## GitHub是一个面向[开源](https://baike.baidu.com/item/%E5%BC%80%E6%BA%90/20720669" \t "https://baike.baidu.com/item/github/_blank)及私有[软件](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6/12053" \t "https://baike.baidu.com/item/github/_blank)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。

Git 分布式版本控制软件；

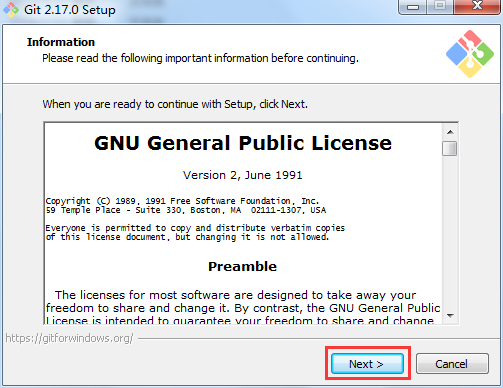
Svn \hg \git

SVN 集中式版本控制软件

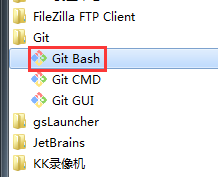
## 下载Windows版Git

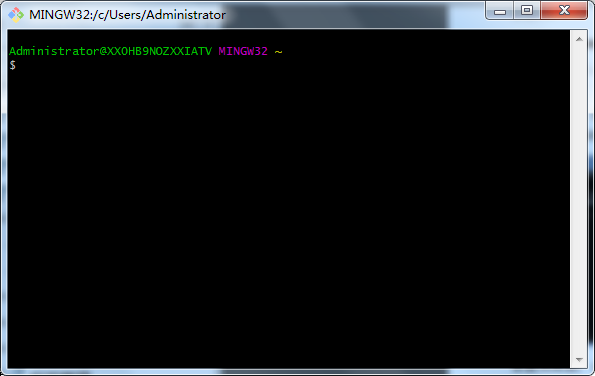


## 双击开始安装

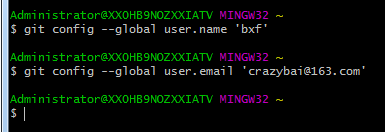


## 打开Git Bash



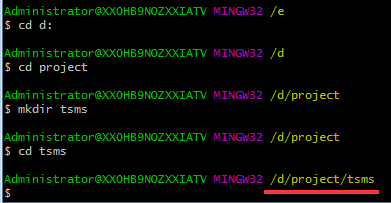


## 创建用户（指定用户名、邮箱）



## 创建版本库

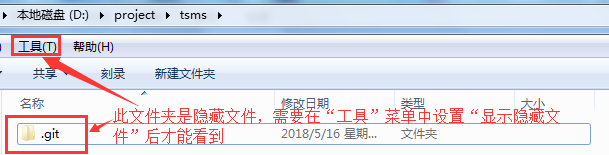
### 创建空目录



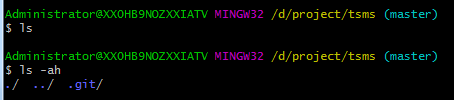


### 使用git init命令将上述目录变成git可管理的仓库





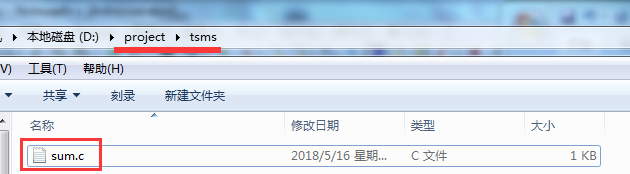
.git目录是隐藏目录，使用ls –ah命令可以看到此目录

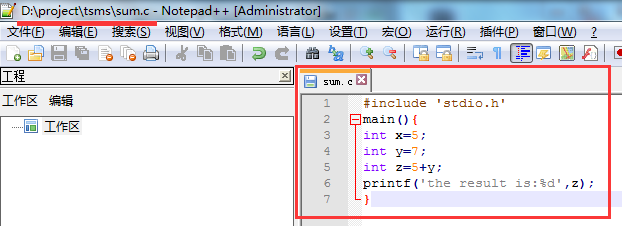


## 将文件添加到版本库

### 在仓库中创建文件

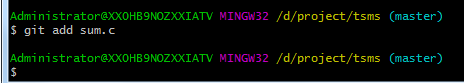
在d:/project/tsms目录中（此目录是Git仓库，在仓库中的文件，才能由Git管理）创建sum.c文件。





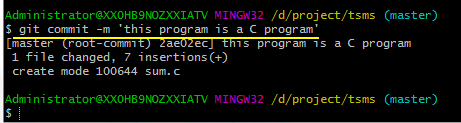
注意，不要使用Windows的记事本创建文件，记事本创建的文件会出现格式或文本混乱的问题，可以使用其他编辑器创建文件，如NotePad++、sublime等编辑器。

