目录

[Git 2](#_Toc31379863)

[一 Git本地库初始化 2](#_Toc31379864)

[1 创建项目文件夹（我这里为/d/Desktop/test） 2](#_Toc31379865)

[2 在项目文件夹下打开git bash 2](#_Toc31379866)

[3 输入git init初始化本地库 2](#_Toc31379867)

[二 设置签名 2](#_Toc31379868)

[1 签名 2](#_Toc31379869)

[2 签名级别： 2](#_Toc31379870)

[3 命令 2](#_Toc31379871)

[三 Git基本操作 3](#_Toc31379872)

[1 查看工作区状态 3](#_Toc31379873)

[2 将工作区的文件添加到暂存区 3](#_Toc31379874)

[3 将暂存区的文件提交到本地库 4](#_Toc31379875)

[4 查看不同的版本 5](#_Toc31379876)

[5 在不同版本间切换 6](#_Toc31379877)

[6 删除文件和找回 9](#_Toc31379878)

[7 文件比较（文本文档） 11](#_Toc31379879)

[四 git分支 12](#_Toc31379880)

[1 创建分支 12](#_Toc31379881)

[2 查看分支 12](#_Toc31379882)

[3 切换分支 12](#_Toc31379883)

[4 分支合并 12](#_Toc31379884)

[5 分支合并冲突及解决 13](#_Toc31379885)

