Git 分支的本质

分支本质上是一个提交对象，所有分支都有机会被 HEAD 所引用（HEAD 一个时刻只会指向一个分支）
当我们有新的提交的时候 HEAD 会携带当前持有的分支往前移动

Git 分支

创建：**git branch branchname**
切换：**git checkout branchname**
创建&切换：**git checkout -b branchname**
版本穿梭（时光机）：**git branch branchname commithash**
普通删除：**git branch -d branchname**
强制删除：**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
    解决冲突 → 打开冲突文件进行修改、add 、commit
查看分支列表：git branch
查看合并到当前分支的分支：**git branch –merged**
    一旦出现在这个列表中，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，就应该观察一下是否需要合并

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 、 完成其他工作后切回原分支
    git stash apply ： 将栈顶的工作内容还原，但不让任何内容出栈
    git stash drop ： 取出栈顶的工作内容后就应该将其删除（出栈）
    git stash pop ： git stash apply 与 git stash drop 的结合
    git stash list ： 查看存储

后悔药

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（仅仅修改注释）
    git commit –amend

reset 三部曲

soft → 重置 HEAD
    git reset --soft commithash ： 用 commithash 的内容重置 HEAD 内容
mixed → 重置 HEAD、暂存区
    git reset [--mixed] commithash ： 用 commithash 的内容重置 HEAD 内容、重置暂存区
hard → 重置 HEAD、暂存区、工作目录
    git reset --hard commithash ：用 commithash 的内容重置 HEAD 内容、重置暂存区、重置工作目录

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，HEAD 本质是指向一个分支，分支的本质是一个提交对象
- 提交对象指向一个树对象，树对象又很有可能指向多个 git 对象，一个 git 对象代表一个文件!!
- HEAD 可以代表一系列文件的状态

git reset [--mixed] commithash filename
    用 commithash 中 filename 的内容重置暂存区

对 checkout 的深入理解

git checkout branchname 与 git reset –-hard commithash 特别像
    共同点：都需要重置 HEAD、暂存区、工作目录
    区别：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

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 n