基于 GitHub 的合作开发流程

（这里都默认大家已经有 GitHub 账号）

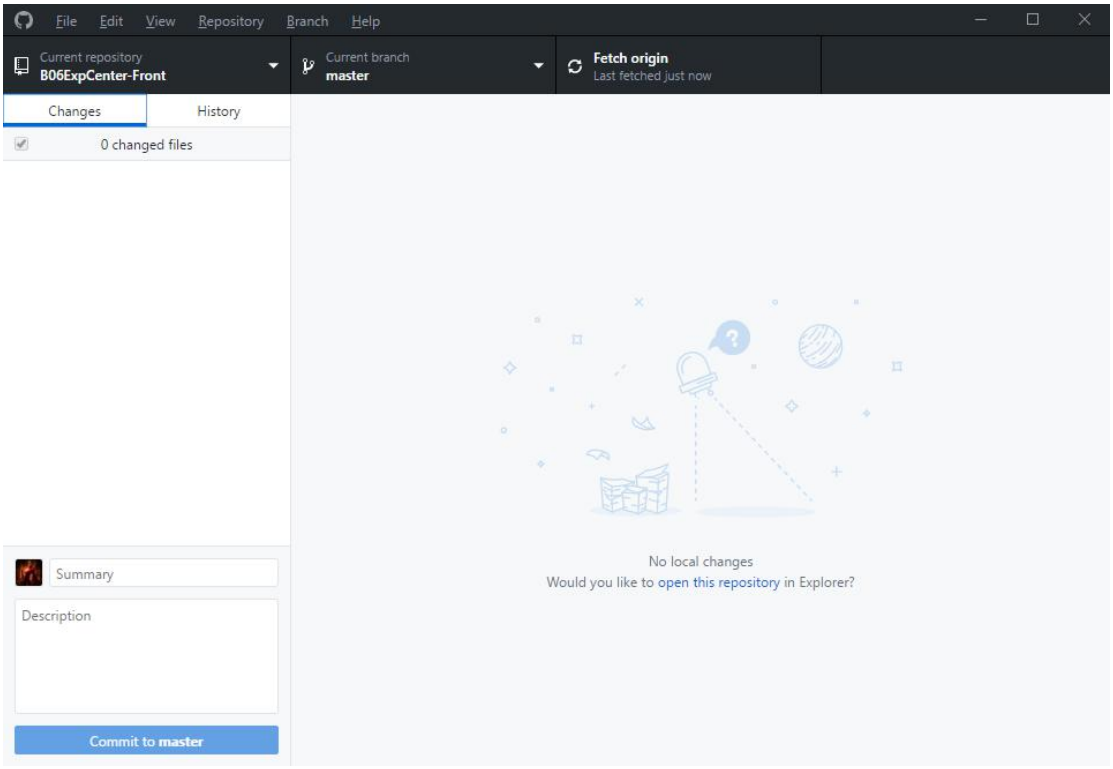
环境部署

[Git]

1. GitHub 是基于 Git 的，下载 Git 工具（这不是 GitHub，是 Git，最基本的依赖工具）：https://git-scm.com/downloads
2. 下载页面选 64-bit Git for Windows Setup；
3. 安装全部下一步；
4. 生成 Git SSH Key（这一步必须，否则不能提交或者 Clone 代码），参考此教程：http://blog.csdn.net/lsyz0021/article/details/52064829

[GitHub]

1. 在群文件里下载 GitHubDesktopSetup.exe；
2. 打开即可用，并且登录你的 GitHub 账号，过程中可能需要输入用户名和邮箱，对应你的 GitHub 用户名和邮箱即可；
3. 进入如下主界面：

