# **Git的使用--如何将本地项目上传到Github（两种简单、方便的方法）**

****将本地项目上传到Github（两种简单、方便的方法）****

****一、第一种方法：****

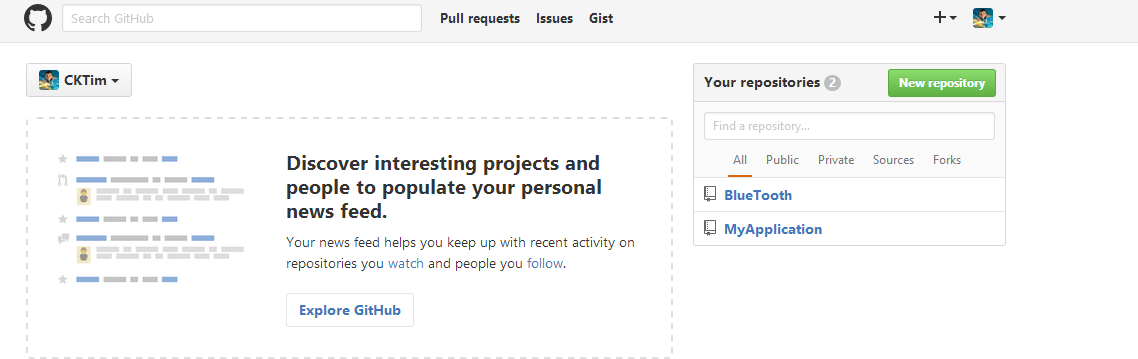
****首先你需要一个github账号，所有还没有的话先去注册吧！****

****https://github.com/****

****我们使用git需要先安装git工具，这里给出下载地址，下载后一路直接安装即可：****

****https://git-for-windows.github.io/****

**1.进入Github首页，点击New repository新建一个项目**



**2.