### 将文件添加到暂存区



### 提交文件

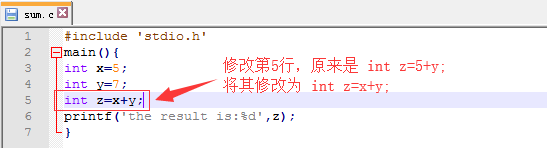
使用git commit命令通知Git，将文件提交到仓库。



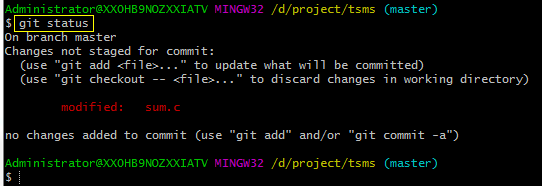
命令执行完毕后，提示1个文件被改动，插入了7行内容

## 修改文件、查看修改前后文件差异、查看仓库状态、提交修改

### 修改sum.c文件内容



### 查看仓库当前状态



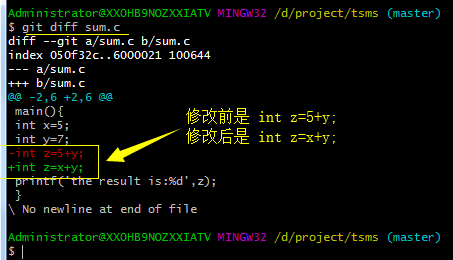
命令执行完毕后，可以看到仓库状态信息：

在master主分支修改了文件，

但修改内容并未提交，

修改的文件是：sum.c

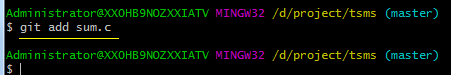
### 查看文件修改前后的差异



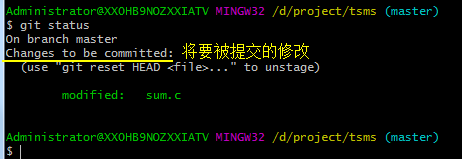
### 提交修改

提交修改 和提交新文件步骤一样，执行git add命令和git commit命令。

1. 将修改的文件添加到仓库

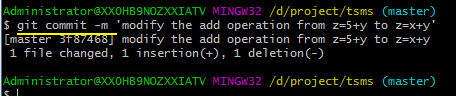


1. 查看添加到仓库后的状态（提交前）

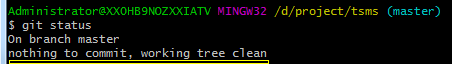


git status命令执行完毕后，提示我们“将要被提交的修改（还未提交）”是sum.c文件。

1. 提交文件



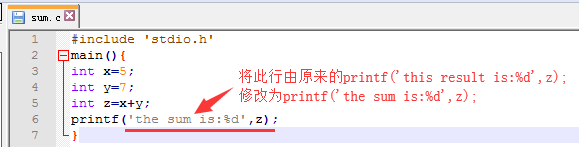
1. 查看提交后的状态



提示，当前没有需要提交的修改，工作目录是干净的。

## 版本回退

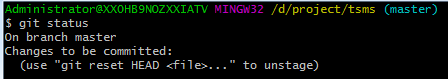
### 再次修改文件并提交



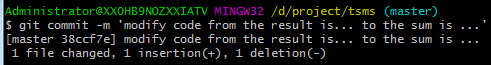
1. 添加



1. 查看仓库状态



1. 提交



1. 查看仓库状态

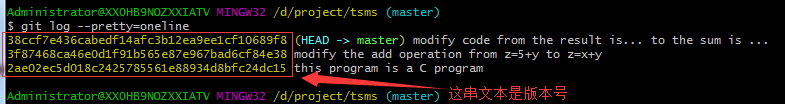


### 查看历史修改记录

使用git log命令查看历史修改记录。

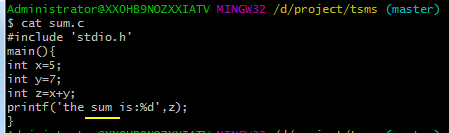


可以在git log命令后加参数 --pretty=oneline 简化输出信息



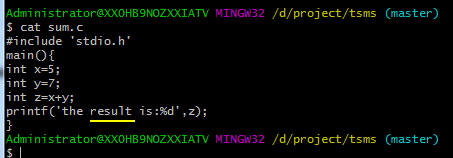
### 回退到上一版本

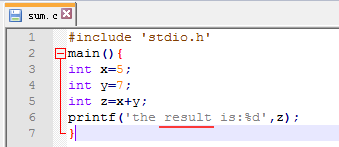
1. 使用cat命令查看当前文件内容



1. 回退到上一版本“from z=x+y”



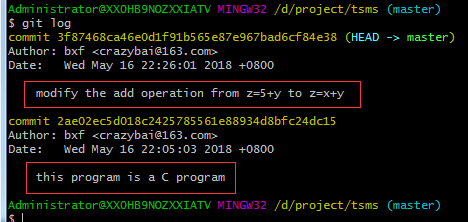




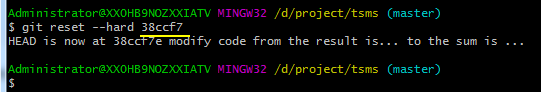
HEAD表示当前版本，上一个版本是HEAD^，再上一个版本是HEAD^^，HEAD~50表示上50个版本。

1. 查看当前版本库状态

可以看到最新的版本库已经不见了

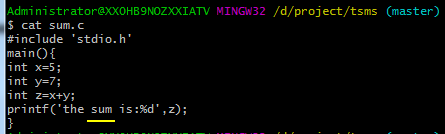


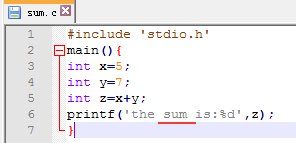
1. 回到未来的某个版本（上面消失不见的之前修改的最近的版本库）



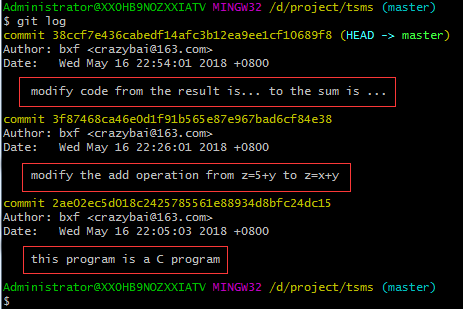
使用版本号（commit id）回到未来的版本，版本号无需写全，只需写前几位即可（能够与其他版本号区别开来即可）。

查看文件内容，可以看到版本已经恢复了



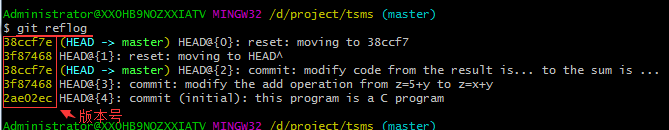


1. 查看历史修改记录



### 使用git reflog命令查看历次的操作命令历史记录

当回退到某个版本后，关闭了Git Bash，此时就看不见了之前的版本号了，我们可以使用git reflog命令查看历次操作记录，在这个操作记录中可以看到版本号，这样就可以使用“git reset –hard 版本号” 命令恢复到未来的版本库了。



