版本控制：是一种用来记录若干文件内容变化，以便将来查看特定版本修订情况的系统；

集中化的版本系统(Centralized version contral system):为了适应多人开发协作模式，从而又产生了集中化的版本系统，像 subversion 就是典型；作为开发者只需安装客户端就能与团队保持协调一致；使用集中化的版本系统(eg.subversion)的好处：集中管理，方便查看 Team 的 Task; 方便管理；缺点是：一旦中央服务出现单点故障，Team 就无法工作；如果故障数据没有备份，那么数据丢失的风险比较大。

分布式版本系统：(Distributed version control system):分布式版本系统是为弥补集中式的版本系统不足而面世，同时它更加适合软件全球化协作模式；与集中化的版本系统的客户端不同，分布式版本系统的客户端并不只是取文件最新快照，而是把原始的仓库完整地镜像来下，这样的操作对中央仓库来说是次完整的备份；

代表 Git:

- 本地操作，对网络依赖度少
    - 查年历史记录
    - 当前版本与历史版本比较
    - 提交更新，联网后只需要上传至中央服务器即可；
- 文件的索引不是根据文件名，而是根据数据的标识符： SHA-1 哈希值：由 40 个十六进制字符组成
- 在 Git 中文件只有三种状态：
    - 已提交： 文件已经完全保存到本地的数据库；
    - 已修改： 文件已经修改，但是还没有提交保存；
    - 已暂存： 把修改后的文件保存到提交清单中；
- git config --global user.name "username"
- git config --global user.email "email name"
- git config --list

具体操作：

1. git init： 初始化后，会产生.git 目录
2. git add *.c
    1. git commit -m "message"
3. git clone "clone url" rename_dir （完成第一步的操作）
4. git status : 查看当前文件状态
5. 添加新的文件使用第二步
6. 修改文件之后，也要使用 git add
7. git commit -a -m " message"
    1. git commit -m "message"
8. git rm
9. git mv file_from file_to
10. git log :查看历史
    1. git log -p -2 : 查看最近二次更新的差异；
11. git commit --amend : 取消最后一次提交

12. git reset HEAD : 取消暂存 ；
13. git remote :查看远程库
    1. git remote -v : 查看远程库并显示 地址
14. git fetch "仓库名" ： 向远程仓库更新数据
    1. 此命令会到远程仓库中拉取所有你本地仓库中还没有的数据。运行完成后，你就可以在本地访问该远程仓库中的所有分支，将其中某个分支合并到本地，或者只是取出某个分支，一探究竟。
    2. 它只是下拉数据，但是还不会合并到本地分支中；git pull 下拉合并
15. git push [remote-name] [branch-name] 推送数据到远程仓库
    1. 如果要把本地的 master 分支推送到 origin 服务器上（再次说明下，克隆操作会自动使用默认的 master 和 origin 名字），可以运行下面的命令：
       $ git push origin master
16. git branch testing ： 建立分支
    1. 第一次默认的分支是 master
    2. 远程主仓库默认叫 origin(master)
17. git checkout testing: 切换分支
    1. git checkout -b testing : 完成 16，17 相同的命名
18. git checkout master 返回主分支
    1. git merge testing: 在主分支进行合并；
    2. git branch -d testing: 删除 testing 分支
    3. git push origin --delete testing 删除远程分支
19. 不同分支修改同一个文件时的冲突问题
    1. git status 查看状态，进行修改
    2. git add 添加在暂存
    3. git commit
20. git branch 不带任何参数，是显示所有分支名称
    1. git branch --no-merged 查看那些分支没有合并
    2. git branch --merged 查看那些分支已经合并入当前分支（也就是说哪些分支是当前分支的直接上游）
    3. git branch -a : 显示所有分支，包含隐藏分支。
    4. 如果远程有多个分支，比如除 master 之外，还有 develop 分支并且想在 develop 上作业可以使用:

       ```
       git checkout -b branch_name remote_branch_name
       ```

21. git fetch 更新数据从远程数据
    1. git push origin 提交数据到远程仓库
    2. git push origin testing 提交数据到远程仓库并在远程仓库 origin 远程分支 testing
22. git co -- <file> # 抛弃工作区修改 git co . # 抛弃工作区修改 (撤销修改)
23. git tag -a "v2.0.0" -m "Release version 2.0.0"

@Dylan Zhang