Fork/github操作手冊

**目錄**

[**相關檔案連結** 3](#_Toc126536931)

[練習操作的專案（可以隨意玩沒關係） 3](#_Toc126536932)

[**從github將專案連線到Fork** 3](#_Toc126536933)

[SSH方法 4](#_Toc126536934)

[HTTPS方法 9](#_Toc126536935)

[**新建專案** 11](#_Toc126536936)

[**加入現有的專案** 13](#_Toc126536937)

[公開專案 13](#_Toc126536938)

[加入私有的專案-專案負責人 17](#_Toc126536939)

[加入私有專案-組員 18](#_Toc126536940)

[**加入分支-共用端和本地端** 23](#_Toc126536941)

[加入共用端分支 24](#_Toc126536942)

[**Fork相關操作** 29](#_Toc126536943)

[本地端 30](#_Toc126536944)

[Fetch、Pull、push、stash 30](#_Toc126536945)

[Commit (將檔案存到個人端空間) 38](#_Toc126536946)

[Revert(退回commit) 39](#_Toc126536947)

[**合併檔案操作** 47](#_Toc126536948)

[專案負責人(審核者) 47](#_Toc126536949)

[審核Merge 47](#_Toc126536950)

[開發人員 55](#_Toc126536951)

[發Merge 55](#_Toc126536952)

[解衝突 61](#_Toc126536953)

# 相關檔案連結

Github研習講義：  
<https://hackmd.io/bIA6KvZMRFOXYBdI9dA9mA>

下載Fork：

<https://git-fork.com/>

## 練習操作的專案（可以隨意玩沒關係）

持有人都是10946002，練習發MR要說一聲，我才能上去審核。

公開專案：

**Dynasty20221209**

私有專案：(要加入需給github的名稱)

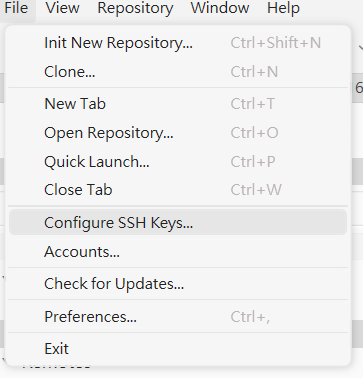
**private-test1216**

# 從github將專案連線到Fork

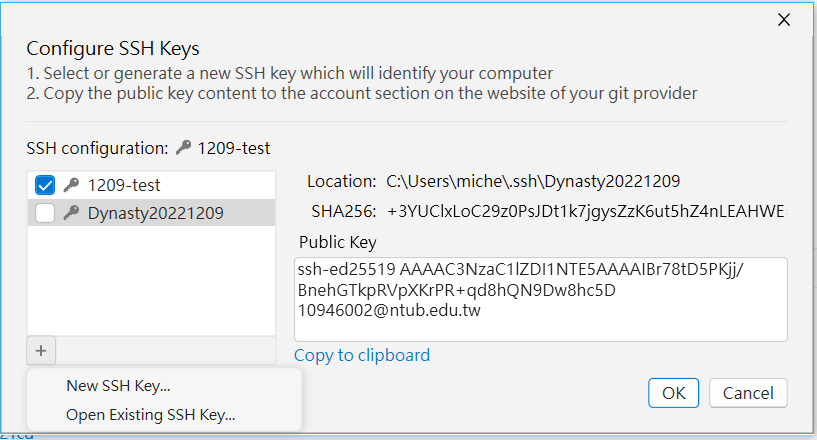
將github的東西放到Fork有兩種一個是用HTTPS一個用SSH，差別在於使用HTTP需要登入github密碼，SSH需要設置金鑰，但是只能用在Public的專案(僅須設定一次即可)，這邊兩個都會講解，**本操作手冊後面的操作都使用HTTPS的方式。**

## SSH方法

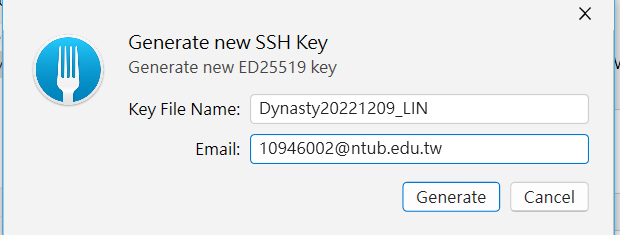
到Fork裡面，選Configure SSH Keys



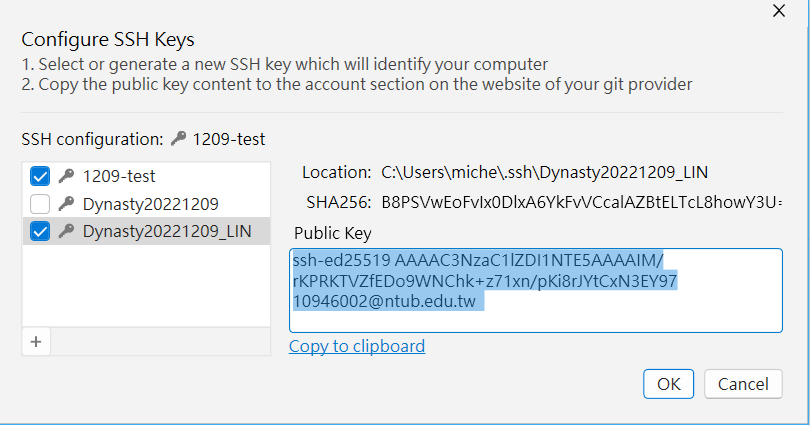
新增SSH Key



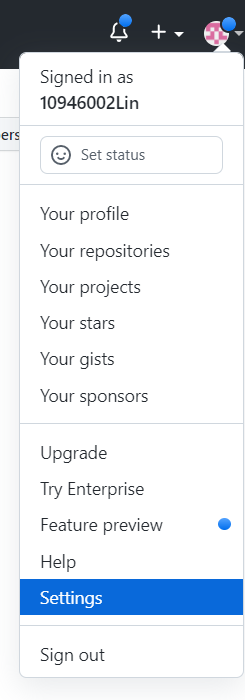
輸入金鑰名稱和信箱



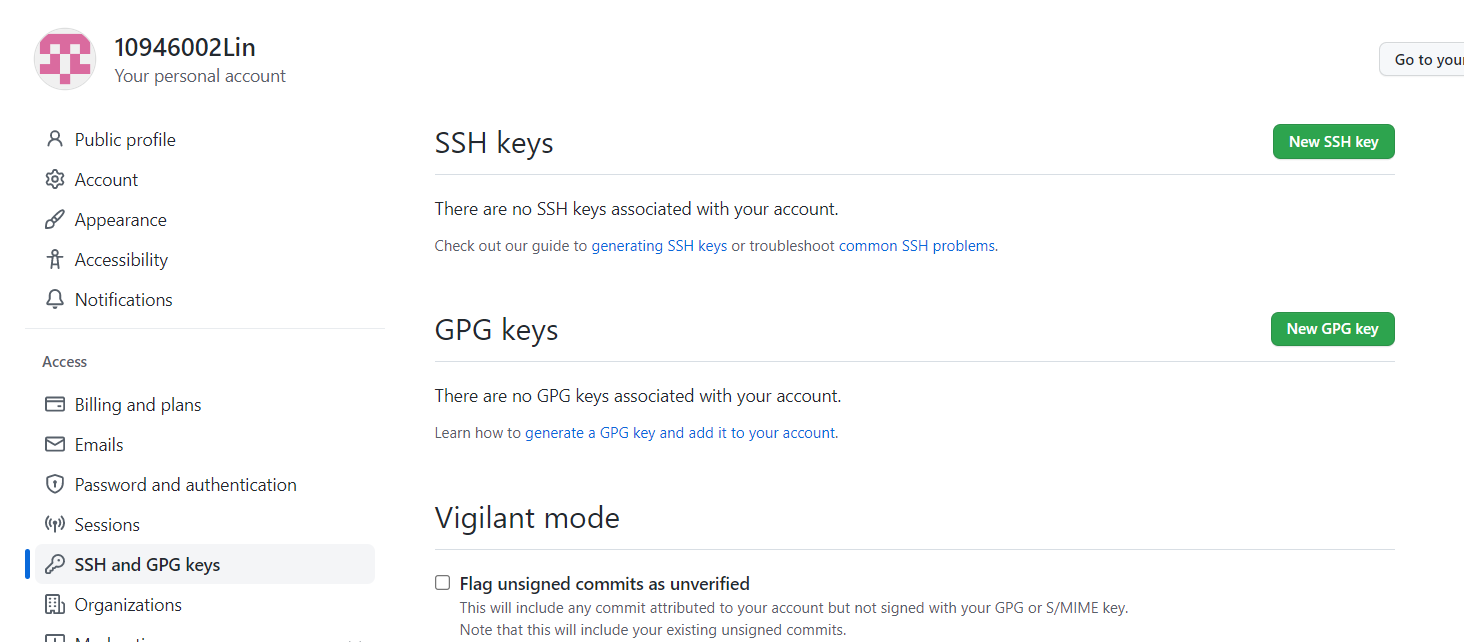
複製剛剛建立的金鑰(按copy也可以)



