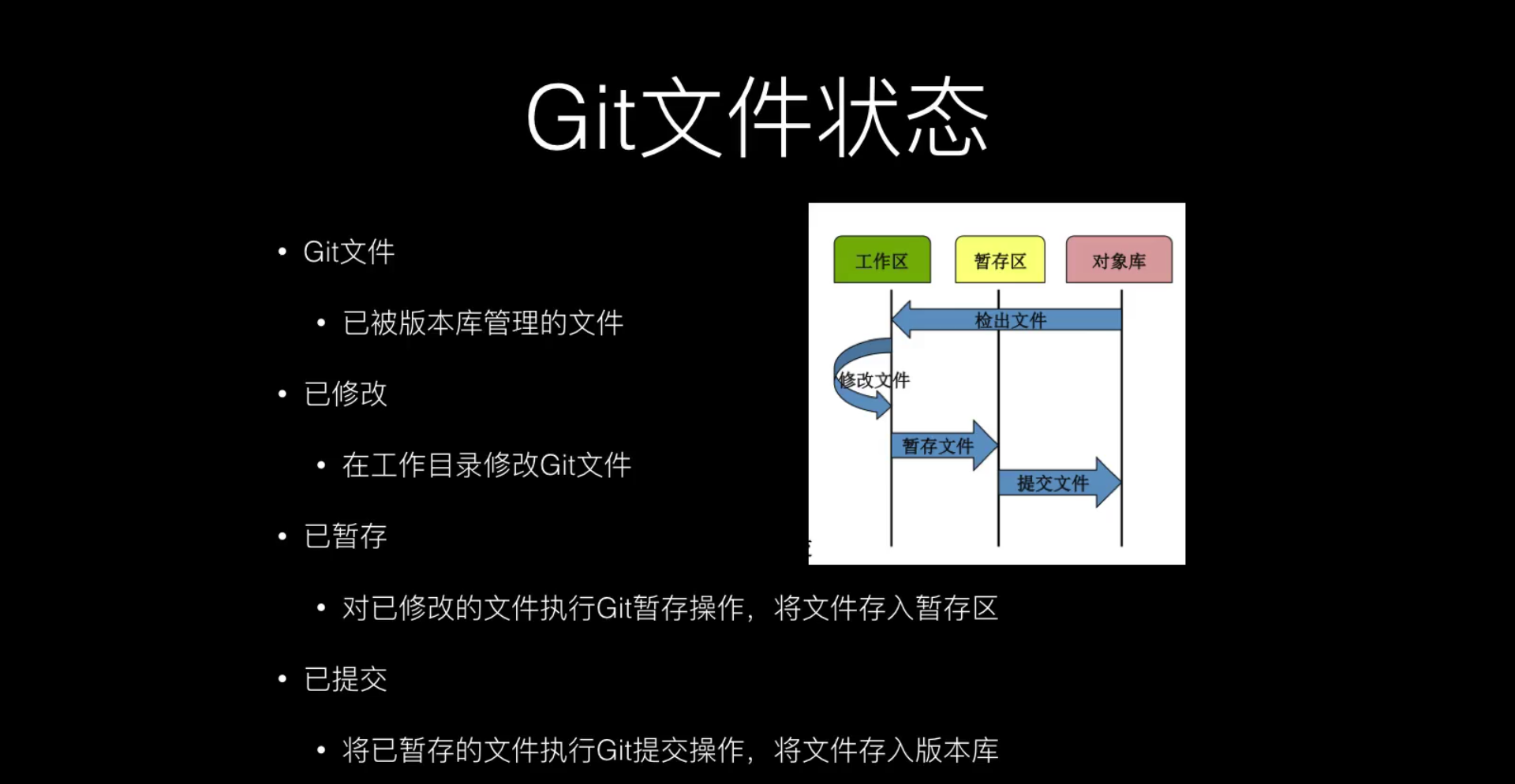
Git基础

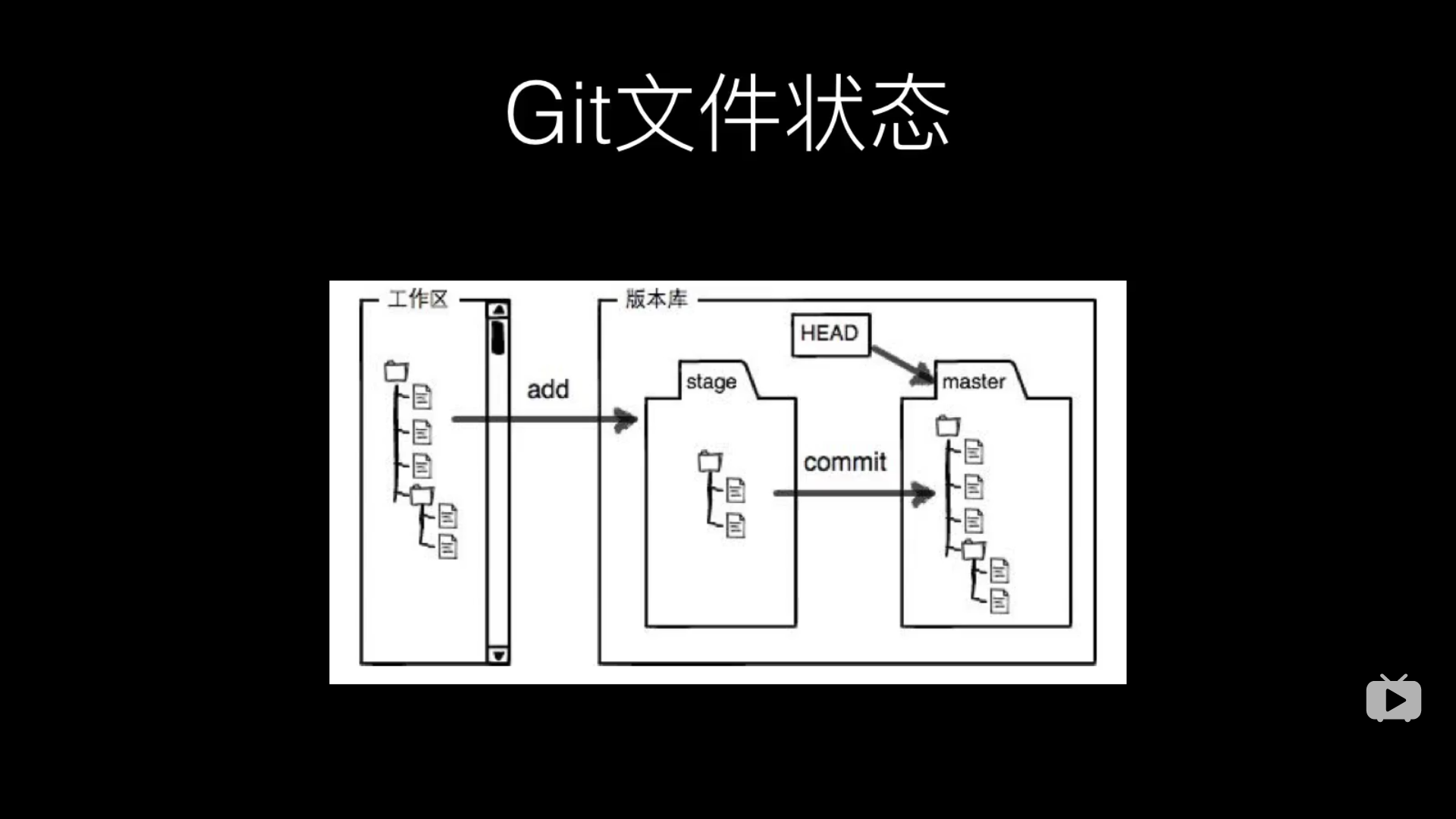
1. 直接记录快照，而非差异比较
2. 几乎所有的操作都在本地执行
3. 时刻保持数据完整性
4. 多数操作仅添加数据
5. 文件的三种状态

