把一个软件版本进行分支,在打分支的一瞬间,二者是一样的

查看当前分支 git branch

创建分支 git branch <branchname>

切换分支 git checkout <branchname>

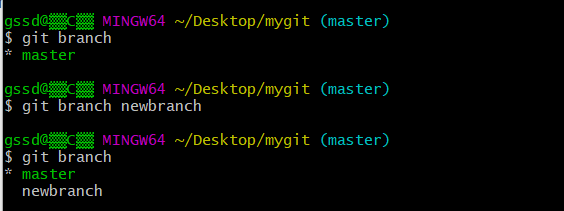
创建分支并切换 git checkout -b <branchName>

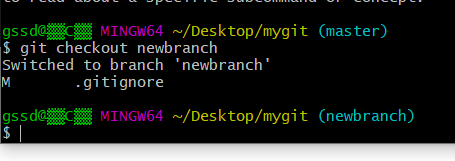
删除分组 git branch -d <branchname> 删除分支 ,自己不能删除自己

其他不能删除的原因 : 1.出现文件未合并

2.新的分支执行了写操作,但是局限在工作区(没有add和commit),如果删除分支a是可以删除的

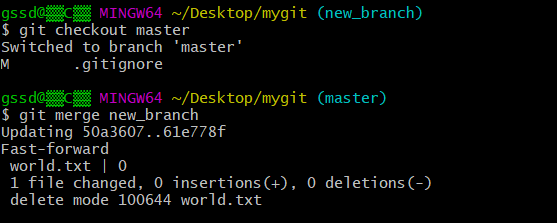
总结: 没有提交的就可以删除分支,提交了就不能删除分支





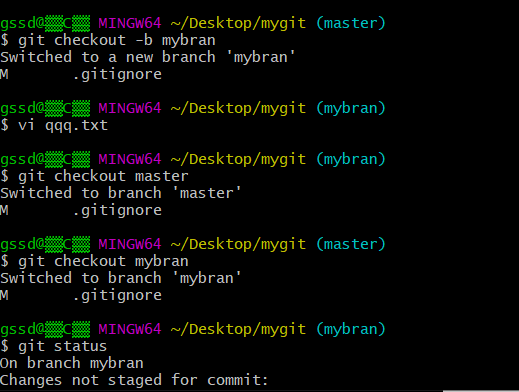
git merge new\_branch 合并分支

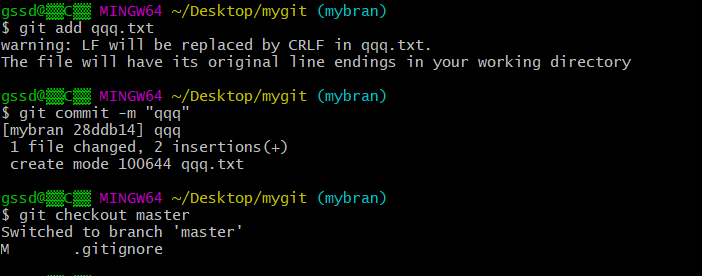
分支合并完了,才可以删除 (删除分支前,建议先合并)



如果在分支a进行了写操作,但是此操作仅在工作区进行,没有进行(add,commit)

在master中能看到此操作,但是如果在分支a进行了commit,则master无法观察到此文件





git branch -v 每个分支最近一次提交的sha1值

分支: 一个commit链.

快照(version),每一次提交之后的所有文件

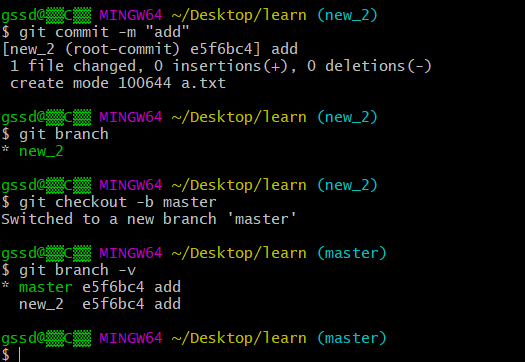
包含:

sha1值,parent(就是上一次提交的sha1值),通过sha1值确定每次提交的路线

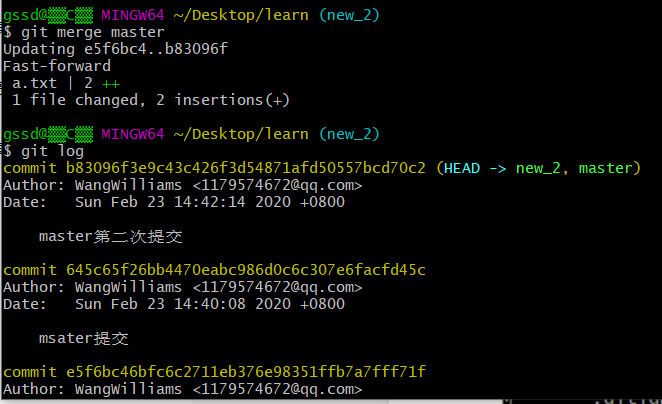
分支是由一系列的sha1值串联起的,通过sha1可以知道分支的存在

分支名只想当前提交的commit,新建一个分支,分支没有修改,则新分支和老分支的sha1值相同

如下图所示:



fast-forward 中间跳跃的一步,645c65开头的那一步



如果有两个分支,一个分支靠前,一个分支落后,如果不冲突,可以通过merge直接追赶上靠前的分支,称作fast forward,挑个的中间commit,仍然会保存.

fast forward 1.两个分支会归为一点commit

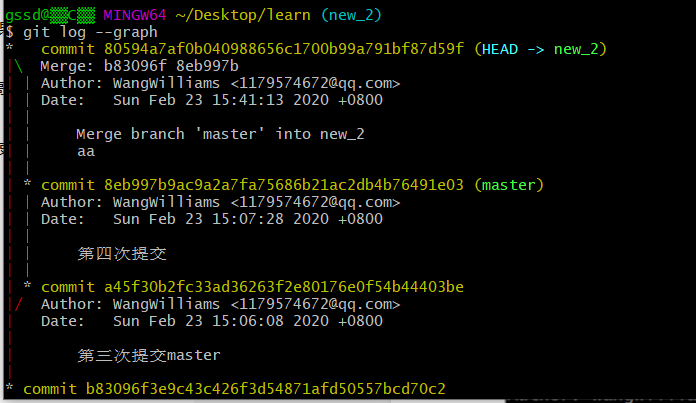
2.没有分支信息(丢失分支信息)

git merge --no-ff <靠前的分支>

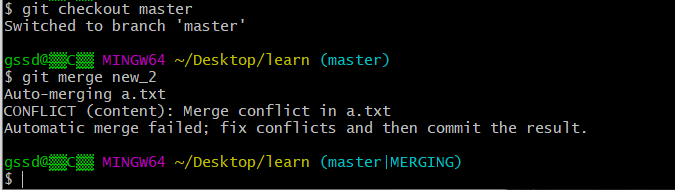
两个分支不会归于一点,(主动发起合并的分支会前进一步)

分支信息完整(不丢失分支信息)

git log --graph

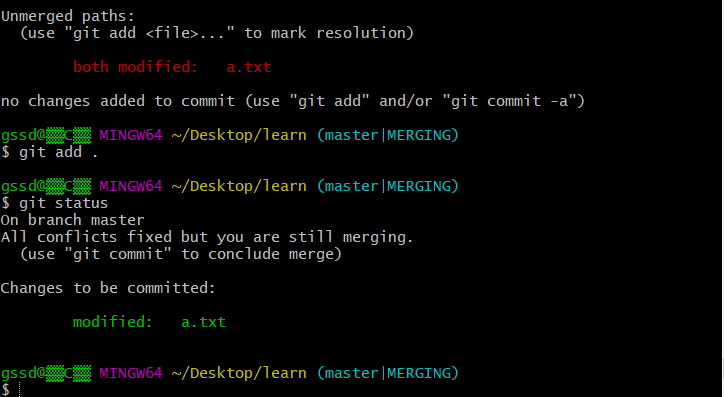


有fast-forward 不保留分支信息,没有fast-forward,保留分支信息



产生冲突,修改文件

vi 修改冲突文件.然后git add .



合并 如果冲突,需要解决冲突

解决冲突:git add xxxx , git commit -m "xx"

git add xxxx(告知git,冲突已经解决)

注意:master在merge时,如果遇到冲突,并解决,则解决冲突会进行两次提交,一次是最终提交一次将对方的dev信息也提交过来了

如果一方落后,另一方前进,则落后方可以直接合并到前进方而不会产生冲突

git log --graph --pretty=oneline --abbrev-commit 只看commit的sha1和commit信息

