

## 创建版本库

mkdir learngit -> cd learngit -> pwd 显示当前目录

git init 把这个目录变成Git可以管理的仓库

把一个文件放到Git仓库: ① git add 添加到暂存区

② git commit 提交到分支 -m " " 双引号为是提交的说明

git status 查看当前状态

git diff 查看修改了什么内容

## 版本回退

git log 查看历史记录 加上 --pretty=oneline 参数, 信息呈现更简洁

HEAD表示当前版本, 上一个版本就是 HEAD^, 以此类推

git reset --hard HEAD^ 回退到上一个版本. 或者用 commit id 的版本号, 回退到指定版本

cat + 文件名 查看文件内容.

git reflog 记录每一次命令

## 工作区和暂存区

版本库里有暂存区(stage/index)、分支、指针

每次修改, 如果不用 git add 到暂存区, 那就不会加入到 commit 中

## 撤销修改

git checkout -- 文件名. 把该文件在工作区的修改全部撤销, 让其回到最近一次 git add 或 git commit 的状态.

git reset HEAD 文件名. → 把暂存区的修改撤销(unstage)

## 删除文件

在文件管理器中或者用 rm 命令删除 rm 文件名.

然后从版本库删除该文件 git rm 文件名. -> git commit -m " "

注: 从来没有被添加到版本库就被删除的文件, 是无法恢复的.

## 远程仓库

① 创建 SSH key. ssh-keygen -t rsa -C "邮件地址"

② 登陆 GitHub. Add SSH Key. 在 Key 文本框里粘贴 id\_rsa.pub 文件内容.

## 添加远程库

登陆 GitHub. 创建一个新的仓库

在本地的仓库下运行命令 git remote add origin git@github.com:Lichunyan3/库名

git push -u origin master 把本地库所有内容推送到远程库上.

## 删除远程库

git remote -v 查看远程库信息.

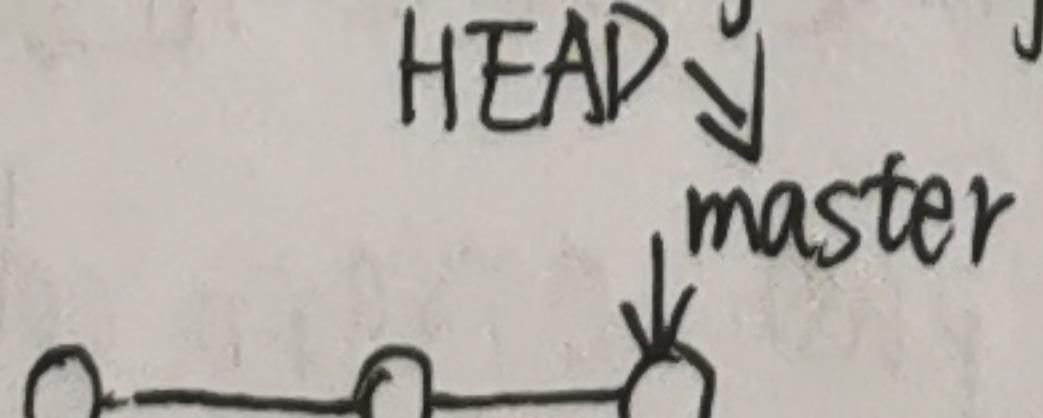
git remote rm <name> 根据名字删除

## 从远程库克隆

登陆 GitHub 创建一个新仓库

git clone git@github.com:Lichunyan3/库名.git. 克隆一个本地库

## 分支管理 创建与合并分支



git checkout -b dev 创建并切换到dev分支，相当于git branch 和 git checkout

git branch 查看当前分支 \*

git ~~branch~~ checkout master 切换回master分支

git merge dev 把dev分支的工作成果合并到 master 分支上(合并指定分支到当前分支)

