商管程式設計 109-1 GitHub Workshop

影片助教 彭晨

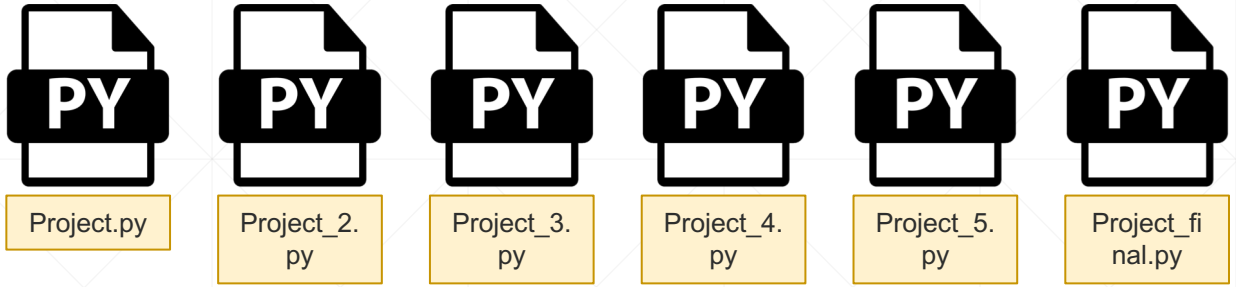
2020/12/07-11

Agenda

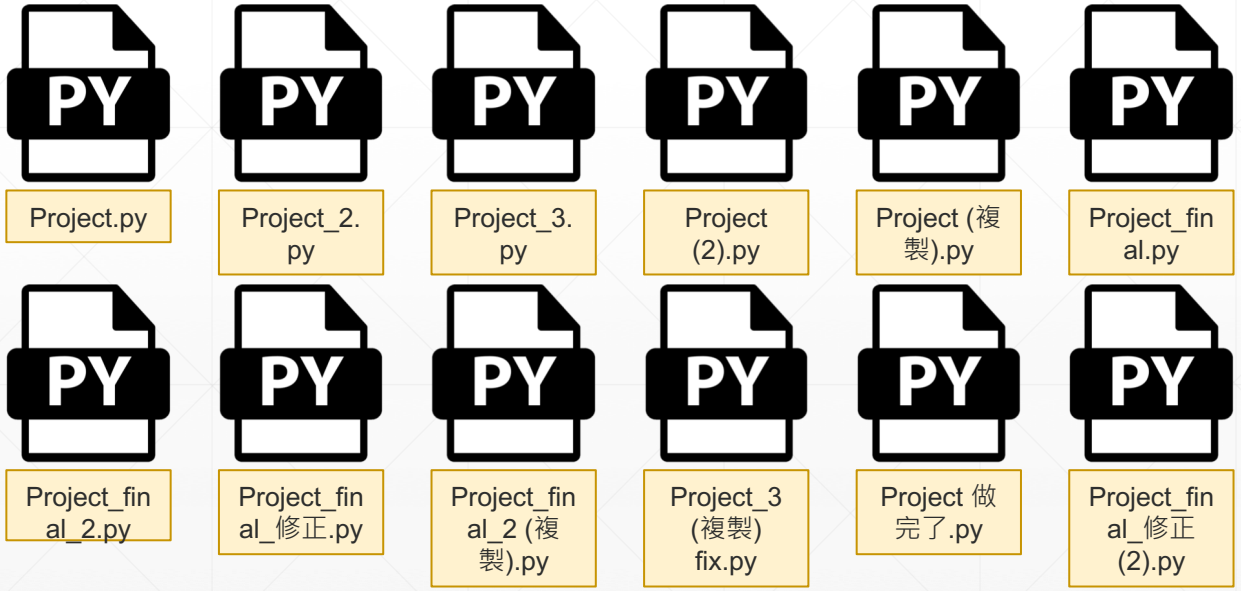
- 版本控制
- GitHub
  - 使用 GitHub
  - GitHub Desktop 操作
- 結語

版本控制

你自己寫 code



大家一起寫 code



版本控制的用途

- 記錄每一次更新的內容
- 檢查新舊版本之間的差異、衝突
- 保存從舊到新的每一份 code，隨時回退版本
- 有效的群組共同編輯
- ~~抓出是哪個老鼠屎把 code 寫爆了~~

VCS – 版本控制系統

- 2005 年，一群寫 code 的人(linux devs) 為了避免悲劇頻繁發生而催生了 git
  - 免費開源！(a.k.a. 任何人都可以用)
- 伺服器上放一份 code
  - 每個人電腦再存一份
- 要寫 code 前 → 從伺服器上下載
- 寫完 code 後 → 更新上去伺服器

應用 git 的服務

GitHub

GitLab

BitBucket

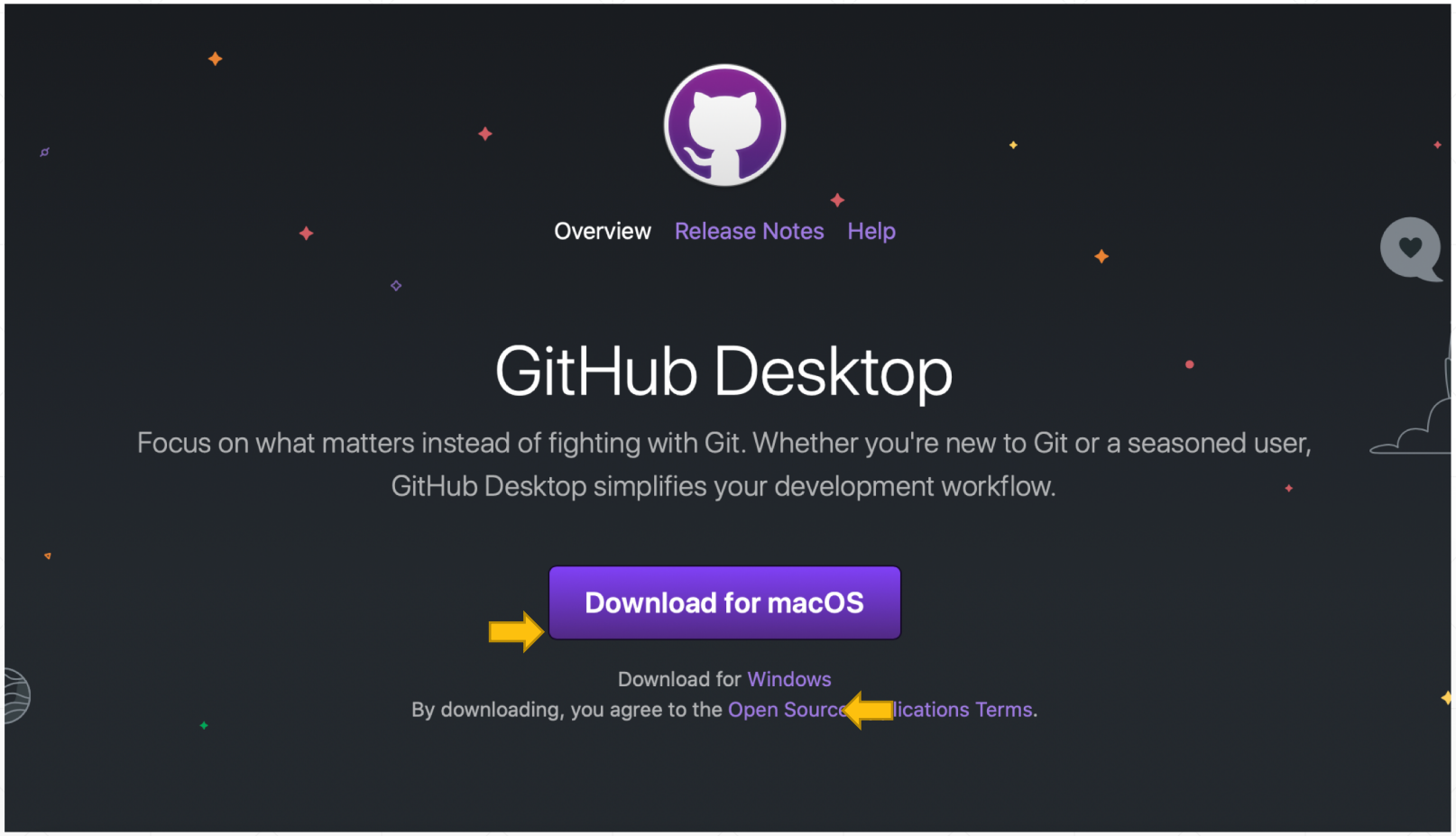
SourceForge

為甚麼今天教 GitHub？

- 最多人用，具有豐富的社群和功能
  - 共同開發同一個大專案
  - 合理的借用別人的 code，發展自己的版本
  - 和別人線上討論、追蹤議題
  - 建立專案的線上 wiki 說明文件
  - 建立自己的 (靜態) 網站
- 當作自己的求職履歷

