Git简介

Git起源

- 在2002年以前，世界各地的志愿者把源代码文件通过 diff[1] 的方式发给Linus，然后由Linus本人通过手工方式合并代码
- 到了2002年,Linus选择了一个商业的版本控制系统BitKeeper，BitKeeper的东家BitMover公司出于人道主义精神，授权Linux社区免费使用这个版本控制系统
- 2005,Linux社区牛人试图破解BitKeeper的协议，BitMover公司要收回Linux社区的免费使用权
- Linus花了两周时间自己用C写了一个分布式版本控制系统——Git！一个月之内，Linux系统的源码已经由Git管理了

版本控制系统[2]

- 集中式：
  - 集中式版本控制系统，版本库是集中存放在中央服务器的，而干活的时候，用的都是自己的电脑，所以要先从中央服务器取得最新的版本，然后开始干活，干完活了，再把自己的活推送给中央服务器。
  - 集中式版本控制系统最大的毛病就是必须联网才能工作，受限于网速
- 分布式：
  - 分布式版本控制系统没有“中央服务器”，每个人的电脑上都是一个完整的版本库，工作的时候，就不需要联网
  - 个人的对库文件的修改只需通过局域网推送给对方即可看到
  - 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
- Git是目前世界上最先进的 分布式版本控制系统[3]

Git 特性

- 直接**记录快照**、而非差异比较
  - Git 只关心文件数据的整体是否发生变化，而大多数其他系统则只关心文件内容的具体差异
  - 实际上，Git 更像是把变化的文件作快照后，记录在一个微型的文件系统中。每次提交更新时，它会纵览一遍所有文件的指纹信息并对文件作一快照，然后保存一个指向这次快照的索引。为提高性能，若文件没有变化，Git 不会再次保存，而只对上次保存的快照作一链接
- 近乎所有的操作都是本地执行
- 时刻保持数据的完整性
  - 在保存到 Git 之前，所有数据都要进行内容的**校验和**（checksum）计算，并将此结果作为数据的唯一标识和索引
  - Git 使用 SHA-1 算法计算数据的校验和，通过对文件的内容或目录的结构计算出一个 SHA-1 哈希值，作为指纹字符串。该字串由 40 个十六进制字符（0-9 及 a-f）组成。**所有保存在 Git 数据库中的东西都是用此哈希值来作索引的，而不是靠文件名**
- 多数操作仅添加数据

安装Git

- Linux：直接用命令行安装
- Win：在官网下载安装程序，并配置机器名字和地址
- 具体操作互联网搜索即可
- 第一次安装后准备过来
  1. 为在这台机器上的commit签名

     - 因为Git是分布式版本控制系统，所以，每个机器都必须自报家门：你的名字和Email地址（名字和邮箱都不会进行验证），这样远程仓库才知道哪次提交是由谁完成的
     - 配置的用户名和邮箱对push代码到远程仓库时的身份验证没有作用，即不用他们进行身份验证；他们仅仅会出现在远程仓库的commits里。
     - 最后，这个用户名和邮箱是可以随便配置的（不提倡），如果你配置的邮箱是github里真实存在的邮箱，则commits里显示的是这个邮箱对应的账号；如果配置的邮箱是一个在github里不存在的邮箱，则commits里显示的

> 是你配置的用户名。

```
git config --global user.name "xxx"
git config --global user.email "email@example.com"
```

> 可先通过`git config --global --list`查看本机上的git配置情况

2. 为这台机器配置ssh

```
cd ~/.ssh       # 进入ssh密匙目录
ls              # 查看本机密匙情况 需要寻找一对以 id_dsa 或 id_rsa 命名的文件，其中一个带有 .pub 扩展名。.pub 文件是你的公钥，另一个则是与之对应的私钥
ssh-keygen -t rsa -C "配置的邮箱"    # -t 密匙的种类；  - C 相当于密匙的名字，用来识别密匙
```

- 将公匙内容复制的github的设置中

版本库

- 版本库（repository）：可以简单理解成一个目录，这个目录里面的所有文件都可以被Git管理起来，每个文件的修改、删除，Git都能跟踪，以便任何时刻都可以追踪历史，或者在将来某个时刻可以“还原”
- 工作目录下面的所有文件都不外乎这两种状态：已跟踪或未跟踪。
  - 已跟踪的文件是指本来就被纳入版本控制管理的文件，在上次快照中有它们的记录，工作一段时间后，它们的状态可能是未更新，已修改或者已放入暂存区。
  - 而所有其他文件都属于未跟踪文件。它们既没有上次更新时的快照，也不在当前的暂存区域

操作过程

- `git init`：在指定目录中使用，可以将该目录变成Git可以管理的仓库
  - 会在该目录下生成一个`.git`目录，是Git用来跟踪管理版本库的，一般不需改动
  - 该仓库是一个本地仓库
- `git add xxx`：把位于指定目录的文件添加到仓库

```
git add .       # 将所有的工作区的修改添加到暂存区
```

- `git commit -m "xxx"`：将文件提交到仓库
  - 一次commit会将之前所有的add都会提交
  - `-a`，自动将所有已跟踪的文件暂存起来一并提交

扩展阅读

- 百科-版本控制
- 集中式（SVN）和分布式（Git）版本控制系统的简单比较

---

1. diff 命令比较文本文件,它能比较单个文件或者目录内容。 ↩

2. 一种软体工程技巧，籍以在开发的过程中，确保由不同人所编辑的同一档案都得到更新。版本控制透过文档控制(documentation control)记录程序各个模组的改动，并为每次改动编上序号 ↩

3. 每个人都可以创建一个独立的代码仓库用于管理，各种版本控制的操作都可以在本地完成。每个人修改的代码都可以推送合并到另外一个代码仓库中 ↩