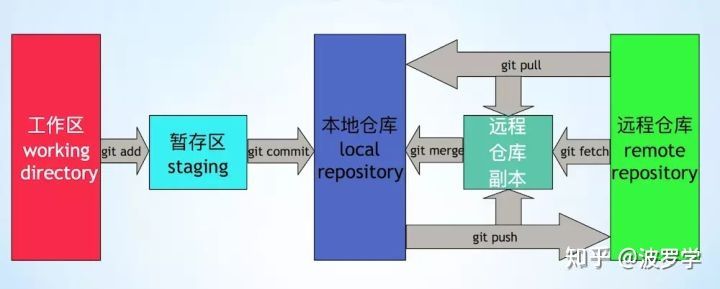
# Git简介

github是通过Git进行版本控制的软件源代码托管服务平台。github除了Git代码仓库托管及基本的Web管理界面以外，它还提供了一些方便社会化共同软件开发的功能，即一般人口中的社群功能，包括允许用户追踪其他用户、组织、软件库的动态，对[软件代码](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=4196027&ss_c=ssc.citiao.link" \t "_blank)的改动和bug提出评论等。

## 工作原理 / 流程



上图展示了git的整体架构，以及和各部分相关的主要命令，其中涉及的各部分。

* Workspace：工作区
* Index / Stage：暂存区
* Repository：或本地仓库
* Remote：远程仓库（Github上建的仓库就相当于远程仓库）

**工作区(working directory)，**简言之就是你工作的区域。对于git而言，就是的本地工作目录。工作区的内容会包含提交到暂存区和版本库(当前提交点)的内容，同时也包含自己的修改内容。

**暂存区(stage area, 又称为索引区index)，**是git中一个非常重要的概念。是我们把修改提交版本库前的一个过渡阶段。查看GIT自带帮助手册的时候，通常以index来表示暂存区。在工作目录下有一个.git的目录，里面有个index文件，存储着关于暂存区的内容。git add命令将工作区内容添加到暂存区。

**本地仓库(local repository)，**版本控制系统的仓库，存在于本地。当执行git commit命令后，会将暂存区内容提交到仓库之中。在工作区下面有.git的目录，这个目录下的内容不属于工作区，里面便是仓库的数据信息，暂存区相关内容也在其中。这里也可以使用merge或rebase将**远程仓库副本**合并到本地仓库。图中的只有merge，注意这里也可以使用rebase。

**远程版本库(remote repository)，**与本地仓库概念基本一致，不同之处在于一个存在远程，可用于远程协作，一个却是存在于本地。通过push/pull可实现本地与远程的交互；

**远程仓库副本，**可以理解为存在于本地的远程仓库缓存。如需更新，可通过git fetch/pull命令获取远程仓库内容。使用fech获取时，并未合并到本地仓库，此时可使用git merge实现远程仓库副本与本地仓库的合并。git pull 根据配置的不同，可为git fetch + git merge 或 git fetch + git rebase。rebase和merge的区别可以自己去网上找些资料了解下。

## Git安装与基本配置

**一、准备工作**

1.Git官网（<https://git-scm.com/>）下载Git。

2.GitHub官网（<https://github.com/>）注册账号。

3. 测试仓库 <https://github.com/liuxinfengabc/cultivate.git>

**二、安装Git**

1.点击安装，选择目录为根目录，文件夹为Git，然后Next->默认安装。

2.安装完成后打开根目录，找到Git文件夹，右键git-bash.exe可执行文件将其固定的工具栏，点击git-bash.exe运行。

3.输入指令 ls 查看文件目录。

此为运行正常安装完成！

**三、运行Git**

1.设置用户名和邮箱。

设置用户名：$ git config --global user.name "Your Name"。

设置邮箱： $ git config –global user,email "email@example.com"。

设置用户名和邮箱，修改后提交到远程仓库能够看到是谁提交的

2.输入 $ git init 启动Git。

**四、在GitHub上设置公钥**

1.新建/连接主存储库。

2.输入 $ ssh-keygen -t rsa -C "[email@example.com](mailto:%2213963639551@163.com%22)" 生成公钥，在C:\Users\Administrator\.ssh下找到id\_rsa.pub文件，记事本打开复制公钥。

3.在GitHub上新建密钥，自定义标题（title）将公钥粘贴保存。

**五、创建git库**

　　git init #在当前目录中生成一个.git 目录（含有.git目录的目录即是git仓库）

**六、注册git用户**

--->用于在团队合作开发中，表明代码作者。

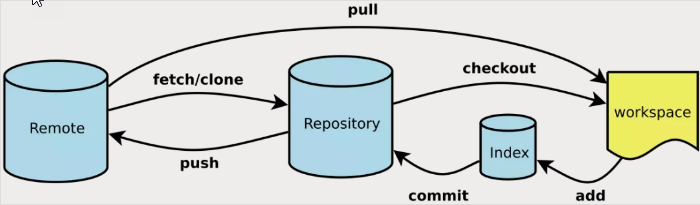
　　git config --global user.name XXX #用户名