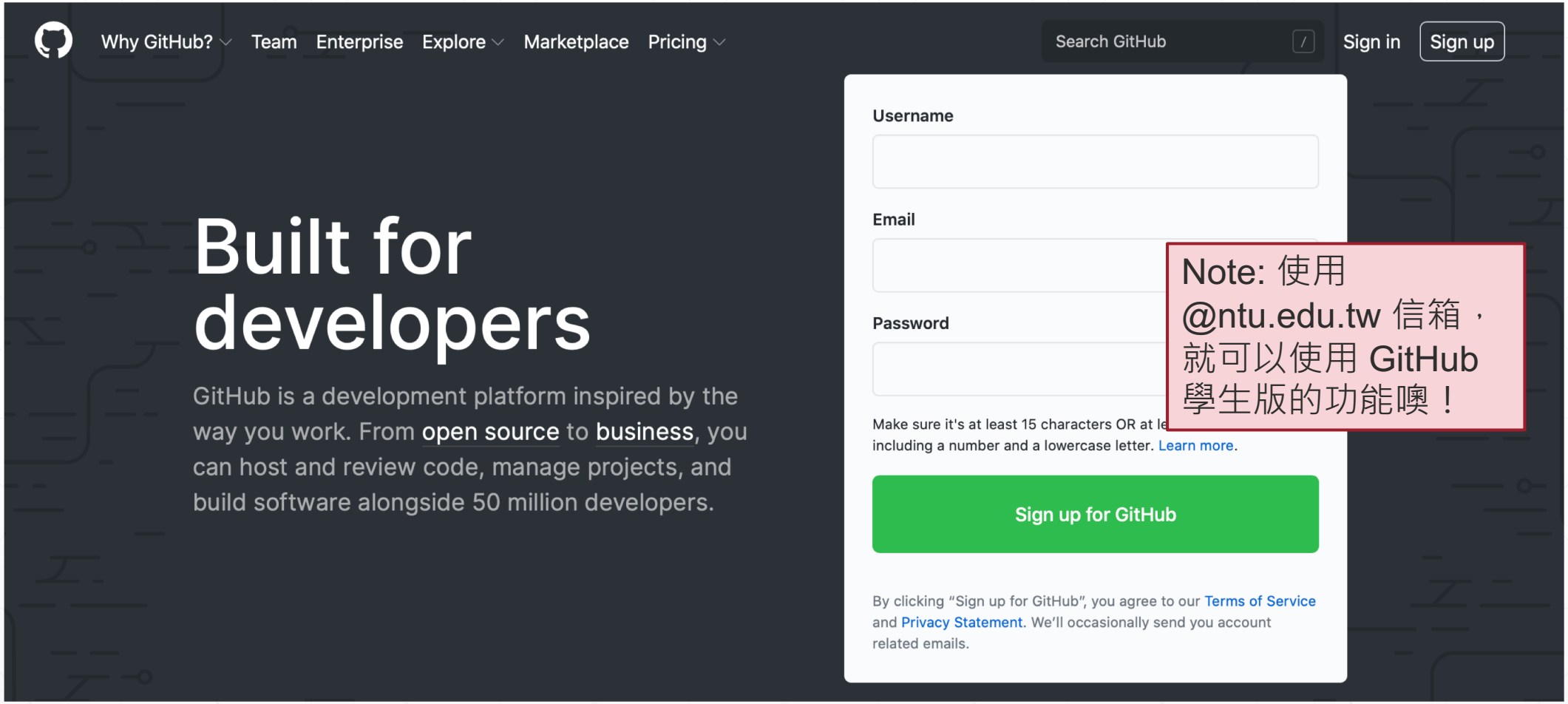
使用 GitHub

下載 https://desktop.github.com/



目前圖片是在 GitHub網站喔

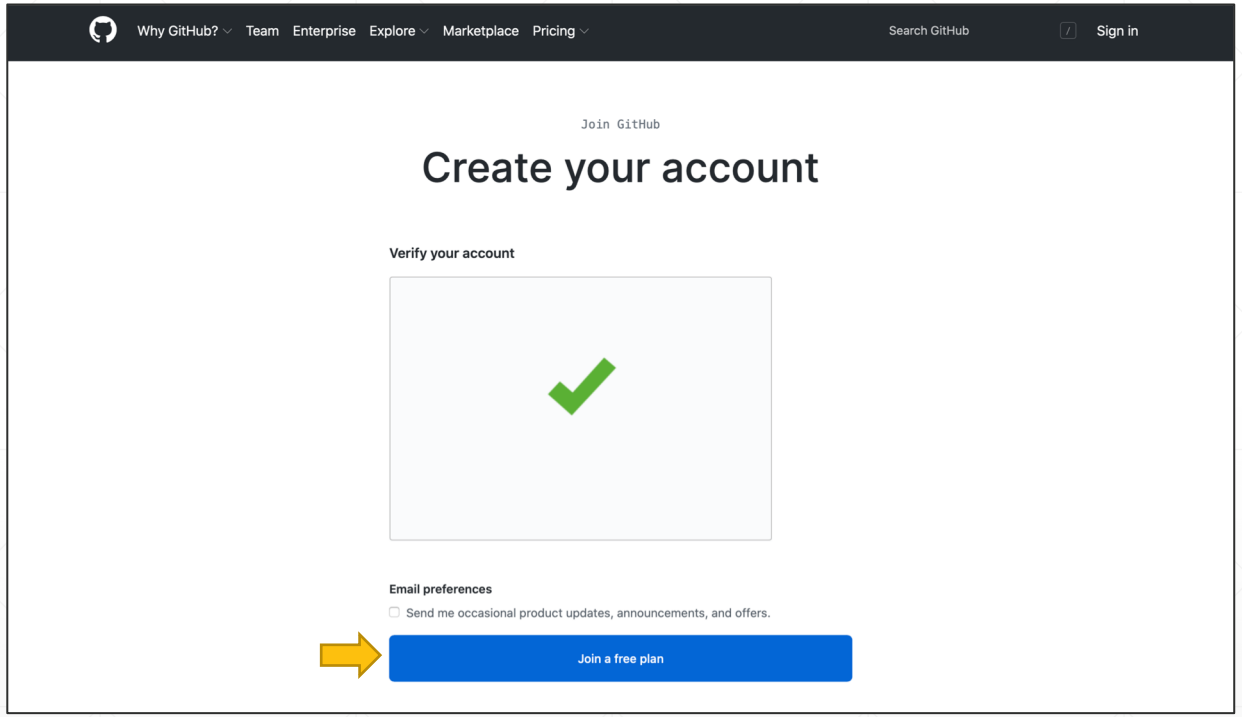
註冊 https://github.com/



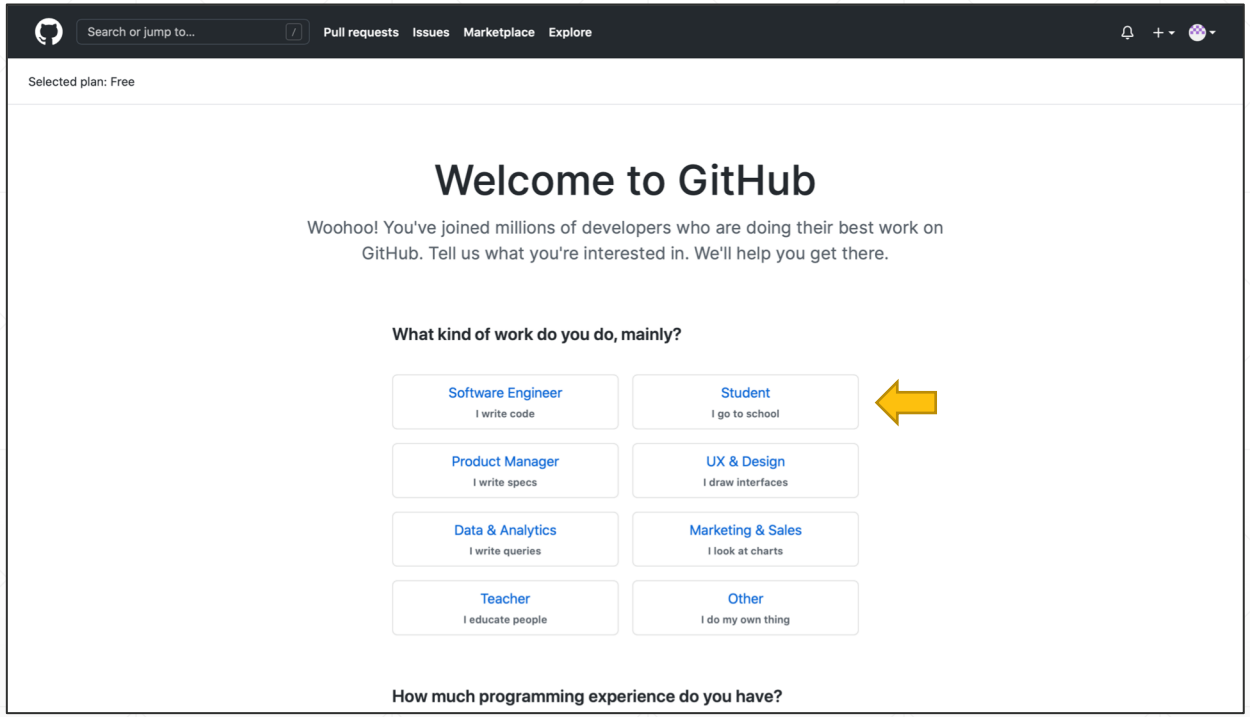
Note: 使用 @ntu.edu.tw 信箱，就可以使用 GitHub 學生版的功能噢！

