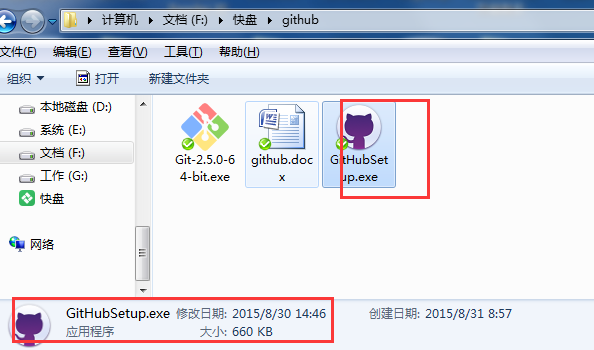
一：git 简单的操作

首先安装github

去github官网上面下载



桌面上会出现



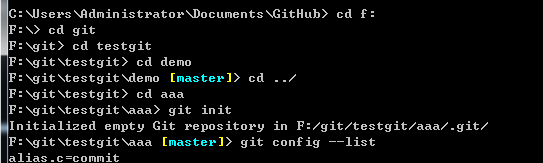
打开Git Shell

1 进入指定的文件夹

cd e:



进入到文件夹目录下面之后执行：git init

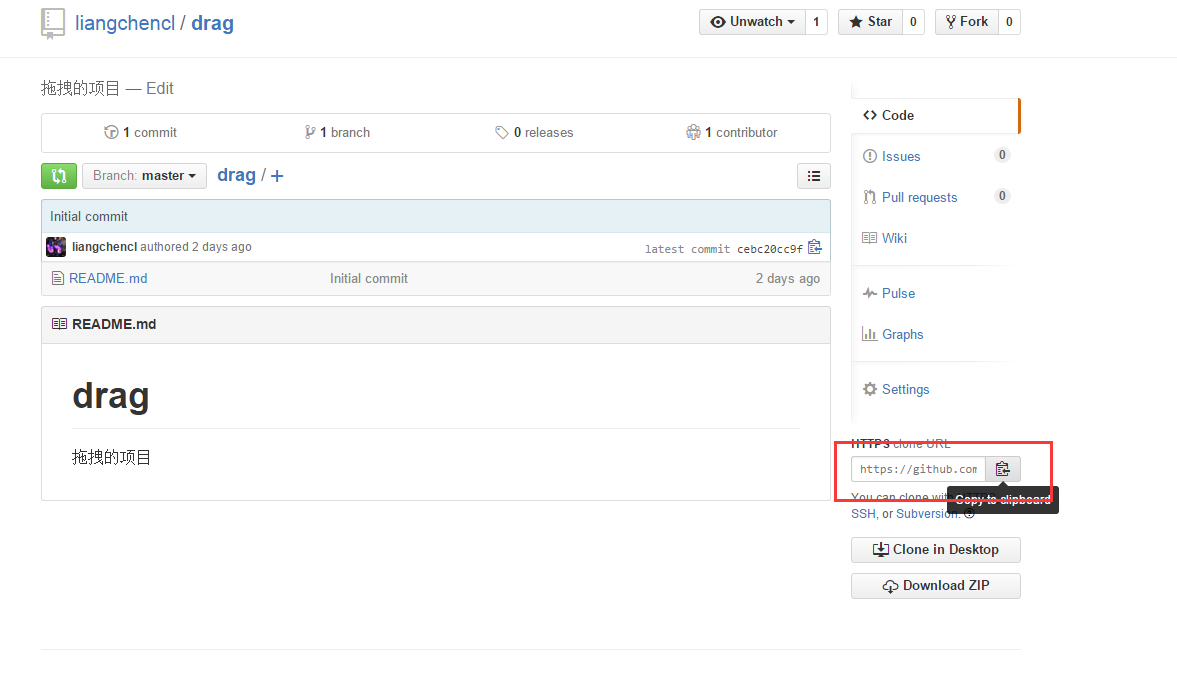


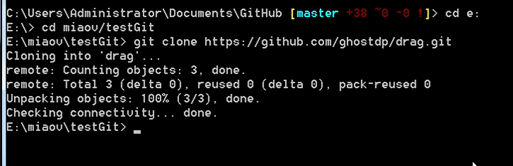
该命令执行完后会在当期目录生成一个 .git 目录。

使用我们指定目录作为Git仓库。

克隆：

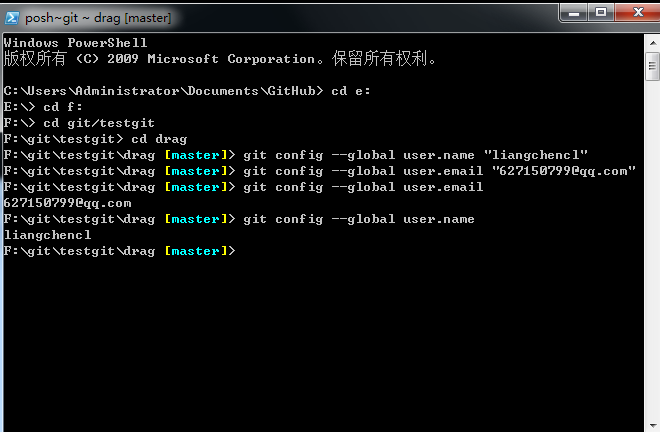
这是github上面的地址：



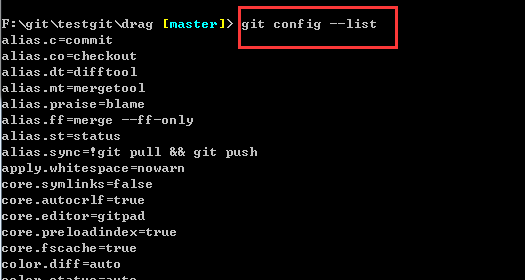


设置用户名邮箱并且查看用户名邮箱！

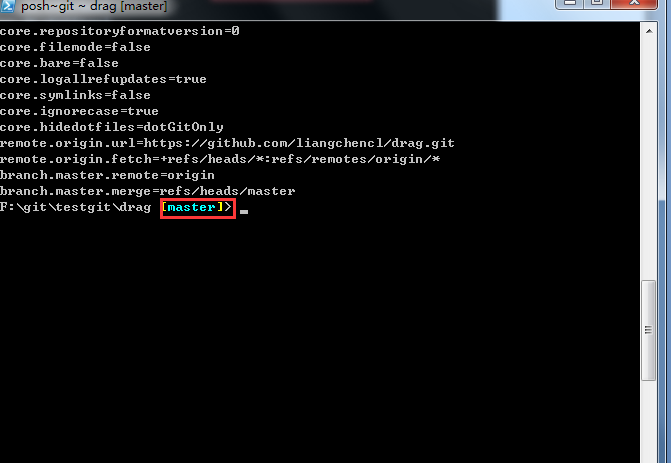
注意：要先进入到文件夹里面去;

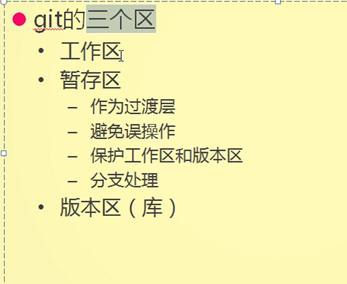


查看所有信息：



master ：主分支的默认名字！







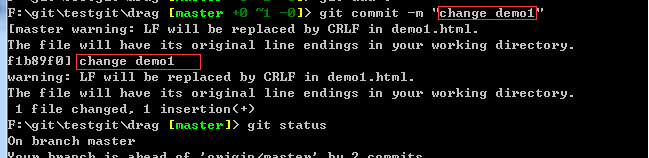
git status 命令是查看工作区和暂存区的当前的状态 （这个要经常用）

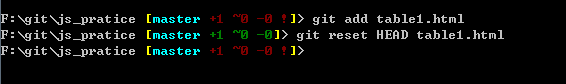
git add 文件名(demo1.html) 把demo1.html 提交到暂存区

git add . 提交所有修改过的文件到暂存区中

git commit 把文件从暂存区提交到版本库 运行这个命令后会弹出一个记事本一样的编辑器，这个是写注释的！

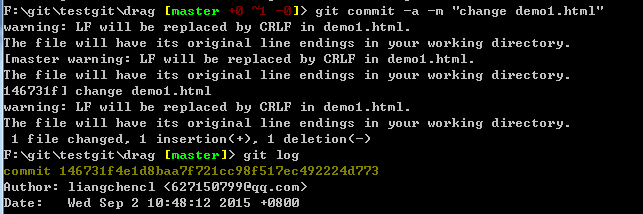
git commit -m "change one" 把文件从暂存区提交到版本库 运行这个命令后不会弹出记事本，





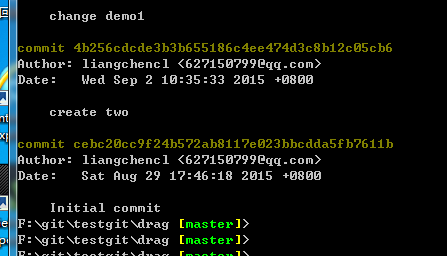
git reset HEAD demo.html 把demo1.html 从暂存区撤会工作区

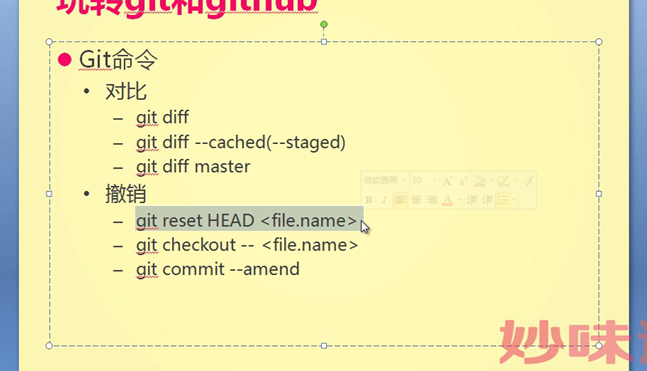
（红色表示在工作区有变动，蓝色表示提交到了暂存区）

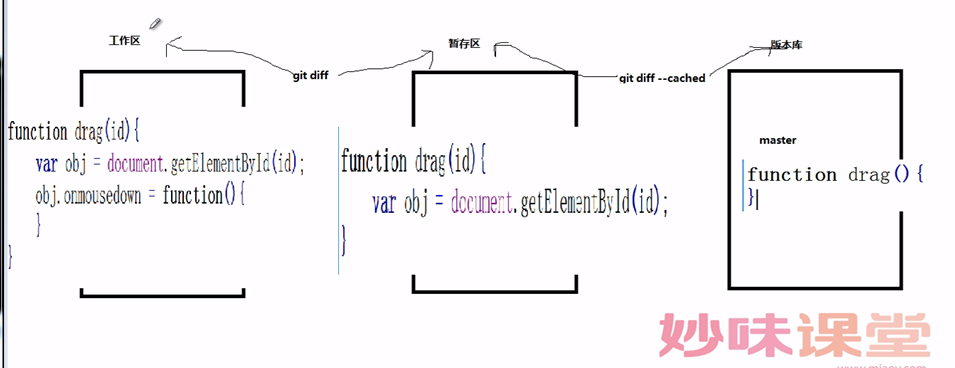


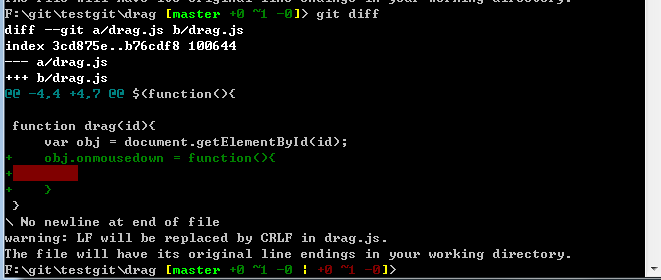
git commit -a -m "change demo1.html" 这个是简写的，把工作区的文件提交到暂存区然后到版本库！

