME：job的名称，这个就是我们需要的。

我们只需要把workspace目录赶出JENKINS\_HOME目录就行了。上配置：

workspace：/home/froad/workspace/${ITEM\_FULL\_NAME} #前面的目录随便你改，只需要在最后带上${ITEM\_FULL\_NAME}

JENKINS\_HOME：/home/froad/.jenkins #给你们对比着看

#好了，这下.svn目录不打架了。我也不用头疼了。

1. 删除已经存在的workspace目录  
     
   为什么要删除已经存在的workspace？想知道结果？跳过这一步，执行下面一步就知道结果了。

find . -type d -name **"workspace"**|xargs rm -rf #看见find后面的那个点了么，改成你的路径就行了

1. 配置Subversion忽略目录

貌似将JENKINS\_HOME目录直接提交到Subversion，数据量不是一般的高啊！我这个Jenkins才跑3个月都10Gb了。

如果要提交上去，花费的时间，不是一般的长啊。

注意：你用的什么用户，就在用户目录下面改。我用的root用户。

进入用户目录的.subversion打开config文件，找到global-ignores。大概在105行。

把global-ignores前的#去掉，注意#后面的空格要删掉。然后在=后面添加 modules

modules目录是maven项目产生的，我的Jenkins就是这个目录大。把这个目录删除了也就几百Mb。如果你有其他目录想忽略的自行添加，记得用空格分隔就行了。

1. 写个shell脚本svn\_commit.sh

这个脚本随便你放在哪儿，我想偷懒。直接放在Jenkins的主目录，顺便也版本控制下。。O(∩\_∩)O~  
声明：脚本写的很丑，勿喷！需要版本控制的文件或目录，命名不能是 ${xxxx} 格式。

复制代码

#!/bin/bash

#Jenkins的Subversion备份脚本

#请将本脚本放到JENKINS\_HOME目录

#作者：zjl

#version：2.0

#time：2014-01-07 18:36

#

#获取当前系统时间

DATE=`date **"+%Y%m%d"**`

SVN\_HOME=/home/froad/svn

JENKINS\_HOME=/home/froad/jenkins

#日志文件

LOG\_HOME=/root/.subversion/logs

LOG\_FILE=${LOG\_HOME}/svn\_commit\_${DATE}.log

TMP\_STATUS\_LOG=${LOG\_HOME}/status.log

echo **"[begin] 当前时间：`date "**+%Y%m%d\_%H%M%S**"`"** >>${LOG\_FILE}

if [ ! -d **"${LOG\_HOME}"** ];then

mkdir -p ${LOG\_HOME}

fi

#进入到svn工作目录，防止出现稀奇古怪的错误

cd ${JENKINS\_HOME}

function svn\_add()

{

#使用svn add命令将文件添加到版本控制

${SVN\_HOME}/bin/svn add ${JENKINS\_HOME}/\* --username=jenkins --force &>>${LOG\_FILE}

#退出码

EXIT\_NUM=$?

echo **"svn add退出码：${EXIT\_NUM} ！"** >>${LOG\_FILE}

}

function svn\_status()

{

#使用svn status命令判断文件是否删除、修改或者其他操作

${SVN\_HOME}/bin/svn status **"/${JENKINS\_HOME}"** > ${TMP\_STATUS\_LOG}

#循环从文件中读取一行

cat ${TMP\_STATUS\_LOG} | while read line;do

#获取第一个字符，根据第一个字符判断情况

l=`echo ${line}|cut -c1`

#状态为A(增加)时，不需要任何操作，直接返回。

if [ **"${l}"** == **"A"** ];then

continue

fi

#状态为D(删除)时，不需要任何操作，直接返回。

if [ **"${l}"** == **"D"** ];then

continue

fi

#状态为M(修改)时，不需要任何操作，直接返回。

if [ **"${l}"** == **"M"** ];then

continue

fi

#状态为?(未受控制)时，调用add命令然后返回。ps：不知道怎么的有时候从文件中读取回来的？变成了0

if [ **"${l}"** == **"0"** ] || [ **"${l}"** == **"?"** ];then

${SVN\_HOME}/bin/svn add **"${JENKINS\_HOME}/`echo ${line} | cut -d ' ' -f 2- `"** >>${LOG\_FILE}

continue

fi

#状态为!(丢失或者不完整)时，调用delete命令然后返回。

if [ **"${l}"** == **"!"** ];then

${SVN\_HOME}/bin/svn delete **"${JENKINS\_HOME}/`echo ${line} | cut -d ' ' -f 2- `"** >>${LOG\_FILE}

continue

fi

echo **"[info]其他情况:"**${line} >>${LOG\_FILE}

done

#删除临时文件

rm -rf ${TMP\_STATUS\_LOG}

}

function svn\_commit()

