# 企业级代码管理工具git的应用

## 1.为什么使用github

### GitHub的特点

· 开源的分布式版本管理系统

· 开源项目集中的代码库

· 所有略有规模的公司都在使用

### GitHub的功能介绍

· 记录多个版本

· 查看历史操作，可以进行版本回退和前进的控制

· 多端共享代码，自动合并

## 2. GitHub的使用

### 安装Git客户端

· Linux：yum install -y git

· Windows：<https://git-scm.com/downloads>

· Mac：brew install git

**· 下载安装包，运行 .exe 文件，设置好路径之后，一直next 即可。**

### 检查安装是否成功

· git --version

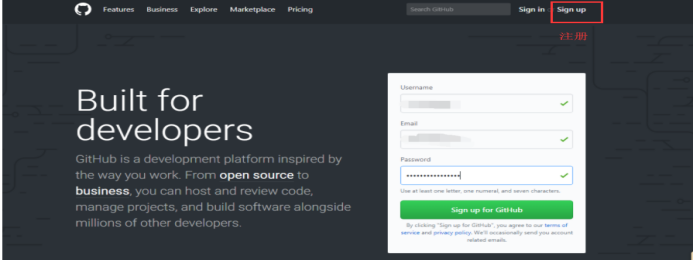
## 3. 拥有一个git账号

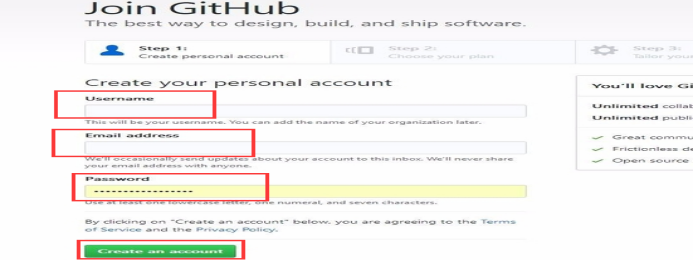
· 主页：<https://github.com>

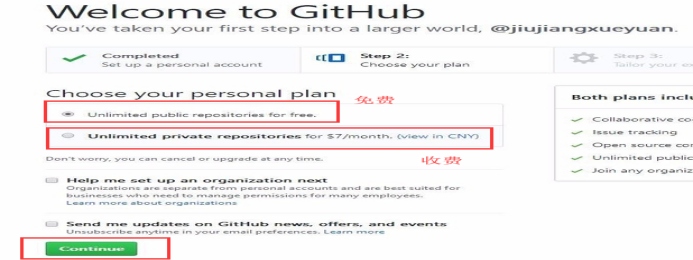
· 推荐使用命令行进行操作 （git 有图形化界面）

· 图形化界面，有些问题相对于来说处理不够清晰明朗，建议使用命令行

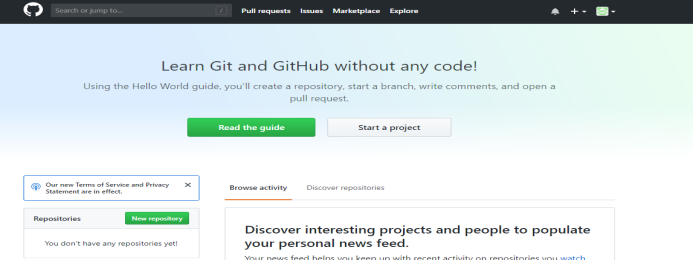