## 工作区和暂存区

* 工作区（Working Directory）

工作区就是在电脑里能看到的，我们前面创建的d:/project/tsms就是工作区。

* 版本库（Repository）

工作区有一个隐藏目录 .git，这个不算工作区，是Git的版本库。

* 暂存区

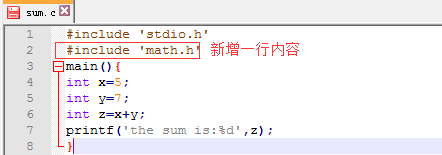
版本库中有一个称为stage的暂存区，还有Git自动创建的第一个分支master，以及指向master的一个指针HEAD。

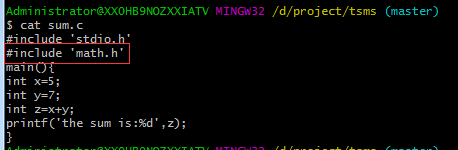
执行 git add 命令添加文件，实际上就是将文件添加到暂存区。

执行 git commit命令提交文件，实际上是把暂存区的所有内容提交到当前分支。

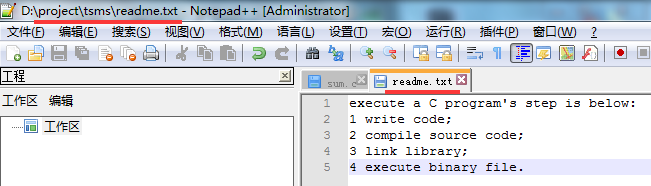
通过练习理解工作区、暂存区的功能和意义

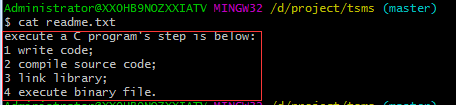
### 修改sum.c文件内容



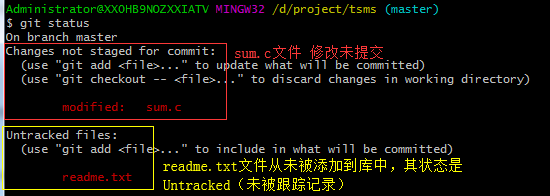


### 在工作区新增一个readme.txt文件





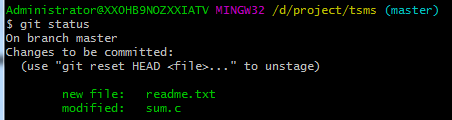
### 查看仓库状态



### 添加sum.c和readme.txt文件到仓库



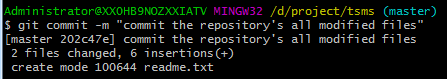
### 查看仓库状态



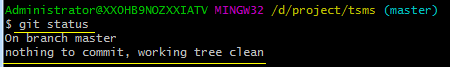
执行git add 命令后，sum.c和readme.txt两个文件已进入暂存区，等待提交。

### 提交修改

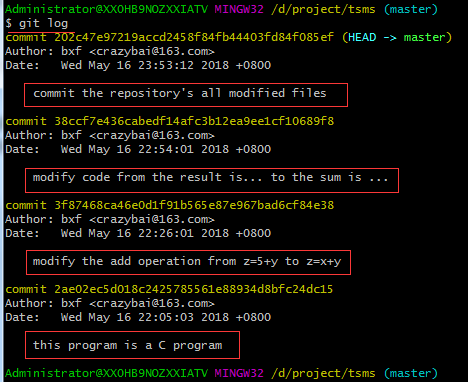
执行git commit命令一次性把暂存区中的所有修改提交到分支。



### 查看仓库状态



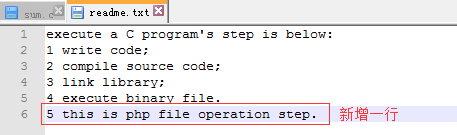
### 查看历史修改记录

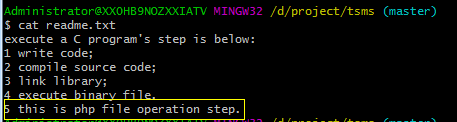


## 撤销修改

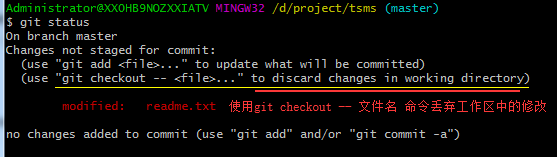
**情况一：修改未被添加到暂存区，需要撤销修改**

### 在readme.txt文件中新增一行





### 查看仓库状态

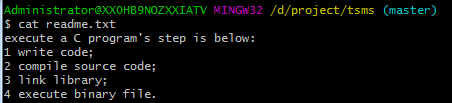


Git提示使用git checkout --file 命令可以丢弃工作区中的修改。

### 执行git checkout 命令，将工作区的修改全部撤销

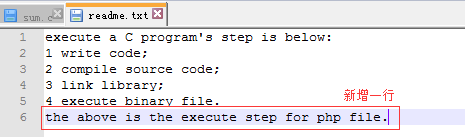


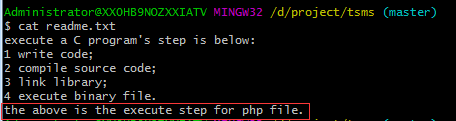
### 再次查看当前状态



**情况二：修改已经添加到暂存区，需要撤销**

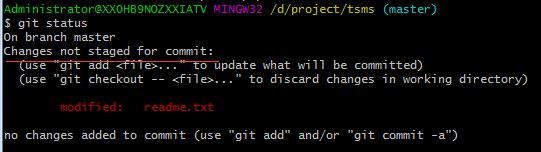
### 在readme.txt文件中新增一行





### 查看状态

修改未添加到暂存区

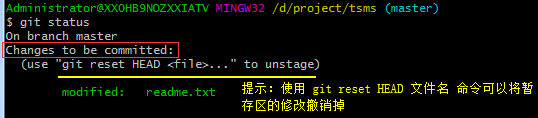


### 将修改添加到暂存区

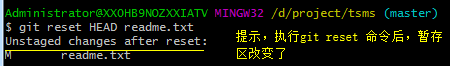


### 查看状态

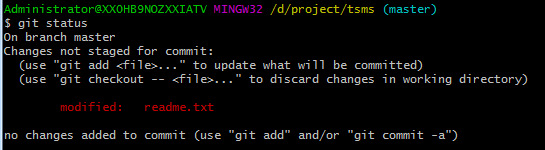
Git提示，可以使用git reset HEAD file 命令，将暂存区的修改撤销掉（将修改回退到工作区）。