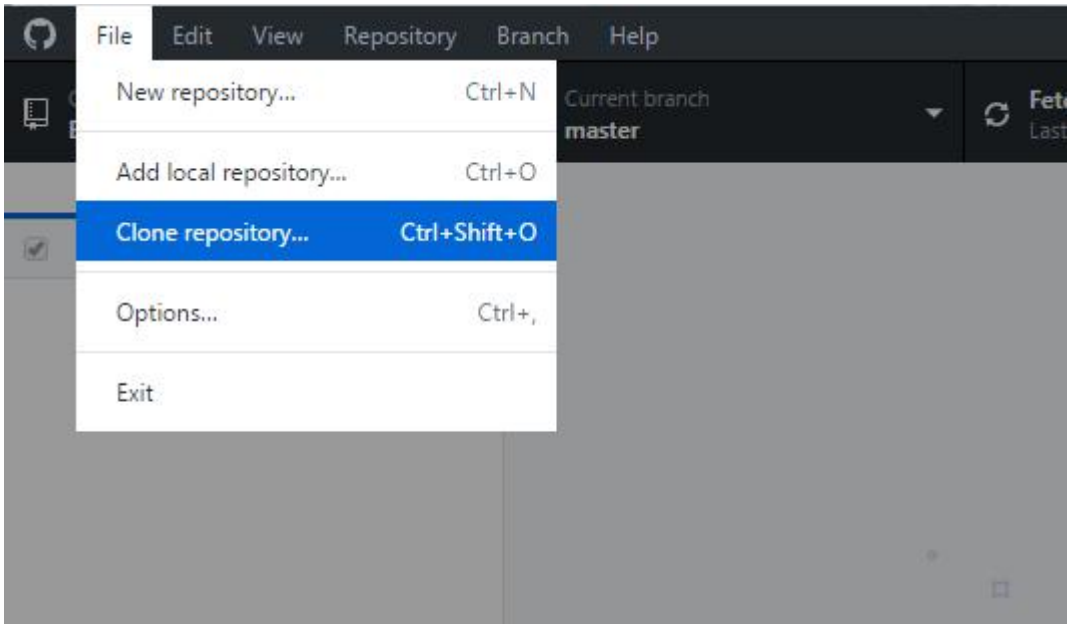


下载仓库

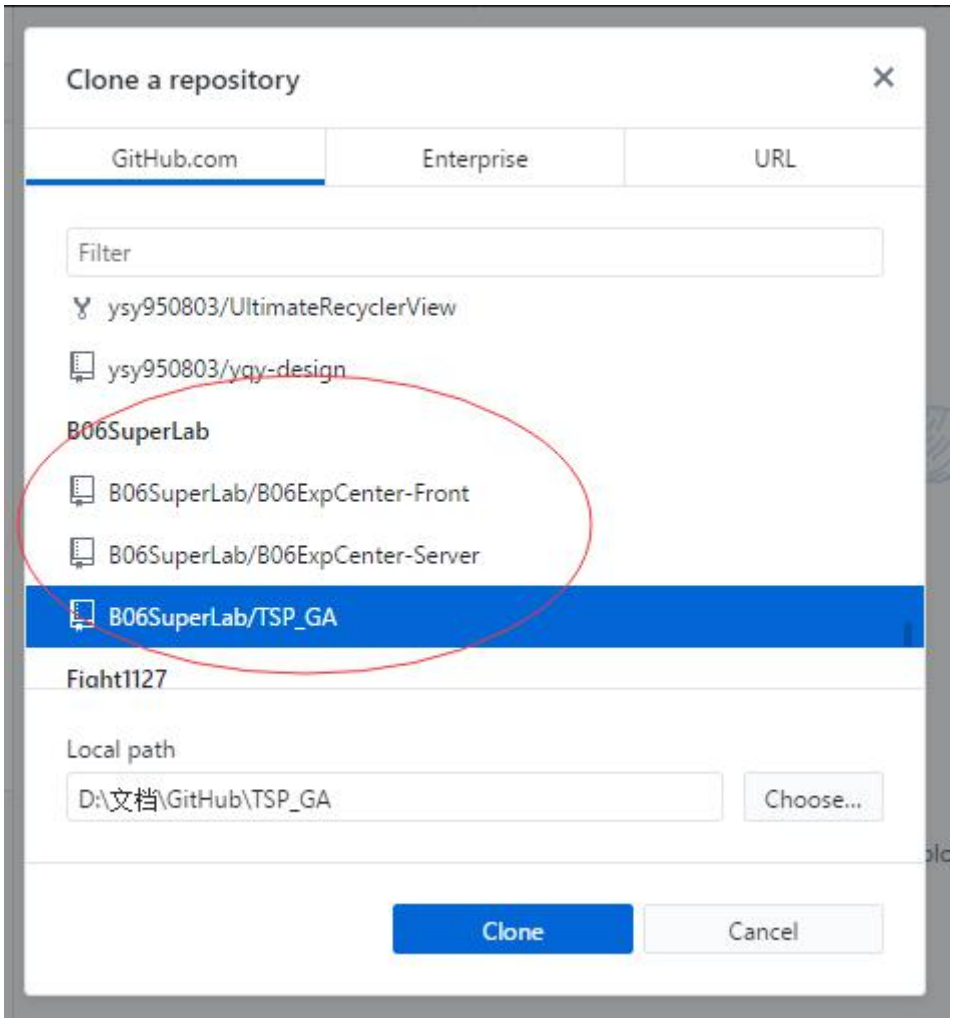
这里我已经给大家把代码仓库创建好了，前端的名称叫 B06ExpCenter-Front，后端叫 B06ExpCenter-Server，大家可在 B06 的 GitHub 网站上查看：https://github.com/B06SuperLab