### · 注册



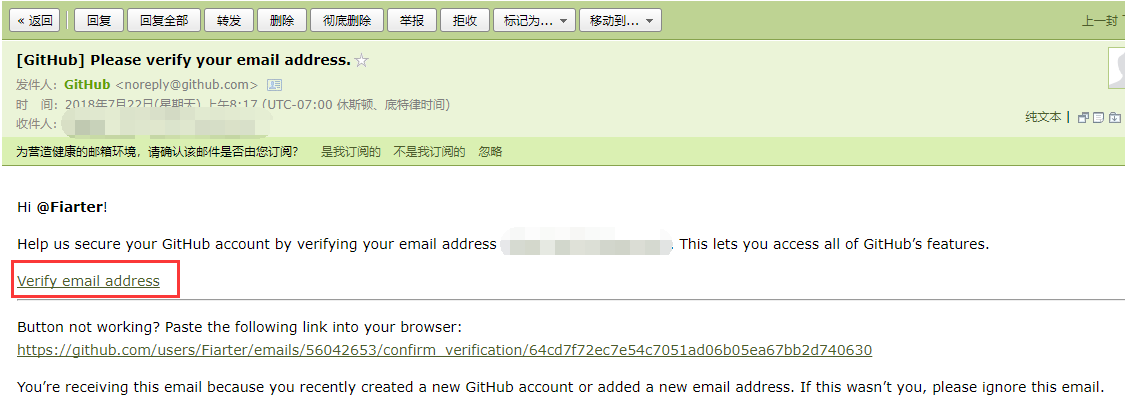




### · submit 提交



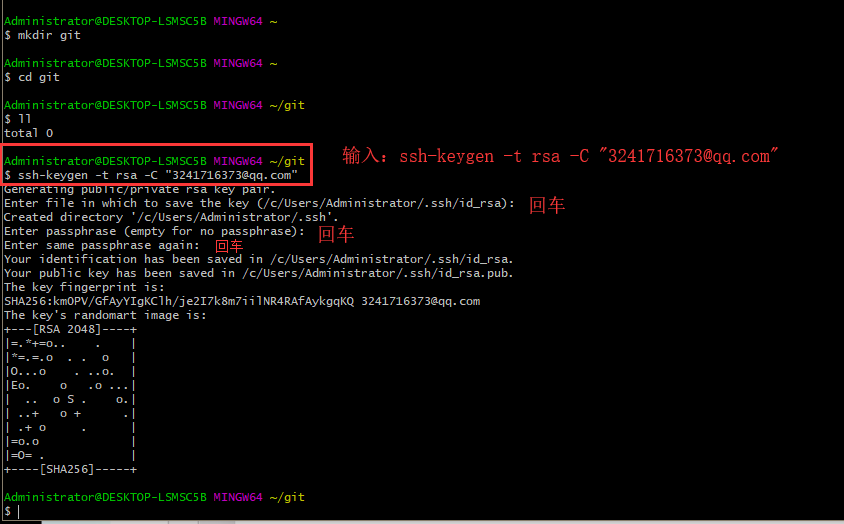
### · 提交后需验证邮箱



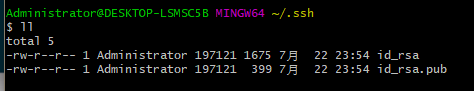
## 4. github的基本使用ssh-key的客户端配置

### · 配置sshkey

#### · 生成key：ssh-keygen -t rsa -C “邮箱地址”



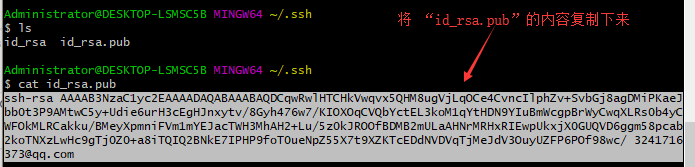
· .ssh 路径下会生成 id\_rsa 与 id\_rsa.pub 文件



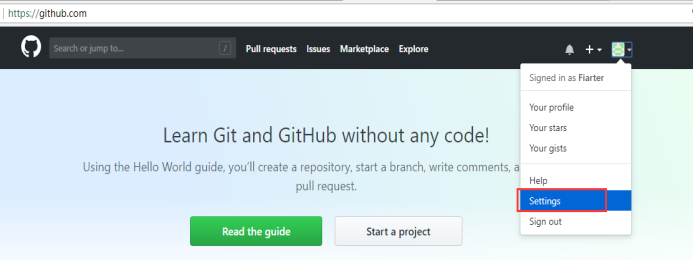
#### · cd ~/.ssh(用户目录下的.ssh文件夹)

#### · 复制id\_rsa.pub的公钥内容到github网站中

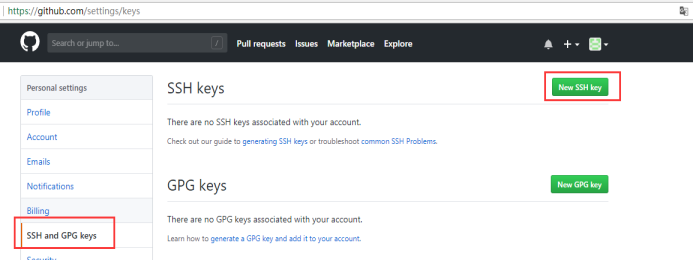
· 复制公钥



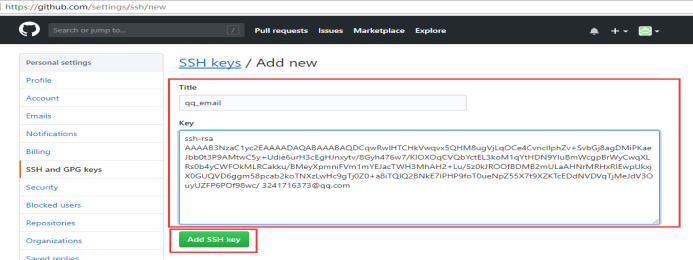
· 进入 github 设置页面



· 选择 “SSH and GPG keys” --> “SSH keys” --> “New SSH key”

