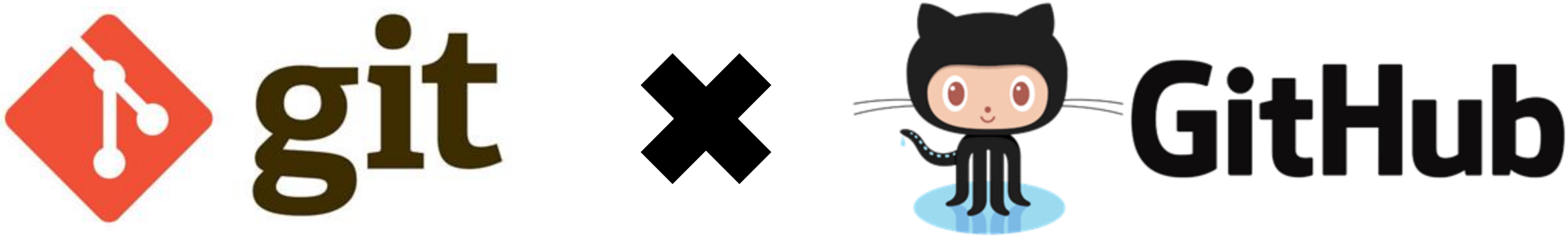


王緯宸

eddiewang@ispan.com.tw

2025 - 07 - 13

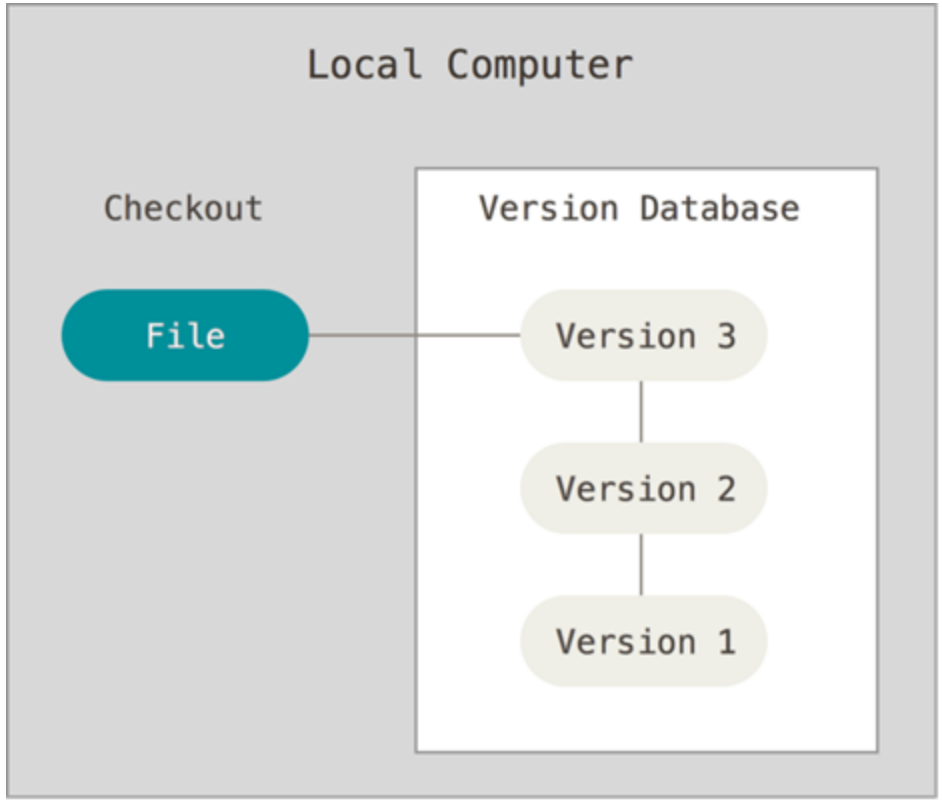
什麼是版本控制?

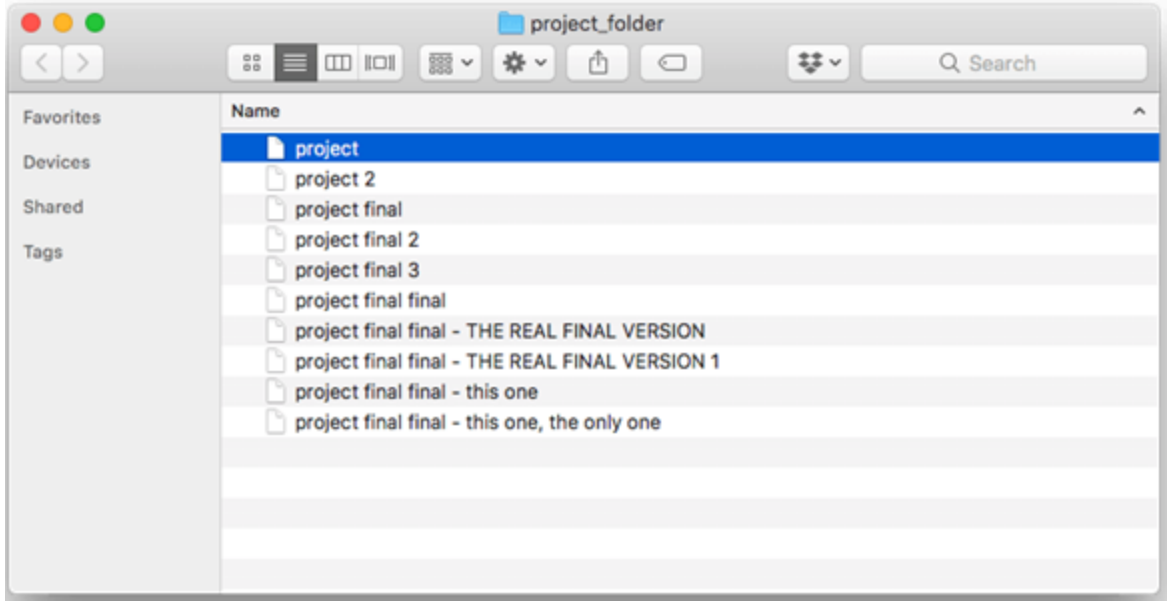
紀錄一個檔案變更的部分，當你未來想回到某一個版本時就可以簡單呼叫。

大致上分為【本地端】、【集中化】、【分散式】三種版本控制。

本地端版本控制

做法簡單，但容易覆蓋錯檔案。



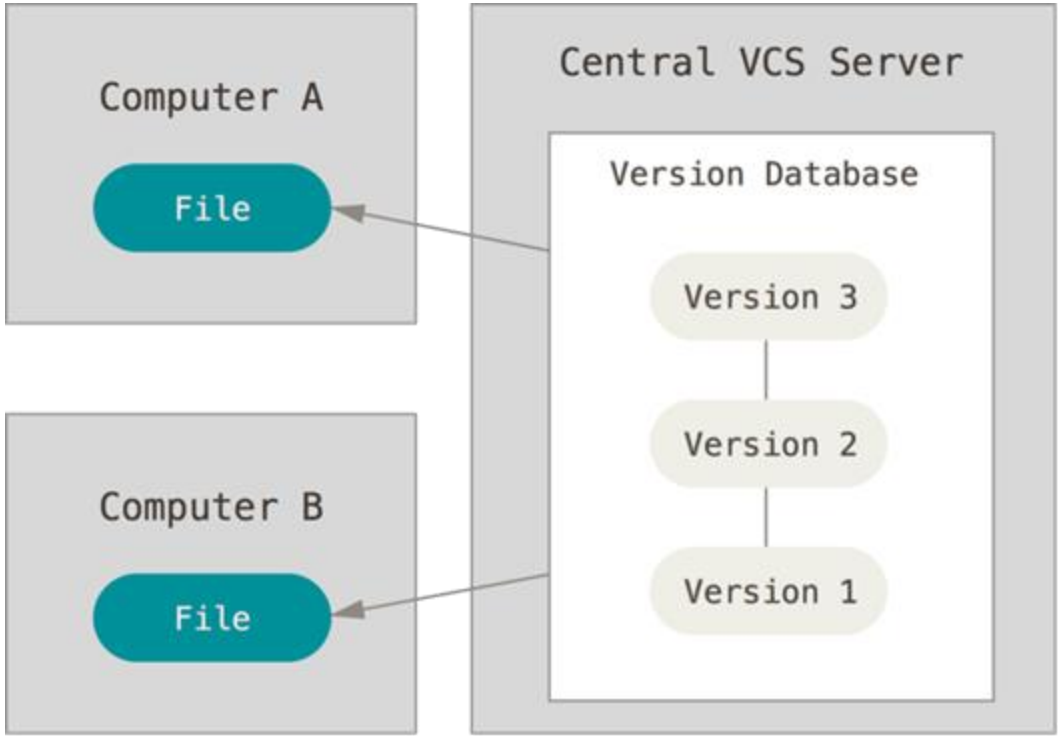


- 使用者運用了「備份」和「檔案名稱」讓開發者能查找不同版本的檔案。
- 這種作法有以下缺點：
  - 無法得知檔案版本之間的差異
  - 無法得知備份的原因
  - 無法追蹤修改者和修改的內容

圖片來自：https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

集中化的版本控制系統

中央伺服器故障時，所有人都無法存取檔案。



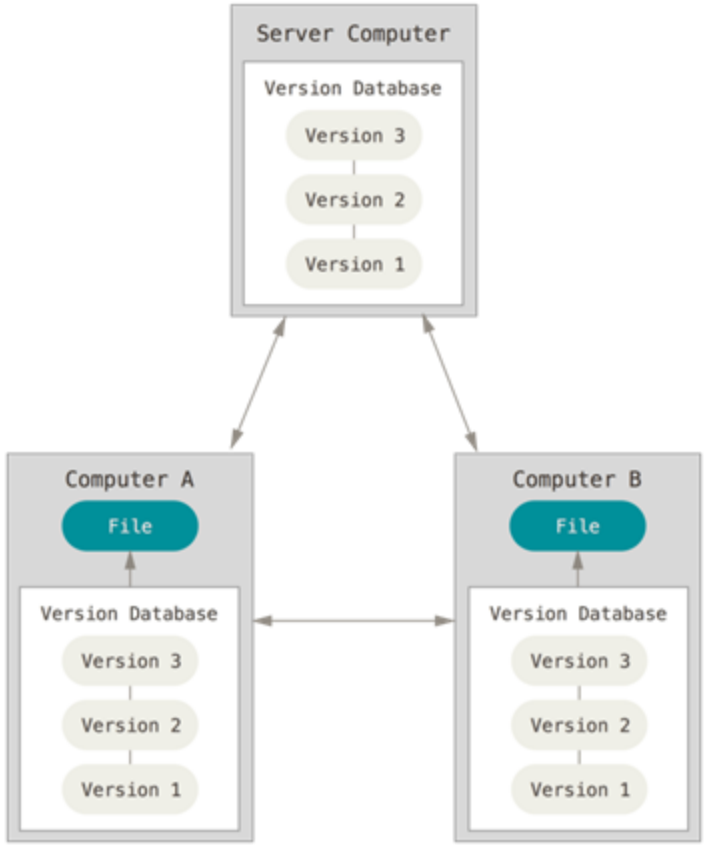
SUBVERSION

PERFORCE

- 資料統一由伺服器管理，開發者連到伺服器讀取資料。
- 解決本地端版本控制無法與其他開發者共同合作的問題。
- 當伺服器損壞時，資料可能會跟著遺失。

圖片來自：https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

分散式版本控制系統



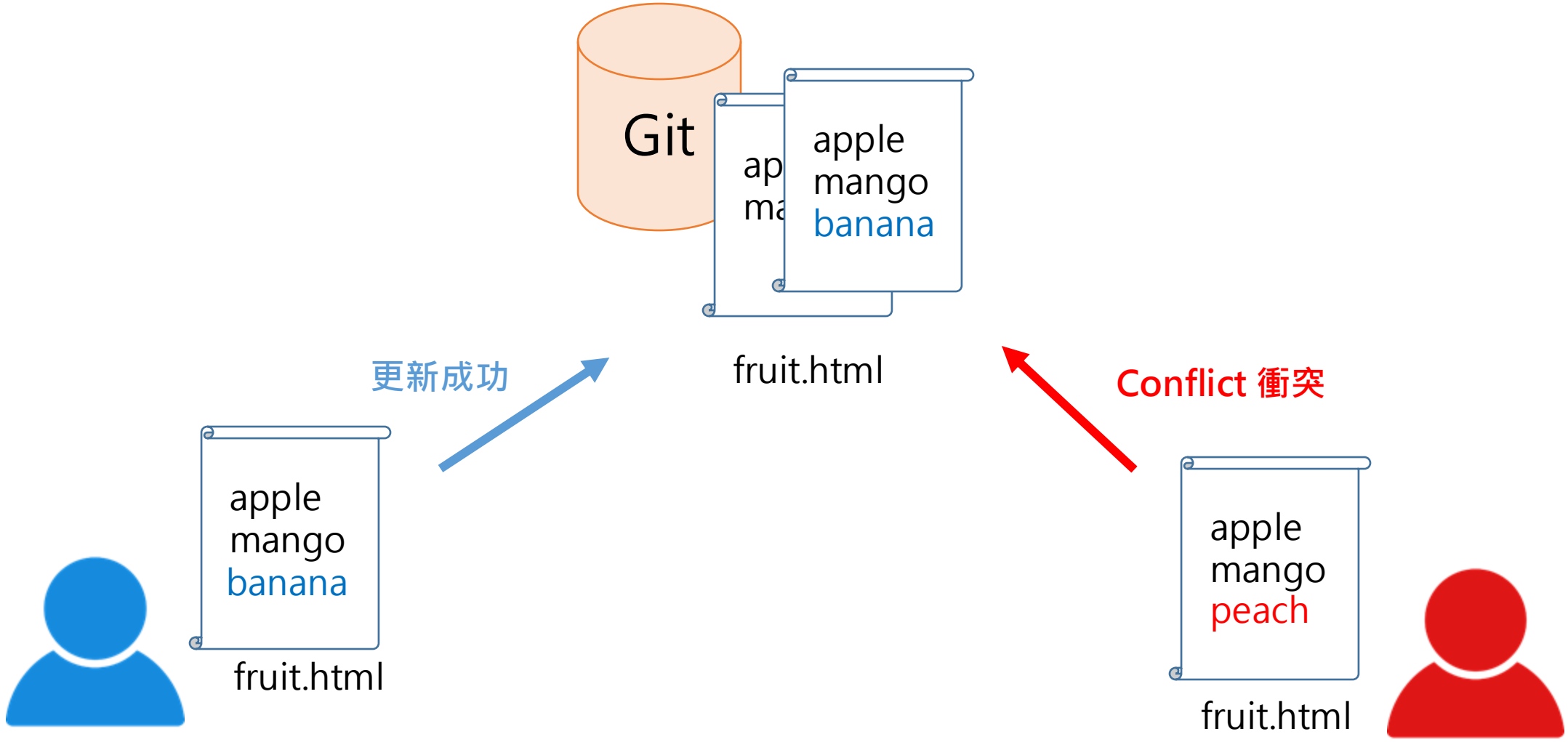
- Distributed Version Control Systems，簡稱 DVCSs
- 開發者不但可以取出最新的檔案資訊，還可以將整個檔案夾備份。
- 假如有任何一個伺服器損壞，就可以採用開發者的資料來進行還原。

圖片來自：https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

什麼是Git?