{

#使用svn commit命令提交到服务器

${SVN\_HOME}/bin/svn commit ${JENKINS\_HOME}/\* --message=**"crontab commit"** --username=jenkins &>>${LOG\_FILE}

EXIT\_NUM=$?

if [ ! ${EXIT\_NUM} == **"0"** ];then

echo **"[error]退出码：${EXIT\_NUM} svn commit失败,请查看日志！"**>>${LOG\_FILE}

return

fi

echo **"svn commit退出码：${EXIT\_NUM} ！"**>>${LOG\_FILE}

}

#调用方法

svn\_add

svn\_status

svn\_commit

echo **"[end] 当前时间：`date "**+%Y%m%d\_%H%M%S**"`"** >>${LOG\_FILE}

#输出空行隔离

echo >>${LOG\_FILE}

复制代码

脚本写好了，建议先运行一次。测试下有没有错误。

1. 添加定时提交
2. crontab -u root -e

\*/30 \* \* \* \* sh /home/froad/.jenkins/svn\_commit.sh #每30分钟提交一次

1. 结尾

普通用户开机自动启动Jenkins：

su 用户名 -c **"sh path"**

#例如：su froad -c "sh /usr/local/jenkins/bin/startup.sh"

将上面的命令加入到/etc/rc.d/rc.local文件中

非root用户不能使用1024以下的端口，比较麻烦。有不懂的，下面留言。有好的建议，下面留言。欢迎交流！

**09配置Linux系统ssh免密码登陆**

ssh认证的完整描述:https://www.ibm.com/developerworks/cn/linux/security/openssh/part1/

说明：[点我去查看](http://baike.baidu.com/subview/16184/5909252.htm)

今天我们只说生成ssh的key，从而达到免密码登陆的目的。

不知道ssh是什么的自己看说明。好了，不废话了。

系统：CentOS 5.8

1. 确认用户

确认当前用户是你需要的用户！

$ whoami

froad #froad用户，根据你自己的需求选择用户。我这儿用froad用户演示。

1. 生成key

复制代码

$ cd ~ #回到用户目录，不回去也没有关系

$ ssh-keygen

#可以使用-t选项选择加密方式，包括 RSA 和 DSA 两种密钥

#例如：$ssh-keygen -t dsa 或者ssh-keygen -t rsa

#加密方式不同，key的名称不同，其他没有区别

#如果没有指定密钥，默认RSA

Generating public/private rsa key pair.

Enter file in which to save the key (/home/froad/.ssh/id\_rsa): #私钥存放的位置，默认会存放在用户目录的.ssh文件夹，直接回车

Enter passphrase (empty for no passphrase): #默认，回车

Enter same passphrase again: #默认，回车

Your identification has been saved in /home/froad/.ssh/id\_rsa.#私钥路径

Your public key has been saved in /home/froad/.ssh/id\_rsa.pub.#公钥路径

The key fingerprint is:

e8:b6:e6:xxxxxxxxxxxxxxxxx:ec:b5:d8 froad@localhost.localdomain

$ cat id\_\*\*\*.pub >> authorized\_keys #\*\*\*代表加密方式，将公钥拷贝到authorized\_keys文件中。如果你有很多的电脑需要配置，将所有的id\_\*\*\*.pub公钥拷贝到一个authorized\_keys文件中即可

复制代码

1. 配置

将authorized\_keys文件拷贝到需要被管理的电脑上。注意：放在用户目录下.ssh文件夹中。Linux用户会限制你的访问权限

1. 验证
2. 复制代码
3. $ ssh root@192.168.2.xxx
4. #root是用户名，可以根据你的需求改变
5. The authenticity of host **'192.168.2.xxx (192.168.2.xxx)'** can**'t be established.**
6. **RSA key fingerprint is ff:07:49:4d:xxxxxxxxxxe:2c:38.**
7. **Are you sure you want to continue connecting (yes/no)? yes**
8. #同意将指纹添加到本地
9. Warning: Permanently added **'**192.168.2.xxx**' (RSA) to the list of known hosts.**

