# 学习进程

先本地，后远程

git是一个文件管理系统。

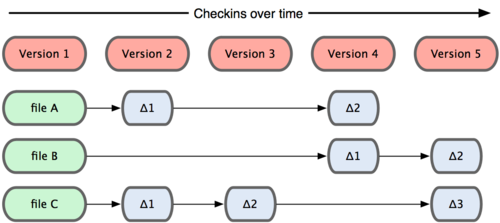
# git结构介绍

1. 功能

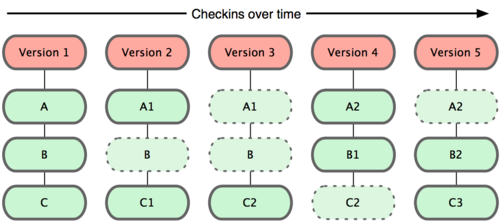
对文件的编辑历史进行保存；一系列的文件构成一个所谓的版本。git和svn都是版本控制的软件，svn属于集中控制，即所有的版本都放在集中的服务器上，所有用户共享同一版本库，需要时必须从服务器下载相应版本。git是分布式版本控制，每个人持有一个独立的版本库。

1. 原理

版本库即repository，物理上就是一个目录，一个软件仓库。git工作在一个当前目录，这个目录的所有非隐藏文件都是属于用户可见可修改的，是实时的文件，而版本都存储在.git目录下.git就是版本仓库，含stage（暂存区），master（主分支），HEAD指针🡪master.



其他版本控制软件记录差异



git记录所有的文件，只要有提交就有记录，若全无变化则不记录

**校验和**

在保存到 Git 之前，所有数据都要进行内容的校验和（checksum）计算，并将此结果作为数据的唯一标识和索引。24b9da6552252987aa493b52f8696cd6d3b00373，由sha-1算法生成。

**三个阶段**

工作目录中文件有三个状态，已修改，已暂存(add)，已提交(commit)。

分别对应三个目录:工作目录，stage目录和正式仓库目录。

使用git status查看此目录

**初始配置**

git config –system/--global分别写配置文件/etc/gitconfig和~/.gitconfig,当前配置文件在.git/config中。配置user.name 和user.email

git config –global user.name ‘ddd’

git config –global user.email ‘..@d.com’’

这两个配置在随后每次提交中都作为提交者身份被记录。

git config –global core.editor vim/emacs --设置默认的编辑器

.. merge.tool vimdiff --设置差异比较程序

git config –list 查看所有的配置

git help 命令 --查看命令的帮助

**克隆仓库**

git clone [url] [newname] –把远程仓库复制过来，当然，也可以是本地仓库

github的远程仓库必须使用git协议: git clone git://github.com/schacon/grit.git mygit

git://决定着访问服务器的哪个端口。

**忽略文件**

.gitignore 使用#注释,可以使用通配符，也可以单独列出文件

**git diff**

会调用diff命令比较文件差异,git diff –cached/--staged 分别和上次提交和这次暂存比较

**log** 查看所有的历史操作

**分支的概念**

分支代表不同的版本特性，譬如测试分支，debug分支，一个分支总是建立在其他分支之上，形如一个树的结构。

理解分支的一个重要特性是考虑分支的作用。git的提交是线性的，也就是说如果没有分支，所有的提交（一次提交构成一个版本）都将属于一个版本，很难追溯特定的提交路径。分支忠实地记录了提交的路径

HEAD指针指向当前分支，HEAD^指向上一个，HEAD^^指向上上一个,HEAD~3 表示前3个分支

创建分支 git branch <new-branch> 每当创建一个新分支时，总是继承了当前的所有commit记录。

切换/删除分支 git checkout <target> / git branch –d <target> **分支切换会改变工作目录的文件为目标的最后一次提交**

