jenkins学习笔记

# 什么是持续集成

1.1什么是持续集成

持续集成是一种软件开发实践，即团队开发成员经常集成他们的工作，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，每次的集成都通过自动化的构建（包括编译，发布，自动化测试）来验证，从而尽早发现集成错误。简单来说，就是持续的定时的在多个团队成员的工作中进行集成，并且给予反馈。

持续集成需要开发人员一天多次的将代码集成到主干，并进行自动化编译、测试等操作，由于这种频繁集成，以及集成后及时开始的编译和测试，可以有效避免我们在提交代码时没有进行必要检查而导致的错误，以及一些超出预期效果的更改，从而保证代码的质量。

由于这种及时性，如果在一次提交后项目集成失败，可以快速的在这次提交中查找问题所在，缩小了找问题的范围，从而减少了一些debug时间。同时如果按照这种实践，那么我们的主干代码时刻都是正确的，这样我们可以更频繁的交付。

1.2为什么要使用持续集成

一般规模较小的项目，对外部系统的依赖和服务调用很小，对于软件的集成不是问题。但是随着软件复杂度的增加，对集成提出了更多的要求，持续集成的好处就体现出来了。

1）对重复的编译发布等操作进行抽象，减少重复过程。

2）及早发现各种冲突和错误，减少风险。

3）任何时间、任何地点生成可部署的软件

1.3怎么持续集成

基本要求：要将这种实践付诸实际，需要一些必要的条件，如下

1）一个自动构建过程，包括自动编译、分发、部署和测试等

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。

3）一个持续集成服务器。

自动化构建成过程，可帮助我们节省大量时间，完成这个过程的自动化后，在以后的开发过程中，我们需要做的，就是只是提交代码到版本库中，构建自动完成，基本不再需要人工干预。

代码仓库作为构建的素材库，构建所需的代码从代码库中获得。

最好有一台服务器单独作为持续集成服务器，一方面保证了环境的纯净，一方面不影响开发，而且持续集成服务器一般是随时准备开始构建的，所以一般也不关机。

1.4持续集成工具

jenkins

travis

gitlab

buddybuild

仅列举了一些典型的，Jenkins 是传统型的工具，前身是 Hudson，04 年到现在已经有十多年的历史，后几个是最近几年出现的新一批，多少都和容器技术有点关系，这里我们主要介绍 Jenkins，因为这个工具比较常用，各种开发实践都可以通过大量的插件来组合实现，可定制性好很多。

# 2.jenkins介绍

Jenkins是一个开源软件项目，是基于Java开发的一种持续集成工具，用于监控持续重复的工作，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。同时也可以处理任何类型的构建或持续集成。

Jenkins是一种软件允许持续集成，jenkins安装在一台服务器也就是构建发生的地方

2.1简单的流程图：

Jenkins提取代码，并部署到环境之中，也可以做一下测试（单元测试）

开发人员提供提交代码

2.2jenkins的功能

1）.定时拉取代码并编译

2）.静态代码分析

3）.定时打包发布测试版

4）.自定义额外的操作，如跑单元测试等

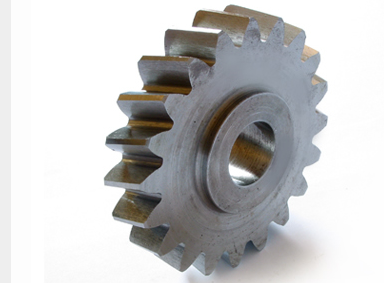
5）.出错提醒

基本上都是持续集成实践中的要求和周边的一些实现措施，如提醒功能等，出错后及时提醒开发者修复，Jenkins 中通过配置 SMTP 配置信息（这个一般的邮件服务提供商都有提供），邮件模板等，创建事件触发器，在事件（如编译失败）发生时，及时发送邮件通知开发者，挺方便的。

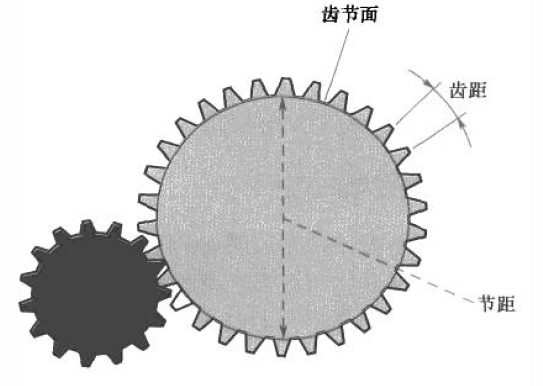
Jenkins 有很多种触发构建的方式，如 webhook，定时更新代码等，同时可以在触发构建后执行自定义的构建操作，通过编辑自定义的构建脚本，几乎可以进行任何构建操作。

