

Git 简单使用教程

0. 前言（仅看安装教程不用看）

为什么要使用 Git，Git 又是个什么东东？

对于我们经常做一些小作品、项目的电子人来说，很多时候一整个项目并不是能直接一口气完成的，我们通常会把整个工程分为很多个部分，不仅是硬件上可以模块化设计，代码也可以。而对代码来说，版本迭代是一件很频发的事。将整个工程进度当做一条线的话，不同的完成进度就是线上的一个个节点。

这时候有个问题：我们通常开发不是一个顺利的过程，会经常遇到问题，有可能今天上午还能运行的一个程序，下午做了一些改动就突然运行不了了。要是改动少的话我们还可以靠着记忆力回退，改动多的话就只能祈祷代码有做保存能回到上午的版本了，要是也没保存就只能重新来过或者找个时光机了。

那随时保存总好了吧？的确是一个办法，但是保存多了自己分不清版本搞不清时间，还占了很大的一部分空间又不敢删除。并且，通常这样的工程代码想合作是一件困难的事情，你很难指望你的小伙伴能分得清你的进度到哪，每次干活还得先复制一个完整工程...那还是自己干吧。

而 Git 是一个版本管理软件，安装后你能使用命令行写命令对你的工程进行存档，并且可以随时回退。有意思的是 Git 的保存方式是差异化保存，每次都保存增加删减的部分而已，这是不会占用而外的空间的。而 Gitee 是网页上的代码托管平台，可以在网页上创建一个仓库和电脑上的文件关联，网页上能随时看到本地提交的更新记录和变更内容，也能和小伙伴一起协作。