目前圖片是在 GitHub網站喔

真人驗證



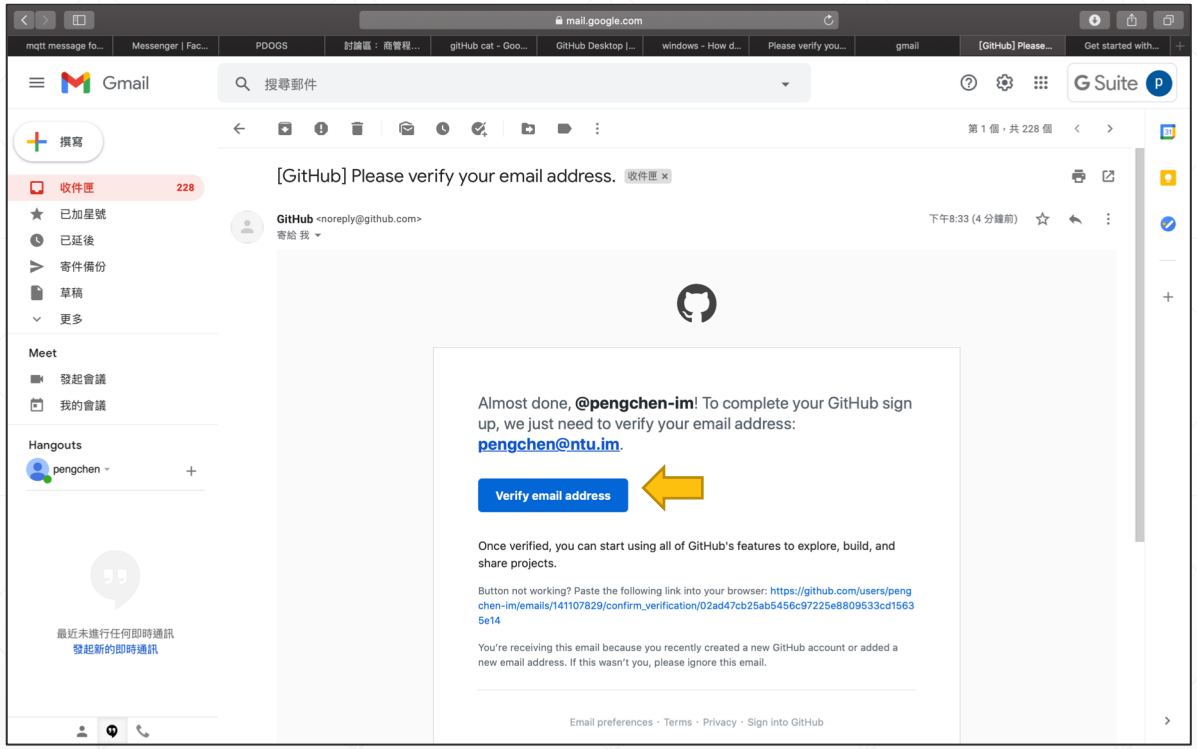
回答問題

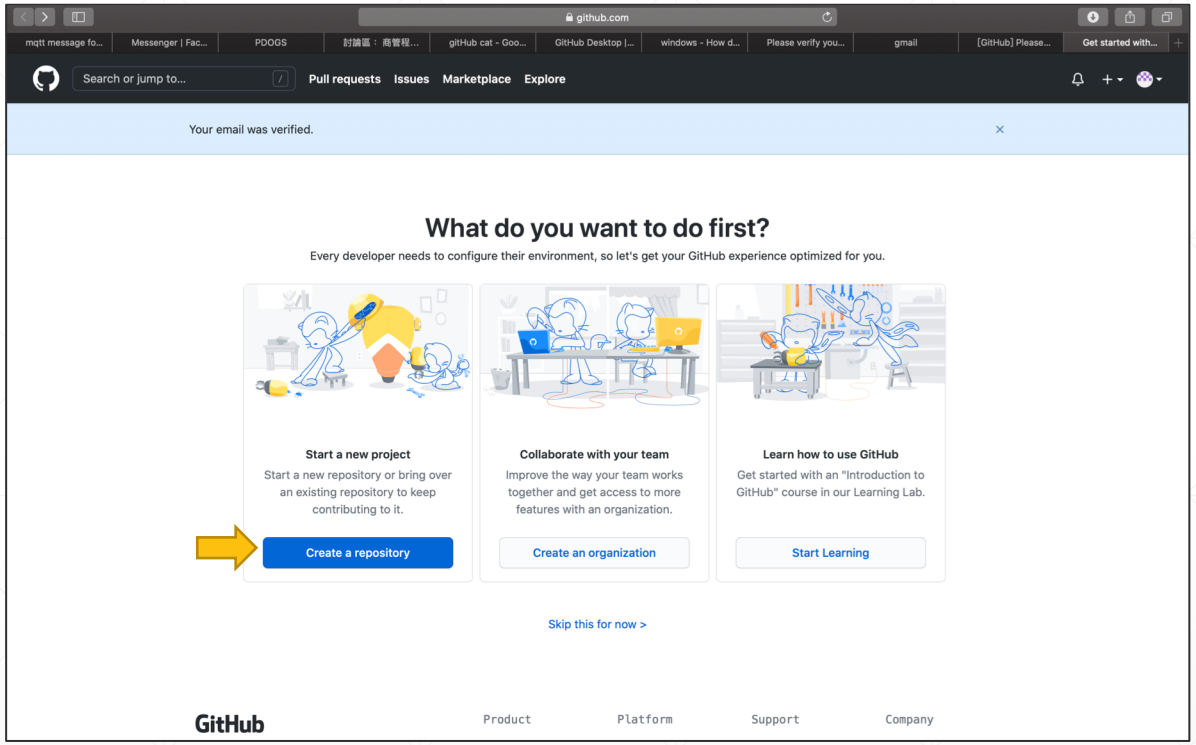


目前圖片是在 GitHub網站喔

信箱驗證

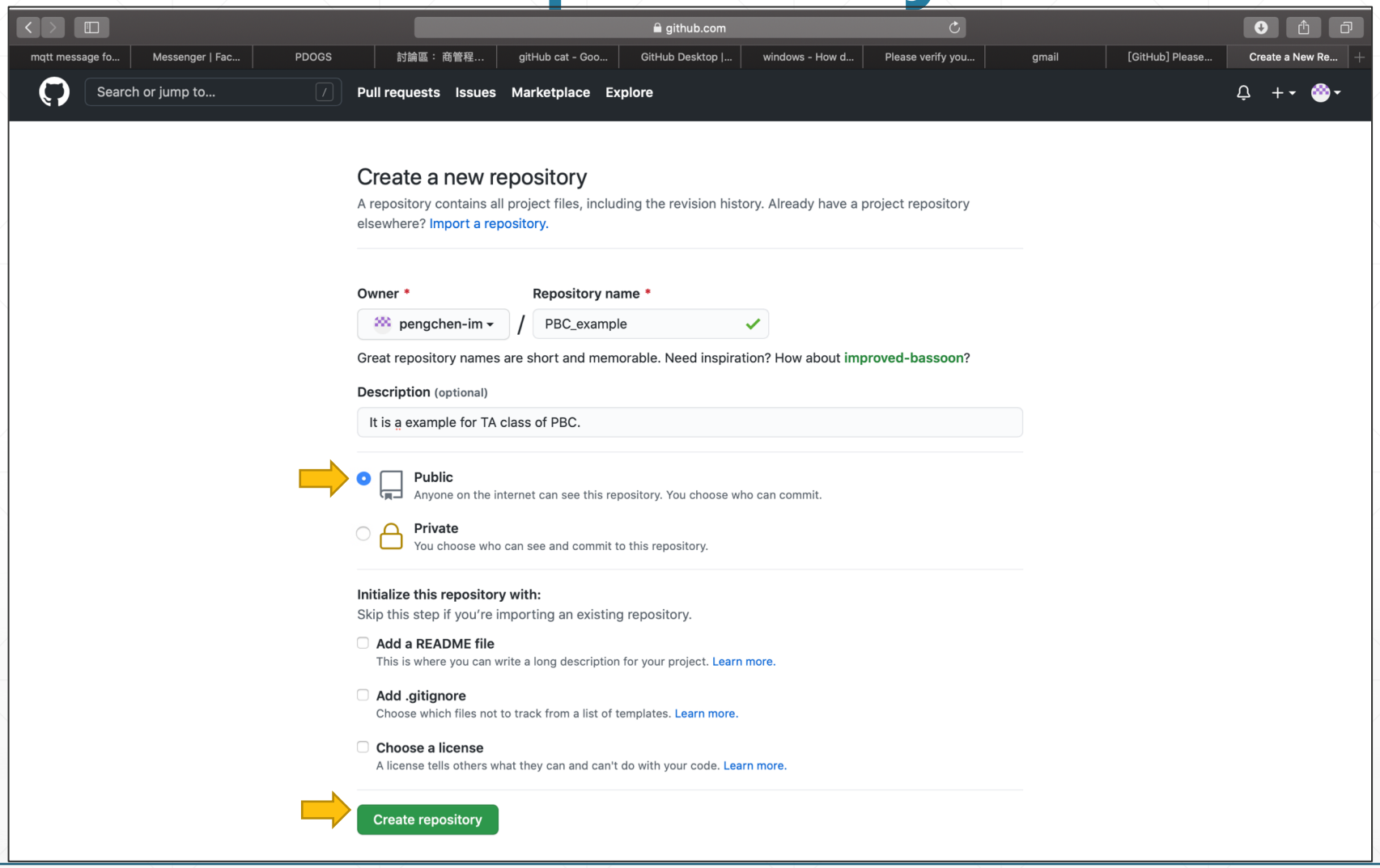
Create a repository



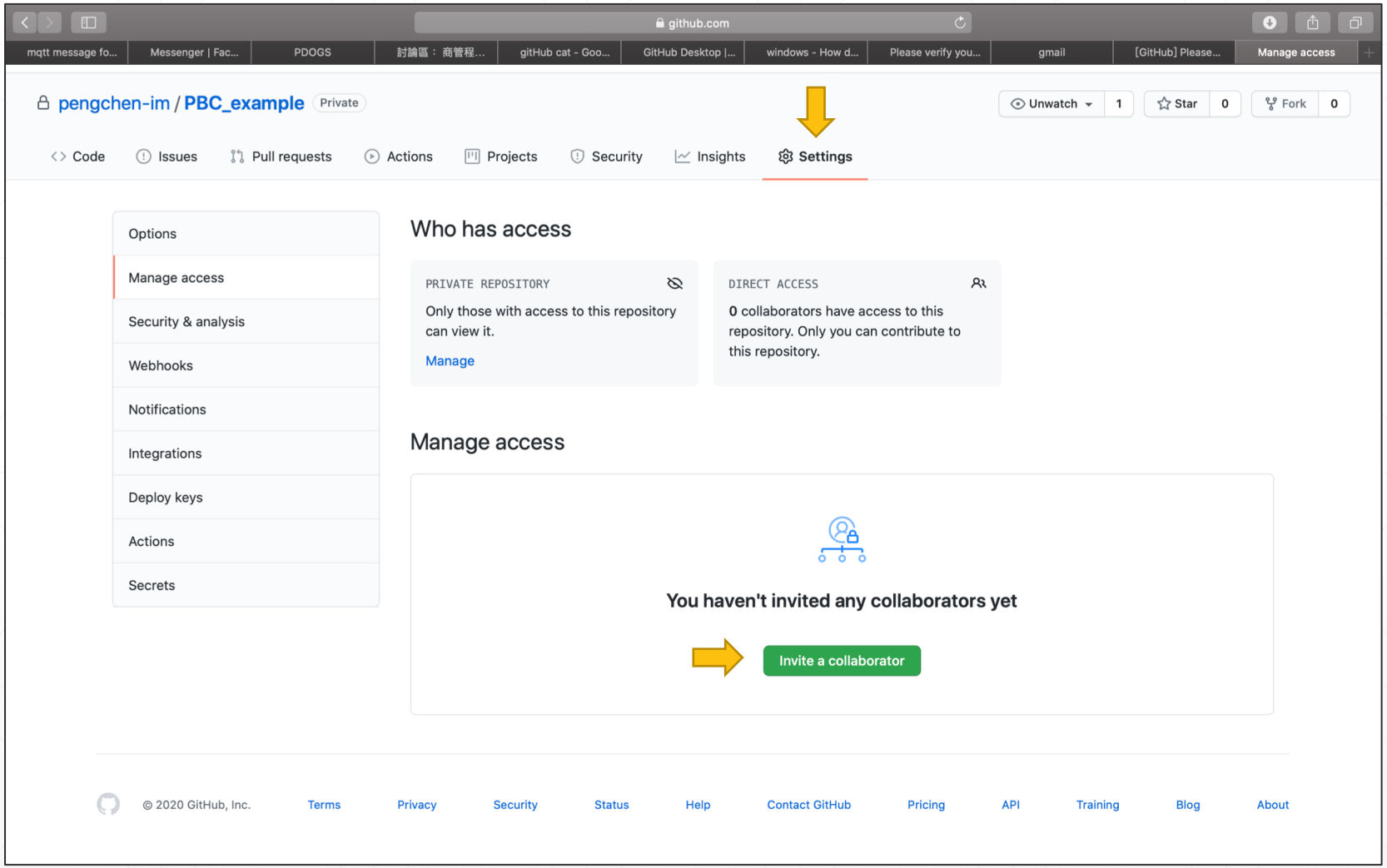


到剛剛註冊使用的信箱收驗證信

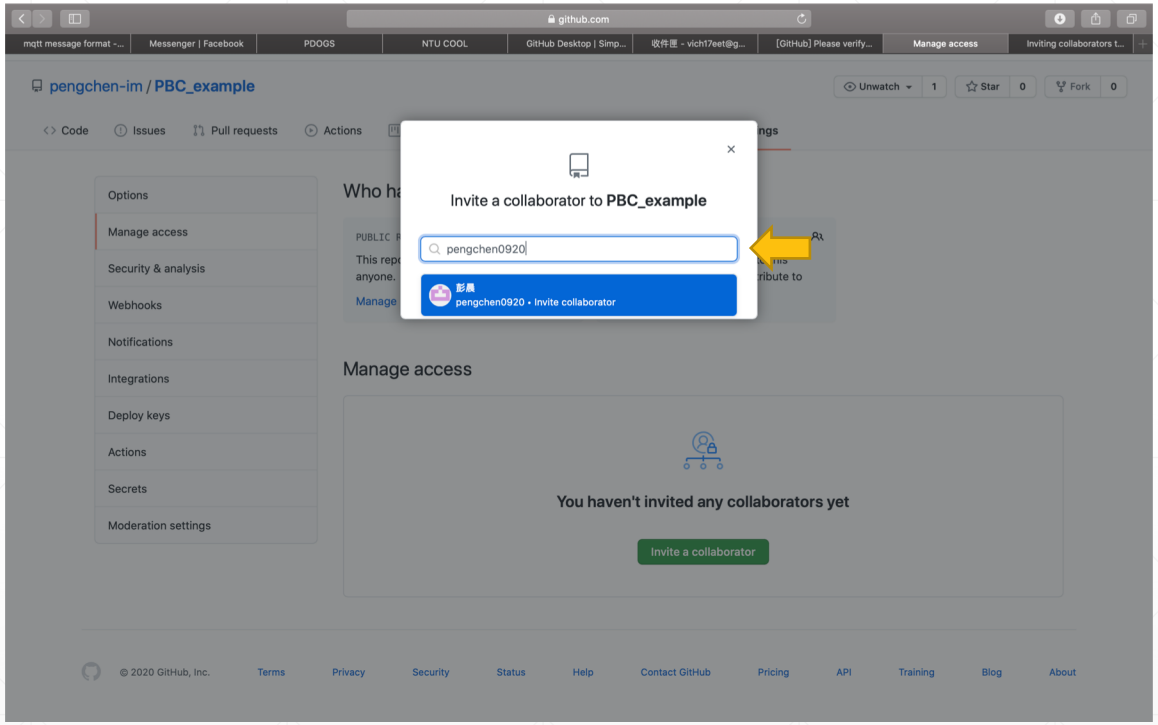
Create a new repository

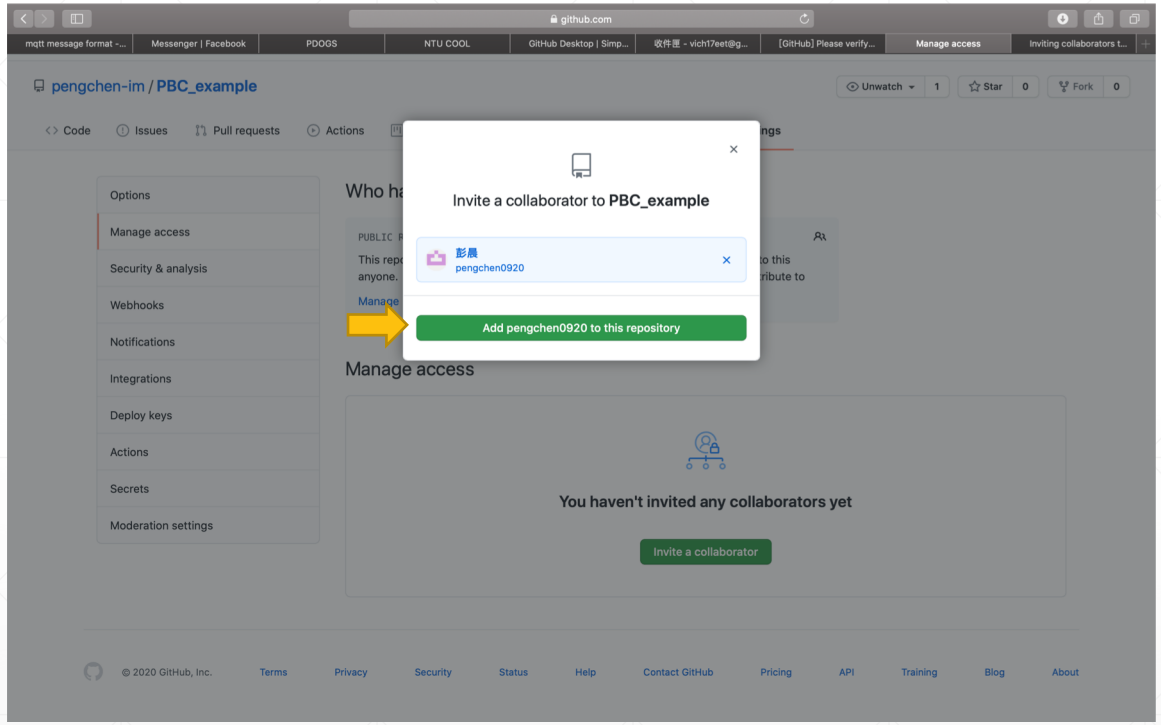


Invite collaborators = 你的組員們



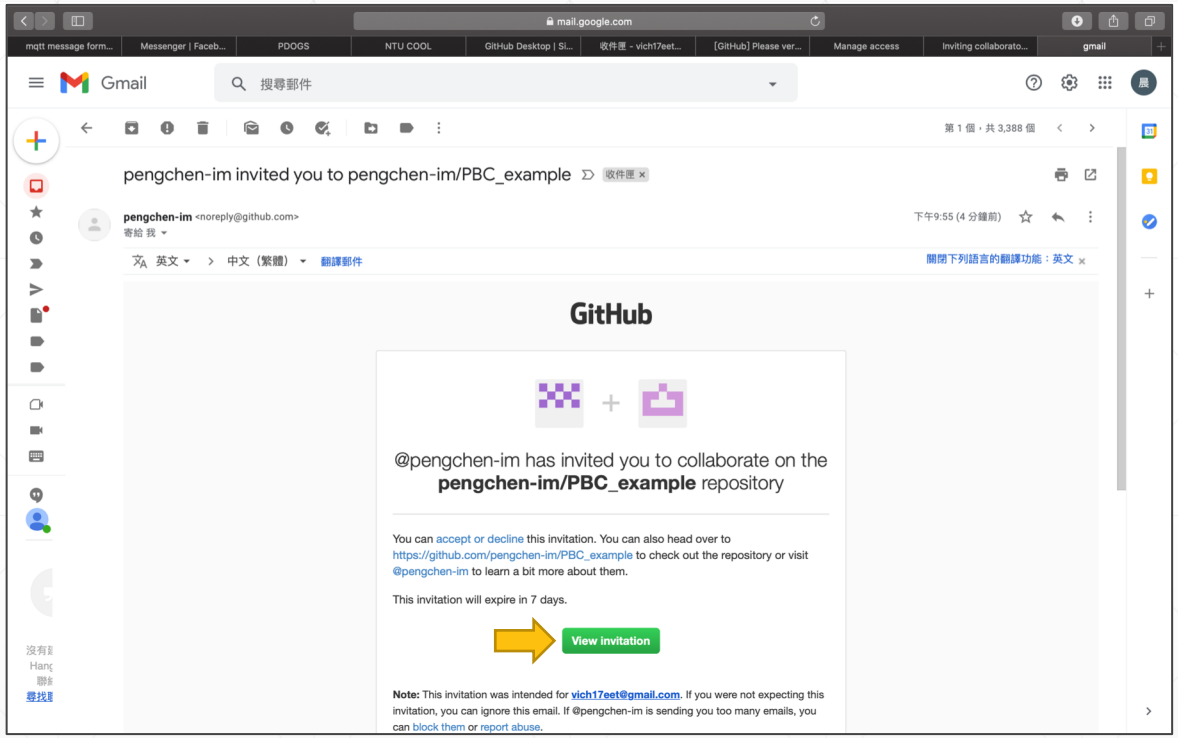
Invite collaborators = 你的組員們

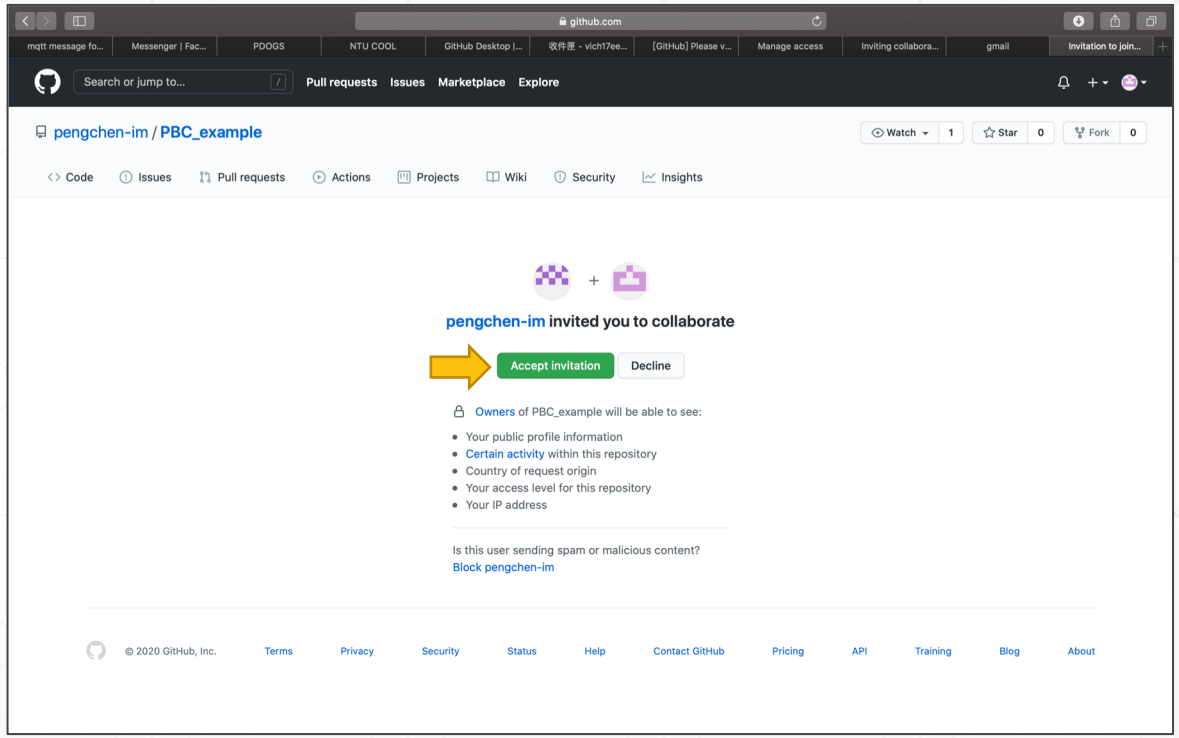




