Git

**1.Git介绍**

Git是一个开源的分布式版本控制系统，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，Git可以使用本地创建仓库与网络仓库，解决了集中管理型版本控制软件存在的一些问题（CVS、VSS、SVN）。

**2.Git安装**

双击 Git-2.9.2-64-bit.exe 文件，一直下一步即可。

**3.Git使用**

安装完成后，在任意的文件目录下，右键都可以开打Git的命令行窗口。



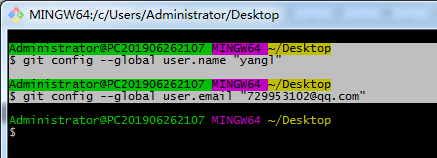
Git是分布式版本控制系统，所以需要填写用户名和邮箱作为一个标识，--global 表示全局属性，所有的git项目都会共用属性。

配置用户名

$ git config --global user.name "yangl7299"

配置邮箱

$ git config --global user.email "729953102@qq.com"



此时在C:\Users\Administrator\.gitconfig文件可以查看到我们的配置信息。

**4.理解工作目录，暂存区，本地仓库**

存储顺序：工作目录 -->> 暂存区 -->> 本地仓库

**工作区(Working Directory)：**电脑上的本地硬盘目录，平时存放项目代码的地方。

**暂存区(stage)**：用于临时存放你的改动，事实上它只是一个文件，保存即将提交到文件列表信息，一般存放在"git目录"下的index文件（.git/index）中，所以我们把暂存区有时也叫作索引（index）。

**本地仓库(Repository)：**工作区有个隐藏目录.git，它就是Git的本地版本库。

git的工作流程：

１、在工作目录中添加、修改文件；

２、将需要进行版本管理的文件放入暂存区域；

３、将暂存区域的文件提交到git仓库。

使用步骤：（首先设置电脑可以查看隐藏文件）

1.在任意位置创建空文件夹，作为项目目录，例如：在G盘右键创建git\_repository文件夹

2.在项目文件夹内右键打开git bash窗口，输入命令: git init

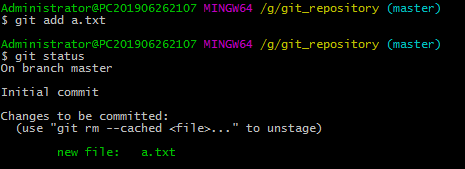


此时git\_repository目录下会生成一个隐藏的 .git 目录。

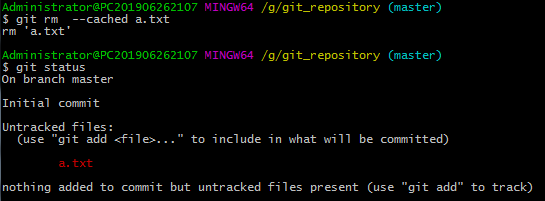
3.新建一个文件，例如：a.txt

4.输入命令：git add 文件名，此时是将文件添加到暂存区当中

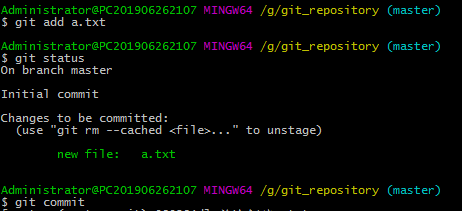
5.输入命令：git status，查看暂存区状态

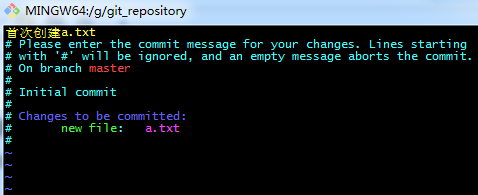


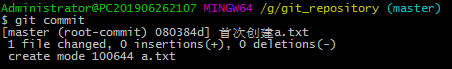
6.如需删除，可以输入命令：git rm --cached 文件名，此时是从暂存区中删除了，工作目录中还会存在该文件，删除之后再次查看，该文件名颜色已经发生变化



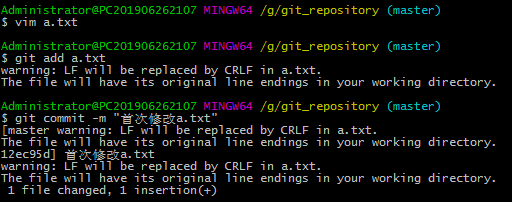
7.如需提交文件到本地库，输入命令：git commit ，提交完成后，会提示在文件中写一些备注（可写可不写），如需写可以使用i/a/o编辑文本（和Linux命令一样），然后保存并退出



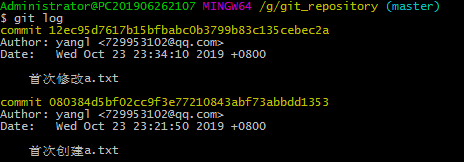




另一种方式：使用git commit –m “注释内容”, 直接带注释提交



8.输入命令：git log 查看所有历史记录，输入命令：git log 文件名，查看该文件的历史记录



9.输入命令：git log --pretty=oneline 查看所有文件历史记录简易信息，输入命令：git log --pretty=oneline 文件名 查看该文件历史记录简易信息



10. 输入命令：git reset --hard HEAD^ ，回退到上一次提交

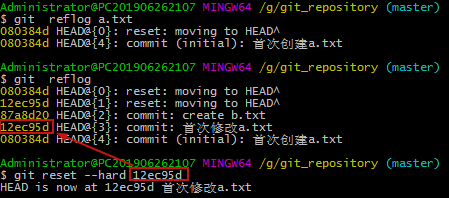
输入命令：git reset --hard HEAD~n，回退n次操作

输入命令：git reset 文件名，撤销文件缓存区的状态



11.输入命令：git reflog[文件名]，查看历史记录的版本号

输入命令： git reset --hard 版本号



12.手动删除工作目录中的文件，然后输入命令：git checkout 文件名，可以恢复工作目录中已经删除的文件



13.手动删除工作目录中的文件，然后输入命令：git add 文件名 （此时该命令表示提交当前删除文件的操作）

然后输入命令：git commit -m "删除文件" （此时才是真正删除了该文件，但是此次删除只是这一次操作的版本号没有了，其他的都可以恢复）



恢复文件某个版本

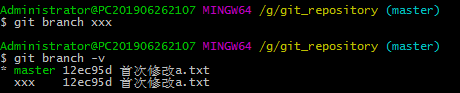


**5.分支相关操作**

创建分支

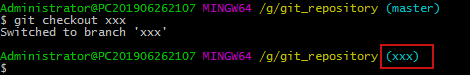
输入命令：git branch <分支名>

输入命令：git branch –v，查看分支

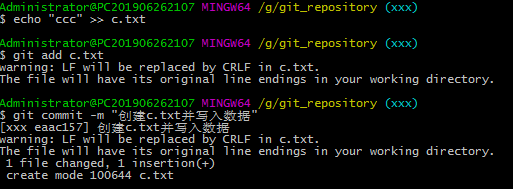


切换分支

输入命令：git checkout <分支名>



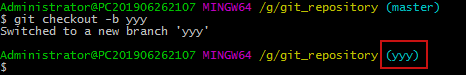
切换到分支后，在分支下的操作，master（主线）不会有任何变化



此时在工作目录中可以查看到c.txt文件，但是当切换到master时，再去查看工作目录已经没有c.txt文件了



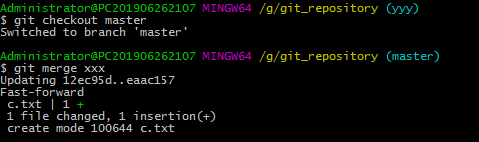
输入命令：git checkout –b <分支名>，将创建分支，切换分支一起完成