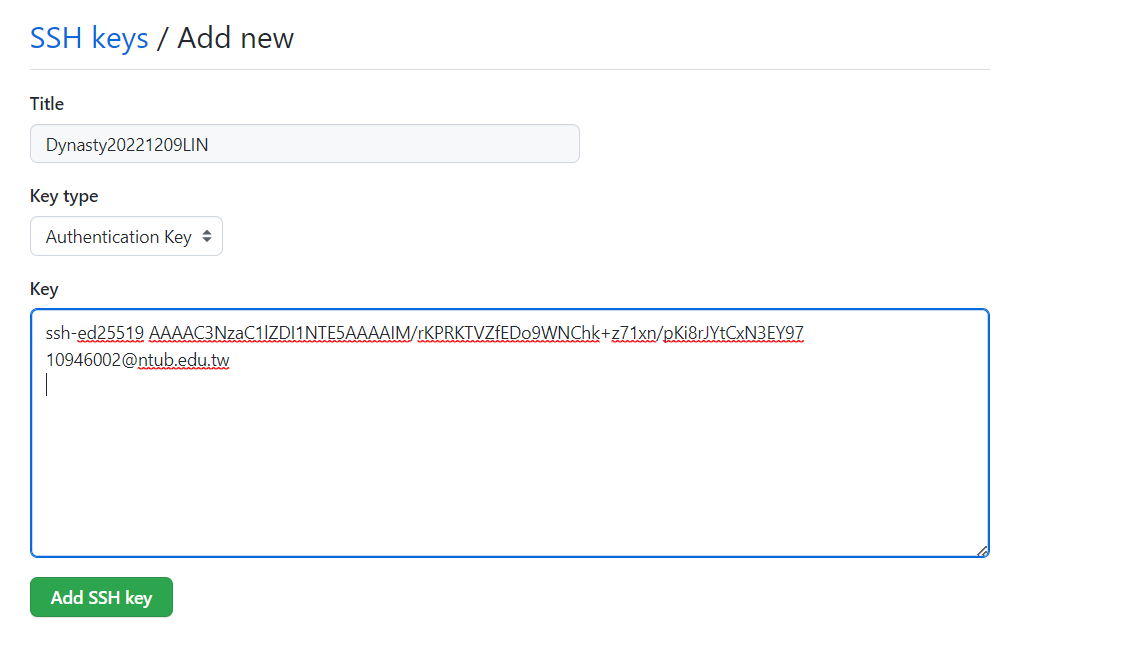
回到github，找到Setting



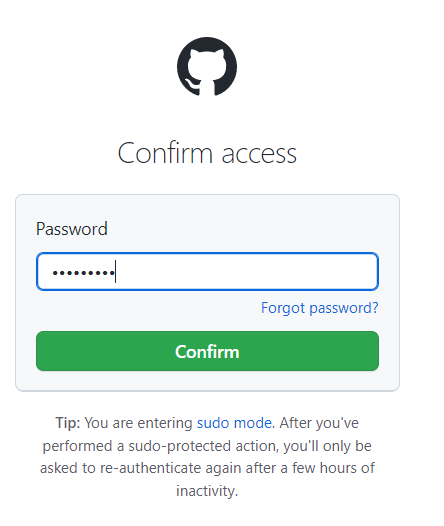
SSH and GPG Keys頁面，建立鑰匙



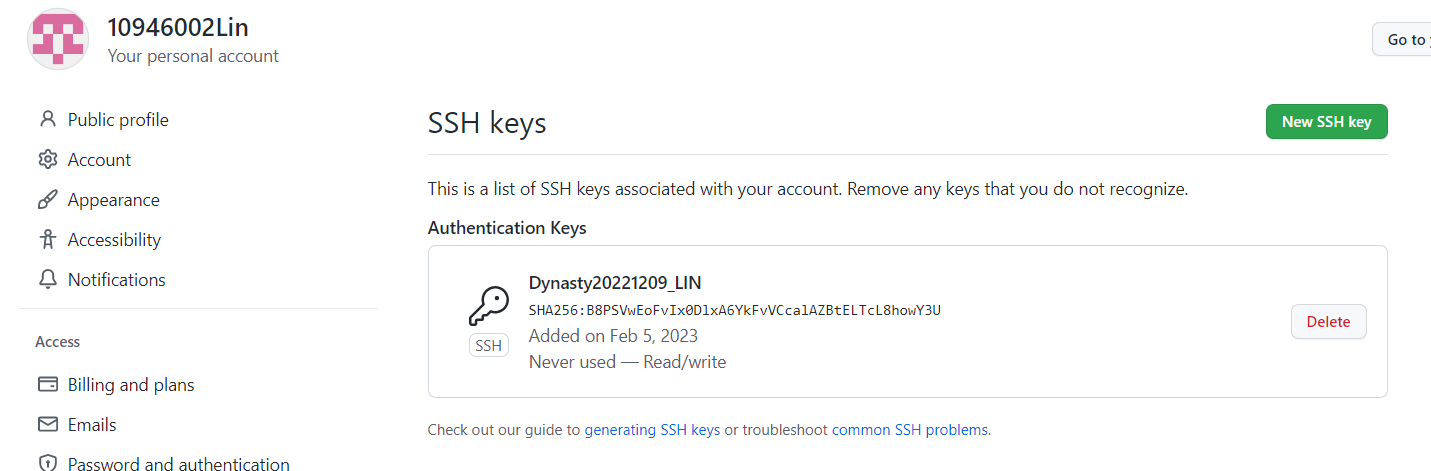
打上鑰匙的名稱(和專案名稱一樣較好辨別)，貼上剛剛複製的金鑰



會要求輸入密碼

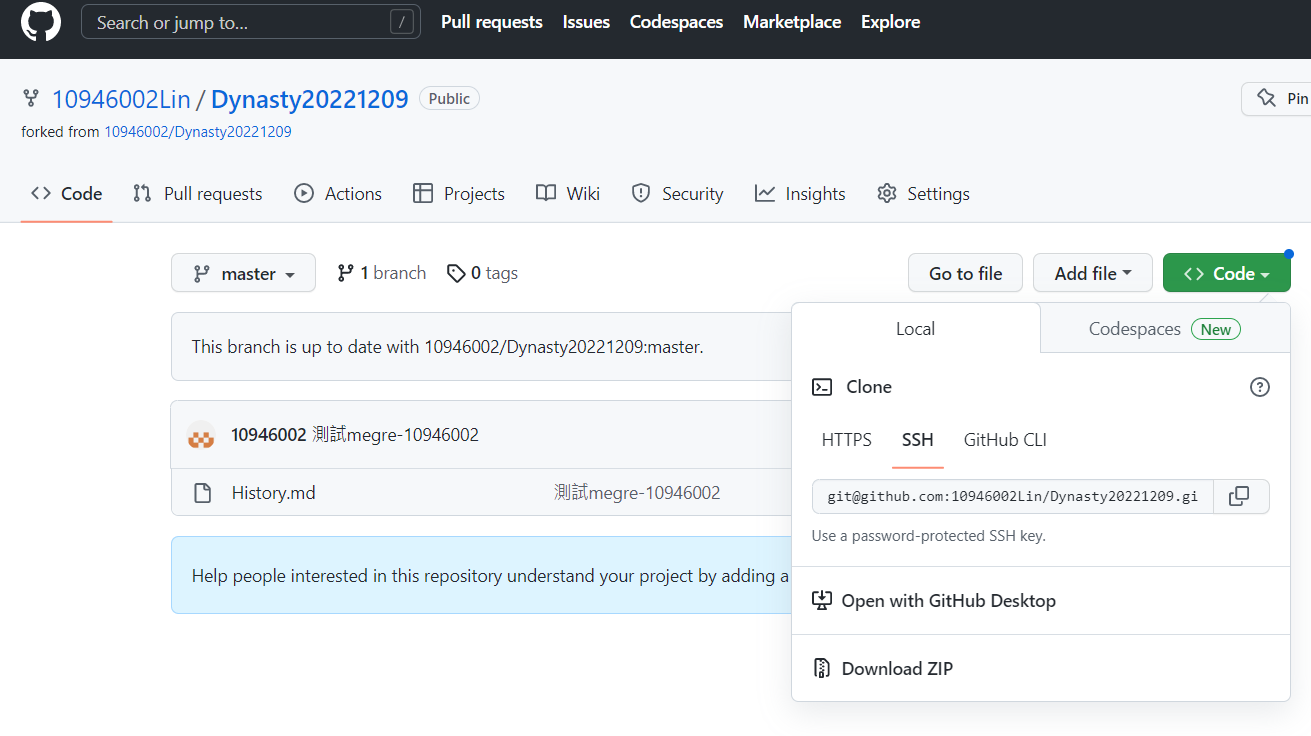


完成畫面

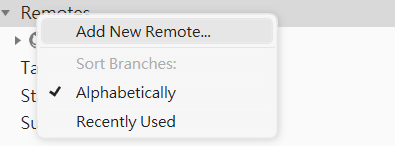


**使用SSH**

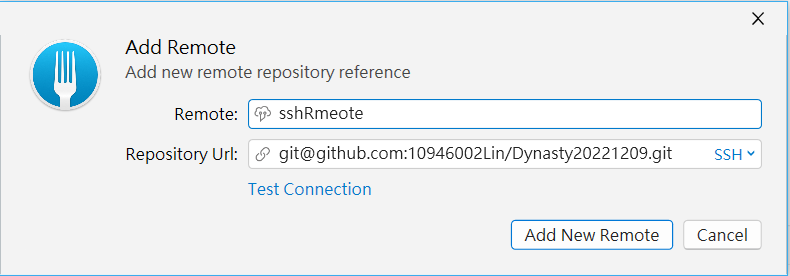
到github裡面Code，要選SSH，複製該連結



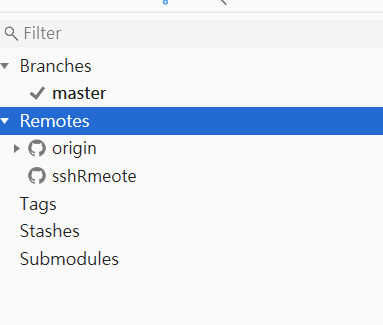
到Fork裡面新建分支



幫分支取名字，貼上剛剛的連結，後方會顯示目前這個是SSH，可以按下Test測試有沒有問題，好了按下Add



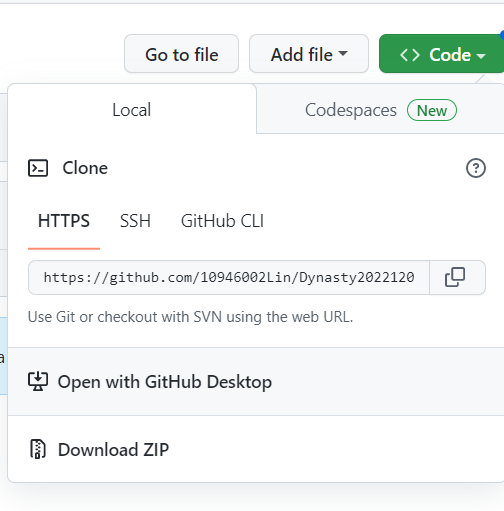
新的分支就建立好了



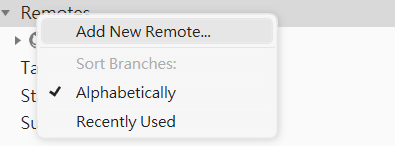
## HTTPS方法

此方法不論是公有還是私有專案都能連

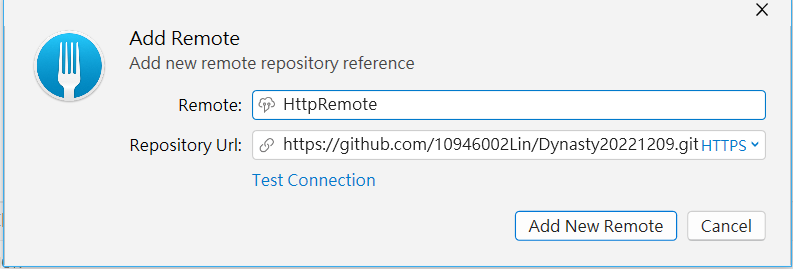
選HTTPS，複製連結



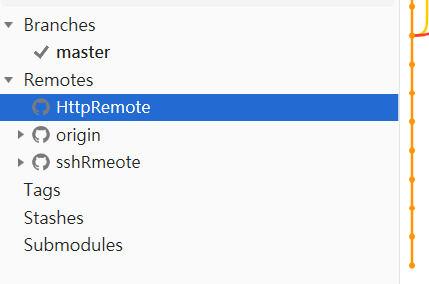
到Fork新建分支



打上分支名稱，將複製的連結貼上去，可以按Test測試有無錯誤，好了後新增

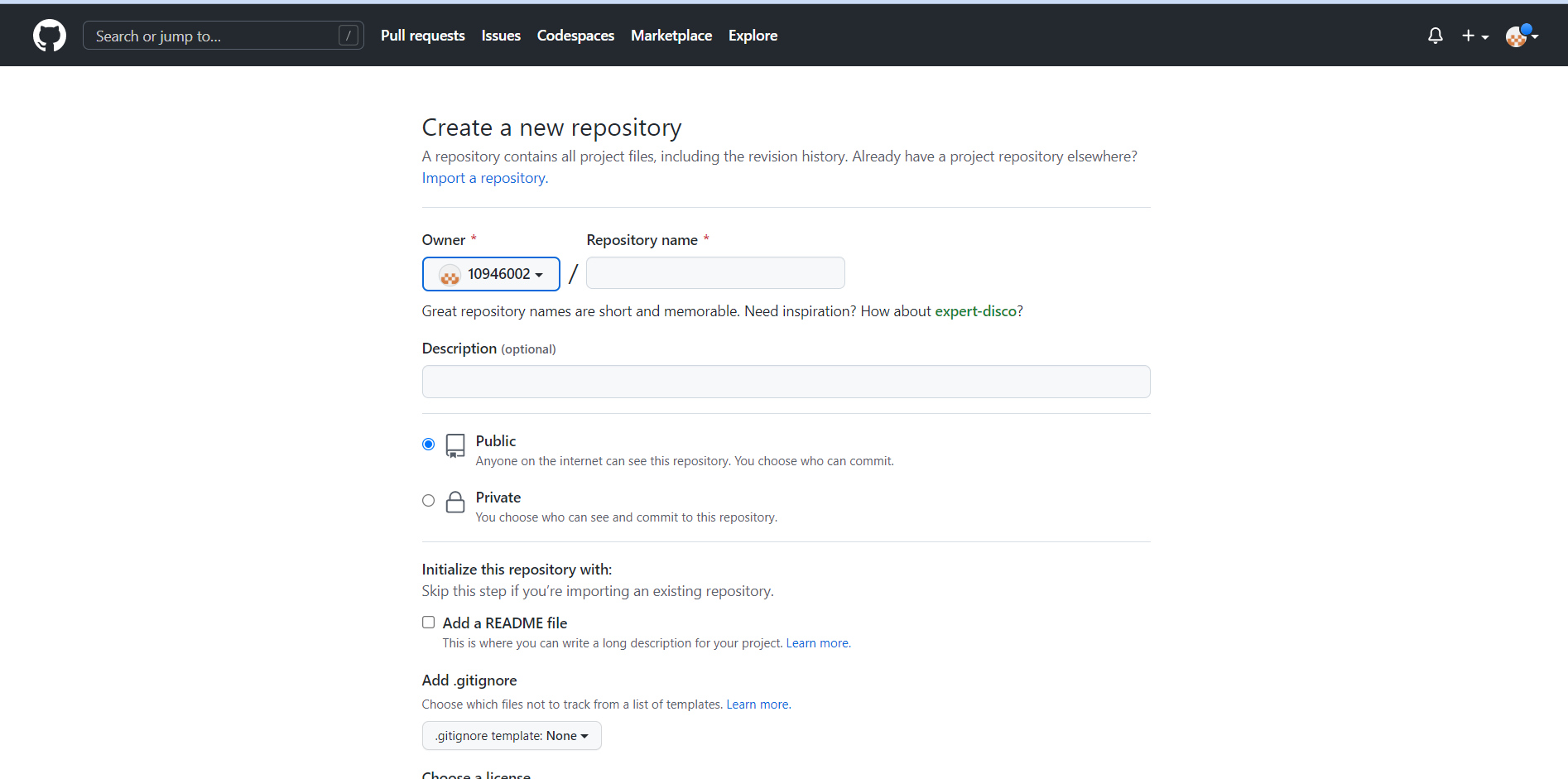


完成畫面

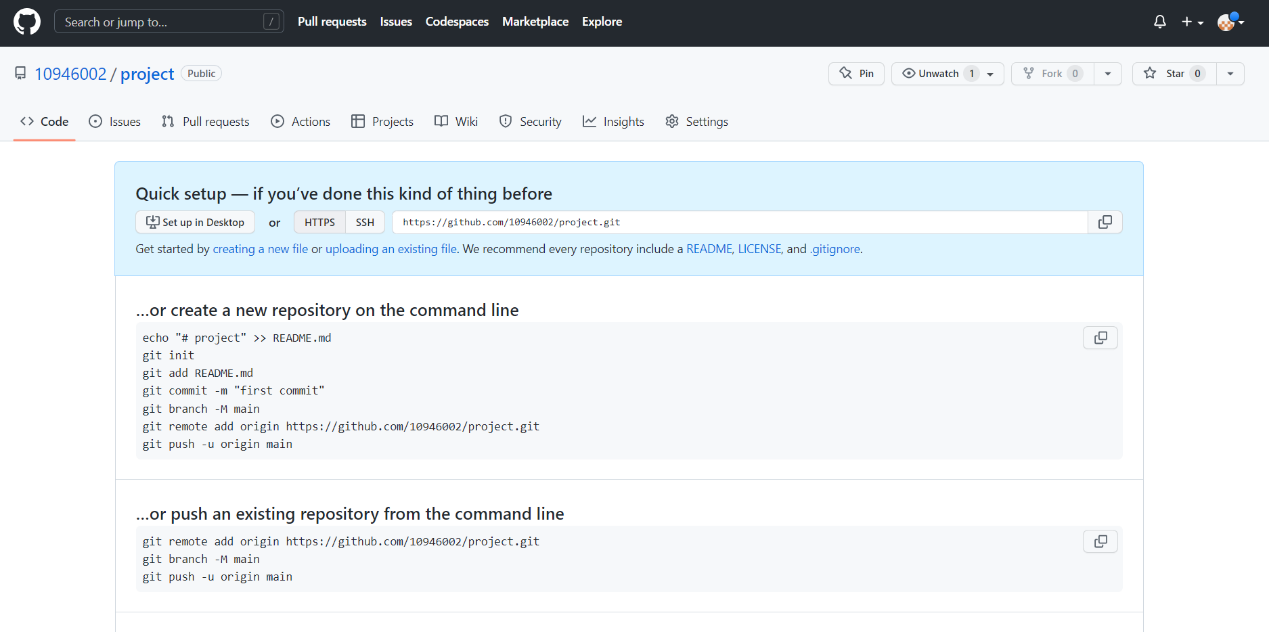


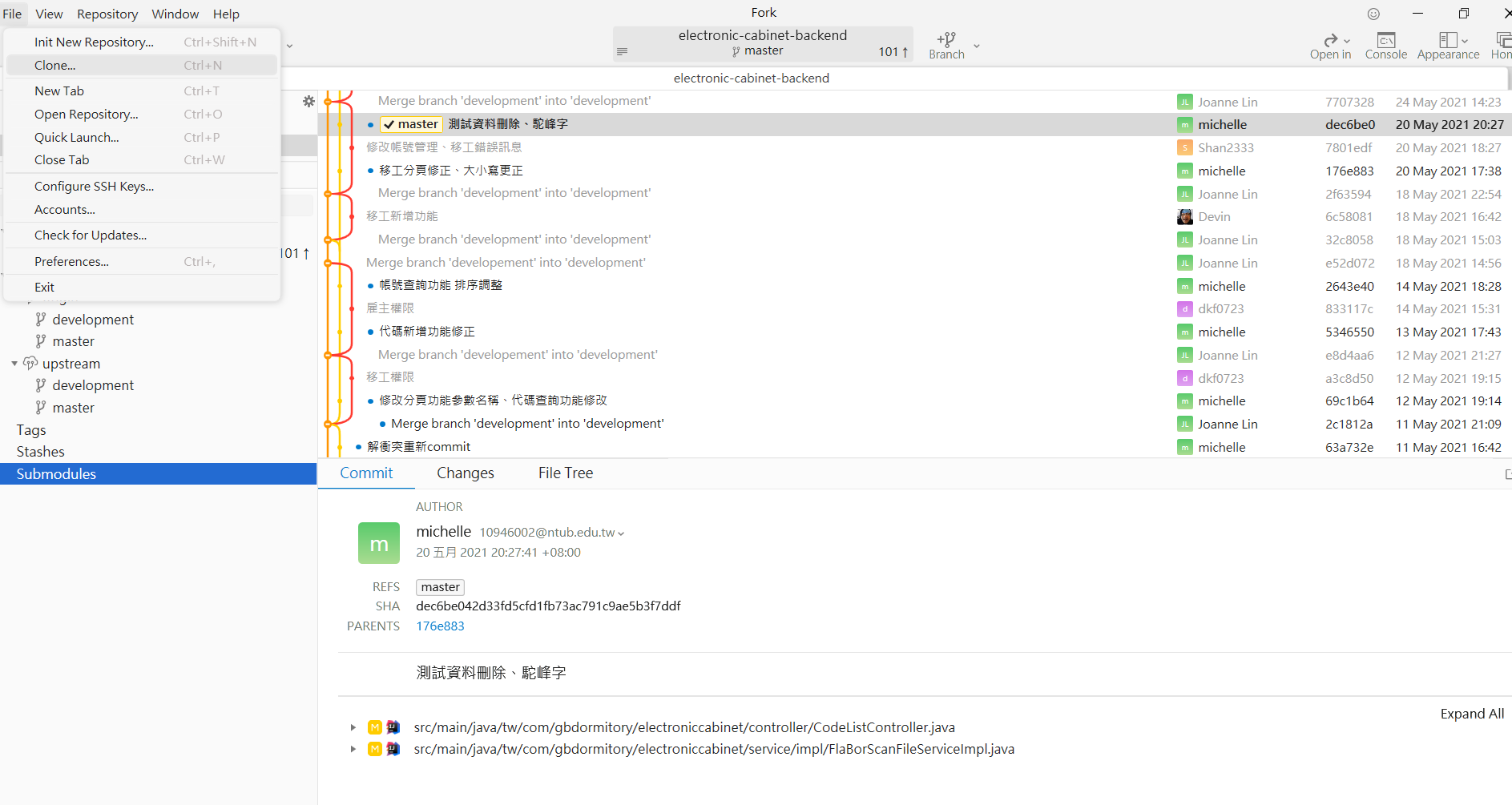
# 新建專案

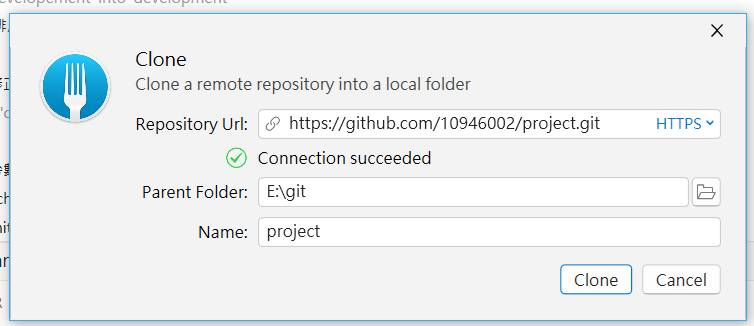
到github輸入專案名稱，並設定公開或是非公開。



選擇HTTPS並複製

打開Fork 點Clone



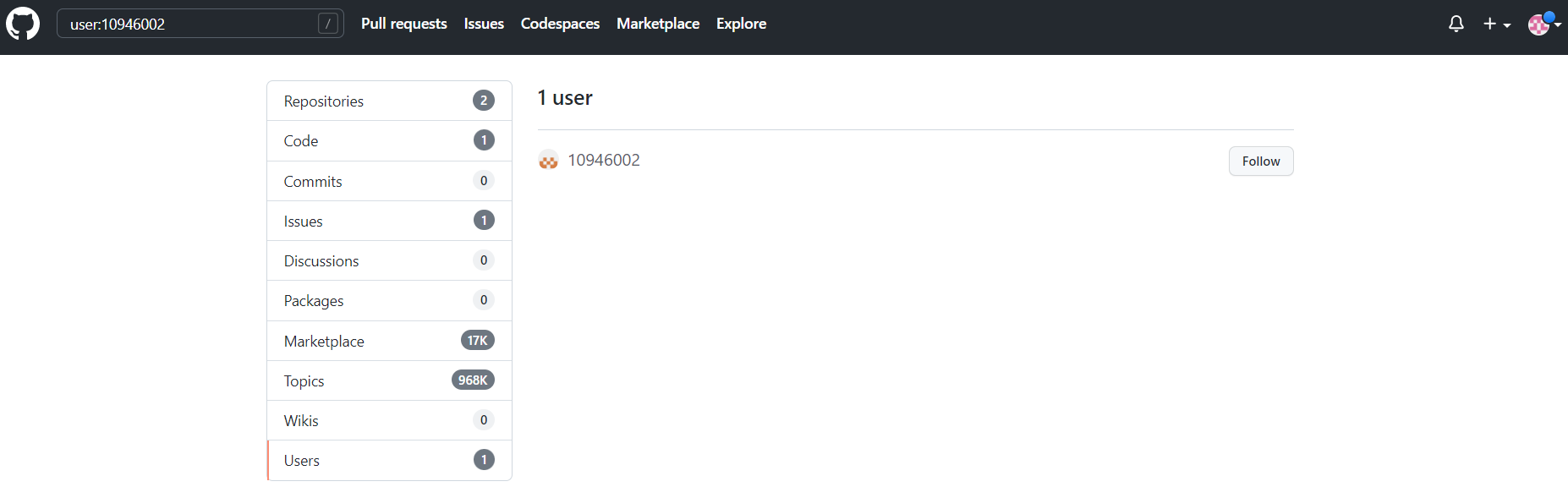
將網址貼上，他預設會將剛剛複製的網址貼上去，可以點Test Connecting 測試連結是否有問題，並選擇存檔的路徑和檔案名稱

\*\*\*專案持有者要先將初始環境設置完畢組員才能使用fork!!!

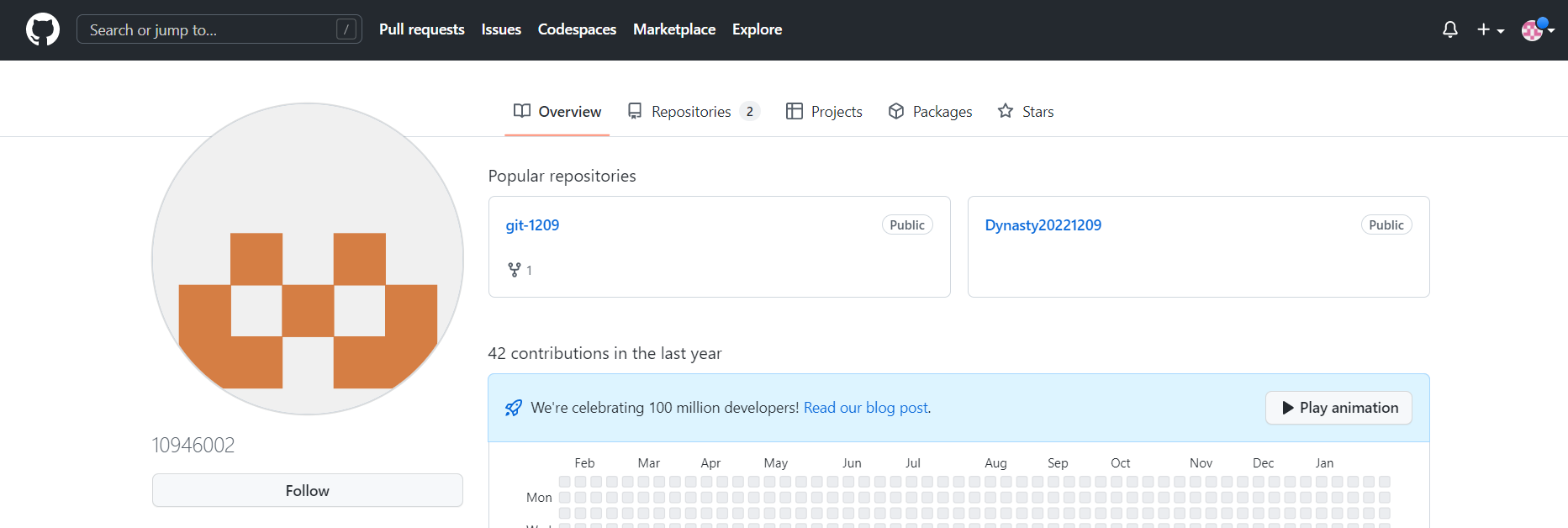
# 加入現有的專案

## 公開專案

搜尋專案持有人的名稱

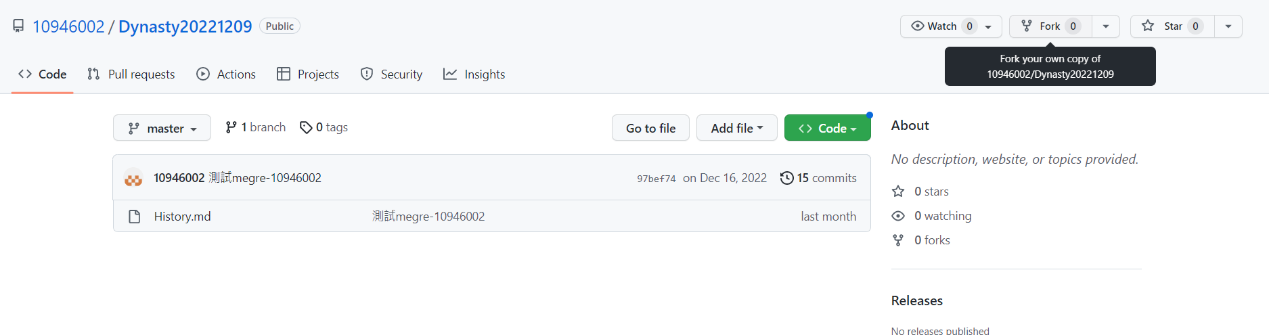


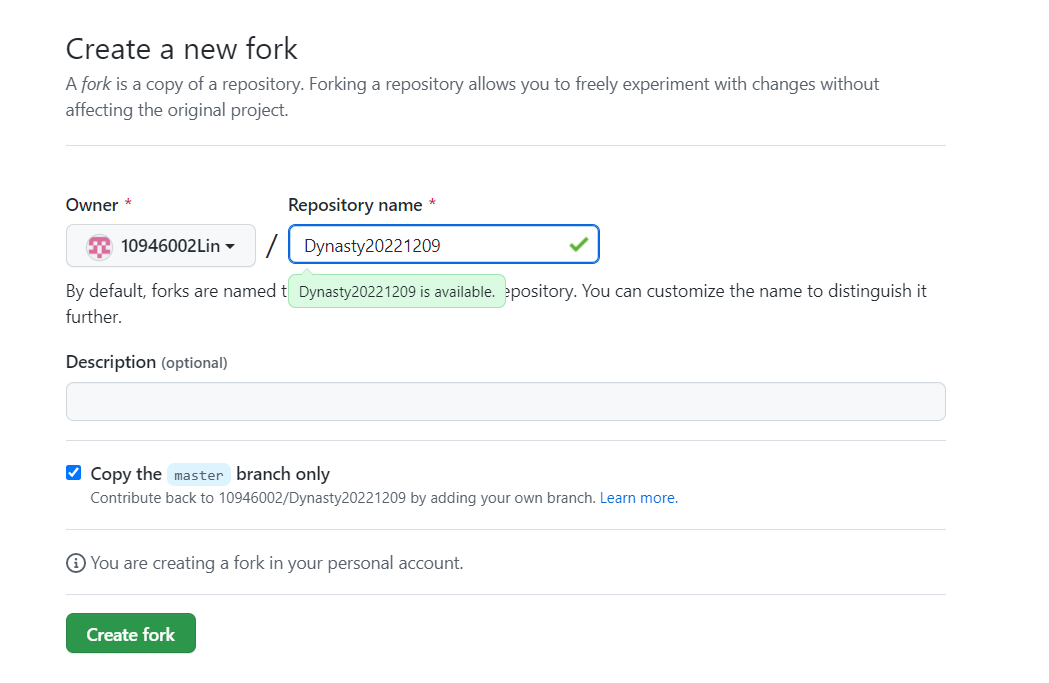
可以看到對方設為公開的專案



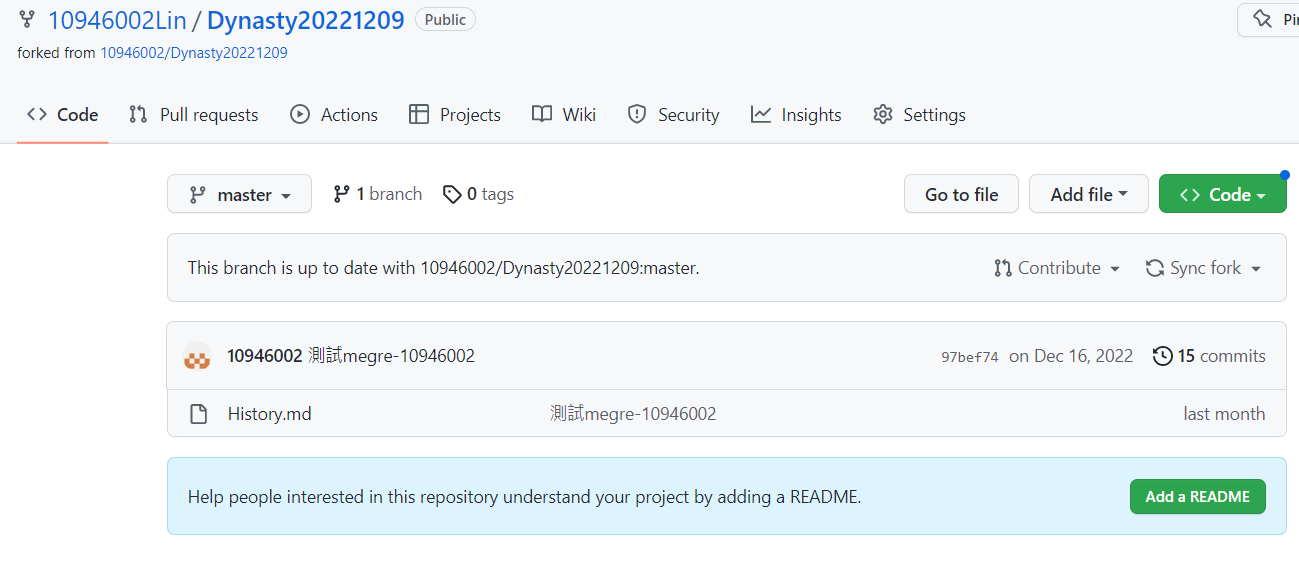
點選要加入的專案點Fork，輸入專案名稱，create fork

(Fork的意思是複製一份當作自己的和原本的空間是分開的)

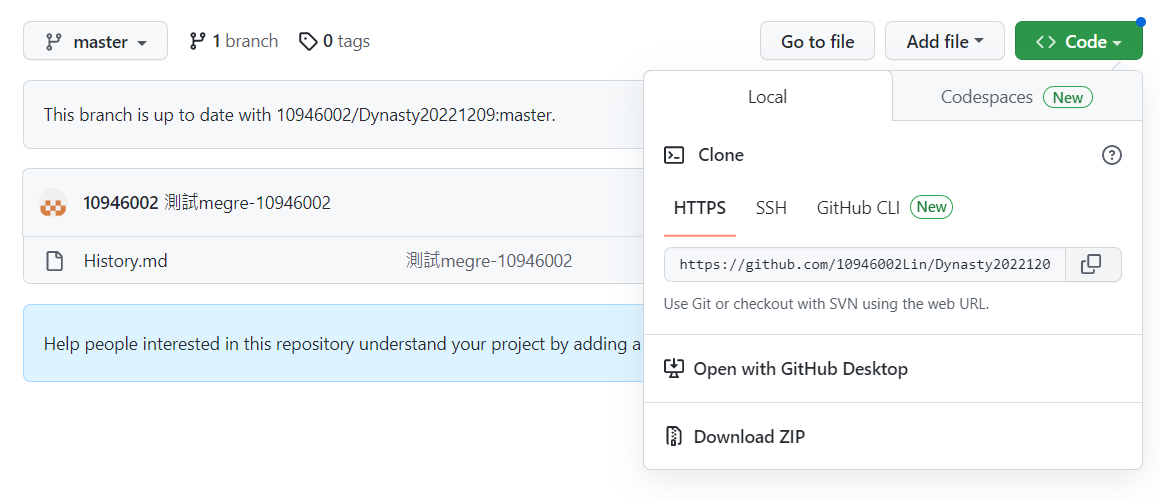




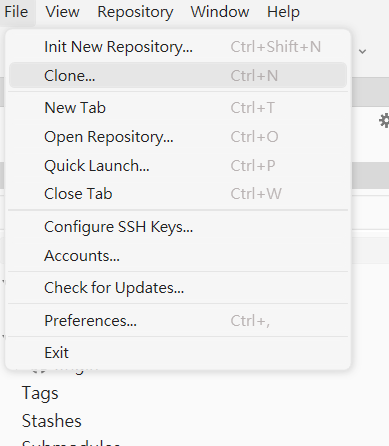
會看到頁面左上角有自己的名稱及專案的名稱，下方有小字顯示這個專案來源



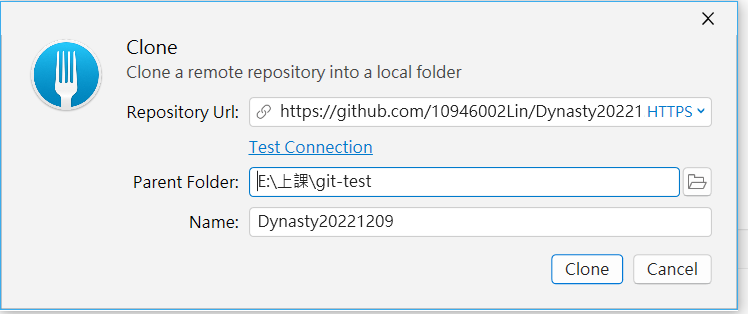
將自己的專案放到Fork裡面，點開code ，選HTTPS，複製網址

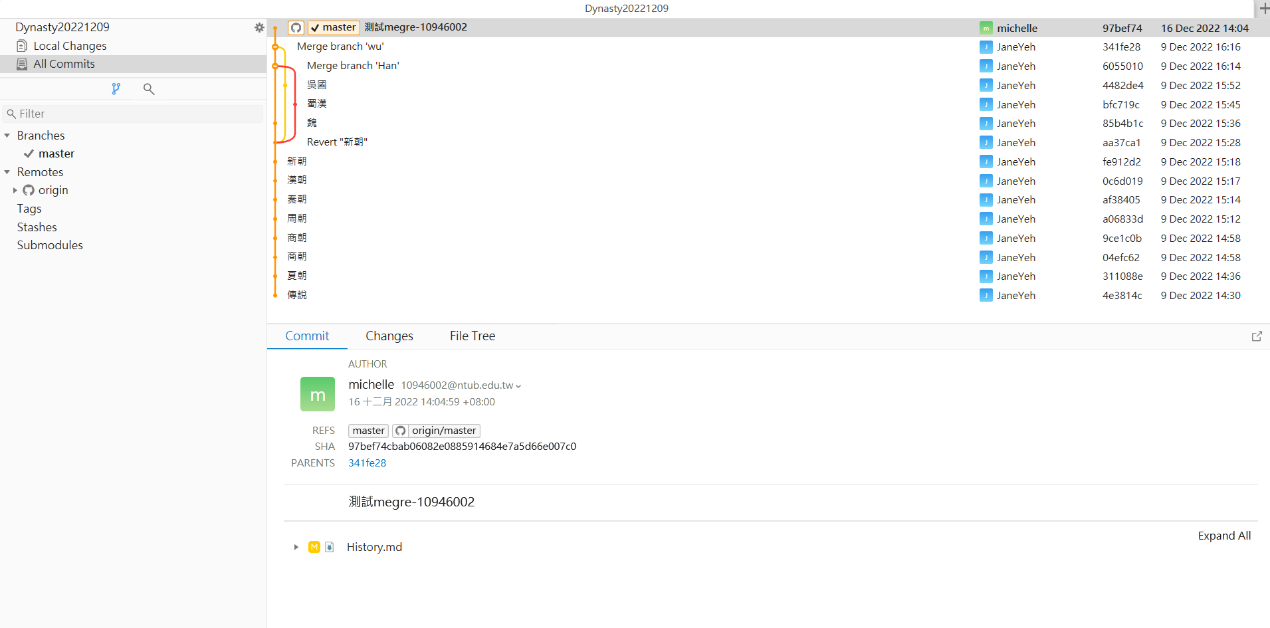


打開Fork，File裡面選Clone



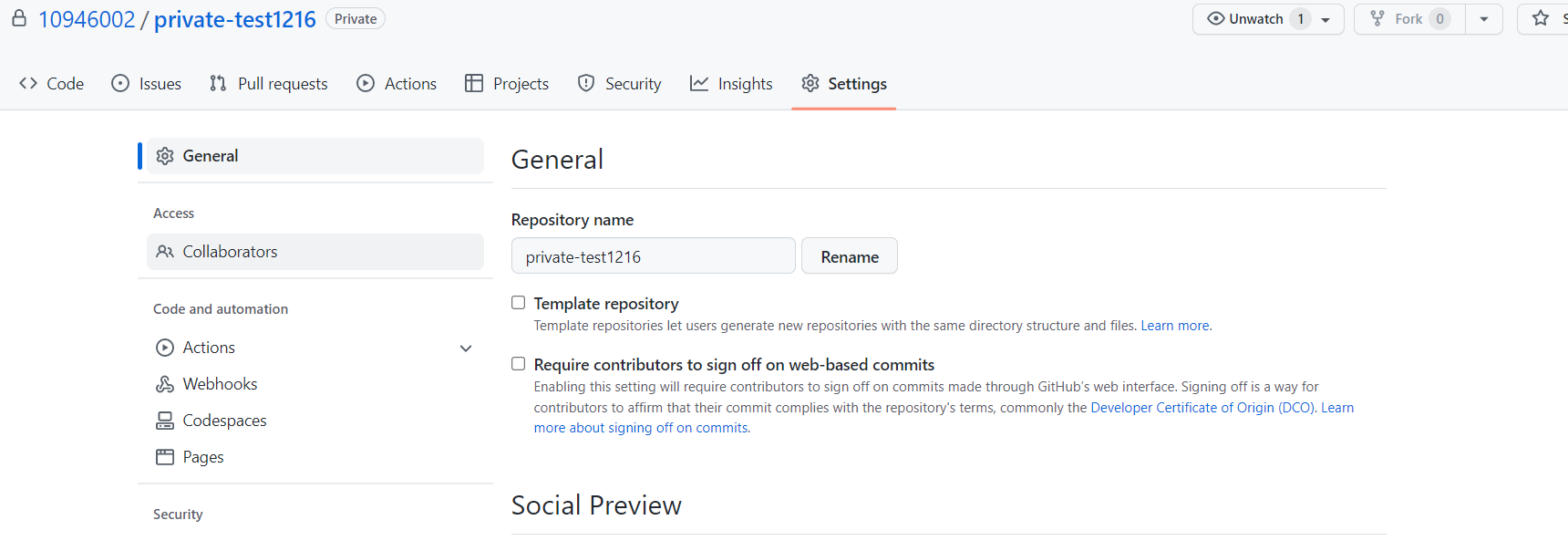
將複製的網址貼上去，這邊要注意是連接自己的檔案空間，可以點Test Connection檢測網址是否正確，選擇檔案的路徑和名稱，完成後按Clone



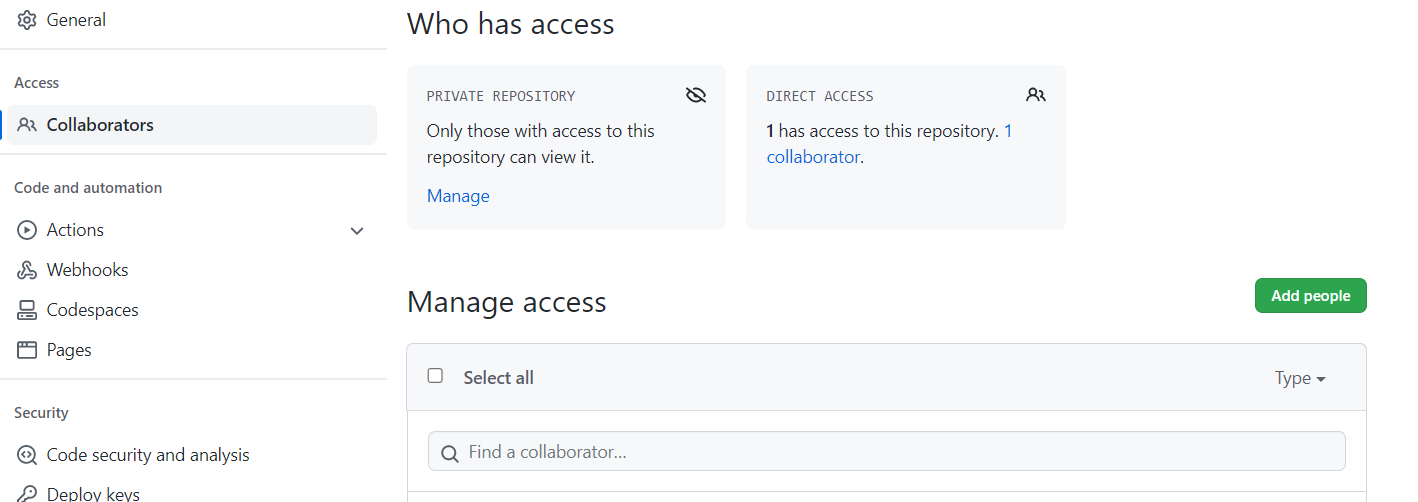
會要求登入github帳號，完成後的畫面，這個就是自己電腦中的檔案位置

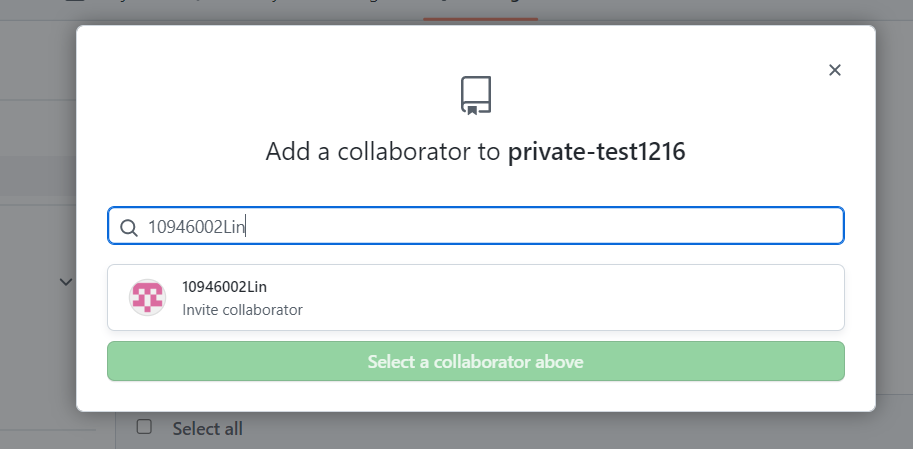
## 加入私有的專案-專案負責人

切到私有專案的setting，選Collaborations



選add people，輸入對方的名字/帳號，同意後通知對方





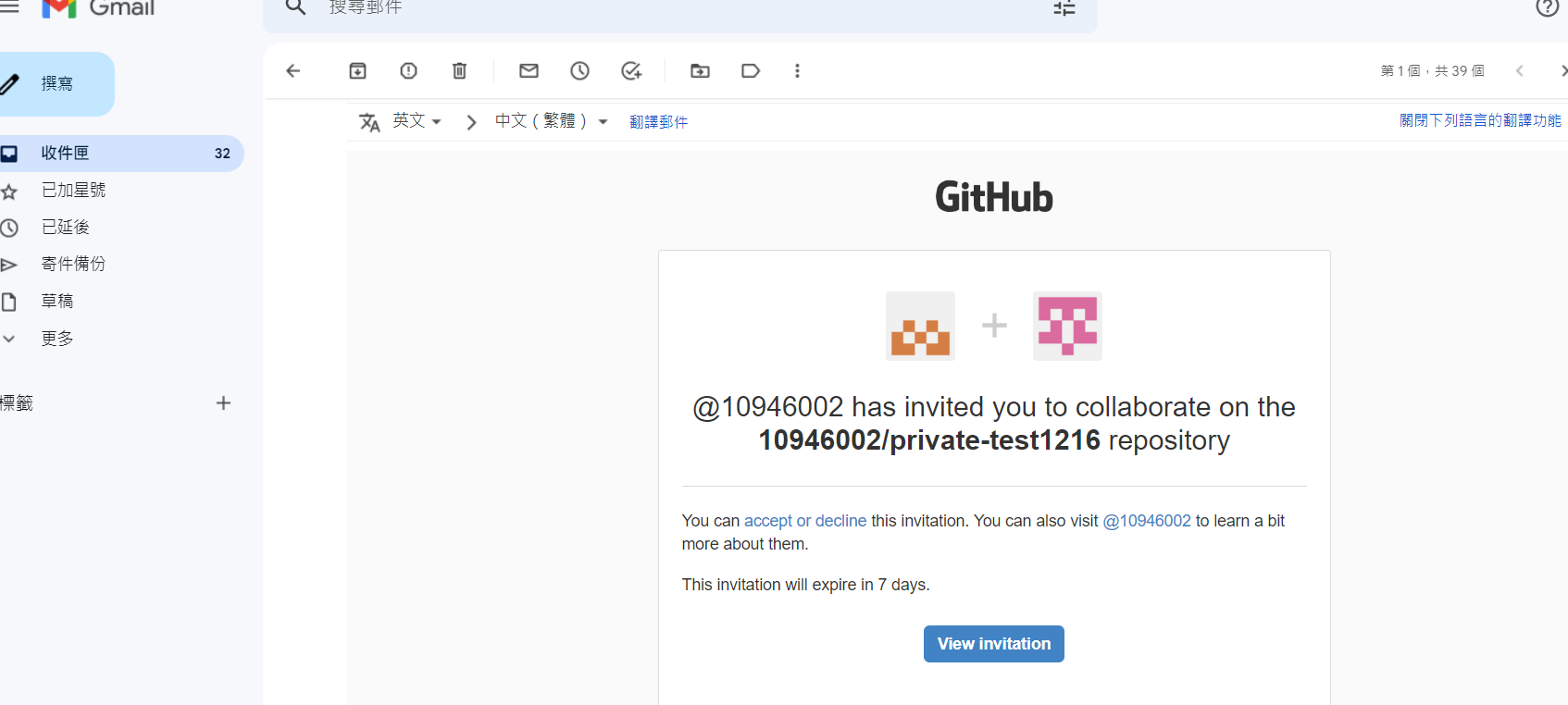
## 加入私有專案-組員

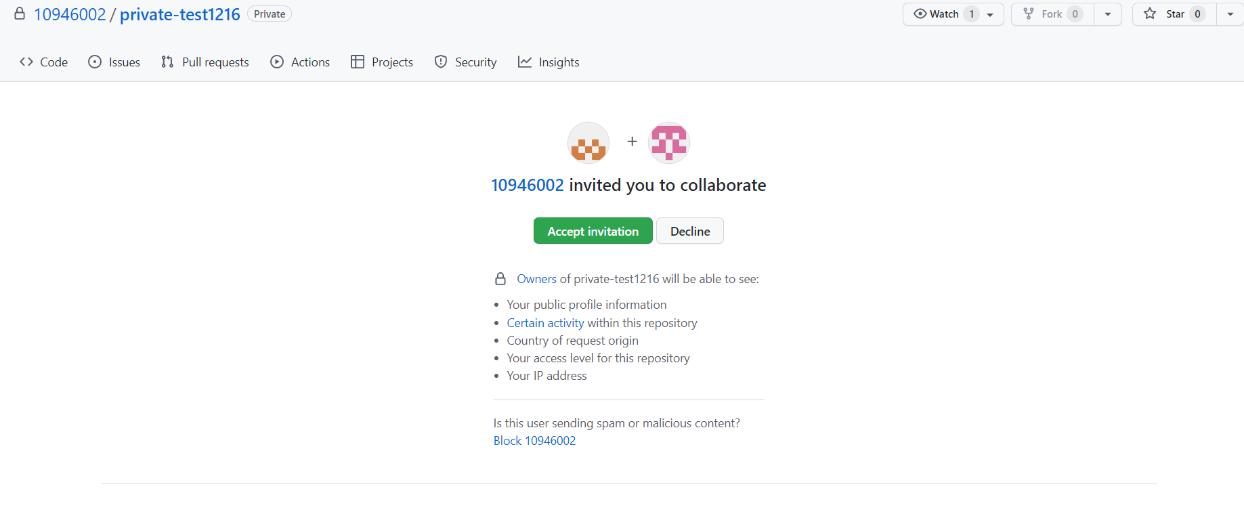
私有專案需要專案持有人的邀請才能加入，須提供給對方**自己的github的名字或是信箱**等對方加自己加進去