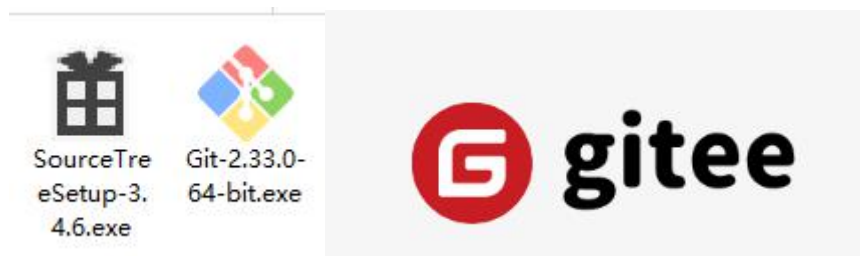
1. 涉及软件和网页

A Git 软件，提供底层支持

B SourceTree，给 Git 提供一个图形 GUI 界面方便使用

C Gitee 国内基于 Git 的代码托管平台（直接百度 gitee）

个人网盘下载：



<https://wvr.lanzoui.com/b02ie6s7g>

密码:7wio

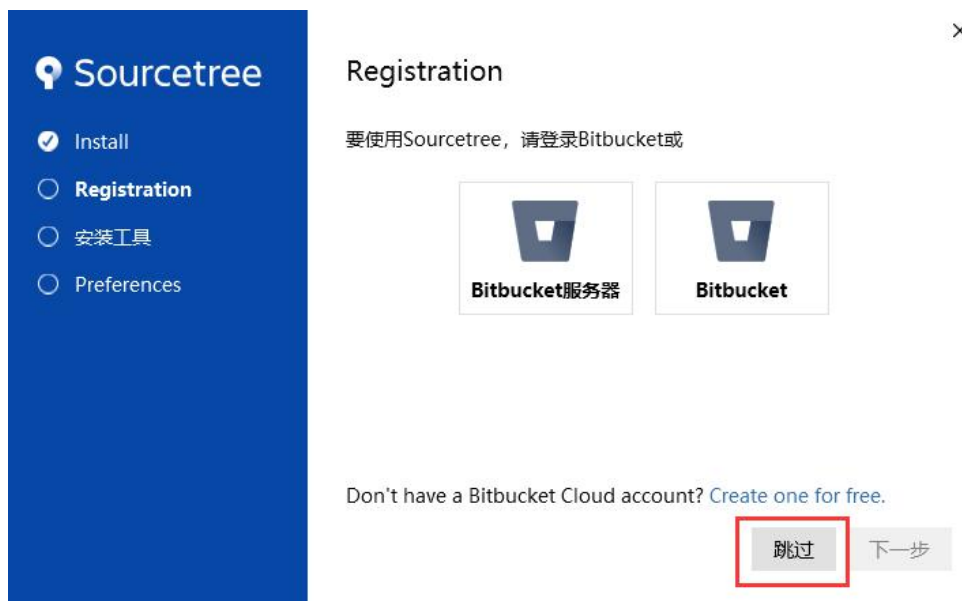
2. 安装 git 软件

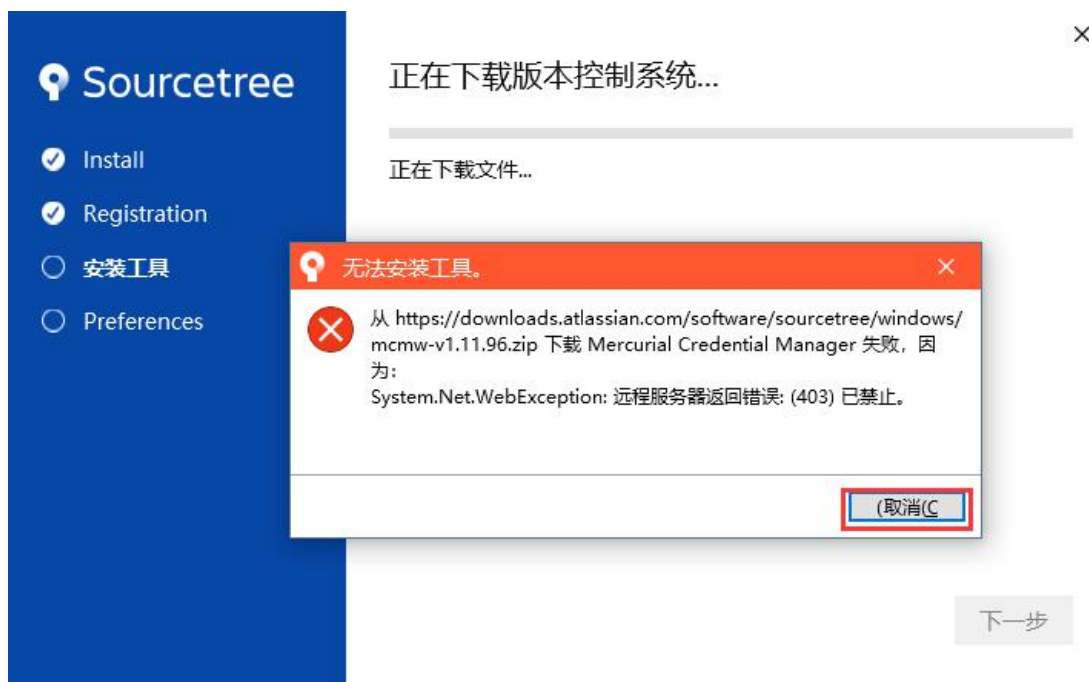
这里无脑下一步就好了，一路走到黑



3. 安装 SourceTree

几乎也是无脑下一步

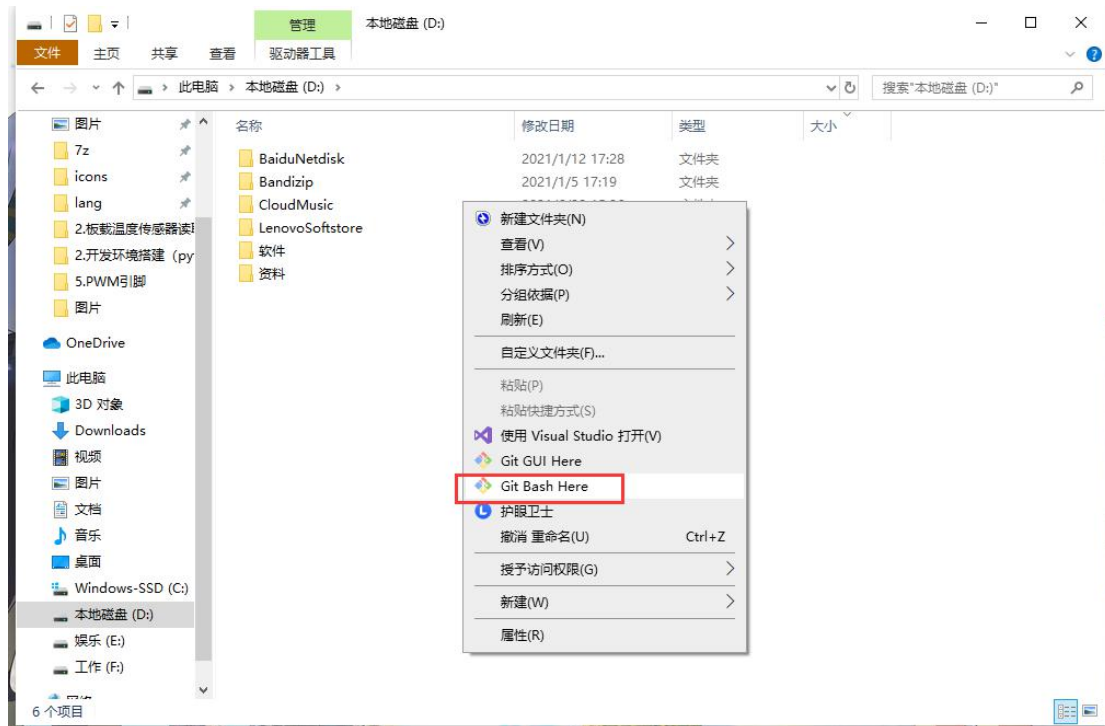




后面就是无脑下一步了，没有需要注意的地方，最后一步是否添加秘钥选否。

4. 生成 SSH 秘钥以及添加到 gitee 上

在任意文件夹右键打开 git 命令窗

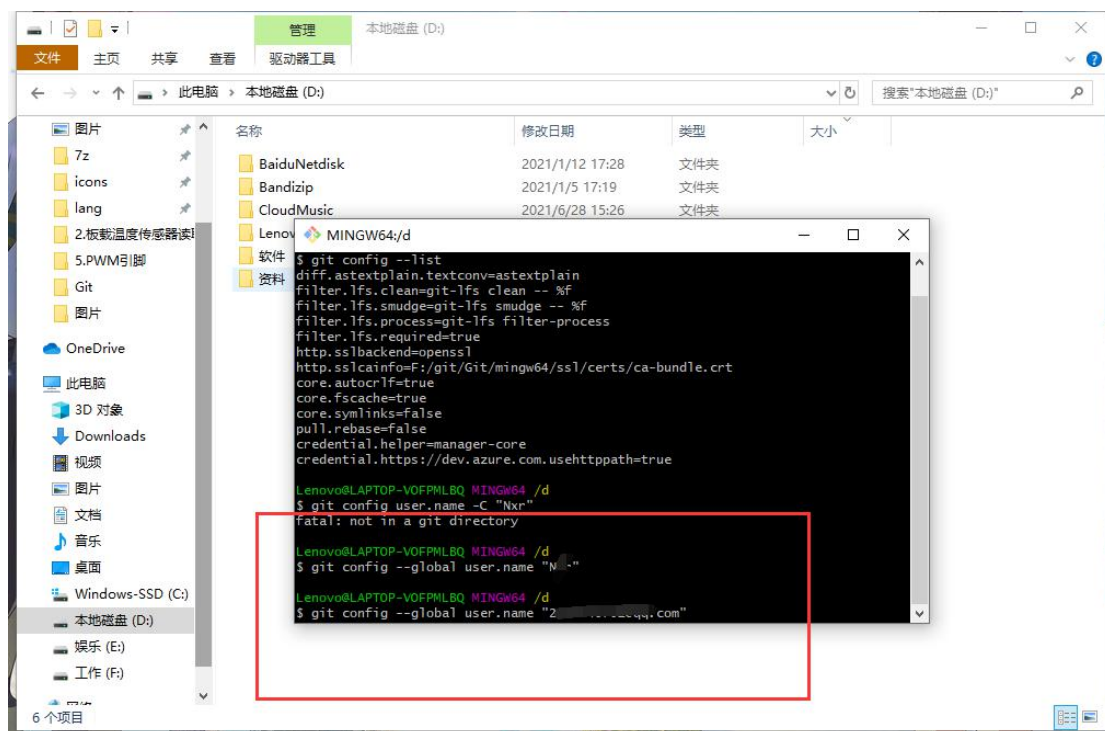


可以先输入 `git config --list` 看看自己名字邮箱正确与否, 如果没有可以用下面两个指令增改:

`git config --global user.name "你的名字"`

`git config --global user.email "你的邮箱"`

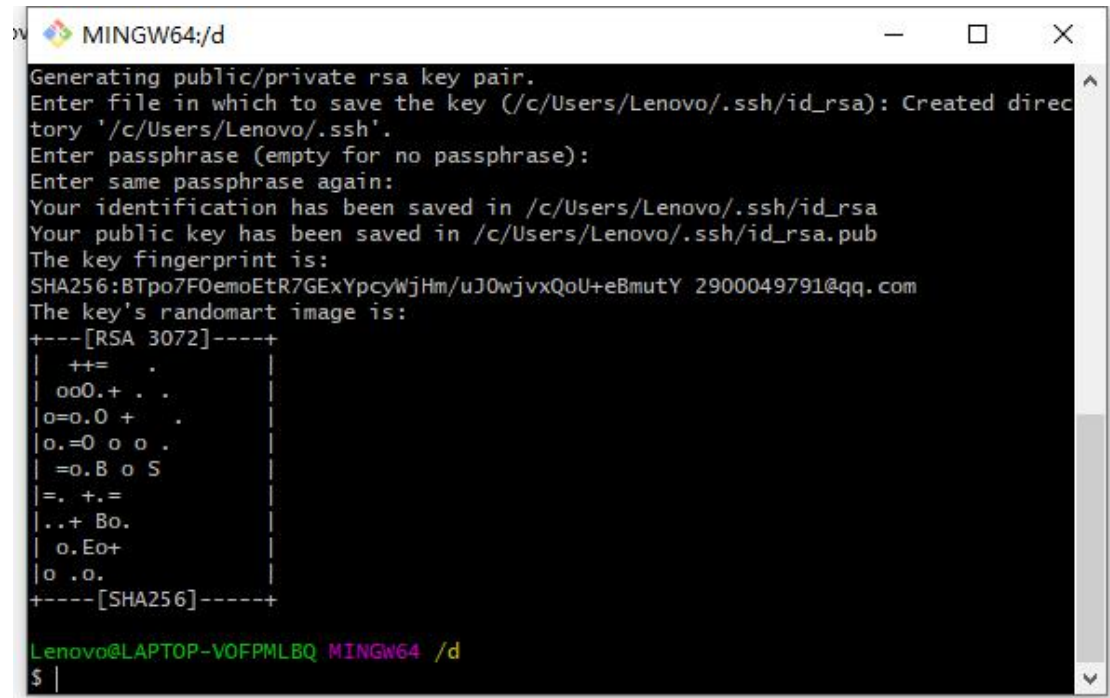
如下图:



然后使用 `SSH-keygen -t rsa -C "你的邮箱"` 生成 SSH 秘钥，这里需要按三次回车

（稍作说明一下：SSH 秘钥相当于自己电脑的身份证，有了身份证才可以在网页 [gitee](#) 上进行身份认证，认证通过了才有权管理仓库，换句话说，电脑的 SSH 泄露后每个人都可以在你的仓库上撒欢，因此不要随意告知别人自己的秘钥）

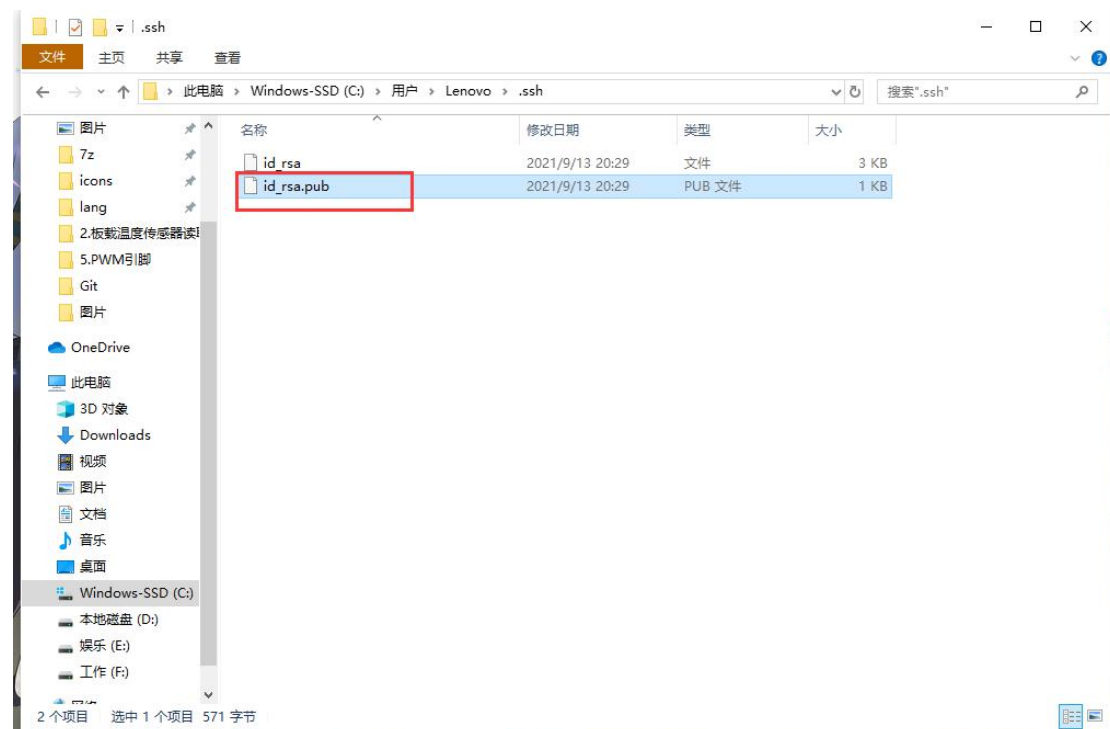
完成如下图：



```
MINGW64:/d
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/Lenovo/.ssh/id_rsa): Created directory '/c/Users/Lenovo/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/Lenovo/.ssh/id_rsa
Your public key has been saved in /c/Users/Lenovo/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:BTpo7F0emoEtR7GExYpcyWjHm/uJ0wjvxQoU+eBmutY 2900049791@qq.com
The key's randomart image is:
+----[RSA 3072]-----+
|  ++=  .                |
| oo0.+  . .             |
| o=o.O +  .             |
| o.=0 o o .             |
| =o.B o S               |
| =. +.=                 |
| ..+ Bo.                |
| o.Eo+                  |
| o .o.                  |
+----[SHA256]-----+

Lenovo@LAPTOP-V0FPMLBQ MINGW64 /d
$
```