合并分支

输入命令：git checkout master，切换到主干

输入命令：git merge <分支名>，合并分支



此时在工作目录中又可以查看到c.txt文件了

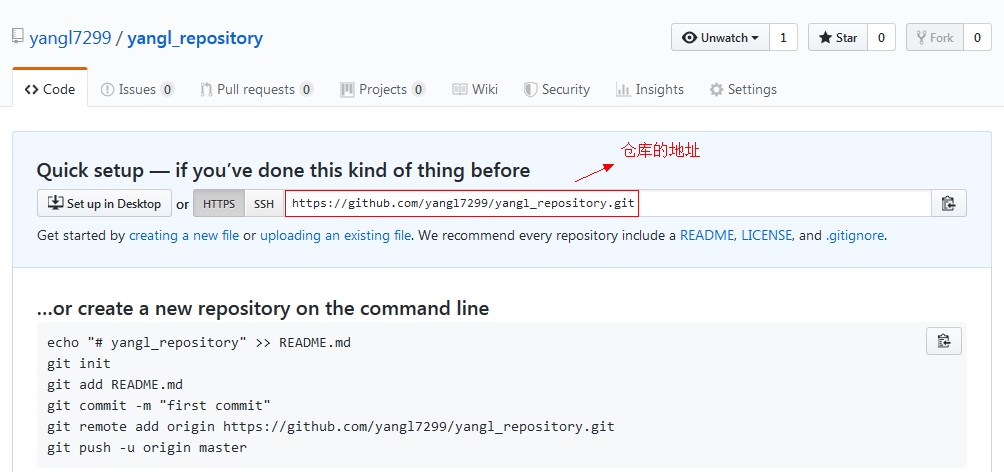
冲突问题：当主线和分支合并之后，分支和主线都会有相同的文件，此时在主线中修改该文件内容并提交，在分支中也修改该文件内容并提交，再次合并时就会出现冲突，此时文件中会显示冲突内容，我们需要手动解决再合并了。

**6.GitHub**

GitHub是一个基于Git的面向[开源](https://baike.baidu.com/item/%E5%BC%80%E6%BA%90/20720669)及私有[软件](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6/12053)项目的托管平台。