### 执行get reset 命令



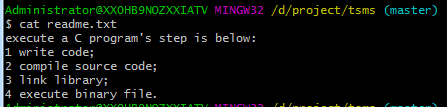
### 查看状态

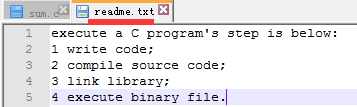


### 执行 git checkout -- readme.txt 命令，丢弃工作区的修改



### 查看文件内容





## 删除文件

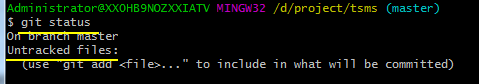
**场景一：**

手动删除工作中的test.txt文件，还未执行git commit命令，此时版本库中还存在test.txt文件，此时可以使用git checkout命令将版本库中的test.txt文件恢复到工作区中。

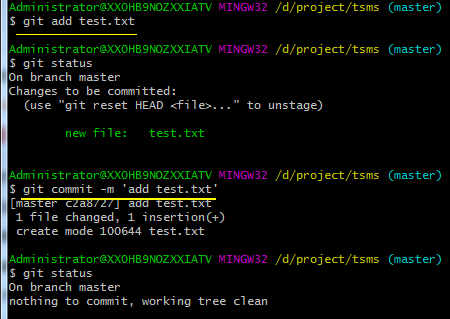
### 新增一个test.txt文件



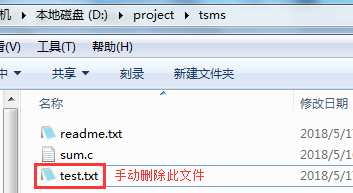
### 查看状态

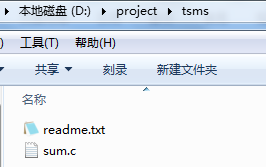


### 添加并提交文件



### 在目录中手动删除test.txt文件





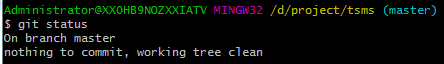
### 查看状态

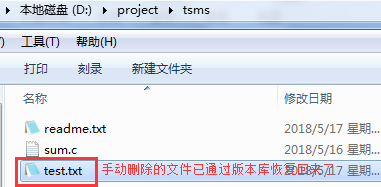


### 使用版本库恢复删除的test.txt文件到工作区



### 查看状态

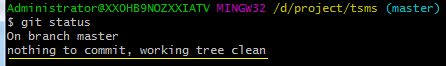




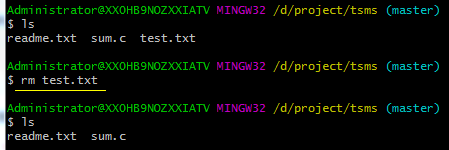
**场景二：**

手动删除工作区中的test.txt文件，此时版本库中还存在test.txt文件，版本库与工作区不一致，此时可执行git rm命令将版本库中的test.txt文件删除。

