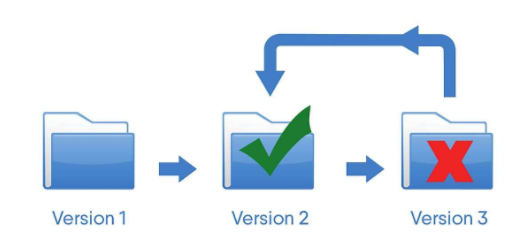
Git&Github

###### 1. 版本控制工具简介

**1.1什么是版本控制工具**

版本控制工具提供完备的版本管理功能，用于存储、[追踪](https://baike.baidu.com/item/%E8%BF%BD%E8%B8%AA/44928)目录（文件夹）和文件的修改历史，是软件开发者的必备工具，是软件公司的基础设施。

版本控制最主要的功能就是追踪文件的变更。它将什么时候、什么人更改了文件的什么内容等信息忠实地了已录下来。每一次文件的改变，文件的版本号都将增加。除了记录版本变更外，版本控制的另一个重要功能是并行开发。[软件开发](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%BC%80%E5%8F%91/3448966)往往是多人协同作业，版本控制可以有效地解决版本的同步以及不同开发者之间的开发通信问题，提高协同开发的效率。并行开发中最常见的不同版本软件的错误(Bug)修正问题也可以通过版本控制中分支与合并的方法有效地解决。



版本控制工具的作用：

（1）协同开发

团队协作共同完成同一个项目

（2）版本管理

以不断提升项目版本的方式逐步完成项目。

（3）数据备份

开发中以版本控制的形式保存每一个历史版本。

（4）权限控制

对团队开发人员进行不同的权限分配。

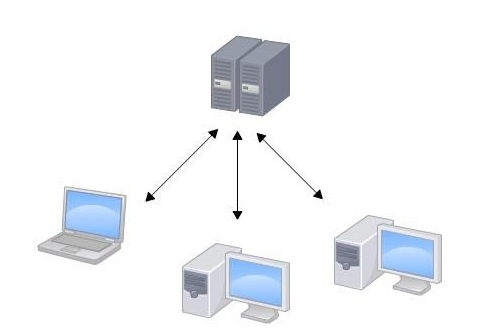
（5）分支管理

允许开发团队在工作过程中多条生产线同时推进任务，进一步提高效率。

**1.2 市场上常见的版本控制工具及分类**

**a. 集中式版本控制工具**

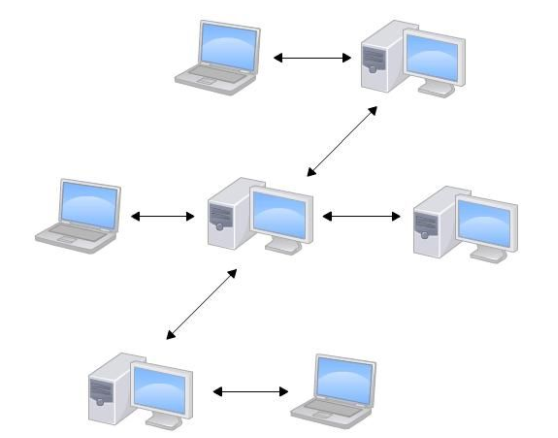
CVS、**SVN**、VSS……

****

不同的客户端直接跟服务器端交互工作，可以进行提交操作、克隆操作等。不同客户端之间是相互独立的。

**b. 分布式版本控制工具**

**Git**、Mercurial、Bazaar、Darcs……



每个用户都有自己在本地的仓库，用来管理版本控制，每个客户端也同时是一个服务器。同时也可以用一个专门的服务器的仓库用来管理。

###### 2. Git简介

**2.1 Git概述**

Git是一个开源的分布式版本控制系统，可以有效、高速地处理从很小到非常大的项目版本管理。Git起初是 Linus Torvalds 为了帮助管理 Linux 内核开发而开发的一个开放源码的版本控制软件。

**2.2 Git历史简介**

Git创建历史解读：

时间 事件

1991年 Linus在1991年创建了开源的Linux，从此，[Linux系统](https://www.baidu.com/s?wd=Linux%E7%B3%BB%E7%BB%9F&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "_blank)不断发展， 已经成为最大的服务器系统软件了

2002年以前 世界各地的志愿者把源代码文件通过diff的方式发给Linus

（Linus坚定地反对CVS和SVN，这些集中式的版本控制系统不 但速度慢，而且必须联网才能使用。有一些商用的版本控制系统， 虽然比CVS、SVN好用，但那是付费的，和Linux的开源精神不 符）

2002年 Linux系统已经发展了十年了，代码库之大让Linus很难继续通过 手工方式管理了。Linus选择了一个商业的版本控制系统 BitKeeper，BitKeeper的东家BitMover公司出于人道主义精神， 授权Linux社区免费使用这个版本控制系统

2005年 开发Samba（Samba是在Linux和UNIX系统上实现SMB协议的 一个免费软件）的Andrew试图破解BitKeeper的协议，被BitMover 公司发现了，于是BitMover公司在愤怒中收回了Linux社区对于 BitKeeper的免费使用权

同是2005年 Linus在没有版本控制系统的窘境下，自己用C语言写了一个分 布式版本控制系统，这就是Git

值得一提的是，Linus仅仅使用两周时间就完成了Git的创建

Git创建完毕的一个月后，Linux系统的源码正式由Git管理

2008年 GitHub网站上线了，它为开源项目免费提供Git存储，无数开源 项目开始迁移至GitHub，包括jQuery，PHP，Ruby等等

如今，GitHub+Git的开发模式已经被全世界所认可，已然成为了全世界应用最广泛的分布式版本控制管理工具。

**2.2 Git官网及Logo**

官网地址：https://git-scm.com/

Logo：

**2.3 Git的优点**

（1）适合[分布式开发](https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F%E5%BC%80%E5%8F%91" \t "_blank)，强调个体。