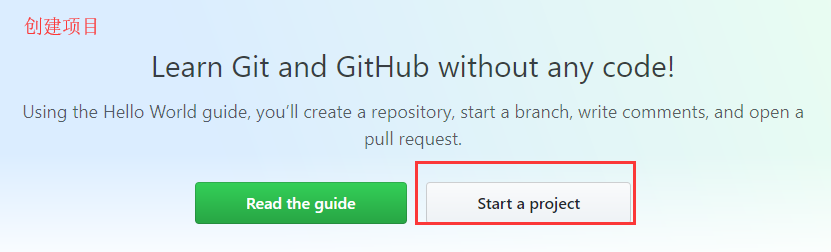


· Title --> Key --> Add SSH key



## 5 创建git仓库及克隆、拉取和推送操作

### 创建项目

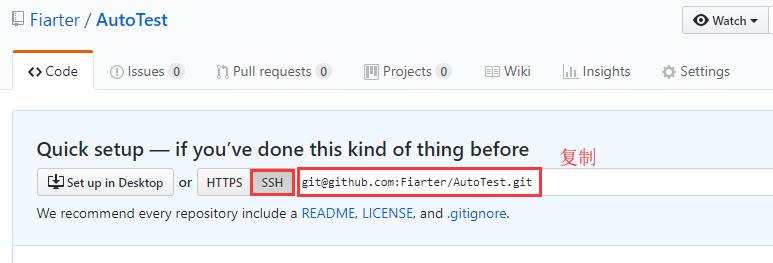


· 有时候创建项目时会验证邮箱，登录邮箱，验证一下即可。



### 项目的克隆、拉取与推送

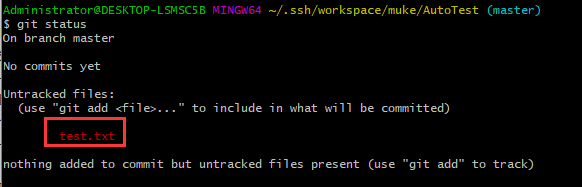
#### · 克隆



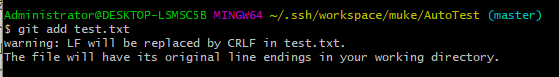


#### · 推送

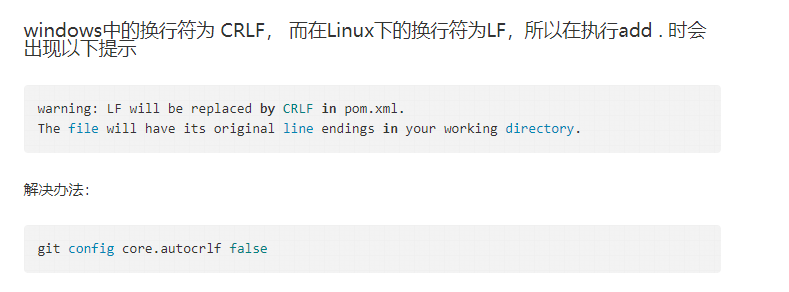
· git status ：查看当前仓库里本地文件的状态



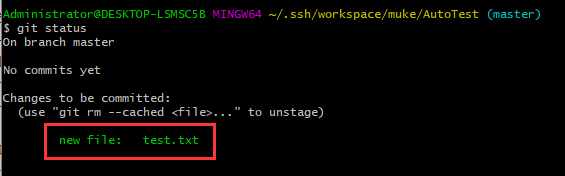
· git add test.txt : 追踪 将 test.txt 文件追踪



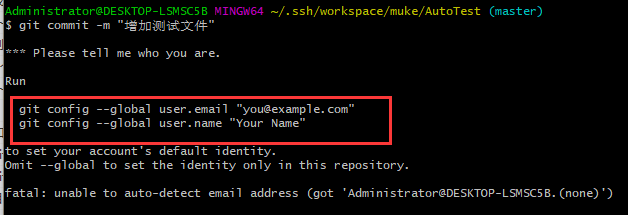
· warning 告警信息解决办法



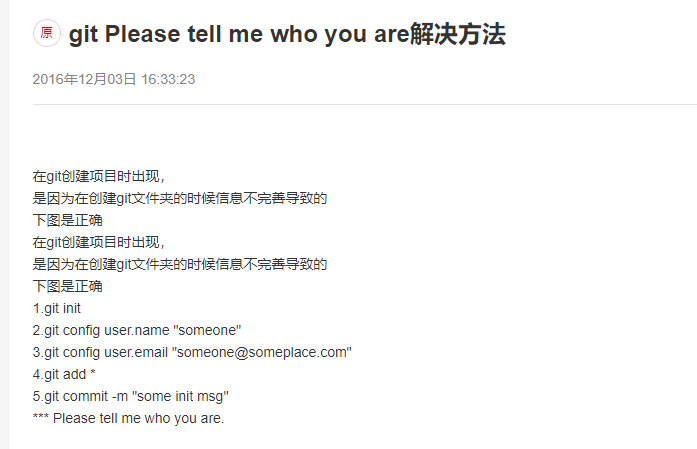
· 再次 git status



· git commit -m "增加测试文件"：输入提交的信息，添加提交的解释说明文件（可中文）

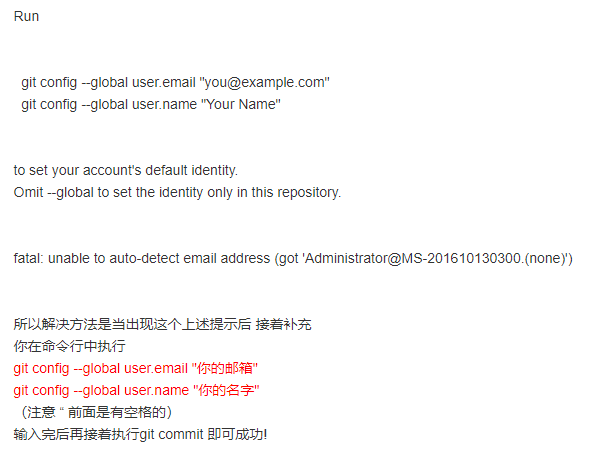


· Please tell me who you are. 解决方法

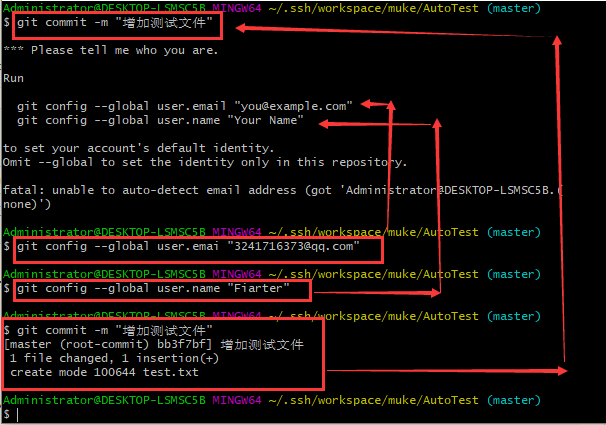


· git config user.name "someone"

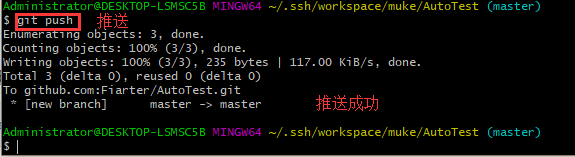
git config user.email "someone@someplace.com"



