# Git分布式版本控制系统学习笔记

1. 什么是Git

Git 是一款分布式版本控制系统。分布式，即没有中央主机，每个用户拥有的都可以是主机；版本控制，即能实现各个版本之间的切换。

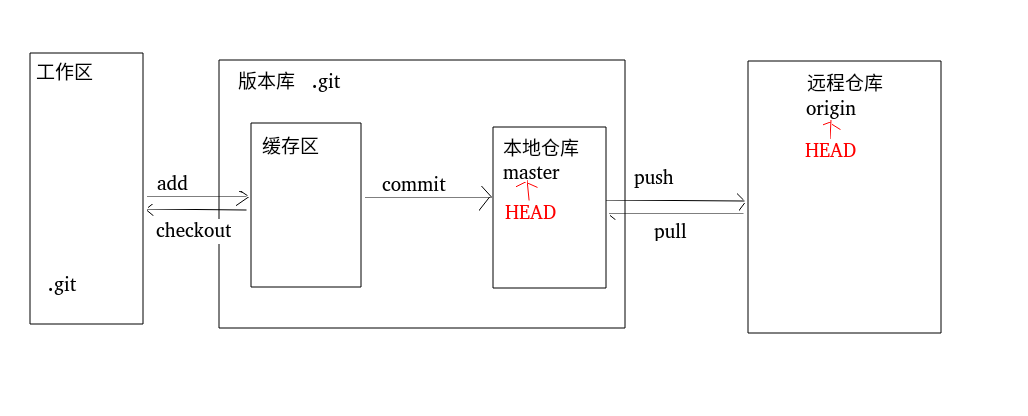
场景1：当我们一个人做项目的时候，每天对自己都有任务要求，当我们有一天在工作的时候，熊孩子爬上了键盘，一顿操作，你写的完美的代码变得缺七少八，混乱不堪，我不知道熊孩子总共修改了代码的哪些地方，所以我们陷入了困境，这个时候，版本控制系统就可以展现他的作用了。如果我们使用git ，则每天完成当天的任务后，都要将成果推送到版本库，再遇到上面的情况时，我们直接将版本回溯到昨天即可。

场景2：一个团队共同在开发一个产品，每个人负责一个模块内容的开发，开发完后，将源代码发给主管，让主管合并代码，如果人少还好，如果团队是100人呢?那样主管得累死。还好有git，git可以默认给项目一个主分支master，这个分支是稳定用来推送新版本的，程序员们可以在这个分支下新建一个分支，专门用来开发，当开发完成，再将开发分支与主分支合并即可。

1. git 的安装

在linux系统下: sudo apt-get install git

1. git的使用
   * 1. Git分四层管理代码版本
        1. 工作区，你选择存放仓库的目录
        2. 缓存区，add操作之后文件存放的位置
        3. 本地仓库，每次commit操作之后文件存放的位置
        4. 远程仓库, 存在于远端，如github,码云等



注：本地仓库的创建有两种方法

1. git init 则当前目录就成为了工作区，本地仓库也在这里。
2. Git clone xxxx，克隆了一个远程仓库到本地，文件名为仓库名，这个克隆的仓库就是工作区

使用git前的准备

1. 要在github上创建一个账号，并创建一个仓库
2. 生成ssh key ： ssh-keygen -t -C “你的注册邮箱”,

默认生成在home目录下的 **.ssh**文件夹下，将id\_rsa.pub 里的内容复制到github个人的ssh key里，这样就可以给本地仓库与远程仓库拉取分支和推送更新提供了基础。

1. 设置用户名与邮箱信息,让git可以识别你

Config --global user.email “邮箱”

Config --global user.name “用户名”

1. 将本地仓库与远程仓库关联起来

Git remote add branch\_name “远程仓库地址”

Branch\_name 为远程仓库的分支名，默认为origin,但是当想将一个本地仓库关联到多个远程仓库，那么就得给每一个远程仓库一个别名，不然所有的都是origin就会出错。

Git remote -v可查看当前关联的远程仓库信息。

配置完这些以后，以后使用就不用再次配置啦。

1. 常用命令
   1. 将文件添加到缓存区: git add 文件名/目录
   2. 将缓存区中添加的文件提交到本地仓库: git commit -m “提示信息”
   3. 将本地仓库push到远程仓库更新: git push origin master

Origin 代表远程仓库名，master代表本地要push的仓库名

* 1. 查看历史操作 git log [options]

返回每一次commit的id,分支信息、author、日期、操作

Options:

--pretty=oneline 只显示一行，包括commit\_id 和 分支信息以及提示信息

--graph 可看到分支合并图

<file\_name> 可查看指定文件的commit记录

* 1. 查看缓存区内的信息 git status
  2. Git diff

查看两个commit之间的区别 git diff com\_id1 com\_id2

查看工作区与缓存区的区别 git diff

查看本地版本库与工作区的区别 git diff HEAD -- file\_name

**版本回溯:**

1. 撤销对工作区内容的修改 git checkout --file\_name
2. 撤销对暂存区内的内容的修改 先git reset HEAD <file> 再 git checkout -- file
3. 已经提交到当地版本库 先git log 得到commit\_id 再 git reset HEAD/commit\_id

**Git 所用协议**

https: 速度慢，每次推送都得输命令

Ssh 即git://形式，默认

**分支管理**

背景：玄幻小说中，主人公修成身外化身，身外化身修炼其他功法，若是与本身不冲突，那么实力更进一步，若是修的属性相冲突，那么二者合体就会出问题。

此处分支也是如此。但是此处分支在合并之前互相独立，互不影响。

注：在本地版本库，master为主分支，指向最新的一次提交，而HEAD指向的是分支，可以是当前分支，也可以是其他分支，HEAD的指向是可变的。

HEAD: 用来记录版本号，每一次commit都会生成一个版本号，HEAD为当前版本，HEAD^ 为上一个版本，HEAD~n 为前n个版本。

创建一个分支 git branch 分支名

切换分支 git checkout 分支名

二者可以合并为一条命令,创建并切换到这个分支 git checkout -b 分支名

查看分支信息 git branch 当前分支之前会有个 ‘ \* ’

合并分支 git merge 分支名 ，一般先切回主分支，再在主分支上合并其他分支的内容。

如果分支合并出了问题，应先修改冲突文件为理想中的样子，再合并。

删除临时分支 git branch -d 分支名 (-D 为强制删除)

**分支管理策略**

Fast-forward 合并模式 无曾经合并的历史

Git merge xx --no-ff 普通模式，有分支合并历史。

Master分支，稳定，不用于开发，用于发布新版本

Dev 分支 用于开发，开发结束后再与master合并

**Bug 分支**

类似于计算机cpu正在处理一个进程，但是有更高优先级的进程到达时，cpu会将当前进程的信息存起来，再切换到那个进程。此处也是一样，当我们在一个分支进行开发，但是上面发现了个bug，让我们去解决，我们要先保存当前工作现场，再创建一个解决bug的分支，切换过去，解决完bug后删除那个临时分支。

保存当前工作现场: git stash

查看被保存的工作现场 git stash list

恢复工作现场 git stash apply(恢复，保存记录仍然存在) 再 Git stash drop 删除保存记录

git stash pop 恢复+删除记录(类似python 列表pop方法)

Feature分支 用于开发项目内的新功能时创建的分支,开发完合并后删除

多人协作：

1. git push origin <local\_branch> 推送自己的修改
2. 如果1失败，则可能远程分支比当前分支版本更新，故要先拉取远程分支并更新

Git pull xxx

1. 若合并有冲突，则解决
2. 无冲突后 git push 成功

注意:如果git pull 显示no tracking information 说明本地分之爱和远程 分支没有建立连接 用git branch --set-upstream -to <local\_branch> origin/<remote\_branch>

在本地创建与远程分支对应的分支 git checkout -b local\_branch origin/<remote\_branch>

建立与远程分支的连接 git branch --set-upstream local\_branch origin/remote\_branch

标签管理

虽然每一个版本，每一个commit都有一个id,但是很长，而且难以辨别故引入标签管理,就像ip地址之于域名。

标签即指向一个commit的指针

创建一个标签 git tag v1.0(标签名)

查看所有标签 git tag

查看某一个标签信息 git show tag\_name

对指定的commit打标签 git tag tag\_name commit\_id

创建指定信息的标签 git tag -a <tag\_name> -m “ tag说明”

Pull-request流程

用于参加开源项目中，给别人贡献代码或者提交bug

1. Fork一份别人的项目到自己的github
2. Git Clone xxx 将该项目克隆到本地
3. 贡献代码或者报bug ，完成后push到自己fork的仓库里
4. 将自己更新后的仓库Pull request给源作者，如果原作者觉得可以，就会将我们的代码跟他的代码合并，这就达到了目的。

一个本地仓库与多个远程仓库相关联

Git remote add origin1 xxorigin1的仓库地址xx

Git remote add origin2 xxorigin2的仓库地址xx

对于重要的文件，让git忽略，不上传，如数据库配置文件,python编译文件.

创建一个 .gitignore文件，将要忽略的文件写入其中，再将此文件提交到git

跟linux命令一样，git的命令也可以起别名

Git config --global alias.co(新名) checkout(默认名)

粗略地总结了下git的知识点，内容不够详细。想深入建议多上官网。

2018-09-22 晚21:24 来自noobcoderr