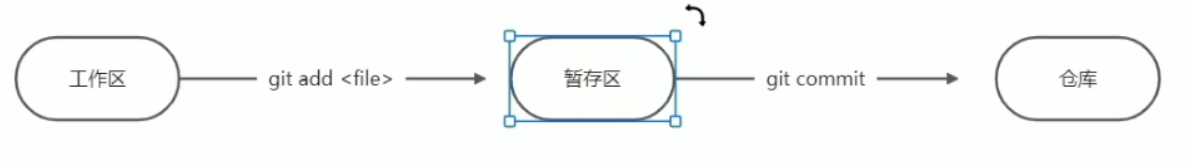
Git基础 -- (vscode版)

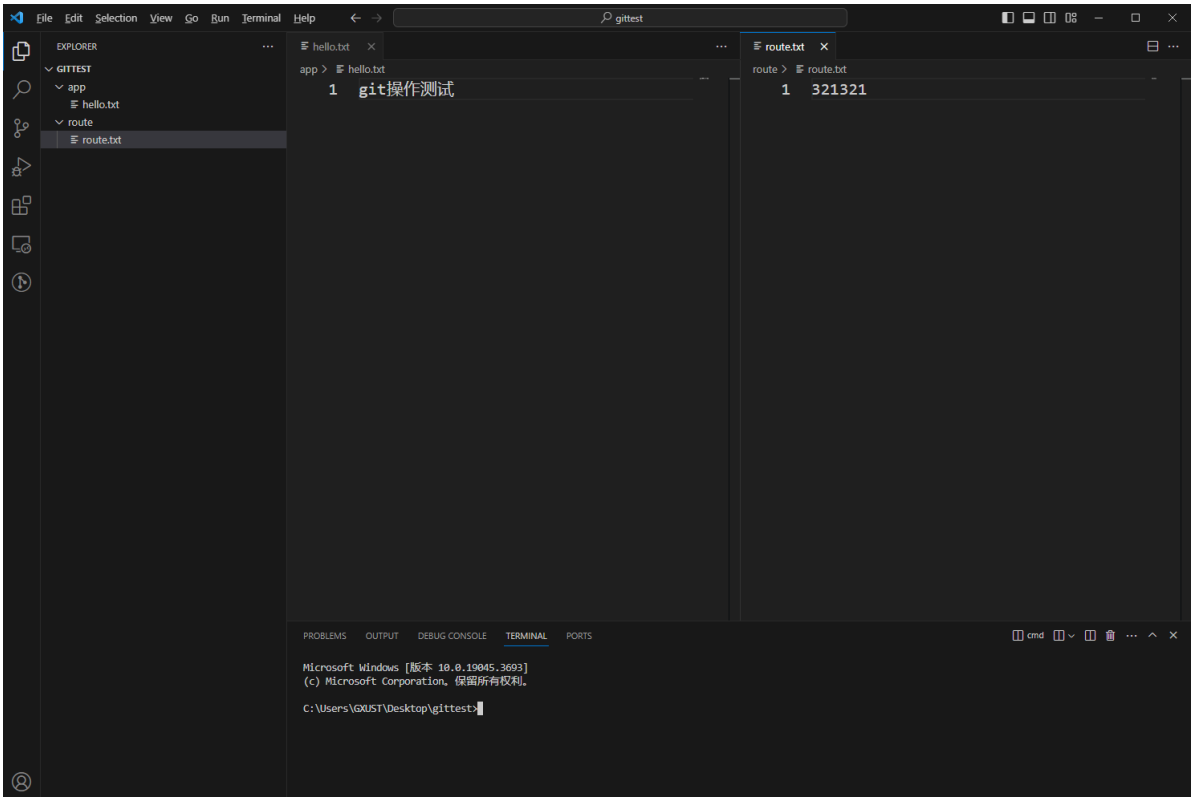
一、仓库和项目的概念

在github中直接下载得到的是一个项目文件，而使用git clone拿下来的是一个项目仓库。文件区别上后者比前者多一个.git文件。

二、本地仓库、暂存区(本地)、远程仓库之间的关系



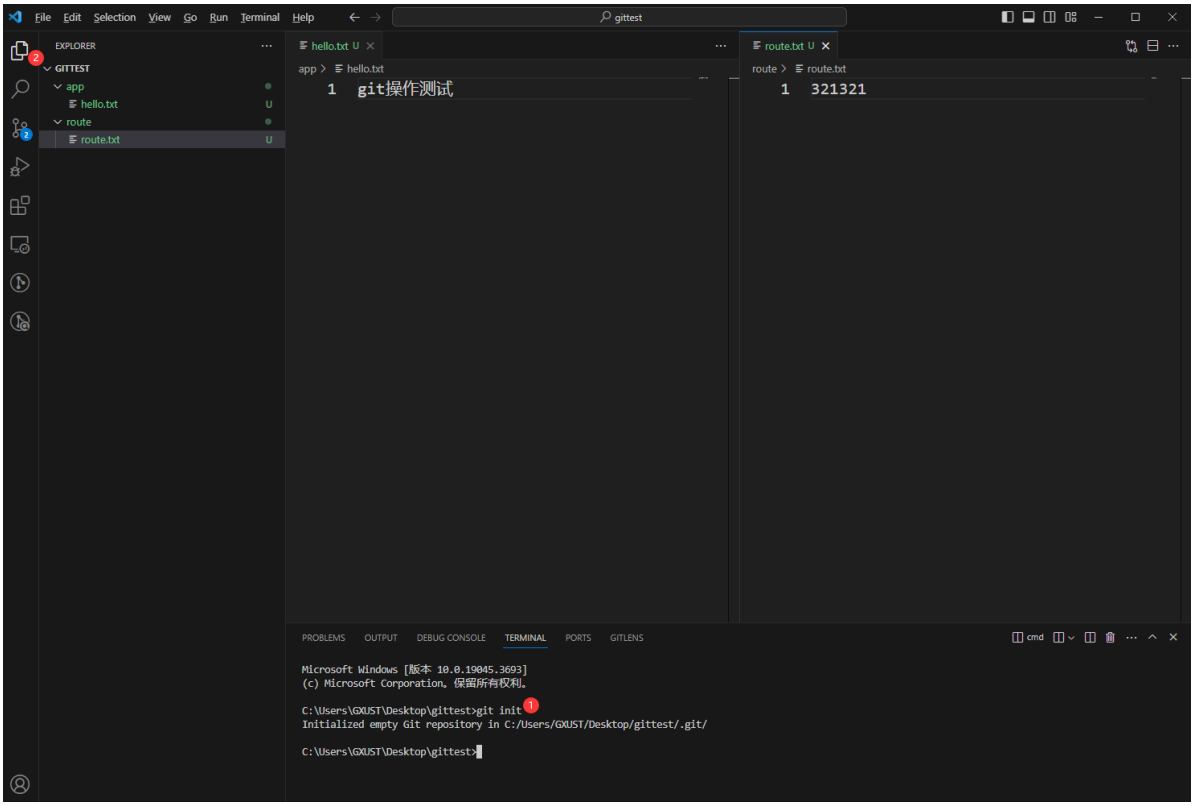
三、本地仓库初始化



本地项目首先使用下述命令将本地文件夹初始化为一个git仓库

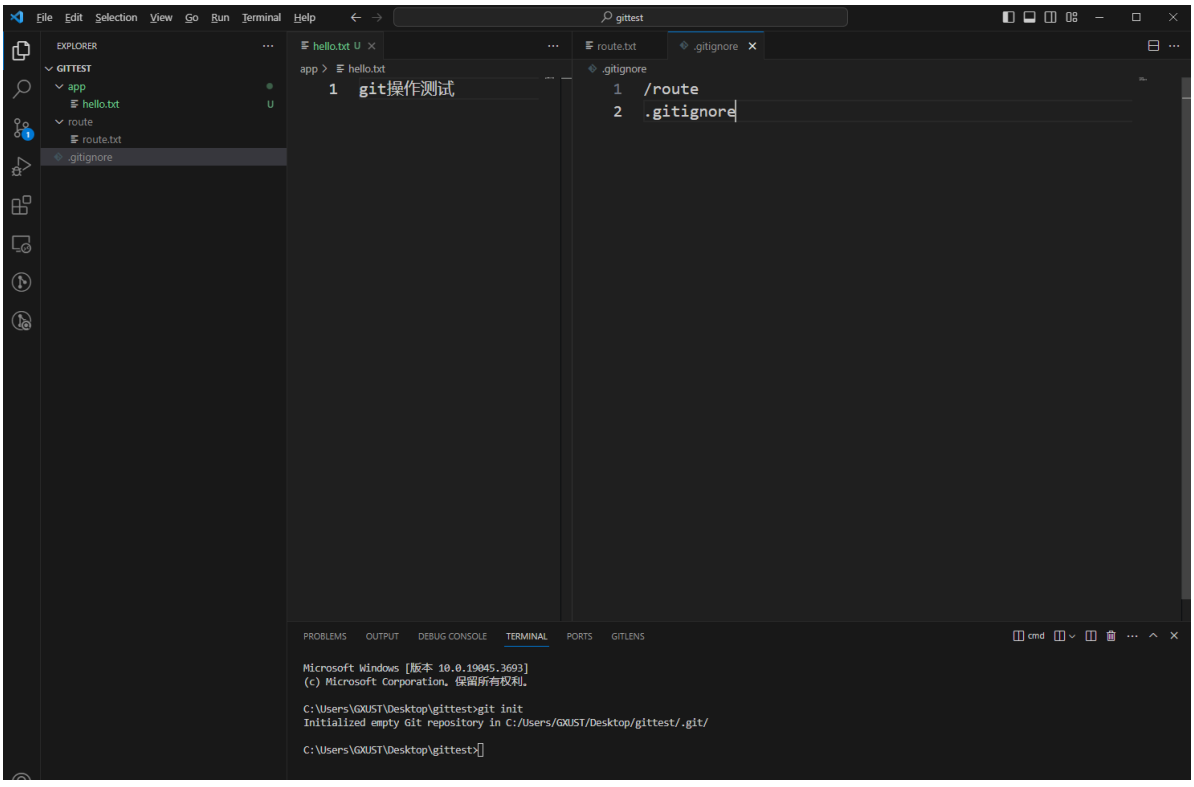
```
git init
```

可见所有的文件成为待提交的绿色



四、.gitignore文件

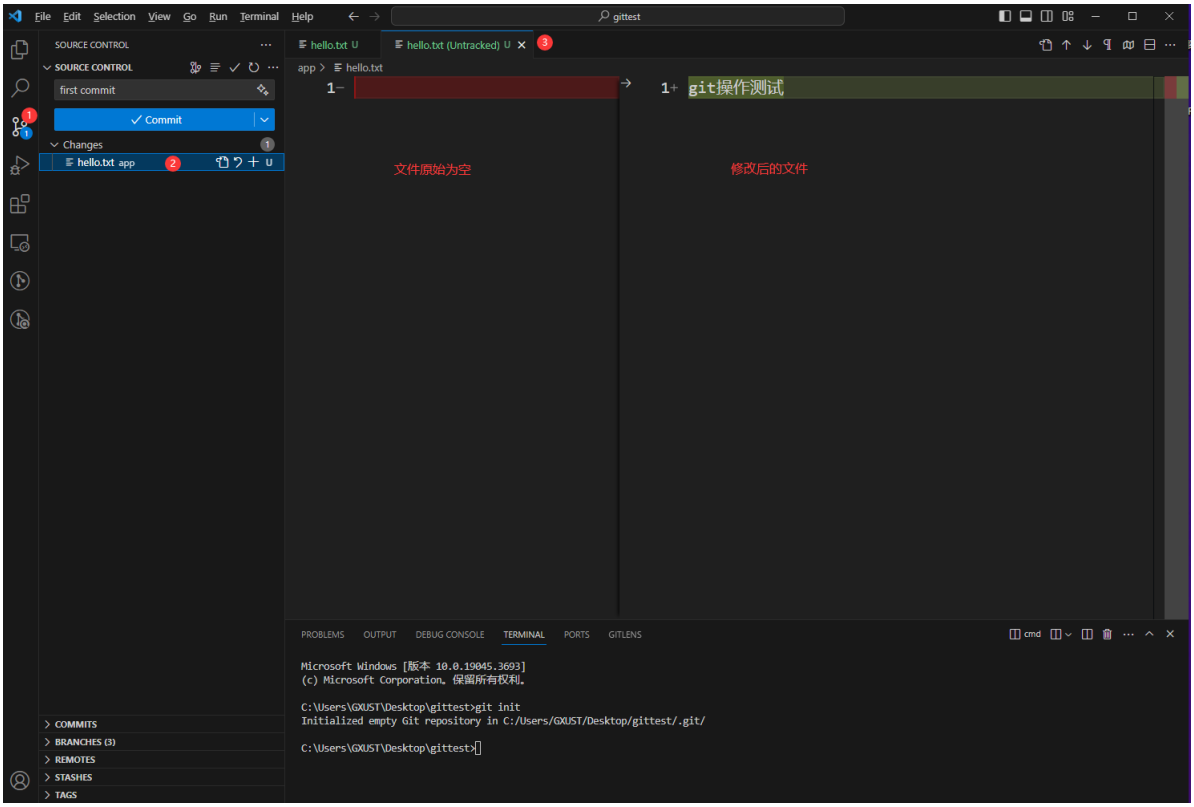
用于忽略不需要提交的文件或文件夹，在仓库目录下创建.gitignore文件，如将.idea文件夹及其下的所有文件忽略提交，被忽略的文件为灰色：



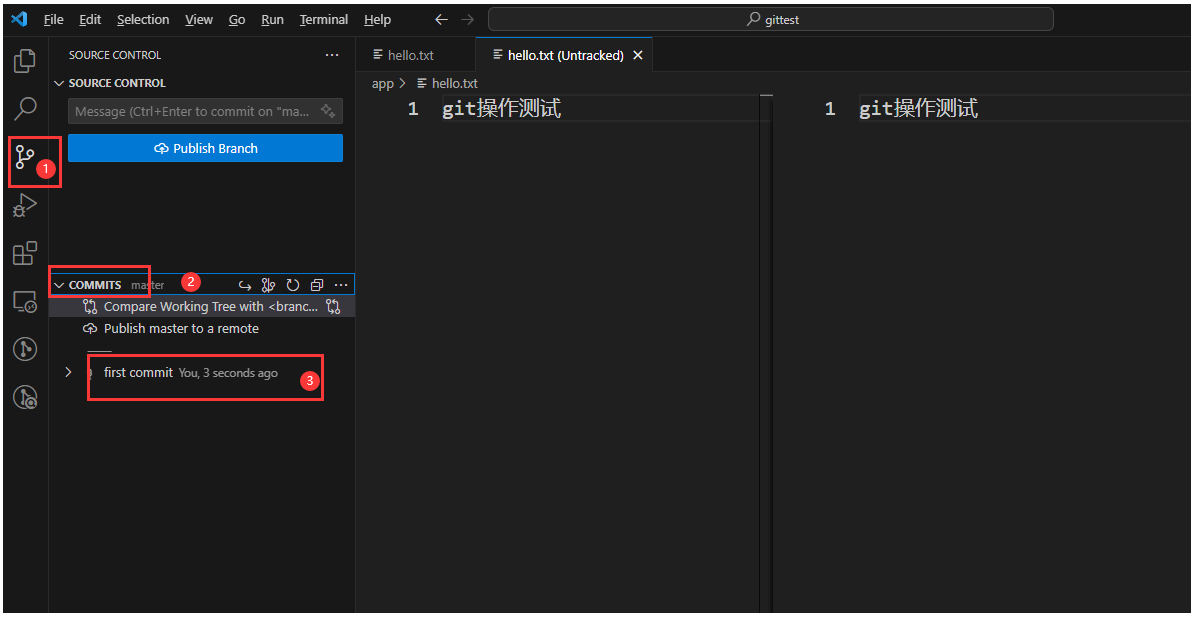

```
#注释              .gitignore的注释
*.txt              忽略所有 .txt 后缀的文件
!src.a             忽略除 src.a 外的其他文件
/todo              仅忽略项目根目录下的 todo 文件，不包括 src/todo
build/             忽略 build/目录下的所有文件，过滤整个build文件夹；
```

```
doc/*.txt       忽略doc目录下所有 .txt 后缀的文件，但不包括doc子目录的 .txt 的文件

bin/:           忽略当前路径下的 bin 文件夹，该文件夹下的所有内容都会被忽略，不忽略 bin 文件
/bin:           忽略根目录下的 bin 文件
/*.c:           忽略 cat.c，不忽略 build/cat.c
debug/*.obj:    忽略debug/io.obj，不忽略 debug/common/io.obj和tools/debug/io.obj
**/foo:         忽略/foo，a/foo，a/b/foo等
a/**/b:         忽略a/b，a/x/b，a/x/y/b等
!/bin/run.sh    不忽略bin目录下的run.sh文件
*.log:          忽略所有 .log 文件
config.js:      忽略当前路径的 config.js 文件

/mtk/           忽略整个文件夹
*.zip           忽略所有.zip文件
/mtk/do.c       忽略某个具体文件

# 上述例子中的根目录/均指.gitignore文件所在的文件夹。
```

五、本地仓库首次提交(工作区 -> 暂存区)

