

Github 怎么使用？

Github是面向开源和私有软件项目的托管平台，程序员编写完成的代码可以托管到这个云平台,方便储存,更新,开源分享。一个工程项目从开始编写到完成的过程中会经历多次修改，形成多个版本（version）, git就是一个版本控制系统，可以对不同版本进行操作,修改,来回切换,方便代码的编写与管理。

1. Git的安装
2. Windows或者mac 直接百度搜索 git 在Downloads页面选择合适的版本安装下载，完成后 运行git bash 进入控制台就可以进行操作了
3. Linux用户直接 apt-get install git 或者yum安装，之后直接在控制台进行指令的输入
4. 进行简单的操作

（1）cd 命令切换当前工作路径（用过控制台的比较熟悉了）

例如 在控制台中输入

cd /home/Kaggle

即切换当前工作路径到home目录下的Kaggle目录

1. 在当前工作路径向，比如在Kaggle文件夹内， 输入

git init

命令创建一个初始化一个本地仓库，完成后当前目录（Kaggle）就是一个git仓库了

（4）在命令行中输入 git status 可以查看当前git仓库的状态

3.进行仓库的管理

你有一堆文件，向提交到仓库里面，保存起来怎么办呢？

Git规定了这样的使用流程：

工作区相当于在文件夹里，使用命令：

git add *your file name*

就可以将指定的文件提交到暂存区（这里你可以提交很多个文件到暂存区，只需要重复使用以上命令即可） ps：add一个文件之后文件的状态就被保存了，这时候如果你修改文件，暂存区中的文件不会被修改

这时候就可以用 git status 命令查看仓库的状态了。

当然，如果不小心 add 错了文件，可以使用命令：

git checkout -- *your file name*

撤销指定文件在工作区的修改

设置暂存区相当于给你一个反悔的机会，减少误提交的几率，在需要的文件全部的 add 完成之后，输入命令：

git commit -m *“note”*

提交你暂存区中的文件到你的git仓库里面，用 git status 就可以查看啦

这里“note”是用以表述说明这次提交的一个字符串，例如“增加了bug”，“删除了源码”，方便你的合作者了解你的提交

一不小心把错误的文件提交到了git仓库而不是暂存区怎么办呢？git 还提供了版本回退的功能。输入命令：

git reset --hard HEAD^

就可以把git仓库还原到上一次提交（commit）的版本，相当于撤销了这一次提交的操作。如果想要一次性退回前两个版本，前三个版本该怎么处理？

其实每次提交的信息，git都把提交的信息写进了一个日志里面，输入命令：

git log

就可以查看日志

其中commit后面一长串的字符串是每一次提交对应的一个id，如果你想回到id对应的那个版本，只需输入命令

git reset --hard *id*

id代表你想回到的版本id，如果id太长，可以只输入id的前几个字母，只要能够区别出你选择的id即可

想查看自己输入的每一个命令有一种更加简洁的方法，输入命令

git reflog

4.配置远程库

仅仅限于本地仓库没有任何意义，仓库应该提交到云服务器上才方便进行修改和分享。

1. 首先，设置自己的github账号和用户名。输入命令：

git config --global user.name *“你的用户名”*

git config --global user.email *“你的邮箱”*

1. 添加远程库

设置你的本地仓库要提交到云端的那个仓库（可以是别人的仓库，但是需要权限）

输入命令：

git remote add origin *“远程仓库的地址”*

添加远程库,这里origin是远程版本库的名称，默认为origin

git remote -v

查看已经添加的远程仓库

1. 添加完远程库，就可以进行和远程库相关的操作了

输入命令：

git push origin master

可以将本地仓库的内容提交到刚才 remote add的远程仓库里面的master分支（分支的内容后面会提到，可以看成一条记录着你每次修改和提交的时间线）

如果需要下载远程仓库的代码到本地仓库

输入命令：

git pull origin master

或者

git fetch origin master

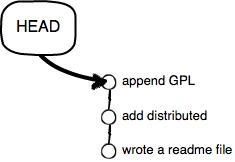
pull 和fecth的区别是pull相当于先fetch取得远程代码，然后再merge（合并分支，后面有介绍）

当然还可以克隆一个指定仓库，无需 remote add（通常是别人的远程仓库），输入命令：

git clone *address*

就可以把相应地址上的远程仓库复制到本地了

5.分支的管理

1.前面提到版本回退，其实原理就是每一次提交对应一个地址，通过改变头指针的指向就可以很容易地在版本之间切换。

git 里面根据提交的时间有一条主的时间线，类似于一个数轴，记录着你的每次提交，成为master分支（主分支）。在主分支上我们可以自主创建其他的分支，并且在其他分支上进行修改等操作而不影响到主分支。

（红色的为我们创建的dev分支）

我们可以在dev分支上进行修改，完成之后再合并到master主分支。以此来完成相对来说安全的操作。



输入命令：

git branch *branch\_name*

创建一个branch分支

输入命令：

git checkout *branch\_name*

就可以切换到指定名称的分支作为当前分支，在该分支上进行你想进行的操作

当然也可以输入命令：

git branch -b *branch\_name*

完成创建和切换的一步到位

输入命令：

git merge *branch\_name*

就可以合并指定的分支到当前分支

合并完成之后，我们就可以删除多余的无用的分支了，输入命令：

git branch -d *branch\_name*

可以删除指定名称的分支

2.前面并没有讲到merge合并的具体操作，一般来说两个分支对于同一个所作不同的操作在合并以后，如果没有矛盾（对同一文件同一内容进行修改），则合并之后得到的是两次修改的合效果，如果存在冲突，git就会报错，这时候需要手动解决冲突。

解决的方法是：在当前分支对该文件冲突的地方手动再次进行一次修改，作为你的最终觉得，然后提交，之后再进行合并。

（一个比较直观的效果）

完成之后删除分支即可

1. bug分支

如果你创建了一个分支，并把已经完成的文件提交到了暂存区，但是还没有提交到仓库，这时候你发现一个bug需要修改。为了不因再创建一个分支修改bug造成分支的混乱，输入命令：

git *statsh*

把你的暂存工作区隐藏起来（类似与备份），这时候用git status 查看你的暂存区已经是空的了。这是你就可以方形地按照上面的方法创建分支，修改，合并分支，删除分支

最后输入命令：

git statsh list

查看被隐藏的工作区

然后输入命令：

git statsh apply

git statsh drop

恢复工作，删除隐藏备份起来的内容

当然还有一个等价的命令：

git statsh pop

一次完成恢复和删除工作

需要注意的是，没有合并的分支是不能被删除的，如果必须删除，输入命令：

git branch -D *branch\_name*

强行删除分支

6..标签操作

前面提到过 commit 操作的note参数是用于给提交添加附加说明，这里介绍另外一种操作

输入命令：

git tag *tag\_name commit\_id*

就可以给id为commit-id的那次提交打上一个名为tag\_name 的tag标签

打完标签以后，就可以输入命令：

git show *tag\_name*

轻松地检索到指定标签所对应提交的相关信息

标签写错了也删除标签。输入命令：

git tag -d *tag\_name*

可以删除指定的标签

还可以将本地标签全部推送到远程仓库

git push origin --tags

以上只是说明了git 指令的一些基本操作，在这之上可以添加一些参数。

例如 -u ,-m等以实现其他的更加精细的调整，上面也部分一些指令加入了这些参数，这里就不做过多的介绍。

End

**有不懂的地方可以参看廖雪峰的git教程，这里的内容几乎都是来自哪里**