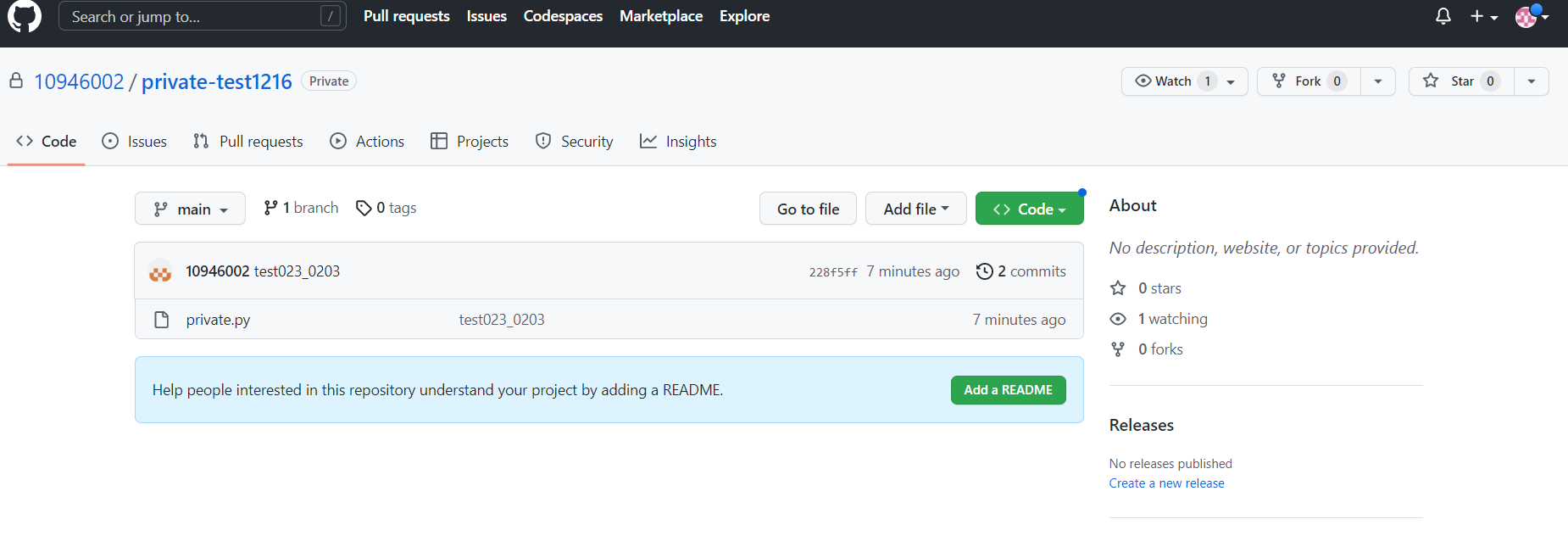
\*查詢用戶看不到對方的私有專案

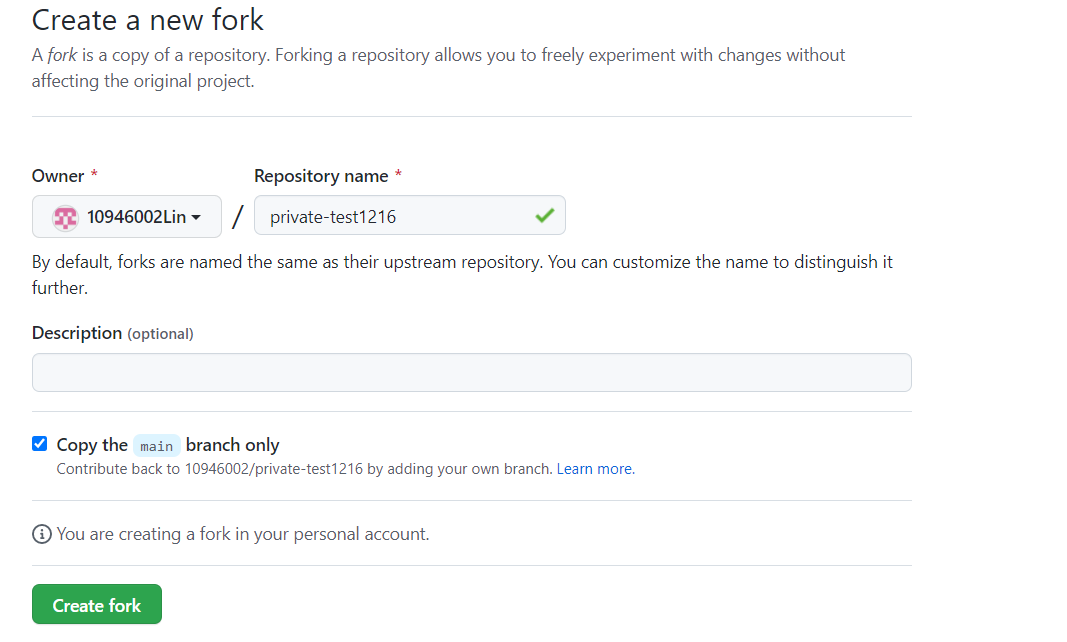


**只能看到public(公開)的專案**

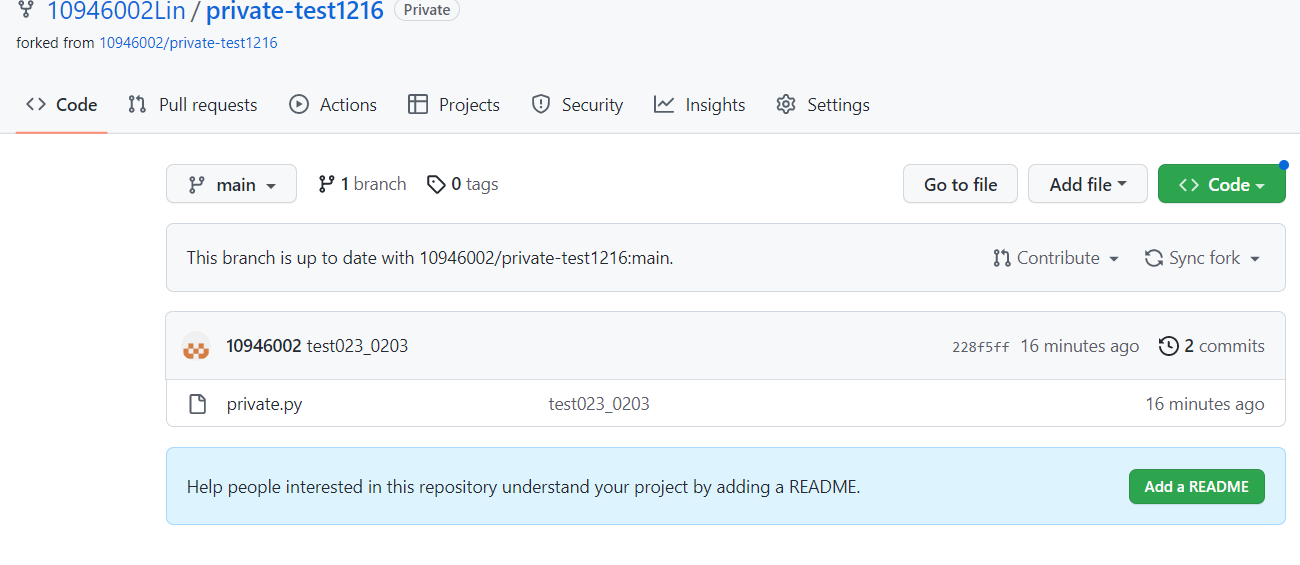
對方通知加入後記得去註冊github的信箱收信，接受邀請



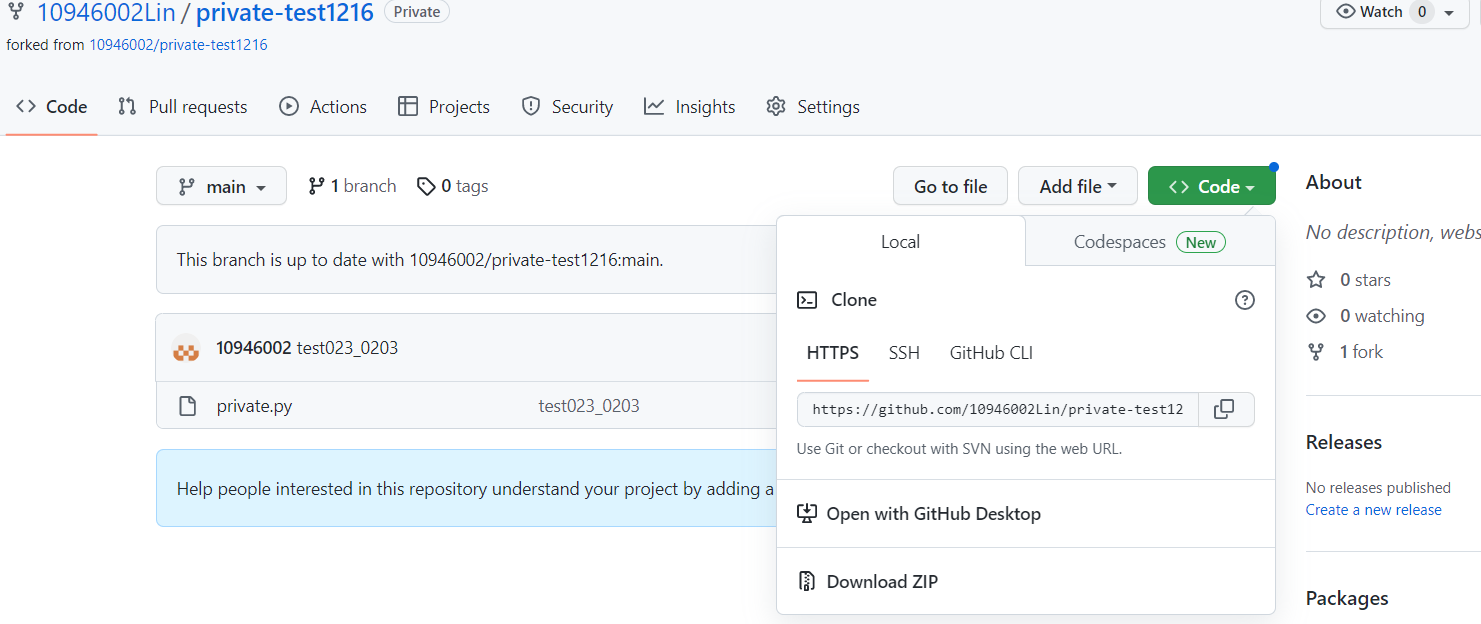
接受後會看到這個畫面，左上角看到的是對方的專案，選Fork，(Fork的意思是複製一份對方的檔案為自己使用)，打上專案名稱後Create fork



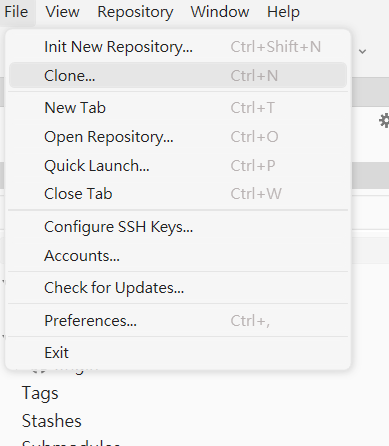
看到這樣的畫面才算成功，右上角會顯示自己的名字和該專案名稱，下方顯示專案的來源，代表這個是自己的空間和原本專案持有人的檔案是分開的



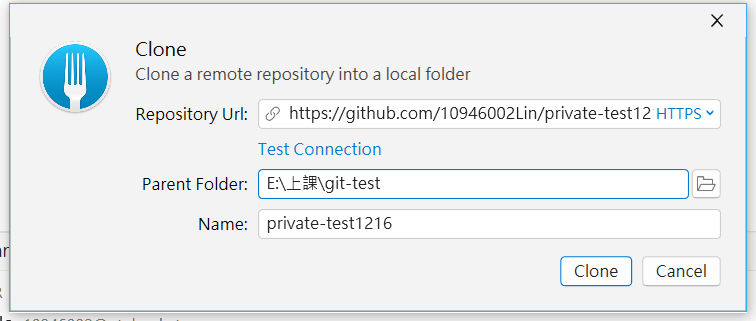
將自己的空間載入fork裡面，確認是自己的空間後點開code，選擇HTTPS，複製網址



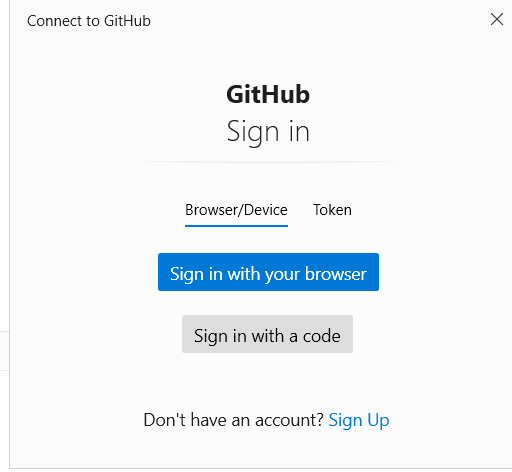
進入Fork裡面的File選Clone

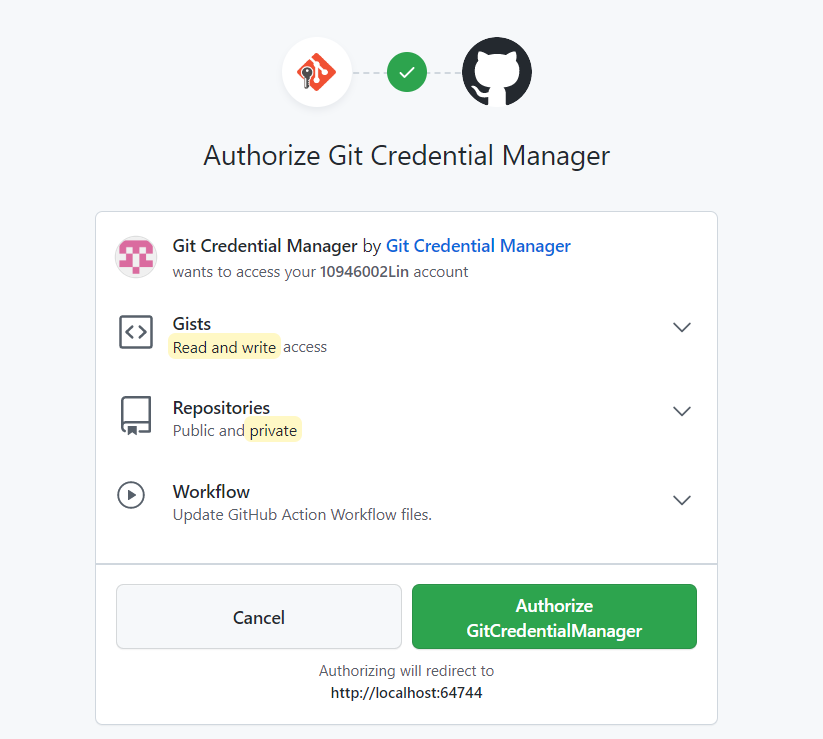


將網址貼上，確認是自己的空間和專案名稱選擇存放檔案的路徑位置和檔案名稱，可按Test Connection 測試網址是否有誤，沒問題按Clone

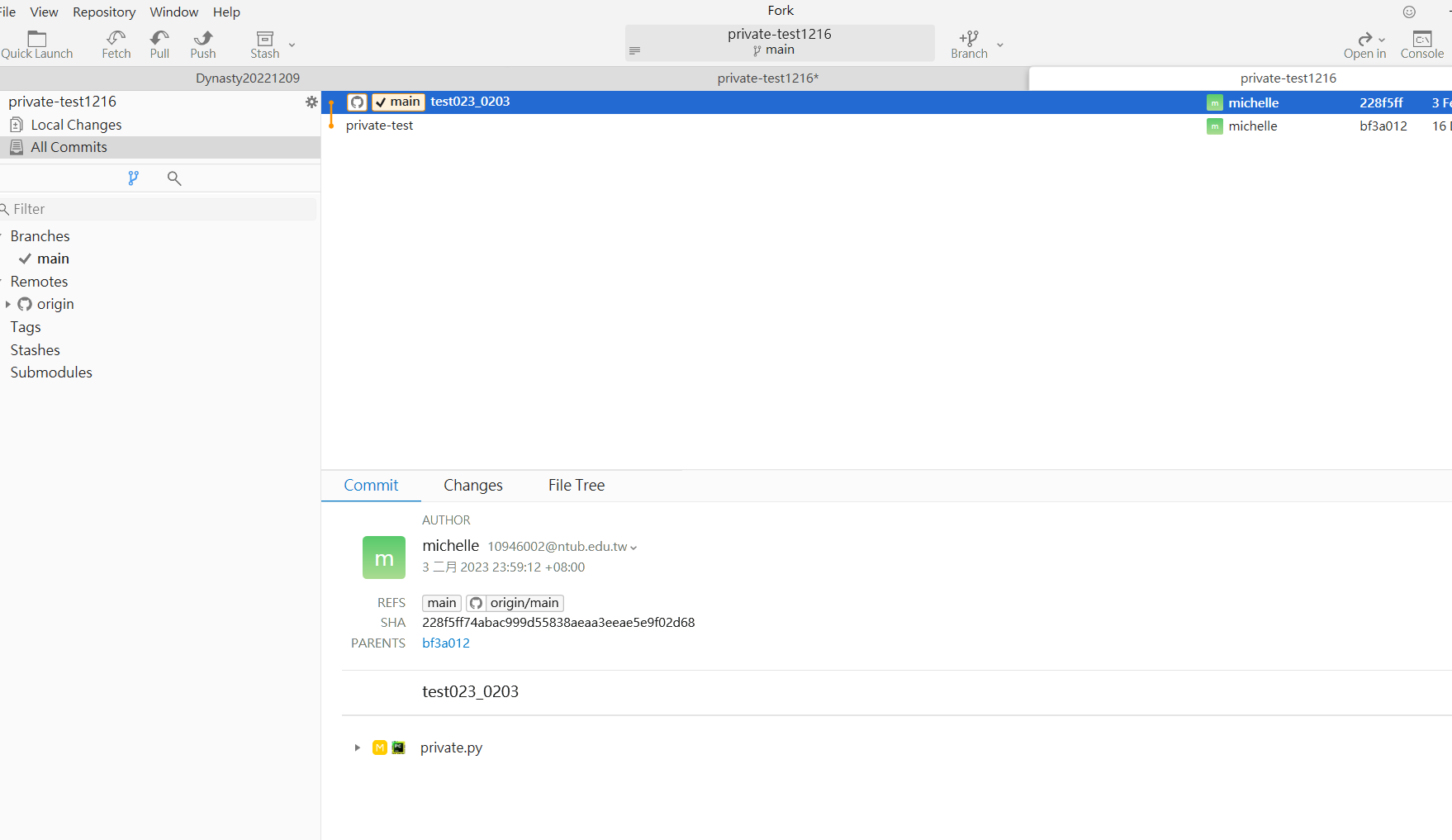


有要求登入github帳號就登入





完成後畫面



# \*加入分支-共用端和本地端

上面Fork的環境裡面，是存在自己的電腦本機端的空間，git功能在於它可以有一個共用雲端和本地端。

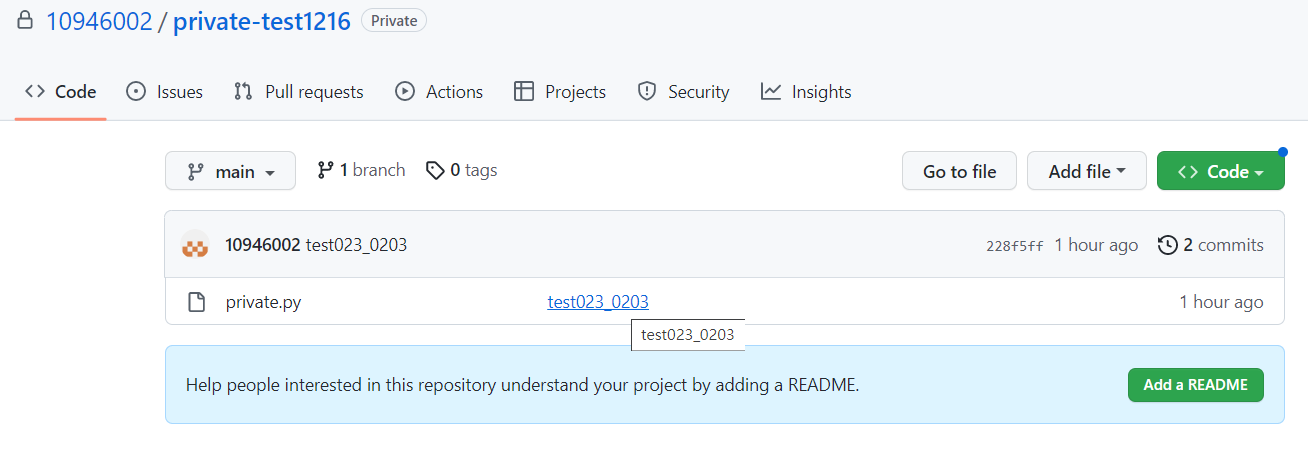
**共用端：**大家能把各自的程式和檔案寫完後一起合併成一個共同的檔案的共用空間

**本地端**：屬於開發人員自己的空間，個人空間的概念，所有自己寫的程式都是在本地端完成後再傳上去共用端

在github裡面要Fork就是為了建立自己的本機端，不然大家都把自己的丟到公共空間萬一程式有問題可能會蓋掉或是刪除別人寫的東西或是導致成是不能跑，整份就毀了。

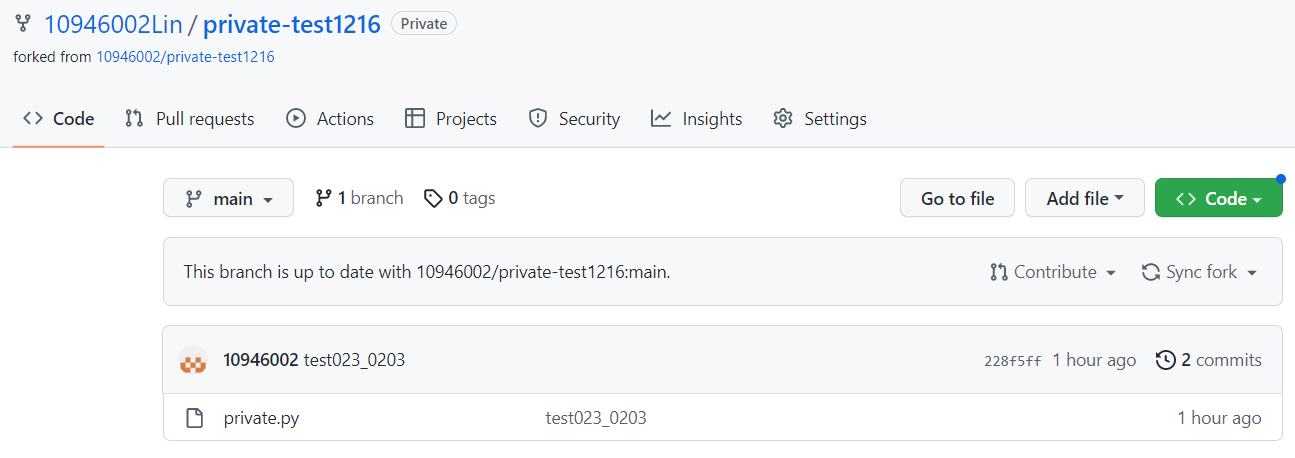
**Github上面區分公用端和本地端**

**共用端**



**公用端專案名稱開頭會標示該專案的建立者，本地端則是會標示自己並附上一行小字表示該專案的建立者是誰**

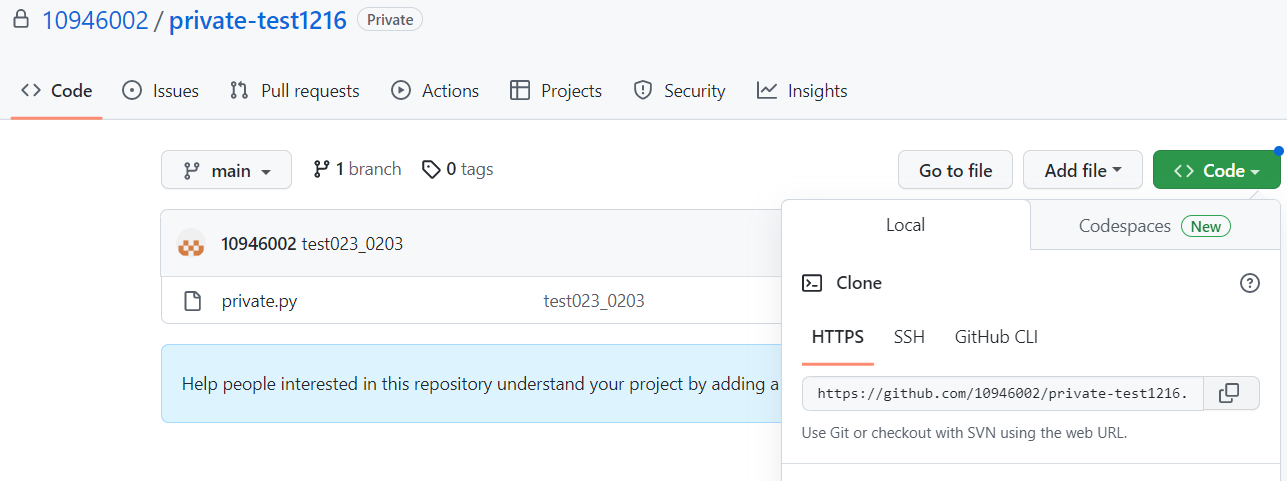
**個人端**



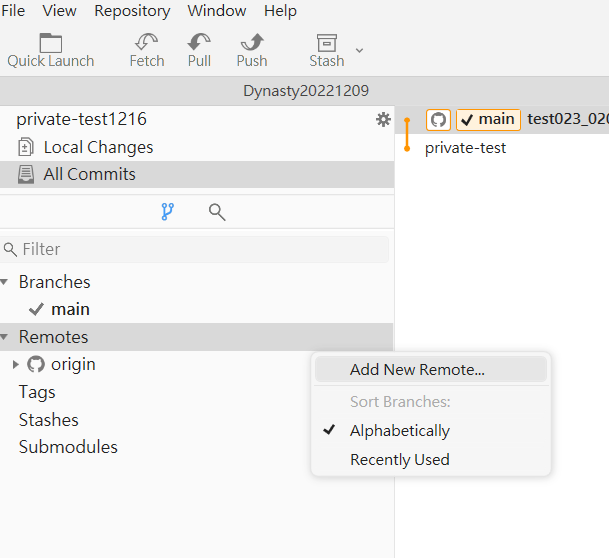
## 加入共用端分支

加入共用端的分支是為了方便合併程式和即時更新進度用的，可以將已經完成放在共用端的程式下載回來個人端，可以即時知道組員寫了甚麼更新了甚麼部分，不用去找別人要程式碼。

