**git命令**

* 1.git init 初始化仓库
* 2.git status 查看当前状态
* 3.git add -A(提交所有的) 提交本地文件到缓存区
* 4.git commit -m"提交信息" 将缓存区的东西提交到本地仓库
* 5.git reset --hard sha 值 回退到某一个版本
* git reset --hard sha 值 回退到某一个版本
* git reset --mixed(默认可以不写) sha 状回退到修改态

git reset --soft sha 回退到暂存区状态

* 6.git push 将本地仓库的内容提交到远程服务器
* 7.git pull 从远程服务器更新
* 8.git log 查看所有的提交日志
* 9.git reflog 查看所有的SHA值

**分支**

* 1. git branch fixBranch(分支名称) 开启分支
* 11.git branch 查看当前分支 有\*的代表当前正在工作的分支
* 12.git checkout fixBranch 切换到fixBranch的分支上
* 13.git merge fixBranch 分支合并 将 fixBranch上的内容合并到master上
* 14.git branch -d fixBranch 删除分支
* git branch -r -d origin/branch-name 删除远程分支
* windows [git下载网址](https://git-for-windows.github.io/)
* mac不用下载自带

**15.git共享仓库**

* git clone 仓库地址
* git clone --bare 地址
* 共享仓库看不到工作区
* 但是里面有内容 他是共享的,别人只能往里面放代码,但是不让修改 如果向获取就直接克隆

文件冲突: 多个人同时改了同一个文件的同一行就会引起冲突

生成 公钥/私钥 ssh-keygen -t rsa

* git remote -v 查看远程仓库地址 默认 origin
* git remote add 仓库名称 仓库地址
* 修改远程仓库地址
  + 1.直接修改

git remote origin set-url [url]

* + 2.先删除后增加
  + git remote rm origin

git remote add origin [url]

* + 3.直接修改config文件

**16.git tag 标签**

* 标签可以针对某一时间点的版本做标记，常用于版本发布
* git tag v0.1.2
* 创建附注标签
* git tag -a v0.1.2 -m “0.1.2版本”
* 切换到标签
* 与切换分支命令相同，用git checkout [tagname]
* 用git show命令可以查看标签的版本信息：
* git show v0.1.2
* 给指定的commit打标签
* git tag -a v0.1.1 9fbc3d0
* 标签发布
* 通常的git push不会将标签对象提交到git服务器，我们需要进行显式的操作：
* git push origin v0.1.2 # 将v0.1.2标签提交到git服务器
* git push origin –-tags # 将本地所有标签一次性提交到git服务器
* git tag -d v0.1.2 # 删除本地标签
* git push :refs/tags/v0.1.2 删除远程tag
* git tag 查看本地的tag

**17.git差异化**

* 比较暂存区和当前的版本的差异 此命令比较的是工作目录(Working tree)和暂存区域快照(index)之间的差异也就是修改之后还没有暂存起来的变化内容。
* git diff
* git difftool 分屏比较
* git diff sha1 sha2 比对2个不同的版本
* git difftool sha1 sha2 分屏对比

**18.文件回退到暂存区**

* git checkout -- 文件名称

**19.保存工作区**

* git stash 保存内容(入栈) 切换分支的时候不会让另一个分支看到
* git stash pop 回到最初保存的内容

**md5加密 不可逆**

* echo -n 123456 | openssl md5
* -n就表示不输入回车符
* 结果 e10adc3949ba59abbe56e057f20f883e

**base64加密/解密**

* 加密
  + echo abc | openssl base64
  + YWJjCg== （编码结果）
* 解密
  + echo YWJjCg== | openssl base64 -d
  + abc (解码结果)
* wc 统计字节数
* wc -l filename 报告行数
* wc -c filename 报告字节数
* wc -m filename 报告字符数

wc -w filename 报告单词数

**gitignore 忽略文件**

* 创建 .gitignore文件, 图形化界面不能创建
* node\_modules/ node\_modules下的所有文件都不提交
* node\_modules/\*.jpg node\_modules/123.jpg图片不提交但是node\_modules/coderYJ/456.jpg可以提交
* .idea/\* .idea 下的所有文件都不提交
* \*.png 忽略所有的 .png 结尾的文件
* !xxoo.png 但排除 xxoo.png

想把某些目录或文件加入忽略规则，按照上述方法定义后发现并未生效，原因是.gitignore只能忽略那些原来没有被track的文件，如果某些文件已经被纳入了版本管理中，则修改.gitignore是无效的。那么解决方法就是先把本地缓存删除（改变成未track状态），然后再提交

* git rm -r --cached .
* git add .
* git commit -m 'update .gitignore'