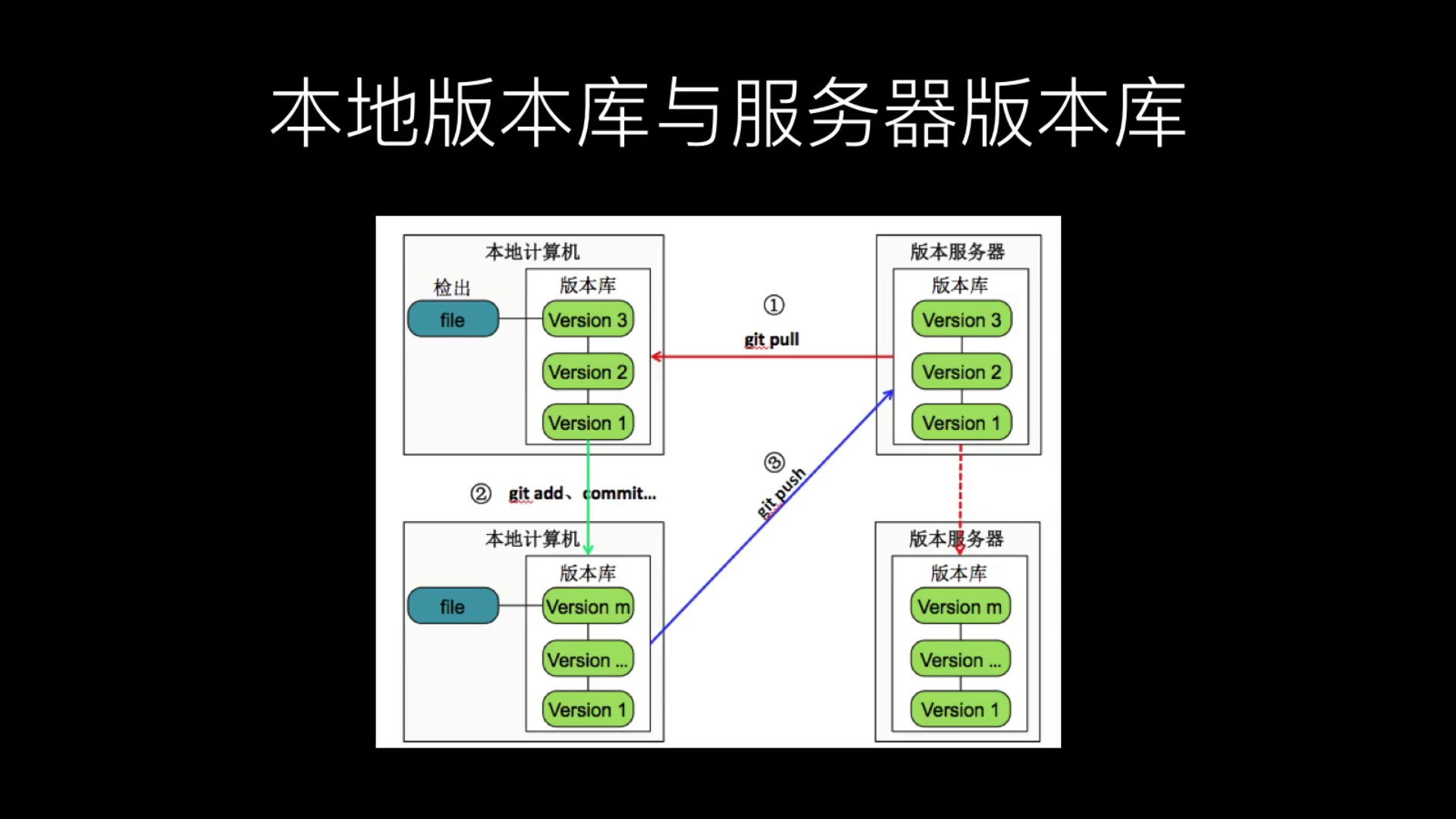
已修改（modified）

已暂存（staged）

已提交（commited）







git rm –-cached 文件名 （实现从暂存区回到已修改的状态）

Git的提交id(commit ID)是一个摘要，这个摘要实际上是通过sha1计算出来的

对于user.name与user.email来说，有3个地方可以设置：

1. /etc/.gitconfig(几乎不会用) （整个计算机的） git config --system
2. ~/.gitconfig(很常用) （针对特定用户） git config --gloabal
3. 特定仓库下/.git/config （针对特定项目） git config –local

git rm:

1. 删除了一个文件
2. 将要删除的文件纳入到暂存区（stage, index）

所以可以马上就git commit -m 将其纳入到版本库当中

若想恢复被删除的文件，需要进行两个动作：

1. git reset HEAD test.txt 将test.txt 从暂存区恢复到工作区
2. git checkout – test.txt 将工作区的修改丢弃掉 真正恢复文件test.txt

系统提供的rm:

仅仅是删除了一个文件，并未将要删除的文件纳入到暂存区（stage，index）

所以还需要自己手动git add 文件名将其纳入到缓存区

git mv 原文件名 新文件

系统mv

它们俩的关系和git rm 与rm 的关系一样

覆盖上次写错的commit消息

git commit --amend -m “修正后的消息”



Doc/\*.txt

Doc/\*/\*.txt 一个\*表示一层

Doc/\*\*/\*.txt 两个\*表示所有层

mkdir mydir && cd mydir

cd - 回到上次那个目录

分支

创建新的分支

git branch 分支名

切换分支

git checkout 你想要切换到的那个分支名

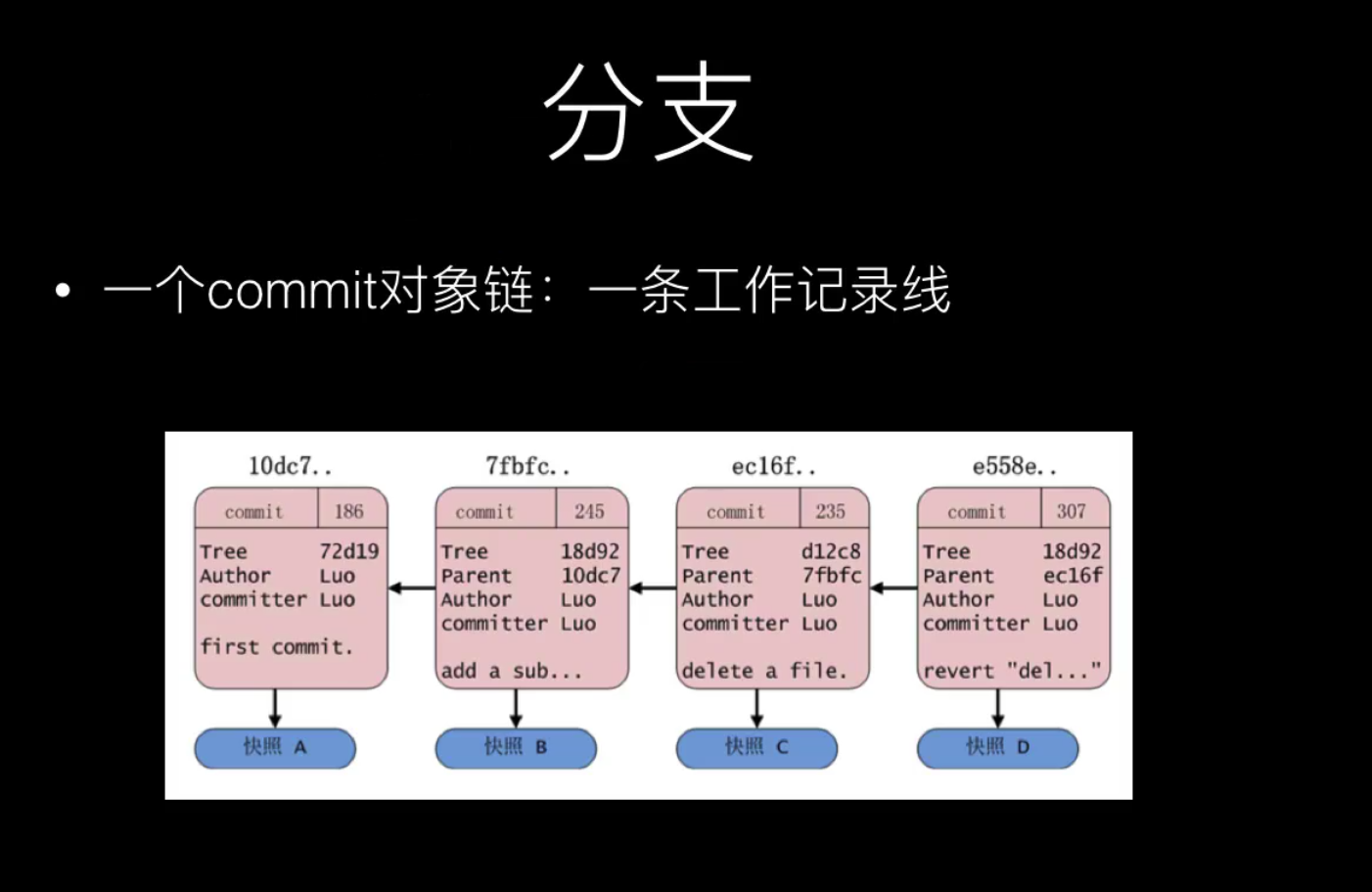
git checkout - 切换到你先前的那个分支

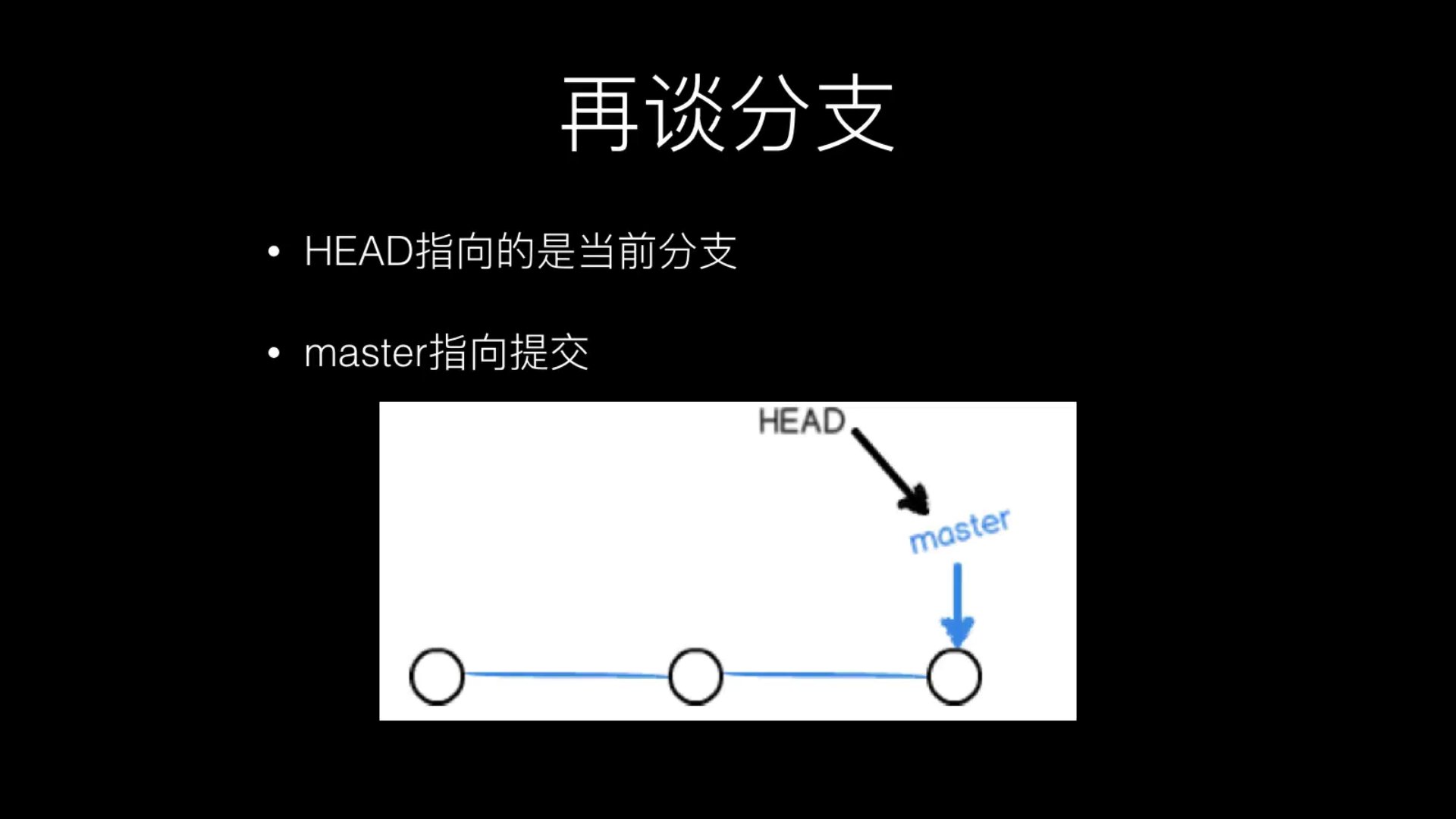
删除分支

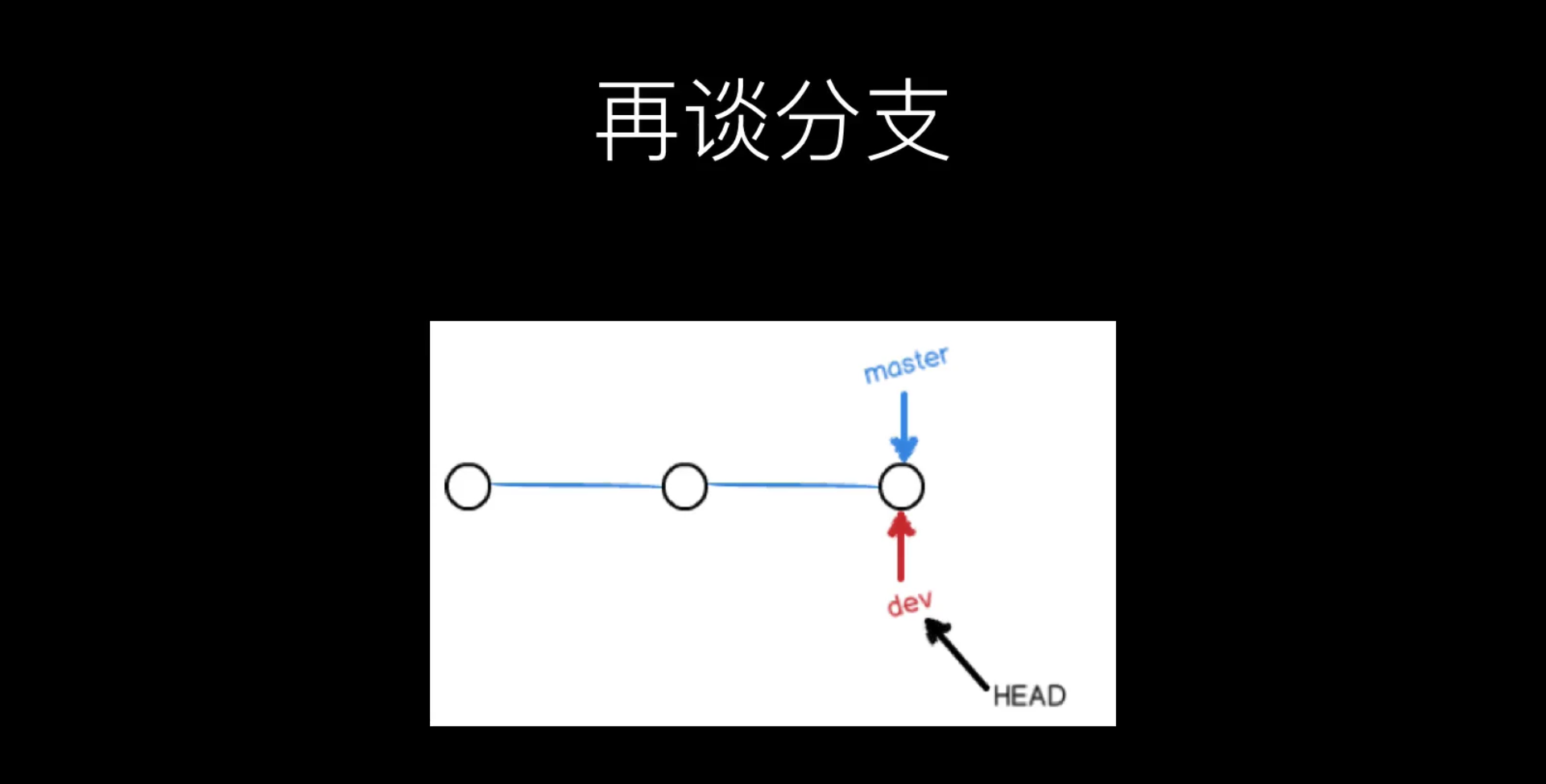
git branch -d 你想要删除的那个分支名

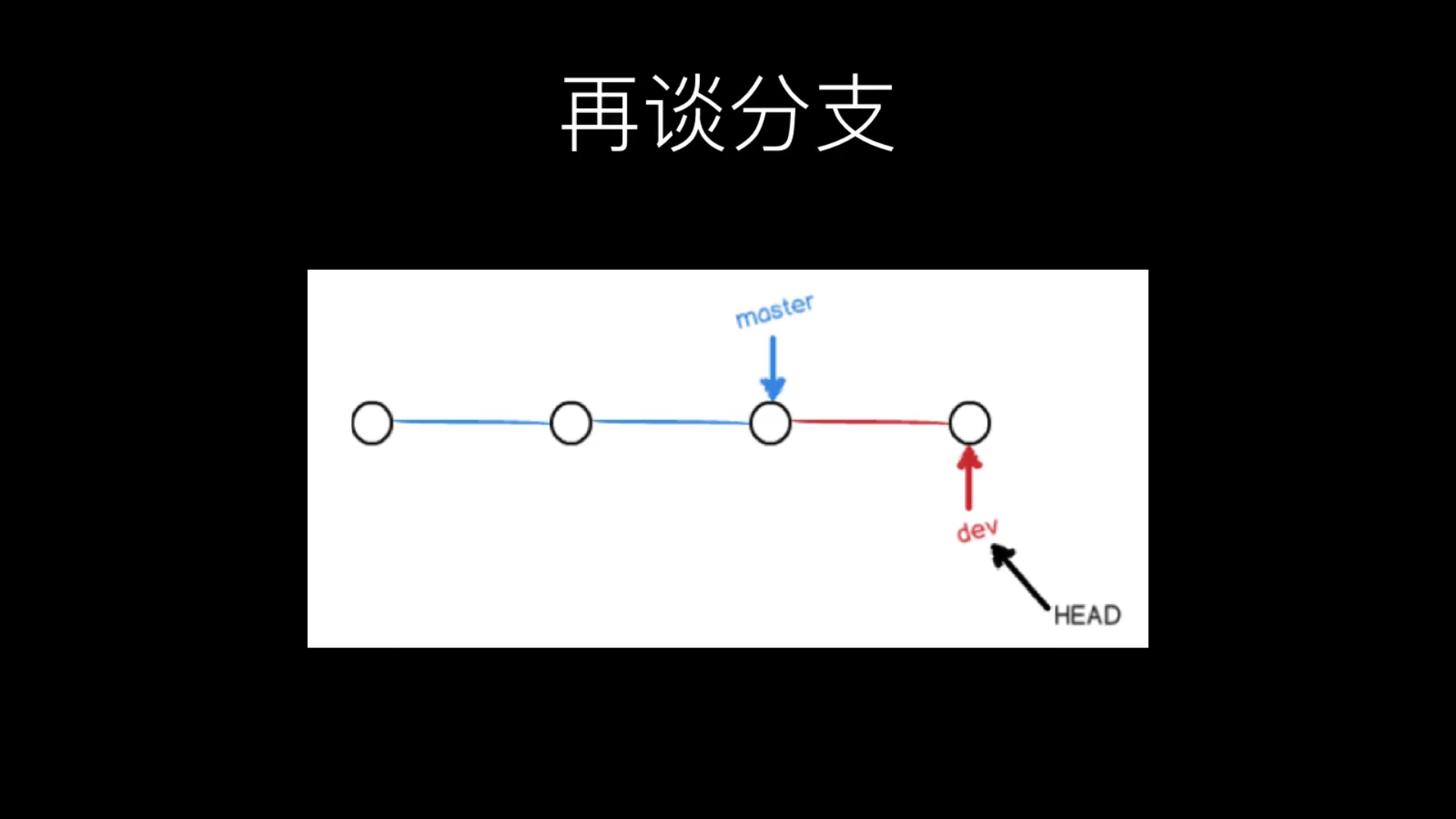
git checkout -b new\_branch 创建分支并切换到新的分支