**Last login: Wed Oct 23 13:58:32 2013 from 192.168.1.xxx**

复制代码

只能在你生成key的电脑上访问authorized\_keys的电脑，如果你想两台电脑互相访问均免密码。那么你还需要重复上面的步骤（机器的配置刚好相反）。

1. 其他

　　如果添加指纹的时候提示添加失败，是因为你以前添加过了这个ip的指纹。

　　 解决办法：将.ssh目录的known\_hosts文件删除掉（好粗暴啊( ⊙ o ⊙ )啊！），也可以打开这个文件把对于ip的那条记录删除（这个就精细多了O(∩\_∩)O哈哈~）

　　如果操作步骤都正确，但是依然要求输入密码。一般是因为权限的问题。命令如下

chmod 644 ~/.ssh/authorized\_keys

**10Publish Over SSH插件**

说明：这个插件可以通过ssh连接其他Linux机器

官方说明：[Publish Over SSH](http://wiki.jenkins-ci.org/display/JENKINS/Publish+Over+SSH+Plugin)

安装步骤：

系统管理→管理插件→可选插件→Artifact Uploaders→[Publish Over SSH](http://wiki.jenkins-ci.org/display/JENKINS/Publish+Over+SSH+Plugin)

# 系统设置（所有的高级全部展开）

公共配置：

Passphrase：密码（key的密码，如果你设置了）

Path to key：key文件（私钥）的路径

Key：将私钥复制到这个框中

Disable exec：禁止运行命令

私有配置：

SSH Server Name：标识的名字（随便你取什么）

Hostname：需要连接ssh的主机名或ip地址（建议ip）

Username：用户名

Remote Directory：远程目录

Use password authentication, or use a different key：可以替换公共配置（选中展开的就是公共配置的东西，这样做扩展性很好）