輸入你組員們 GitHub 的名字或是註冊的信箱

Accept invitation

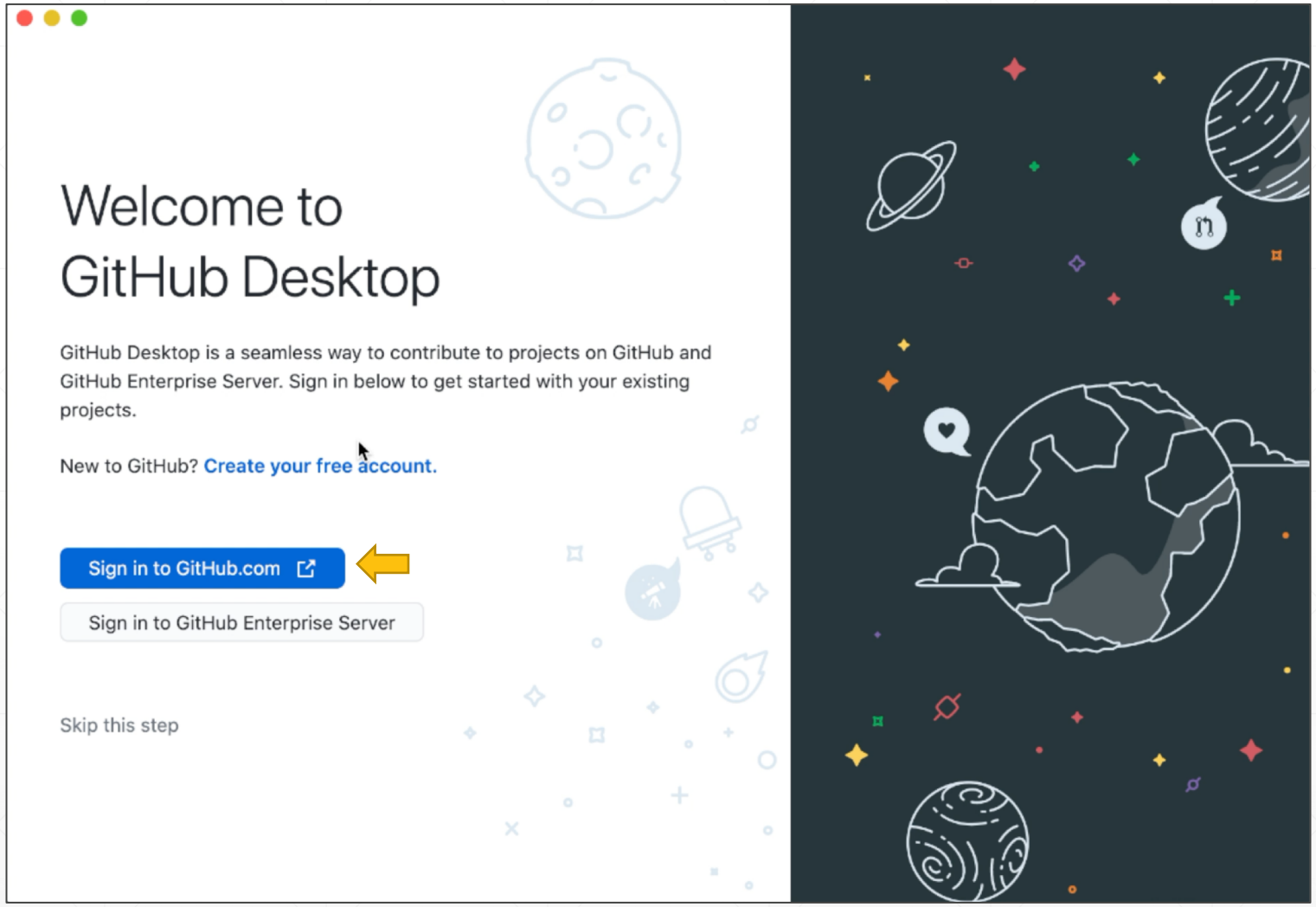




「被邀請的組員」記得去註冊的信箱收信！！！

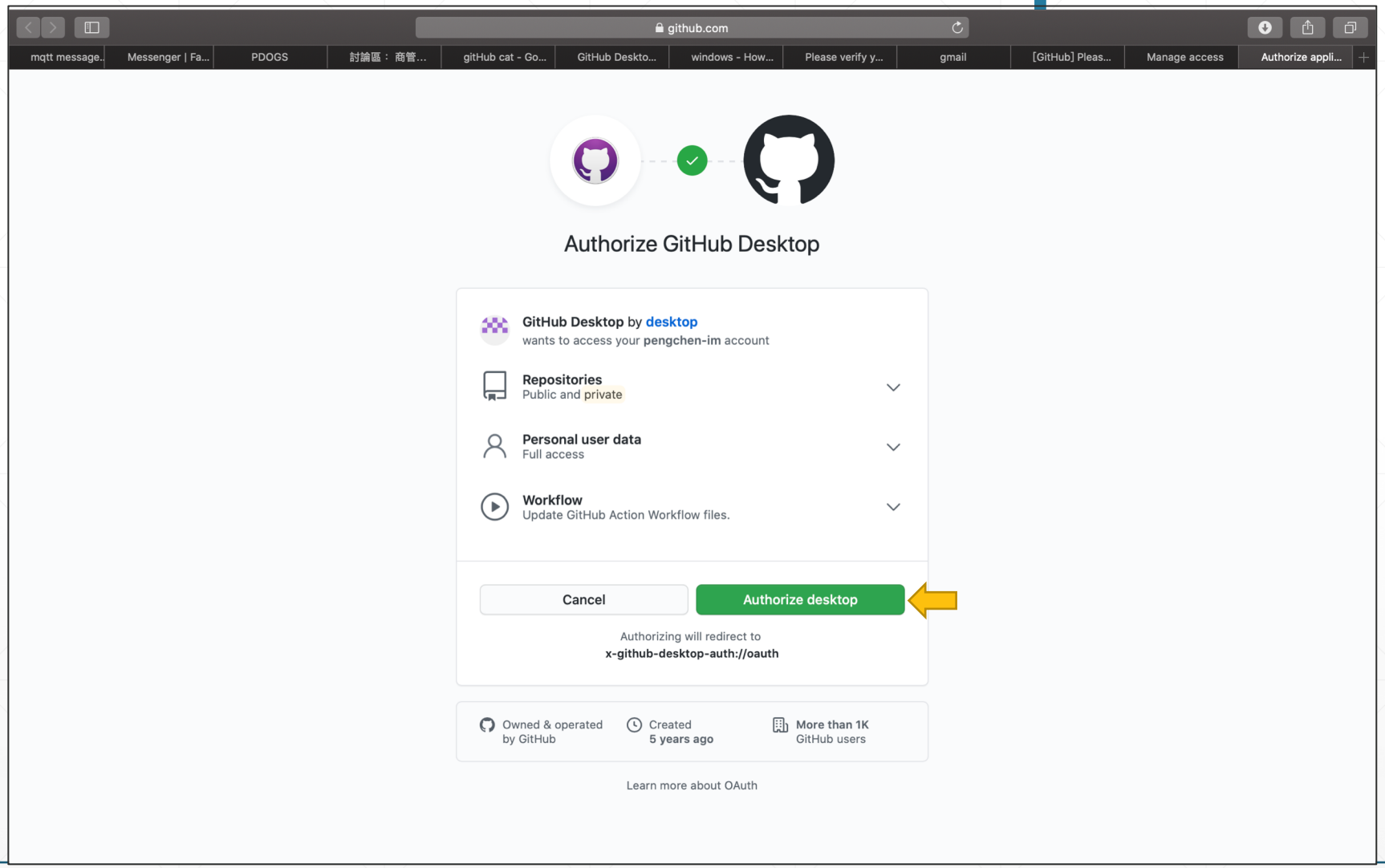
被邀請的組員按下 **Accept invitaion**

打開 p.10 安裝的 GitHub Desktop

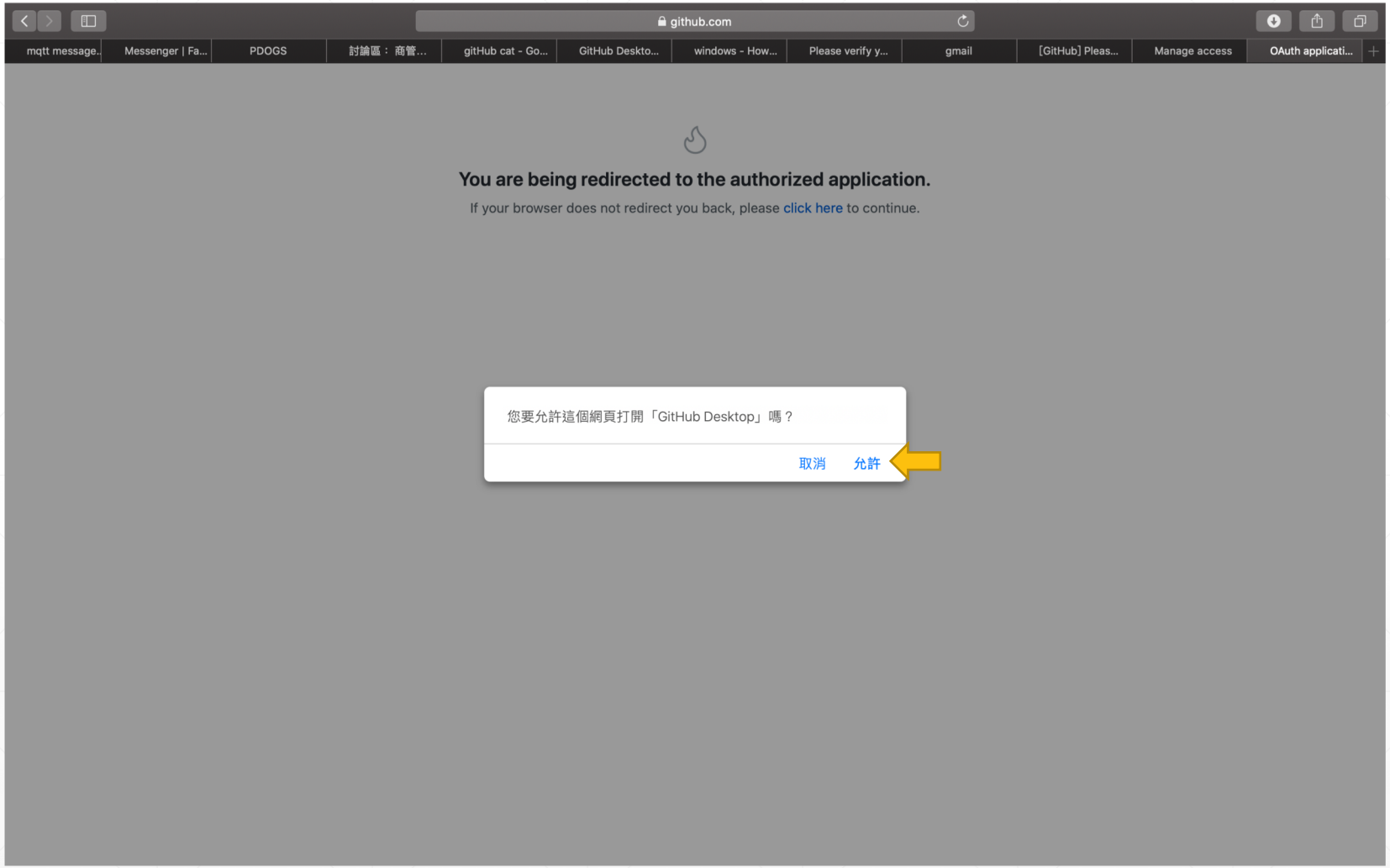


此時你的 browser會跳出 Authorize GitHub Desktop的頁面

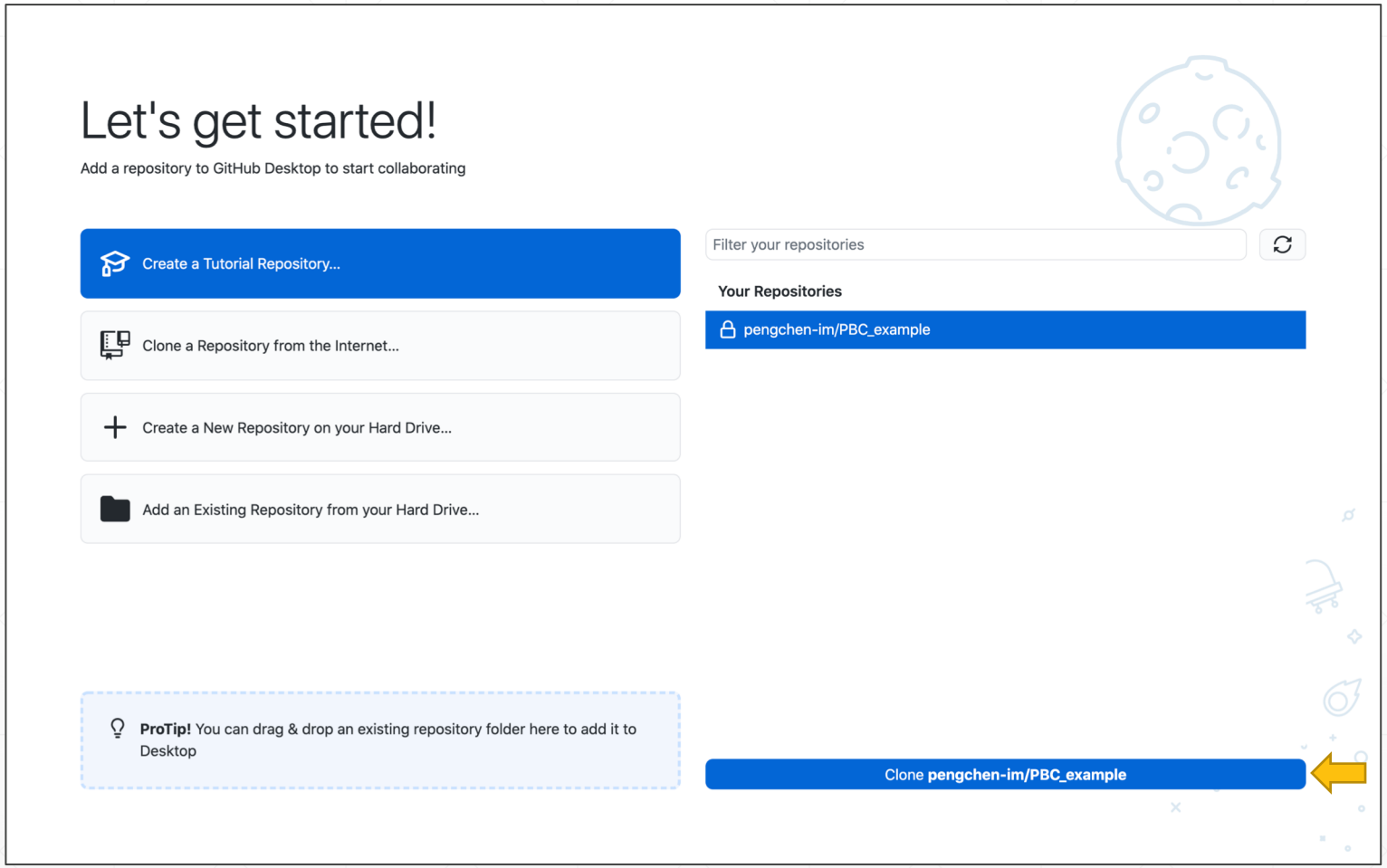
Authorize GitHub Desktop



Authorize GitHub Desktop

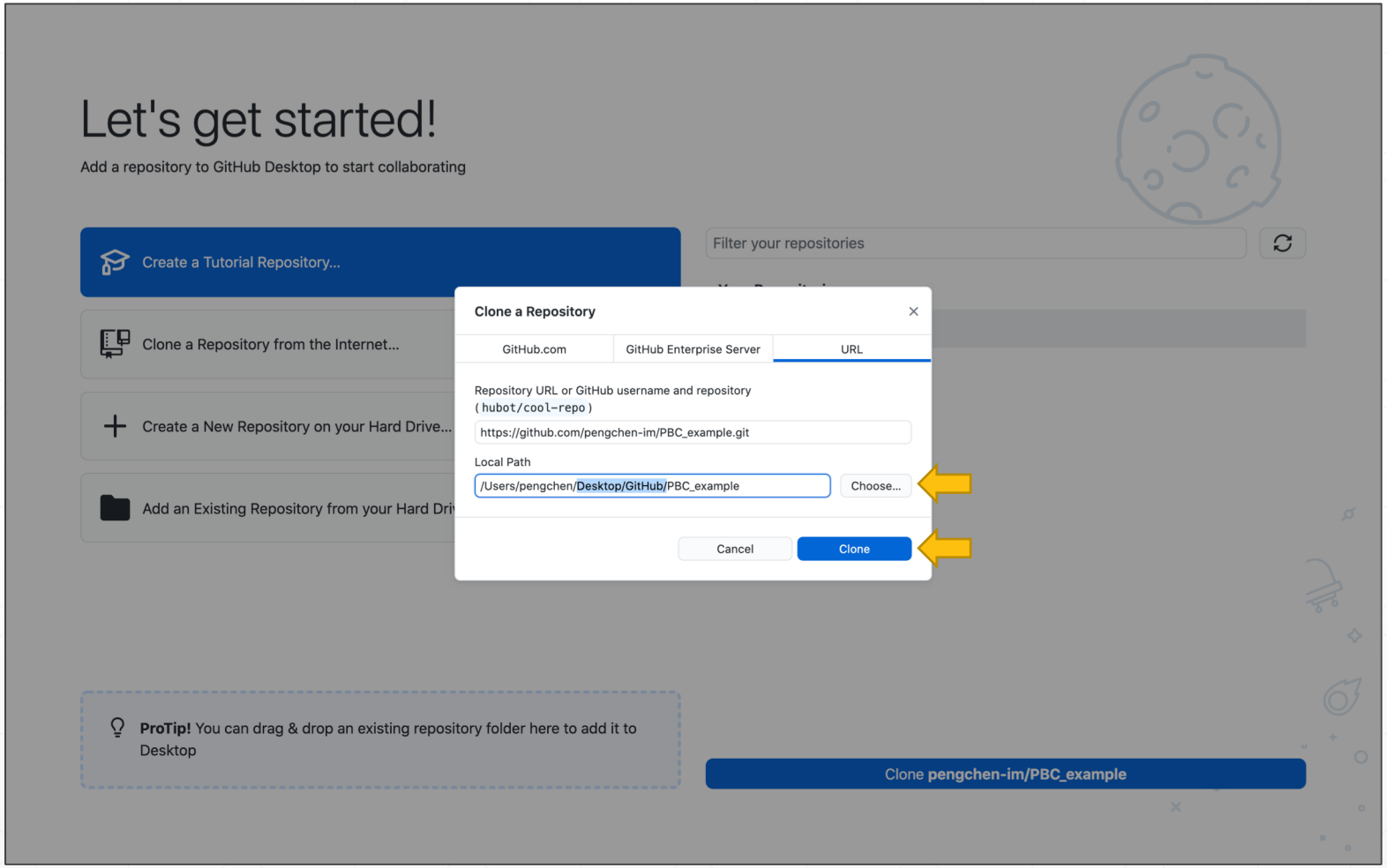


選擇剛剛在網站建立的 repository



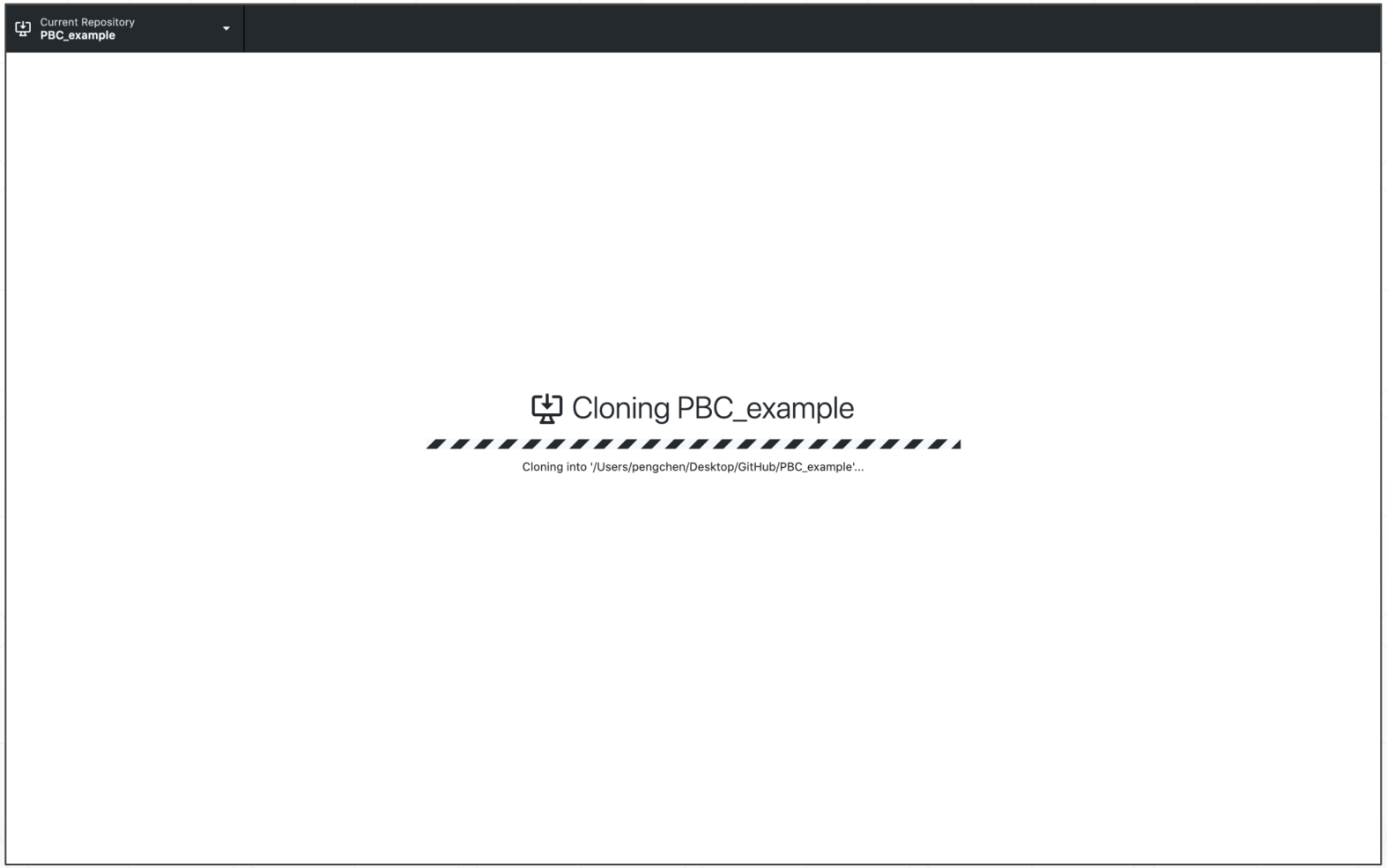
我們目前已在 GitHub Desktop 了