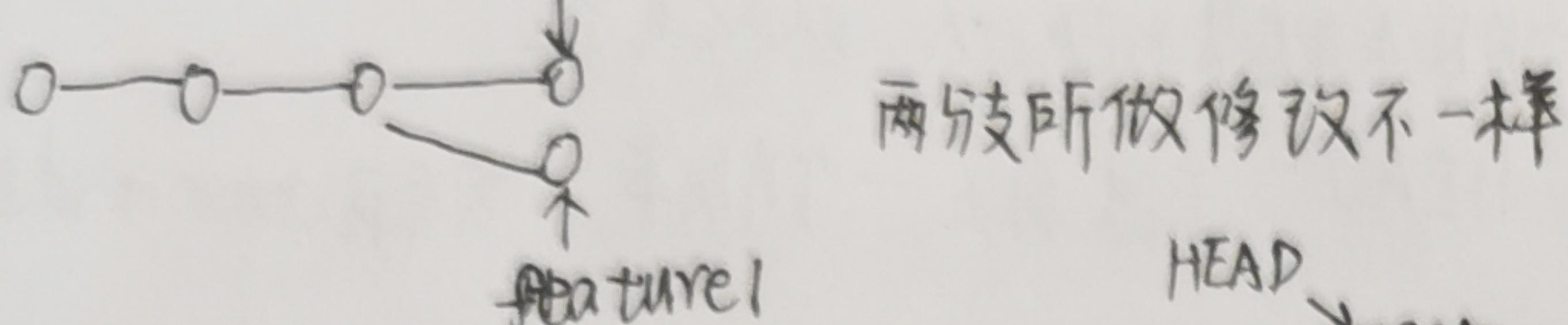
git branch -d dev 删除dev分支

git switch -c dev 创建并切换到新的dev分支

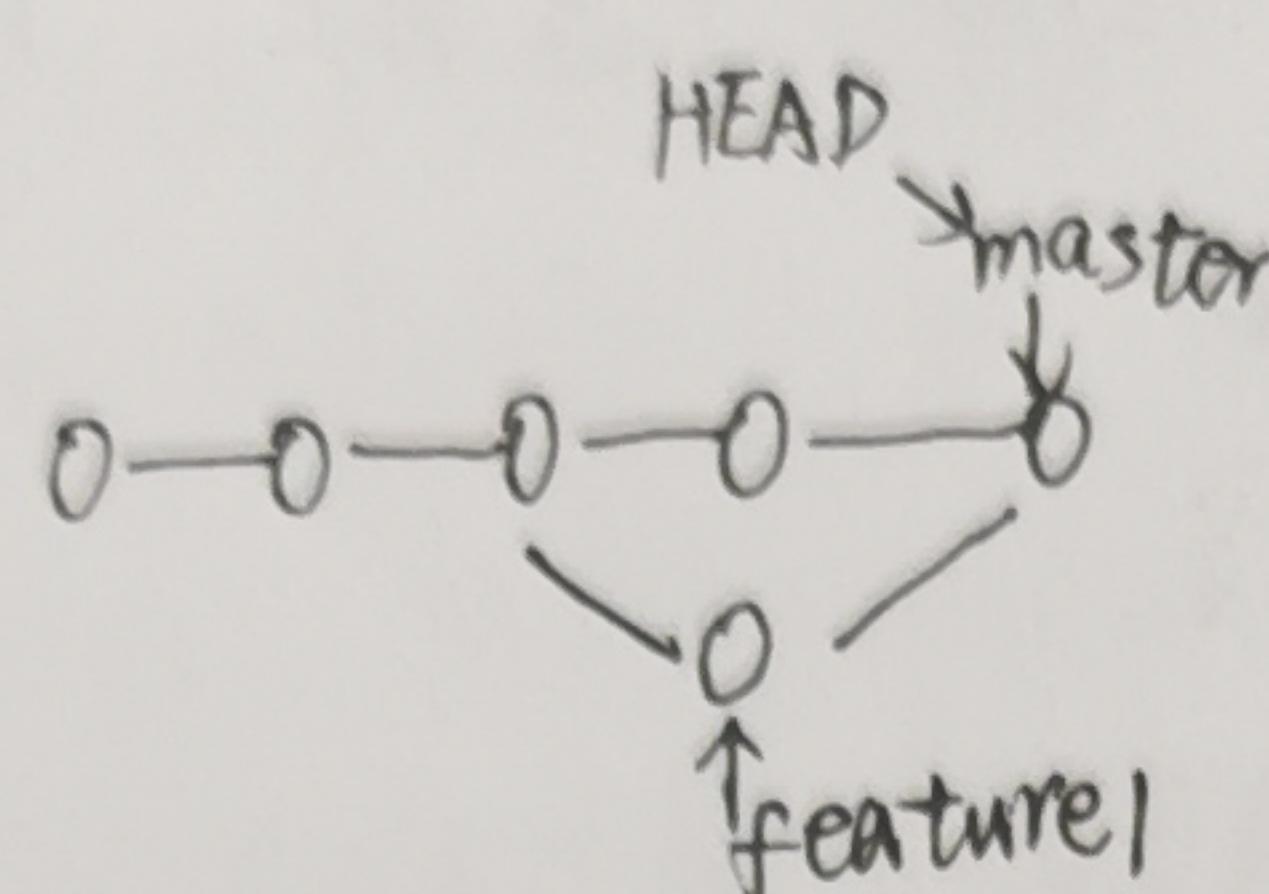
git switch master 切换到已有的master分支

HEAD → master

## 解决冲突



对文件修改后保存再提交。



~~git log --graph~~ git log --graph --pretty=oneline --abbrev-commit

## 分支管理策略

git merge --no-ff -m "描述" dev 禁用Fast forward模式，并生成一个新的commit

## Bug 分支

git stash 把当前工作现场(如未提交修改)“储藏”起来，等以后恢复现场后继续工作

假定在master分支上修复bug，就从master创建临时分支。

git checkout master → git checkout -b issue-101 → 修复提交后 → git switch master  
→ 合并 git merge ...

回到dev分支干活 git switch dev. ~~git~~

git stash list 查看工作现场储存位置。

git stash pop 恢复工作现场并删除 stash 内容

若多次stash，git stash apply stash@{0} 恢复指定的stash git stash drop 删除  
dev分支也存在同的bug.

git cherry-pick 4c805e2(修改提交的代号) 复制一个特定的提交到当前分支，但是是两个不同的commit.  
即commit id不同。

创建了一个分支 feature-vulcan

在 feature 分支准备和 dev 分支合并时，需将 feature 分支销毁。

git branch -d feature-vulcan. — 销毁 ~~失败~~