　　git config --global user.email XXX #用户邮箱

　　git config --list #查看用户信息

注：加--global，全局设置。

## 命令状态机



1.Workspace---->index 向git库添加修改

**git add [path］ #会把对应目录或文件，添加到stage状态**

　　git add . #会把当前所有的untrack files和changed but not updated添加到stage状态

实际上是为修改内容添加index索引。

2.index--->Repository 向版本库提交修改

**git commit –m “XXXX” #提交修改,添加注释**

注：git 提示： 未有add红色字体，未有commit绿色字体，已提交则worktree是干净的

3、Repository---->Remote 向远端库推送修改（提交修改）

**git push origin master【分支名】**

4、Remote----> Repository 远端库到本地库

**git fetch**

5、Repository---->WorkSpace 本地仓库到工作区

**Git checkout**

6. Remote----> Workspace    下拉指定主机的指定分支，并与本地的指定分支合并：

**git pull <origin> <远程master>:<本地master>**

7、查看当前代码库的状态

　　git status

8、查看版本信息--->实际是查看修改提交信息

　　git log

git log --graph #以图形化（节点）展示当前git库的提交信息。

## 常用操作

### 克隆远程存储库到本地

新建一个文件夹作为本地仓库，右击点击 Git Bash Here，把远程github的项目clone到本地，远程仓库的地址<https://github.com/liuxinfengabc/cultivate.git>

命令：

$ git clone <https://github.com/liuxinfengabc/cultivate.git>

Git clone相当于以下四个步骤

>git init(创建本地存储库)  
>git remote add(将URL添加到该存储库)  
>git fetch(从该URL中获取所有分支到本地存储库)  
>git checkout(创建工作树中主分支的所有文件)

### 添加并推送文件到GITHub

1.把项目文件复制到本地的 ./Git/XXXX 存储库中。

2.将项目文件添加到工作区中。

命令：git add path

3.将工作区中的项目文件存入master分支中。

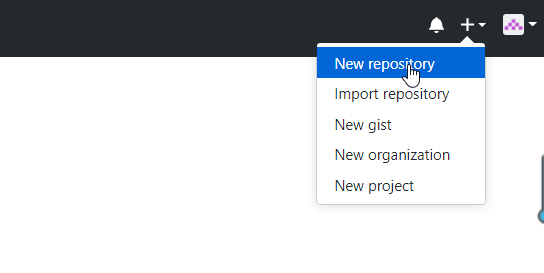
命令：git commit -m "XXXXX"

