Git與GitHub

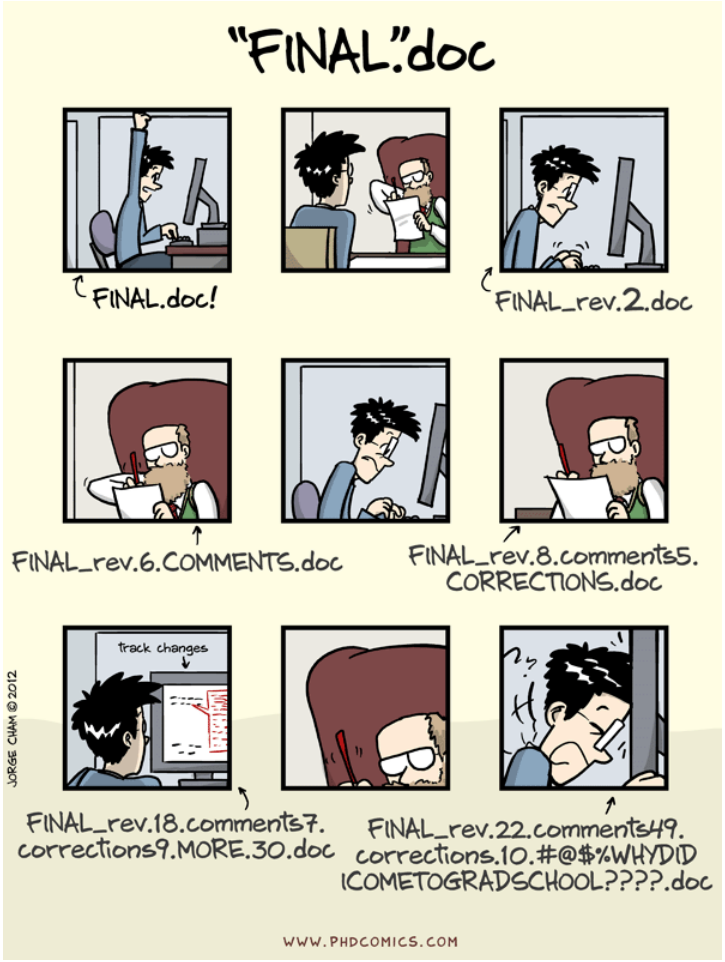
原始教材編撰(SourceTree)：逢甲大學資訊工程學系 陳錫民 教授

課程改編講授(GitHub Desktop)：臺灣海洋大學資訊工程系 馬尚彬 教授

https://www.openedu.tw/course?id=1566

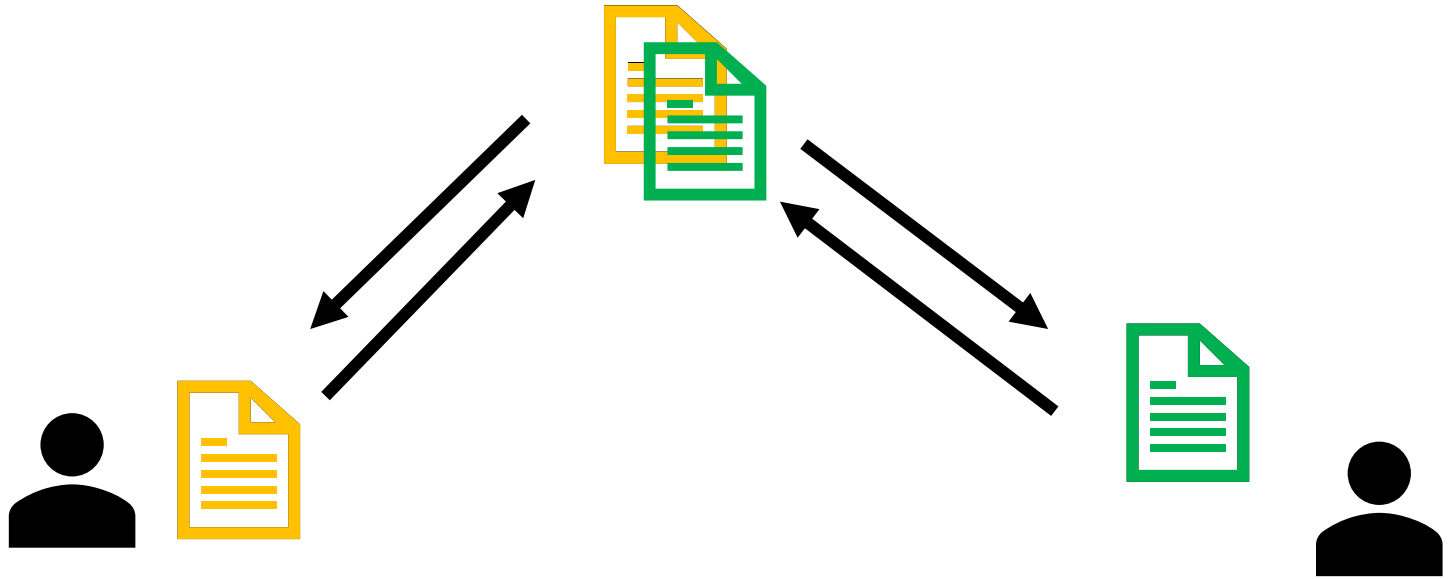
- **為什麼需要版本控制？**

  當你撰寫報告時，如果你想備份，你會怎麼做？



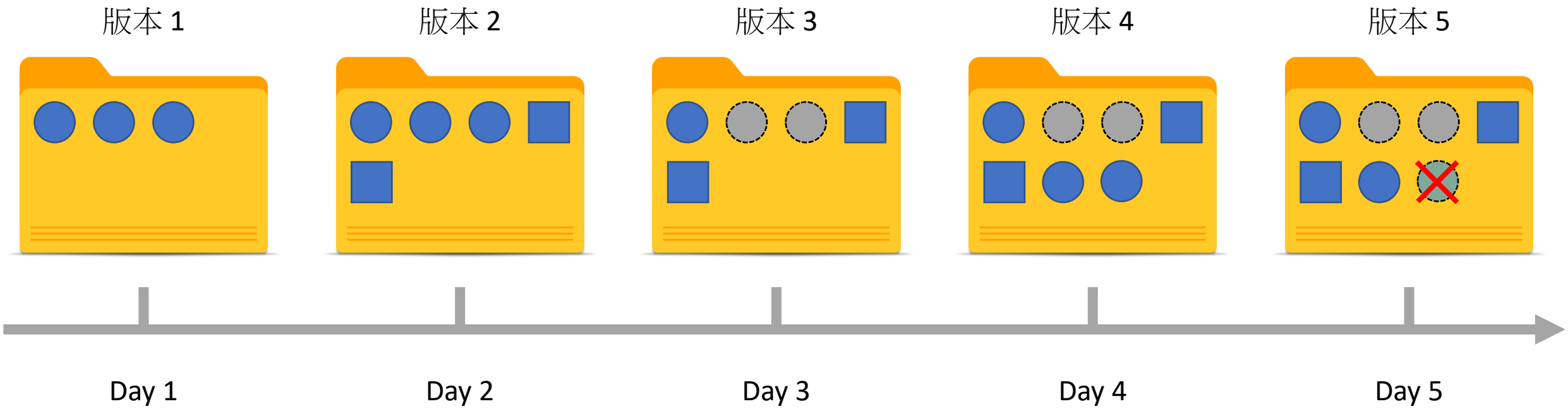
- **會發生什麼事？**
  - 很多版本檔案

- **會發生什麼事？**
  - 跟別人合作時.....



- **為什麼要版本控制**
  - 「凡走過必留下痕跡」
    - 追蹤歷程
    - 改了東西，不會改不回來
  - 「三個臭皮匠勝過一個諸葛亮」
    - 大家一起改，不會互相干擾
    - 大家一起改，還能清楚知道對方改了什麼

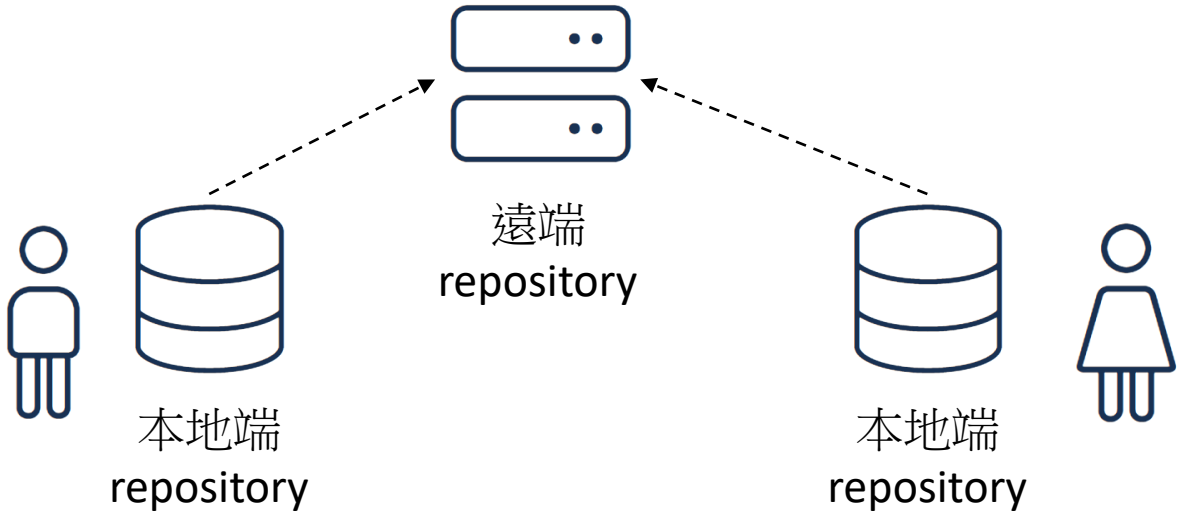
- 什麼是「版本」？



- **版本控制**
  - 什麼類型的內容適合版本控制
    - 程式原始碼
    - 測試程式碼
    - 伺服器的設定檔
    - 文件
    - 書籍
    - 網站內容

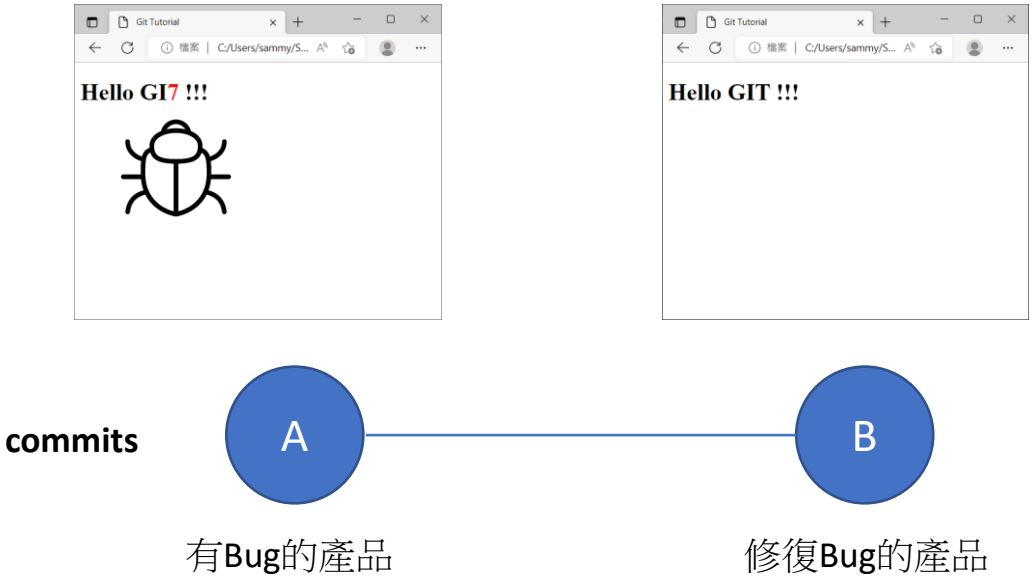
- **Git是什麼?**
  - 分散式的版本控制系統
  - 開源軟體(open-source software)
    - 活耀的開源社群
    - 豐富的生態系支持
  - 適合於多種不同類型的專案與工作流程
    - Linux作業系統開發即運用Git進行管理
  - 是大部分 IT公司都會運用的工具

- **分散式版本控制系統**
  - 分散式版本控制系統具有三特性
    - 每一位使用者皆有自己本地端的專案歷程(repository)
    - 使用者可以離線使用版控系統
    - 可以方便地進行儲存庫(Repository)內容的整合工作

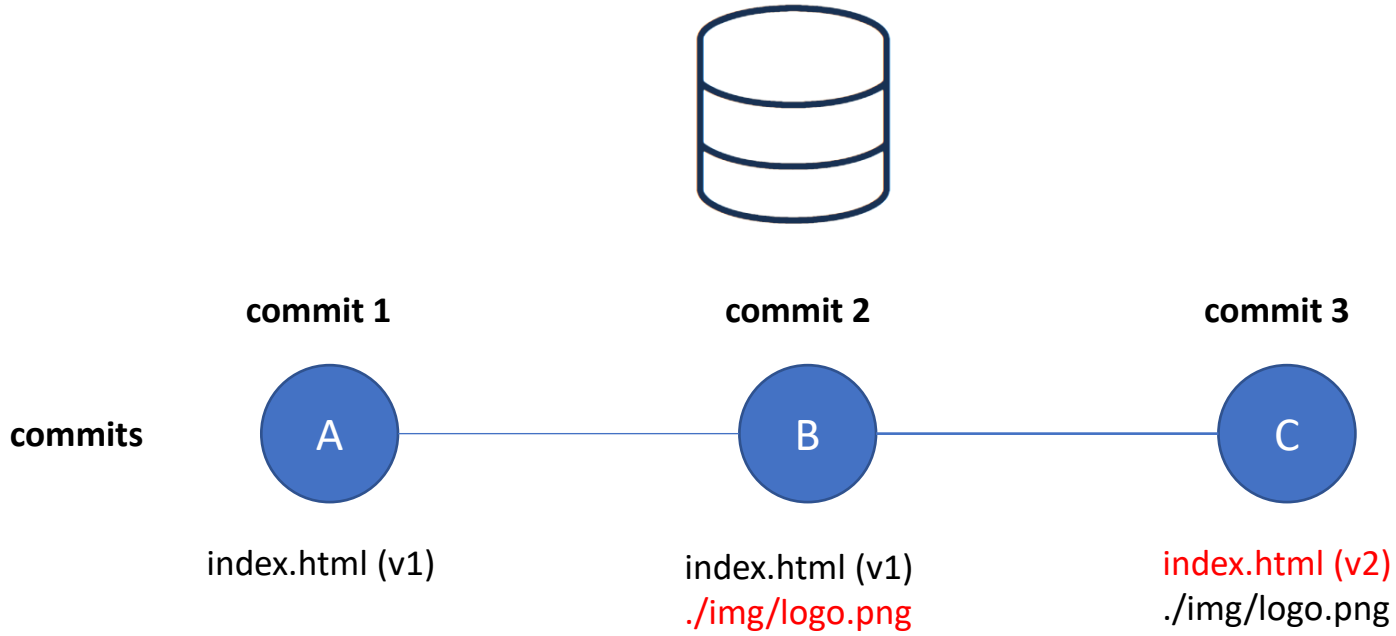


Git Commits

- Git記錄專案在不同時間的狀態，構成一個專案開發的**歷程**
- 專案中每個記錄版本稱為一個commit (交付或提交)
- 每一次commit可視為專案在一個給定的時間點上的快照(snapshot)

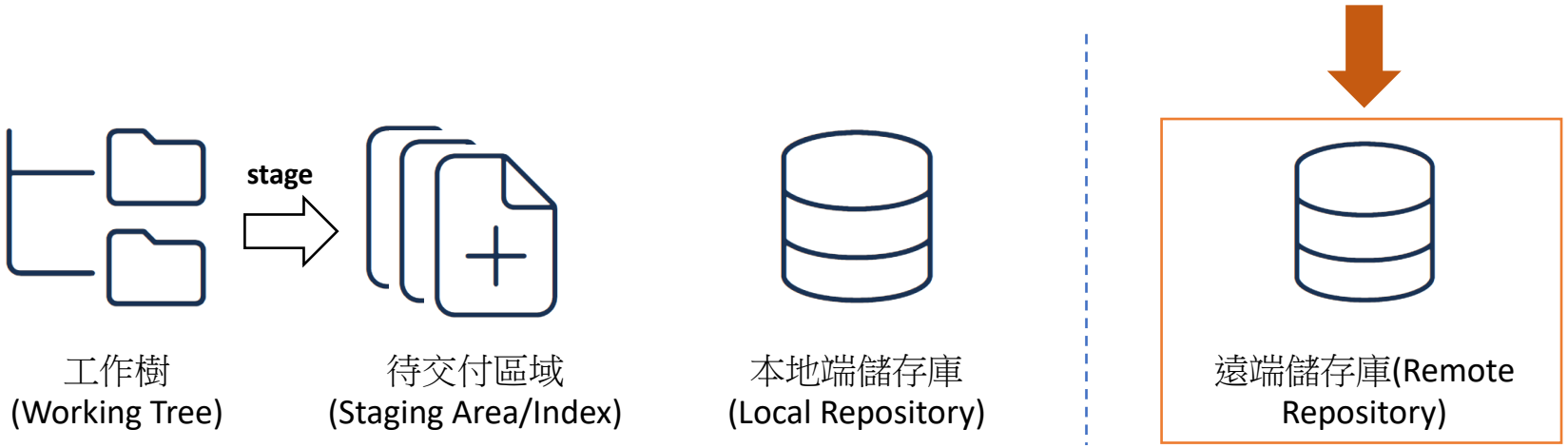


- **Git Repository (Repo) 是什麼?**
  - 一連串的專案快照(snapshot)，包含多個交付(commit)



遠端儲存庫(Remote Repository)

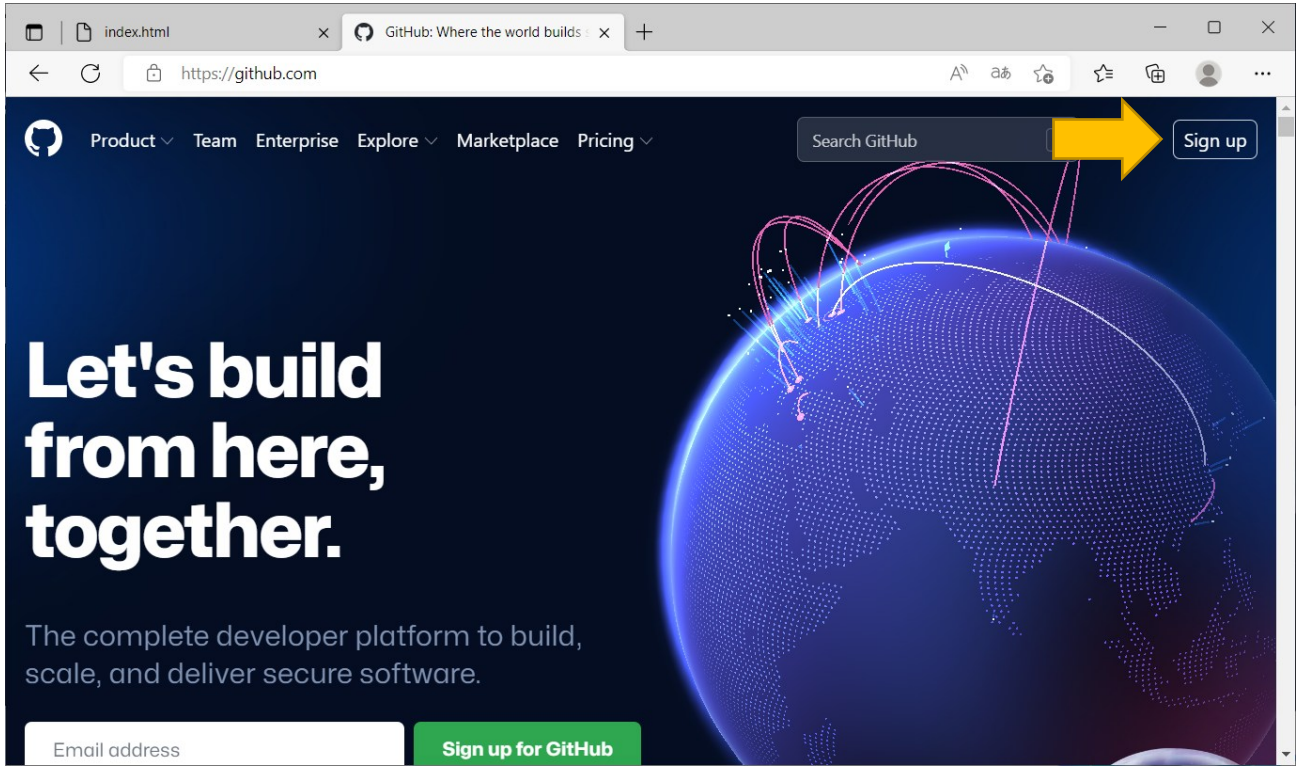
- 雲端的儲存庫
- 反映軟體專案的官方程式碼狀態
- 可與其他系統整合
  - 專案管理系統、議題追蹤系統等



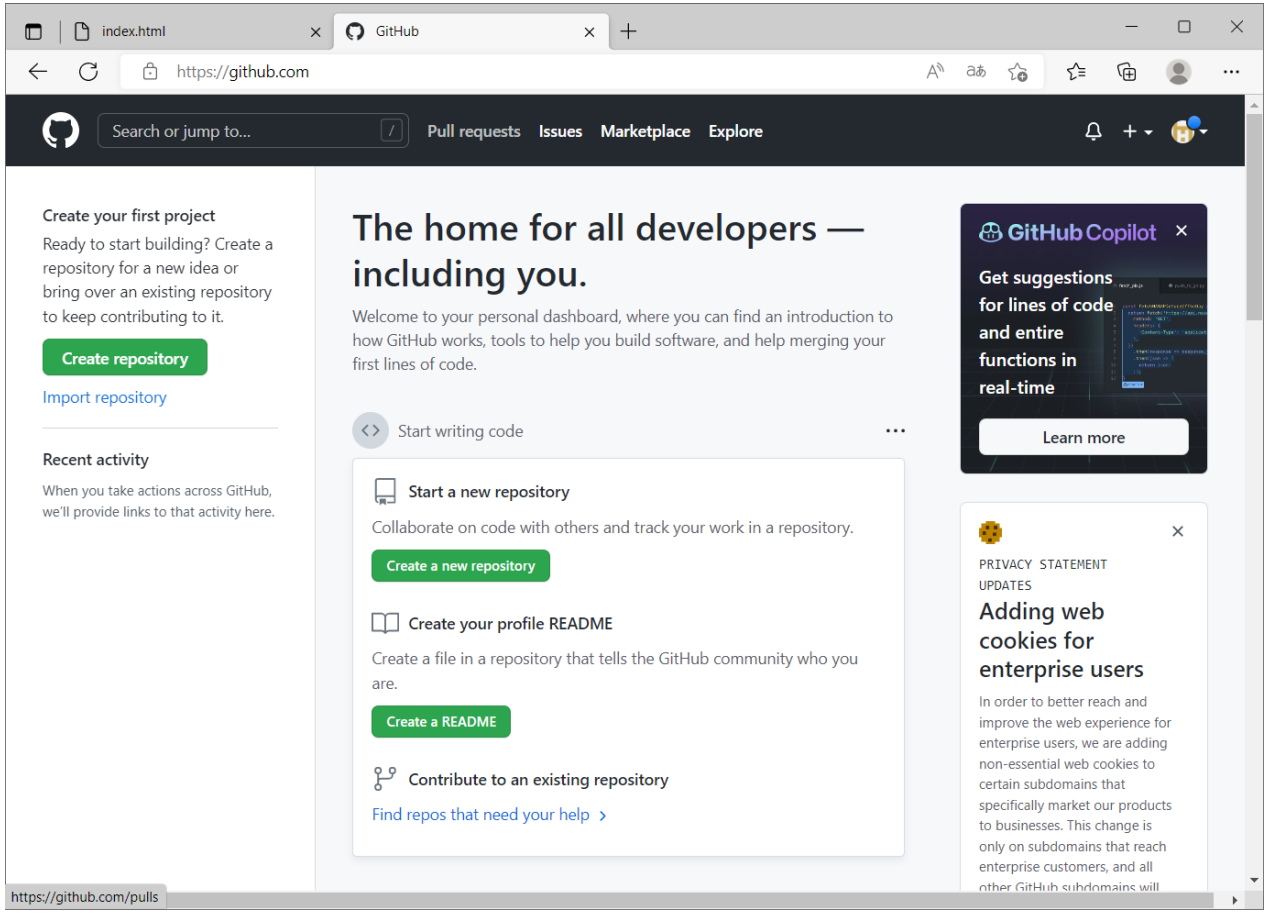
- **遠端儲存庫(Remote Repository)**
  - 遠端儲存庫代管服務
    - GitHub
      - https://github.com/
    - GitLab
      - https://about.gitlab.com/
    - Bitbucket
      - https://bitbucket.org/
    - ….