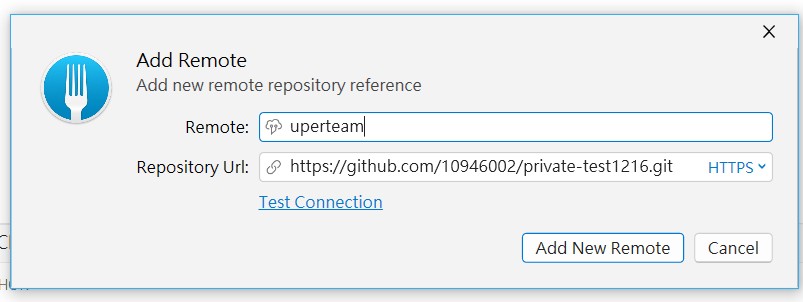
在github裡面到**共用端**的專案裡面code，選HTTPS，複製網址



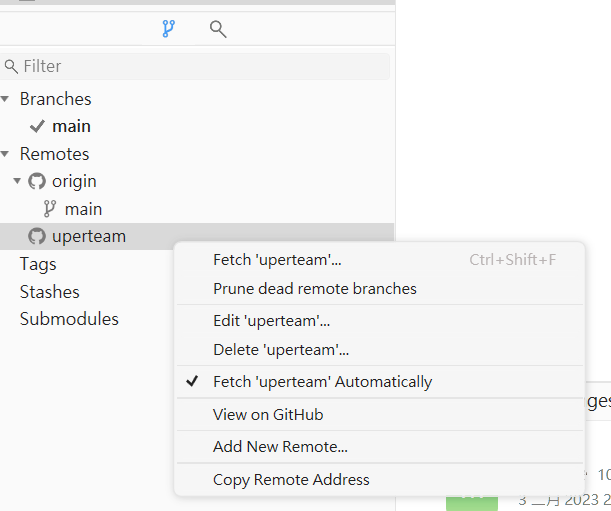
回到Fork，在Remotes按右鍵，Add New Remote



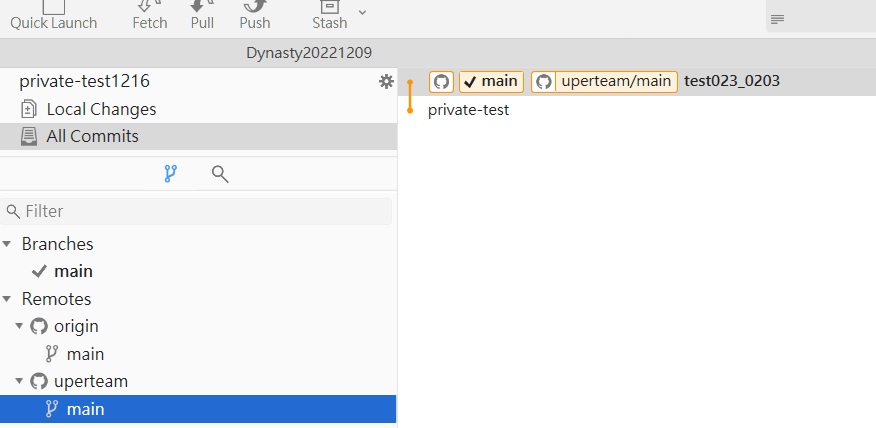
將剛剛複製的網址貼上，確認是公共端的名稱，Remote打上該分支的名稱，可以按Test Connection測試複製的網址有沒有問題，沒問題後Add New Remote



右鍵點擊剛剛創建的分支，Fetch uperteam



完成畫面，右邊會跑出兩個，Branches裡面的main代表自己電腦中的檔案和origin裡面的main是相同的，要分得更細可以再建立一個分支代表自己本地電腦的分支，Remote裡面origin中的main代表自己的本地雲端，uperteam代表共用端，此圖代表目前自己的所有進度都是進度是相同的。

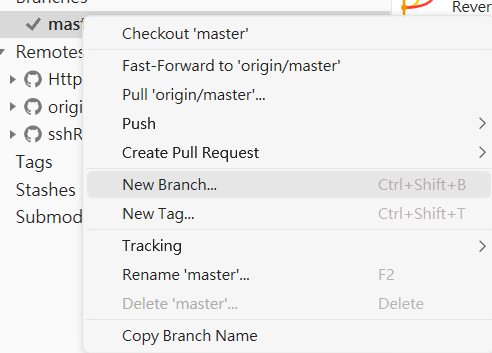


**共用雲端**

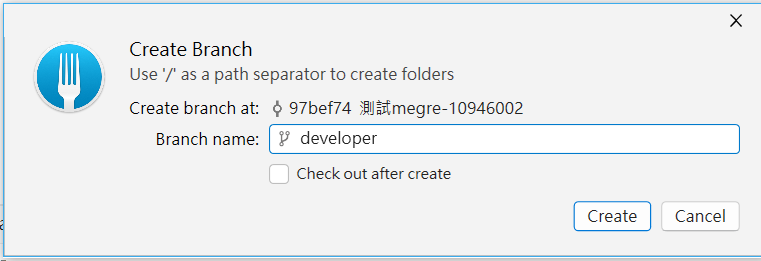
**個人雲端**

**本地電腦中檔案**

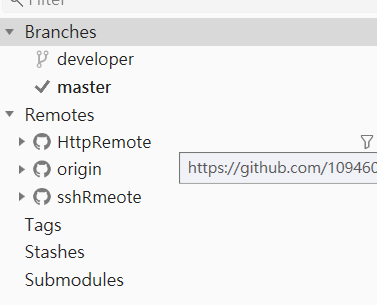
建立新分支，Branch右鍵，New Branch



打上分支名稱，create

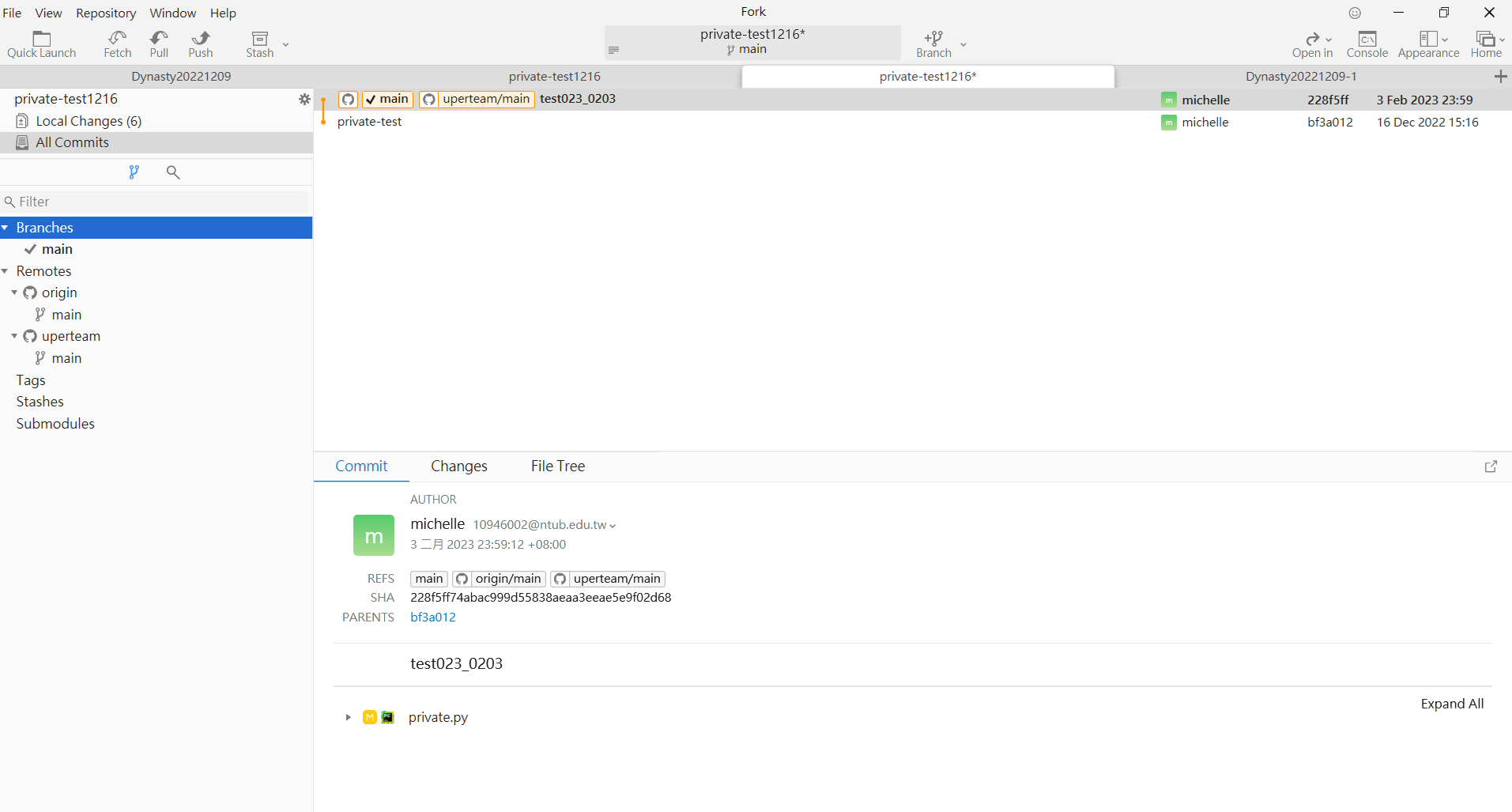


完成畫面，新的分支會依據目前最新的檔案進度建立，有打勾代表目前在哪個分支裡面，兩個分支內容是獨立的。



# Fork相關操作

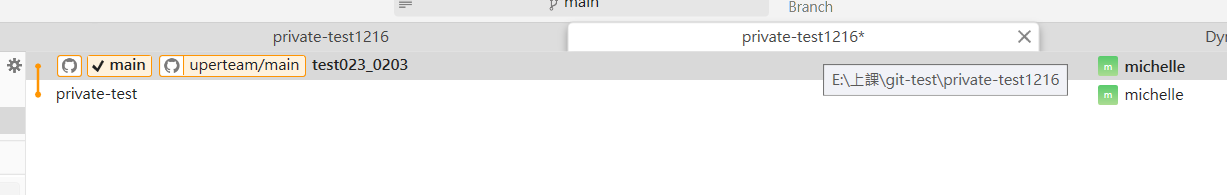
Fork裡面可以查看所有的專案修改紀錄，針對專案所有的修改上面都會顯示

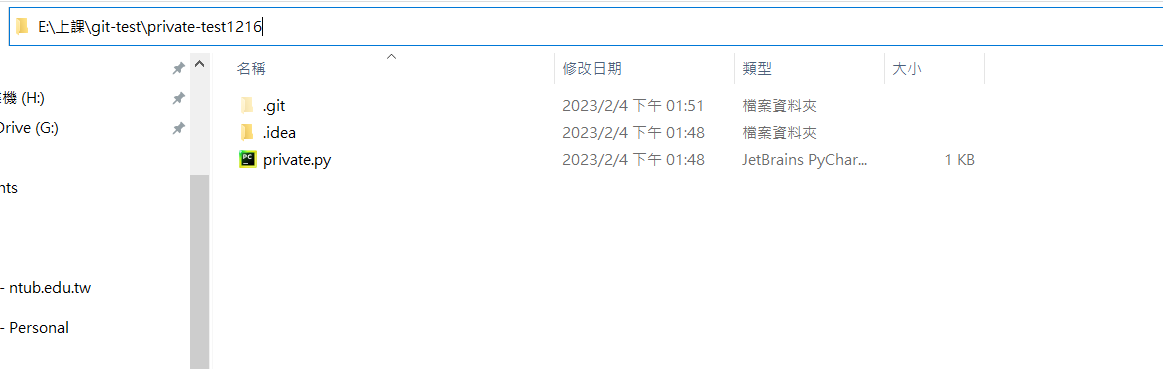


可以查看每一筆修改的細項和修改的檔案內容(僅顯示有修改的部分)

專案的總體紀錄，可以知道上傳者是誰還有上傳日期

將滑鼠放在專案的頁籤上面可以知道該專案的檔案路徑，只要在這個路徑內的所有修改都會顯是在fork裡面。





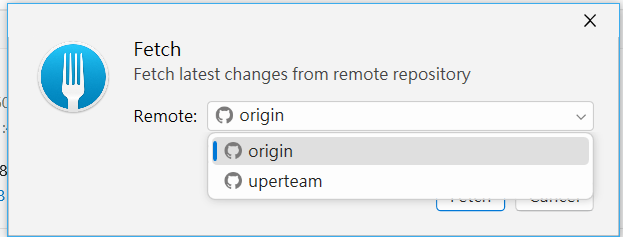
## 本地端

### Fetch、Pull、push、stash

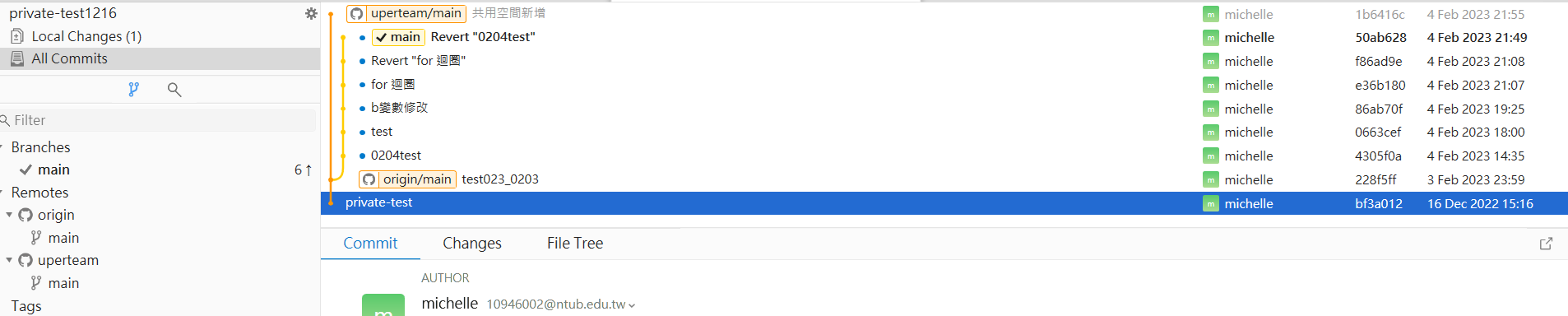


**Fetch** 重新整理，可針對分支做Fetch，原本的資料不會消失

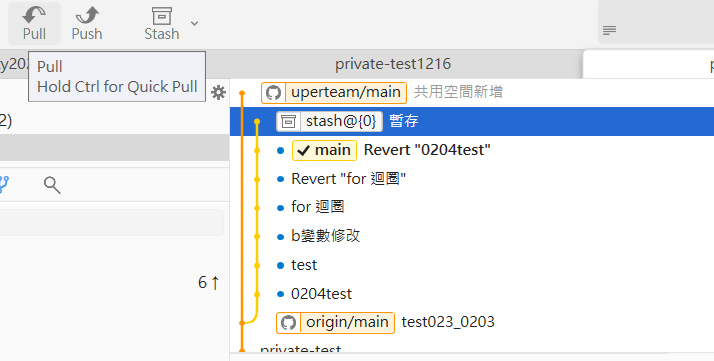
問你要重整哪個分支



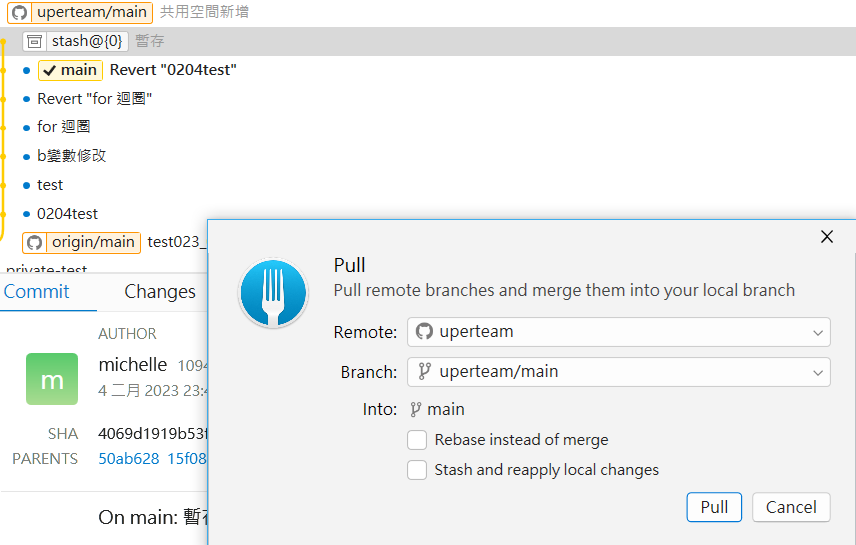
**Pull**  下載，將共用空間的檔案下載回到電腦

共用空間有更新的檔案的話，可以看到出現了兩個分支，黃色代表自己的，橘色代表共用端，從test023\_0203開始的紀錄就和本地端不同步

選Pull



會問你要下載哪個分支到自己的本地端，這邊選upstream的main，即共用端。

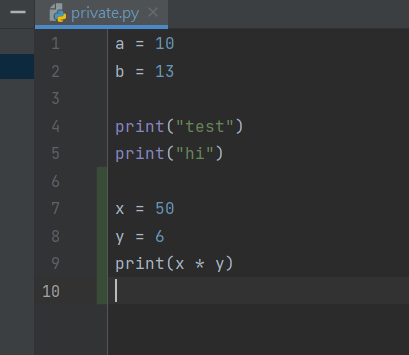


成功後會發現線條接再一起了，代表已經將公共空間的檔案拉下來了，沒有在線上的點就是尚未和公共空間同步的修改

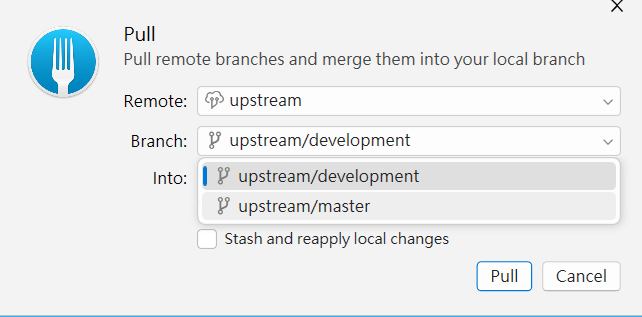
**\*如果Local change有修改紀錄卻還沒commit的話，他會沒辦法下載，看要不要先暫存起來在Pull**



程式碼的地方多出了共用空間的檔案內容

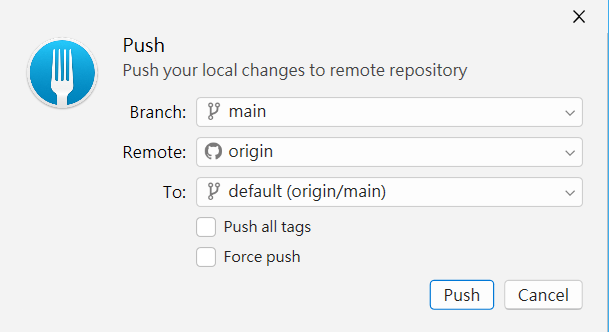


**\*如果是已經有上線的專案，upstream的分支大部分會分已上線和開發中兩個不同的分線，大部分狀況是拉開發中的那個，這樣才會拿到其他同伴修改的部分。(下圖中Developer為開發中的分支，master為正式上線的分支)**

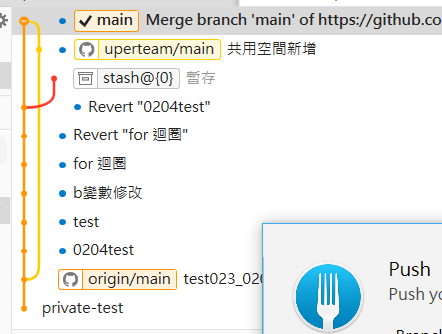


**Push** 上傳 將自己本地電腦中的檔案上傳到本地端/共用端，**切記不要隨意點，不然會直接蓋掉共用空間的檔案**(後面會說明合併檔案的步驟)

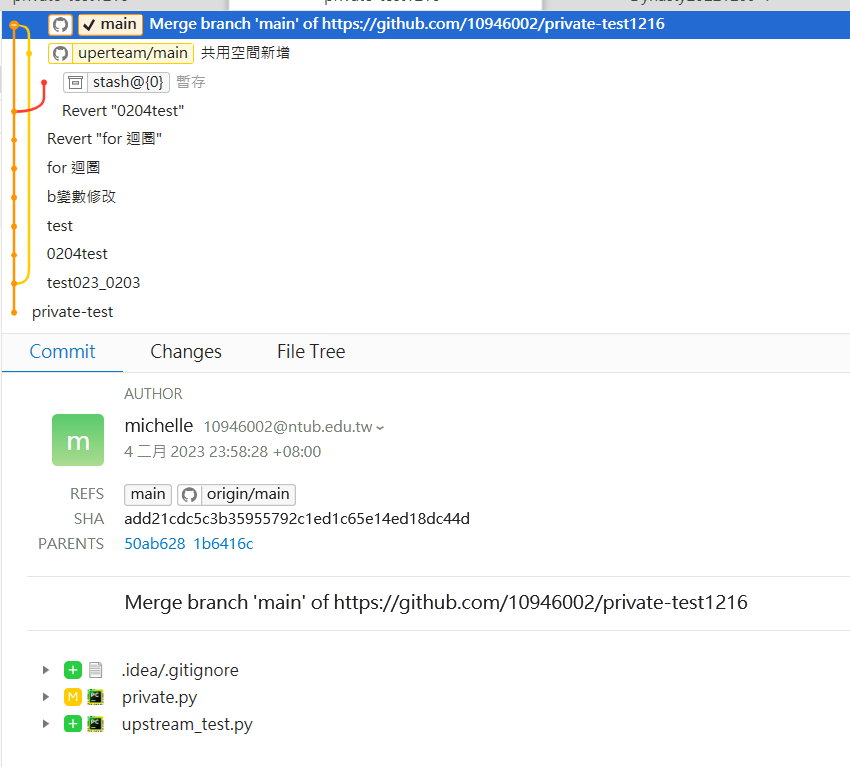
一般push，只要將本地電腦中檔案上傳到個人端就好了，這邊會問要上傳到哪裡，這邊都選**個人端切記不要選錯**，不然共用端檔案就沒了



推上去之前，orgin/main在最底下，



推上去後和main在同一層，就代表電腦中檔案已經上傳到個人端去了

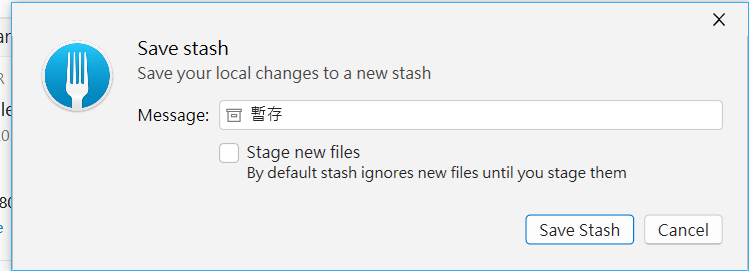


**Stash** 暫存，將目前已經修改但是還沒Commit的程式暫存起來，未來有需要可以倒回去

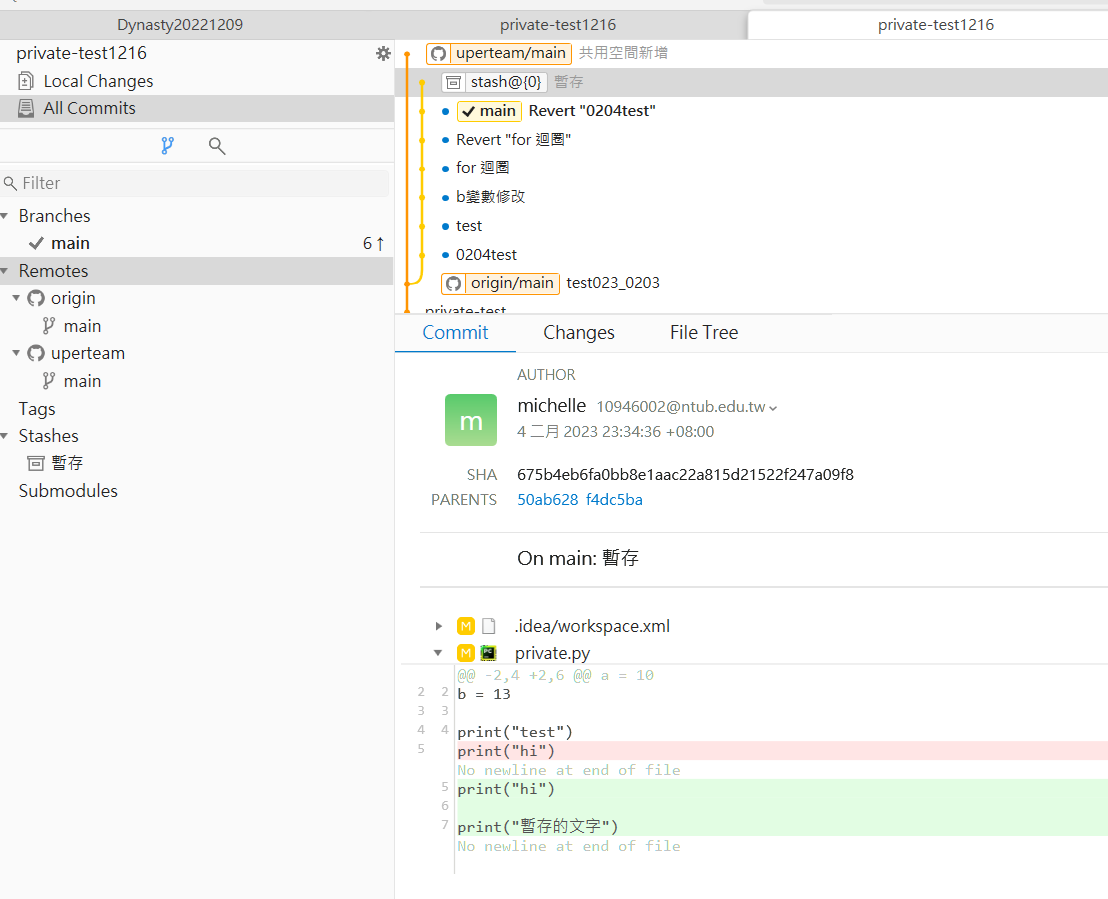
這是目前還沒commit的部分，按下stash



輸入該暫存的名稱(依自已能辨識為主)