4.将本地的项目文件推送到GitHub服务器上。

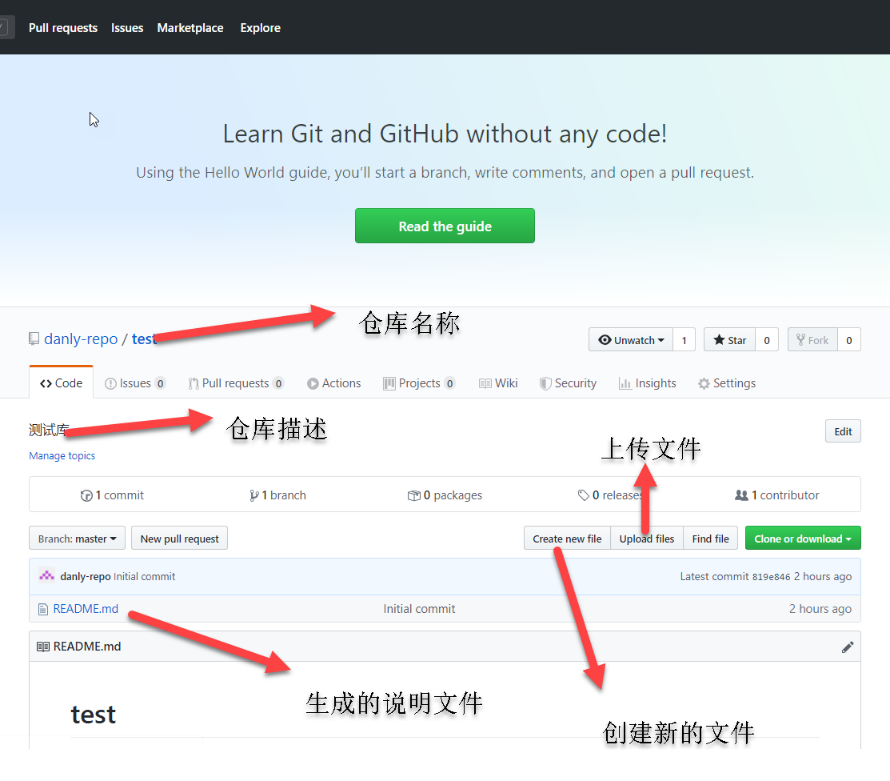
命令：git push origin master

# GitHub操作

## 创建仓库

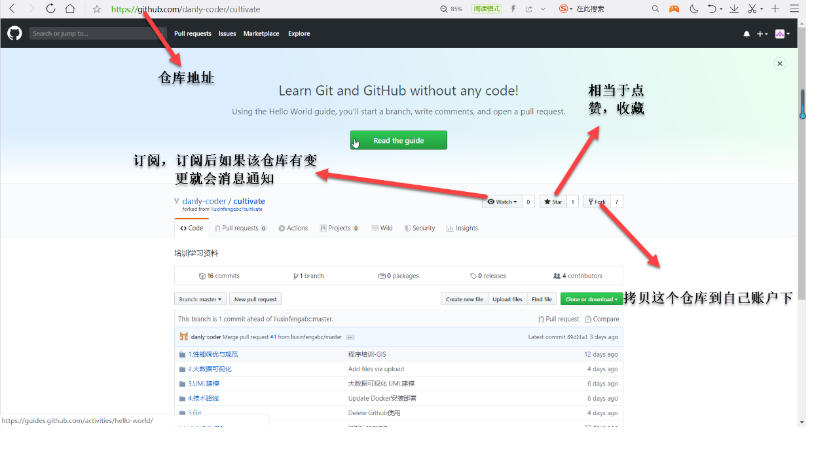


生成的仓库如下