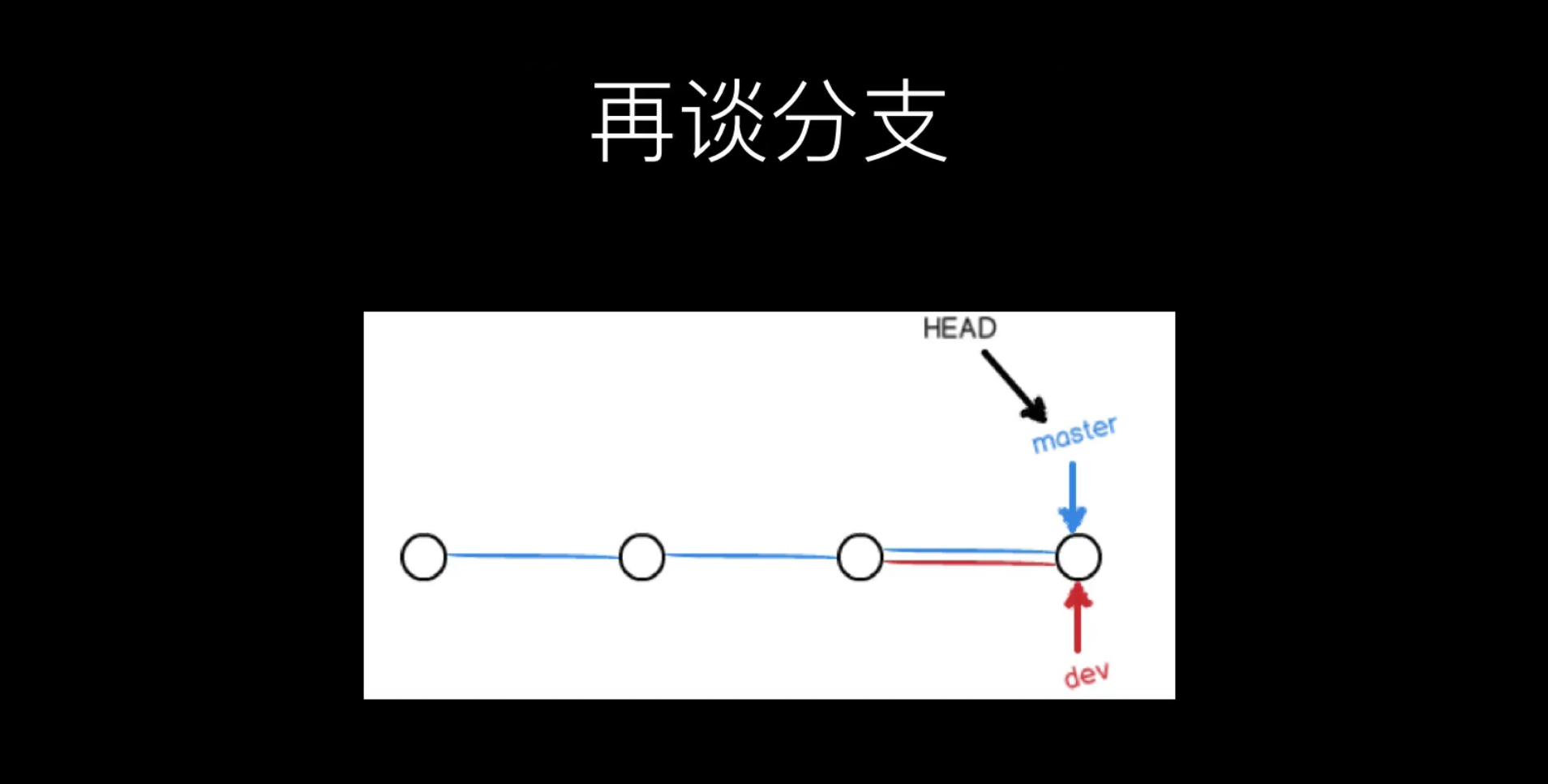
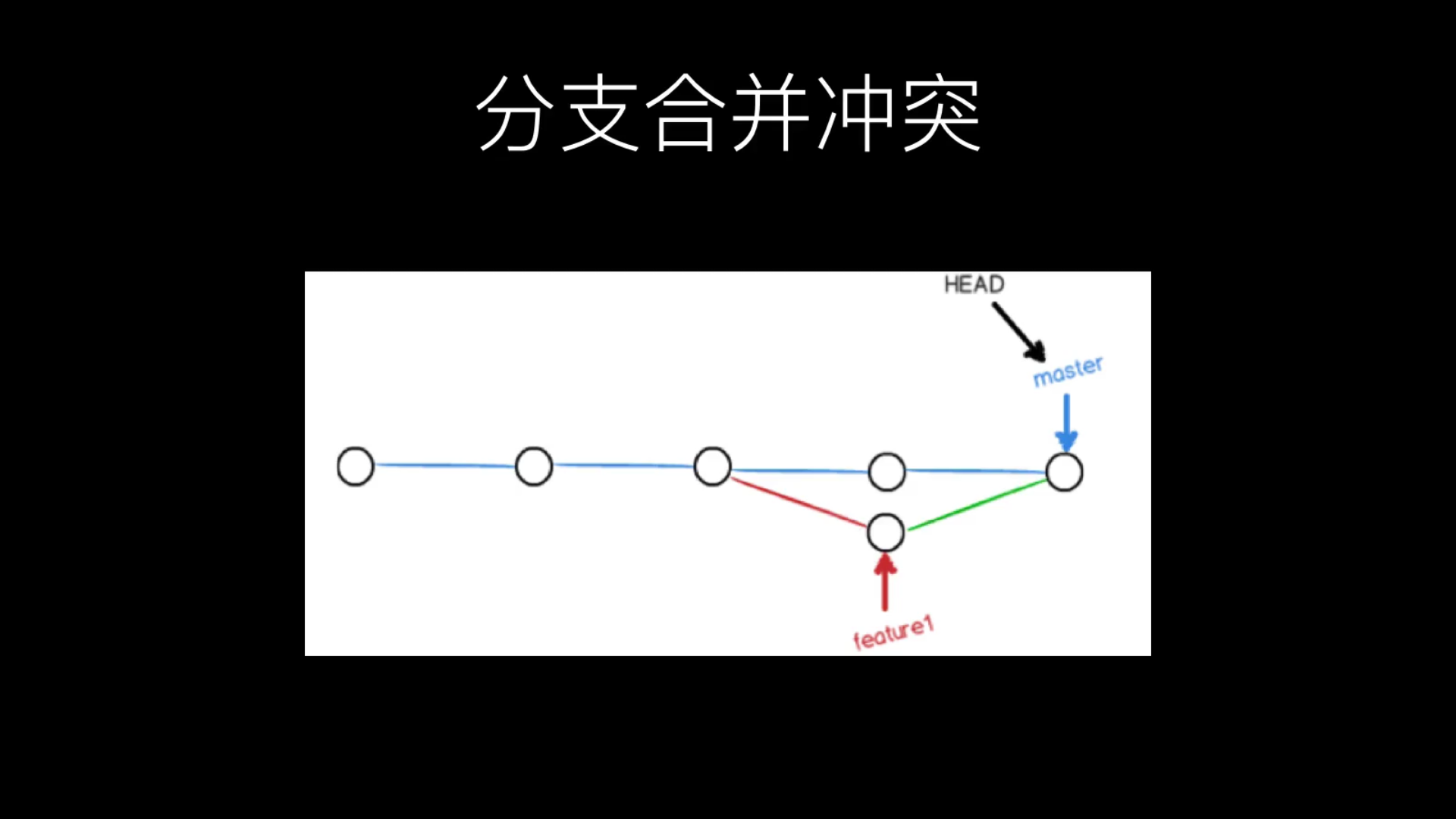
git merge 分支名 将分支名的那个分支合并到当前分支