填写相应信息后点击create即可**

****Repository name: 仓库名称****

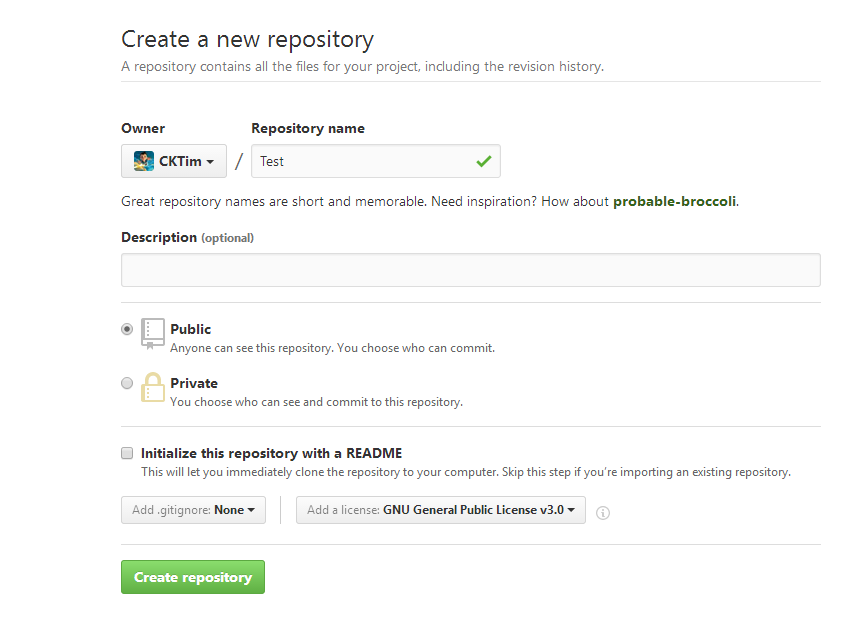
****Description(可选): 仓库描述介绍****

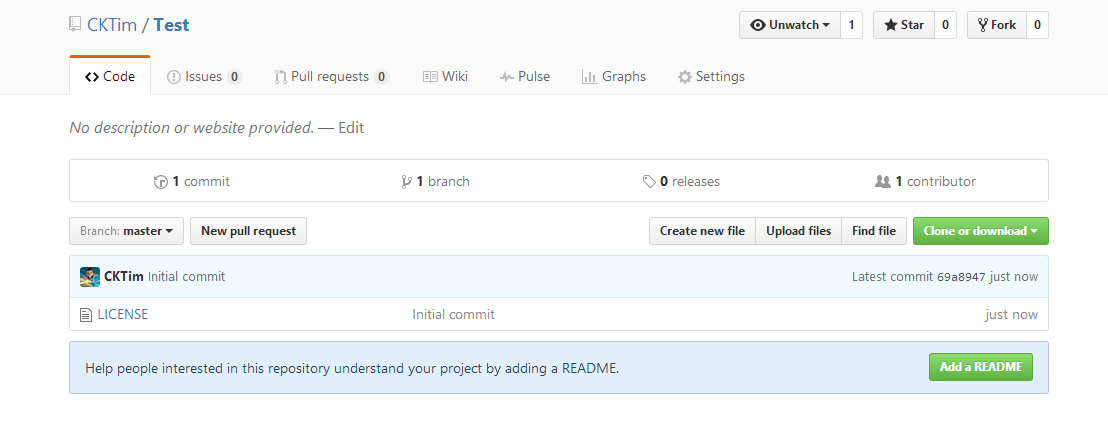
****Public, Private : 仓库权限（公开共享，私有或指定合作者）****

****Initialize this repository with a README: 添加一个README.md****

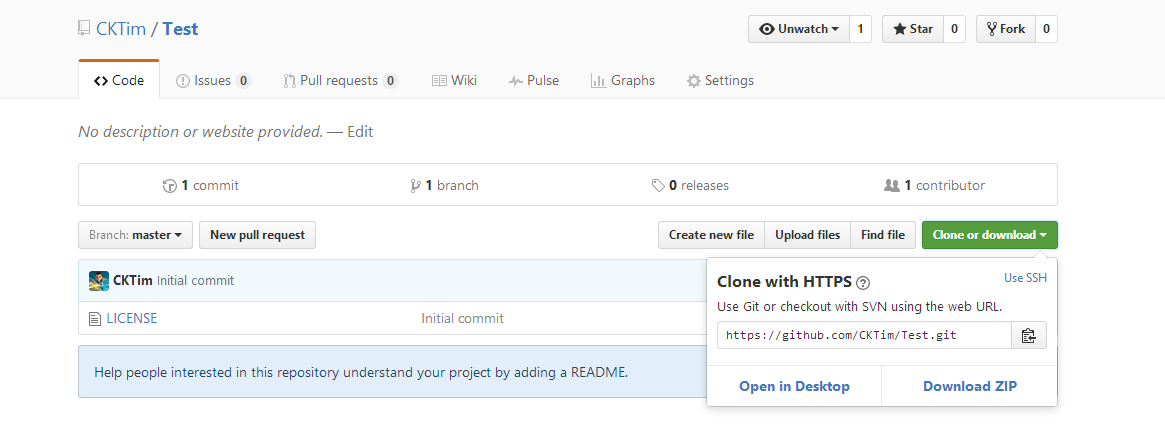
****gitignore: 不需要进行版本管理的仓库类型，对应生成文件.gitignore****