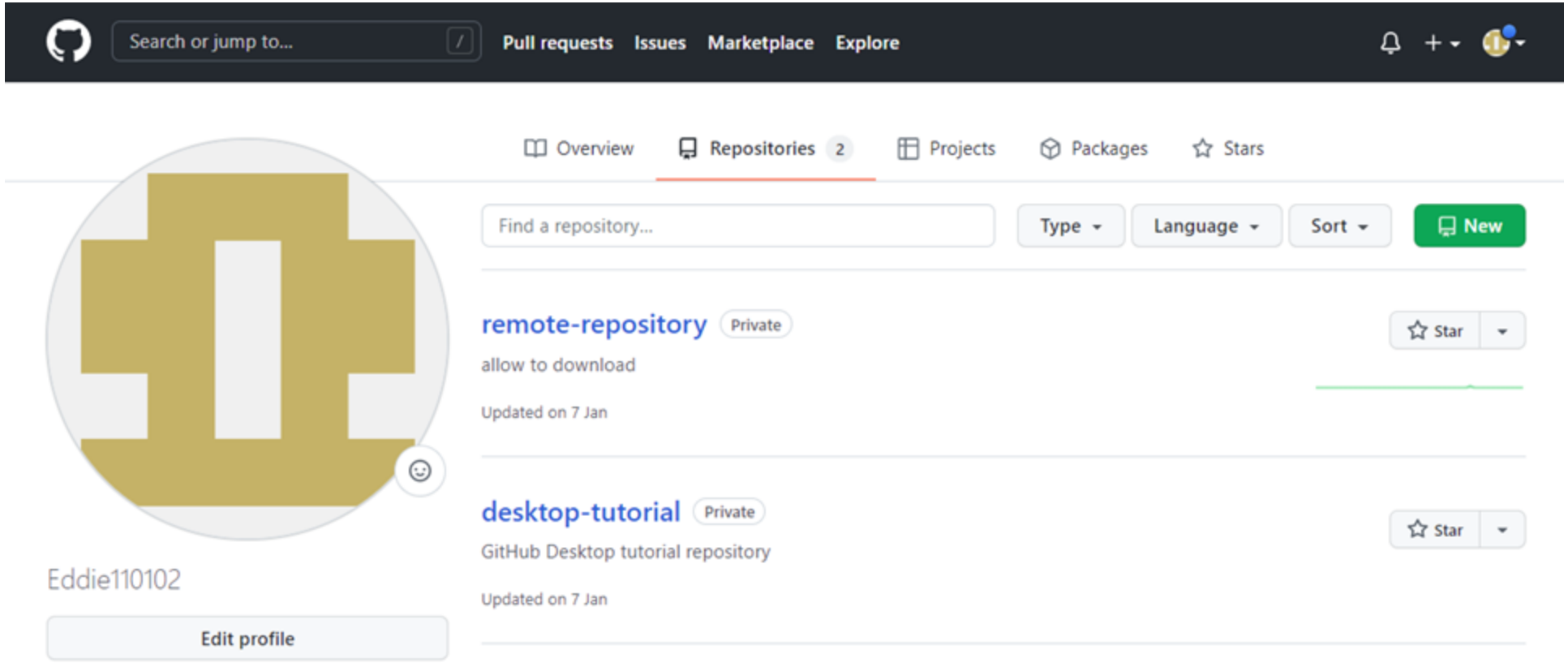
- Git是分散式版本控制系統，是一種工具。
- 優點及特色：
  - Git可以把檔案的狀態作為更新歷史記錄保存起來，因此可以把編輯過的檔案復原到以前的狀態。
  - 顯示編輯過內容的差異。
  - 當有開發者想將編輯過的舊檔案上傳到伺服器、覆蓋其他人的最新檔案時，系統會發出警告，因此可以避免在無意中覆蓋他人的編輯內容。

避免在無意中覆蓋他人的編輯內容



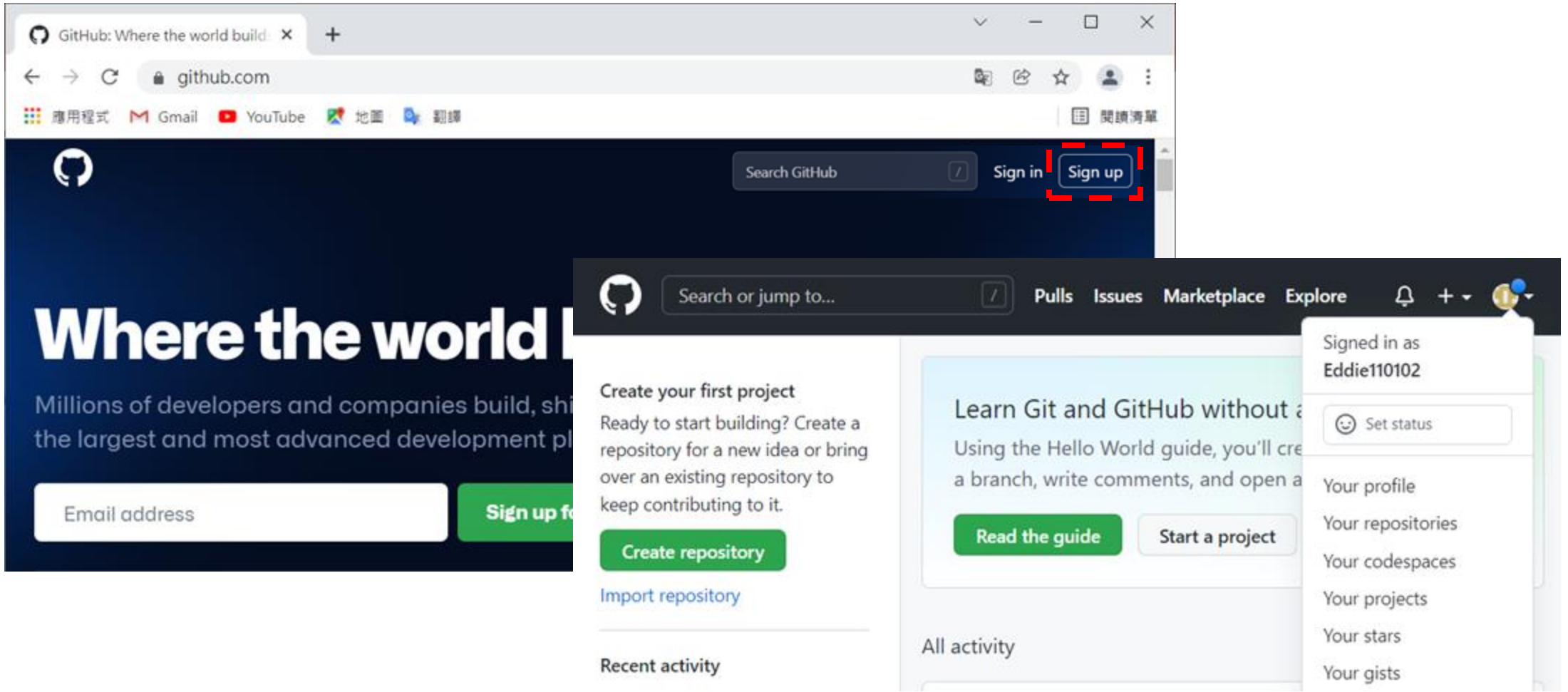
什麼是GitHub?

- GitHub 是一個 Git Server 網站。



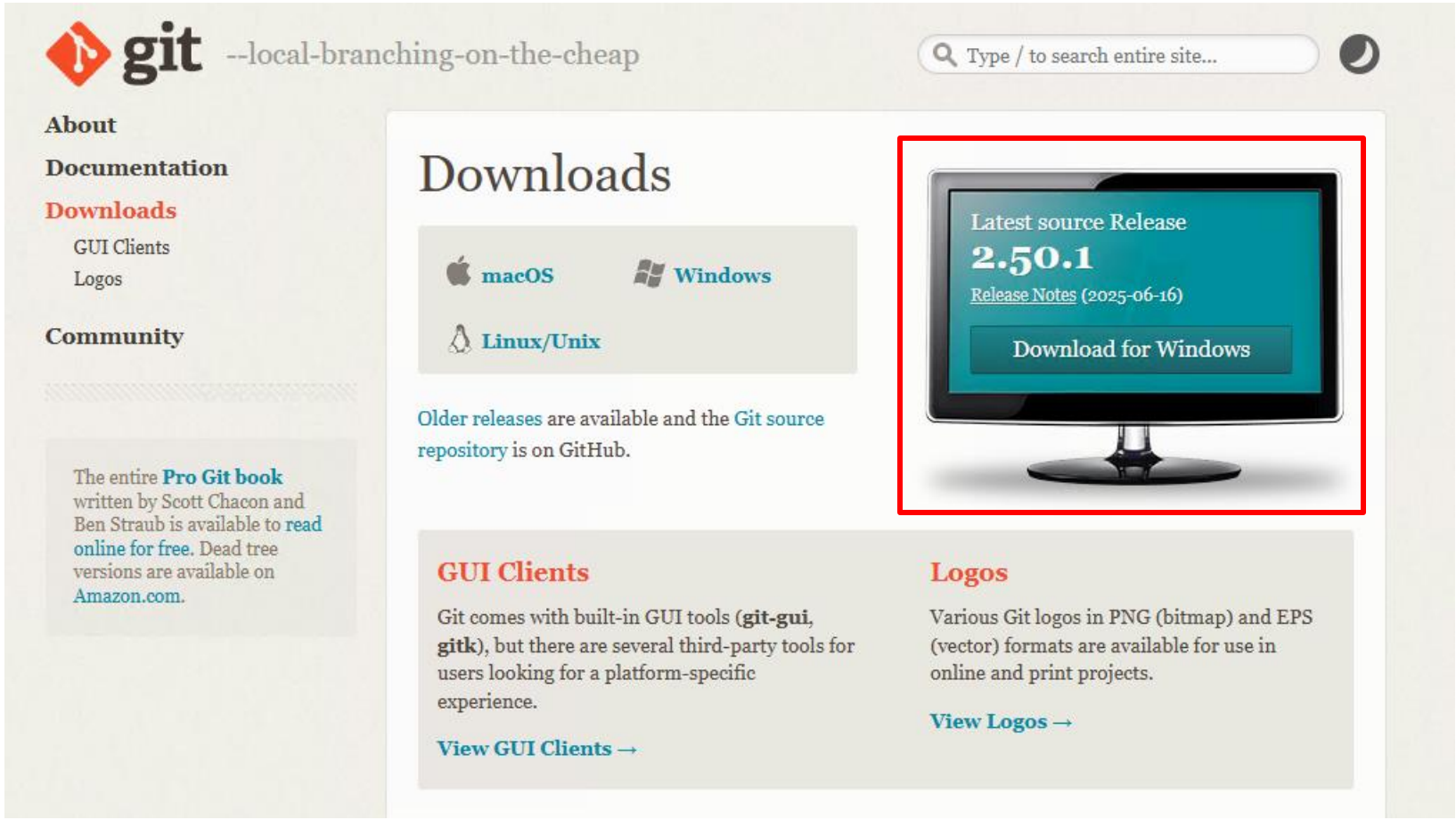
註冊GitHub

網址： https://github.com/



下載流程

1. 至 git官網 ，點擊 Download for Windows 。



下載流程

2. 選擇 Standalone Installer 版本。

Download for Windows

Click here to download the latest (**2.50.1**) **x64** version of **Git for Windows**. This is the most recent maintained build. It was released **14 days ago**, on 2025-07-08.

Other Git for Windows downloads

Standalone Installer 一般版程式

Git for Windows/x64 Setup.

Git for Windows/ARM64 Setup.

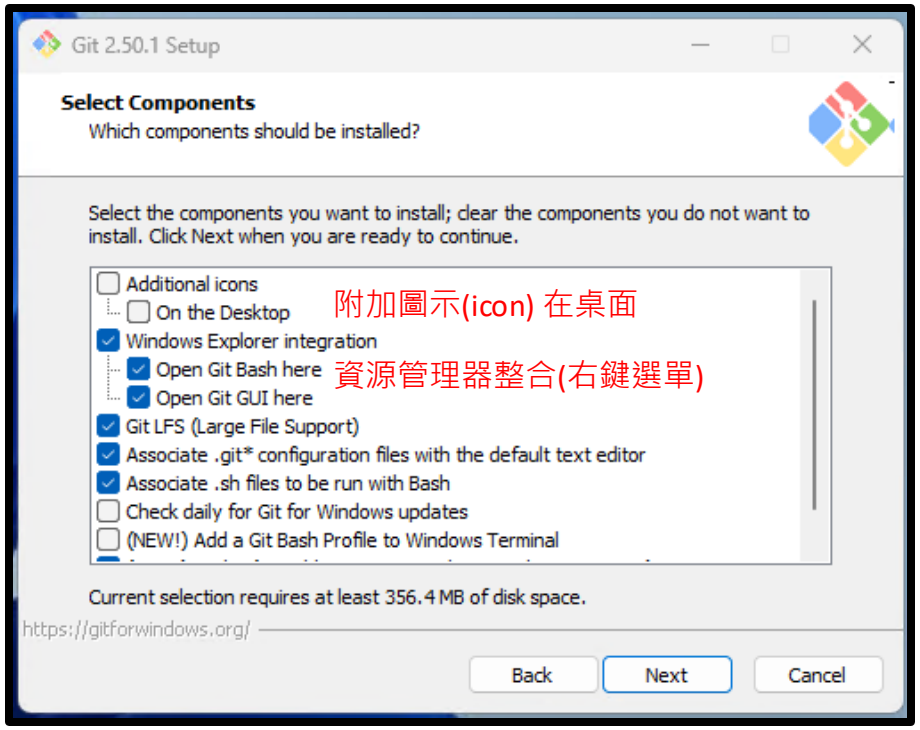
Portable ("thumbdrive edition") 可攜式程式

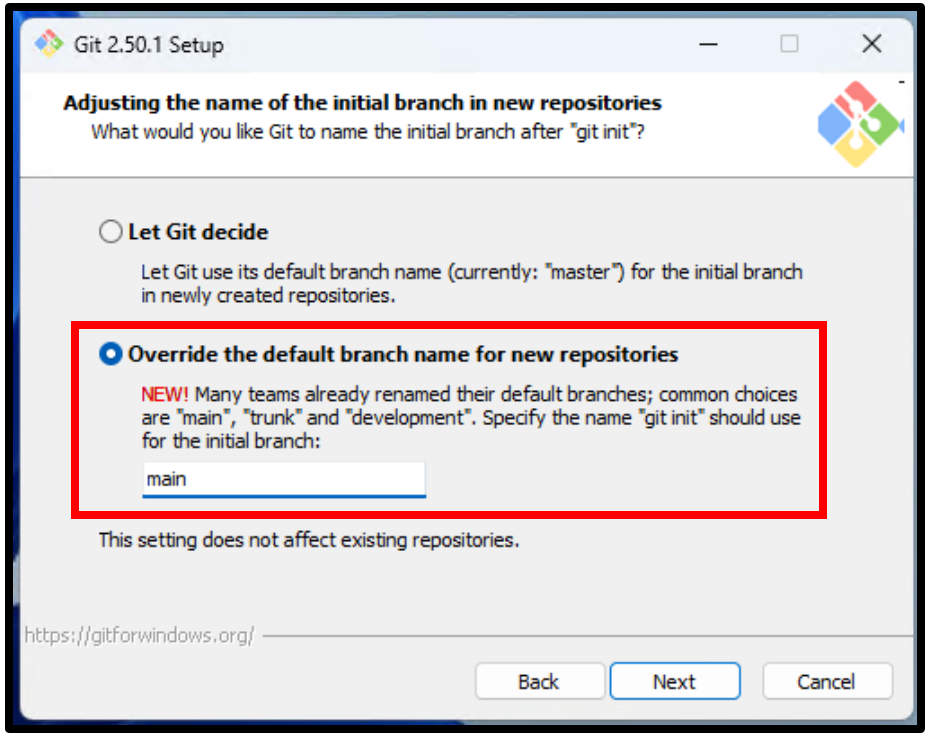
Git for Windows/x64 Portable.

Git for Windows/ARM64 Portable.

下載流程

3. 點擊安裝，選擇元件。
4. 設定Branch 名稱。（原為master）





下載流程 - mac版本（方法一）

1. 選擇 Homebrew 方式：

   開啟內建終端機，輸入「 /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)" 」 -> 安裝完成後，繼續輸入 「 brew install git 」

2. 檢查方式：

   在終端機輸入「 git --version 」確認 git 版本。

下載流程 - mac版本（方法二）

1. 選擇 Binary installer 方式：

   點擊 installer -> Download

下載流程 - mac版本（方法二）

2. 允許安裝不明來源 app：

執行安裝檔(.dmg) ->

至「系統偏好設定」->

「安全性與隱私權」->

強制打開 -> 即可安裝

文章參考來源：https://support.apple.com/zh-tw/HT202491

Linux 指令

Linux 指令

- 檢查 git 版本

```
git --version  ||  git -v
```

```
WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~/Desktop
$ git --version
git version 2.50.1.windows.1

WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~/Desktop
$ git -v
git version 2.50.1.windows.1
```

- 查看目前所在位置

```
pwd
```

```
WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~/Desktop
$ pwd
/c/Users/WDAGUtilityAccount/Desktop
```

Linux 指令

- 移動目前所在位置

```
cd <目的路徑>
```

「 . 」 ⇒ 現在的檔案夾位置
「 .. 」 ⇒ 回上一層檔案夾
「 ~ 」 ⇒ 目前使用者的目錄

```
WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~/Desktop
$ pwd
/c/Users/WDAGUtilityAccount/Desktop

WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~/Desktop
$ cd .

WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~/Desktop
$ pwd
/c/Users/WDAGUtilityAccount/Desktop

WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~/Desktop
$ cd ..

WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~
$ pwd
/c/Users/WDAGUtilityAccount

WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~
$ cd Desktop/

WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~/Desktop
$ pwd
/c/Users/WDAGUtilityAccount/Desktop
```

Linux 指令

- 建立新的資料夾

```
mkdir <資料夾名稱>
```

- 檢視目前資料夾中的檔案

```
ls
```

```
WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~/Desktop
$ mkdir git-lesson

WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~/Desktop
$ ls
desktop.ini  git-lesson/
```

Linux 指令

- 參數的意義，不同指令會不一樣，常用到指令的參數如下

  -a ⇒ all 全部

  -l ⇒ 檢視詳細資訊

  -f ⇒ force 強制

- 檢視目前資料夾中的檔案（包含隱藏檔 / 呈現詳細資訊）

```
ls -a
```

```
ls -al
```

```
WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~/Desktop
$ ls -a
./  ../  desktop.ini  git-lesson/

WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~/Desktop
$ ls -al
total 5
drwxr-xr-x 1 WDAGUtilityAccount 197121   0 Jul 22 18:00 ./
drwxr-xr-x 1 WDAGUtilityAccount 197121   0 Jul 22 18:00 ../
-rw-r--r-- 1 WDAGUtilityAccount 197121 282 Jul  9 07:15 desktop.ini
drwxr-xr-x 1 WDAGUtilityAccount 197121   0 Jul 22 18:00 git-lesson/
```