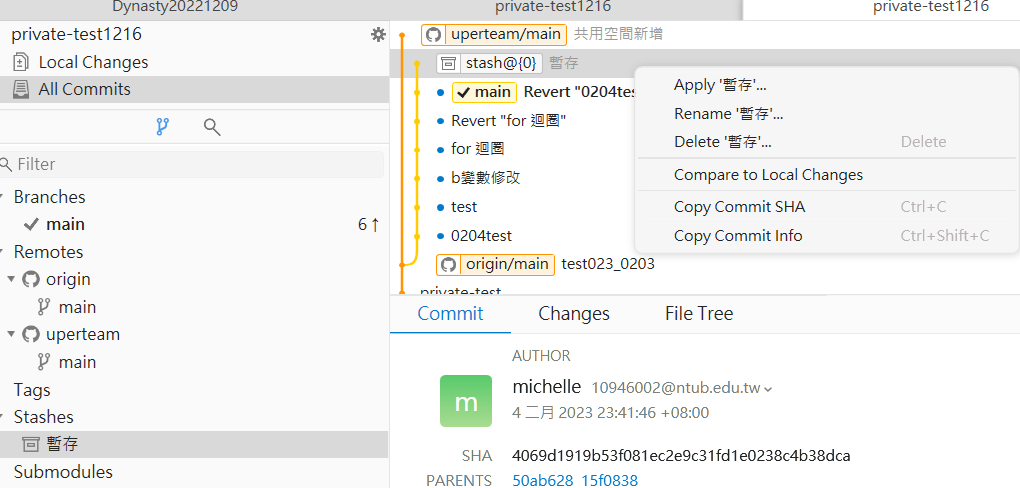
暫存下後local的部分會歸零，並生出一個暫存紀錄



暫存的內容

對stash按右鍵或是左邊視窗可以選擇倒回來還是要刪除，

Apply是倒回去、Delete是刪除

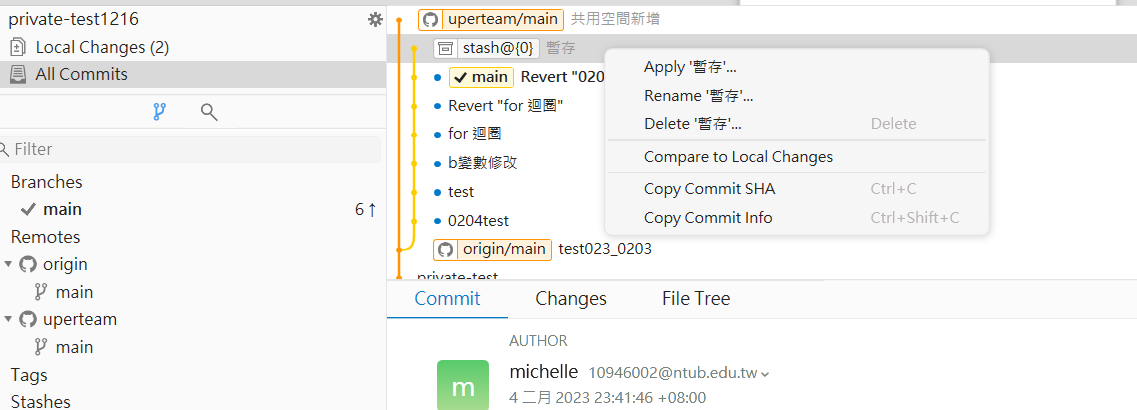


這邊按右鍵也可以

倒回去會問你要不要將暫存的紀錄刪除，不打勾的話也能手動刪除

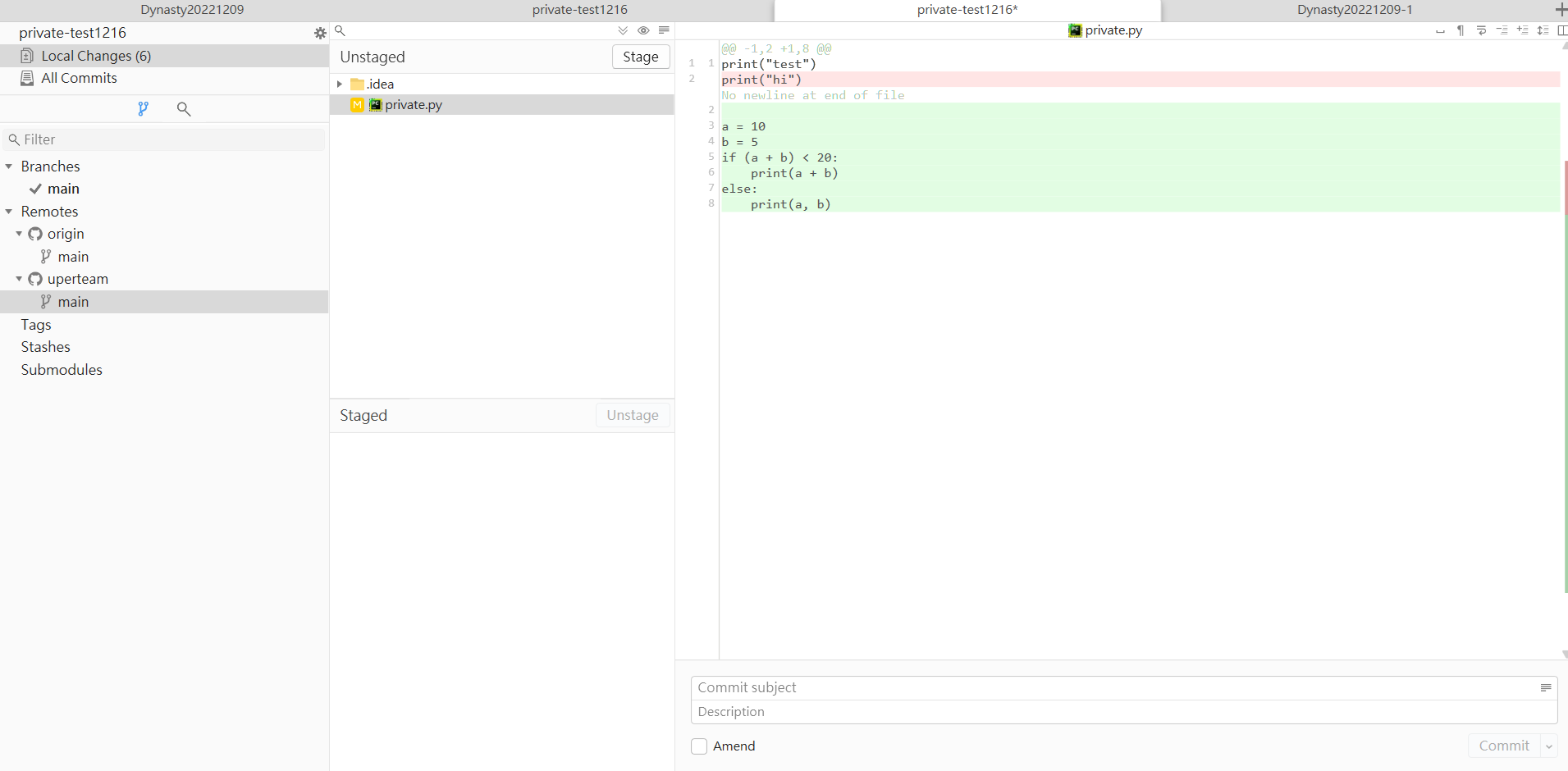


暫存的內容已經倒回去了(Local change有東西)，可以按右鍵選擇要不要刪掉暫存，暫存不會上傳到共用端裡面

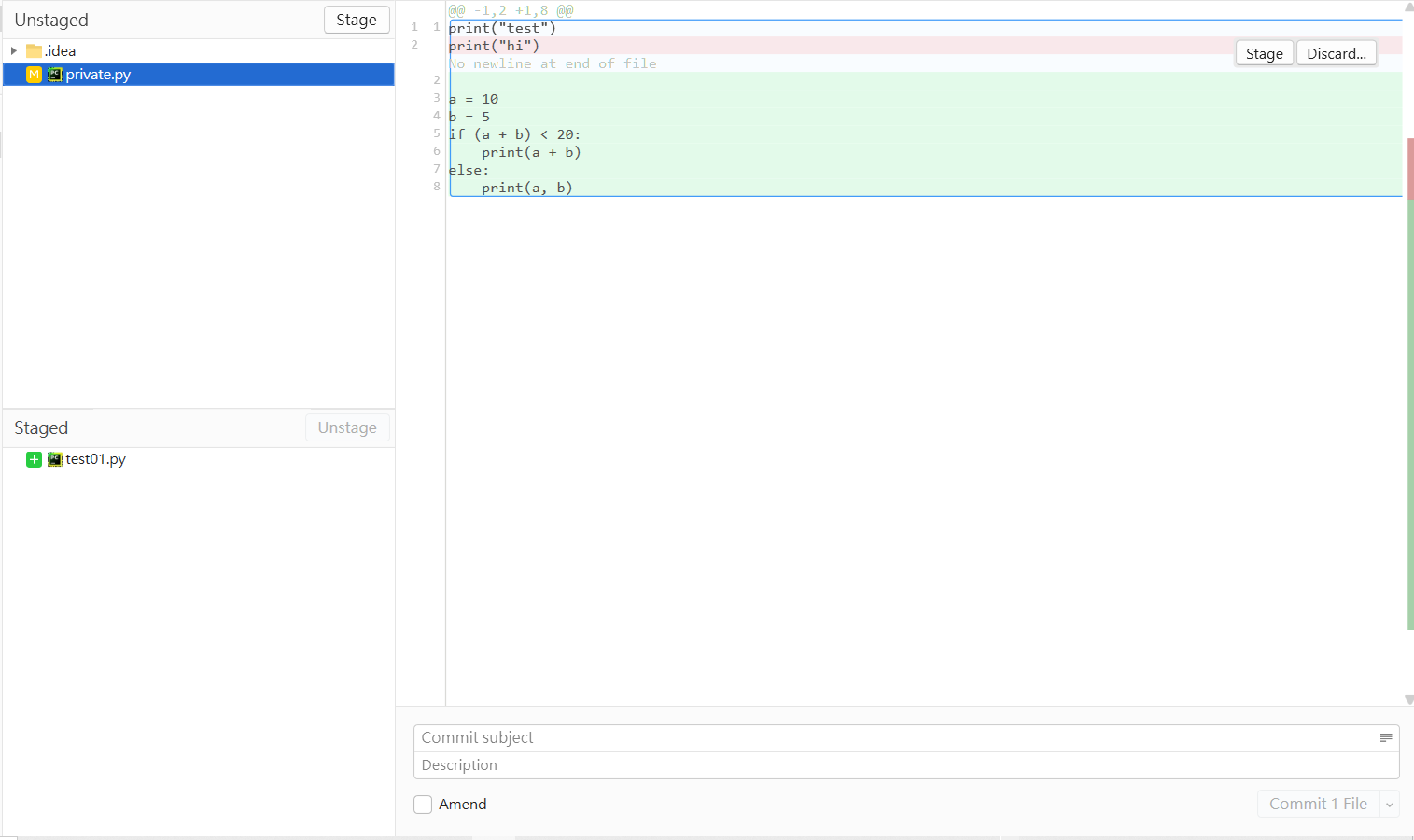


### Commit (將檔案存到個人端空間)

本機修改的檔案在Local Chagne項目，綠色代表新增項目，紅色是刪除的項目



選擇要上傳的項目，可以按stage全部上傳也可以只上傳部分，要上傳的檔案會討到下面的視窗

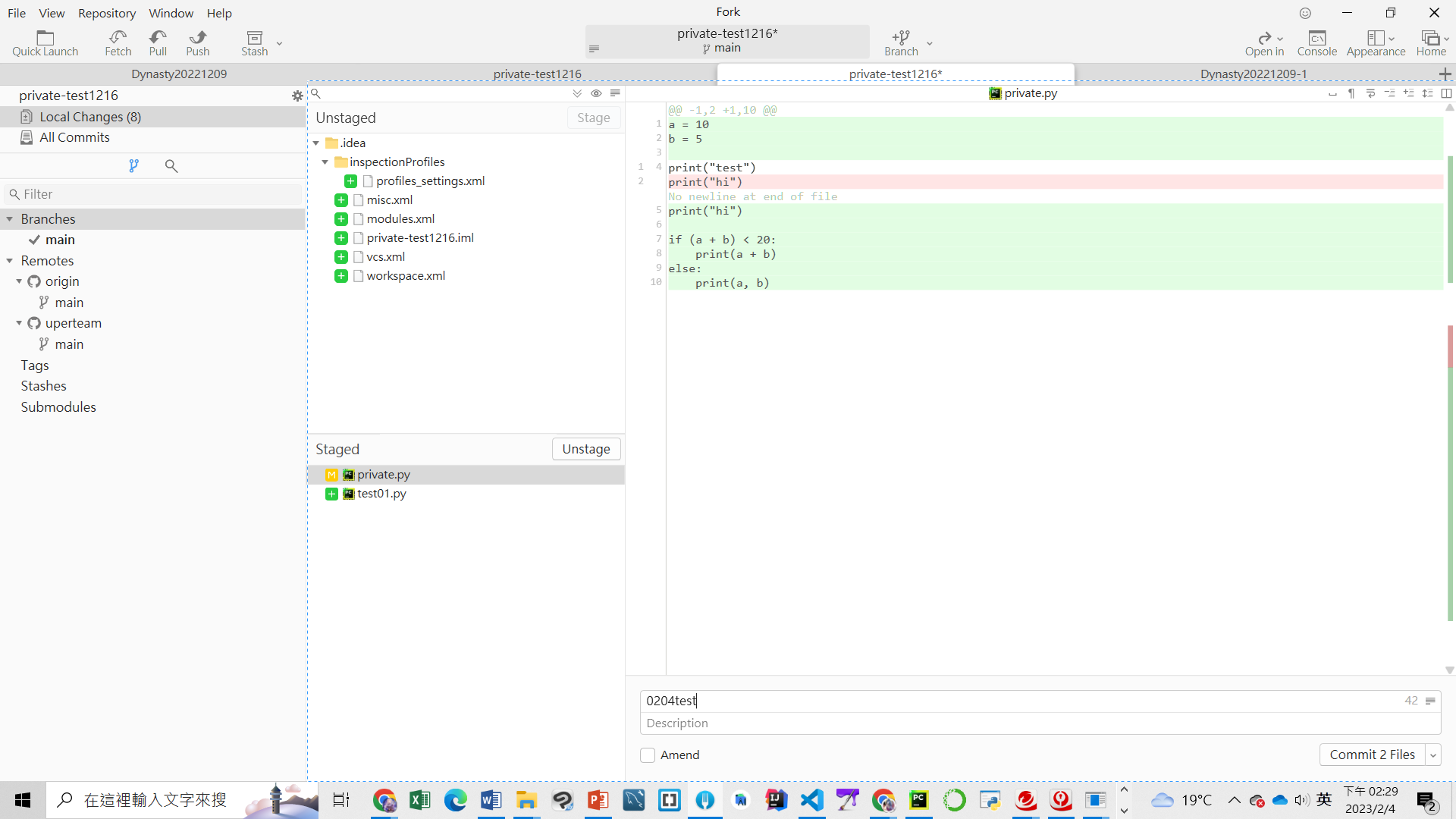


快點兩下可以將檔案移到下方

要上傳的檔案會在這裡

private.py檔案內的修改內容，同個檔案裡面可以只選擇部分上傳，要上傳的部分按stage就會移動到下面的視窗了(以藍色框起來的區域為一組)

確認上傳內容後須輸入更改的項目內容，ex.「XXX功能修改」、「新增XXXX」、「更新XXX」，讓大家比較好辨別該紀錄做了甚麼，不需要輸入日期，Fork會記錄上傳時間，按下commit



Commit的項目會顯示在最上面，因沒有上傳到共用端，其他人不會看到這筆修改也無法拿到有更改的檔案，不需要全部寫完再commit，可以完成部分就commit一次，多一些紀錄

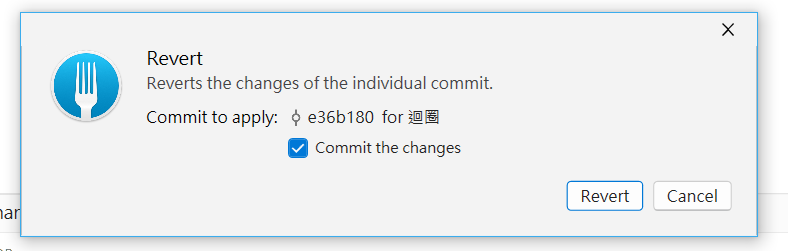


### Revert(退回commit)

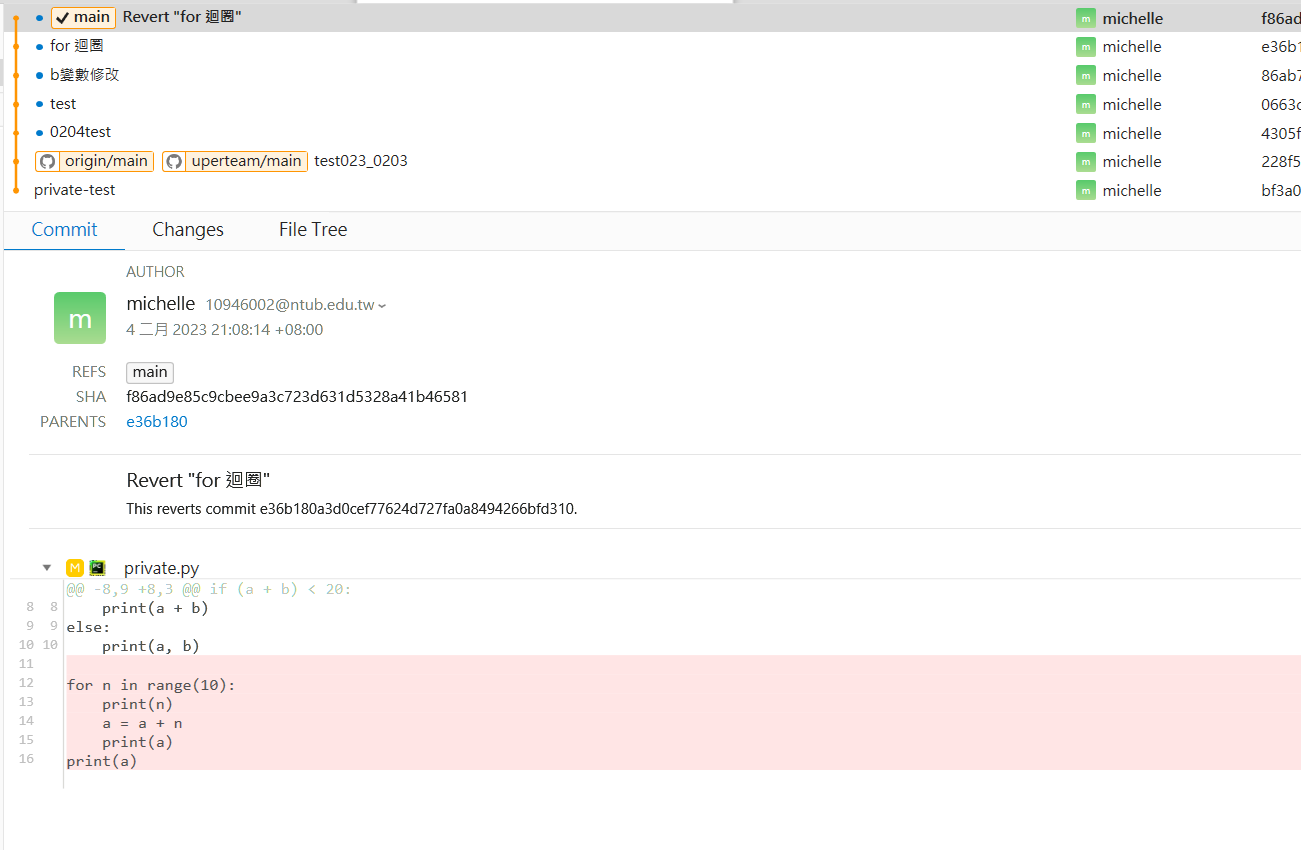
萬一本地端上傳錯了，可以利用Revert功能將commit撤銷，fork會生出一筆紀錄代表你刪除了該commit，不過要注意Revert的內容會直接刪除

選擇要撤回的commit，右鍵Revert commit

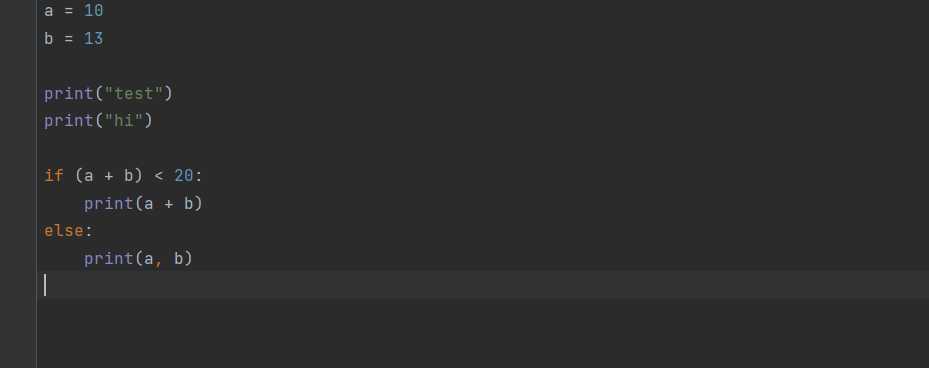




Fork會自動生出一筆紀錄，顯示刪除的內容，類似做一筆分錄抵銷錯誤的概念

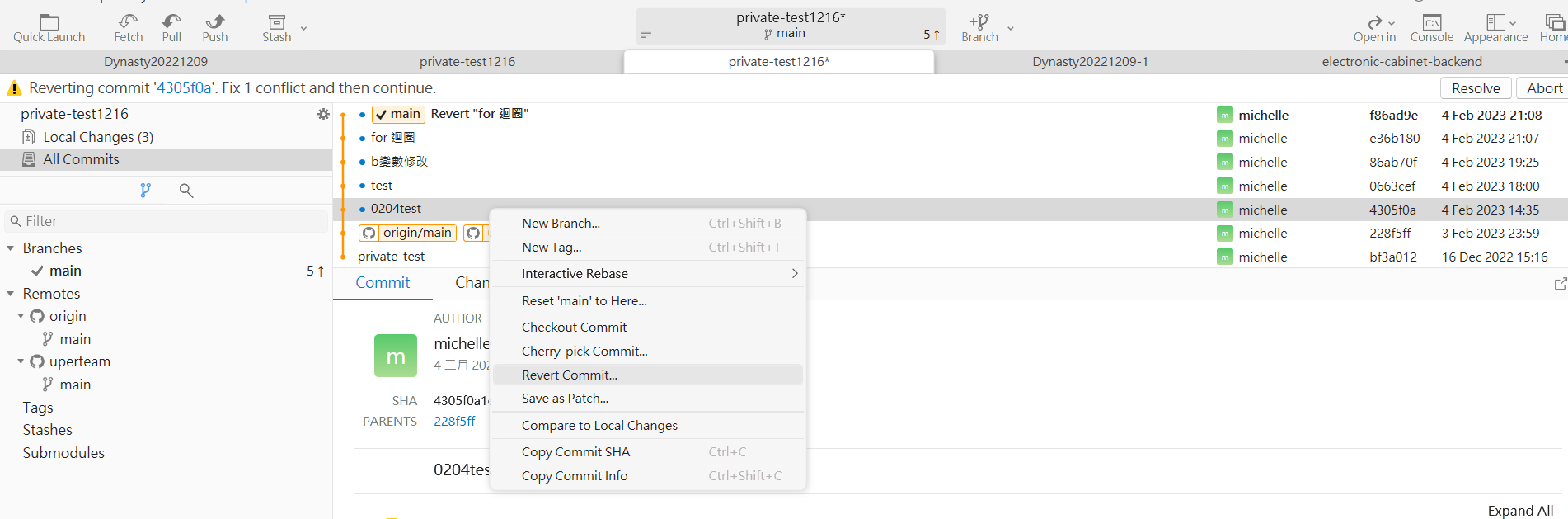


程式碼裡面FOR迴圈的部分已經消失

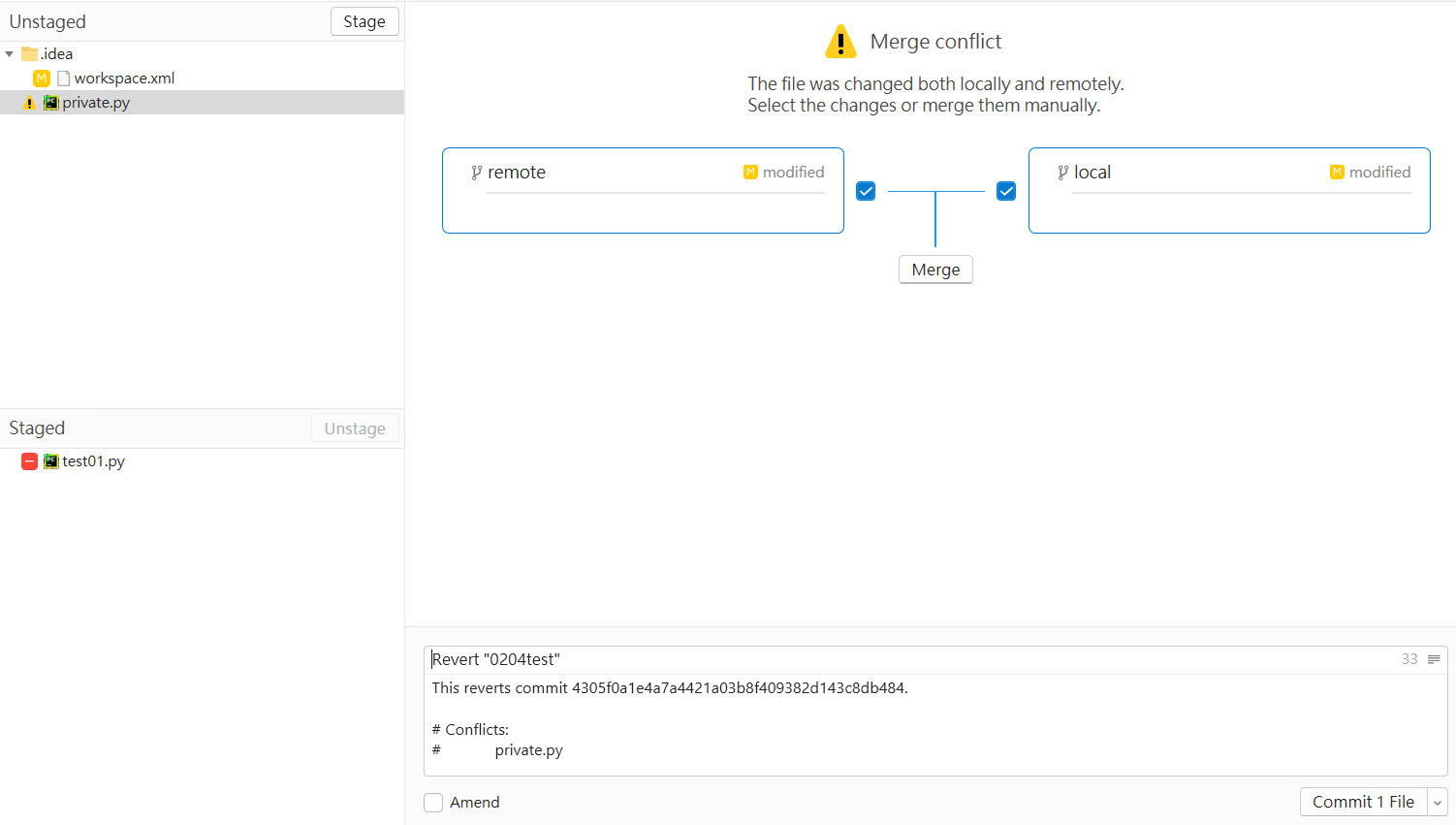


**\*若是中間的commit 要復原的話**

這邊選了中間0204test這筆，上方會跳出一個通知，代表要恢復的檔案和目前的檔案有衝突，按下Reslove。



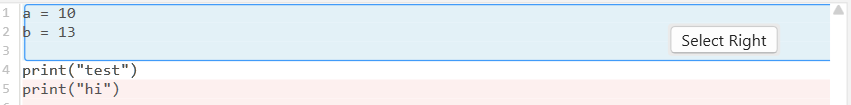
選中間的Merge



左邊和右邊分別代表目前和要回復的檔案內容，依此例子來看Remote是test0204這個commit裡面的檔案，Local是目前電腦裡面的檔案，下方視窗是兩個檔案合併後的結果，這邊可以選擇要留下哪個部分(左邊的empty或者是右邊的藍色區域)也可以兩個都保留

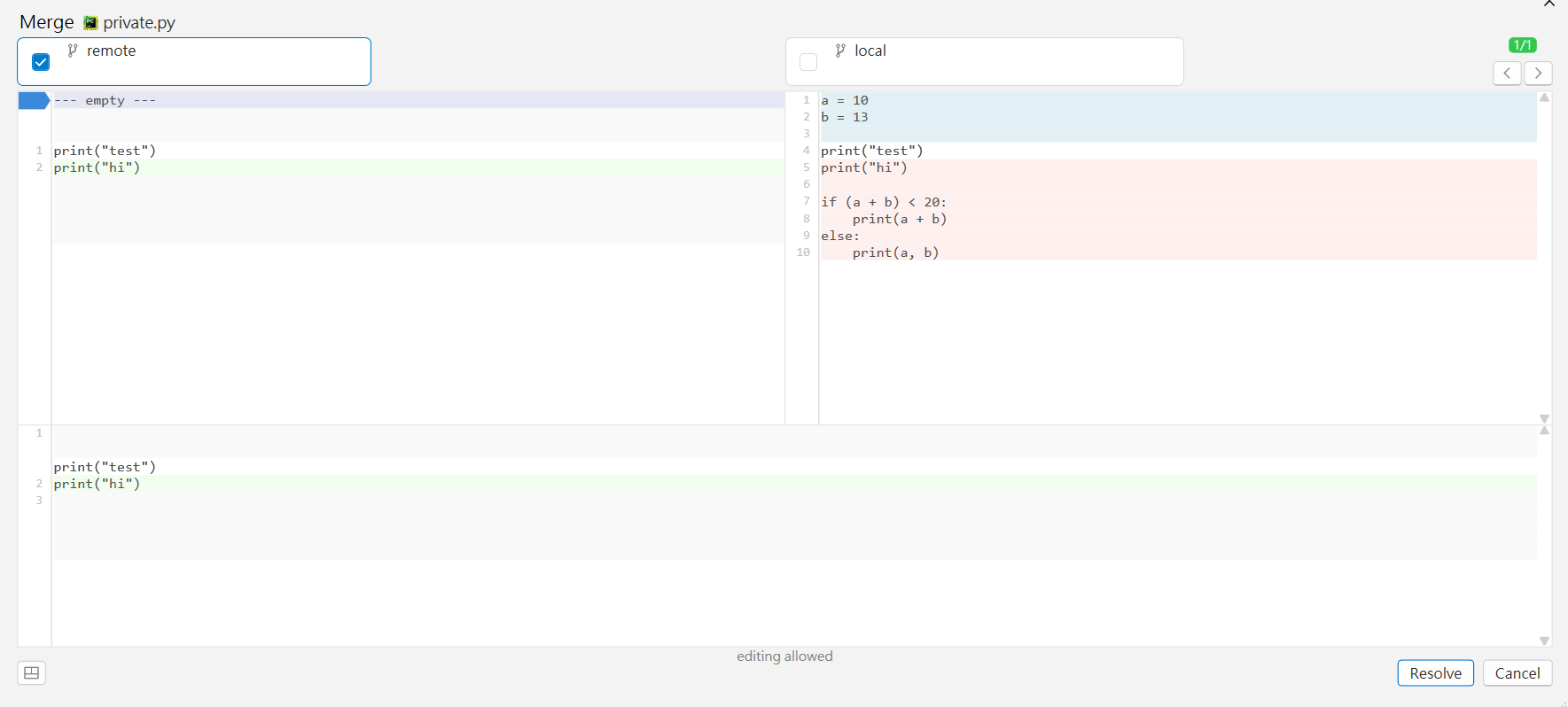


滑鼠放在藍色區域可以看到Select Right的按紐，點下去後可以選擇保留

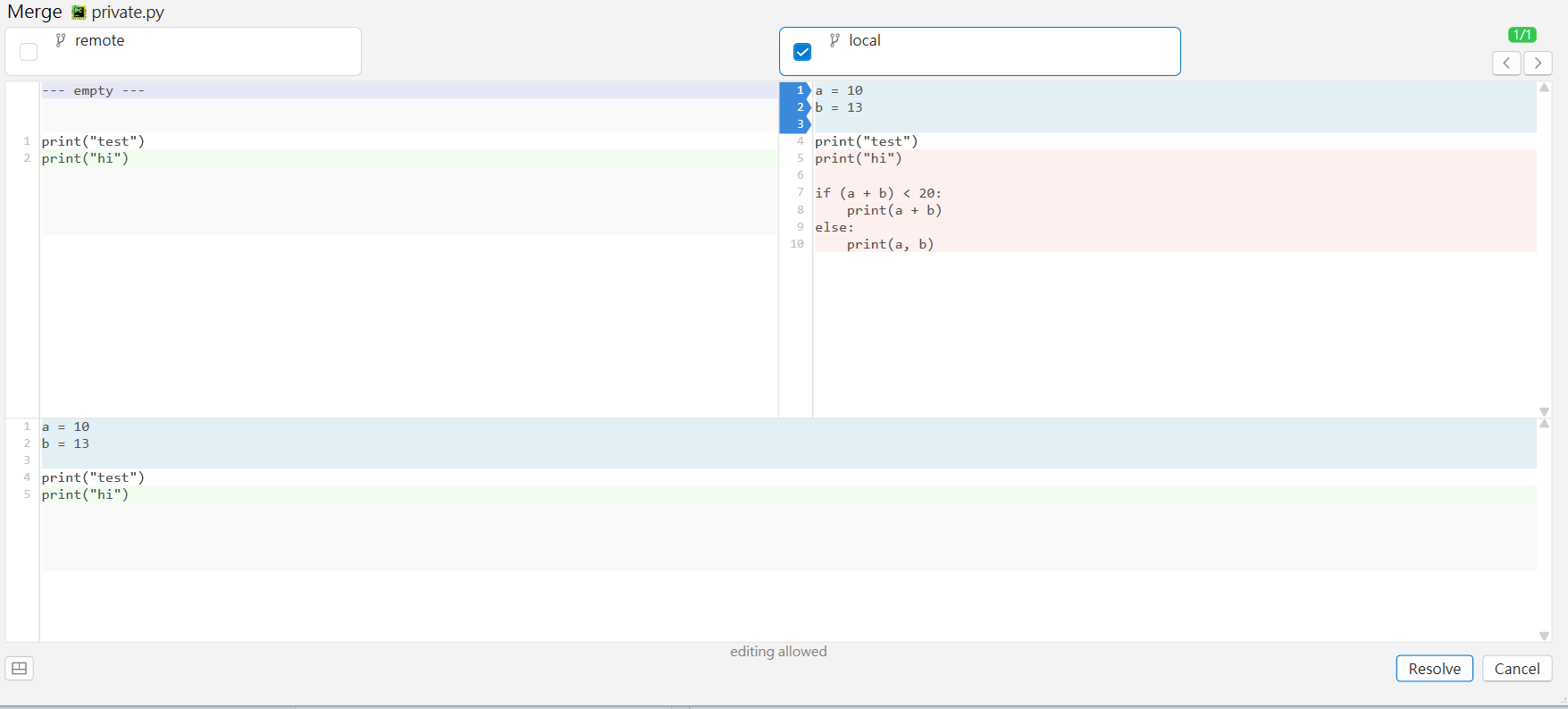


有選擇合併的部分結果可以在下方查看預覽

只保留左邊(下方預覽可以看到上面多出一列空白，灰色的數字代表第N行)