2.3对jenkins的理解

1）齿轮：如果将java/maven/ant/git/tomcat/jenkins等软件比喻为齿轮，如下图



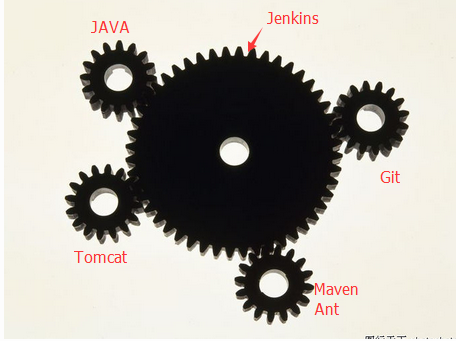
2）两个软件在一起可以驱动另外一个软件，如下图



3）如果把这些软件要集成在一起工作，那么这个软件就可以存在其他软件的中间来驱动各个软件工作，如下图



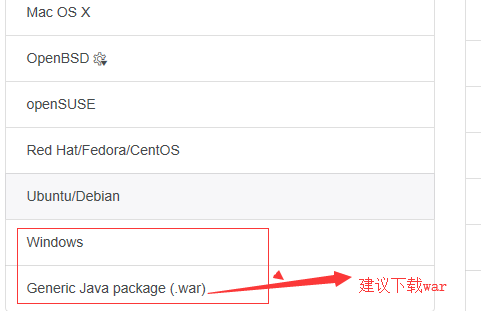
jenkins就是类似中间那个齿轮，来驱动其他软件的集成一起工作，如下图



# 3.jenkins在windows下的环境配置

3.1jenkins的下载

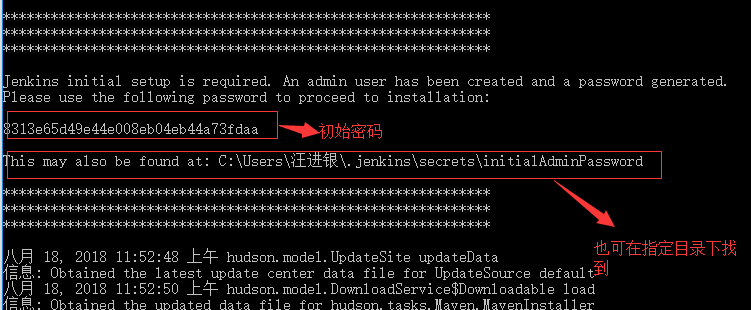
第一步：官网下载：<https://jenkins.io/download/>



放在d:Jenkins目录下

第二步：运行 Win+R 打开 命令行窗口 进入d:Jenkins目录下

运行：java -jar jenkins.war



第三步：在浏览器中访问：<http://localhost:8080> 默认端口号为8080，若与tomcat服务器端口号冲突也可修改其默认端口号

首次可以修改登录账号和密码 本人已修改



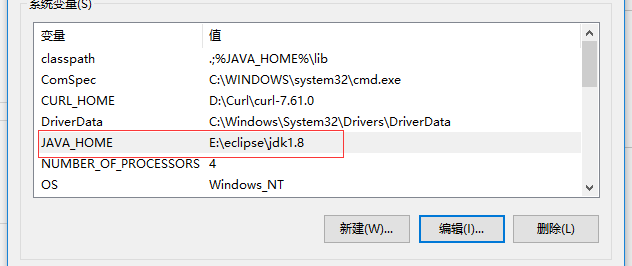
Web 管理页面



3.2Jenkins+maven+git/svn搭建项目自动化部署集成环境

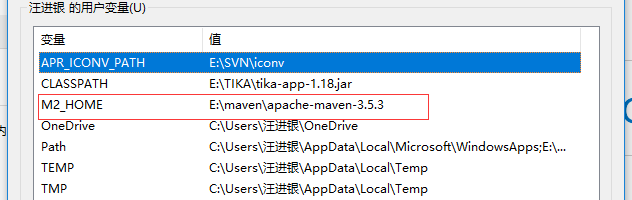
3.2.1 jdk的安装及配置环境变量

配置java\_home E:\eclipse\jdk1.8



3.2.2 安装maven

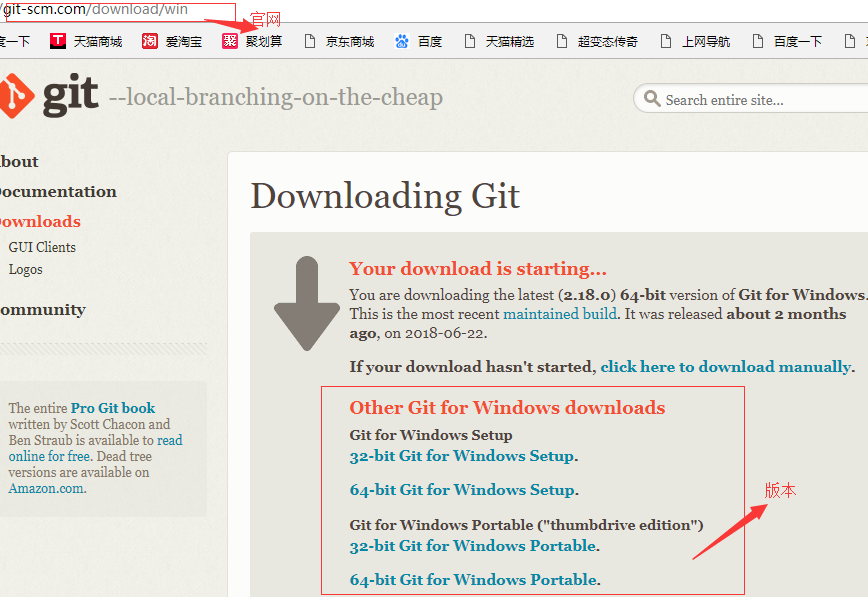