Linux 指令

MacOS / Linux 指令	說明
pwd	顯示目前所在路徑
cd <目的地路徑>	改變檔案目錄 change directory
mkdir <資料夾名稱>	建立新的資料夾 make directory
ls	列出目前資料夾中的檔案（不包含隱藏檔） list
touch <檔案名稱.副檔名>	建立檔案
cat <檔案名稱.副檔名>	檢視檔案內容
cp <被複製的檔案> <新檔名>	複製檔案 copy
mv </原本路徑/檔名> </新的路徑/檔名>	移動檔案 move
mv <舊檔名> <新檔名>	更改檔名
rm <檔案名稱.副檔名>	刪除檔案 remove
rm -r <檔案夾名稱>	刪除資料夾
clear	清空畫面
logout	關閉終端機指令

Git 初始設定

使用者設定

- 開啟終端機設定使用者資訊

```
git config --global user.name "使用者名稱"
git config --global user.email "電子郵件"
```

```
WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~/Desktop
$ git config --global user.name "Eddie110102"

WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~/Desktop
$ git config --global user.email "eddiewang@ispan.com.tw"
```

- 檢視目前的設定

```
git config --list
```

- 檢視單一設定

```
git config user.name
```

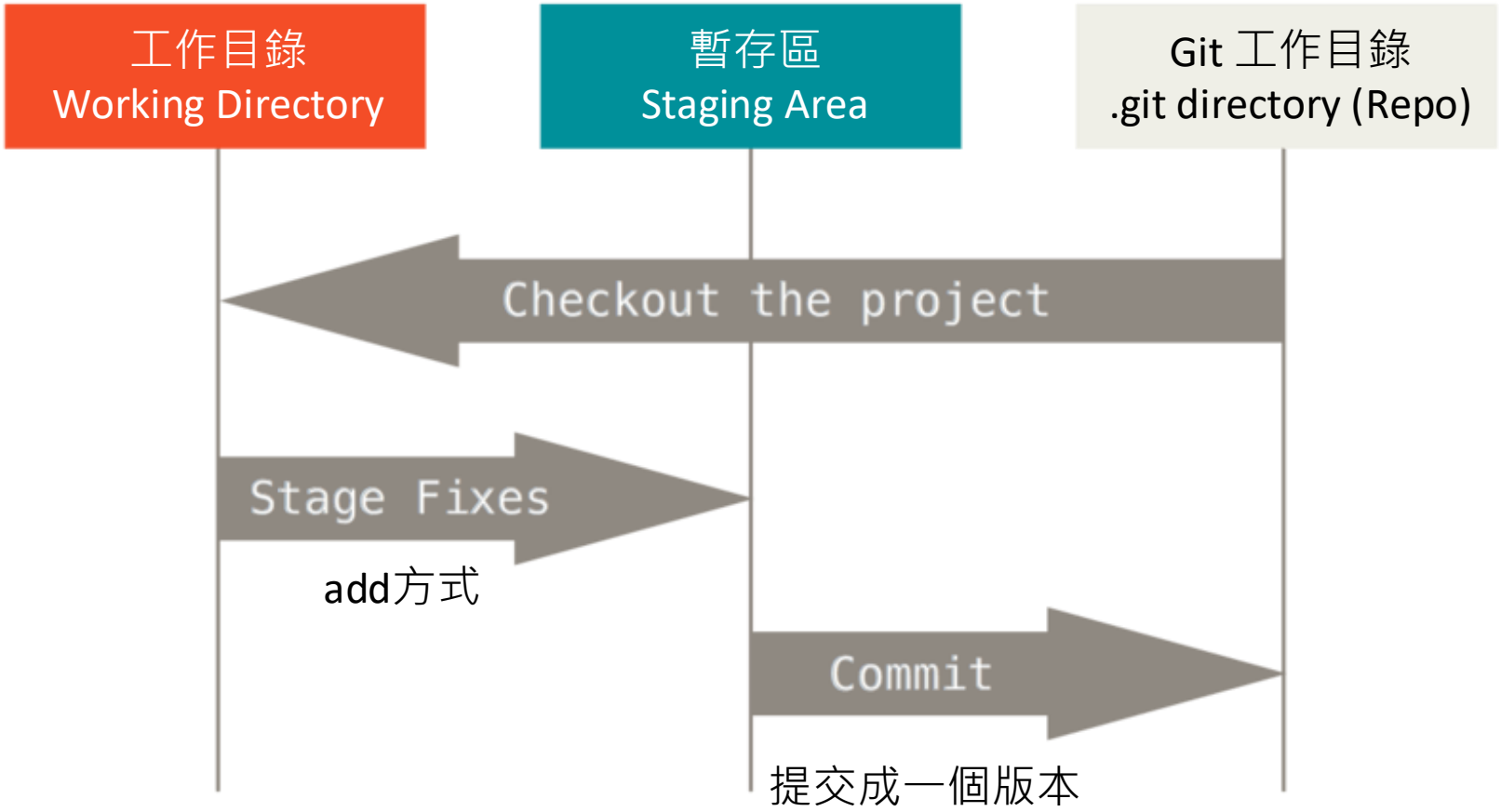
- 設定的資料儲存在
  - C:\Users\<使用者帳號>\.gitconfig

.gitconfig

C: > Users > WDAGUtilityAccount > .gitconfig

```
1   [user]
2       name = Eddie110102
3       email = eddiewang@ispan.com.tw
```

Git 主要工作區



基本的 Git 工作流程如下：

1. 在工作目錄中修改檔案。
2. 將你想要提交的檔案添加到暫存區。
3. 提交更新，將暫存區資料存儲到 Git 目錄。

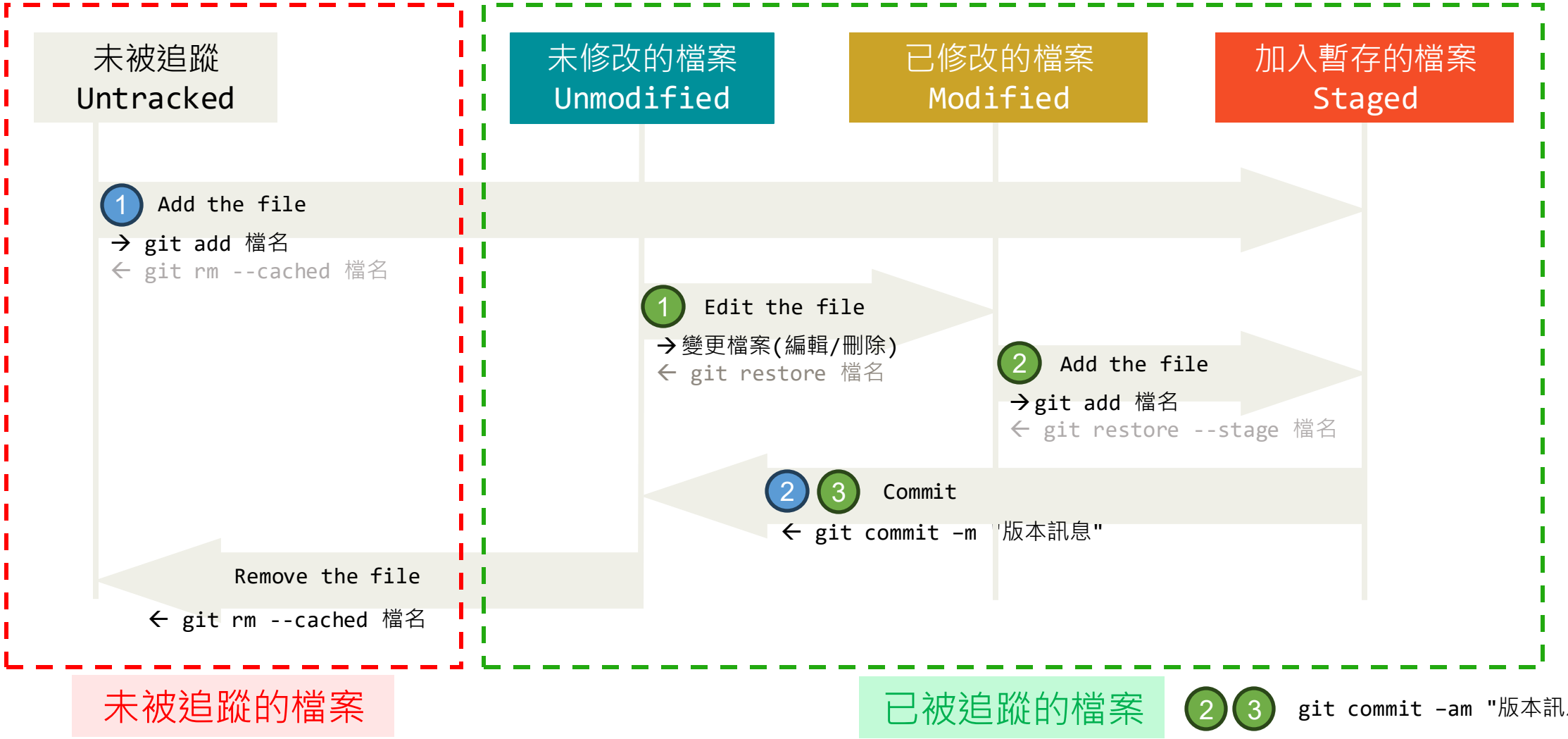
圖片來自：https://git-scm.com/book/en/v2/Getting-Started-Git-Basics

工作目錄(Working Directory)中的檔案

- 兩種基本狀態
  - tracked files：已經被Git控管(追蹤)的檔案，其中又可以分成三種狀態
    - Staged (Changes to be committed)：加到暫存區(staging area)的檔案
    - Modified (Changes not staged for commit)：Commit後，又修改的檔案
    - Unmodified：Commit後，還沒修改過的檔案 (不會顯示)
  - untracked files：還沒有被Git控管(追蹤)的檔案

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   page2.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   page1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        page3.txt
```

檔案狀態的生命週期



圖片來自：https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository

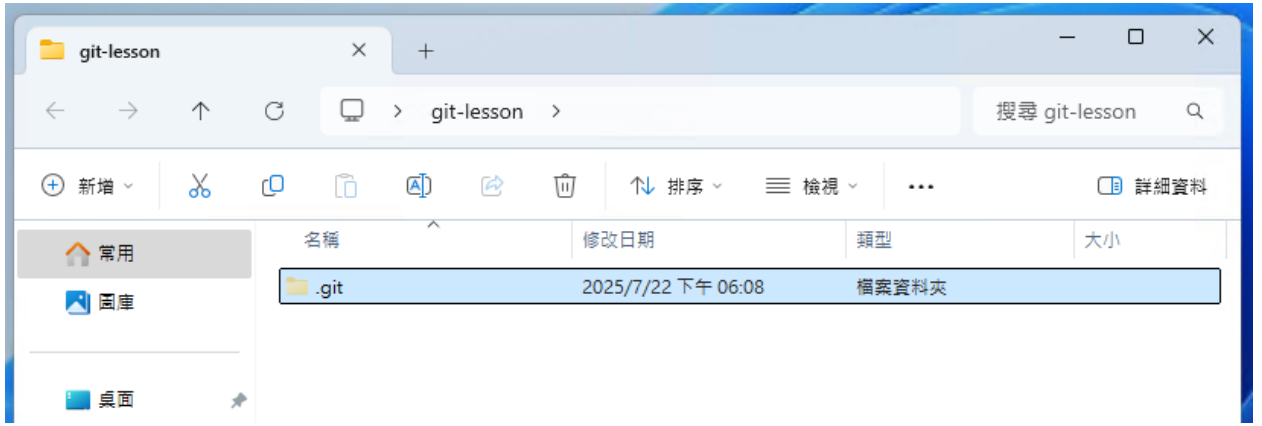
開始使用 – 建立 Git儲存庫(Repository)

git init

- 建立一個名為「 .git 」隱藏檔案夾
  - .git目錄中紀錄了儲存庫中的所有更動的紀錄

```
WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~/Desktop/git-lesson
$ git init
Initialized empty Git repository in C:/Users/WDAGUtilityAccount/Desktop/git-lesson/.git/
```

- 檢視隱藏的項目
  - 檔案總管 -> 檢視 -> 勾選「 隱藏的項目 」。



顯示檔案狀態 git status

- 建立一個檔案(page1.txt)到工作目錄中，然後顯示工作目錄中檔案的狀態

git status

```
WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~/Desktop/git-lesson (main)
$ git status
On branch main

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

沒有新的檔案可以被追蹤

```
WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~/Desktop/git-lesson (main)
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        page1.txt

nothing added to commit but untracked files present (use "git add" to track)
```

未被追蹤檔案

將檔案加入到暫存區(Staging Area)

untracked files 可以透過git add指令加到暫存區staging area，並讓Git追蹤

- 新增單一檔案

```
git add <檔案名稱.副檔名>
```

- 新增副檔名為txt的所有檔案

```
git add *.txt
```

- 新增所有檔案

```
git add –all   ||   git add *   ||   git add .
```

從暫存區(Staging Area)移除

git rm --cached <檔案名稱.副檔名>

```
WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~/Desktop/git-lesson (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   page1.txt
```

綠色為已加入追蹤 (暫存區) 的檔案

```
WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~/Desktop/git-lesson (main)
$ git rm --cached page1.txt
rm 'page1.txt'

WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~/Desktop/git-lesson (main)
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        page1.txt

nothing added to commit but untracked files present (use "git add" to track)
```

紅色為未被追蹤的檔案 / 已追蹤且有修改的檔案

從暫存區進入到Git工作目錄(Repository)

- 暫存區中的檔案，可以透過Commit指令，將檔案確認存到Repository

git commit -m " 本次新增/修改的描述(可以自行撰寫) "

```
WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~/Desktop/git-lesson (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   page1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        page2.txt
        page3.txt


WDAGUtilityAccount@65dcdca2-fcf1-4f00-8083-acf29650b99a MINGW64 ~/Desktop/git-lesson (main)
$ git commit -m " note what you do, both chinese or english are fine"
[main (root-commit) afd1789] note what you do, both chinese or english are fine
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 page1.txt
```

檢視commit紀錄

- 使用log指令，會由新到舊顯示版本紀錄

```
git log
```

- 單行顯示版本紀錄

```
git log --oneline
```

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git log
commit 1c77073031c19ec4fc55a09bd74ccdef4fa385d6 (HEAD -> main)
Author: Eddie110102 <eddiewang@ispan.com.tw>
Date:   Wed Jul 23 10:28:11 2025 +0800

    note what you done, both Chinese or english are fine.

WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git log --oneline
1c77073 (HEAD -> main)  note what you done, both Chinese or english are fine.
```

檢視所有版本歷史紀錄

- 列出所有的歷史紀錄（包含移動版本紀錄）

```
git reflog
```

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git reflog
146de76 (HEAD -> main) HEAD@{0}: reset: moving to 146de76
9bb732c HEAD@{1}: reset: moving to 9bb732c
9bb732c HEAD@{2}: reset: moving to 9bb732c
146de76 (HEAD -> main) HEAD@{3}: commit: add two sentences
9bb732c HEAD@{4}: commit: add one sentence
1c77073 HEAD@{5}: commit (initial): note what you done, both Chinese or english are fine.
```

多久要commit一次？什麼情境下該commit？

- 合理的 commit 時機：
  - ✔ **功能完成一小段**（例如完成一個按鈕事件、完成一個 API 呼叫）
  - ✔ **修正一個 bug**
  - ✔ **重構程式碼**（但不改功能）
  - ✔ **修改 UI 樣式或文案**
  - ✔ **加上註解或整理格式**（可單獨 commit）
  - ✔ **完成單一任務清單中的一項**

一個好的 commit 是「小而完整，清楚描述變更」，讓未來的自己或其他協作者可以輕鬆還原、理解與除錯。

- 不建議的情況：
  - ❌ 完成一大段功能才 commit（不好分辨）
  - ❌ 加了很多東西但只 commit 一次（不利追蹤錯誤）
  - ❌ 把測試中未完成的半成品 commit

修改檔案

- 修改已Commit過的檔案後，可以透過「 git status 」查詢檔案的狀態

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   page2.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   page1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        page3.txt
```

- 使用「 git add 」指令、「 git commit 」指令再存成一個版本紀錄。
- 這兩個動作可以合併成一次 「 git commit -am " 本次新增/修改的描述(可以自行撰寫) "

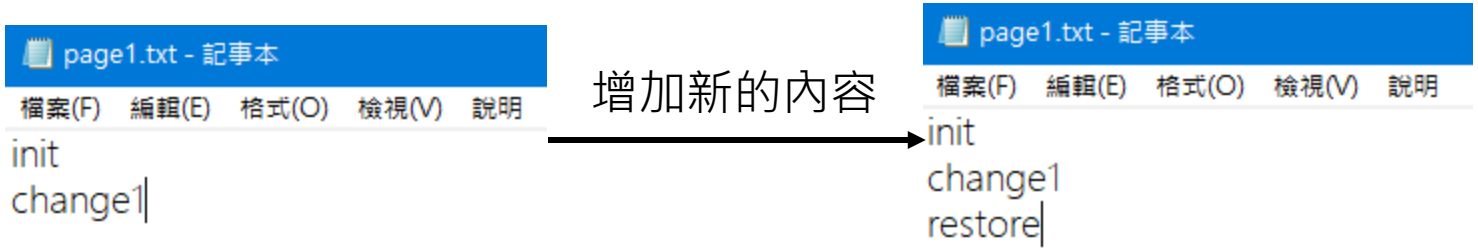
git commit -am " 本次新增/修改的描述(可以自行撰寫) "

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git commit -am " add one sentence "
[main 9bb732c]  add one sentence
 2 files changed, 1 insertion(+)
 create mode 100644 page2.txt
```

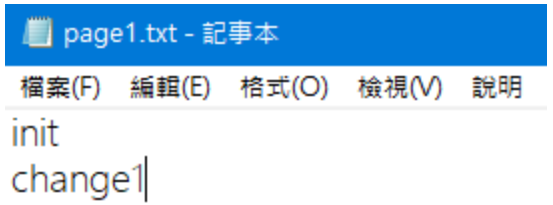
還原修改 git restore

- 檔案修改後，尚未加入到暫存區(Staging Area)，要還原成修改前的版本

```
git restore <檔案名稱.副檔名>
```

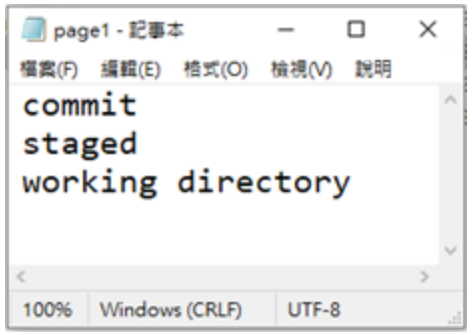



```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git restore page1.txt
```

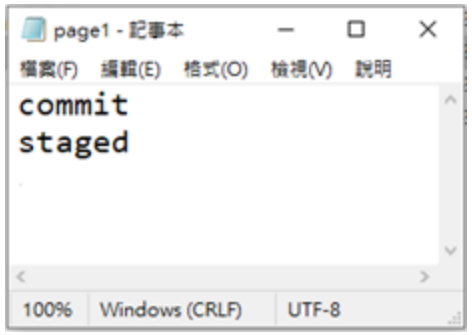


檔案差異比較 git diff – 情境1

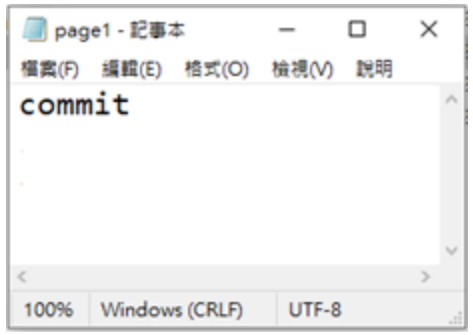
工作目錄
Working Directory



暫存區
Staging Area



Git 工作目錄
.git Directory (Repo)




```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   page1.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   page1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        page3.txt
```

檔案差異比較 git diff – 情境1

- 比對工作目錄(Working Directory)與暫存區(Staging Area)檔案之間的差異

```
git diff
```

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git diff
diff --git a/page1.txt b/page1.txt
index 204be94..10daf4f 100644
--- a/page1.txt
+++ b/page1.txt
@@ -1,2 +1,3 @@
第一次commit後又撰寫的文字。
-第二次commit後又增加的文字，這行文字已經加進暫存。
\ No newline at end of file
+第二次commit後又增加的文字，這行文字已經加進暫存。
+在檔案加進status後再進行編輯。
\ No newline at end of file
```

黑色文字表示相同內容
--- 表示舊版本檔案的內容
+++ 表示新版本檔案的內容
-1,2 表示舊的檔案中內容變更在1-2行
+1,3 表示新的檔案中內容變更在1-3行

- 將比較的結果匯出

```
git diff > page.diff
```

檔案差異比較 git diff – 情境2

- 比對工作目錄(Working Directory)與Git工作目錄(Repository)檔案之間的差異

```
git diff HEAD
```

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git diff HEAD
diff --git a/page1.txt b/page1.txt
index 765d762..10daf4f 100644
--- a/page1.txt
+++ b/page1.txt
@@ -1 +1,3 @@
-第一次commit後又撰寫的文字。
\ No newline at end of file
+第一次commit後又撰寫的文字。
+第二次commit後又增加的文字，這行文字已經加進暫存。
+在檔案加進status後再進行編輯。
\ No newline at end of file
```

檔案差異比較 git diff – 情境3

- 比對暫存區(Staging Area)與Git 工作目錄(Repository)檔案之間的差異

git diff --cached HEAD

git diff --staged HEAD

git diff --cached

git diff --staged

```
$ git diff --cached HEAD
diff --git a/page1.txt b/page1.txt
index 765d762..542b9ca 100644
--- a/page1.txt
+++ b/page1.txt
@@ -1 +1,2 @@
-第一次commit後又撰寫的文字。
\ No newline at end of file
+第一次commit後又撰寫的文字。
+第二次commit後又增加的文字，這行文字已經加進暫存。
```

檔案差異比較 git diff – 情境4

- Git 工作目錄(Repository)中任意兩個版本 (7碼) 檔案之間的差異

```
git diff <commit1> <commit2> <檔案名稱.副檔名>
```

○ - 為commit1版本中的資料，+ 為commit2版本中的資料

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git diff 1c77073 9bb732c page1.txt
diff --git a/page1.txt b/page1.txt
index e69de29..765d762 100644
--- a/page1.txt
+++ b/page1.txt
@@ -0,0 +1 @@
+第一次commit後又撰寫的文字。
\ No newline at end of file
```

切換到之前的版本 - 情境1

- 保留工作目錄(Working Directory)中的內容，Git 工作目錄(Repository)回到指定的版本
- 指定版本的幾種寫法

```
git reset HEAD^^
```

```
git reset HEAD~2
```

```
git reset <commit>
```

切換到之前的版本 - 情境2

- 工作目錄(Working Directory)，Git 工作目錄(Repository)的內容都回到指定的版本

```
git reset --hard HEAD~~
```

切換到之前的版本 - 情境3

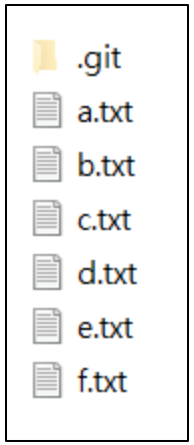
- 將單一檔案切回到某一個版本的內容

```
git checkout <commit> <檔案名稱.副檔名>
```

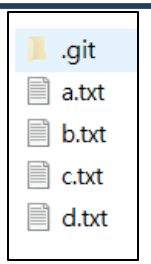
```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git log --oneline
12e456e (HEAD -> main) add one sentence
146de76  add two sentences
9bb732c  add one sentence
1c77073  note what you done, both Chinese or english are fine.

WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git checkout 146de763 page2.txt
Updated 1 path from 7aaa61c
```

Reset比較

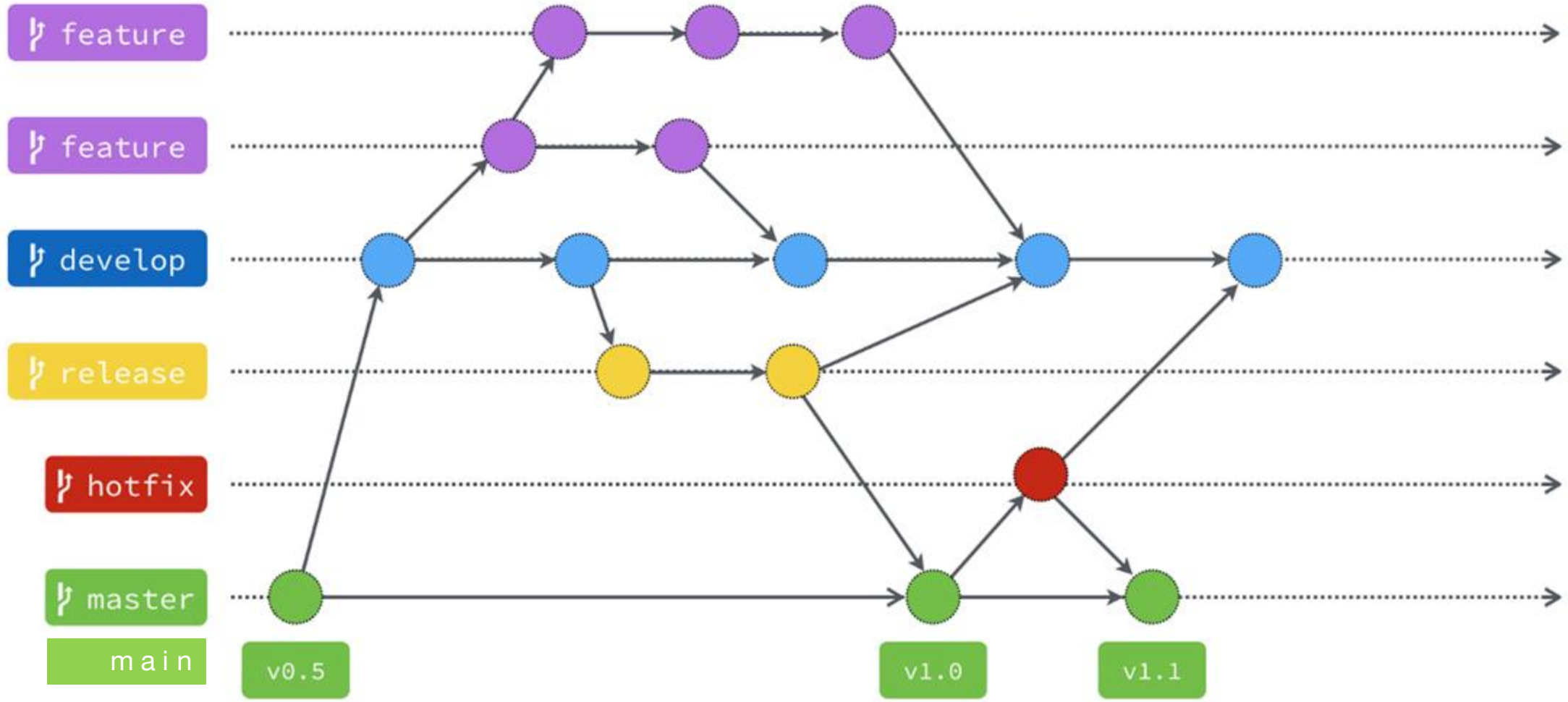
```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git log --oneline
61d5a7a (HEAD -> main) add files ( e.txt & f.txt )
a060f0a add files ( c.txt & d.txt )
8736cf2 add files ( a.txt & b.txt )
```

git reset a060f0a	WDAGUtilityAccount@dc3ef9c
	$ git reset --soft a060f0a
	
	WDAGUtilityAccount@dc3ef9c
	$ git status
	On branch main
	Changes to be committed:
	(use "git restore --stag
	new file: e.txt
	new file: f.txt
		

git reset --soft a060f0a	WDAGUtilityAccount@dc3ef9c
	$ git reset a060f0a
	
	WDAGUtilityAccount@dc3ef9c
	$ git status
	On branch main
	Untracked files:
	(use "git add <file>..."
	e.txt
	f.txt	

git reset --hard a060f0a	WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 $ git reset --hard a060f0a HEAD is now at a060f0a add files (c.txt & d.txt)	

Git Flow



資料來源：https://gitbook.tw/chapters/gitflow/why-need-git-flow

branch 分支

- 將程式開發從開發主線上分離開
  - 不同的版本就使用不同的分支，ex：1.0、1.0.1、1.1、2.0、....
  - 不同的軟體版本週期使用不同的分支，ex：Alpha、Beta、RC、RTM、.....
  - 單一功能的開發使用不同的分支
  - 不同的開發人員使用不同的分支
  - 為了修復問題也可以使用不同的分支
- 預設的分支名稱：以前是master，現在是main
- 透過HEAD指標，指定目前使用中的branch

新增分支

- 查詢目前專案有哪些分支

```
git branch
```

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git branch
* main
```

- 新增分支

```
git branch <branch分支名稱>
```

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git branch develop

WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git branch
  develop
* main
```

切換分支

- 切換分支

```
git switch <branch分支名稱>
```

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git switch develop
Switched to branch 'develop'

WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (develop)
$
```

- 同時建立及切換分支

```
git switch -c <branch分支名稱>
```

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (develop)
$ git switch -c hotfix
Switched to a new branch 'hotfix'

WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (hotfix)
$ git branch
  develop
* hotfix
  main
```

修改分支名稱

git branch –m <舊的分支名稱> <新的分支名稱>

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (hotfix)
$ git branch
  develop
* hotfix
  main

WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (hotfix)
$ git branch -m hotfix eddie

WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (eddie)
$ git branch
  develop
* eddie
  main
```

刪除分支

- 欲刪除正在使用的分支，要先切換到其他分支

```
git branch –d(-D) <分支名稱>
```

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (eddie)
$ git branch
  develop
* eddie
  main

WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (eddie)
$ git branch -d develop
Deleted branch develop (was 12e456e).

WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (eddie)
$ git branch
* eddie
  main
```

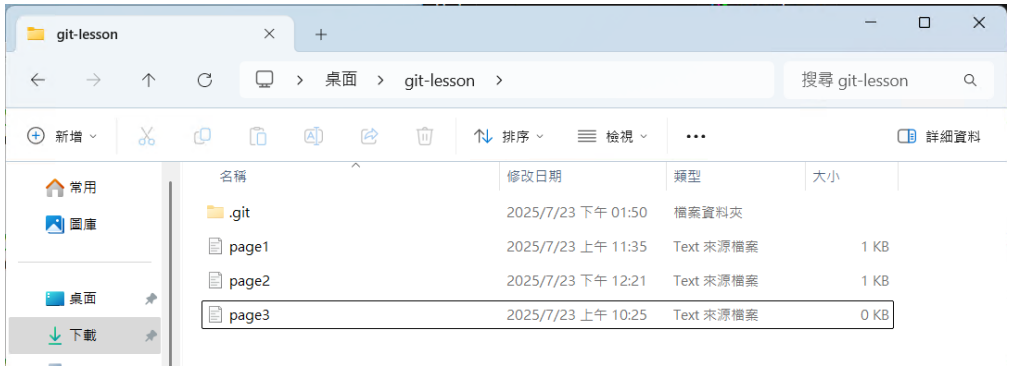
-d：會先檢查檔案是否有合併到其他分支，已合併的分支可安全刪除
-D：不檢查檔案，確定不要再保留該分支，強制刪除

合併分支

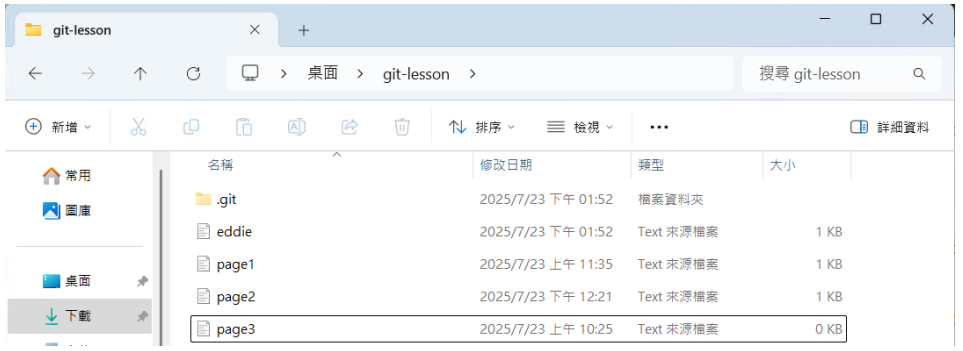
- 如果要將main分支合併其它分支，就要先切換到main分支