git branch -D feature-vulcan. 强行删除

git remote 查看远程库的信息。加上 -v 显示更详细的信息

git push origin master 推送分支。 git push origin dev 推送 dev 分支

另一人从远程库克隆只能看到本地的 master 分支 <sup>他在</sup> 在 dev 分支上开发必须创建远程 origin 的 dev 分支到本

## Feature 分支

## 多人协作

地：git checkout -b dev origin/dev 然后修改提交 push 到远程.

碰巧自己也对同样的文件作了修改并提交推送会发现失败.

解决方法：① git ~~branch~~ branch --set-upstream-to=origin/dev dev 设置 dev 和 origin/dev 的连接

② git pull 把最新的提交从 origin/dev 拉下来

先本地合并③ 解决冲突再提交然后 push

Rebase git rebase 将分叉的提交变成一条直线

标签管理 标签是版本库的一个快照... 跟某个 commit 绑在一起

创建标签 先切换到要打标签的分支上. 然后 git tag <标签名>. 打一个新标签

git tag 查看所有标签. 默认标签是打在最新提交的 commit 上.

git tag <标签名> <commit id> 给指定提交打标签.

git show <标签名> 查看标签信息

git tag -a <标签名> -m "说明文字" <commit id>

git tag -d <标签名> 删除标签.

git push origin <标签名> 推送某个标签到远程.

git push origin --tags. 推送全部未推到远程的本地标签.

若已推送到远程，删除步骤：① git tag -d <标签名>

② git push origin :refs/tags/<标签名>

配置别名 git config --global alias.st status. 将 st 作为 status 的缩写.