

商管程式設計 108-1

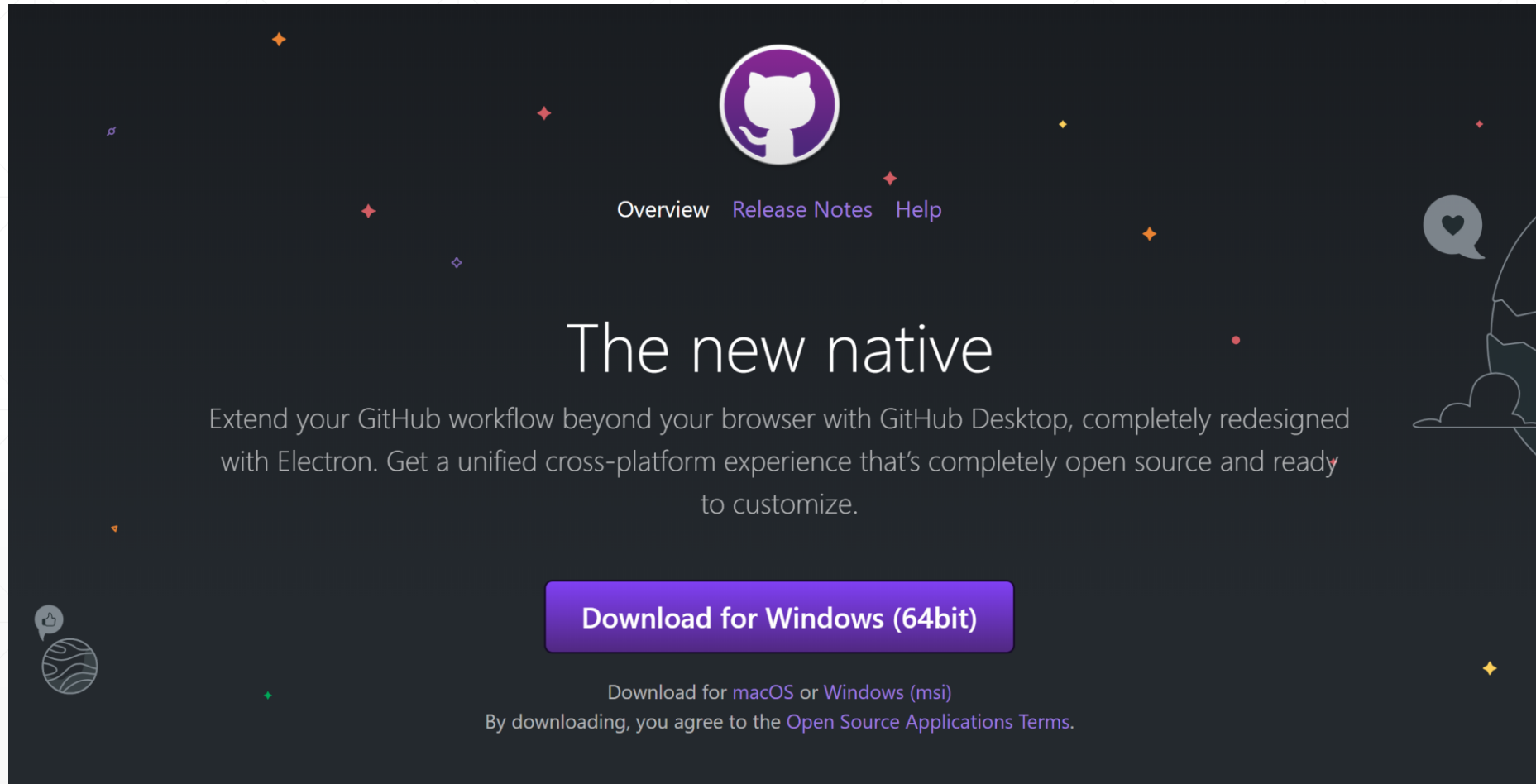
GitHub Workshop

2019/12/11-12

Agenda

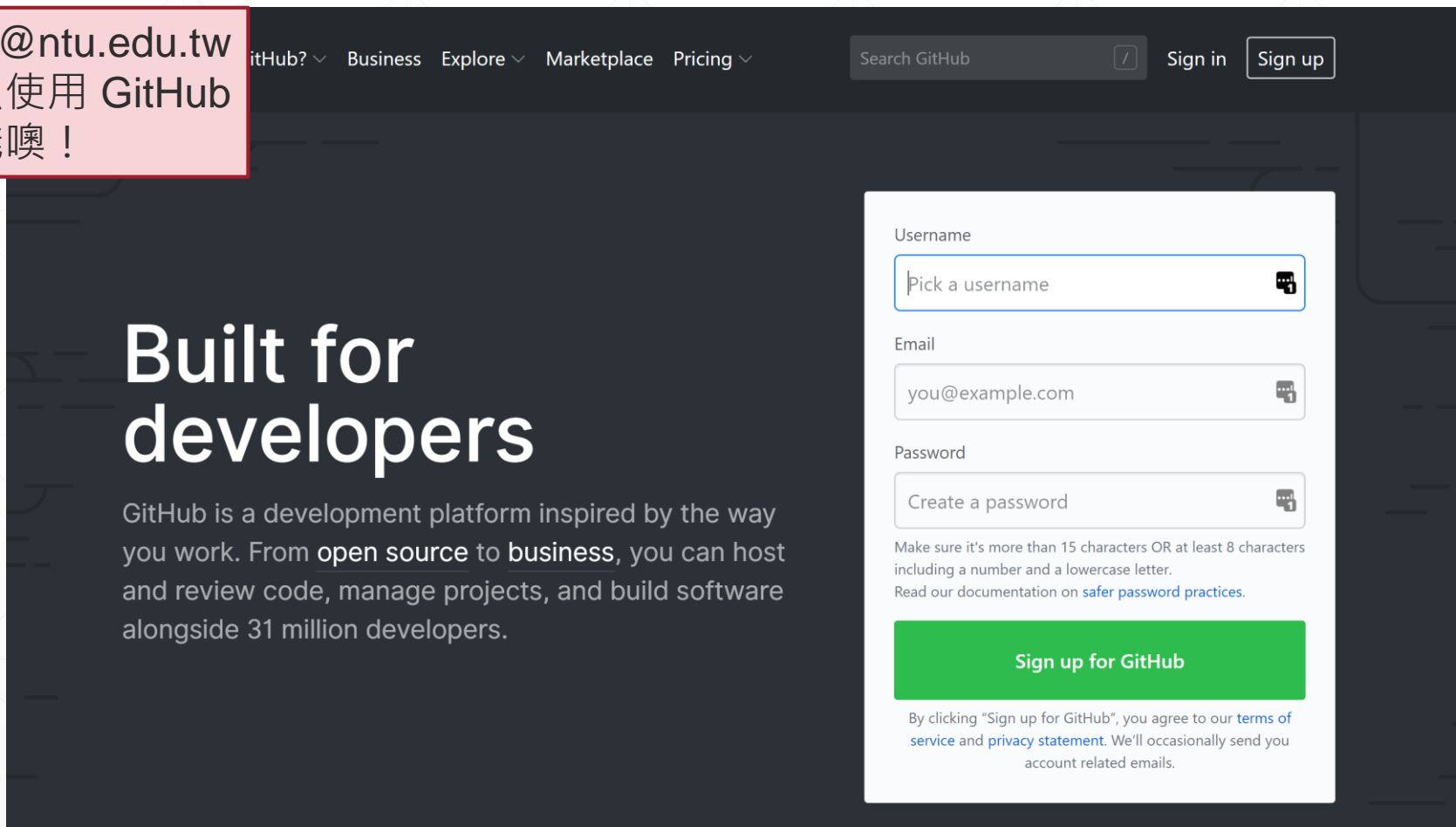
- 版本控制
- GitHub
 - 使用 GitHub
 - GitHub Desktop 操作
- 結語

節省時間，先下載 <https://desktop.github.com/>



節省時間，先註冊 <https://github.com/>

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



The screenshot shows the GitHub website's sign-up interface. At the top, there's a navigation bar with links like 'GitHub?', 'Business', 'Explore', 'Marketplace', and 'Pricing'. A search bar and 'Sign in'/'Sign up' buttons are also present. The main content area features the text 'Built for developers' and a description of GitHub as a development platform. On the right, there's a sign-up form with fields for 'Username', 'Email', and 'Password'. The 'Username' field has a placeholder 'Pick a username'. The 'Email' field has a placeholder 'you@example.com'. The 'Password' field has a placeholder 'Create a password' and a note about password requirements. A green 'Sign up for GitHub' button is at the bottom of the form. Below the button, there's a disclaimer about agreeing to terms of service and privacy statement.

Username

Pick a username

Email

you@example.com

Password

Create a password

Make sure it's more than 15 characters OR at least 8 characters including a number and a lowercase letter.
Read our documentation on [safer password practices](#).

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy statement](#). We'll occasionally send you account related emails.