安装过程省略

环境变量配置: maven\_home

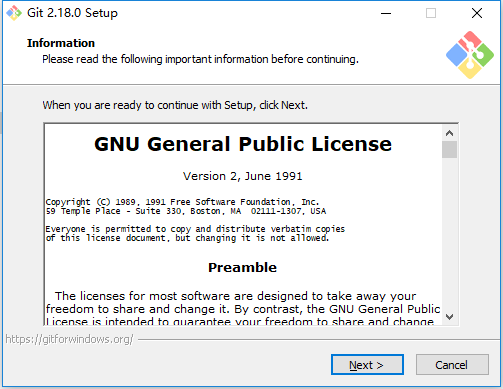
3.2.3 git安装

1）Git的目的：通过git管理github托管项目代码

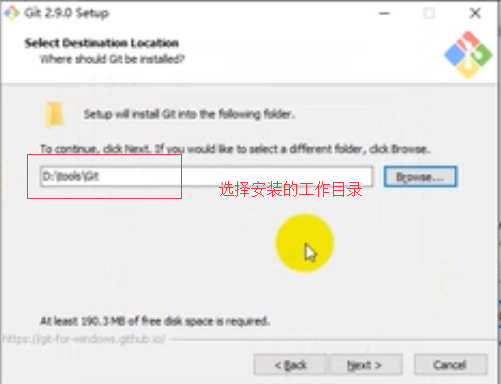
第一步：下载git 官网：https://git-scm.com/download/win



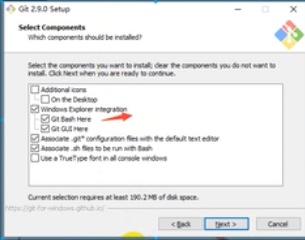
第二步：Git-2.18.0-64-bit.exe 双击下载的.exe文件



选择安装的工作目录

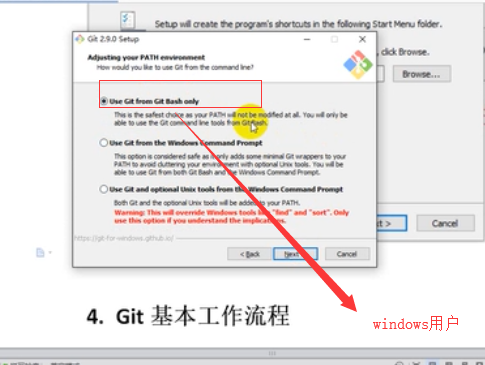


选择组件（默认下一步）



下一步：默认下一步

选择使用的命令行环境



以下三步：傻瓜式安装 全部点击next

验证是否安装成功

2）.git的工作流程：

暂存区：暂存已经修改的文件，最后统一提交仓库中。

工作区域： 添加，编辑，和修改文件

仓库（repository）:最终将文件保存到仓库中

Git commit –m “描述”

暂存区：暂存已经修改的文件，最后统一体检到仓库中

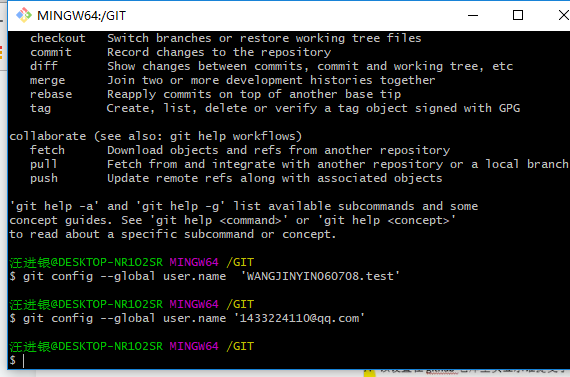
Git.add

工作区：创建，修改 文件等工作

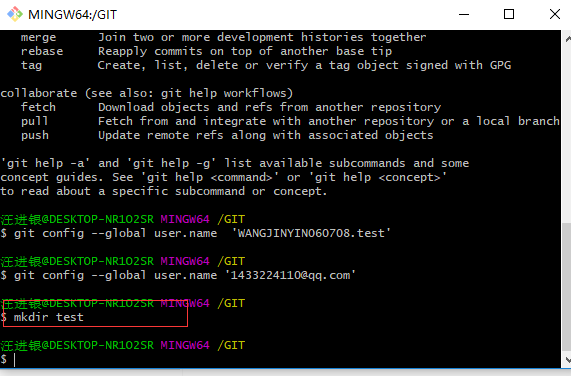
3）.git初始化及仓库创建和操作

基本信息设置

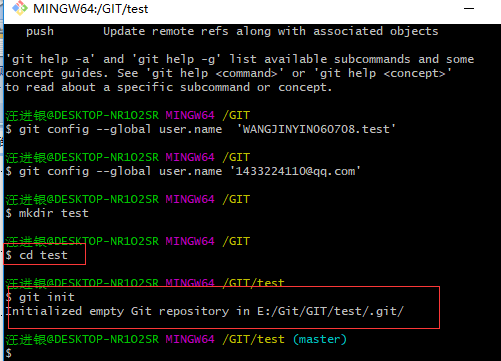
|  |
| --- |
| 1. 用户名 git cofig --global user.name 2. 邮箱 git cofig --global user.name 1433224110@qq.com |

