# [使用Jenkins自动编译部署web应用](http://www.cnblogs.com/hhhyaaon/p/5854977.html" \t "http://blog.csdn.net/gavid0124/article/details/_blank)

# 写在前面

初步接触持续集成自动化过程，本篇主要介绍基于Jenkins实现持续集成的方式，通过案例介绍线上自动编译及部署的配置过程

# 持续集成

[持续集成](http://baike.baidu.com/link?url=8FOvoIYfLbreDpdQ_mZAOIT6BFf_g5xgBRS3A7_6lIBkit5PkYPmWQbvg1cDBFyLUwSXT1u-Z5FYJFnMuDvqtK" \t "http://blog.csdn.net/gavid0124/article/details/_blank)是一种软件开发实践，即团队开发成员经常集成它们的工作，通过每个成员每天至少集成一次，也就意味着每天可能会发生多次集成。每次集成都通过自动化的构建（包括编译，发布，自动化**[测试](http://lib.csdn.net/base/softwaretest" \o "软件测试知识库" \t "http://blog.csdn.net/gavid0124/article/details/_blank)**）来验证，从而尽早地发现集成错误。

# Jenkins

Jenkins是基于**[Java](http://lib.csdn.net/base/java" \o "Java 知识库" \t "http://blog.csdn.net/gavid0124/article/details/_blank)**开发的一种持续集成工具，用于监控持续重复的工作，功能包括：  
1、持续的软件版本发布/测试项目。  
2、监控外部调用执行的工作。

更多信息请戳[官网](https://jenkins.io/" \t "http://blog.csdn.net/gavid0124/article/details/_blank)

# 安装

网上有很多Jenkins安装配置教程，详情请戳[Jenkins安装与配置](http://jingyan.baidu.com/article/d169e186525b50436711d850.html" \t "http://blog.csdn.net/gavid0124/article/details/_blank)

