

#### Working Directory：工作区——.git所在的文件夹

#### stage/index:暂存区

#### Repository：本地仓库——.git所在的文件夹，被git管理

Remote:远程仓库

git status:查看状态

建议做什么操作前都用一下这个

git add <file>...:把工作区的改变添加到暂存区（stage操作）

git restore --staged <file>... :把工作区恢复成暂存区的样子（unstage操作，stage的逆操作）

Git diff :比较文件不同

git commit -m ”<一些注释>”:把暂存区的内容提交到仓库

一旦被commit，就相当于永久备份，随时可以恢复到某个commit过的版本

git pull:把远程仓库的内容更新到本地仓库

Git push:把本地仓管更新到远程仓库上去，一般push前要pull，避免冲突

切换版本

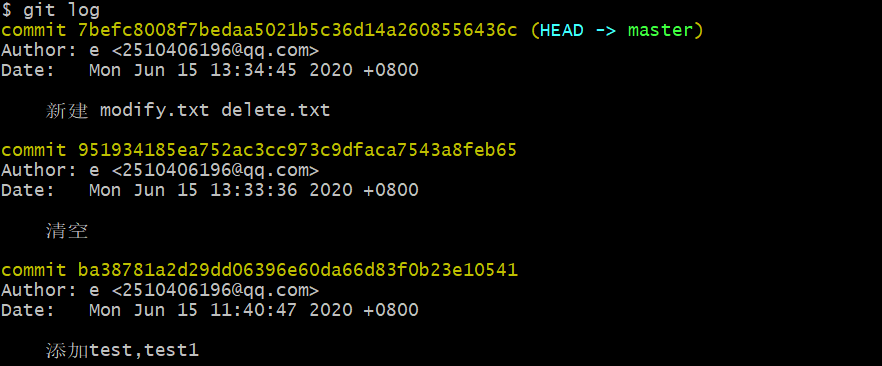
git reset --hard HEAD:把工作区恢复成版本库的最新版本

git reset --hard HEAD^:把工作区恢复成版本库的上一版本，多少个’^’就往前推几个版本，window的控制台需要变成HEAD”^”,HEAD”^^”.......，git Bash不需要

git reset --hard HEAD~n:把工作区恢复成版本库往前推n个版本，

git reset --soft HEAD:把暂存区恢复成版本库的最新版本，

Git log：查看提交日志 参数--pretty=oneline可以查看简易信息



Git reset --hard <版本号的前几位，足以区分就行> :将工作区恢复成任意版本

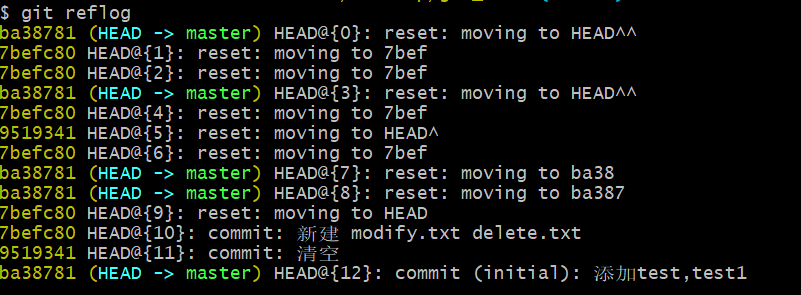
版本号是上图commit 后的十六进制序列

Git log只能看到之前的版本，所以你恢复到之前的某个版本后，就看不到之后的版本了

这个时候可以用

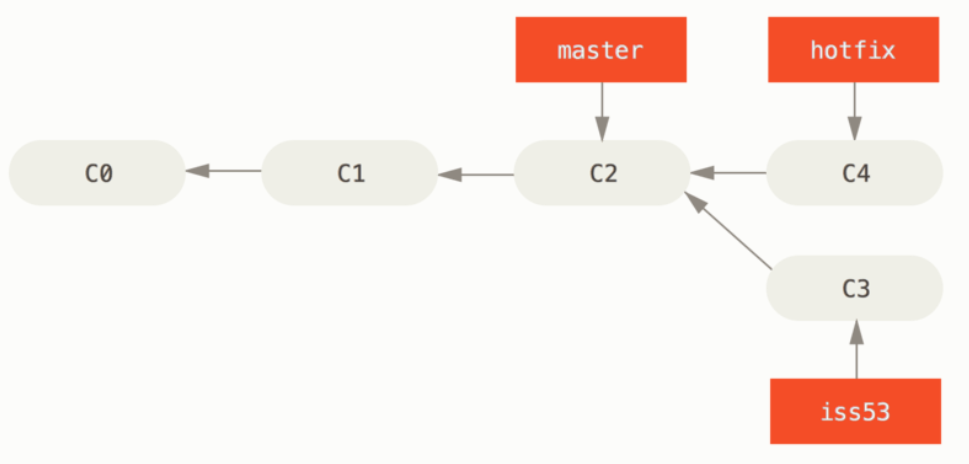
Git reflog

不仅可以看到commit操作，也可以看到reset操作



这样就可以找到之后版本的版本号（每条记录最前面的十六进制序列），然后恢复回之后的版本了

分支



很多个版本可以构成一棵树，每个版本就是一个节点，如上图的c0~c4

一个分支是一个指向某个版本的指针，如上图的master，hotfix，iss53

Master 分支是自动构造的默认分支

HEAD是一个指向分支的指针（指针的指针），代表当前工作的分支，即HEAD指向上图的master，hotfix或iss53

当有了多个分支之后，就可以通过改变HEAD在不同的版本节点间切换。

比如自己未完成的工作，当然是不可以直接提交到主分支，这样别人就无法工作了。

但是又想要备份自己未完成的工作，就可以创建一个分支，在这个分支，把自己未完成的工作备份好，等全部完成，再与合并到主分支中，创建的分支就可以删除了

git branch <新分支名>：创建新分支

Git checkout/switch <分支名>：切换到分支

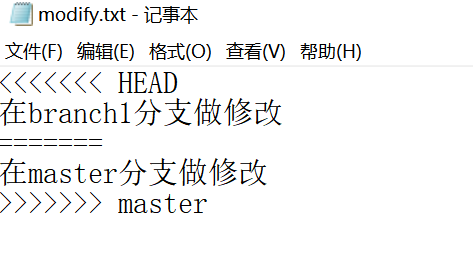
Git checkout -b <新分支名> / Git switch -c <新分支名>：创建并切换到新分支

Git branch:查看所有分支，带’\*’号的是当前分支，即HEAD指向的分支

Get merge <分支名>：合并当前分支和某分支

Git log --graph：可以看分支合并图

如果合并的两个分支都对同一个文件做了修改，则合并后的文件会同时有两个分支的内容



一般，其他人是不会对你处理的文件做修改，你也不应该对其他人处理的文件做修改，只是对别人处理的文件进行读取