版本控制

你自己寫 code



Project.py



Project_2.py



Project_3.py



Project_4.py



Project_5.py



Project_final
.py

大家一起寫 code



Project.py



Project_2.py



Project_3.py



Project
(2).py



Project (複製)
.py



Project_final
.py



Project_final
_2.py



Project_final
_修正.py



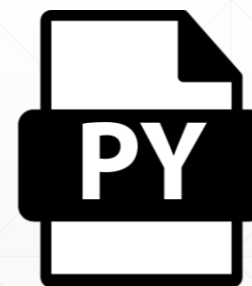
Project_final
_2 (複製).py



Project_3
(複製) fix.py



Project 做完了
.py

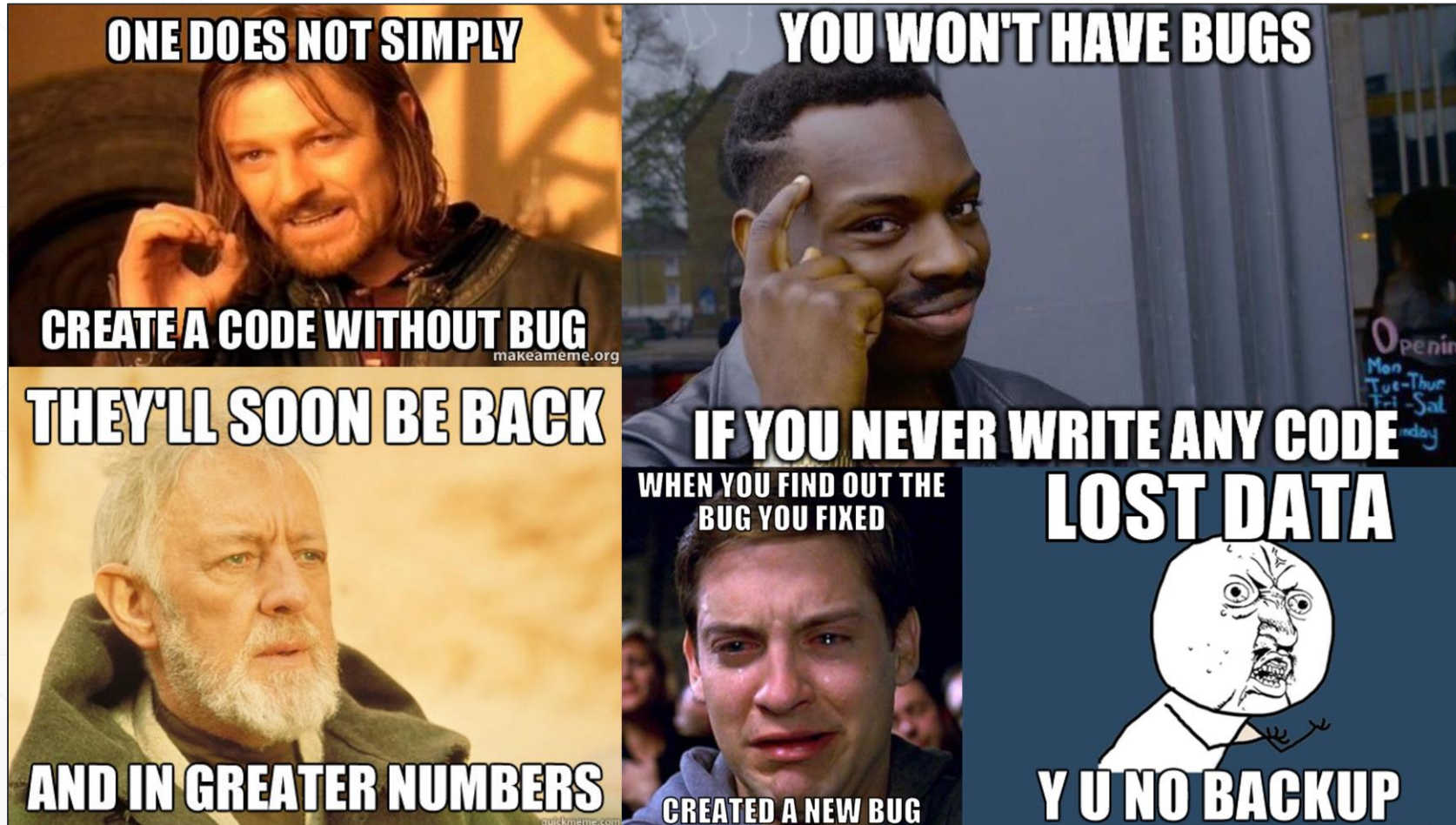


Project_final
_修正 (2).py

大家一起寫 code (進階)

- 我改了昨天晚上的 code，你也改了，都傳到群組
- 隔天又有人改了，但忘記是用誰的版本下去改的
- 一個禮拜後發現 code 寫錯了，不知道誰寫的
- 為了要把 code 改過來，改掉某一段
- 又過了一個禮拜，發現改掉的那一段造成另一個功能壞掉
- 去群組找備份，裡面有 20 個檔案，不知道哪個是哪個
-

你的 code 一塌糊塗，而且沒有人愛你



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



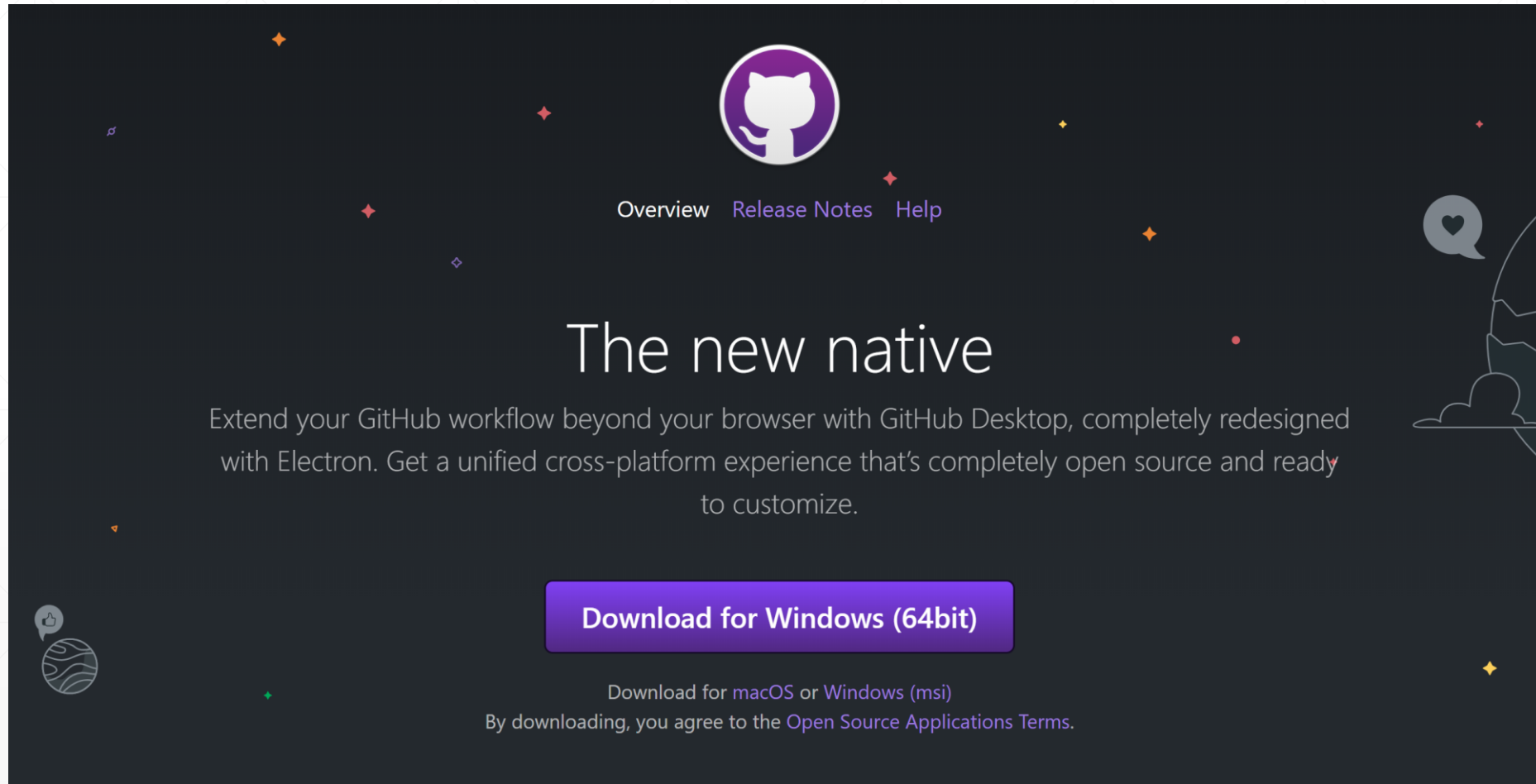
SourceForge

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 (然後被採納回去)
 - 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

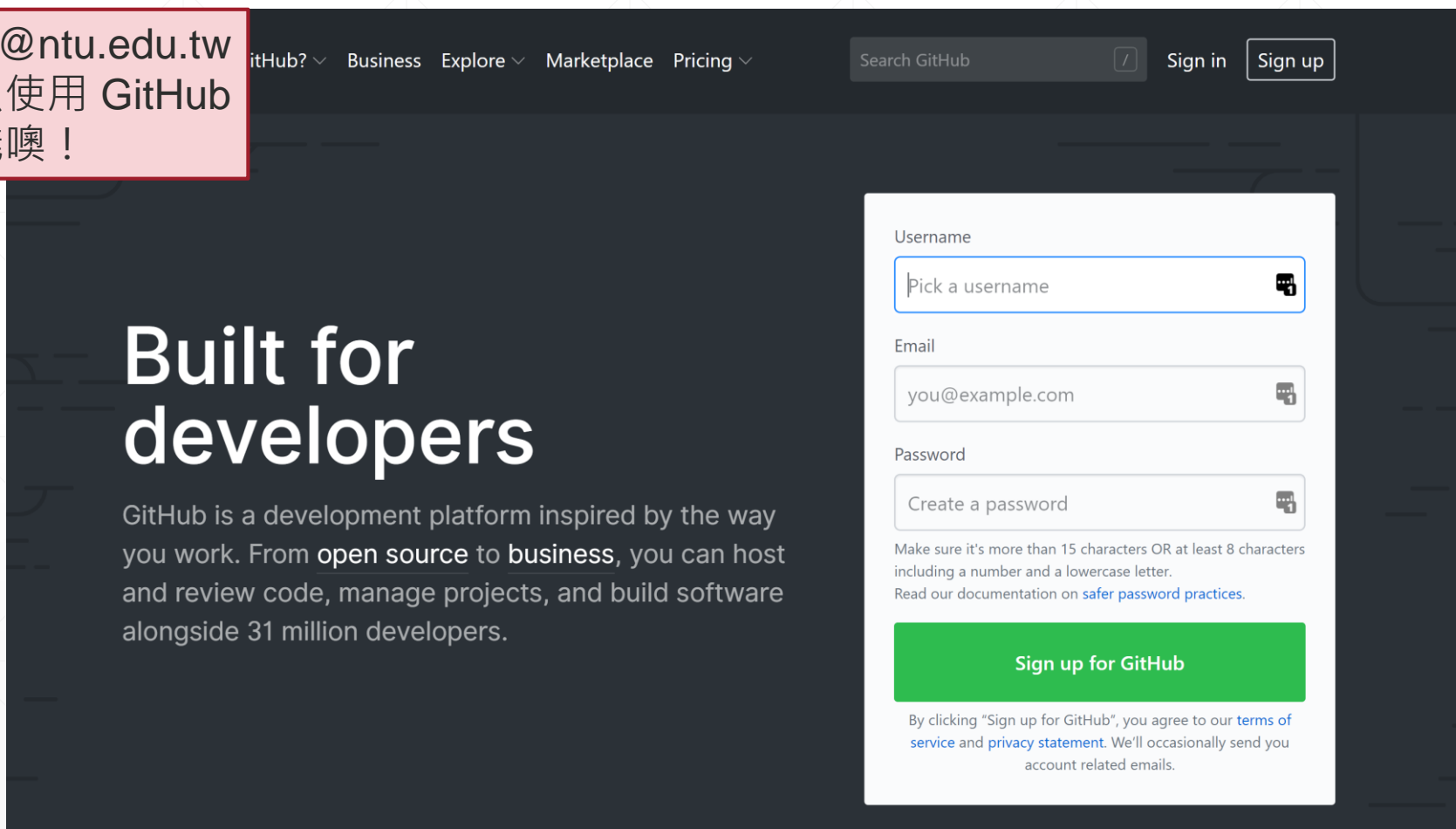
使用 GitHub

下載 <https://desktop.github.com/>



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！



The screenshot shows the GitHub sign-up page. The header includes navigation links like 'GitHub?', 'Business', 'Explore', 'Marketplace', and 'Pricing', along with a search bar and 'Sign in'/'Sign up' buttons. The main content area features the text 'Built for developers' and a description of GitHub as a development platform. On the right, there is a sign-up form with fields for Username, Email, and Password, each with a strength indicator icon. Below the password field, there is a green 'Sign up for GitHub' button and a disclaimer about agreeing to terms of service and privacy statement.

