## 05-版本切换

1.工作区添加aaa.txt文件

git init

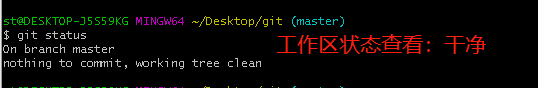
git add .

git commit -m ‘提交说明’

执行以上操作可以理解成一个独立的版本



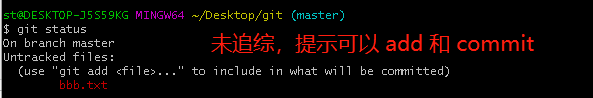
Git status



Git log 如果无法退出输入Q



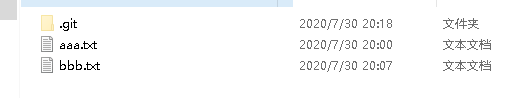
2.接下来在工作区添加bbb.txt文件，直接查看状态



对bbb.txt进行 add 和 commit 操作,然后查看状态



3. 此时工作区有两个文件 aa.txt 和 bbb.txt，如何回到只有aaa.txt时期

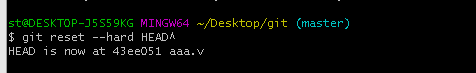


第一步 查看提交日志

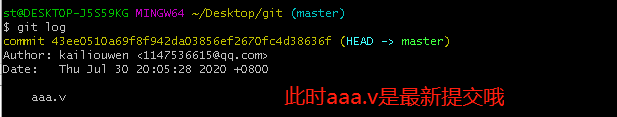


上面可以看到bbb.v是最近一次提交，我需要回到它前面一次提交

第二步 git reset --hard HEAD^

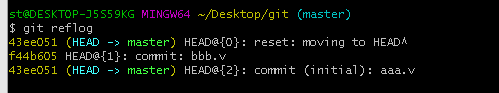


此时回到bbb.v前面一步提交，再次查看日志



4. 回到bbb.v提交时期

git reflog 查看当前版本，查看当前目录所有提交日志



git reset --hard f44b60 找到bbb.v版本id ,回到bbb.v版本

然后查看当前最新一次版本



## 06-git撤销修改

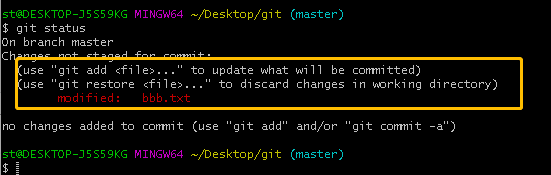