其中包含了秘钥所在路径：

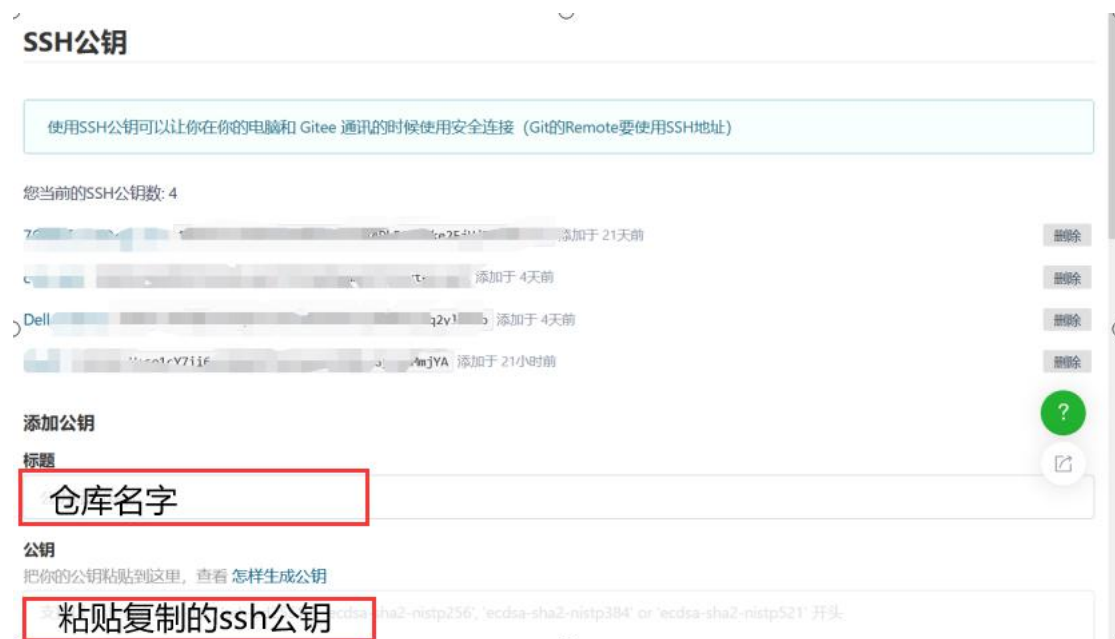


用记事本打开如上文件复制其中所有的内容。

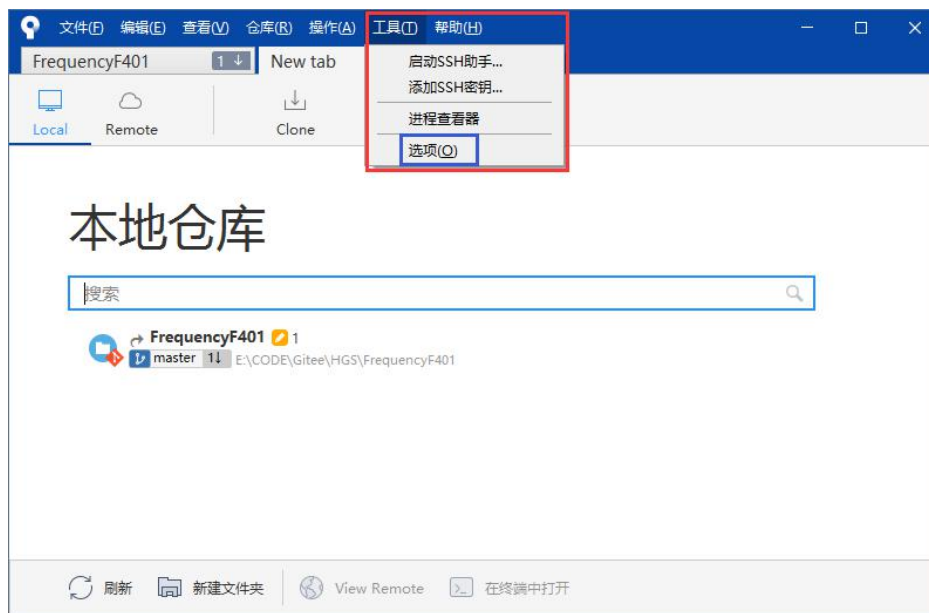
接着打开自己的 gitee 主页（没有赶紧创建一个，名字弄个自己喜欢的/原名），设置 SSH：



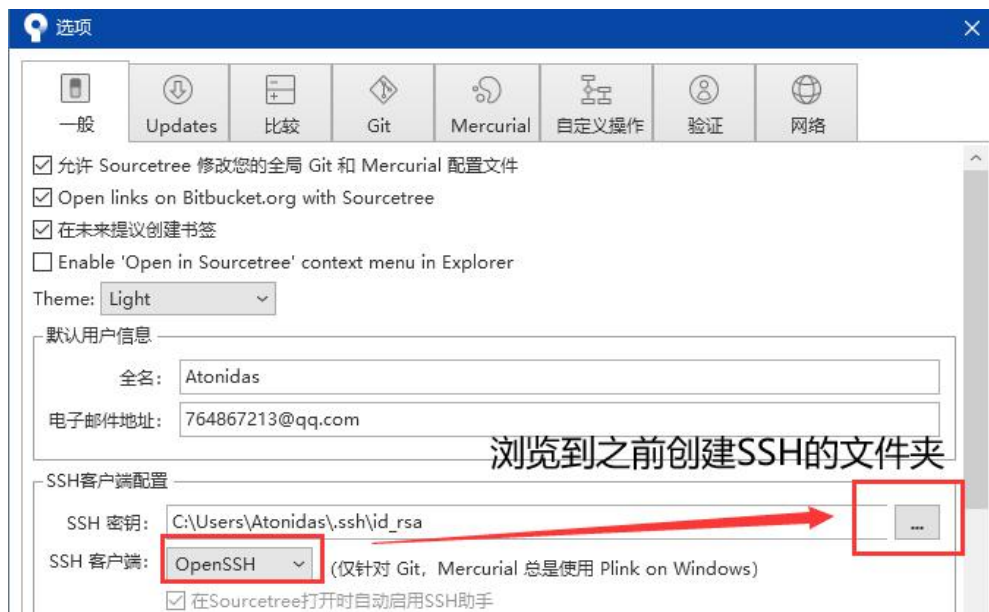
在如下地方填写自己这份公钥的标题，并粘贴刚刚复制的 SSH 公钥



接着打开 SourceTree，也添加一下 SSH 公钥：



更改 ssh 客户端的类型后通常会自动定位 SSH 秘钥的位置，如果不正确也可以自行添加路径：



到此，安装教程到此结束，

使用教程

1.仓库的创建：

接着可以简单创建一个仓库，并试着同步一次代码：

首先先在 gitee 网页上创建一个仓库，仓库名称自定义，一般以工程作名字：

其中：

1.仓库介绍这一块建议用心写一下，方便自己的管理，也可以通过修改设置模板中的 readme.md 文件中的内容进行修改，其他的根据实际情况勾选：



仓库配置仅供参考：

仓库名称 * ✓

GiteeTest

归属 ✓ **路径 *** ✓

Atonidas / gitee-test

仓库地址: <https://gitee.com/Atonidas/gitee-test>

仓库介绍 22/200

创建一个个人的代码仓库，并对其进行简单的配置

☐ 开源 (所有人可见)

☒ 私有 (仅仓库成员可见)

☐ 企业内部开源 (仅企业成员可见) ?

☒ 初始化仓库 (设置语言、.gitignore、开源许可证)

选择语言 添加 .gitignore 添加开源许可证 许可证向导 ↗

C/C++ 请选择 .gitignore 模板 请选择开源许可证

☒ 设置模板 (添加 Readme、Issue、Pull Request 模板文件)

☒ Readme 文件 ☐ Issue 模板文件 ? ☐ Pull Request 模板文件 ?

☒ 选择分支模型 (仓库创建后将根据所选模型创建分支)

单分支模型 (只创建 master 分支)