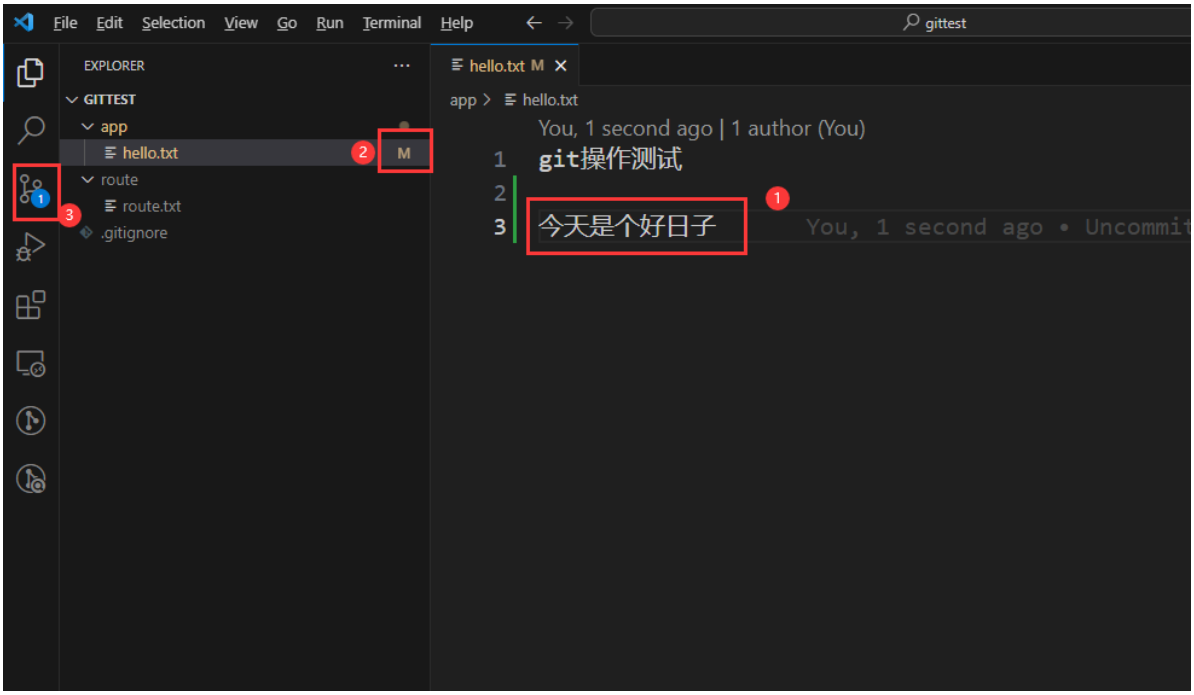


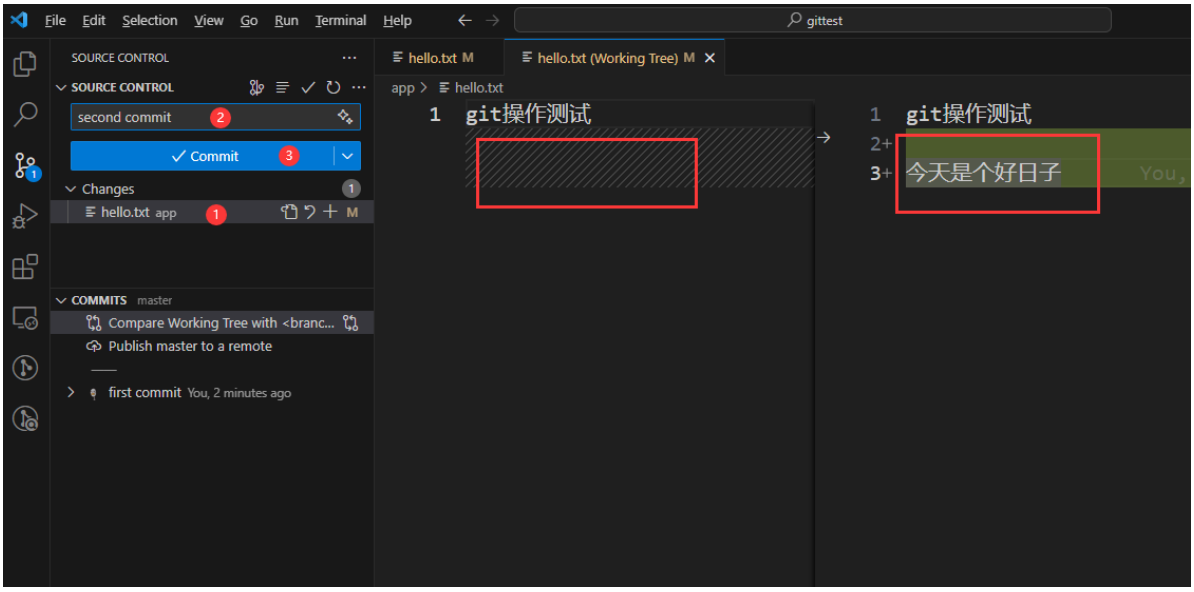
在左侧commit上方添加提交注释，然后点击commit即可将文件提交到暂存区



六、文件修改后提交到暂存区

在第一次commit的基础上，若代码作了修改需要第二次commit到暂存区。

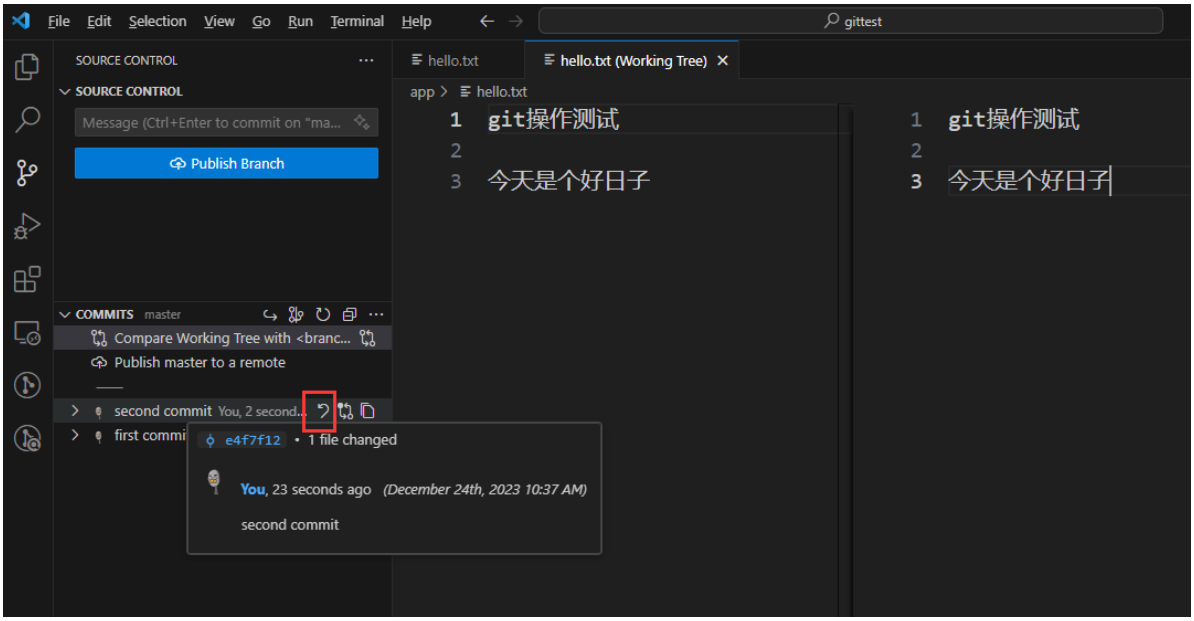


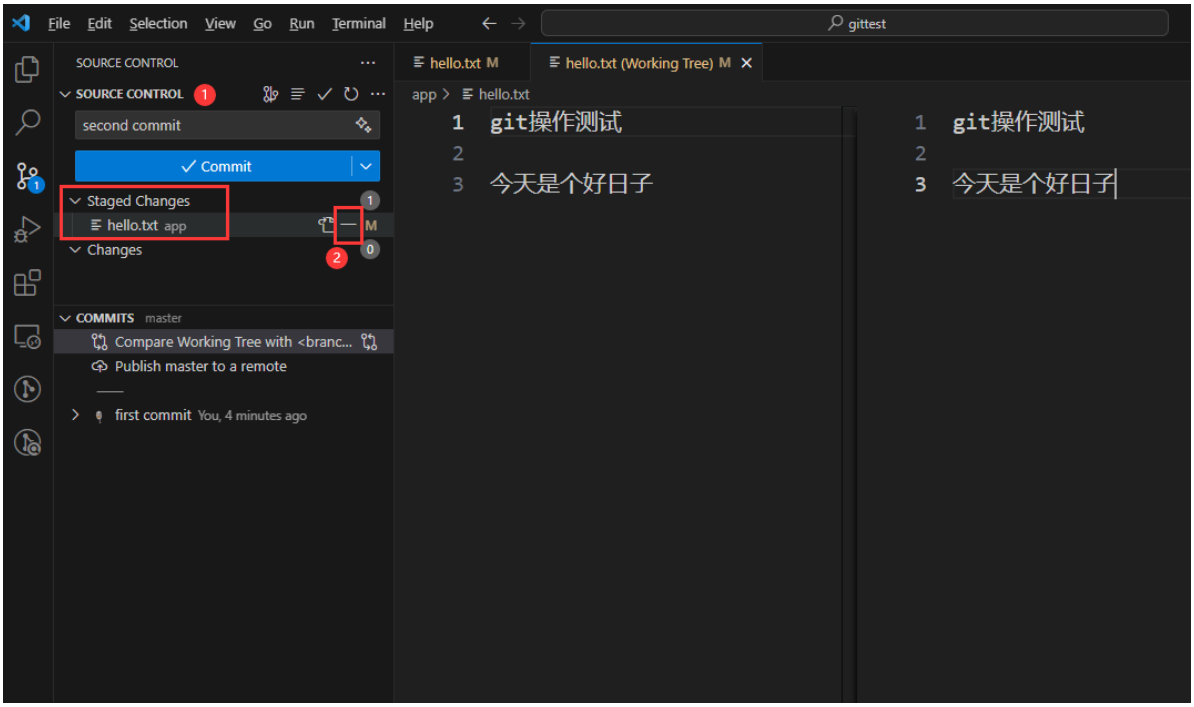


上述修改完后写好提交说明 点击commit完成修改后的文件提交，提交到暂存区

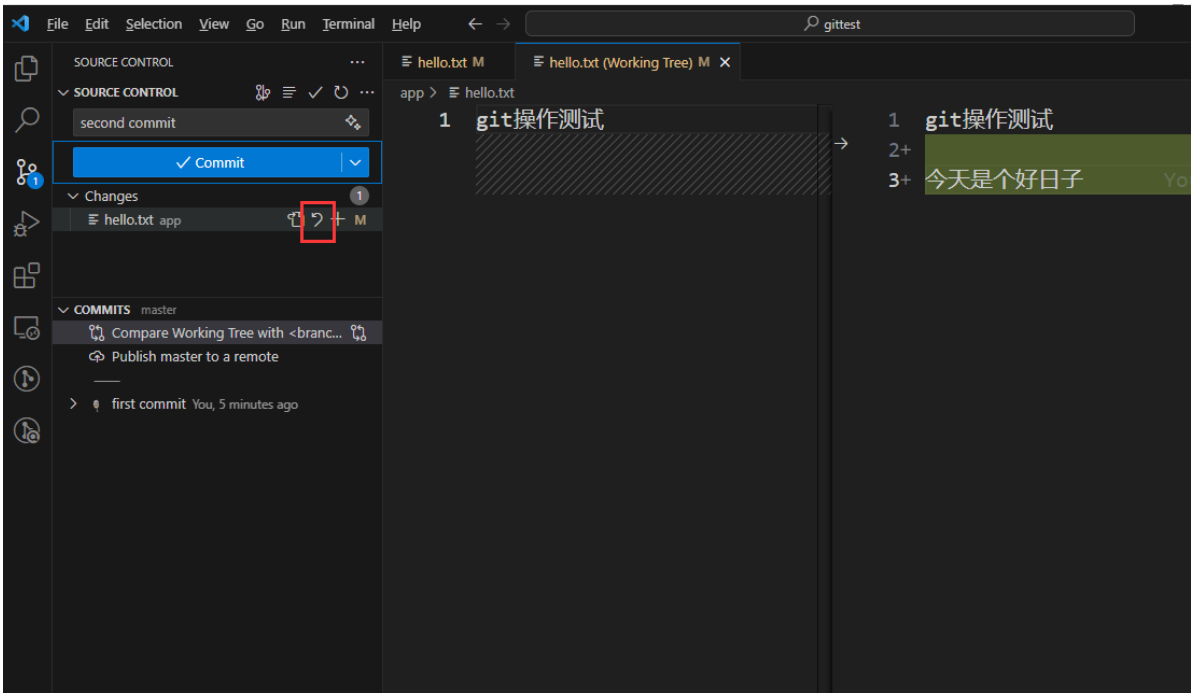
七、提交回滚

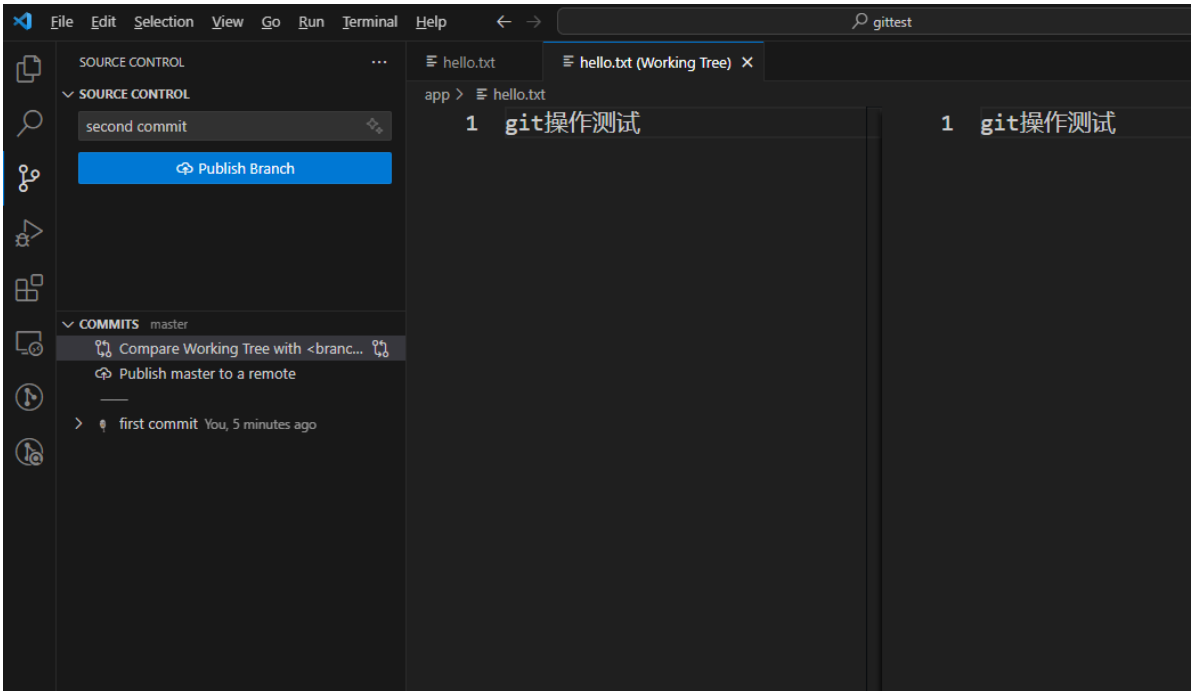
如果上述情况下修改完之后又不想修改了，想要回原来版本的代码，可以在COMMITS下找回刚刚的提交操作，使用回退操作将提交回滚到工作区





在工作区的修改文件取消更改，然后再执行一次工作区的回滚，即可回退到删除了代码前的版本





上述是已提交到暂存区的情况，如果刚刚改完但是还没提交到暂存区的话，同样在changes位置直接回滚到修改前即可，少了一步从暂存区回滚到工作区这一步。

```
# 流程：已提交（暂存区 -> changes -> 工作区）
#       未提交 （changes -> 工作区）
```