### 1.工作区恢复

1. 现在在bbb.txt中添加以下内容（ccc 和 ddd），然后保存



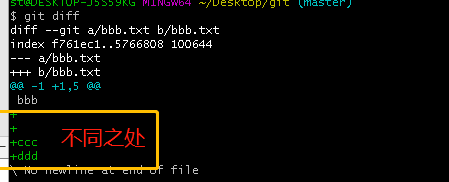
接下来查看状态

显示没有提交到暂存区，和如何恢复原先版本内容（git restore）



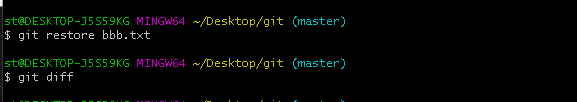
现在工作区内容和之前已经不同,查看不同之处

git diff

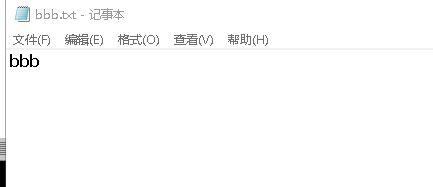


回到工作区修改之前

git restore bbb.txt



现在恢复到工作区修改之前，查看工作区bbb.txt



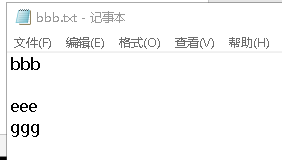
此时 ccc 和 ddd 已经没有了

注意：以上是没有存到暂存区的情况

### 2.暂存区恢复

如果已经添加到暂存区了呢

接下继续修改bbb.txt文件

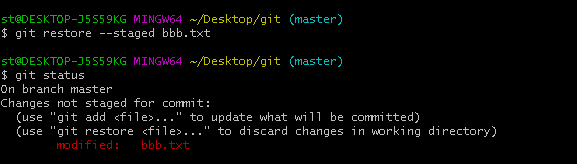


添加到暂存区,查看状态：绿色



图片显示如果要恢复到原先版本使用git restore –staged

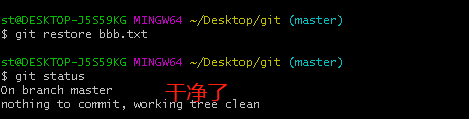
Git restore –staged bbb.txt



现在暂存区已经恢复了，但是工作区还是更改了



继续使用 git restore 恢复工作区原先状态



此时干净了，查看工作区bbb.txt

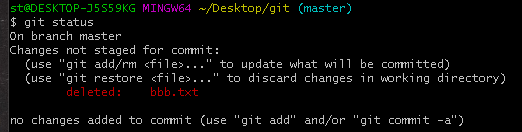