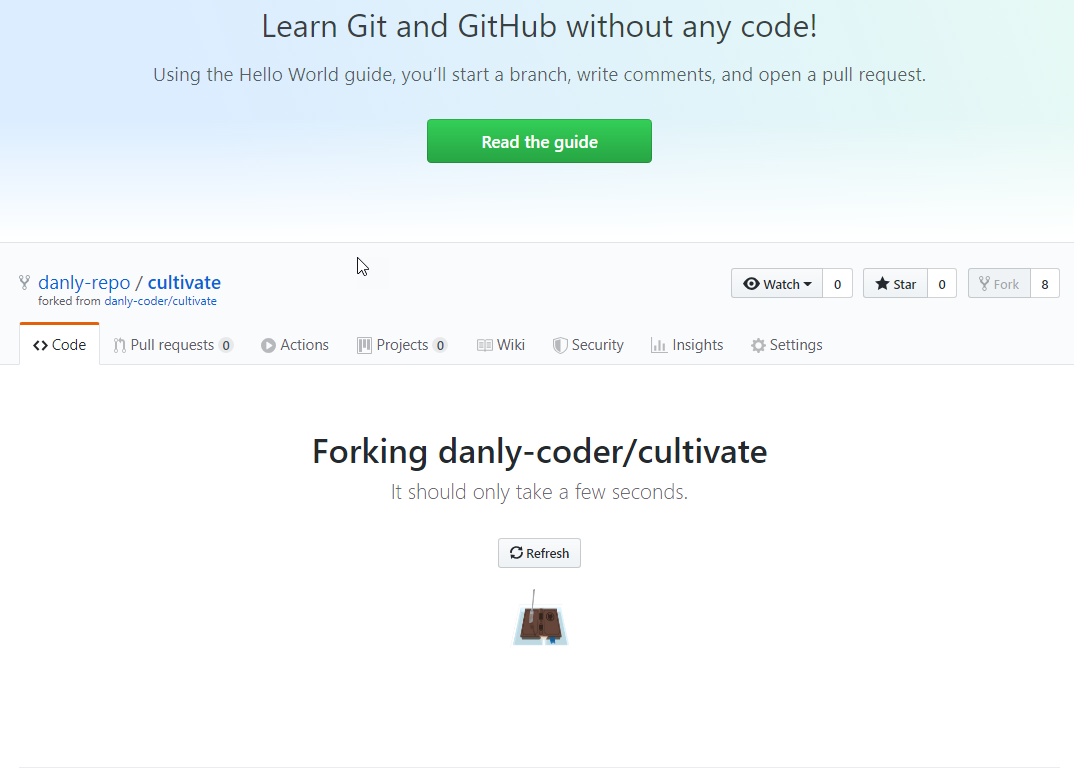


打开另外一个仓库的网址<https://github.com/danly-coder/cultivate>

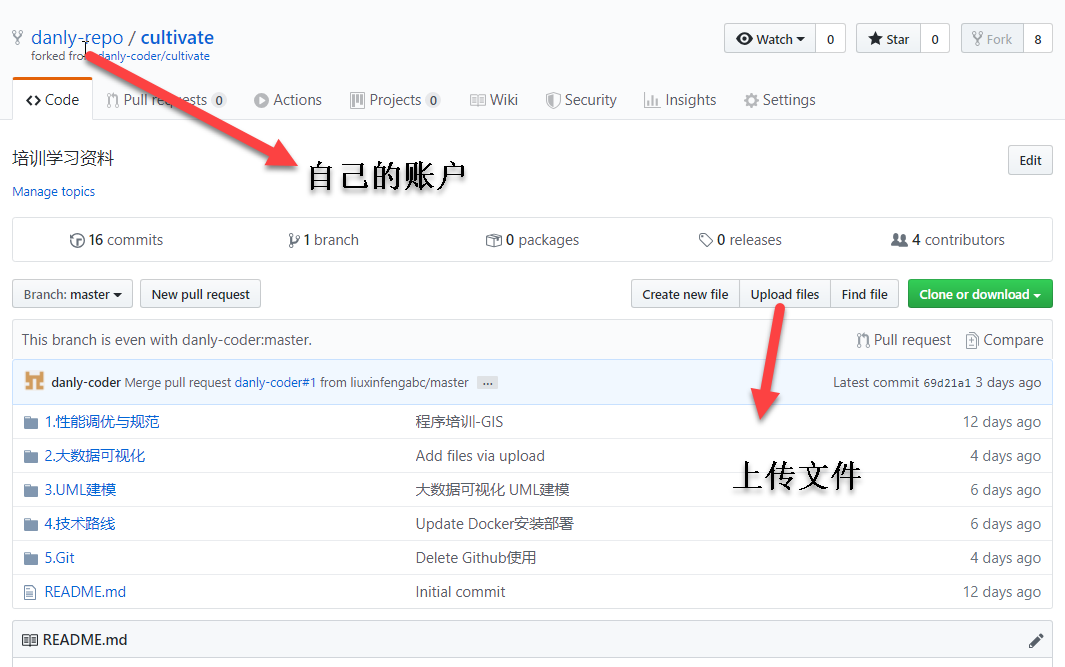
这是danly-coder用户下cultivate库地址



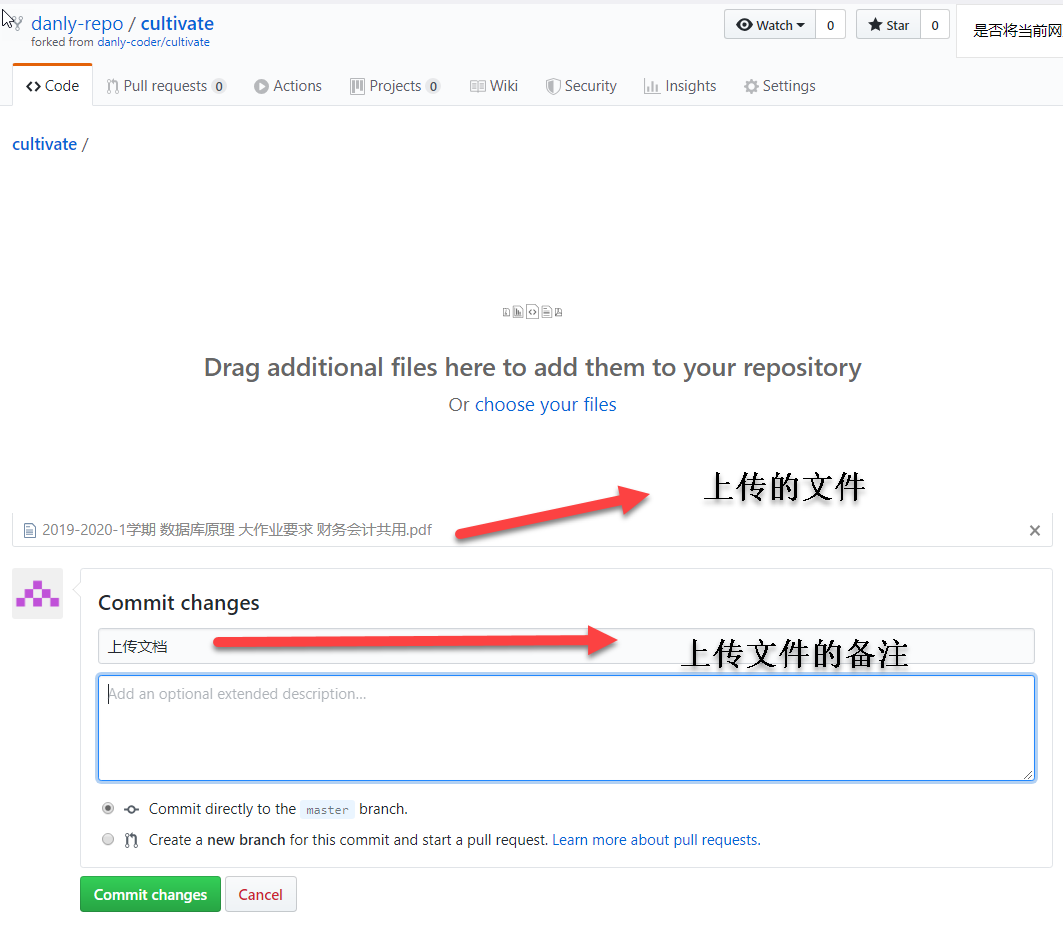