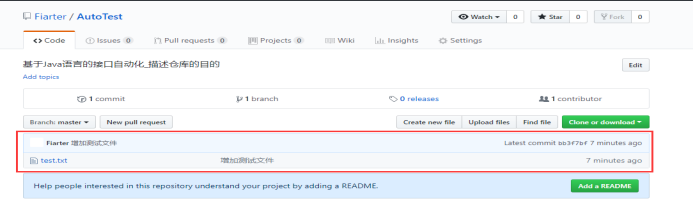
· 解决成功后



· 推送 git push



· 进入 github 查看



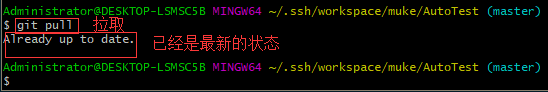
#### · 拉取

· git pull

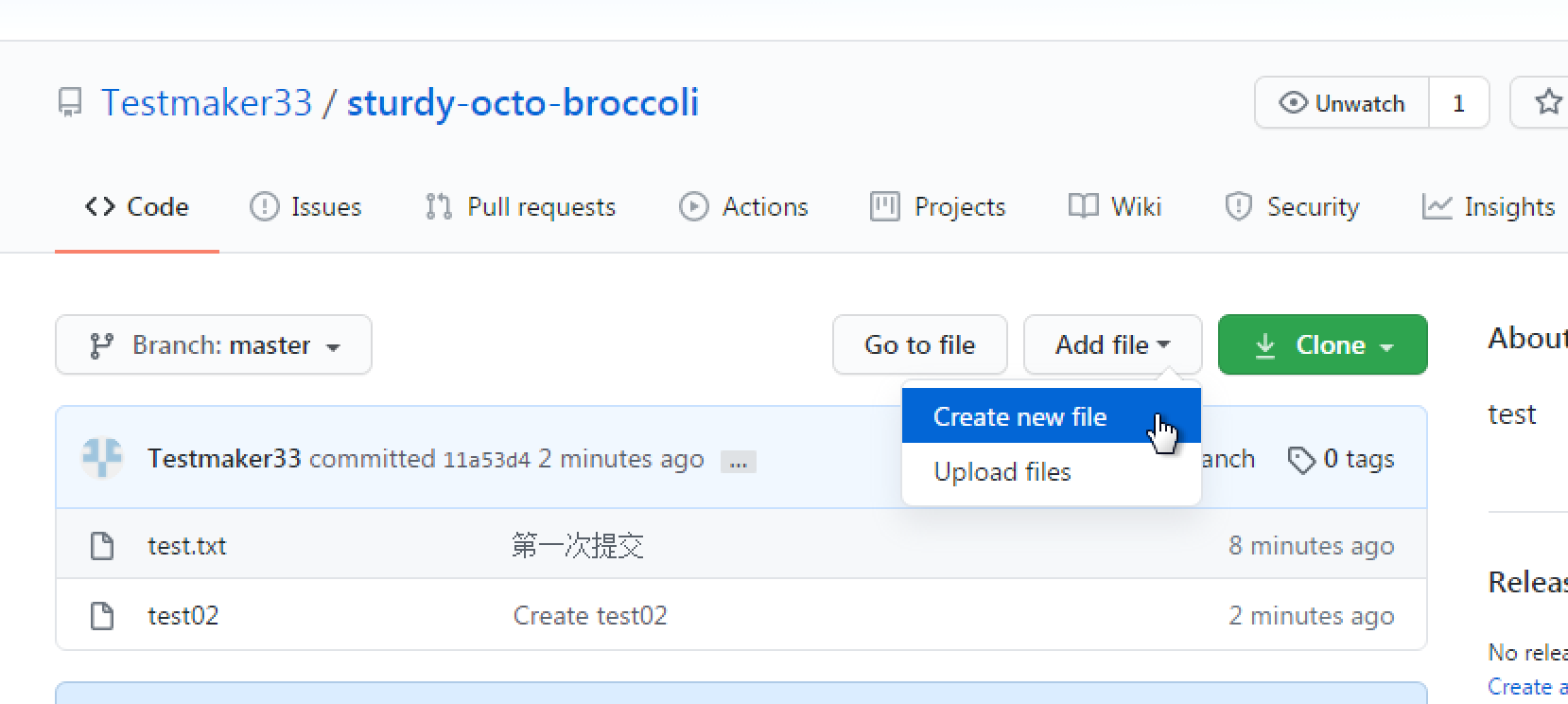
· 什么时候会用到git pull 命令？

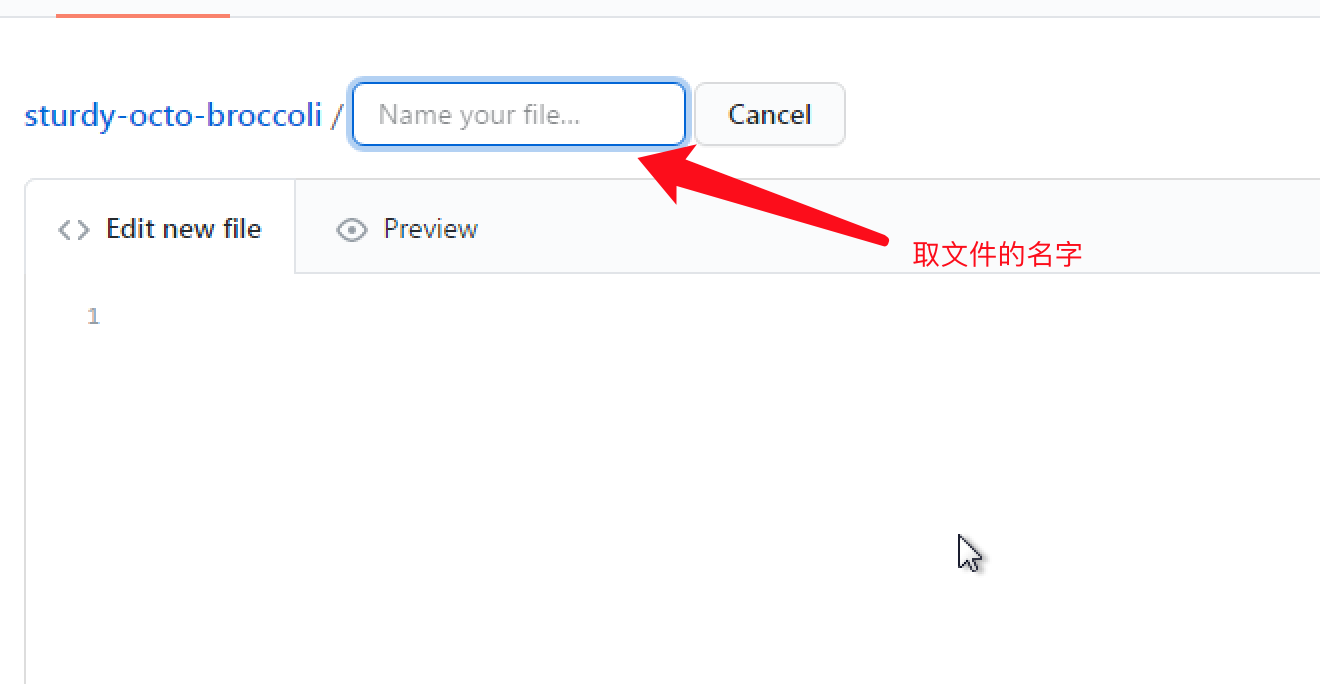
· 本身git 是一款协同开发工具，在其他开发人员同事更新了远端的git仓库，这