****license: 证书类型，对应生成文件LICENSE****

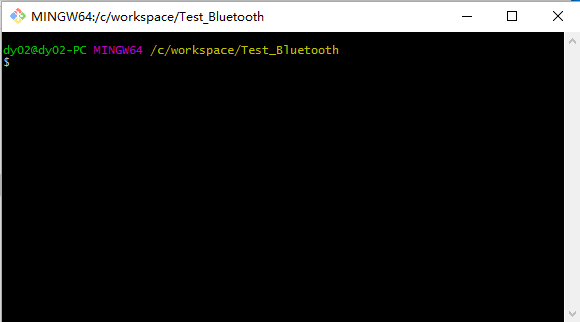




**4.点击Clone or dowload会出现一个地址，copy这个地址备用。**

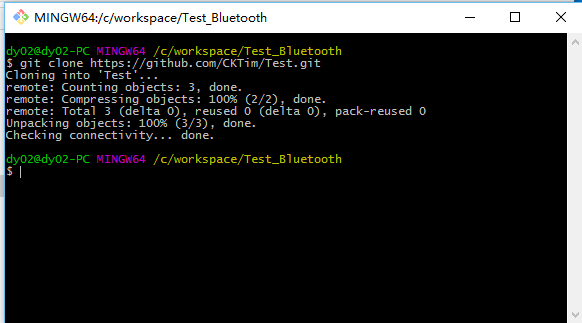


**5.接下来就到本地操作了，首先右键你的项目，如果你之前安装git成功的话，右键会出现两个新选项，分别为Git Gui Here,Git Bash Here,这里我们选择Git Bash Here，进入如下界面，Test\_Bluetooth即为我的项目名。**

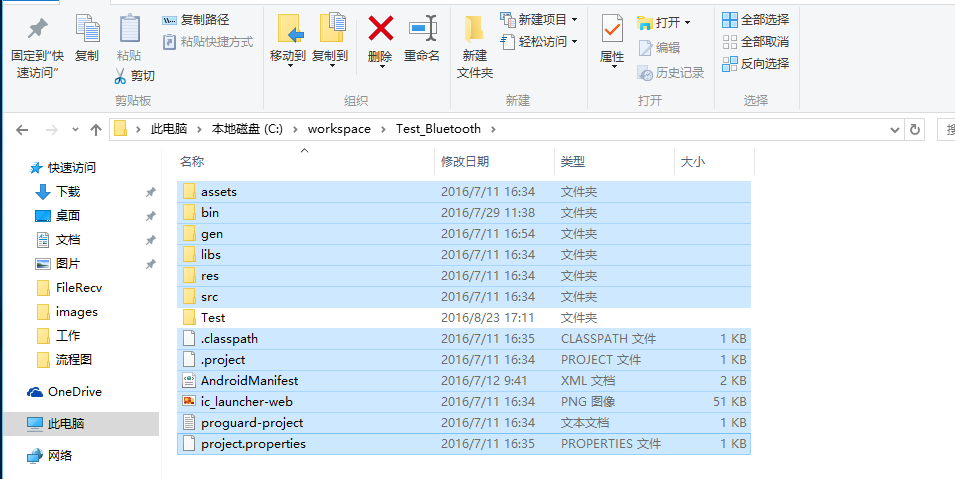


**6.接下来输入如下代码（关键步骤），把github上面的仓库克隆到本地**

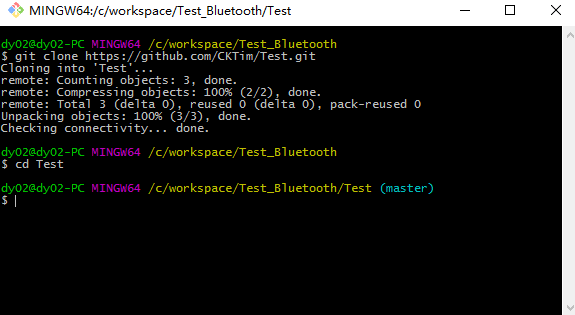
**git clone https://github.com/CKTim/BlueTooth.git（https://github.com/CKTim/BlueTooth.git替换成你之前复制的地址）**



