**版本管理的演变**

VCS（版本控制系统）出现前：用目录拷贝来区别不同版本（利用不同文件夹来区分不同版本）

缺点：公共文件容易被覆盖，成员沟通成本高，代码集成效率低下

集中式VCS

有集中的版本管理服务器，服务器内存放着整个项目每个文件以及每个文件夹的版本演变的历史，具备文件版本管理能力和分支管理能力。不同的分支之间可以做集成，集成效率有明显的提高。但是每个客户端内不包括完整的版本库，必须时刻和服务器相连。

分布式VCS

服务端和每个客户端都有完整的版本库

脱离服务端，客户端照样可以管理版本

查看历史和版本比较等多数操作，都不需要访问服务器，比集中式VCS更能提高版本管理效率

**Git:**

Cmd-git –version查看git版本

配置user信息

git config --global user.name ‘name’配置用户名

git config --global user.email ‘email’配置用户邮箱

config的三个作用域：缺省等同于local

git config --local 只对某一个仓库有效

git config --global 对当前用户所有仓库有效

git config --system 对系统所有登录的用户有效

显示config的配置，加--list

**建git仓库**

两种场景

1. 把已有的项目代码纳入git管理

cd 项目代码所在的文件夹

git init

1. 新建的项目直接用git管理(建立一个裸仓库)

cd 某个文件夹

git init your\_project #会在当前路径下创建和项目名称同名的文件夹

cd your\_project

git init git\_learning #建一个git\_learning仓库

cd git\_learning #进入git\_learning仓库中

ls -al #查看当前目录中所有文件夹

touch readme #新建一个readme文件

git commit -m’add readme’ #给当前操作版本添加一个操作描述，当前操作为添加一个readme

但是第一次git commit报错，readme文件目前还没有被git管控，所以git commit这个命令对该文件不生效，要先用git add命令将该文件名添加到git中，再执行git status查看当前状况。此时再执行git commit -m’add readme’就可以了

git log #查看编辑历史记录

往仓库中添加文件

4次提交，一个像模像样的静态页面生成了

cmder

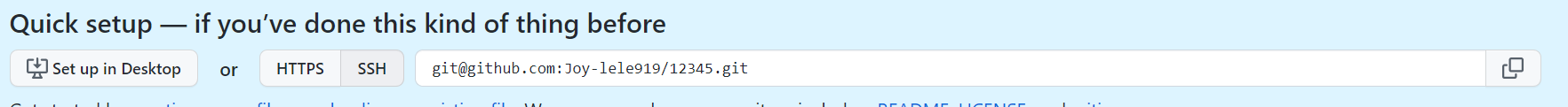
cd ..返回上一层目录

git status查看当前状态

git log查看版本提交记录

在cmder框中进入到想要推的文件所在的文件夹

1. 在github内创建一个仓库并命名
2. 在cmder框内输入git init，将当前所在的文件夹变成一个仓库，并在文件夹内创建一个本地的git版本仓库，用来存储该仓库所进行的一切操作行为以及配置文件，此时文件夹仓库内有一个初始化分支master，此时仓库内所有文件位于master分支上，可用ls -la查看当前分支上所有文件列表
3. git add .将需要编辑的文件添加到暂存区内
4. git commit -m”描述语言”添加版本描述语言
5. git remote add origin 仓库地址（github上创建完仓库就有）：给远程仓库（github上新建的仓库）添加一个名字origin，并指明仓库地址



1. git push origin master将本地的master推到origin远程仓库中

修改

1. 在cmder框中进入到想要存放的路径中
2. git clone 需要修改的仓库地址（如果是新文件夹，用clone，老文件夹，用pull）
3. 在本地对需要修改的文件进行修改
4. git add .
5. git commit -m”版本信息”
6. git push origin master