GitHub遠端儲存庫

- 申請GitHub帳號
  - https://github.com/



- # GitHub遠端儲存庫
  - 註冊成功

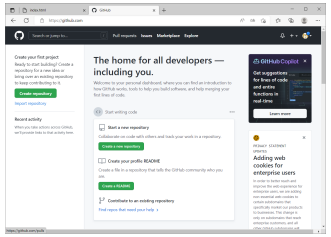


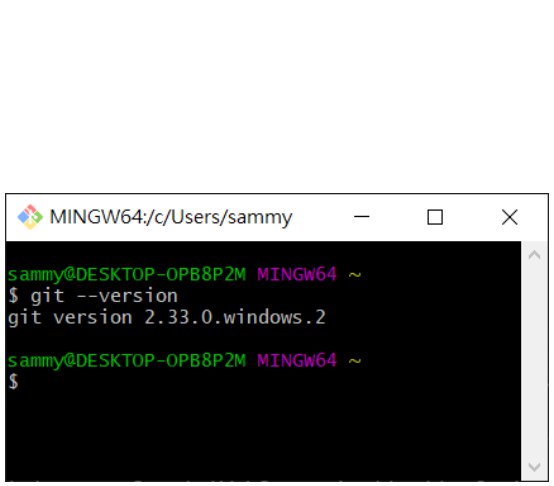
- # GitHub遠端儲存庫
  - GitHub遠端儲存庫的**名稱**都以.git做為後綴

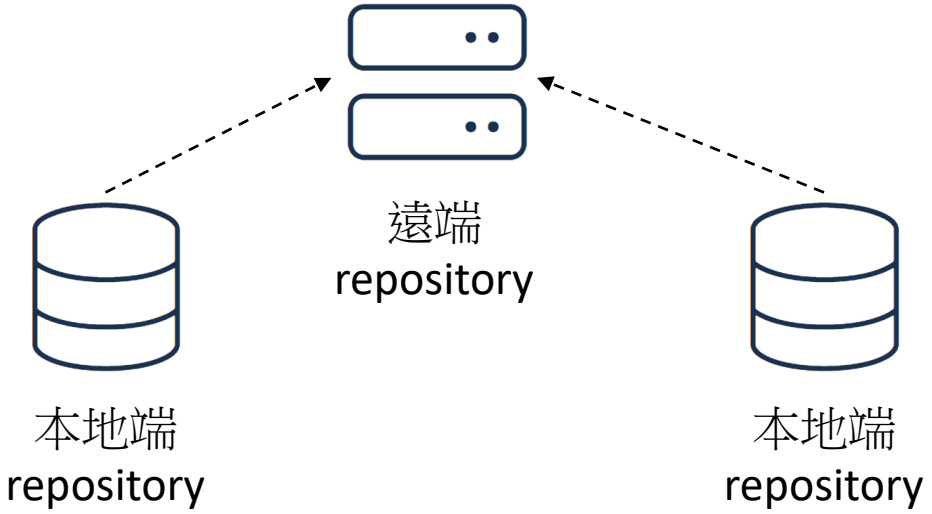
遠端儲存庫(Remote Repository)

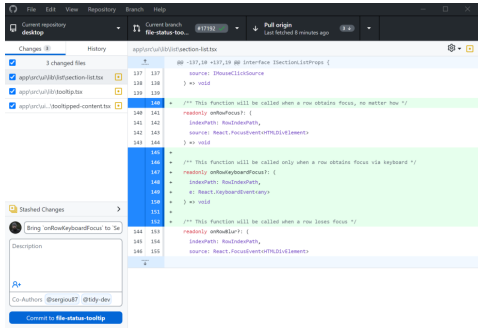
https://github.com/<username>/<repo-name>.git

- # Git工具
  - 指令(command line)模式與使用者介面(GUI)模式

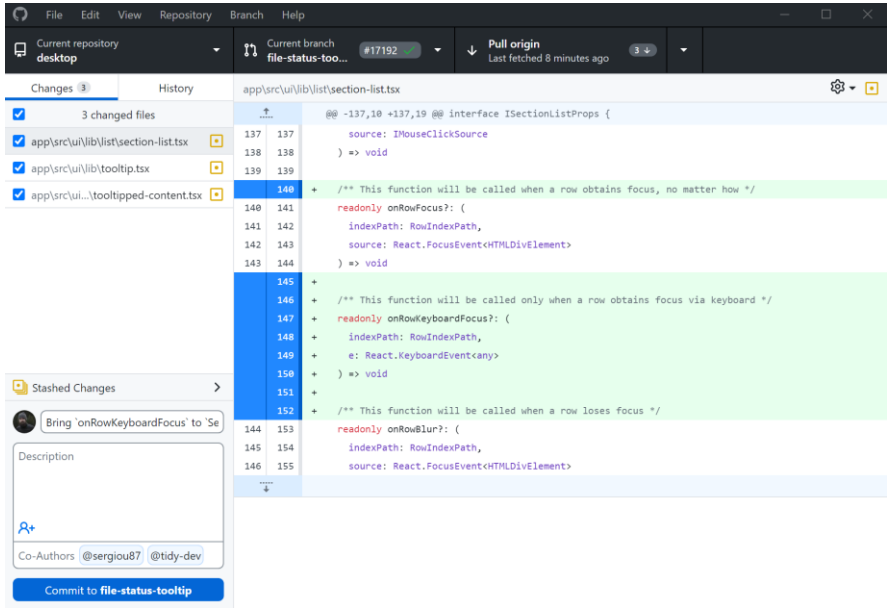






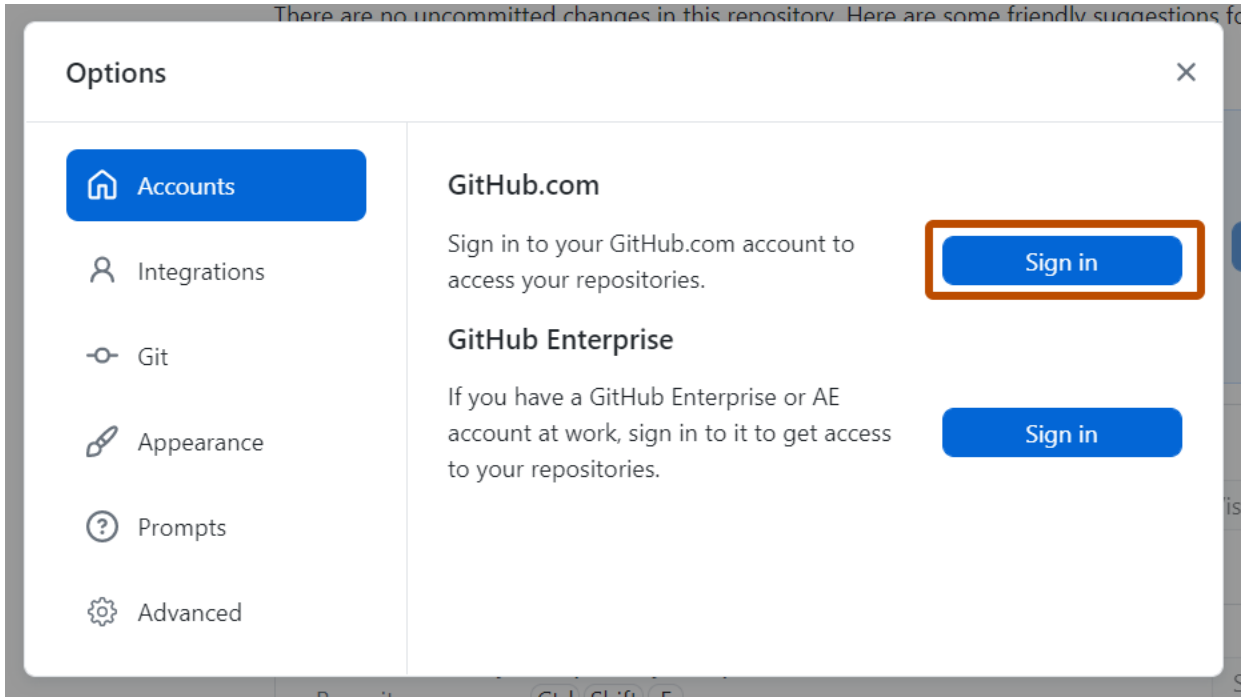


- # 安裝圖形化Git客戶端工具
  - GitHub Desktop
    - https://desktop.github.com/
    - 原生支援與GitHub之串連
  - SourceTree
    - https://www.sourcetreeapp.com/
  - 其他圖形化Git客戶端工具
    - https://git-scm.com/downloads/guis/



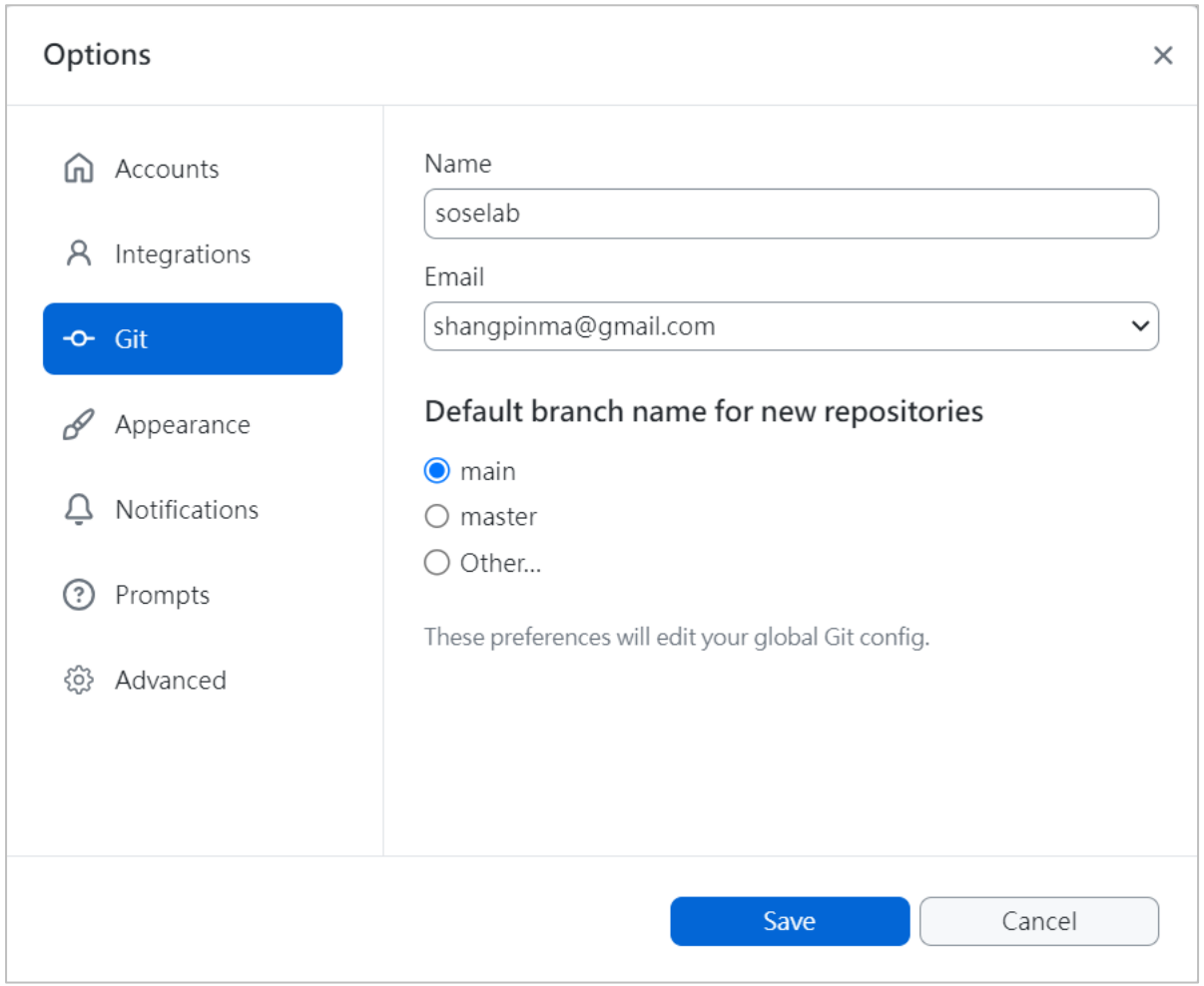
GitHub Desktop使用者帳號設定

- 安裝 GitHub Desktop 後，可以使用 GitHub 帳號對應用程式進行驗證。
  - 透過身份驗證，你可以連接到 GitHub 上的遠端儲存庫。
- 操作方式：[File]->[Options]->[Accounts]，並透過GitHub帳號登入。
  - 如果還沒有GitHub帳號，請立即申請：https://github.com/。

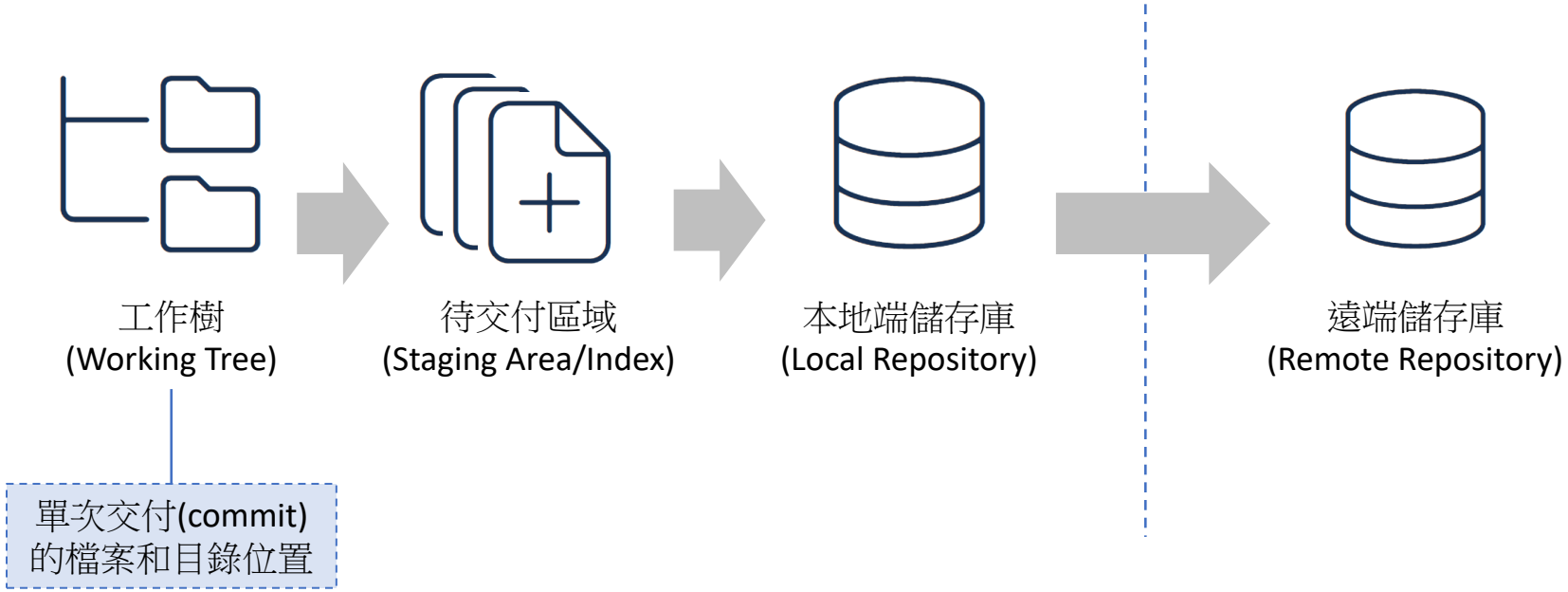


GitHub Desktop使用者帳號設定

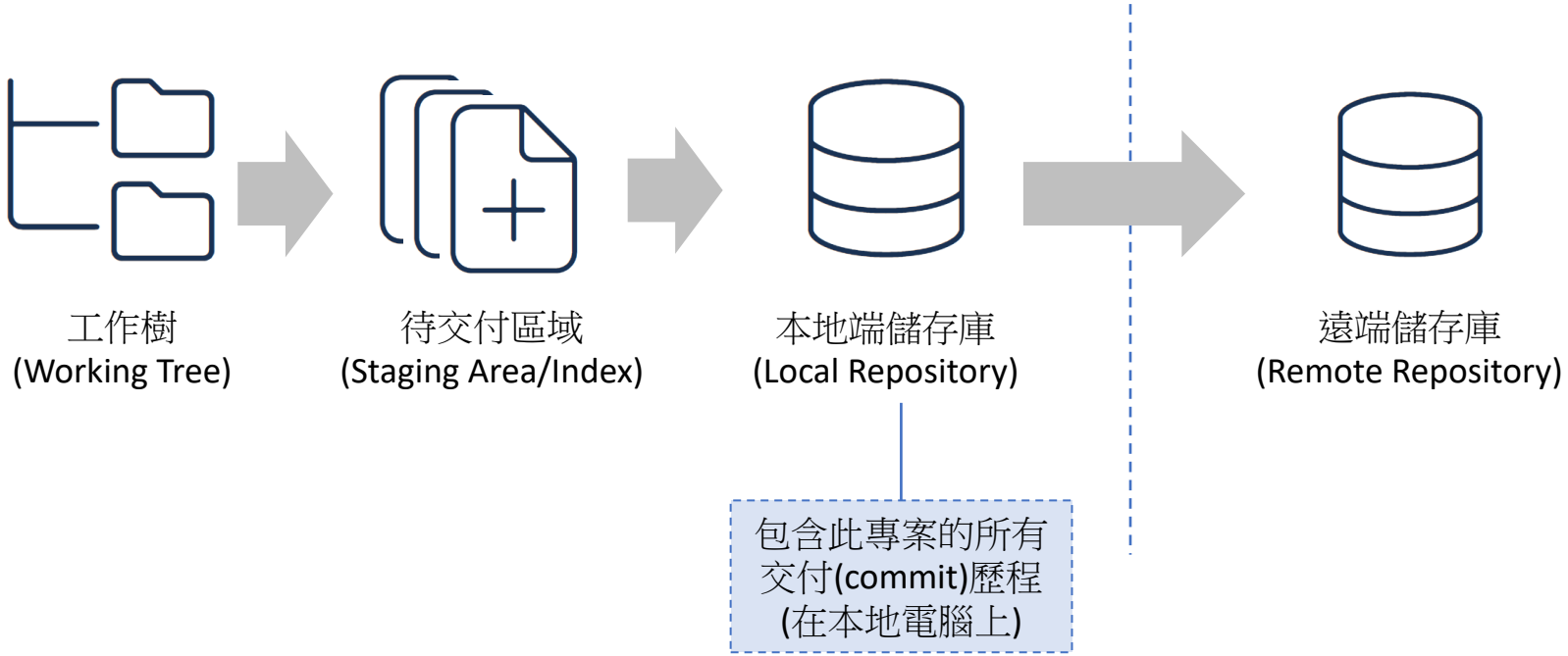
- 在"Git"頁籤的設定部分，請設定與GitHub帳號相同的Email，以確保之後的commit之作者資訊正確。
- 此外，預設分支名稱請選擇"main"，讓Git與GitHub都具有相同的預設分支名稱。
  - 同學們可於此門課程後再去了解分支的運用。



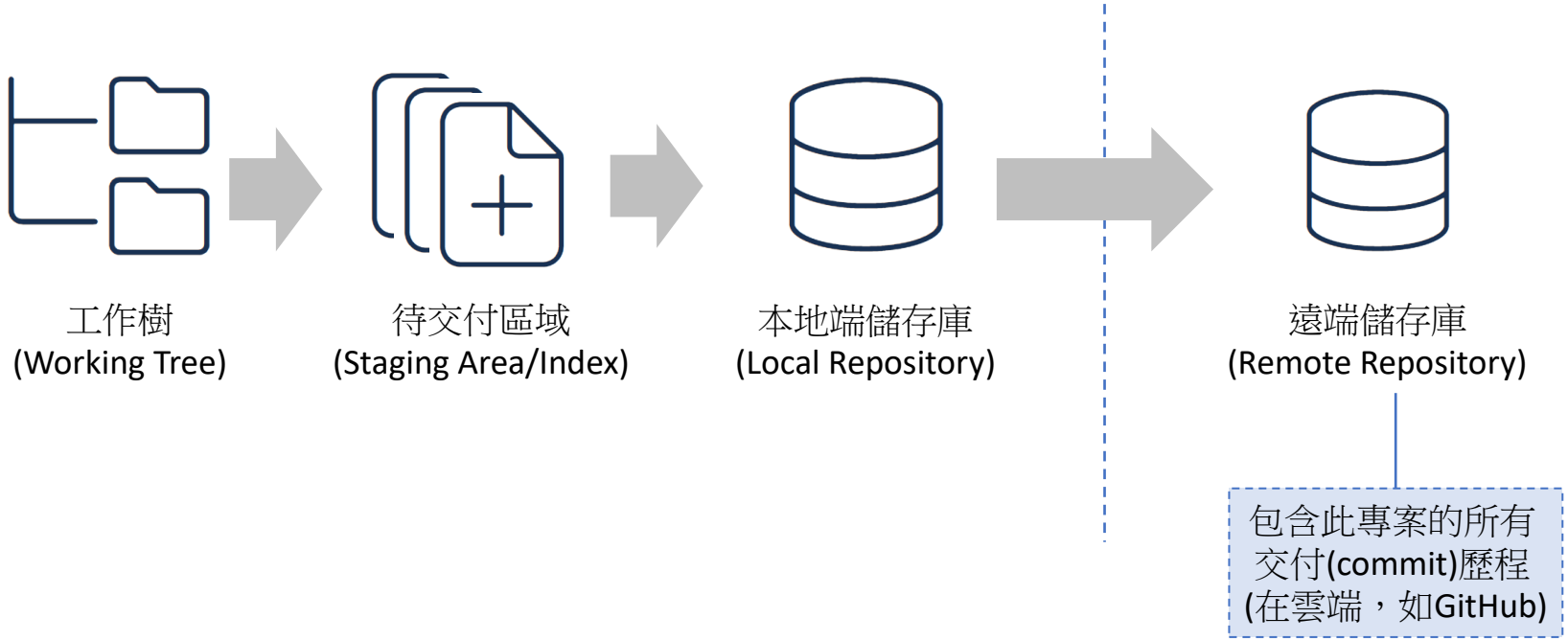
- # Git Locations



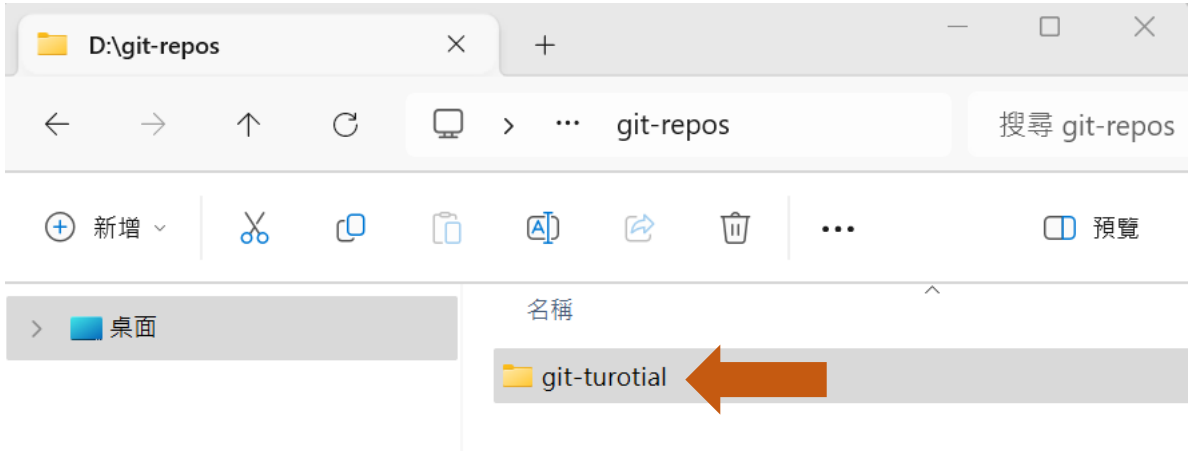
- # Git Locations



- **Git Locations**

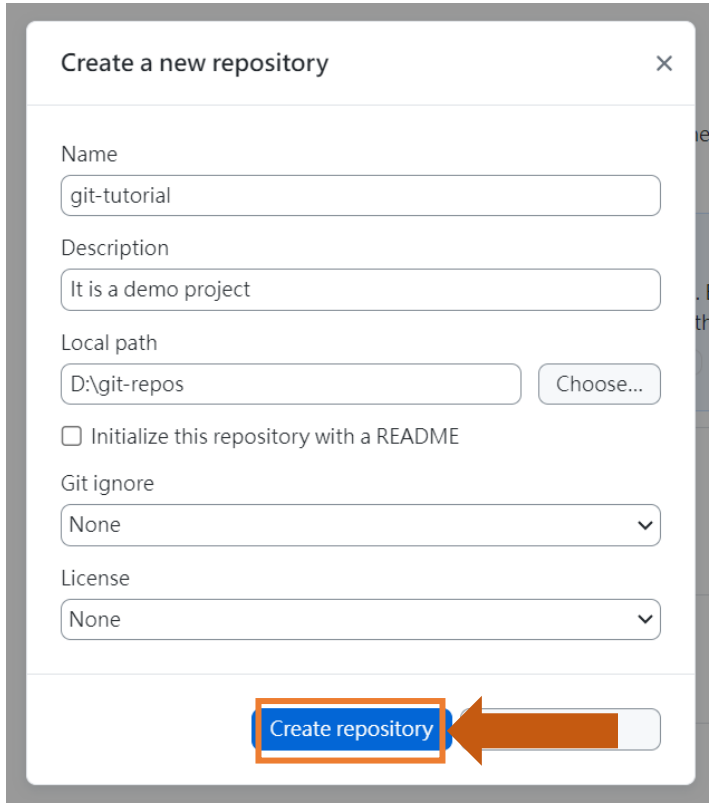


- **建立本地端儲存庫(Local Repository)**
  - 開啟檔案總管，建立版控專案**根**目錄
    - 用以管理所有的Git專案
    - e.g., 目錄名稱 *git-repos*
  - 接著建立版控專案目錄(Working Tree)
    - e.g., 目錄名稱 *git-tutorial*

