1. 集中式版本控制系统最大的毛病就是必须联网才能工作，如果在局域网内还好，带宽够大，速度够快，可如果在互联网上，遇到网速慢的话，可能提交一个10M的文件就需要5分钟，这还不得把人给憋死啊。

2分布式版本控制系统的安全性比集中式的高很多，因为每个人电脑里都有完整的版本库，某一个人的电脑坏掉了不要紧，随便从其他人那里复制一个就可以了。而集中式版本控制系统的中央服务器要是出了问题，所有人都没法干活了。

3. 分布式版本控制系统通常也有一台充当“中央服务器”的电脑，但这个服务器的作用仅仅是用来方便“交换”大家的修改，没有它大家也一样干活，只是交换修改不方便而已

4.命令总结

git init ：创建版本库

git add <file> 添加多个文件

git commit -m “message” 提交

git diff <file> 查看difference

git diff HEAD -- <file> 查看工作区和版本库里面最新版本的区别

git status 查看仓库当前的状态

git log 查看提交日志，获取对应的commit\_id，以便确定要回退到哪个版本（git log --pretty=oneline 可减少输出信息）

git reflog查看命令历史，以便确定要回到未来的哪个版本

git reset --hard commit\_id 回退到指定版本

git checkout -- <file> 直接丢弃工作区的修改(修改还没提交到暂存区)

git reset HEAD <file> 文件已经提交到暂存区，回退到最新版本

git rm <file> 从版本库中删除该文件，并且git commit（另一种情况是删错了，版本库里还有，可以通过git checkout -- <file> 把误删的文件恢复到最新版本）

本地仓库和远程仓库管理：

1. 关联一个远程库

git remote add origin 远程仓库路径

2.把本地文件添加到本地仓库

**git add . 然后** git commit -m "备注"

3. 本地库的所有内容推送到远程库上

git push -u origin master

第一次推送master分支时，加上了-u参数，Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令

注意：当执行3时，终端显示

src refspec master does not match any.

说明本地库为空，先执行2，再重新执行3

4.以后本地推送到远程都不用 -u

git push origin master

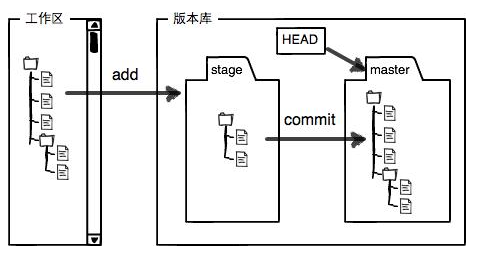
克隆一个仓库：

git clone 远程库地址

工作区（Working Directory）: 就是在电脑里能看到的目录

版本库（Repository）: Git的版本库里存了很多东西，其中最重要的就是称为stage（或者叫index）的暂存区，还有Git为我们自动创建的第一个分支master，以及指向master的一个指针叫HEAD。

git add命令实际上就是把要提交的所有修改放到暂存区（Stage），然后，执行git commit就可以一次性把暂存区的所有修改提交到分支。



5.终端快捷键

pwd 显示当前文件夹所在的路径

ls –a 显示目录下所有文件（包括隐藏文件）

rm file 删除文件