選擇 Local Path



可以在桌面新增一個資料夾，這樣同學們以後比較方便作業

Cloning……



把在伺服器上的 repository 拷貝一份到本地端（你的電腦剛剛設的 Local Path）

甚麼是 repository (repo) ?

- 等於一個**線上倉庫**
  - 一個專案 or 一個共享資料夾

- 把需要版本控制的東西通通丟上去：`.py` `.html`……
  - 不用版本控制的也通通丟上去：`.mp3` `.jpg` `.csv`……
  - 反正倉庫裡面想放甚麼就放甚麼 (放不用錢)

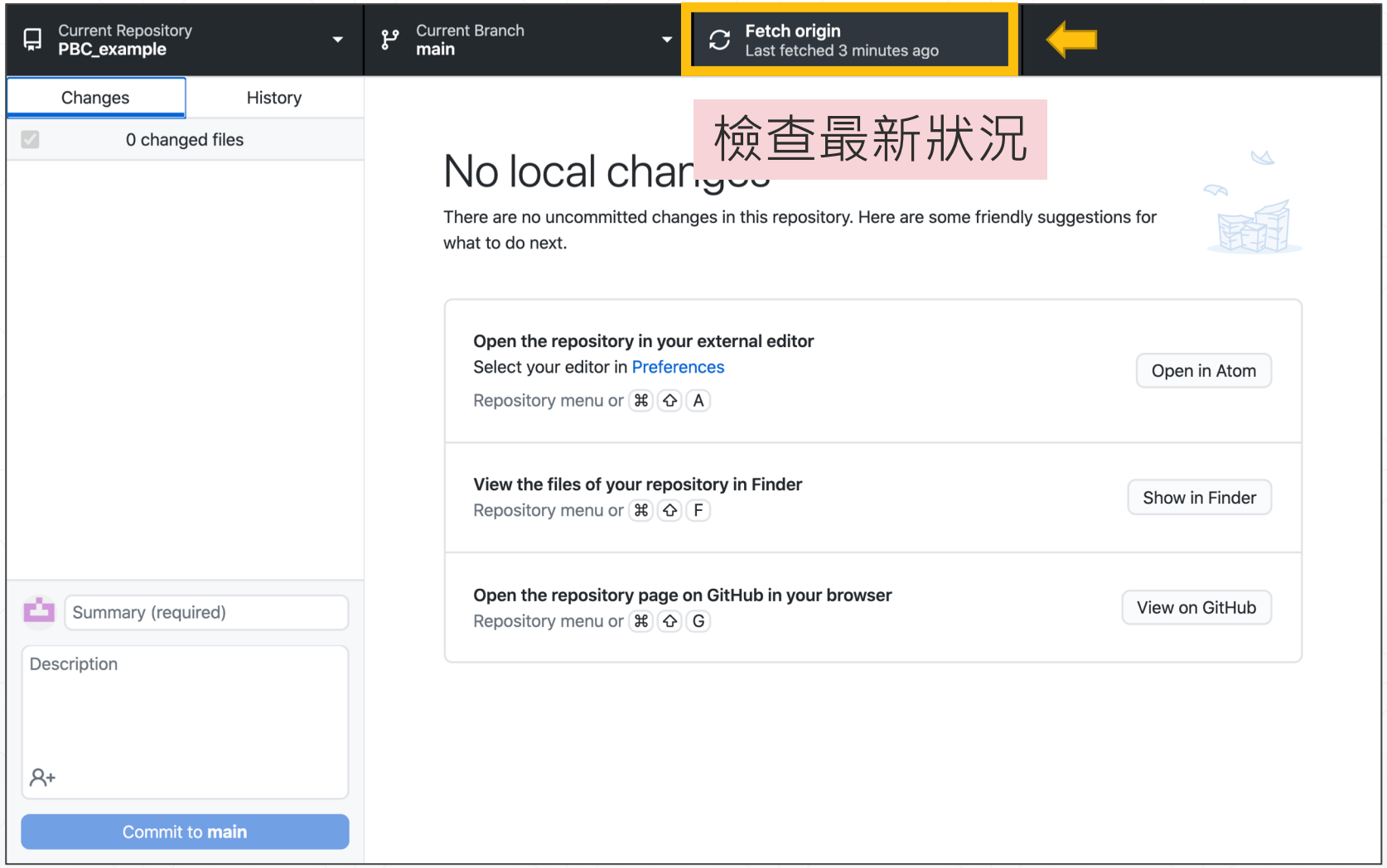
GitHub Desktop 操作

GitHub vs. GitHub Desktop?

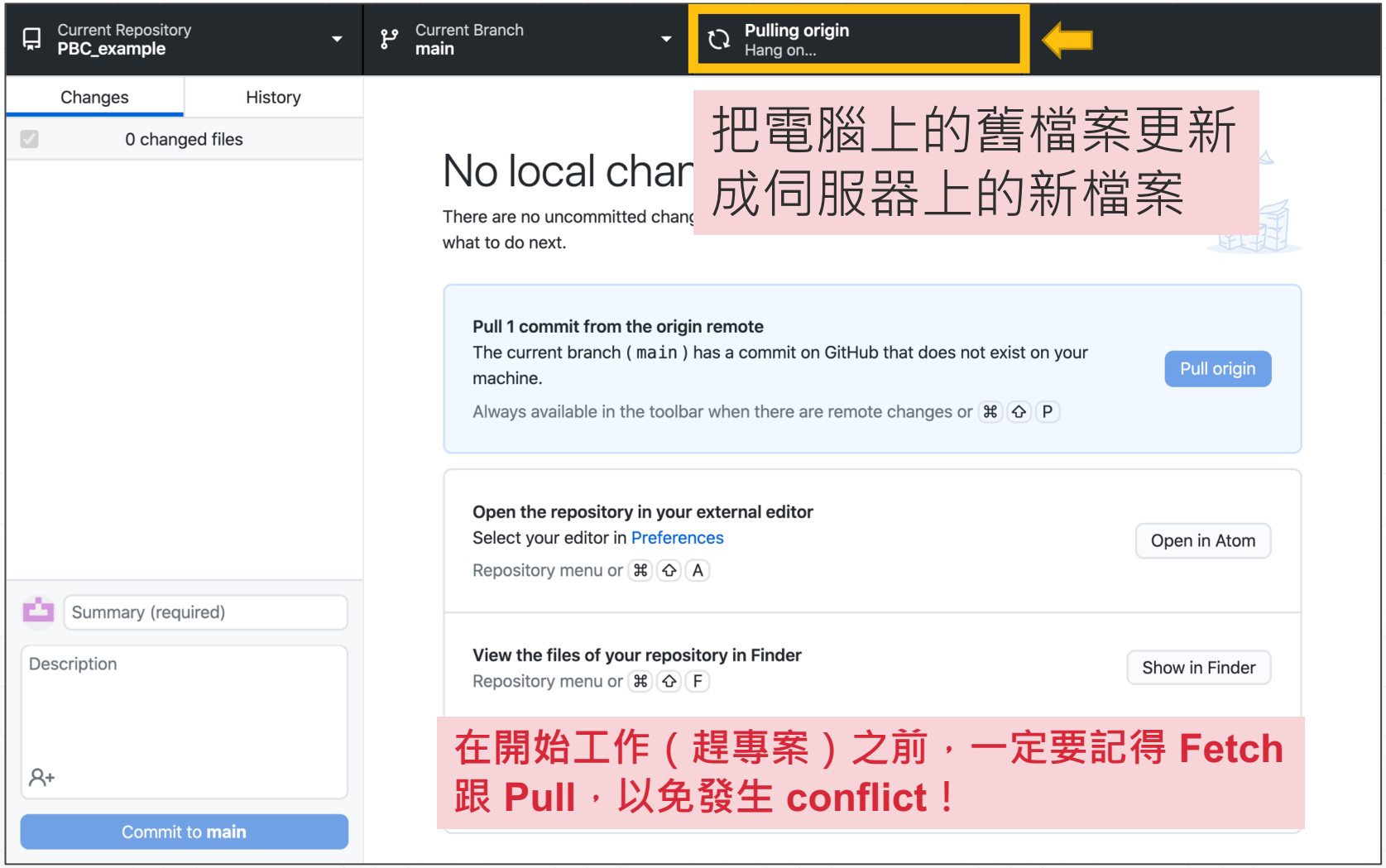
- GitHub 網頁：管理專案
- GitHub Desktop：管理 code

- GitHub：把專案管理和 code 管理包成一個服務
  - 包成一個 repository

Fetch (檢查他人進度)