官网：https://github.com/

登录官网 -->> 注册 -->> 邮箱确认 -->> 创建仓库



把本地仓库项目上传到GitHub上步骤：

**增加远程地址**

**git remote add <远端代号> <远端地址>**

**<远端代号> 是指远程链接的代号，一般直接用origin作代号，也可以自定义。**

**<远端地址> 默认远程链接的url**

**例：git remote add origin https://github.com/xxxxxx.git**

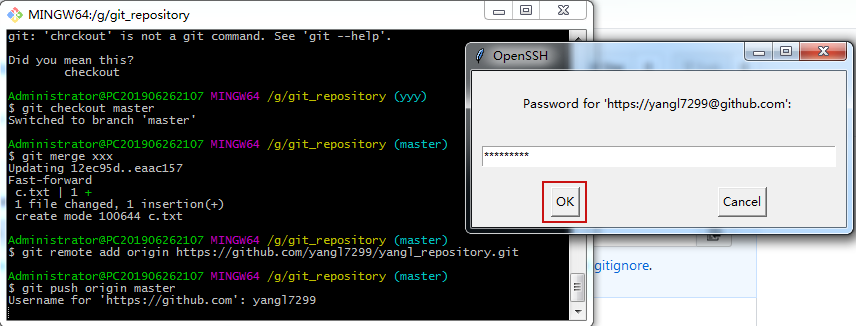
**推送到远程库**

**git push <远端代号> <本地分支名称>**

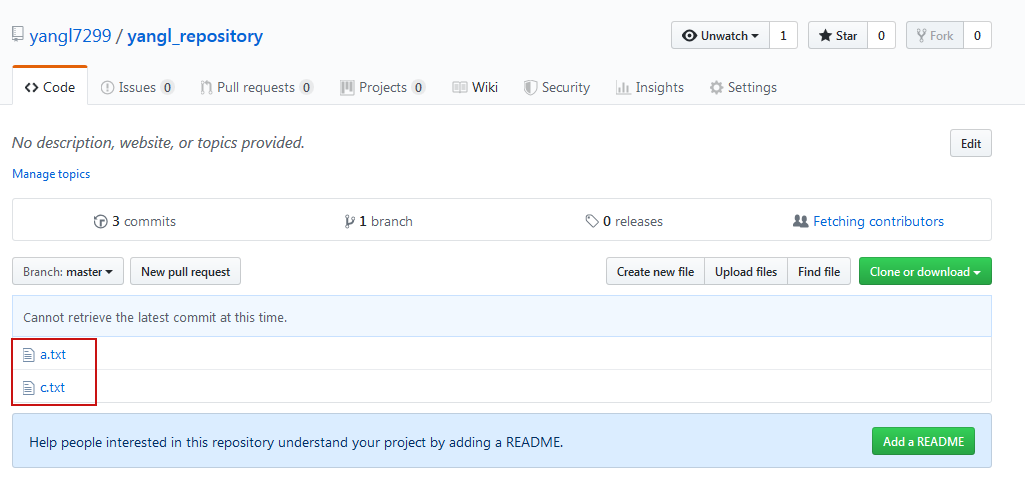
**<远端代号> 是指远程链接的代号。**

**<分支名称> 是指要提交的分支名字，比如master。**

**例：git push origin master**



提交完成之后，刷新浏览器，会显示工作目录提交的数据



**从GitHub上克隆（复制）一个项目**

**git clone <远端地址> <新项目目录名>**

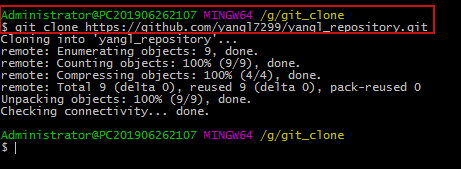
**<远端地址> 是指远程链接的地址。**

**<项目目录名> 是指为克隆的项目在本地新建的目录名称，可以不填，默认是GitHub的项目名。**

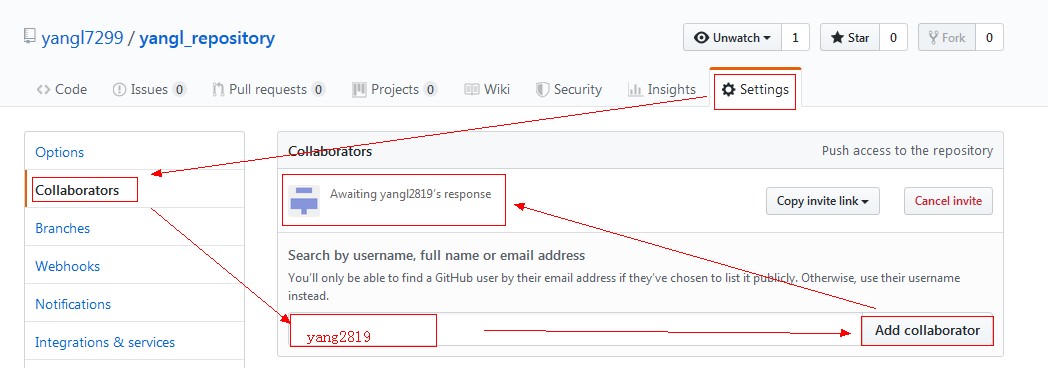
**命令执行完后，会自动为这个远端地址建一个名为origin的代号。**

**例：git clone https://github.com/xxxxxxx.git 文件夹名**

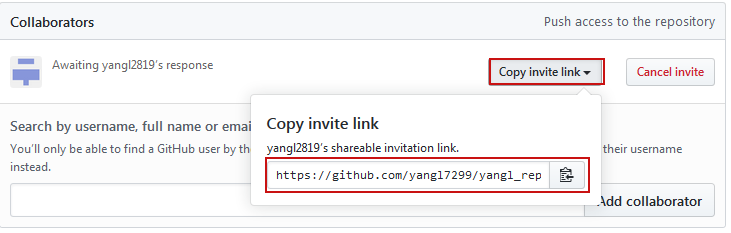
注意：在G盘下创建任意目录，然后右键打开git bash，再输入克隆命令

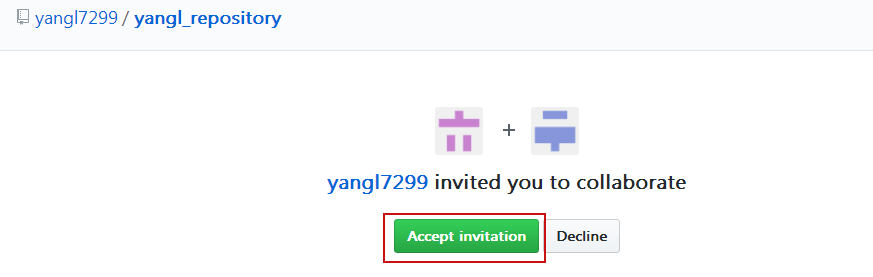


注意：如果是其他用户克隆项目之后，对项目进行了修改，然后想要推送到GitHub上时，需要当前用户授权才行！（例如：yangl2819想要对yangl7299的GitHub上的项目进行修改，需要yangl7299授权才行）



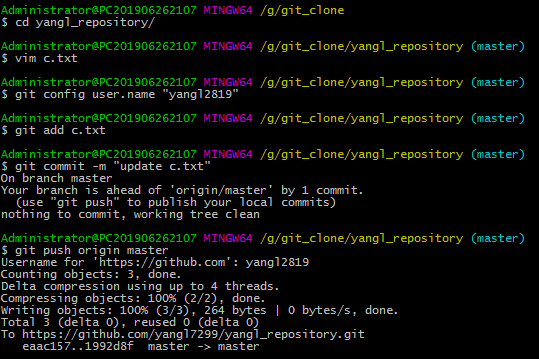
当前用户邀请其他用户成为伙伴时，需要其他伙伴同意，此时会发送一封邮件，需要其他用户登录网站进行同意，登录之后，拷贝邀请链接，用浏览器打开，同意该邀请





成为合作伙伴之后就可以更新仓库中的项目文件了

git config 用于配置当前提交文件的用户信息，可以配置用户名以及邮箱



合作伙伴修改完文件之后，推送到GitHub中的仓库中，此时仓库中的文件内容已经发生变化，当前用户如果需要和GitHub中的已经更新的文件内容保持一致，则需要更新项目

**从GitHub更新项目**

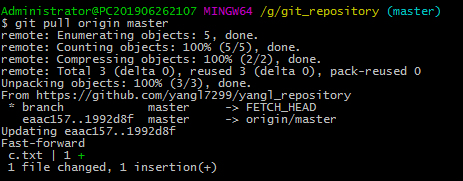
**git pull <远端代号> <远端分支名>**

**<远端代号> 是指远程链接的代号。**

**<远端分支名>是指远端的分支名称，如master。**

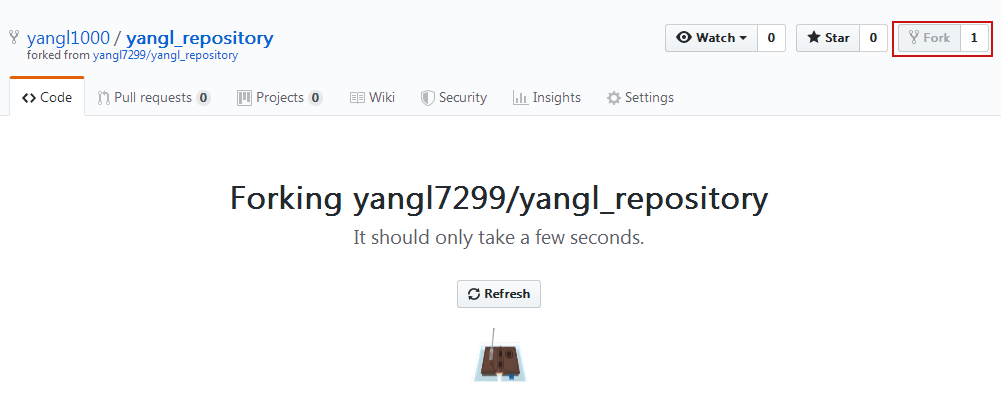
**例：git pull origin master**

打开当前用户的git bash 窗口，进行更新：

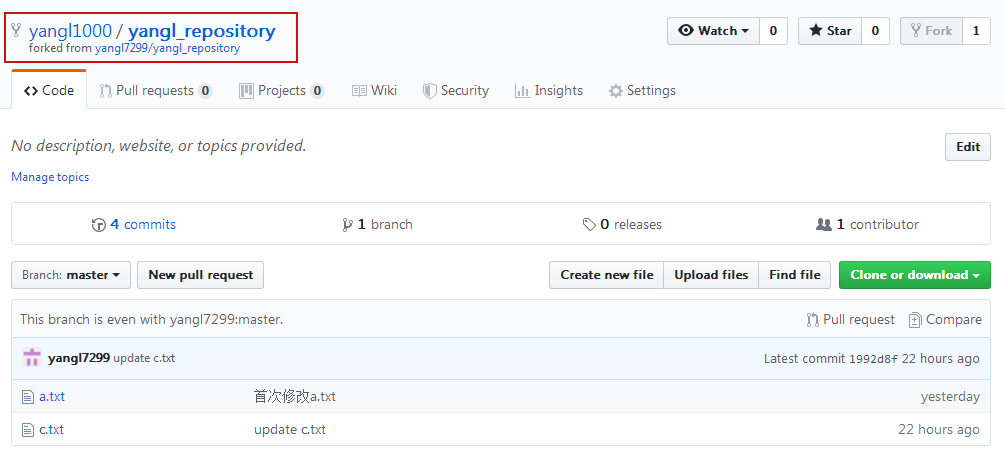


另一种情况：A，B都是该公司员工，可以共享同一个GitHub仓库，B无法解决项目中的问题，找C帮忙，C不是该公司员工，则只能查看该仓库，不能向仓库提交代码，如何来解决？