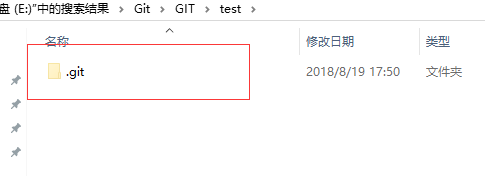
 初始化一个Git仓库

第一步：新建一个文件夹



第二步：创建一个仓库

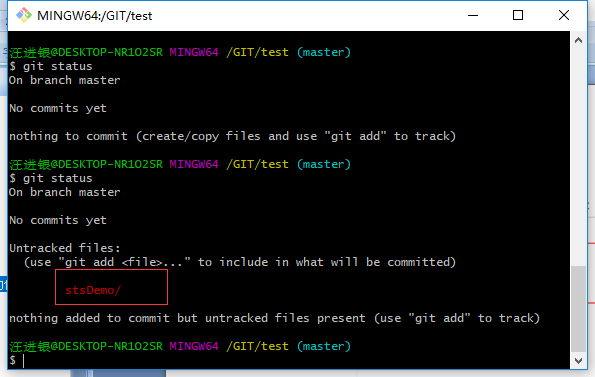




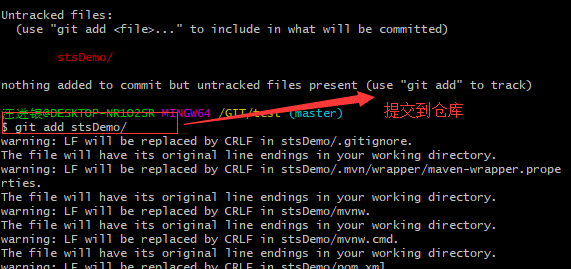
本地仓库配置完毕

第三步：向仓库中添加文件

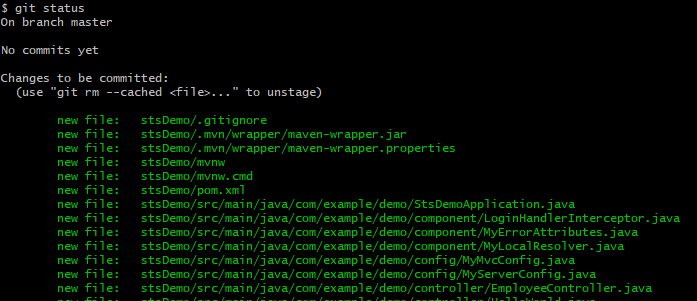
将工程文件上传到test文件夹下（用touch 命令或直接拷贝文件到文件夹下）



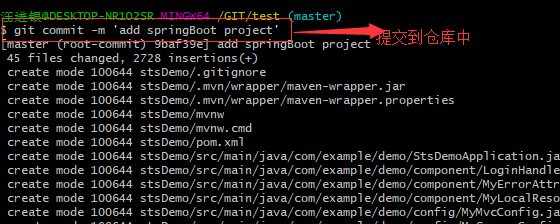
提交到暂存区中git add ….