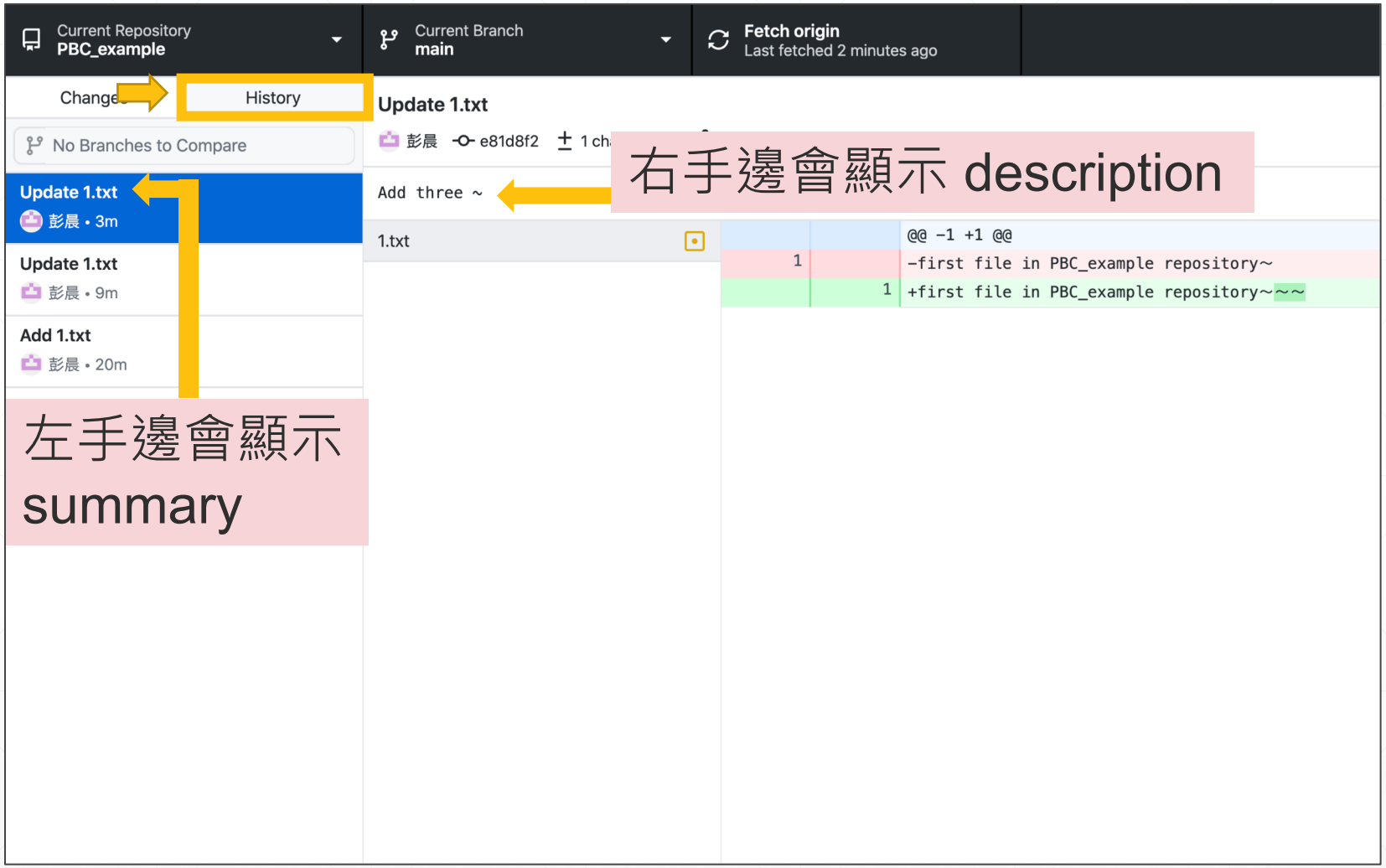
Pull (從雲端更新到電腦)



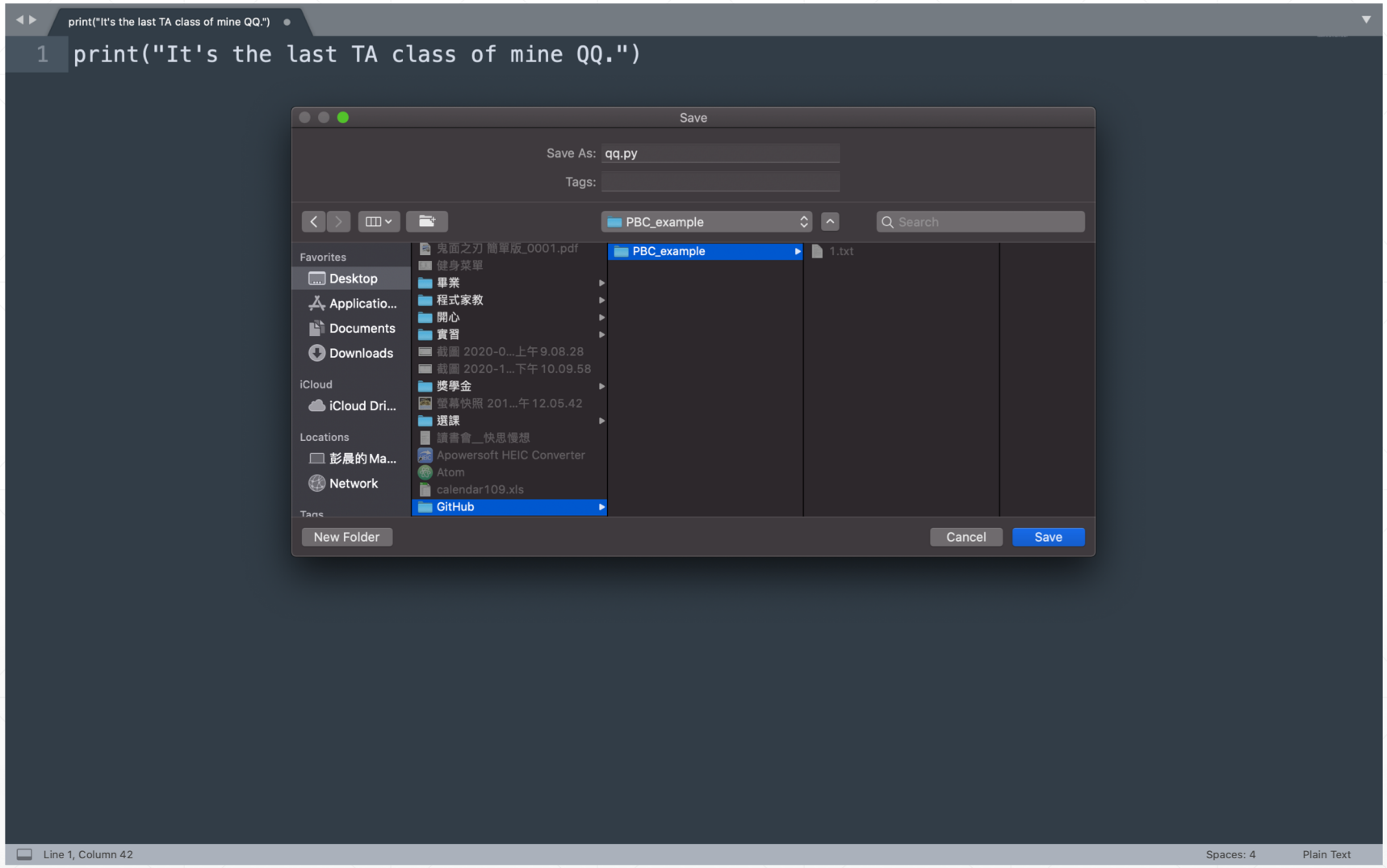
把電腦上的舊檔案更新成伺服器上的新檔案

在開始工作（趕專案）之前，一定要記得 Fetch 跟 Pull，以免發生 conflict！

History (檢視他人更改內容)



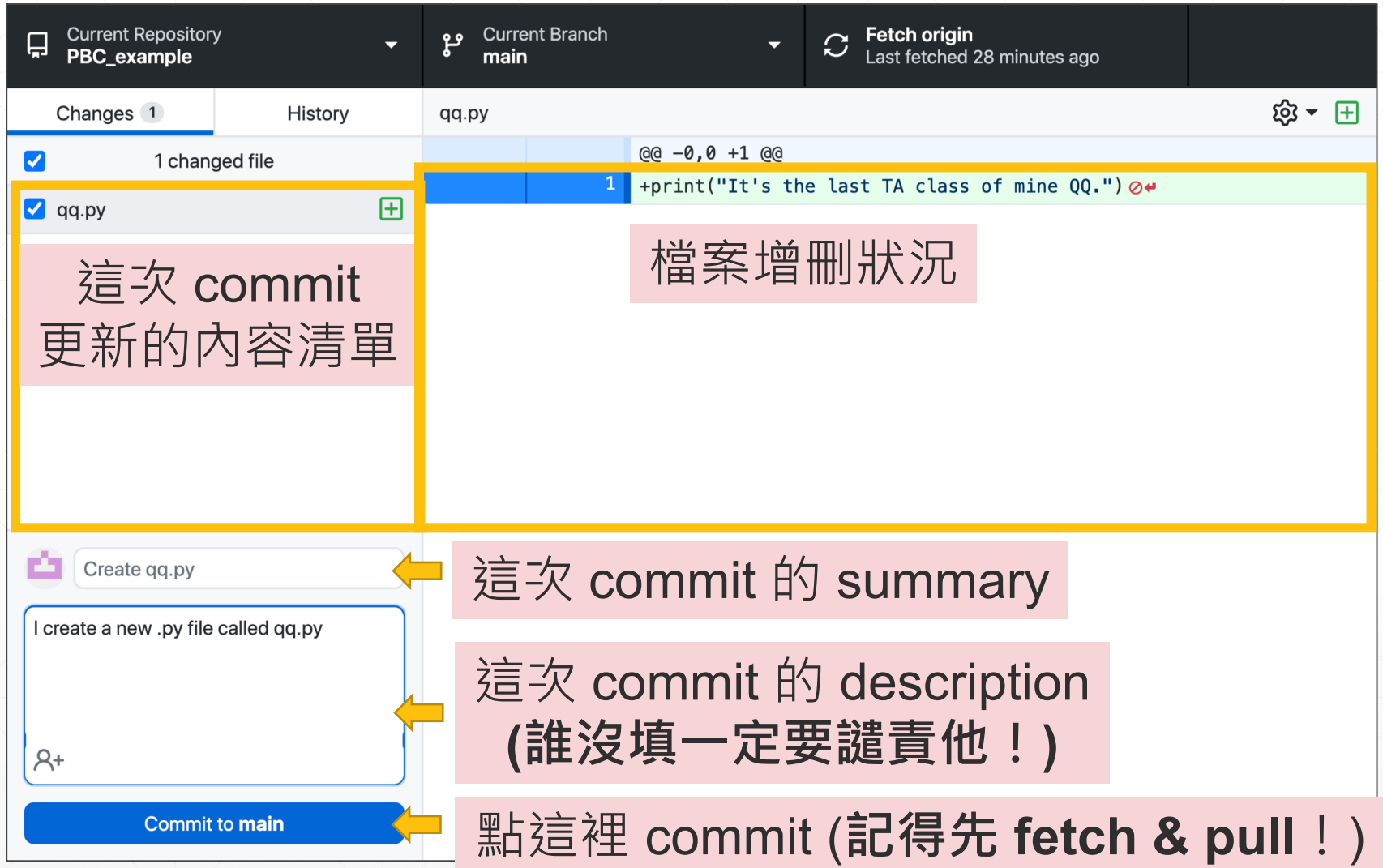
開始打 code



例如：在剛剛的 local path 新增一個 .py 檔

現在可以在我的 local path ，看到剛剛我夥伴新增的 1.txt 這個檔案，因為我剛剛有 Fetch & Pull

Commit (設定存檔點)

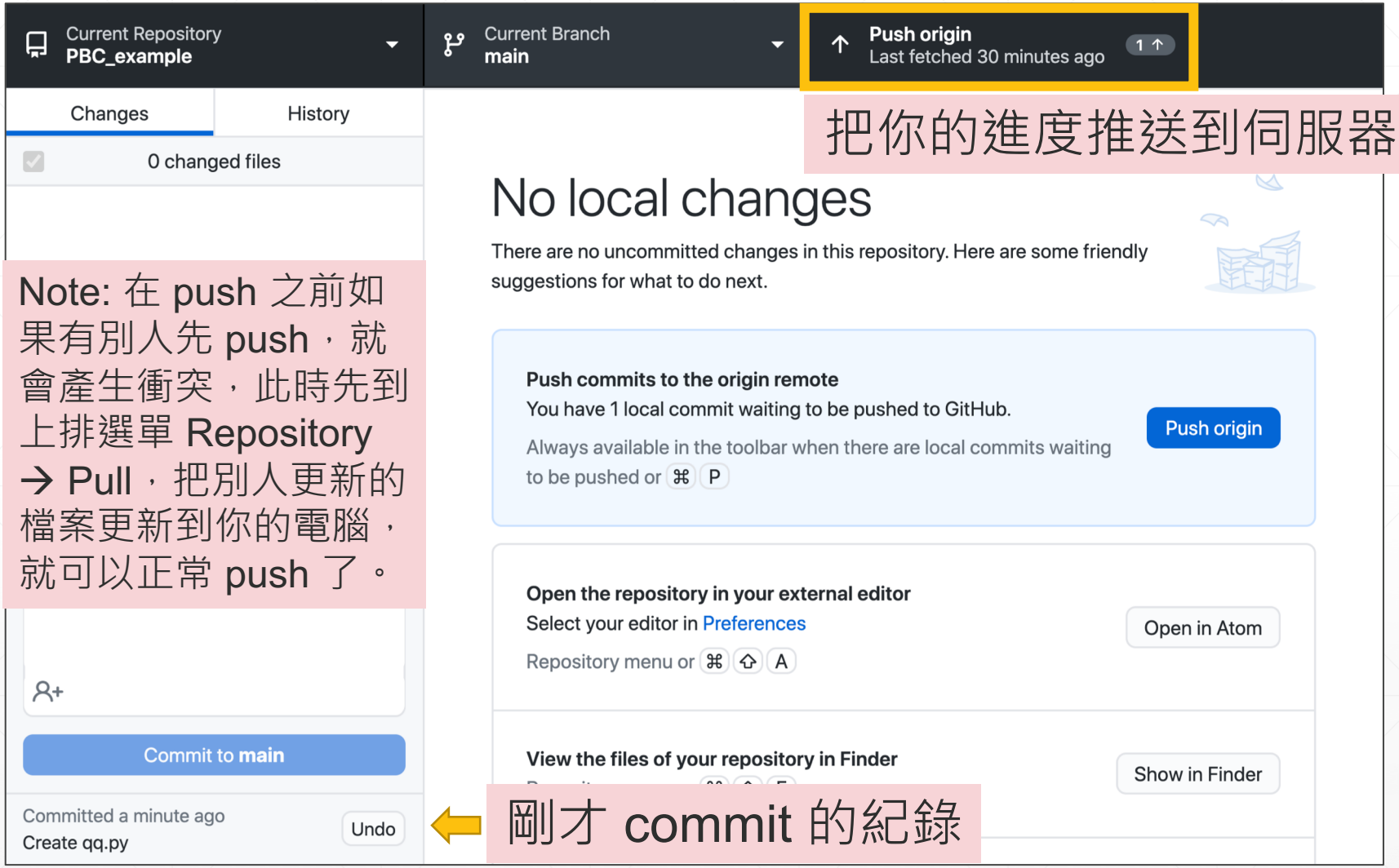


這次 commit 的 summary

這次 commit 的 description
(誰沒填一定要譴責他！)

點這裡 commit (**記得先 fetch & pull**！)

Push (推送 / 存檔到伺服器)