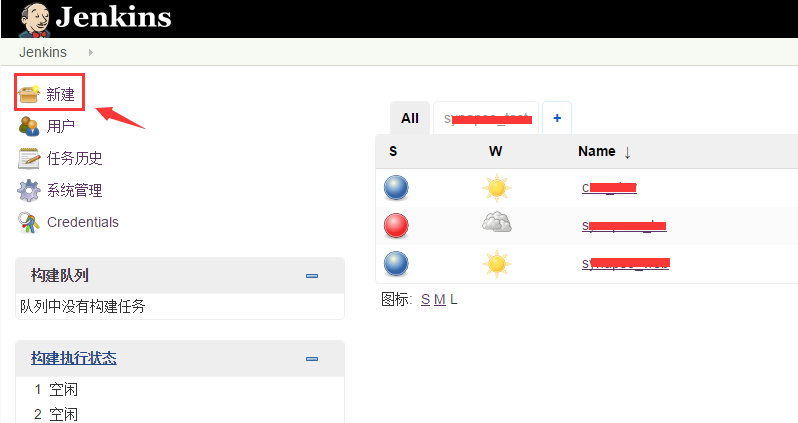
# 开始一个DEMO

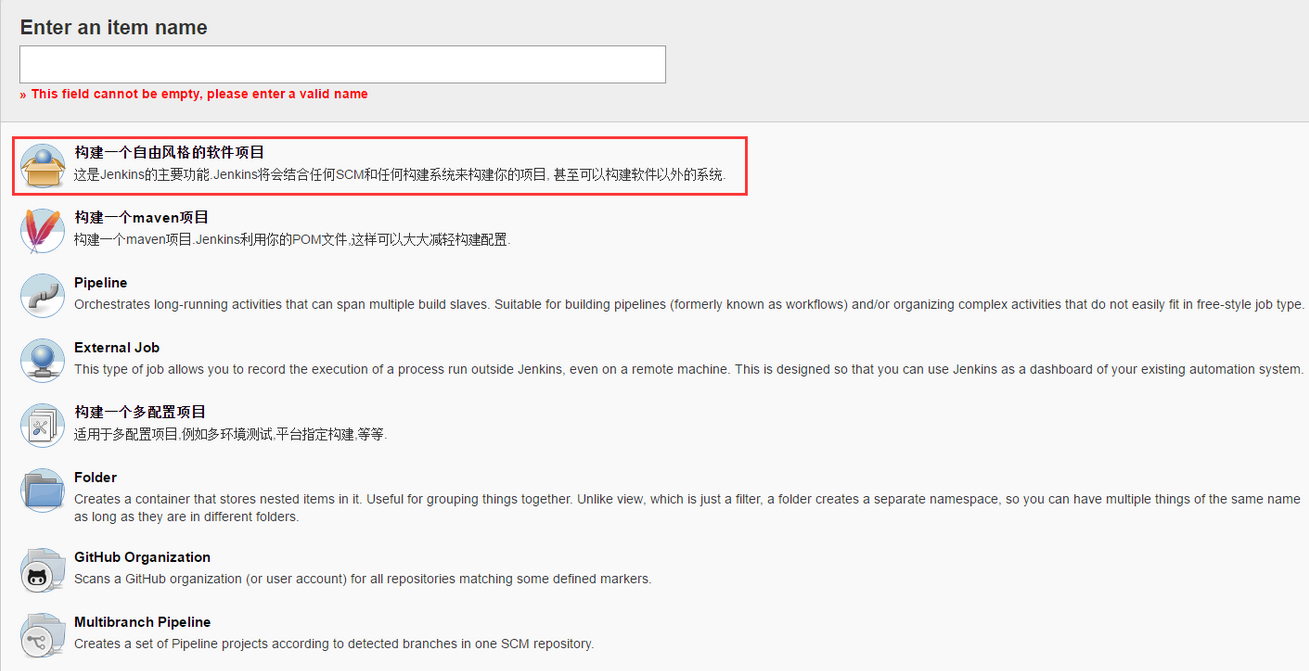
## 背景

Jenkins运行在**[Linux](http://lib.csdn.net/base/linux" \o "Linux知识库" \t "http://blog.csdn.net/gavid0124/article/details/_blank)**服务器上，尝试在此服务器上编译并部署一个web应用，编译过程需要node环境，编译后文件需ssh到测试服务器

Jenkins主要完成任务如下：  
1、自动集成编译环境  
2、线上编译应用  
3、ssh到目标环境并运行

## 新建一个Job

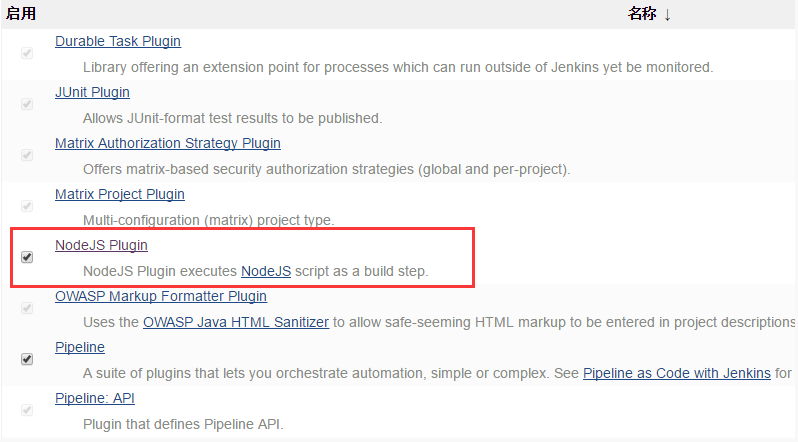
登录进入Jenkins，可以看到当前所有已集成项目的构建情况；  
1、点击左上角的【新建】，创建一个新任务  