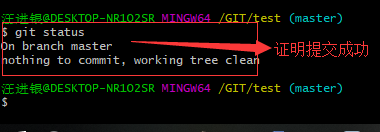


查看当前的状态



提交仓库中

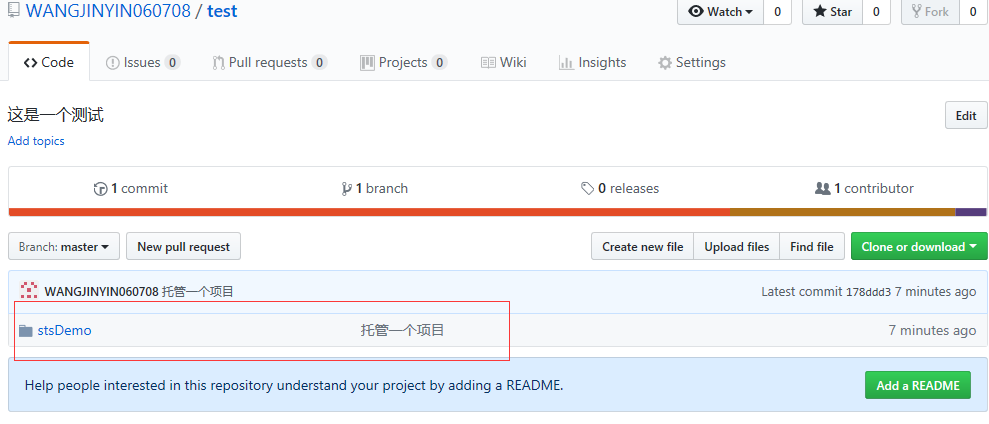




修改仓库

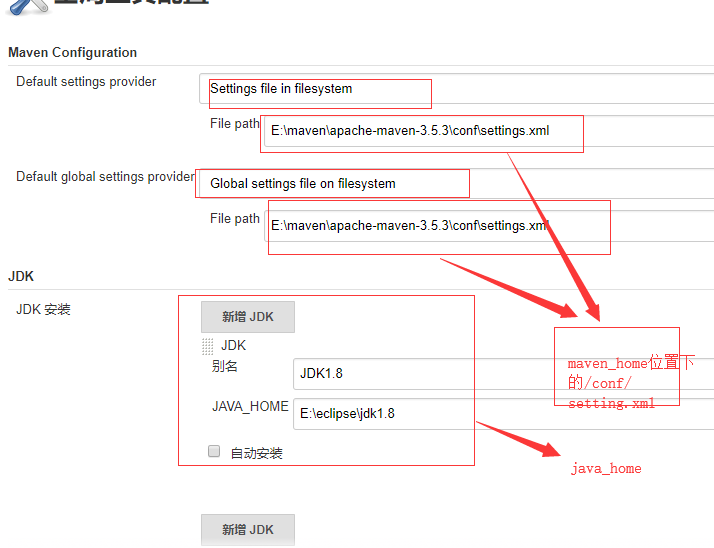
删除仓库

Git远程仓库怎么将本地仓库的代码提交到远程仓库上

* 第一步：git 克隆操作
* 复制github上的 地址 <https://github.com/WANGJINYIN060708/test.git>
* 使用git clone <https://github.com/WANGJINYIN060708/test.git>
* 在执行代码提交
* Git add …
* Git commit
* Git push
* 

3.2.4 将maven，jdk，git 集成到jenkins

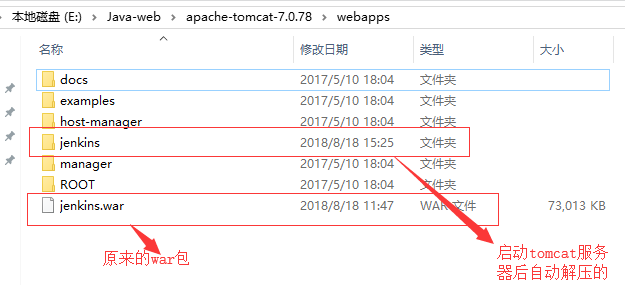




git集成

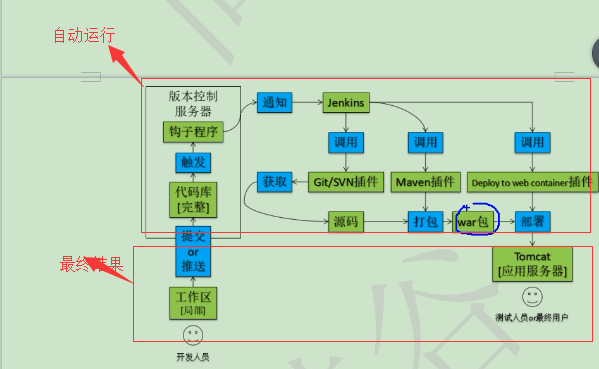
3.3jenkins部署到tomcat服务器上

直接将下载下来的jenkins.war拷贝到 E:\Java-web\apache-tomcat-7.0.78\webapps目录下



# 4.jenkins的使用

整个流程图



4.1新建一个maven工程（apple）

|  |
| --- |
| package com.nanhua.study;  public class Apple {    private String app="tom";    public String getApp() {  return app;  }  } |

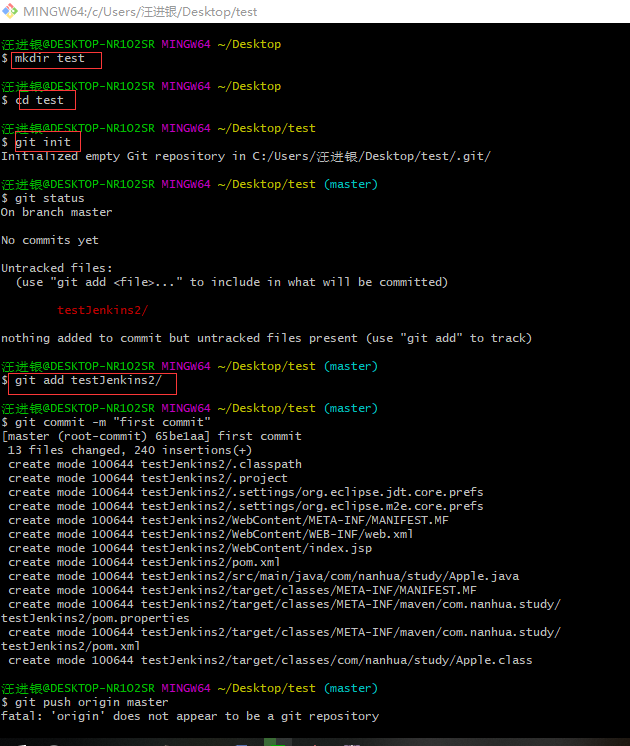
3.2pom.xml

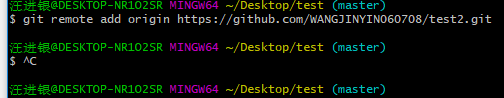
|  |
| --- |
| <dependencies>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>javax.servlet-api</artifactId>  <version>3.0.1</version>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>javax.servlet.jsp</groupId>  <artifactId>jsp-api</artifactId>  <version>2.1</version>  <scope>provided</scope>  </dependency>  </dependencies> |

