Git 那些事儿

作者：junze
原文链接：https://ld246.com/article/1465371955823
来源网站：链滴

Git是目前世界上最先进的分布式版本控制系统，适合多人协作开发的大型项目。我平常也经常使用gi，来管理自己的几个小项目。简单说说git的原理和git的特点！（只有知道了一个工具的运行原理，计思路，才能更好的使用这个工具）

#1. 自己对SVN和Git的体验

在公司一直用SVN，自己折腾的业余项目用Git我的Github。个人认为SVN用起来比较快捷，方便，交代码只需要 commit一下就行了，适合小团队的代码版本管理。但是一个大型的开源项目，可能有百或者上千个开发者提交代码，SVN就显得力不从心了！SO　Git大法横空出世了！

#2. Git的诞生背景

自2002年开始，林纳斯·托瓦兹决定使用BitKeeper作为Linux内核主要的版本控制系统用以维护代码因为BitKeeper为专有软件，这个决定在社区中长期遭受质疑。在Linux社区中，特别是理查德·斯托与自由软件基金会的成员，主张应该使用开放源代码的软件来作为Linux核心的版本控制系统。林纳斯托瓦兹曾考虑过采用现成软件作为版本控制系统（例如Monotone），但这些软件都存在一些问题，别是性能不佳。现成的方案，如CVS的架构，受到林纳斯·托瓦兹的批评

2005年，安德鲁·垂鸠写了一个简单程序，可以连接BitKeeper的存储库，BitKeeper著作权拥有者拉里麦沃伊认为安德鲁·垂鸠对BitKeeper内部使用的协议进行逆向工程，决定收回无偿使用BitKeeper的权。Linux内核开发团队与BitMover公司进行蹉商，但无法解决他们之间的歧见。林纳斯·托瓦兹决定行开发版本控制系统替代BitKeeper，以十天的时间，编写出第一个git版本

资料来自维基百科Git-维基百科

#3. Git于SVN的主要区别

SVN是集中式版本控制系统，版本库是集中放在中央服务器的，而干活的时候，用的都是自己的电脑所以首先要从中央服务器哪里得到最新的版本，然后干活，干完后，需要把自己做完的活推送到中央务器。集中式版本控制系统是必须联网才能工作，如果在局域网还可以，带宽够大，速度够快，如果互联网下，如果网速慢的话，就纳闷了。

Git是分布式版本控制系统，那么它就没有中央服务器的，每个人的电脑就是一个完整的版本库，这，工作的时候就不需要联网了，因为版本都是在自己的电脑上。既然每个人的电脑都有一个完整的版库，那多个人如何协作呢？比如说自己在电脑上改了文件A，其他人也在电脑上改了文件A，这时，们两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。

#4. 浅析Git原理

git的底层从其本质上讲是一个内容寻址文件系统,然后基于这个内容寻址文件系统实现了一套vcs(版本制系统)的高层接口,方便我们使用.当然git也提供了底层接口,便于我们使用之做出符合自己需求的系统.

我们把文件内容交给git进行管理,总得有一个地方来存放这些内容是吧!

是的,在git中,所有的文件内容都保存在git仓库的objects目录中.

初始化git库

我们初始化一个git仓库有两种方式,git init和git –bare init

这两者的区别是,前者会在当前目录下生成一个.git目录(此目录即为git库的目录),而当前目录为我们的作目录,一般是checkout后的文件,我们编程时所读写的内容都在此目录下.

后者的bare的意思就是裸的意思,也就是直接把当前目录当作git库的目录,这个一般用在远程git库上,为我们在远程git库上没有checkout的需求,只是用作单纯的git库

git库还有个优点就是直接拷贝到另一个地方就可以直接用了,只要你的相应的机器上安装了git即可.

git对象

git中一个非常重要的概念就是git对象,我们可以把git系统想象成一个强大的key-value存储,每一个对都对应着一个40位的哈希值.通过这个哈希值我们便可以很容易的取得对象(当然我们可以为这些哈希取一些有意义的别名,方便我们使用).我们可以把这个哈希值看作指针.而对应的对象就是指针所指向的体.对象和对象之间还可以通过通过指针进行一些关联的操作.

