```
工作区 Working Directory —— 在电脑上就能看到的目录
                                                                                                               但是不算做工作区
                                                                                                  一 在工作区里面的一个隐藏的目录 .git
            基础知识 —— 工作区、暂存区、版本库 — 版本库 Repository
                                                                                                                        - 作用 — git add 把文件的修改添加到了 暂存区中
                                                                                     - 结构
                                                                                                                        dev —— 自己建的 —— 在上面工作
                                                                                                     一个HEAD指针 —— 指向switch 到的branch
                                             文件夹是工作区
                                            进入文件夹 —— cd learngit
                                             建文件夹 —— mkdir learngit
                                             显示当前目录 —— pwd
                              初始化
                                             把目录变成repository
                                                                                     在当前的目录下生成了一个隐藏的.git 目录 —— 是Git 用来跟踪管理版本的库
                                                                     · 文件首先要放在工作区里面,即放在learngit 这个文件夹里面
                                                                     · 然后通过git add fileName 将文件的修改放入stage 即暂存区当中 —— 这个过程可以重复多次
                                                                  ──最后使用 git commit -m "write the message here " —— 使得放在暂存区里的所有修改 被当前所在分支采纳
                                                     查看当前的branch 的状态 —— git status
                                                     查看从最近到最远的提交日志 —— 好看一点 —— git log --pretty=oneline git log --pretty-oneline --abbre
                                                                                       — 注意 —— 其中大串的数字是commit id ,即版本号 —— 每一次commit 都有一个id
                                                                                                                                                      指向当前分支,即branch
                                                     查看工作区里面的文件和当前分支中的对应文件有什么不同
                                                                                                                   git diff HEAD -- fileName
                                                                                                                          git reset --hard HEAD^
                                                                                      使用HEAD
                             对修改进行操作
                                                                                                       再往前一个 —— git reset --hard HEAD^^
                                                     退回到历史的某一个版本
                                                                                                    └ 往前100个 --- git reset --hard HEAD~100
                                                                                   — 使用commit id —— git reset --hard commit_id —— 异中Commit_id
Git 会自动去找
                                                                                                                                               其中commit id 可以在log中看,并且不必全部输入,只需要输入前几位就可以了,
                                                                                     如果reset 了之后通过git log 也看不到了
                                                                                                — 即使reset 了,也可以看到某一个历史版本
— 可以通过这样查看commit id —— 来进行git reset --hard commit_id
                                                                                          一 如果没有git add 过 —— git restore —— 那么就回到和版本库里一模一样的状态
                                                                                            如果已经git add 到了暂存区里面 —— git restore --staged fileName —— 那么就回到unstaged 的状态
                                                                  — git —— git rm fileName —— 然后提交更改 —— git commit -m "remove file ......"
                                             我们本地的仓库中加一个远程的库 —— create 完了一个repository 后按照提示在 cd 到想要上传的仓库后 粘贴 —— $ git remote add origin git@github.com:用户名/learngit.git
                                                                                     一创建SSH Key —— 先看一下C盘的用户adminstrator 里面有没有.ssh 目录 —— 如果有看一下有没有id_rsa 和 id_rsa.pub 如果没有 —— 打开Git Bash —— $ ssh-keygen -t rsa -C "youremail@example.com" —— 输入一个邮件地址 然后一直回车就行了
                                            与github相连 —— 步骤
                                                                                     ─ 打开.ssh 里面的id_rsa.pub  —— 打开github 在settings 中找到SSH Key 的设置  —— 在Key里面粘贴 id_rsa.pub 的内容
                                                                                       git push origin branchName
Git
                                             把当前分支推到github连着的库里面
                                                                                        如果是第一次推送,并想要把本地的master分支和远程的master 分支关联起来 —— git push -u origin master
                                              clone 之后就可以直接push 了
                                                                                                                                                                 按一下,把地址复制下来
打开Git Bash 后不要cd 到其他地方 —— 直接 git clone ...... —— ...... 处粘贴复制的地址
                              远程库
                                            把github上面的库克隆到本地 —— 在your repositories 里面的库 —— 可以找到clone or downlowd 绿色的键
              具体操作
                                                                  从远程库克隆后只能看到本地的master
                                            创建远程的origin 的dev 到本地
                                                                                 — git checkout -b dev origin/dev —— 也可以git switch -c dev origin/dev
                                                                                 — 要推送到远程的dev分支中的时候 —— git push origin dev
                                                              后push 的同学要先pull
                                            当你push 的时候发现你的当前的branch 已经behind 了 —— 但是要先把本地的dev 和远程的origin/dev 链接起来才能往里面拿东西 —— git branch --set-upstream-to=origin/dev dev 可以git pull 了 —— 子主题
                                                             可以想象成有一条路径,每一次commit 都会做一个站点,每个站点都有一个
                                                             message 同时有一个 commit id
                                                                                                      指向一个个站点 —— 这一个个站点就是一个个仓库的不同版本
                                                                                                                                                          每有一个commit 这个当前的指针就移动到最新的位置
                                          分支的本质
                                                            分支 branch 实际上是一个个指针
                                                                                                                                                         即指向最新的那个版本
                                                                                                    - 而没有switch 到的指针则停留在最后的那版本
                                                         ─ HEAD指针永远指向那个最新的站点
                                                       查看有哪些branch —— git branch
                                                       用图看分支的情况 —— git log --graph --pretty=oneline --abbrev-commit
                                                      创建并转换到一个branch中 git branch branchName — 然后 git switch branchName dit switch branchName 
                                                      删除一个branch —— git branch -d brancName
                                                      强行删除 —— git branch -D branchName
                                                                     一般称作把其他分支的工作结果合并到这个分支上
                             - 分支
                                          操作
                                                                  git merge otherBranch
                                                                                                                 听以速度非常快
                                                      合并
                                                                              fast-forward
                                                                                                    直接把当前branch 的指针的值给改了
                                                                  方式
                                                                                这种方式合并可以留下历史
                                                                                                                   能看出曾经做过合并
                                                                               用commit 的方式进行合并 —— git merge --no-ff -m "message"
                                                                                master 应该要非常稳定
                                                                                                         —— 一般只用来正式发布新版本
                                                             分支的策略
                                                                                  ·般建一个dev 分支 —— 用来工作 —— 平常的东西往dev上面合并就行了
                                                                                                                      如果先从master 分了一个dev
                                                                                                                                                            更改了dev, 也更改了master
                                                                                                                                                                                                那么master 分支里面不能git merge dev
                                          问题和策略
                                                                                如果两个分支分自同一个版本
                                                             合并的问题
                                                                                                                    二者不能相互merge
                                                                               如果两个分支中有矛盾
                                                   简短有力,便于提及
                                                                              先git tag tagName —— 然后接下来的一个commit 就有这个标签
                                               给一个commit 附上标签
                                                                              给往日的commit 打上标签 —— git tag tagName commit id
                                                                        git show tagName —— 看对应的标签的信息
                                                                                                   git tag -a tagName -m "message about this tag"
                              标签 tag
                                                                                                   git tag -a tagName -m "message about this tag" commit id
                                                               删除本地的标签
                                                                                      git tag -d tagName
                                                                删除远程库的标签
                                                                                                         git push origin :refs/tags/tagName

    git push origin tagName
```

git push origin --tags