## 07-git删除文件

1. 在工作区删除bbb.txt文件

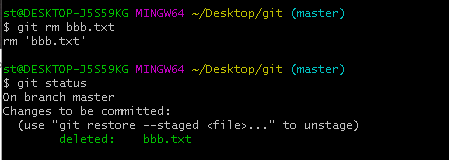


查看状态



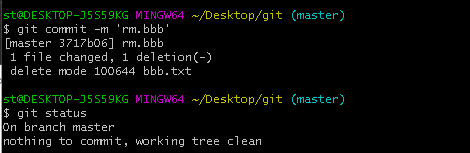
现在和git版本管理暂存区中不同步

Git rm bbb.txt



此时git 暂存区同步删除 绿色

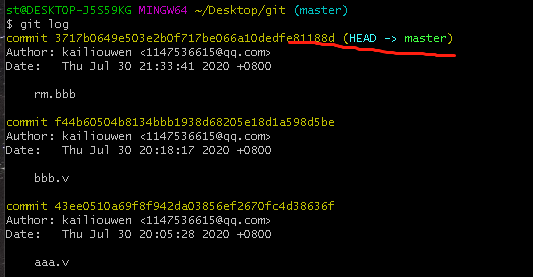
添加当前到本地仓库



工作区是干净的

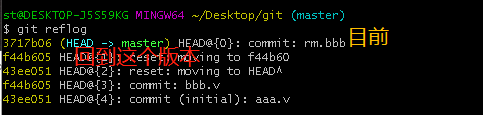
现在恢复删除bbb.txt

Git log



查看当前版本

Git reflog



恢复某一版本

git reset --hard f44b60 f44b60这个是id



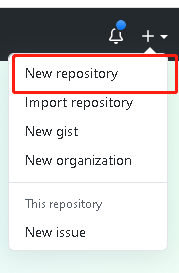
此时bbb.txt回来了



## 08-更新到远程仓库和从远程仓库拉取数据

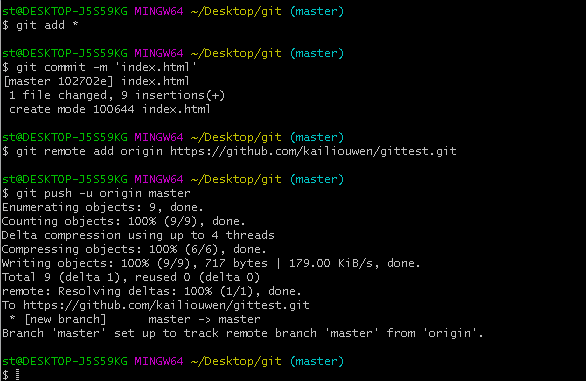
### 1.更新到远程仓库 GitHub

创建远程仓库

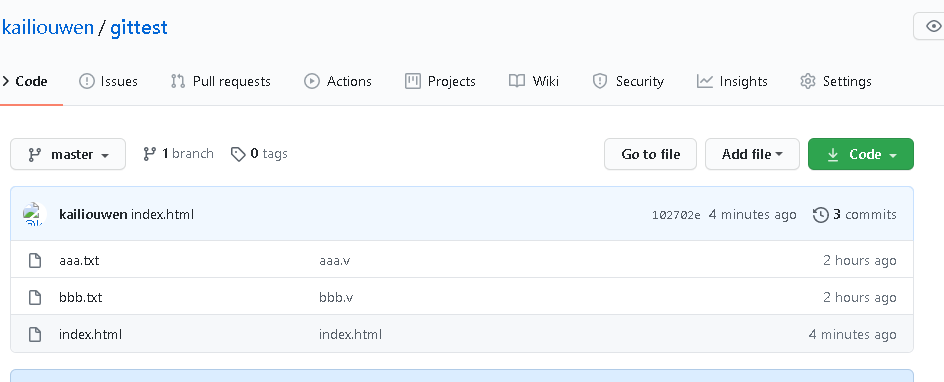




提交到远程仓库



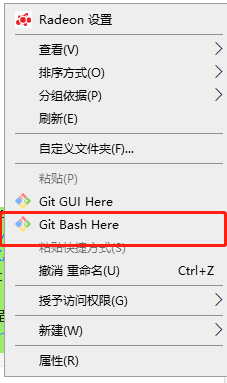
刷新远程仓库页面



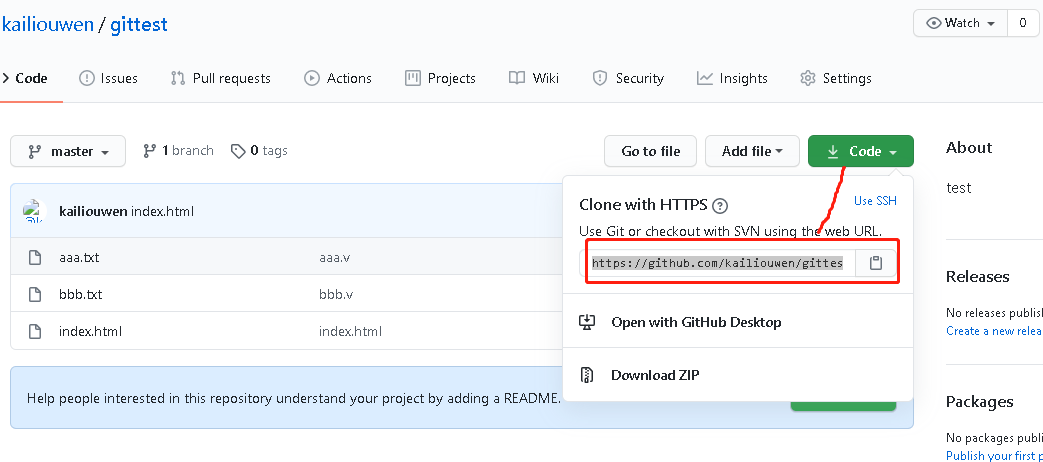
### 2. 从远程仓库拉取

假设我在github看到项目，然后我要拉取

创建一个文件夹 Pull -> 进入文件夹 -> 右键