然后我们利用刚才安装的 GitHub 客户端来 Clone 仓库：

1. 点左上角 File，选择 Clone：



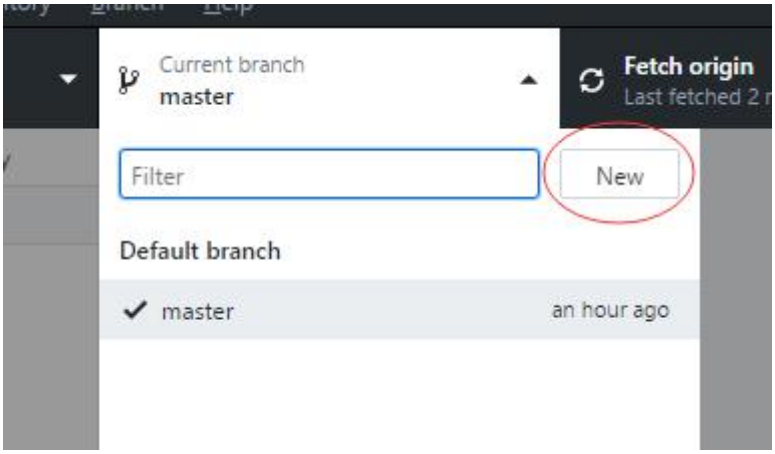
2. 在弹出的窗口中找到你对应的项目，然后 Clone 到本地：



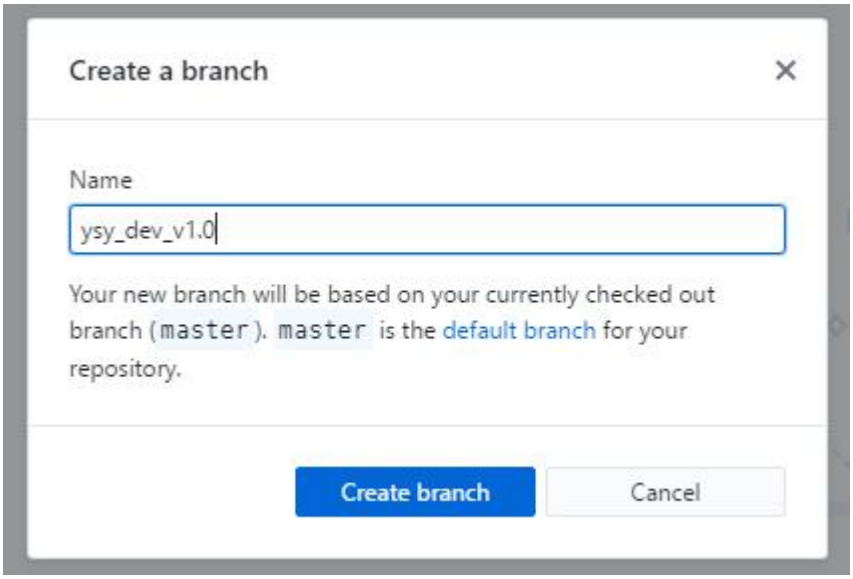
创建分支

项目默认是有一个 master 主分支的，但是我们不能在这个上面开发，多人合作就需要各自建立自己的分支，这样就不会干扰主分支，在各自的分支上开发完成后，合并到主分支。

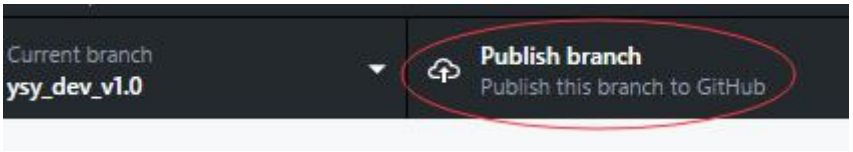
1. 点 Current branch 会展开一个菜单，然后点 New 就是创建新分支：