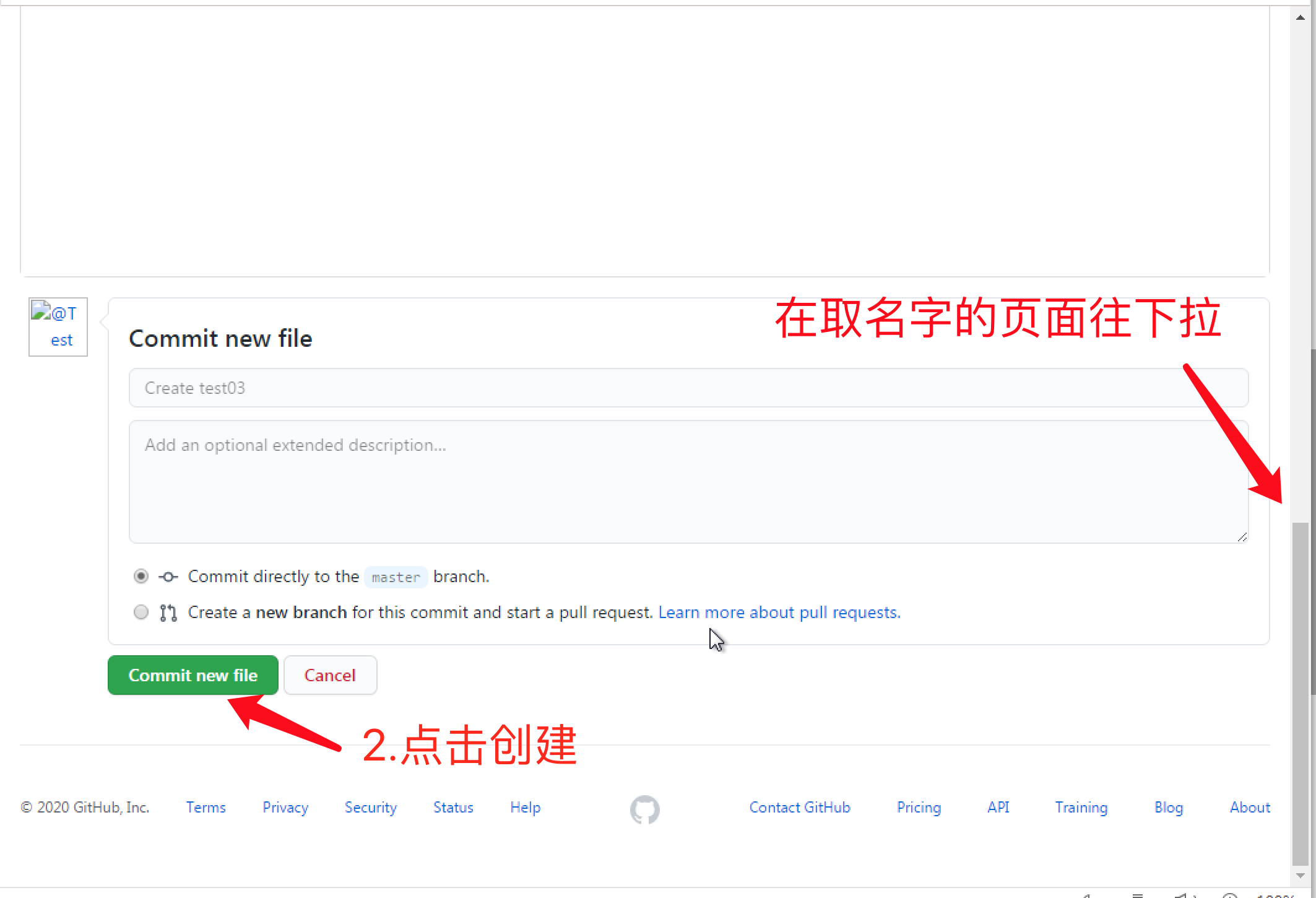
个时候， git pull 命令就会将远端更新的文件，拉取到本地

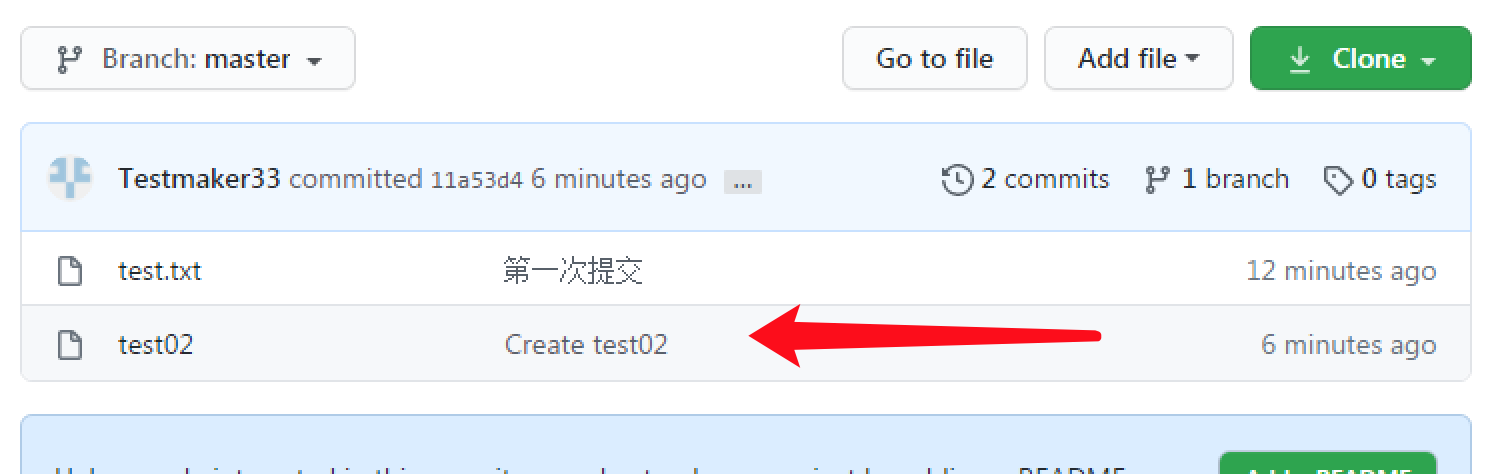


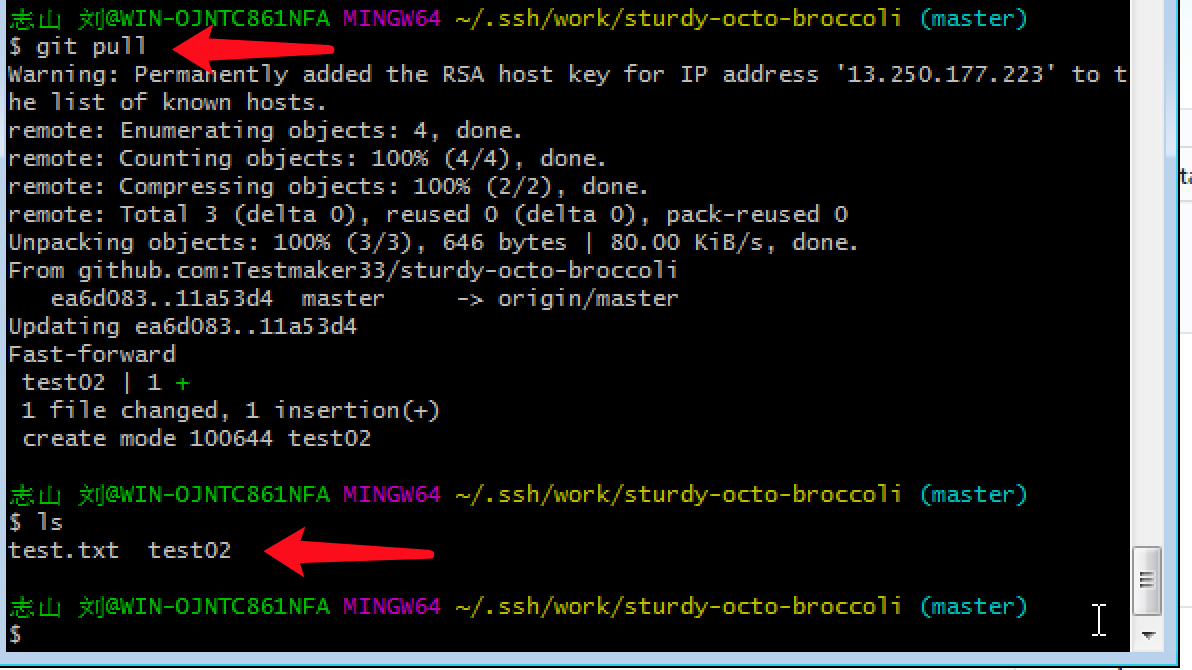
在github的仓库中添加新文件









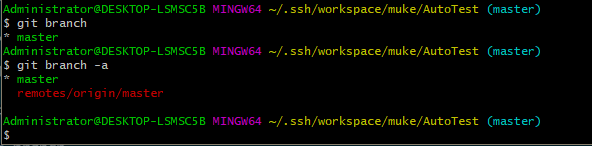


## 6 分支操作-如何创建分支

### 项目的分支操作

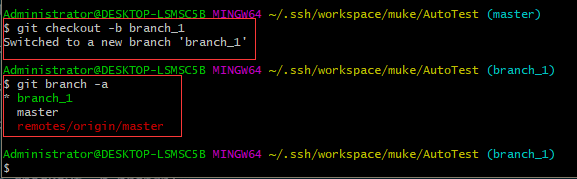
· git branch 查看本地分支

· git branch -a 查看远端远程分支

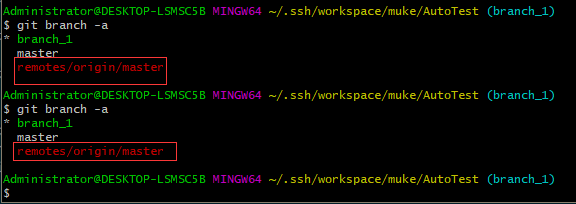


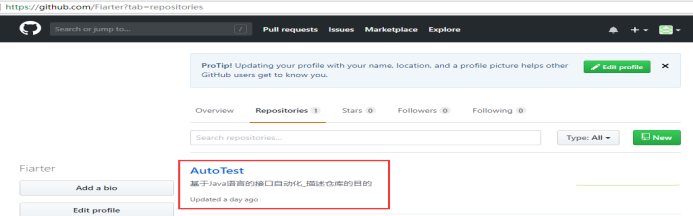
### 本地创建分支

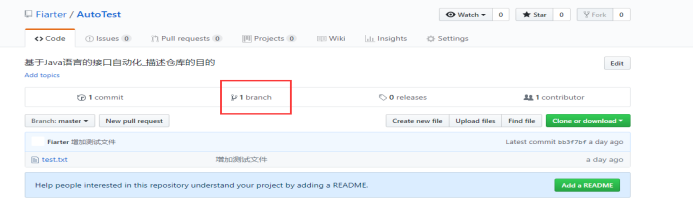
· git checkout -b branch\_1

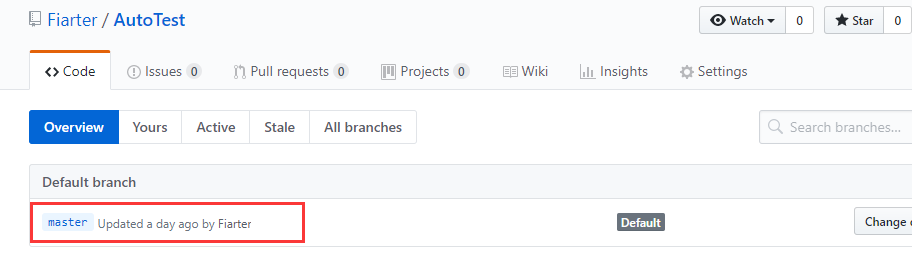


