# git分支本质

分支本质是一个提交对象，所有的分支都会有机会被HEAD所引用（HEAD同时只能只想一个分支）

当我们有新的提交时，HEAD回携带当前持有的分支往前移动

# git分支命令

创建分支：git branch branchname

切换分支：git checkout branchname

创建&切换分支：git checkout -b branchname

版本穿梭（时光机）：git branch branchname commitHash

普通删除分支：git branch -d branchname

强制删除分支：git branch -D branchname

合并分支：git merge branchname

快进合并 --> 不会产生冲突

典型合并 --> 有机会产生冲突

解决冲突 --> 打开冲突文件，修改，再add、commit

查看分支列表：git branch

查看合并到当前分支的分支列表：git branch --merged

一旦出现再这个列表中，就应该删除

查看没有合并到当前分支的分支列表：git branch --no--merged

一旦出现在这个列表中，就应该观察以下是否需要合并

# git分支的注意点

在切换的时候，一定要保证当前分支是干净的！！！

允许切换分支：

分支上所有的内容处于：已提交状态

（避免）分支上的内容是初始化创建 处于未跟踪状态

（避免）分支上的内容是初始化创建 第一次处于已暂存状态

不允许切分支：

分支上所有的内容处于已修改状态 或 第二次以后的已暂存状态

在分支上的工作做到一半时，如果有切换分支的需求，我们应该将现有的工作存储起来

git stash：会将当前分支上的工作推到一个栈中

分支切换 进行其他工作 完成其他工作后切换回原分支

git stash apply：将栈中的工作内容还原

git stash drop：删除栈中还原的工作内容

git stash pop：git stash apply + git stash drop

git stash list：查看存储

# 后悔药（撤销、重置）

撤销工作目录的修改：git checkout --filename

撤销暂存区的修改：git reset HEAD filename

撤销提交：git commit --amend

# reset三部曲

git reset --soft commithash -----> 用commithash的内容重置指针HEAD

git reset [--mixed] commithash -----> 用commithash的内容重置HEAD及暂存区

git reset --hard commithash -----> 用commithash的内容重置指针HEAD、暂存区、工作目录

# 路径reset

所有的路径reset都要省略第一步！！！

* 第一步是重置HEAD内容，但是HEAD本质指向一个分支，分支的本质是一个提交对象；
* 提交对象指向一个树对象，树对象可能指向多个git对象，一个git对象代表一个文件！！！
* HEAD可以代表一系列的文件状态！！！

git reset [--mixed] commithash filename

用commithash中filename的内容重置暂存区

# checkout深入理解

git checkout branchname 跟 git reset --hard commithash特别像

共同点：都需要重置HEAD 暂存区 工作目录

异同点：checkout对工作目录是安全的 reset --hard是强制覆盖

checkout动HEAD时不会带着分支走而是切换分支

reset --hard时是带着分支走的

git checkout + 路径：

git checkout commithash filename

重置暂存区

重置工作目录

git checkout filename

重置工作目录