# Git版本控制系统

## Git版本控制系统的使用优点

①更方便的存储版本信息

②可以在各个版本之间进行切换

③更方便进行版本对比

④方便协同合作

## 原始的本地版本控制缺点

①本地会存在很多以前到现在一直维护的各个版本文件，附带日期和版本信息（使用Git版本控制系统后，本地管理的内容基本只有一份，不需要很注重命名方式，更新完提交即可，提交后会在版本控制系统里进行保存）

②回退版本会对目前在更新到一半的版本造成麻烦（使用Git版本控制系统后，所有更改过的信息都会记录在系统中，可以直接回退或者切换到最新的版本）

③对比新旧文件过于麻烦，只能打开两个文档逐行逐行比对（使用Git版本控制系统后，可生成对比信息，并且有明显标注）

④不方便于协同合作，难以得知当前其他成员对文档的最新更改

## Git的下载

下载地址：<https://git-scm.com/>

## Git的最小配置

如果在配置前进行操作，git会出现相应的警示提醒；

也是为了多人协作时，代码后面的维护可能需要找到相应的人。

### 1、查询git配置信息

|  |
| --- |
| git config --global --list |

### 2、配置名称和邮箱

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

## Git的基础操作及指令

### 1、版本查询

|  |
| --- |
| git --version |

### 2、建立Git仓库

①已经有文件时

|  |
| --- |
| cd target\_folder  git init |

②没有文件时

|  |
| --- |
| cd target\_folder  git init projectname |

### 3、配置信息

git config默认是--local，而local只对当前仓库有效，global则是对登录用户所有的仓库有效，system对系统所有用户都有效

|  |
| --- |
| git config --local //如果配置了则会有限找到local对应信息  git config --global //再然后找global信息  git config --system //最后再找system信息 |

### 4、提交版本

|  |
| --- |
| git add file //把修改过得文件提交到临时缓存区，还不是正式版本  git add . //如果文件太多，想提交当前文件夹下的所有，可执行  git commit -m "版本信息" // |

### 5、查看当前项目的信息

|  |
| --- |
| git log |

### 6、版本状态查看

当提交完一个项目到git，都会进行版本状态的查看，看看文件修改成什么样，在版本控制系统中，处于一个什么状态（比如：目前处于暂存区间？只是刚好处于修改状态？或者已经完全被保存到版本控制系统）

|  |
| --- |
| git status  git status -s  git status --short |

示例：