git log 查看记录，这里只是显示一部分，可以点击enter 查看，如果想退出的话就直接按q

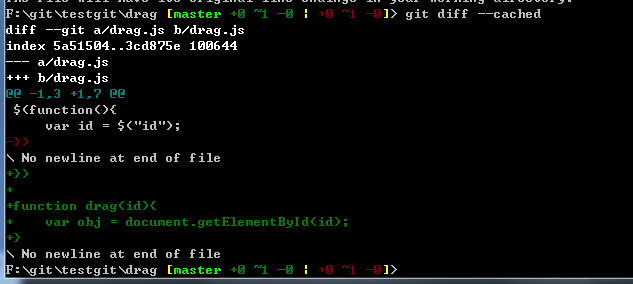




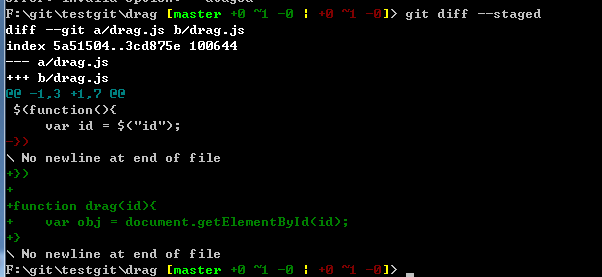




git diff 工作区和暂存区之间的差异对比



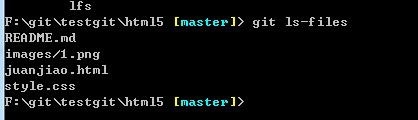
git diff --cached 暂存区和版本库之间的对比



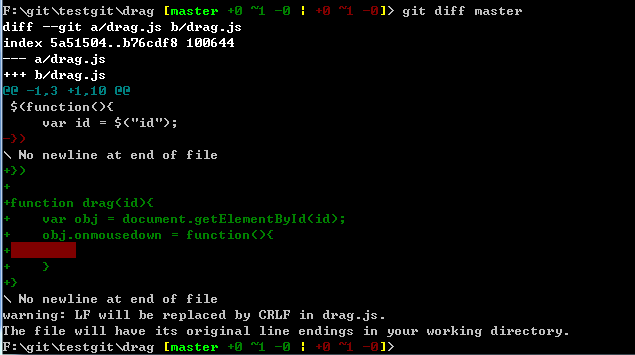
git diff --staged 暂存区和版本库之间的对比 (和上面的是一样的，这两个命令功能一样)

|  |
| --- |
|  |

git ls-files 显示工作区和暂存区的文件



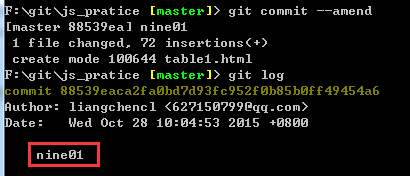
注意： ls-files 中间没有空格



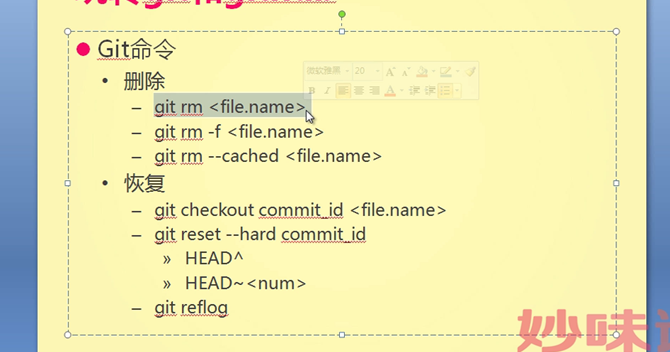
git diff master 工作区与版本库之间的差异对比

git checkout -- drag.js 还原工作区里面的内容到版本区的状态 执行之后就跟版本区里面的内容是一样的 (可以理解为：版本区里面有一个版本，然后在工作区修改了的，之后发现修改的不对，执行这行命令之后就还原到版本区了！)

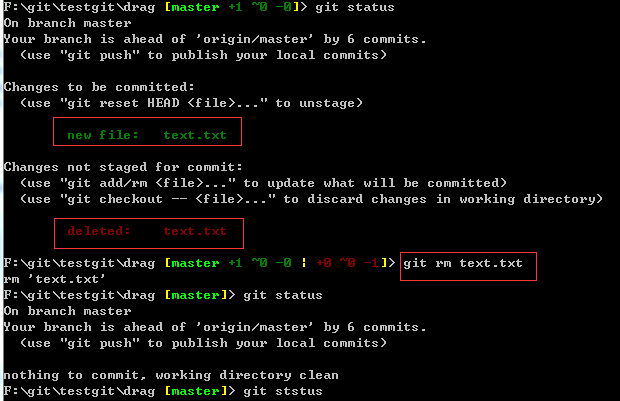
 git commit --amend 修改最后一次提交的信息



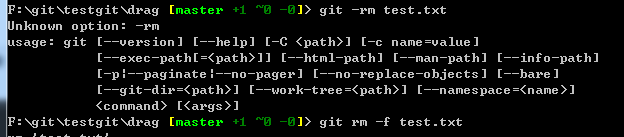
运行 git commit --amend 之后会弹出一个记事本，直接修改第一行的（这里是把nine改成了nine01）内容然后直接关闭，再运行q 退出，然后运行git log 查看提交的信息，就可以看到最后一次的提交信息已经修改了



git rm text.txt 针对 在暂存区里面有但是工作区里面没有的文件进行删除



如果工作区和暂存区都有text.txt 文件的话执行这句话是错误的，会报错



所以：如果工作区和暂存区都有test.txt 文件的话 可以执行

git rm -f test.txt



git rm --cached test.txt 这行命令是删除暂存区的内容，但保留工作区的内容

这是和上面的那行命令（git rm -f test.txt）的不同

**区别：**

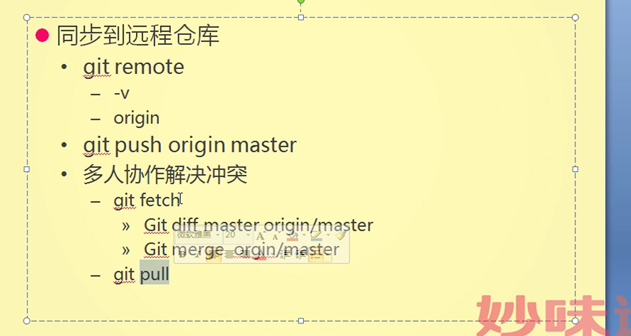
**git rm -f test.txt**

**这行命令是删除暂存区和工作区里面的内容(全部删掉了)**

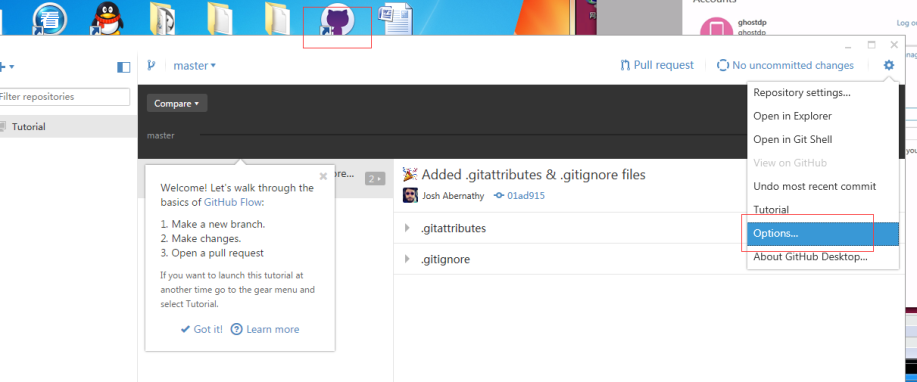
**git rm --cached test.txt**

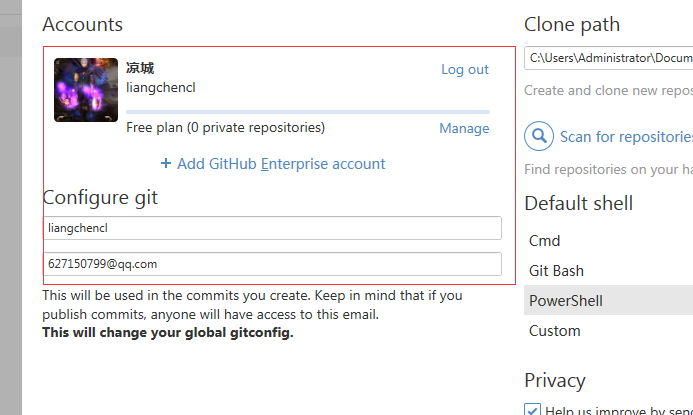
**这行命令是删除了暂存区里面的内容，但是工作区里面的还是保留着的！**

同步到guthub 上面：



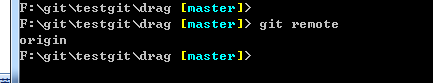
先查看下登陆的帐号：



****

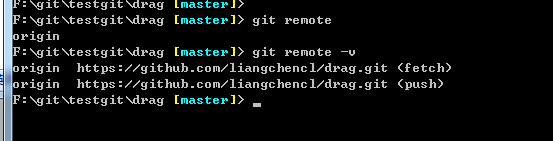
git remote

查看远程仓库的名字： (这个是可以改的！)