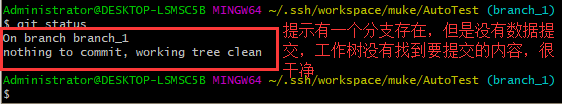
· 本地创建分支成功后，我们会发现远程端并没有分支产生











### 远程端创建分支

· 重新编辑“test.txt”文件

· vim test.txt

· 编辑 增加一些测试内容

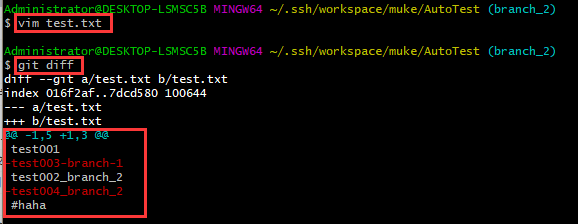
· 举例：test001

test001\_branch-1

#haha

· ：wq （保存）

· git diff (比对下修改前后的内容)



· git status (查看文件状态)

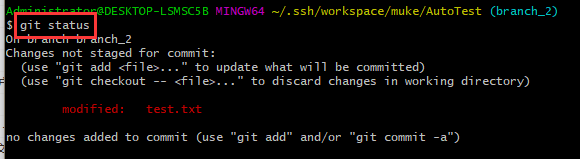
· git add “test.txt” (添加推送文件)

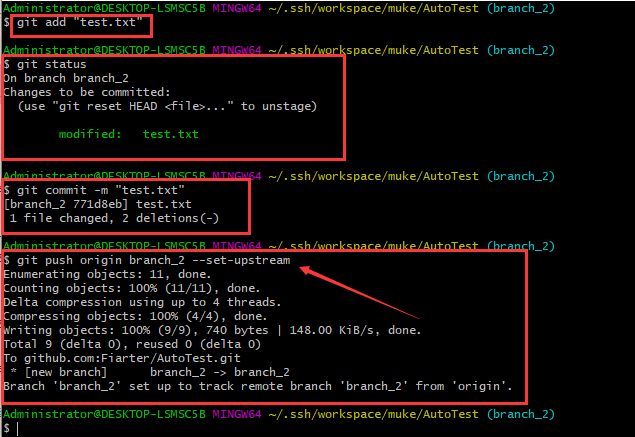
· git status (再次文件状态)

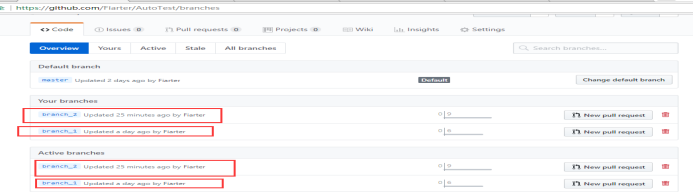
· git commit -m “test.txt” (提交文件)

· git push origin 分支名 --set-upstream (推送)