（2）公共服务器压力和数据量都不会太大。

（3）速度快、灵活。

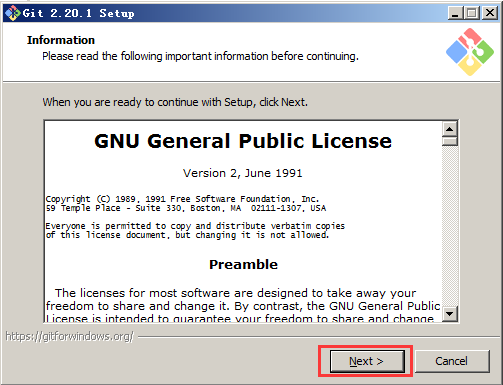
（4）相对容易的解决冲突。

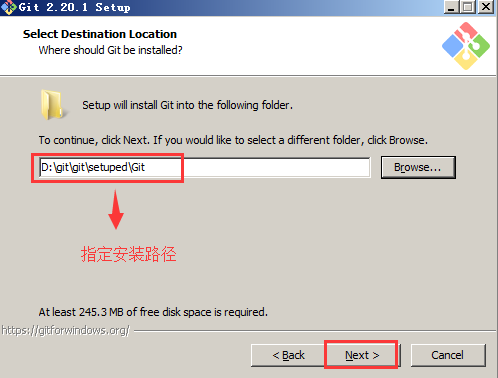
（5）大部分操作在本地完成，不需要联网。

###### 3. Git安装

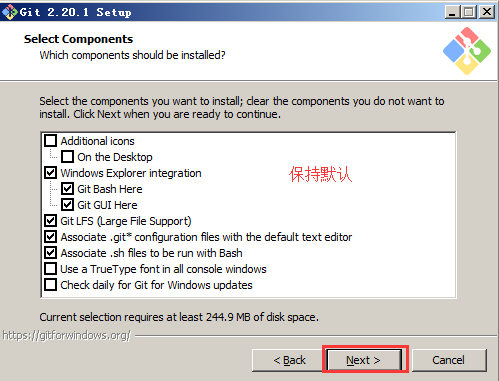
双击安装文件

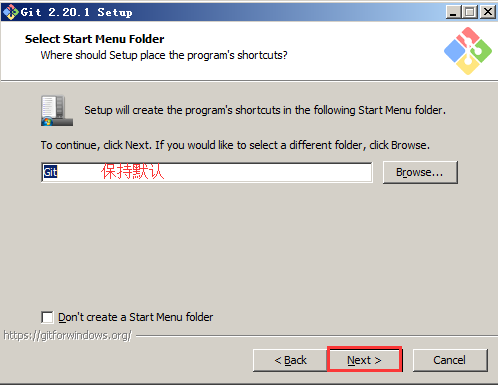


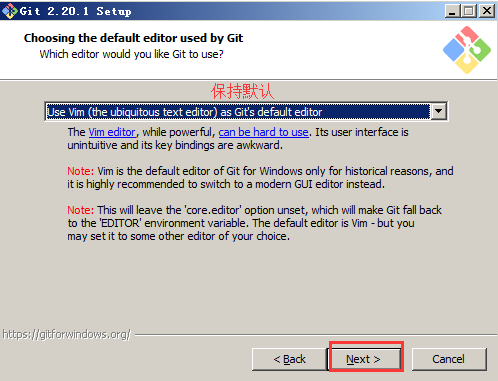


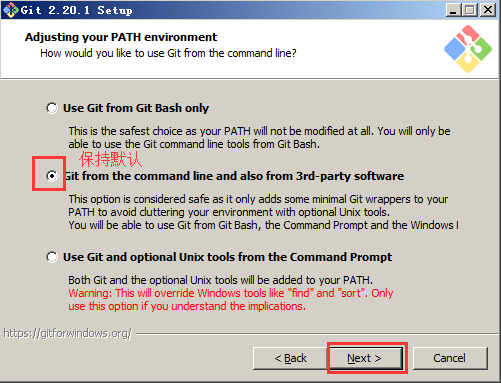


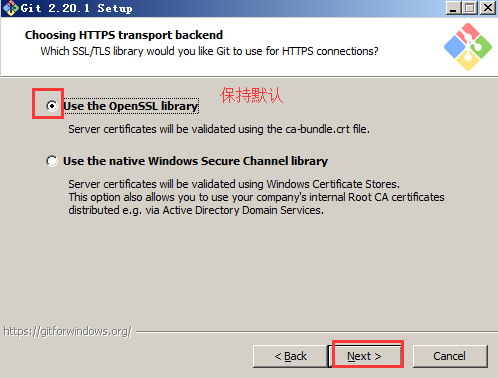
注意安装路径不能有中文和空格

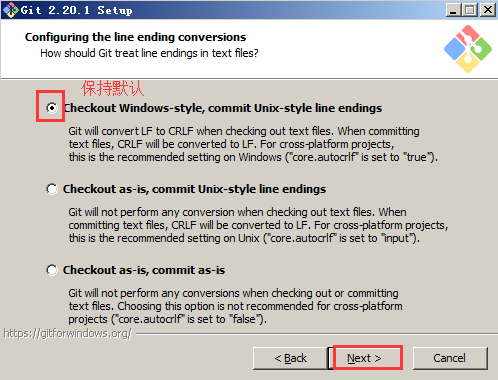


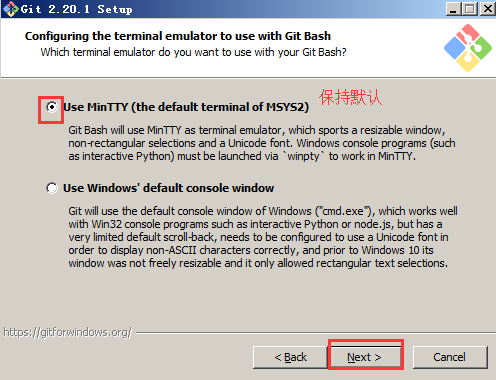


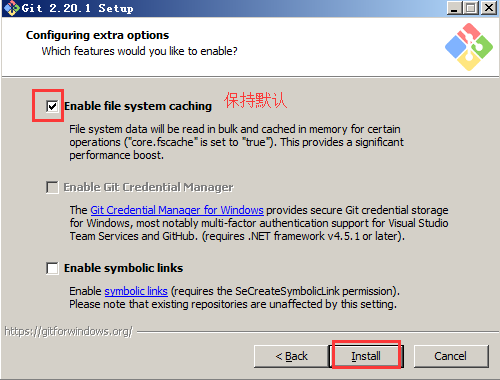


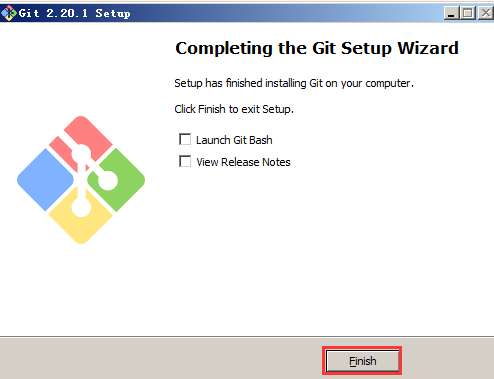




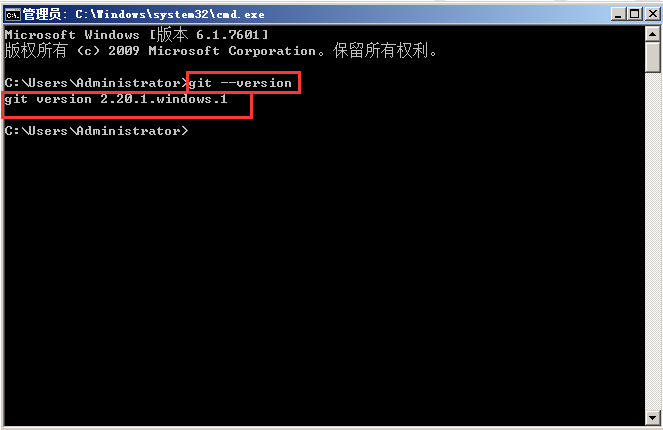








启动cmd命令行工具，以查看版本号的形式验证Git是否已经安装成功



###### 4. Git应用方式

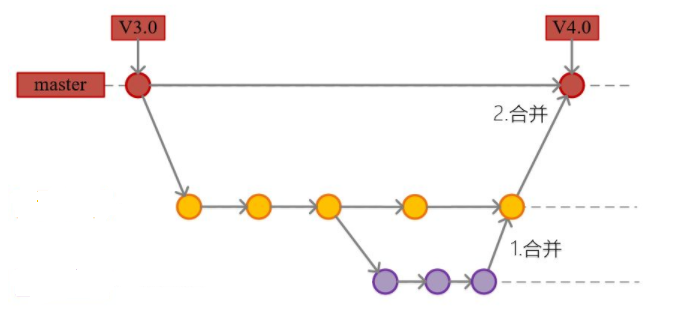
**3.1 Git版本控制方式**

**关于基本操作**



将工作目录的代码先提交到暂存区，然后再由暂存区提交到本地仓库

**关于分支操作**



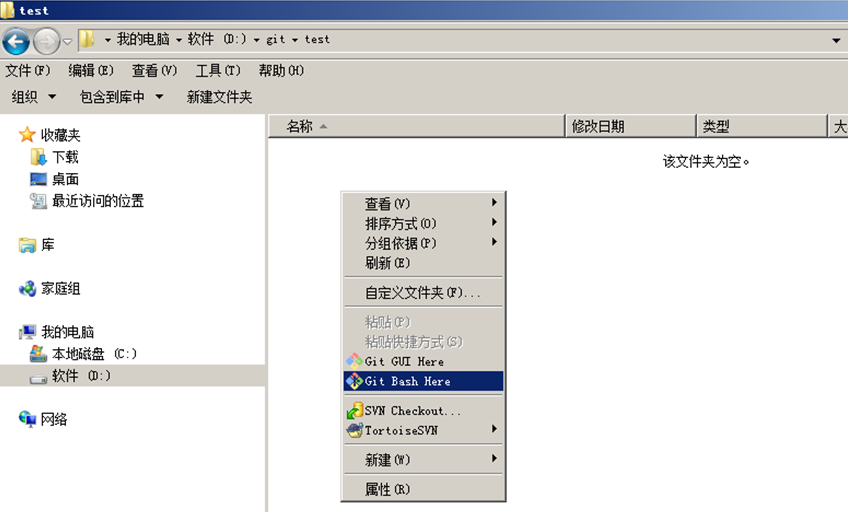
在版本控制过程中，使用多条线同时推进多个任务

**3.2 以命令行方式操作本地库**

首先找到一个任意的路径当做本地库目录

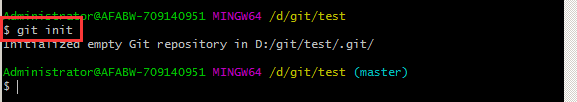
例如：目录D:\git\test

点击右键 --> Git Bash Here

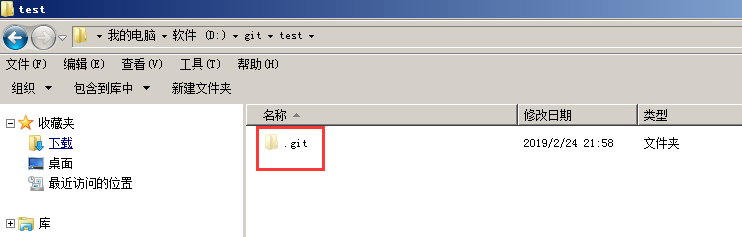


初始化本地库

命令：git init



执行完该命令后，在当前目录会出现 文件夹 .git



.git目录的出现表示本地库初始化成功（表示这个.git目录就是我们的本地库了）

注意：.git目录存放的是和本地库相关的文件，不要修改或者删除

Git目录说明

工作目录：任意目录下，我们开发代码的目录

暂存区域：.get目录下，作用：有个后悔（返回撤销）的余地

本地仓库：.get目录下，Git存储项目的仓库

**3.3 设置签名**

设置签名的作用：区分不同开发人员的身份

注意：为Git设置签名与远程库（代码托管中心）的账号密码没有任何关系

设置签名命令：

本地库级别设置签名方式：

git config user.name zs

git config user.email zs@bjpowernode.com

信息保存位置：./.git/config 文件

系统用户级别设置签名方式：

git config --global user.name zs

git config --global user.email zs@bjpowernode.com

~/.gitconfig 文件

优先级按照就近原则：项目级别优先于系统用户级别，二者都有时采用项目级别

的签名

###### 4. GitHub应用方式

**4.1 GitHub概述**

GitHub是一个面向[开源](https://baike.baidu.com/item/%E5%BC%80%E6%BA%90/20720669" \t "_blank)及私有[软件](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6/12053)项目的托管平台，因为只支持Git 作为唯一的版本库格式进行托管，故名GitHub。

2018年6月4日，微软宣布，通过75亿美元的股票交易收购代码托管平台GitHub。

2018年10月26日，微软以75亿美元收购GitHub交易已完成。同年10月29日微软开发者服务副总裁奈特·弗里德曼(Nat Friedman)将成为GitHub的新一任CEO。