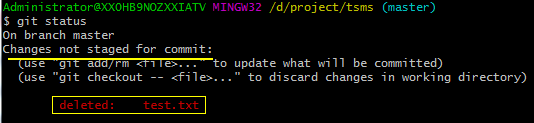
### 查看状态



### 删除工作区中的test.txt文件



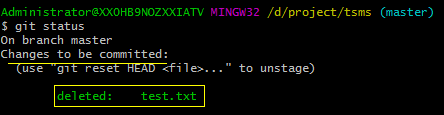
### 查看状态



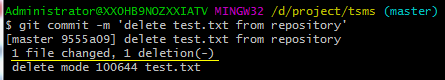
### 删除版本库中的test.txt文件



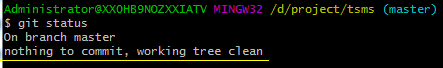
### 查看状态



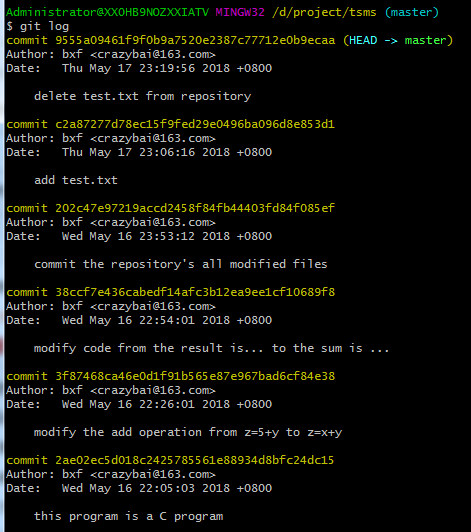
### 提交



### 查看状态

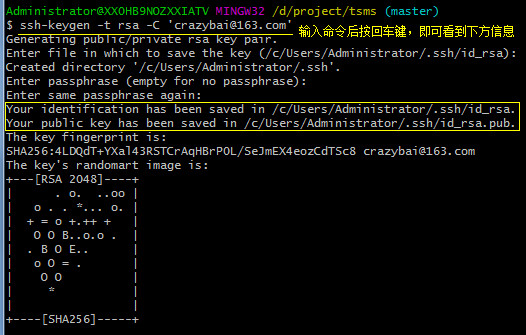


### 查看日志

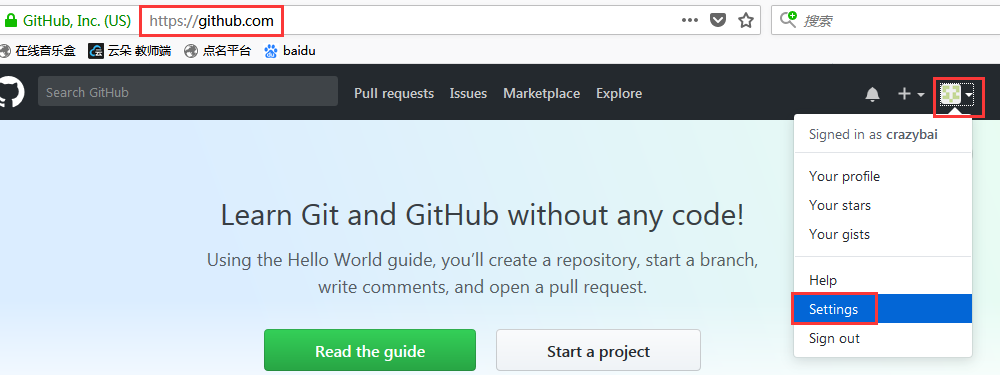


## 远程仓库

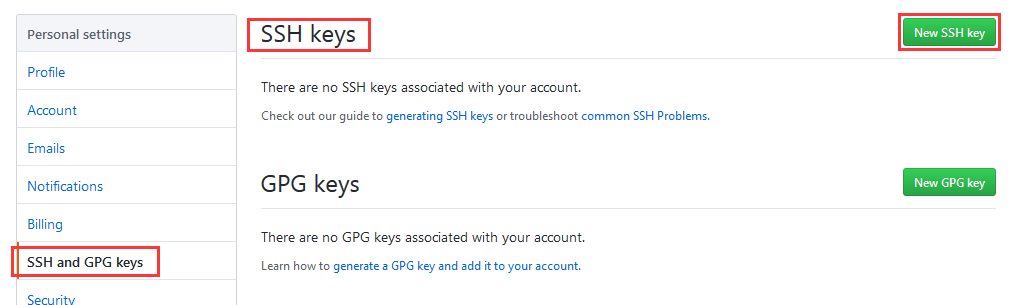
### 创建SSH Key



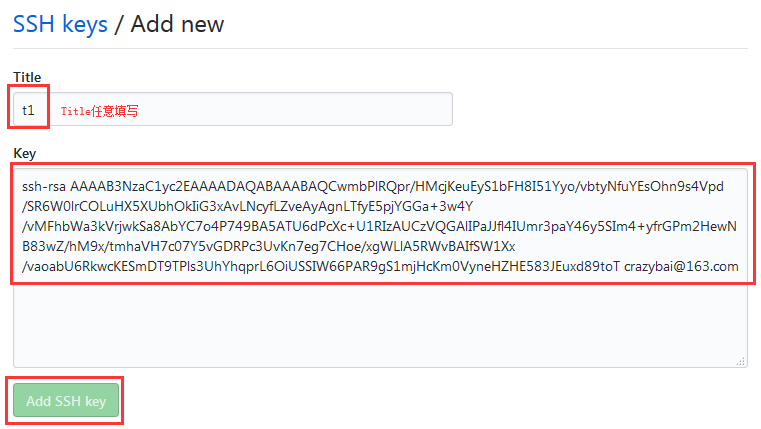
### 登陆GitHub，打开“ settings”页面



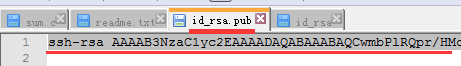
### 选择“SSH and GPG Keys”项，单击右上角“New SSH Key”按钮