git对象可分为四种类型:

- blob对象 用来存放文件数据
- tree对象 对应着目录,tree的内容为blob对象的指针或者其他tree对象的指针
- commit对象 每一次commit都会产生一个新的commit对象,其包含了一个指向tree对象的指针,指前一次commit对象的指针,还包含了commit的时间,作者和注释等信息,就相当于为项目做了一次snapsot,通过commit对象我们可以跟踪到前一次commit对象,这样就可以实现log功能了
- tag对象 一种特殊的commit对象

git库目录

接下来分析git库目录中各个文件的作用

```
Wujunze-MacBook:test.git Junze$ ls -al
total 32
drwxr-xr-x  11 Luke  staff   374 Jun  4 20:21 .
drwxr-xr-x  24 Luke  staff   816 Jun  4 20:21 ..
-rw-r--r--   1 Luke  staff    23 Jun  4 20:21 HEAD (当前分支的指针)
drwxr-xr-x   2 Luke  staff    68 Jun  4 20:21 branches
-rw-r--r--   1 Luke  staff    85 Jun  4 20:21 config
-rw-r--r--   1 Luke  staff    73 Jun  4 20:21 description
drwxr-xr-x  12 Luke  staff   408 Jun  4 20:21 hooks (可以实现在特定操作的前或者后触发一些动作)
drwxr-xr-x   3 Luke  staff   102 Jun  4 20:21 info
drwxr-xr-x  64 Luke  staff  2176 Jun  4 20:21 objects (blob,tree,commit,tag 对象)
-rw-r--r--   1 Luke  staff    85 Jun  4 20:21 packed-refs
drwxr-xr-x   4 Luke  staff   136 Jun  4 20:21 refs (指向各个分支的指针)
```

objects保存的时候,以40位哈希值的前两位作为子目录的名称,后38位作为对象的文件名

git系统会定期对所有的objects进行打包操作,这样可以减少磁盘占用空间

git中最新版本的都是直接保存的,以前版本是通过引用最新的文件以及差异进行获取的,这是因为大都时候我们对最新的分支代码更为关注

#5.Git的学习

先熟悉Git的运行原理和设计思路，然后把自己的项目迁移到Git。自己动手用Git，才能真正的熟练使Git！

推荐一个不错的Git教程,廖雪峰的Git教程！**最浅显易懂的Git教程**

也欢迎大家加QQ群213470752一起学习交流Git的使用!

#6.Git常用命令

PS: 一些Git命令使用了别名 co=checkout ci=commit br=branch 等

查看、添加、提交、删除、找回，重置修改文件

git help <command> # 显示command的help

git show # 显示某次提交的内容 git show $id

git co -- <file> # 抛弃工作区修改

git co . # 抛弃工作区修改

git add <file> # 将工作文件修改提交到本地暂存区

git add . # 将所有修改过的工作文件提交暂存区

git rm <file> # 从版本库中删除文件

git rm <file> --cached # 从版本库中删除文件，但不删除文件

git reset <file> # 从暂存区恢复到工作文件

git reset -- . # 从暂存区恢复到工作文件

git reset --hard # 恢复最近一次提交过的状态，即放弃上次提交后的所有本次修改

git ci <file> git ci . git ci -a # 将git add, git rm和git ci等操作都合并在一起做 git ci -am "some c mments"