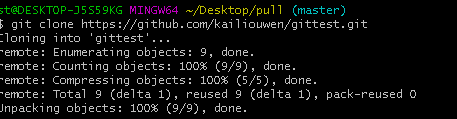


找到远程仓库地址



执行克隆命令

git clone <https://github.com/kailiouwen/gittest.git>（写你自己项目地址）

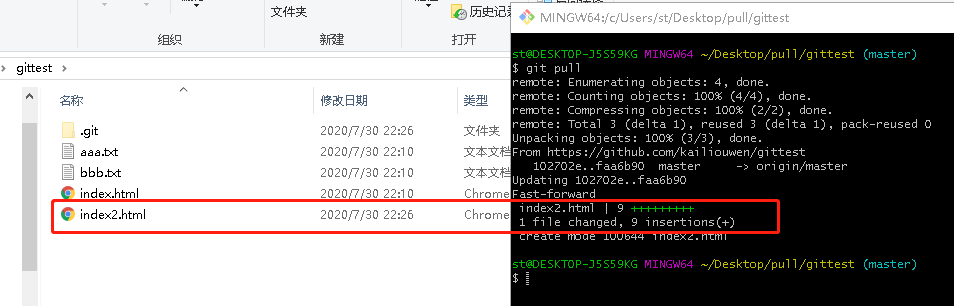


此时查看pull文件夹



由于我克隆的远程仓库更新了，我拉取的仓库也需要更新

Git pull



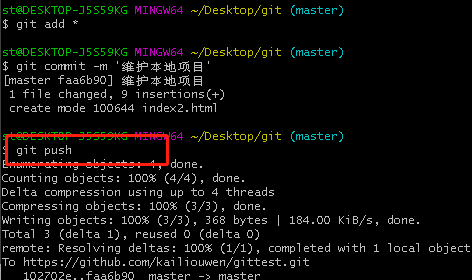
此时拉取到一个更新文件index2.html

### 3. 维护本地项目

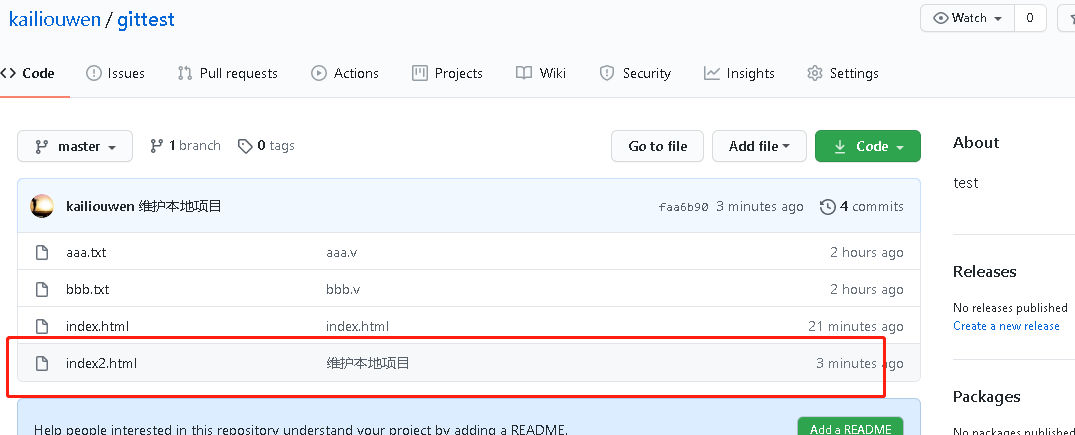
新添加了index2.html文件



然后把他添加到远程仓库，注意:之前已经和远程仓库建立联系,不用再进行远程关联



刷性远程仓库页面，此时多了index2.html



## 09-分支的创建\_合并\_删除

分支：可以理解成不同版本库，每个库放自己这个库的内容，可以合并

### 1.创建分支

在主分支master下创建c++文件夹，并提交到本地仓库



创建一个分支 dev，

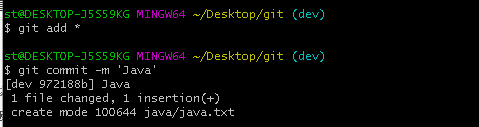
git checkout -b dev



接着，在dev分支工作区创建文件夹java/java.txt



然后，提交到本地仓库



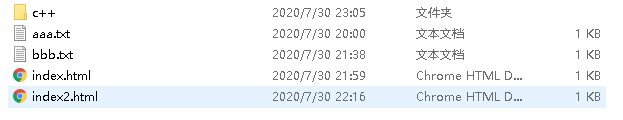
当前目录结构如下：



切换到master分支

git checkout master

