Git 使用

Widnows 上面有图形化控制操作这里一般介绍实际使用代码操作：

$ git config –global user.name “liulikeqian”

$ git config –global user.email liulikeqian@gmail.com

以上是配置用户的名字以及邮箱

三种方法获取帮助

$ git help <verb>

$ git <verb> --help

$ man git -<verb>

$ git help config 一般我自己习惯这种

初始化一个目录，做好push准备

$ git init 执行以后出现README

$ git add \*.c 添加文件，以.c结尾的

$ git add README 添加README

$ git commit –m ‘initial project version’ 提交工程版本注释

推送更新到远程服务器

git [远程名] [本地分支]:[远程分支]

git push origin master

查看文件状态

git status

查看提交历史记录 git log

如果在这里退出bash重启电脑之类的，后面回到这里还是一样的结果，因为之前的信息保存在README里面。

LF will be replaced by CRLF in myvim.txt.

出现以上错误一般是换行符的问题，插入的时候 echo ‘liulikeqian’ >> mybash.txt这样就会出现以上警告，若更改如此解决： echo ‘# liulikeqian’ >> mybash.txt 即可消除warming、

$

提交更新【并未推送到服务器】

$ git commit

git commit 加上 -a 选项，Git 就会自动把所有已经跟踪过的文件暂存起来一并提交，从而跳过 git add 步骤：

git commit –m “注释” //commit只是提交并未推送，需要push

删除已跟踪的文件

$ git rm hello.txt //在缓存区移除hello.txt文件,并且删除工作目录下的对应文件

$ git rm –cached hello.txt //在缓存区移除hello.txt，保留本地工作目录对应文件

以上修改需要push提交到才能同步更新

$ git rm log/\\*.log

删除log/下所有 .log结尾的文件//不用shell扩展，git有它支持的命令，【译注：实际上不加反斜杠也可以运行，只不过按照 shell 扩展的话，仅仅删除指定目录下的文件而不会递归匹配。上面的例子本来就指定了目录，所以效果等同，但下面的例子就会用递归方式匹配，所以必须加反斜杠】

如

$ git rm \\*~

删除所有~结尾的文件，用递归匹配需要在\*前加\

//如果clone错了仓库，那么可以使用 $ git clean –fd untracked 目录

文件移动：

$ git mv nameA nameB //git不跟踪移动的文件

$ git mv ⬄等价于

$ git nameA nameB

$ git rm nameA

$ git add README

$ git log //查看历史操作记录，默认显示注释、SHA-1校验和、作者(email)、时间、版本号

其中

$ git log –p -2

-p：显示每次提交的差异

-2：近两次提交的更新

$ git log --stat //显示简要的增改行数统计

$ git log –pretty=online

--pretty：可以指定使用挖暖不用于默认的方式展示commit历史，

online：每个提交放一行显示

format：可以定制要显示的记录格式

比如：

$ git log –pretty=format：“%h - %an， %ar ：%s”

上述命令日志显示：

提交对象的尖端哈希字符串、作者名字、作者修订日期，按多久的方式、提交说明【注释】

**选项 说明**

**%H 提交对象（commit）的完整哈希字串**

**%h 提交对象的简短哈希字串**

**%T 树对象（tree）的完整哈希字串**

**%t 树对象的简短哈希字串**

**%P 父对象（parent）的完整哈希字串**

**%p 父对象的简短哈希字串**

**%an 作者（author）的名字**

**%ae 作者的电子邮件地址**

**%ad 作者修订日期（可以用 -date= 选项定制格式）**

**%ar 作者修订日期，按多久以前的方式显示**

**%cn 提交者(committer)的名字**

**%ce 提交者的电子邮件地址**

**%cd 提交日期**

**%cr 提交日期，按多久以前的方式显示**

**%s 提交说明**

**选项 说明**

**-p 按补丁格式显示每个更新之间的差异。**

**--stat 显示每次更新的文件修改统计信息。**

**--shortstat 只显示 --stat 中最后的行数修改添加移除统计。**

**--name-only 仅在提交信息后显示已修改的文件清单。**

**--name-status 显示新增、修改、删除的文件清单。**

**--abbrev-commit 仅显示 SHA-1 的前几个字符，而非所有的 40 个字符。**

**--relative-date 使用较短的相对时间显示（比如，“2 weeks ago”）。**

**--graph 显示 ASCII 图形表示的分支合并历史。**

**--pretty 使用其他格式显示历史提交信息。可用的选项包括 oneline，short，full，fuller 和 format（后跟指定格式）。**

限制输出长度：【-n的参数少用，git提供自动翻页的功能，要使用的时候手动翻页即可】

$ git log --since=2.week

$ git log –pretty=”%h:%s” –author=gitster –since=”2018-01-01” \

--before=”2018-10-10” --no-merges -- t/

**选项 说明**

**-(n) 仅显示最近的 n 条提交**

**--since, --after 仅显示指定时间之后的提交。**

**--until, --before 仅显示指定时间之前的提交。**

**--author 仅显示指定作者相关的提交。**

**--committer 仅显示指定提交者相关的提交。**

撤销操作，注释提交错误的时候或者遗漏几个文件没有添加，可以是用撤销操作，有些撤销是不可逆的

$ git commit --amend

此命令将使用当前的暂存区域快照提交。如果刚才提交完没有作任何改动，直接运行此命令的话，相当于有机会重新编辑提交说明，但将要提交的文件快照和之前的一样。

启动文本编辑器后，会看到上次提交时的说明，编辑它确认没问题后保存退出，就会使用新的提交说明覆盖刚才失误的提交。

如果刚才提交时忘了暂存某些修改，可以先补上暂存操作，然后再运行 --amend 提交：

“改写提交内容，使用上述命令”

Tips:描述很微妙需要手动去操作才知道怎么的结果

误用了add,将两个修改后的A.txt,B.Log文件添加到一起，本来是要分开提交的，如下操作，

$ git add .

$ git status

|

A.txt

B.Log

|

-----------🡪

$ git reset HEAD A.txt A.txt:locally modified

$ git status

-----------🡪

如下撤销文件的修改，此前所有修改的撤销，有些危险，小心操作：

$ git checkout – A.txt

$ git status

文章说后面【第九章】使用stashing和分支类处理会比较好

远程仓库的使用:

查看当前的远程库：

$ git remote