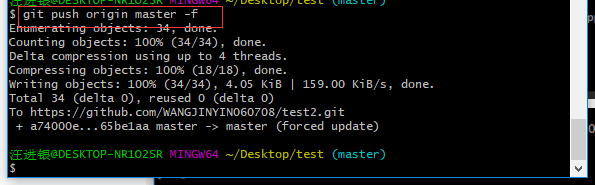
3.3jsp页面

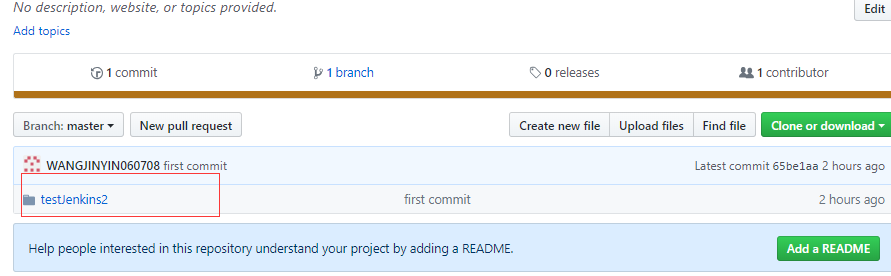
|  |
| --- |
| <%@page import="com.nanhua.study.Apple"%>  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">  <title>Insert title here</title>  </head>  <body>  <%= new Apple().getApp()%>  </body>  </html> |

