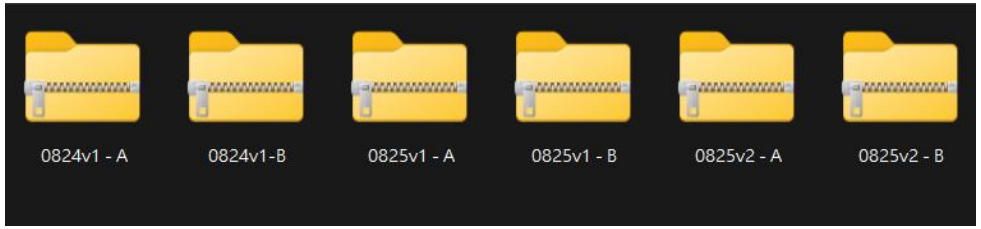
Git & GitHub

為何需要版本控制?

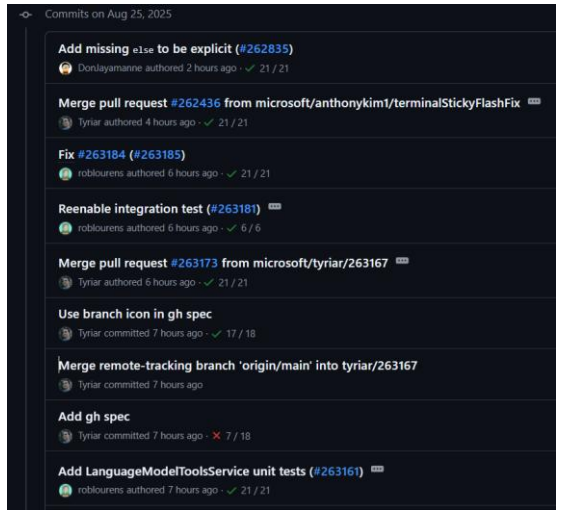
大量不同日期、不同人的壓縮檔



頻繁的傳程式來同步大家的內容

分工、實驗的不同版本

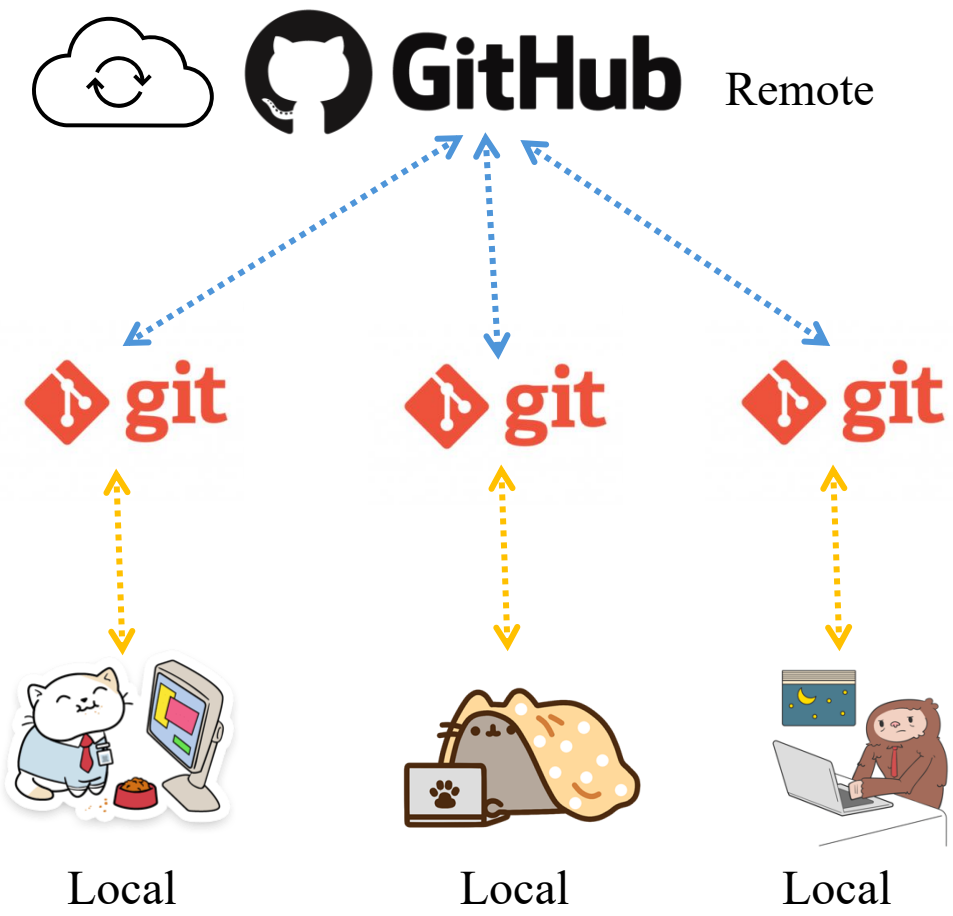
詳細的版本日誌



Git ? GitHub?

git ≠

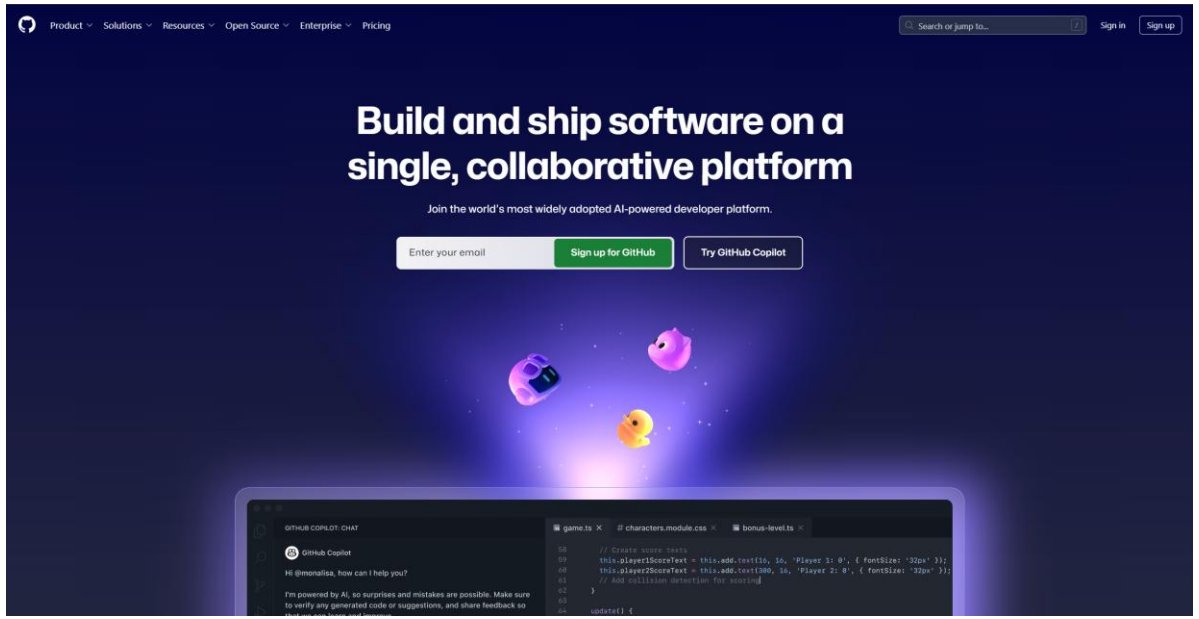
本質	分散式版本控制系統（軟體工具）	基於 Git 的雲端托管平台
安裝位置	安裝在你的電腦上	網頁服務，無需安裝
主要功能	版本控制、分支管理、合併	遠端儲存、協作、Issue 追蹤、PR
使用方式	命令列或圖形介面	網頁介面 + Git 命令
離線使用	✅ 完全可離線工作	❌ 需要網路連線
類比	像 Word 的「追蹤修訂」功能	像 Google Drive 的共享資料夾
競爭對手	SVN, Mercurial	GitLab, Bitbucket