2. 这里强调一下分支命名规范，这样方便以后管理，如下（ysy 表示你的名字或者昵称，这个随意，dev 表示这是开发分支，可能还有 release、test 什么的，v1.0 表示版本号）：



3. 创建好自己的分支后，由于你只是在本地操作，还需要把分支推送到 GitHub 服务器：



提交代码

提交一次代码到 GitHub 远程仓库大致流程是：

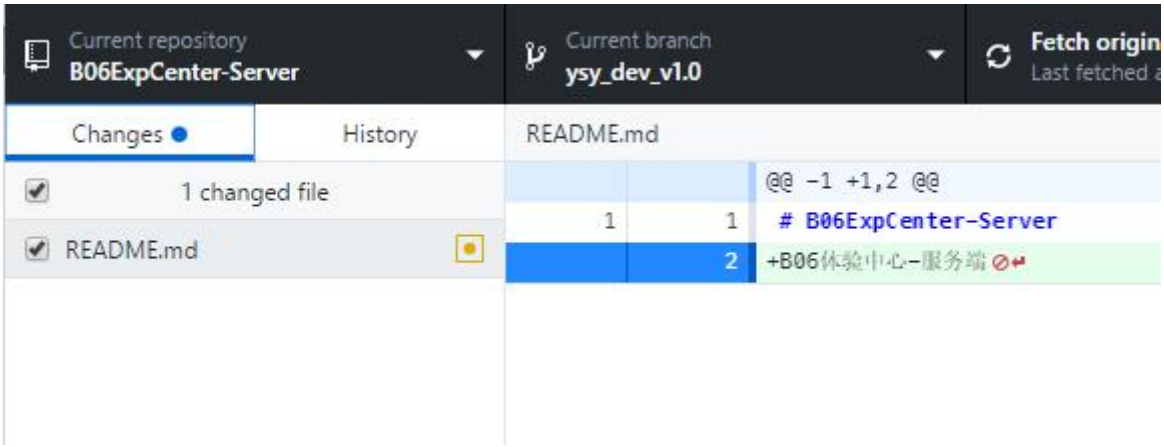
1. 本地修改代码（产生了 change）；

2. 提交（commit）代码到自己的分支，注意这个时候代码都只是在本地，并没有同步到服务器；
3. 推送（push）到远程仓库，完成了一次代码提交。

具体步骤示例：

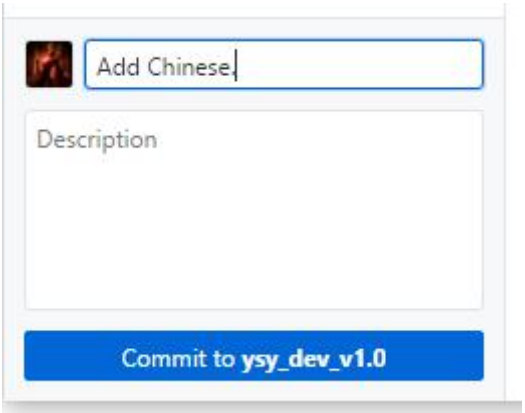
1. 我们以 README.md 这个文件为例，先在本地修改这个文件：

2. 然后你就可以在 GitHub 客户端看到改动：

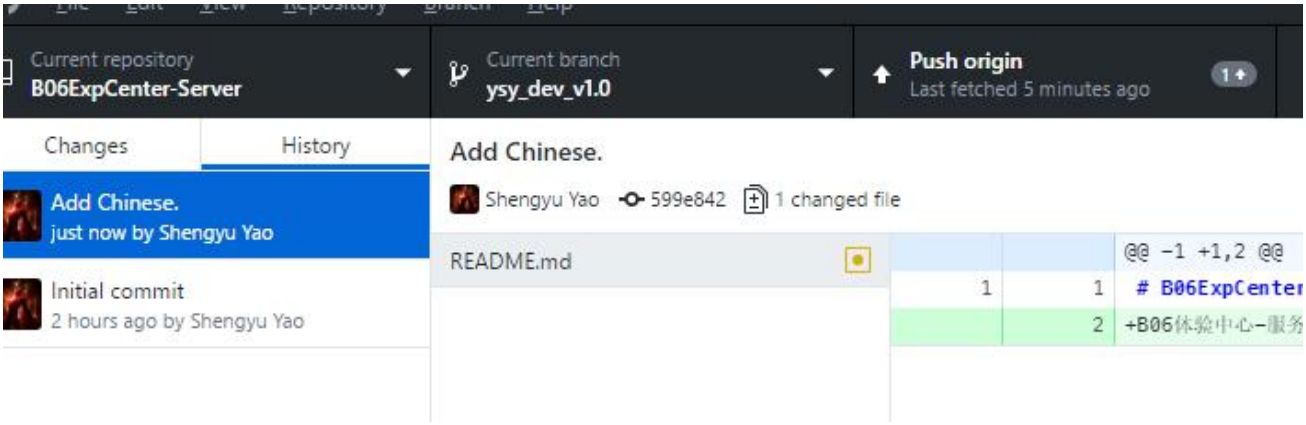


3. 填写提交信息，这里的 Summary 是必填项，说明你这个 change 主要是做了什么（比如改了什么 BUG，增删改了什么功能），Description 是非必填的，可以详细描述一下自己的代码为什么这么写。

填好了点击 Commit：



4. 提交成功后还可以 Undo 撤销，然后我们可以看到 History 当中已经有你刚才的 commit 了，这个时候直接点击 Push origin，即可把修改后的代码推送到远程仓库：