主分支是最终的版本分支，用于发布；develop分支用于日常开发；release预发布；fixbu\*修补补丁；feature定义新的功能

**git merge命令：**

git checkout target --切换到目标分支

git merge mbranch --合并其他分支到当前分支

合并的原理：一个分支就是一个commit（sha-1算法标识）链表，合并的时候，将链表的根指针指向新的分支即可。即merge从原理上来说就是简单地改变指针。

**git reset**

git reset HEAD <file> : 丢弃暂存区的内容

git reset –hard|soft|mixed|merge|kepp <commit|HEAD> --会改变分支的头指针和当前目录内容

--hard 丢弃目标commit之后的所有提交

--soft HEAD指向目标commit，目标commit之后的提交都存到暂存区，不删除

--mixed 默认的方式,仅仅改变头指针，不改变目录内容，相较于soft，所有的新内容都没有在暂存区

# git命令

**远程命令**

clone ：url格式支持ssh协议，[ssh协议可以写成user@web.com](mailto:ssh协议可以写成user@web.com)。

remote ：为了方便管理而为远程主机指定主机名，-v选项可以查看主机名—网址。remote add <hostname> <host> //host可以是本机的一个路径

push <host> <local branch> 就是一个远程的merge命令，将远程的分支指向当前的分支。

fetch :fetch <主机仓库> <分支>，例如git fetch origin master

# git底层命令

config core.crlf 显示一个属性

config core.crlf false 配置属性

git存储文件的本质实验所用的命令

git hash-object -w –stdin | file 表示从stdin读取输入,然后获取hash值,写入到git的仓库

git cat-file -p HASH\_VALUE [file] 从git的仓库中获取一个key的value值,如果提供了file参数,相当于文件恢复

cat-file -t HASH\_VALUE 获取type, 可能是blob

-t接收的类型有: blob, tree, commit, tag

也就是说git支持的hash的类型就是这4种.

git update-index 其实是add的低级接口, 它用来更新缓存.缓存是映射当前的working tree的

git write-tree 为当前缓存的working tree创建一个key-value pair,类型为tree.和git hash-object -w 类似.

git commit-tree TREE\_HASH\_VALUE 为了解决引用多个tree时需要多个HASH\_VALUE的麻烦, 可以创建一个commit对象指代一个tree对象,然后可以引用其中的文件.

引用: 在 .git/refs下, 每个文件名对应的文件中的值就是一个HASH值

典型的有 .git/refs/heads/master

.git/refs/tags/\*

使用git update-ref refs/heads/master HASH\_VALUE 来更新一个引用.

HEAD文件指向当前的引用.它一般不包含HASH值,而是一个二级引用,即形如 ref: refs/heads/master. 这种二级引用在git称作symbolic-ref.可以使用这个命令查看/更新

git symbolic-ref HEAD [VALUE] 如果VALUE(一般是另一个一级引用)没有提供,则显示

Tags

tag也非常类似于其他3中类型,只不过tag指向的是commit(注意,commit指向的是tree)

git tag NAME VALUE 来创建tag

Remote引用

记住,refs/remotes下面存有所有的引用. remote也是一种引用,大概像这样:

refs/remotes/origin/master

gc 垃圾回收

引入packfile的概念,

运行git gc 会创建packfile, 然后将原来的文件的引用重新指向此文件.

注:所有的HASH值都是指SHA-1

# 自动补全和别名

git config –global alias.<别名> <目标名> 设置git别名

在/usr/share/bashcompletion下找到git文件，将之复制到~/.git\_completion,source即可

# 远程仓库github使用和ssh指纹

1.生成一个系统用户对应的密钥，使用邮箱作为标签

ssh-keygen -t rsa -b 4096 -C [*your\_email@example.com*](mailto:your_email@example.com)

然后会提示用一个文件存储密钥。

这个生成的ssh密钥一方面存储在计算机上，一方面用在远程服务器上。

此时，使用ssh-agent –s会生成一个字符串命令，使用eval $(ssh-agent –s)可以启动这个服务器。

然后使用ssh-add <ssh-file.private>可以添加这个密文。

1. 验证

使用ssh –T [git@github.com](mailto:git@github.com)验证是否登陆成功。

1. 添加ssh的远程连接

git remote add ssh\_origin [git@github.com:yourname/yourepos.git](mailto:git@github.com:yourname/yourepos.git)

现在，你可以使用push来同步仓库。

# 常见问题

## gitignore

## 删除一个存在于已有提交中的文件

只需要将那个已经存在的commit删除即可，如果是init commit，就将.git文件夹删除即可。