4.2将代码提交到github上





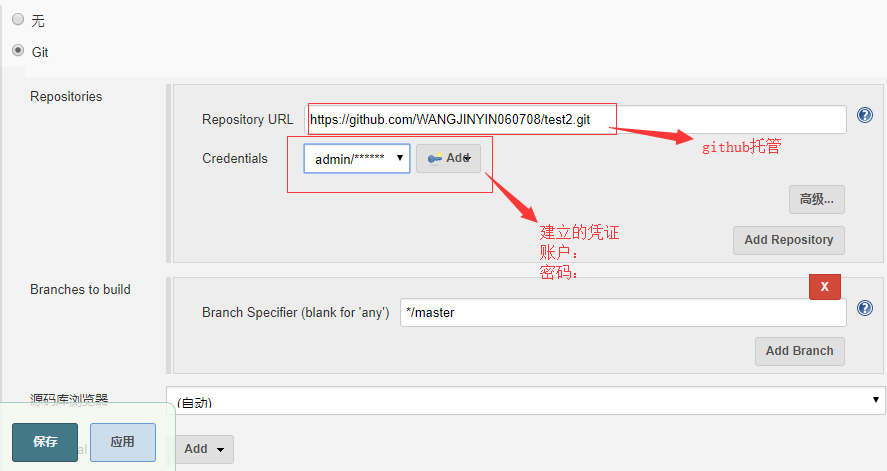




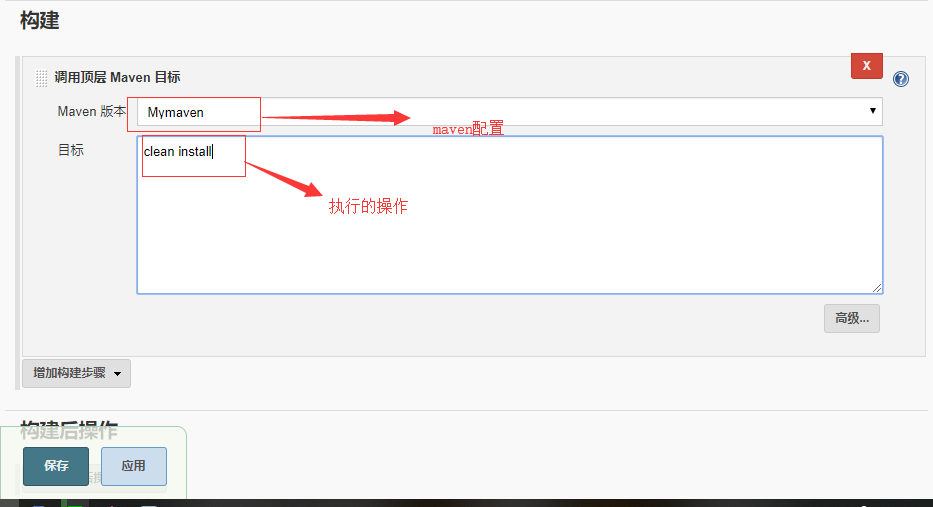
4.3新建job



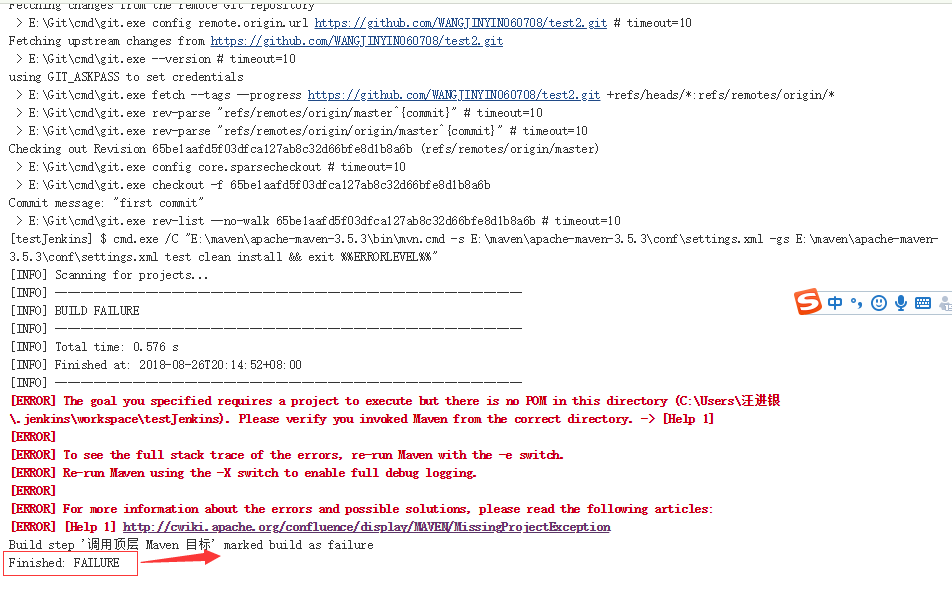
4.3.1Git配置



4.3.2Maven的配置（Mymaven是自己配置的全局maven）

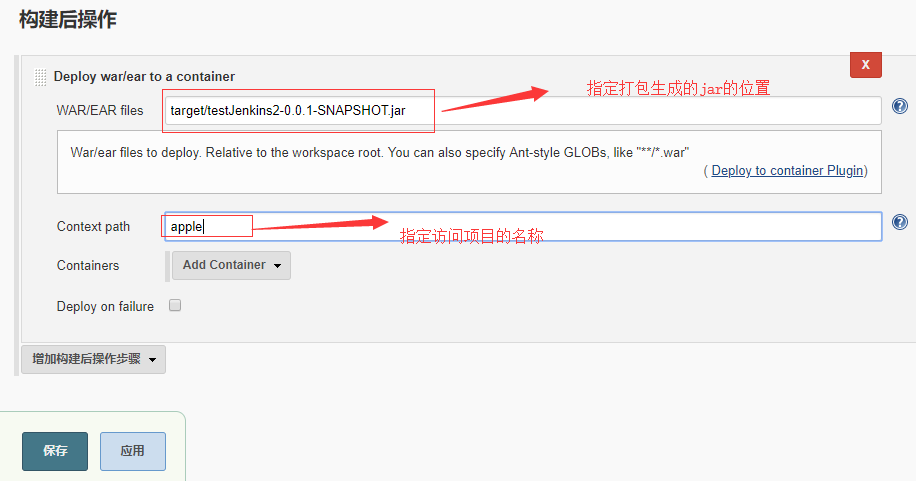


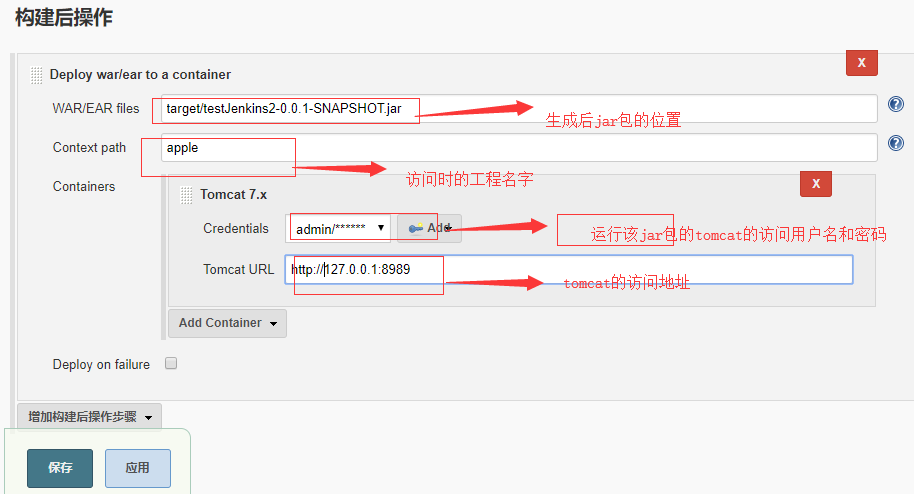
4.3.3立即构建（如构建成则会生成一个war或jar包）



4.3.4构建后的配置：

构建后会自动生成war包: 部署到tomcat服务器上





4.3.5 配置自动构建的地址

当访问固定的地址就触发远程构建 当访问该地址<http://127.0.0.1:8989/jenkins/job/testJenkins/build?token=NANHUA_TOKEN> 时就会触发构建从而达到自动构建的目的