Username

Pick a username

Email

you@example.com

Password

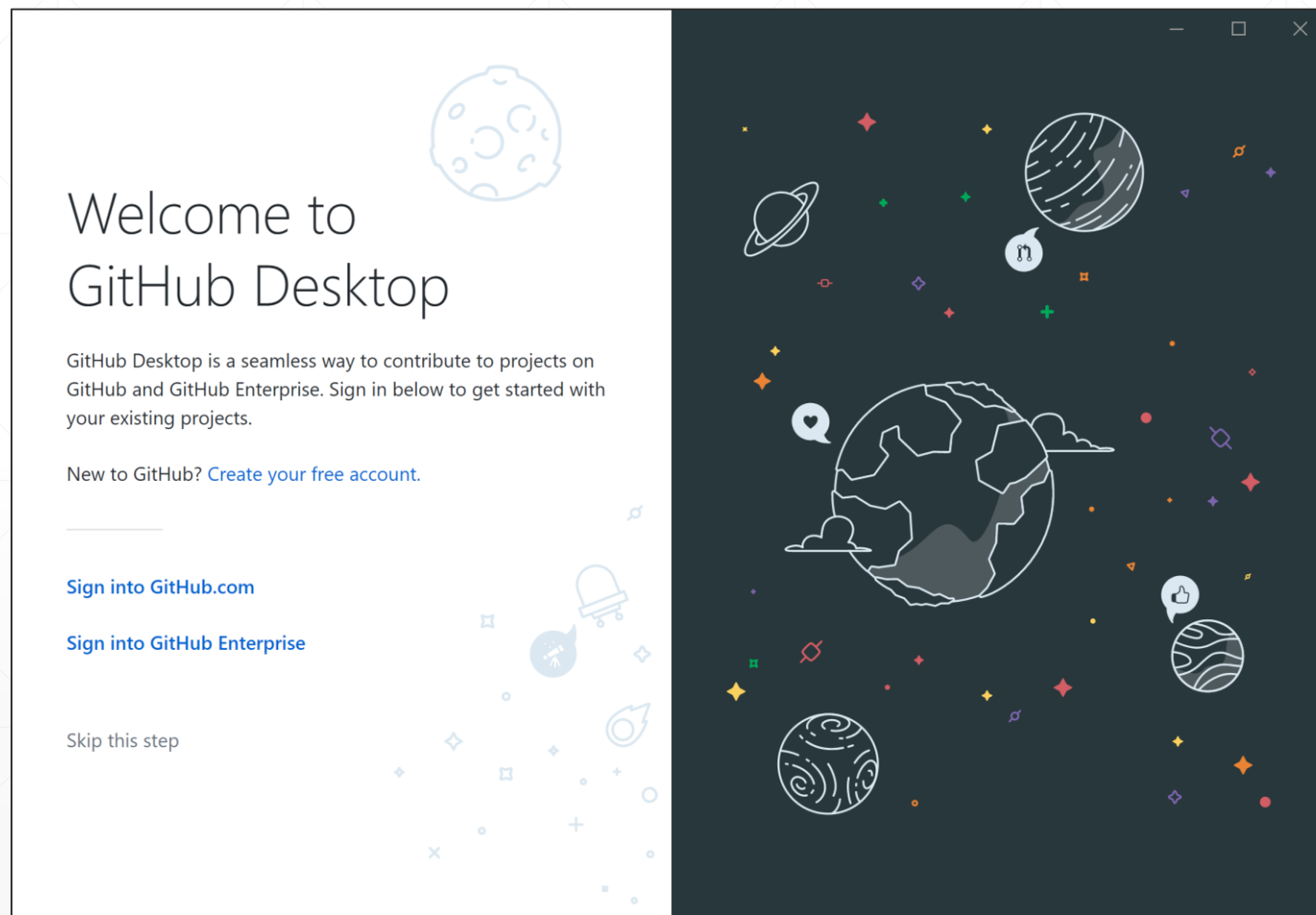
Create a password

Make sure it's more than 15 characters OR at least 8 characters including a number and a lowercase letter.
Read our documentation on [safer password practices](#).

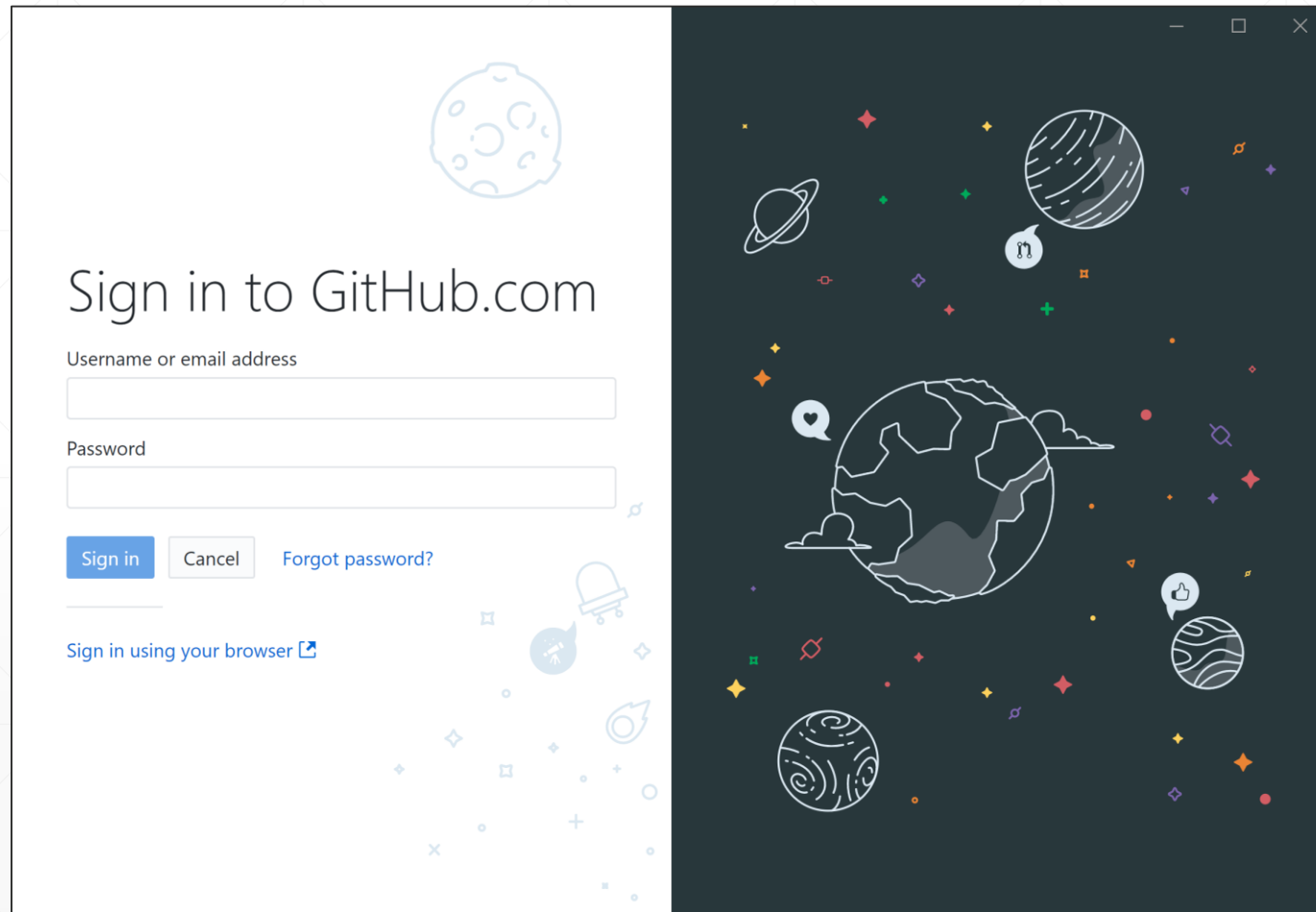
Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy statement](#). We'll occasionally send you account related emails.

打開



登入



The image shows the GitHub login page with a space-themed illustration. The left side is a white panel with the GitHub logo (a blue octocat) at the top. Below it, the text "Sign in to GitHub.com" is displayed. Underneath, there are two input fields: "Username or email address" and "Password". Below the password field are three buttons: "Sign in" (blue), "Cancel" (gray), and "Forgot password?" (blue). At the bottom of the white panel, there is a link "Sign in using your browser" with an external link icon. The right side of the image is a dark space-themed illustration with various celestial bodies, including a large planet with a ring, a planet with a heart, a planet with a thumbs up, and several smaller planets and stars. The background is a dark blue/black space with white stars and colorful geometric shapes.