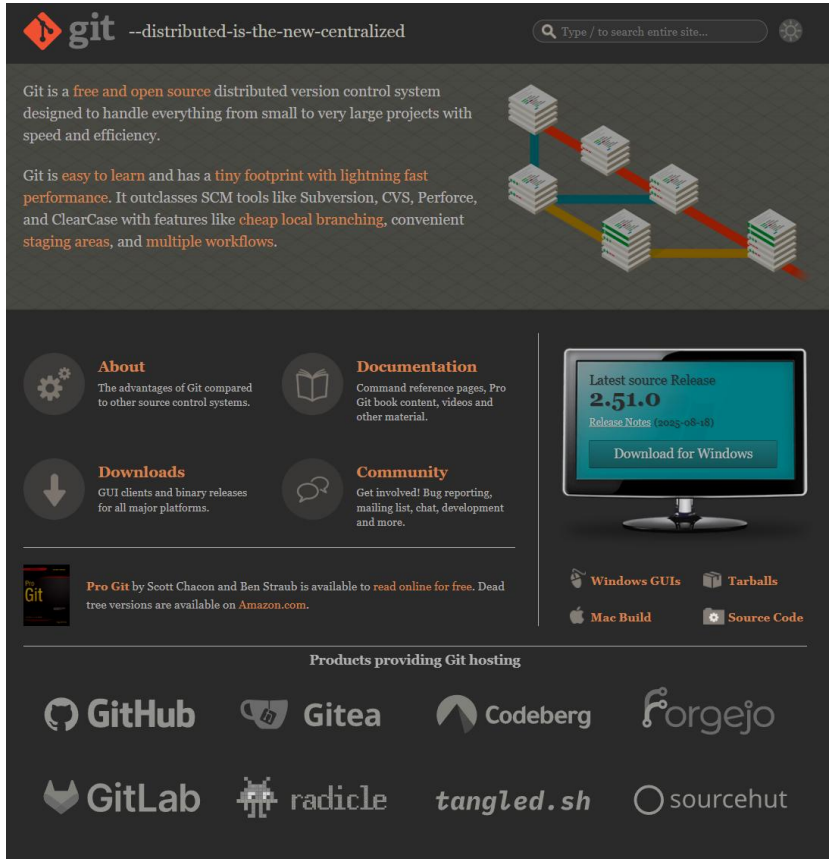
安装

行前準備

連結:GitHub



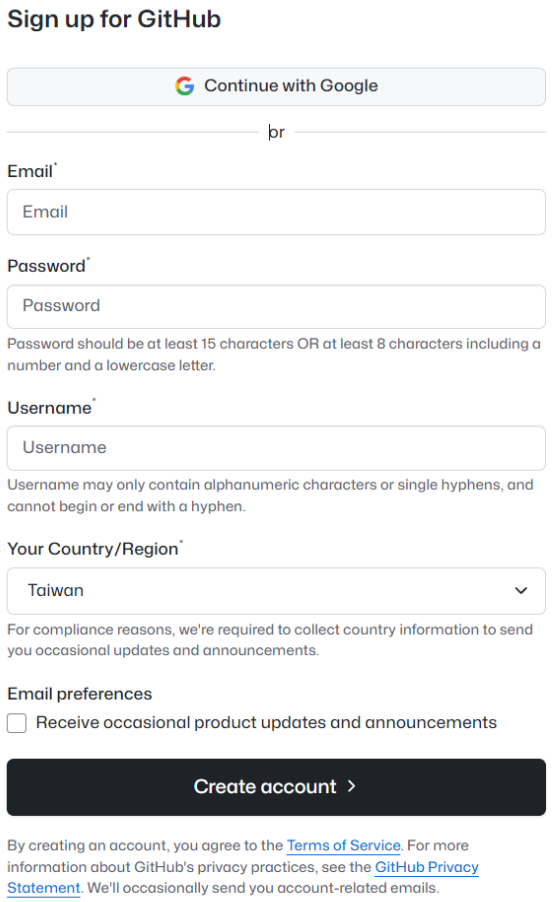
連結: Git



建立GitHub帳號