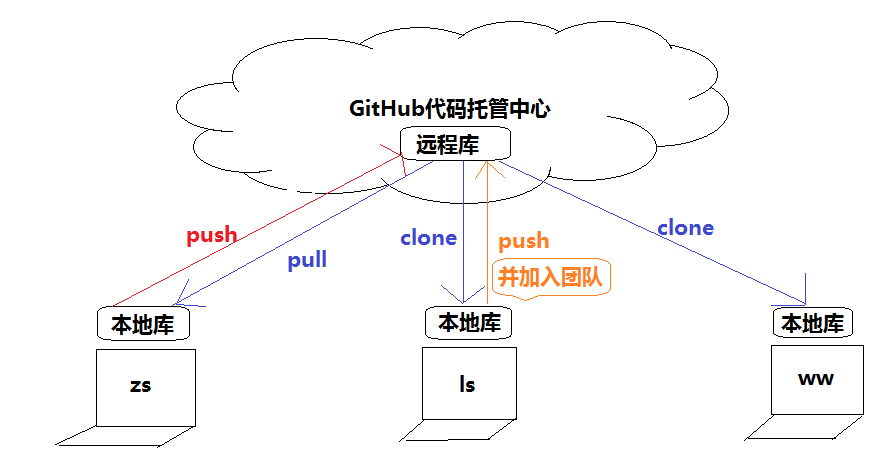
**4.2 GitHub官网**：

https://github.com/

官网首页就是注册页面

**4.3 GitHub应用流程**

团队协作开发方式如下图所示：

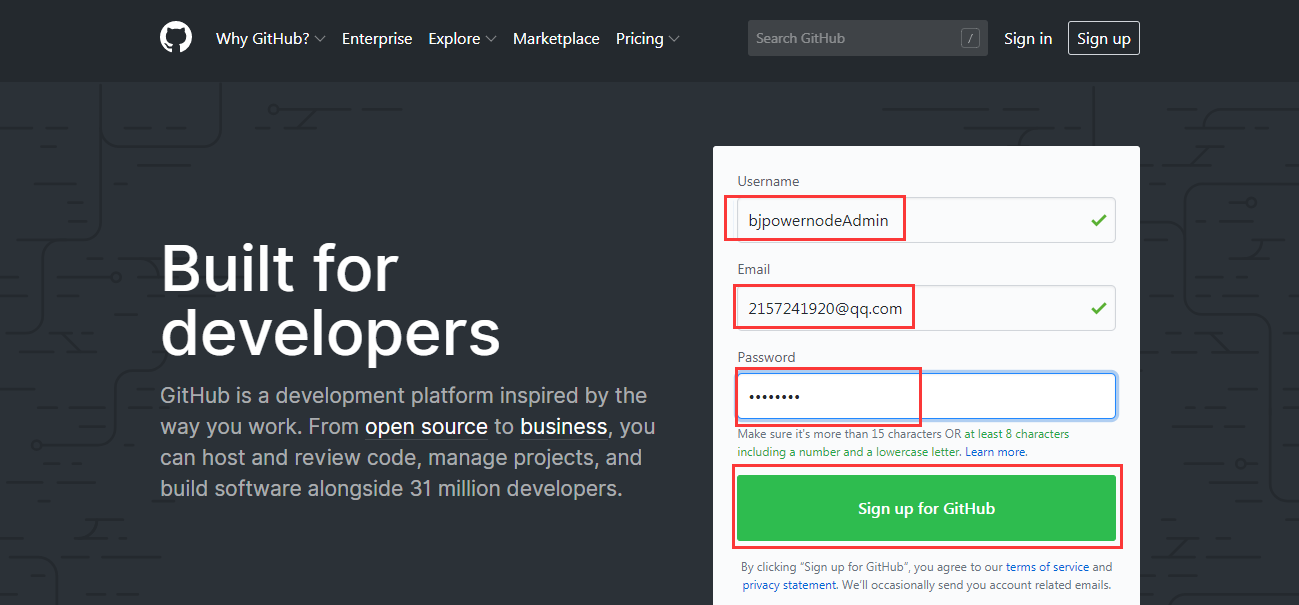


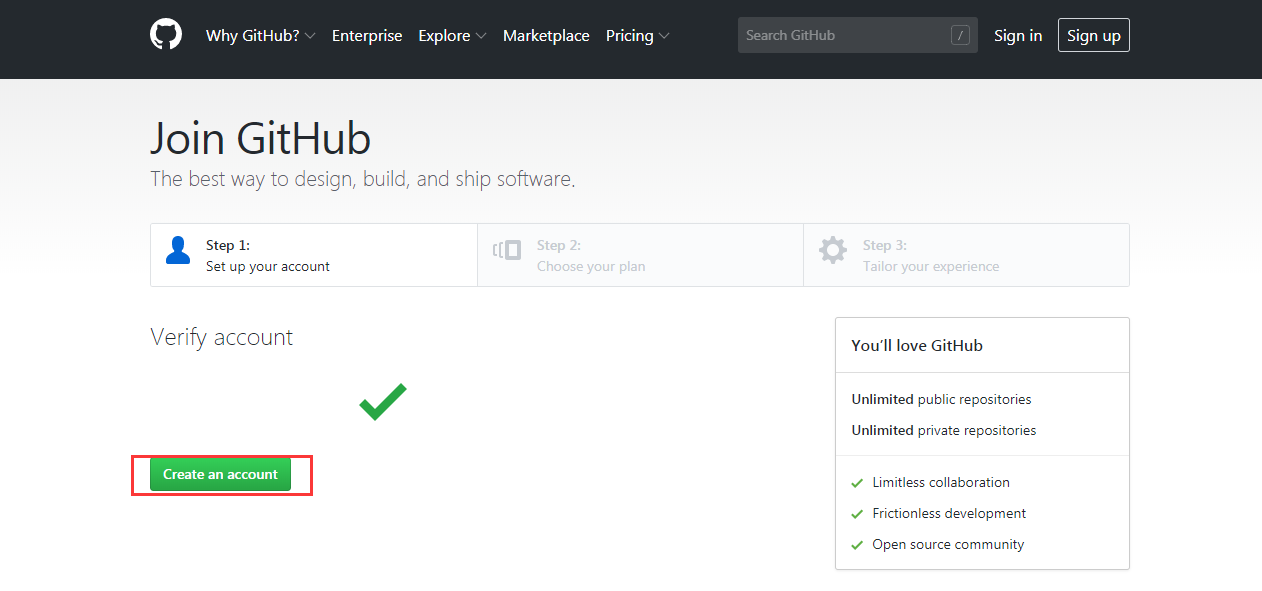
**4.4 GitHub注册流程**

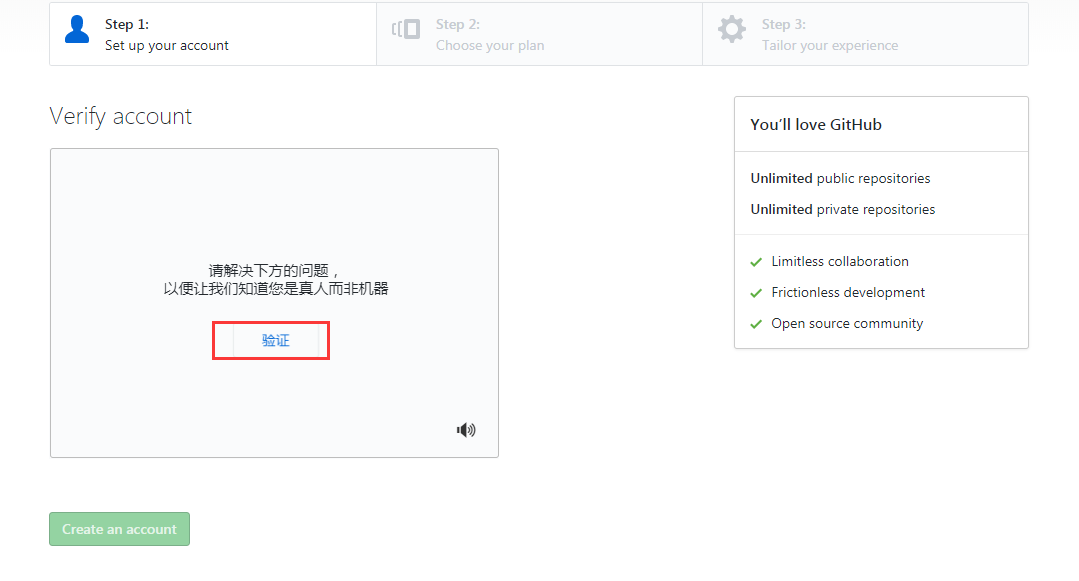
打开官网

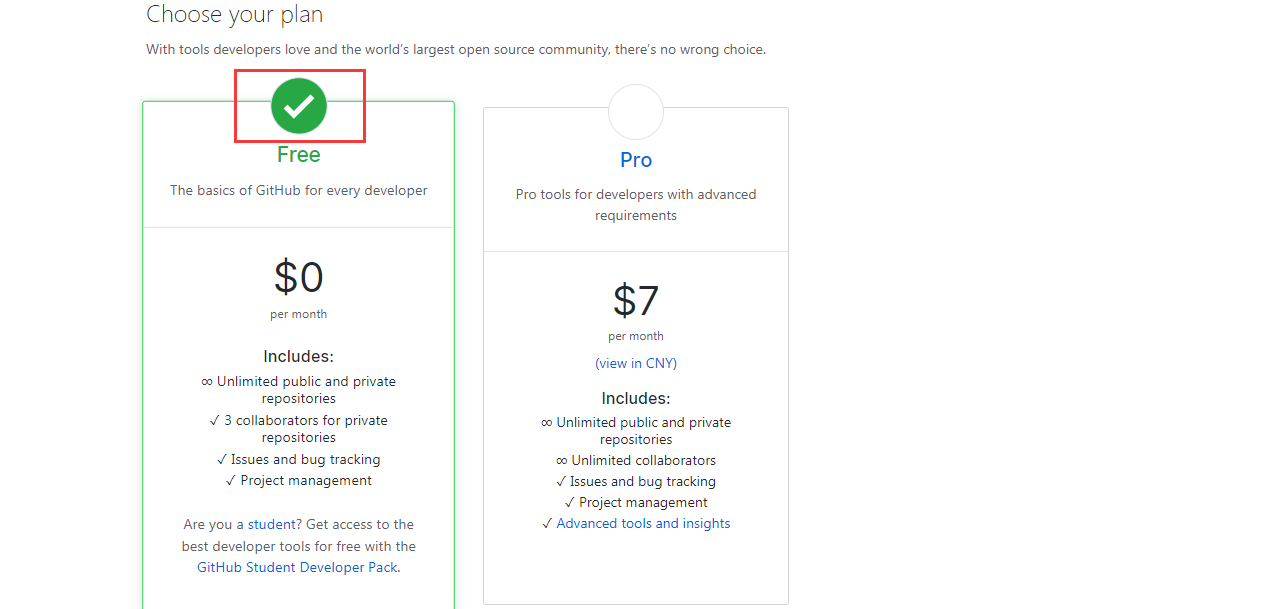
填写注册信息

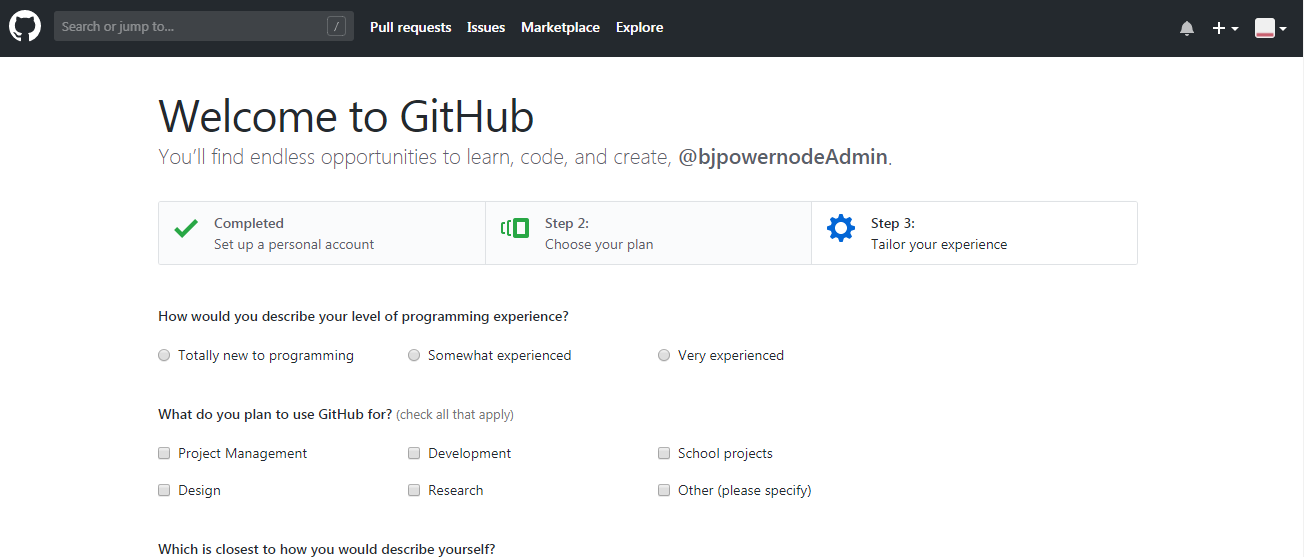
**注意：此处必须使用真实邮箱**







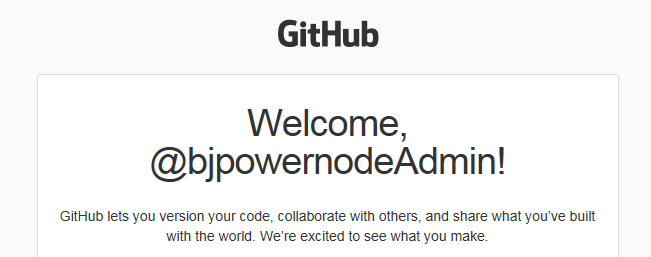




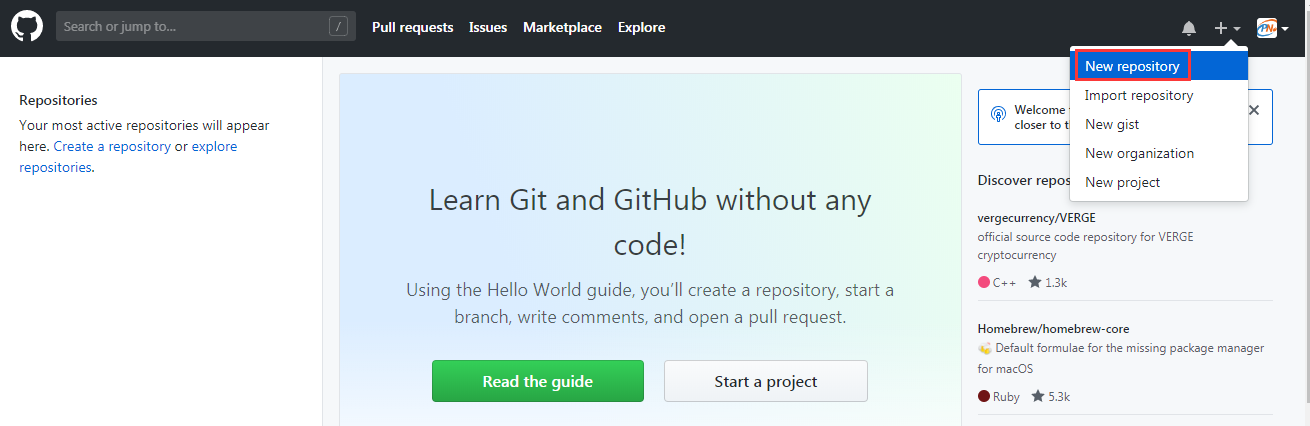
注意：注册完毕后，会收到GitHub为我们发送的邮件

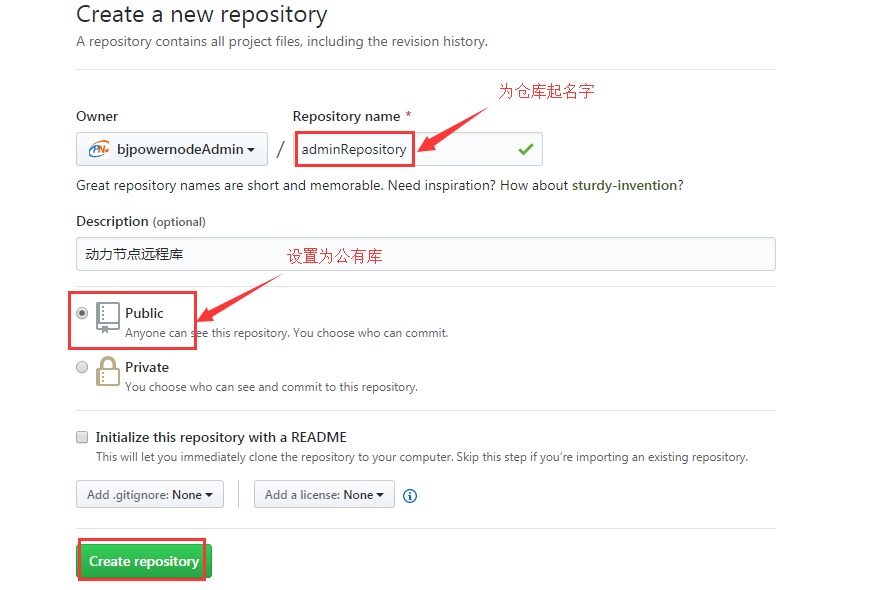
我们打开邮件，点击Verify email address

收到欢迎邮件表示完成注册

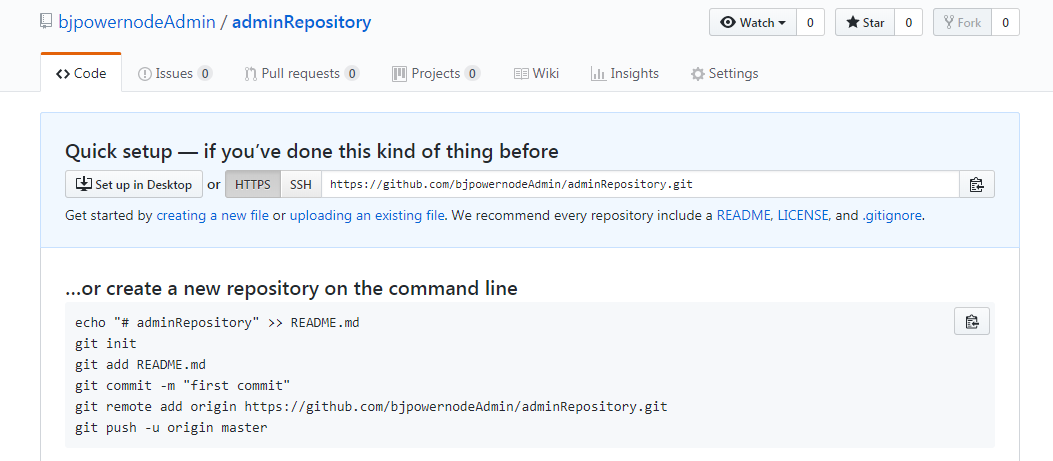


