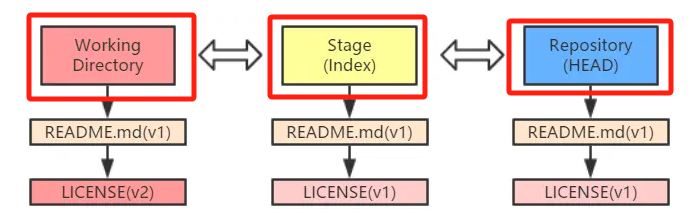
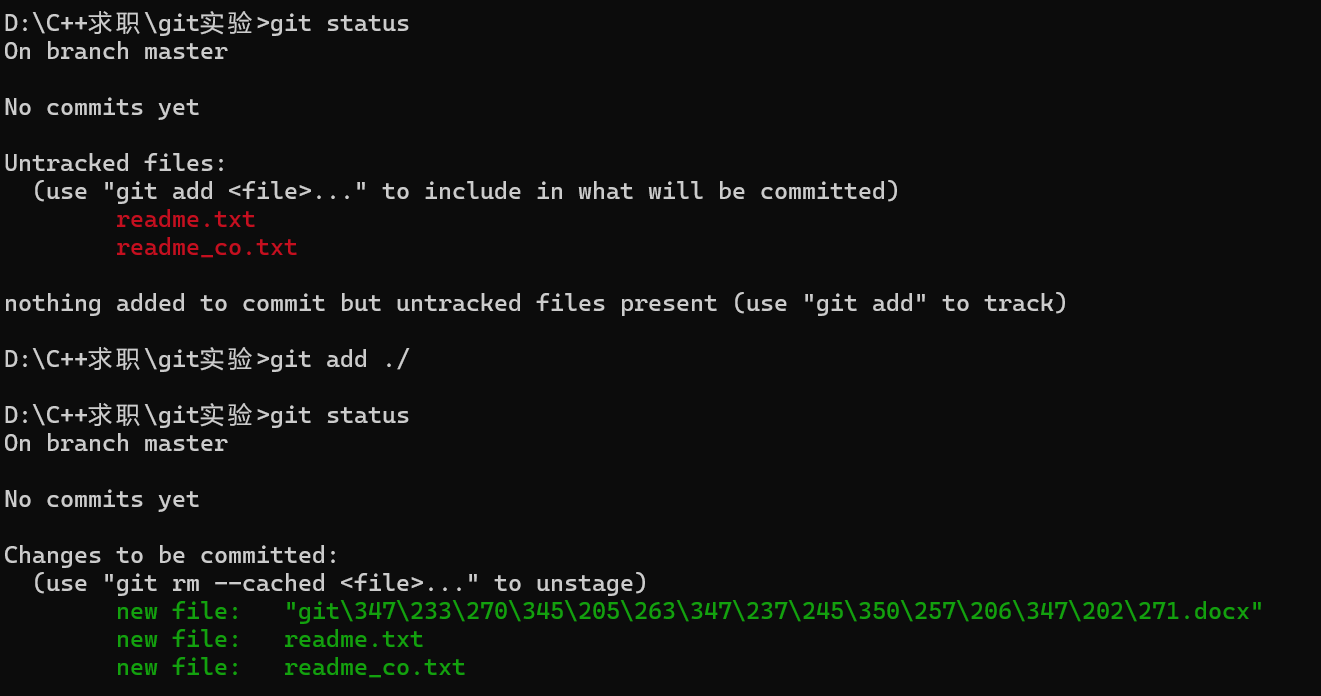
0 先验知识：快照指每一次提交的版本，每个版本称为快照

Git有三棵树，一颗是本地工作区，一颗是暂存区，一颗是仓库Repository

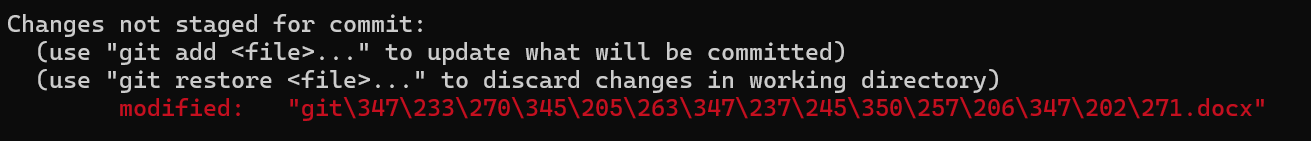


1 git status 查看git repository的状态，包括哪些文件是追踪的，未追踪的文件

2 git init 初始化当前目录的.git文件夹，否则git status检测不到repository的状态



修改了docx文件但是没有add时



每次git add都会将需要将文件的最新状态添加上去/修改

3-1 git reset FILE表示（在没提交之前）撤销FILE的添加/修改，若无FILE则表示撤销上一次的添加/修改，如果上一次是add的多个文件，那就撤销的就是多个文件

git reset HEAD~表示**回退上一次（COMMIT\_ID为A）的提交（用撤销不太合适，因为如果使用git reset COMMIT\_ID(A)还是可以回到的提交版本A的，实质上该命令只是将HEAD的指向改变了）**，即将repository回退至上一次提交的版本，默认是soft版本，不会将暂存区的文件还原到本地工作区中（此时暂存区的文件应该是上一个提交版本在提交时的数据），如果是hard版本的话那本地工作区也会还原到上一次提交的版本

git reset COMMIT\_ID验证了git reset并不意味着撤销/删除之前的一些提交版本，因为通过git reset 之前的提交ID还可以回到那些看似被删除的提交版本，有这么一个场景，如果有提交版本C是从提交版本B的基础上进一步提交的，并且通过git reset B再git log就会发现没有出现C的版本了，如果再在B的基础上提交版本D，git log会显示出版本D，如果此时再用git reset C，然后git log，就会看到C但是没看到D，也就是说C、D是分岔的，因此repository被称为树

git reset COMMIT\_ID FILE 是指回退当前版本的FILE到指定版本的FILE的状态

3-2 git commit提交的是暂存区的信息，必须要有-m选项描述提交，否则会“abort because of empty message”，并且只有有修改才会提交，如果md5检测到没有改变则不会提交，有这么一个场景：如果使用add提交本地工作区到暂存区中，之后再对本地工作区进行修改，那么在提交的时候仅仅会把缓存区中的信息提交上去，而之后本地工作区的修改不会提交，这种情况下可以通过git commit -am "change the license file"一步到位，其中a是add

4 git log 获取直到当前HEAD指向的版本为止提交的历史操作记录

git reflog 获取所有过去的提交的历史操作记录

5 git diff 命令是比较暂存区域与工作目录/本地工作区的文件内容

git diff --cached 命令是比较暂存区域与最新提交的快照

git diff COMMIT\_ID1 COMMIT\_ID2 命令是比较两个快照

git diff COMMIT\_ID1 命令是比较本地工作区和指定快照

~~可以发现默认git是本地工作区占比较一方，而加了—cache选项则是缓存区域占比较一方~~

@@ -1 +1,3 @@ 表示(-代表旧文件，当然也可以认为是删除)旧文件从第一行开始的所有行，新文件从第一行开始的三行有不同

6 git checkout 将（~~最后一次？~~还是HEAD指向？其实应该是HEAD指向，也就是当前提交版本所在的暂存区域，因为要想HEAD可以通过reset绕开比较新的提交并改变指向的提交版本）暂存区域的相关文件恢复到工作目录，如果是git checkout FILE则是恢复指定文件

7 git rm 只会删除本地工作区和暂存区的文件，不会将快照中的文件记录一并删除，因此若想完全删除文件则