2、填写任务名称，构建的项目类型可根据实际情况进行选择，本次选择第一种即可  


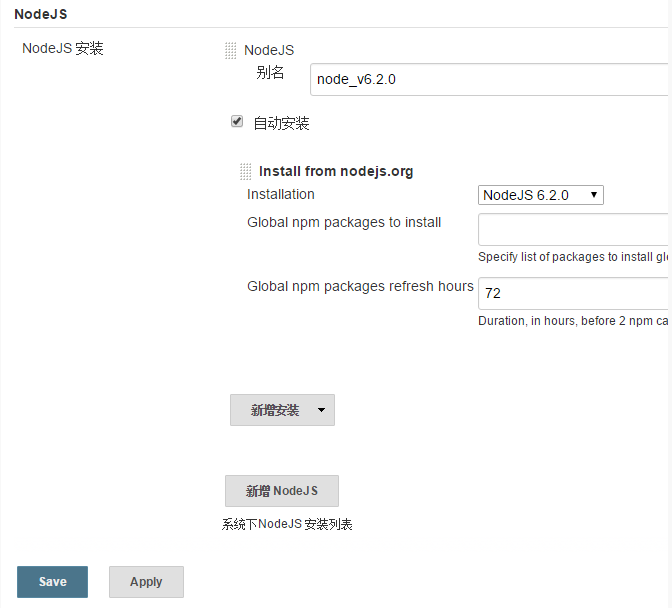
3、选择完毕，点击【OK】，回到首页，发现创建的Job已显示在列表中

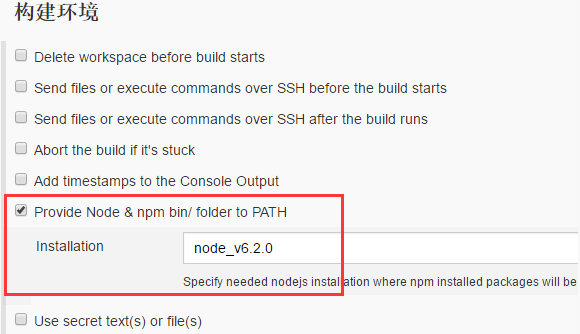
## 配置编译环境

在这个过程中，我们需要为应用配置基于node的编译环境。  
关于安装node，一般可以下载执行文件安装或编译源码安装，而通过Jenkins，我们则可考虑使用其提供的插件进行自动安装  
1、选择系统管理->管理插件  

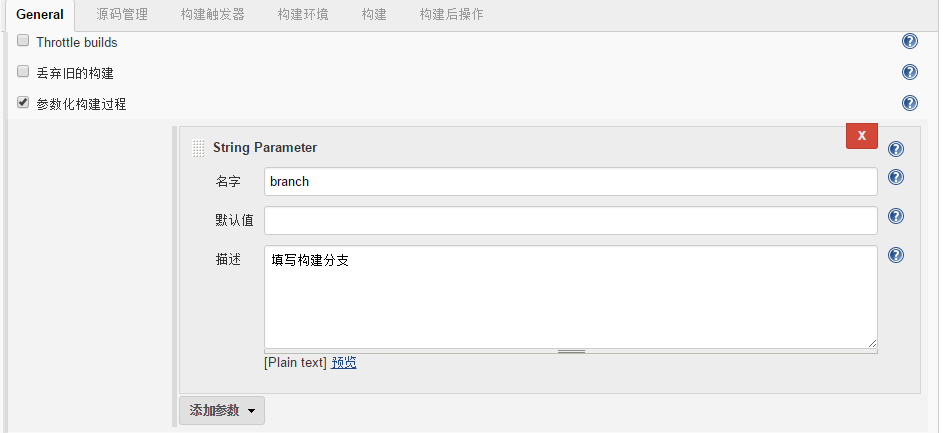

2、可以看到，Jenkins提供了丰富的插件供开发者使用，找到我们需要的[NodeJS Plugin]，勾选后点击安装即可  


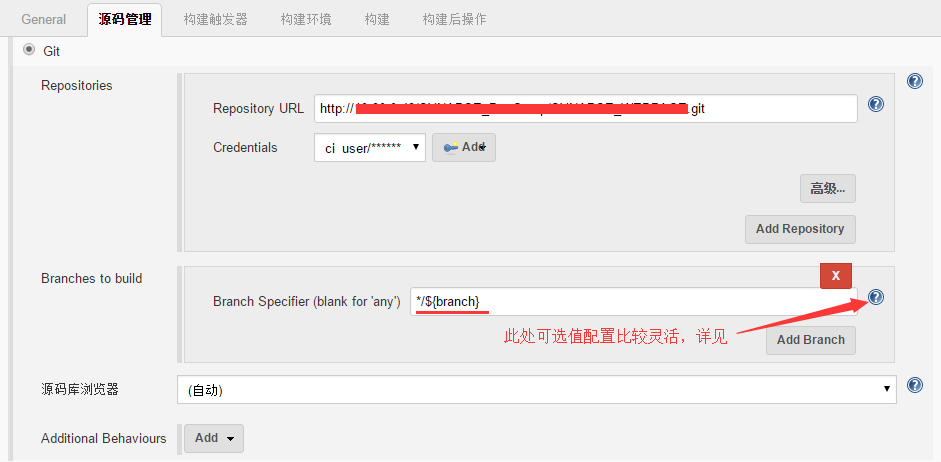
3、安装完毕后，选择系统管理->Global Tool Configuration，配置node下载及安装  