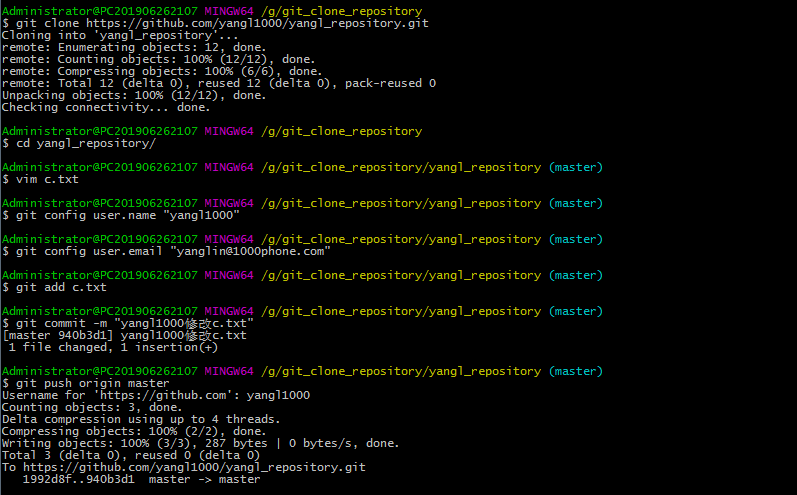
1.再创建一个新用户，然后登录，拷贝仓库地址（https://github.com/xxxxxxx.git），在浏览器执行，然后点击Fork，表示拷贝之前的仓库到该用户下



拷贝成功后如下：

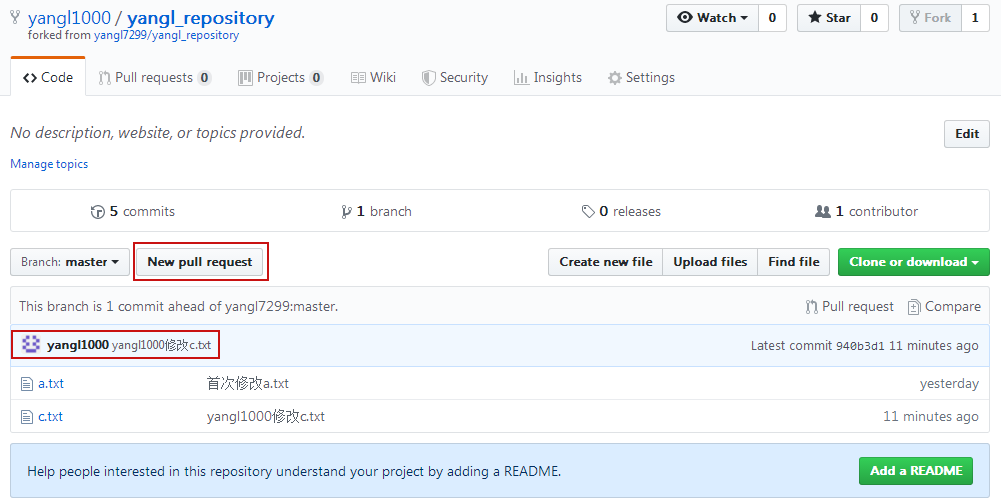


此时，相当于在GitHub中有了两个相同的仓库，一个为master，一个为Fork，然后在磁盘再创建一个目录，在目录下右键git bash 打开新窗口，进行如下操作：