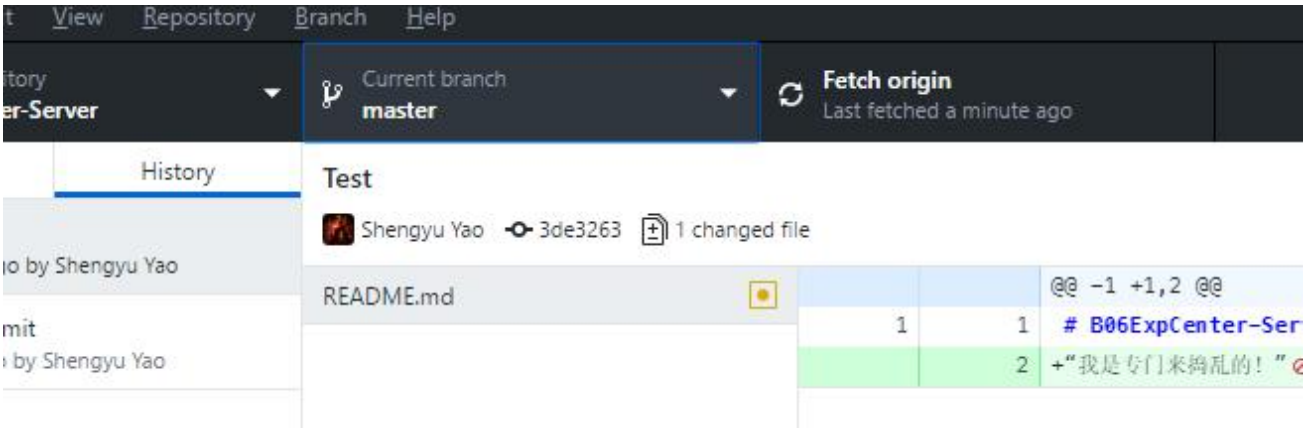


合并代码

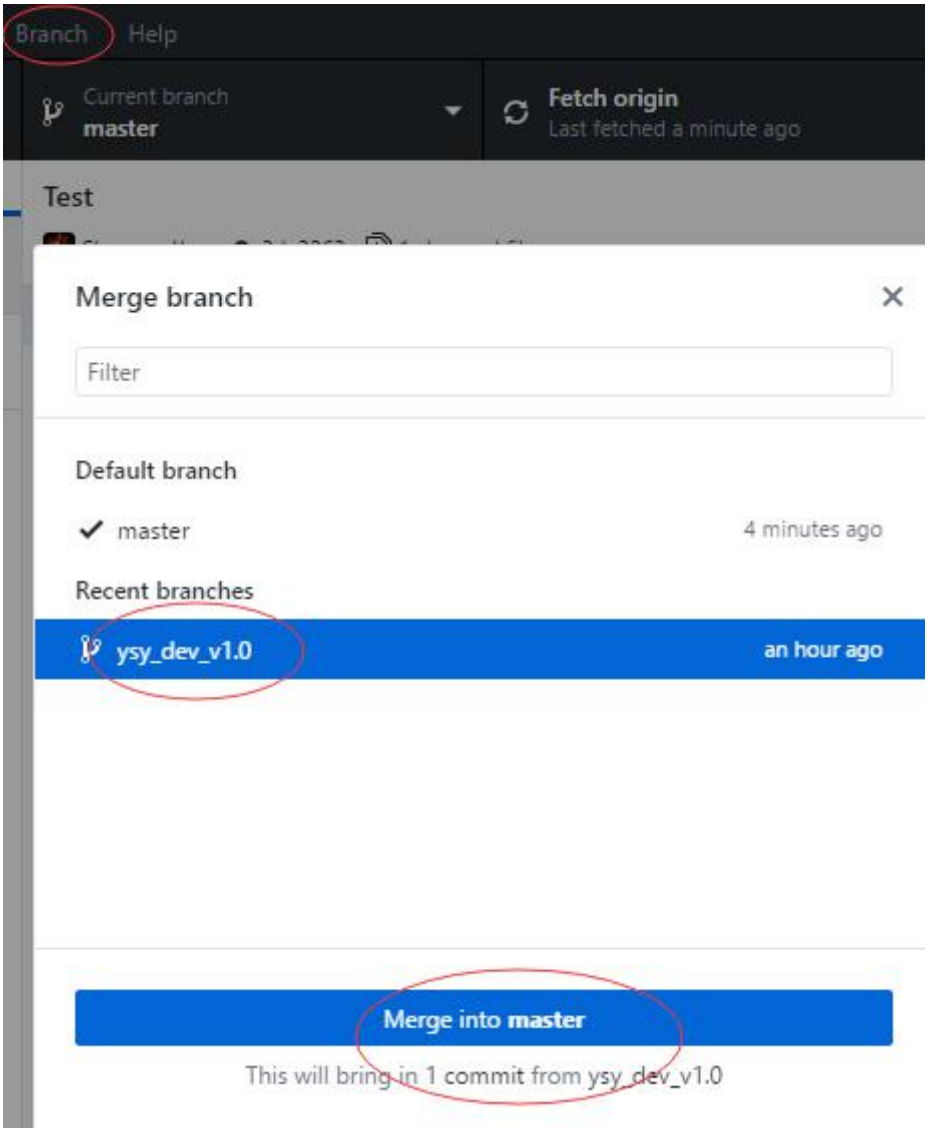
刚才我们只是把自己修改的代码提交到了自己的分支上，但最终我们肯定是要合并到主分支的。为了提高大家的开发效率，我这里就不设什么代码提交权限了，大家都是扁平化的。所以也就暂时不用到 Pull Request（即把自己分支合入主分支时提交一个请求，等主分支的管理者来审查要不要合入你的代码）。

所以这里讲一个**具体步骤示例**：