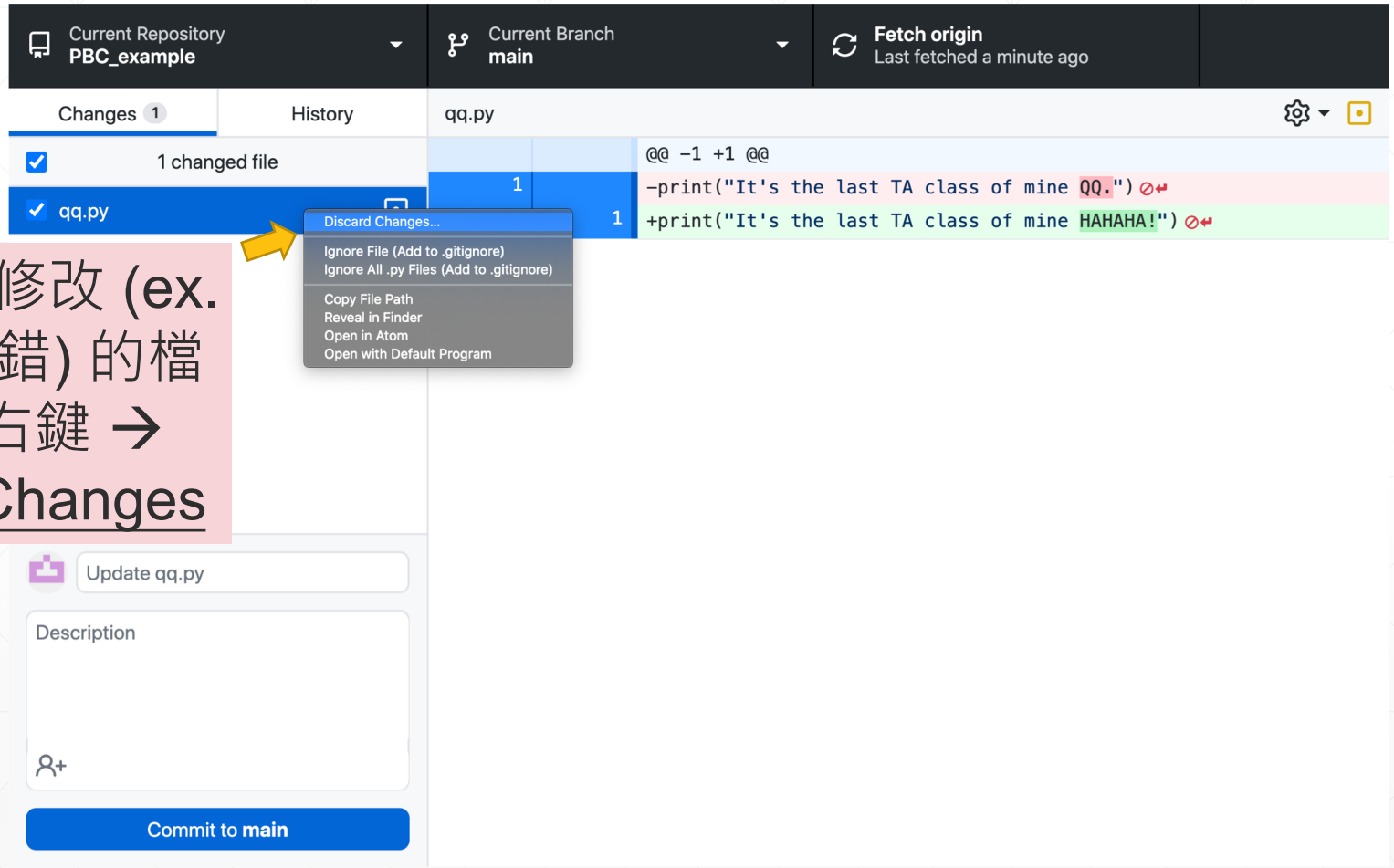


把你的進度推送到伺服器上

Note: 在 push 之前如果有別人先 push，就會產生衝突，此時先到上排選單 Repository → Pull，把別人更新的檔案更新到你的電腦，就可以正常 push 了。

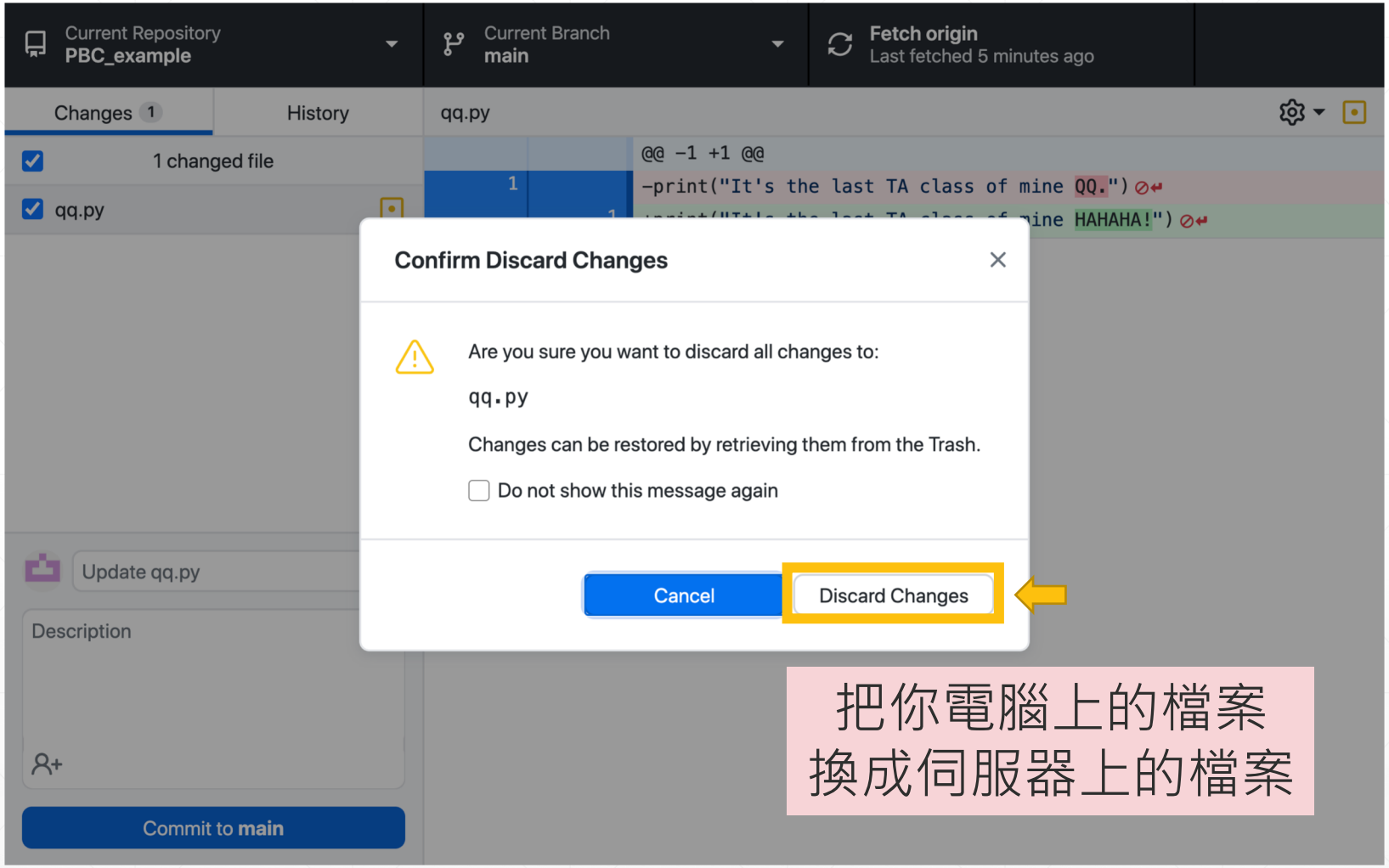
剛才 commit 的紀錄

Discard Changes (取消修改)



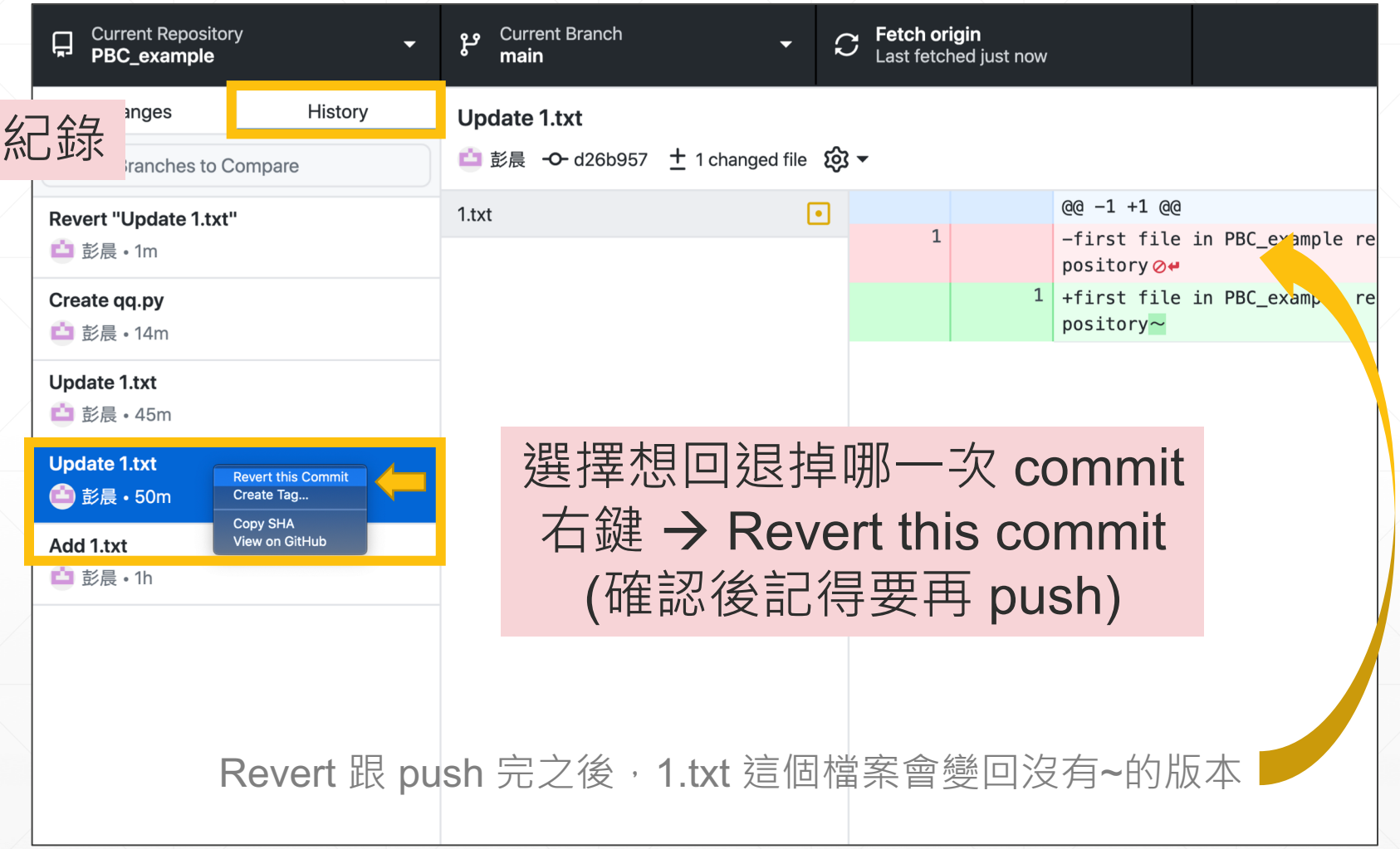
在要取消修改 (ex. 誤刪、改錯) 的檔案上按右鍵 → Discard Changes

Discard Changes (取消修改)



Revert (回退 commit)

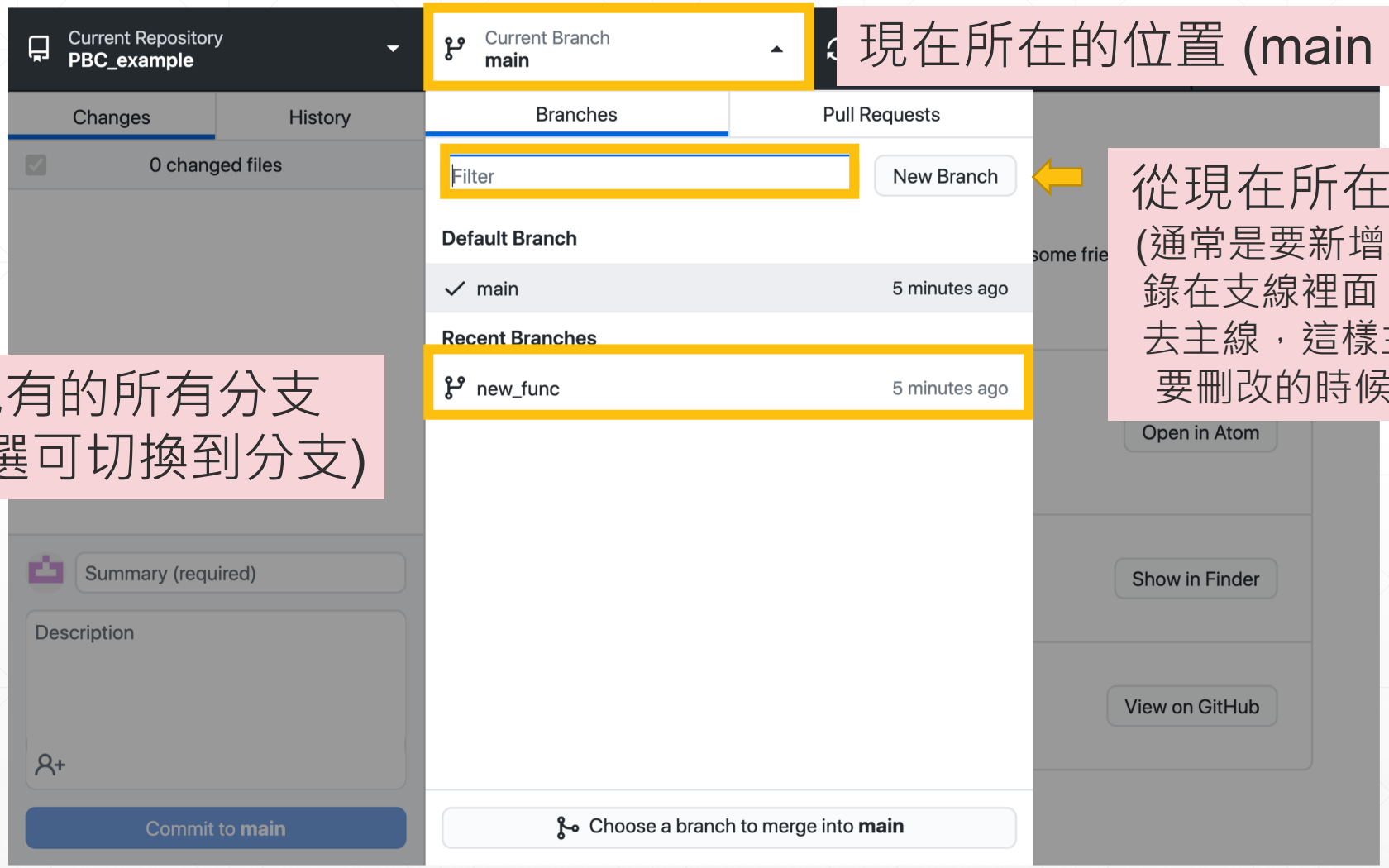
切換到推送紀錄



選擇想回退掉哪一次 commit
右鍵 → Revert this commit
(確認後記得要再 push)