分支及其合并操作

分支的用处是为了在一个项目仓库中进行多人协作，不同的开发人员首先按照主分支作为副本新建分支，然后再分支上进行代码修改，最后再将分支修改的内容合并的主分支上，完成分支合并。主分支是一切分支的起点(修改起点)，也是所有分支的终点(分支合并)

一、查看分支

```
git branch

C:\Users\GXUST\Desktop\gittest>git branch
* master
```

二、新建分支

注意：新建分支时，当前所在的分支位置就是新建分支的复制副本(新建分支和当前所在分支的内容是一样的)，并且分支之间的代码是相互隔离的，意味着修改分支下的代码并不会影响其他分支的内容

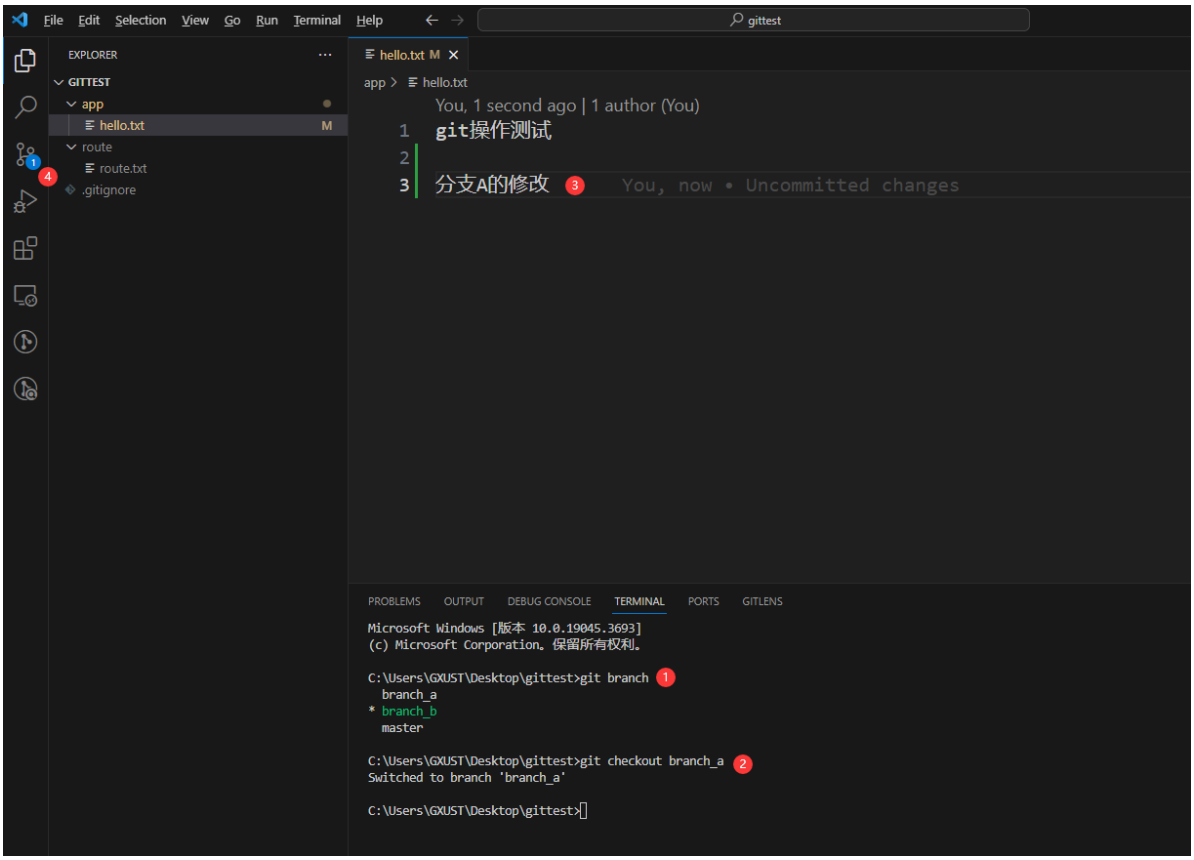
```
git checkout -b <新分支名称>

git checkout -b branch_a  # 新建即切换到该分支
C:\Users\GXUST\Desktop\gittest>git checkout -b branch_a
Switched to a new branch 'branch_a'

# 回到主分支(分支切换)
git checkout master

# 新建分支
C:\Users\GXUST\Desktop\gittest>git checkout -b branch_b
Switched to a new branch 'branch_b'
```

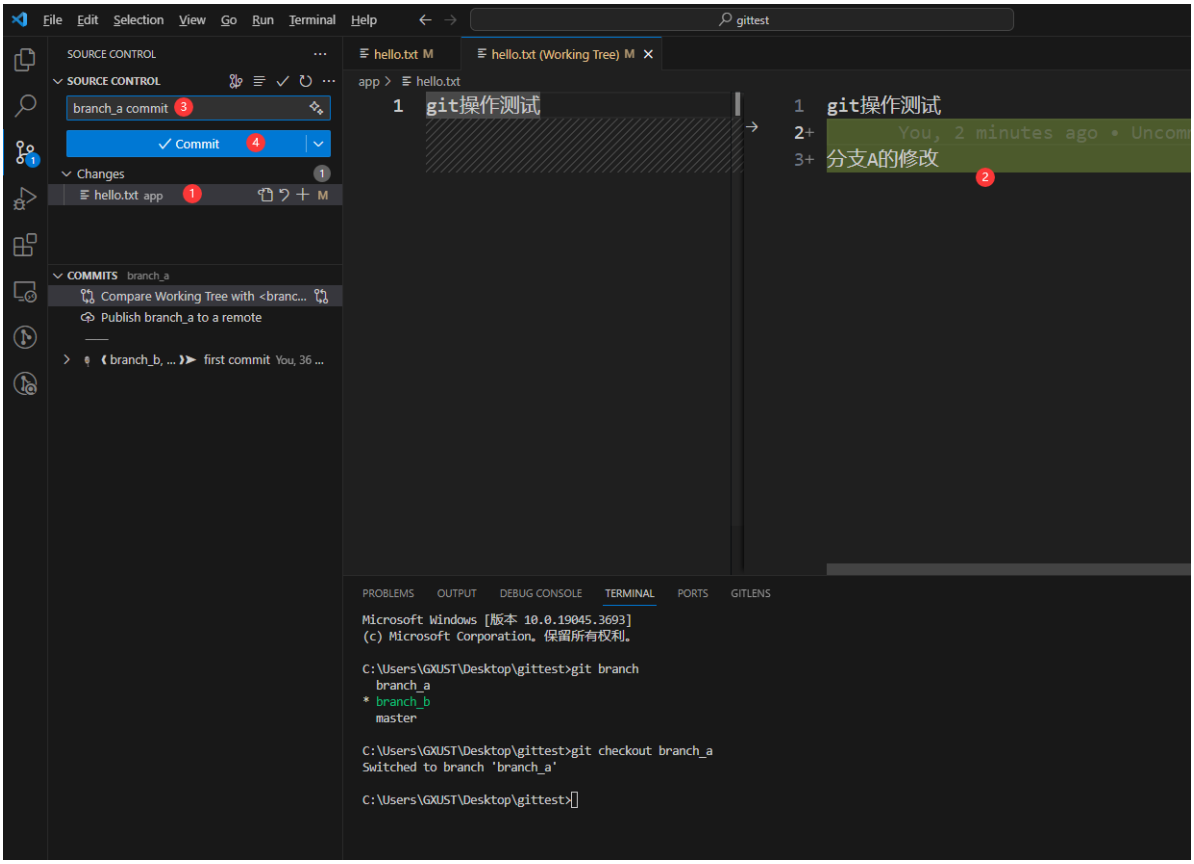
三、分支切换

```
git checkout <分支名称>
```

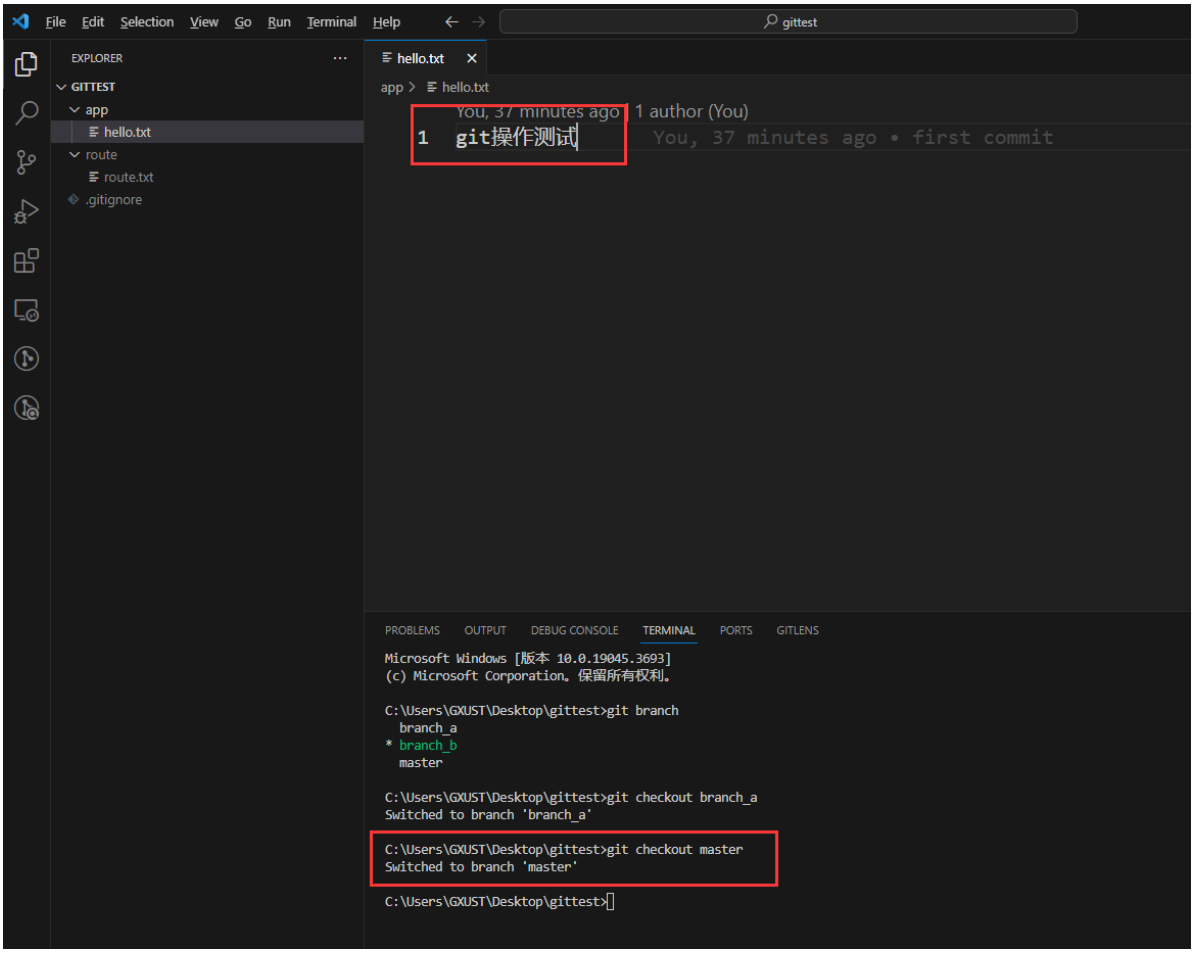
四、分支合并

分支合并是指将当前分支合并到上级分支，这里是合并到主分支上，因为层级只有两层；在分支上修改的内容，需要提交到暂存区后再与主分支合并（先提交分支修改，再到主分支下合并）