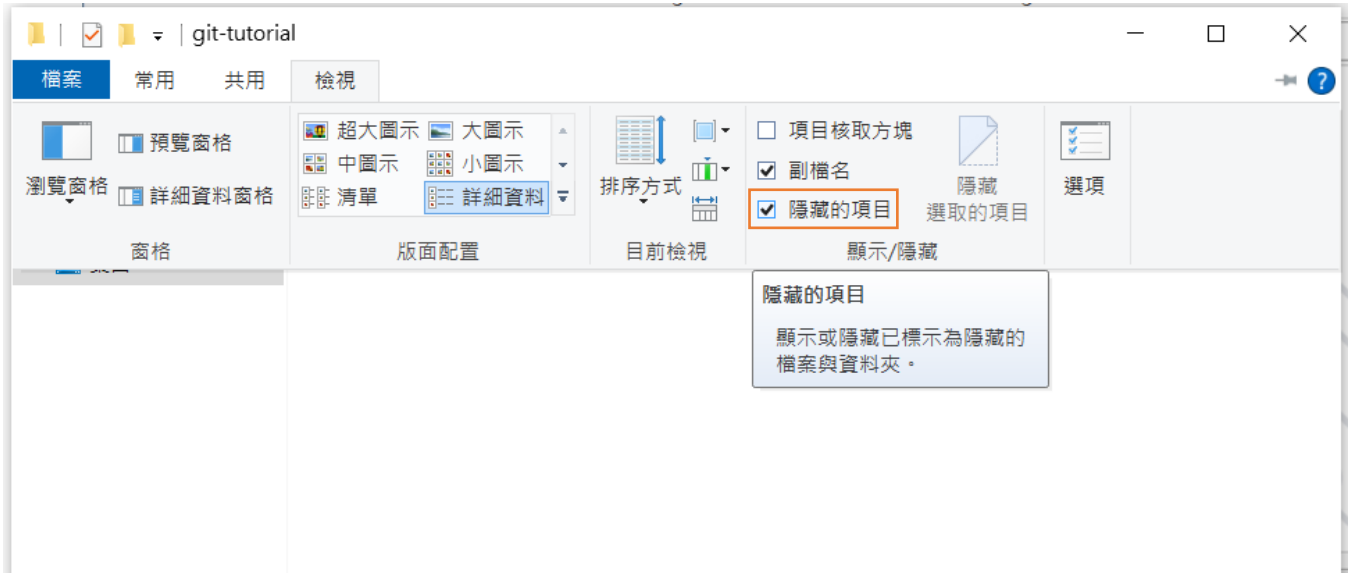


建立本地端儲存庫(Local Repository)

- 透過GitHub Desktop設定版控工作目錄(working tree)
  - [File]->[New Repository]

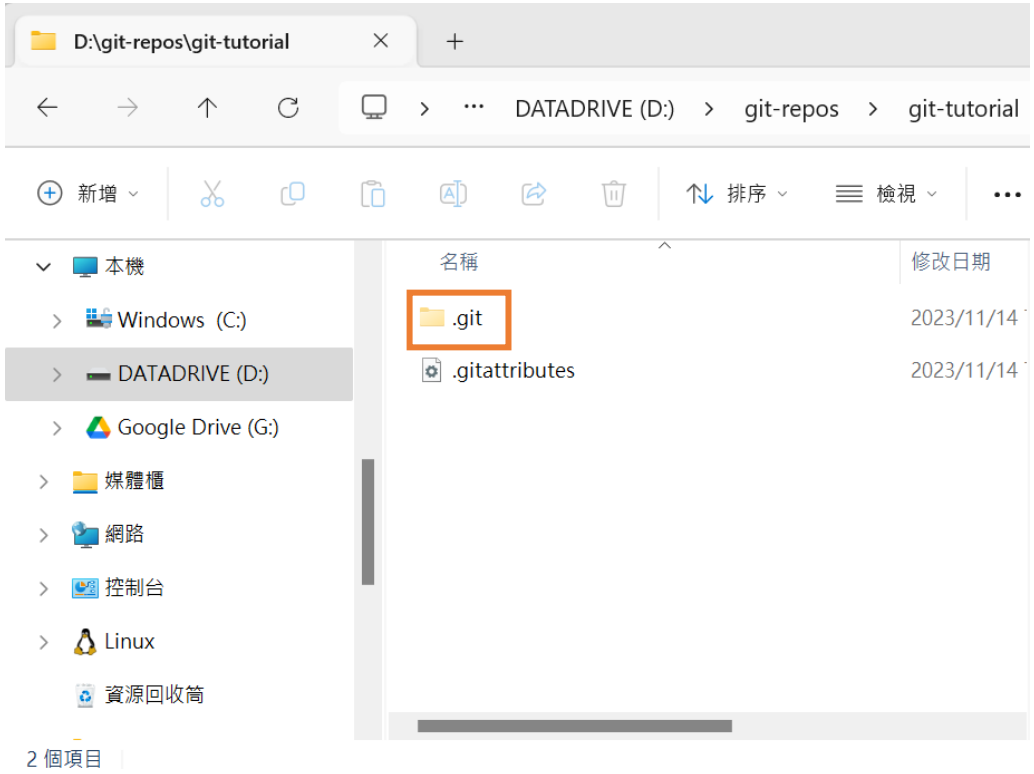


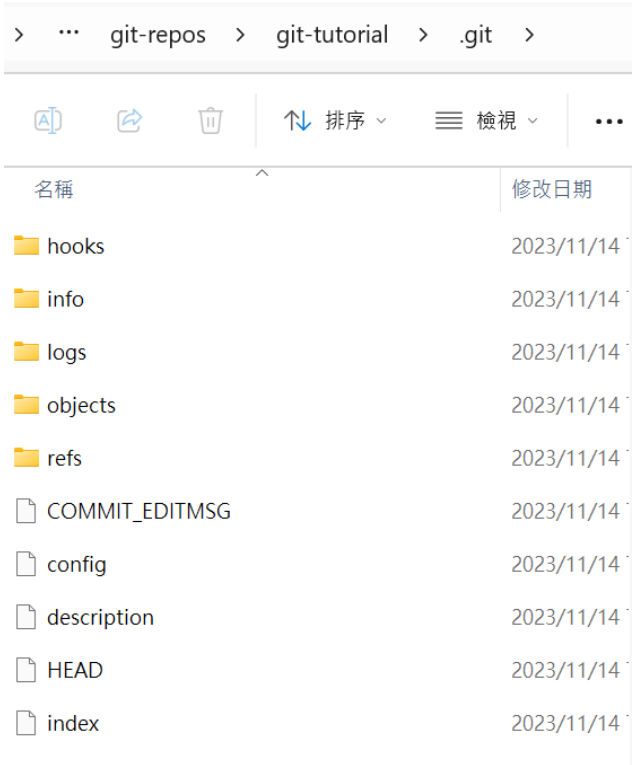
- # 建立本地端儲存庫(Local Repository)
  - 開啟檔案總管，開啟顯示隱藏項目



建立本地端儲存庫(Local Repository)

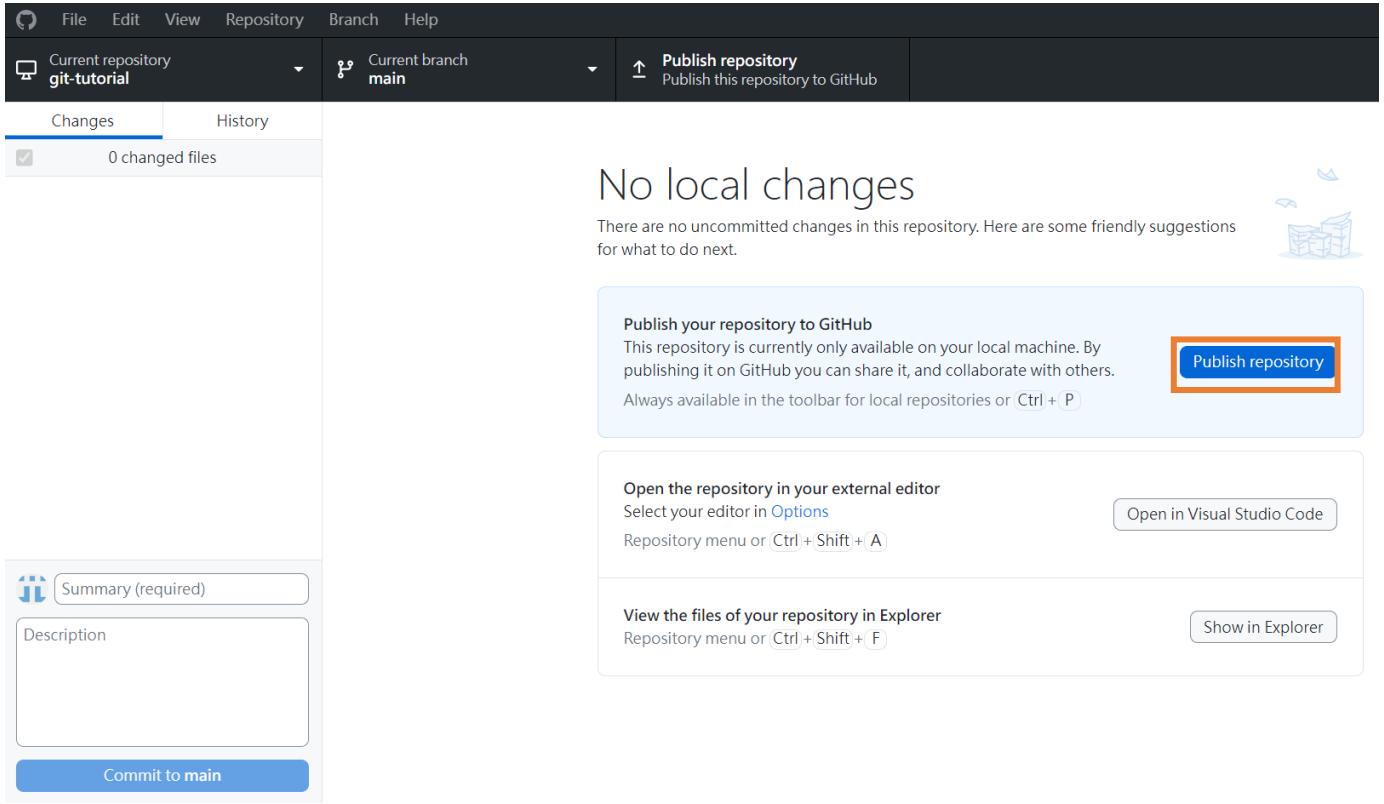
- 開啟檔案總管，查看本地端存庫(.git)



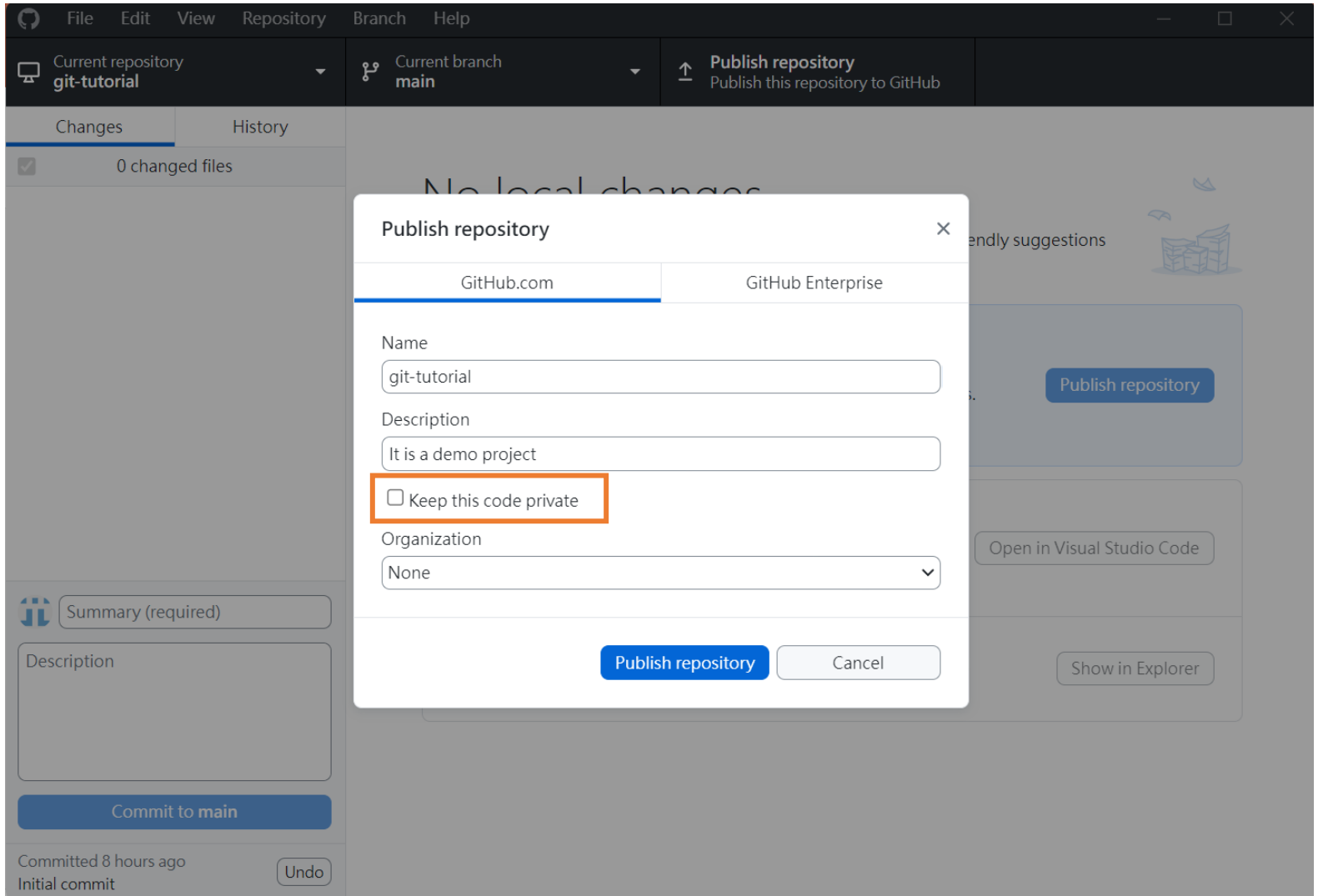


建立與連結遠端儲存庫(Remote Repository)

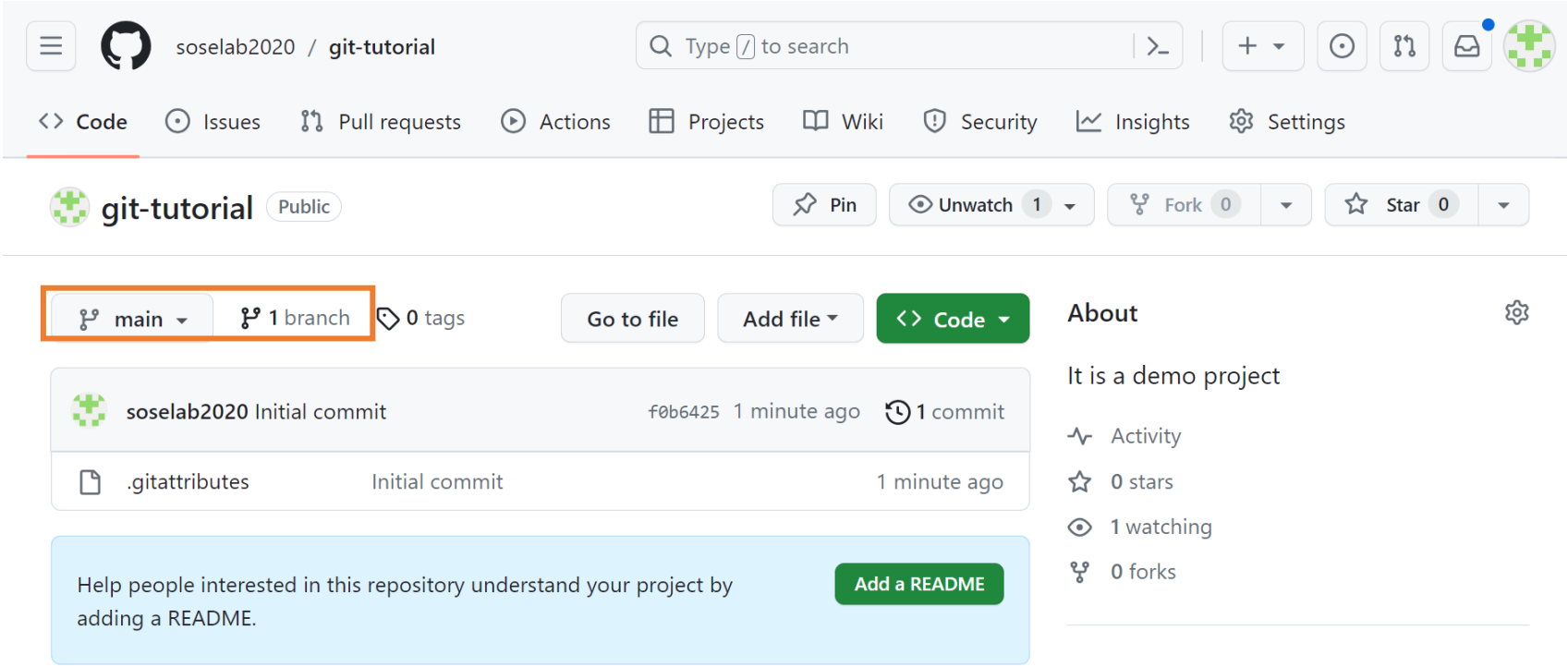
- 我們可直接將本地端儲存庫與GitHub遠端儲存庫串連起來
  - 點擊"Publish Repository"功能即可完成
  - 等同將專案發布於Internet，日後可方便與他人協同合作



- 建立與連結遠端儲存庫(Remote Repository)



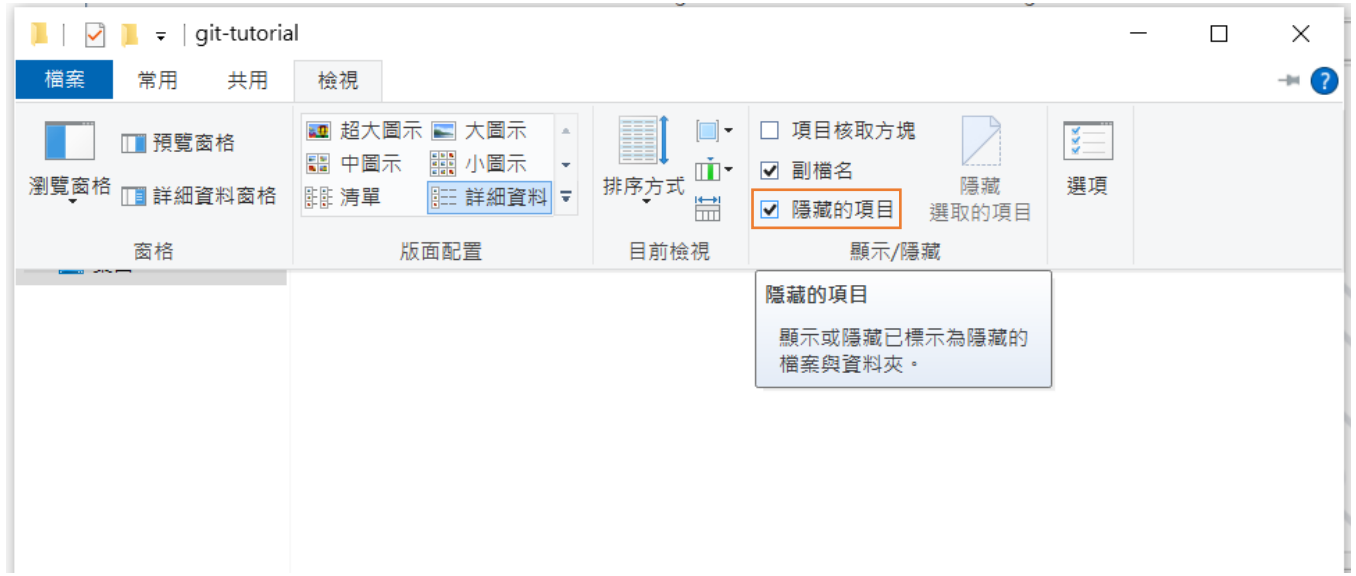
- # 建立與連結遠端儲存庫(Remote Repository)
  - 連結GitHub，將會看到遠端儲存庫也成功地被建立出來。
    - 一個分支(branch)可視為專案的獨立開發支線
    - 以目前來說，可先都以main分支為主，日後請大家務必要再了解分支的處理方式



