# Git学习

1. 什么是git?

git是分布式版本控制系统，可以记录修改的动作及多人协作，方便文件的管理。

**Tips:** 版本控制只能针对文本内容，像word这样的二进制内容，所谓版本，就是文件快照，不能比较差异，只能算备份的网盘。

1. 集中式和分布式的区别

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。中央服务器就好比是一个图书馆，你要改一本书，必须先从图书馆借出来，然后回到家自己改，改完了，再放回图书馆。缺点就是必须联网才能工作，因为你必须从服务器拿到最新的版本进行修改。

分布式版本控制系统：首先，分布式版本控制系统根本没有“中央服务器”，每个人的电脑上都是一个完整的版本库，这样，你工作的时候，就不需要联网了，因为版本库就在你自己的电脑上。既然每个人电脑上都有一个完整的版本库，那多个人如何协作呢？比方说你在自己电脑上改了文件A，你的同事也在他的电脑上改了文件A，这时，你们俩之间只需把各自的修改推送给对方，就可以互相看到对方的修改了。

和集中式版本控制系统相比，分布式版本控制系统的安全性要高很多，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。

在实际使用分布式版本控制系统的时候，其实很少在两人之间的电脑上推送版本库的修改，因为可能你们俩不在一个局域网内，两台电脑互相访问不了，也可能今天你的同事病了，他的电脑压根没有开机。因此，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。——分布式版本控制的每个节点都是完整仓库（by 廖雪峰）

1. 安装git

安装过程是极其简单的，在这里就不多说了。安装完成后。需要最后一步设置，在命令行输入：$ git config --global user.name “Your Name”

$ git config --glogbal user.email [“email@example.com”](mailto:\“email@example.com\”)

Tips: git config命令的--global参数表示这台机器上的所有仓库都会使用上面的配置，当然也可以对某个仓库指定不同的用户名和Email地址。

1. 创建版本库

第一步，创建版本库非常简单，首先在一个合适的地方，创建一个空目录：

$ mkdit learngit //创建learngit目录

$ cd learngit //进入该目录

$ pwd //显示当前目录

第二步，通过git init 命令把上面创建的目录变成Git可以管理的仓库（其实完全可以在已有的项目目录下使用该命令，从而省去第一步）：

$ git init

Initialized empty Git repository in /Users/michael/learngit/.git/

创建好git仓库之后，项目子目录下会出现一个隐藏的.git目录。通过ls-ah可查看。

**Tips:** 版本控制职能跟踪文本问价的改动，比如TXT文件、网页、程序代码等等。它可以告诉你每次的改动，而图片、视频这些二进制文件，没有办法跟踪文件的变化只知道大小的变化，至于改了啥，版本控制系统是不得而知的。

1. 把文件添加到版本库

第一步：$ git add . //把当前目录的所有文件都添加上，后面的 . 表示所有的意思。

第二步：$ git commit -m “提交文件” //-m后面的内容表示本次提交的说明，方便从历史记录中找到改动记录，因此该说明最好做到让人一目了然。

**Tips:** 可以使用git stauts命令掌握工作区的状态，如果git status告诉你有文件该修改过，用git diff可以查看修改内容。（最好是修改到一定的程度，完成某些特定的部分之后再使用git commit提交到版本库，并加以注释说明）

1. 版本回退（后悔药）

工作中，一旦我们把文件改乱了，或者误删了，可以从最近的一个commit恢复，然后继续工作。使用git log命令可以查看修改（commit）的历史记录，git log命令显示从最近到最远的提交日志。如果输出信息太多，可以加上--pretty=online参数过滤。

在我们进行版本回退的时候，我们需要知道当前版本是哪个版本，HEAD表示当前版本，也就是最新的提交，上一个版本就是HEAD^，上上个版本就是HEAD^^，二HEAD~\*\*\* 表示\*\*\*版本。使用git reset命令；

$ git reset --hard HEAD^，使用该命令后用git log，最新版本就看不到了，如果想回到之前的版本，只要命令窗口没有被关闭，依然可以回到最新版本：

$ git reset --hard 79606 (版本号没必要写全，但也不能只写前一两位，因为Git可能会找到多个版本号，就无法确定是哪一个了)

如果我关闭了命令窗口抑或是关了电脑，而我又想重新回退到最新版本，但是我根本不知道commit id！！怎么办？在Git中，它提供了一个命令git reflog用来记录每一次命令：

$ git reflog 通过该命令行可以查看commit的内容注释和id，由此可见commit -m后面的注释内容可以帮助你识别之前commit提交的内容变动情况。

**小结：HEAD指向的版本就是当前版本，使用命令git reset --hard commit\_id便可以穿梭未来；穿梭前，git log 可以查看提交历史，以前确定要回退到哪个版本；要重返未来，用git reflog查看命令历史，以便确定要回到未来的哪个版本。**

1. 工作区和暂存区

工作区：就是在电脑里看到的目录，也就是你在本地操作版本库的地方。

版本库：工作区有一个隐藏的目录.git，这个是Git的版本库。里面有一个称为stage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。将文件添加到Git版本库的时候，是分两步进行的：

第一步、用git add把文件添加进去，实际上就是把文件修改添加到暂存区；

第二步、用git commit提交更改，实际上就是把暂存区的所有内容提交到当前分支。

在创建Git版本库的时候，Git自动为我们创建了唯一的master分支，所以git commit就是往master分支上提交更改。也就是说，可以一次性提交暂存区的所有修改。

1. 管理修改

git管理的是文件的修改，每次修改，如果不add到暂存区，那就不会加入到commit中。比如说：第一次修改 -> git add -> 第二次修改 -> git commit

在这里只会把第一次修改的内容提交到版本库中。如果我们需要把全部已经修改好的文本内容提交上去，则在你修改后仍旧需要使用git add -> git commit这两个命令操作。

1. 撤销修改

一种是test.txt自修改后还没有被放到暂存区，现在，撤销修改就回到和版本库一模一样的状态；

一种是test.txt已经添加到暂存区后，又作了修改，现在，撤销修改就回到添加到暂存区后的状态。

如果已经从暂存区提交到了版本库，可以进行版本回退。

**Tips:**

场景1：当你改乱了工作区某个文件的内容，想直接丢弃工作区的修改时，用命令git checkout -- file。

场景2：当你不但改乱了工作区某个文件的内容，还添加到了暂存区时，想丢弃修改，分两步，第一步用命令git reset HEAD file，就回到了场景1，第二步按场景1操作。

场景3：已经提交了不合适的修改到版本库时，想要撤销本次提交，参考版本回退一节，不过前提是没有推送到远程库。

1. 删除文件

rm命令用于删除一个文件。（从工作区中删除）

如果一个文件已经被提交到版本库，那么你永远不用担心误删，但是要小心，你只能恢复文件到最新版本，你会丢失最近一次提交后你修改的内容。

**Tips:** 恢复文件使用git checkout -- file（从版本库中恢复）

git rm （从版本库中删除），并且git commit

1、git rm test.txt ，git commit -m 'test.txt'

提交了删除操作，版本库就没有 test.txt 文件了，不能用 git checkout 撤销，要找回只能退回版本号。

1. git checkout主要是针对工作区的操作，前提是你没有提交操作到版本库，也就是没有执行命令 git commit -m 'test.txt'。
2. 远程仓库