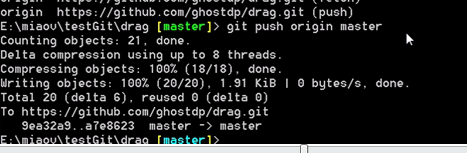
查看远程的地址

git remote -v

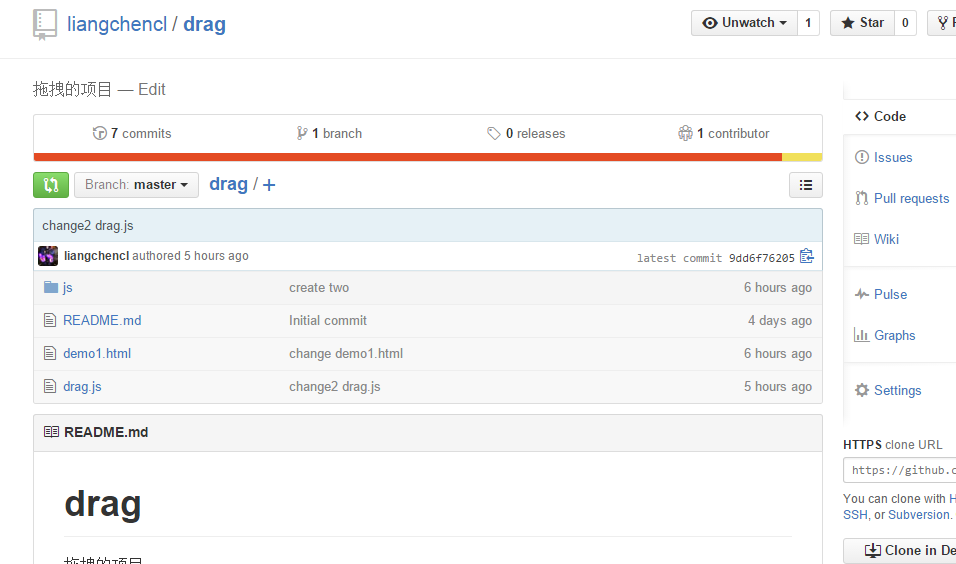


同步到github 上

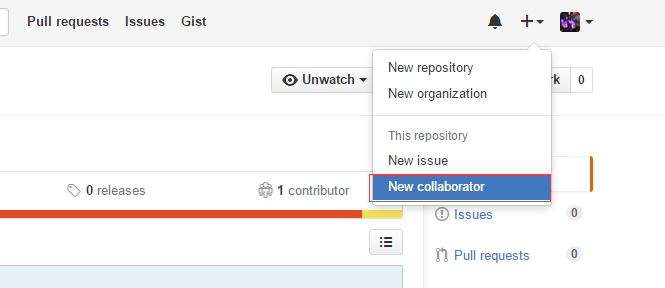
git push origin master

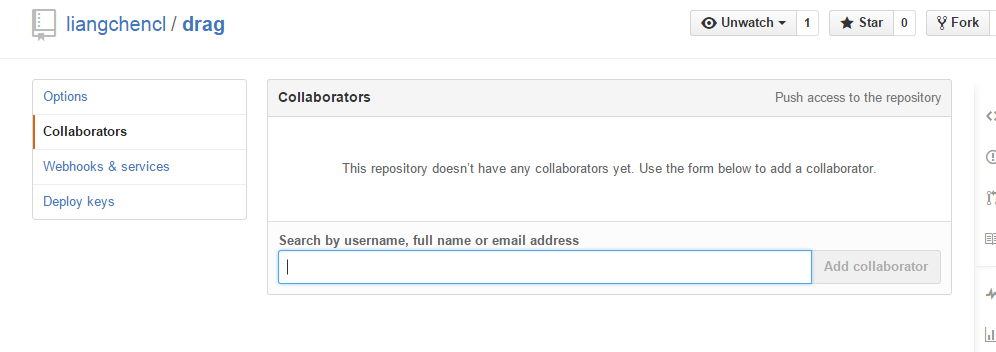


然后去github官网上可以看到：



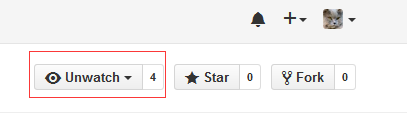
添加新成员：



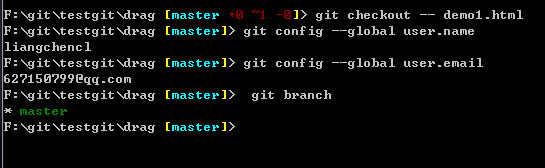


搜索用户！然后添加上去，这里会自动发一封邮件到对方的邮箱里面！

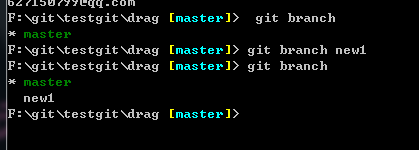
这里添加的用户就是让该用户关注这个项目！

添加了之后可以看到  这里会加1.也可以点进去看下成员

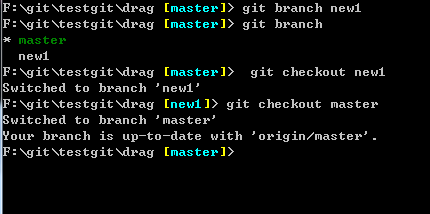
* **star** 的作用是收藏，目的是方便以后查找。
* **watch** 的作用是关注，目的是等作者更新的时候，你可以收到通知。
* **fork** 的作用是参与，目的是你增加新的内容，然后 Pull Request，把你的修改和主仓库原来的内容合并。



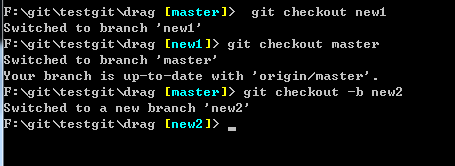
git bracnch 查看分支，前面的那个\* 表示当前的分支



git branch new1 创建new1 这个分支！



git checkout new1 切换到new1 这个分支上面

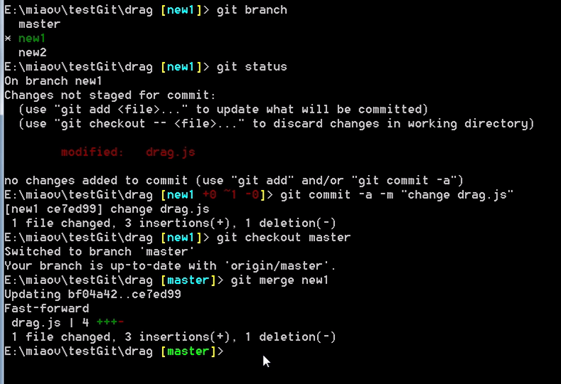


git checkout -b new2 创建new2 分支冰切换到这个分支上面

切换到new1 这个分支，然后改一写drag.js里面的内容

然后提交到版本库：

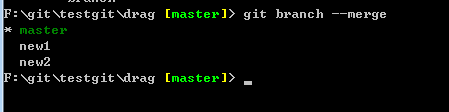
(由于是在new1分支上面改的内容，所以master分支上面并没有，还是以前的内容，这个时候new1分支和master分支上面的内容就不一样了)



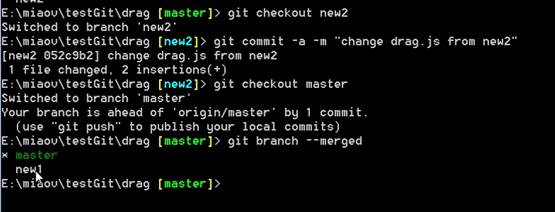
合并分支：

切换到master分支上面，执行：

git merge new1 把new1分支合并到master分支上面去！



git branch --merge 查看当前分支所合并的分支！就是在同一级目录下的



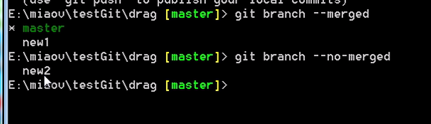
如果切换到new2分支，然后在更改一些内容，然后提交：然后再切换到master分支上面：

运行：

git branch --merged

这样就显示如上图：

是因为new2分支已经提交了一次，new2分支已经不再和new1，master分支在同一级了！

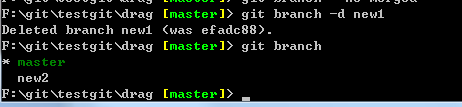


查看不在同一分支下的：

git branch --no-merged

删除分支：

git branch -d new1



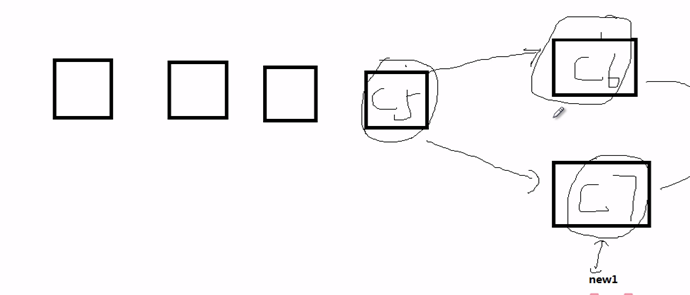
注意：如果两个分支有区别（例如这里的master分支和new2）是不能用这个命令删除的

如果想强制的删除不同的分支（new2）：

执行： (把小写的d改成大写的D即可)