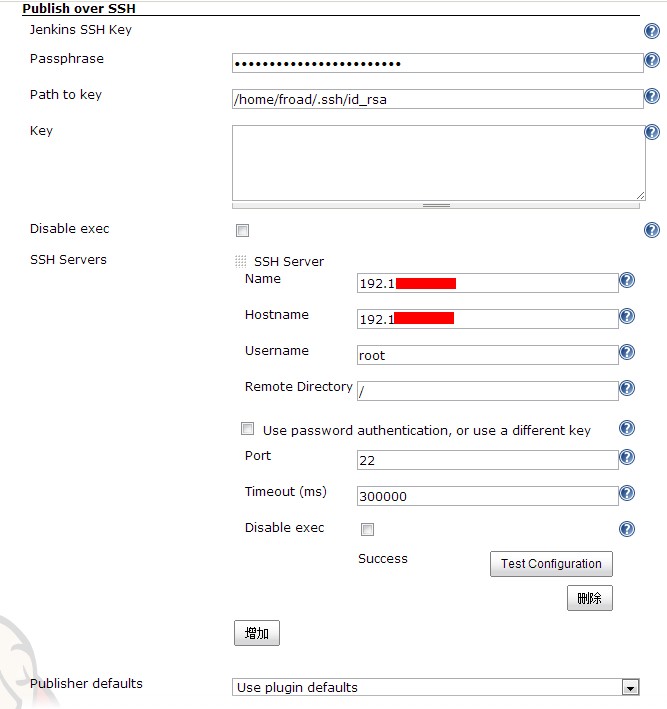
私有配置的高级：

Port：端口（默认22）

Timeout (ms)：超时时间（毫秒）默认即可

Disable exec：禁止运行命令

Test Configuration：测试连接



Publisher defaults：插件底层的设置，有兴趣的自己折腾。

# 项目配置

启用步骤：

构建后操作→Add post-build action→Send build artifacts over SSH

SSH Server Name：选个一个你在系统设置里配置的配置的名字

Transfer Set Source files：需要上传的文件（注意：相对于工作区的路径。看后面的配置可以填写多个，默认用,分隔）

Remove prefix：移除目录（只能指定Transfer Set Source files中的目录）

Remote directory：远程目录（根据你的需求填写吧，因为我这儿是测试,所以偷懒没有填写。默认会继承系统配置）

Exec command：把你要执行的命令写在里面

（新版本的要求Transfer Set Source files和Exec command都要填写，可以不了他。老版本的允许只填写一个）

高级：

Exclude files：排除的文件（在你传输目录的时候很有用，使用通配符，例如：\*\*/\*.log,\*\*/\*.tmp,.git/）

Pattern separator：分隔符（配置Transfer Set Source files的分隔符。如果你这儿更改了，上面的内容也需要更改）

No default excludes：禁止默认的排除规则（具体的自己看帮助）

Make empty dirs：此选项会更改插件的默认行为。默认行为是匹配该文件是否存在，如果存在则创建目录存放。选中此选项会直接创建一个目录存放文件，即使是空目录。（个人理解）

Flatten files：只上传文件，不创建目录（除了远程目录）

Remote directory is a date format:远程目录建立带日期的文件夹（需要在Remote directory中配置日期格式），具体格式参考下表：

|  |  |
| --- | --- |
| **Remote directory** | **Directories created** |
| 'qa-approved/'yyyyMMddHHmmss | qa-approved/20101107154555 |
| 'builds/'yyyy/MM/dd/'build-${BUILD\_NUMBER}' | builds/2010/11/07/build-456 (if the build was number 456) |
| yyyy\_MM/'build'-EEE-d-HHmmss | 2010\_11/build-Sun-7-154555 |
| yyyy-MM-dd\_HH-mm-ss | 2010-11-07\_15-45-55 |

Exec timeout (ms)：运行脚步的超时时间（毫秒）

Exec in pty：模拟一个终端执行脚步

Add Transfer Set：增加一个配置

恩，这个插件就介绍到这里。很简单的\(^o^)/~

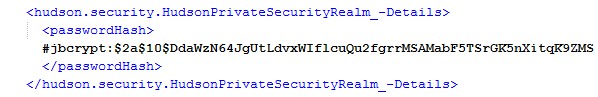
**11修改Jenkins用户的密码**

说明：本方法仅适用于jdk6+、tomcat6+和Jenkins专有用户数据库的Jenkins！

很多童鞋在使用jenkins的时候忘记密码了，然后各种蛋疼。最近闲着无事，折腾了下。好了，闲话少扯。

Jenkins专有用户的数据存放在JENKINS\_HOME/users目录。users目录的结构我就不扯了，你一看就懂。users/xxx(各种用户名)

打开忘记密码的用户文件夹，里面就一个文件config.xml。嗯，就是它了。打开config.xml，里面有一堆的东西，找找。。。找到了<passwordHash>节点，如图：



把<passwordHash>节点的内容（图中黑色的那一串）换成#jbcrypt:$2a$10$DdaWzN64JgUtLdvxWIflcuQu2fgrrMSAMabF5TSrGK5nXitqK9ZMS

（注意前后有木有空格）

保存，重启Jenkins程序。

然后输入用户名，密码111111

**12详解Jenkins节点配置**

Jenkins有个很强大的功能：分布式构建(在Jenkins的配置中叫做节点)，分布式构建能够让同一套代码在不同的环境(如：Windows和Linux系统)中编译、测试等。这么做的好处一大堆，自己去想。而且Jenkins构建的代码和产物最后自动拷贝到主节点。

注意：如果节点主机上不存在JDK，Jenkins会去自动下载，但Oracle对程序自动下载做了限制，会导致下载失败，然后一直循环这个问题。

建议：所有Unix或者Windows机器的环境路径统一(如：JDK、Ant、Maven)，好处是便于管理、不容易出现奇葩问题。

Jenkins版本：1.532(不同版本的配置可能不同)

进入节点配置界面：

系统管理→管理节点→新建节点(左上角)

节点名称：建议使用字母、数字或字母和数字的组合。最好见名知意。不建议使用标点符号和中文(中文命名没有问题，但Job中无法引用)

Dumb Slave：新建一个节点

复制现有节点：从已存在的节点中复制一份配置(如果存在节点才会显示)

点击ok进入下一步配置

Name：节点名称

Description：节点描述，支持中文

# of executors：最大同时构建数量(根据机器的性能定，单颗四核cpu建议不要超过5)【必须为数字】

Remote FS root：节点的根目录(注意：如果目录不存在，会自动创建目录。你必须对该目录有读写权限，不然会报错：hudson.util.IOException2: Failed to copy xxxx)

Labels：标记(又叫做标签)用来对多节点分组,标记之间用空格分隔.例如'refression java6'将会把一个节点标记上'regression'和'java6'.