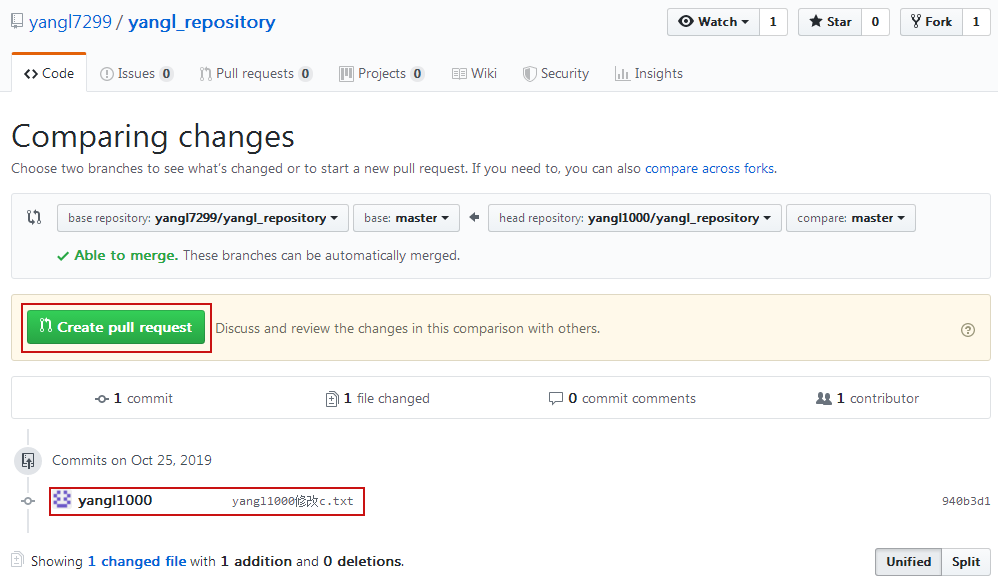


此时C把A仓库克隆下来之后进行了修改，然后又提交到了C自己的仓库，但实际运行的是A仓库中的代码，所以C需要向A发送请求，A需要同意请求，然后合并即可。

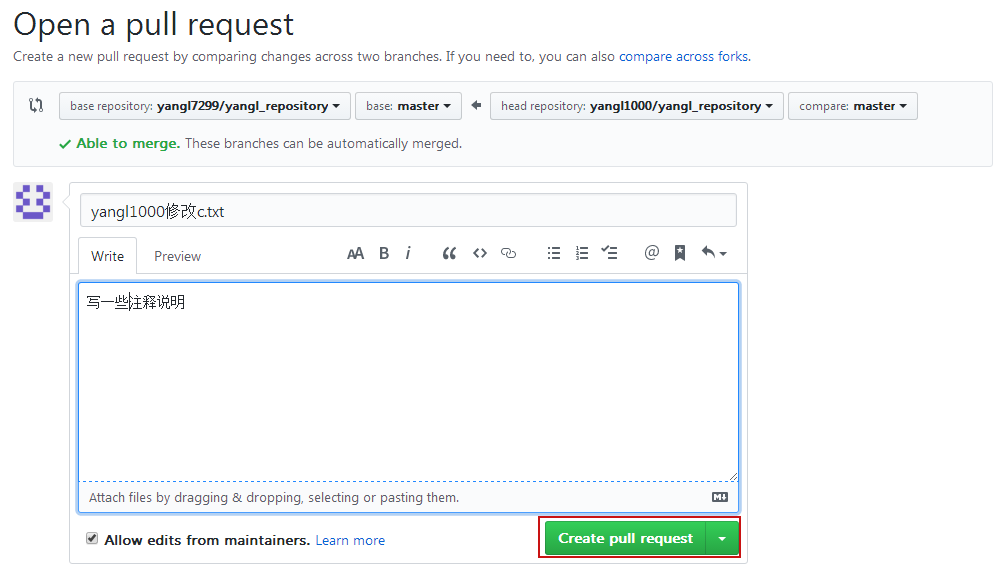
C发送请求：（首先刷新C当前浏览器页面，点击New pull requst）



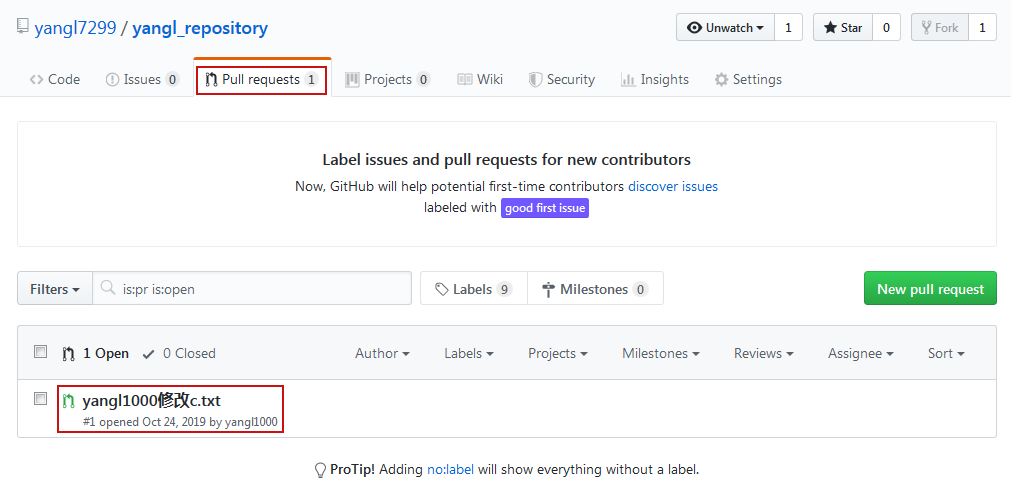
确认发送请求



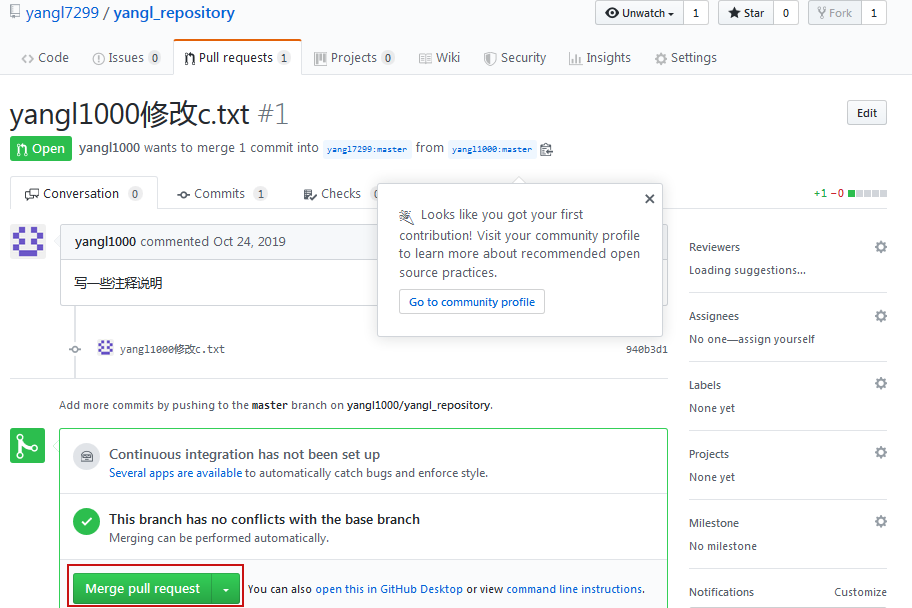
再次确认发送请求



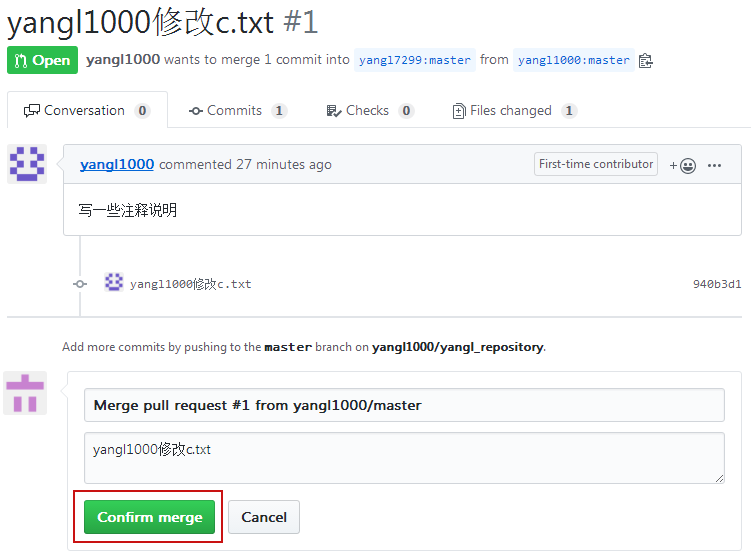