4.3.6利用curl来触发构建

Linux 的curl 命令用来发送HTTP 请求。

-X 参数：指定请求方式

-v 参数：显示响应结果

-u 参数：携带用户名/密码

-H 参数：携带请求消息头信息

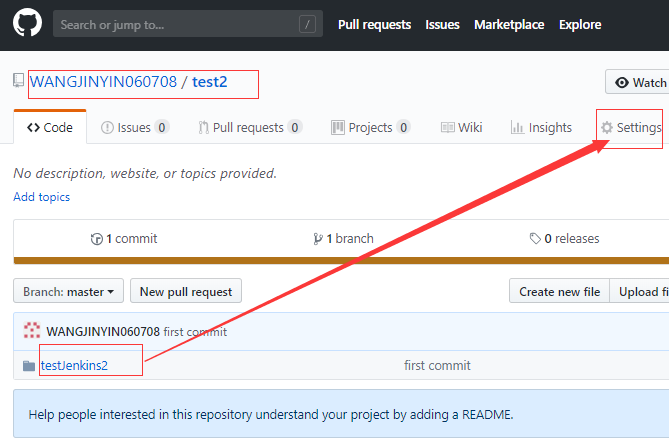
|  |
| --- |
| curl -X post -v -u [Jenkins 用户名]:[Jenkins 密码] -H "请求消息头信息" http://[服务器IP 地址]:[服务器端口  号]/jenkins/job/[Jenkins 项目名称]/build?token=[身份验证令牌] |
| curl -X post -v -u admin: 8313e65d49e44e008eb04eb44a73fdaa -H  "Jenkins-Crumb:88a12946e07d82b3b0d567c7c4610c9a"  <http://127.0.0.1:8989/jenkins/job/testJenkins/build?token=NANHUA_TOKEN> |

4.3.7钩子程序 将生成的curl明亮注入到钩子程序中

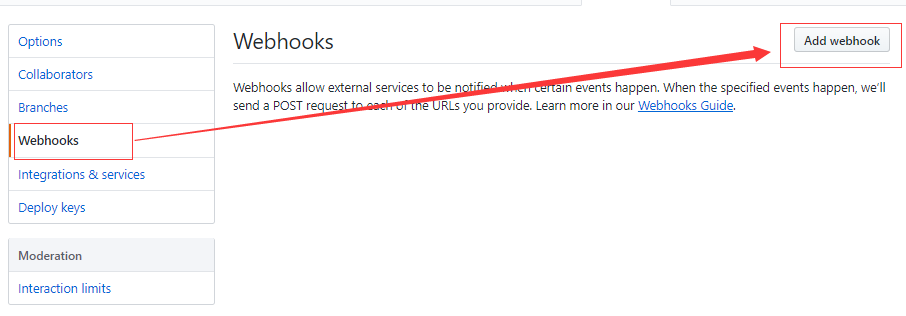
针对于svn

而对于github需要去配置WebHook 相当于svn的钩子程序

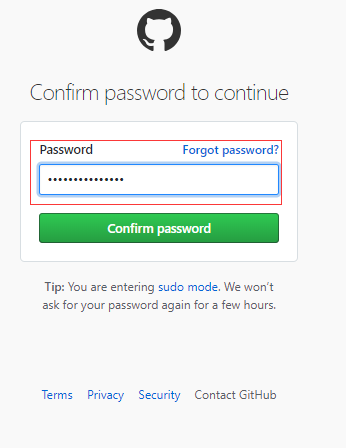
第一步：登录github 进入你需要配置的仓库



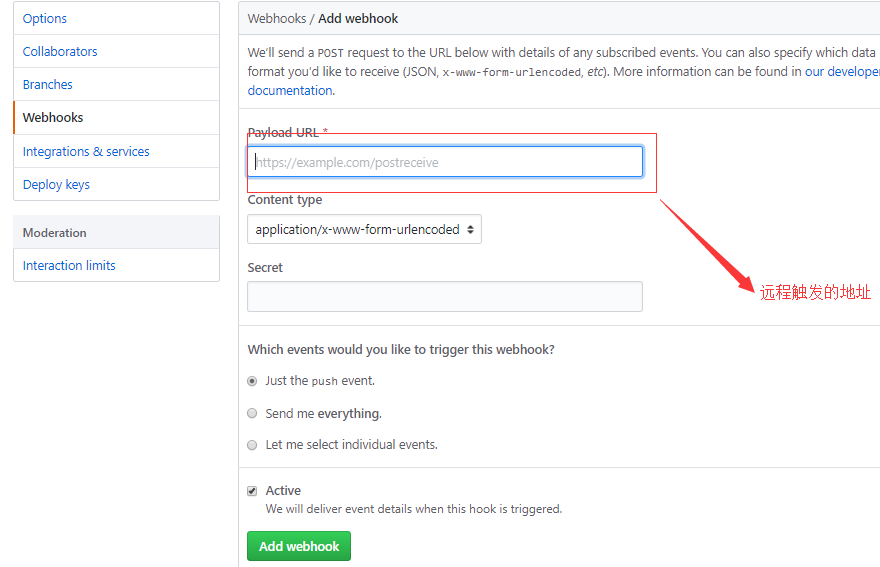
第二步：增加webHook



第三步：密码验证



第四步：配置webHook的地址



第五步：点击确定就行

最后一步： 在Jenkins 内关闭“防止跨站点请求伪造”