　　举例来说,如果你有多个Windows系统的构建节点并且你的Job也需要在Windows系统上运行,那么你可以配置所有的Windows系统节点都标记为'windows', 然后把Job也标记为'windows'.这样的话你的Job就不会运行在除了Windows节点以外的其它节点之上了.

用法：尽可能的使用这个节点/只允许运行绑定到这台机器的Job(根据你的需求，二选一)

Launch method：运行方式有四个选项。建议选择第1、2种方式配置。详细如下：

1. 【推荐】Launch slave agents on Unix machines via SSH 在Unix(包括Linux)机器上通过SSH通道连接节点 (适用于Unix和Linux)

　　Host：节点主机的ip地址

　　Credentials：凭据(如果为空或者不可选择，请在系统管理→Manage Credentials中配置。Manage Credentials的配置非常简单，这里就不在描述了。Manage Credentials配置完成后，需刷新节点配置页面才会显示。)

　　Port：端口默认22

　　JavaPath：[可选]JDK路径，默认和master节点相同。路径必须指定到Java程序，如：/path/bin/java

　　JVM Options：[可选]JVM可选参数

　　Prefix Start Slave Command：[可选]不知道干什么用的参数

　　Suffix Start Slave Command：[可选]不知道干什么用的参数

　　 测试可以使用Unix命令，会自动拼接在[SSH] Starting slave process:[Prefix Start Slave Command] cd '/path' && /path/bin/java -jar slave.jar [Suffix Start Slave Command]

1. 【推荐】Launch slave agents via Java Web Start 通过Java Web Start连接节点 (适用于所有支持Java程序的系统)

　　Tunnel connection through：[可选]在端口转发这种情况下使用

　　JVM options：[可选]JVM可选参数

　　这种方法的缺点：如果该节点宕机了，主节点无法自动重启它。

1. Launch slave via execution of command on the Master 通过主节点的控制台连接节点

　　 在写内容前吐槽一下Jenkins的帮助，说的云里雾里的。应该去看该选项下Launch command的帮助。

　　 Jenkins的开发者考虑到某些企业可能有N++ 个节点(N>=你猜!)。如果在界面配置，那么升级版本之类的操作会很麻烦。所以允许你使用shell脚本去配置管理节点(貌似很方便的样子)。具体的脚本需要你自己写。

　　Launch command：Unix运行脚本的命令，如：sh aaa.sh

1. 【不建议使用】Let Jenkins control this Windows slave as a Windows service 让Jenkins节点添加到Windows服务中

　　 这个选项比Launch slave agents via Java Web Start添加为服务更加稳定(帮助文档是这么说的)。采用这种运行方式，那么这个系统不能登录任何用户。这种配置方式是非常的麻烦和折腾。具体请查看 [点我去看帮助](https://wiki.jenkins-ci.org/display/JENKINS/Windows+slaves+fail+to+start+via+DCOM)

　　Administrator user name：域\管理员账号

　　Password：密码

　　Host：节点主机IP或者域名

　　Run service as：

　　　　Use Local System User：使用本地系统用户

　　　　Log on using a different account：使用不同的用户登录

　　　　　　User name：账号

　　　　　　Password：密码

　　　　Use Administrator account given above：使用上面的用户登录

　　Path to java executable：[可选]JDK路径。必须指定到Java程序,如：C:\Windows\system32\java.exe

　　JVM options：[可选]JVM可选参数

Availability：

* Keep this slave on-line as much as possible：尽可能保持节点在线【推荐】
* Take this slave on-line according to a schedule：根据时间表在线(类似于Linux的定时任务)

Startup Schedule：类似于Linux定时任务的时间，如下：

# every fifteen minutes (perhaps at :07, :22, :37, :52)

H/15 \* \* \* \*

# every ten minutes in the first half of every hour (three times, perhaps at :04, :14, :24)

H(0-29)/10 \* \* \* \*

# once every two hours every weekday (perhaps at 10:38 AM, 12:38 PM, 2:38 PM, 4:38 PM)

H 9-16/2 \* \* 1-5

# once a day on the 1st and 15th of every month except December

H H 1,15 1-11 \*  
 如果使用 H Jenkins会自动提前一段时间连接节点，避免出现同一时间高并发的问题

Scheduled Uptime：超过任务时间后延迟多少分钟离线。如果此数值大于在线总时间(单位：分)，就会一直保持在线【必须为数字】

Keep on-line while jobs are running：当有Job在构建时(到达离线时间了)继续保持在线

* Take this slave on-line when in demand and off-line when idle：让Jenkins根据需求自动连接或者离线