1. 切换当前分支为 master，刚才是 ysy_dev_v1.0，然后我们发现主分支的 README.md 文件也被修改了：

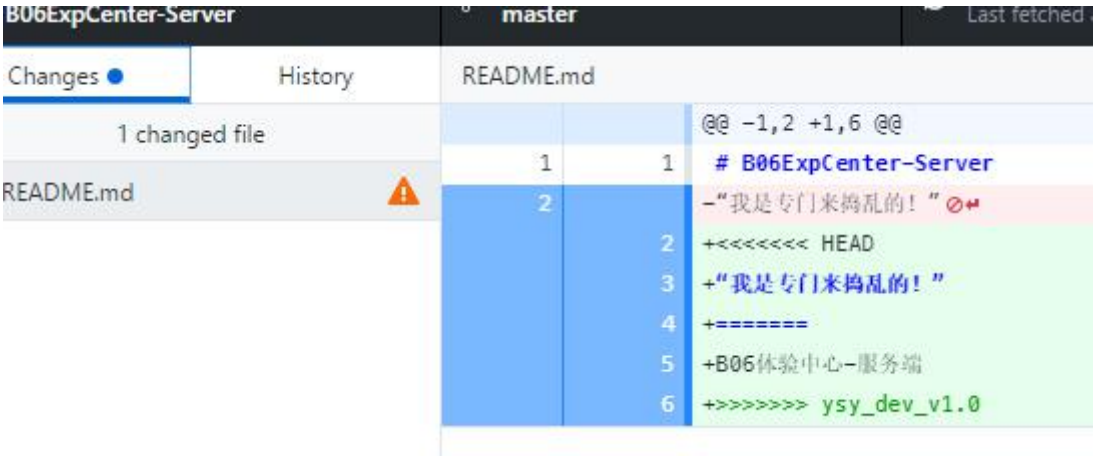


2. 点上面的 Branch，Merge into current branch，并选择要合并的分支：



3. 点了 Merge into master 后，果然出现了 Error，这里显然就是因为两个分支修改了同一个文件的代码，所以出现了冲突（conflict），到底听谁的呢？这就要你们自己去商量了。当然，实际开发中，我们尽量分工，尽量保证不产生冲突，但有了冲突也不要怕，解决冲突就行了：