**4.5 创建远程库**

****

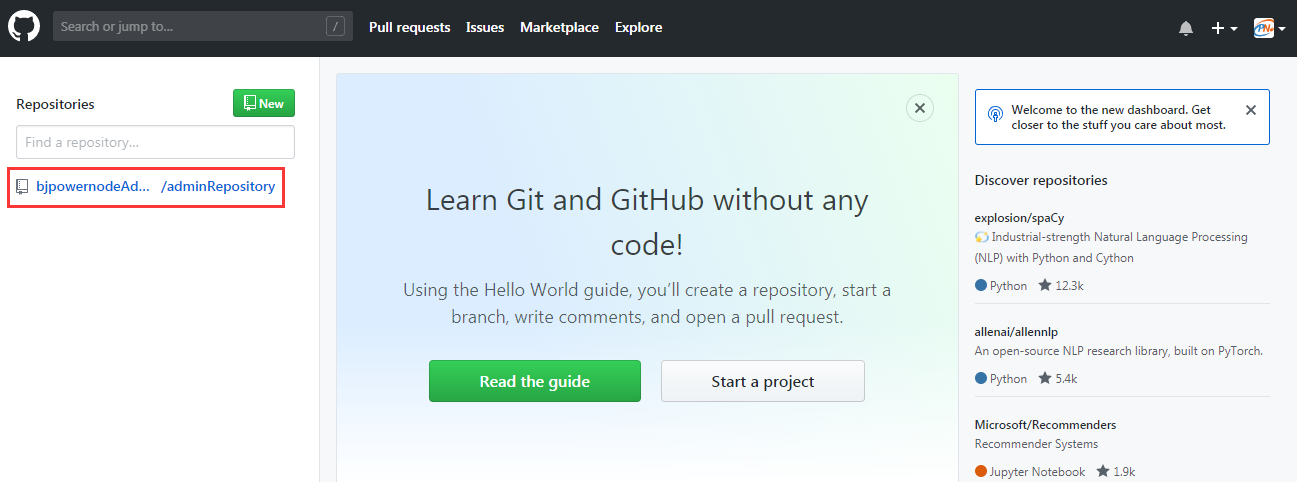


看到以下远程库信息页面证明远程库已经创建成功



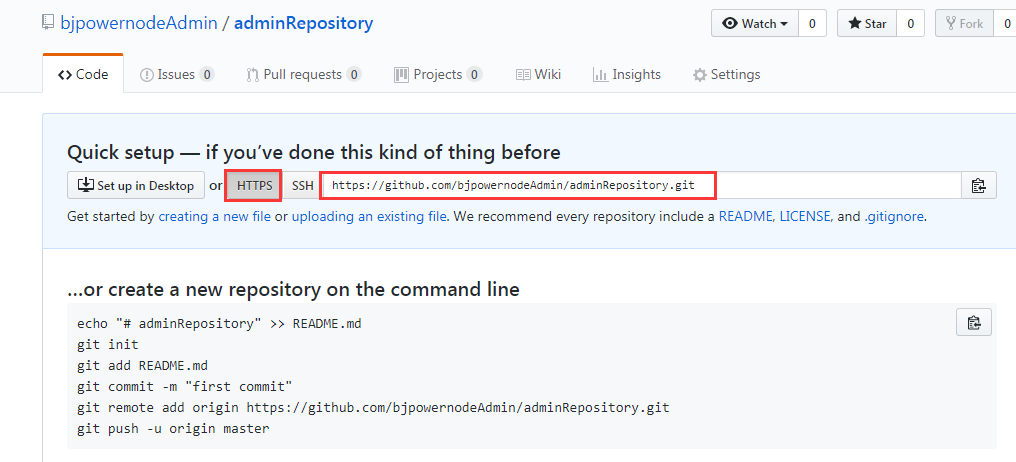
如果是刚登录进来，欢迎页面的左侧也会看到远程库的菜单项

点击左侧菜单中的远程库，也会进入到该远程库信息页面

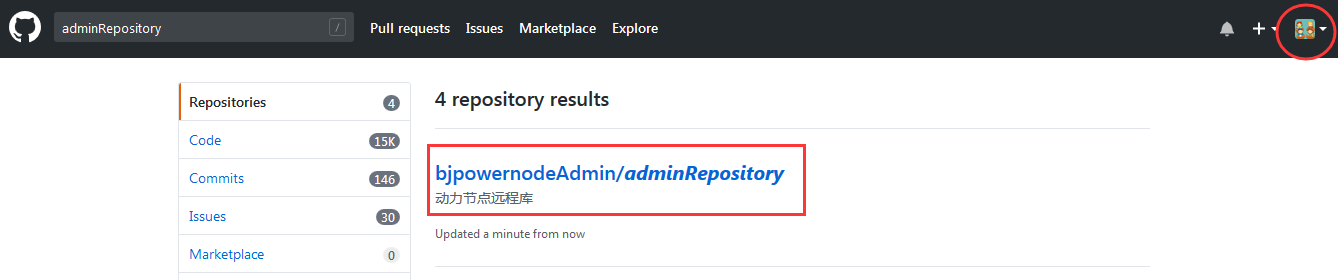


观察如下详细信息页面

点击HTTP，会得到访问远程库的地址

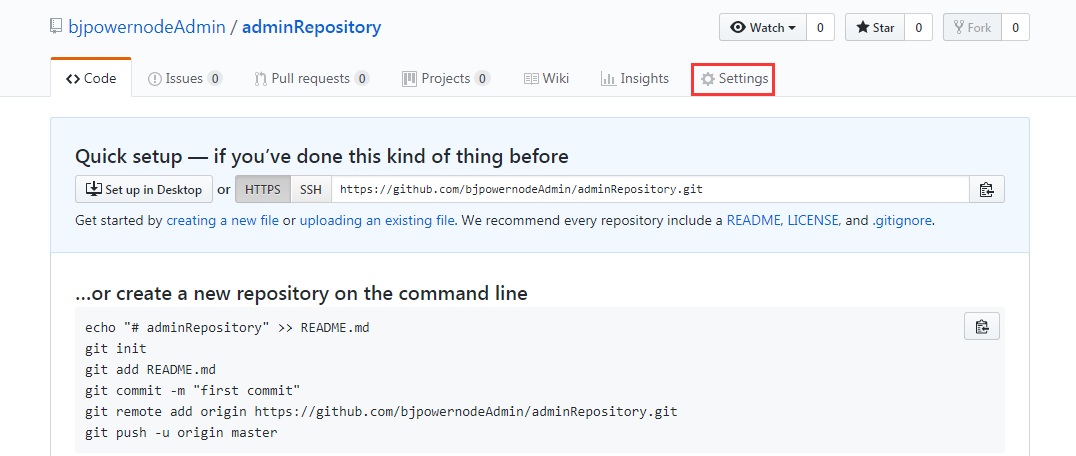


使用另一个账号登录，可以搜索到别人创建的远程库

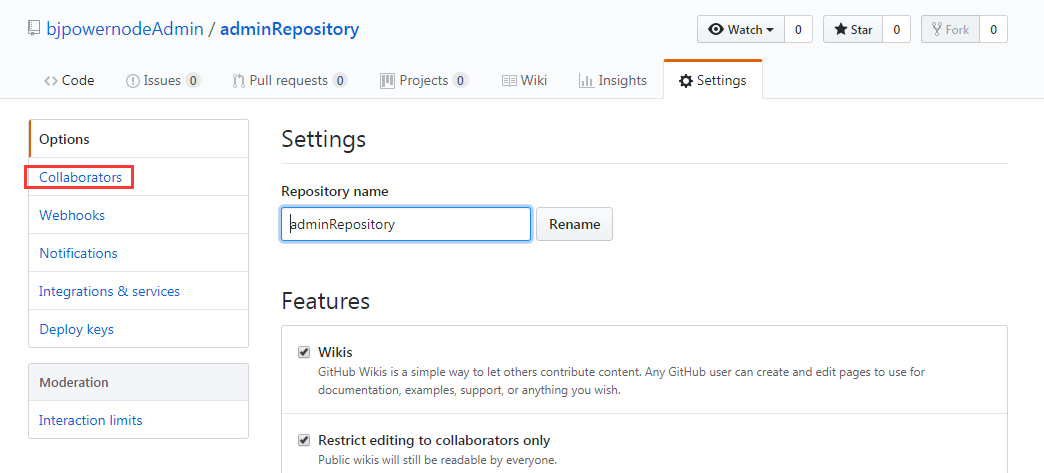


**4.6 邀请其他GitHub用户加入开发团队**

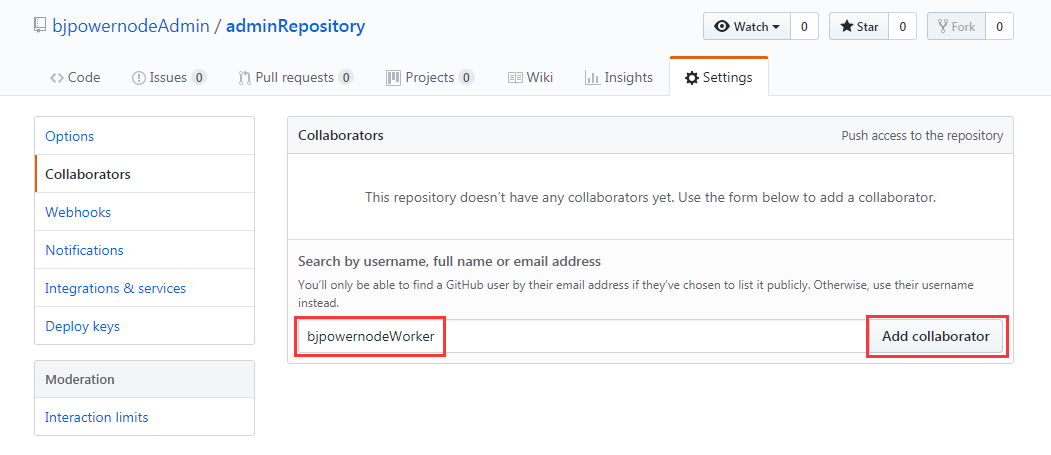
打开远程库信息页面，点击settings

****

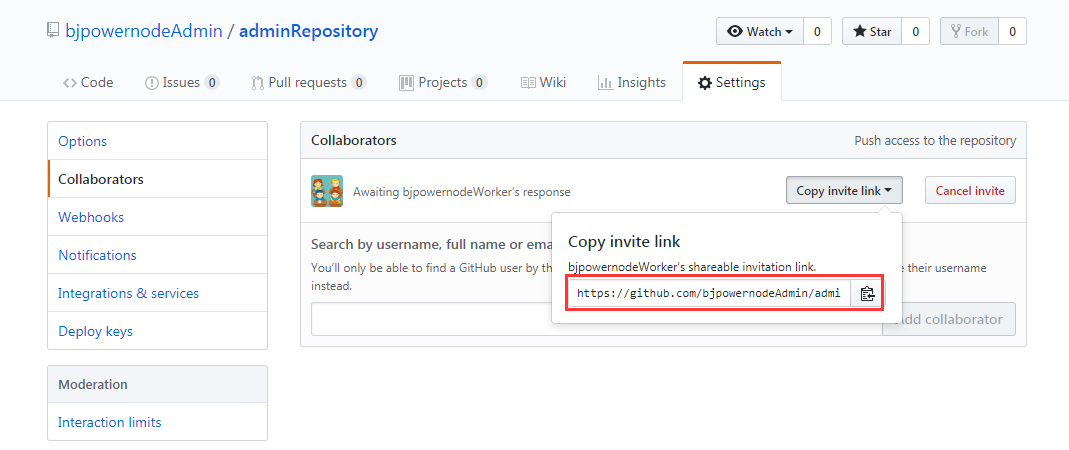
点击Collaborators



输入被邀请人账号，点击Add collaborator



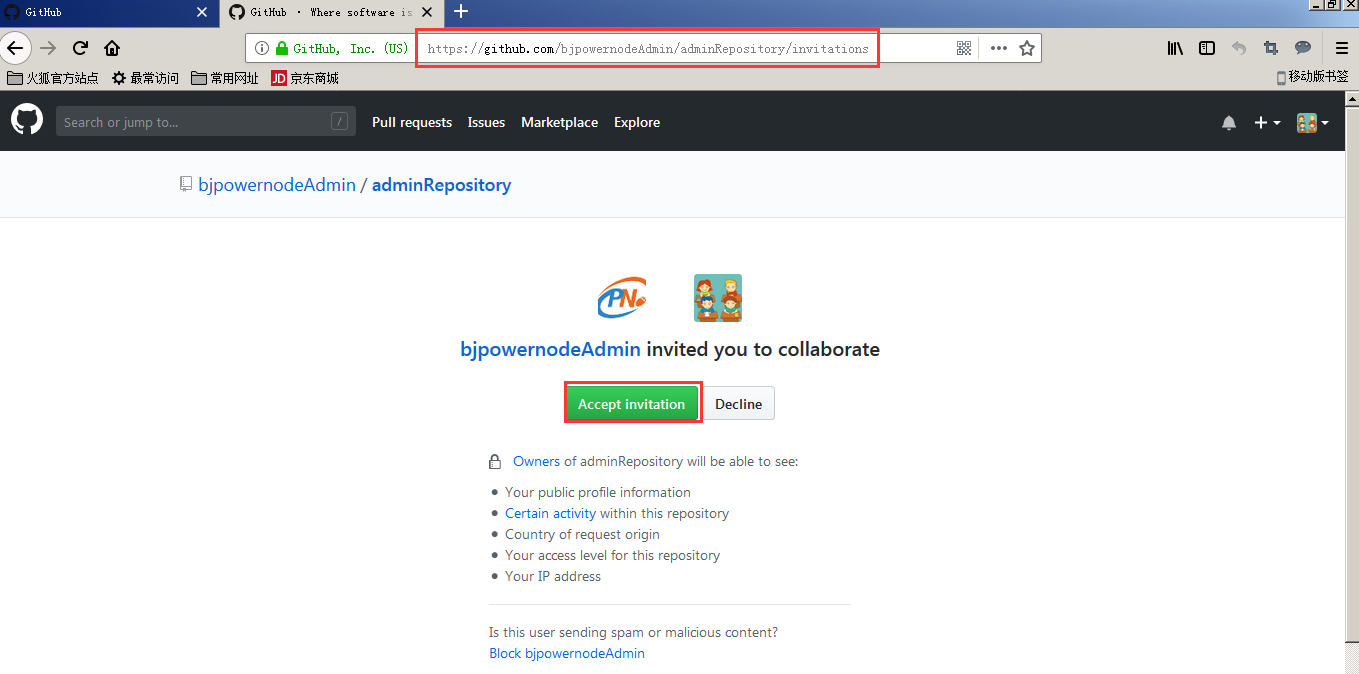
复制邀请链接



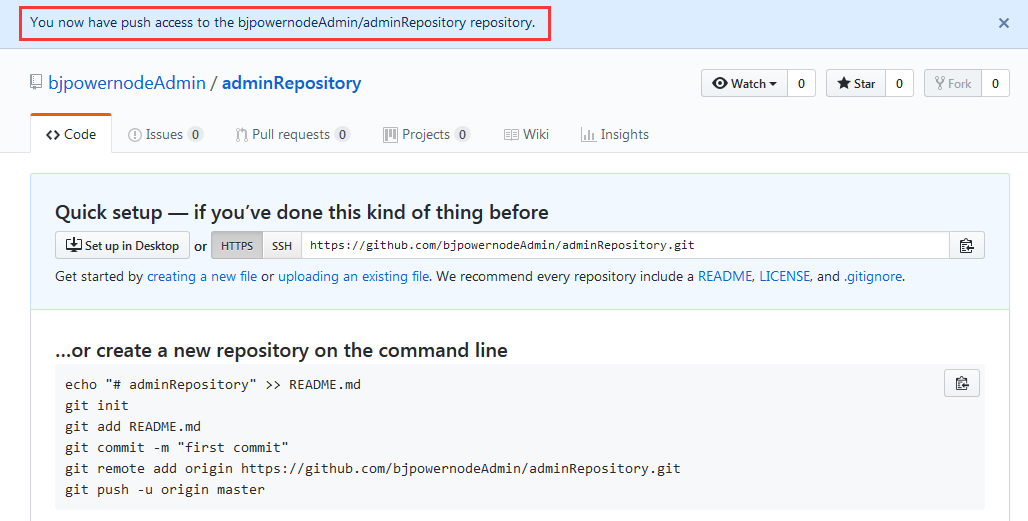
登录被邀请账号

打开浏览器新窗口，在地址栏访问刚刚复制的链接