In demand delay：告诉Jenkins如果有Job需要在此节点构建，需要在任务队列等待多长时间才会进入任务状态进行构建【必须为数字】

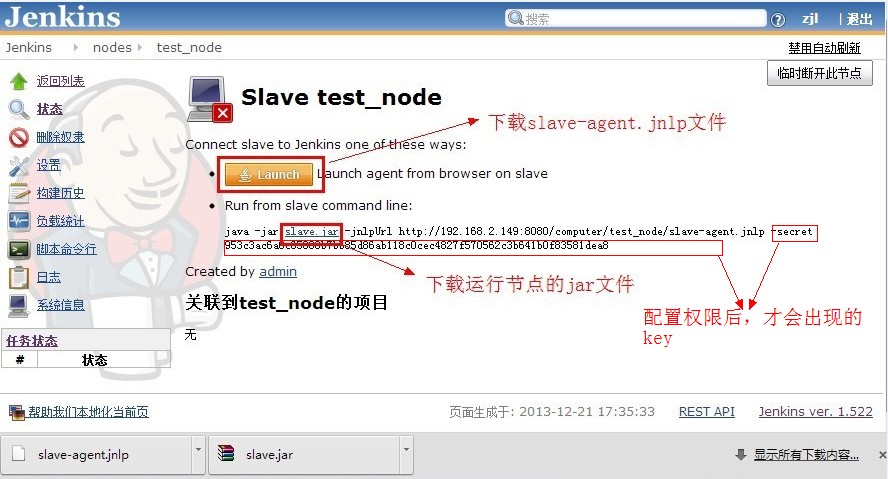
Idle delay：告诉Jenkins多少分钟内如果没有Job需要构建就离线【必须为数字】

Node Properties：

* Environment variables:配置环境变量(可以在脚本中引用，好方便啊\(^o^)/~)
* Tool Locations：工具的目录【推荐】。说明：可以替换系统设置的各种工具目录。如：JDK目录、Ant目录、Maven目录等。好处就是在不更改Job配置的情况下，不同环境(如：Windows和Linux) Job配置通用。好方便啊\(^o^)/~

到此，节点的配置工作完成了，点击Save保存。进入下一步节点的连接：

只介绍Launch slave agents via Java Web Start第二种连接方式，Launch slave agents on Unix machines via SSH点击保存后会自动连接。因为Launch slave via execution of command on the Master和Let Jenkins control this Windows slave as a Windows service选项没有配置测试过，所以大家自行折腾吧。



如上图所示，有两种方式可以启动节点(都是JNLP方式。JNLP连接需要端口，默认连接端口是随机的，端口更改 系统设置→Configure Global Security→JNLP节点代理的TCP端口)

你有两种方式启动：

1.Launch agent from browser on slave 下载文件slave-agent.jnlp文件，双击打开。

一般用在Windows系统上，需要javaws.exe(在Java的bin目录中可以找到)程序才能打开。如果提示错误，请卸载JDK后重新安装。成功启动如下图所示：



点击左上角的File选择Install as a service就可以添加为Windows的服务了(默认开机自动启动)。

2.Run from slave command line javaws http://xxxx/slave-agent.jnlp 如果你配置了权限那么后面还有一串看不懂的随机Key

下载slave.jar到本地，然后进入存放slave.jar的目录，复制粘贴并运行 java -jar slave.jar -jnlpUrl http://xxxxx 即可启动。

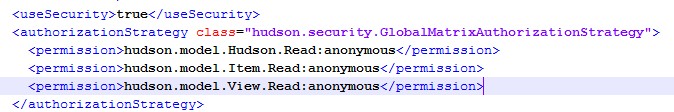
好了，Jenkins节点的配置展示告一段落了。如有疑问，请留言！

**13修改Jenkins权限控制**

很多童鞋在使用jenkins的时候忘记配置权限或者权限配置错误，然后各种蛋疼。最近闲着无事，折腾了下。好了，闲话少扯。

Jenkins的权限配置文件存放在JENKINS\_HOME目录。

进入JENKINS\_HOME目录，找到config.xml文件。嗯，就是它了。打开config.xml，里面有一堆的东西，找找。。。找到了<useSecurity>和<authorizationStrategy>节点。<useSecurity>节点代表是否使用用户权限，<authorizationStrategy>节点代表用户权限是怎么划分的。如图：