```
git merge <分支名稱> -m "message"
```

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (eddie)
$ git switch main
Switched to branch 'main'

WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git merge eddie -m " eddie.txt merge into main "
Updating 12e456e..5bb3daf
Fast-forward (no commit created; -m option ignored)
 eddie.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 eddie.txt
```



main 分支

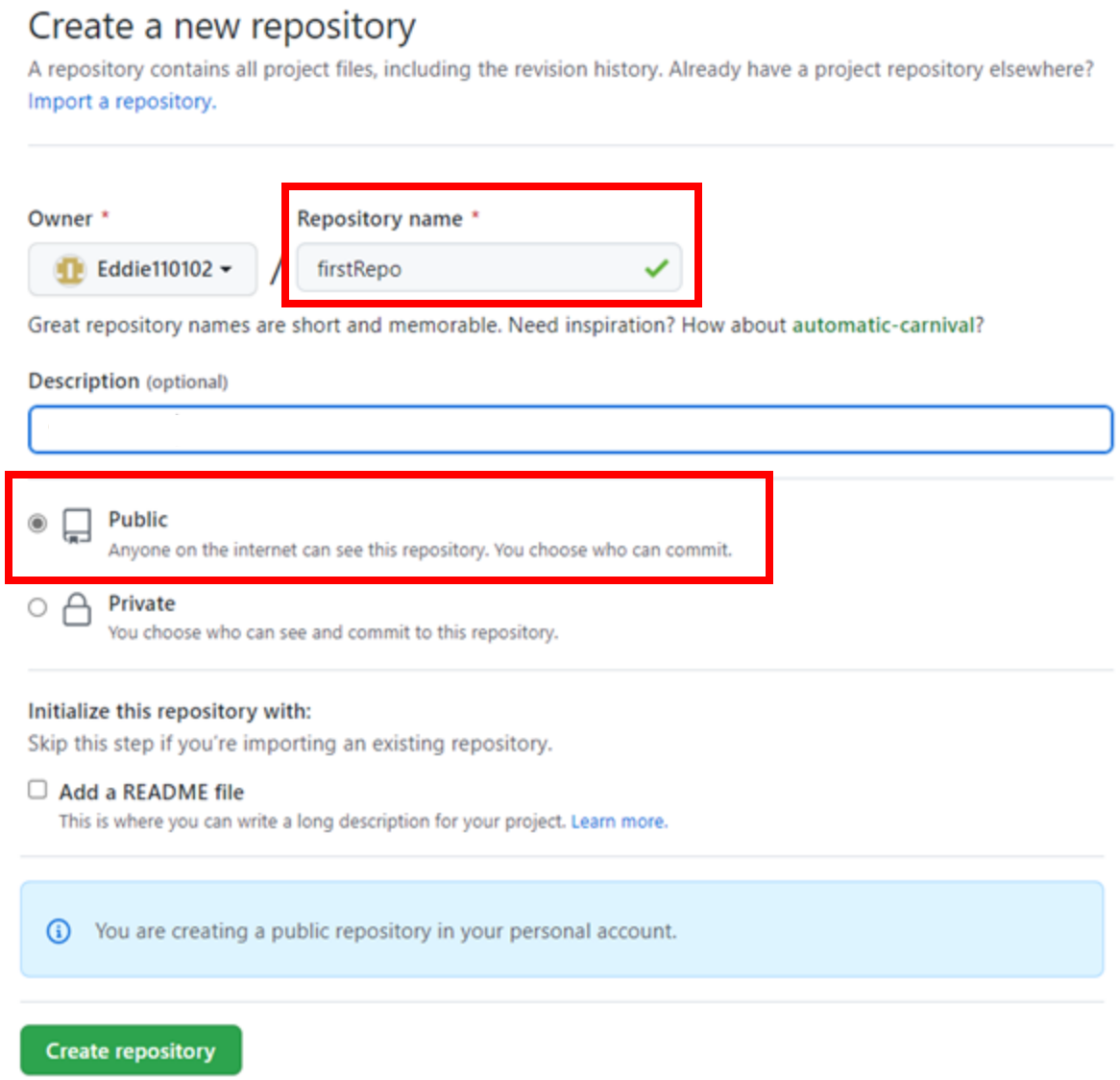


eddie 分支

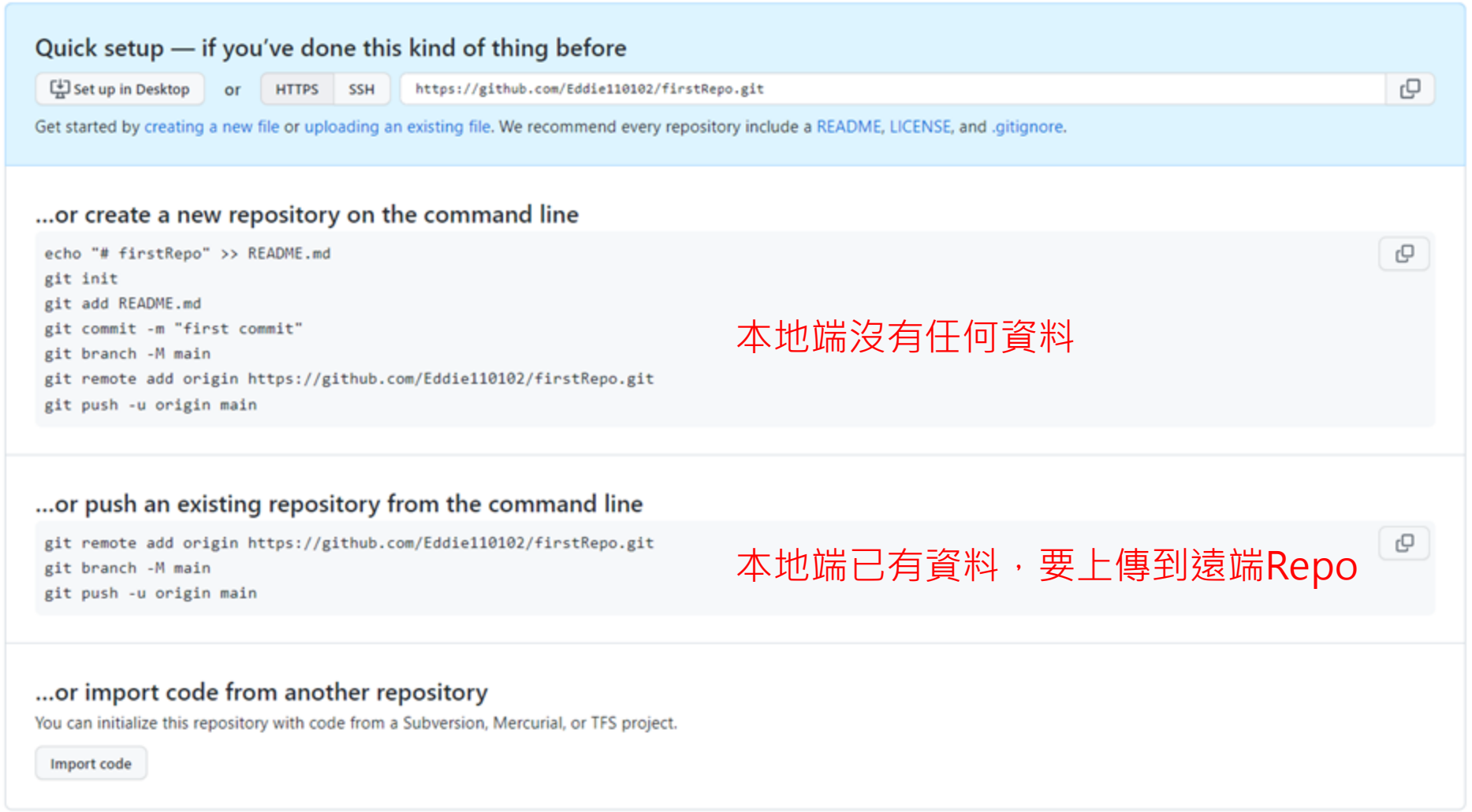
建立遠端資料庫

- Git Repository
  - Local Repository：本地端的Git工作目錄
  - Remote Repository：讓開發團隊成員分享各自的local repository資料而建立
- 可以自行建置 Git Server或使用坊間的Git Repository託管服務
  - Git Server：GitHub、GitLab、Bitbucket、Gitorious

建立新專案


Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.
Owner *
Eddie110102
Repository name *
firstRepo
Great repository names are short and memorable. Need inspiration? How about automatic-carnival?
Description (optional)
Public
Anyone on the internet can see this repository. You choose who can commit.
Private
You choose who can see and commit to this repository.
Initialize this repository with:
Skip this step if you're importing an existing repository.
Add a README file
This is where you can write a long description for your project. Learn more.
You are creating a public repository in your personal account.
Create repository

完整的使用方式說明


Quick setup — if you've done this kind of thing before
Set up in Desktop
or
HTTPS
SSH
https://github.com/Eddie110102/firstRepo.git
Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.
...or create a new repository on the command line
echo "# firstRepo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/Eddie110102/firstRepo.git
git push -u origin main
本地端沒有任何資料
...or push an existing repository from the command line
git remote add origin https://github.com/Eddie110102/firstRepo.git
git branch -M main
git push -u origin main
本地端已有資料，要上傳到遠端Repo
...or import code from another repository
You can initialize this repository with code from a Subversion, Mercurial, or TFS project.
Import code

設定remote repository

- 在專案資料夾中設定remote repository

```
git remote add <自取的名稱> <remote repository的URL>
```

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git remote add origin https://github.com/Eddie110102/firstRepo.git
```

  - remote repository的URL格式
    - https://github.com/<GitHub帳號>/<GitHub上的Repository名稱>.git

- 檢視所有remote repository的設定

```
git remote -v
```

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git remote -v
origin  https://github.com/Eddie110102/firstRepo.git (fetch)
origin  https://github.com/Eddie110102/firstRepo.git (push)
```

設定remote repository

- 檢視某個remote repository的設定

git remote show <remote名稱>

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git remote show origin
* remote origin
  Fetch URL: https://github.com/Eddie110102/firstRepo.git
  Push  URL: https://github.com/Eddie110102/firstRepo.git
  HEAD branch: (unknown)
```

修改remote repository名稱

- 檢視所有remote repository的設定

```
git remote
```

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git remote
origin
```

- 改變remote repository設定的名稱

```
git remote rename <舊的名稱> <新的名稱>
```

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git remote rename origin eddie
Renaming remote references: 100% (3/3), done.

WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git remote
eddie
```

刪除 remote repository

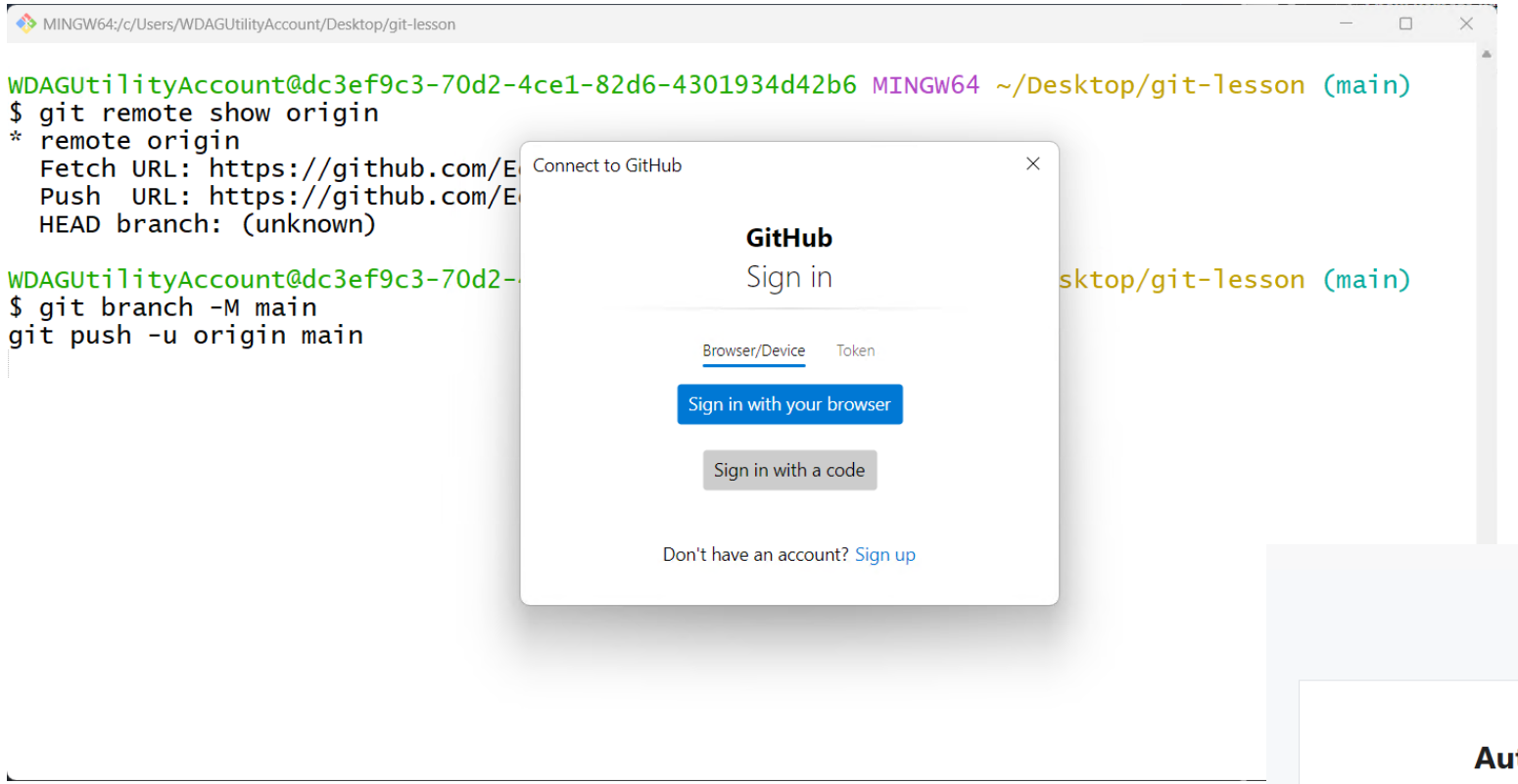
- 刪除remote repository的設定

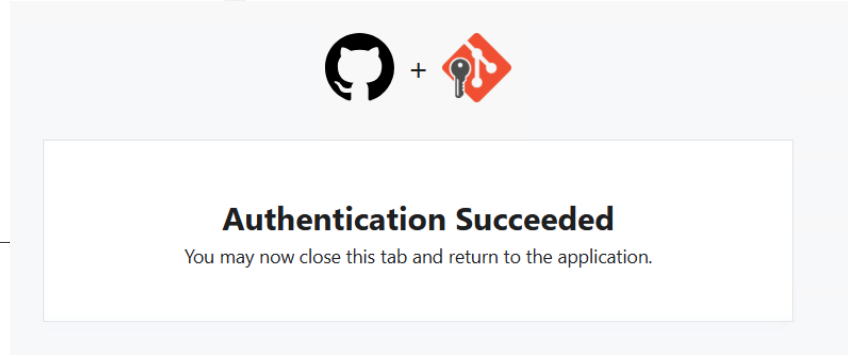
git remote remove <設定的名稱>

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git remote
eddie

WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git remote remove eddie

WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git remote

WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git remote add origin https://github.com/Eddie110102/firstRepo.git

WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git remote
origin
```

上傳資料（首次需要登入）

- 如未串聯GitHub情況下，首次會需要登入，可以使用瀏覽器/Token。





上傳資料

- local repository的資料**上傳**到remote repository

```
git push <設定的名稱> <branch的名稱>
```

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git push -u origin main
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 8 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (15/15), 1.45 KiB | 296.00 KiB/s, done.
Total 15 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Eddie110102/firstRepo.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

- -u (--set-upstream 的意思)，之後使用push及pull都會自動連結這個分支

```
git push -u <設定的名稱> <branch的名稱>
```

下載資料

- remote repository的資料**下載**到local repository

```
git pull <設定的名稱> <branch的名稱>
```

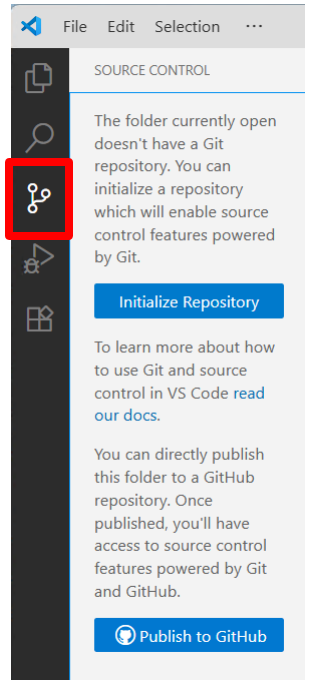
如果在push時有先加上參數 –u，這邊就可以省略後面的設定。

```
WDAGUtilityAccount@dc3ef9c3-70d2-4ce1-82d6-4301934d42b6 MINGW64 ~/Desktop/git-lesson (main)
$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 1.05 KiB | 154.00 KiB/s, done.
From https://github.com/Eddie110102/firstRepo
   5bb3daf..7b1e677  main       -> origin/main
Updating 5bb3daf..7b1e677
Fast-forward
 README.md | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 README.md
```