# Git

## Git本地库初始化

### 创建项目文件夹（我这里为/d/Desktop/test）

### 在项目文件夹下打开git bash

### 输入git init初始化本地库

命令执行完后会产生.git目录,有了这个.git目录就能对这个项目文件夹进行操作了,注意里面的内容不要动

## 设置签名

### 签名

包括用户名和邮箱地址,目的是区分不同的开发人员,可随意设置

### 签名级别：

项目级别、用户级别，项目级别仅当前项目可用，用户级别在本电脑用户下的所有项目均可使用，二者都有时，项目级别优先级更高，二者必须有一个

### 命令

项目级别

**git config user.name li**

**git config user.email** [**frhngiv@qq.com**](mailto:frhngiv@qq.com)

此签名信息保存在该项目文件夹下.git目录下的config文件中。

用户级别

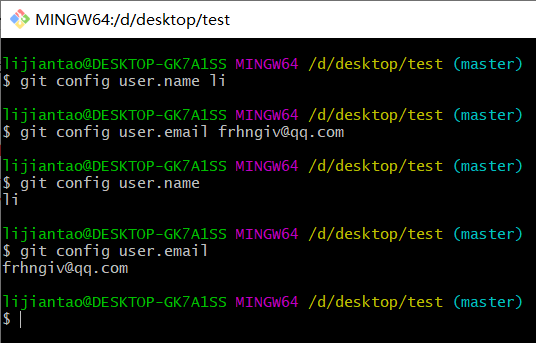
**git config --global user.name li**

**git config --global user.email** [**frhngiv@qq.com**](mailto:frhngiv@qq.com)

此签名信息保存在电脑用户目录下的.gitconfig文件中

我的这个目录是C:\Users\lijiantao。

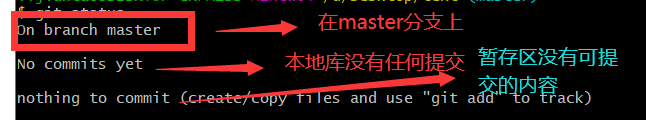
若使用该命令时未加用户名或邮箱可查看当前项目所使用的用户名和邮箱。