4、进入后，撸到底部，发现全局可用工具中已经出现了NodeJS的配置项  
5、插件支持多种方式安装nodeJS，点击[新建安装]，选择习惯的安装方式即可，此处给出从node官网下载并安装的配置  


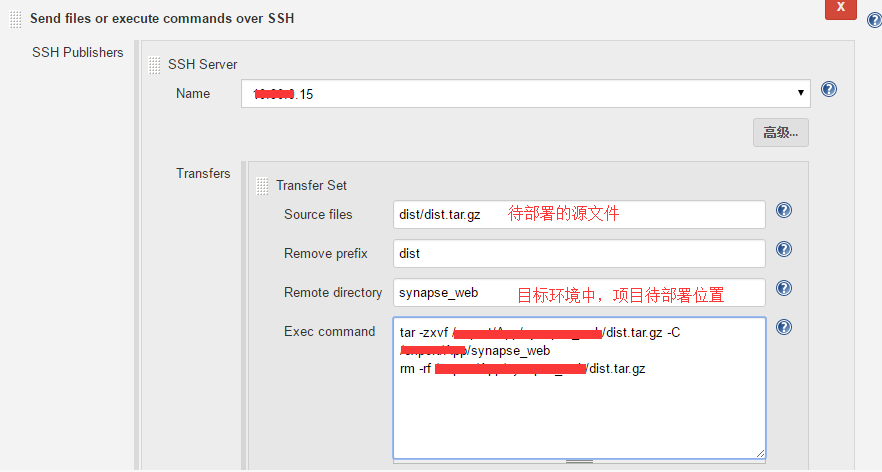
6、保存完毕后，通过列表，进入刚刚新建的Job，找到【构建环境】，勾选Provide Node & npm bin/ folder to PATH，选择刚刚添加的配置即可；此后当每次build，都会首先执行环境构建，环境构建无误后，才会开始真正的构建过程  


## 配置构建过程

在这个过程中，我们可以配置构建所需的参数，用来指明构建的位置（从哪个分支/哪个版本...）或其他一些状态；并通过命令执行构建。  
1、通过列表进入Job详情，开始相关配置；  
2、【General】一栏需要配置项目基础信息和一些构建需要的参数。选择【参数化构建】，此构建方式允许我们自定义构建过程中的参数。如下：配置一个branch参数，用于指明要构建的**[Git](http://lib.csdn.net/base/git" \o "Git知识库" \t "http://blog.csdn.net/gavid0124/article/details/_blank)**分支  


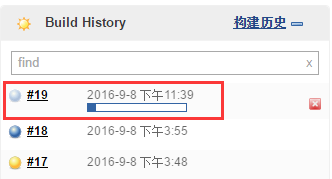
3、【源码管理】一栏需要填写待构建项目的源码地址，其中【Branches to build】用来指明待构建的分支，当然，也可以选择某个提交号，同时此处也可以使用我们在General中定义变量（详细配置见输入框右侧的[帮助]按钮）  


4、【构建】一栏需要输入相关命令来实现自动化构建，此过程在【环境构建】后，故我们可以编写如下代码执行构建  


5、线上编译完毕，下面将build完成的文件发送至目标环境（测试环境/生产环境）。继续在【构建】一栏中添加配置，指定目标环境相关信息；命令行配置ssh后的一些操作（解压文件，并删除压缩包）  


## 执行构建

至此，整个构建的配置算是基本完成，下面开始构建。  
回到列表页，点击构建即可：  


可以看到构建历史列表中显示当前构建的进度  


点击左侧圆形按钮，即可进入控制台查看输出  
构建完毕后，可以通过列表中的晴雨状态查看构建结