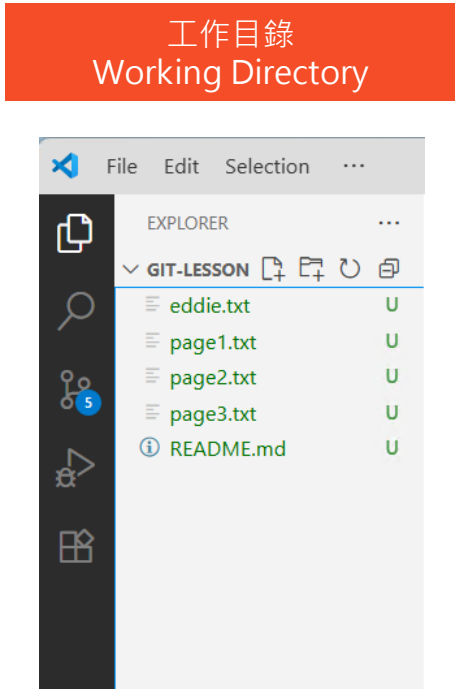
VS Code 工具

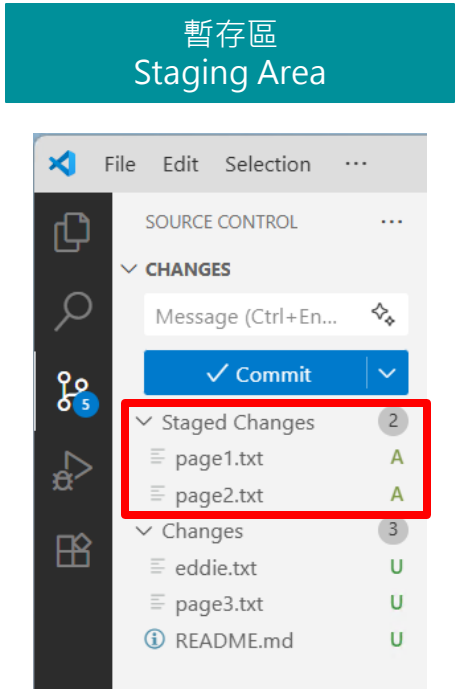
Visual Studio Code + GitHub

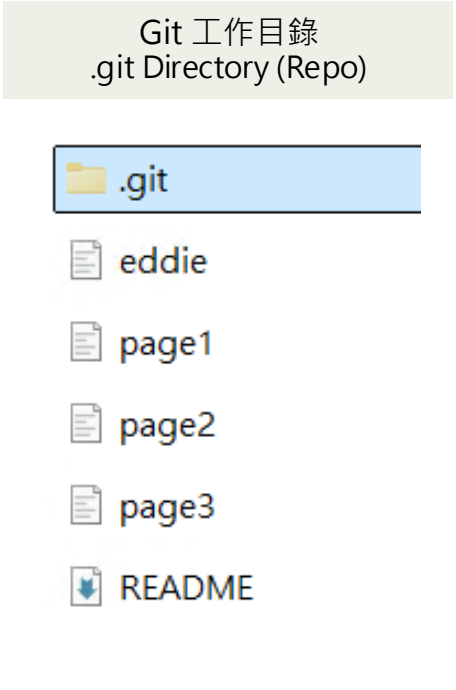
- git init：開啟專案資料夾 -> 原始檔控制 -> 將存放庫初始化
- git status：使用GUI介面