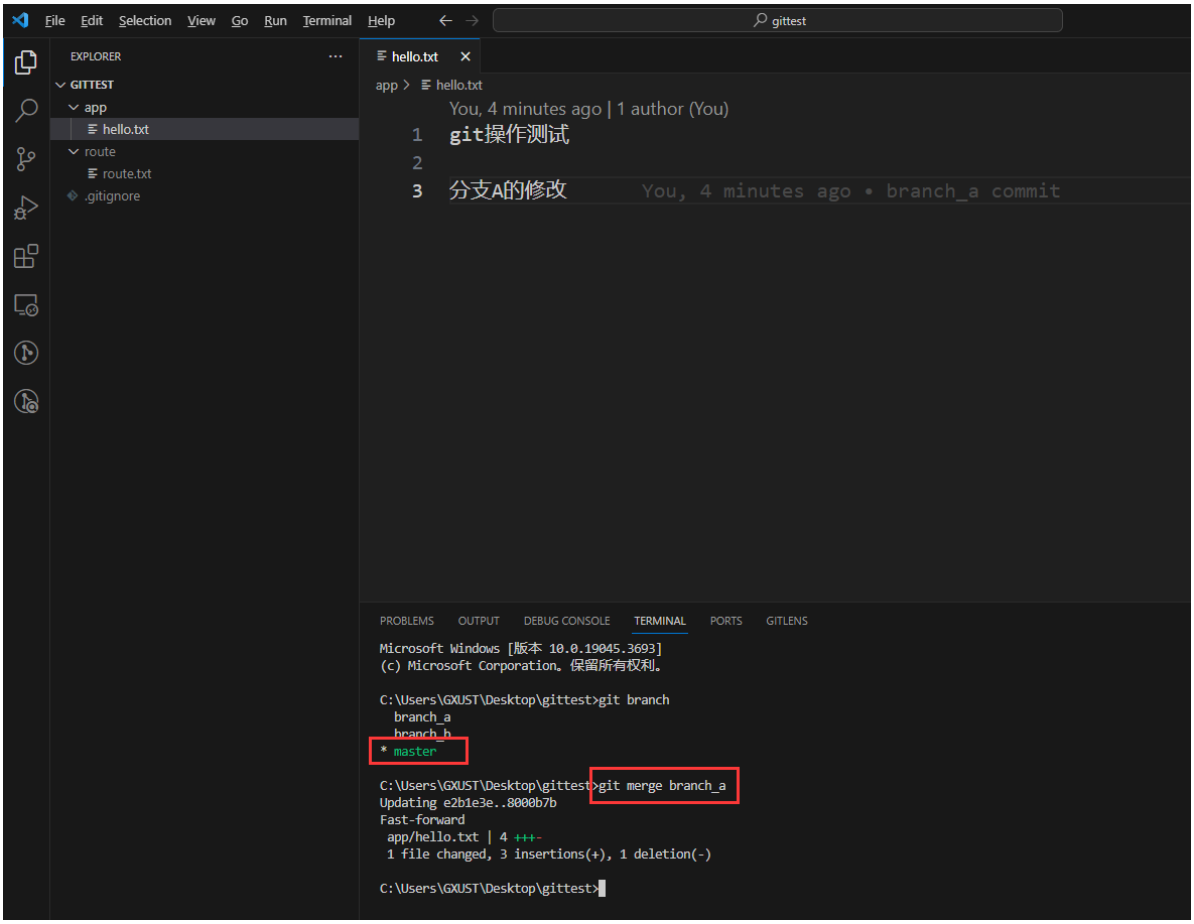




此时的branch_a分支已经完成修改提交，但这是的master分支还是不变的



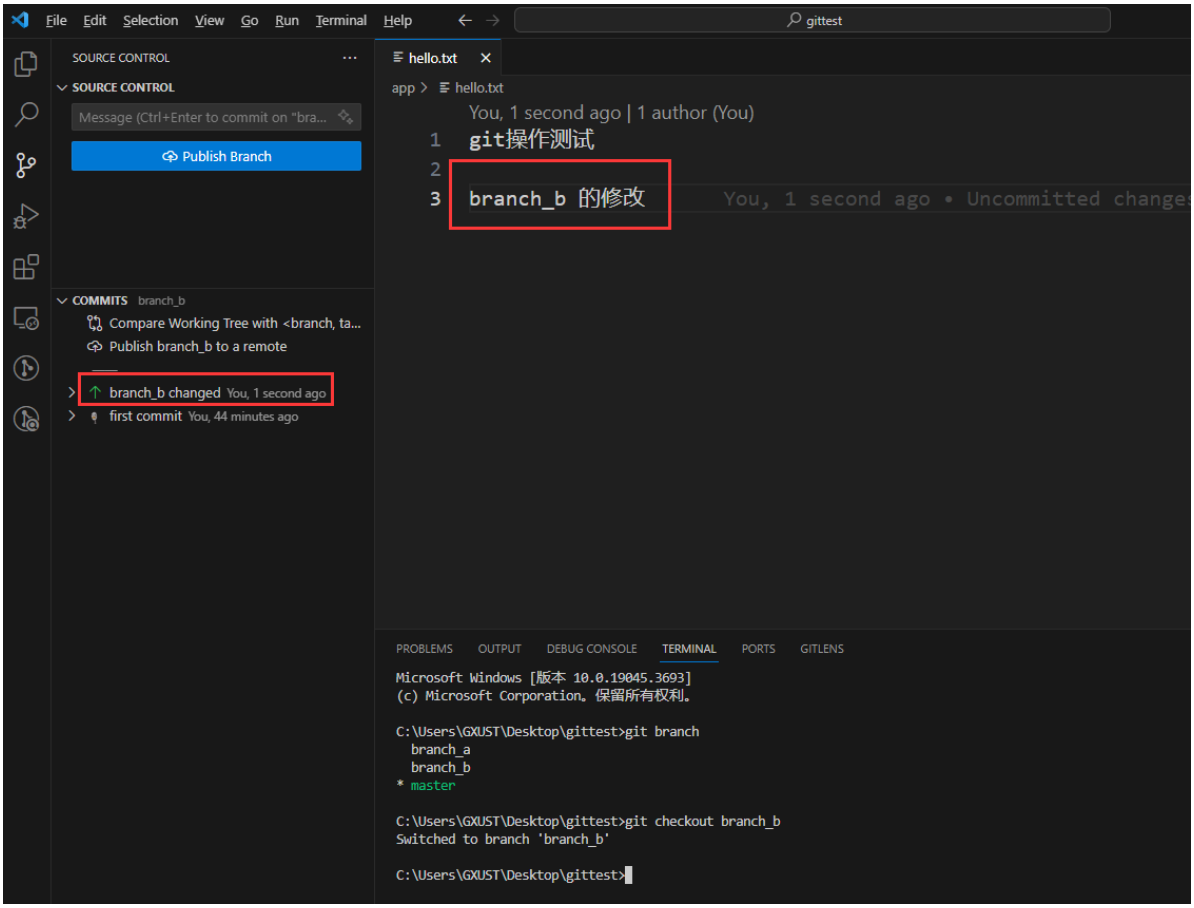
```
# 合并分支到主分支
git merge <分支名>

# 回到主分支下
git checkout master
git merge branch_a

C:\Users\GXUST\Desktop\gittest>git merge branch_a
Updating e2b1e3e..8000b7b
Fast-forward
 app/hello.txt | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)
```



五、合并冲突

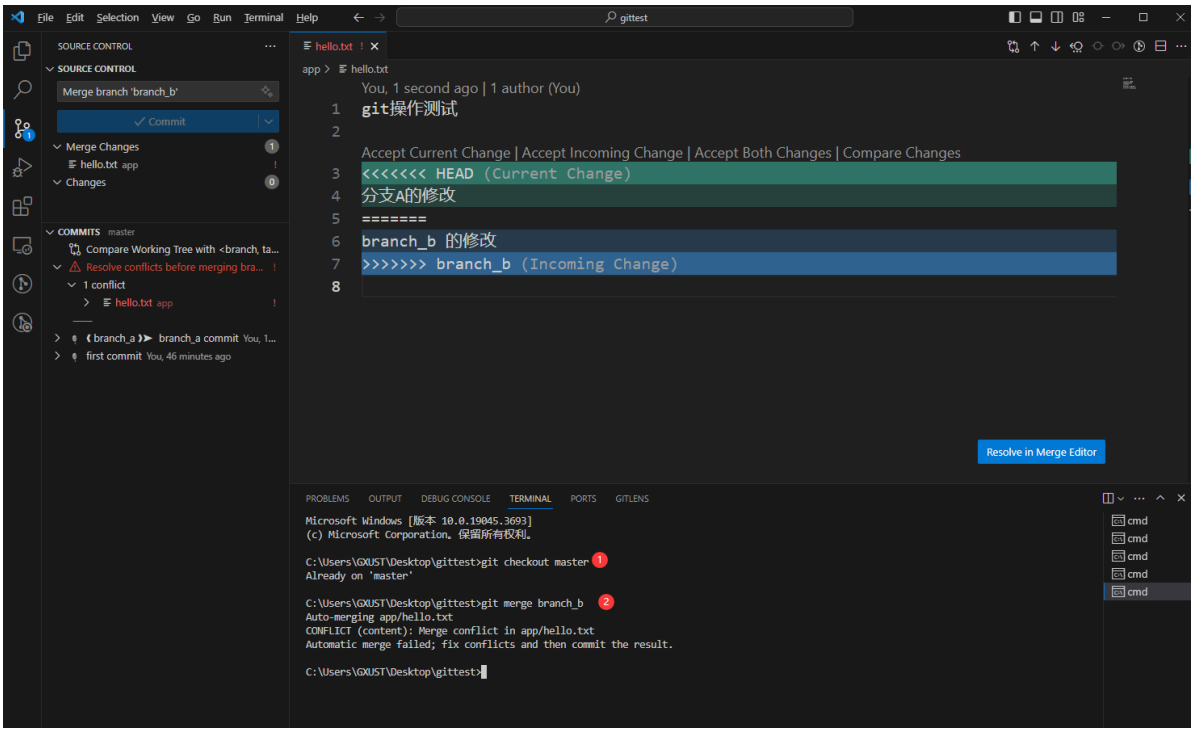
这种情况是 branch_a修改了hello.txt文件的前提下，branch_b也修改了该文件，在分支合并的时候git不知道要保留合并后那个分支的修改，这时候需要解决冲突的情况