**7.这个步骤以后你的本地项目文件夹下面就会多出个文件夹，该文件夹名即为你github上面的项目名，如图我多出了个Test文件夹，我们把本地项目文件夹下的所有文件（除了新多出的那个文件夹不用），其余都复制到那个新多出的文件夹下，**



**8.接着继续输入命令 cd Test，进入Test文件夹**

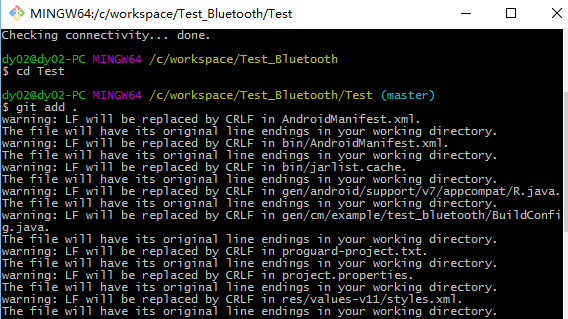


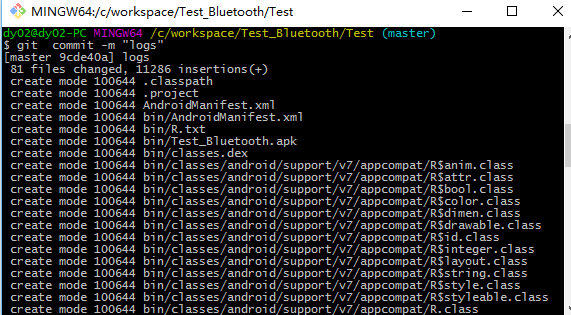
**9.接下来依次输入以下代码即可完成其他剩余操作：**

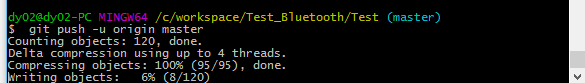
****git add .        （注：别忘记后面的.，此操作是把Test文件夹下面的文件都添加进来）****

****git commit  -m  ”提交信息”  （注：“提交信息”里面换成你需要，如“first commit”）****

****git push -u origin master   （注：此操作目的是把本地仓库push到github上面，此步骤需要你输入帐号和密码）****





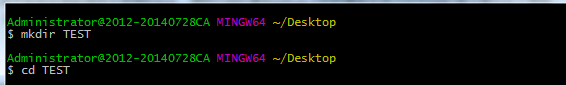


**二、第二种方法：**

第一步：我们需要先创建一个本地的版本库（其实也就是一个文件夹）。

       你可以直接右击新建文件夹，也可以右击打开Git bash命令行窗口通过命令来创建。

       现在我通过命令行在桌面新建一个TEST文件夹（你也可以在其他任何地方创建这个文件夹），并且进入这个文件夹

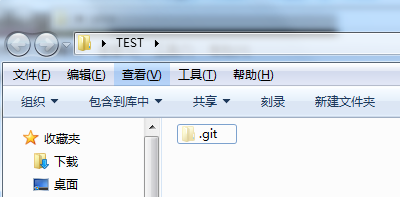




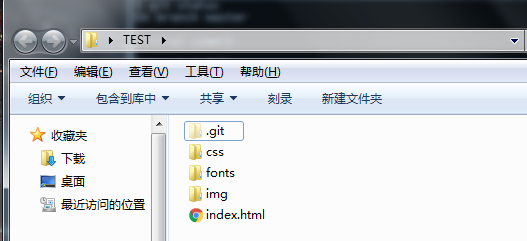
       第二步：通过命令git init把这个文件夹变成Git可管理的仓库

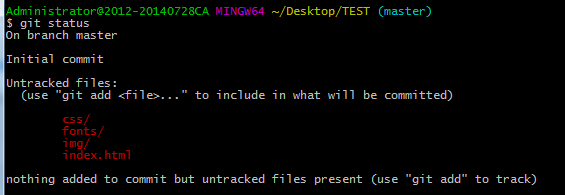
IMG_269

       这时你会发现TEST里面多了个.git文件夹，它是Git用来跟踪和管理版本库的。如果你看不到，是因为它默认是隐藏文件，那你就需要设置一下让隐藏文件可见。

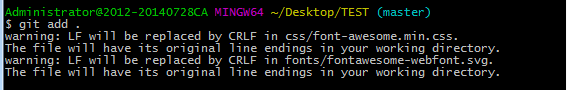


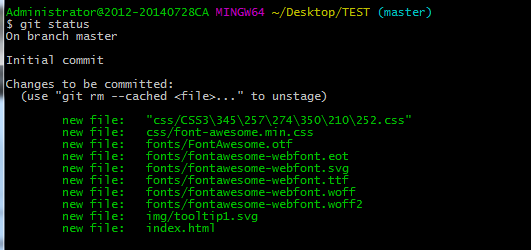
       第三步：这时候你就可以把你的项目粘贴到这个本地Git仓库里面（粘贴后你可以通过git status来查看你当前的状态），然后通过git add把项目添加到仓库（或git add .把该目录下的所有文件添加到仓库，注意点是用空格隔开的）。在这个过程中你其实可以一直使用git status来查看你当前的状态。



