`git commit`—提交

`git branch 分支名`—创建新分支

`git checkou 分支名`—切换到该分支

`git checkout -b 分支名`—创建该分支的同时切换到该分支上

`git merge 分支名`—将该分支合并到所在分支

`git rebase 分支名`—将所在分支的工作移接到该分支上（实际上还是并行执行）（不能向下走）

`cat .git/HEAD`—查看HEAD指向

`git symbolic-ref HEAD`—查看HEAD指向的引用

可以用切换到某个提交记录上来分离HEAD

`git log`—查看提交记录的哈希值

可以用标识提交记录的前几个字符来处理哈希值

`git checkout 分支名或节点名^`— 向上移动一个节点（一个^代表一个节点）

`git checkout 分支名或节点名~num`—向上移动n个节点（不跟数字时与^相同）

`git branch -f 分支名 节点名`—将该分支移动到此节点上

`git branch -f 分支名 HEAD~num`—将该分支向HEAD的第n个父节点提交（使用前需切换到该分支上）

`git reset 分支名`—撤销（单人使用）

`git revert 分支名`—撤销（远程使用，同时会生成一个新的提交记录）

两者后均可加~或^

`git cherry-pick 节点名`—将节点转移到所在节点

`git rebase -i HEAD~num`—将所在节点以上n个节点进行调整，包括此节点

`git commit --amend`—对所在节点进行修改

`git tag 标签名 节点名`—给指定节点打上标签（如果不指定则会指向HEAD）

`git describe 节点名`—输出离此节点最近的标签名*距离*g该节点名

`git clone`—创建远程仓库

`git fetch`—从远程仓库中进行下载

`git pull`—fetch和merge的结合

`git push`—上传至远程仓库

`git pull --rebase`—pull和rebase的简写

`git push origin 分支名`—将该分支上传至远程仓库

`git checkout -b 分支名 o/main`—可以创建一个分支追踪o/main

`git branch -u o/main 分支名`—同上（但是必须先创建分支）

`git push origin 分支名`—将该分支更新到远程仓库

此指令可删除远程仓库中的分支

`git push origin 本地文件分支名`：远程仓库分支名—将本地文件的分支更新至远程仓库中的指定分支中（可加^或~）

`git fetch origin 分支名`—将远程文库的该分支更新到本地文件

`git fetch origin 远程仓库分支名`：本地文件分支名—将远程仓库的分支更新至本地文件中的指定分支中（可加^或~）

此指令可在本地文件中创建一个该分支

`git pull origin 远程仓库分支名`—将该分支更新至本地文件并在当前检出的位置将其合并

`git pull origin 远程仓库分支名`：本地文件分支名—在下载的基础上在进行合并