点击Accept invitaion 接受邀请



变为团队成员后可以推送代码到远程库了

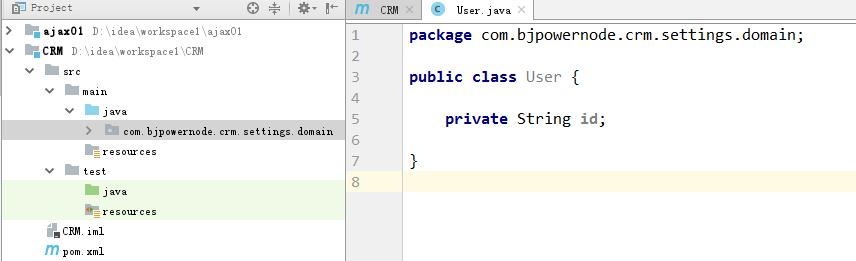


###### 5. IDEA对于Git的支持

**5.1 IDEA对GitHub和Git的基本配置**

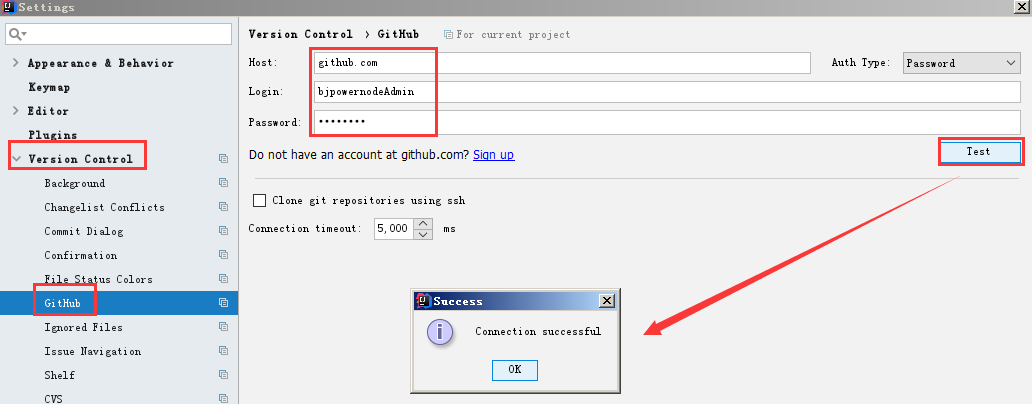
案例演示

首先建立一个演示项目（Maven项目即可），然后建立一个User类，里面写上初始测试内容



打开settings --> Version Control --> GitHub

填写GitHub网址，账号，密码，然后点击Test测试



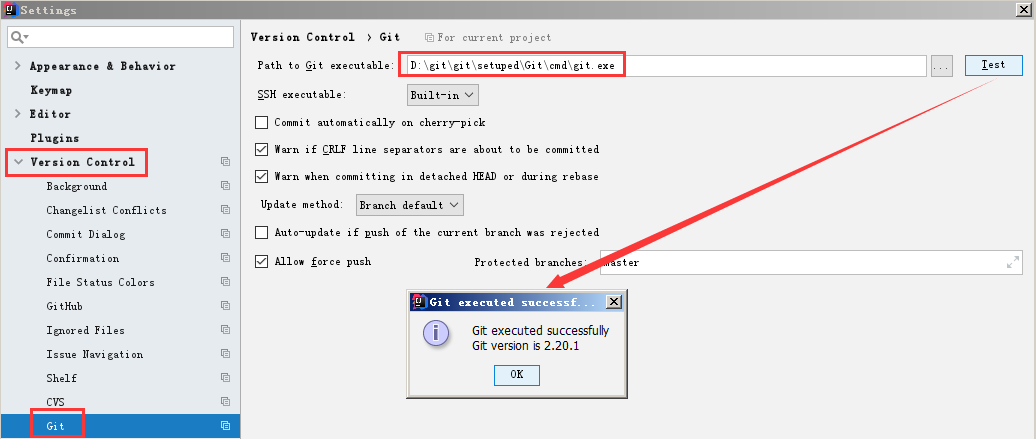
上述测试成功后，配置Git

打开settings --> Version Control --> Git

第一栏文本框会自动识别到本地Git安装路径下的git.exe

点击测试

**其他项维持默认即可**

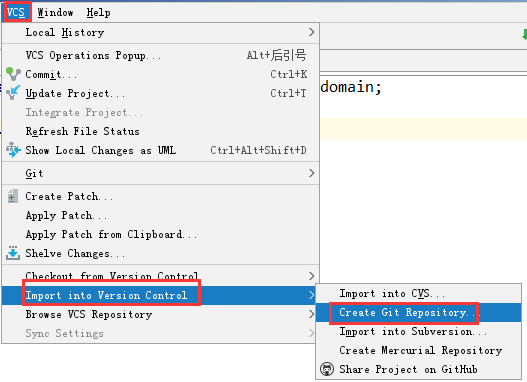


以上就是IDEA对于GitHub和Git的基本配置

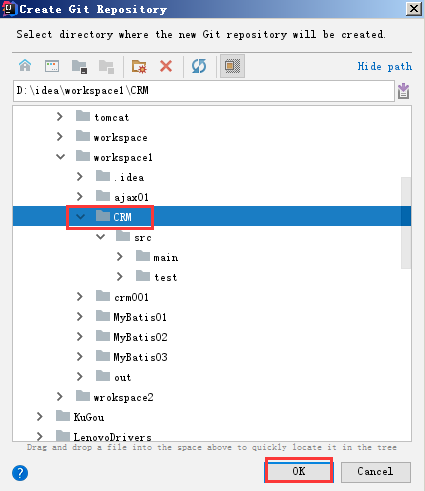
以上配置完毕后，我们来操作项目

**5.2 将IDEA项目push（推送）到GitHub**

创建本地库

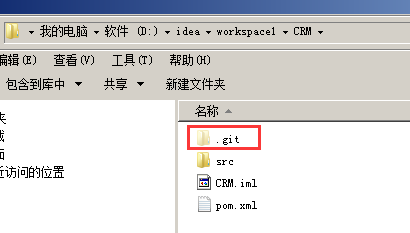