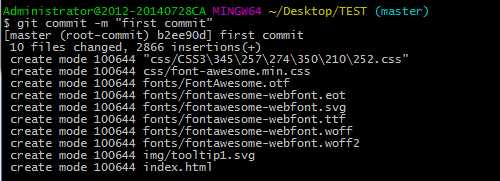


       这里提示你虽然把项目粘贴过来了，但还没有add到Git仓库上，然后我们通过git add .把刚才复制过来的项目全部添加到仓库上。





 第四步：用git commit把项目提交到仓库。



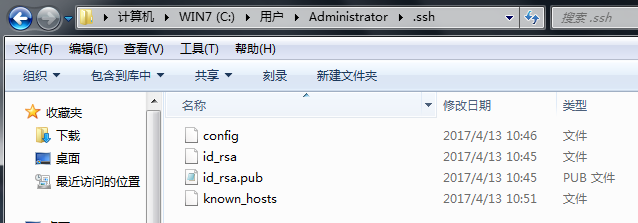
        -m后面引号里面是本次提交的注释内容，这个可以不写，但最好写上，不然会报错，详情自行Google。 好了，我们本地Git仓库这边的工作做完了，下面就到了连接远程仓库（也就是连接Github）

      由于本地Git仓库和Github仓库之间的传输是通过SSH加密的，所以连接时需要设置一下：

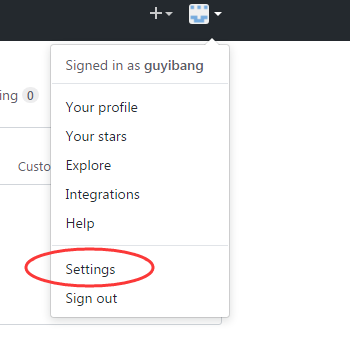
      第五步：创建SSH KEY。先看一下你C盘用户目录下有没有.ssh目录，有的话看下里面有没有id\_rsa和id\_rsa.pub这两个文件，有就跳到下一步，没有就通过下面命令创建

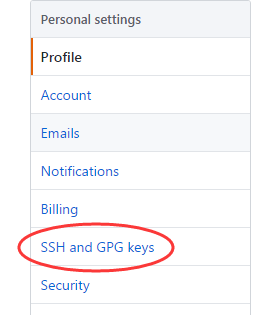
$ ssh-keygen -t rsa -C "youremail@example.com"

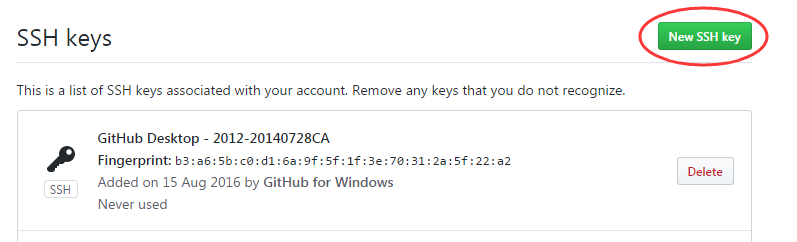
       然后一路回车。这时你就会在用户下的.ssh目录里找到id\_rsa和id\_rsa.pub这两个文件

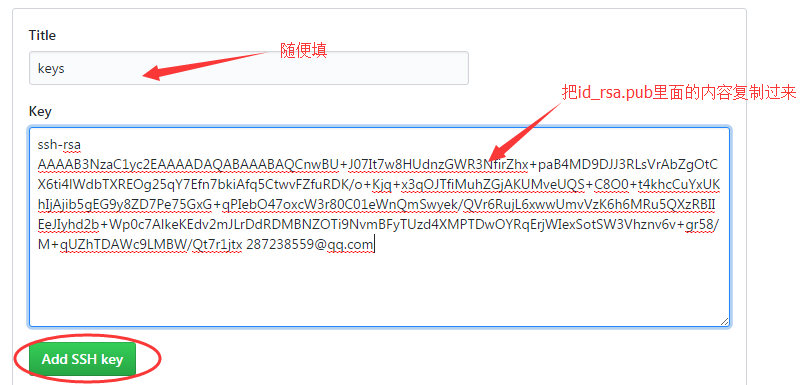


      第六步：登录Github,找到右上角的图标，打开点进里面的Settings，再选中里面的SSH and GPG KEYS，点击右上角的New SSH key，然后Title里面随便填，再把刚才id\_rsa.pub里面的内容复制到Title下面的Key内容框里面，最后点击Add SSH key，这样就完成了SSH Key的加密。具体步骤也可看下面：