完成后就能看到自己创建的仓库了：



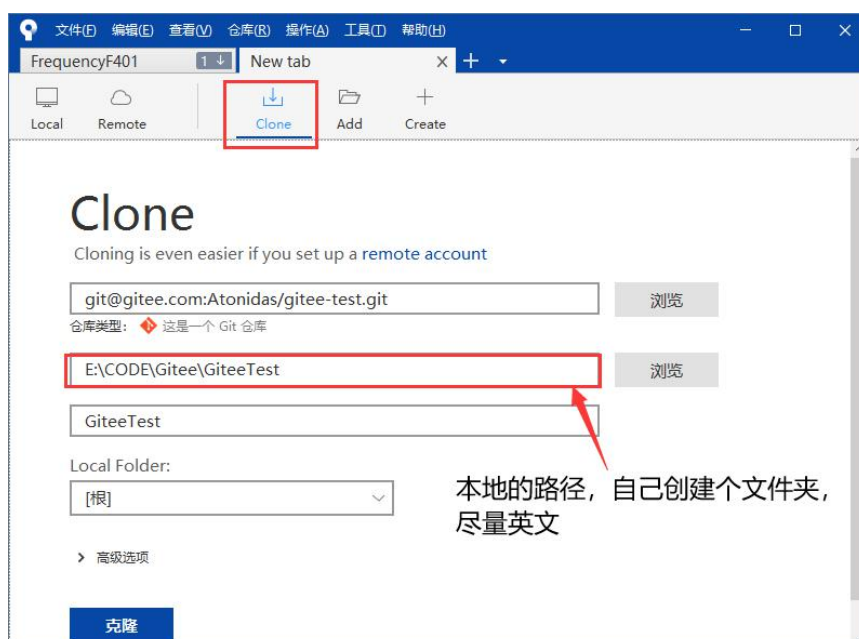
2. 仓库的克隆：

首先打开刚刚创建的仓库，复制仓库的 SSH 连接：

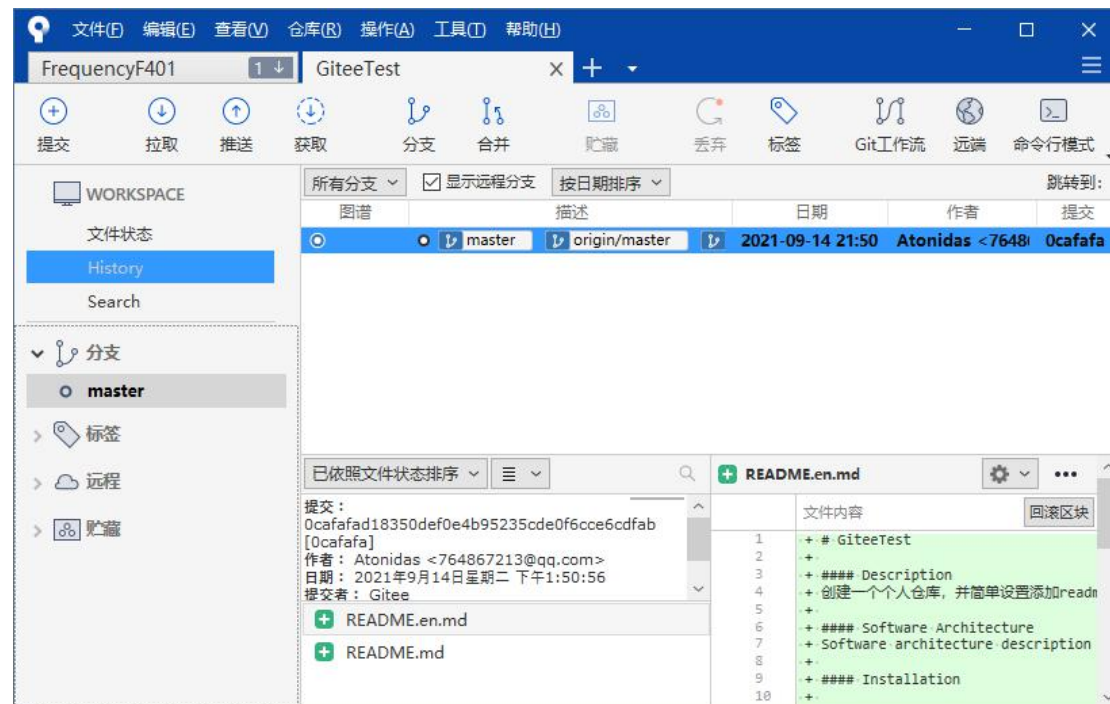


之后就可以利用 SourceTree 进行克隆，粘贴仓库 SSH 地址选择路径后克隆即可：

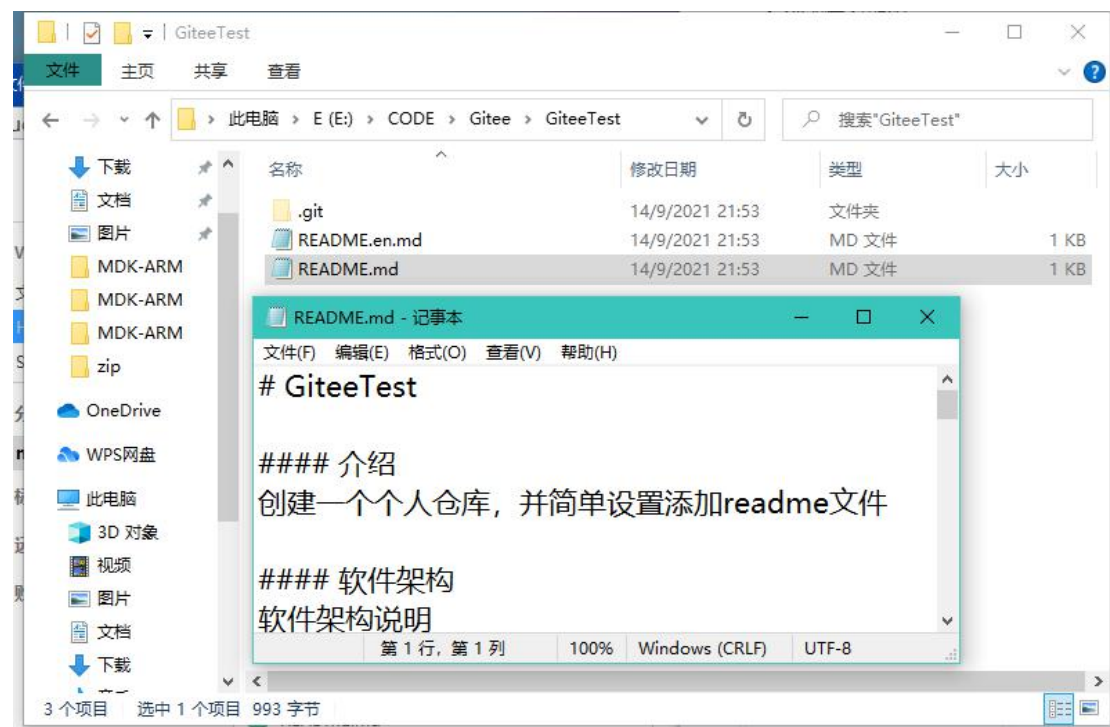
1. 第一行粘贴刚刚的 ssh 公钥
2. 第二行选择本地的路径，创建一个文件夹，通常文件夹与网上仓库名一致