选中当前项目作为本地库



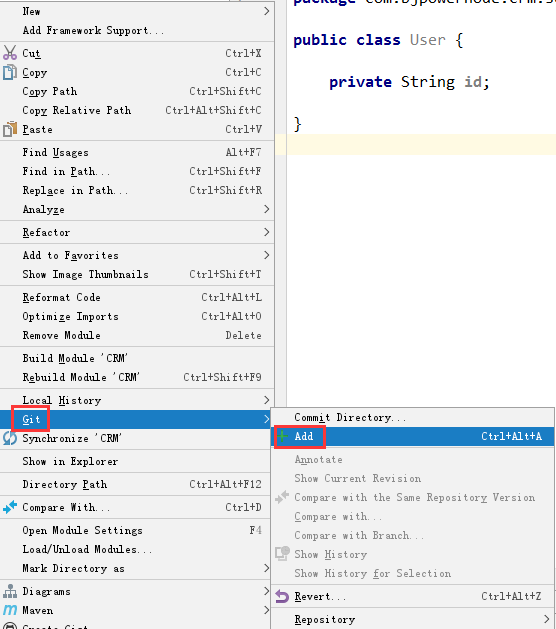
以上操作之后，会在当前项目的本地看到.git文件夹

说明当前项目文件夹作为本地库存在



将项目提交到暂存区

项目右键 --> Git --> Add

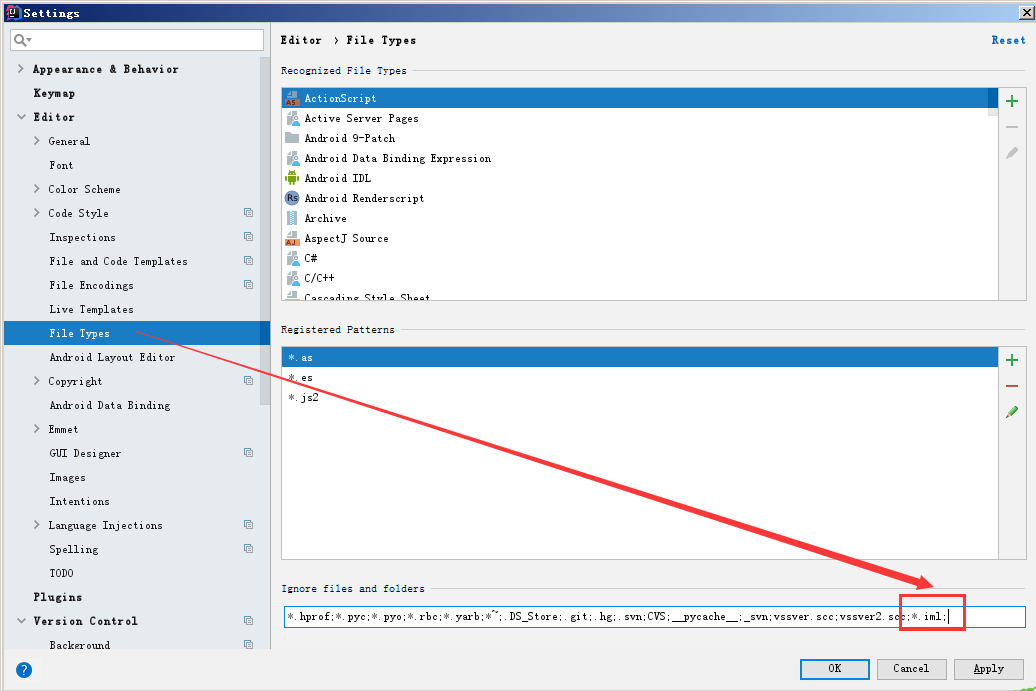


将项目提交到本地库

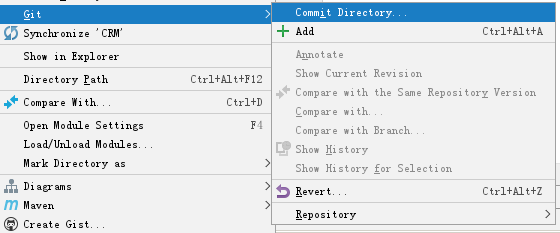
首先我们在提交前，需要忽略掉.iml文件

settings --> Editor --> File Types

添加\*.iml

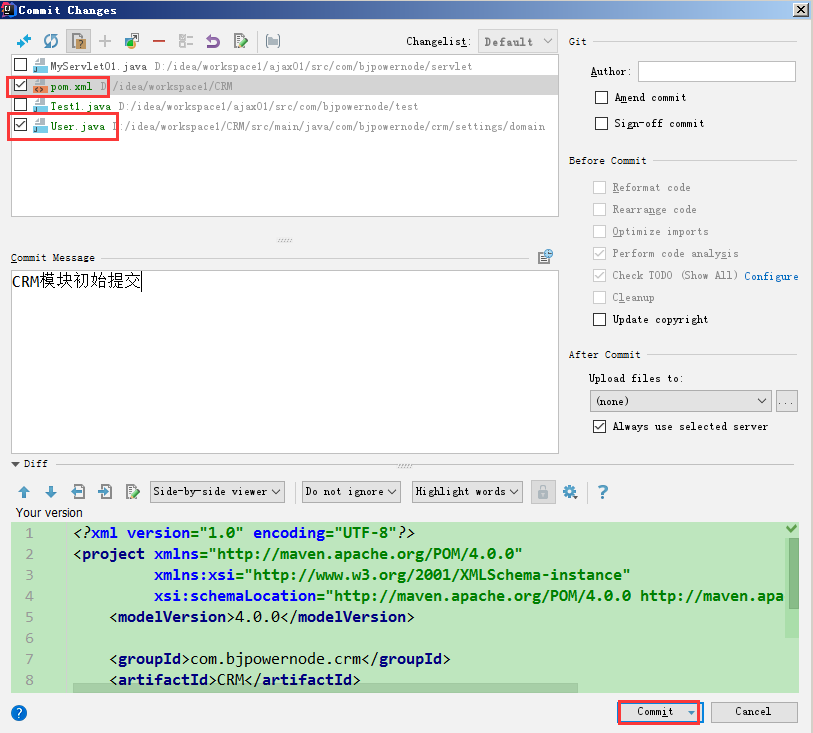


项目右键

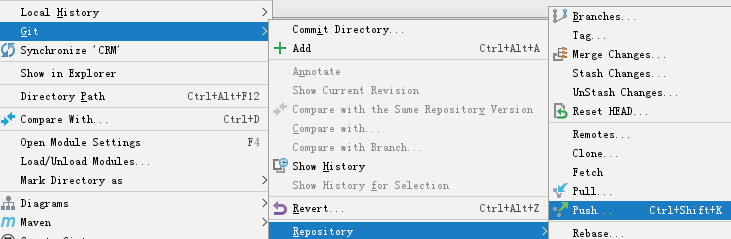


选中刚刚模块中写好的未提交过的文件

点击Commit提交



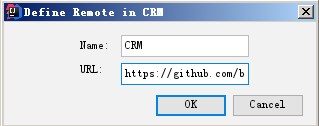
将本地库中的文件上传到远程库



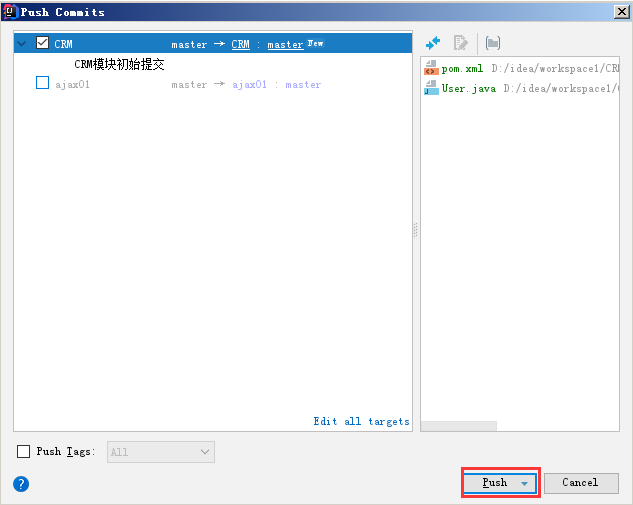
第一次提交，需要先写入提交的GitHub库的名字，以及上传地址



在GitHub上创建一个新仓库CRM



填写完以上信息后，推送项目到GitHub远程库

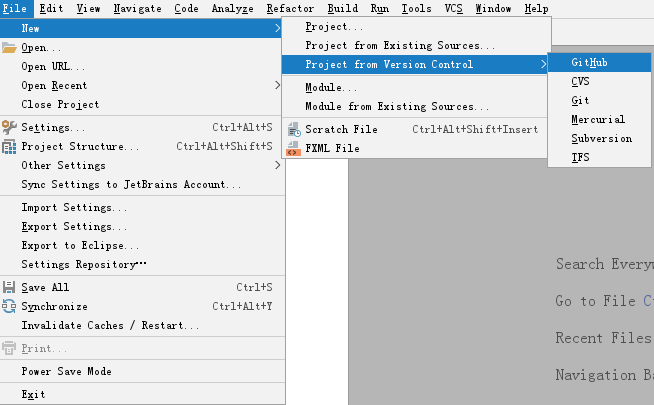