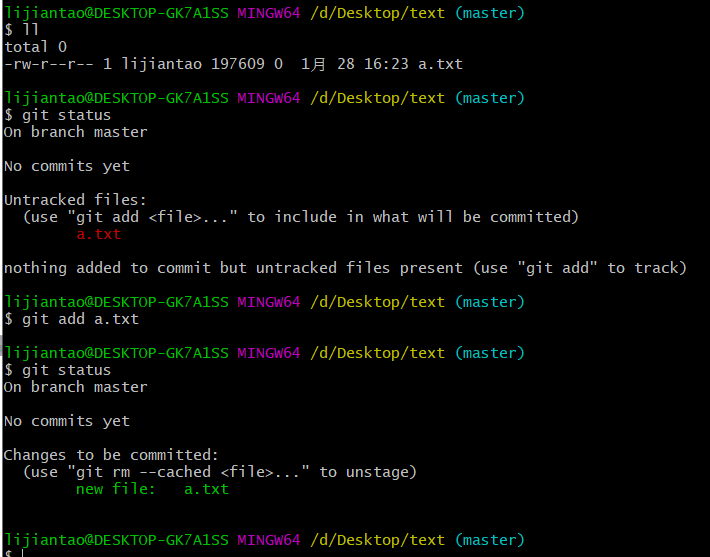
## Git基本操作

### 查看工作区状态

git status



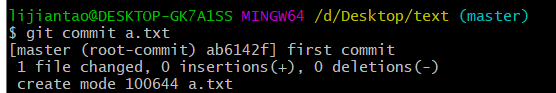
### 将工作区的文件添加到暂存区

git add <file>

使用git rm --cached <file>可将提交到暂存区的文件撤回到暂存区

### 将暂存区的文件提交到本地库

#### git commit <file>



提交到本地库时，会跳转到Vim编辑器中输入提交信息，按i开始输入，输入完后按Esc键，接着输入：wq，再按回车键即可完成提交。

#### git commit –m “commit message” <file>

这种方式提交可不进入Vim编辑器

提交完成后代码的状态



### 查看不同的版本

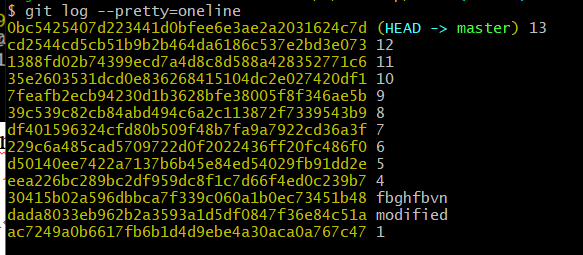
#### git log

查看完整的版本日志，按空格键翻页，b键上一页，q键退出



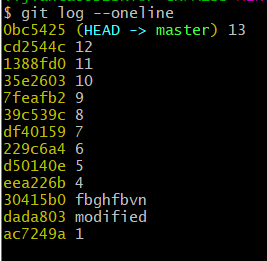
#### git log --pretty=oneline

将日志显示为一行，含有完整的hash值，提交信息，HEAD指针位置



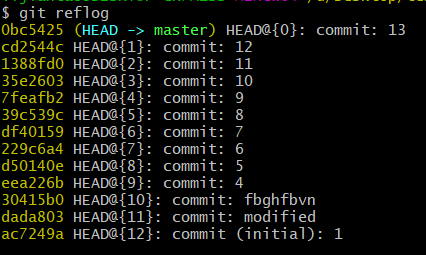
#### git log –oneline

这里会显示较短的hash值



#### git reflog

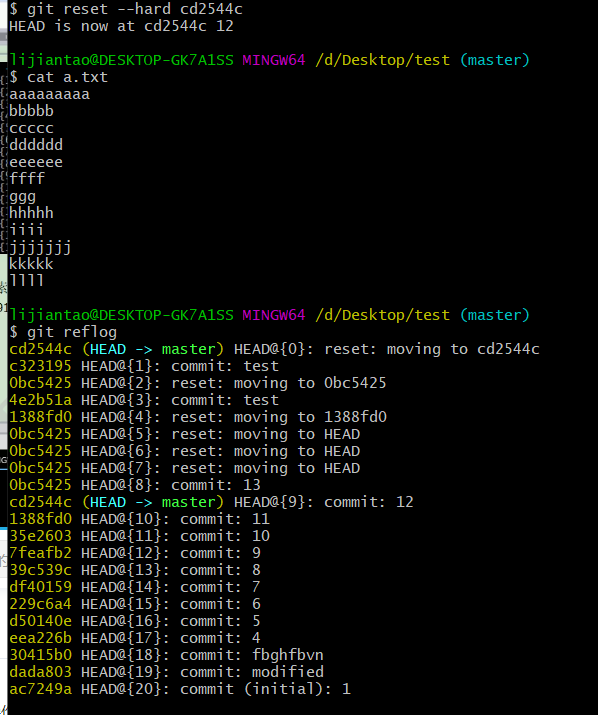
这里大括号中显示的数字是指将HEAD指针移动到该版本所需的次数