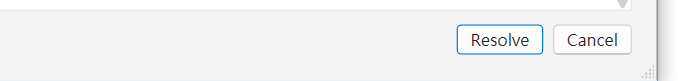
只保留右邊



兩邊都保留(多出第六行空白)

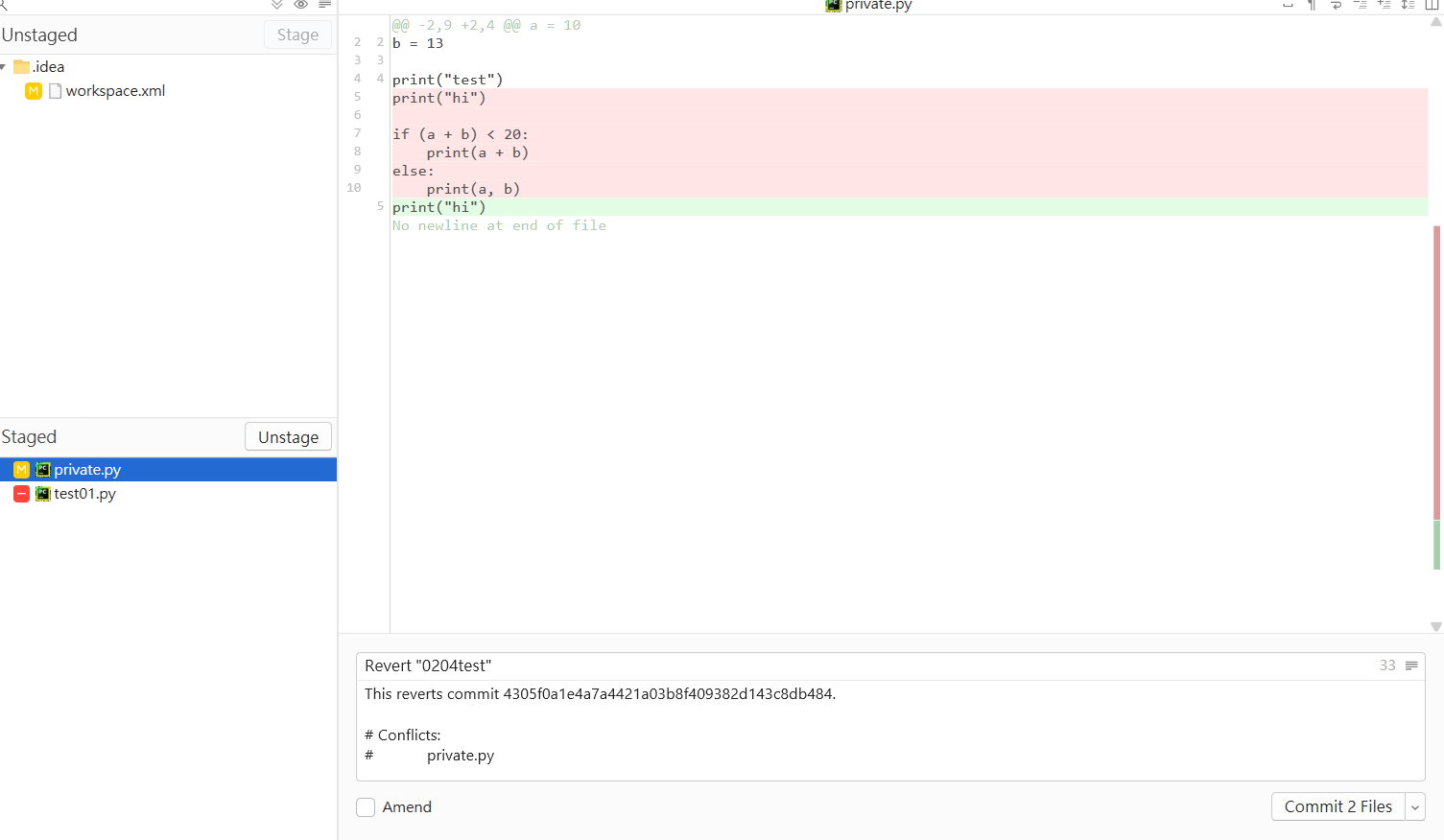


確認要合併的內容後 按下右下角Resolve

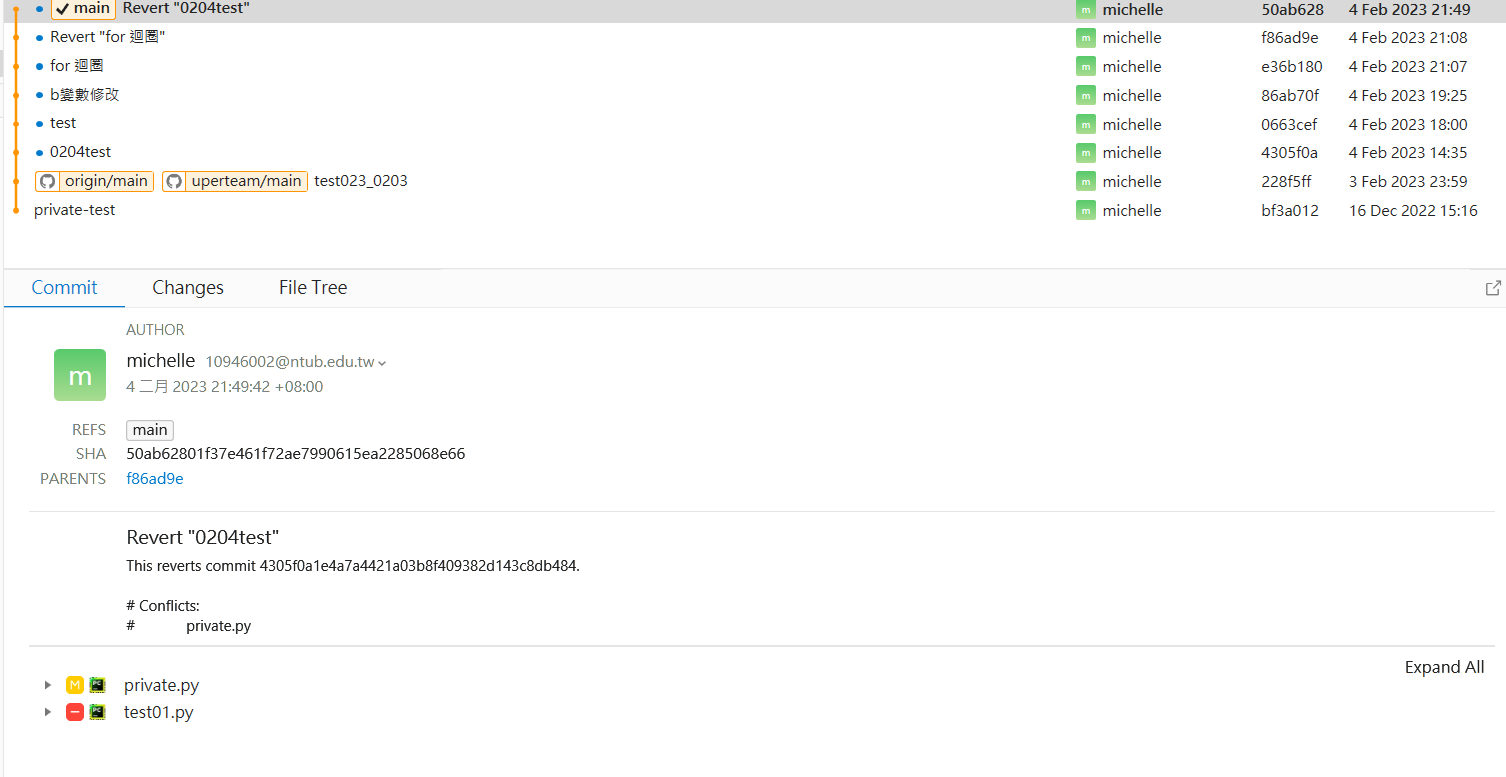


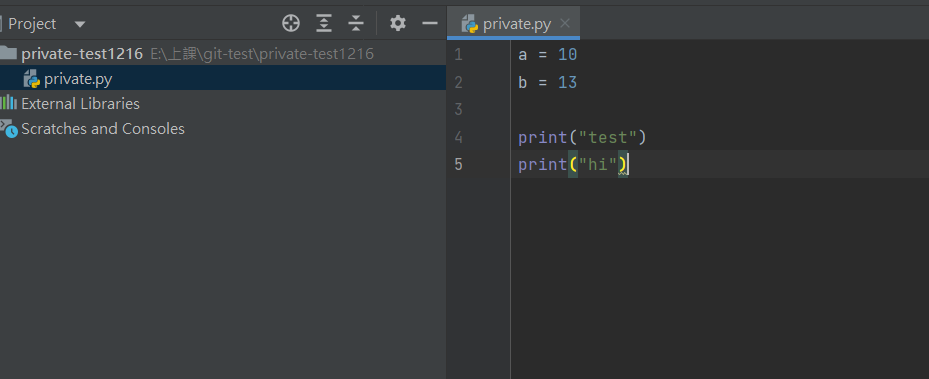
這邊會確認修改的內容後按下Commit

(test0204 這筆紀錄還沒有test01.py這個檔案所以會顯示紅色代表整個刪除)



回覆後的結果和程式碼(這邊是選保留右邊Local的部分)





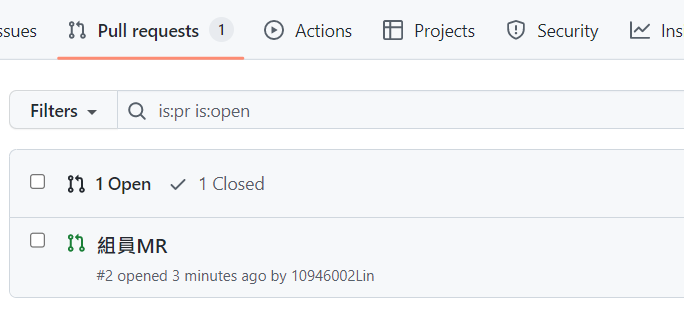
# 合併檔案操作

檔案需先上傳到個人雲端才可以合併到共用端了，需要去github發請求，需要該專案的建立者也就是專案負責人審核

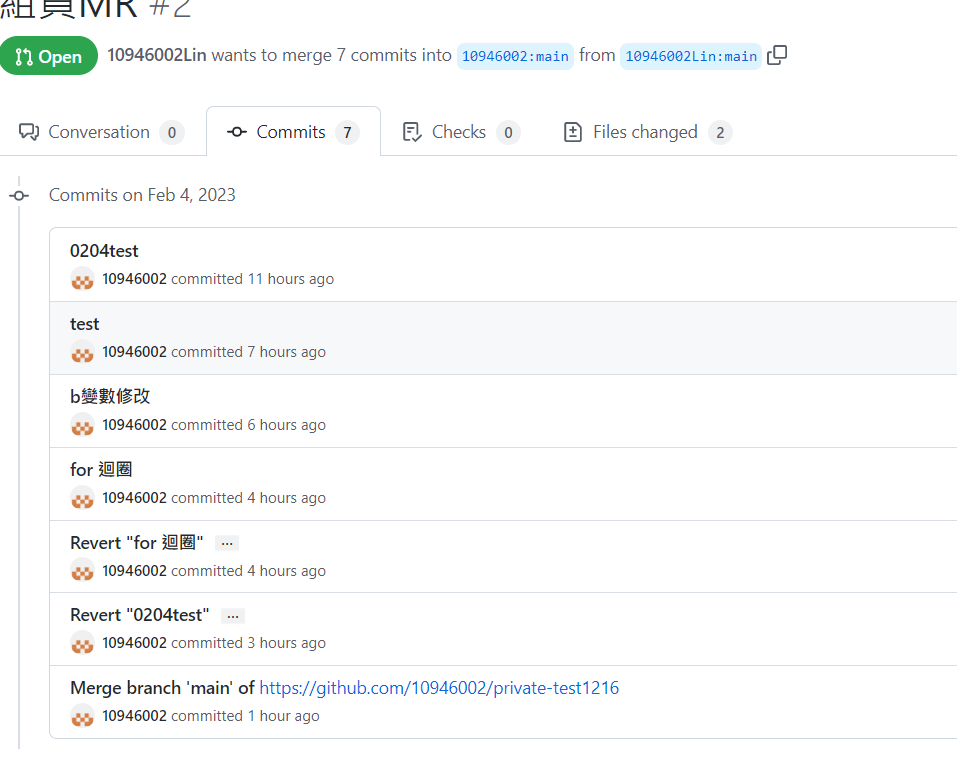
## 專案負責人(審核者)

### 審核Merge

組員會通知自己發了MR，到pull request裡面可以看到所有的MR



Commit裡面可以看到該次所有的commit可以逐一點進去查看程式碼



可以點右邊的Review changes輸入討論問題

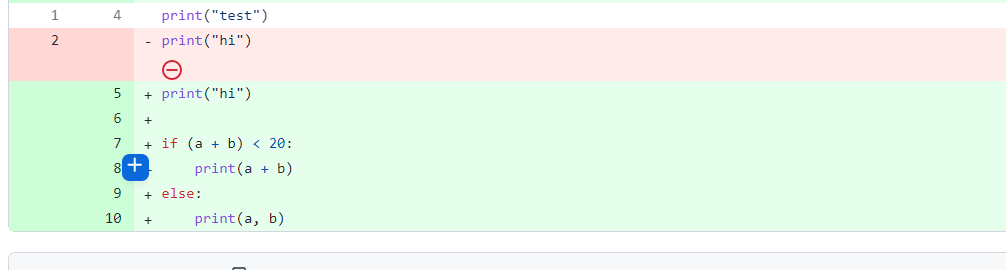
Commit問題討論

Approve此項修改通過

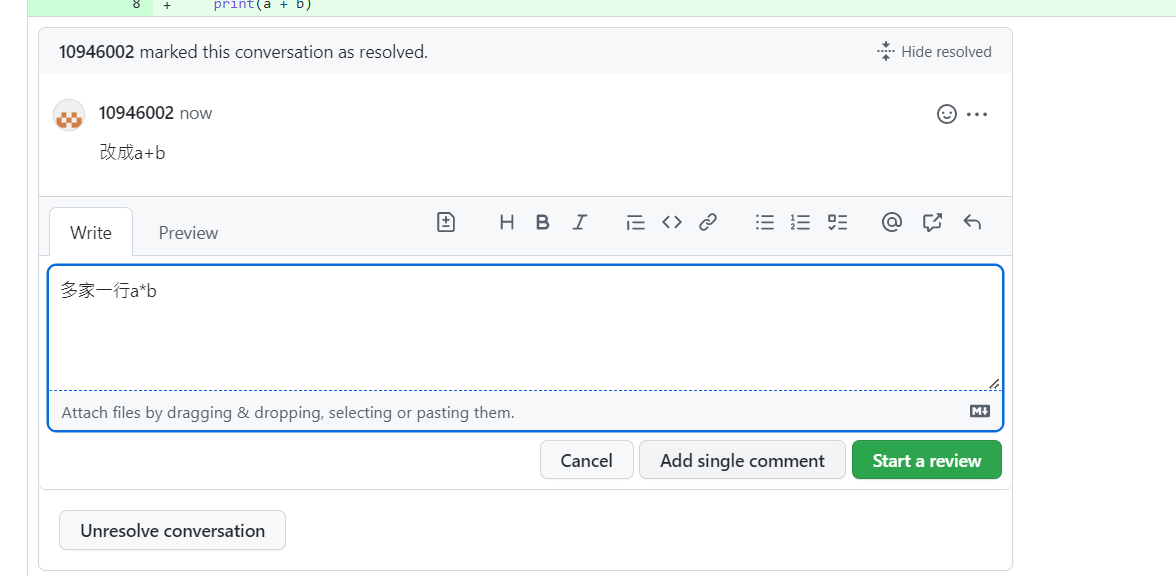
Request change 此項需要更改



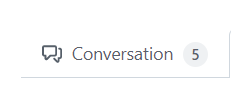
也能針對單項程式做建議事項，將需要修改的程式碼前面有個藍色+號，點下去輸入需修改的問題

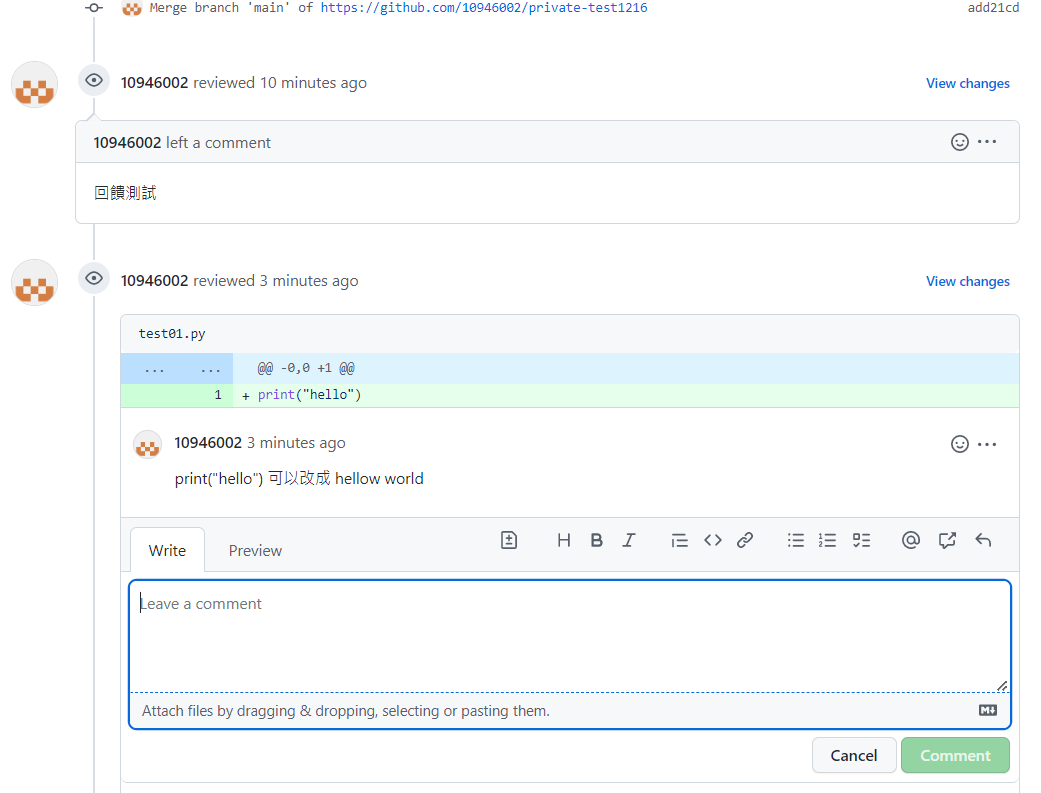


點Add single comment

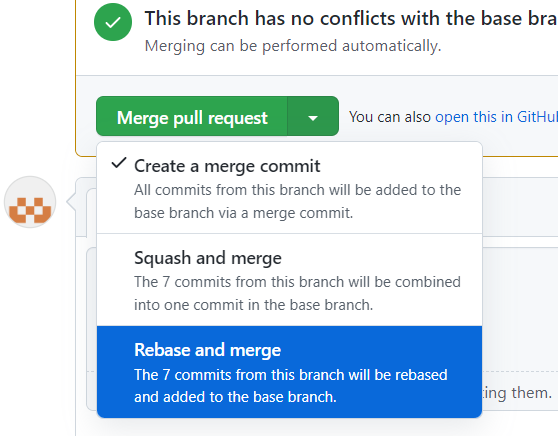


回到第一頁Converastion可以看到修改的項目

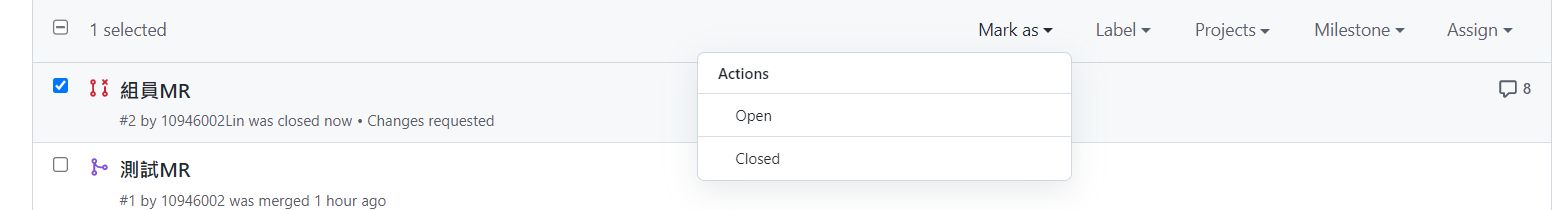


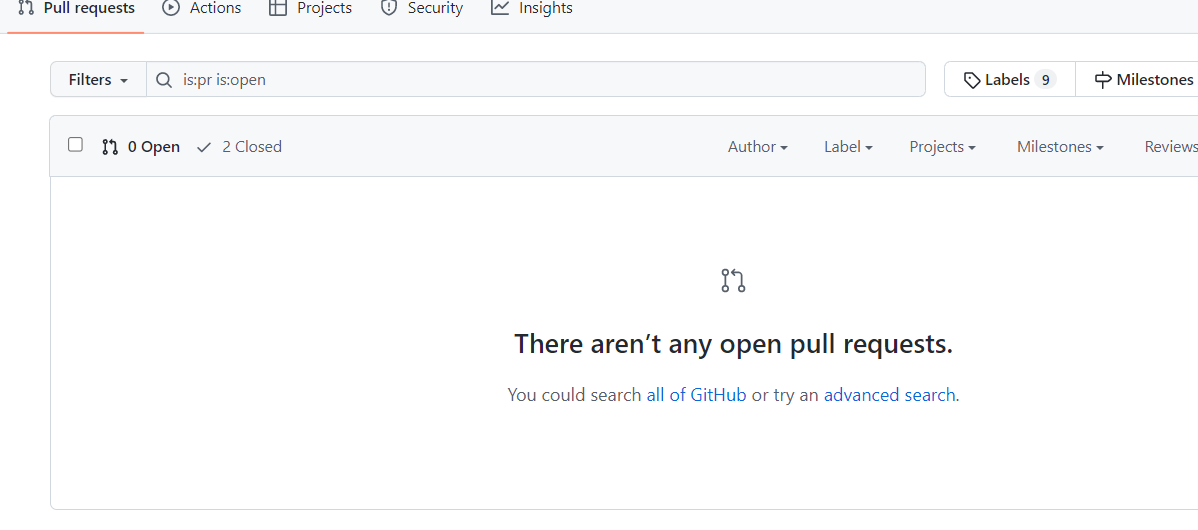


如果覺得這次的內容需要再修改可以將Me退回去

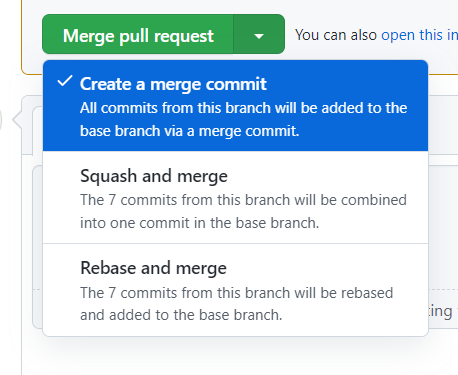


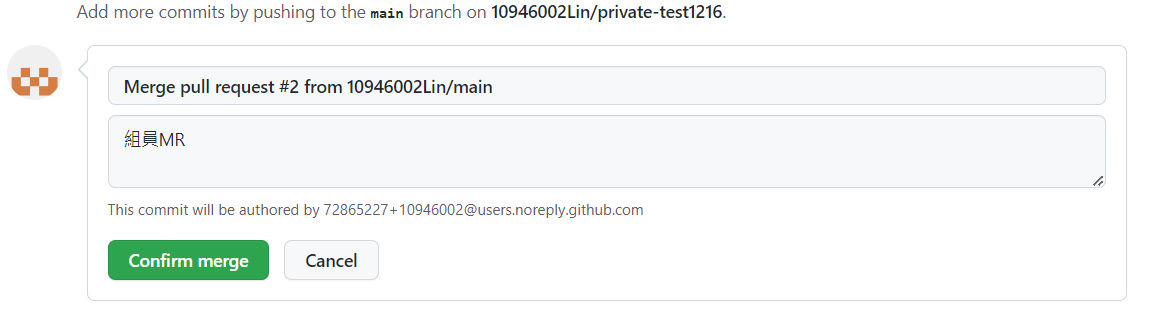
也能將MR關閉，選擇要關閉的MR，Make as 裡面有開和關的選項，關閉後此筆MR扔可以查看但是無法審核通過(可以重新打開)