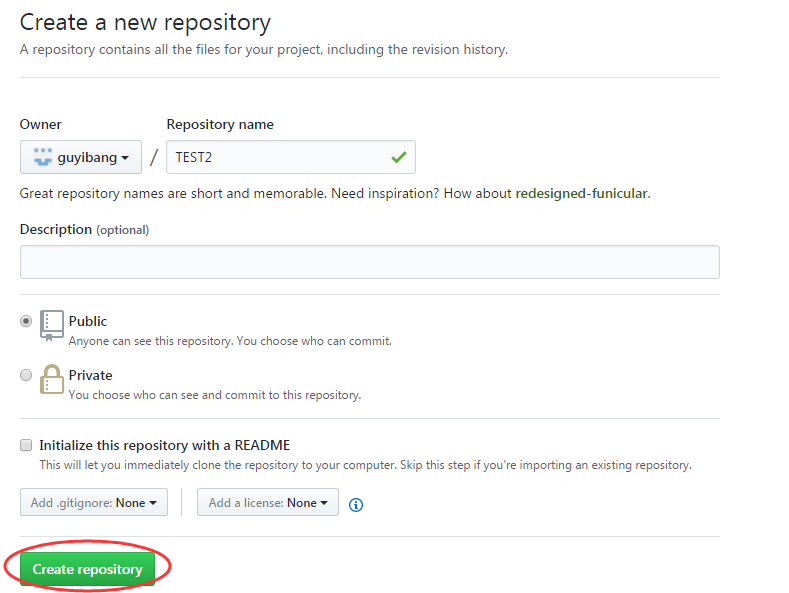






        第七步：在Github上创建一个Git仓库。

     你可以直接点New repository来创建，比如我创建了一个TEST2的仓库（因为我里面已经有了一个test的仓库，所以不能再创建TEST仓库）。



        第八步：在Github上创建好Git仓库之后我们就可以和本地仓库进行关联了，根据创建好的Git仓库页面的提示，可以在本地TEST仓库的命令行输入：

$ git remote add origin https://github.com/guyibang/TEST2.git

IMG_282

        注意origin后面加的是你Github上创建好的仓库的地址。



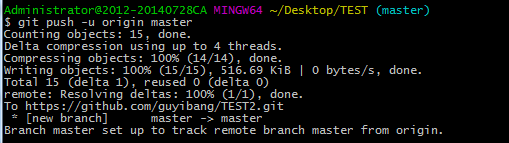
      第九步：关联好之后我们就可以把本地库的所有内容推送到远程仓库（也就是Github）上了，通过：

$ git push -u origin master

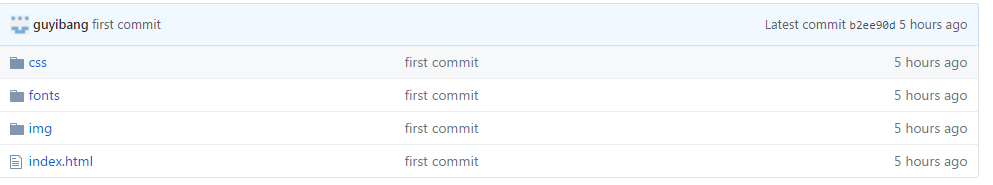
       由于新建的远程仓库是空的，所以要加上-u这个参数，等远程仓库里面有了内容之后，下次再从本地库上传内容的时候只需下面这样就可以了：