### 在不同版本间切换

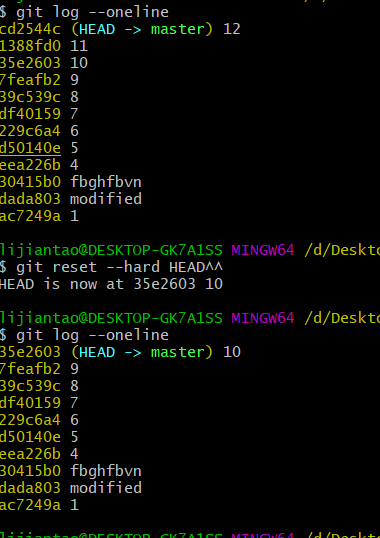
#### 基于索引值的操作

git reset --hard cd2544c，索引值就是较短的hash值



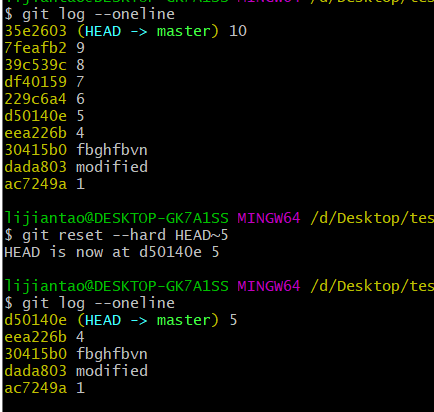
#### 基于^符号（只能后退）

git reset --hard HEAD^^（几个^符号后退几个版本）



#### 基于~符号（只能后退）

git reset --hard HEAD~5（数字代表后退的版本数）



#### git reset参数的比较

--soft仅移动本地库的指针

--mixed移动本地库的指针并重置暂存区

--hard移动本地库指针重置暂存区和工作区

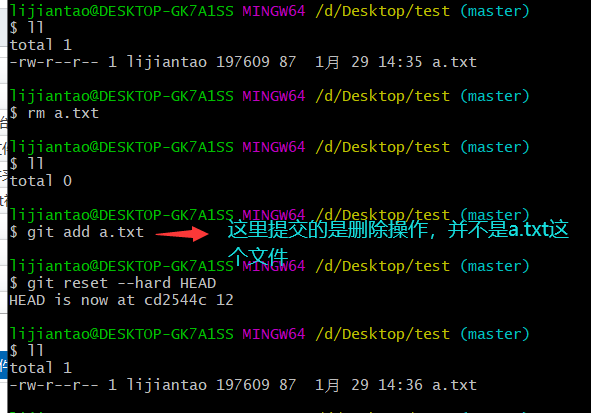
### 删除文件和找回

删除可通过rm <file>或直接选中文件单击右键选择删除

#### 删除未做任何提交操作

请使用专业的文件恢复软件

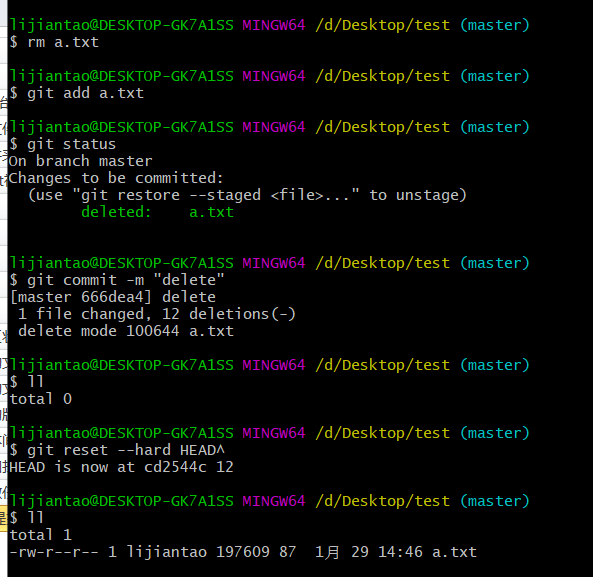
#### 删除操作只提交到暂存区



git reset --hard HEAD

这个命令相当于刷新了一下暂存区和工作区，没有保存到本地库的操作会被删除

#### 删除操作提交到本地库

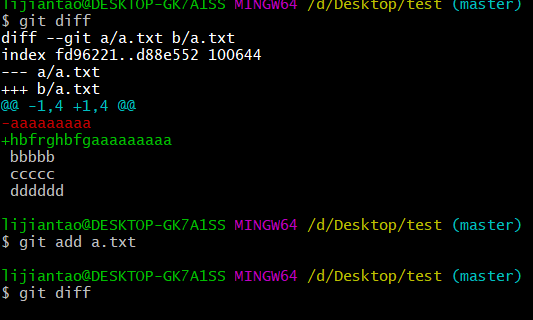


git reset --hard HEAD^

使用该命令回到以前未删除文件的版本

### 文件比较（文本文档）

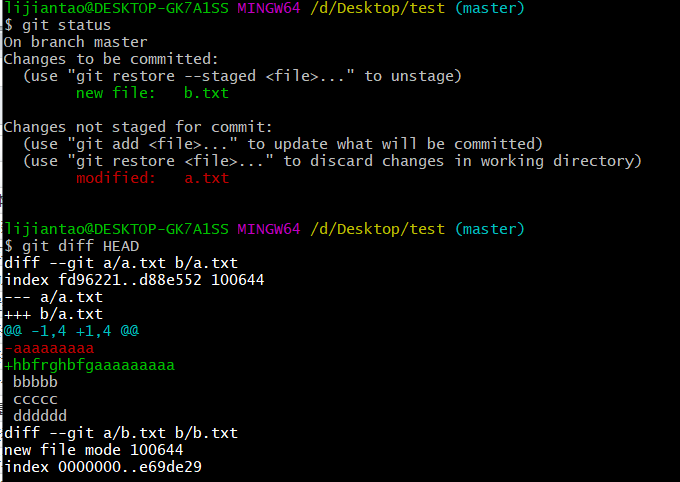