Fork该仓库



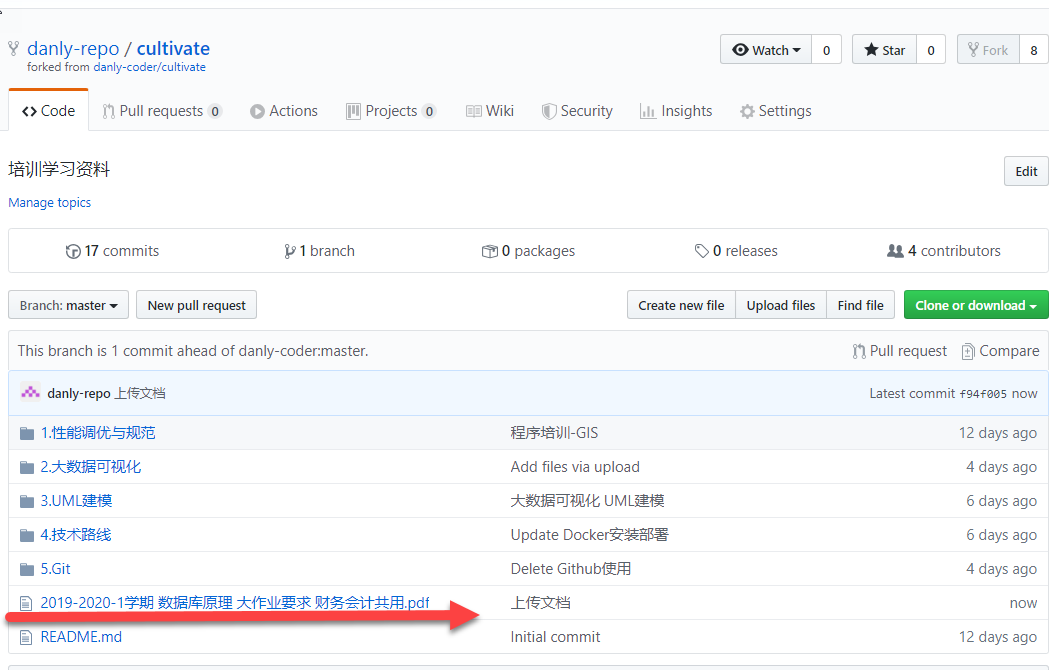
在自己的账户就是danly-repo下就可看到cultivate库



点击上传文件后进行上传，添加备注后提交改变。

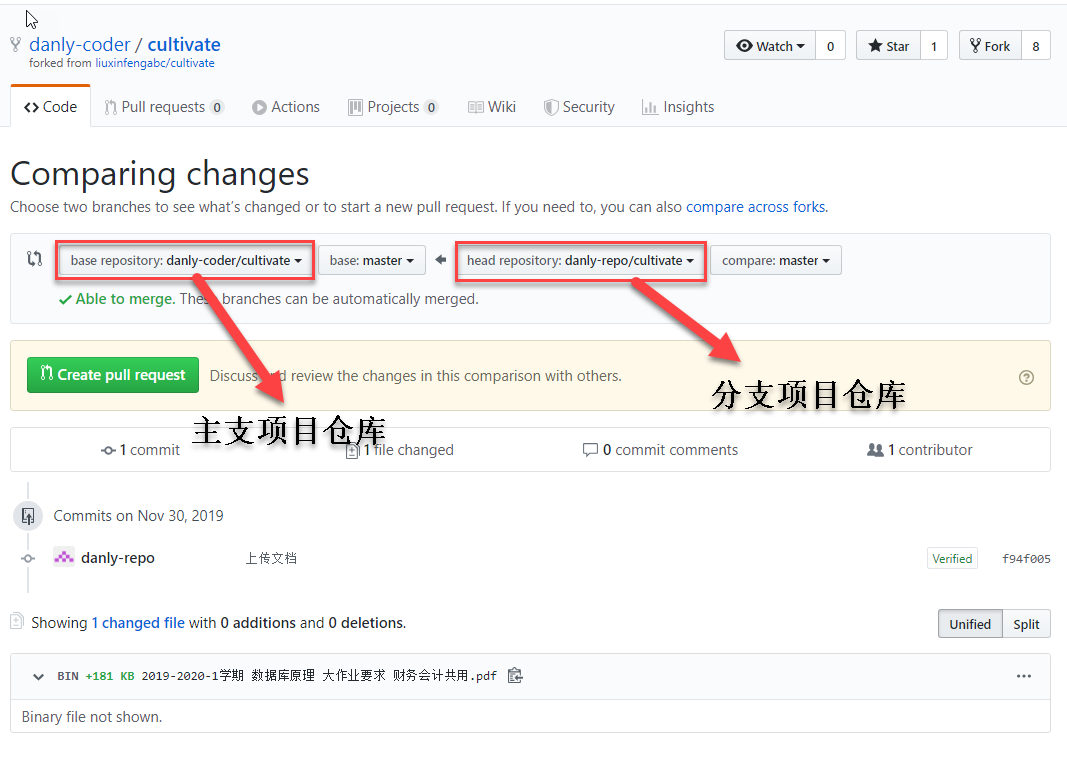


