一、将文件添加到暂存区，以及提交到本地库

● 1、加载当前目录下的详细资源

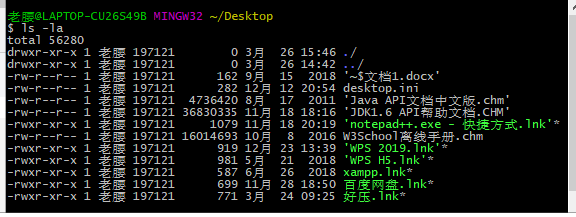
ll

ll .git/ .git 代表的是文件名



● 2、列出当前目录下的资源（包括隐藏资源）

ls -la

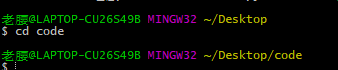


● 3、列出当前目录下的资源,分屏查看资源（包括隐藏资源）

ls -l|less

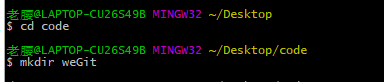
● 4、打开资源目录文件

cd code



● 5、新建一个目录（项目名）

mkdir weGit weGit: 代表文件目录名



● 6、初始化该目录，变成一个git仓库 （在该目录下打开git bash here）

git init



pwd 查看当前的路径

注意：

.git目录下存放的是本地库相关的子目录和文件，不要删除也不要修改

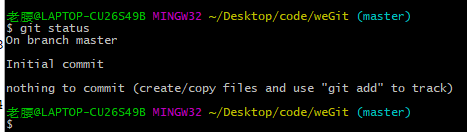
● 7、设置免密公钥，区分用户

git config --global user.name "用户名"

git config --global user.email 邮箱地址

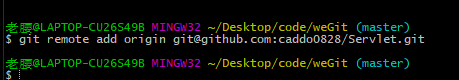
● 8、查看git工作区存储的状态

git status



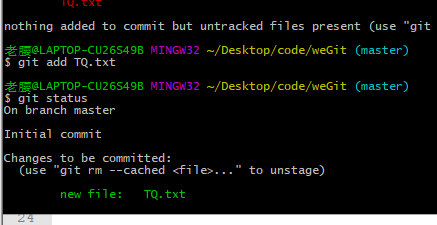
● 9、与远程仓库进行关联 (ssh地址信息)

git remote add origin git@github.com:caddo0828/Servlet.git



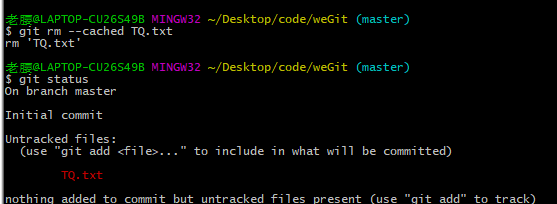
● 10、追踪文件，添加到暂存区 git add . 表明提交当前目录下所有文件

git add 文件名



● 11、将文件从暂存区中移除

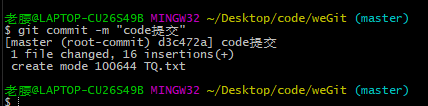
git rm --cached 文件名



此时再次查看git 状态，文件出现红色，表明未提交到暂存区，未进行追踪

● 12、从暂存区提交到本地库

git commit -m "提交注释的内容"

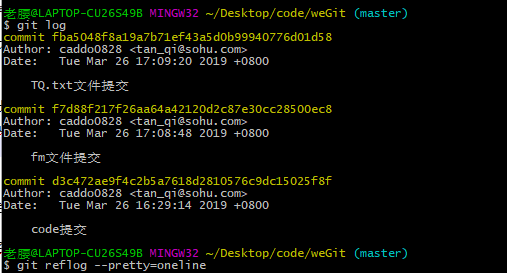


● 13、从本地库第一次提交推送到远程仓库

git push -u origin master

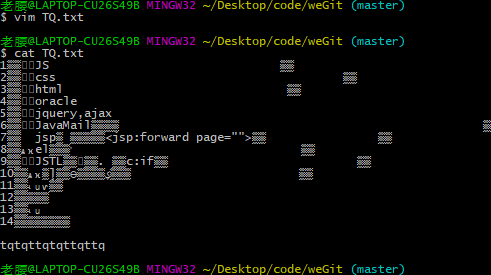
二、历史记录信息:

● 1、获取全部的提交记录 git log



cat 文件名 // 查看文件内容

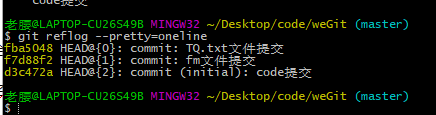
Vim 文件名 【i】 //进行编辑模式



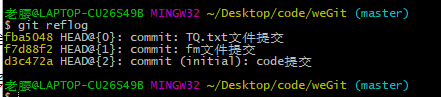
git log --pretty=oneline 以行显示哈希值

或者

git log --oneline



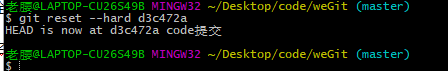
git reflog 前进后退，方便移动指针，知道指针要移动多少次

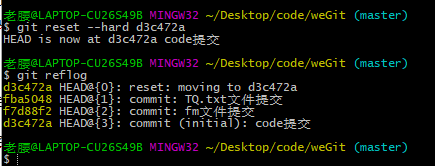


HEAD@（移动到当前版本需要多少步）

● 2、根据索引值前进后退（历史记录） reset 命令

\* git reset --hard hash值

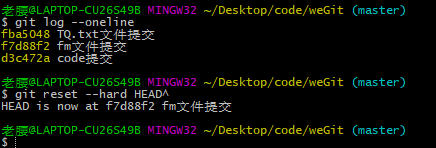




\* 前进后退的其他版本，只能往后退

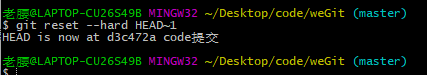
git reset --hard HEAD^

（^一个此符号代表后退一次, 以此类推有几个符号就退几步）



git reset --hard HEAD~n

（ n代表一次性后退的次数）



● reset 命令的参数解释

》 --soft [哈希值]