下面提供2种方法供大家使用：

1. 恢复默认设置

直接删除<useSecurity>和<authorizationStrategy>节点

1. 配置管理员权限

　　　　这种方法适用于已经存在一堆的权限，重新配置麻烦。

　　　　在<authorizationStrategy>节点中添加内容如下：

复制代码

　　<permission>hudson.model.Hudson.Administer:anonymous</permission>

<permission>hudson.model.Hudson.ConfigureUpdateCenter:anonymous</permission>

<permission>hudson.model.Hudson.Read:anonymous</permission>

<permission>hudson.model.Hudson.RunScripts:anonymous</permission>

<permission>hudson.model.Hudson.UploadPlugins:anonymous</permission>  
  
　　ps：anonymous可以更改成你的登录名。提供给大家的是匿名用户的配置。

复制代码

改完之后记得保存额，然后重启Jenkins。

最后给大家说说在配置文件里面怎么辨别使用是哪种权限控制模式

<authorizationStrategy>节点上有个class属性，这个属性控制着使用那种授权模式。

|  |  |
| --- | --- |
| hudson.security.FullControlOnceLoggedInAuthorizationStrategy | 登录用户可以做任何事 |
| hudson.security.ProjectMatrixAuthorizationStrategy | 项目矩阵授权策略 |
| hudson.security.GlobalMatrixAuthorizationStrategy | 安全矩阵 |
| hudson.security.LegacyAuthorizationStrategy | 遗留模式 |

**14配置Jenkins用户和权限**

使用Jenkins专有用户数据库的配置，和一些常用的权限配置。

# 配置用户注册

在已运行的Jenkins主页中，点击左侧的系统管理—>Configure Global Security进入如下界面：



设置如上图，保存后系统管理中就出现管理用户的选项。页面右上角也会出现登录/注册的选项。

点击注册：



点击sign up按钮，提示你现在已经登录.

# 配置用户权限

点击左侧的系统管理—>Configure Global Security

选择授权策略中的安全矩阵或者项目矩阵授权策略

说明：安全矩阵和项目矩阵授权策略的配置是一模一样的，唯一的区别是项目矩阵授权策略支持在Job的配置页面再次配置授权策略。

添加用户到矩阵中，如下图。



各种权限如下(在配置页面将鼠标放到该权限上即可查看帮助)：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Overall(全局) | | | | | Credentials(凭证) | | | | | Slave(节点) | | | | | | Job(任务) | | | | | | | | View(视图) | | | |
| Administer | Read | RunScripts | UploadPlugins | ConfigureUpdateCenter | Create | Update | View | Delete | ManageDomains | Configure | Delete | Create | Disconnect | Connect | Build | Create | Delete | Configure | Read | Discover | Build | Workspace | Cancel | Create | Delete | Configure | Read |
| 管理员(最大) | 阅读 | 运行脚本 | 升级插件 | 配置升级中心 | 创建 | 更新 | 查看 | 删除 | 管理域 | 配置 | 删除 | 创建 | 断开连接 | 连接 | 构建 | 创建 | 删除 | 配置 | 阅读 | 重定向 | 构建 | 查看工作区 | 取消构建 | 创建 | 删除 | 配置 | 阅读 |

其中有一些比较特别的权限：

最大的权限是Overall的Administer，拥有该权限可以干任何事情。

最基本的权限是Overall的Read，用户必须赋予阅读的权限，不然什么都看不到。

Job的Discover权限是一个奇葩的权限，帮助说Discover比Read的级别更低。如果匿名用户(没有访问job的权限)直接访问一个Job的Url将重定向到登陆页面。(经测试，这个权限应该是被废弃了。)