- Lab 1
  - 請完成GitHub Desktop之安裝與設定。
    - 請完成與GitHub帳號的串連。
    - 請完成Git所需的使用者帳號與email設定。
  - 請透過GitHub Desktop建立一個空的Local Repository (本地儲存庫)，並透過"Publish Repository"功能建立對應的Remote Repository (遠端儲存庫)。
    - 此本地儲存庫我們後續將其稱之為LR1。
    - 遠端儲存庫我們後續將其稱之為RR。

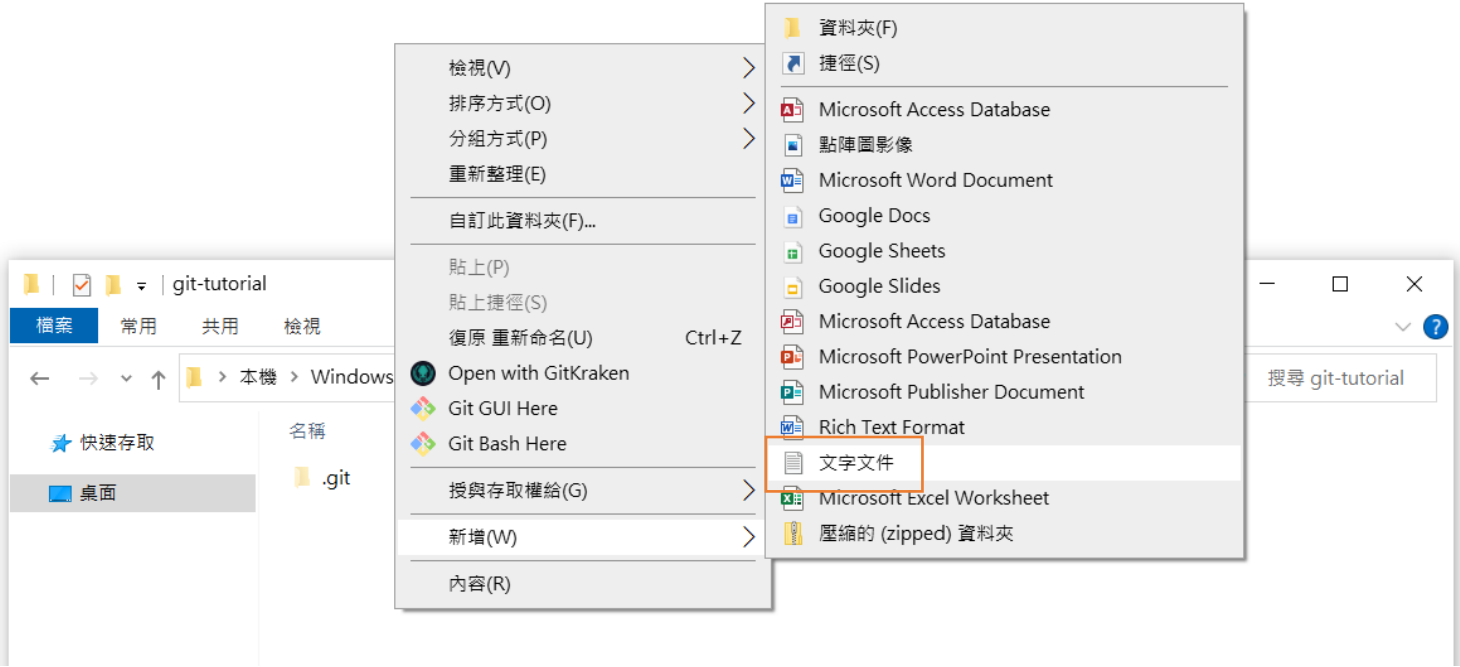
交付檔案至本地端儲存庫

- 首先開啟檔案總管，開啟顯示附檔名。



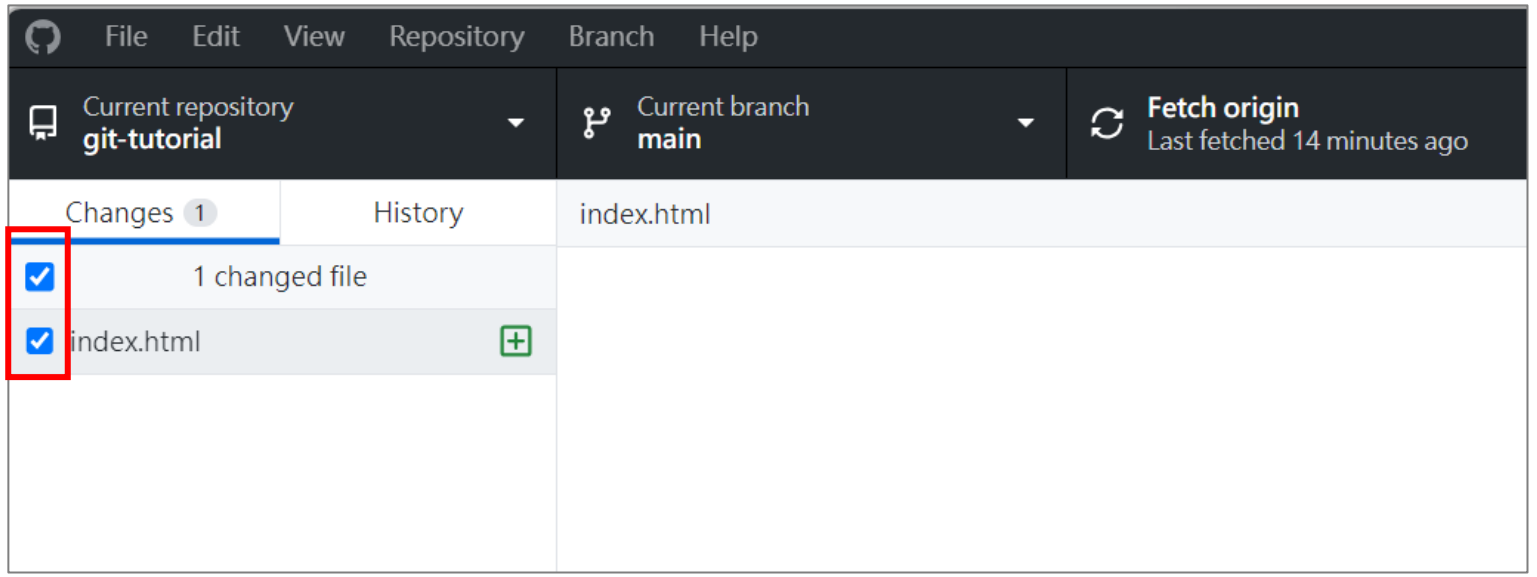
交付檔案至本地端儲存庫

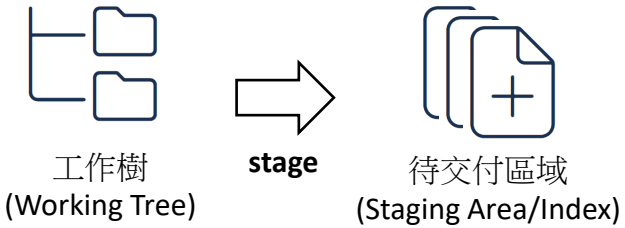
- 加入**未追蹤檔案**至工作目錄
  - e.g., *index.html*



• 檢視檔案狀態

GitHub Desktop會預設勾選所有新加入檔案，以讓我們將未追蹤檔案加入待交付區域





- # 交付檔案至本地端儲存庫

工作樹
(Working Tree)

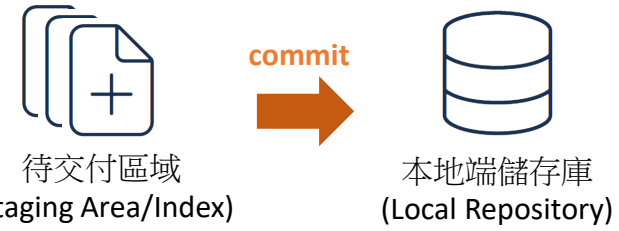
待交付區域
(Staging Area/Index)

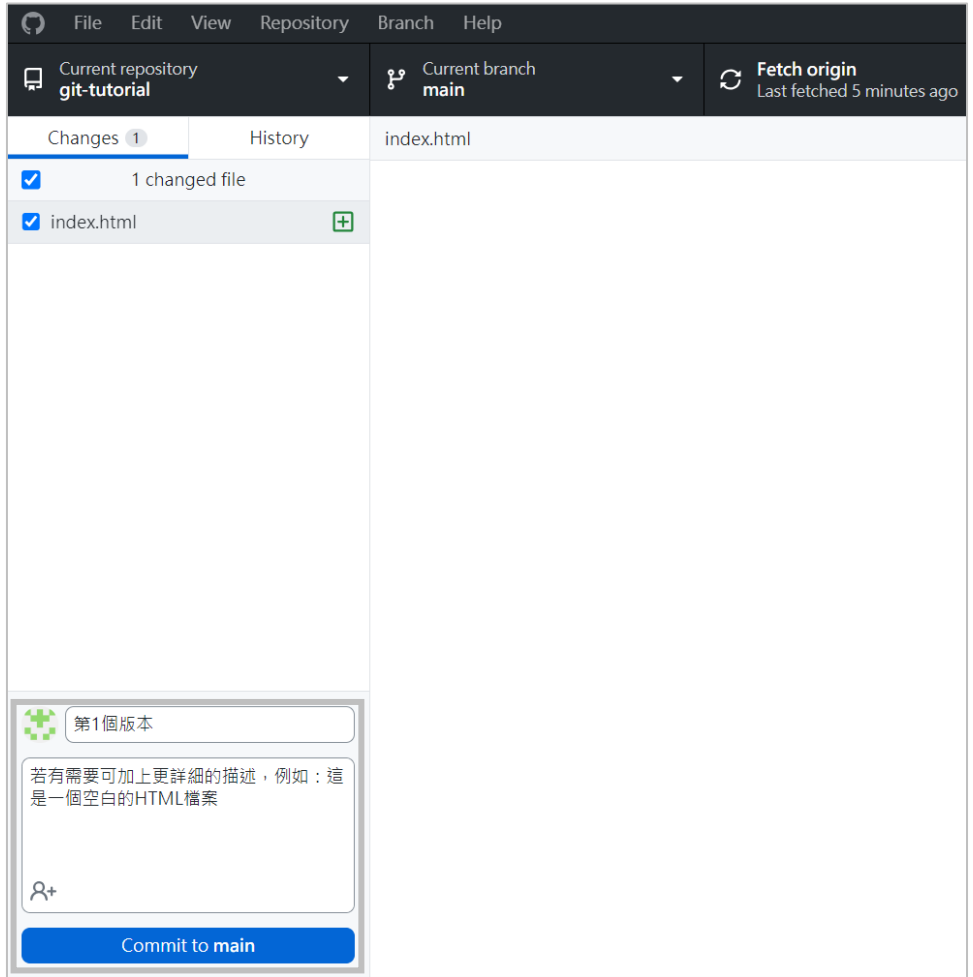
本地端儲存庫
(Local Repository)

遠端儲存庫(Remote Repository)

交付檔案至本地端儲存庫

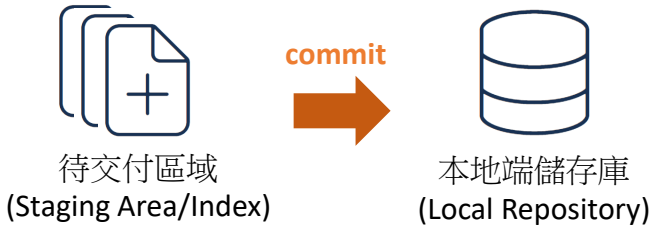
- 我們要撰寫交付訊息(commit message)，實際建立commit。

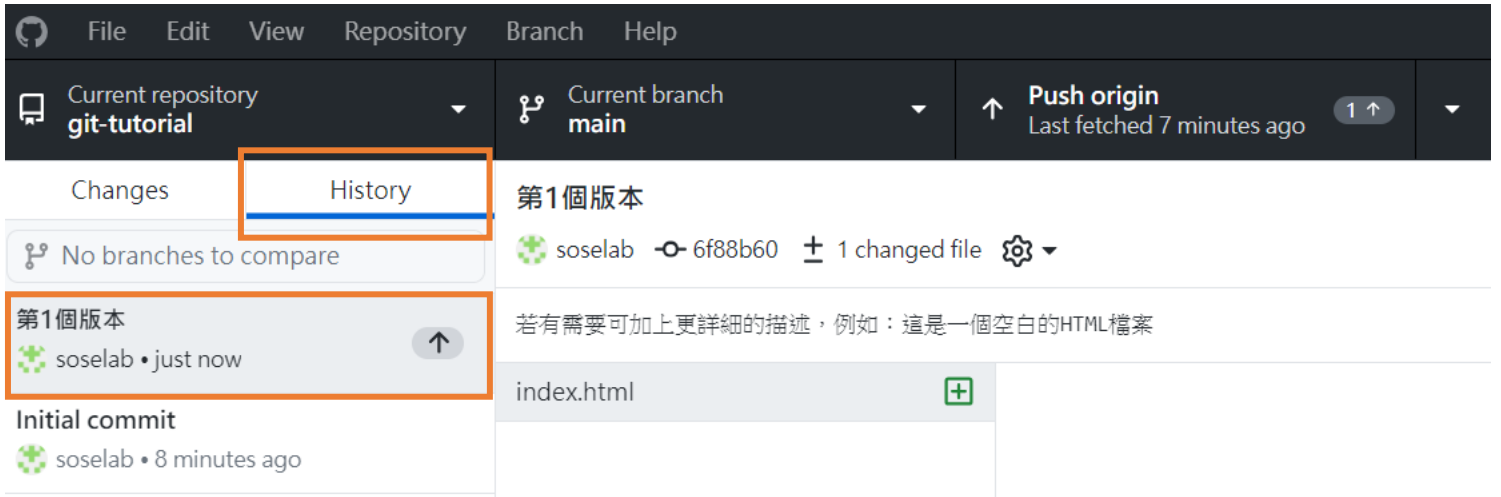




交付檔案至本地端儲存庫

- 檢視交付歷程(history，也就是log)

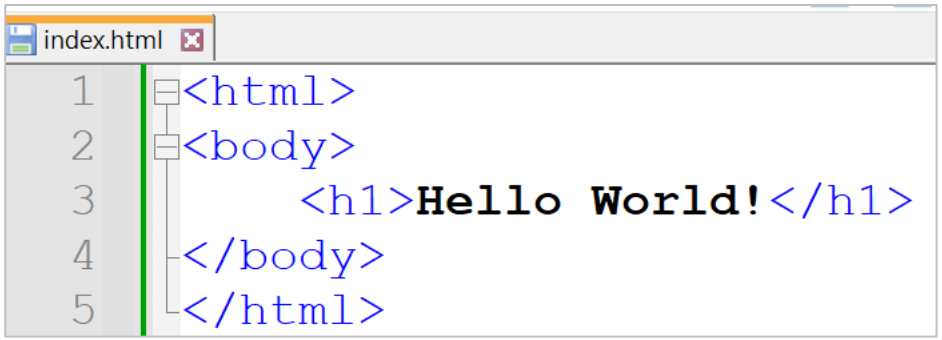




小量改進 - 修改內容

- 修改/變更工作目錄(working tree)中的*index.html*

```
<h1>Hello World!</h1>
```

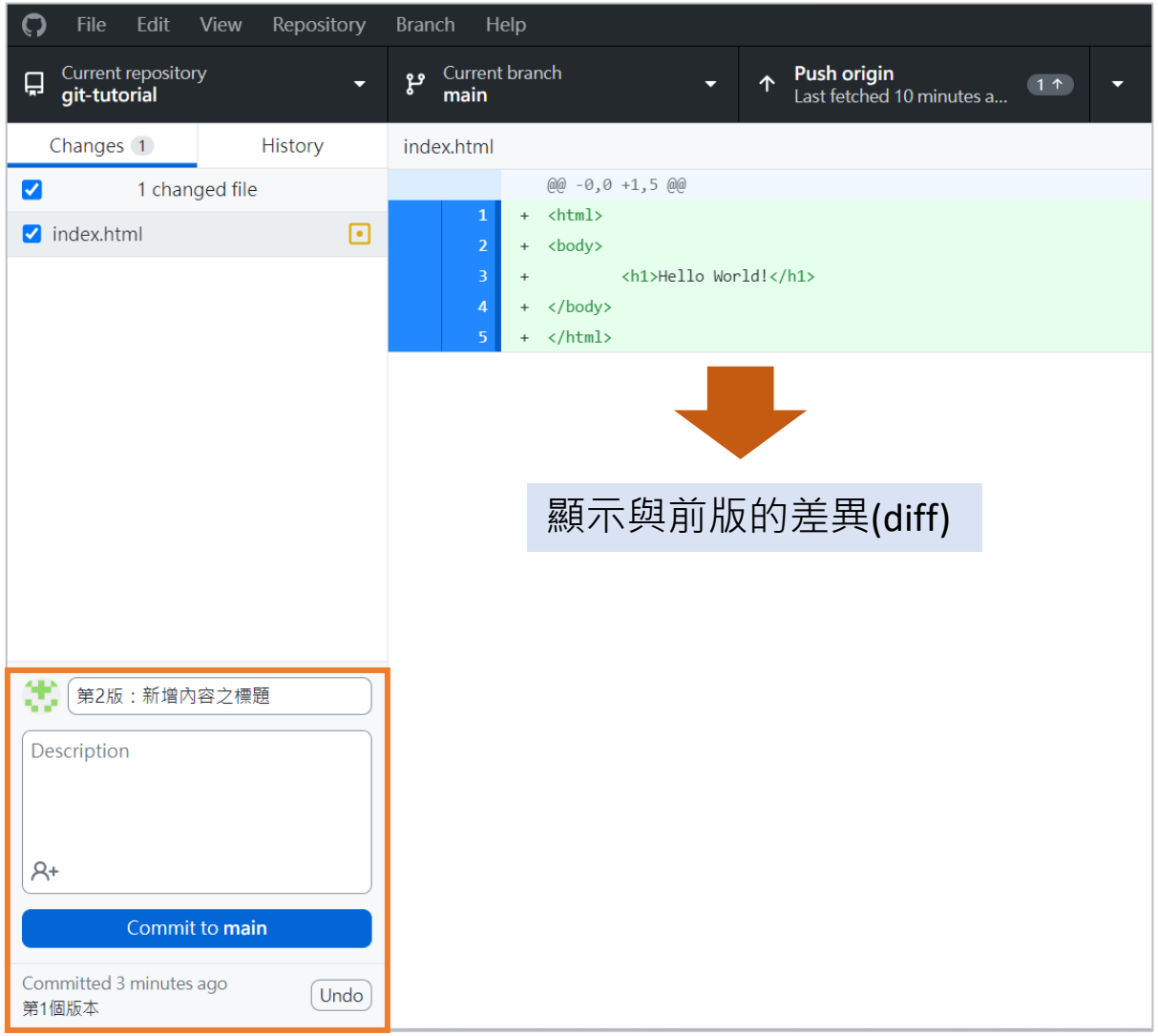


(Notepad++)

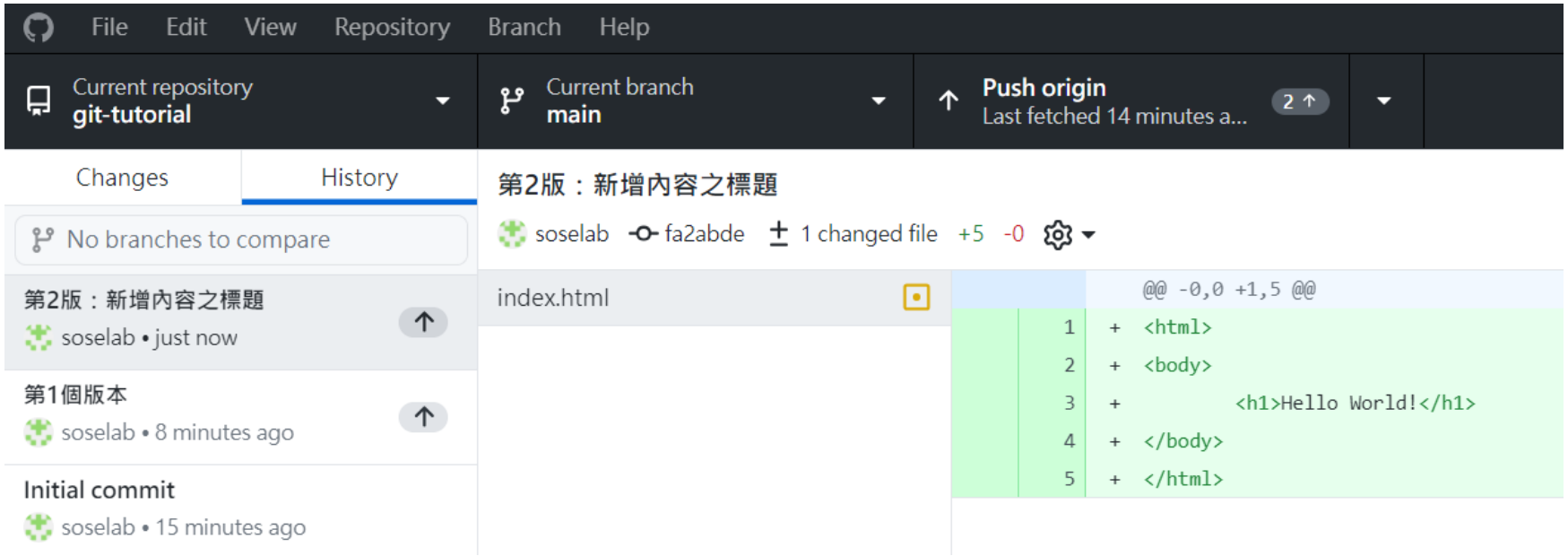
(Chrome)