此时请求已经发送，A登录后查看请求，若已登录可直接刷新当前浏览器页面



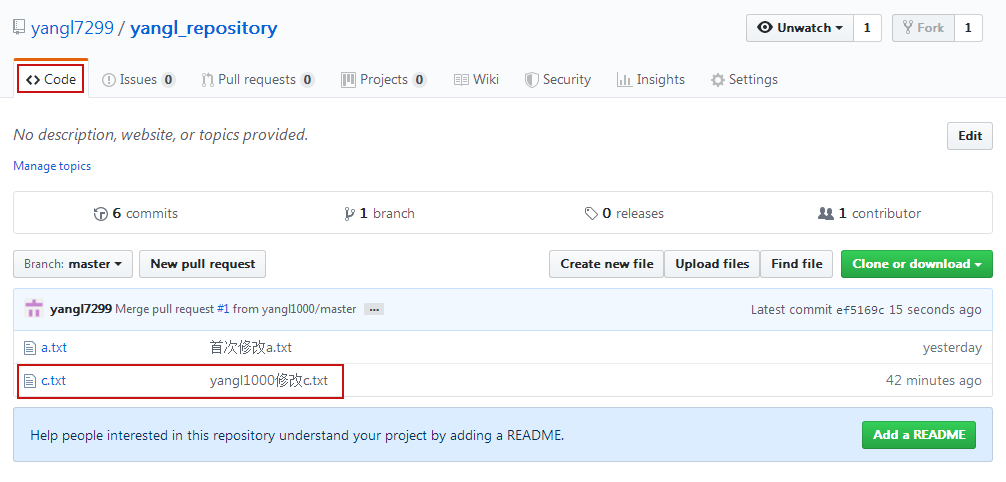
点击查看请求后，同意合并请求



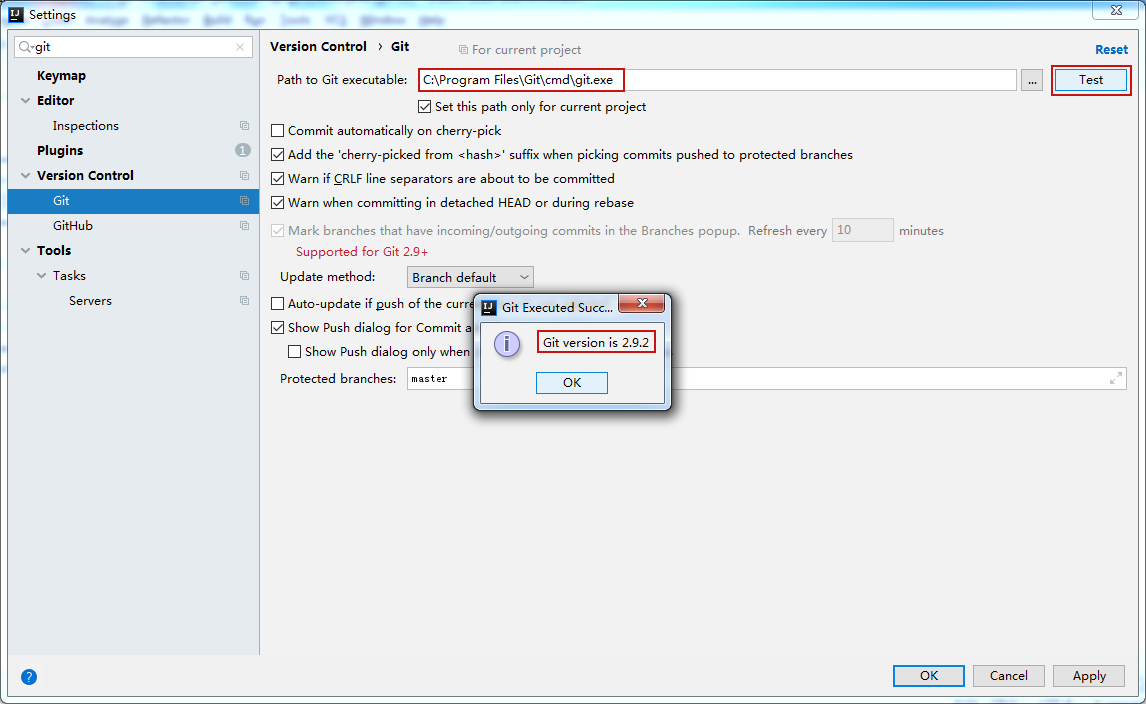
再次确认合并

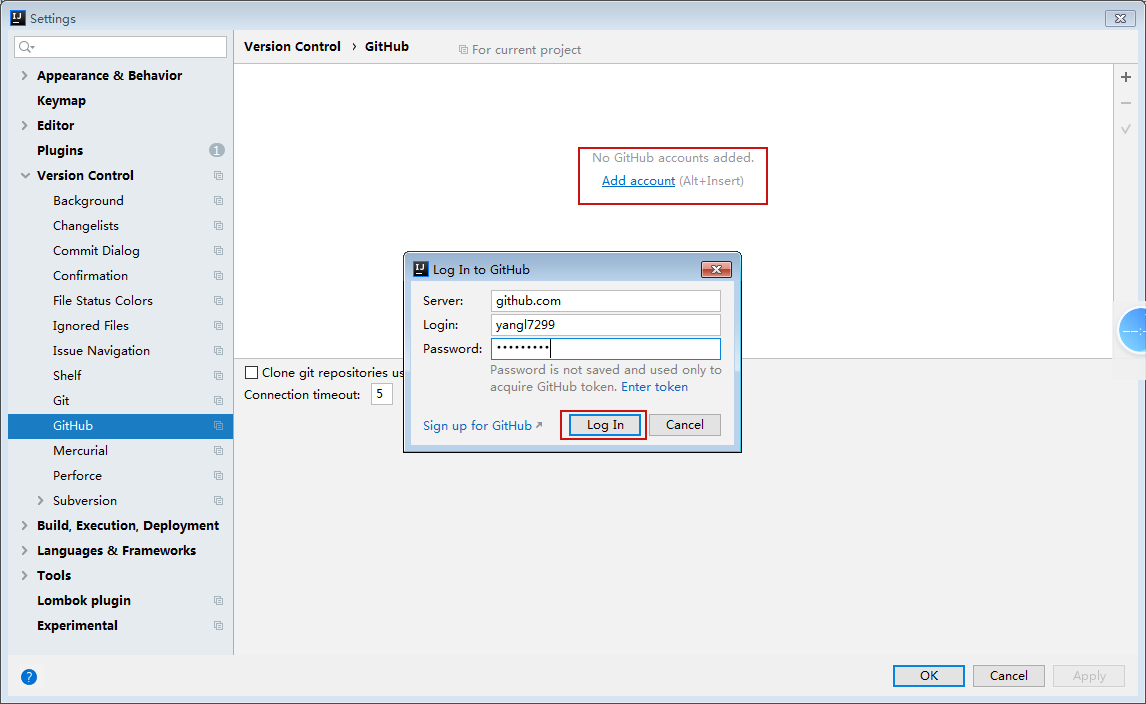


查看状态



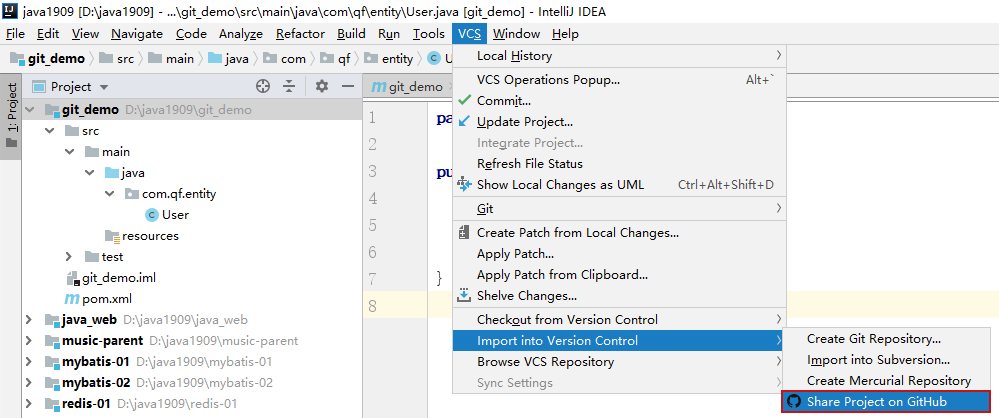
**7.IDEA中使用Git**



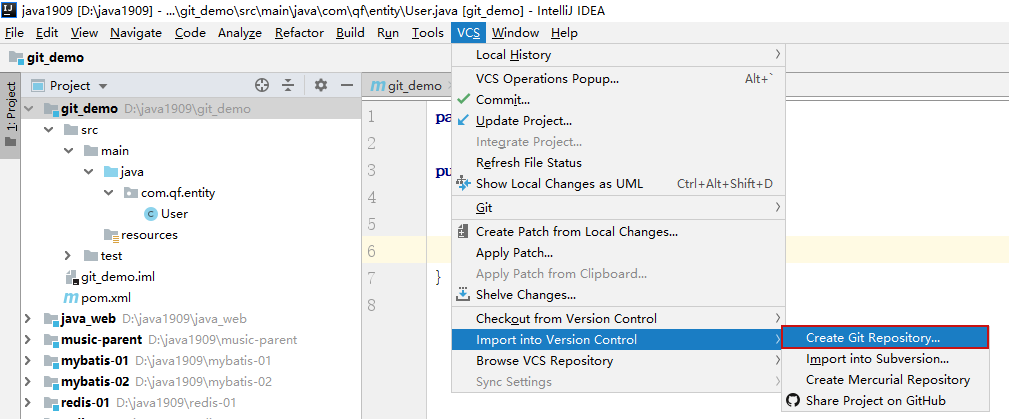