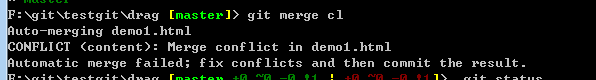
git branch -D new1

合并有冲突的分支：



切换到master分支上面 然后执行

git merge new2 (这里的cl代表new2)会提示报错：

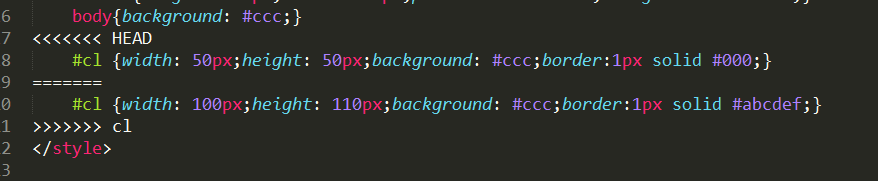


原因是mester分支和new2分支里面的文件有冲突，是不能智能合并的！

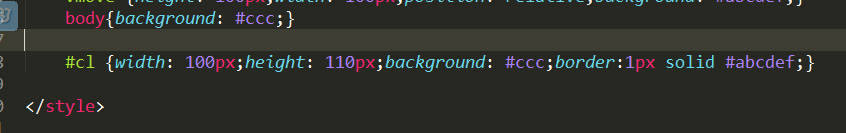
**解决冲突：需要手动解决：**

git status 命令查看冲突，

也可以直接在编辑器里面查看！



然后手动删除不要的即可！



保存！然后再提交一次即可：



然后可以删除new2这个分支，当然也可以保存起来！

合并分支：git merge 分支名字

git status 查看文件之间的冲突！



+ 加号 表示有添加

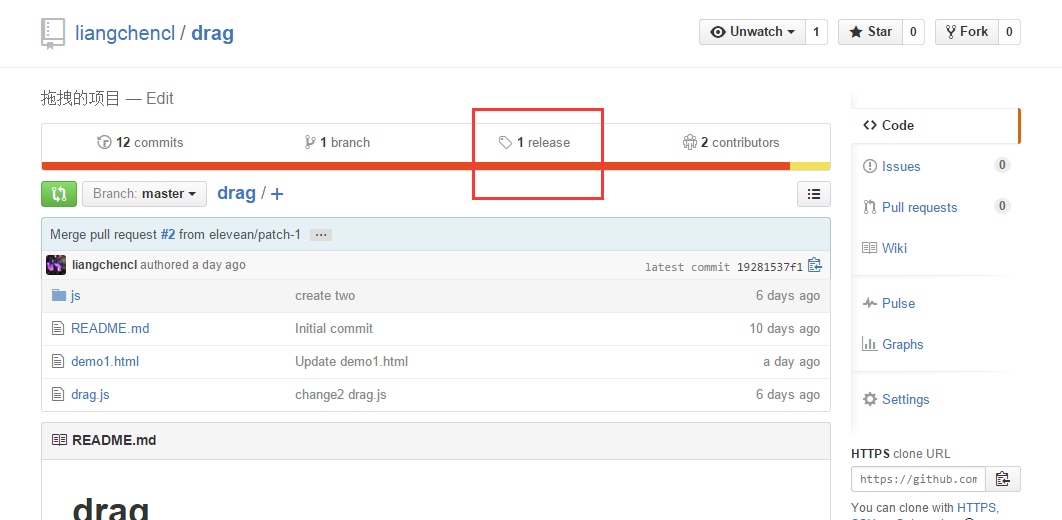
~ 波浪线表示有修改

— 减号表示有删除

！ 感叹号表示有冲突

蓝色表示暂存区

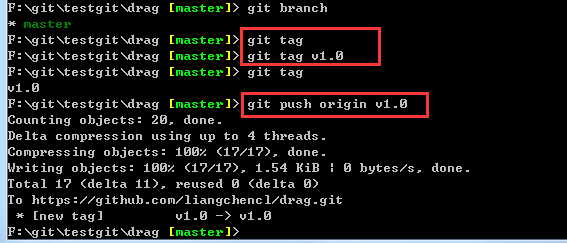
红色表示工作区



上图中的1 release

在github上面这里相当于是一个版本库：

**运用命令添加版本库：**



运行：

git tag 查看当前有没有版本库：

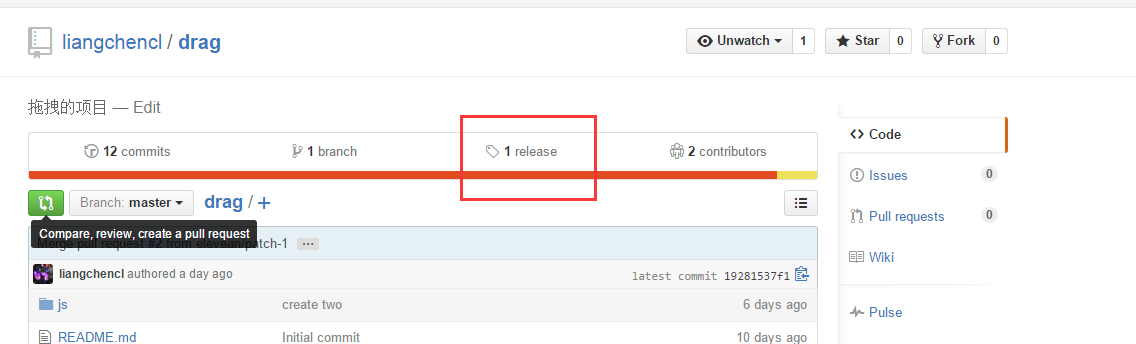
git tag v1.0 添加版本库名字叫做 v1.0

git push origin v1.0 把这个版本库提交到github上面去！

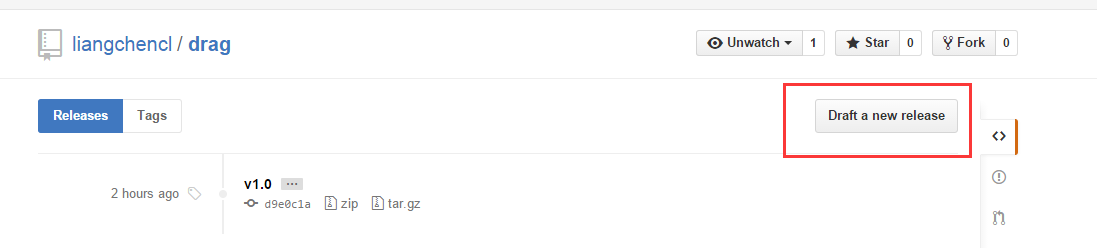
然后在github上面刷新就会出现了

**也可以直接在github上面直接添加：**

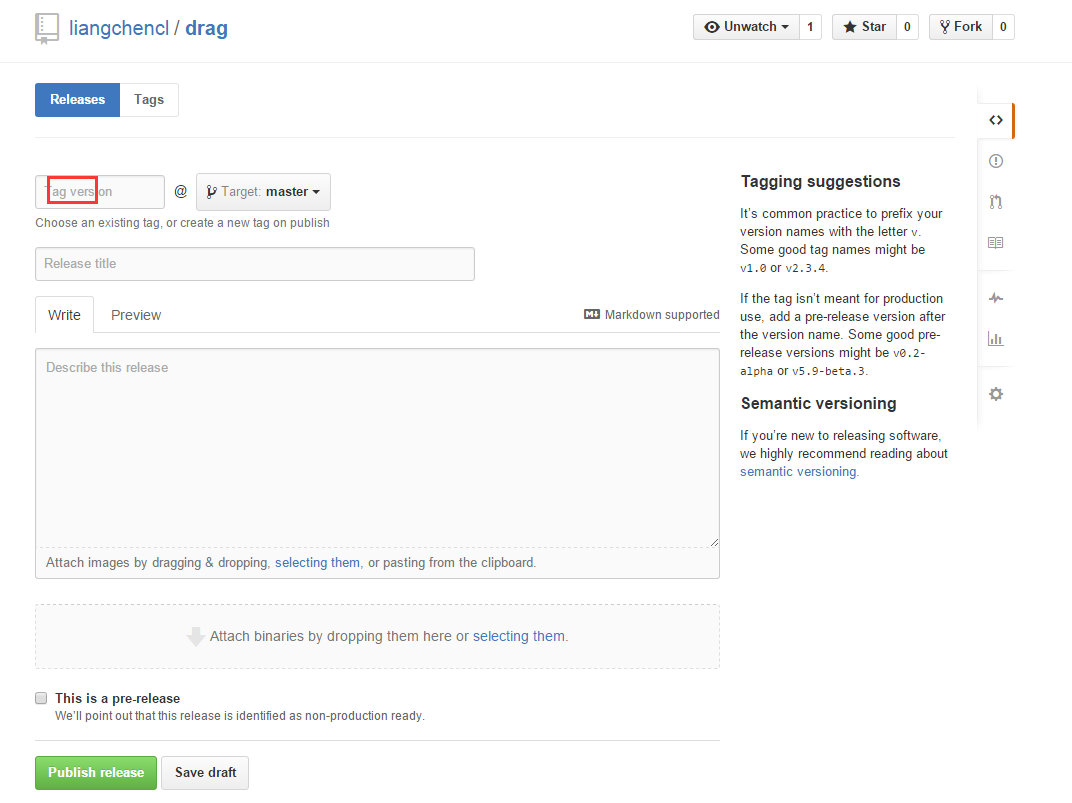
1：点击

****

2：

****

3：

****

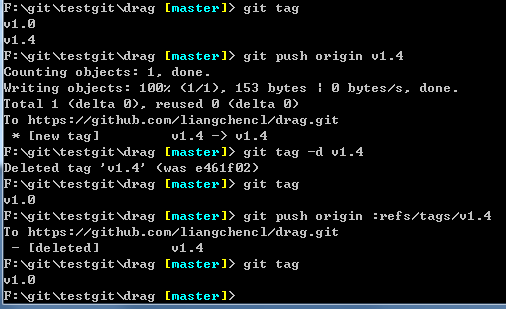
删除github上面的版本：

第一步先删除本地的版本

运行命令git tag -d版本名称(如v1.4)