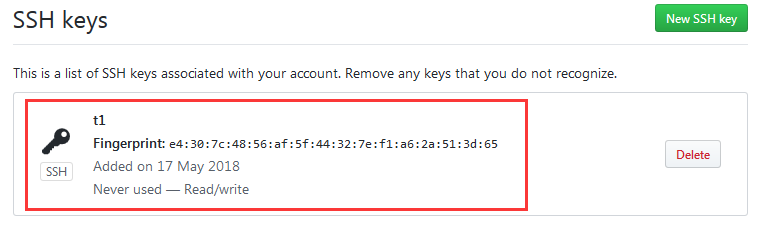
### 输入Titke和Key



**复制id\_rsa.pub文件中内容到上图中的Key文本域中**

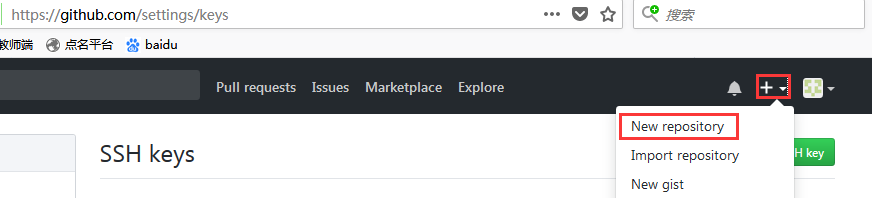


### 单击“Add SSH Key”按钮后完成操作，效果如下

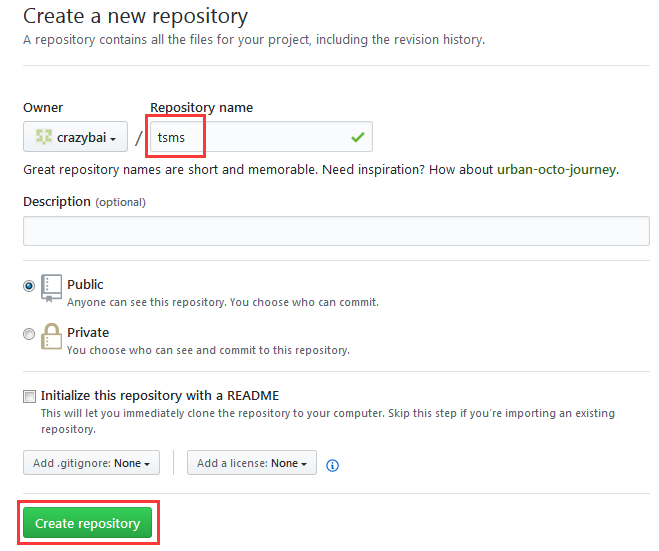


## 添加远程库

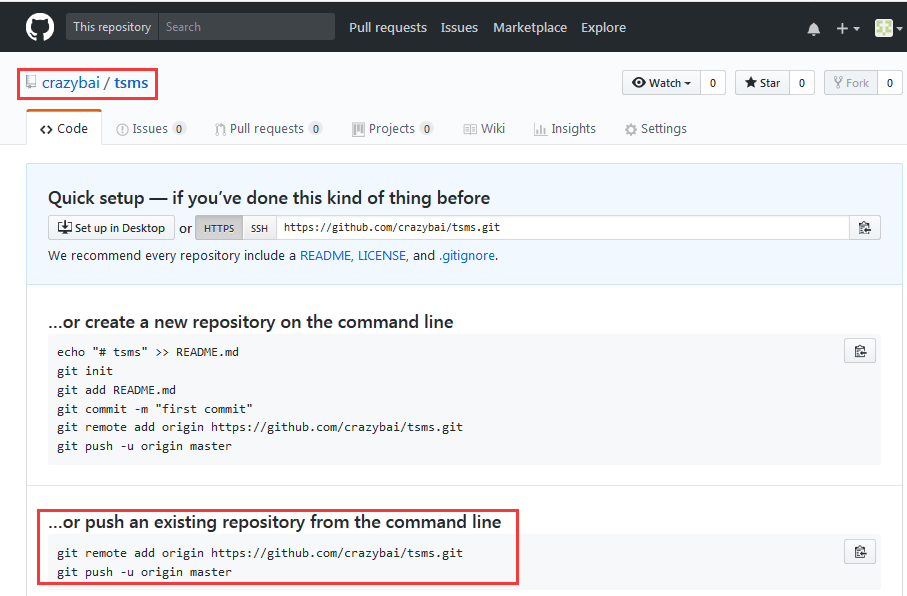
### 在GitHub上创建新的仓库



### 输入仓库名称



### 仓库创建完成



### 本地关联远程库

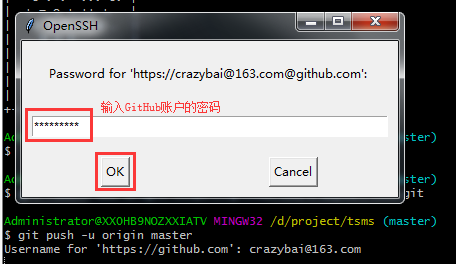


### 将本地仓库内容推送到GitHub上的远程仓库

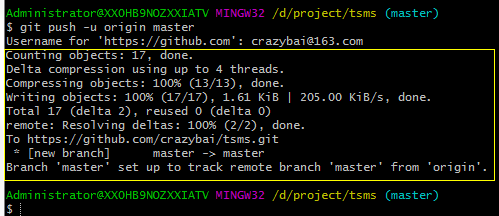
1. 要求输入GitHub站点的用户名



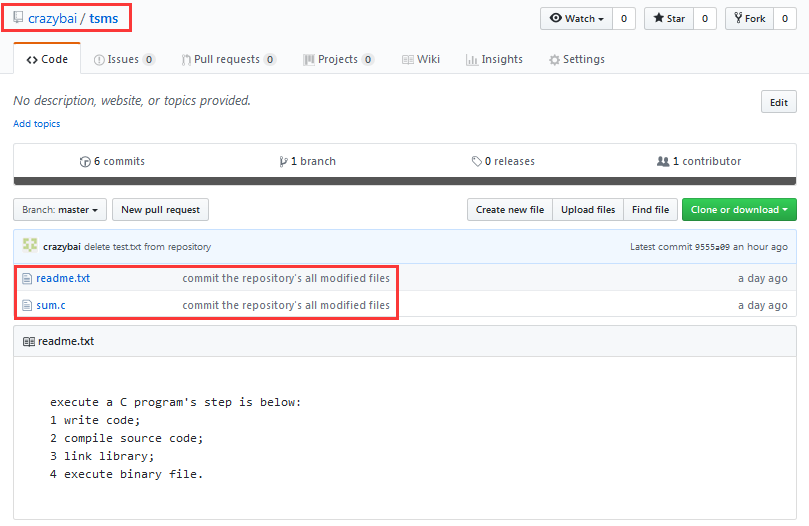
1. 要求输入GitHub站点的用户密码



1. 推送成功



### 查看远程仓库内容



## 分支管理

### 查看分支



### 创建dev分支



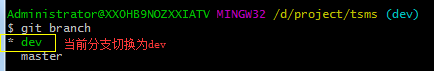
### 查看分支



### 切换到dev分支



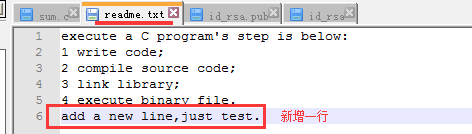
### 查看分支



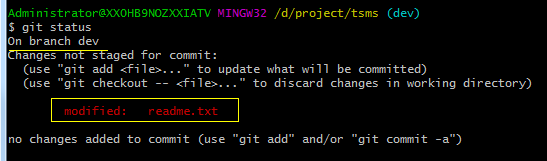
**注意：可以使用“git checkout -b dev”实现创建dev分支，并切换到dev分支（即将当前分支设置为dev）**