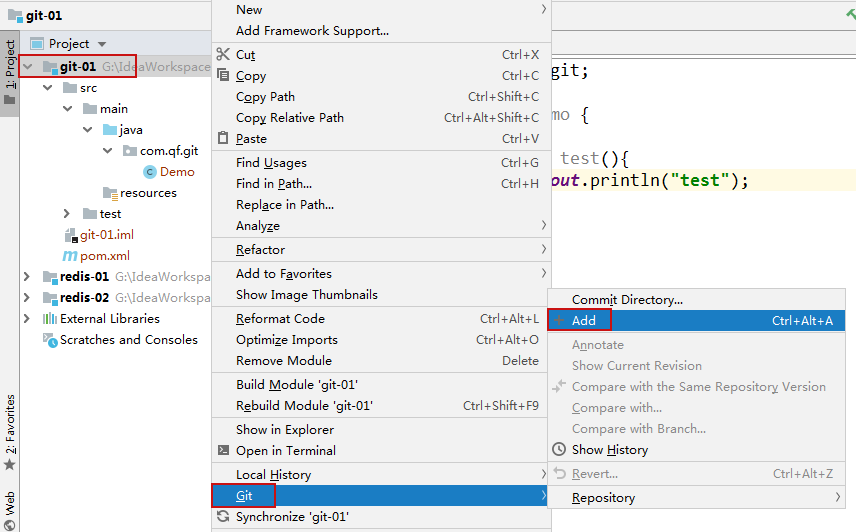
**本地项目推送到GitHub**

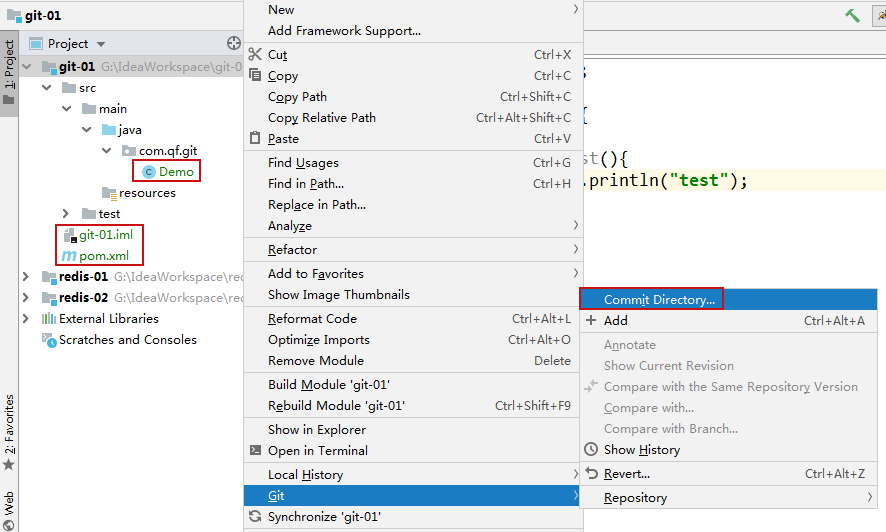
**第一种情况：如果需要在GitHub创建新的仓库，进行如下操作**

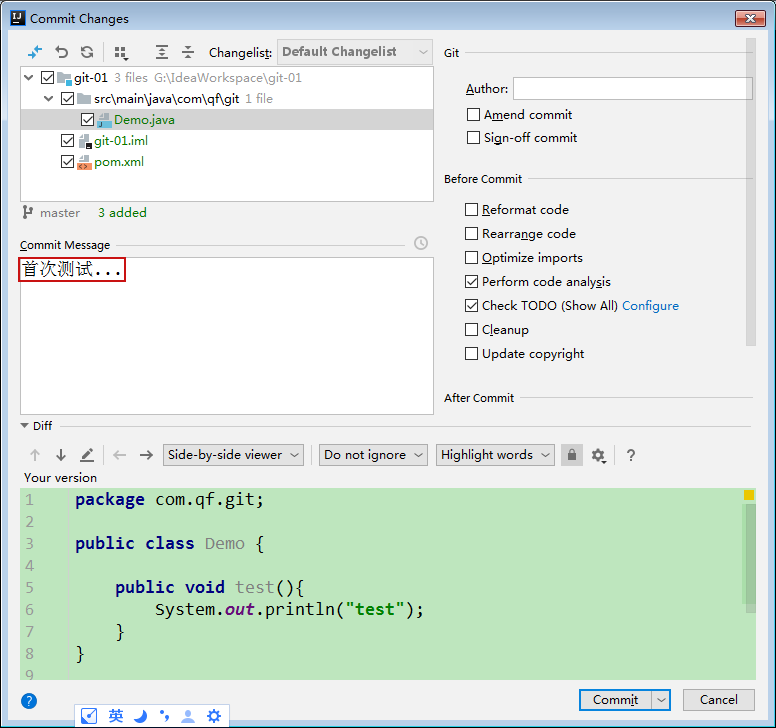


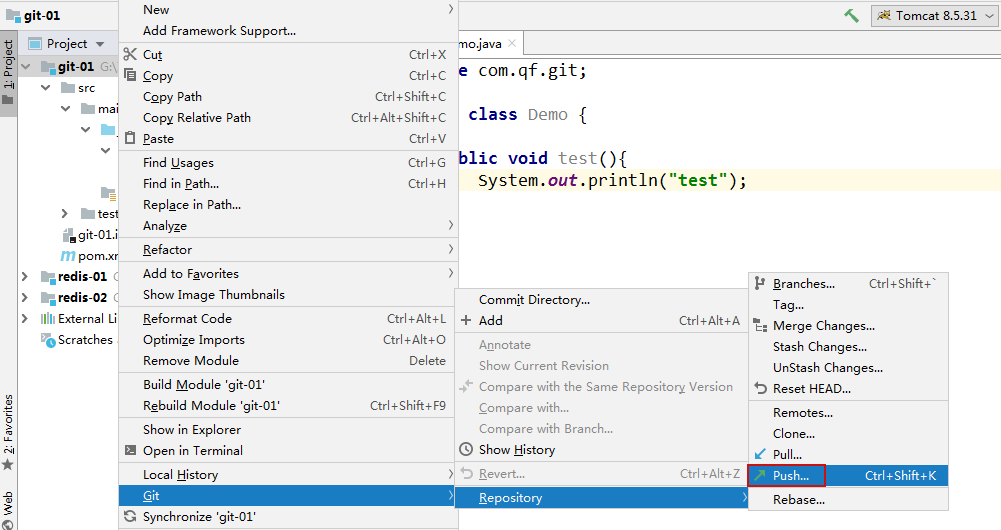
**第二种情况：GitHub上已经有仓库了，进行如下操作**

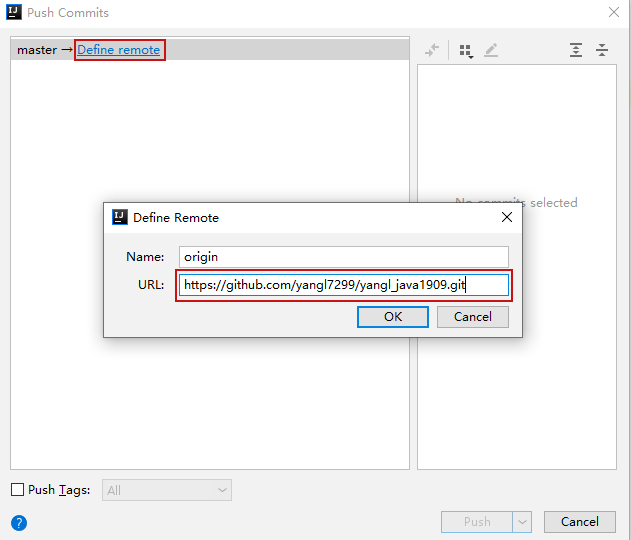


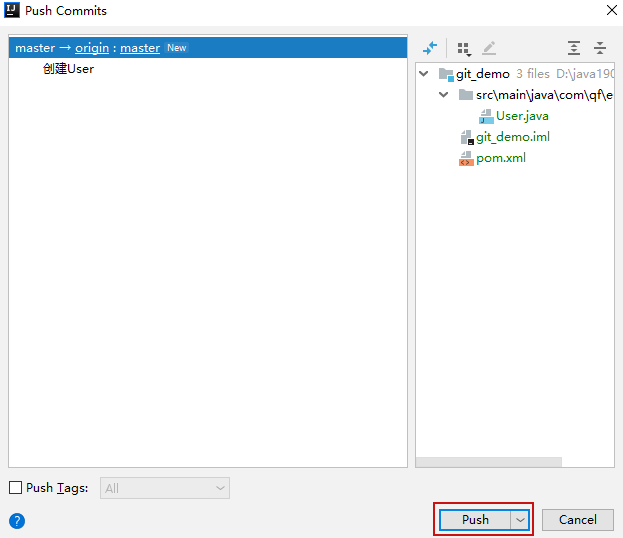