然后

运行命令git push origin :refs/tags/v1.4 提交到github上面删除github上面的版本

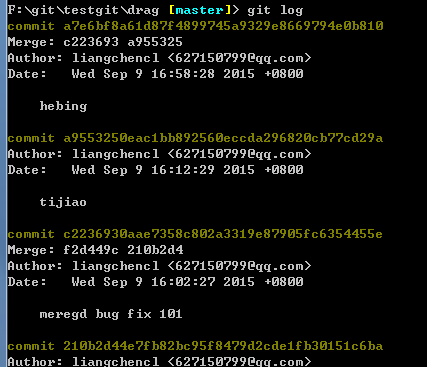


默认的版本库是打在最新提交的commit上的。有时候，如果忘了写版本库，比如，现在已经是周五了，但应该在周一打的版本库没有打，怎么办？

方法是找到历史提交的commit id，然后打上就可以了：

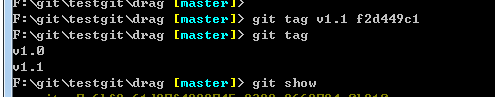
首先运行

git log



找到要打版本号的id即可：

git log 出来的cimmit是一串很长的数字，只需要写一部分即可：

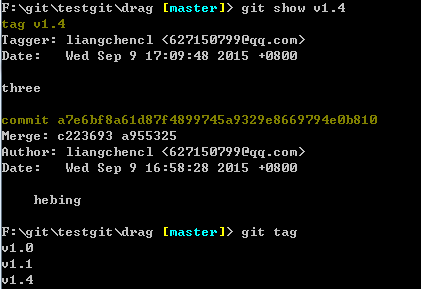


添加版本并写描述

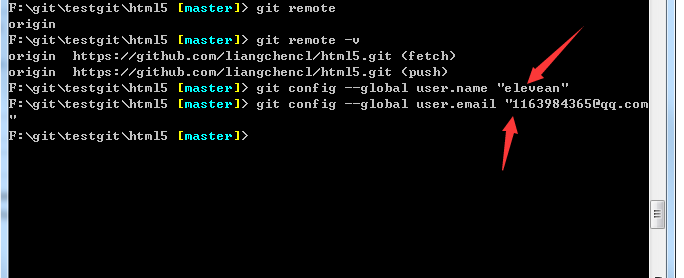
git tag -a v1.1 -m "描述" f2d449c1

查看某一个版本的信息：

git show v1.4



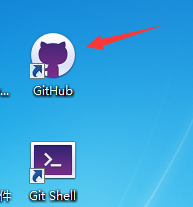
切换用户：

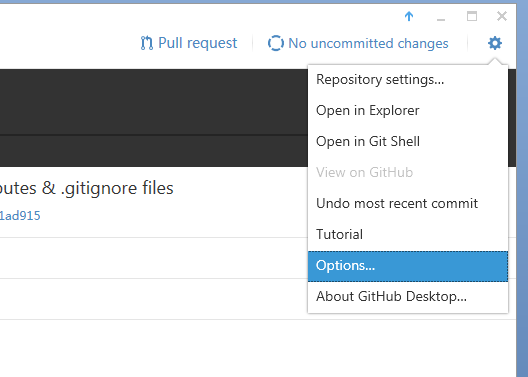


首先在命令行更改用户名和邮箱！

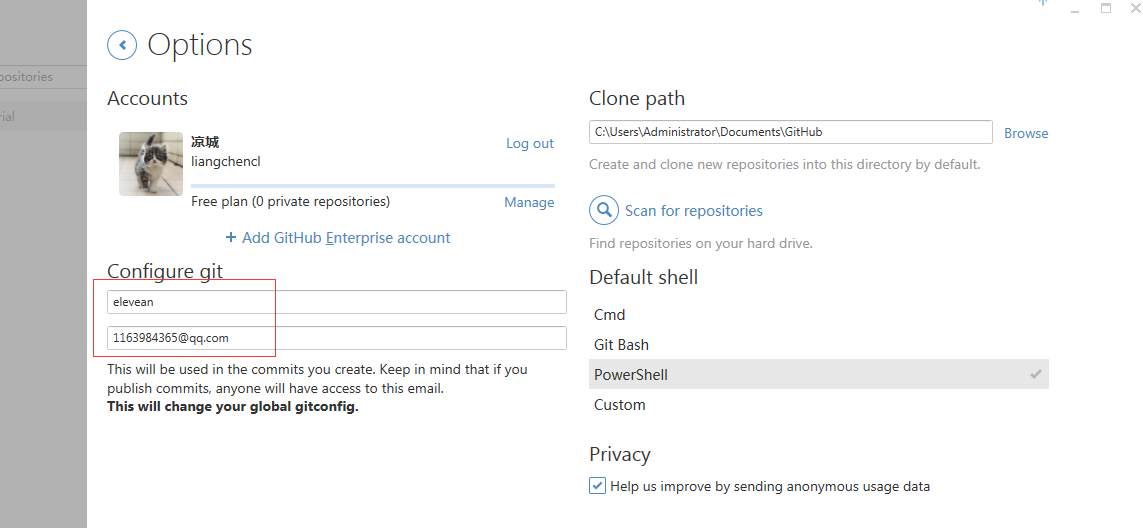
然后打开 github查看：

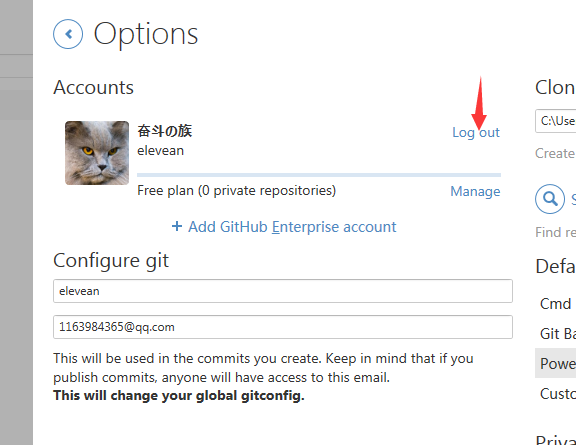
打开options

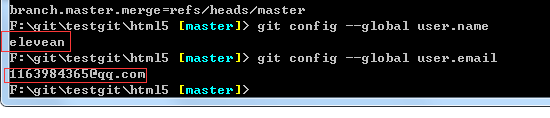




可以看到configure git 那里已经更改过来了，但是帐号并没有改过来，这里需要点击退出，然后重新登录elevean 这个帐号既可！







可以看到 已经切换过来了



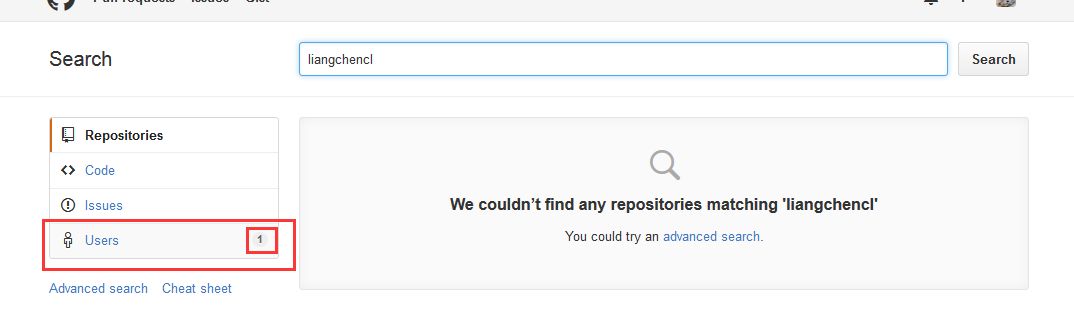
想参加别人的项目但是你没有权限的时候，你可以先fork他的项目到自己的github上，然后进行更改，然后pull request 发送请求给作者！