Sign in to GitHub.com

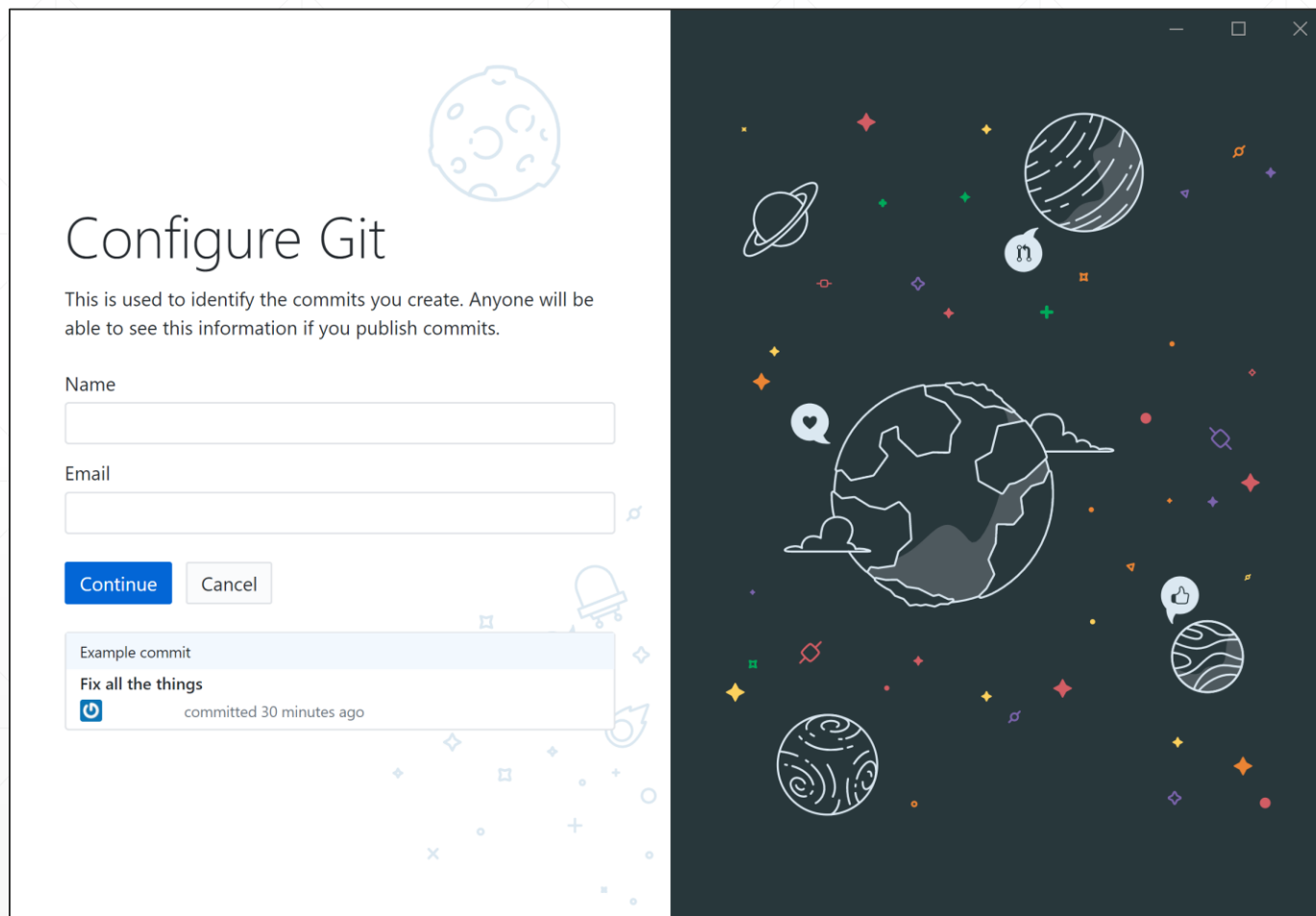
Username or email address

Password

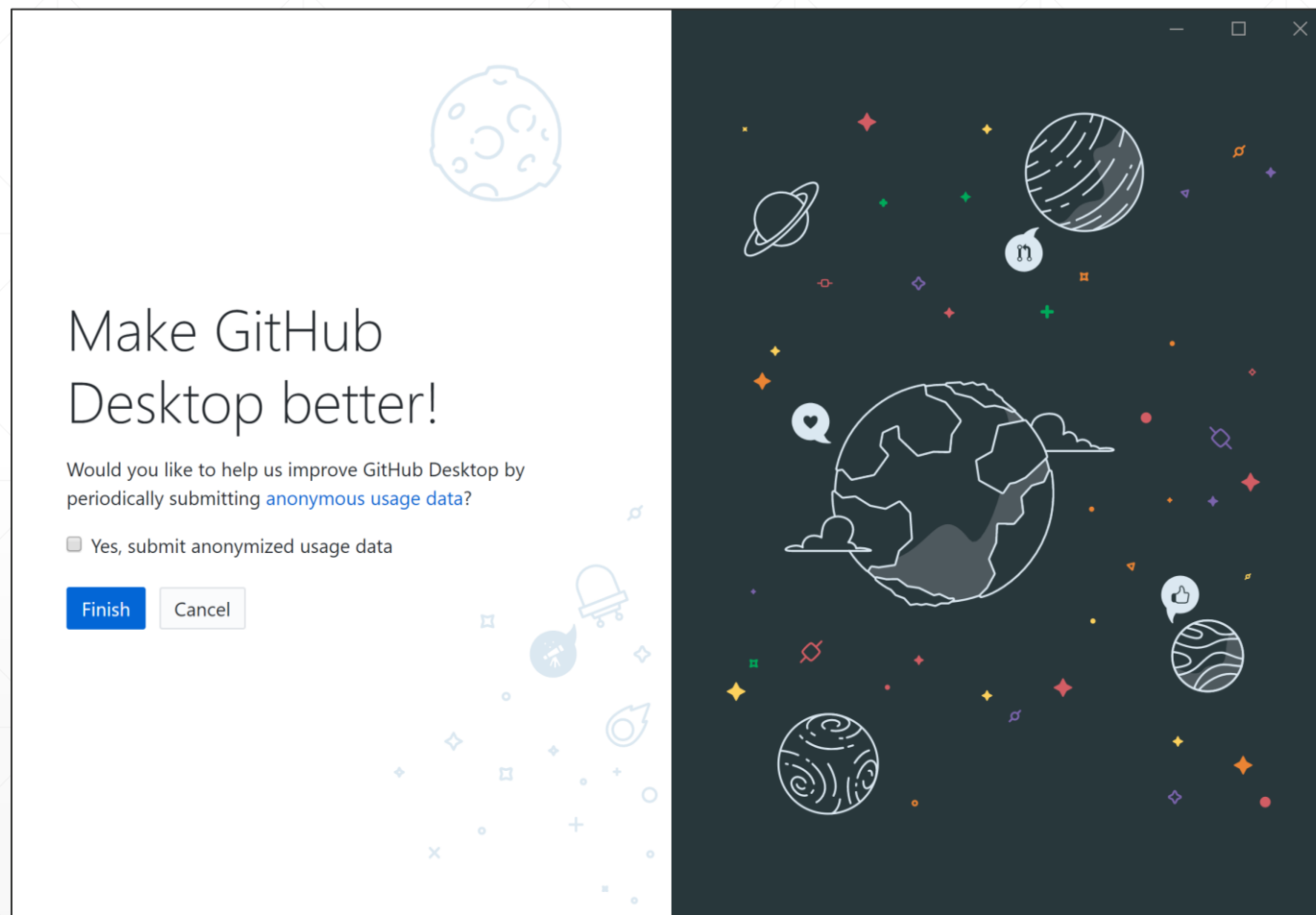
Sign in Cancel Forgot password?

Sign in using your browser

設定自己的名字

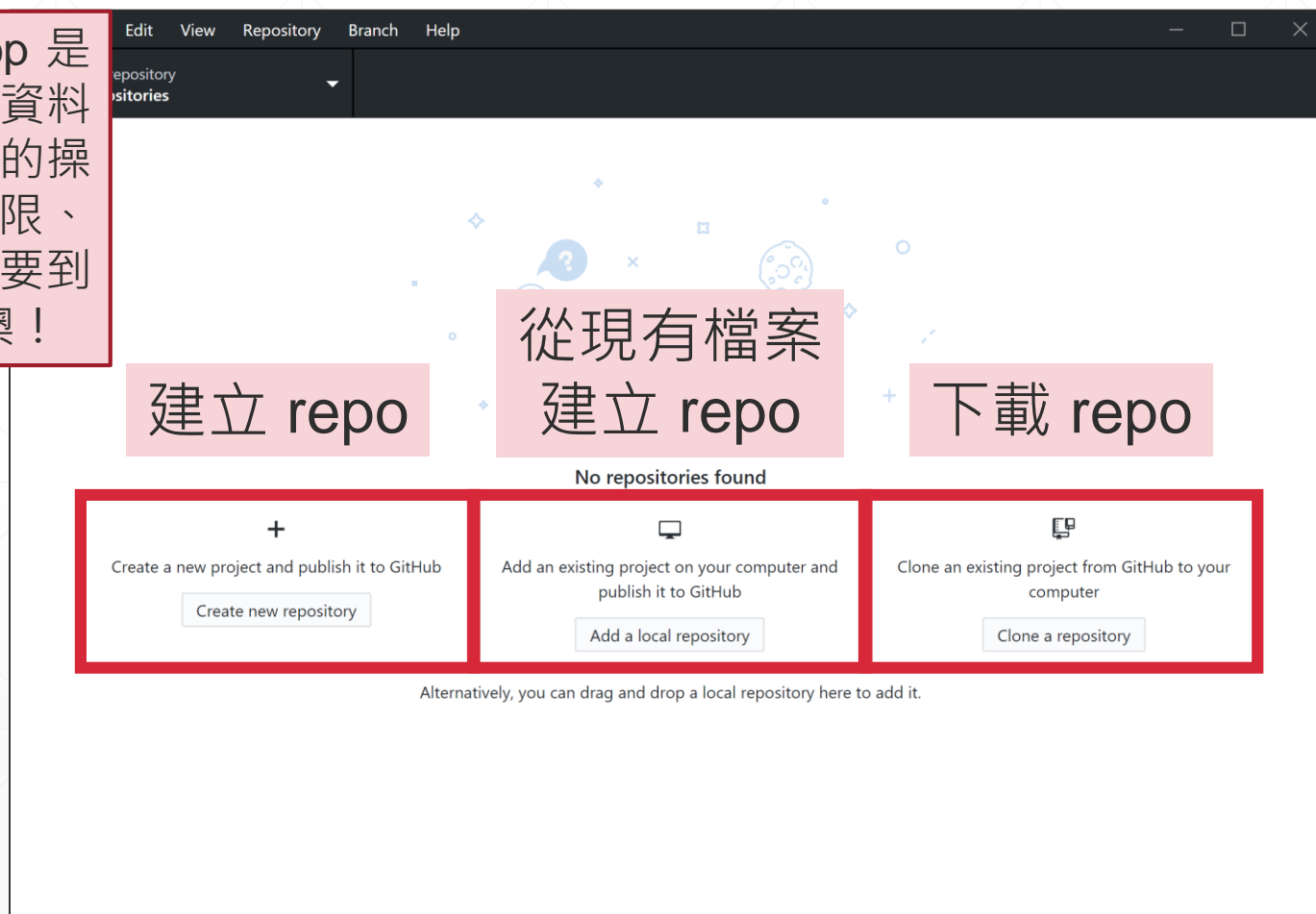


設定結束



主畫面

Note: GitHub desktop 是幫助掌管電腦上同步資料的程式，有些純雲端的操作 (ex. 設定上傳權限、刪除雲端資料) 仍然要到 GitHub 網站上操作噢！



甚麼是 repository (repo) ?

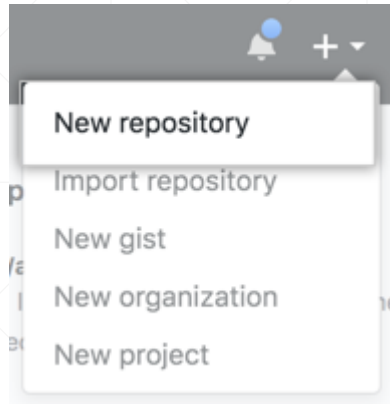
- 等於一個**線上倉庫**
 - 一個專案 or 一個共享資料夾
- 把需要版本控制的東西通通丟上去：.py .html.....
 - 不用版本控制的也通通丟上去：.mp3 .jpg .csv.....
 - 反正倉庫裡面想放甚麼就放甚麼 (放不用錢)
- 所有 repo 都是**公開**的
 - GitHub 的不公開 repo 要錢 or 要學生帳號 or 不要用 GitHub

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

Create Repository (網頁操作)




Create a new repository


A repository contains all the files for your project, including the revision history.