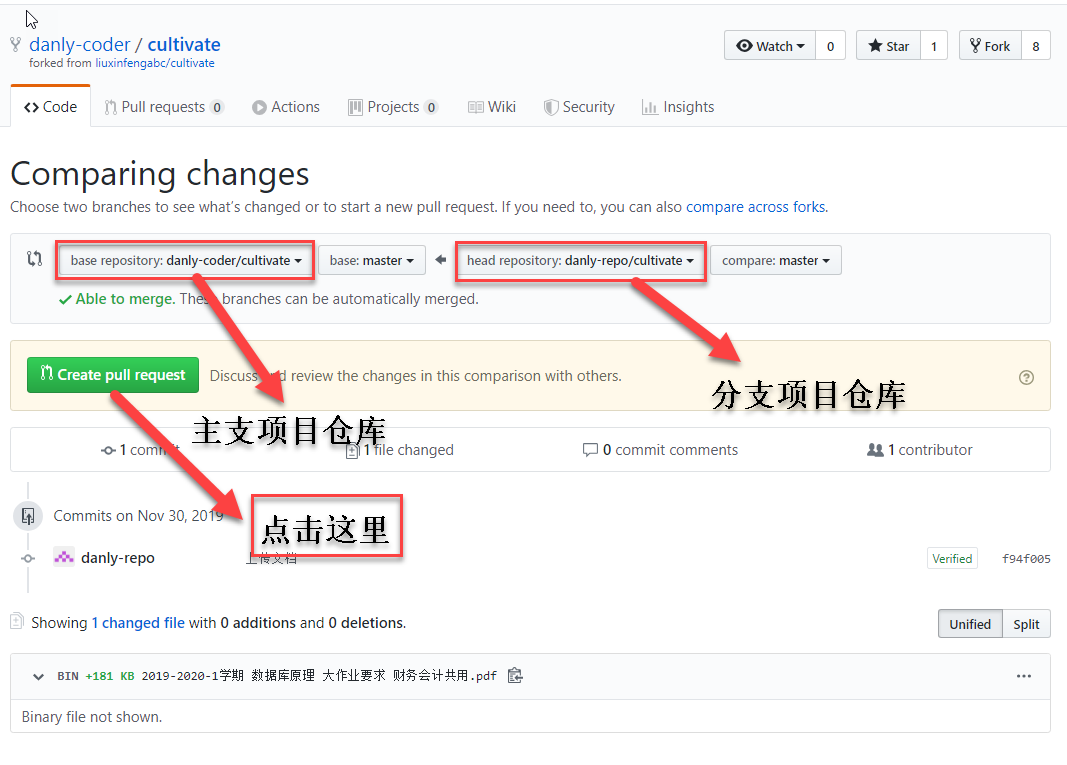
可以看到这个账号下cultivate库增加了文件



Repo账号的cultivate是从coder账户下拷贝过来，相当于coder账户的一个分支，现在分支里的内容发生了改变，主支没有改变，我们可以改变后的内容new pull request,这样就能够把修改后的文件提交给主支。