步骤：

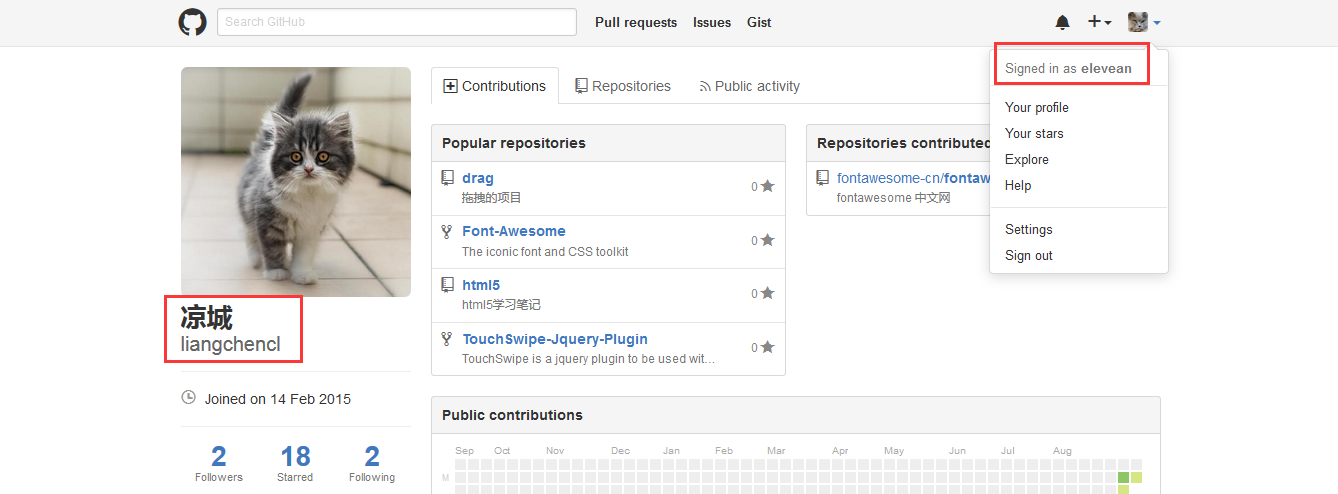
先在github上面随便找一个项目：



这里我采用的是我的另一个帐号 elevean登录的，然后我在上面搜liangchencl：结果如下



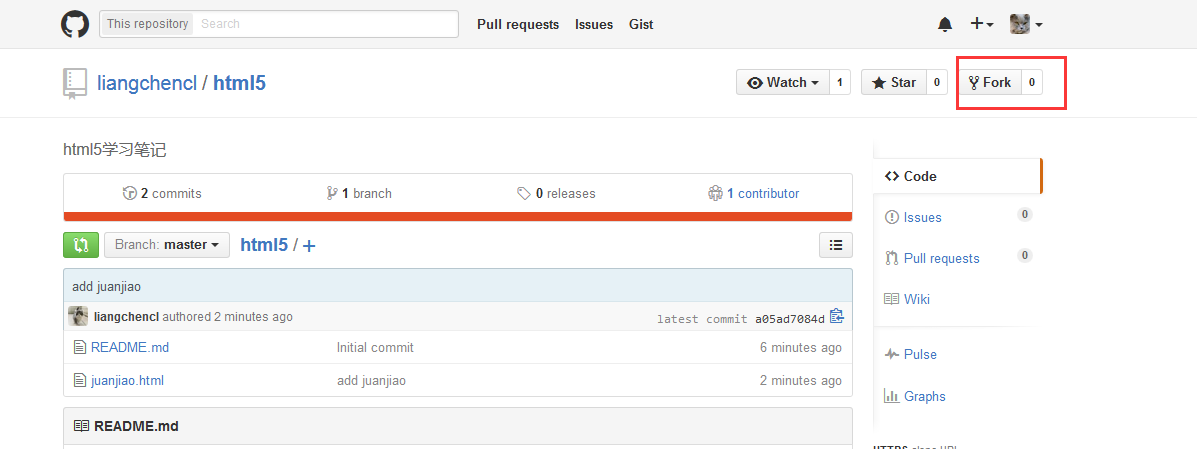
显示有一个用户是叫liangchen的 点击打开：



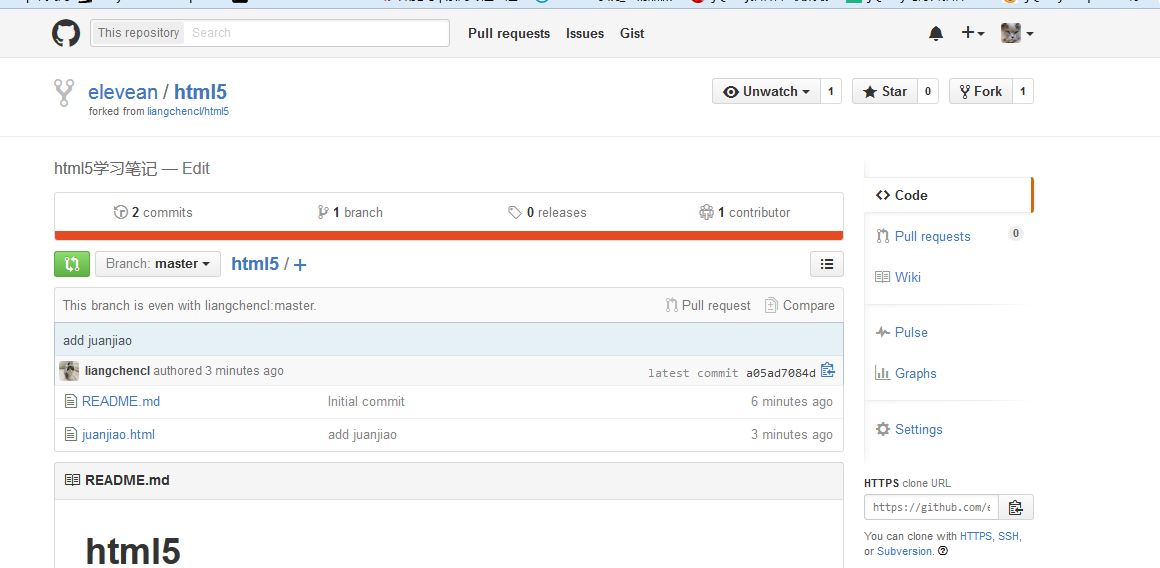
我们看到他有几个项目：

先点击一个html5 吧！

然后点击右上角的 fork！

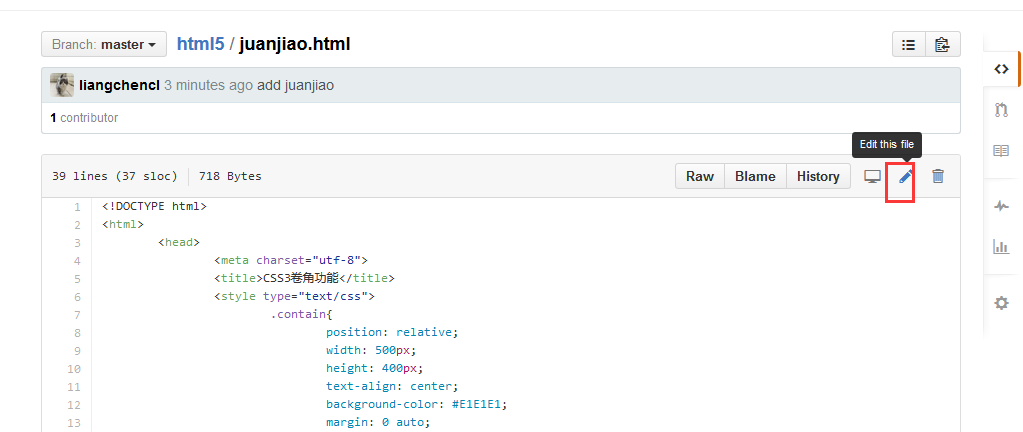


这样就克隆到我们自己的项目下面了！

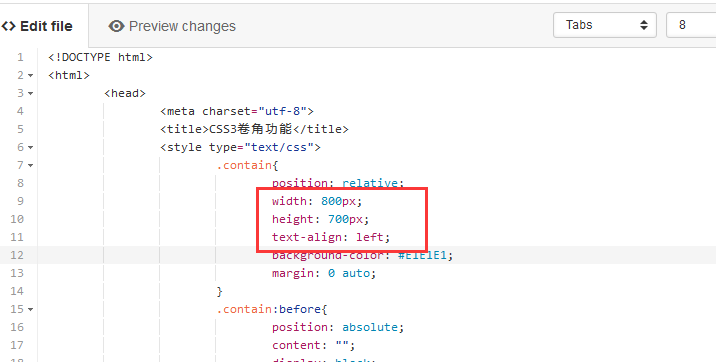


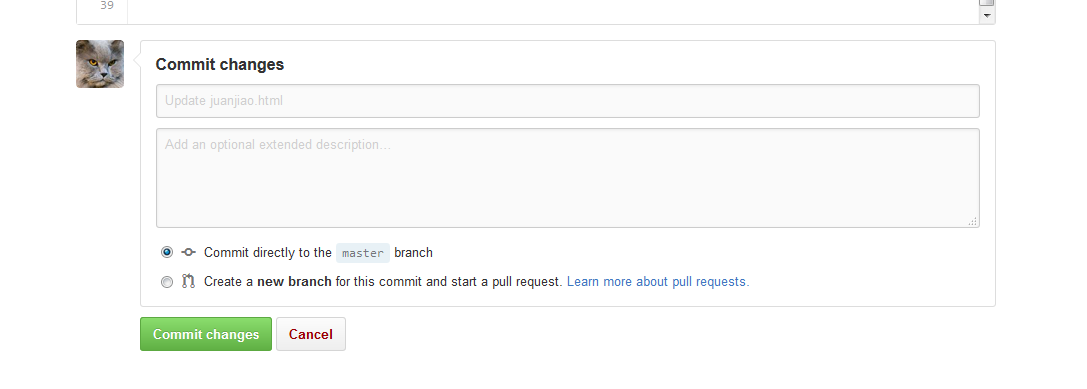
然后可以对它进行修改：

打开juanjiao.html，然后进行编辑：



我改动一些地方：

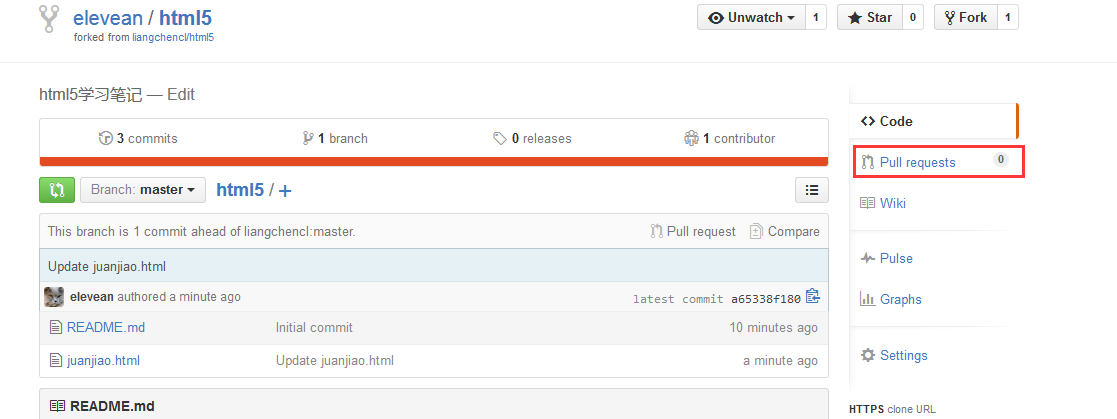


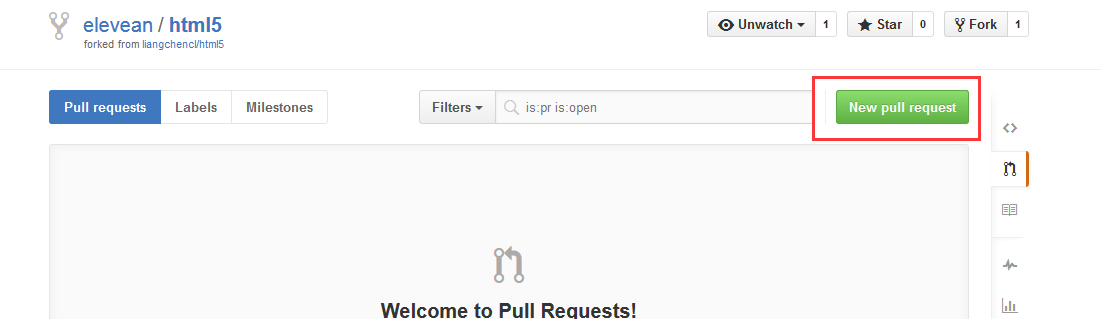


这里可以不用写，直接用默认的就可以了。然后点击提交！

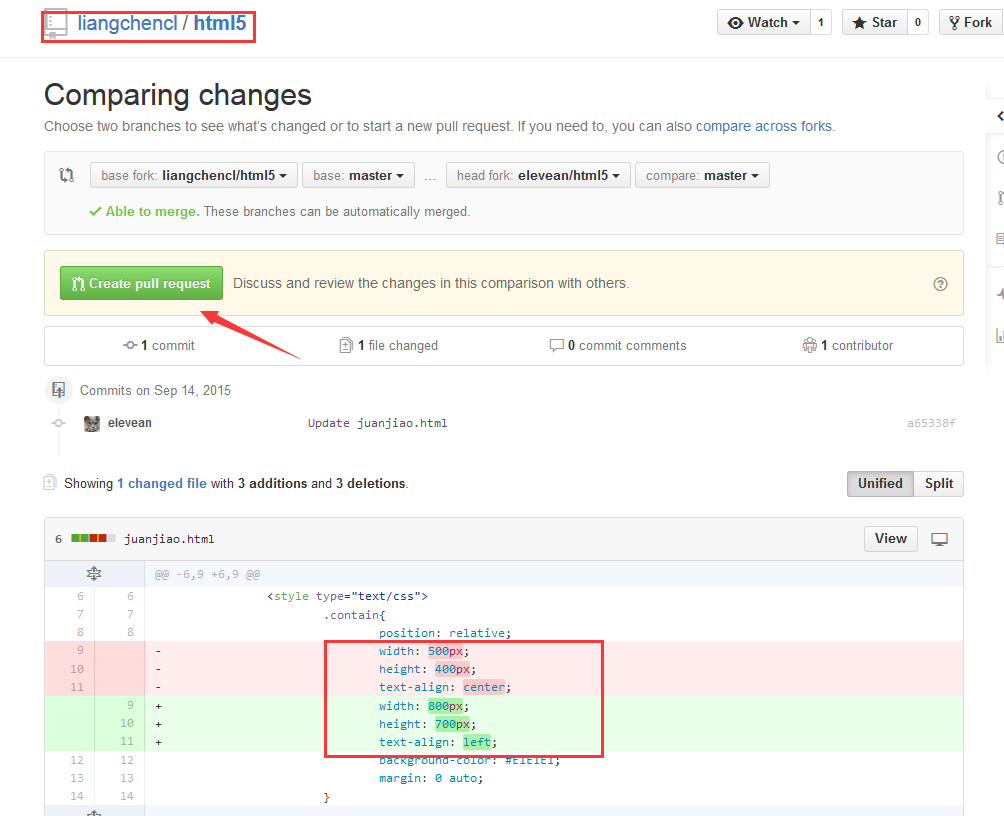
这样！在我们自己的仓库里面就已经更改了html5 这个项目，但是这个项目的作者那边还没有！这时我们要把更改的提交给他：

点击右边的 pull request





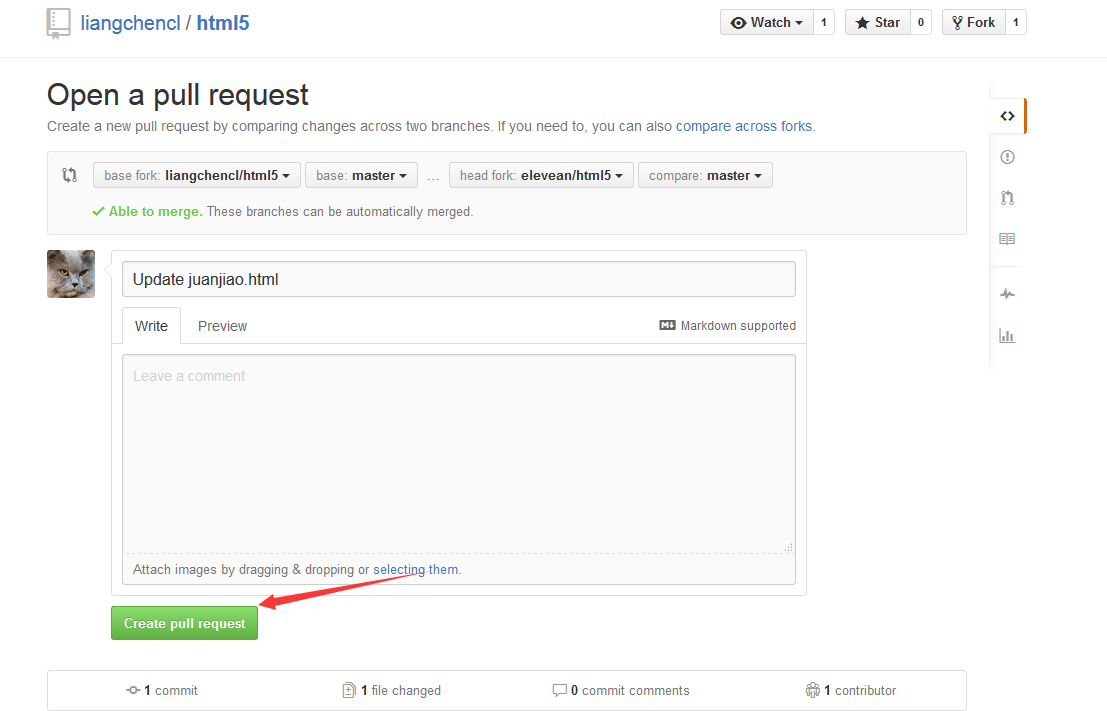
点击绿色按钮之后就直接跳到了liangchencl的项目里面来了,并且标记出了你修改的地方：