新 repo 的名字和敘述

Owner

 octocat

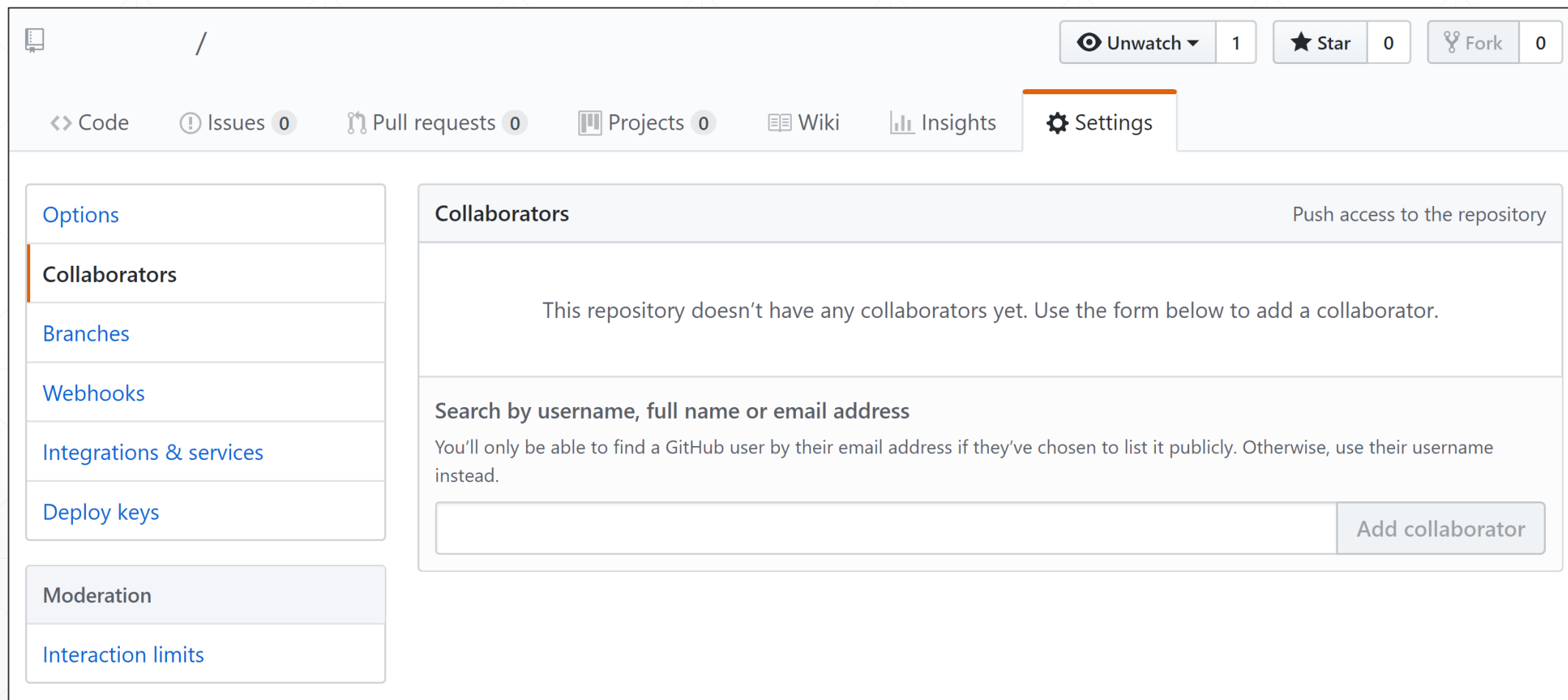
Repository name

hello-world 

Great repository names are short and memorable. Need inspiration? How about **potential-eureka**.

Description (optional)

Collaborators (共編成員)



The screenshot shows the GitHub repository settings page for a repository. The top navigation bar includes links for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings (selected). On the right, there are buttons for Unwatch (1), Star (0), and Fork (0). The left sidebar contains a list of settings: Options, Collaborators (selected), Branches, Webhooks, Integrations & services, Deploy keys, Moderation, and Interaction limits. The main content area is titled 'Collaborators' and includes a sub-header 'Push access to the repository'. Below this, a message states: 'This repository doesn't have any collaborators yet. Use the form below to add a collaborator.' A search bar is provided with the text 'Search by username, full name or email address' and a note: 'You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.' The search bar is followed by an 'Add collaborator' button.

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Options

Collaborators

Branches

Webhooks

Integrations & services

Deploy keys

Moderation

Interaction limits

Collaborators Push access to the repository

This repository doesn't have any collaborators yet. Use the form below to add a collaborator.

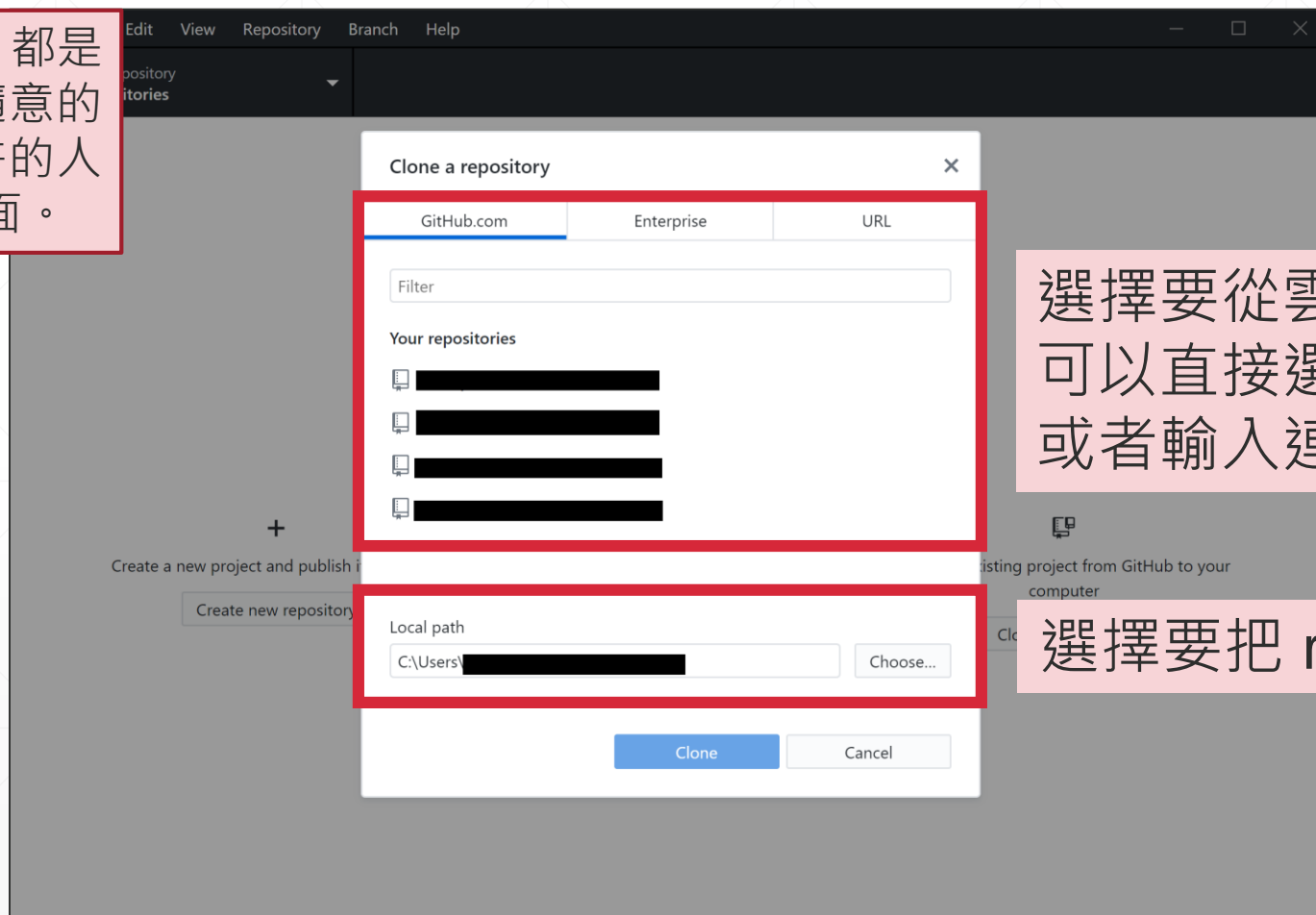
Search by username, full name or email address

You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.

Add collaborator

Clone (從雲端下載 repo)

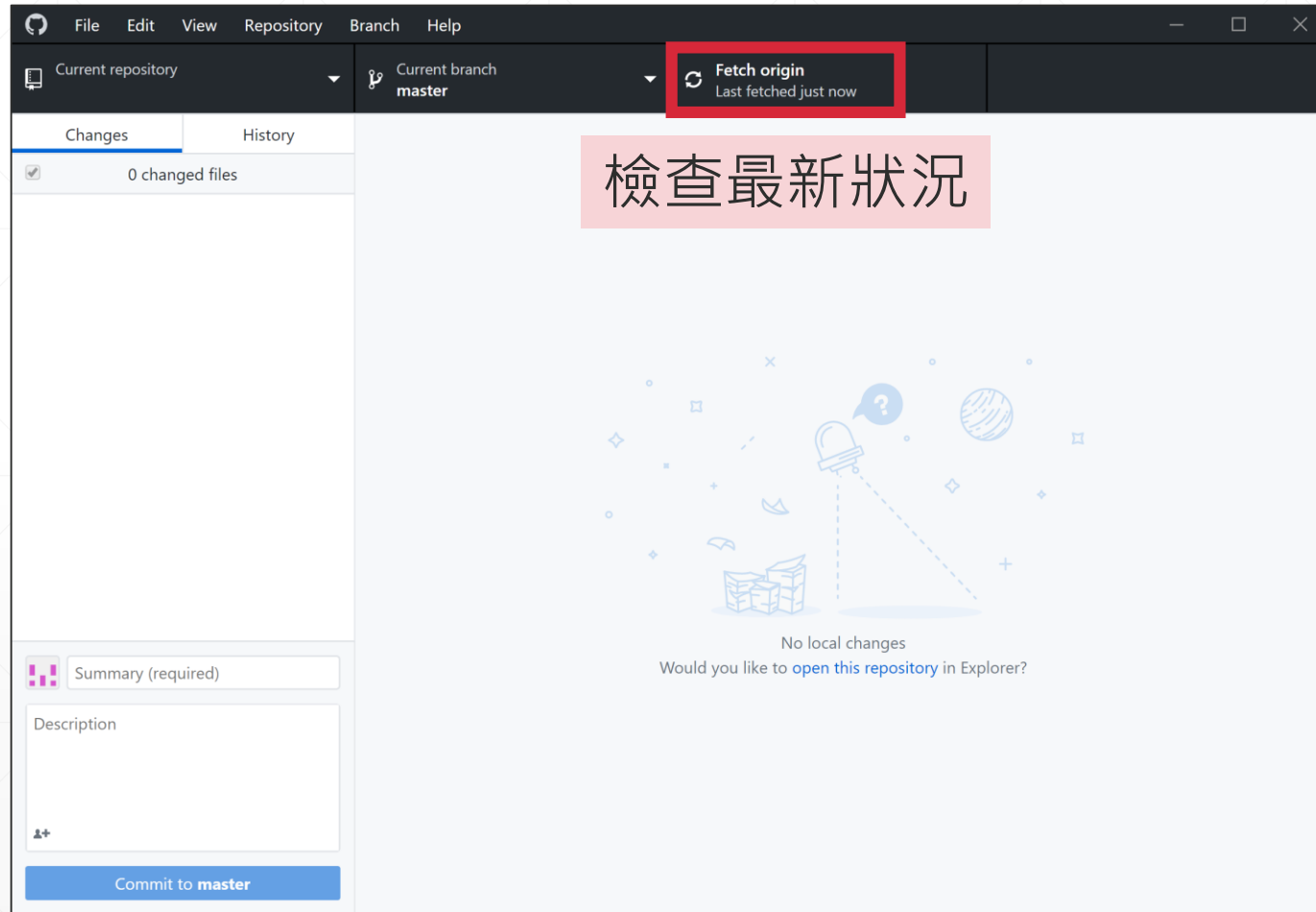
Note: 因為所有 repo 都是公開的，所以可以隨意的下載，但只有被允許的人才能上傳到 repo 裡面。