· git push origin branch\_2 --set-upstream







## 7 分支操作-删除分支

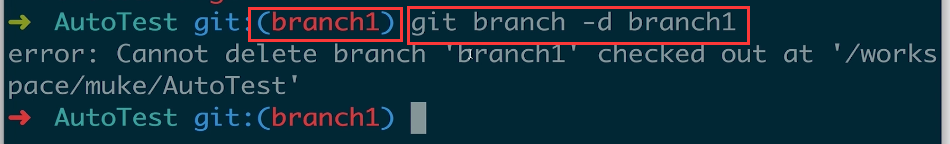
### 本地分支删除

**附：当前所在的工作分支是不能够被删除的，见下图！**

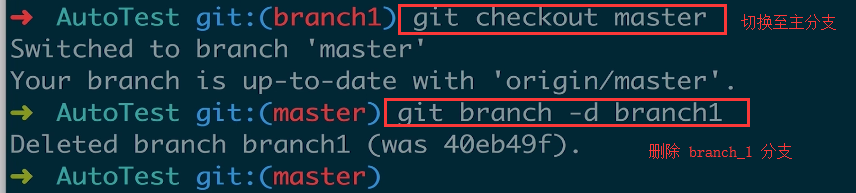
**git branch-1 -d branch\_1**

**· git 分支名 -d ：删除**

**· git 分支名 -D ：强制删除 （一般情况下不用）**



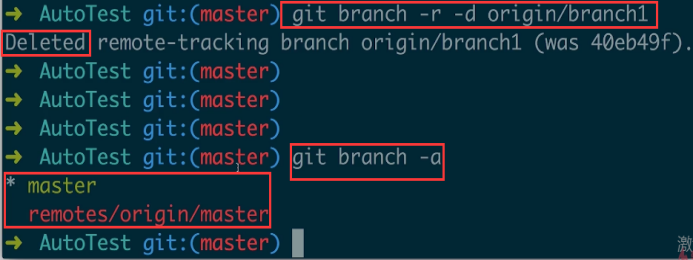
**附：删除某分支（以上图branch\_1为例），需要切换到其他分支,再执行删除的操作**



### 远程端分支删除

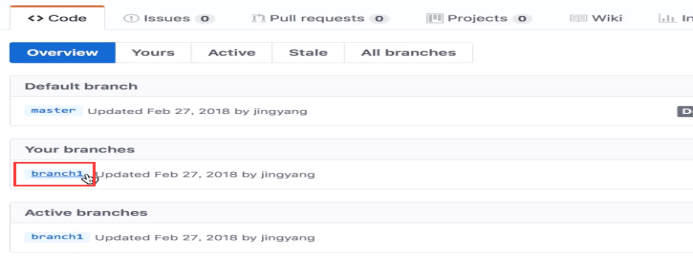
输入命令：git branch -r -d origin/分支名

本地查看：git branch -a



从输出的日志和查看的结果可以看出 远程端的分支 已经被删除了，那么实际结果如何

呢？我们去Web端查看一下

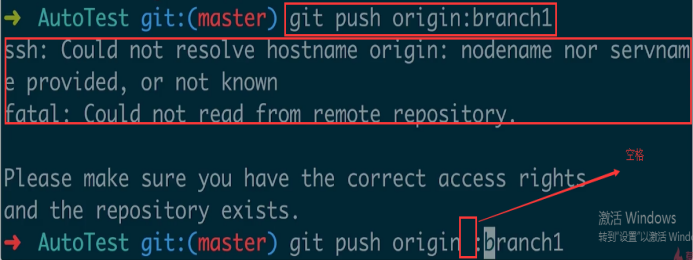


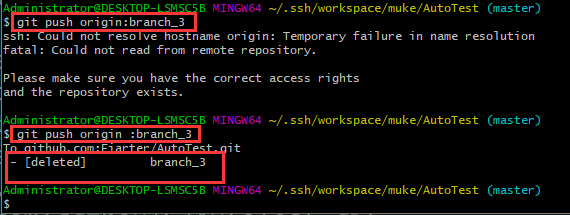
从WEB端我们可以看到分支 branch\_1 并没有被删除。其实我们的操作并没有问题。

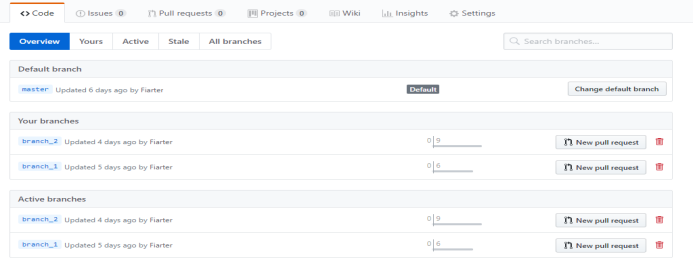
需要将删除的操作推送到远程端。

**· git push origin :branch\_1**

**· 需要注意的一点是 origin 后 有一个空格，不然会出现下图的报错。**







## 8 分支操作-合并分支

### 新建需要合并的分支

· 这里我们新建一个 mergedemo 分支，作为将要合并的分支

· git checkout -b mergedemo



### 编辑该分支需要合并的内容

· vim test.txt

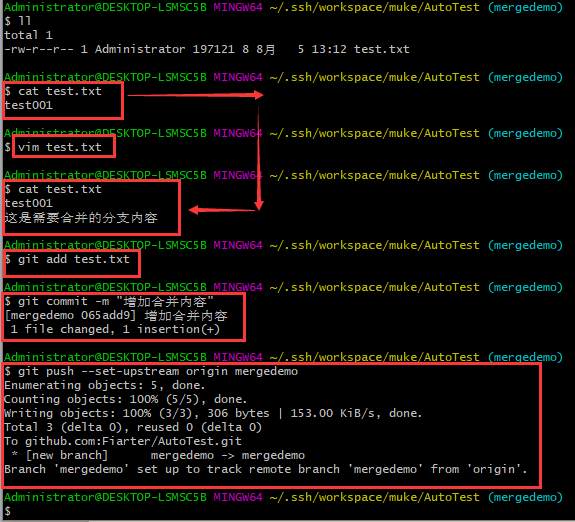
· 增加 “这是需要合并的分支内容”

### 推送至远程端

· git add test.txt

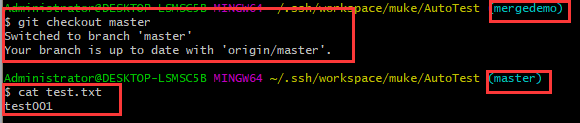
· git commit -m “增加合并内容”