#### 工作区与暂存区文件比较



git diff <file> 未加文件名则对比暂存区和工作区的所有文件

#### 本地库与工作区和暂存区的文件比较

git diff HEAD <file>未加文件名比较所有文件

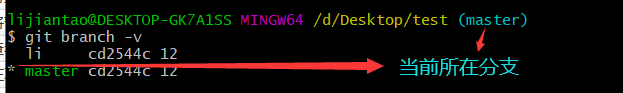


## Git分支

### 创建分支

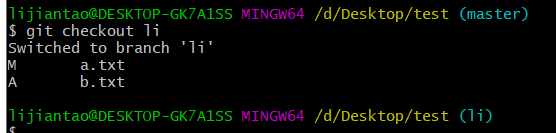
git branch 分支名

### 查看分支



### 切换分支

git checkout 分支名



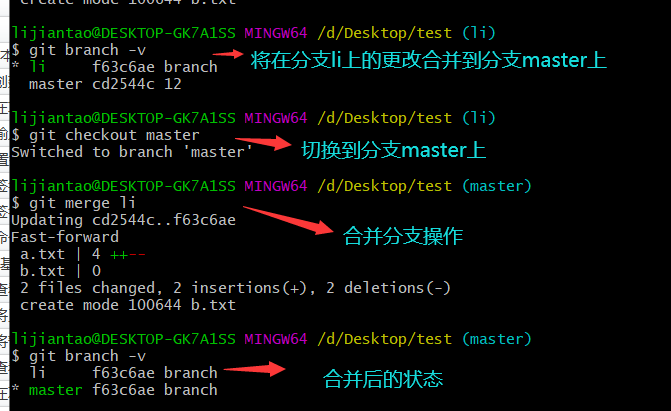
### 分支合并

#### 目的

把一个分支上做的修改合并到另一个分支上

#### 方法

* 切换到接受合并的分支上
* 执行git merge 被合并的分支



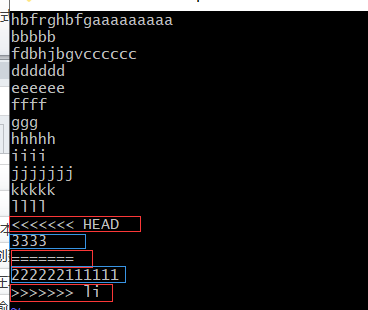
### 分支合并冲突及解决

#### 原因

当不同分支对同一个文件的同一个位置进行更改,再合并时,git无法决定使用哪个更改,这时需要开发人员解决冲突

#### 方法

* 打开冲突的文件并编辑



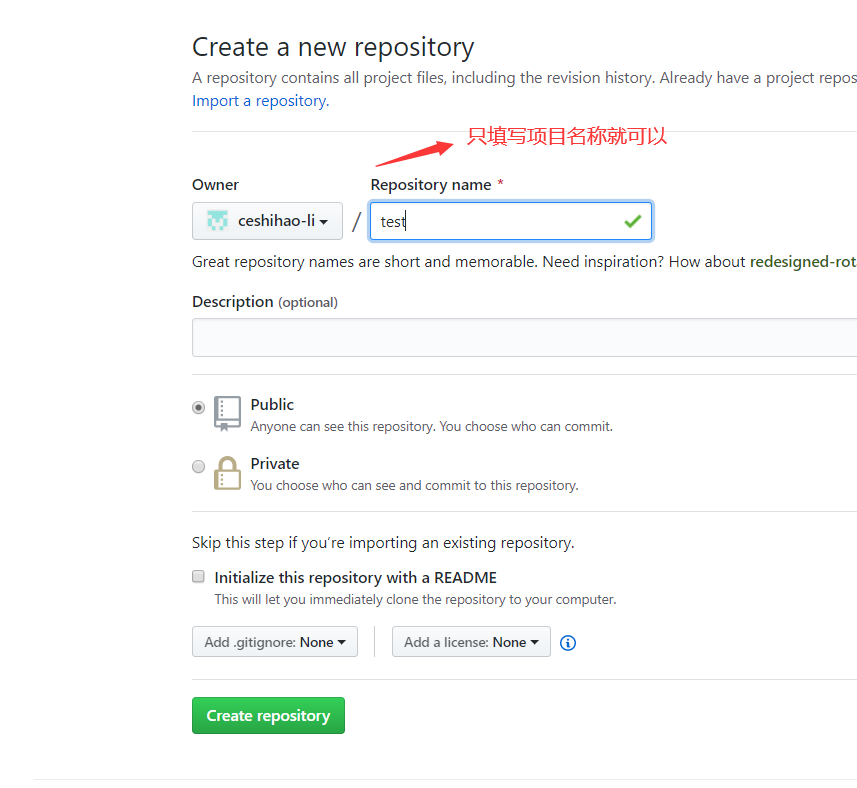
* 红色部分是提示冲突的位置,直接删除,蓝色部分是不同人员做的修改有开发人员决定如何处理
* 使用git add <file>将修改好的文件上传到暂存区
* 使用git commit -m "commit message"将修改上传到本地库,合并即完成



