# Git

## git的优势

* 1. 是分布式的、分支操作非常流畅
  2. 支持离线工作、本地提交可以稍后提交到服务器上
  3. 项目有完整性保障、无需过多依赖中央服务器（每个本地也有全部的信息）
  4. 向中央服务器传输内容依托的是 文件流传输，速度比SVN快倍

## 工作的流程及原理

1. 项目仓库 $ git init
2. 原理
   * 1. 工作区
     2. 暂存区
     3. 历史区
     4. 工作区 --> 暂存区 --> 历史区 --> 中央服务器
3. 工作区 ( $ git add 到达暂存区)
4. 暂存区 ( $ git commit 到达历史区 )
5. 历史区

# 安装

1. 查看配置信息 $ git config -l
2. 查看全局配置信息$ git config --global -l
3. 配置全局信息，用户名和邮箱
   1. $ git config --global user.name ‘xxx’
   2. $ git config --global user.name ‘xxx@xx.xx’
4. （当没有配置全局信息而查看全局配置信息时，会报错）
5. 检测安装成果的命令： git --version
6. 第一次安装后需要配置Git的全局（告诉Git我是谁？ 我的签名是什么，也就是邮箱，用来辩证身份）
7. 创建仓库完成版本控制
8. $ git init //会生成一个隐藏的文件夹 .git 这个文件夹不能删（因为这里储存了暂存区、历史区等信息）
9. 将工作区 --> 暂存区
   1. $ git add xxx //将工作区的某个文件或文件夹提交到暂存区
   2. $ git add . | $ git -A //将工作区中修改的所有文件提交到暂存区
10. 将暂存区 --> 历史区
    1. $ git commit -m’提交文件的描述’
11. 查看当前文件的状态
    1. $ git status
       1. 文件颜色 红色：文件在工作区内
       2. 绿色 ：文件在暂存区内
       3. 没有当前修改信息：文件在历史区内
12. 查看历史区版本信息（历史记录）
    1. $ git log (不包含回滚的信息)
    2. $ git reflog (包含回滚的信息)
13. 回滚到指定版本信息(相当于撤回到指定版本)
    1. $ git reset --hard xxxxxxxxx(指定的版本号)

# 把本地仓库提交到 远程仓库

* 1. 查看本地仓库与哪些远程服务器保持连接 $ git remote -v
  2. 把本地仓库和远程仓库连接 $ git remote add origin xxxxxxxxxxxx(git连接地址)
  3. 删除关联信息 $ git remote rm origin
  4. 提交前先拉去信息 $ git pull origin master
  5. 把本地仓库（历史区）提交到远程仓库 $ git push origin master(以弹出的形式出现：用户名和验证码)

实际操作中的流程：

1. 负责人创建一个中央仓库（增加开发者）
2. 小组成员基于 $ git clone 把远程仓库及默认内容克隆岛本地（解决了三件事：
   1. 初始化一个本地仓库： $ git init
   2. 和对应的远程仓库保持连接：$ git remote add origin xxxxxxxxxxxx(git连接地址)
   3. 把远程仓库默认的内容拉取到本地： $ git pull origin master）
3. 小组成员基于 $ git add / $ git commit 把自己修改的内容存放到暂存区，然后$ git pull / $ git push 把本地信息和远程仓库信息同步