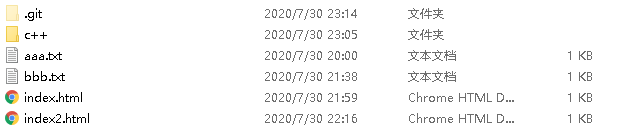
当前目录结构如下：



### 2.合并分支

在master下合并dev分支

未合并时：

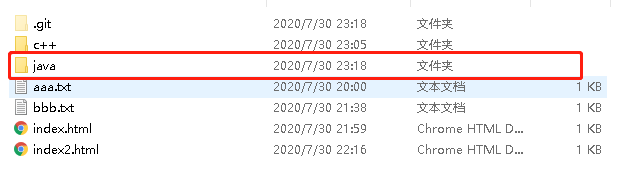


合并分支 dev

git merge dev



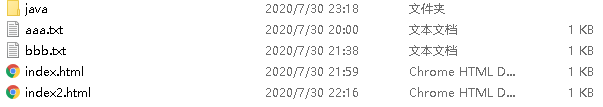
合并之后 dev(java file)



此时 分支 dev 还保留自己的内容

切换分支查看

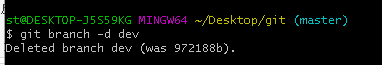




### 3.删除分支

现在master已经合并dev, 我想删除dev分支

git branch -d dev

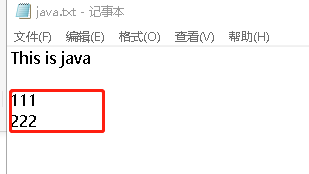


查看dev 显示没有这个分支了



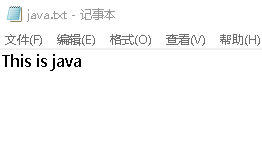
## 10-解决冲突

创建一个分支dev，修改java.txt 添加111 222 文本

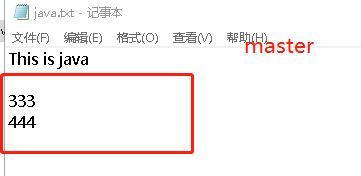




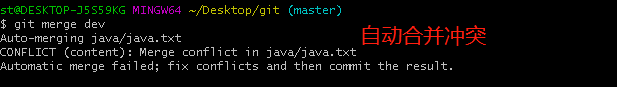
再次切换回master



接着添加333 444 文本,提交到本地仓库



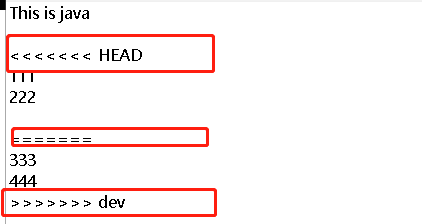
现在看到一个情况，dev分支添加文本 111 222；master主分支添加文本 333 444 ;这种情况下如果合并就会冲突