## 远程库的使用及操作

### 注册github账号，建立远程仓库

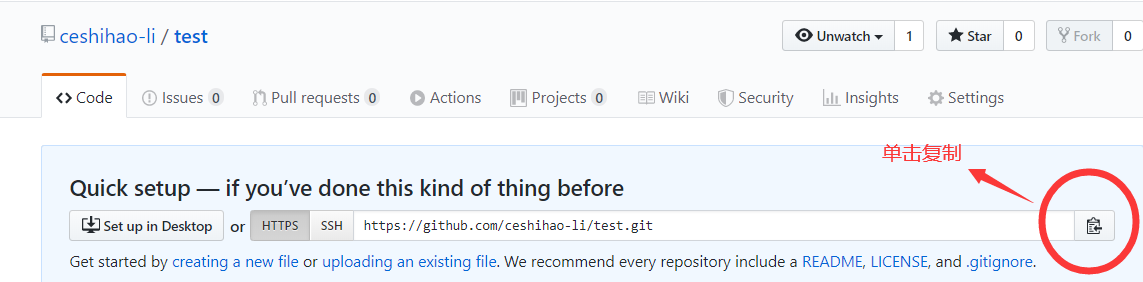
<https://github.com>



### 在本地库中创建远程库别名

#### 获取远程库地址

打开项目，复制远程库地址



#### 添加远程库地址别名

添加：git remote add 别名 远程库地址

查看：git remote –v

删除：git remote remove 别名



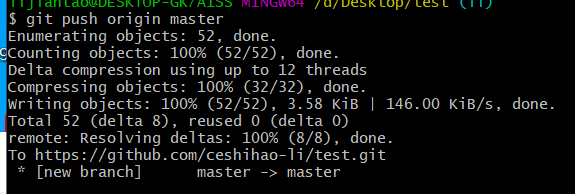
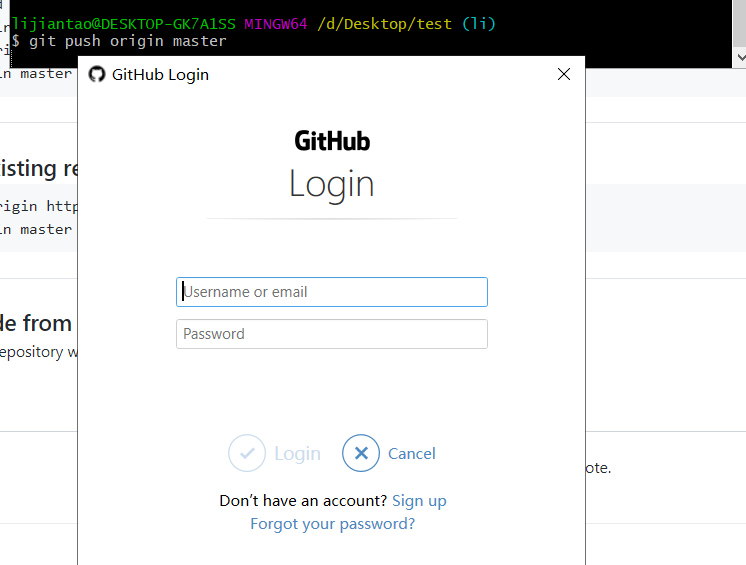
### 本地库向远程库推送