git init



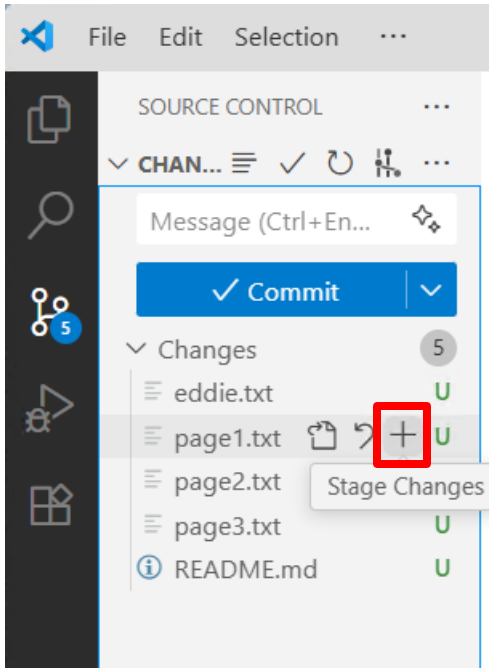




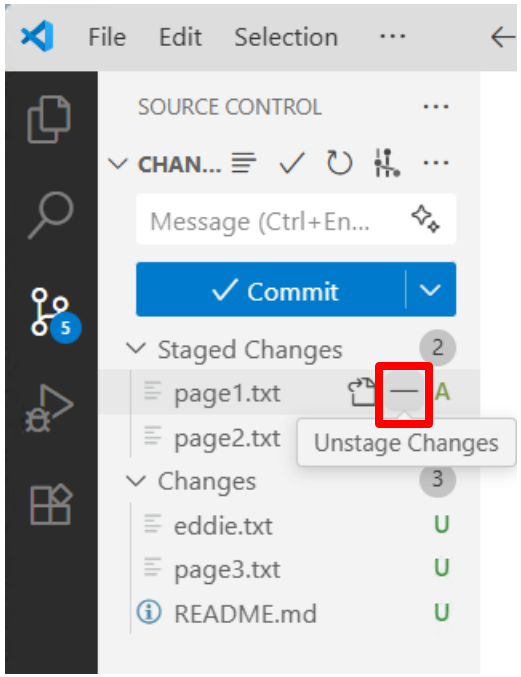
git status

Visual Studio Code + GitHub

- git add ... 檔案加入 / git rm --cached ... 移出暫存區



將檔案加入暫存區

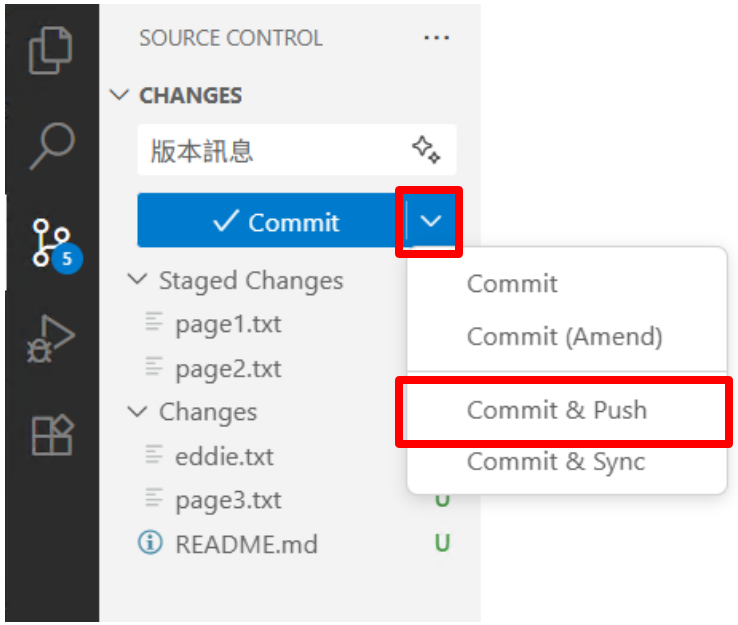


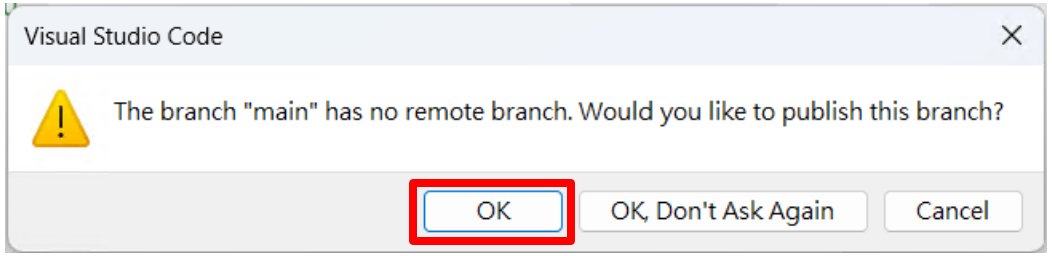
將檔案移出暫存區

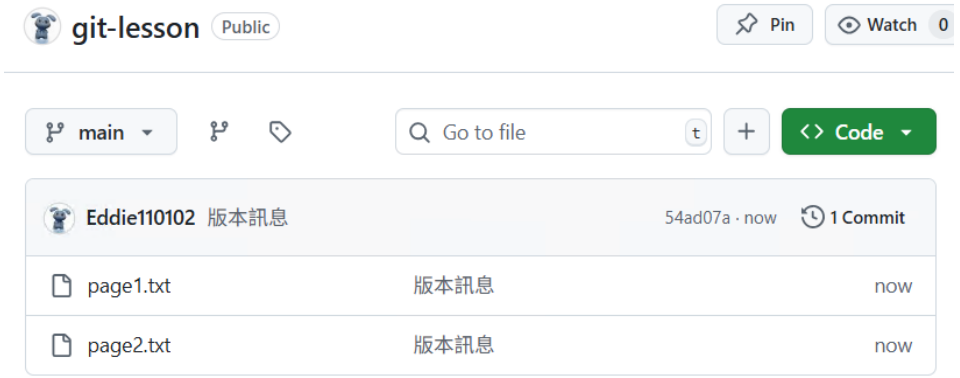
U：Untracked
A：Added
M：Modified

Visual Studio Code + GitHub

- git commit & git push：Commit & Push -> 建立分支 -> 確定

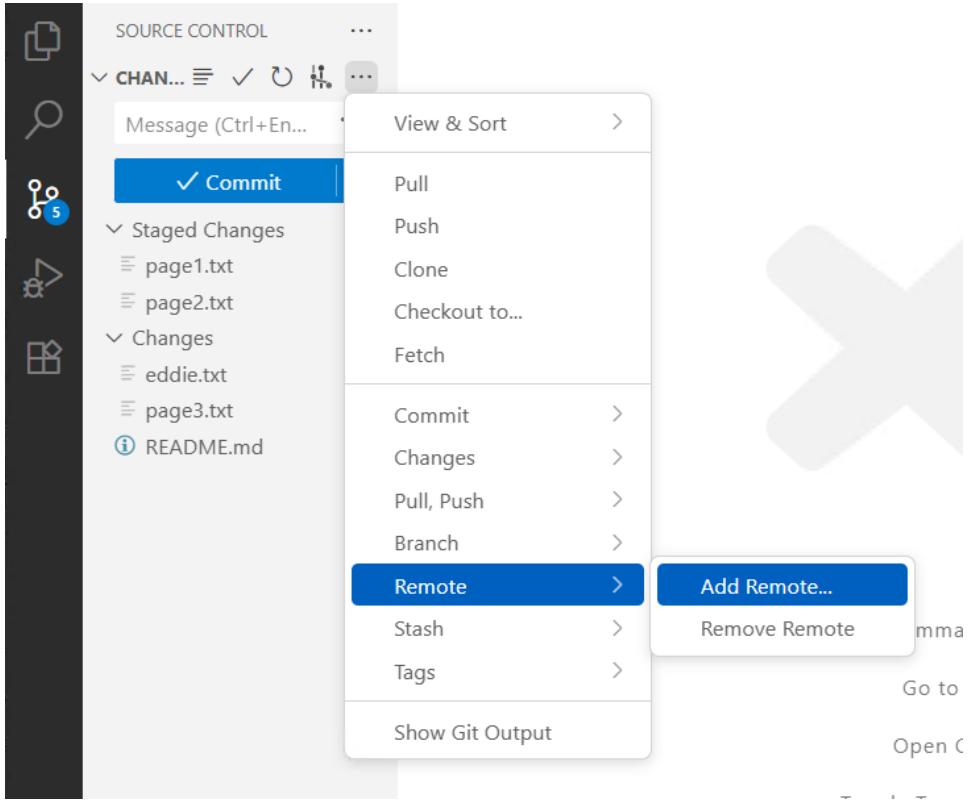


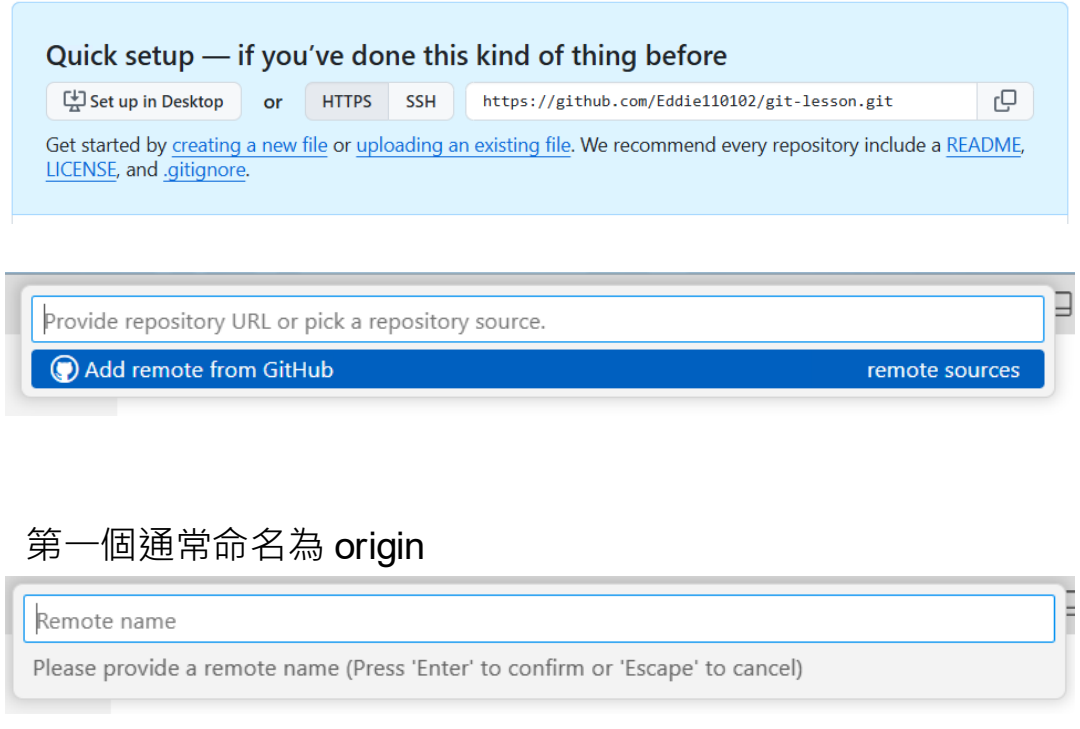




Visual Studio Code + GitHub

- git remote add：在GitHub中建立資料庫 -> 複製url -> 將資料庫命名

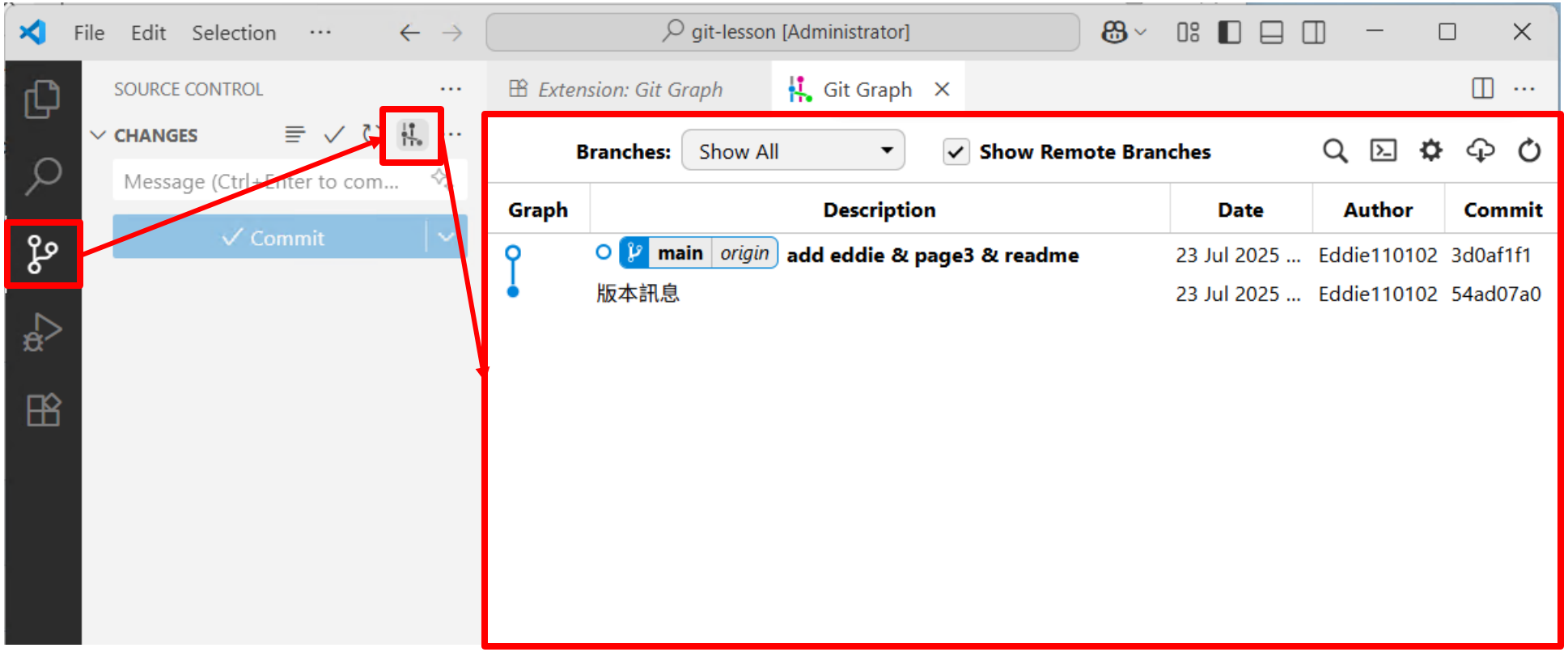




第一個通常命名為 origin

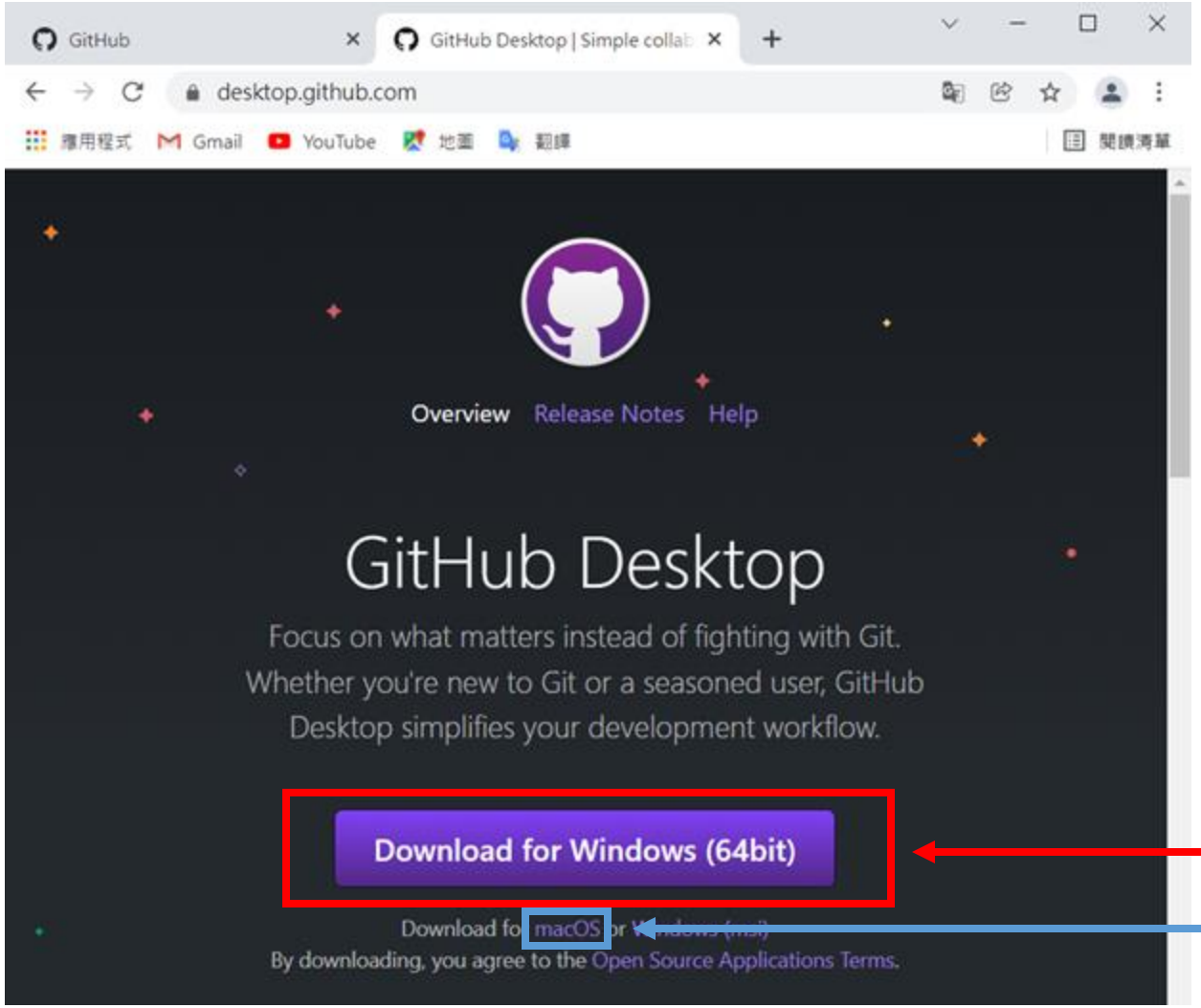
Git Graph 套件

- Git Graph 可以圖形化檢視版本變化及演進。



GitHub Desktop工具

下載 desktop版本

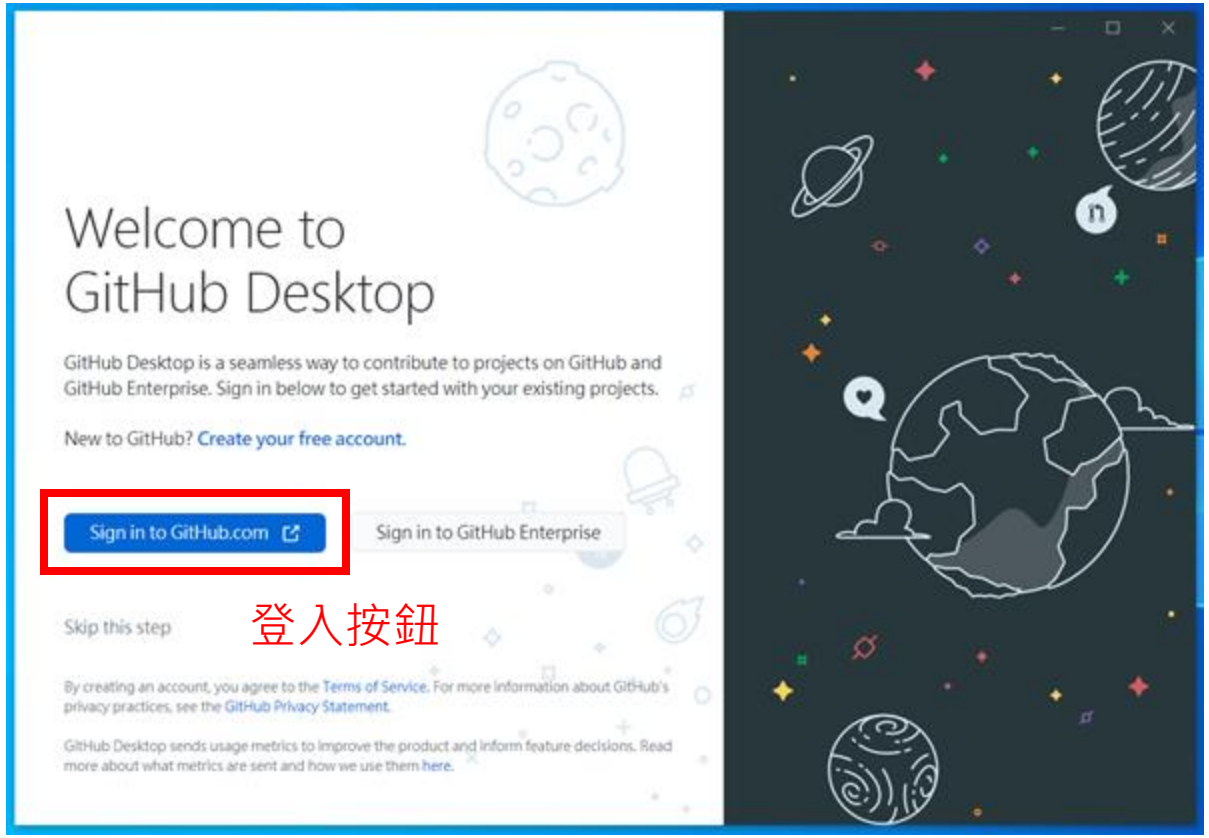


Windows 64位元 下載按鈕

macOS 下載按鈕

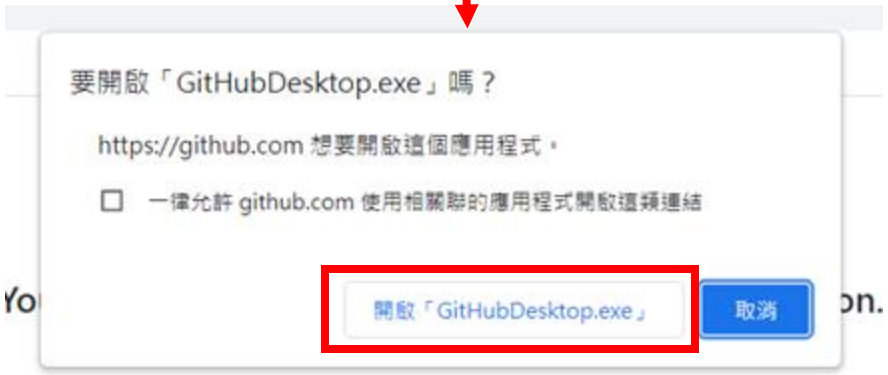
網址： https://desktop.github.com/

登入及授權 desktop版本

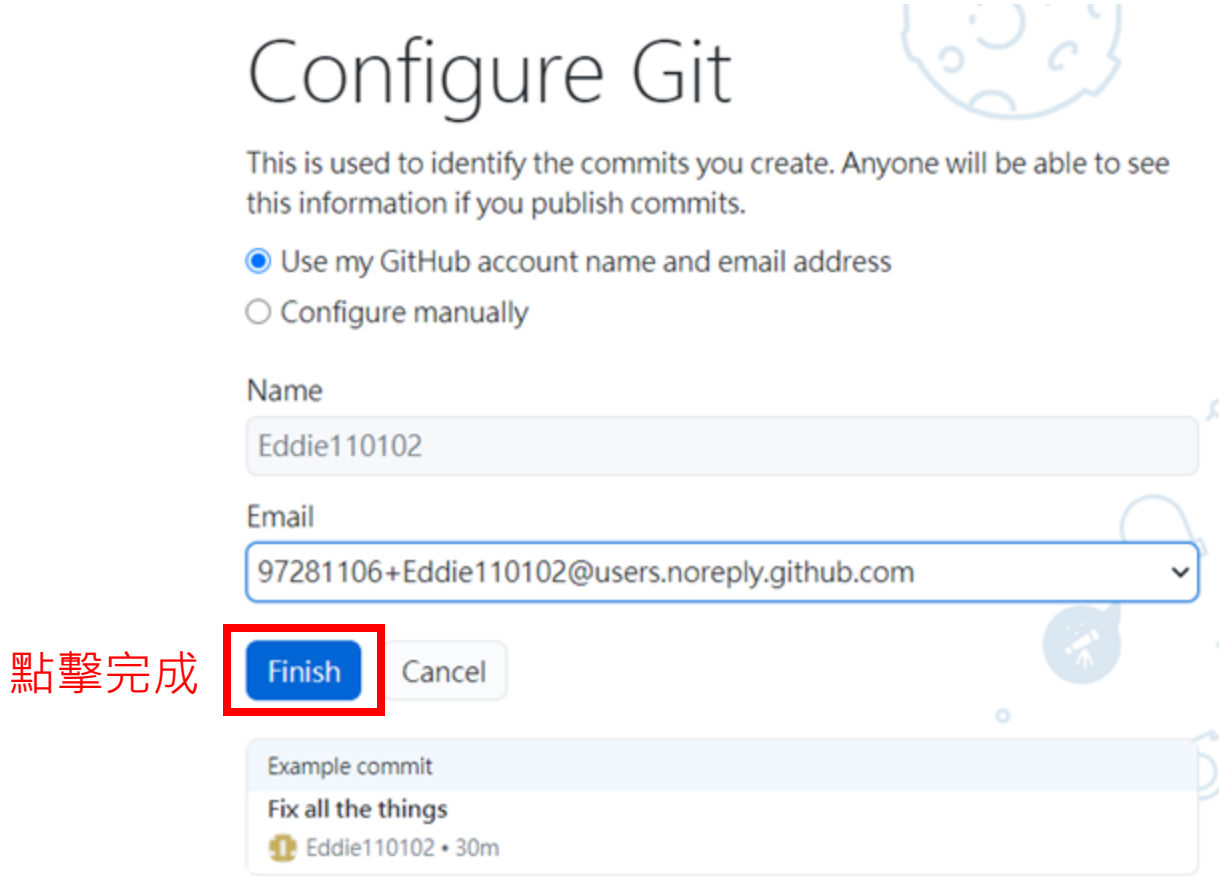


登入按鈕

授權GitHub桌面版

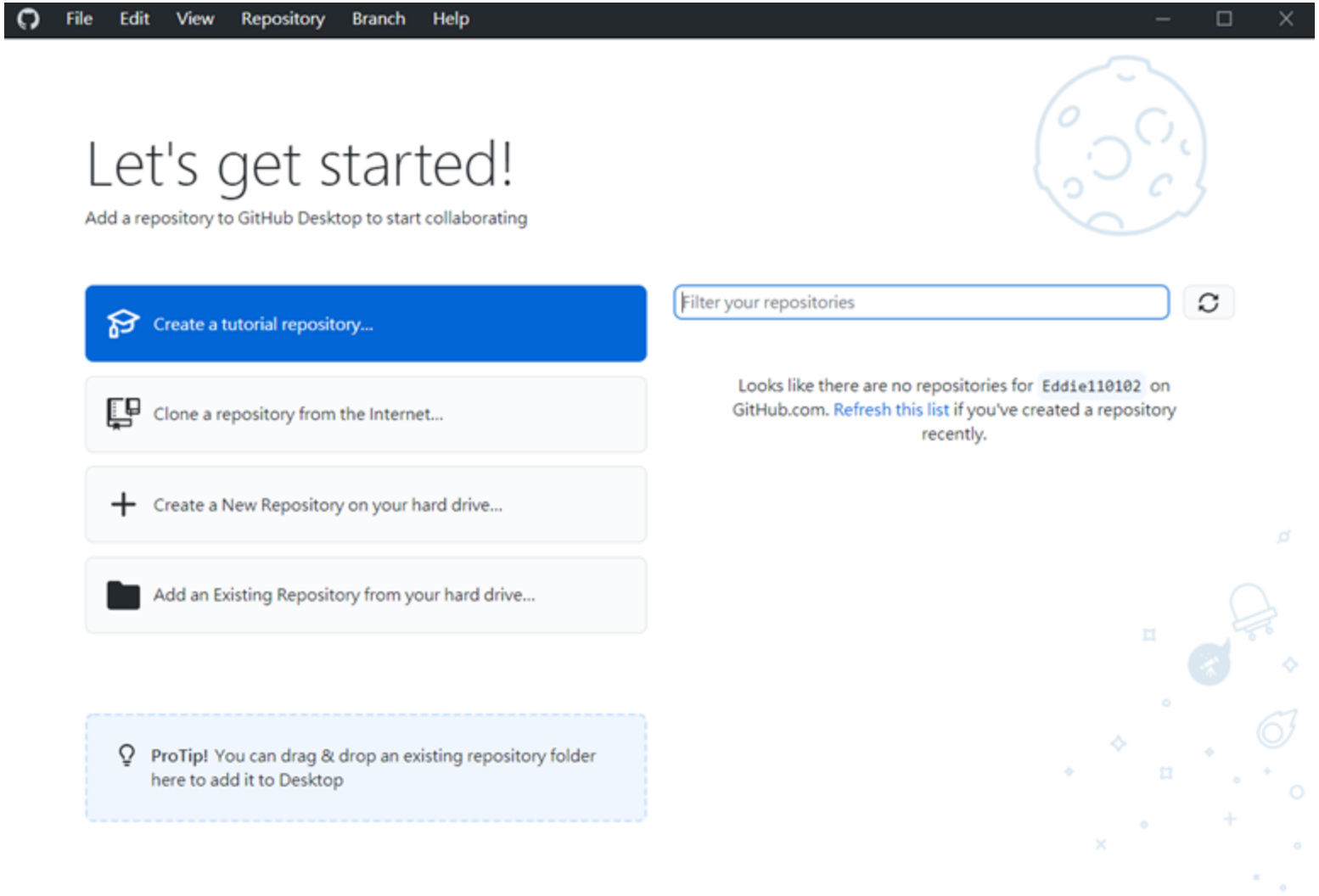


登入及授權 desktop版本



點擊完成

成功登入


File
Edit
View
Repository
Branch
Help
Let's get started!
Add a repository to GitHub Desktop to start collaborating
Create a tutorial repository...
Clone a repository from the Internet...
Create a New Repository on your hard drive...
Add an Existing Repository from your hard drive...
ProTip! You can drag & drop an existing repository folder here to add it to Desktop
Filter your repositories
Looks like there are no repositories for Eddie110102 on GitHub.com. Refresh this list if you've created a repository recently.