### 在dev分支上修改readme.txt文件并提交

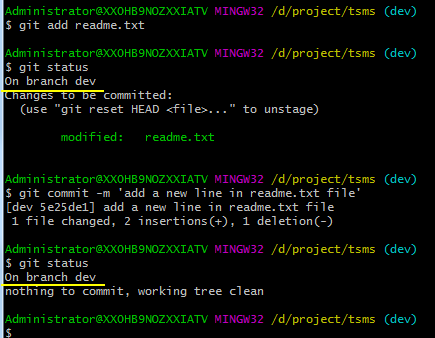
1. 修改readme.txt文件，新增一行内容



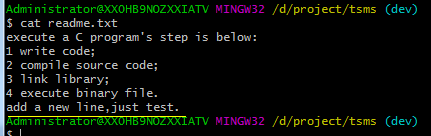
1. 查看状态



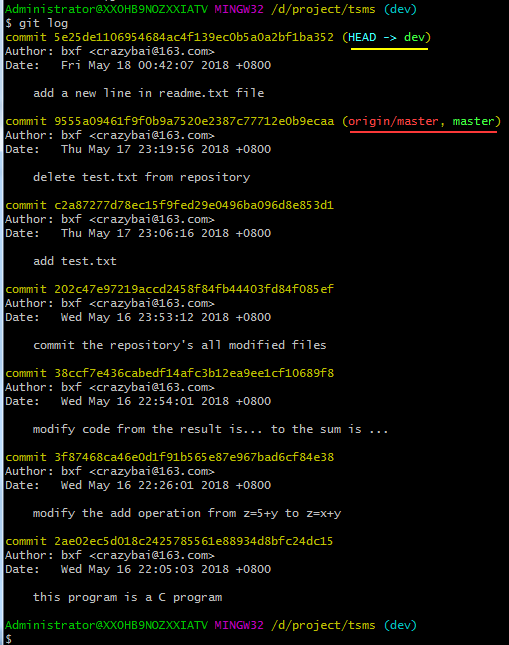
1. 添加并提交



1. 查看文件内容

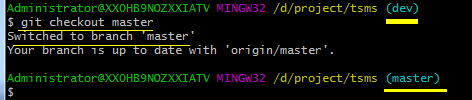


1. 查看历史记录

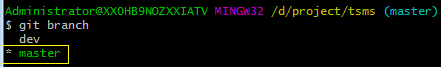


### 切换回master分支

### 切换分支

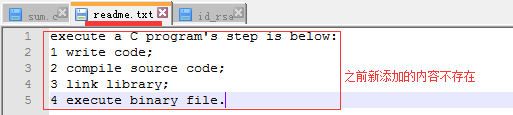


### 查看分支

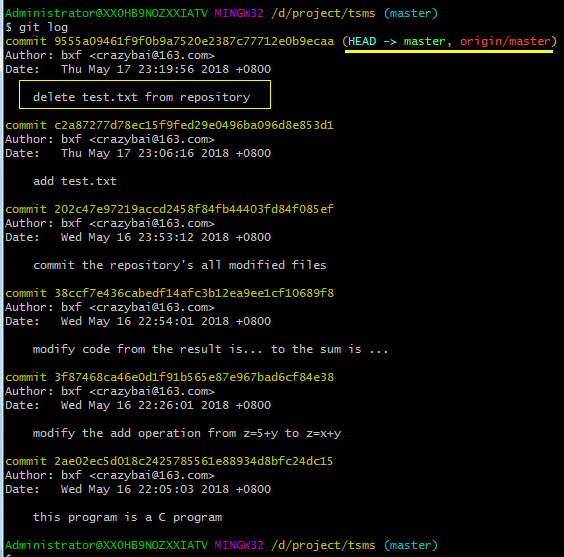


### 查看文件内容



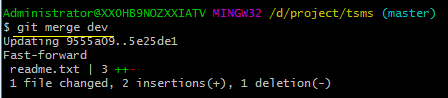


1. 查看历史记录

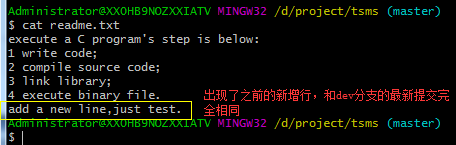


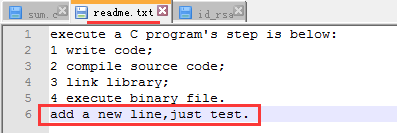
### 将dev分支合并到master分支上

1. 合并分支

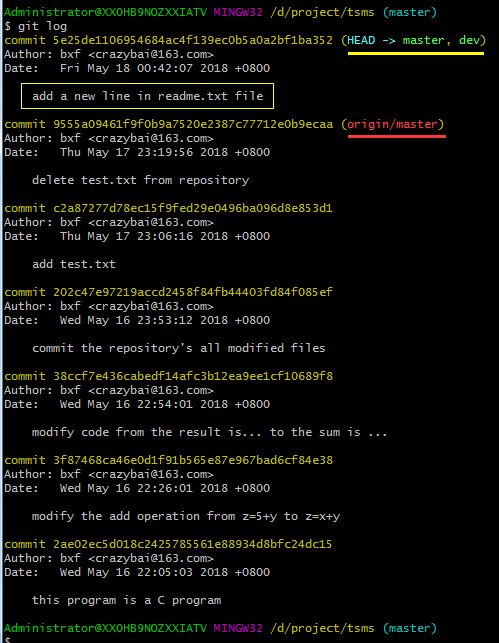


1. 查看内容





1. 查看历史记录



### 删除分支

1. 将dev分支合并到master分支后，就可以删除dev分支了



1. 查看分支

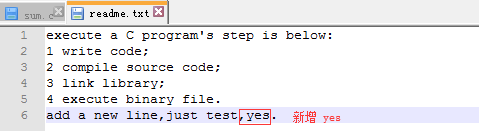


## 解决冲突

### 创建并切换到新分支feature1



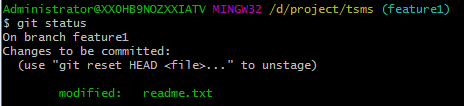
### 修改readme.txt文件内容

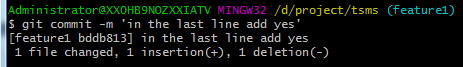


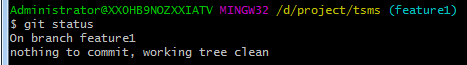


### 添加并提交



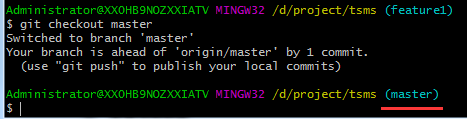




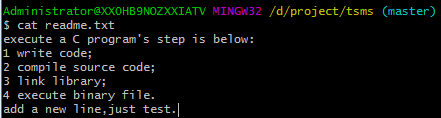


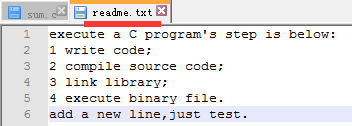
### 切换到master分支

切换

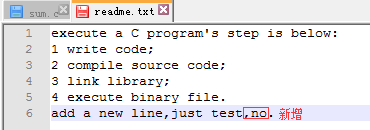


查看文件内容



