Git基本功能

之前一直注册了github帐号，也不断在更新自己的代码，不过上传下载代码一直都是通过网页版进行的，非常麻烦。而且一直也没能完全弄清git和github的联系与区别，最近随着代码越写越多，还是有必要学习下git的基本功能的。对于工具的学习，我一向坚持刚开始只学需求范围内的内容，高级功能以后有需求再学。

首先说git，它是分布式版本控制系统，版本控制简而言之就是在开发时分成多段开发时候对各阶段当前代码进行控制和管理，不仅可以更新最新的代码还可以回退到之前的版本。分布式其实很容易理解，就是一个工作被分配在多台对等功能的计算机上协作完成处理工作，如果开发过程中遇到分工协作的话，对这个概念会理解更深刻些。而github实际上原本和git本没有什么直接联系，它只是免费的远程仓库，相当于一个帮助我们收集汇总代码的服务器，之前把github当做网盘来用实在是太蠢了。

所以基本框架可以理解成：在一个多人协作工程中，每个人可以在本地创建一个git的代码仓库，各人可以独立地在自己的仓库下写代码，git提供了版本控制的基本功能。一般需要一个专门的git服务器用来收集项目的全部代码，可以自己搭建也可以使用github的免费服务，不过在不付费情况下，托管在github中所有代码都是公开的。每个人都可以将github代码库中的代码克隆到本地进行修改并提交到上面（如果有权限的话），同样也可以fork别人的项目到自己的github仓库下，修改别人的代码并通过pull request向原拥有者提出提交申请，得到通过的话，对代码的修改可以被添加到别人的代码库中。

相比于其他很多分布式版本管理系统而言，git一个很大的优点是可以在没有网络的情况下工作。所以我们先从本机下使用git入手，最后逐步使用github。

本文的环境是ubuntu 16.04，所有的操作均为命令行操作。

首先是安装git，如果在命令行下输入git能够显示git的具体使用说明，则表示本机已经安装了git，可以直接进行接下来的操作。如果没有可以输入apt-get install git完成git的安装，安装之后再输入git检查是否已经完成。

安装完成之后需要设置用户名和密码，输入如下指令：

$ git config --global user.name "example\_name"

$ git config --global user.email "example\_mail"

这里的global表示在本台机器上所有的仓库都会使用这个配置，通过global完成的配置会被保存在本用户home目录下的隐藏文件.gitconfig中。

接下来需要做的就是创建版本库，我们需要创建一个目录，暂且起名叫code，之后进入code目录，输入以下目录就可以将其变为git管理仓库了：

$ git init

往这个仓库中提交代码的过程有三个步骤：编码保存，add，commit。这三个指令分别对应不同的存储区。



图1：工作区、暂存区和分支

我们编码工作完成在工作区，当输入git add之后便将代码提交到了版本库的暂存区，只有在暂存区的文件才可以被commit到分支。所以当我们完成在工作区的编码工作是没有办法直接将编辑完成的代码直接添加到分支去的。如果输入commit能够成功，提交的一定是之前add到暂存区但尚未添加到分支的文件。

另外需要说明的是，暂存区的设计理念实际就是一个缓冲区。所以我们可以每次编辑完一个文件之后就add到暂存区，这样暂存区就会缓存下多个多次添加的文件，之后可以由commit一次性提交所有的代码到分支中。

在提交代码的时候，需要加上-m和本次提交的说明，这对我们接下来进行版本控制尤为关键，基本的指令如下：

$ git add file1

$ git add file2

$ git commit –m "add two file"

按照上面的操作file1和file2已经被提交分支中，此时如果再对file1修改，我们可以通过下面命令来看出当前文件仓库的状态以及被修改文件与已前一个版本的file1之间的区别：

$ git status

$ git diff file1

其中，只要当前工作区的代码进行改动，无论是修改还是删除，通过git status都可以看到哪些文件与版本库中文件相比做了改动。而git diff则是用来比较二者区别的。在确认完对文件的改动后，如果要提交到分支，则还需要通过add和commit两个步骤来完成。

如果某个文件已经被提交到分支内，此时删除了工作区的文件的话，此时输入git status会显示该文件已经删除，需要通过以下两条指令来将版本库中该文件也删除，否则会一直有状态提示。删除的指令是：

$ git rm file1

$ git commit –m "remove a file"

版本控制就像是给每个状态的文件做了快照，所以当我们进行了不当修改或者误删时，可以找回之前的版本，而且可以回退到之前的所有次更改的版本。我们可以通过下面的指令来查看之前提交（commit）的版本：

$ git log

通过git log可以看到之前每次提交的记录、提交说明（-m之后的内容，用来提示版本信息）以及git为每个版本生成的哈希值。我们可以通过以下方式回退到之前的版本：

git reset --HEAD^

git reset –HEAD^^

git reset –HEAD~k

第一中表示回退到上一个版本，第二种表示回退到上上一个版本，如果要往前回退k个版本，就通过HEAD~k来实现。回退完之后，首先版本库分支中的文件会回退到指定的版本，另外会将分支中的数据也同步到工作区中。这里具体的原理就不说了。返回版本的指令不仅仅之后通过HEAD指针一种，还可以通过输出的哈希值来直接完成，比如如下方式：

$ git reset HASH（部分即可，具体见log显示）

如果上一步回退多了还有补救方式，就是通过下面指令找到待恢复版本的哈希值用于恢复以前的版本：

$ git reflog

$ git reset HASH

刚刚说的是已经提交到分支的文件的版本恢复，实际上只通过git add添加到暂存区的文件同样可以用下面的指令把暂存区内容恢复：

git reset HEAD file

还有一种情况就是文件仅仅在工作区进行了修改，想要放弃该修改，使工作区文件恢复到之前的状态可通过下面指令完成撤销：

git checkout –file

以上分别列出了已经提交到分支、提交到缓冲区和在工作区的版本回退问题，足以满足日常基本操作。

下面说一下git远程连接github的问题。首先github提供了免费的代码仓库功能，我们可以将所有本地提交到分支内的仓库push到github中，所有被提交到github中的仓库就成了公开状态，可以被分发使用。尤其需要说明的是，能够提交到github中的代码都是在本地经过add、commit被提交到分支的代码。

如何连接github的问题网上有详细的教程，这里就不再多说了，只要有github帐号都可以享受免费服务。

比较常用的功能就是将github上代码通过git clone下载到本地进行修改，修改完成后在通过git push推送到远程github服务器。因为在大部分工作都是本地的git操作，所以连接github之后通常只是上传代码改动和下载代码的操作。

顺便提一下，默认的github连接是通过https传输数据的，所以每次commit都需要输入github账户和密码。有两种常见的解决方法：一种是将https传输方式改为ssh，另一种就是下载ksshaskpass来帮我们保存帐号密码，第一次输入过之后以后每次都可以避免相同的输入操作了。

Have a fun.