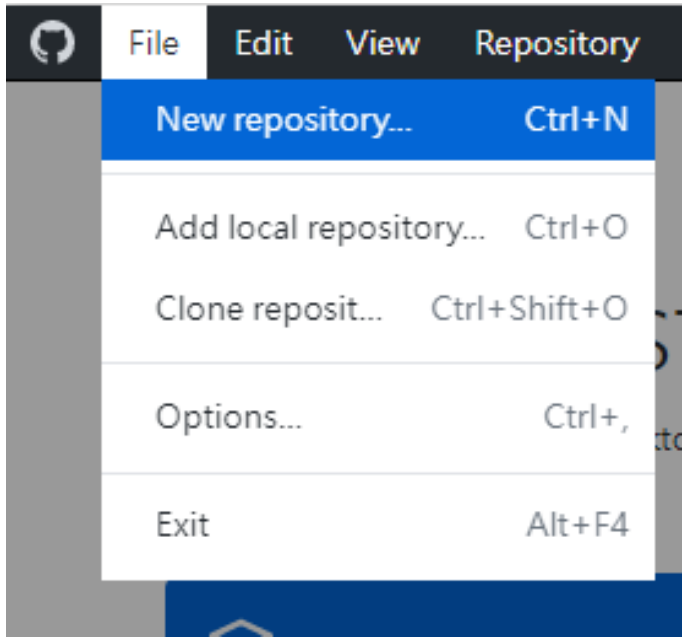
建立local工作目錄

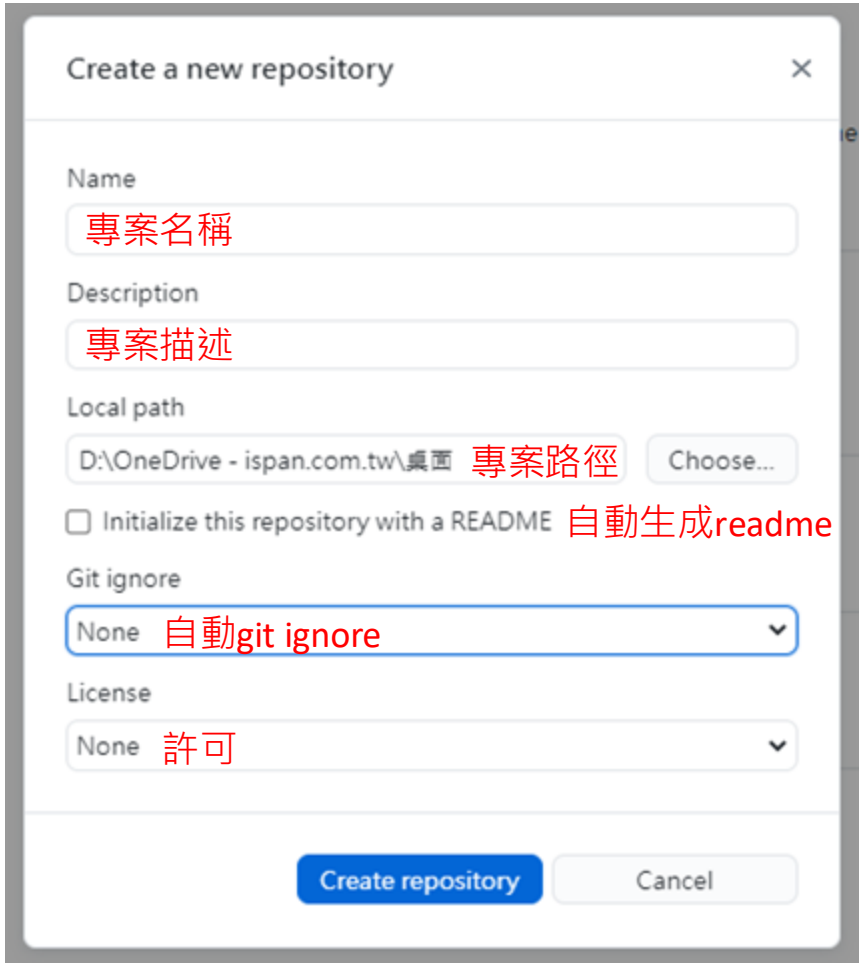
發布到遠端repo

完全沒有資料，從零開始。

建立新的本地工作目錄

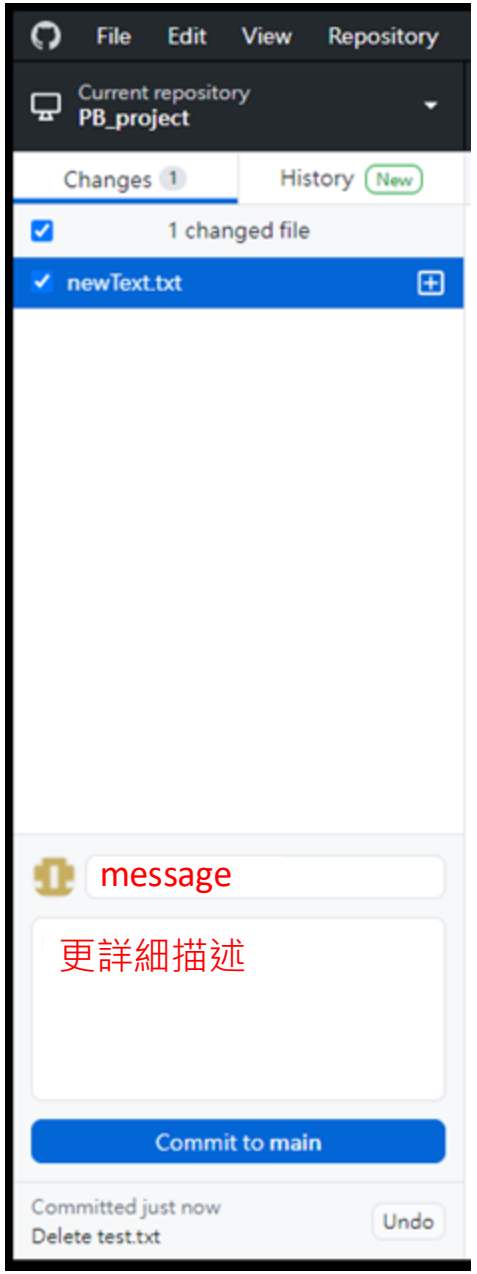
- 點擊 File -> New repository





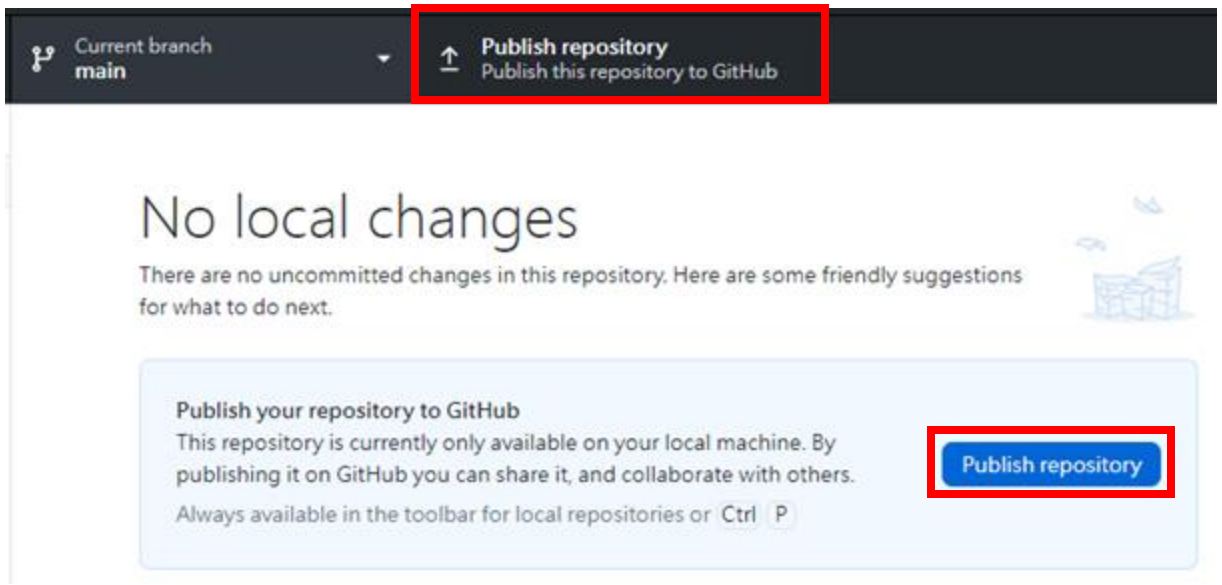
編輯工作目錄中的檔案

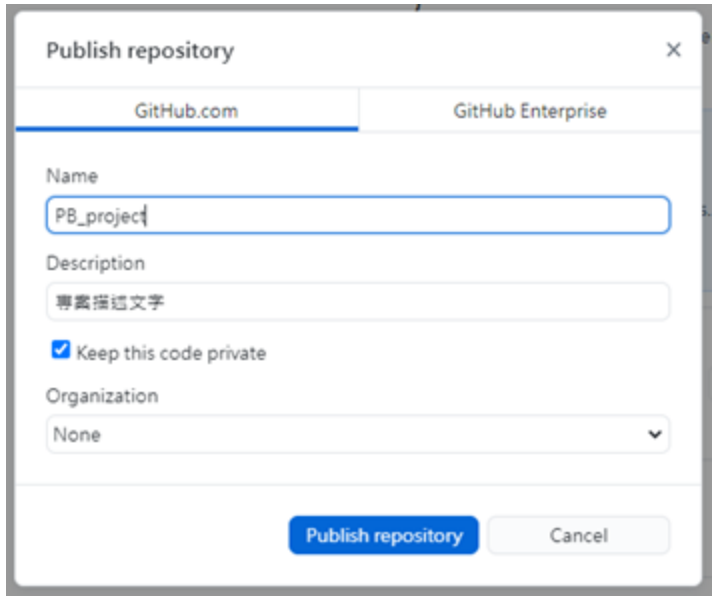
- 新增 / 刪除 / 編輯檔案後，在Desktop中即顯示該檔案，即可填寫message並commit。



發佈到遠端repo

- 點擊 Publish repositiory -> 發布到遠端 repo -> 填寫遠端repo名稱
- Keep this code private：將repo設為private





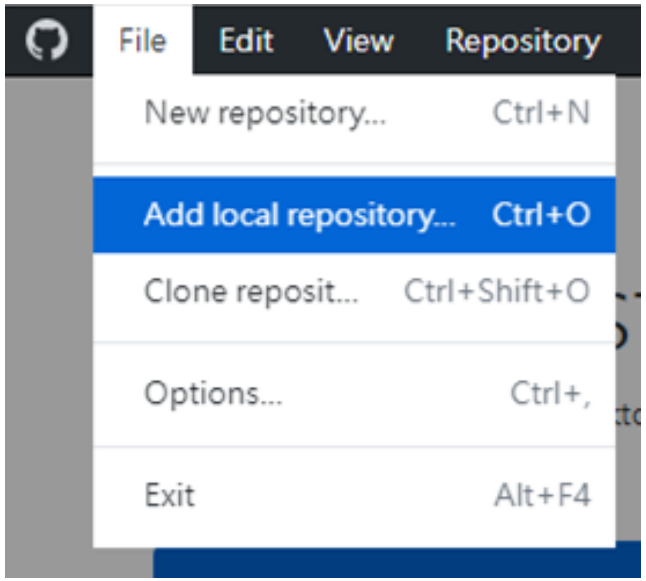
已有local工作目錄

發布到遠端repo

已經有專案資料，已有 .git 資料夾 / 尚未執行 git init 的專案 。

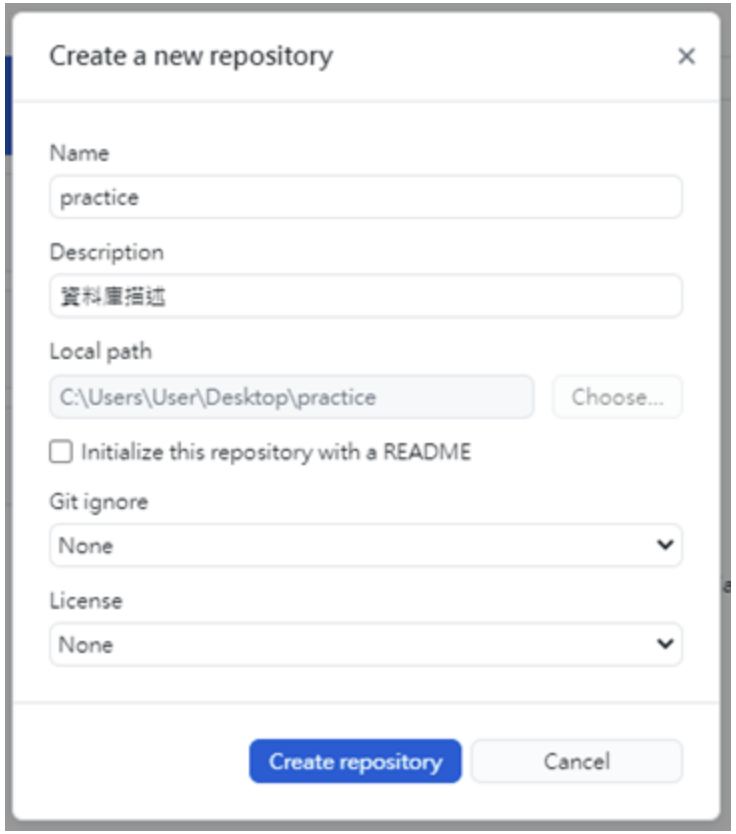
將本地工作目錄加到Desktop (已有 .git)

- 點擊 File -> Add local repository -> publish repository



將本地工作目錄加到Desktop (沒有 .git)

- 點擊 File -> Add local repository
  -> create a repository -> Create repository



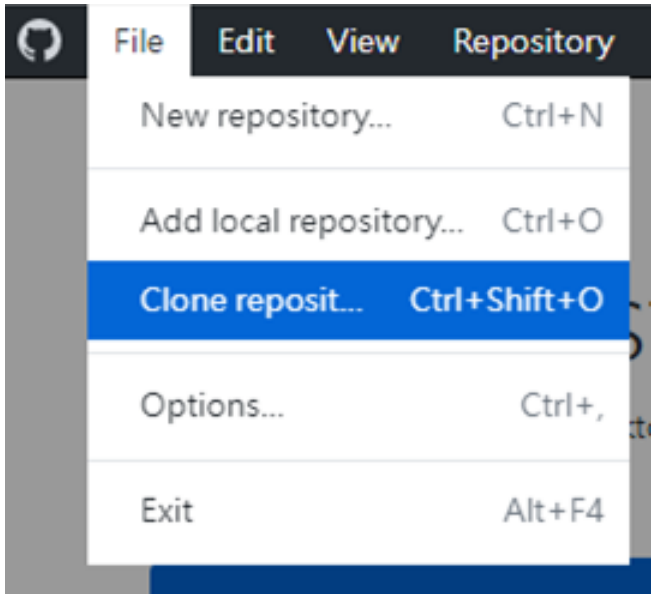
從遠端repo

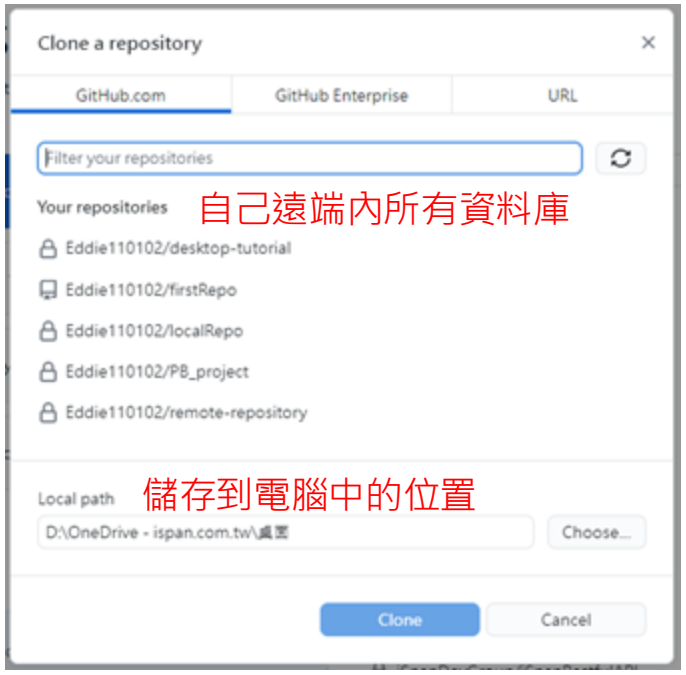
clone到本地

將遠端專案，複製到本機電腦。

從遠端repo下載到本地

- 點擊 File -> Clone repository -> 選擇遠端資料庫 -> 選擇儲存路徑





疑難雜症

Push時 顯示repo 找不到

- 先確認 repo的URL是否正確。
- 如果是共同協作的repo，是否有權限使用。
- 如果有多重帳號問題，在本機的控制台 -> 使用者帳戶 -> 認證管理員 -> windows認證中尋找 github相關帳號並移除。
- 重新 push 時 會需要再驗證一次帳戶即可。

疑難雜症

當我們輸入 git status 的時候，如果檔名或路徑中有中文字，而且畫面顯示亂碼

- git config --global core.quotepath false。

```
User@DESKTOP-LBN7FI8 MINGW64 /d/OneDrive - ispan.com.tw/桌面/test (main)
$ git st
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        新文字文件.txt
```

資料來源：https://blog.miniasp.com/post/2017/09/17/Git-for-Windows-Command-Prompt-Display-Chinese-Issues