今天重新学习总结了一下git

磨刀不误砍柴工，我觉得好好理解一下工具我觉的还是很有必要的。

学习参考的是廖雪峰大神的教程，以下只是我学习的一些总结。

大神的博客 http://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000

安装好git之后就可以在自己的电脑上创建属于自己的版本库了，

版本库，可以添加，修改，删除，回滚到以前任意版本等非常强大的功能

以前只是无脑的从gitlab上clone下项目来，然后每次都是无脑的上传

平时常用的命令和含义

Git status 检查当前版本库的状态,

Git add –all

Git add +名字 这两个命令是添加修改到版本库，第一个是无脑的把所有的都添加进去

第二个就可以选择性的添加了

Git commit –m’此处填写注释’ 这个命令是将刚才的修改提交的版本库，并备注，

当然，如果不用添加到远程版本库，到这里也就完成了。

如需添加到远程版本库 ，还要下面的两条命令。

Git pull 这个是从远程版本库拉代码，看远程版本库是否更新，在使用push命令之前

最好先pull一下，可以防止上传时的冲突

Git push’oringin master 将刚才的修改推到远程版本库，到这里才真正完成了一次上传

第二波学习总结，这次主要是版本回滚的一些命令

Git log 查看每次提交的版本日志



Git reset –hard HEAD^ 这个命令是提示回滚到上衣个版本，最后的^指上一个版本，

如果最后有两个^ 就是回滚到上上个版本，如果版本太多要出入

很多^的话，可以简写 比如说要回滚到倒数第56个版本

git reset –hard HEAD~56 即可

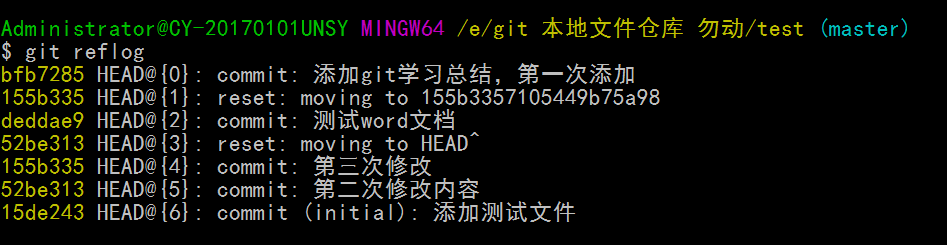
git reset –hard +版本号 那 如果有时候要回退的版本我们不好数是倒数第多少个，

这时候直接看日志里面的commit 后面的一长串字符就可以了

这里的字符就是代表这个版本号，当然，版本号不用输全，输入前几位就可以了，git会自动去找

Git reflog 这个命令是看我们所有的操作日志，包裹回滚的日志，

(git log是看不到回滚的日志的)，主要用于，我们回滚之后后悔了，又想回到原来的版本，就可以用这个命令看到以前的版本号了。



第三波学习总结，暂存区和分支

先理解概念，

工作区，就是我们能看到的文件目录

版本库，我们有一个隐藏的文件夹 .git 这个文件夹里面就是git的版本库了

版本库里面又分两个部分，暂存区和分支

我们第一波总结的地方有两个命令，git add 和 git commit

Git add 其实就是讲我们的修改提交到暂存区，这个时候修改还没有到版本库

Git commit 这个命令才是将我们暂存区的修改一次性的提交到版本库，这时候才安全。

今晚最后一点，写完睡觉

理解git管理的修改，git管理的其实是修改的内容，而非文件，也就是说，新增，或删除一行，都是修改，新建或者删除某个文件也是修改，每个操作git都会追踪到，git非常强大。

第二次学习，两条回退的命令

Git checkout – file(文件名字) 这条命令我觉得有点鸡肋，可能有时候有用，就是将你工作区还未提交到缓存区的内容回退掉，这个命令用于还未 git add时

Git reset HEAD file(文件名字) 很多时候这个命令配合上面这条命令用，这个命令的意思是讲我们添加到缓存区的内容退回到工作区 就是将 git add了但没有git git commit 的状态回退成 git add 之前的状态

第三波，本地git仓库和远程git仓库，

我们在github上创建一个仓库之后有两个方法可以关联到本地

第一个方法已经弄明白了，第二个还不太明白

先说第一个，我们直接在github上创建一个项目，然后 git clone到本地，然后再正常的上传就下拉就可以了，可以直接使用git pull和git push

第二种是导入本地的库好像，我们在github上创建空的库之后，在本地创建文件夹，然后

Git init 添加一点文件 git add –-all git commit –m’xxx’

git remote add origin //这里不是很明白

然后 git push origin master 这样也可以，此时本地的文件夹和github的仓库名字可以不一样

还有就是，自己玩了玩发现文件夹的名字真的不重要，只要这个文件夹里面 git init过就可以了，还有就是要和远程库关联，

关联一种就是直接git clone 另一种就是现在还不是特别明白的git remote add origin