分支b修改

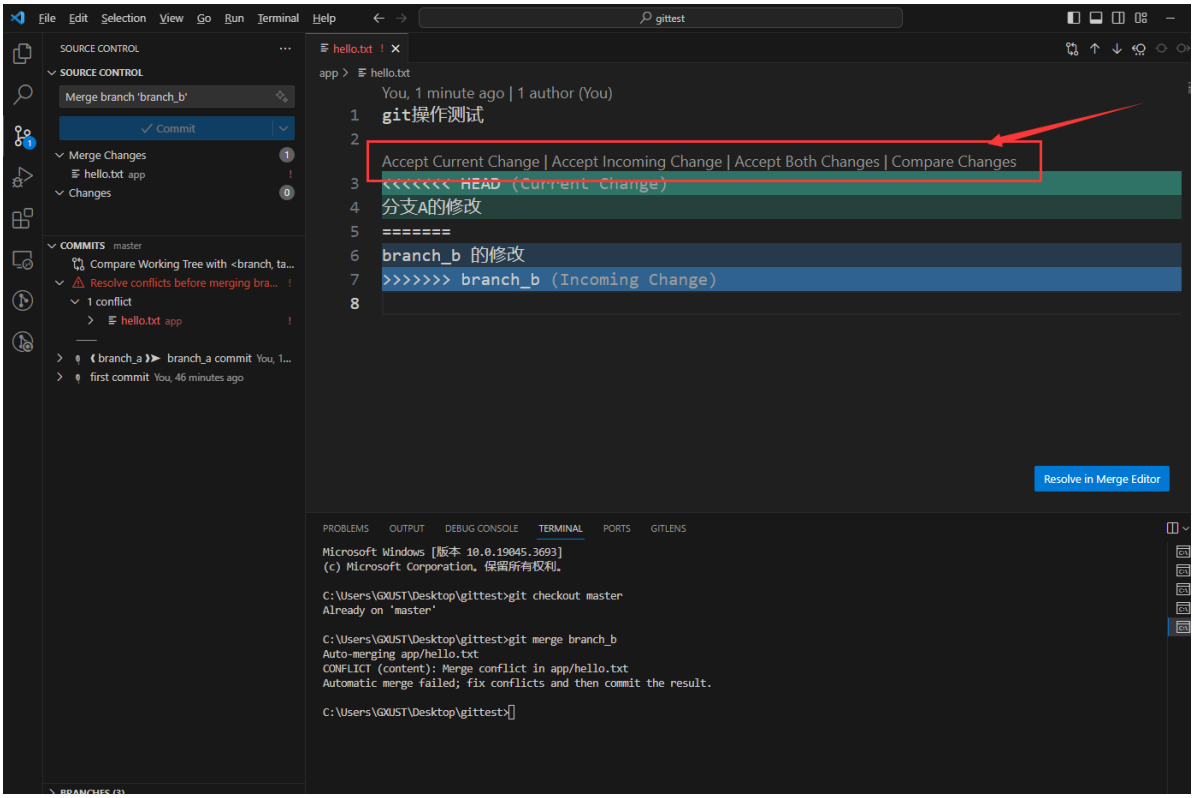


分支b合并，提示合并冲突

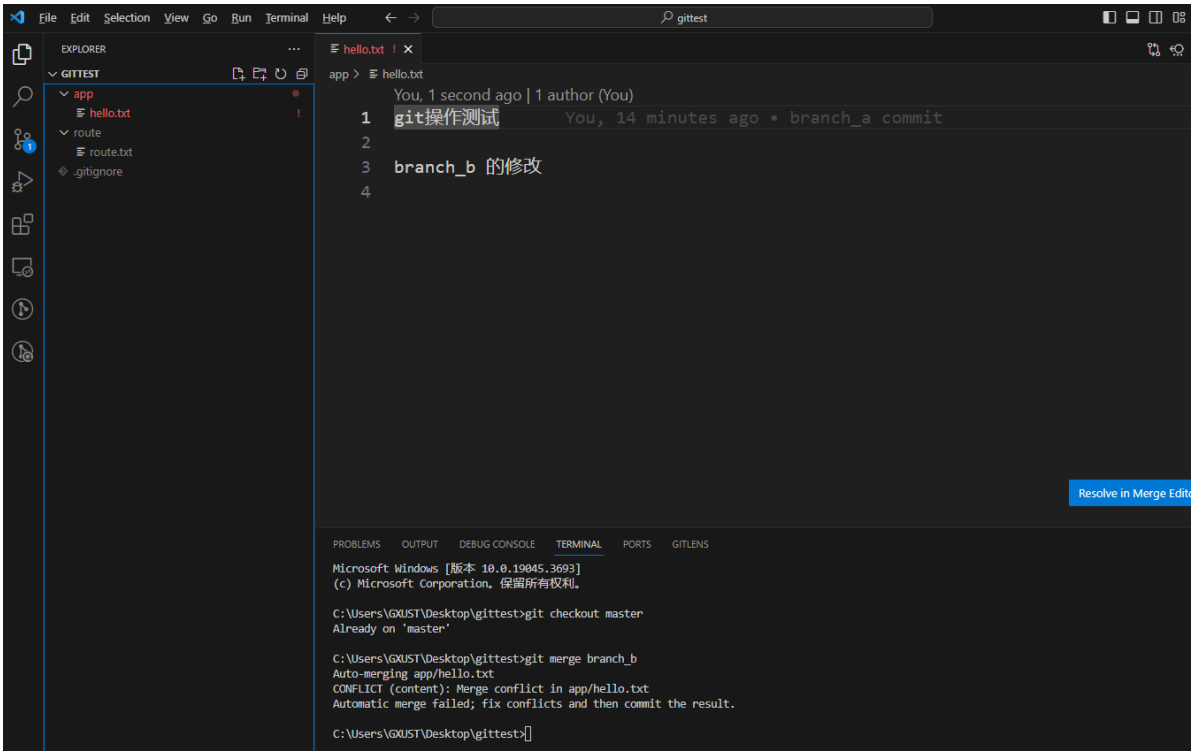
```
git checkout master
git merge branch_b
```



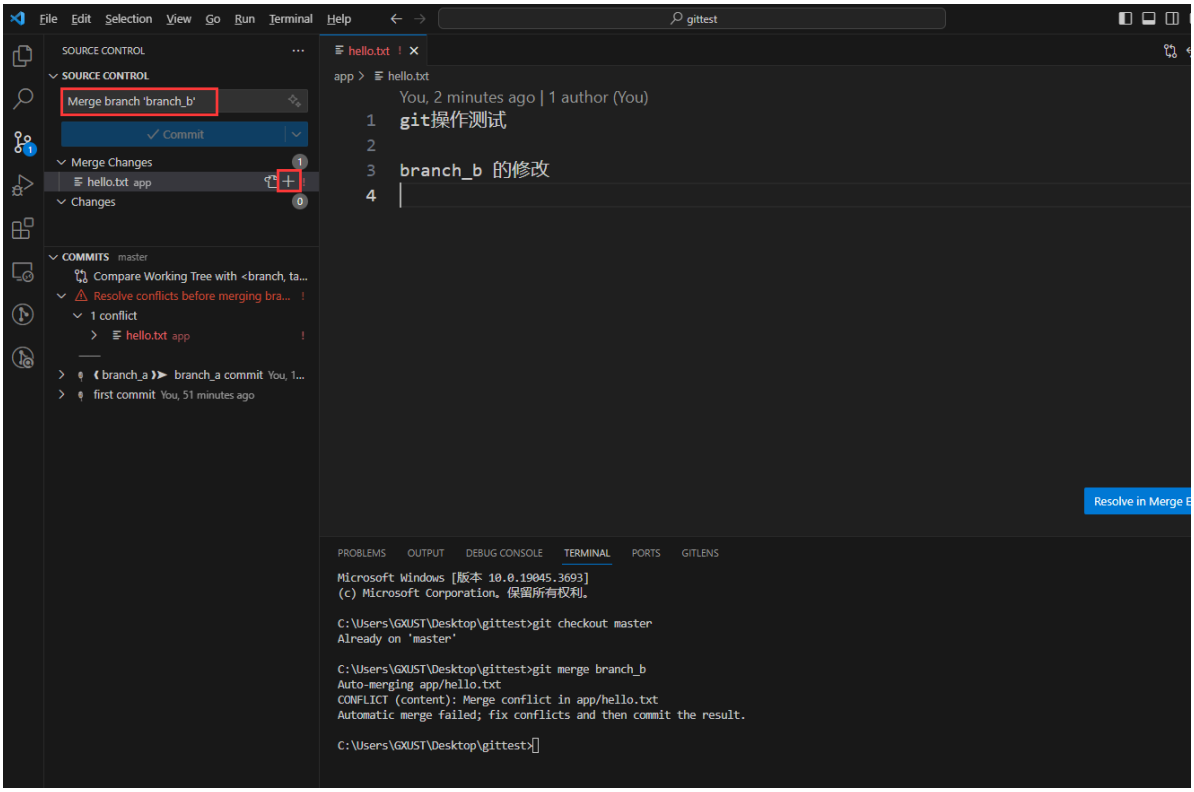
这时候可以直接选择要合并保留的分支版本

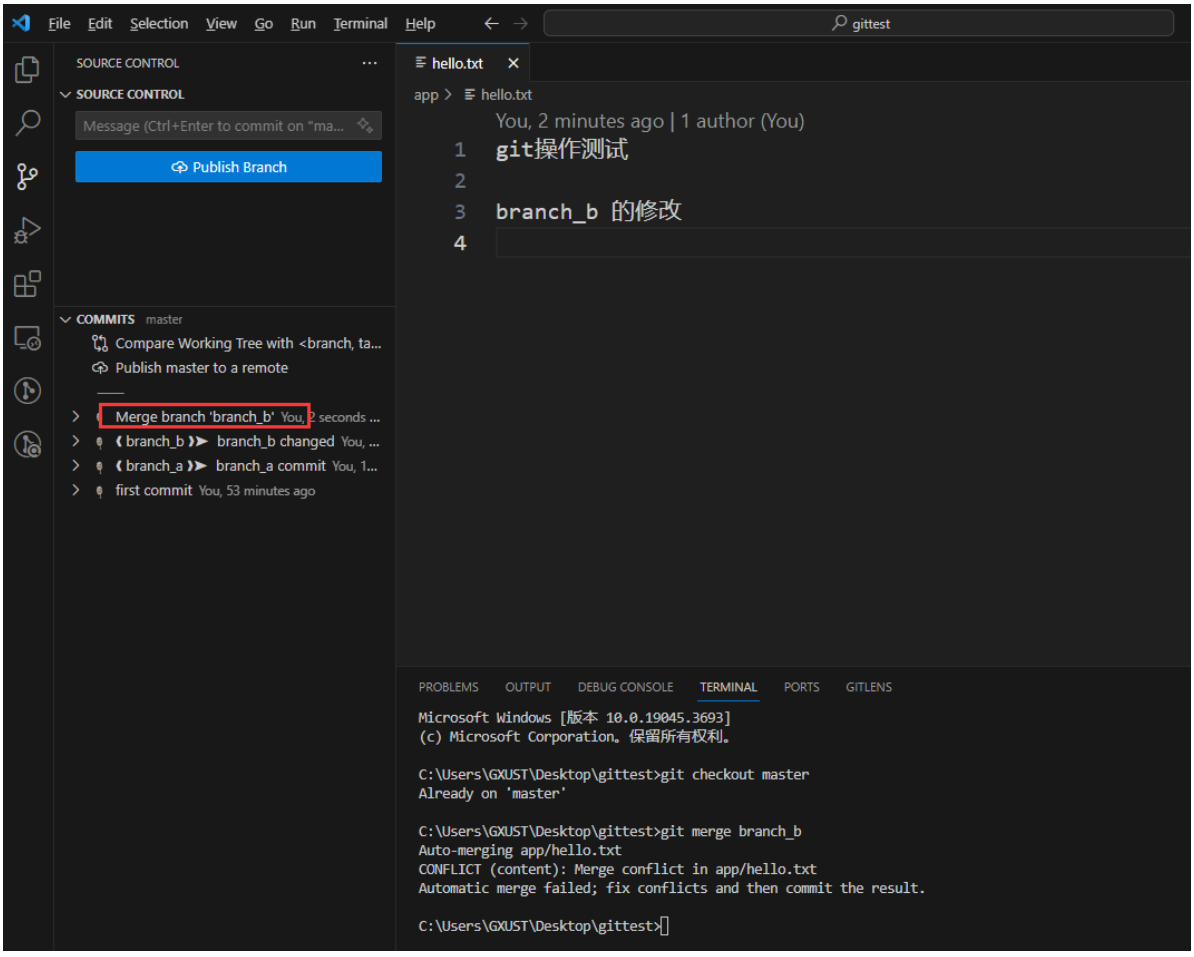



```
Accept Current Change: 保留先合并的版本
Accept Incoming Change: 保留后来合并的版本
Accept Incoming Changes: 保留两个合并的更改
```