小量改進 - 修改內容

- 可查看差異，並交付新版

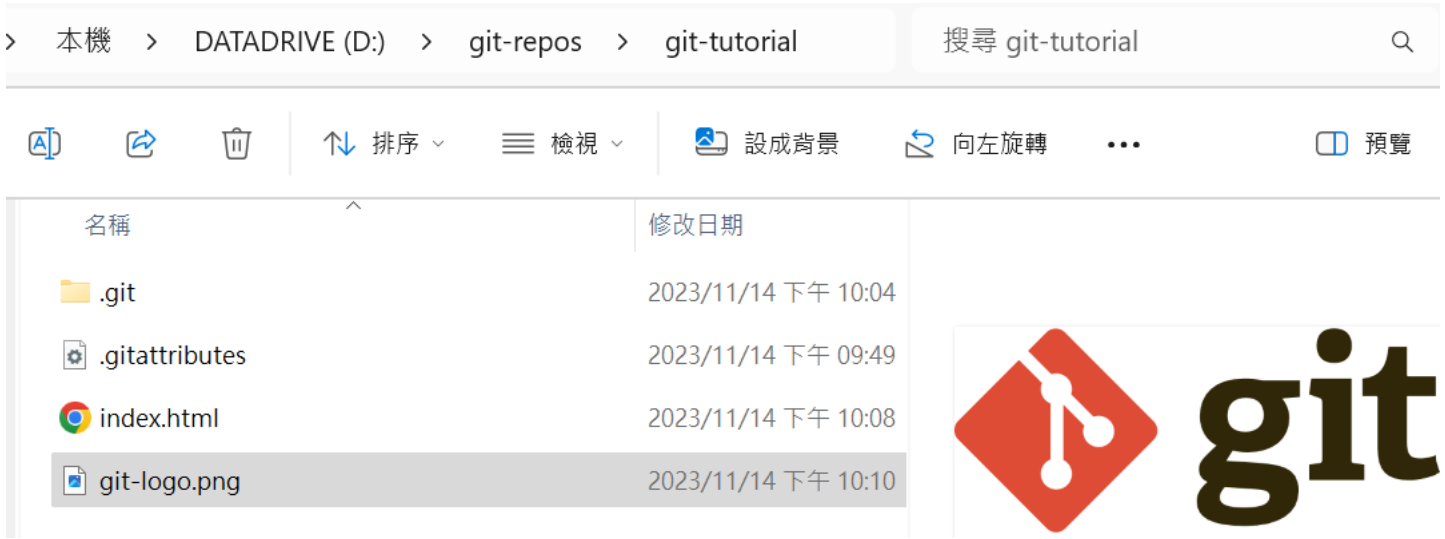


- # 小量改進 - 修改內容
  - 檢視交付歷程



小量改進 - 新增檔案

- 加入**未追蹤圖片**至工作目錄
  - e.g., *git-logo.png*



小量改進 - 新增檔案

- 修改/變更工作目錄中的*index.html*

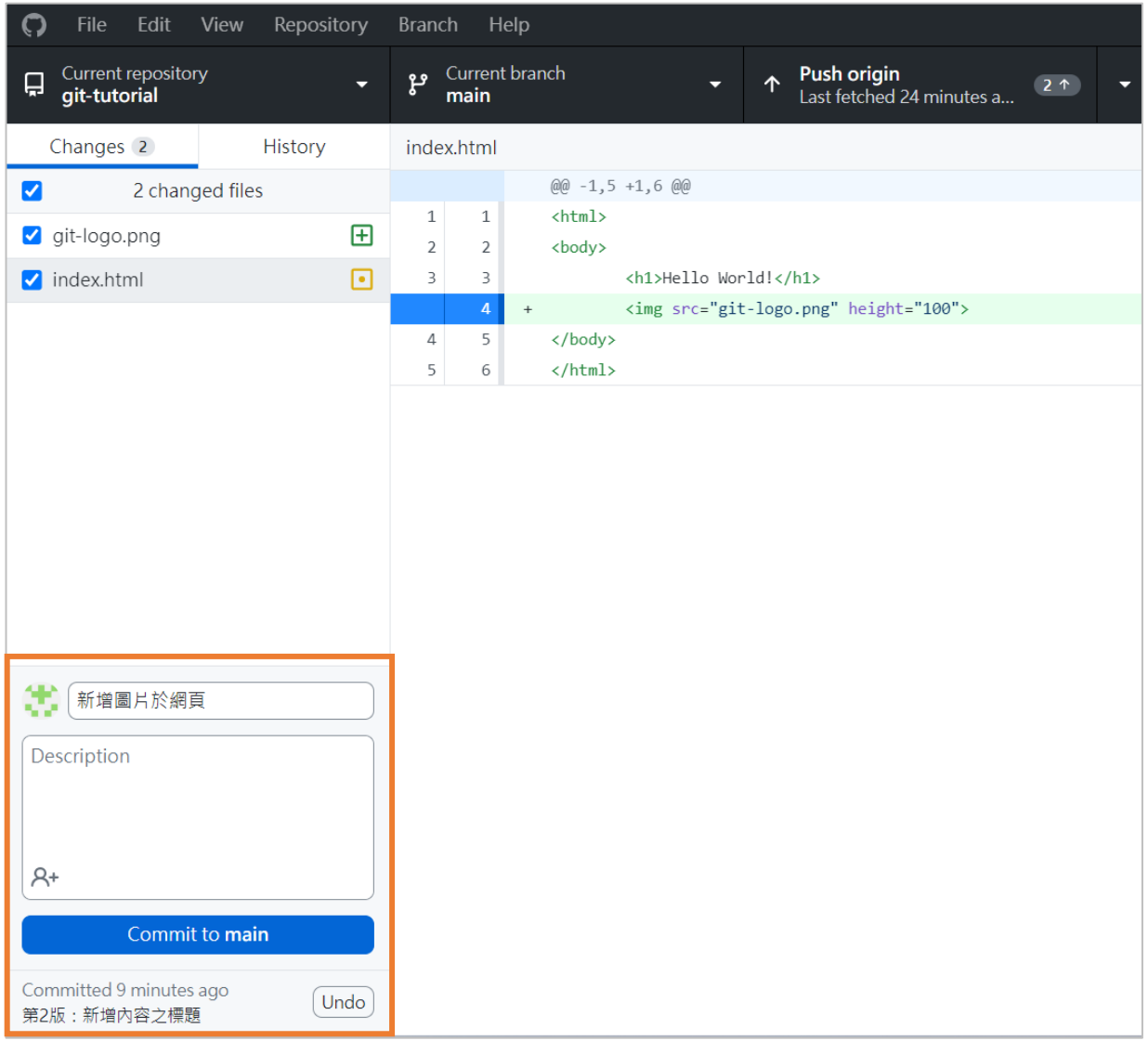
```
<h1>Hello World!</h1>
<img src="git-logo.png" height="100">
```

index.html

```
1  <html>
2  <body>
3      <h1>Hello World!</h1>
4      <img src="git-logo.png" height="100">
5  </body>
6  </html>
```

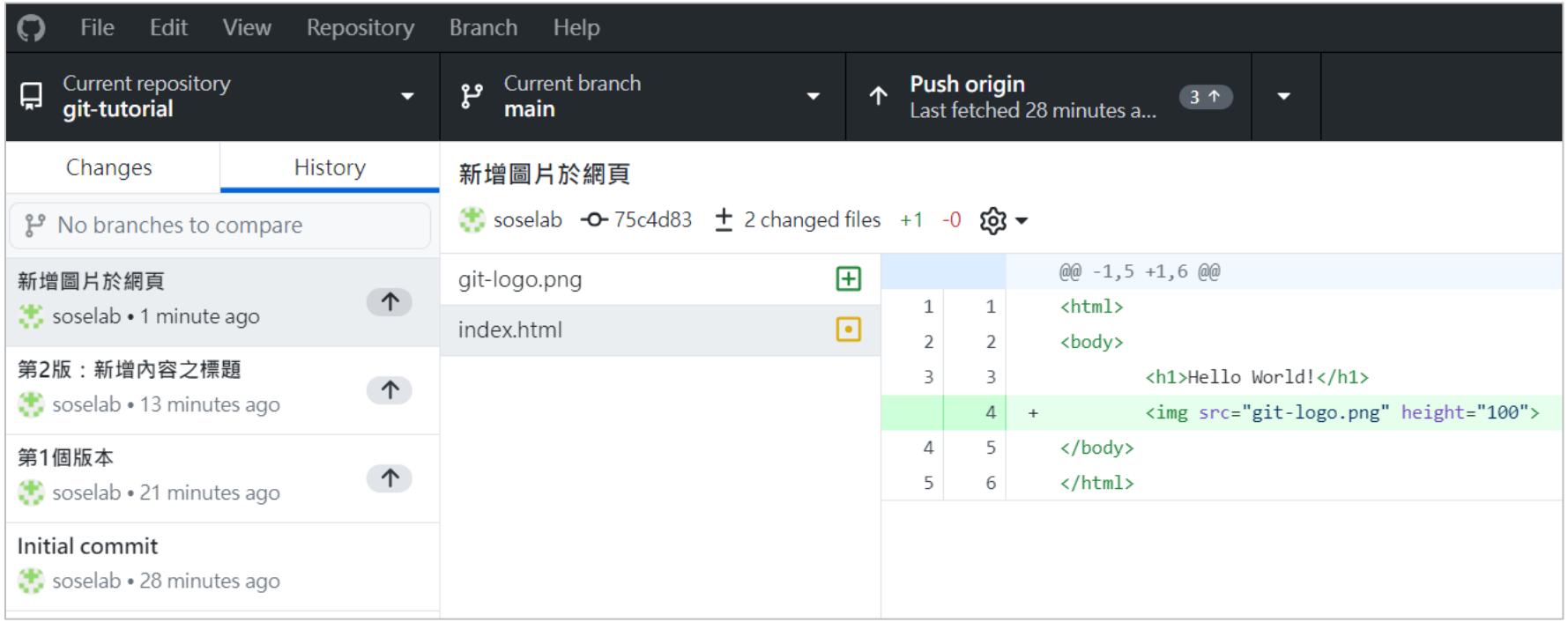
小量改進 - 新增檔案

- 可再查看差異，並交付新版



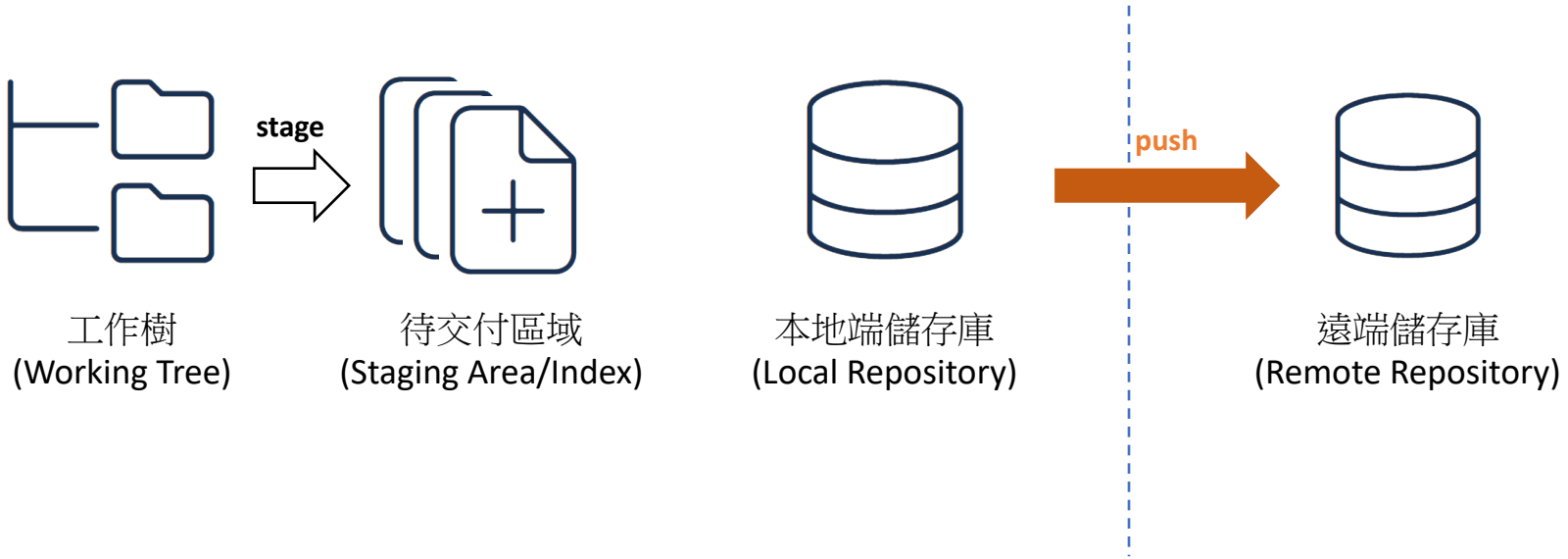
小量改進 - 新增檔案

- 檢視交付歷程
  - 我們可透過此模式逐步新增原始碼與建立交付版本

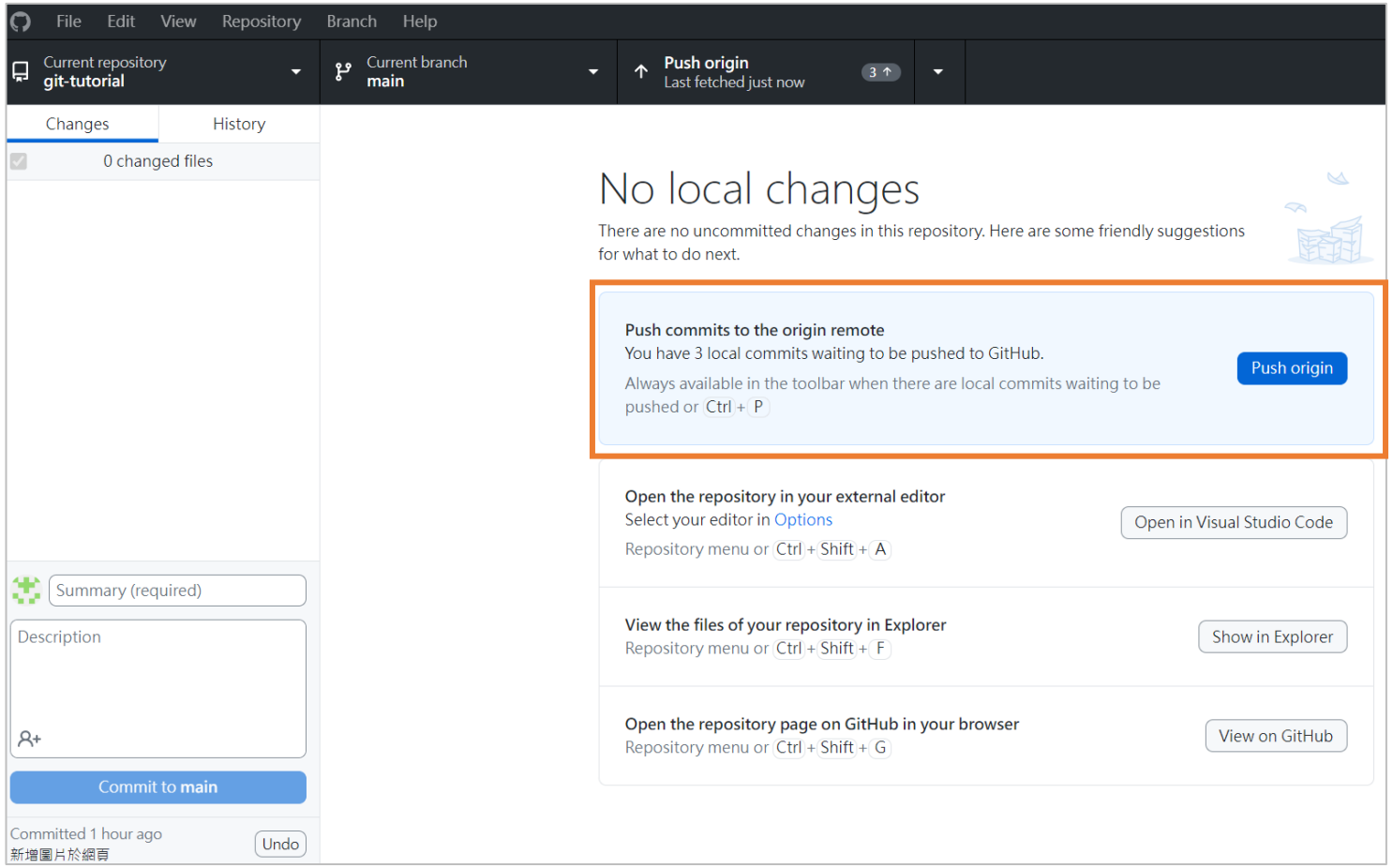


同步遠端儲存庫

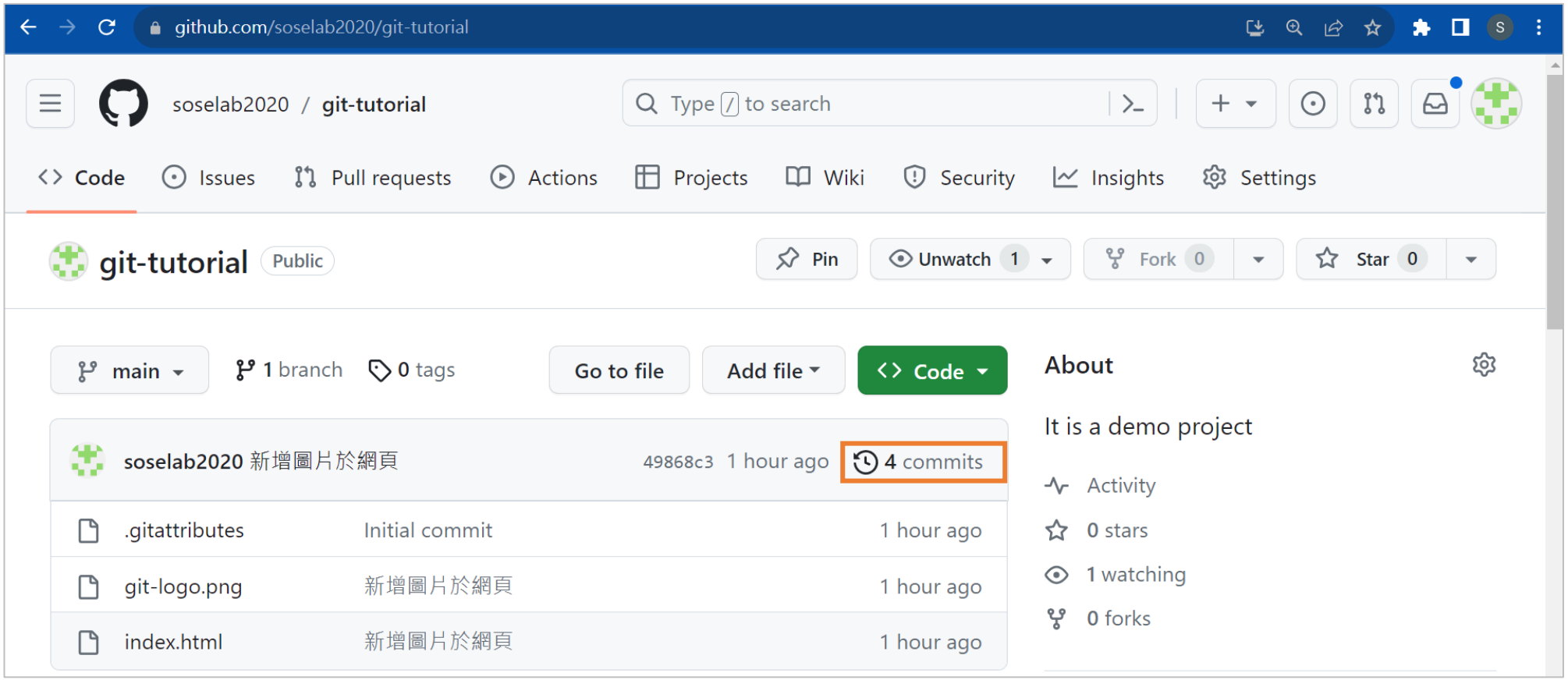
- 我們可將本地端儲存庫的一個交付歷程**推送(push)**至遠端儲存庫



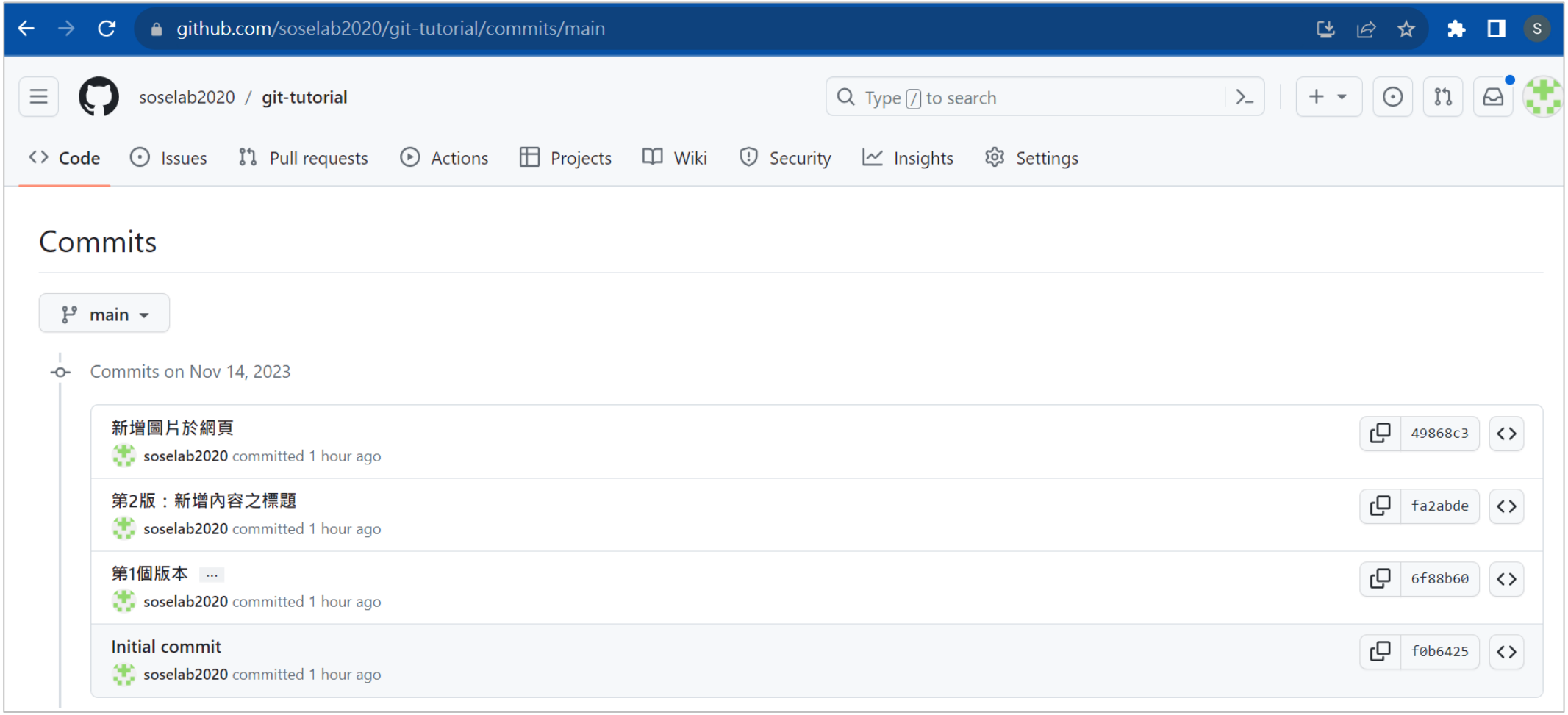
- # 同步遠端儲存庫
  - 點選[Push origin]



- # 完成遠端儲存庫同步
  - 重整遠端儲存庫
    - 可看到所有commit均以推送至遠端儲存庫

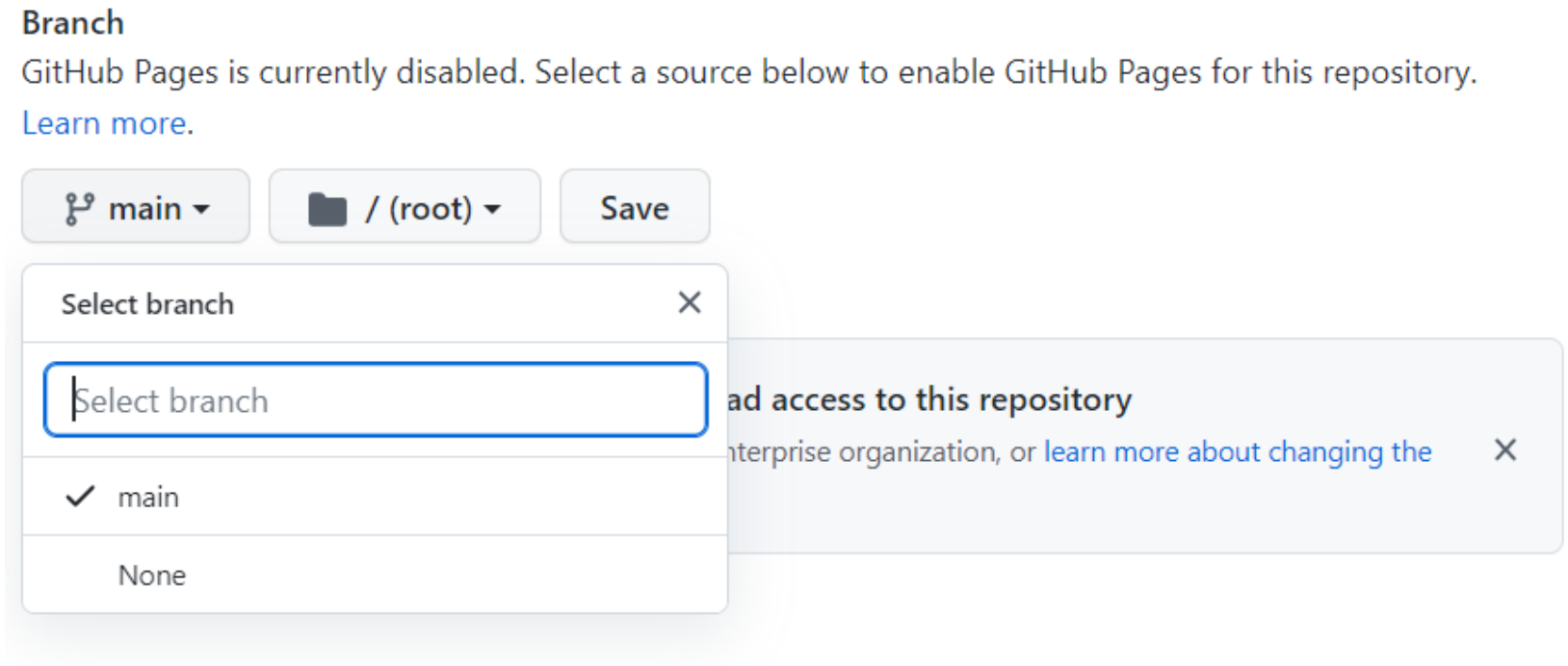


- 查看遠端儲存庫上的交付歷程



- Lab 2
  - 請選定任何一支程式(如HTML、C、Python)等，將其放入剛剛創建的空儲存庫(LR1)，並對其進行修改，以建立多個commit。
    - 請隨時觀看History了解交付歷程。
  - 將本地端的commit都推送到遠端GitHub儲存庫(RR)。
    - 請觀看GitHub之"Commits"內容，以觀察遠端儲存庫與本地儲存庫之交付版本內容是否一致。