**5.3 从GitHub远程库clone项目到IDEA**

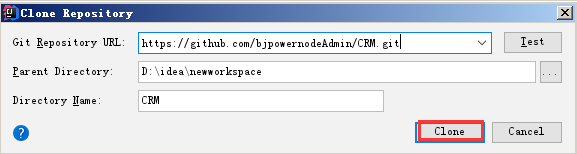
新建一个空的Project，在另一个窗口打开（同时保留原有的项目窗口）



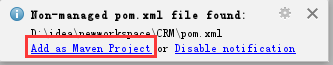
开始clone项目



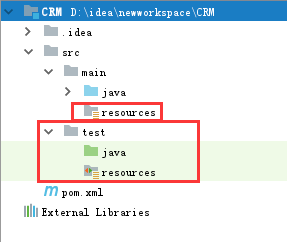
填写基本信息



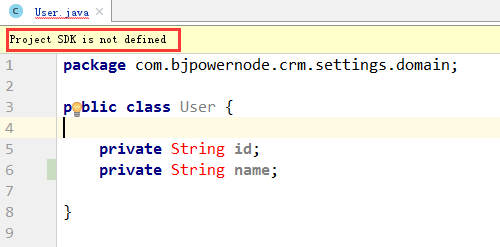
作为一个maven项目导入



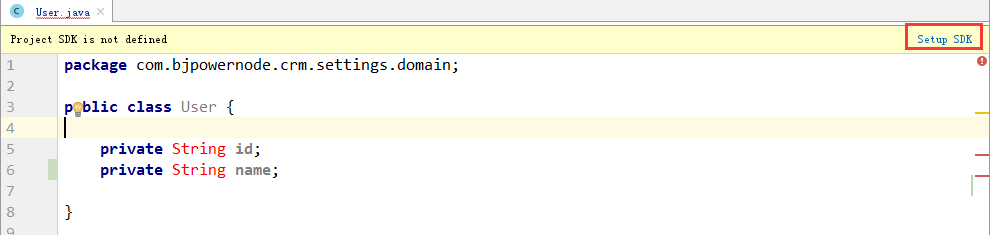
将Maven项目目录结构写完整（注意文件夹颜色要赋予正确的颜色）

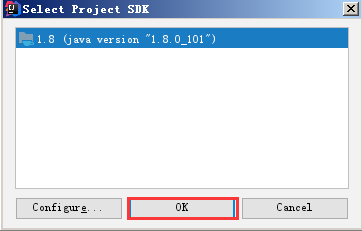


当编写java代码时，会提示没有编译环境



点击右侧的setup sdk





将加入了新属性的User类推送到远程库

**注意：推送前，不要忘记先Add！！！**

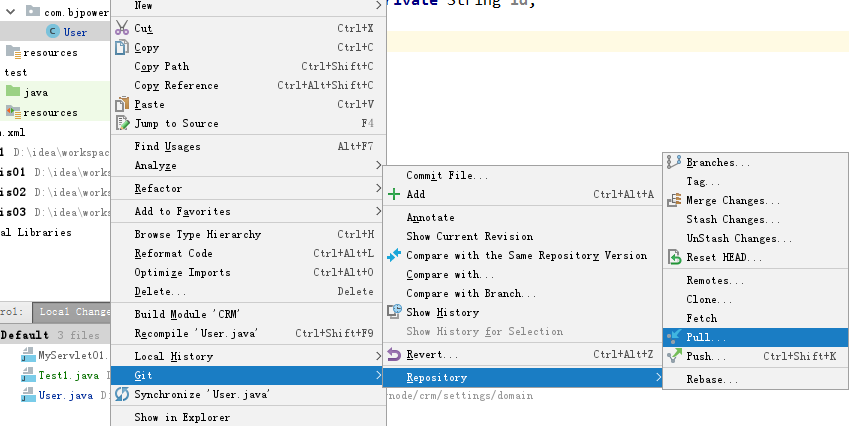
**Add之后Commit and push**

切换到第一个项目

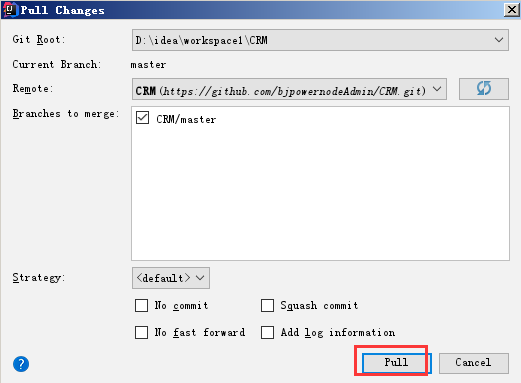
暂时只有id属性（第二个项目有id属性和name属性）

所以第一个项目需要将name属性拉取，将更新为最新版

右键User类

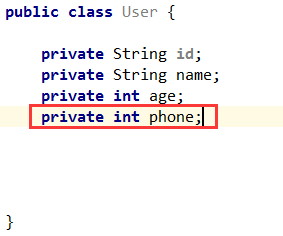