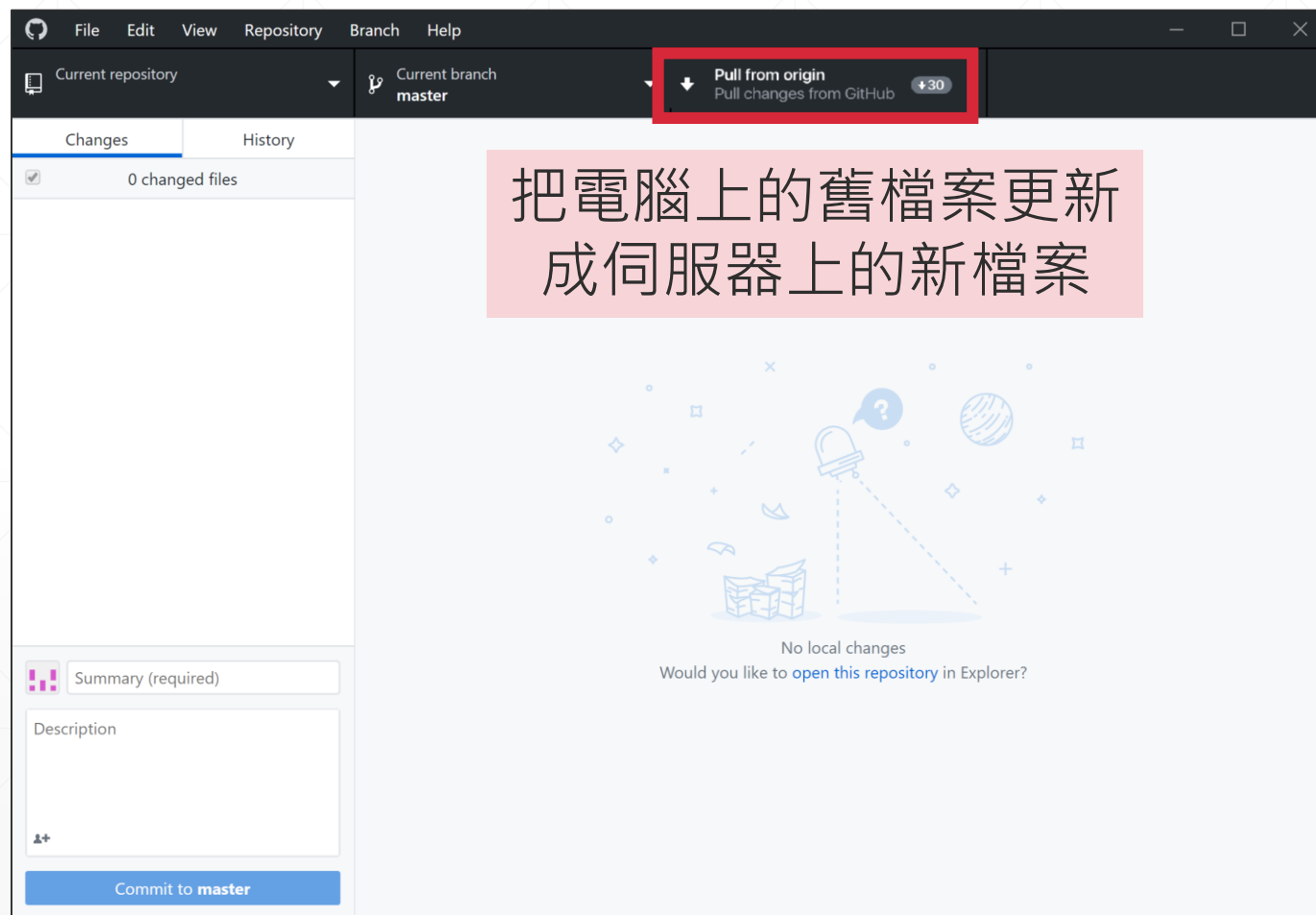
選擇要從雲端下載的 repo 可以直接選取自己的，或者輸入連結下載別人的

選擇要把 repo 存在哪裡

Fetch (檢查他人進度)



Pull (從雲端更新到電腦)

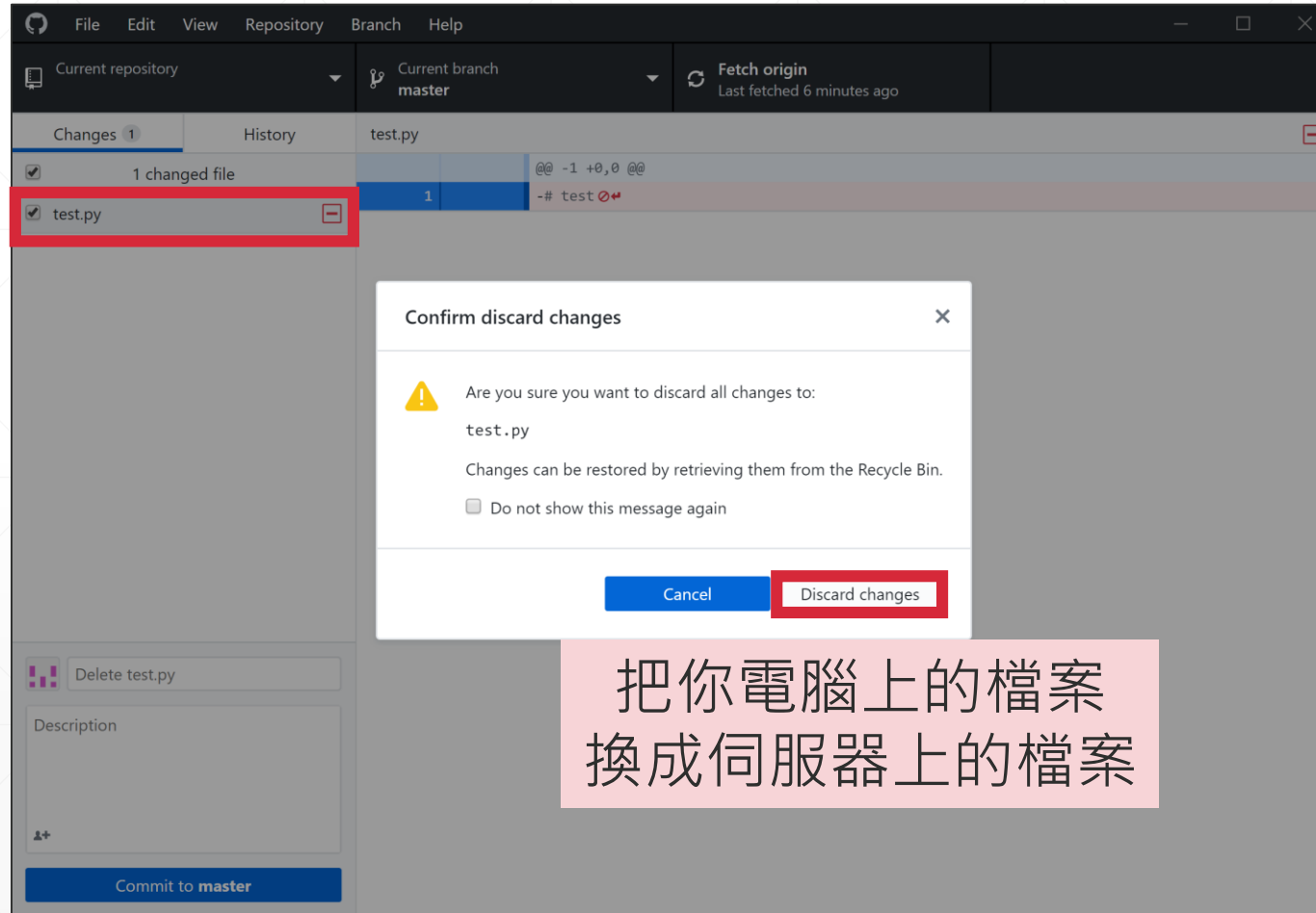


開始打 code

```
31     def __init__(self, settings):
32         self.file = None
33         self.fingerprints = set()
34         self.logdupes = True
35         self.debug = debug
36         self.logger = logging.getLogger(__name__)
37         if path:
38             self.file = open(os.path.join(path, 'requests.log'),
39                             'a')
40             self.file.seek(0)
41             self.fingerprints.update(self.request_fingerprint(request) for request in self.requests)
42
43     @classmethod
44     def from_settings(cls, settings):
45         debug = settings.getbool('SUPERFILTER_DEBUG')
46         return cls(job_dir(settings), debug)
47
48     def request_seen(self, request):
49         fp = self.request_fingerprint(request)
50         if fp in self.fingerprints:
51             return True
52         self.fingerprints.add(fp)
53         if self.file:
54             self.file.write(fp + os.linesep)
55
56     def request_fingerprint(self, request):
57         return request_fingerprint(request)
```

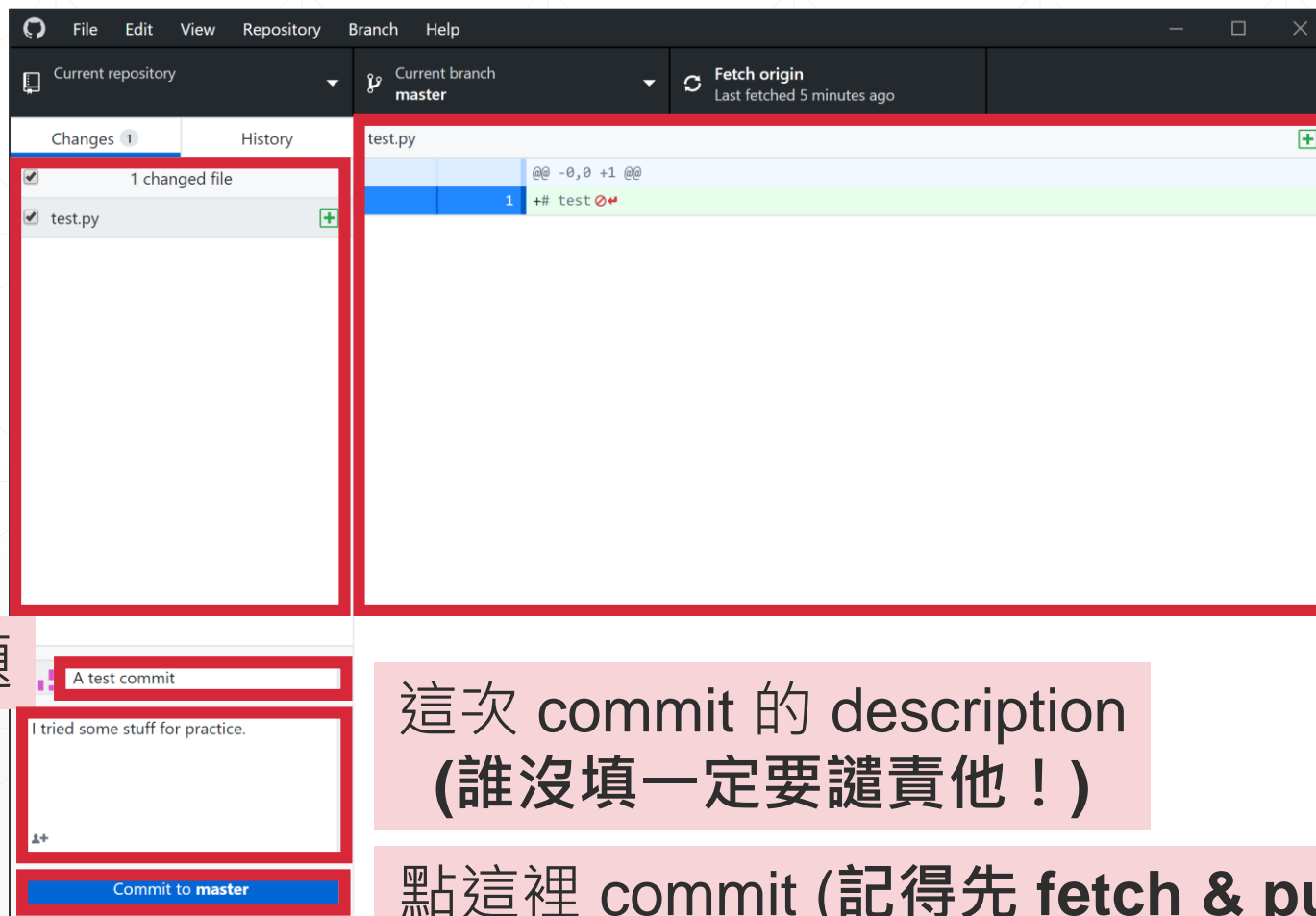
Discard Changes (取消修改)