- **透過GitHub發布網頁1**
  - 進入Setting頁面左側，點選Pages選項。
  - Branch (分支)選擇main (主分支)，然後點選Save存檔。



- **透過GitHub發布網頁2**
  - 連結你的專屬網址：

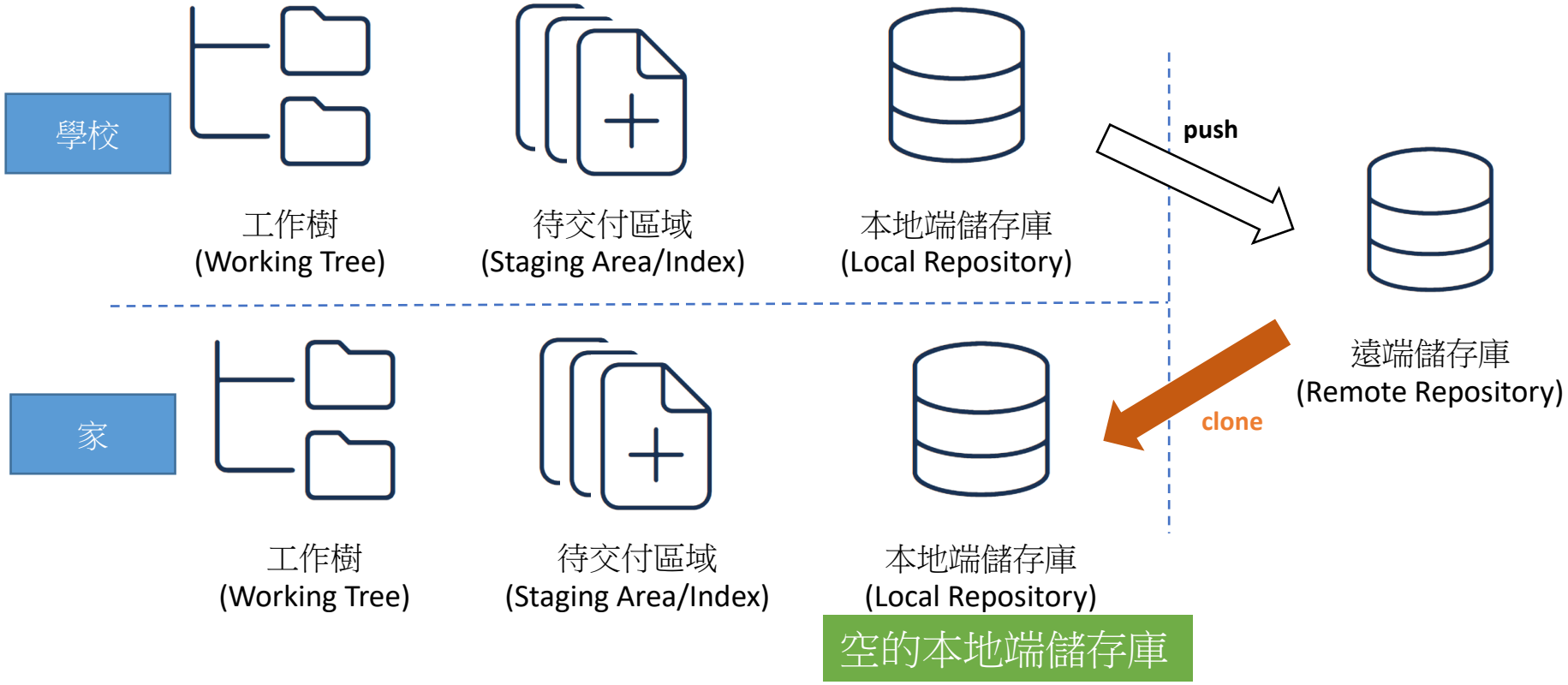
    https://你的帳號. github.io/你的儲存庫名稱/你的網頁檔名

    - 例如：https://albertma2020.github.io/testpage/lab-1-1-example.html
    - 上傳檔案後可能需要稍待1分鐘左右，才能看到網頁。
    - 可將網頁檔名改成為index.html，這樣直接打基底網址就可以連到你的網頁。
    - 用此方法可以發布多個網頁。

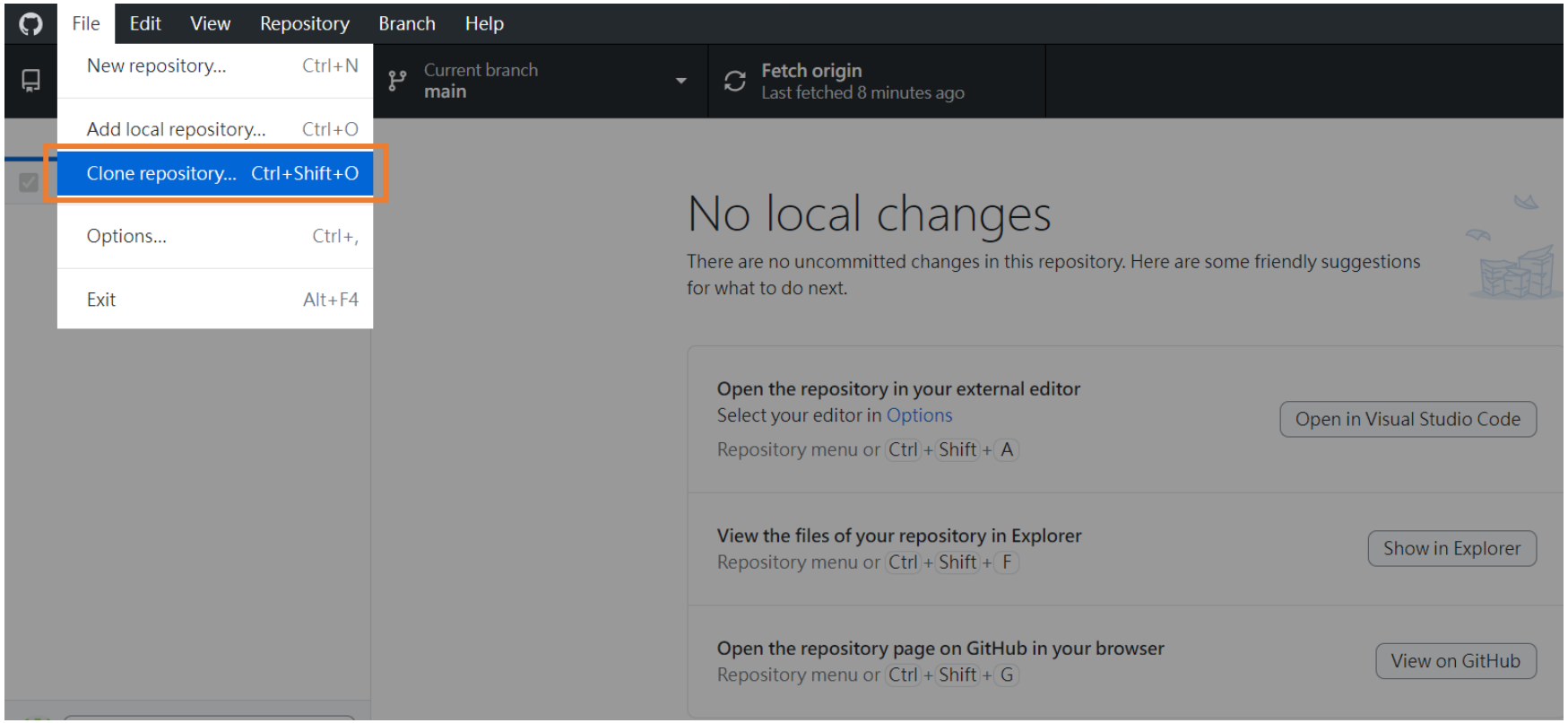
- **Lab 3**
  - 寫一個描述虛擬人物的簡單履歷網頁。
  - 提交(Commit)並推送(Push)程式碼和資源到GitHub。
  - 設定 GitHub 頁面。
  - 提供你在 TronClass 上的 github-pages URL。

複製遠端儲存庫至本地端

- 我們可將**遠端儲存庫**的交付歷程**複製(clone)**到**本地端儲存庫**。
  - 通常發生於多人協作時，或單純取得開源專案原始碼。
  - 多工作環境亦可適用(下面先以此情境進行說明)。

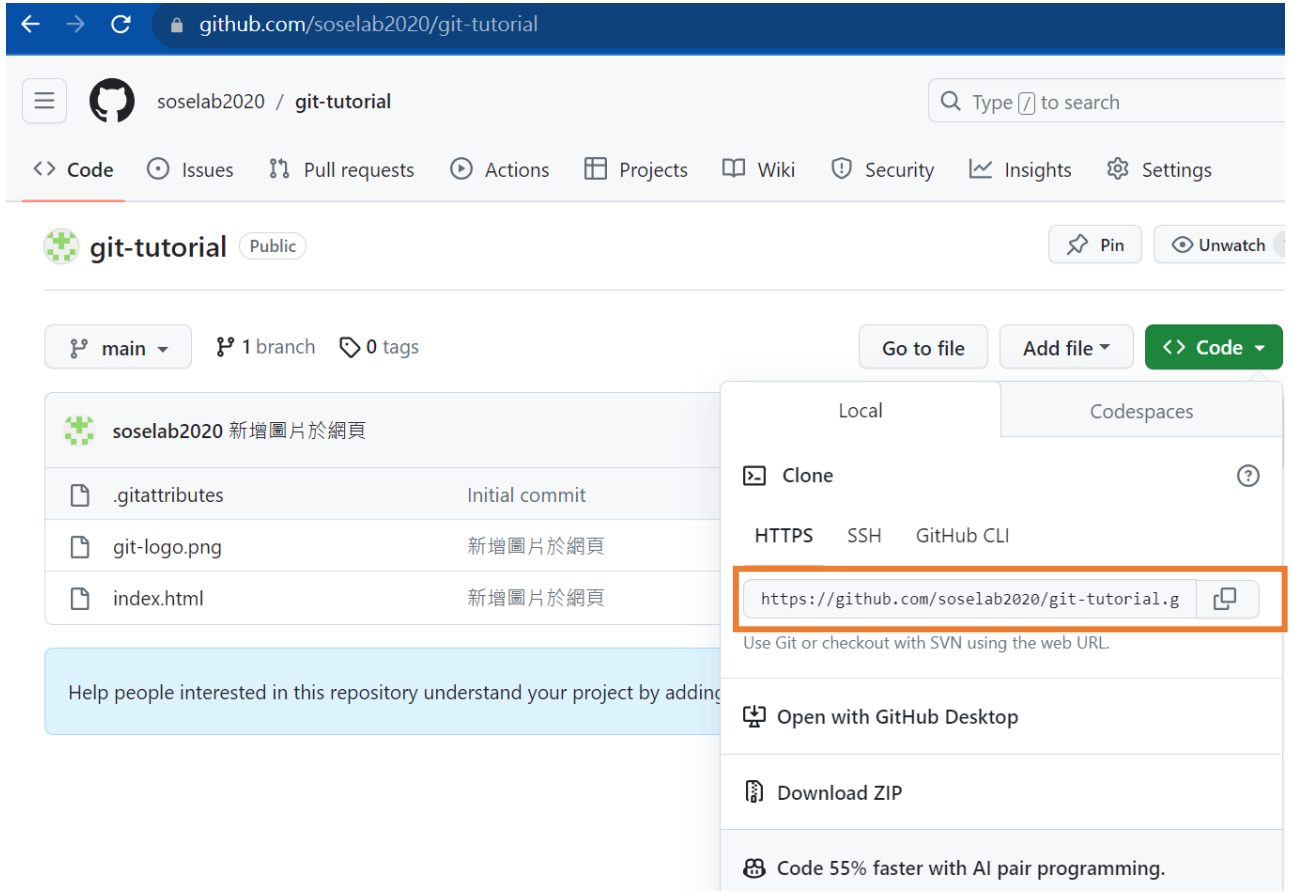


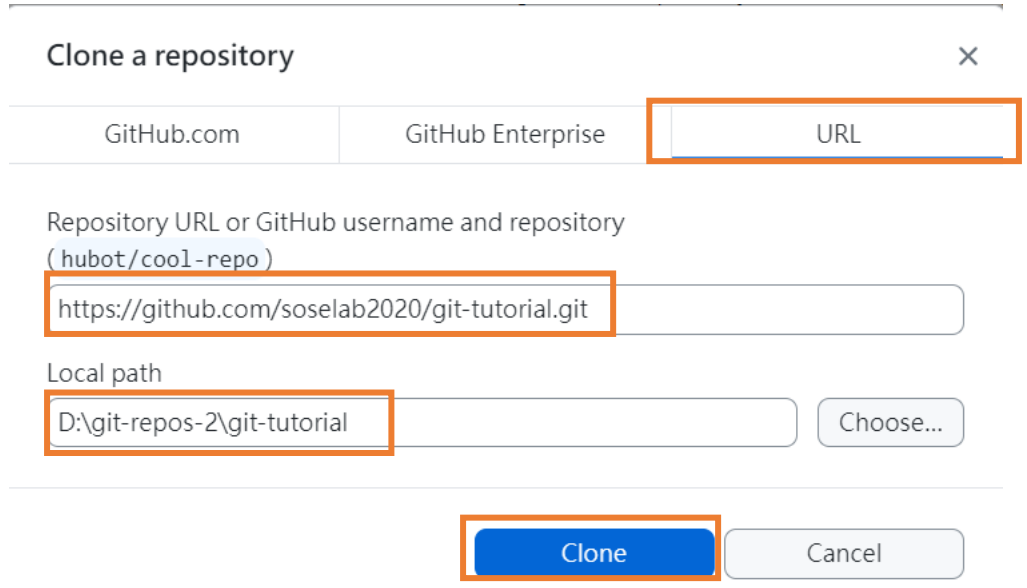
- **複製遠端儲存庫至本地端**
  - 開啟GitHub Desktop，建立遠端儲存庫副本



複製遠端儲存庫至本地端

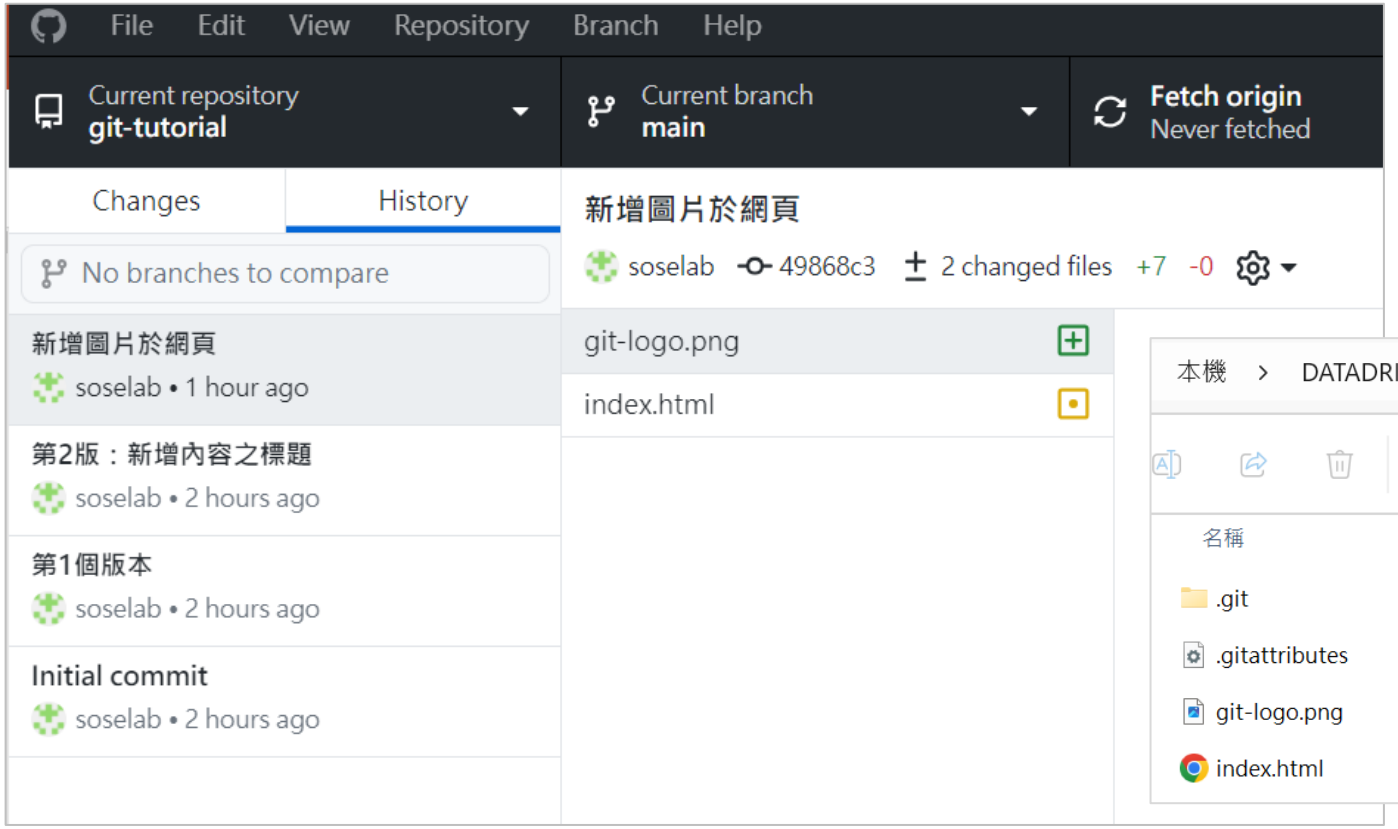
- 設定遠端儲存庫網址與本地端工作目錄(working tree)



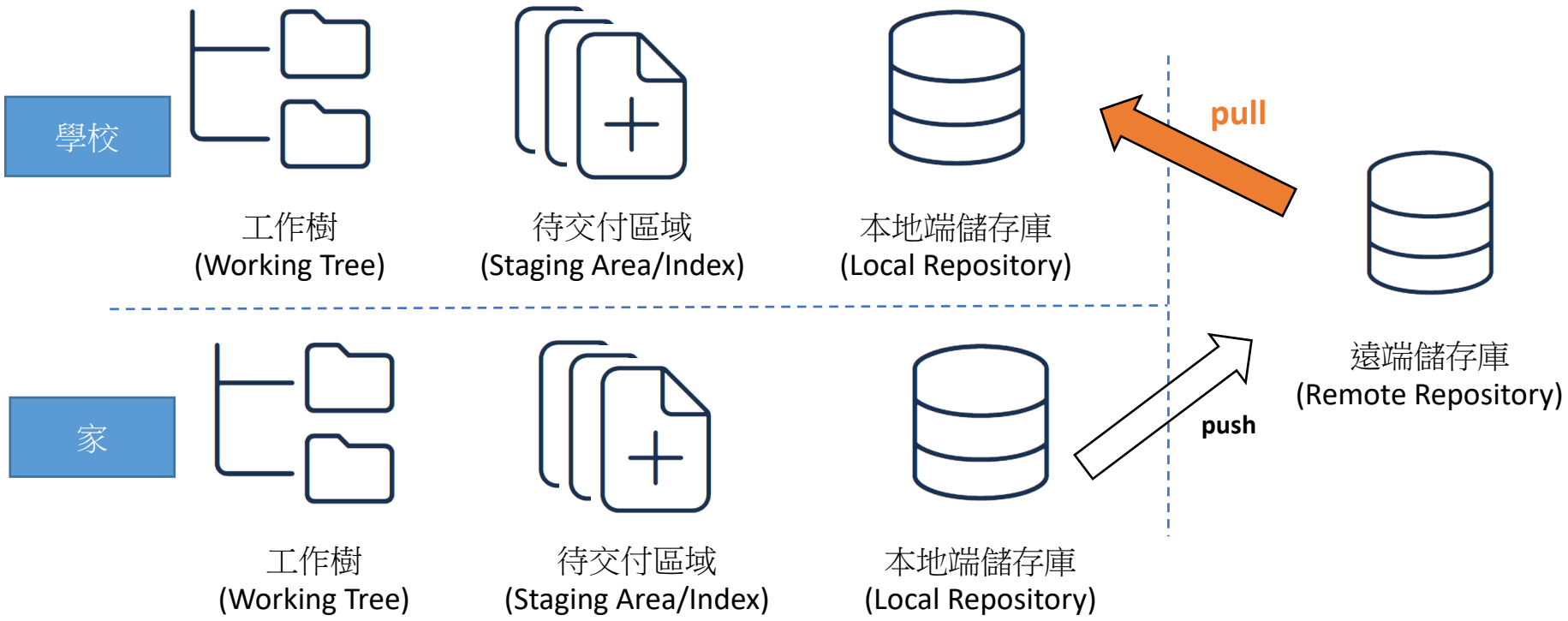


(我們這邊以不同目錄來模擬
不同台電腦之情況)