仅仅在本地库移动HEAD指针

》 --mixed

在本地库移动HEAD指针

重置暂存区

》 --hard

在本地库移动HEAD指针

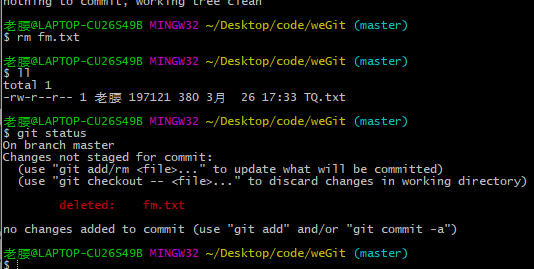
重置暂存区

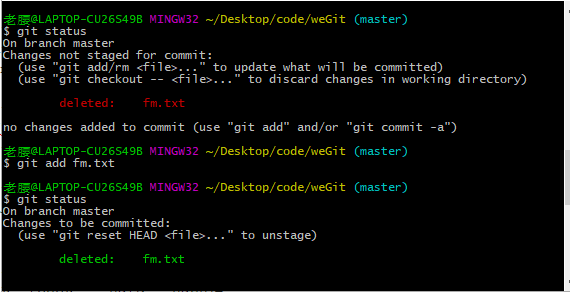
重置工作区

三、删除文件找回：

● 1、删除文件

rm 文件名





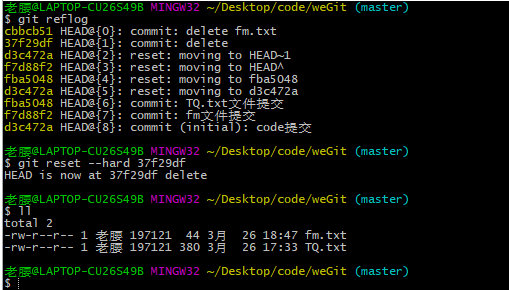
查看暂存区状态、将删除的文件添加到暂存区，此时暂存区会保留此文件被删除的记录

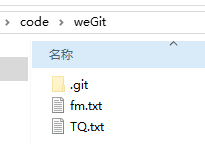
● 2、文件的找回

利用git reset --hard [指针位置] 回到之前的版本就可以找回之前的文件

\* 删除操作已经提交到本地库： 指针位置指向历史记录 ，就是HEAD的哈希值

\* 删除操作尚未提交到本地库： 指针位置使用HEAD



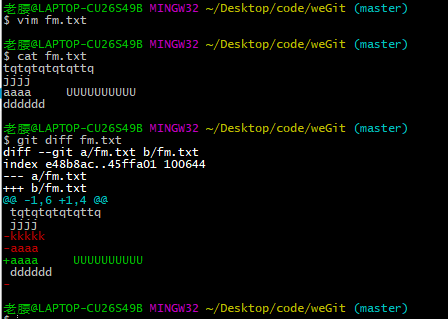


● 3、文件的比较

git diff 文件名

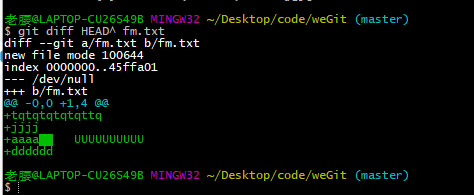
\* 将工作区的文件与暂存区的进行比较

\* - 号代表删除的行 ， + 代表增加的行



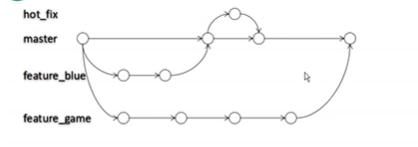
git diff HEAD^ 文件名

\* 将工作区文件与本地库历史记录进行比较



\*不带文件名，比较多个文件

1. git 分支的创建以及删除



● 1、创建新分支 （避免直接在主干中进行开发， 但是又是从主干中继承来的）

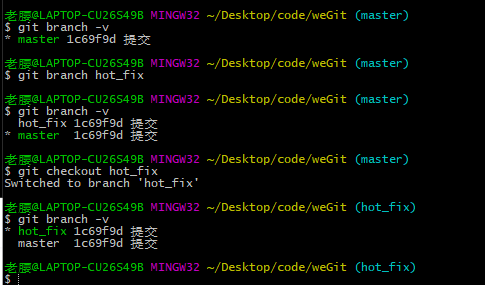
git branch 分支名

● 2、查看分支

git branch -v

● 3、切换分支

git checkout 分支名



修改分支上的内容，在查看，此时主干和分支就内容跟不一致了

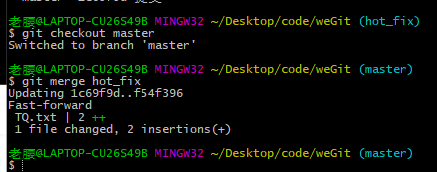


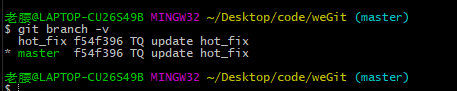
● 4、合并分支

\* 第一步： 切换接收修改的分支上（被合并，增加新内容）

\* 第二步： 执行命令

git merge 分支名

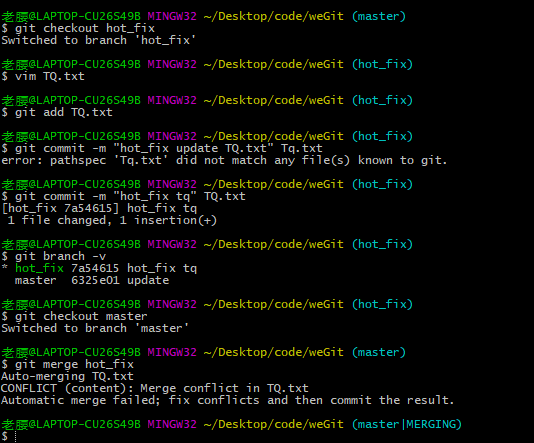




● 5、解决冲突

\* 当分支中修改了文件，主干中也修改了文件，将两个文件进行合并时，会出现冲突

\* 冲突如下

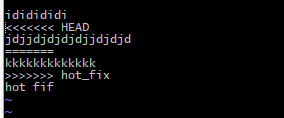


\* 【master| MERGING】

代表手动合并失败，正处于合并状态中

第一步： 使用vim 文件名

【：set】进入文件编辑模式中显示产生冲突的地方

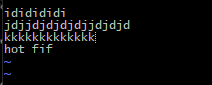


【HEAD】 表示当前分支内容

【hot\_fix】表示其他分支内容

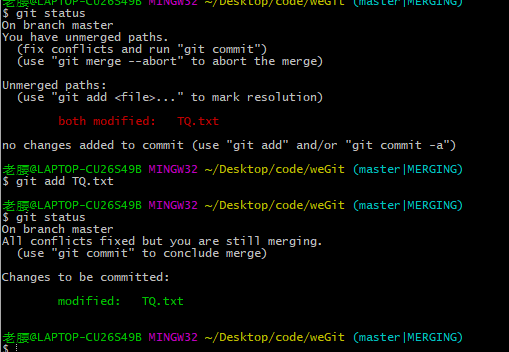


第二步、 \* 修改文件中的特殊符号，去除掉冲突提醒标志



第三步、\* 重新add 冲突文件到暂存区

git add 文件名

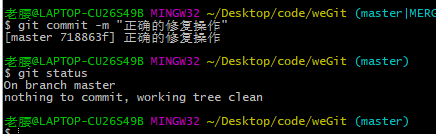


第四步、\* 重新提交文件到本地仓库

（此时不能带文件名进行commit， 否则抛一个致命性的错误, 如图示报错）



git commit -m “注释内容” （正确操作）



此时【MERING】冲突就被解决了

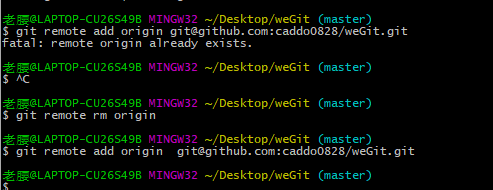
五、远程仓库与本地仓库的交互

● 1、将远程仓库与本地联系起来

git remote add origin ssh的地址标识符

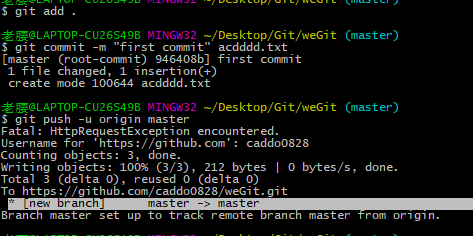
\* 如果说已经存在origin ，那么先删除原先的在添加

git remote rm origin



● 2、推送到远程仓库

git push origin master



● 3、克隆项目 git clone ssh地址标识符

git clone [git@github.com:caddo0828/Servlet.git](mailto:git@github.com:caddo0828/Servlet.git)

● 4、从远程仓库中获取修改的信息

git pull