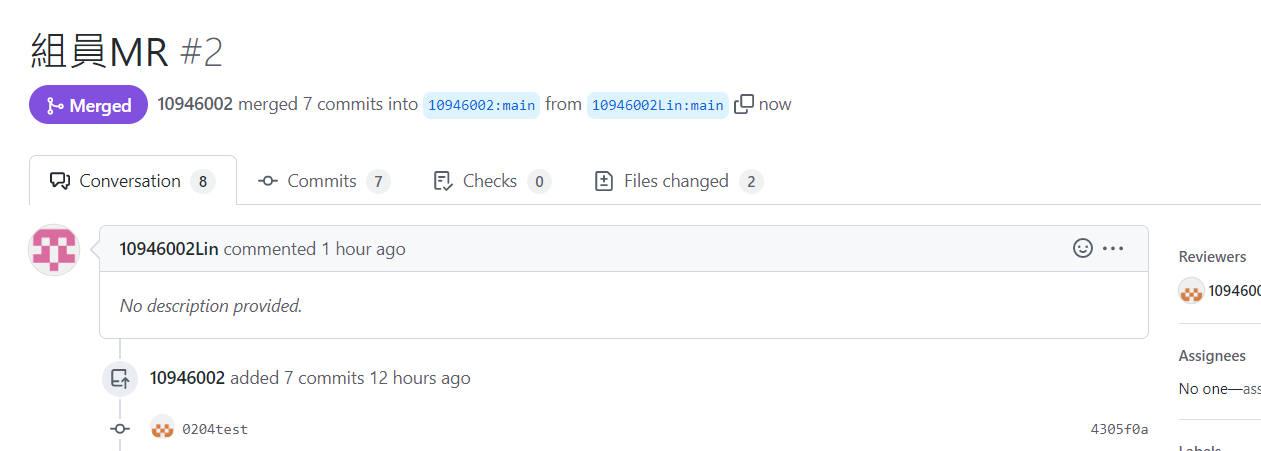


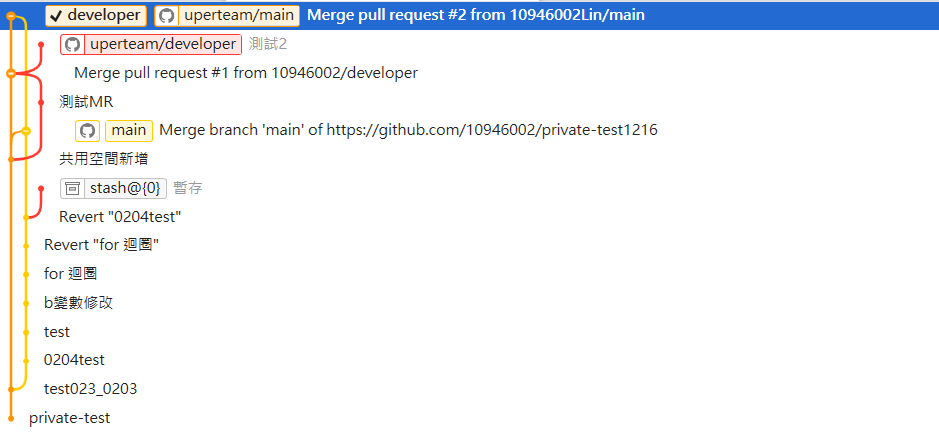
都修改完畢沒問題後，就可以審核通過了



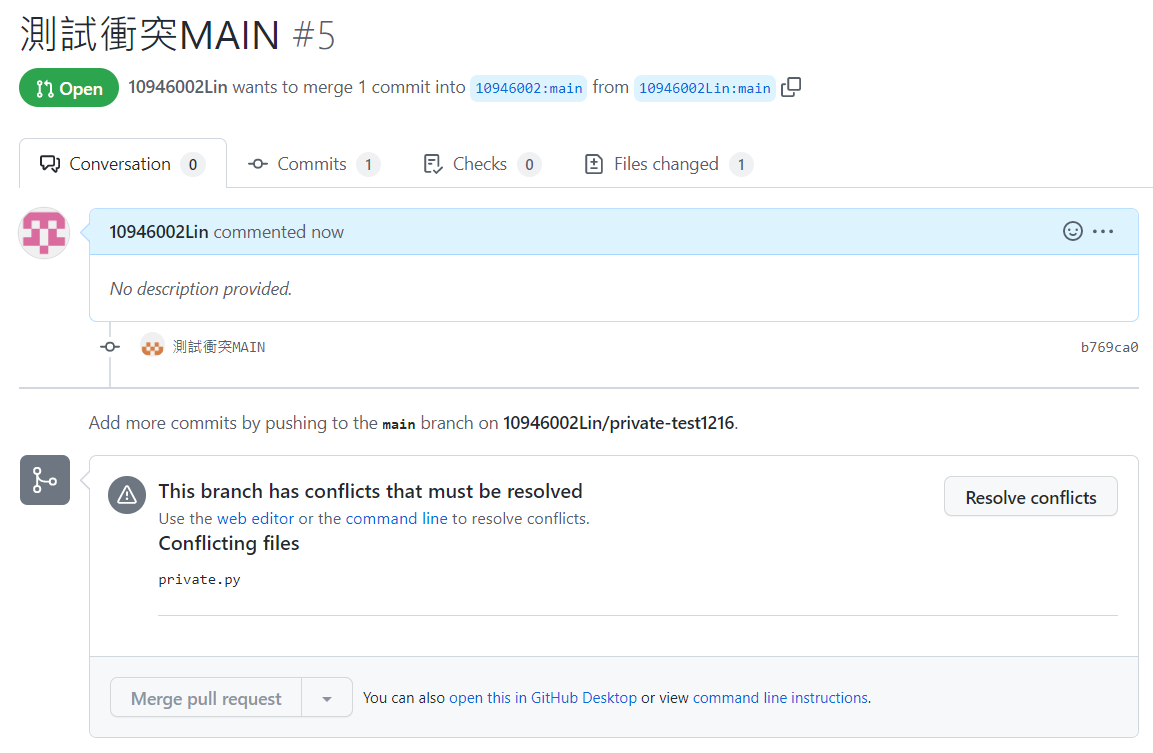


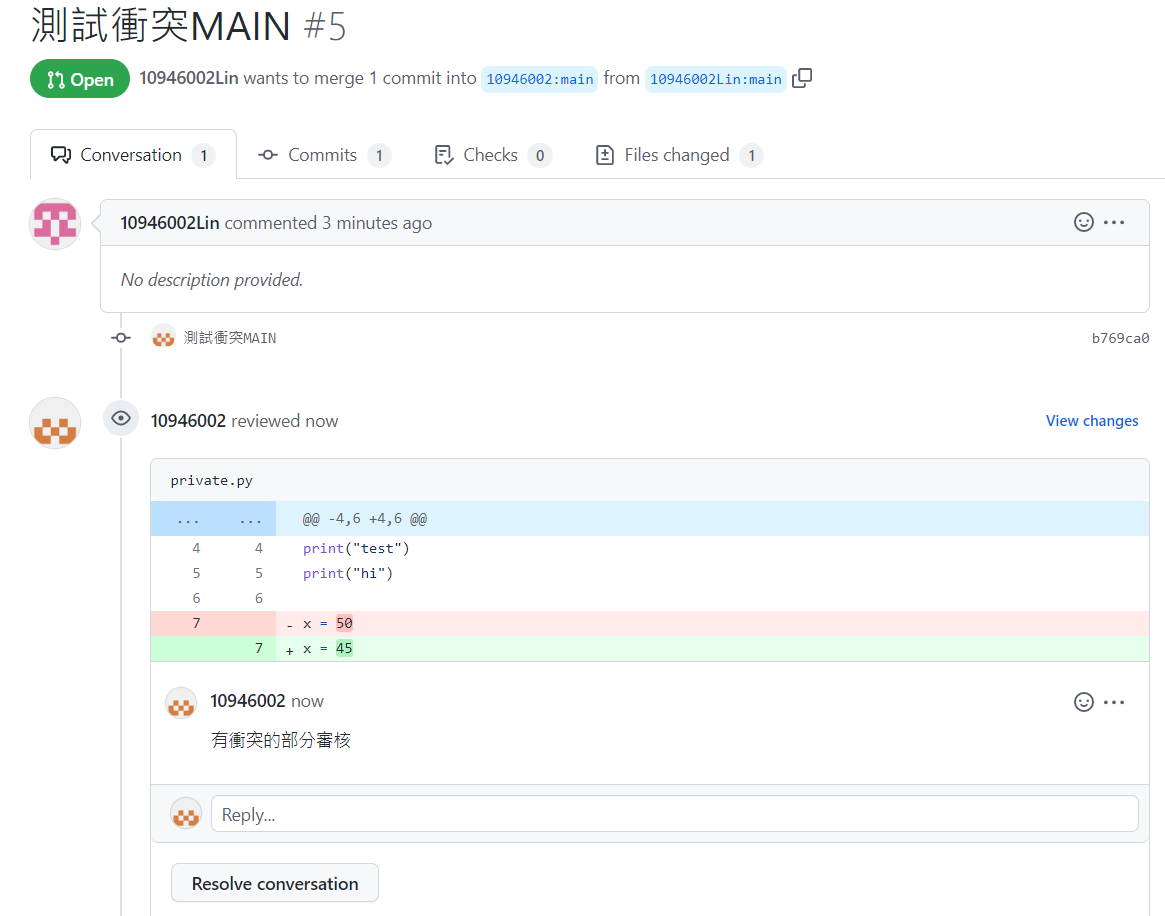
合併成功的MR狀態會改成紫色，就能通知大家已經通過可以更新檔案了，Fork也會有紀錄



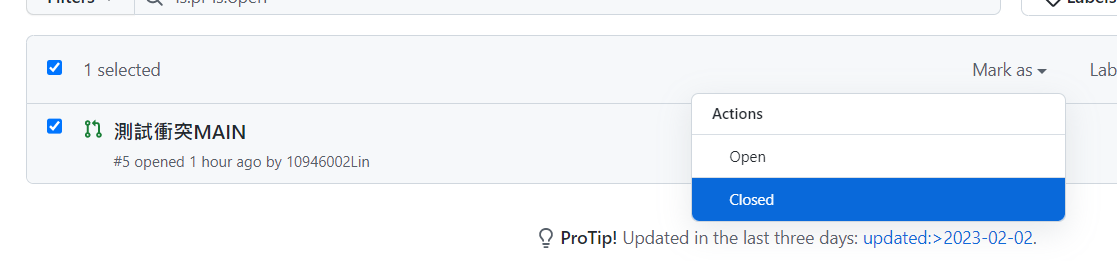


處理有衝突的MR

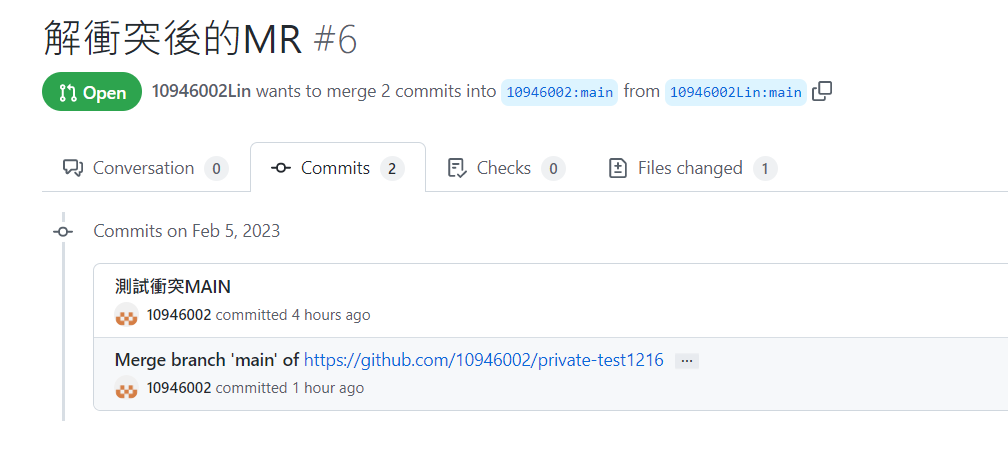
收MR時有時候會出現不同人改了同一段程式碼的問題，如下圖，這樣沒辦法執行合併不過一樣可以先審查程式碼，步驟和上面相同

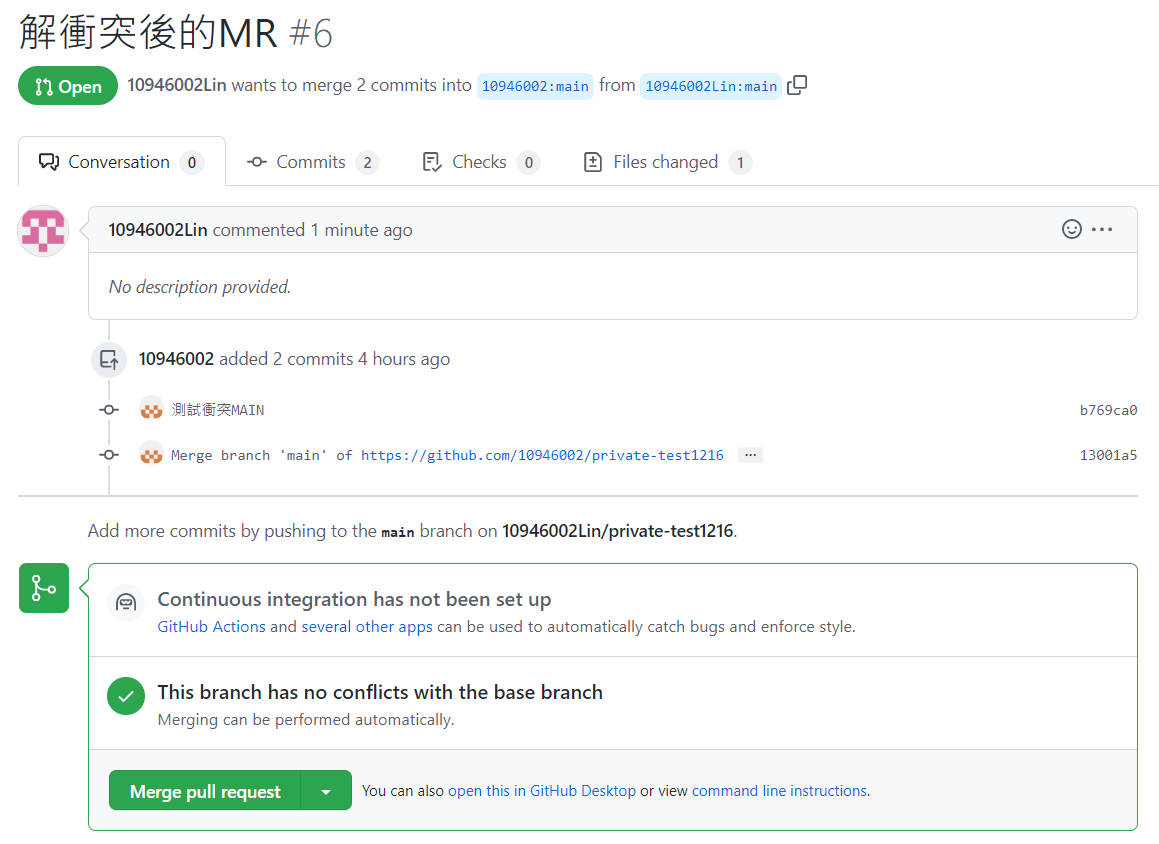


審核沒問題後要通知組員，請兩方協商要留下誰的檔案，此筆MR要關起來避免後面的檔案無法合併。(此筆MR並沒有包含組員商議後最終的修改的Commit所以無法通過)



重新收到的MR，包含衝突和解除衝突的檔案，審核過就能通過合併了





衝突已解決

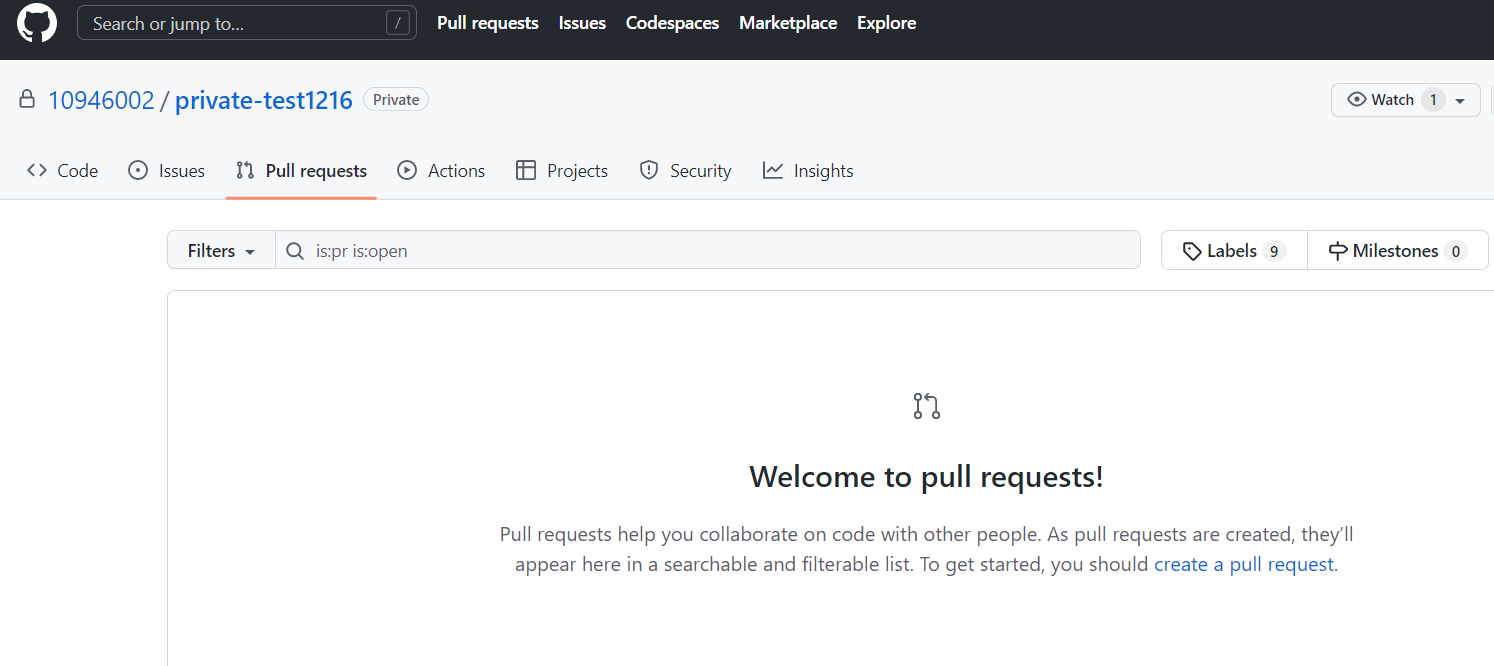


## 開發人員

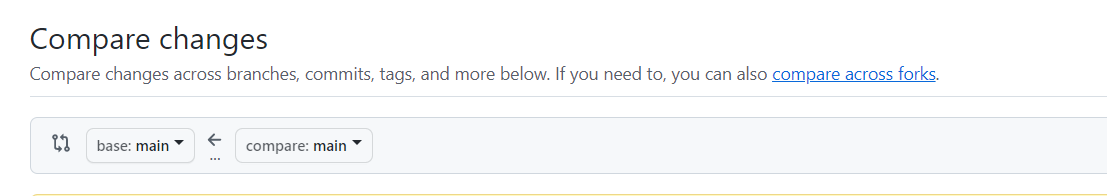
### 發Merge

去github上面發請求將個人端檔案合併至共用端。

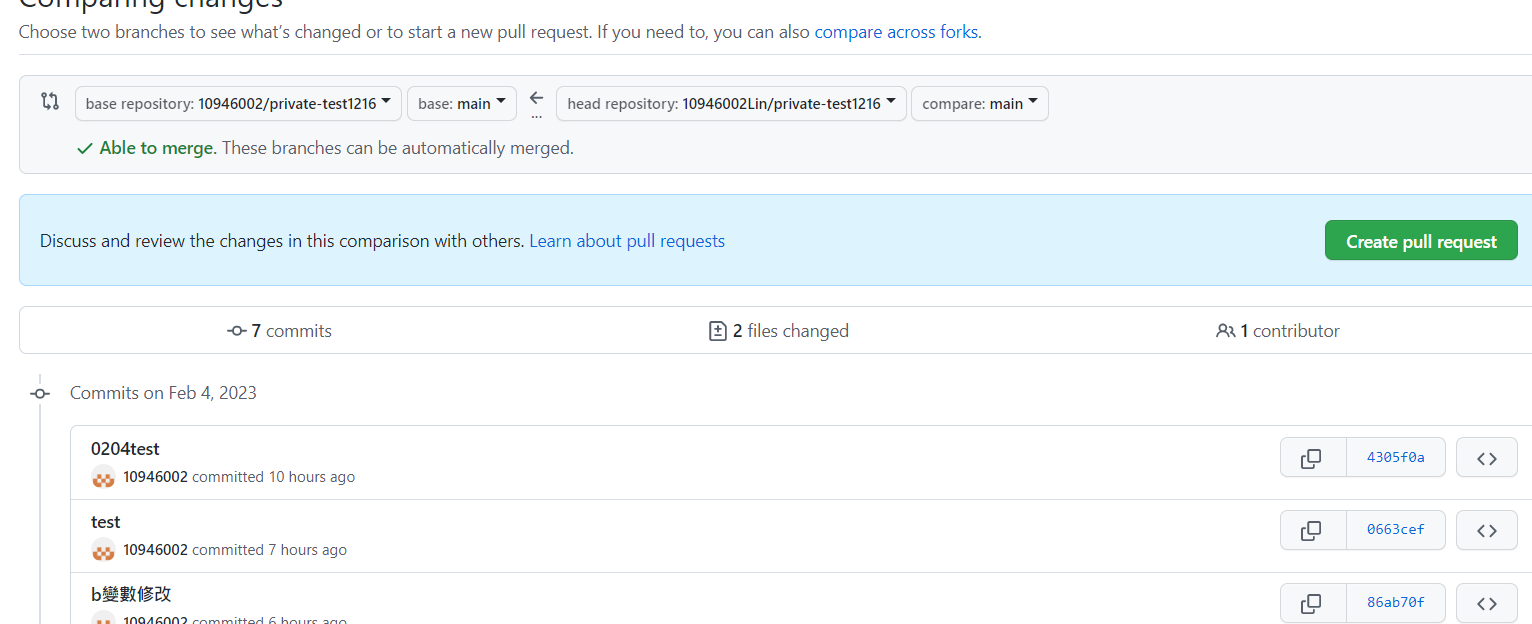
到**共用端的專案，**選擇pull Request，選create a pullrequest



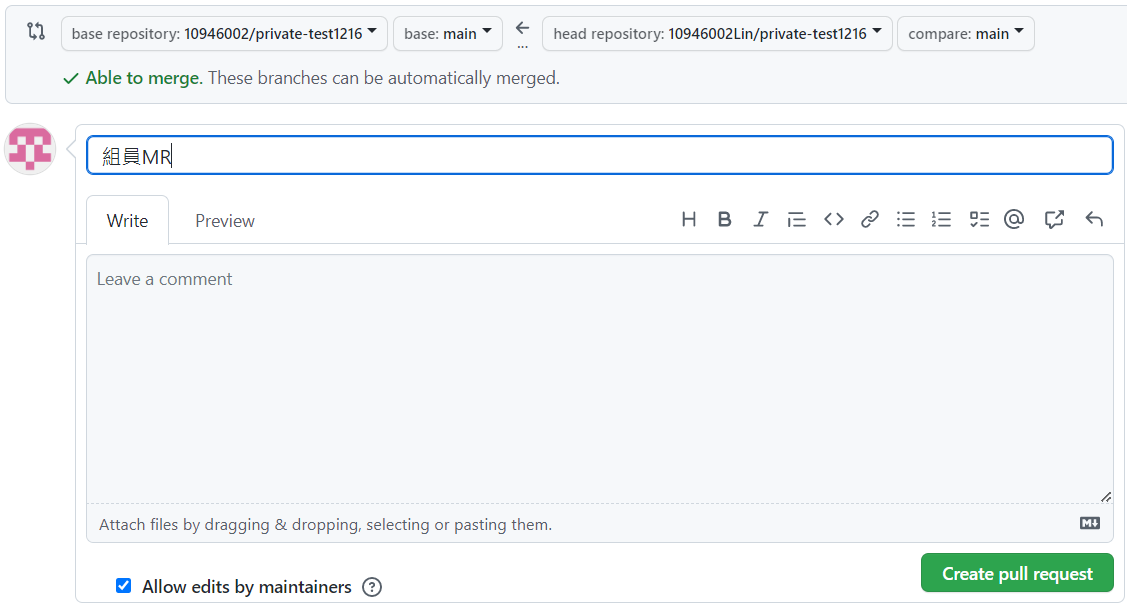
選擇compare across forks

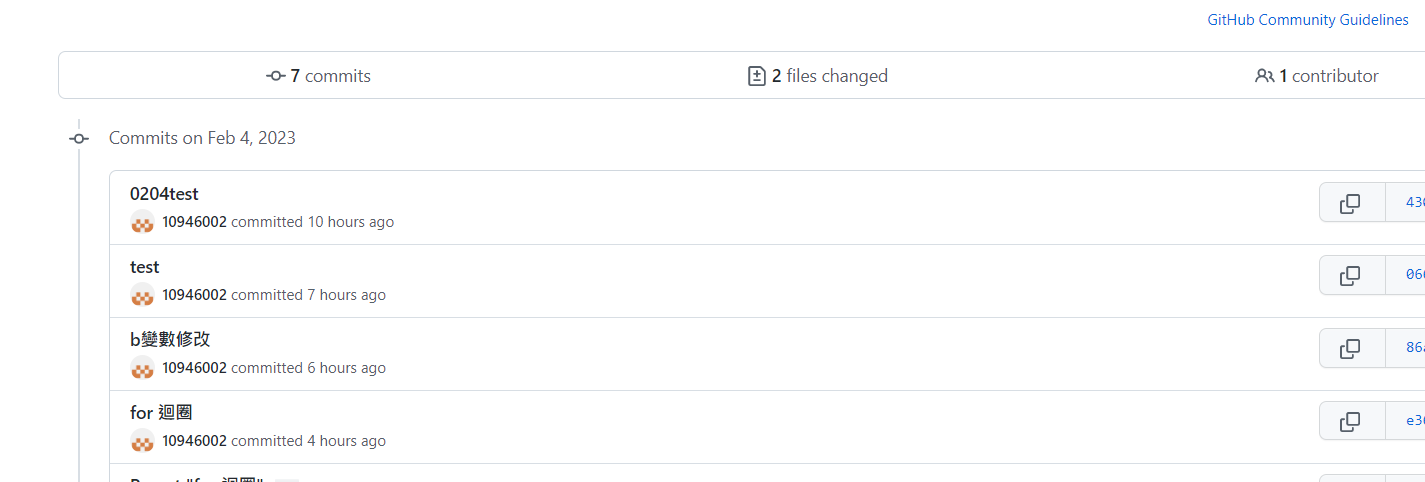


左邊選自己的個人雲端和其分支，右邊選共用端的專案和分支**(注意共用端的分支要是開發中的那個不要選到正式上線的那個)，**選完成後按create pull request

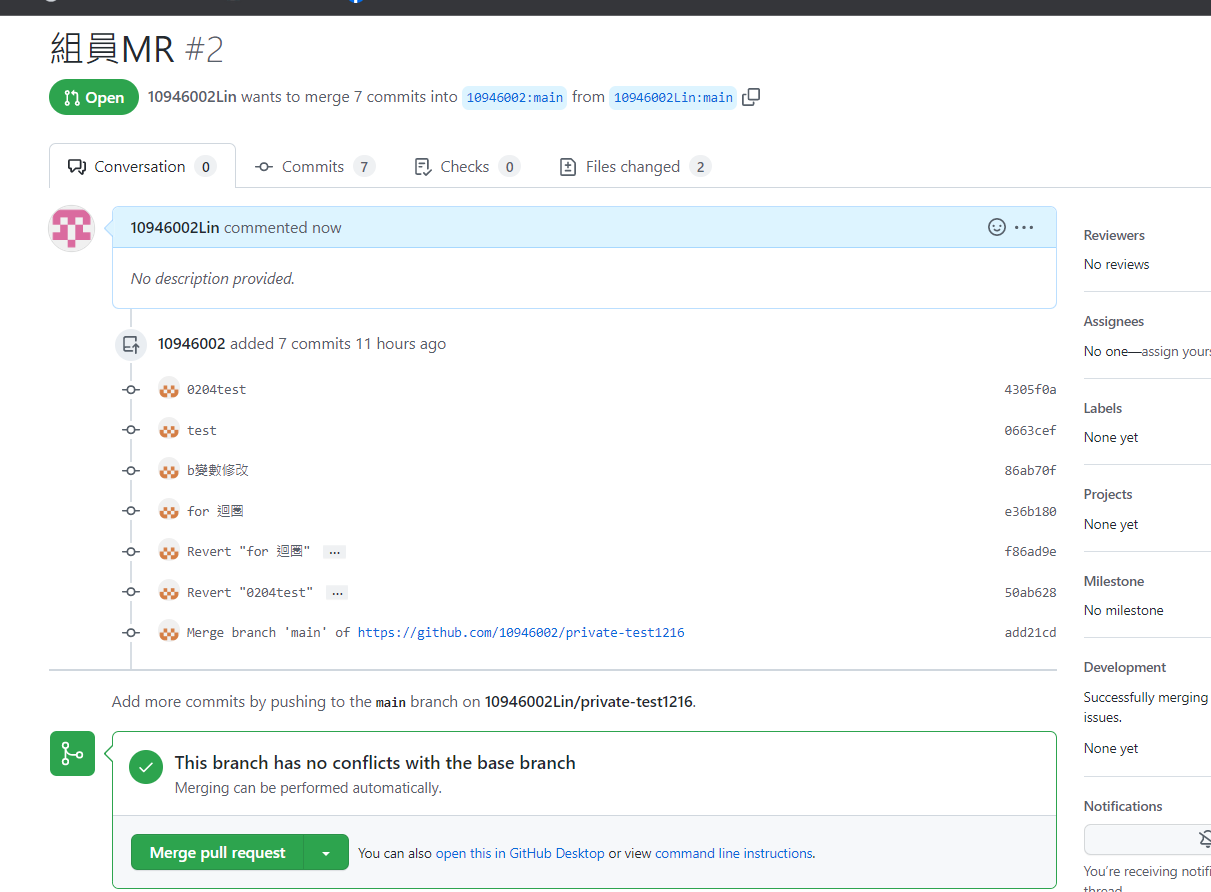


打上這次合併的標題名稱(必填)，可以參照commit打的標題去寫，字太多可以寫下面write裡面，頁面下方有本次的Commit可以看，大完後按下create pull request

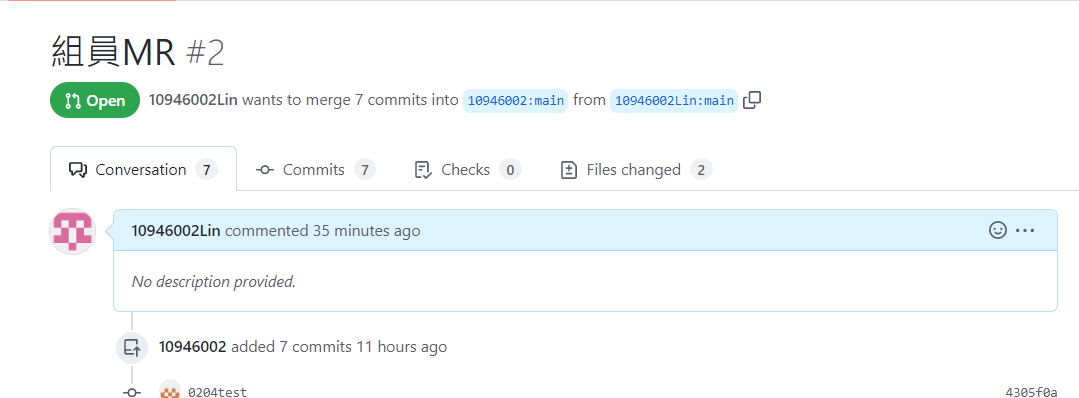




到這個頁面後，通知審核人員審核，**不要按通過，按了就相當於併上去了**



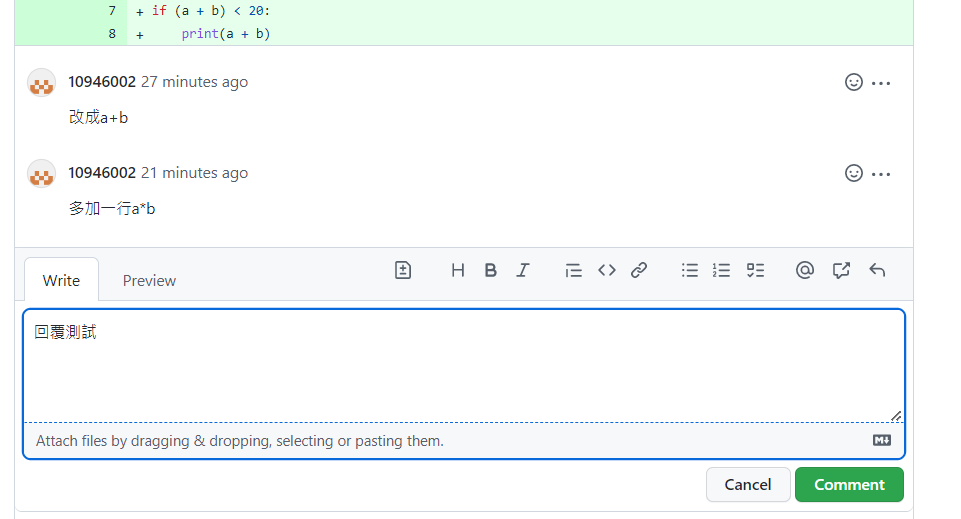
如果審核人通知有修改可以回到頁面的conversation，查看修改事項，點開可以看到細項