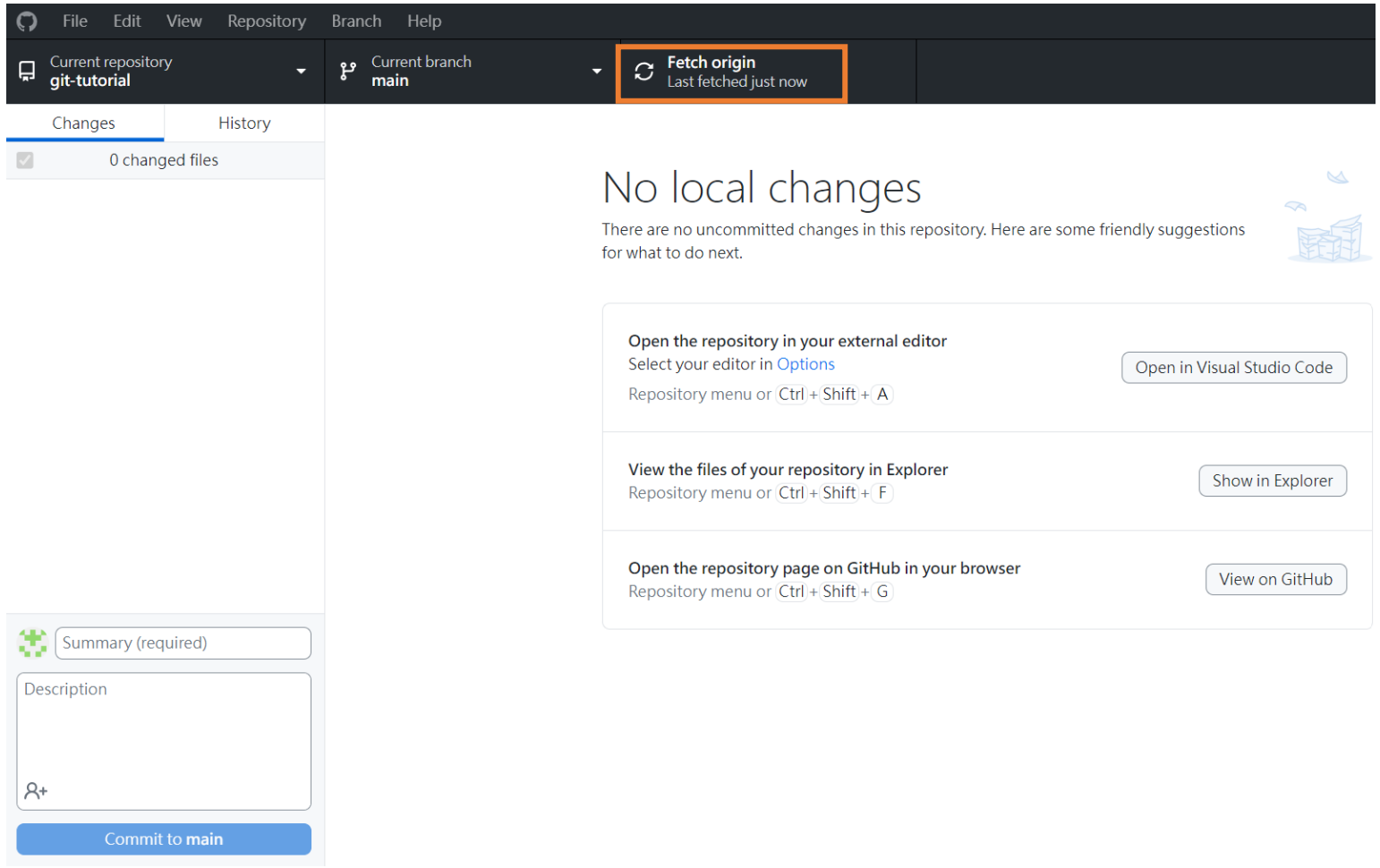
- # 完成本地端儲存庫同步
  - 後續即可在此本地端儲存庫繼續工作，包含新增新的commit，以及將commit推送至GitHub



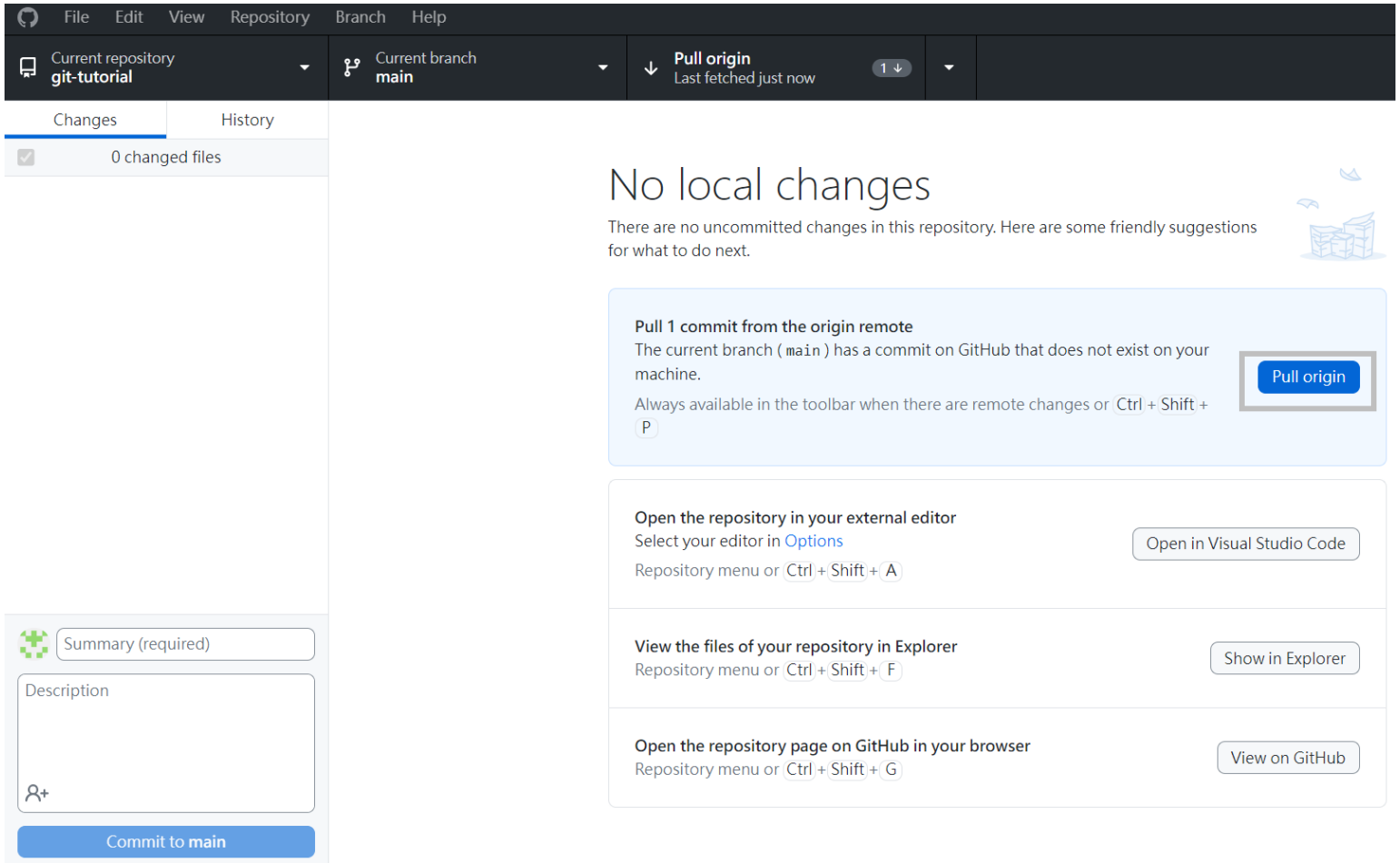
- 不同的本地端儲存庫間的同步



- # 不同的本地端儲存庫間的同步
  - 先點選[Fetch origin]，偵測遠端之版本變化



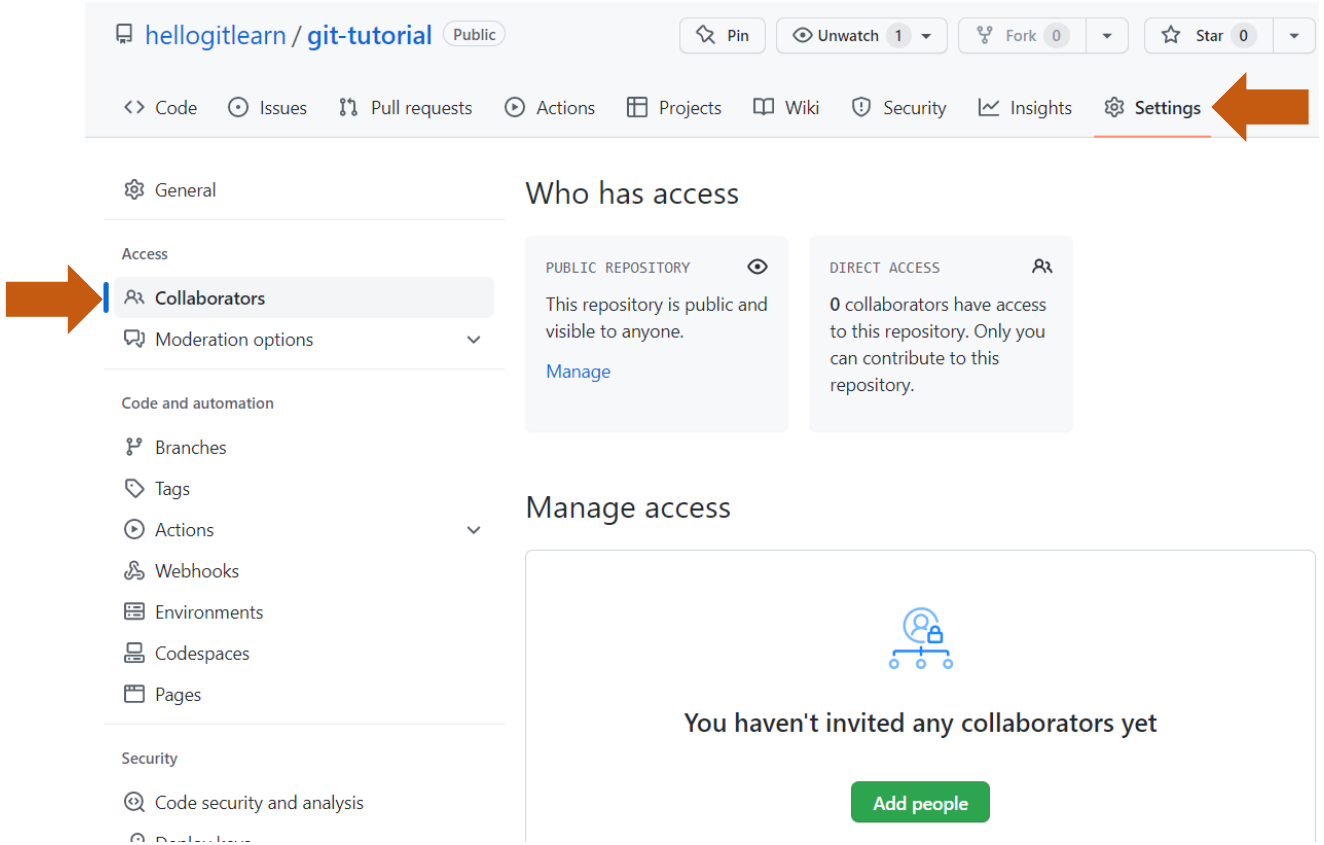
- # 不同的本地端儲存庫間的同步
  - 先點擊[Pull origin]，即可將新版原始碼更新回來



- Lab 4
  - 請透過Clone功能將剛剛的遠端儲存庫(我們稱之為RR)複製到不同目錄下，以建立另一個本地儲存庫(我們稱之為 LR2)。
    - 我們是透過此做法來模擬多個工作環境之程式碼同步。
  - 請對程式進行修改後建立1~N個commit，並推送至GitHub。
    - 請觀察GitHub之Commits是否都與LR2一致。
  - 請切回原本的本地儲存庫(LR1)，再透過Fetch與Pull功能將剛剛LR2新增的commit更新進來。
    - 請觀察LR1的History是否也與LR2一致。

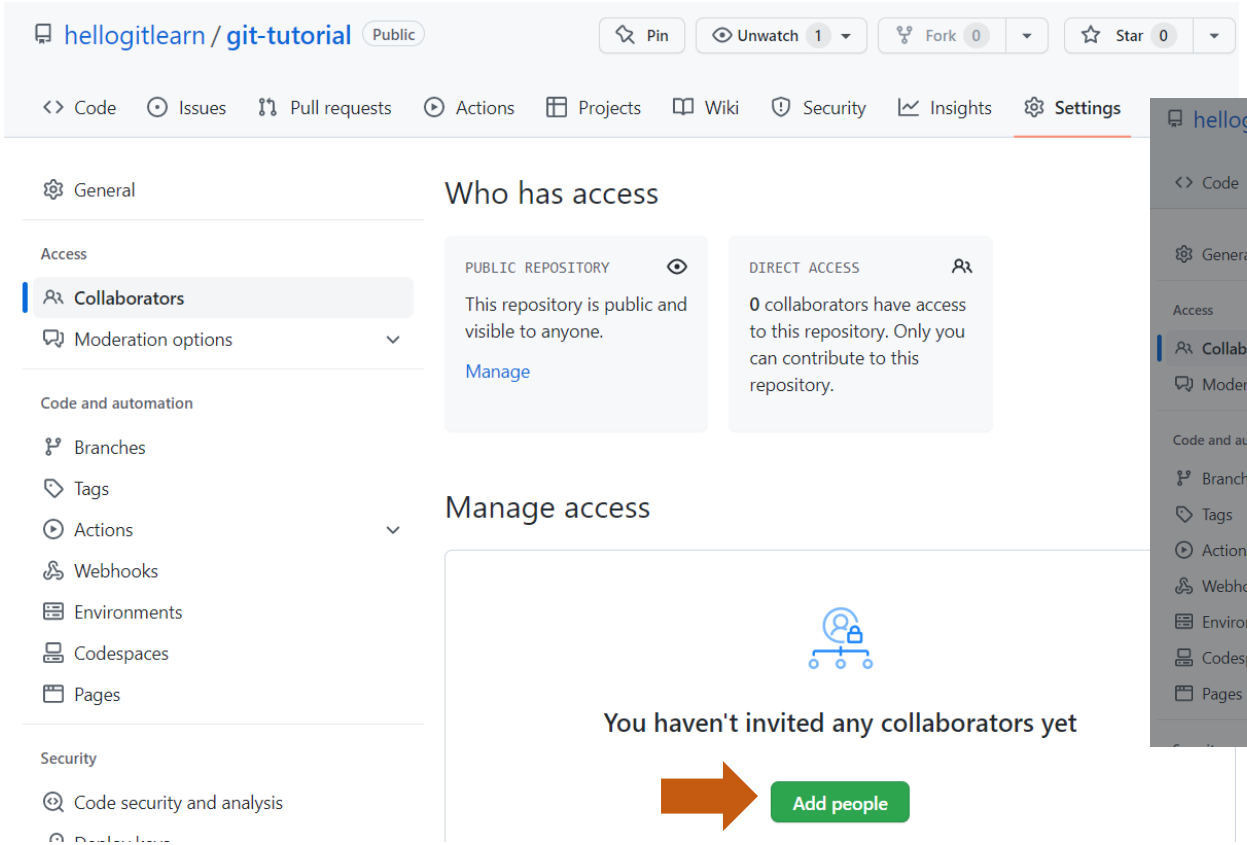
GitHub 多人合作 - Collaborators

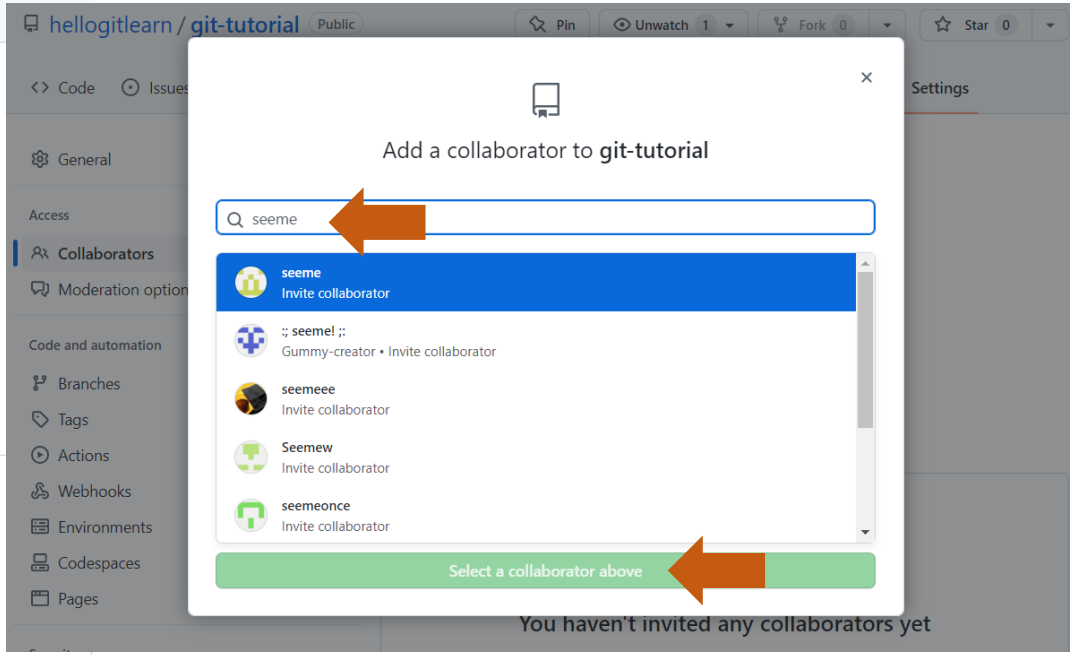
- Collaborators適用於一個team之間的合作
- 負責開這個repository的人，進入settings，加入collaborators的帳號



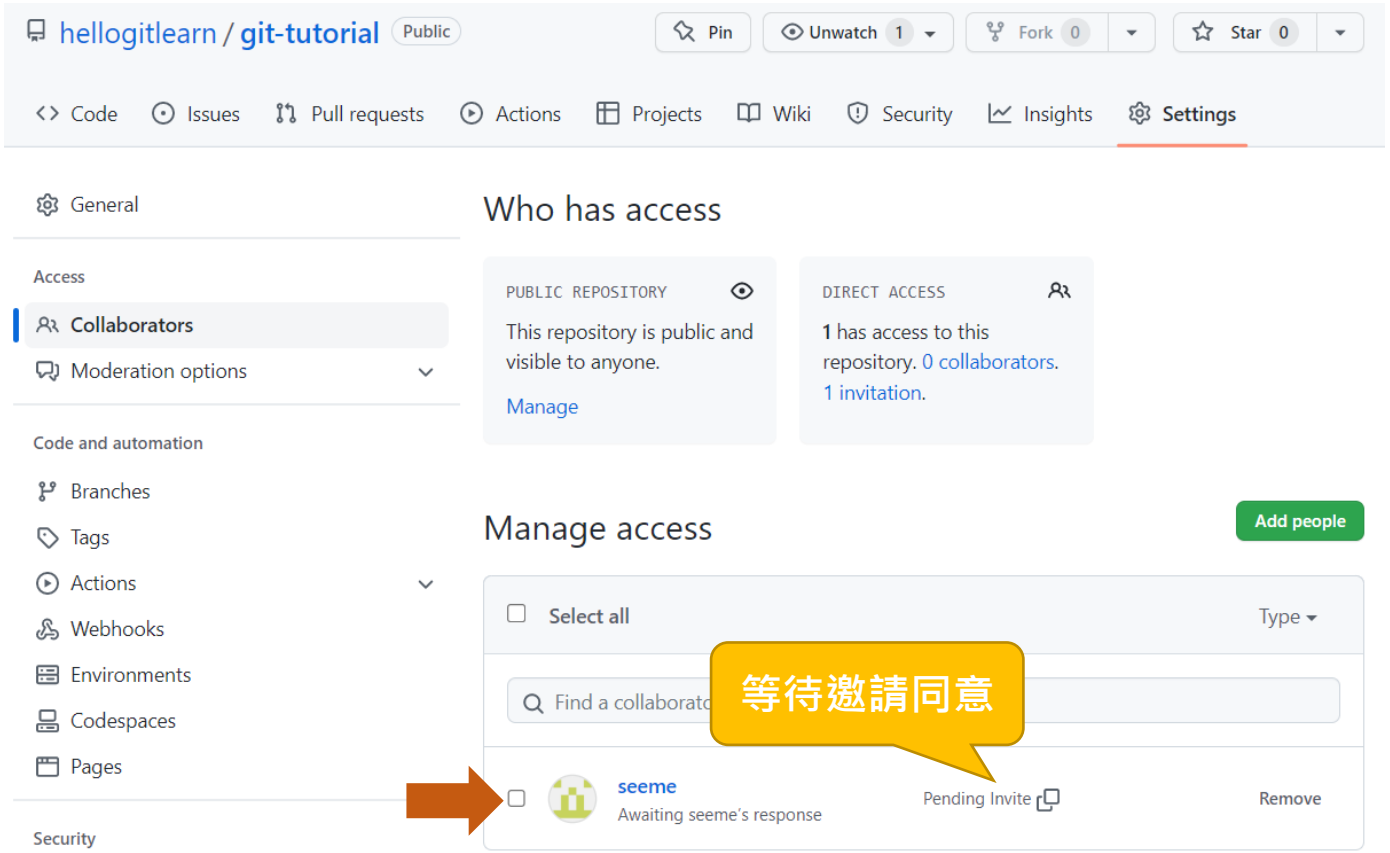
GitHub 多人合作 - Collaborators

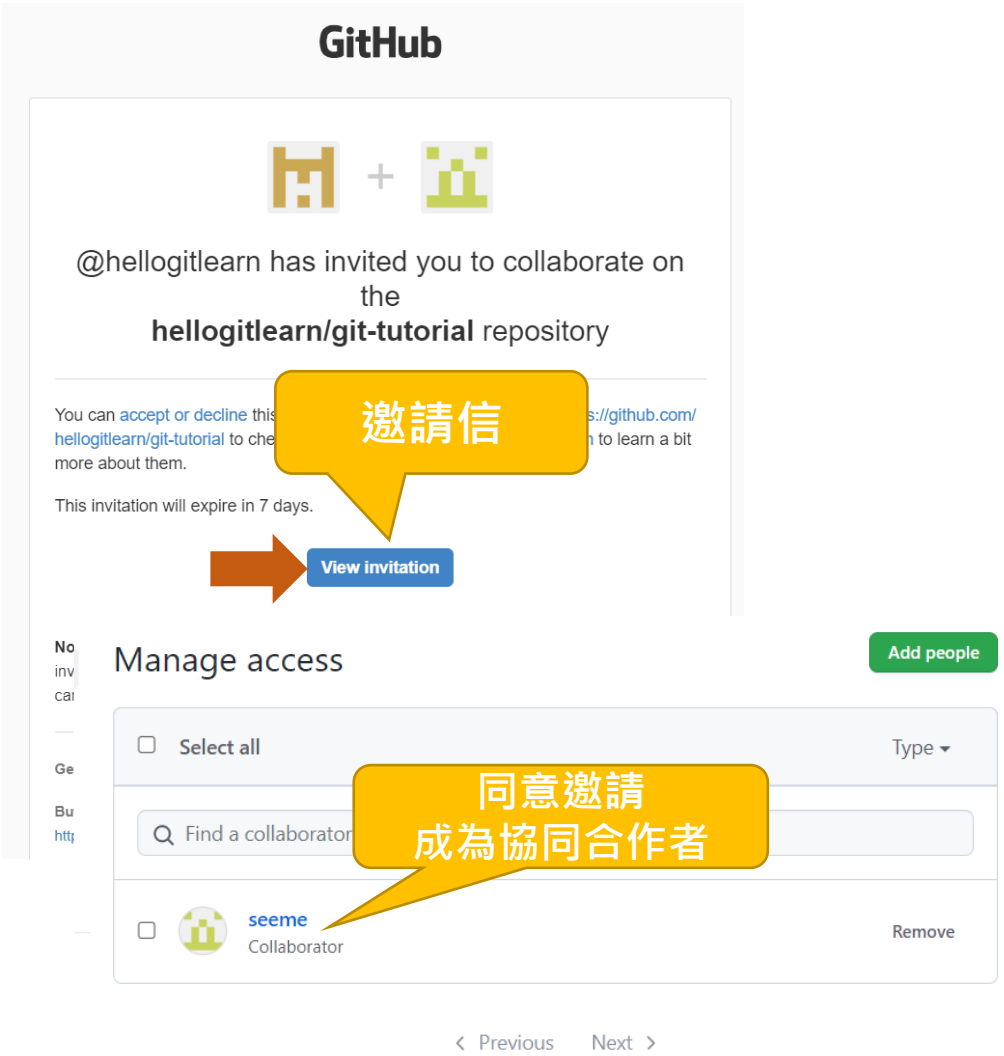
- 新增協同合作者(Collaborator)