#### 命令

git push 远程库地址 推送的分支名

#### 填写用户名和密码

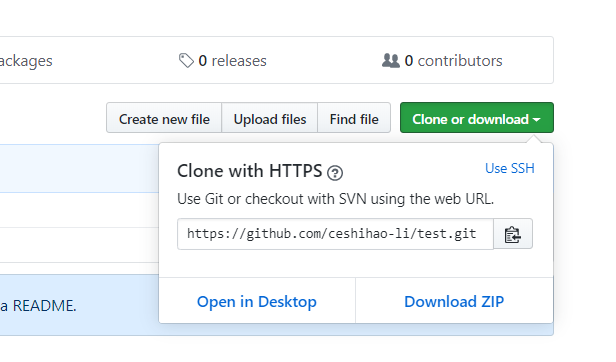
首次推送需要登录github账号，并且会收到一封邮件



## 团队协作与冲突

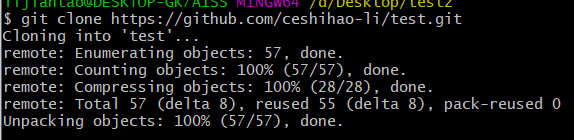
### 加入团队

#### 复制远程库地址（开发成员）

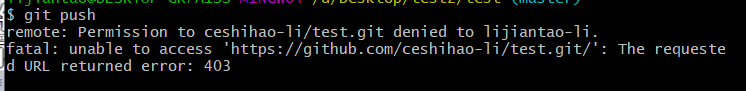


#### 把远程项目克隆到本地

* 在项目文件下打开git bash
* 执行git clone 远程库地址

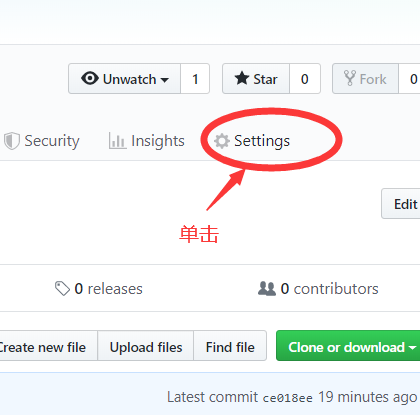


* 克隆效果
  + 初始化文件夹
  + 远程库的所有文件复制到本地
  + 远程仓库别名创建完成
  + 无法推送到远程库

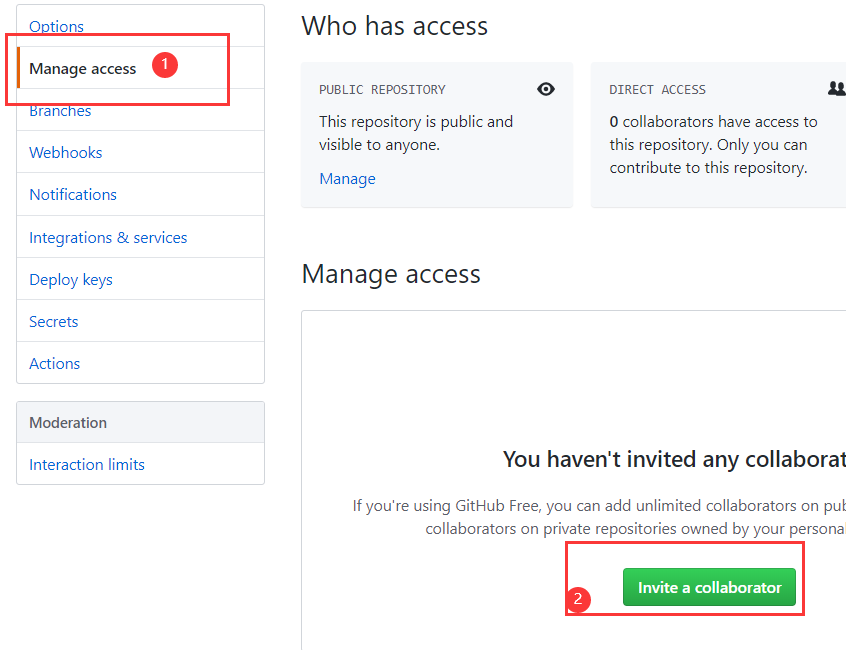


#### 远程库所有者邀请开发人员进入项目

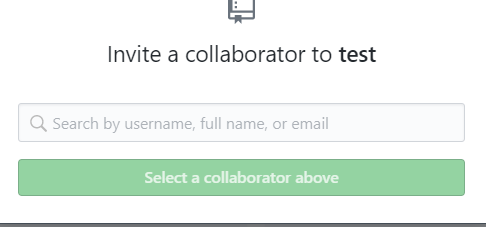
* 进入远程库点击settings

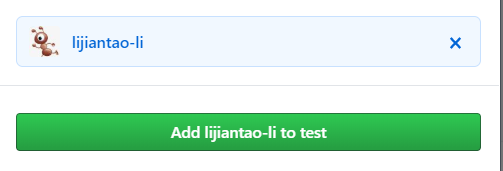


* 点击第二项，找到invite a collaborator

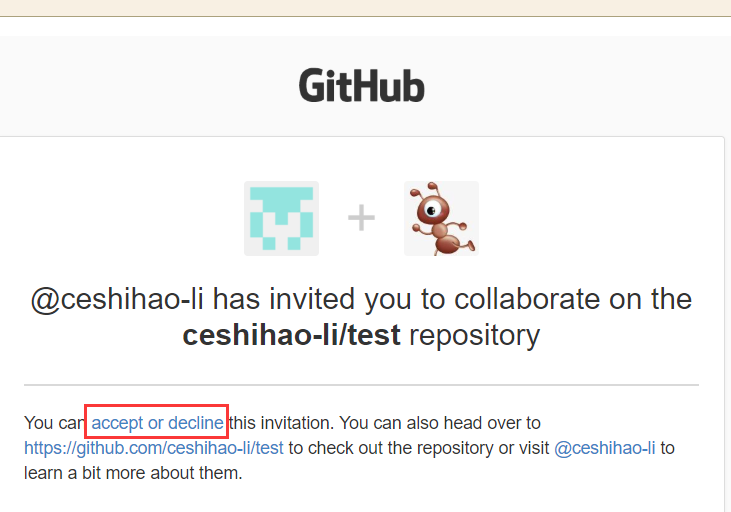