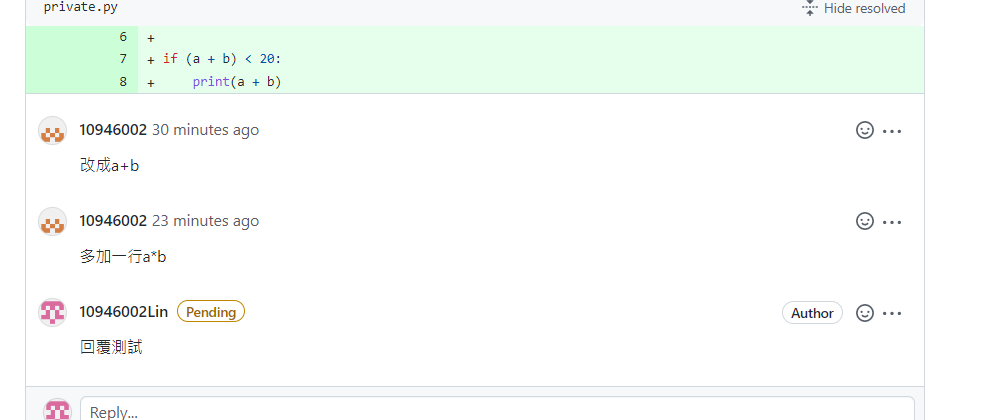




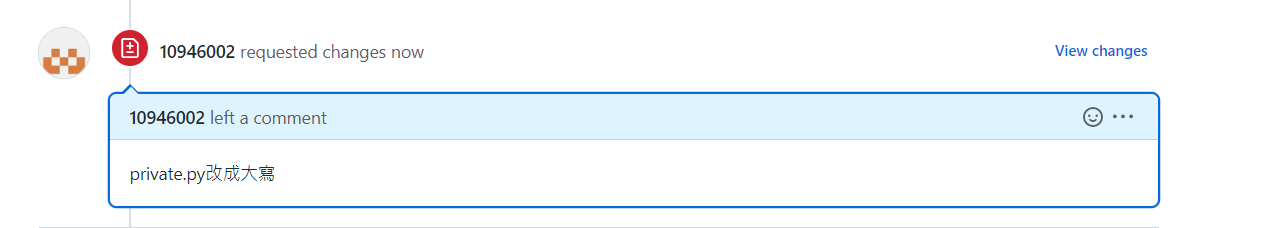
可以打字回覆或只按下Resolve代表已經解決



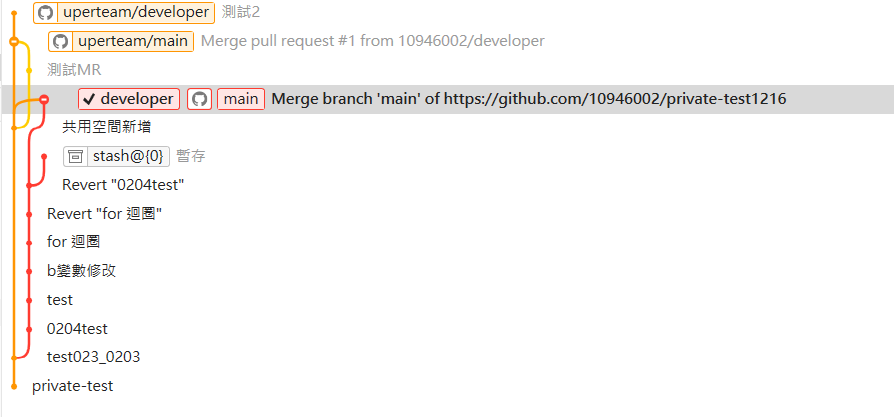




如果是這種的代表該黨按需要修改



很不幸若MR未通過的話，Fork裡面會有紀錄

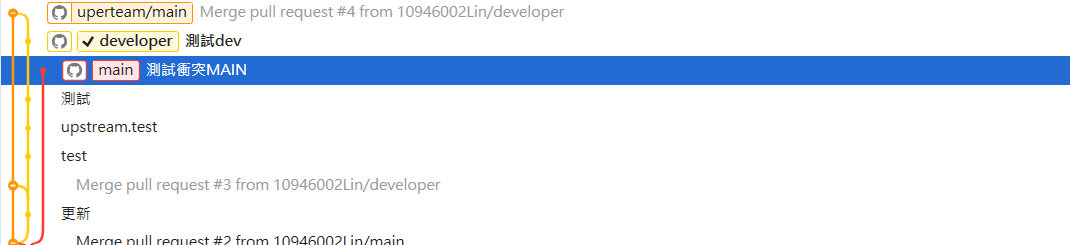


**因為請求不通過，所以分支變成紅色也和upstream分開了。**

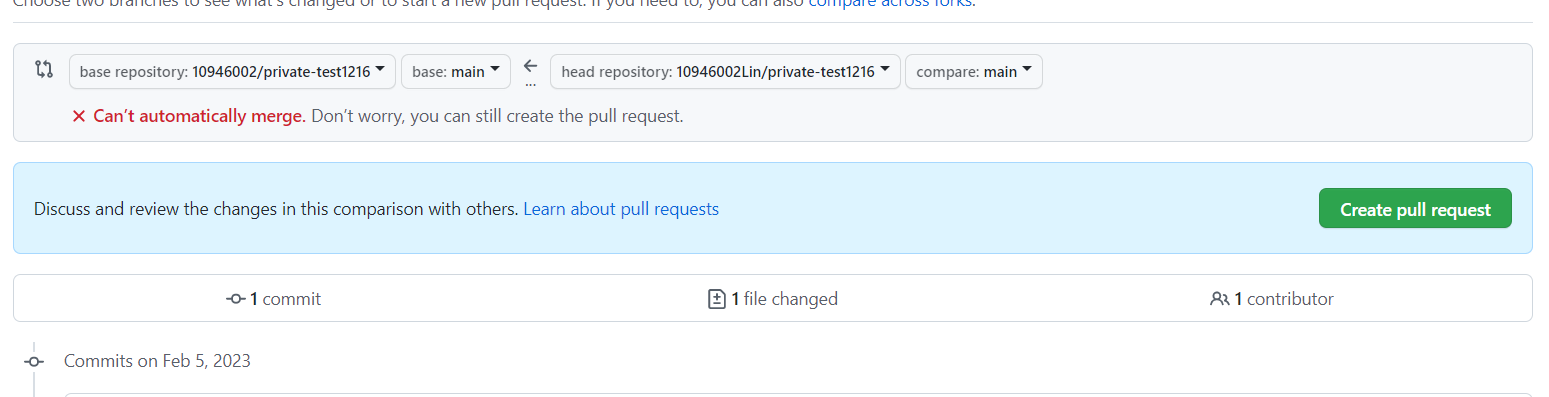
檔案修改好後，將修改過的commit上傳到個人雲端再發一次megre，請審核人員查看。

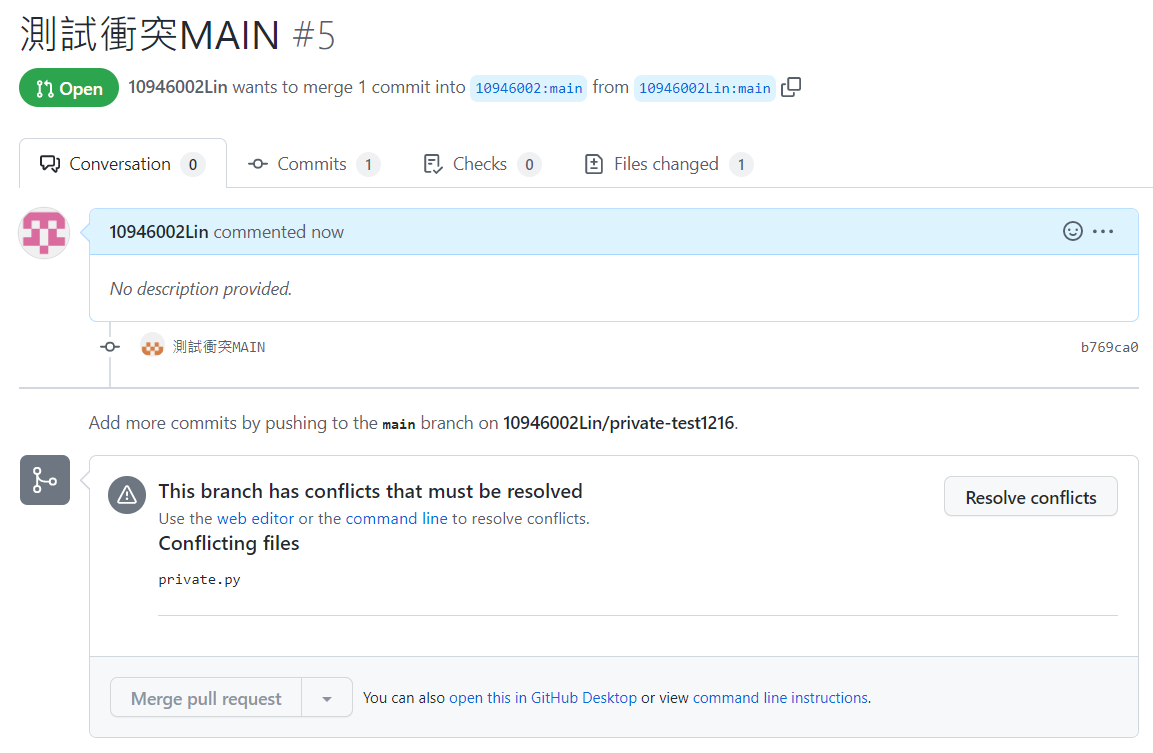
### 解衝突

要合併專案時，兩個人改了同一段程式碼的時候，要合併時就會發生衝突，大多情況長這樣，黃色和紅色代表不同的組員，現在他們要全部合併到共用端(橘色)裡面，這時候紅色那位組員去github發MR就會出現錯誤

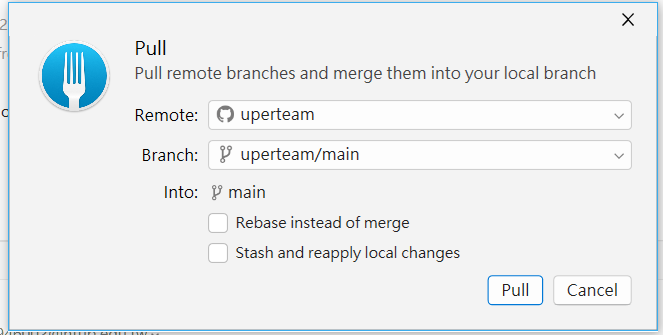


這時候先按下create pull request ，先讓審核人員看一下自己的程式碼有沒有問題之後找到有修改到一樣地方的組員要留下誰的程式碼，**務必要溝通清楚**

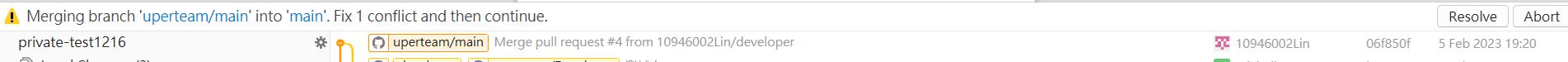




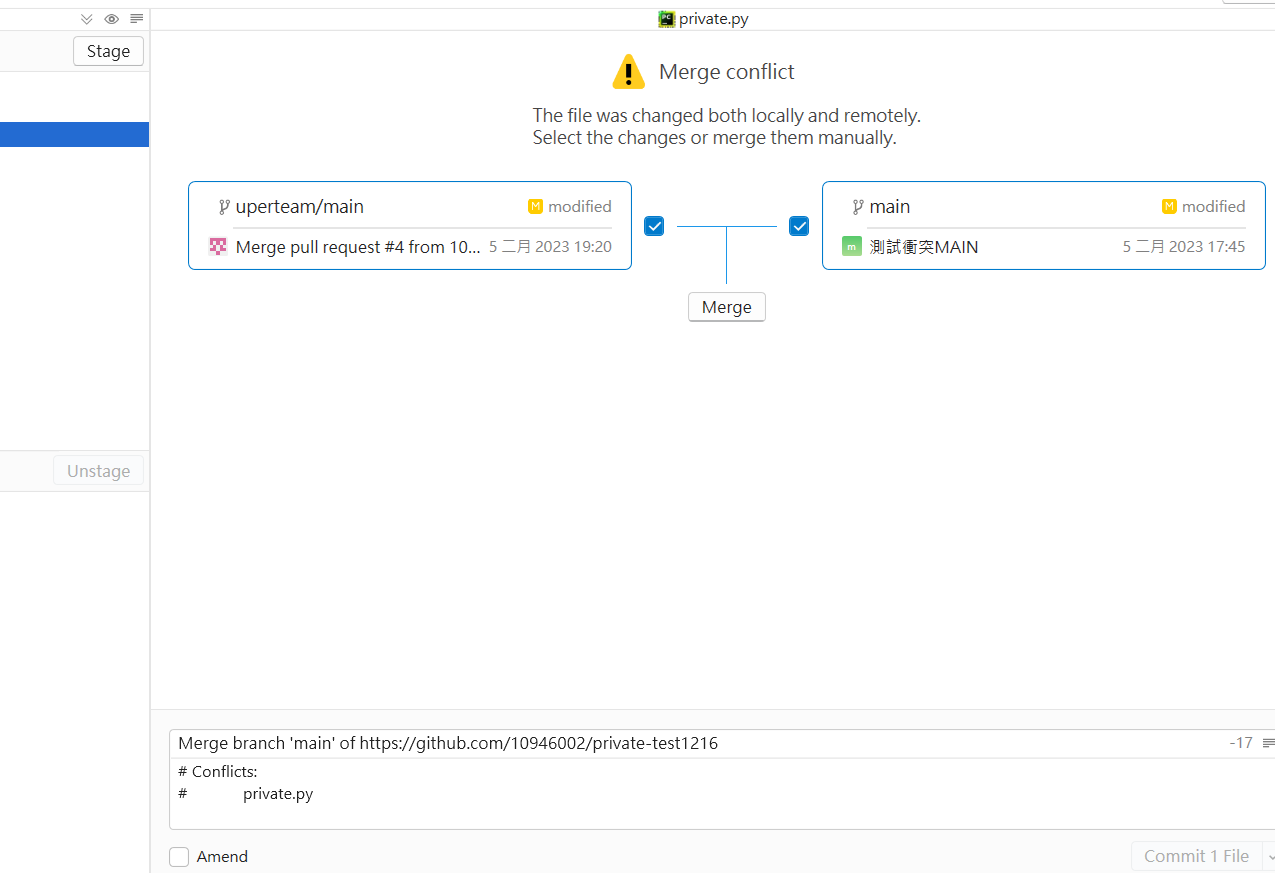
回到Fork中要記得去pull共用端地的內容



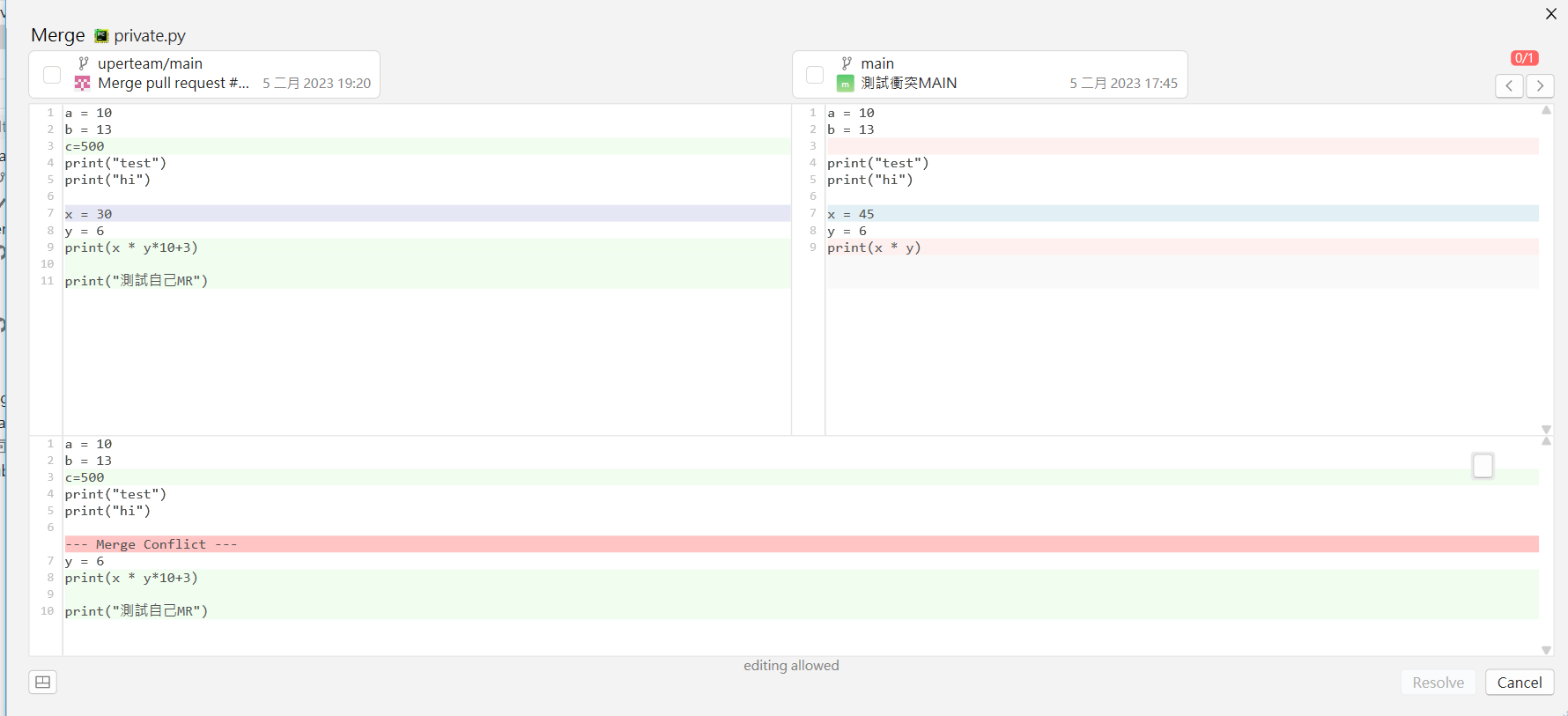
完成後會看到跳出訊息，按下Resovle



按下merge



視窗左邊是共用端的檔案，右邊是組員的檔案，這邊可以選擇要留下誰的程式碼(紫色和藍色的部分)



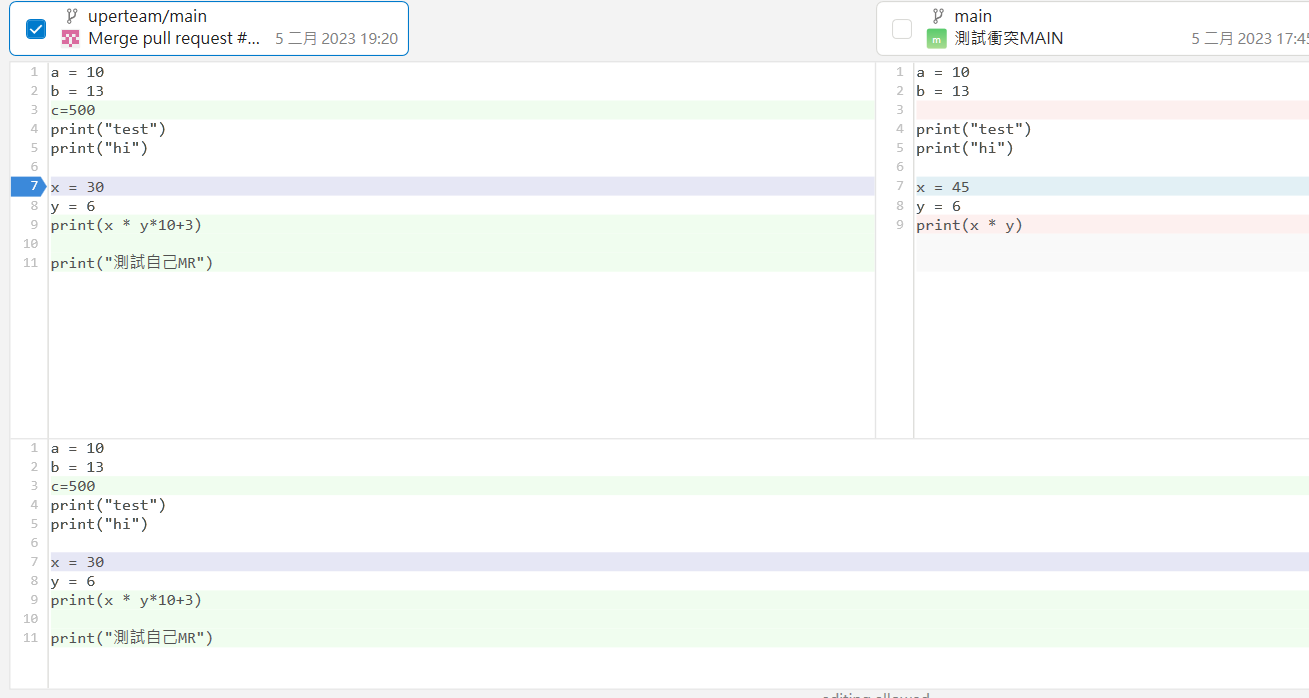
合併後的樣子預覽

本地端

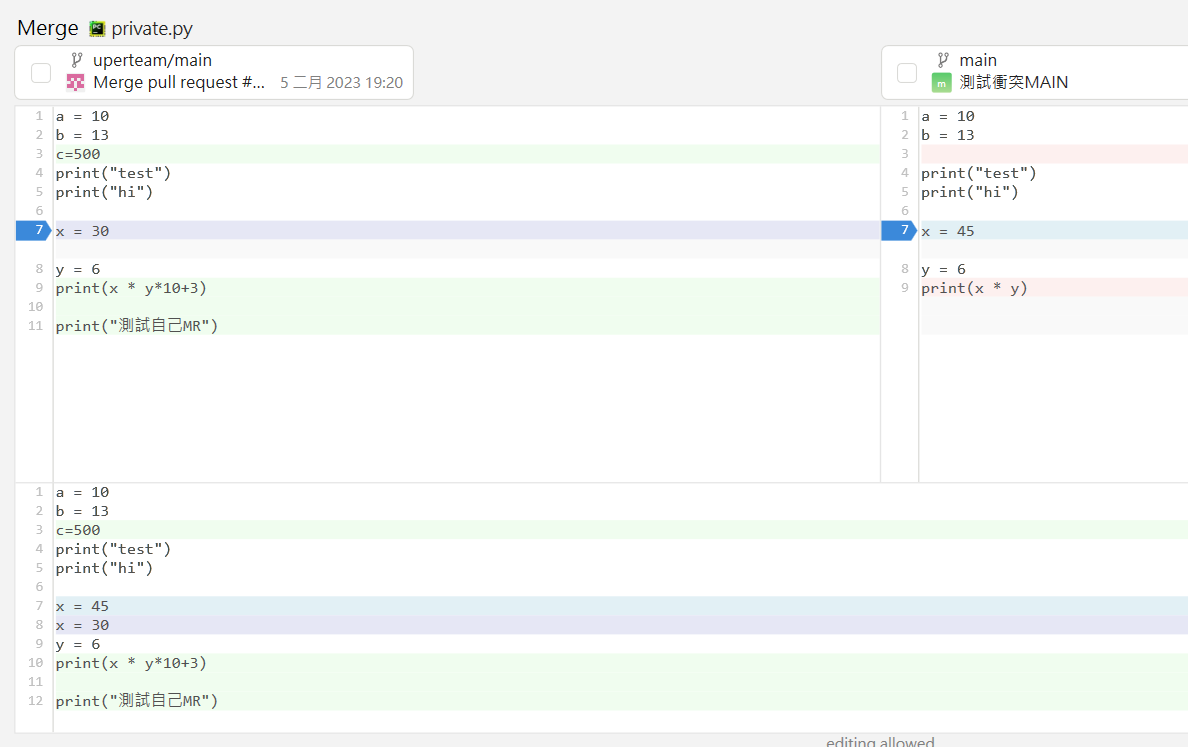
Upstream共用端

可以保留一邊，也能兩個都保留(這樣程式會出問題，最好不要)

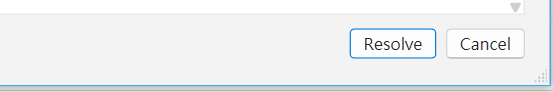
保留左邊(x=30)



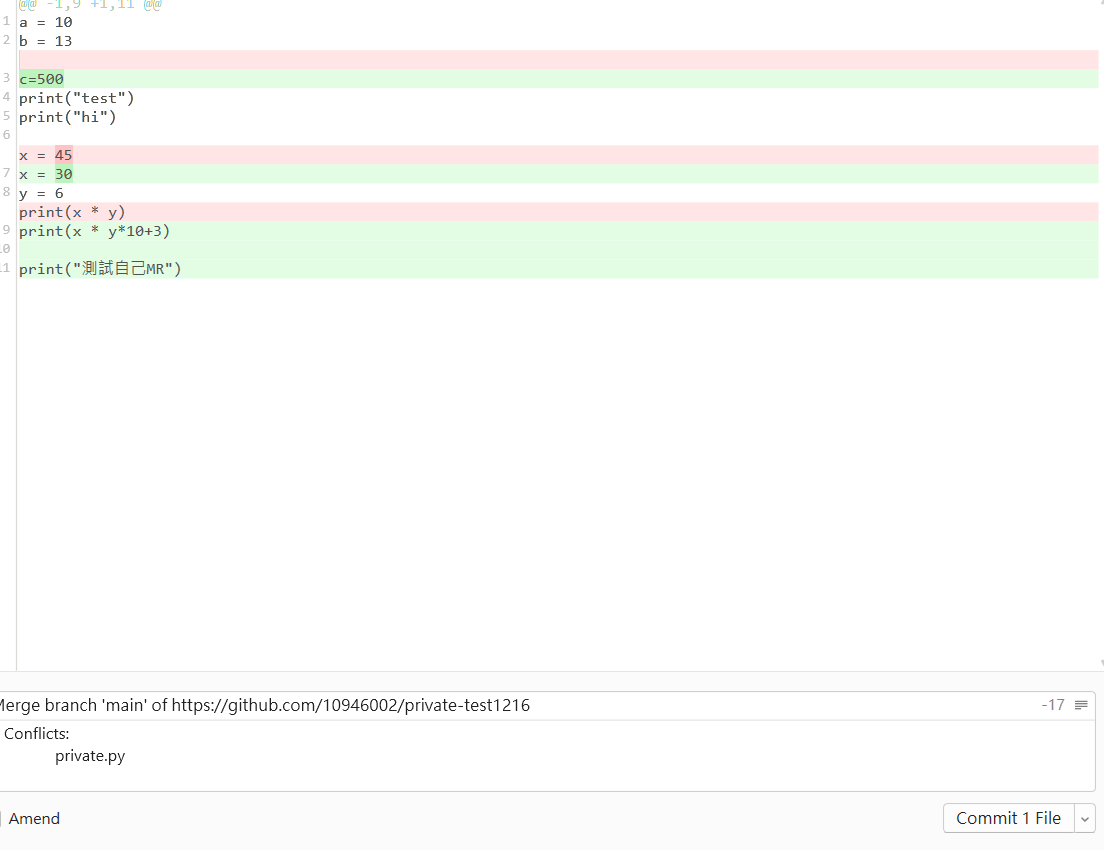
兩邊都保留(不建議)



確定留下的內容後右下角的Resolve送出



要記得commit送出

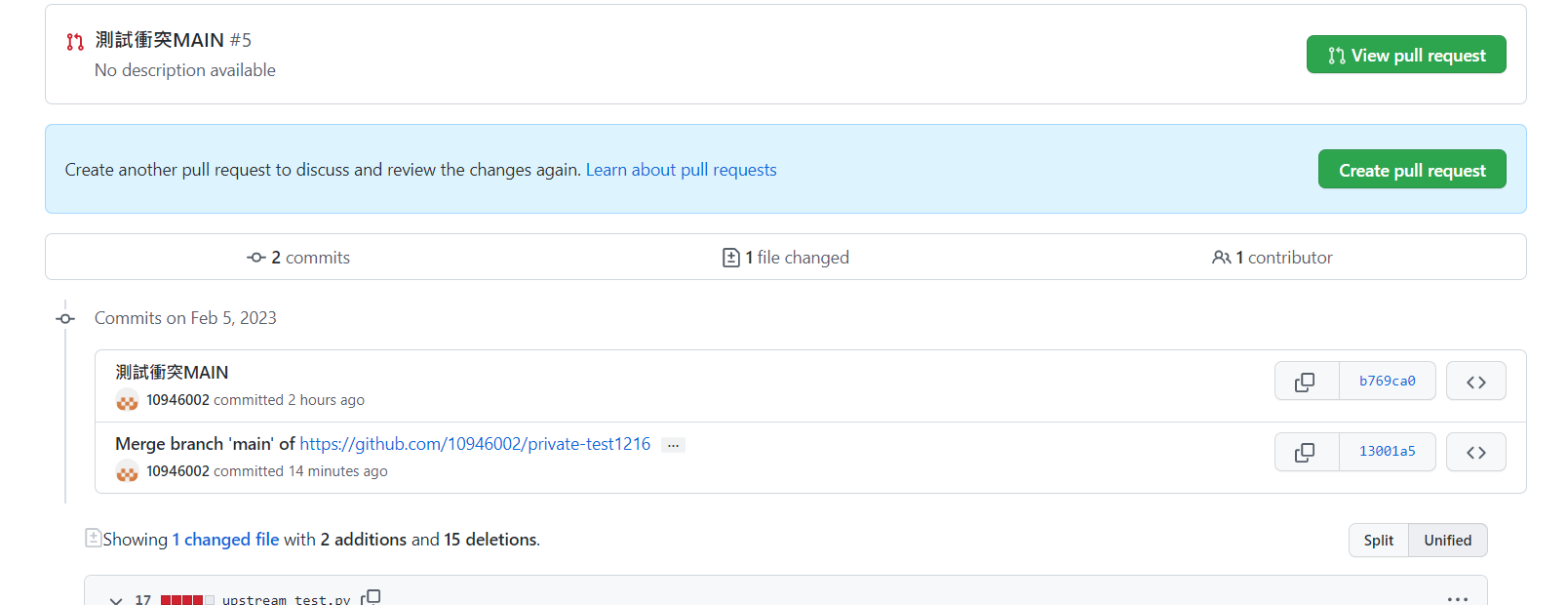


訊息顯示已修復衝突



**請重新發一次MR，給審核人員**

應包含自己要推上去的所有commit和已經解過衝突的commit



待審核人員github那邊審核通過後就能去pull最新的檔案了。



最上面那筆就是自己剛剛發的那筆merge，且所有的線已經接再一起了