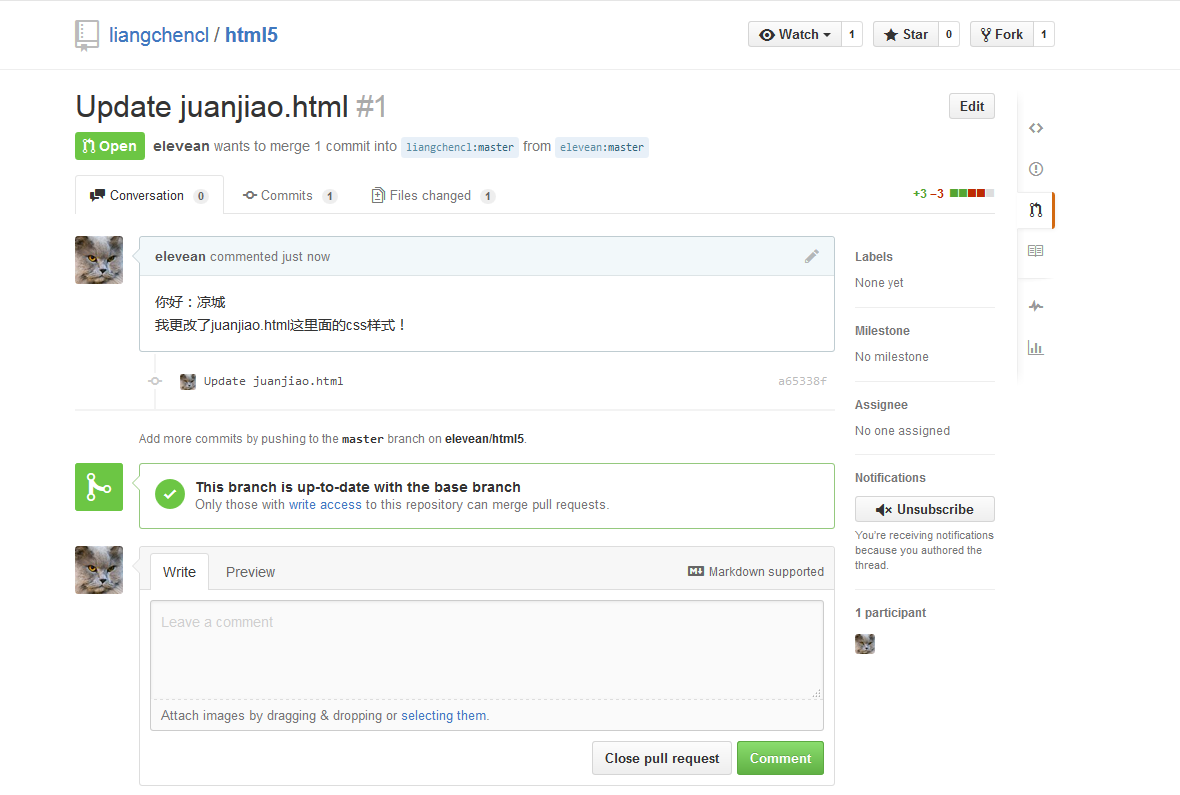
可以看到更改的地方已经标记出来了！

然后再点击绿色按钮：

这里是发送一份右键给原作者，可以自己写几句话也可以什么都不用写！不过建议写上具体的更改文件会比较好点！这样会好找些。



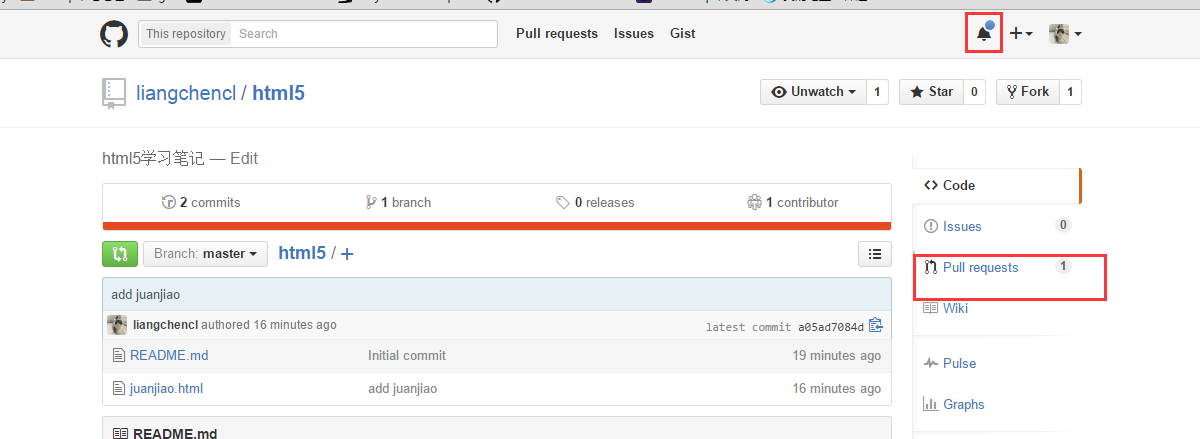
然后提交：

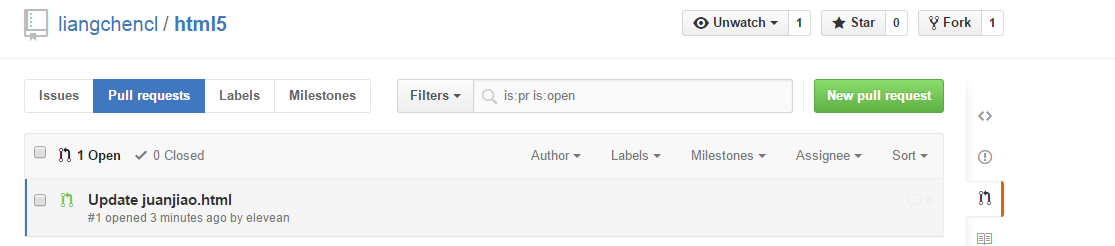


这样，原作者就可以在他的项目里面看到你做的修改！他觉得ok才会把你改的代码合并到他 的项目里去！

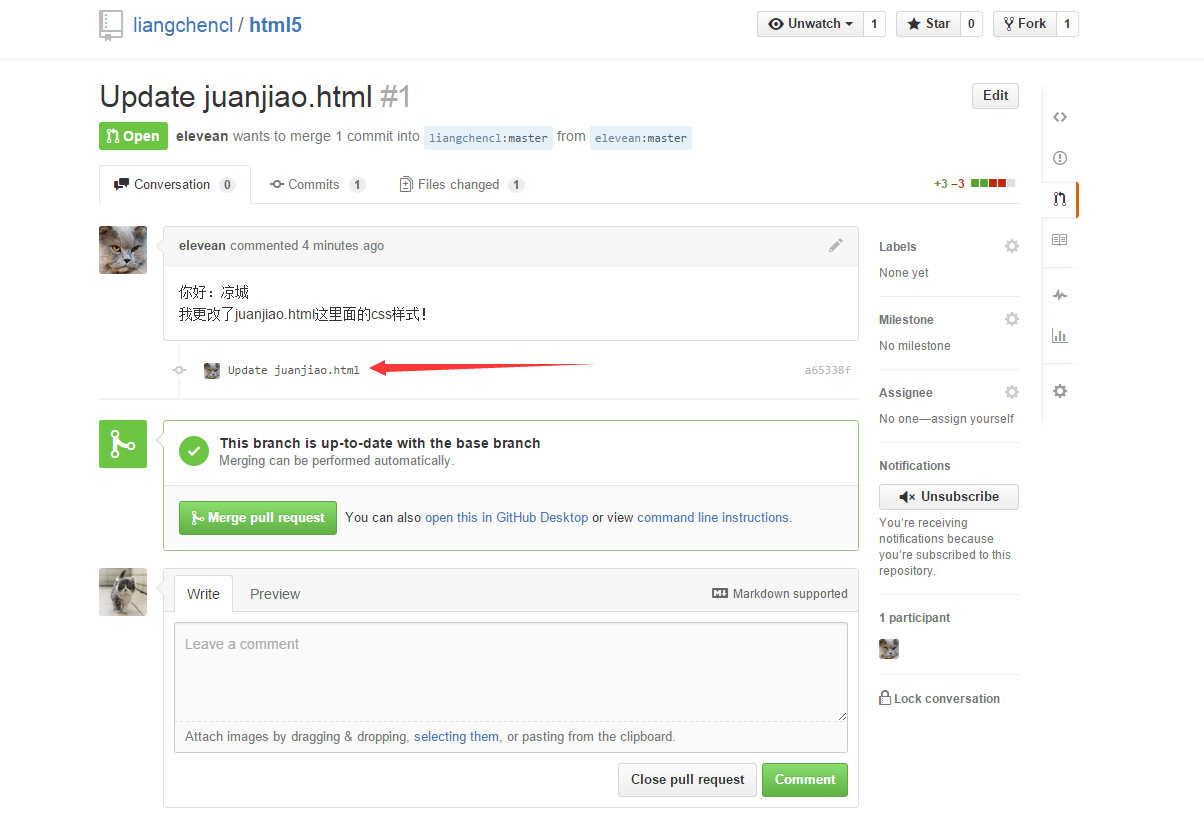
这是liangchencl 帐号登录的：

可以看到有消息：我们点开看下，



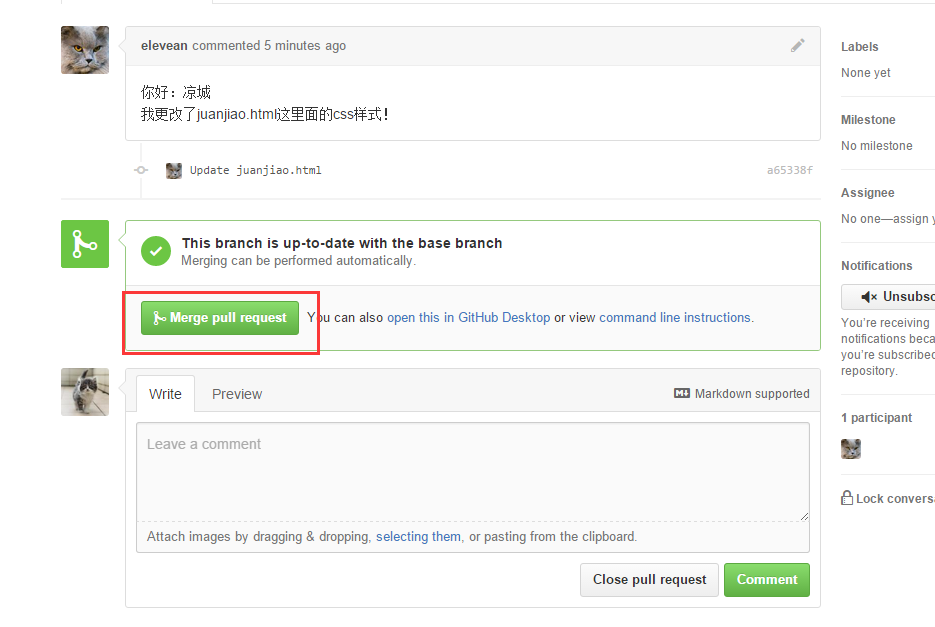


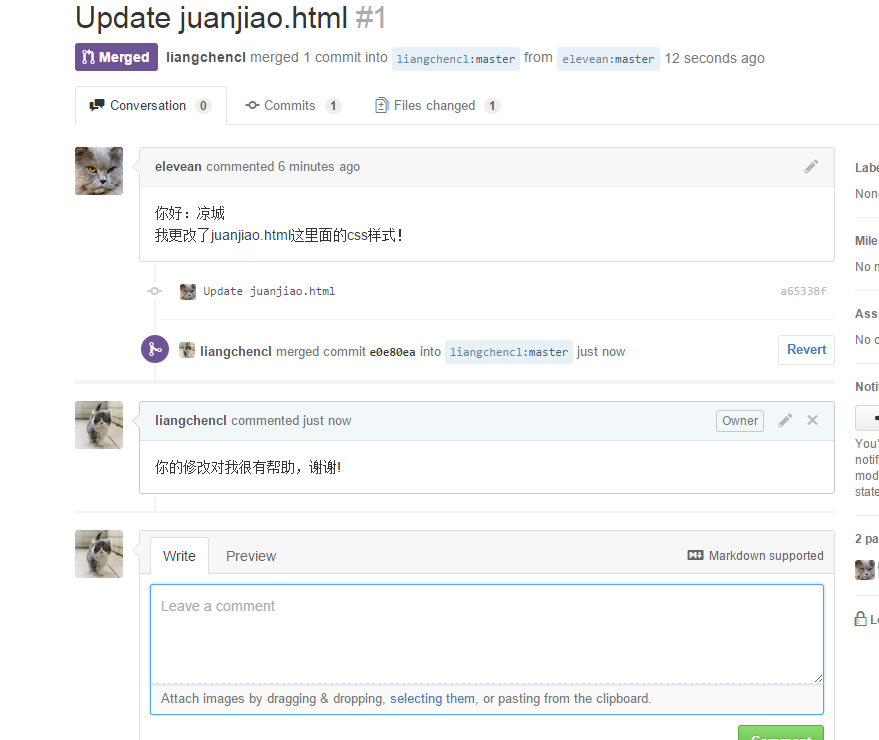
可以打开看具体的更改代码：