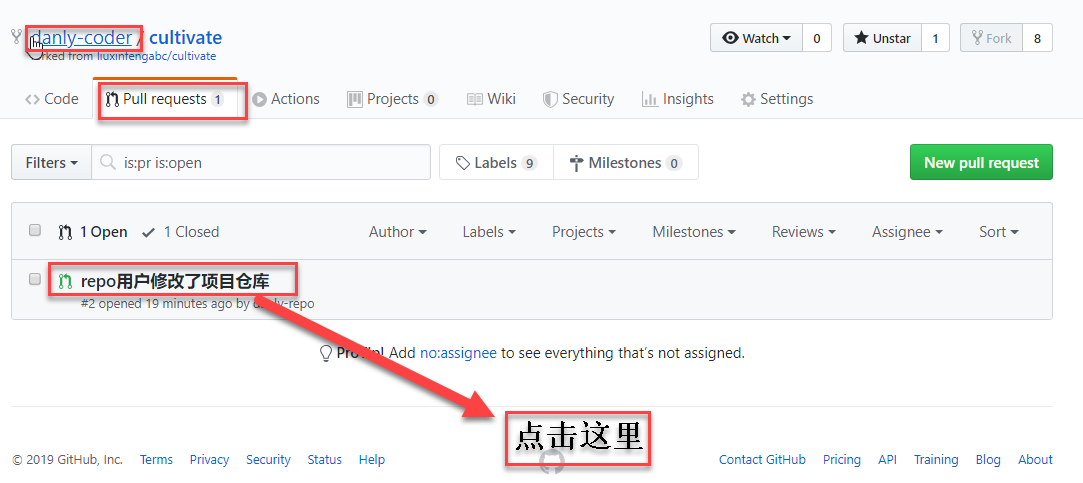


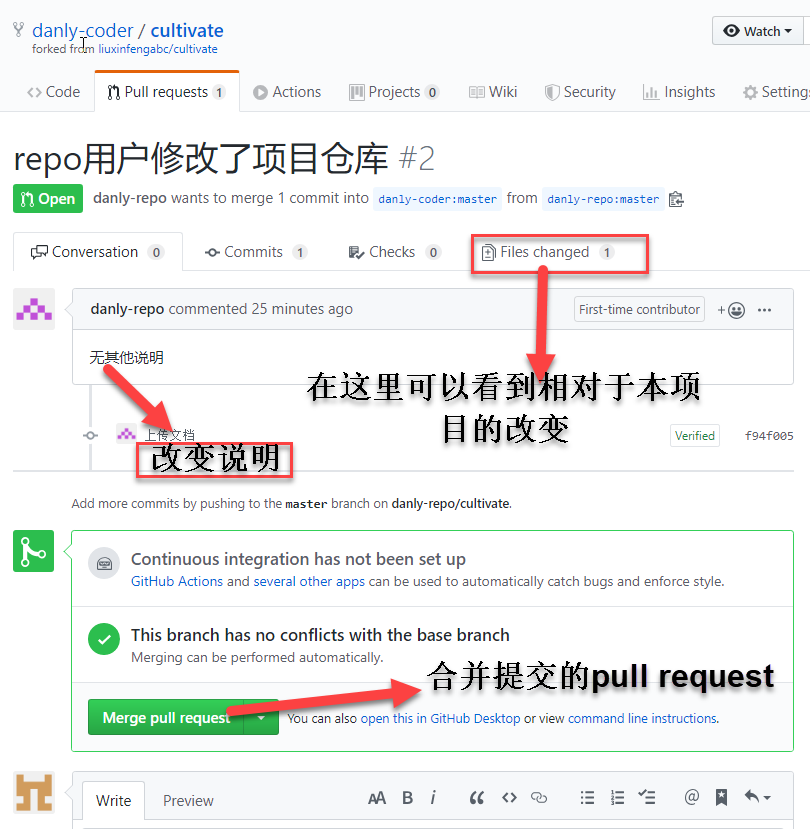




这样就完成了pull request，向主支仓库提交了修改后的项目。

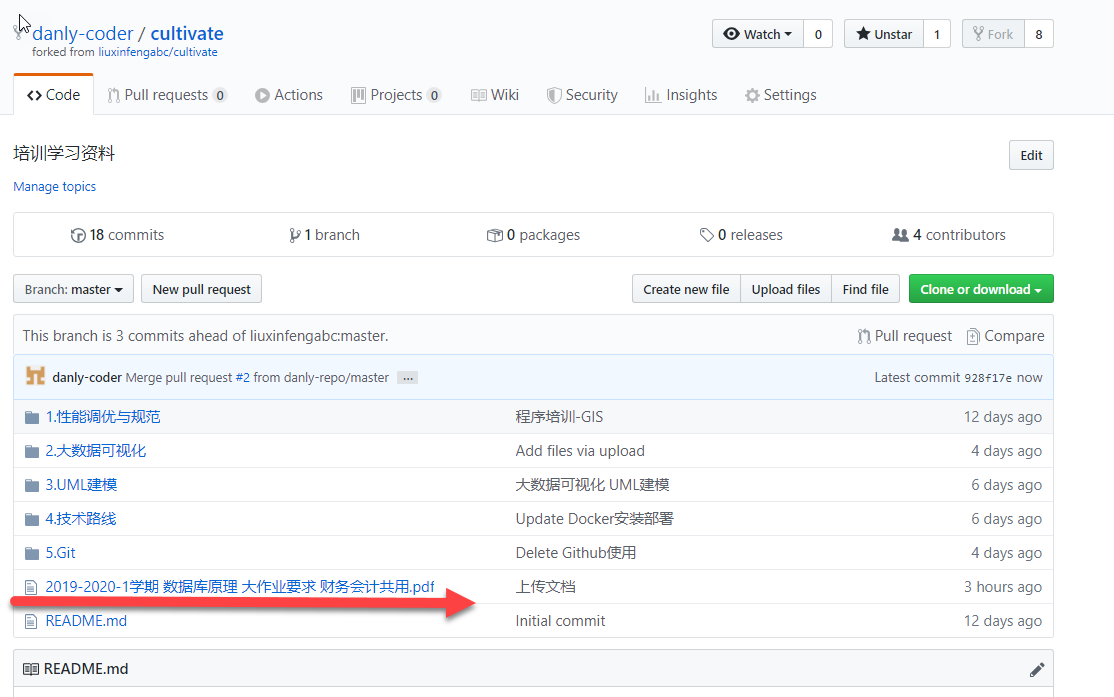
登录coder账户，点击Pull request，可以看到repo用户修改了项目仓库。





Coder账户如果觉的pull request的项目仓库可以，就可以Merge pull request（合并到主项目项目），这样主支项目仓库和分支项目仓库达成一致。如果觉的修改的不合适，就不用理睬。