完成克隆后点击 master 主分支就能看到第一个节点：

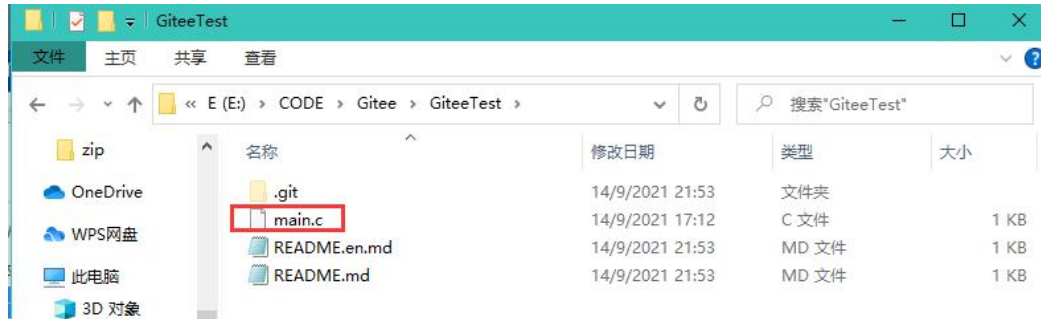


打开对应的文件夹也能看到相应的文件：

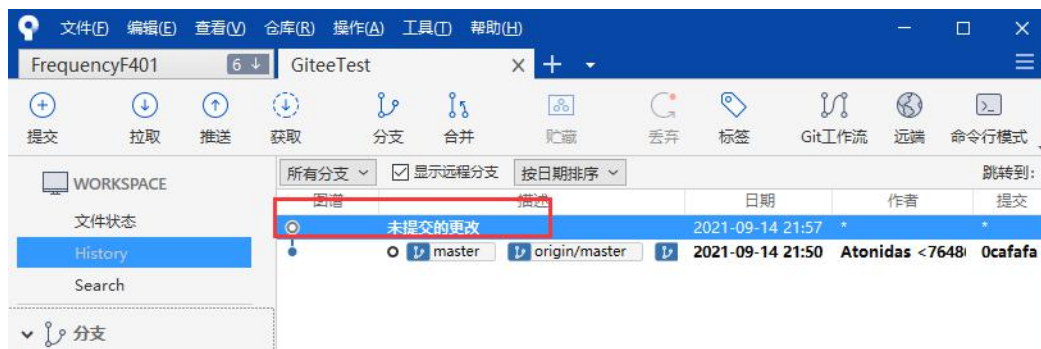


3.进行一次代码的提交：

我们随便抓个 hello world 进去（这里也可以丢一整个工程文件夹进去或者说对其中的文件进行一些内容的修改）：

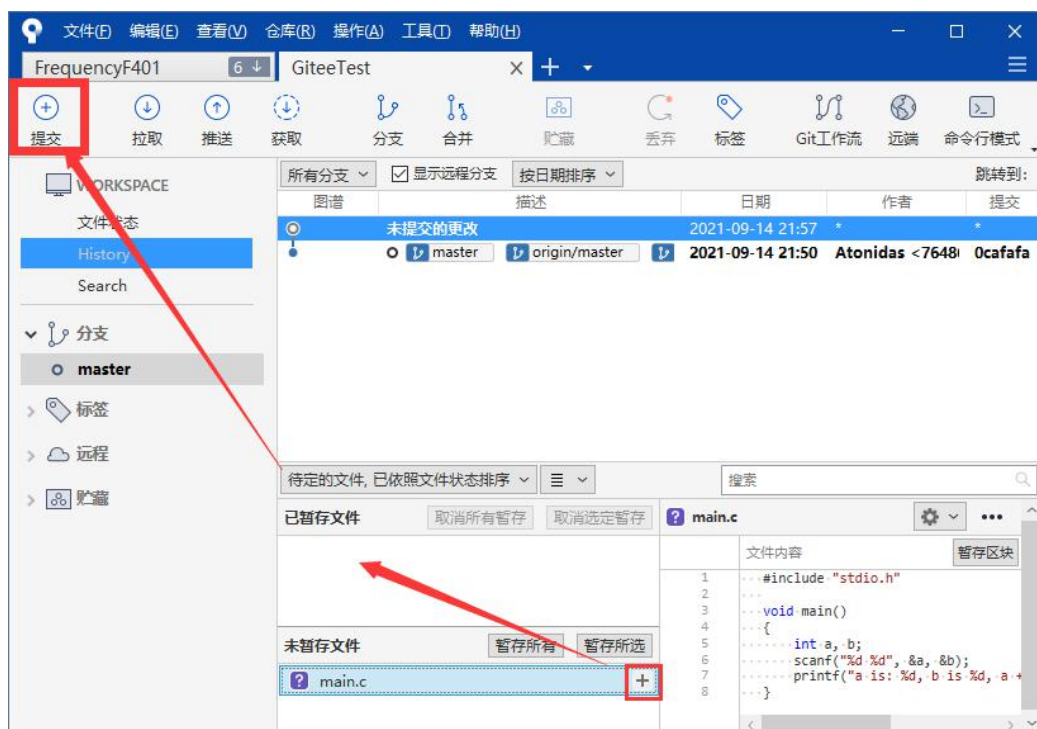


之后打开 SourceTree 能开到这次的变化：



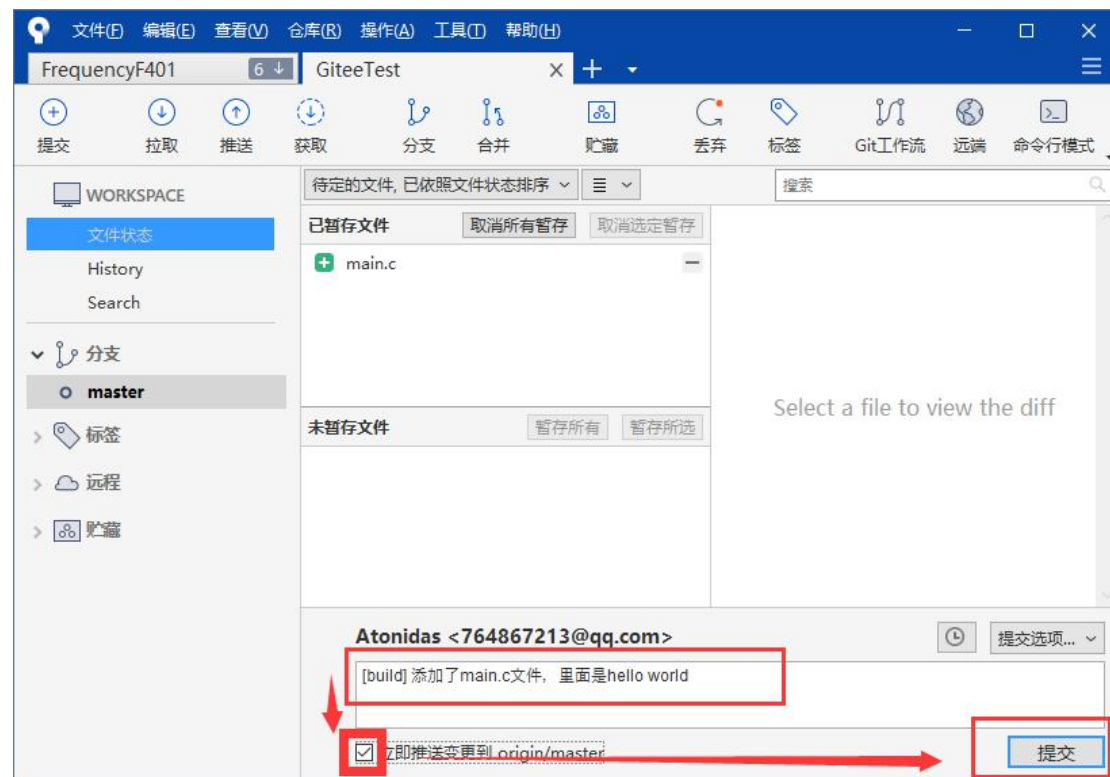
其中能看到未暂存的文件，我们可以把它暂存→提交，之后进行推送：

（在这里，如果是刚刚创建文件夹首次导入工程，我们直接点暂存所有即可；通常对于一个 MDK 工程来说，我们通常只上传改动的.c 和.h 文件即可；视情况而定）

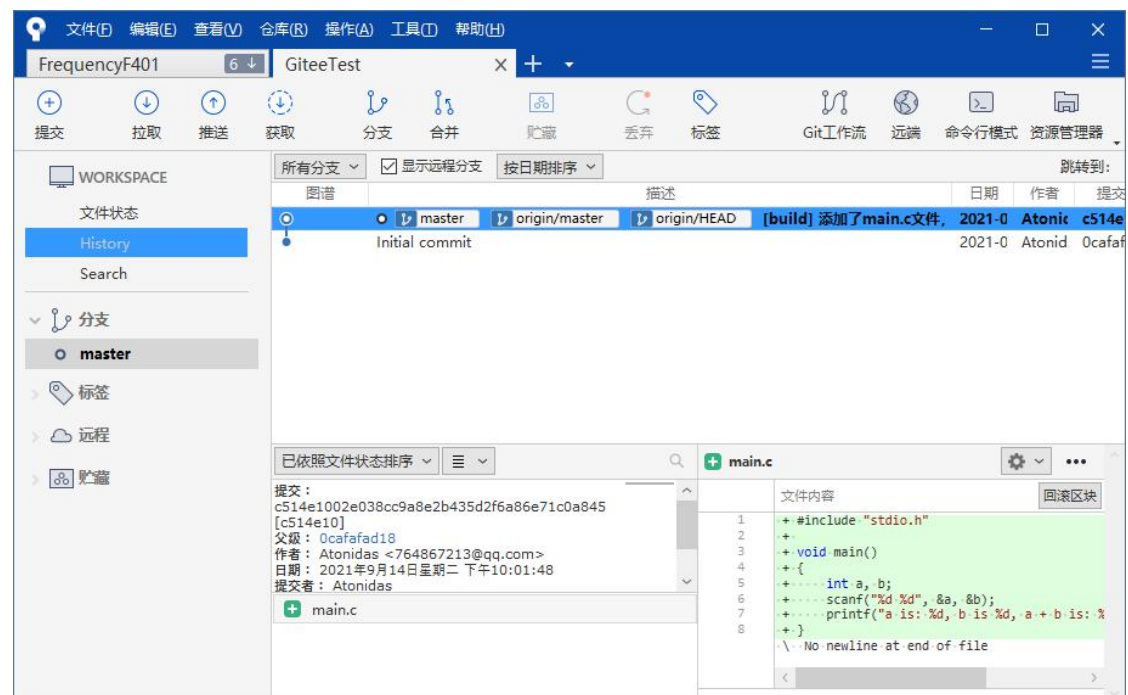


这里着重说明：

下方的第一个红框是本次上传的提交日志，养成认真写提交日志的习惯可以方便我们对自己工程回顾，或者方便在我们多人协作时能看到不同人的进度。通常日志的格式是[提交类型]+简单描述这次的提交内容/项目进度，如果需要详细说明则在换行两次后输入：



完成提交后点击 master 就能看到提交的记录了：



网页上的仓库刷新后就能看到提交记录了：

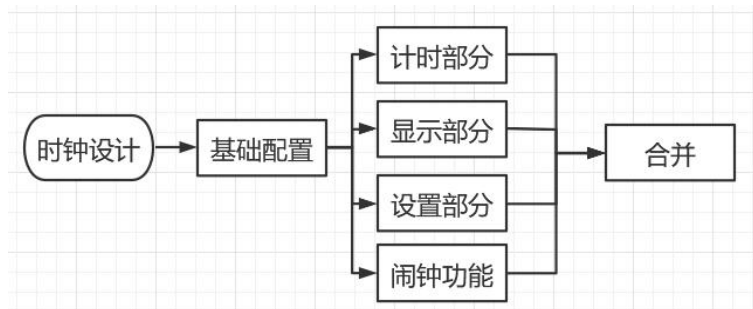


4.分支的操作

在讲版本回退之前得先引入分支的概念:之前有提到,开发过程像是一根有很多个节点的线:

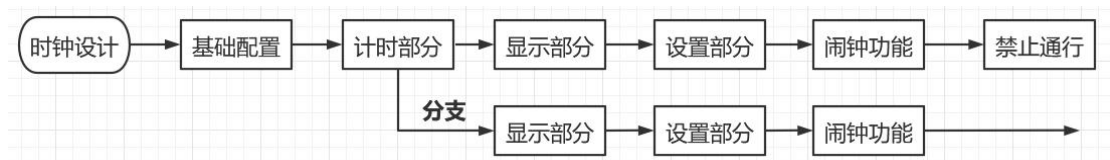


但有时候我们并不是独立一个去开发，或者一个人也可以并线开发：



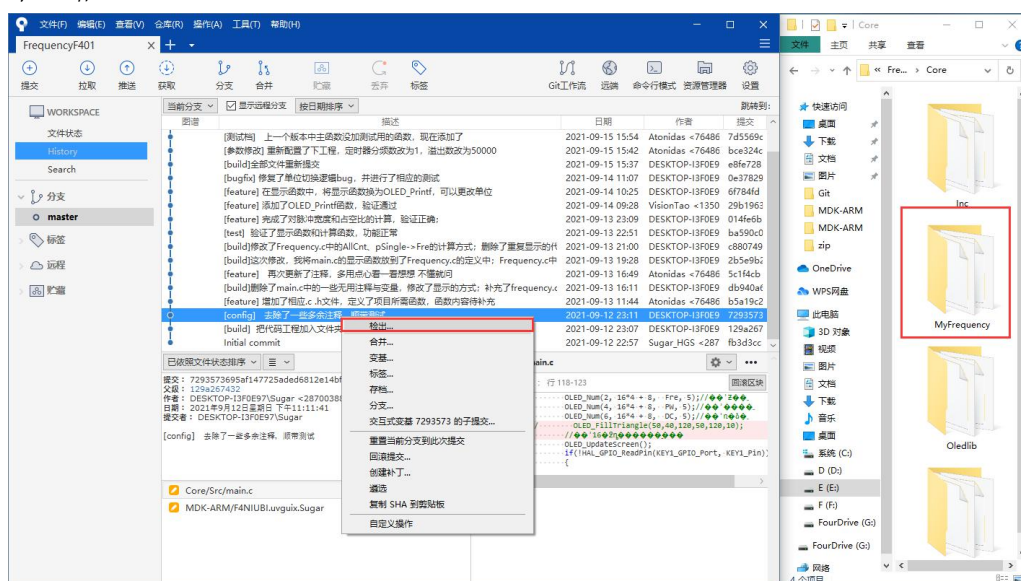
在上面的演示图中表示了将一个项目分成了四个分支，可以互不干涉的开发，再合并。

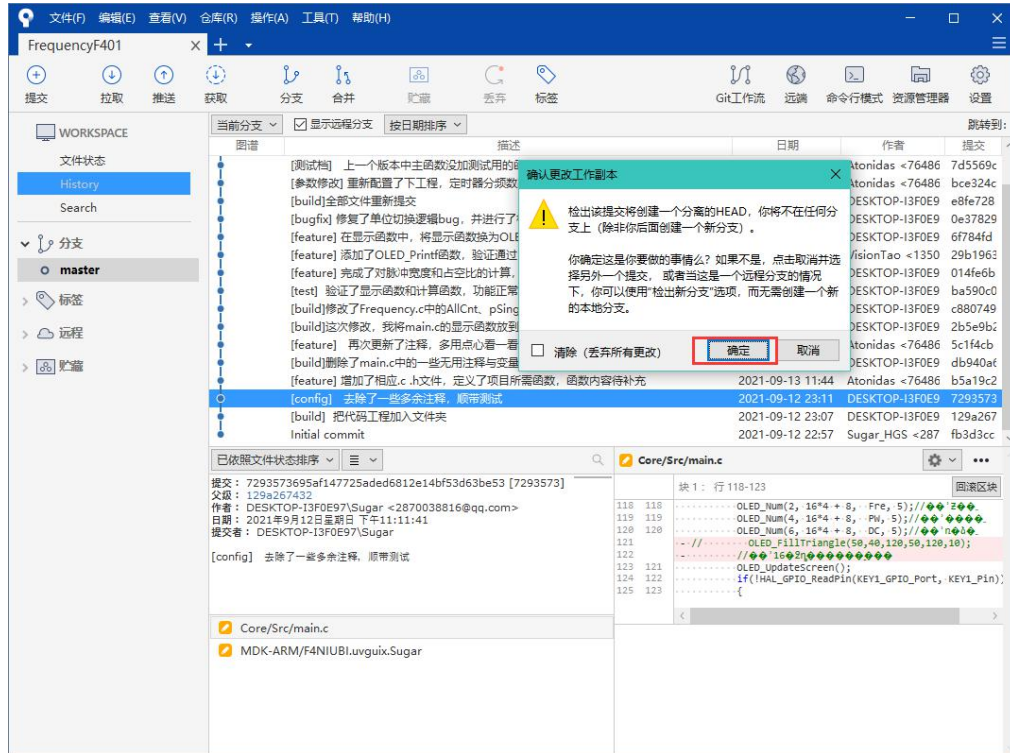
当然也有可能一直向前走发现走不通想在某处 重新来过 的情况:



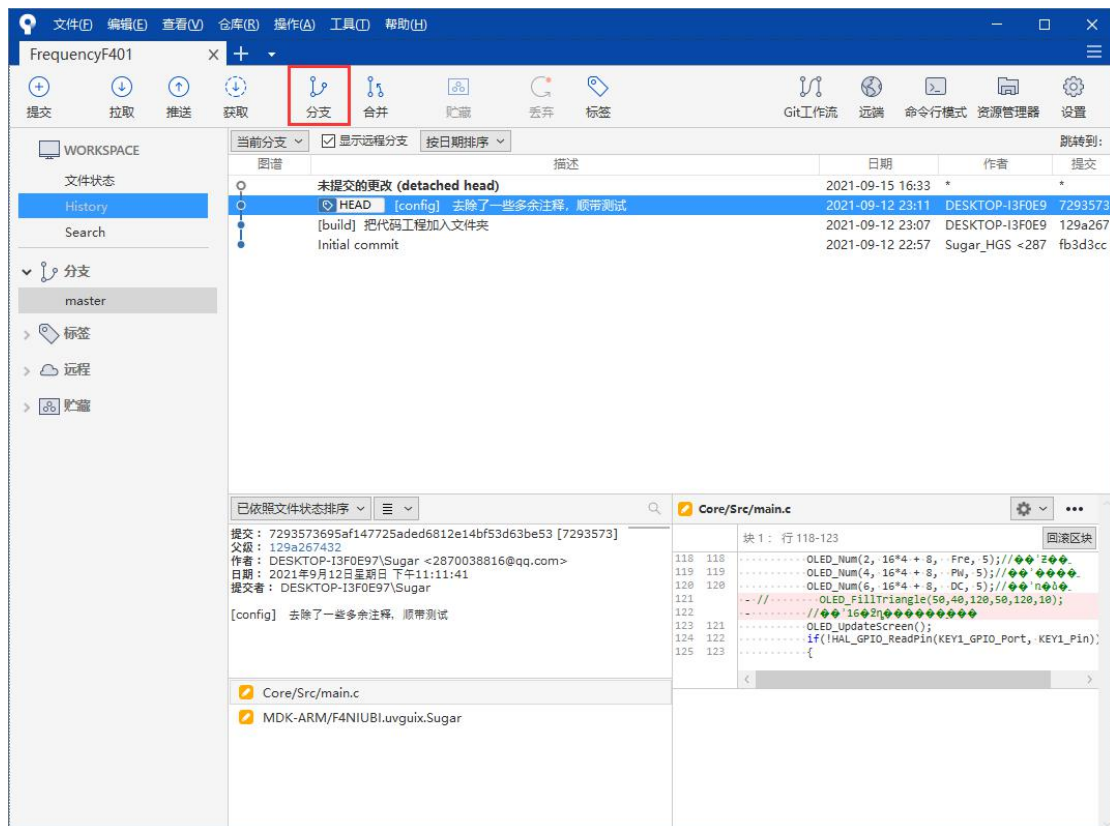
这个时候可以利用分支功能在指定的的时间点创建一个分叉路口：

先创建一个“分叉路口”：（在这点之后，右边文件夹会被创建，我们在创建前的版本弄一个“路口”）

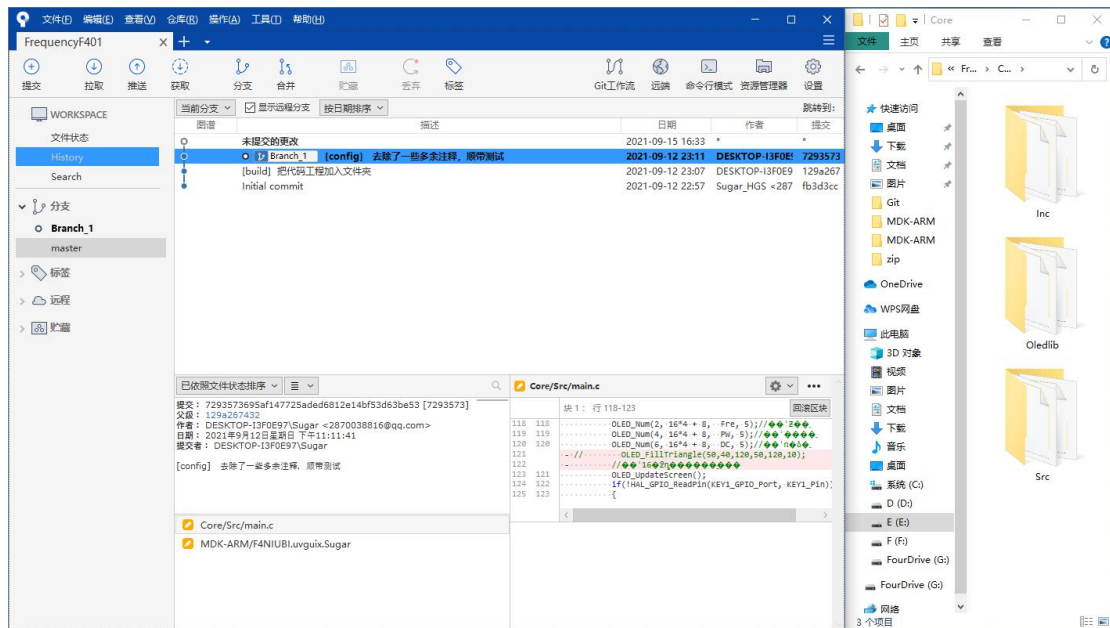




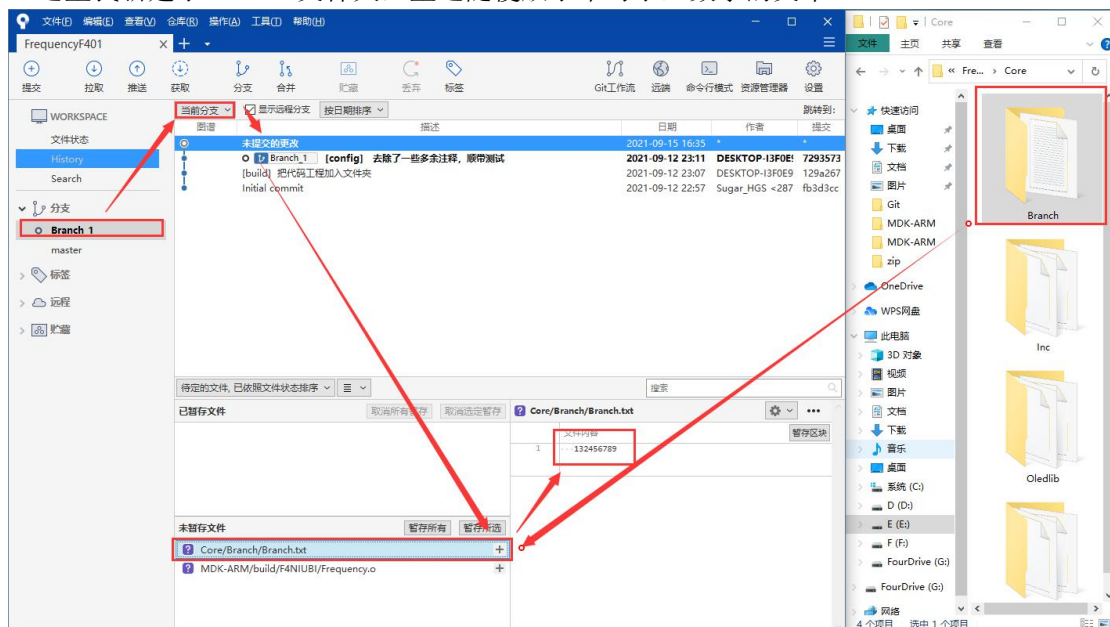
并以这个路口为开始点创建一个分支:



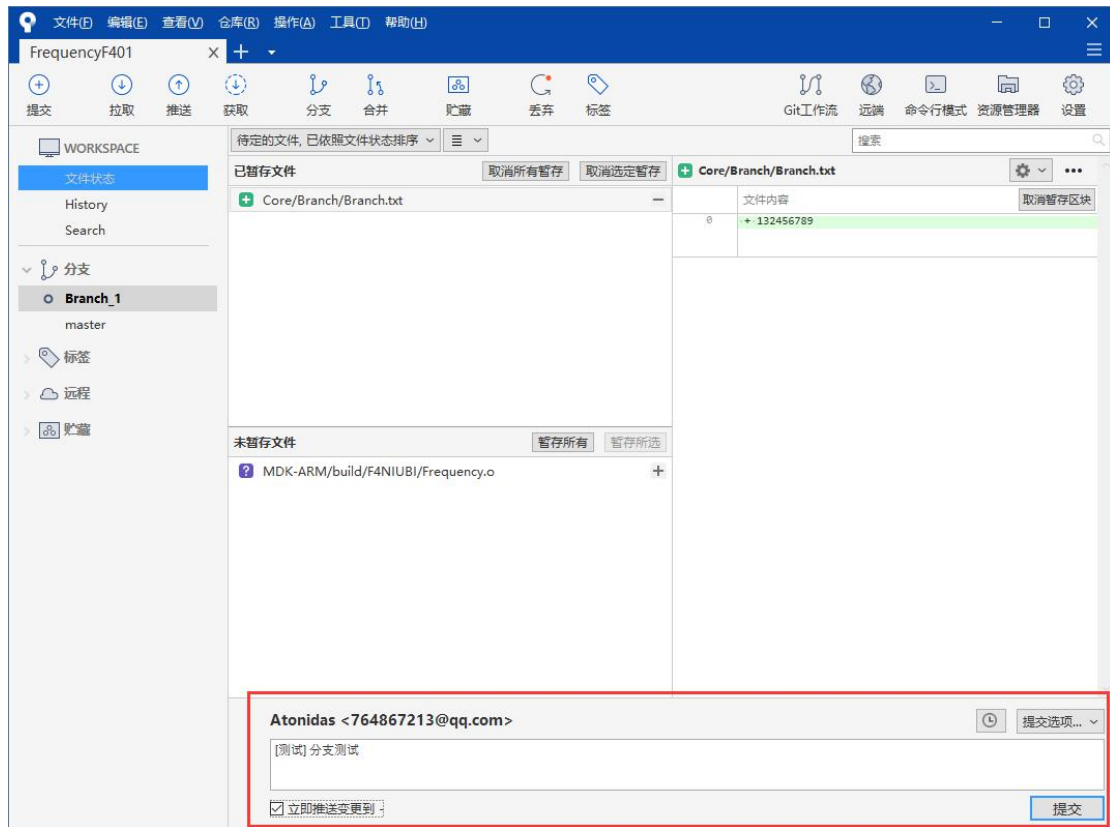
创建后左边选择新分支后，在当前分支选项中就只看路口之前的日志了，同时右边对应的文件夹也不见了：