如下图所示，所有默认，点击Pull

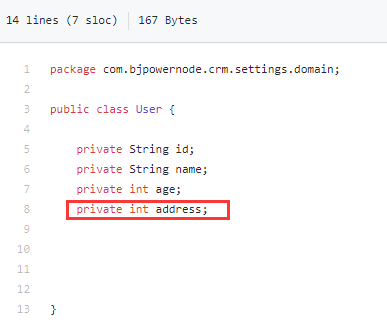


如果本地更改过文件（没有及时上传），拉取时会产生冲突

例如本地要上传的新属性是phone



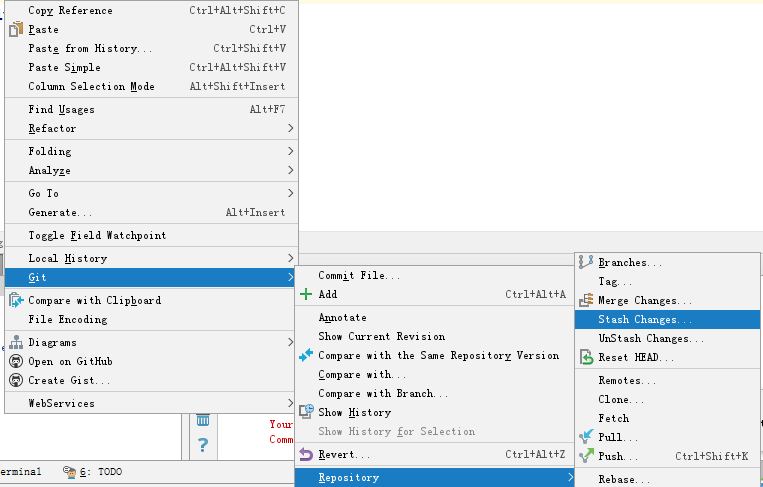
但是远程库，最新的属性是address

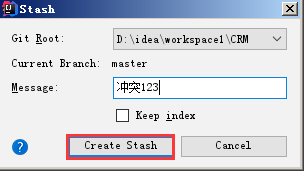


这样会产生冲突，pull会失败

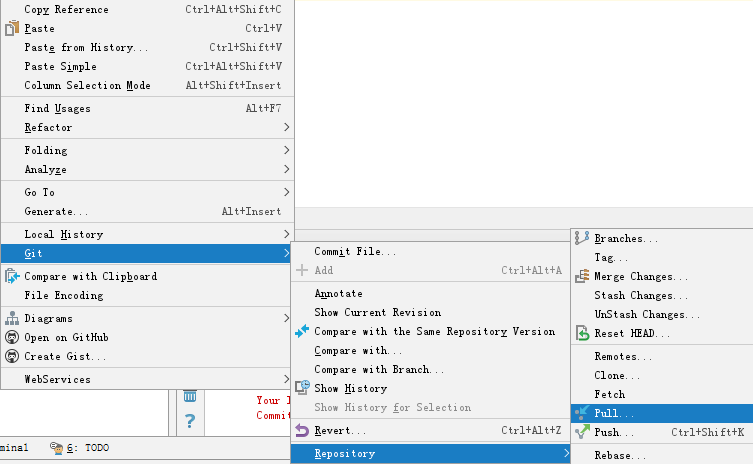


需要先将自己本地库最新的版本先以Stash的方式保存

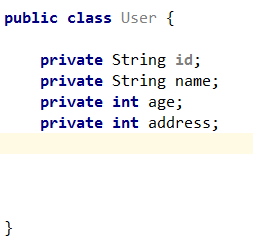




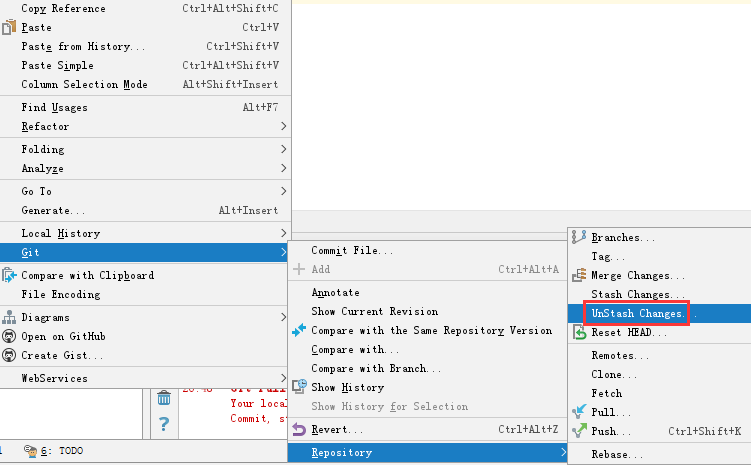
然后再拉取

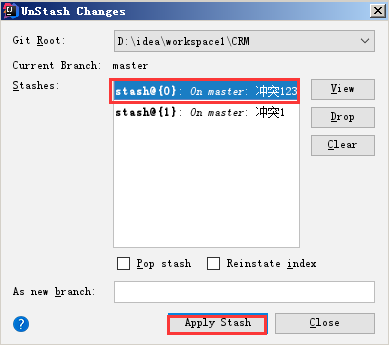


就会拉取成功了

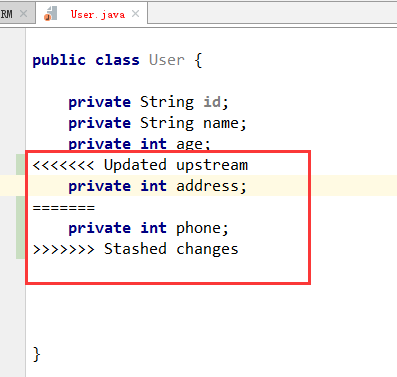


最后将本地最新版（属性phone）合并进去





我们会看到即保留了刚刚pull远程库的最新版本的代码（address），又保留了自己本地库（phone）的代码



调整冲突后（同时保留phone和address），push到远程库

对于远程库

GitHub

Gitee

GitLab

Gitblit