$ git push origin master

        上传项目的过程可能需要等一段时间，完成之后是这样的：



        这时候你再重新刷新你的Github页面进入刚才新建的那个仓库里面就会发现项目已经成功上传了：



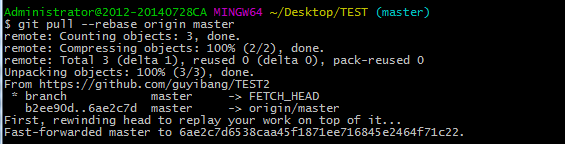
        至此就完成了将本地项目上传到Github的整个过程。

      另外，这里有个坑需要注意一下，就是在上面第七步创建远程仓库的时候，如果你勾选了Initialize this repository with a README（就是创建仓库的时候自动给你创建一个README文件），那么到了第九步你将本地仓库内容推送到远程仓库的时候就会报一个failed to push some refs to https://github.com/guyibang/TEST2.git的错。



      这是由于你新创建的那个仓库里面的README文件不在本地仓库目录中，这时我们可以通过以下命令先将内容合并以下：

$ git pull --rebase origin master



       这时你再push就能成功了。

     总结：其实只需要进行下面几步就能把本地项目上传到Github

     1、在本地创建一个版本库（即文件夹），通过git init把它变成Git仓库；

     2、把项目复制到这个文件夹里面，再通过git add .把项目添加到仓库；

     3、再通过git commit -m “注释内容”把项目提交到仓库；

     4、在Github上设置好SSH密钥后，新建一个远程仓库，通过git remote add origin https://github.com/guyibang/TEST2.git将本地仓库和远程仓库进行关联；

     5、最后通过git push -u origin master把本地仓库的项目推送到远程仓库（也就是Github）上；（若新建远程仓库的时候自动创建了README文件会报错，解决办法看上面）。

## ****三、Git命令 查看、添加、提交、删除、找回，重置修改文件****

git help <command> # 显示command的help

git show *# 显示某次提交的内容 git show $id*

git co *-- <file> # 抛弃工作区修改*

git co . *# 抛弃工作区修改*

git add <file> *# 将工作文件修改提交到本地暂存区*

git add . *# 将所有修改过的工作文件提交暂存区*

git rm <file> *# 从版本库中删除文件*

git rm <file> *--cached # 从版本库中删除文件，但不删除文件*

git reset <file> *# 从暂存区恢复到工作文件*

git reset *-- . # 从暂存区恢复到工作文件*

git reset *--hard # 恢复最近一次提交过的状态，即放弃上次提交后的所有本次修改*

git ci <file> git ci . git ci -a *# 将git add, git rm和git ci等操作都合并在一起做　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　git ci -am "some comments"*

git ci *--amend # 修改最后一次提交记录*

git revert <$id> *# 恢复某次提交的状态，恢复动作本身也创建次提交对象*

git revert HEAD *# 恢复最后一次提交的状态*

查看文件diff

git help <command> # 显示command的help

git show *# 显示某次提交的内容 git show $id*

git co *-- <file> # 抛弃工作区修改*

git co . *# 抛弃工作区修改*

git add <file> *# 将工作文件修改提交到本地暂存区*

git add . *# 将所有修改过的工作文件提交暂存区*

git rm <file> *# 从版本库中删除文件*

git rm <file> *--cached # 从版本库中删除文件，但不删除文件*

git reset <file> *# 从暂存区恢复到工作文件*

git reset *-- . # 从暂存区恢复到工作文件*

git reset *--hard # 恢复最近一次提交过的状态，即放弃上次提交后的所有本次修改*

git ci <file> git ci . git ci -a *# 将git add, git rm和git ci等操作都合并在一起做　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　git ci -am "some comments"*