解决冲突后需要将冲突合并完后的主分支添加到合并修改(merge changes)，然后再添加提交描述，提交到暂存区



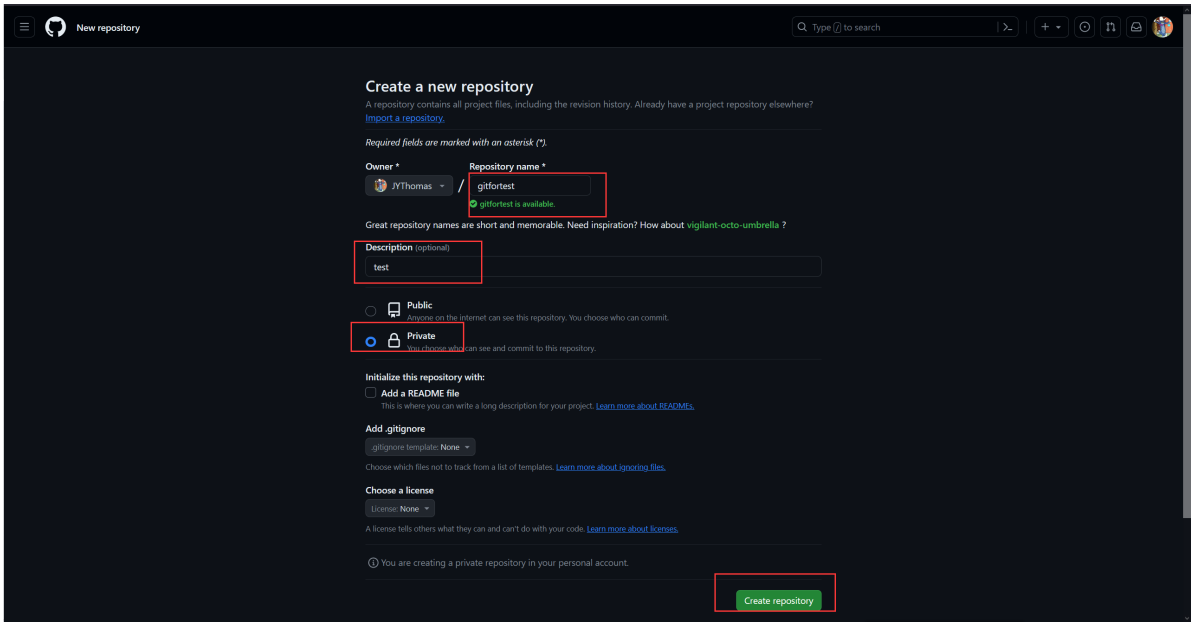


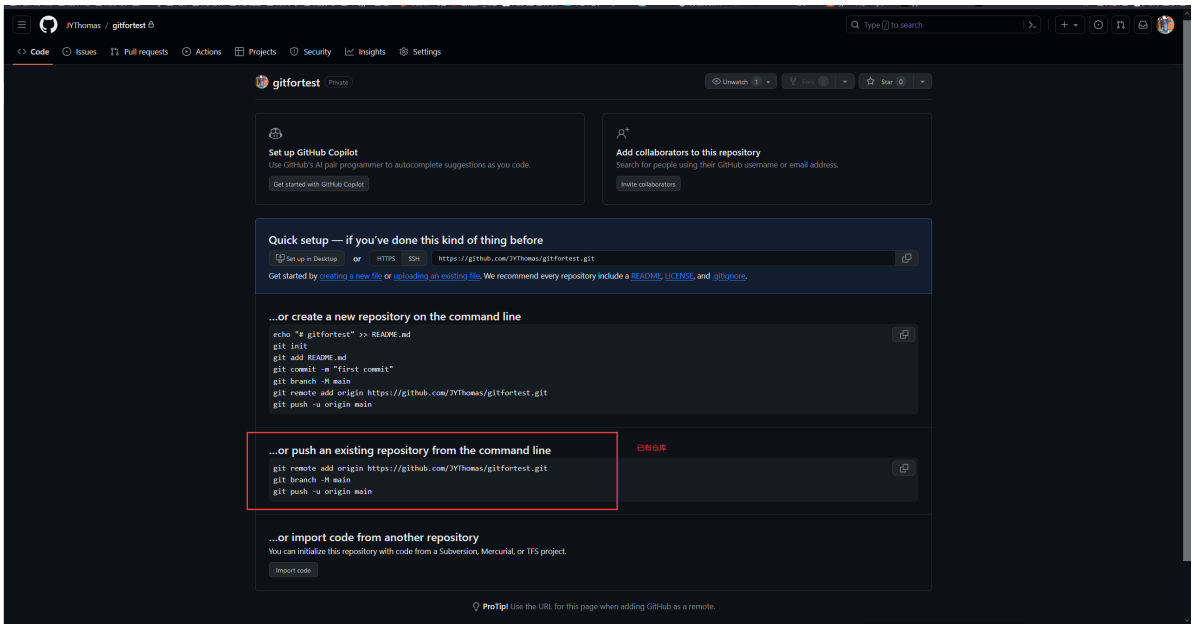
六、删除分支

```
# 首先切换到主分支下
git checkout master
# 删除分支
git branch -D <分支名称>
```

本地仓库同步到远程仓库

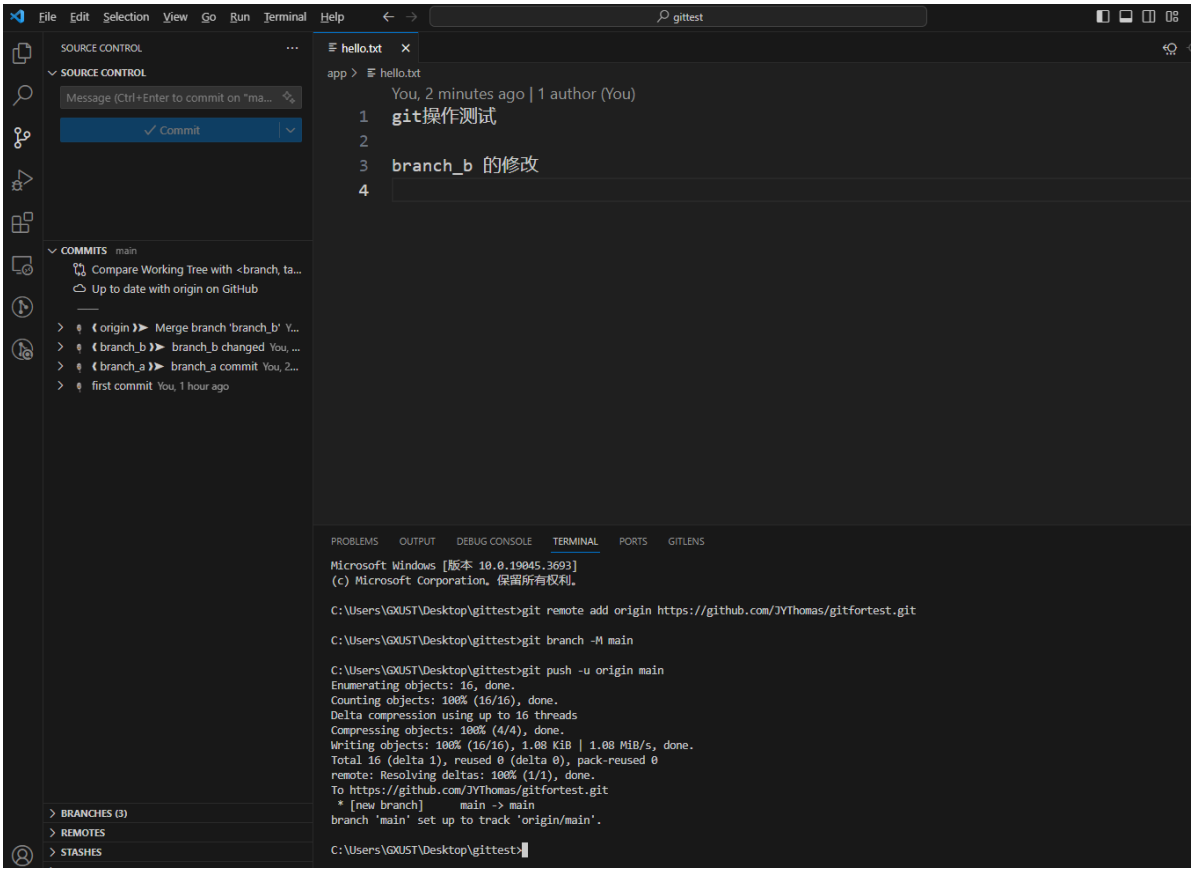
一、创建远程仓库





二、本地仓库推送到远程仓库

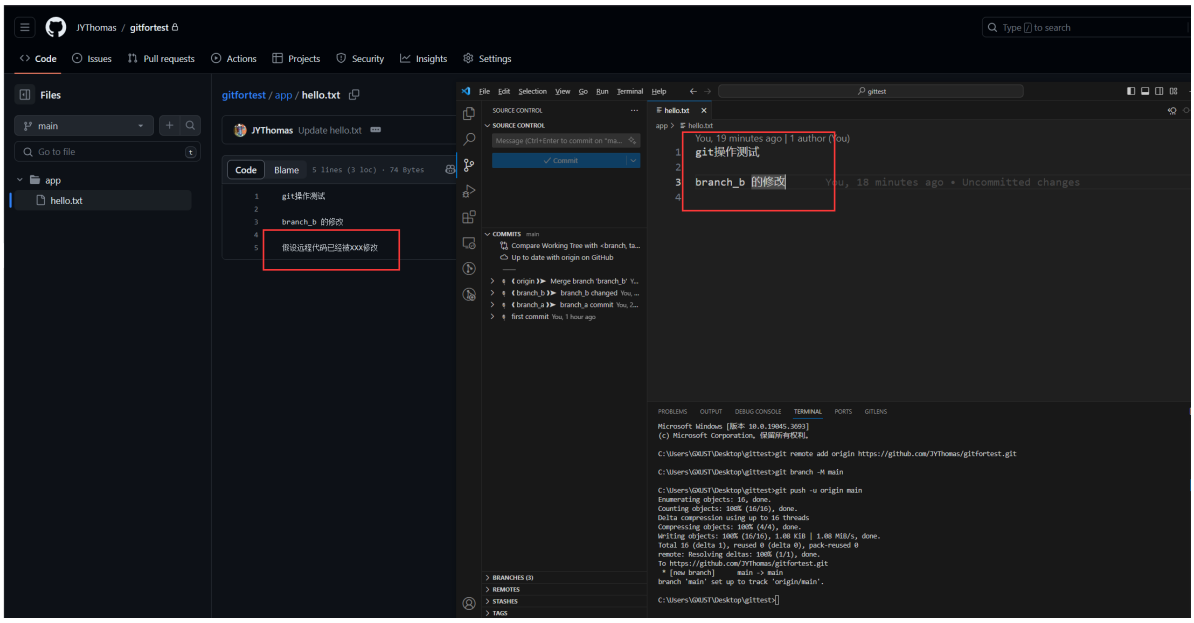
```
# 选定要推送的远程仓库
git remote add origin https://github.com/JYThomas/gitfortest.git
# 远程仓库切换到main分支
git branch -M main
# 将本地暂存区的主分支提交到远程仓库
git push -u origin main
```