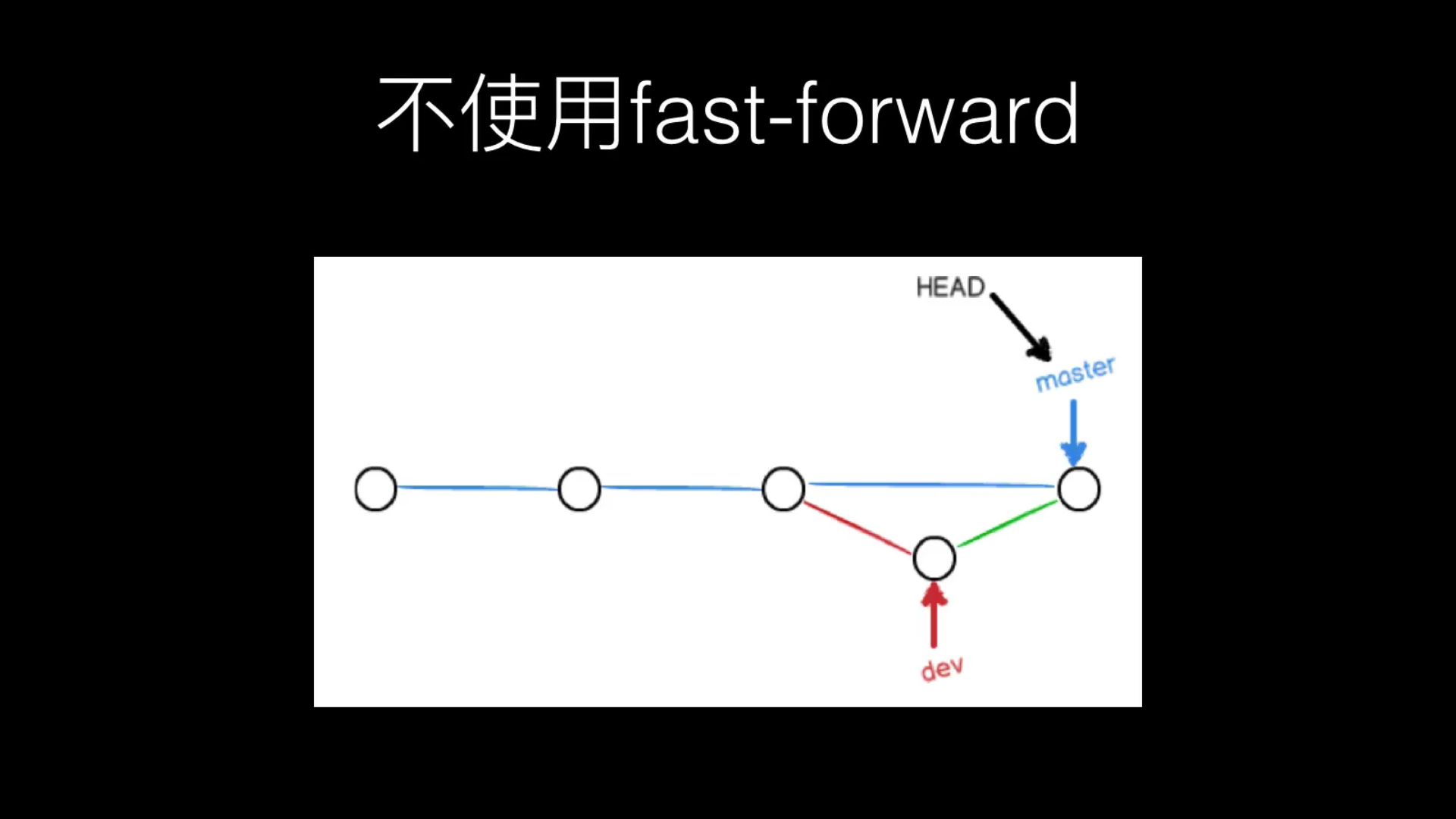
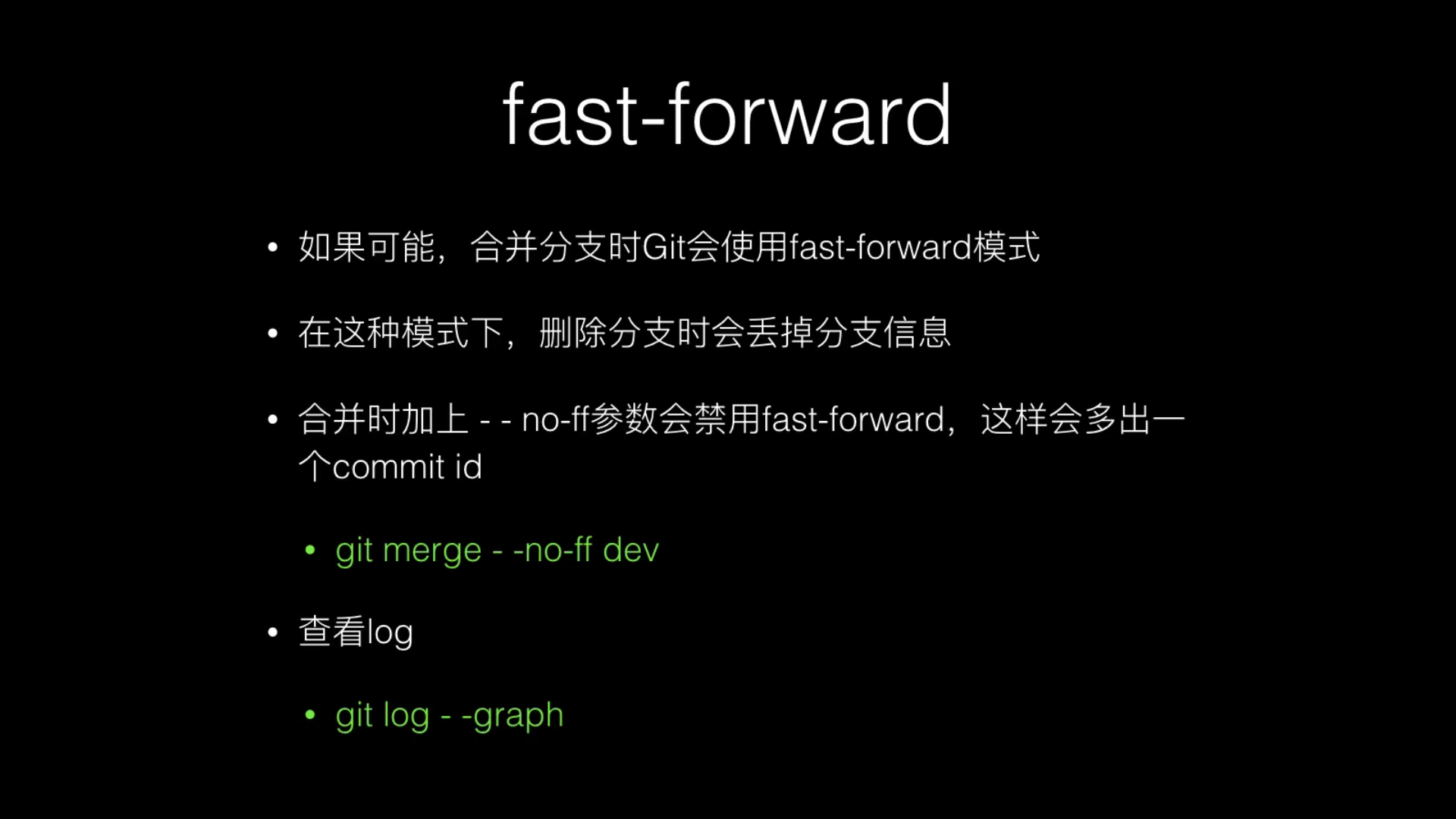