Revert 跟 push 完之後，1.txt 這個檔案會變回沒有~的版本

Branch (分支 / 不想打擾別人用)

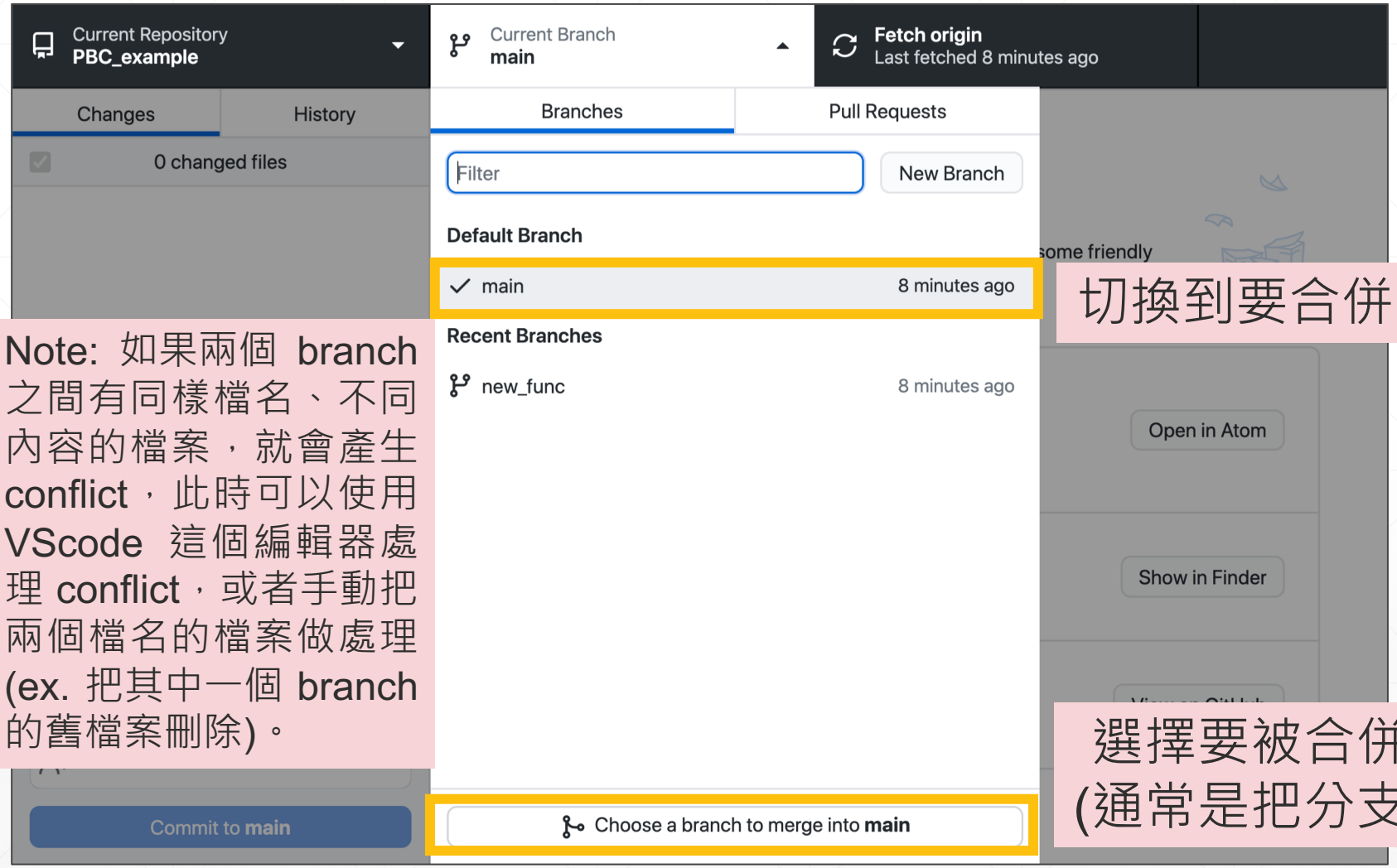


現在所在的位置 (main = 主線)

從現在所在的位置分出去一條
(通常是要新增功能，把寫新東西的過程錄在支線裡面，等到新東西寫好了再併去主線，這樣主線比較不會亂；萬一寫要刪改的時候，也不會動到主線的內容

現有的所有分支
(點選可切換到分支)

Merge (合併)

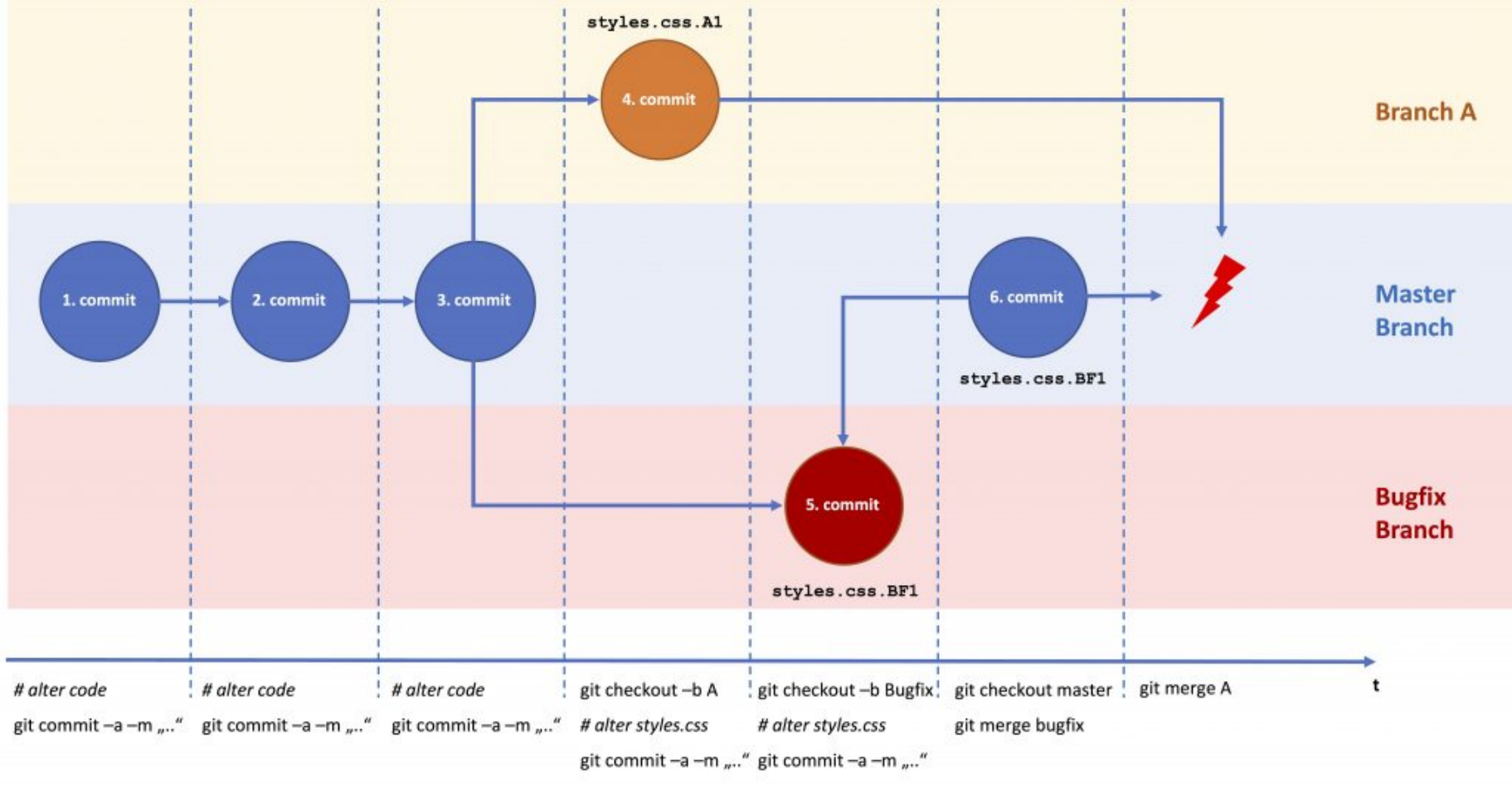


切換到要合併進入的主分支

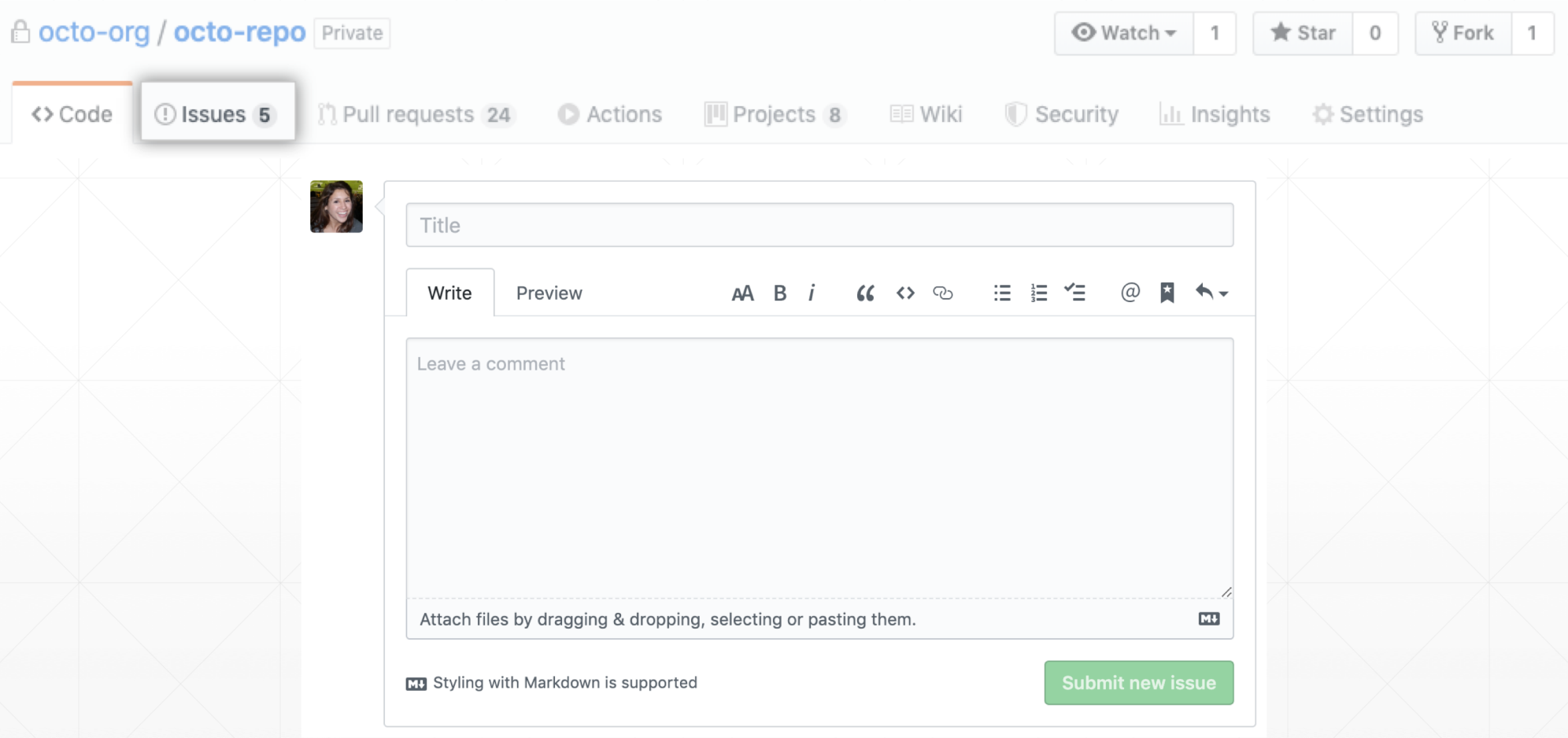
選擇要被合併的旁分支
(通常是把分支併入主線)

Note: 如果兩個 branch 之間有同樣檔名、不同內容的檔案，就會產生 conflict，此時可以使用 VScode 這個編輯器處理 conflict，或者手動把兩個檔名的檔案做處理 (ex. 把其中一個 branch 的舊檔案刪除)。

Conflict


styles.css.A1
4. commit
Branch A
1. commit
2. commit
3. commit
6. commit
styles.css.BF1
Master Branch
5. commit
styles.css.BF1
Bugfix Branch
t
alter code
git commit –a –m „..“
alter code
git commit –a –m „..“
alter code
git commit –a –m „..“
git checkout –b A
alter styles.css
git commit –a –m „..“
git checkout –b Bugfix
alter styles.css
git commit –a –m „..“
git checkout master
git merge bugfix
git merge A

Issue


octo-org / octo-repo
Private
Watch
1
Star
0
Fork
1
Code
Issues 5
Pull requests 24
Actions
Projects 8
Wiki
Security
Insights
Settings
Title
Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting or pasting them.
Styling with Markdown is supported
Submit new issue

其他功能

- Fork
  - 把別人的 repo 分一份出去，變成自己的 repo 自己的進度線
- Pull Request
  - 當你 fork 了別人的 repo、並且做出了修訂之後，可以發送 pull request 給原本的主人，請他參考你的版本
  - 如果他覺得你寫得很好，就可以把你的版本 pull 回去合併

常見開發流程

- 利用 Issue 紀錄需要做的事情
  - E.g. 新增 Issue 13「開發歷史紀錄功能」
- 開一個 branch 處理 X 功能
  - E.g. feature/issue_13、feature/add_history
  - 在 branch 上做，過一兩天後做好功能
- 把 branch merge 回去 master、在 Issue 紀錄完成
  - 也可以用發 Pull Request 的方式將 branch merge 回去 master，順便昭告天下我已經做完了

請 (主動) 做好版本控制

結語

期末報告的建議

- 一定要寫**好**你的 code
  - **要有註解**，變數和 class 名稱不要亂取，依照統一格式 (e.g. PEP 8)
- 做好版本控制
- 多和組員溝通討論
  - 組內互評佔 10%

期末報告的建議

- 如果對自己有信心，可以嘗試接觸 PyCharm 等 IDE
  - Git 整合、內建 PEP 8 排版功能、追蹤變數 / import 流向
- 善用 Google 搜尋和 Stack Overflow 網站
  - 用英文搜尋 python3 *your question*
- **注意死線，不要遲交** (也不能遲交)

更多 GitHub 參考資料

- 連猴子都懂的 Git 入門指南
  https://backlog.com/git-tutorial/tw/
- 五倍紅寶石 - 為你自己學 Git
  https://gitbook.tw/
- Mr. Opengate - Git 與 GitHub 版本控制超簡易教學
  http://mropengate.blogspot.com/2015/04/git-GitHub.html
- Will 保哥 - 30 天精通 Git 版本控管
  https://GitHub.com/doggy8088/Learn-Git-in-30-days
- 助教的萬年書籤：GitLab (另一個 Git 平台) 教你用 Git 的各種復原功能
  https://docs.gitlab.com/ee/topics/git/numerous_undo_possibilities_in_git/