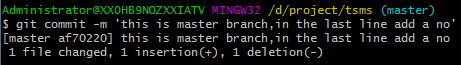


### 修改readme.txt文件内容

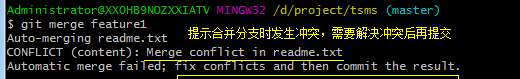


### 添加并提交



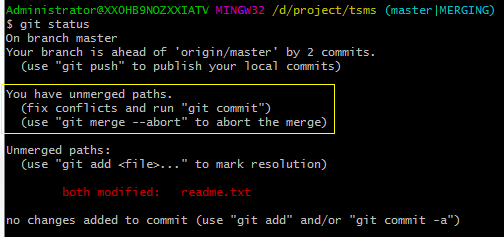


### 合并feature1分支到master分支



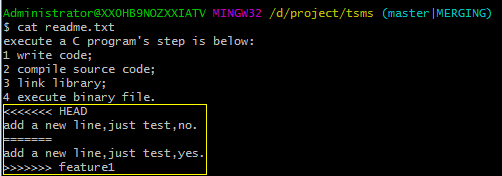
### 查看状态

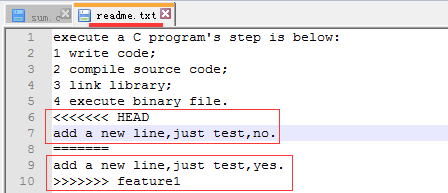
提示有冲突存在，需要处理



### 查看文件内容

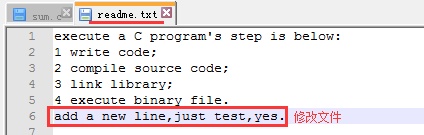
Git使用“=======”符号标记出不同分支的内容





### 解决冲突

1. 手动修改readme.txt文件



1. 添加文件



1. 查看状态



1. 提交



## 多人协作

### 查看远程仓库信息

简要信息

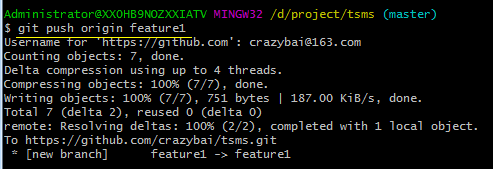


详细信息

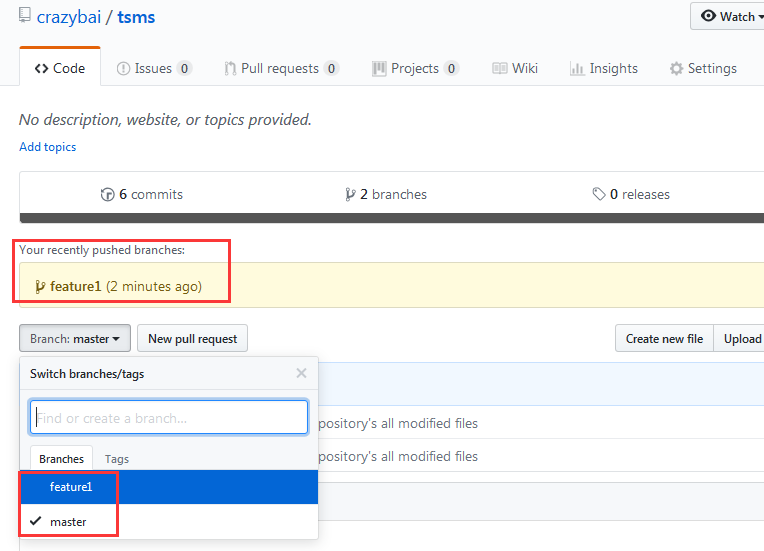


### 向远程仓库推送分支

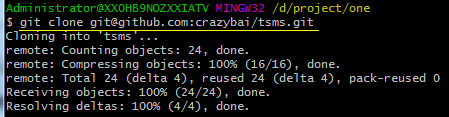
推送



在GitHub上查看分支



### 抓取分支



### 创建选择orign的feature1分支到本地

Clone远程库后，默认情况下，只能看到本地的master分支



创建远程origin的feature1分支到本地

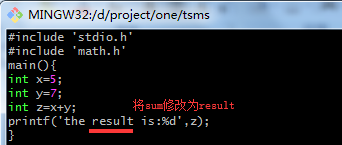


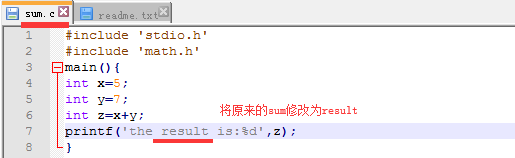
本地feature1分支创建成功



### 修改feature1分支上的内容

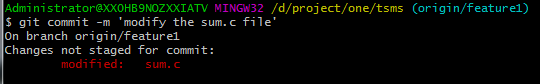






### 将dev分支提交、推送到远程库

提交



推送

Git push

拉下来

Git pull

项目git操作流程：

在自己的分支上

git add . // 添加文件到暂存区

git commit -m’备注’ //暂存区文件提交到自己的分支

git checkout master //切换到master 分支

git merge 自己的分支 //合并分支（有冲突解决冲突）

git pull //拉取

git push //推送

git checkout 自己的分支 //切换到自己的分支

git merge master //合并master分支内容