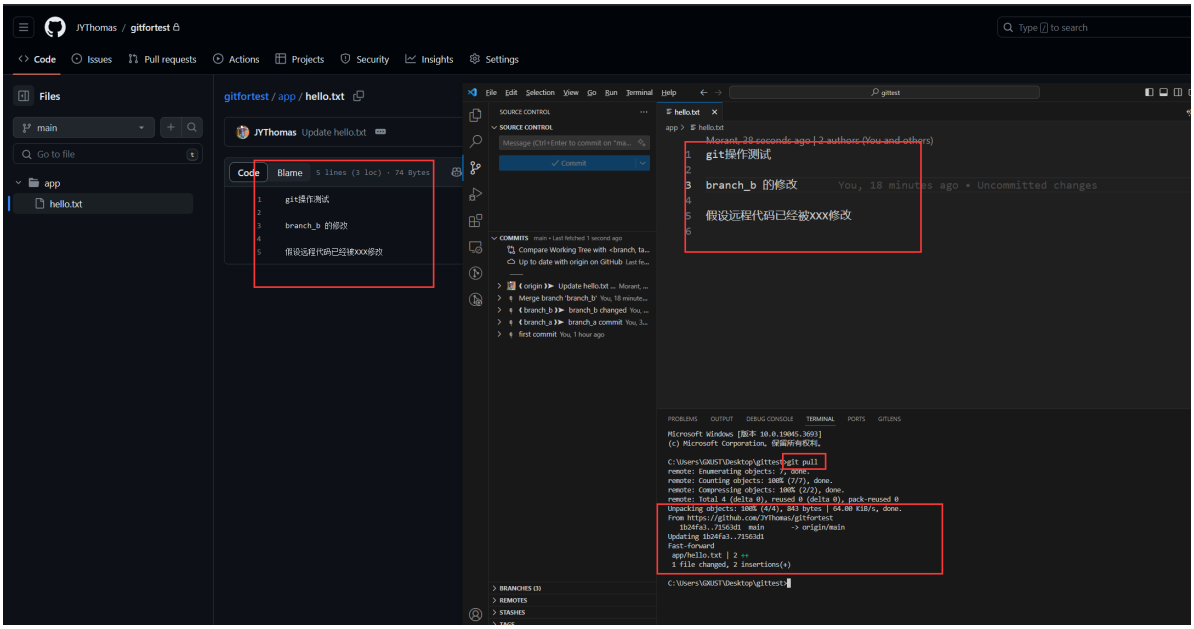
推送的时候可能会出现推送不成功 timeout的情况是因为需要设置git代理服务器，参考：https://www.cnblogs.com/chig/p/17560850.html

推送完成之后即可看到本地仓库推送到远程仓库中，并且被.gitignore的文件不会被推送到远程仓库

三、本地仓库同步远程仓库(拉取远程仓库)



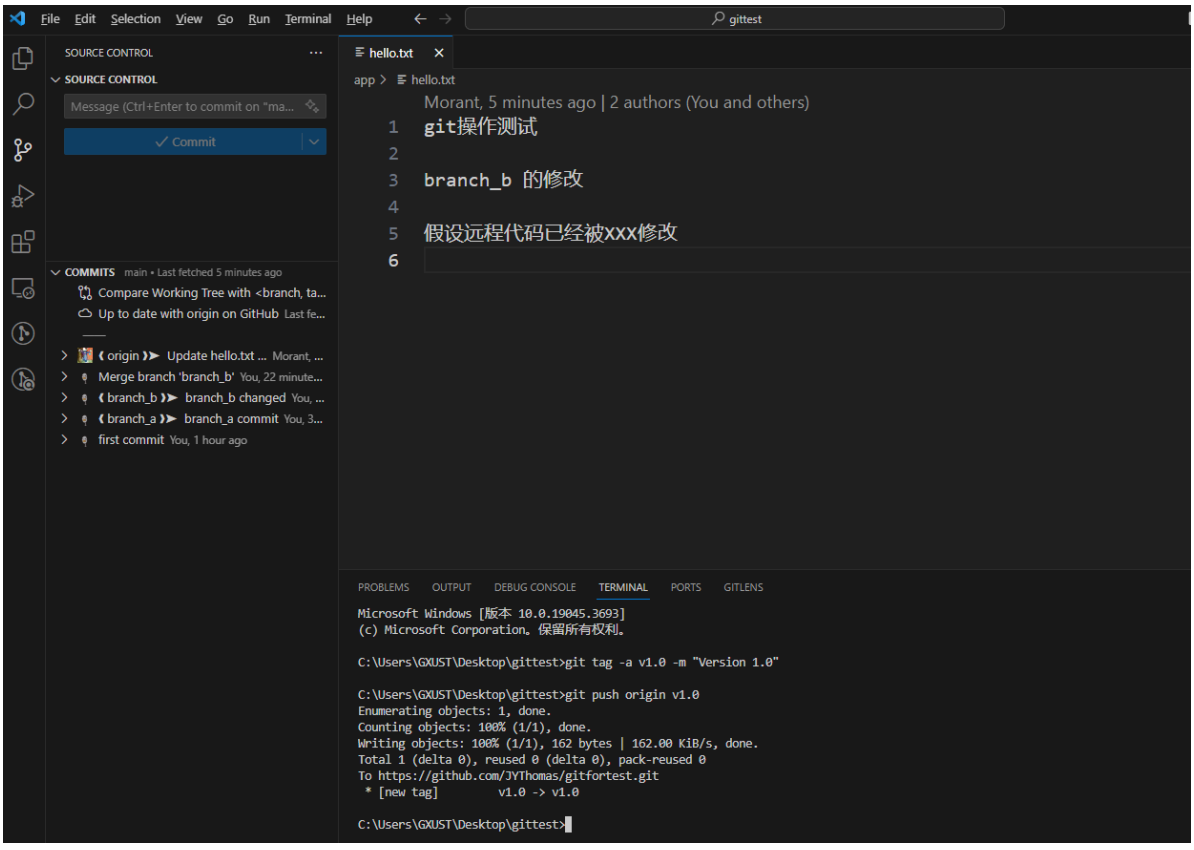

```
# 在进行代码修改前，首先要拉取最新一版的远程仓库代码再进行修改
git pull
```

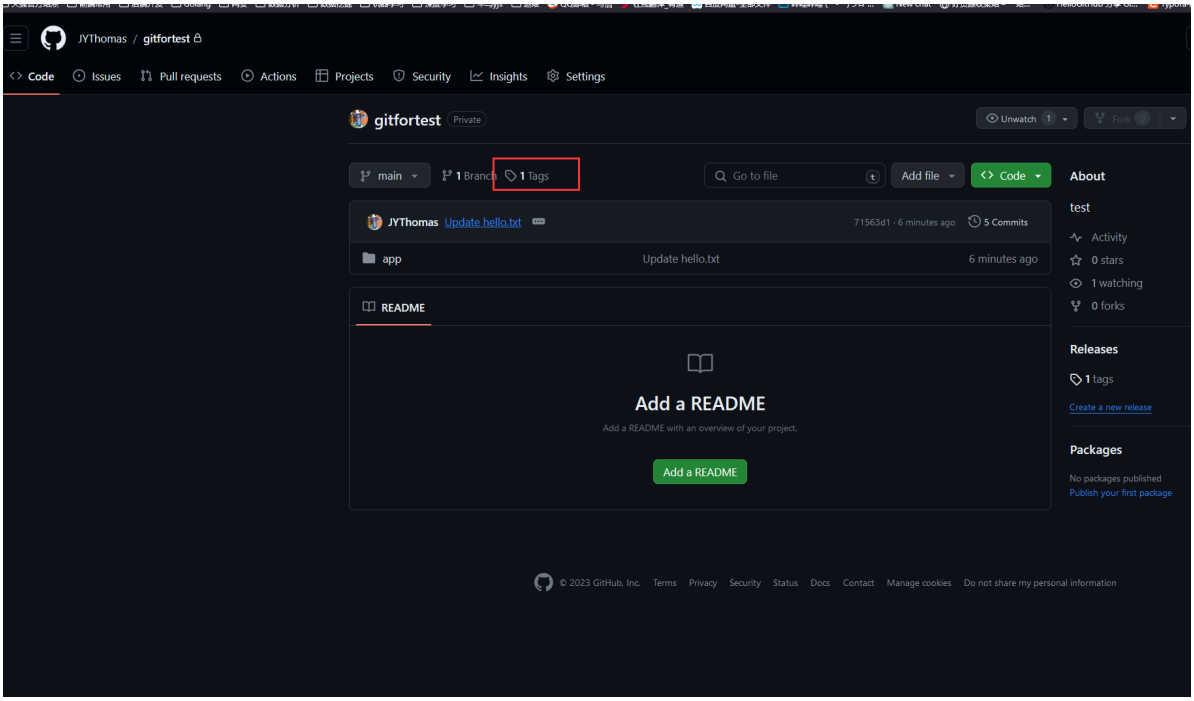


四、版本迭代

1.在软件开发中，每次发布一个新版本时，可以创建一个标签，以便轻松地查找和恢复该版本的代码。

```
git tag -a v1.0 -m "Version 1.0"

# 推送标签到远程
git push origin v1.0
```





2.**回退到特定版本：** 如果需要回退到以前的某个版本，可以使用标签来指定版本号。（**如果代码已经更改，可以通过tag标签回退到某个版本状态的代码**）

```
# 切换到标签对应的版本
git checkout v1.0
```

注意，使用标签切换会处于“分离头指针”状态，这意味着你不能在该状态下直接进行提交。如果需要在某个特定版本上工作，可以创建一个新的分支：**在回退版本的分支上进行代码更新修改，在标签上是无法修改代码的**

```
# 创建并切换到新分支
git checkout -b new_branch v1.0
```

五、代码合并流程

在添加新功能时 新建对应的分支，在完成所有功能之后，新建一个release分支，然后将新功能分支合并到release分支上，然后将release分支提交推送到远程仓库中，最后再远程仓库的PR中将release分支合并到主分支。参考：

https://www.bilibili.com/video/BV1V24y1W7qo/?vd_source=bc3fd2b42eb94a414c6746120ff5ddc6