git ci --amend # 修改最后一次提交记录

git revert <$id> # 恢复某次提交的状态，恢复动作本身也创建次提交对象

git revert HEAD # 恢复最后一次提交的状态

查看文件diff

git diff <file> # 比较当前文件和暂存区文件差异 git diff

git diff <id1><id1><id2> # 比较两次提交之间的差异

git diff <branch1>..<branch2> # 在两个分支之间比较

git diff --staged # 比较暂存区和版本库差异

git diff --cached # 比较暂存区和版本库差异

git diff --stat # 仅仅比较统计信息

查看提交记录

git log git log <file> # 查看该文件每次提交记录

git log -p <file> # 查看每次详细修改内容的diff

git log -p -2 # 查看最近两次详细修改内容的diff

git log --stat #查看提交统计信息

tig

Mac上可以使用tig代替diff和log，brew install tig

Git 本地分支管理

查看、切换、创建和删除分支

git br -r # 查看远程分支

git br <new_branch> # 创建新的分支

git br -v # 查看各个分支最后提交信息

git br --merged # 查看已经被合并到当前分支的分支

git br --no-merged # 查看尚未被合并到当前分支的分支

git co <branch> # 切换到某个分支

git co -b <new_branch> # 创建新的分支，并且切换过去

git co -b <new_branch> <branch> # 基于branch创建新的new_branch

git co $id # 把某次历史提交记录checkout出来，但无分支信息，切换到其他分支会自动删除

git co $id -b <new_branch> # 把某次历史提交记录checkout出来，创建成一个分支

git br -d <branch> # 删除某个分支

git br -D <branch> # 强制删除某个分支 (未被合并的分支被删除的时候需要强制)

分支合并和rebase

git merge <branch> # 将branch分支合并到当前分支

git merge origin/master --no-ff # 不要Fast-Foward合并，这样可以生成merge提交

git rebase master <branch> # 将master rebase到branch，相当于： git co <branch> && git rebase master && git co master && git merge <branch>

Git补丁管理(方便在多台机器上开发同步时用)

git diff > ../sync.patch # 生成补丁

git apply ../sync.patch # 打补丁

git apply --check ../sync.patch #测试补丁能否成功

Git暂存管理

git stash # 暂存

git stash list # 列所有stash

git stash apply # 恢复暂存的内容

git stash drop # 删除暂存区

Git远程分支管理

git pull # 抓取远程仓库所有分支更新并合并到本地

git pull --no-ff # 抓取远程仓库所有分支更新并合并到本地，不要快进合并

git fetch origin # 抓取远程仓库更新

git merge origin/master # 将远程主分支合并到本地当前分支

git co --track origin/branch # 跟踪某个远程分支创建相应的本地分支

git co -b <local_branch> origin/<remote_branch> # 基于远程分支创建本地分支，功能同上

git push # push所有分支

git push origin master # 将本地主分支推到远程主分支

git push -u origin master # 将本地主分支推到远程(如无远程主分支则创建，用于初始化远程仓库)

git push origin <local_branch> # 创建远程分支， origin是远程仓库名

git push origin <local_branch>:<remote_branch> # 创建远程分支

git push origin :<remote_branch> #先删除本地分支(git br -d <branch>)，然后再push删除远程支

Git远程仓库管理

GitHub

git remote -v # 查看远程服务器地址和仓库名称

git remote show origin # 查看远程服务器仓库状态

git remote add origin git@ github:robbin/robbin_site.git # 添加远程仓库地址

git remote set-url origin git@ github.com:robbin/robbin_site.git # 设置远程仓库地址(用于修改程仓库地址) git remote rm <repository> # 删除远程仓库

创建远程仓库

git clone --bare robbin_site robbin_site.git # 用带版本的项目创建纯版本仓库

scp -r my_project.git git@ git.csdn.net:~ # 将纯仓库上传到服务器上

mkdir robbin_site.git && cd robbin_site.git && git --bare init # 在服务器创建纯仓库

git remote add origin git@ github.com:robbin/robbin_site.git # 设置远程仓库地址

git push -u origin master # 客户端首次提交

git push -u origin develop # 首次将本地develop分支提交到远程develop分支，并且track

git remote set-head origin master # 设置远程仓库的HEAD指向master分支

也可以命令设置跟踪远程库和本地库

git branch --set-upstream master origin/master

git branch --set-upstream develop origin/develop

##总结

Git是工具,是开发者用工具,开发者利用工具让项目的管理更加方便!开发者不要被Git所限制,不能被工牵着走!

使用Git,可以自己搭建Git服务,可以可以使用第三方提供的免费服务！例如：GitHub OSC Coding等

大家有什么好的学习Git学习心得或者方法的可以邮件1017109588@qq.com一起交流学习哦!

原文链接https://wujunze.com/git_something.jsp转载文章请保留原文链接

参考

Git官方文档

Git原理浅析

廖雪峰Git教程

等技术文档