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



把你電腦上的檔案
換成伺服器上的檔案

Commit (設定存檔點)



這次 commit 更新的內容清單

檔案增刪狀況

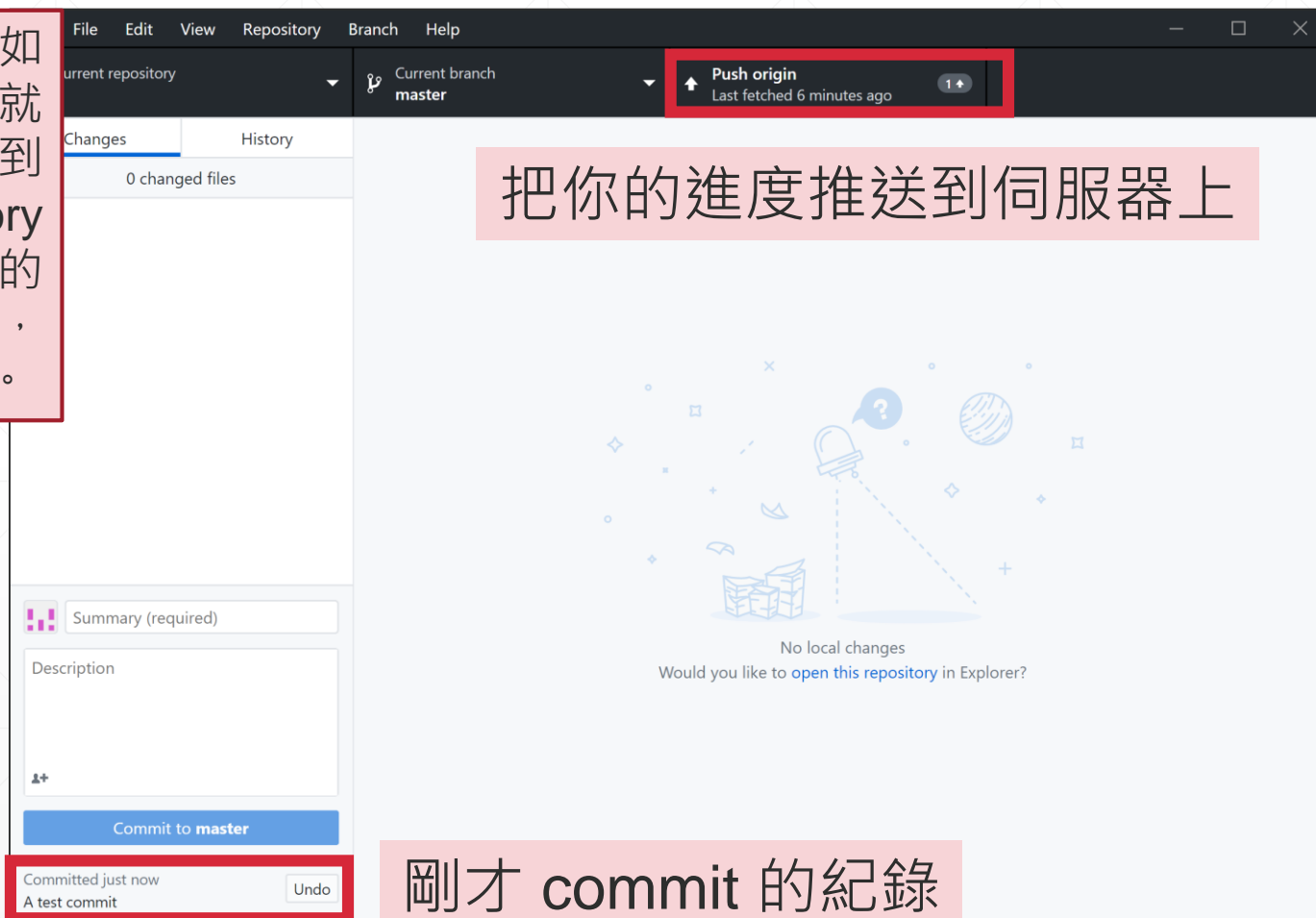
這次 commit 的標題

這次 commit 的 description
(誰沒填一定要譴責他！)

點這裡 commit (記得先 fetch & pull !)

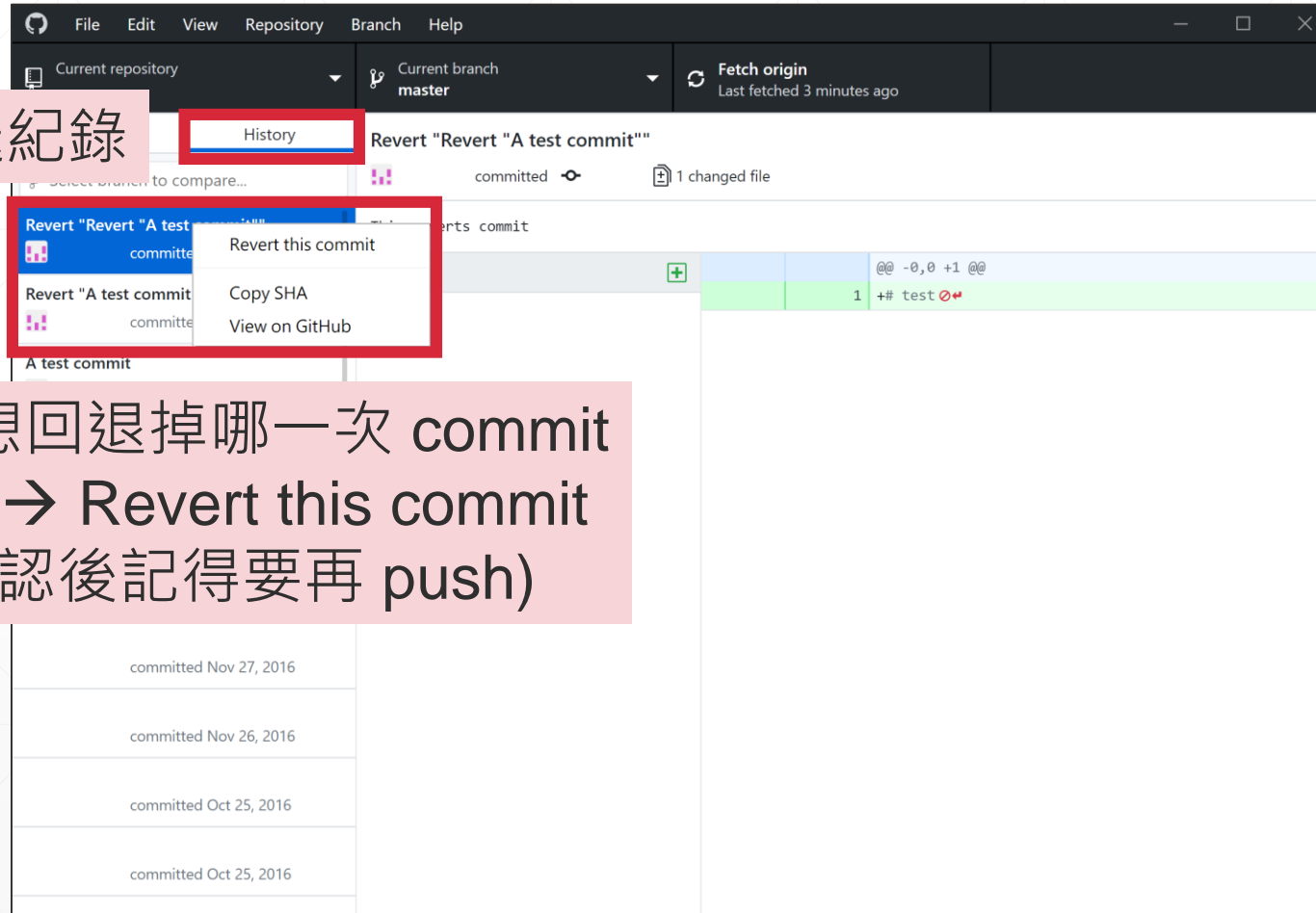
Push (推送 / 存檔到伺服器)

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。



Revert (回退 commit)