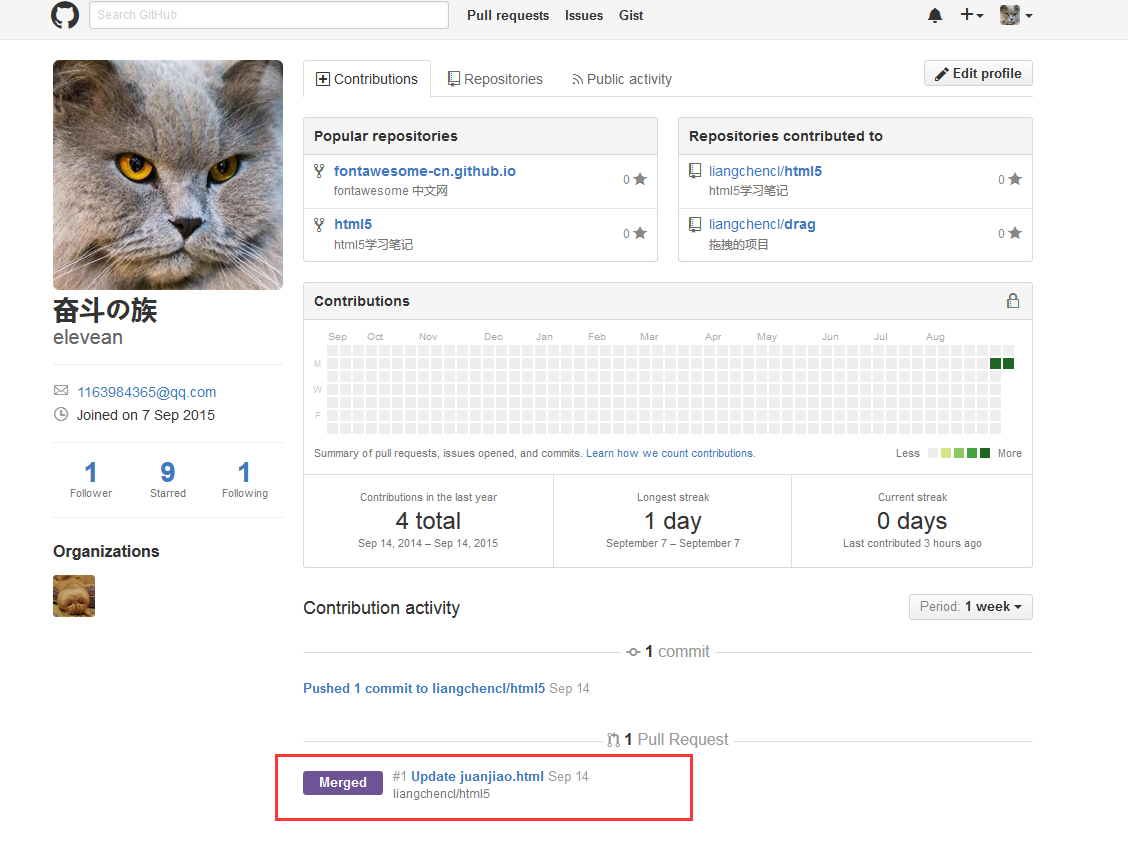
如果作者觉得你修改的是可以的，可以合并，他就直接点击 merge pull request 按钮即可：

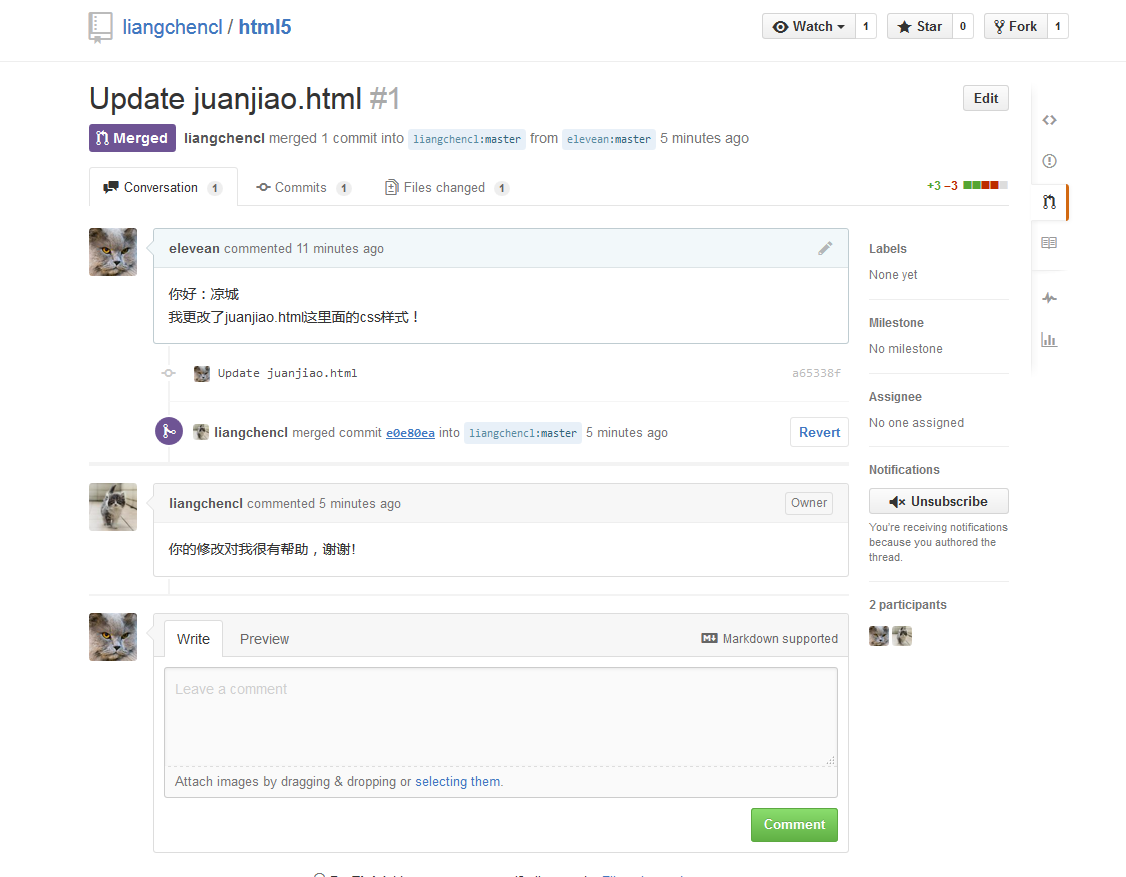
并且可以给你回消息：



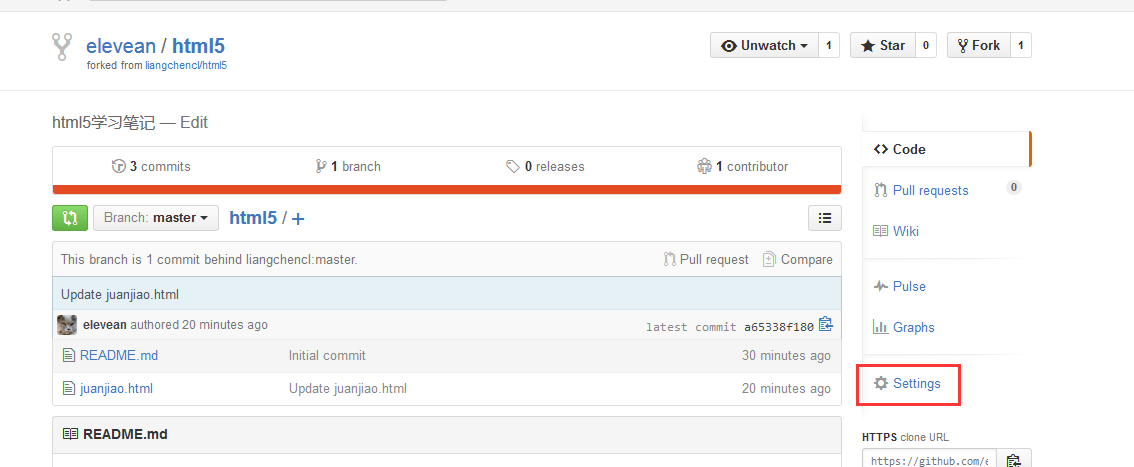


然后我们在自己的github上面会看到作者的回复：

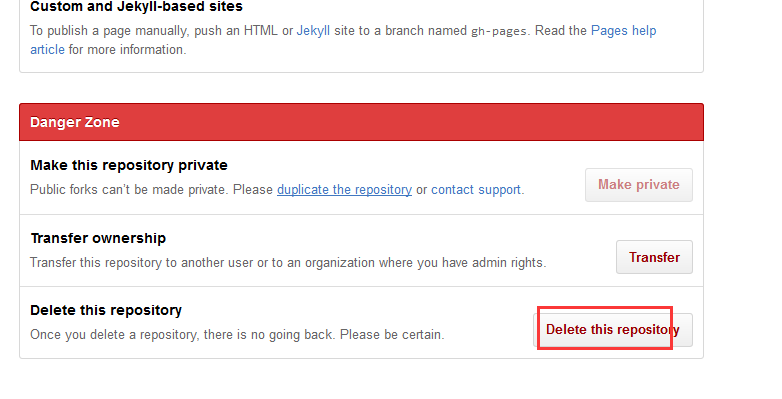




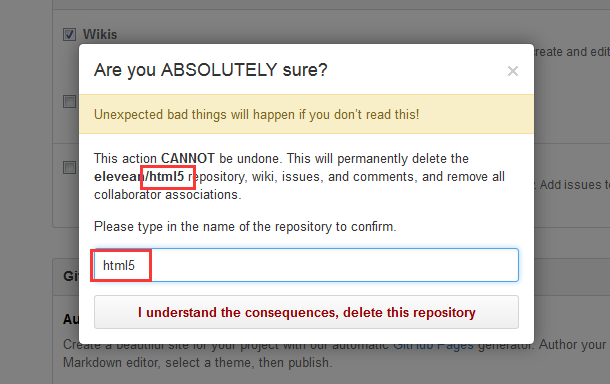
最后一步删除！



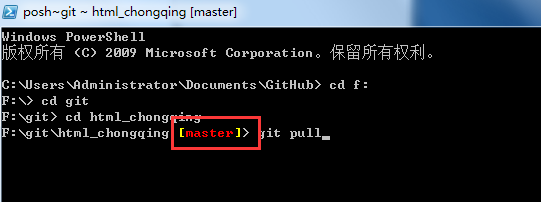
最下面有个红色的按钮：点击



会弹出你要删除的仓库：注意要写正确！删除即可



在多人协作的时候，如果打开



这里是红色的 就表示有其他人提交了，但是你这里还没有拉取，记得先拉取一下！然后再写你自己的代码，然后提交。

拉取命令 就是 git pull