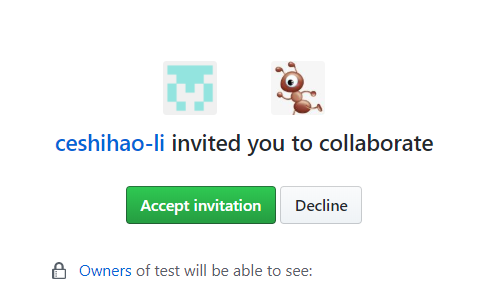
* 输入待邀请人员github账号或用户名，查找完成后点击add

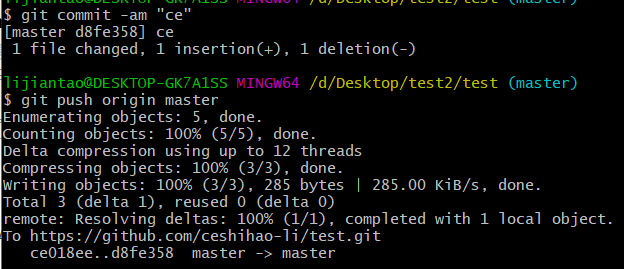




* 开发人员会收到一封邮件点击下图查看详情



* 点击accept invitation，然后就可以push了



### 团队合作操作

#### push操作

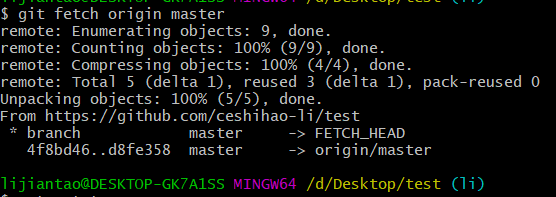
将本地的修改推送到远程库

git push 远程库别名 本地分支 远程分支

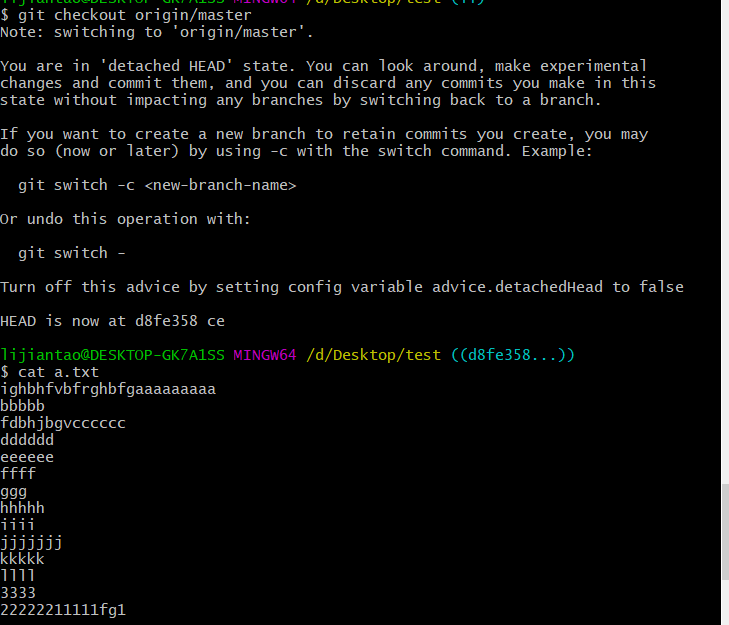
本命令中汉字部分可依次省略

#### fetch,merge操作

* git fetch 远程库别名 远程分支
  + 作用将远程的修改下载到本地并不对本地文件进行修改

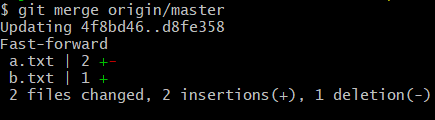


* + 切换到修改过的分支，就可以查看所做的修改



* git merge 远程库别名/待合并分支

效果与分支合并相同



#### pull操作

git pull 远程库别名 远程分支:本地分支

相当于先进行fetch操作，再进行merge操作

### 冲突及解决

#### 原因

当本地库中的修改与待拉取远程库的修改是同一文件的同一位置时，git无法决定保留哪一更改就会出现冲突

#### 解决方法

与分支合并冲突解决方法一样

## 跨团队合作