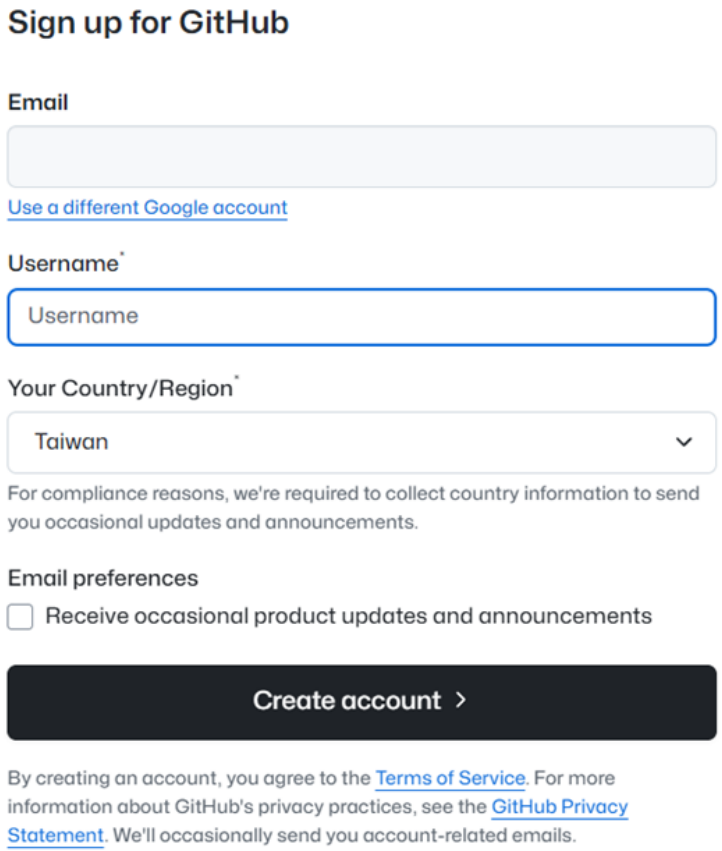
連結: GitHub

1.使用goolge或其他信箱

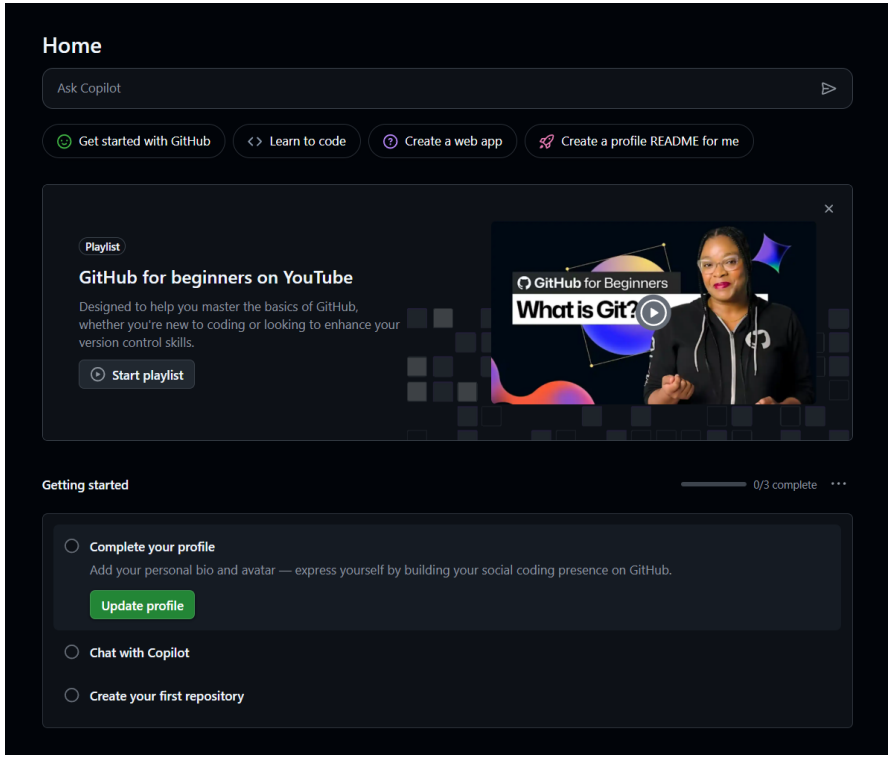


2.取個名字

(盡量不要更改，會導致儲存庫連結更改)