git checkout -- 文件名

作用是：丢弃掉相对于暂存区中最后一次添加的文件内容所做的变更

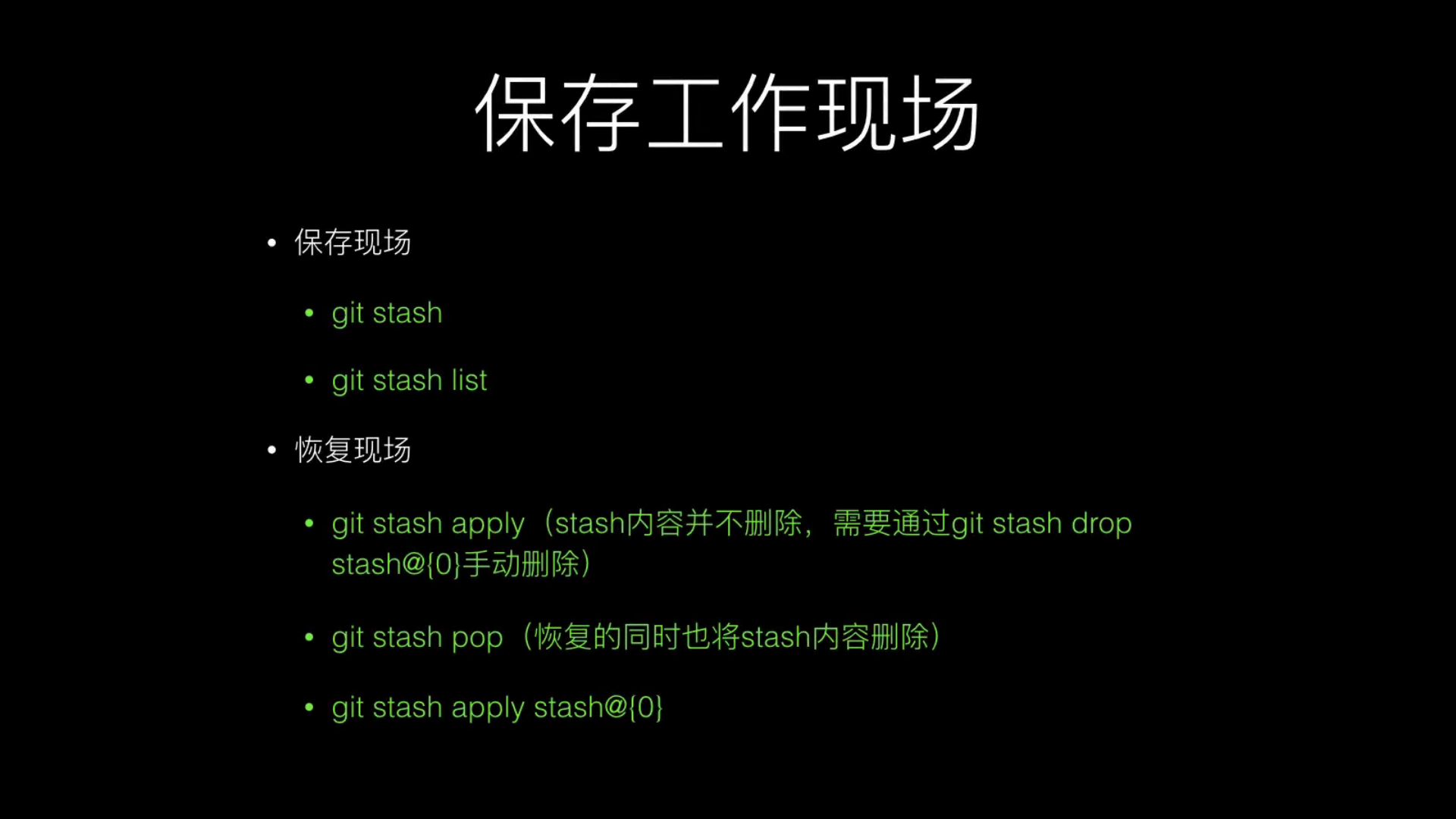
对比暂存区 修改工作区中的那个文件

git reset HEAD 文件名

作用是： 将之前添加到暂存区的内容从暂存区移除到工作区

git branch -m master master2

作用是：把master这个分支名字改为master2





git blame 文件名

查看是哪些人修改过

git diff:  
比较暂存区和工作区的差别

git diff HEAD： （HEAD可以换成任意一个commit\_id）

比较最新的提交（版本库）和工作区的差别

git diff –cached:

比较最新的提交（版本库）和暂存区的差别