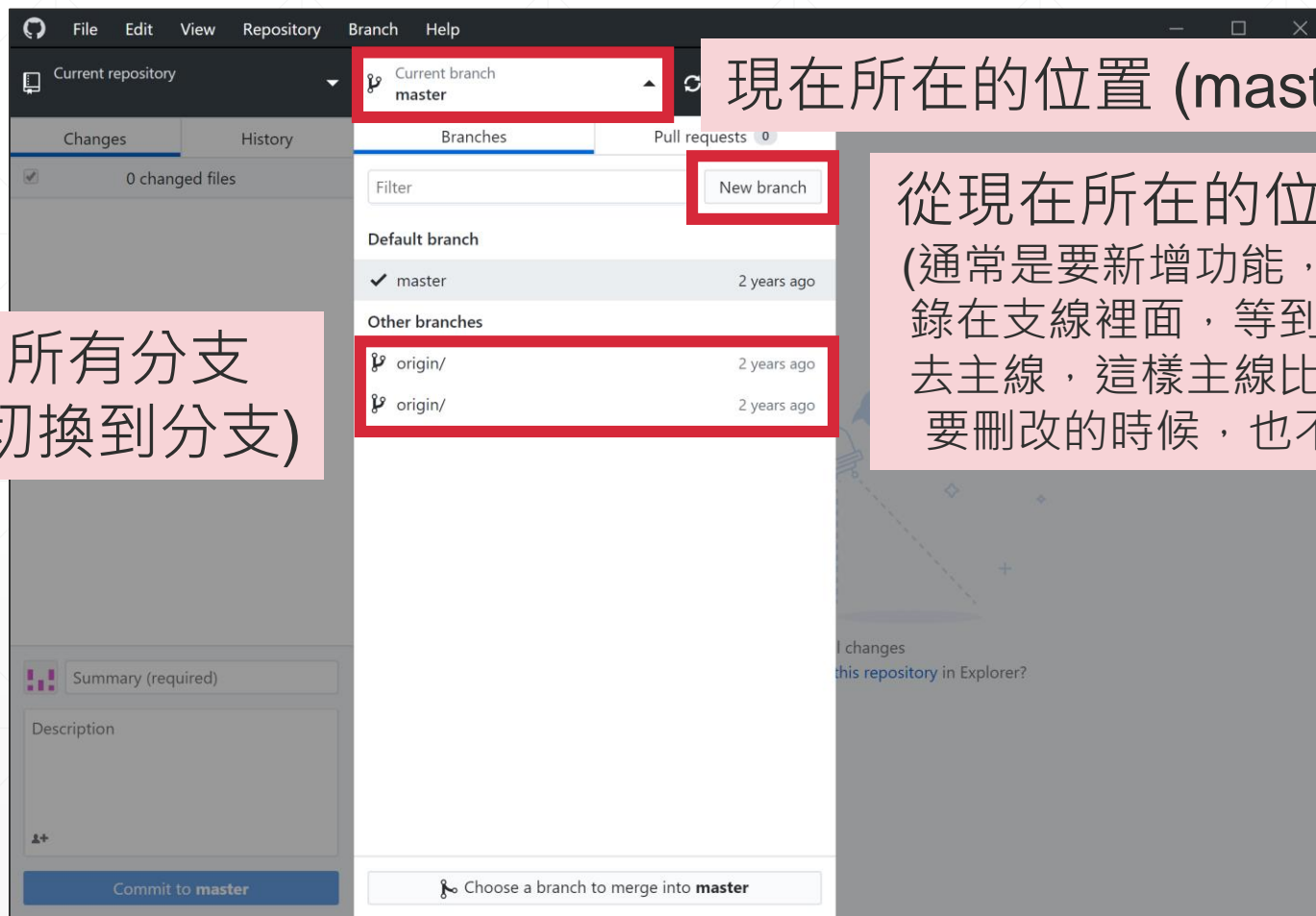
切換到推送紀錄



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

Branch (分支 / 不想打擾別人用)

現有的所有分支
(點選可切換到分支)

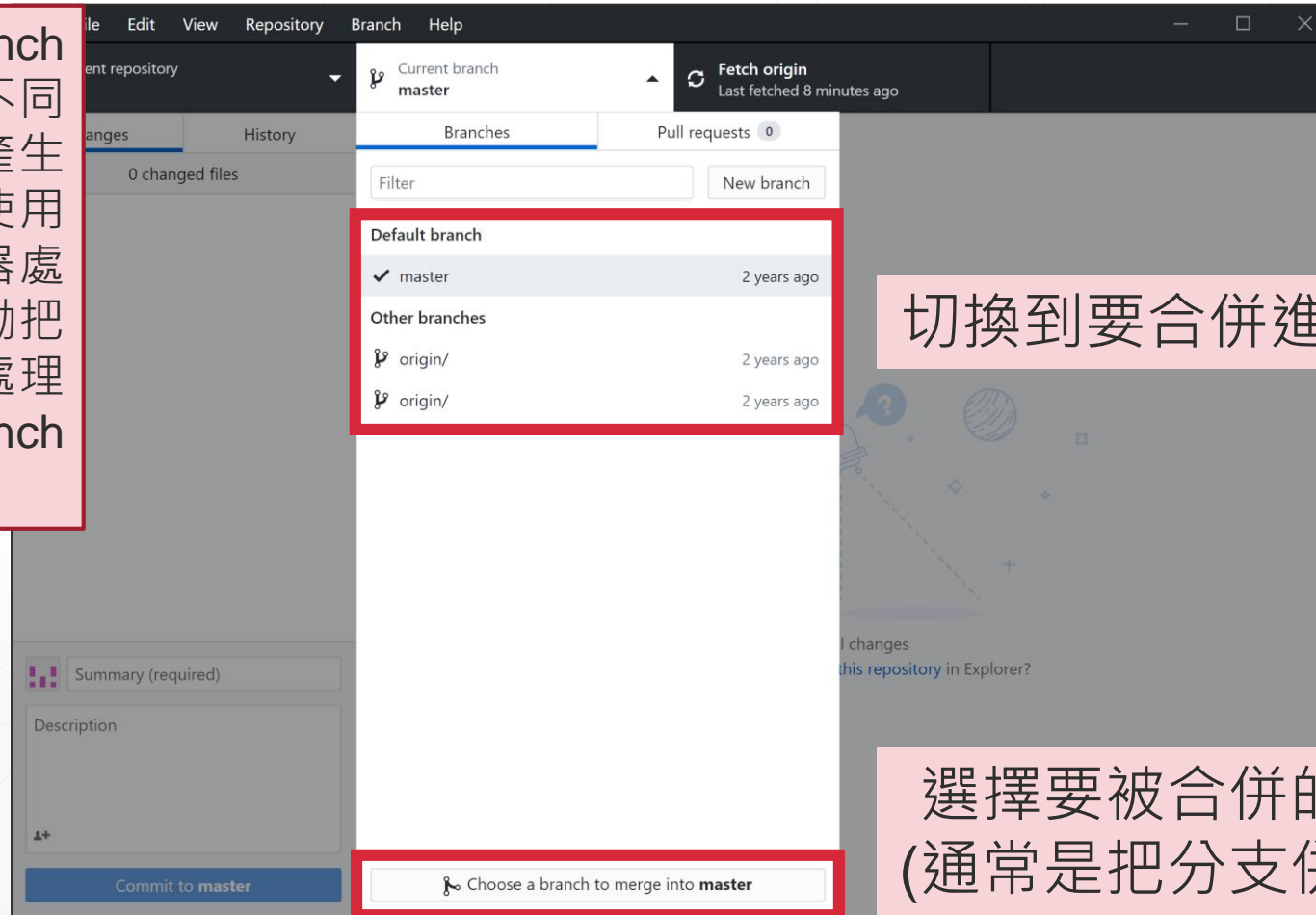


現在所在的位置 (master = 主線)

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

Merge (合併)

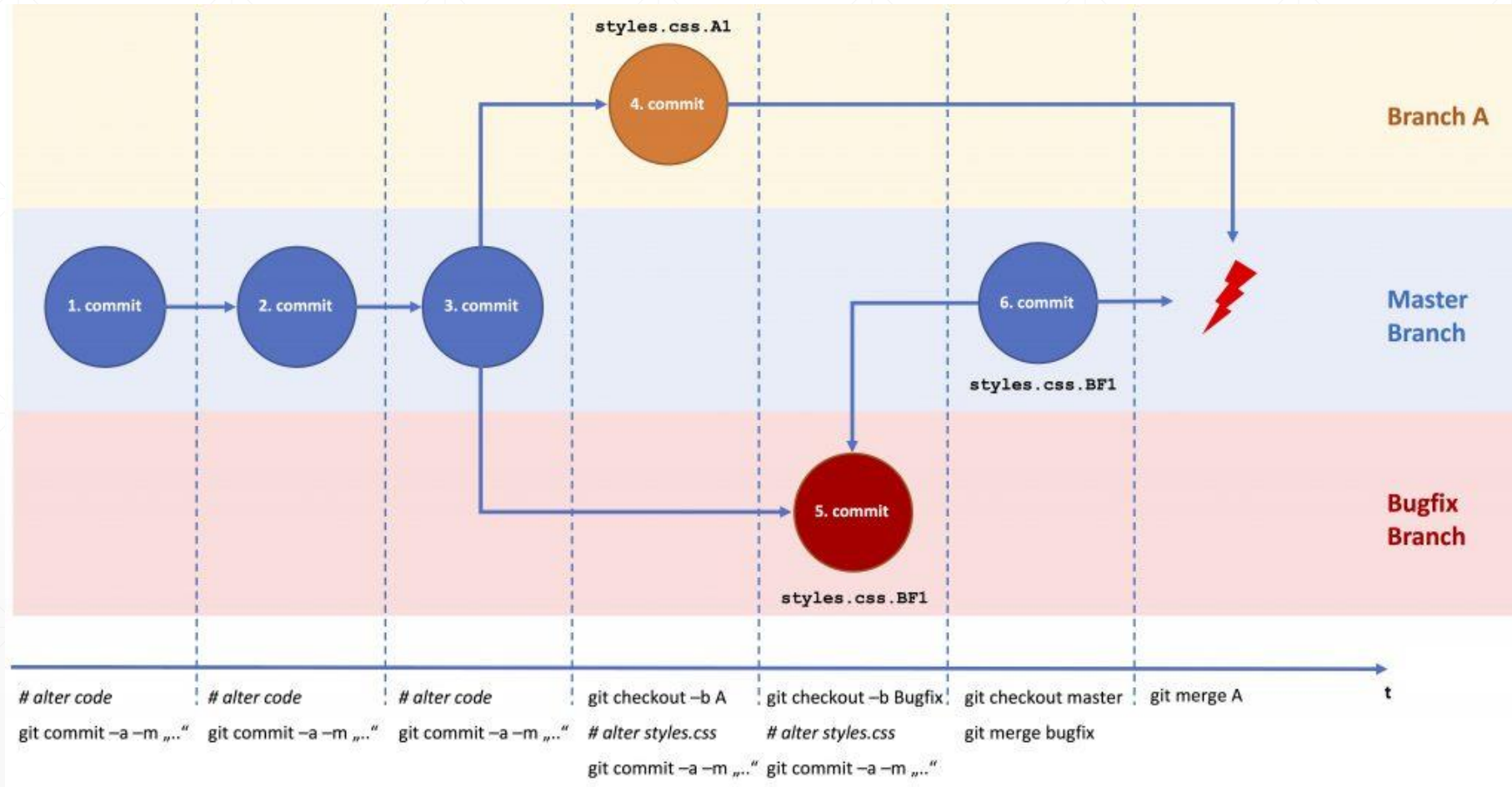
Note: 如果兩個 branch 之間有同樣檔名、不同內容的檔案，就會產生 conflict，此時可以使用 VScode 這個編輯器處理 conflict，或者手動把兩個檔名的檔案做處理 (ex. 把其中一個 branch 的舊檔案刪除)。




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)

Conflict




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](#)




Write


Preview

AA B i “ <> 🔗 ☰ ☷ ✓ @ ★ ↶

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)
 - Code 的用途是對電腦下指令 & 和人類溝通
- 做好版本控制
- 多和組員溝通討論，口頭報告要好好準備
 - 組內互評估 20% 報告成績，全班互評估 15%

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/