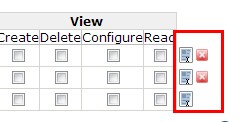
Credentials的ManageDomains这个权限没有看懂干嘛的，有懂的大家一起交流哈！

ps：如果有个用户被赋予了Overall的Read，并没有被赋予Job的Read权限，那么该用户就无法访问job。原因：没有权限。

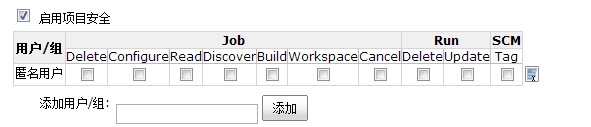
其他都是一些基本的权限，大家根据自己的需求选择。

小技巧：

每个用户后都有1-2个图标，第一个是反选功能(删除当前已选择的权限，选择其他所有权限)，第二个是删除功能(删除该用户)



在Job中配置项目安全，如下图：



**00答疑解惑**

Jenkins其实就是一个工具，这个工具的作用就是调用各种其他的工具来达成你的目的。比如你要获取Subversion上最新的源代码，Jenkins会去调用SVNKIT(插件的核心Jar的名称)。然后你需要编译源代码(假设是用maven编译)，Jenkins会去调用maven的插件。最后你可能需要发布程序到服务器上(假设是使用的Tomcat提供服务)，你可以通过Tomcat自己的API发布程序(有个插件就是通过这个功能实现的发布)，你也可以通过ssh通道自己写shell脚本去实现。总之，Jenkins就是一箱工具，在它能力范围内你想干什么都可以。

1. Q:Jenkins是什么？

A:百度一下你就知道！

1. Q:Jenkins有哪些版本？

A:这个版本太多了，具体去看官网吧！www.jenkins-ci.org

1. Q:应该选择哪个版本的Jenkins？

A:如果你是公司正式使用推荐长期支持版(LTS)，原因：稳定。如果你是学习，随便哪个版本都可以。

1. Q:JDK应该安装哪个版本的？

A:推荐安装JDK7，原因：某些插件必须要求JDK7的支持。比如：Monitoring 插件

1. Q:Tomcat应该用哪个版本的？

A: Tomcat6或者7都可以。如果你要用Tomcat5.5的版本，我劝你趁早扔掉。不然一堆稀奇古怪的错误，看得你头晕。

1. Q: 可以直接用命令启动么？

A: 可以。命令：Java –jar jenkins.war –httpsPort=8443 –httpPort=-1

httpsPort是访问的端口，httpPort=-1代表关闭http访问方式启用https。

1. Q:Jenkins的帮助在哪儿？

A: 每个配置项后都有一个http://images.cnitblog.com/blog/529896/201401/161619458149.jpg帮助按钮，有疑问可以尝试自己解决。

1. Q:有哪些使用Jenkins的案例？

A: <https://builds.apache.org/>

<https://ci.jenkins-ci.org/>

1. Q：使用Jenkins需要什么技能？

A:推荐你学习这些：Windows的批处理、ant脚本、shell脚本、maven、版本控制软件相关知识。如果你会Java就更好了，因为Jenkins是用Java开发的。

1. Q：Jenkins的svn插件无法获取到最新的源代码？

A：请检查你的SVN服务器时间和Jenkins服务器时间是否同步。

**15Maven获取Jenkins的Subversion的版本号**

Maven怎么在Jenkins中获取Subversion的版本号。

Jenkins自己的环境变量中包含了很多有用的的参数，详情查看：http://xxxx/env-vars.html

从环境变量中可以发现，我们可以直接获取到Subversion的版本号。

最重要的一点解决了，下面我们就要想办法让Maven获取到Jenkins环境变量的参数。

Maven提供了它自己的解决方案：

　　在《Maven权威指南》中文版第266页提到，在任何Maven项目中都有一些隐式的属性，这些隐式的属性是：

　　　　project.\*

　　　　　　Maven的项目对象模型(POM)。你可以使用该project.\*前缀来引用任何在Maven POM中的值。

　　　　settings.\*

　　　　　　Maven Settings。你使用该settings.\*前缀来引用~/.m2/settings.xml文件中Maven Settings的值

　　　　env.\*

　　　　　　环境变量如PATH和M2\_HOME都可以使用env.\*前缀来引用。

通过上面的介绍知道，可以通过${env.SVN\_REVISION}参数获取到Jenkins中的Subversion版本号。

好了，Maven中已经获取到需要的内容了。现在你想干什么就干什么O(∩\_∩)O哈哈~

最后给大家上一张我做测试的图，有图有真相额：