4. 查看 master 分支的 Changes 区域，会发现如下：

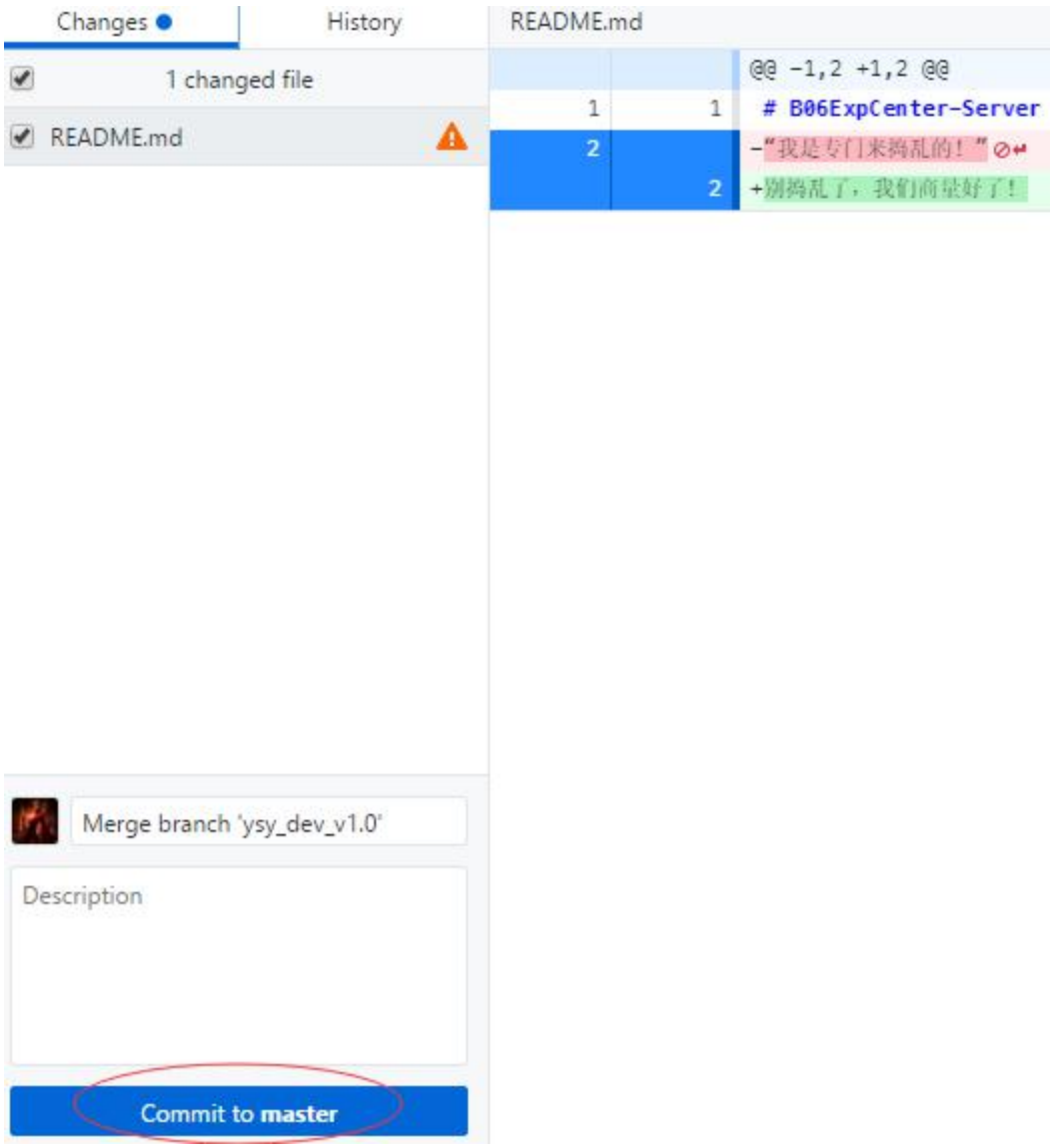


5. 然后在本地编辑器当中去解决冲突，这里的<<<< HEAD 标记表示当前分支的冲突代码，=====是分割线，分割线到>>>> xxx_dev_v1.0 标记是被合并分支的冲突代码，这些标记的目的就是为了让你代码编译出错，强制你去修改冲突的。