· git push --set-upstream origin mergedemo



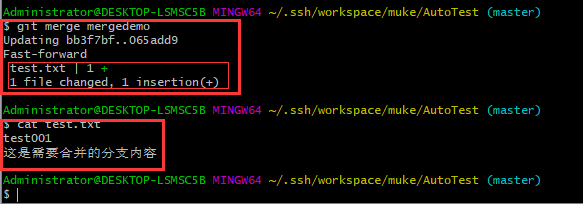
### 切换至主分支

· git checkout master



### 将分支 mergedemo 合并到主分支 master

· git merge mergedemo



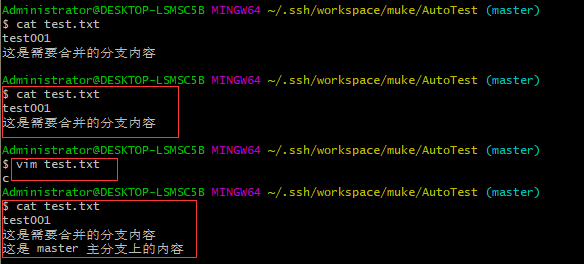
## 9 分支操作-解决合并分支的冲突问题

### master主分支增加需要添加的内容

· 进入 master 主分支

· 如果没有在主分支下，通过 checkout 命令进入

· 见下图

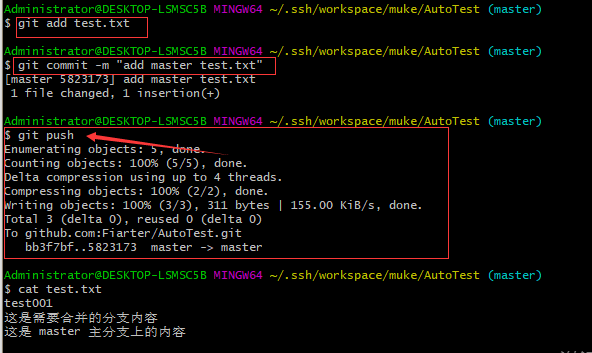


· 将 增加了内容的 test.txt 文件推送至远程端

· git add test.txt

· git commit -m “add master test.txt”

· git push / git push --set-upstream origin master

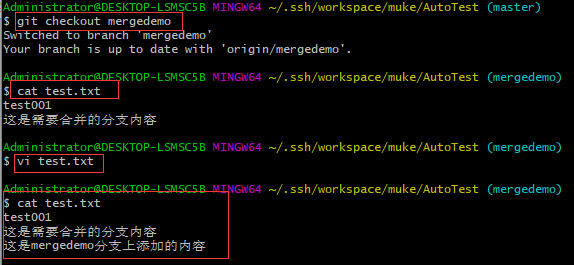


### mergedemo 分支增加需要添加的内容

· 切换 mergedemo

· git checkout mergedemo

· 增加需要添加的内容

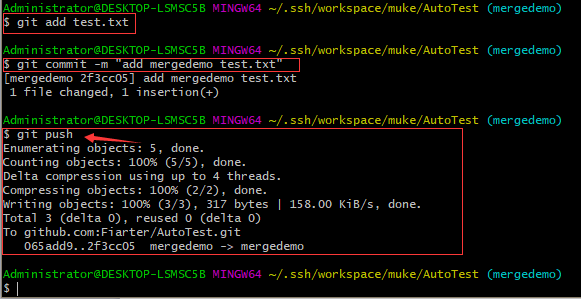


· 将 增加了内容的 test.txt 文件推送至远程端

· git add test.txt

· git commit -m “add master test.txt”

· git push / git push --set-upstream origin mergedemo



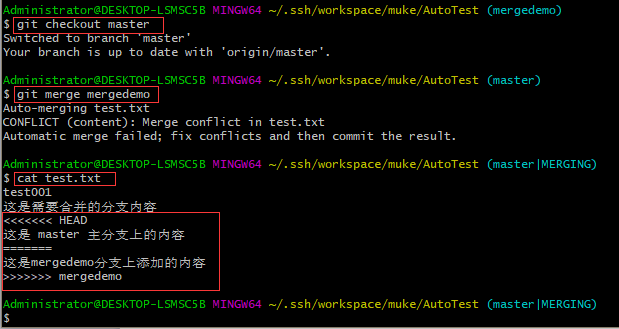
### 合并 master 与 mergedemo

· 切换至 master

· git checkout master

· git merge mergedemo

· 见下图



### 解决合并的冲突问题

**附：其实就是修改内容，然后推送到远程端就行了。**

· vi 修改

· 删除该删除的

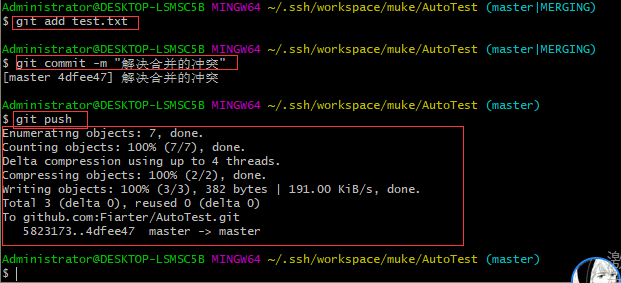
· 修改合并的



· git add test.txt

· git commit -m “解决合并的冲突”

· git push / git push --set-upstream origin master



## 10 git项目版本操作

· 举例

### 编辑 多个版本 推送至远程端

· vim test.txt

· 增加 Version\_1

· 知识点：在存在多个或者大量的文件需要推送至远程端的情况下使用以下命令

· git add .

· 该命令添加分支下需要推送的所有文件

· git commit -m “Version\_1”

· git push

· vim test.txt

· 增加 Version\_2

· git add .

· git commit -m “Version\_2”

· git push

### 本地回退，再将修改过的内容推送至远程端

· git reset --hard HEAD^

· 下图中的476d2f5 是 git 自己记录的版本ID号，唯一的

· ^ :表示是前一个版本

· ^^ :表示是前两个版本

· ^^^ :表示是前三个版本

· 以此类推

· 当需要回到很多个版本之前，使用命令

· git reset --hard HEAD~100

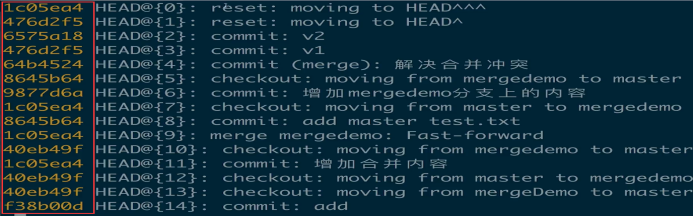
· 会退到前100个版本



### 回到指定版本

· 获取版本ID 号

· git reflog



· 选中想要回退的版本号

· git reset --hard 6575a18 6575a18为想要回退的版本号

· cat test.txt