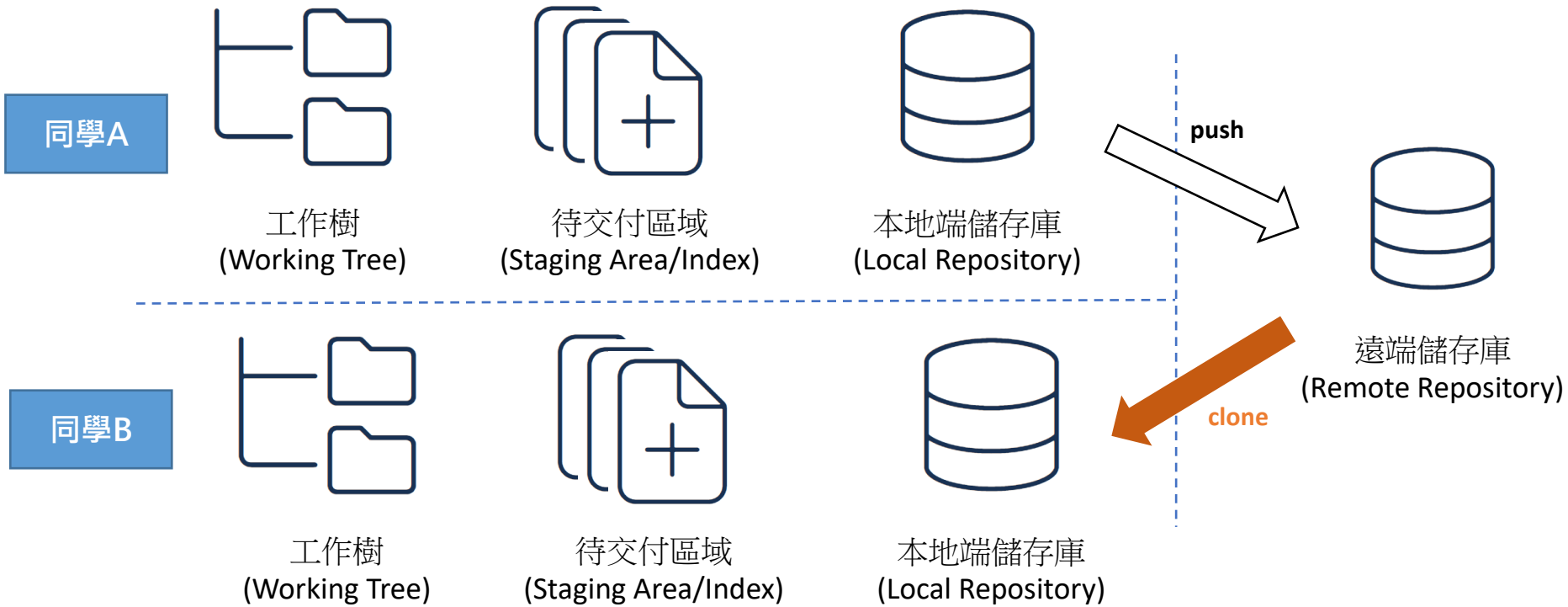
• GitHub 多人合作 - Collaborators





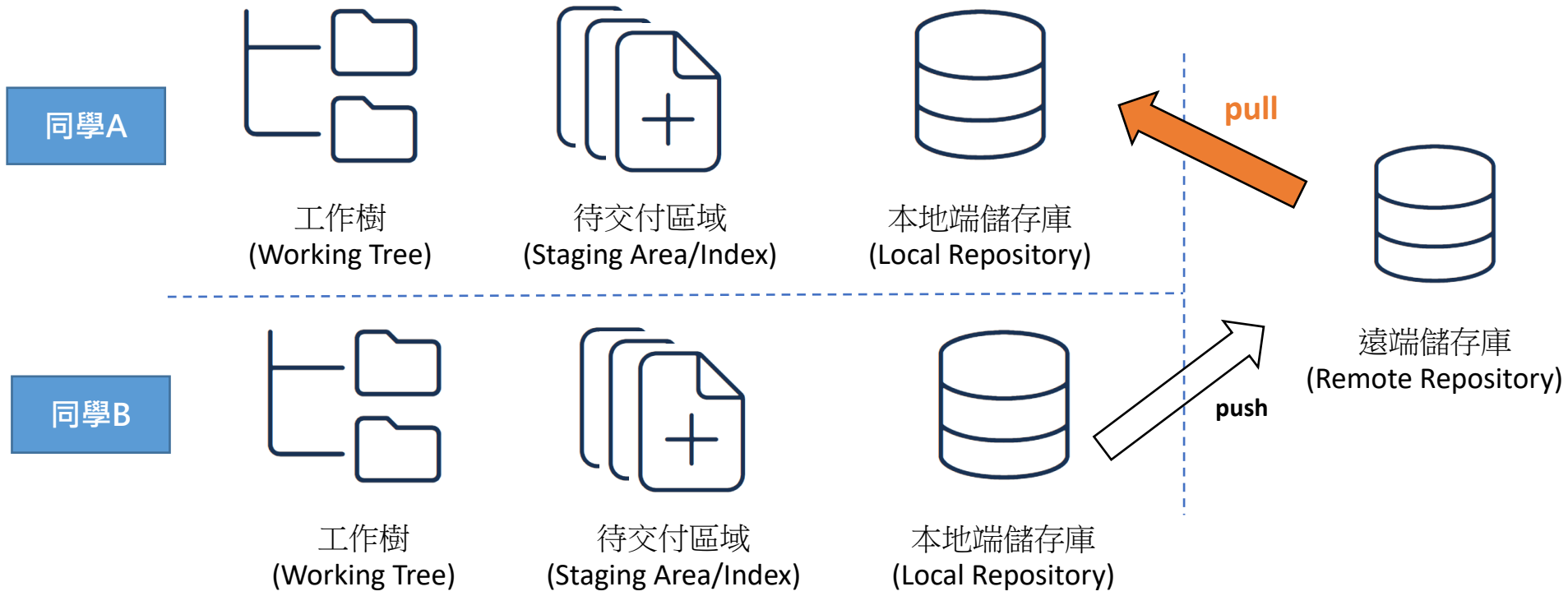
GitHub 多人合作 - Collaborators

- 將**遠端儲存庫**(擁有者所維護)的交付歷程**複製(clone)**到**本地端儲存庫**(協同合作者所維護)



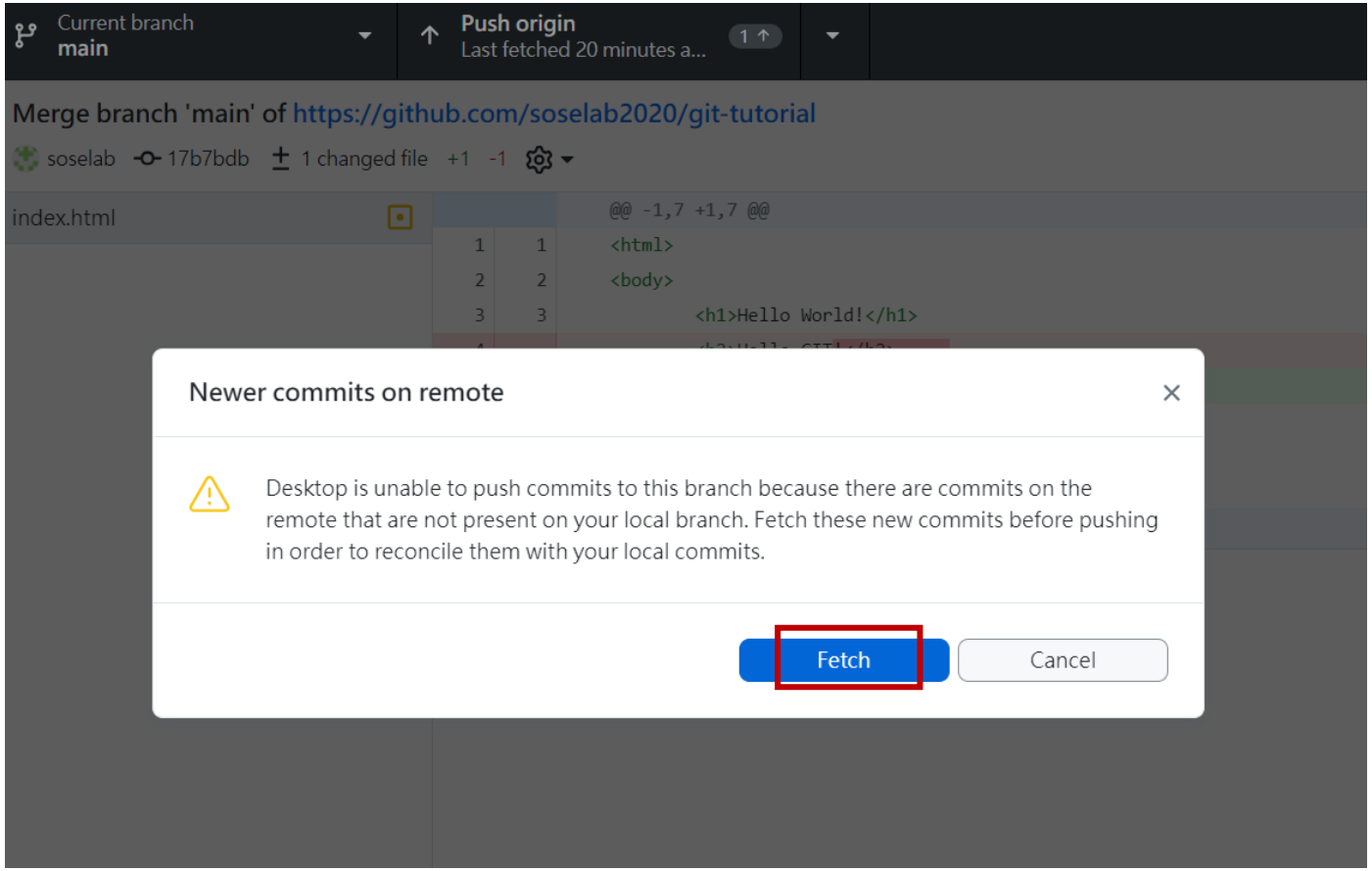
GitHub 多人合作 - Collaborators

- 協同合作者可以將修改的交付透過push指令同步至遠端儲存庫
- 而其他團隊成員可以透過pull指令將遠端儲存庫的最新交付版本同步至本地端儲存庫



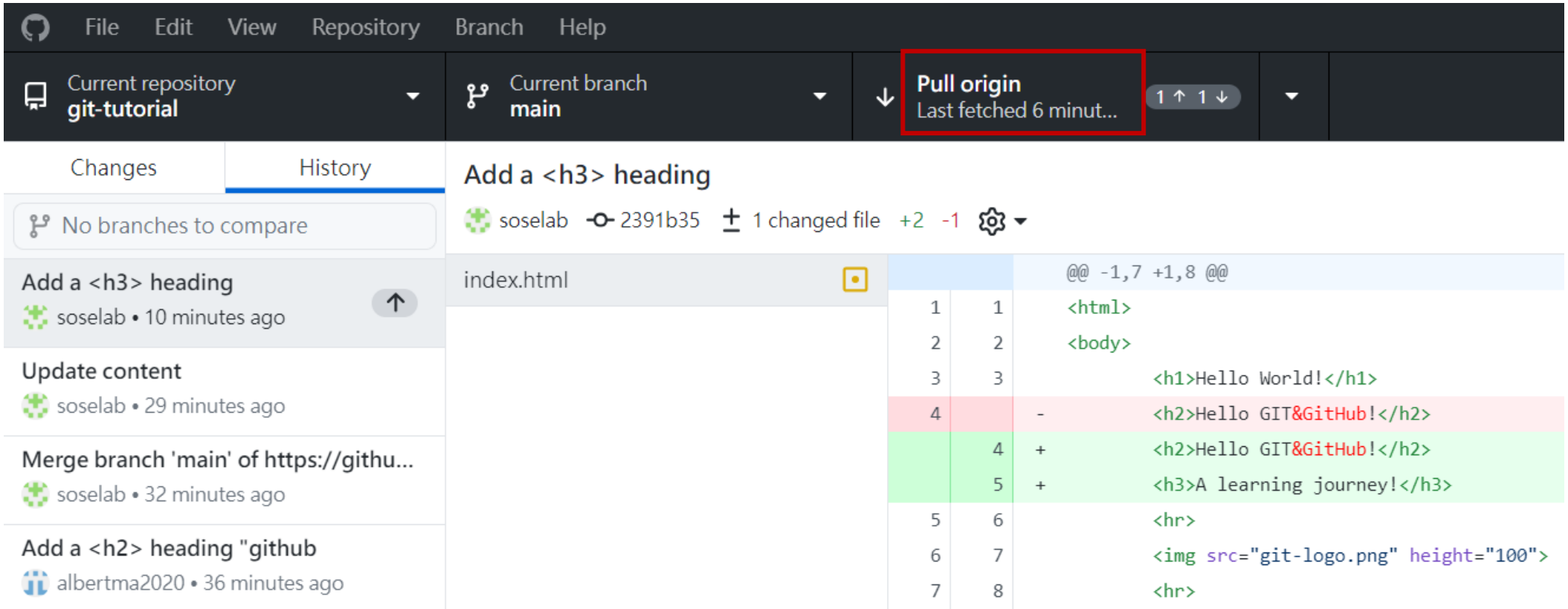
GitHub 多人合作 – 衝突解決

- 可能會遇到無法Push至GitHub的情況：
  - 因為協作者已經先推送了新版，這時應先Fetch遠端版本



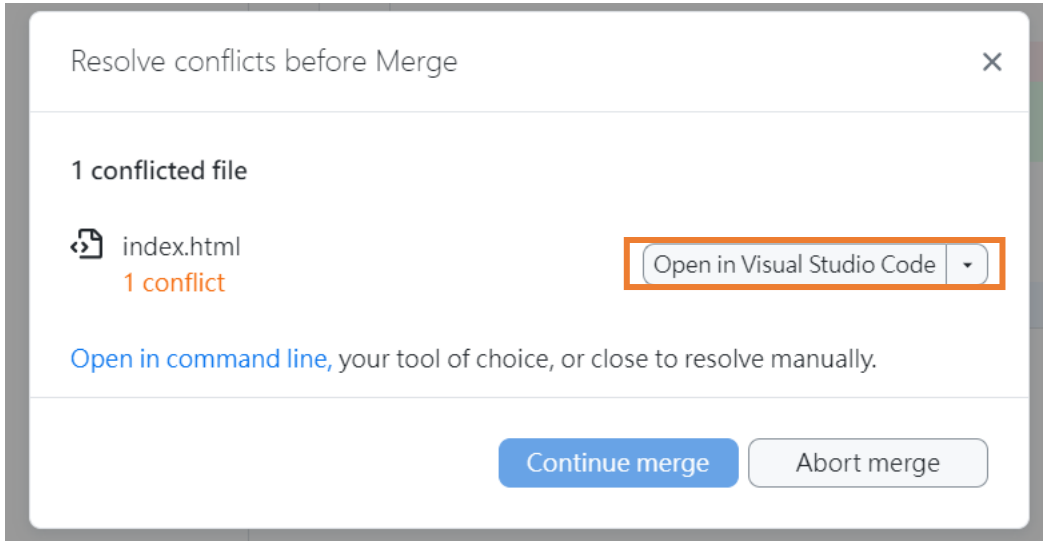
GitHub 多人合作 – 衝突解決

- 此時介面上會出現"Pull origin"的選項，可直接點擊。



GitHub 多人合作 – 衝突解決

- Pull後可能會出現"conflict"之警告訊息，我們只要開啟VS Code去解決即可。



• GitHub 多人合作 – 衝突解決

- 有四種修正選項，可選擇"Accept Both Changes"，再去做適度修改。
  - 請兩個版本的作者要進行討論，以決定最終的合併方式。

index.html

D: > git-repos > git-tutorial > index.html > html > body > ?

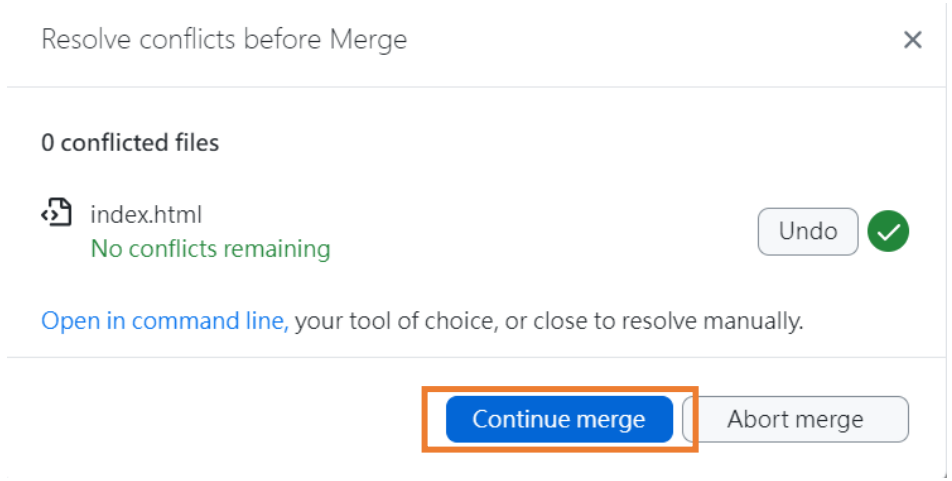
```
1   <html>
2   <body>
3       <h1>Hello World!</h1>
4       <h2>Hello GIT&GitHub!</h2>
        Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
5   <<<<<<< HEAD (Current Change)
6       <h3>A learning journey!</h3>
7   =======
8       <h3>Have fun!</h3>
9   >>>>>>> 60add20c631a438888784480109a6fda4a4d1daf (Incoming Change)
10      <hr>
11      <img src="git-logo.png" height="100">
12      <hr>
13  </body>
14  </html>
15
```

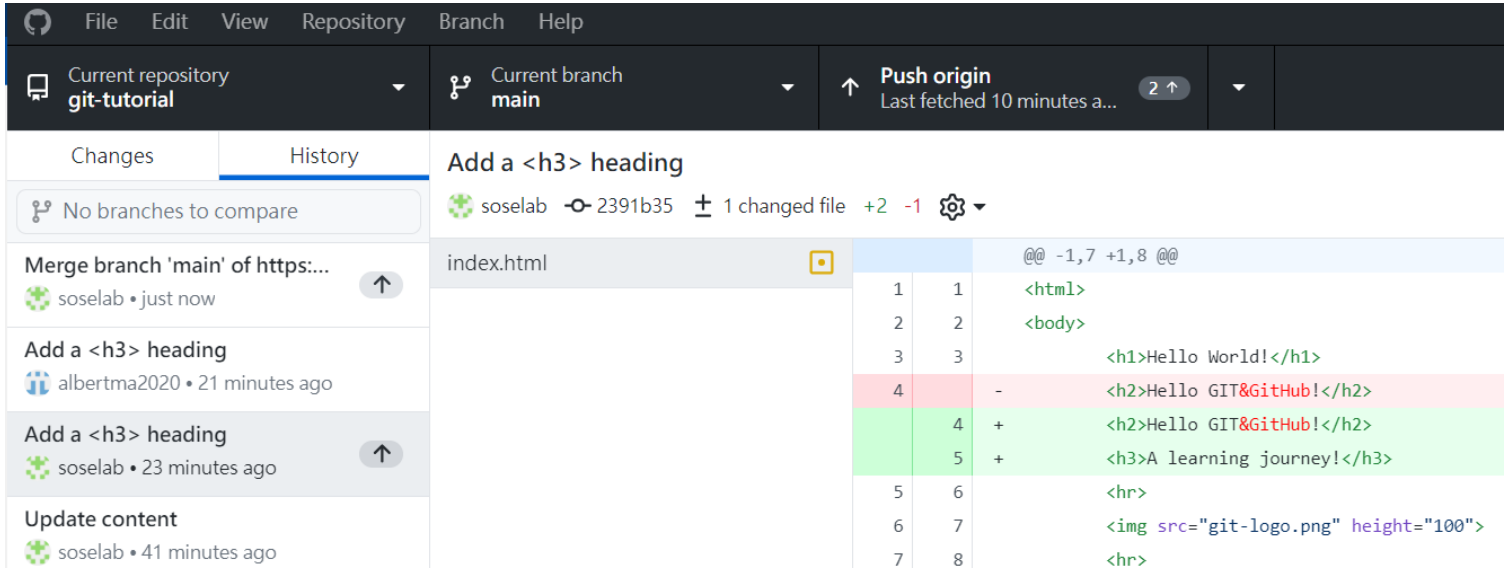
index.html

D: > git-repos > git-tutorial > index.html > ...

```
1   <html>
2   <body>
3       <h1>Hello World!</h1>
4       <h2>Hello GIT&GitHub!</h2>
5       <h3>A learning journey!</h3>
6       <h3>Have fun!</h3>
7       <hr>
8       <img src="git-logo.png" height="100">
9       <hr>
10  </body>
11  </html>
```

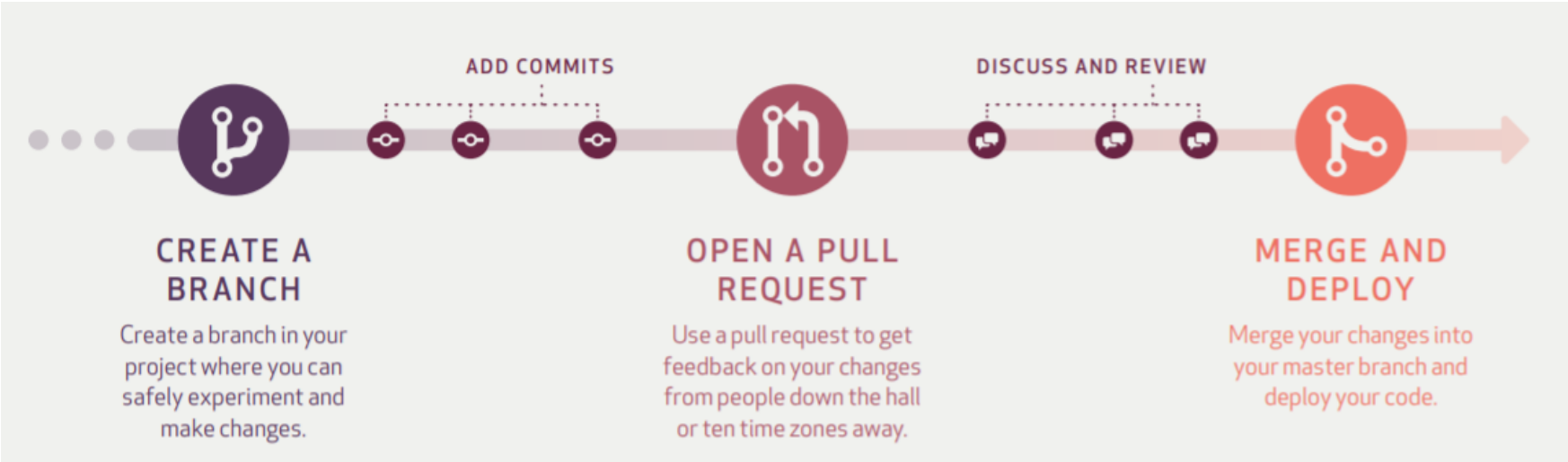
- # GitHub 多人合作 – 衝突解決
  - 修正結束後，回到GitHub Desktop即可再點擊"Continue merge"，即可完成此次的版本合併。





- **Lab 5**
  - 請大家邀請一位夥伴成為你GitHub Repo之Collaborator，你也成為你夥伴的GitHub Repo之Collaborator。
  - 請Clone你夥伴的Repo，並對其進行修改與發佈commits，且推送至GitHub。
  - 請Fetch與Pull你夥伴在你的Repo新增之commit，確保雙方貢獻的內容都有包含在歷程中。
  - 請練習兩位夥伴同時對同一個Repo進行Push與Pull，若有Conflict請練習將其解決。

- # GitHub 多人合作 – Push, Pull, Pull Request
  - 後續可類似先前多工作環境的模式，在多人間對同一個儲存庫(同一個分支)持續Push與Pull。
  - 若要更好的合作，應各自建立分支(branch)後，再進行合併(merge)。
  - 更理想的合作模式可參考GitHub Flow，透過Pull Request來提醒團隊有分支合併之請求。



iGER

Initiating General Education Renaissance