处理方法：手动处理冲突

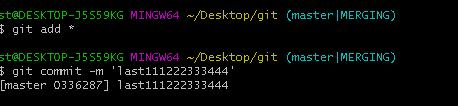
查看java.txt

找到冲突位置，删除红色区域多于字符，



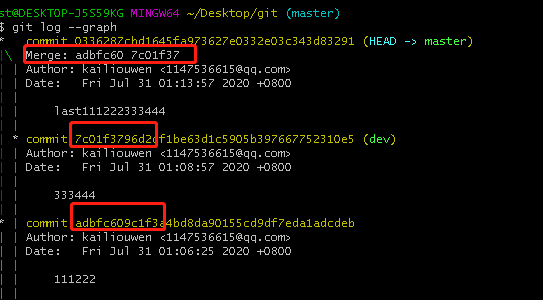
再次手动添加到本地仓库





查看合并历史

git log –graph



## 11-分支管理

中途需要修改其他分支，然后本分支因为没开发完成，所以不能提交，可以先保存当前现场状态

git stash



修改其他分支回来恢复现场

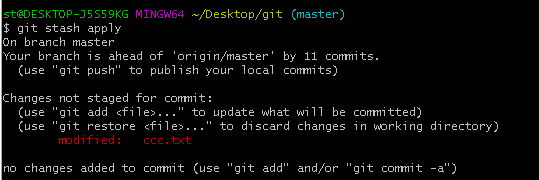
git stash list



这里表示保存了两个stash

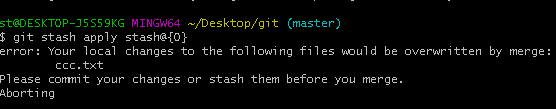
接着恢复现场

git stash apply



也可以指定恢复那个现场

git stash apply stash@{0}

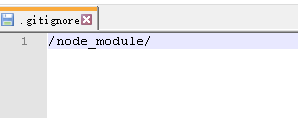


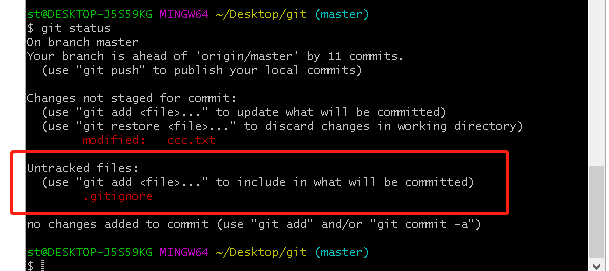
## 12-git忽略文件

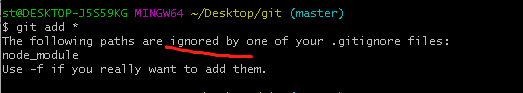
开发项目中node\_module这个文件不需要经常提交，减少内存占用

此时可以忽略这个文件

忽略node\_module,创建.gitignore文件，输入以下代码



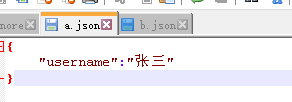


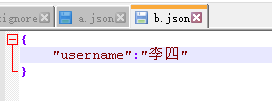


此时提交到本地仓库node\_module文件夹被忽略

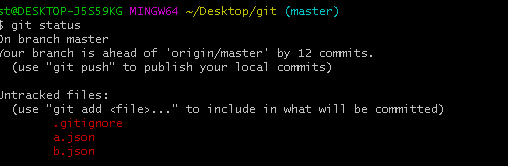
现在忽略后缀名为.json的文件

创建a.json 和 b.json



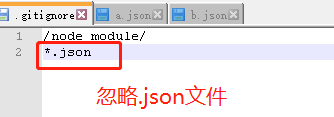


查看状态

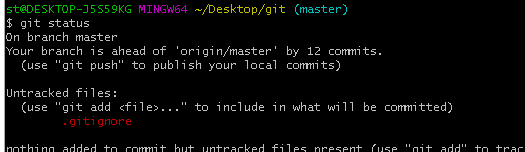


此时a.json 和 b.json 是未追踪

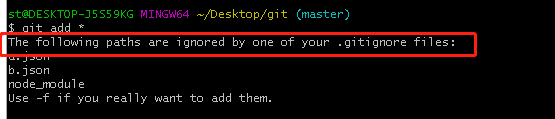
接下来忽略这两文件



在看状态，此时已忽略所有..json文件



这些都是忽略的文件



不能忽略

!.gitignore