这里我就改成一个我们都商量好的内容即可：

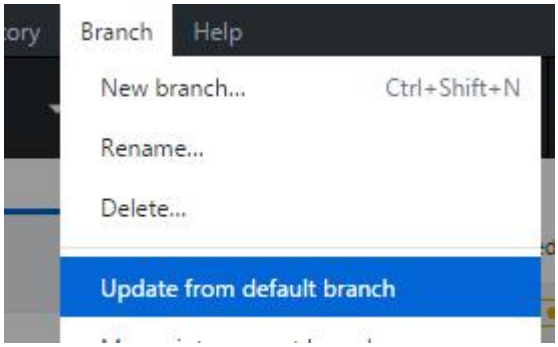
6. 解决完冲突后，我们再次提交到主分支即可（GitHub 客户端已经非常贴心地给你自动填写了 Summary，你只用补充一点 Description 或者什么都不写），

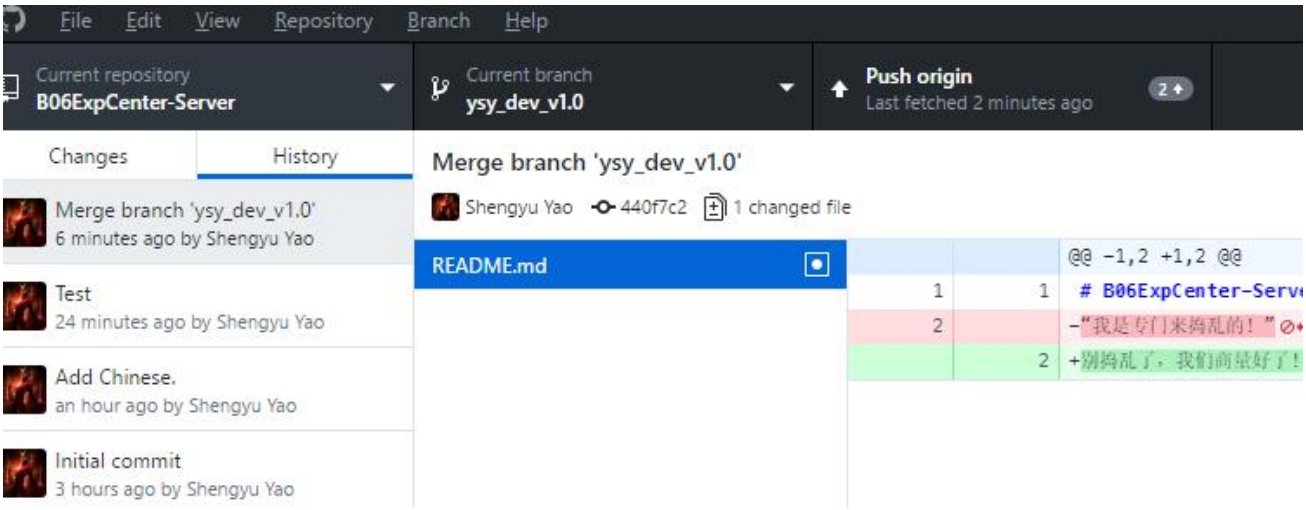
提交完之后别忘了 Push 到远程仓库：



7. 解决完冲突后就完事了吗？没有！刚才的操作你只是把自己分支里的代码成功合并到了主分支并且解决了可能存在的冲突，别忘了还要更新你自己的分支，保证每一次开发新代码之前，自己的分支都和主分支保持一致，换句话说，你的开发版是基于稳定版的最新版来开发的。

所以先切换到自己的 dev 分支，点 Branch，Update from default branch，即可同步主分支的代码到自己的分支：





可见，我们已经把主分支的所有 commit 都更新到了自己的分支里，保证了全局的一致性，所以这个时候你自己的分支里也有其他同学提交的代码了，当然最后也别忘了 Push 到远程服务器。

最后，有什么问题随时在群里问，或者私聊我。

姚圣禹
admin@ysy950803.cn
QQ：632599008
微信/手机：17888836853