git ci *--amend # 修改最后一次提交记录*

git revert <$id> *# 恢复某次提交的状态，恢复动作本身也创建次提交对象*

git revert HEAD *# 恢复最后一次提交的状态*

查看提交记录

git log git log <file> *# 查看该文件每次提交记录*

git log -p <file> *# 查看每次详细修改内容的diff*

git log -p -2 *# 查看最近两次详细修改内容的diff*

git log *--stat #查看提交统计信息*

tig

Mac上可以使用tig代替diff和log，brew install tig

Git 本地分支管理  
查看、切换、创建和删除分支

git br -r *# 查看远程分支*

git br <new\_branch> *# 创建新的分支*

git br -v *# 查看各个分支最后提交信息*

git br --merged *# 查看已经被合并到当前分支的分支*

git br --no-merged *# 查看尚未被合并到当前分支的分支*

git co <branch> *# 切换到某个分支*

git co -b <new\_branch> *# 创建新的分支，并且切换过去*

git co -b <new\_branch> <branch> *# 基于branch创建新的new\_branch*

git co $id *# 把某次历史提交记录checkout出来，但无分支信息，切换到其他分支会自动删除*

git co $id -b <new\_branch> *# 把某次历史提交记录checkout出来，创建成一个分支*

git br -d <branch> *# 删除某个分支*

git br -D <branch> *# 强制删除某个分支 (未被合并的分支被删除的时候需要强制)*

分支合并和reba

git merge <branch> # 将branch分支合并到当前分支

git merge origin/master --no-ff # 不要Fast-Foward合并，这样可以生成merge提交

git rebase master <branch> # 将master rebase到branch，相当于： git co <branch> && git rebase master && git co master && git merge <branch>

Git补丁管理(方便在多台机器上开发同步时用)

git merge <branch> # 将branch分支合并到当前分支

git merge origin/master --no-ff # 不要Fast-Foward合并，这样可以生成merge提交

git rebase master <branch> # 将master rebase到branch，相当于： git co <branch> && git rebase master && git co master && git merge <branch>

Git暂存管

git stash *# 暂存*

git stash list *# 列所有stash*

git stash apply *# 恢复暂存的内容*

git stash drop *# 删除暂存区*

Git远程分支管理

git pull *# 抓取远程仓库所有分支更新并合并到本地*

git pull *--no-ff # 抓取远程仓库所有分支更新并合并到本地，不要快进合并*

git fetch origin *# 抓取远程仓库更新*

git merge origin/master *# 将远程主分支合并到本地当前分支*

git co *--track origin/branch # 跟踪某个远程分支创建相应的本地分支*

git co -b <local\_branch> origin/<remote\_branch> *# 基于远程分支创建本地分支，功能同上*

git push # push所有分支

git push origin master *# 将本地主分支推到远程主分支*

git push -u origin master *# 将本地主分支推到远程(如无远程主分支则创建，用于初始化远程仓库)*

git push origin <local\_branch> *# 创建远程分支， origin是远程仓库名*

git push origin <local\_branch>:<remote\_branch> *# 创建远程分支*

git push origin :<remote\_branch> *#先删除本地分支(git br -d <branch>)，然后再push删除远程分支*

Git远程仓库管

git remote -v # 查看远程服务器地址和仓库名称

git remote show origin # 查看远程服务器仓库状态

git remote add origin git@ github:robbin/robbin\_site.git # 添加远程仓库地址

git remote set-url origin git@ github.com:robbin/robbin\_site.git # 设置远程仓库地址(用于修改远程仓库地址) git remote rm <repository> # 删除远程仓库

创建远程仓库

git clone --bare robbin\_site robbin\_site.git # 用带版本的项目创建纯版本仓库

scp -r my\_project.git git@ git.csdn.net:~ # 将纯仓库上传到服务器上

mkdir robbin\_site.git && cd robbin\_site.git && git --bare init # 在服务器创建纯仓库

git remote add origin git@ github.com:robbin/robbin\_site.git # 设置远程仓库地址

git push -u origin master # 客户端首次提交

git push -u origin develop # 首次将本地develop分支提交到远程develop分支，并且track

git remote set-head origin master # 设置远程仓库的HEAD指向master分支

也可以命令设置跟踪远程库和本地库

git branch --set-upstream master origin/master

git branch --set-upstream develop origin/develop

***方法一转自：****[http://www.cnblogs.com/cxk1995/p/5800196.html](http://www.cnblogs.com/cxk1995/p/5800196.html" \t "https://blog.csdn.net/u014135752/article/details/_blank)*

***方法二转自：****[http://blog.csdn.net/zamamiro/article/details/70172900](http://blog.csdn.net/zamamiro/article/details/70172900" \t "https://blog.csdn.net/u014135752/article/details/_blank)*