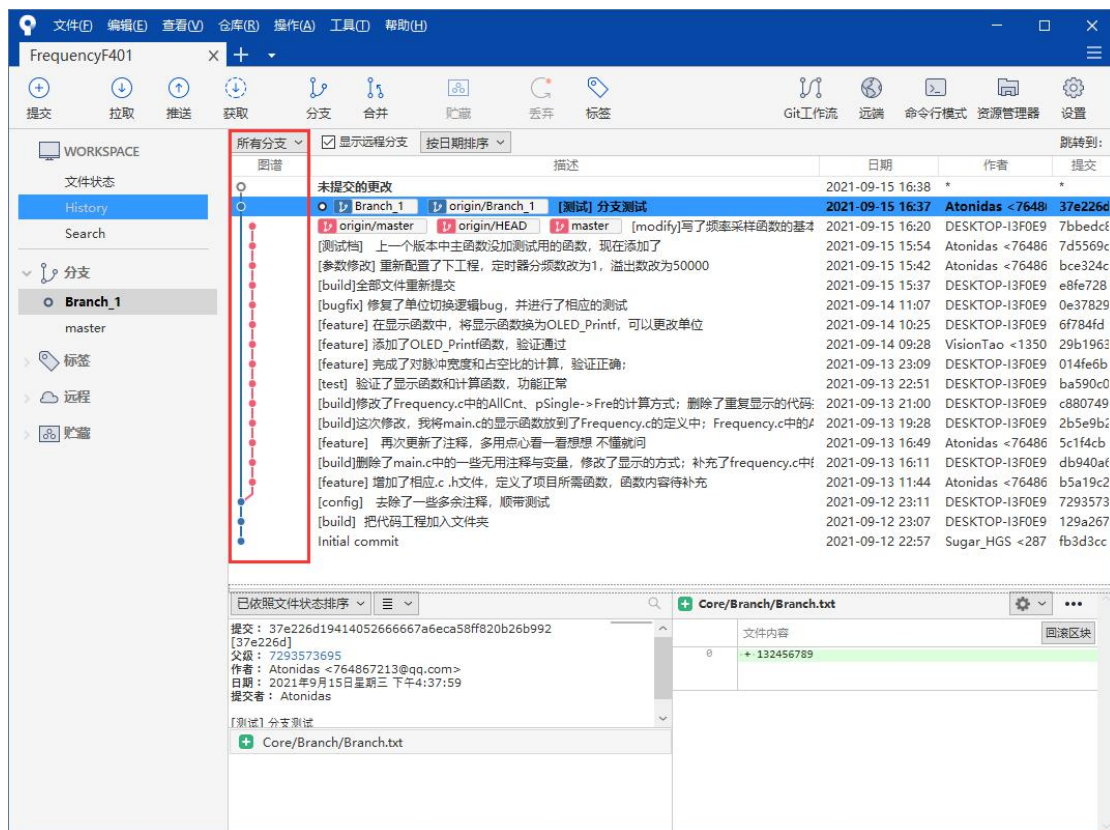
我们试着在这条路上往前走→创建/更改文件并正常提交推送一次：
(这里我新建了 Branch 文件夹，里边随便放了个写了些数字的文本)



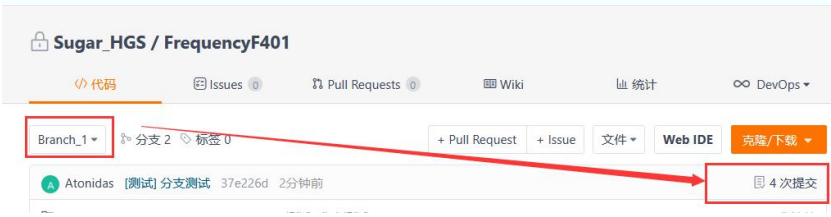
正常提交写日志：



完成提交和推送后点击 所有分支就能看到两个“树杈”：



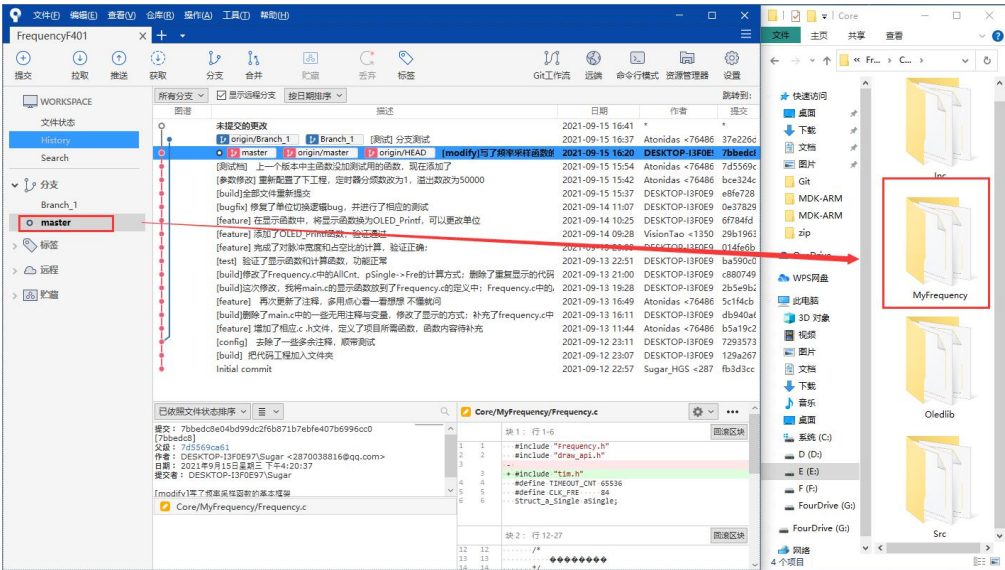
在网页上也能看到新的分支被建立，以及相应的提交记录：



[测试] 分支测试



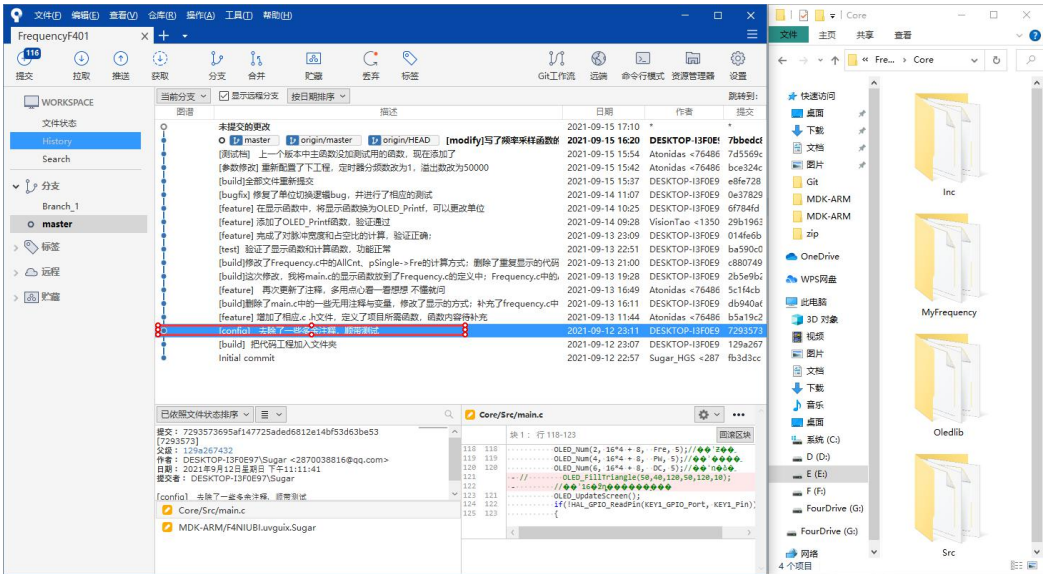
我们也可以随时切换分支，直接双击要切换到分支即可：
(这里切换回主分支，相应文件夹出现)