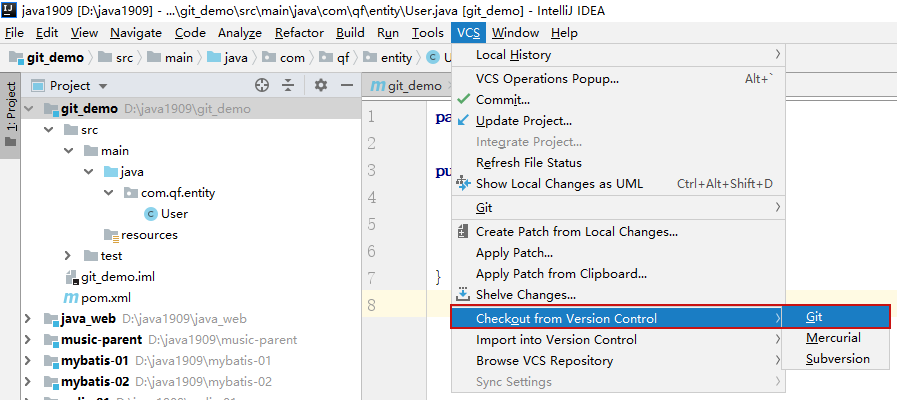


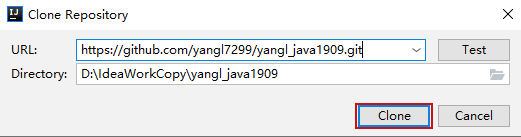




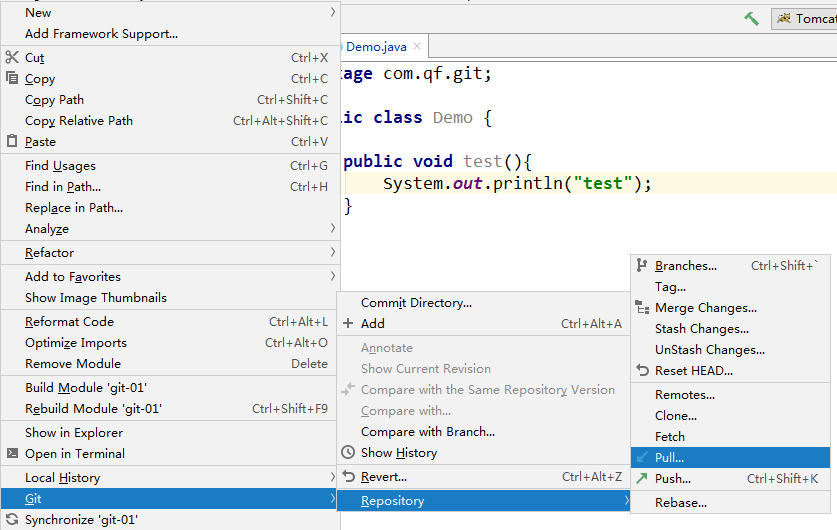


**从GitHub上下载项目**

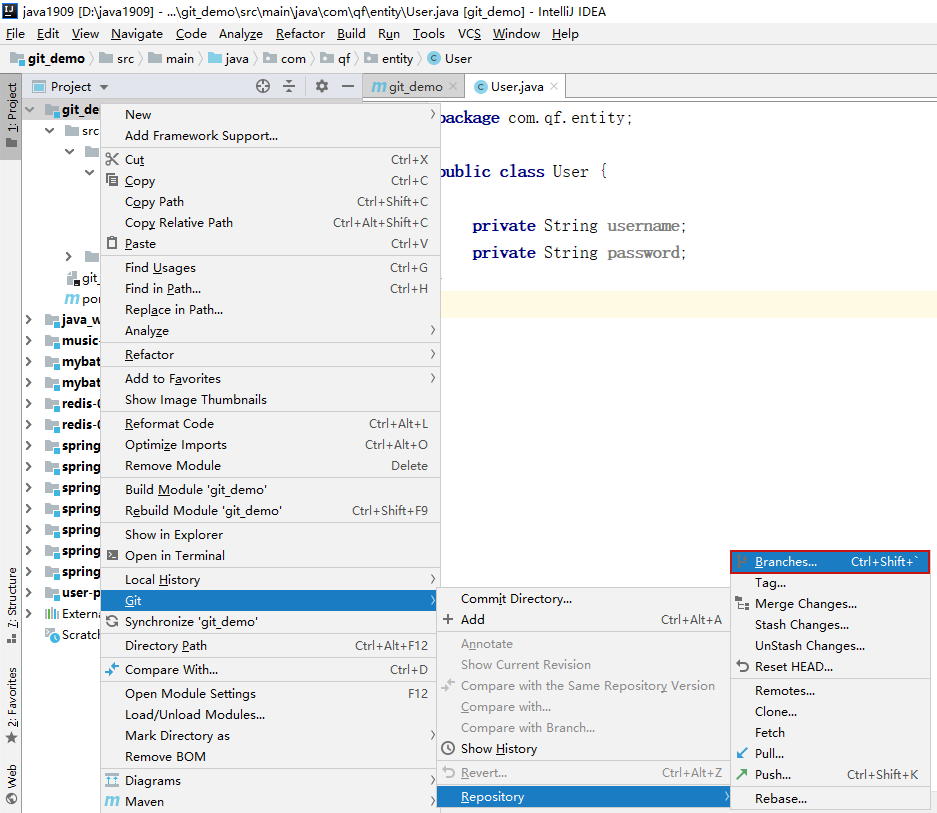




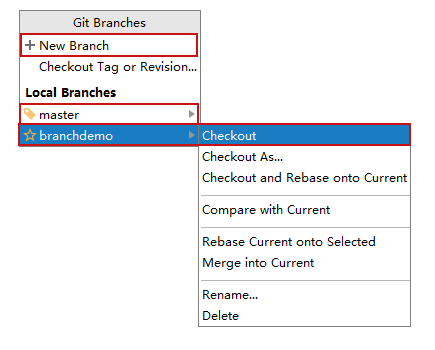
**从GitHub上更新项目**



**Git分支相关操作**



**创建分支，切换分支，切换主线**



测试时，先切换到分支，然后添加代码，提交，然后再切换到主线，分支下添加的代码就不显示了

**合并分支**