可以看到coder项目仓库下就产生了PDF文件。

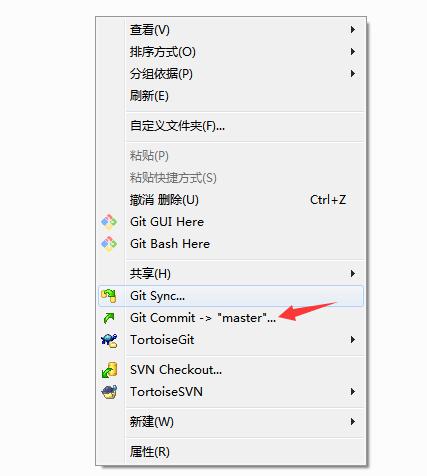


# 开发环境中的使用

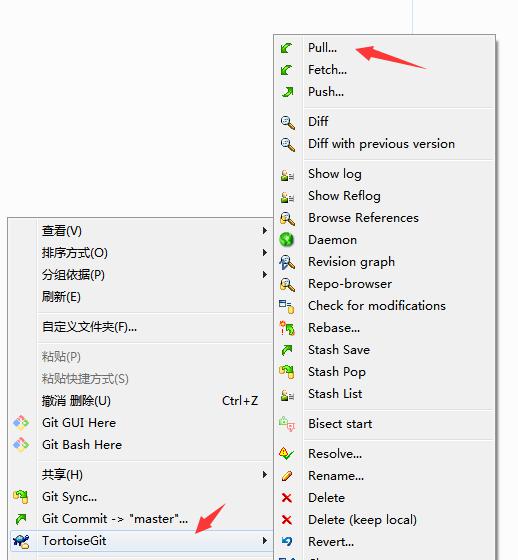
## tortoiseGit使用(windows操作）

第一步：详细见（二、Git本地仓库创建及操作）

第二步：修改了文件后，右键选择commit，输入Message后，点Commit提交到本地仓库。



第三步：拉取服务器文件，git pull，点击OK，后点击close



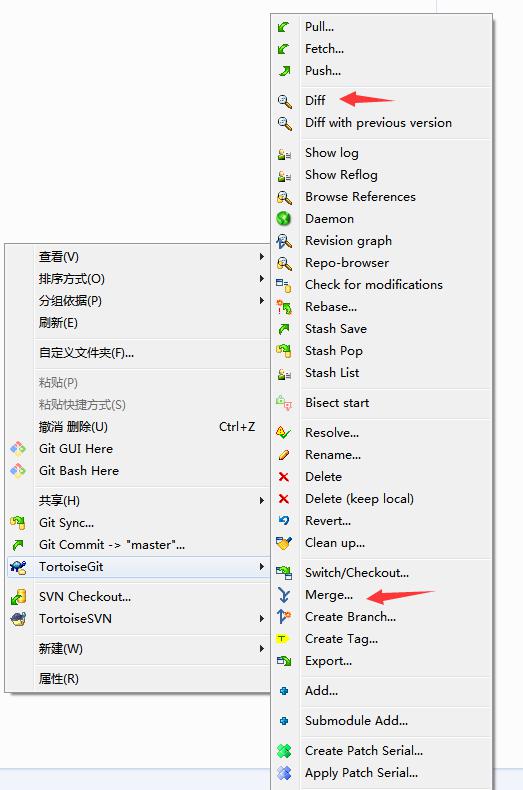
如果文件出现感叹号则表示有冲突，必须先解决冲突。

冲突解决方法：

1.在点击close后可以直接在弹出的窗口中点击yes来查看冲突，也可以通过点击Diff看本地仓库文件和服务器上面的文件哪里冲突了。

2.在弹出的窗口中双击打开冲突文件，解决冲突问题。

3.冲突解决完之后，删除在目录下生成出的多余文件，再次commit。



第四步：提交代表到服务器， git push



新增篇：

第一步：在仓库中新增文件后，先pull，然后可以在diff中查看和仓库区别

第二步：右键->commit，选择你需要提交的文件，然后点击commit

第三步：push文件

## eclipse中使用Git导入工程

1.打开Git Repositories窗口，点击右上角Add an existing local Git Repository to this view图标

2.在Directory中选择你本地Git总仓库，在下方选择你想导入的Git仓库目录

3.打开你导入的仓库->Working Tree，选择你想导入的工程，右键->Import Projects...

4.调整选项，导入工程

5.右键已导入工程->Gradle->Refresh Gradle Project

## eclipse中使用Git同步工程

1.右键工程->Team->Synchronize Workspace进入同步窗口

2.对双向红箭头文件，打开，进行代码合并操作，操作完成后 右键->Mark asMerged

3.选择要上传的文件，右键->Add to Index/在Unstaged Changes中右键->Add to Index

4.在Staged Changes中可以查看要提交的文件，在Commit Message中输入提交内容备注，以及作者相关，点击commit至本地

5.Pull拉取服务器上更新文件

6.最后Push至服务器