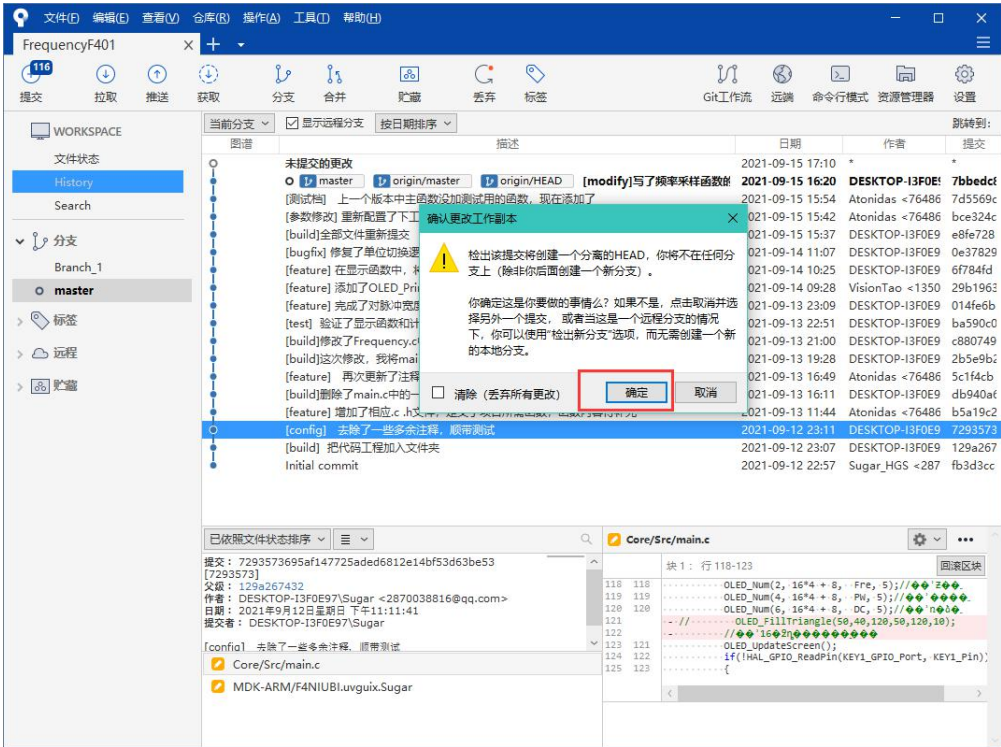
到此，简单的分支操作和概念结束，分支合并操作会在后续教程版本更新。
不难发现，这样也是一种版本回退的方法，只不过这时的回退会引出新的分支，下面会介绍不产生新的分支的回退方法。

5. 单次回退：

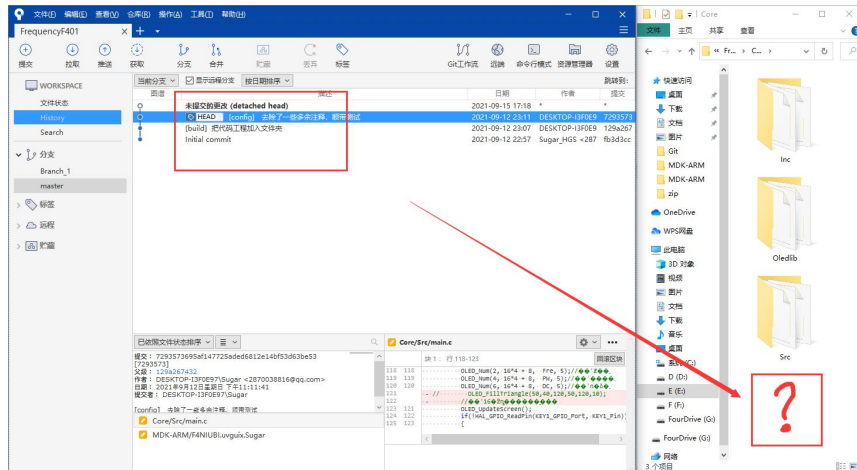
有时候我们只是想回去烧录个代码，或者演示一下，就相当于是在路口停一下，不会往前走不需要创建一个新的分支。这时候我们直接双击想回退的版本就行了：
(同样以文件夹创建前的版本为例子)



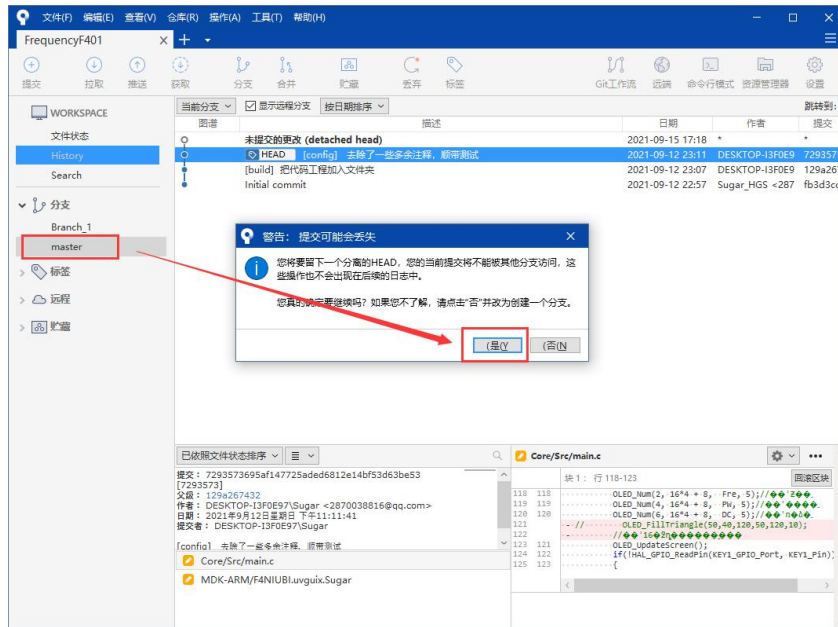
这个时候会暂时创建一个 HEAD 分支在虚空，可以理解为在分支的路口驻留：



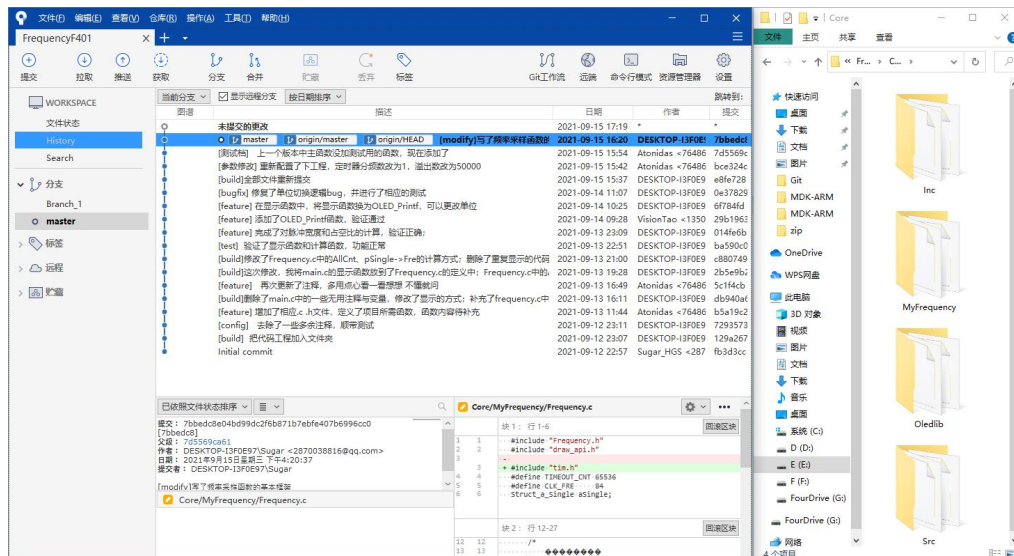
直接确定后我们能看到我们回溯到了文件夹创建之前，后续的日志也没了：



想要从“过去”回到“现在”时双击 master 主分支即可，驻留时的事均不保留：



接着又是“现在”的景象了:(回退内容到此结束)

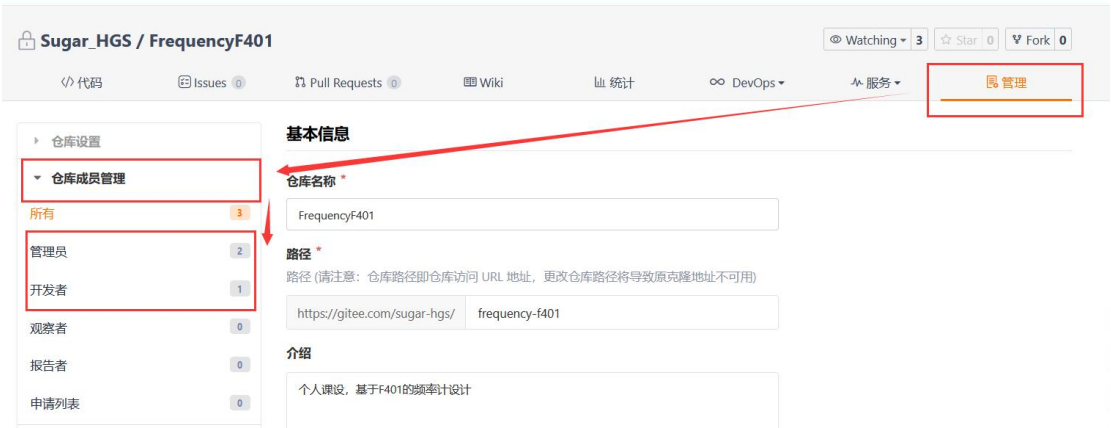


6. 多人合作

开始多人合作时需要给你的小伙伴分配个仓库的权限：

其中：

- 拥有者：可以开除管理员，删除仓库，老大哥；
- 管理员：拥有所有权限，可以编辑仓库，邀请成员；
- 开发者：可以编辑，但是不能邀请成员；
- 观察者：只能观察仓库，无权编辑。



邀请用户

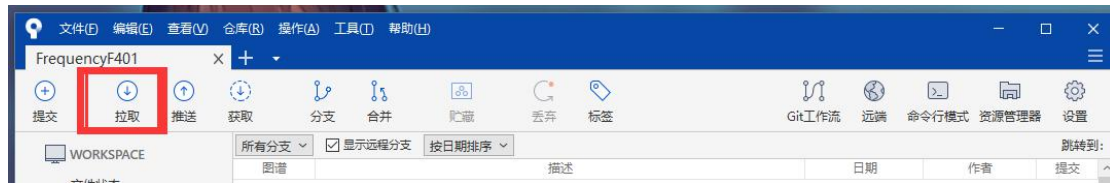


通过分享链接地址或二维码邀请成员：



接着皆可以开始和小伙伴的合作之旅啦：

PS：开始干活前记得拉去一下仓库，保持最新状态：



教程版本：V1.1

作者 Gitee：Atonidas

校对：在找着

声明：任何组织和个人，都可以随意分享使用文档，欢迎反馈问题或一同参与完善。