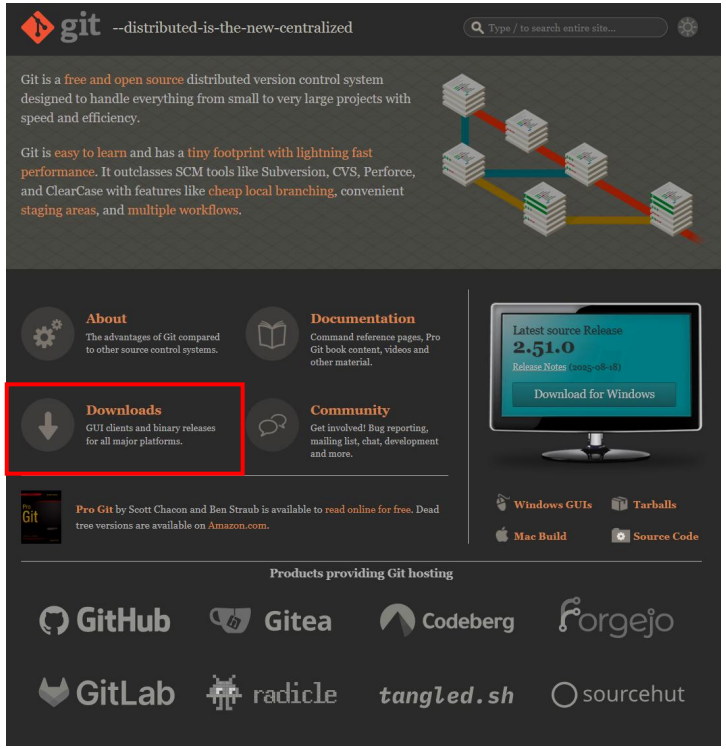
2.完成!



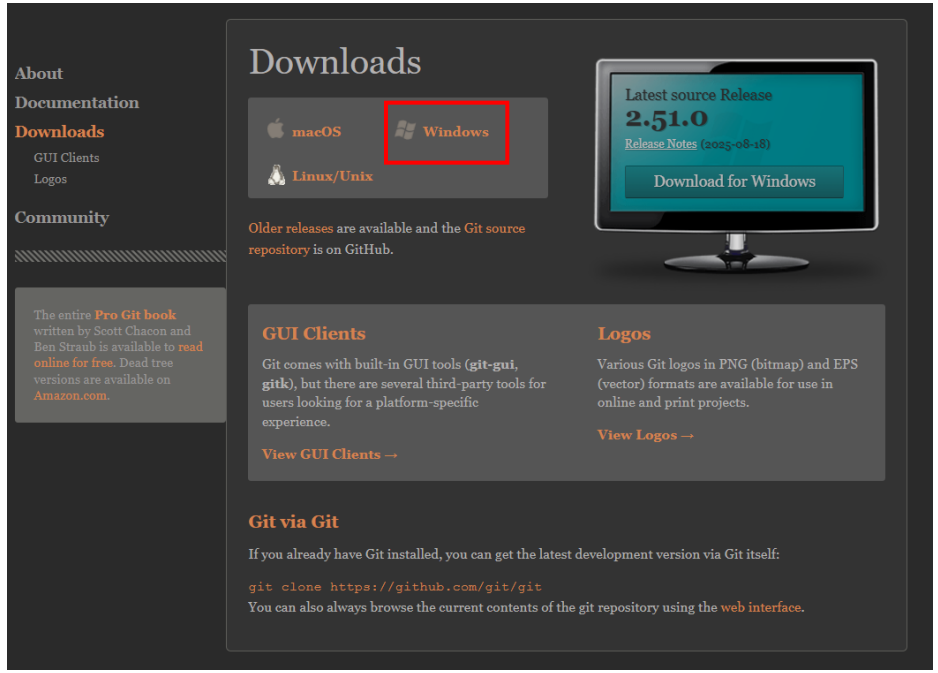
安裝Git

連結: Git

1.點擊下載



2.選擇系統
(macOS、Linux使用指令安裝)



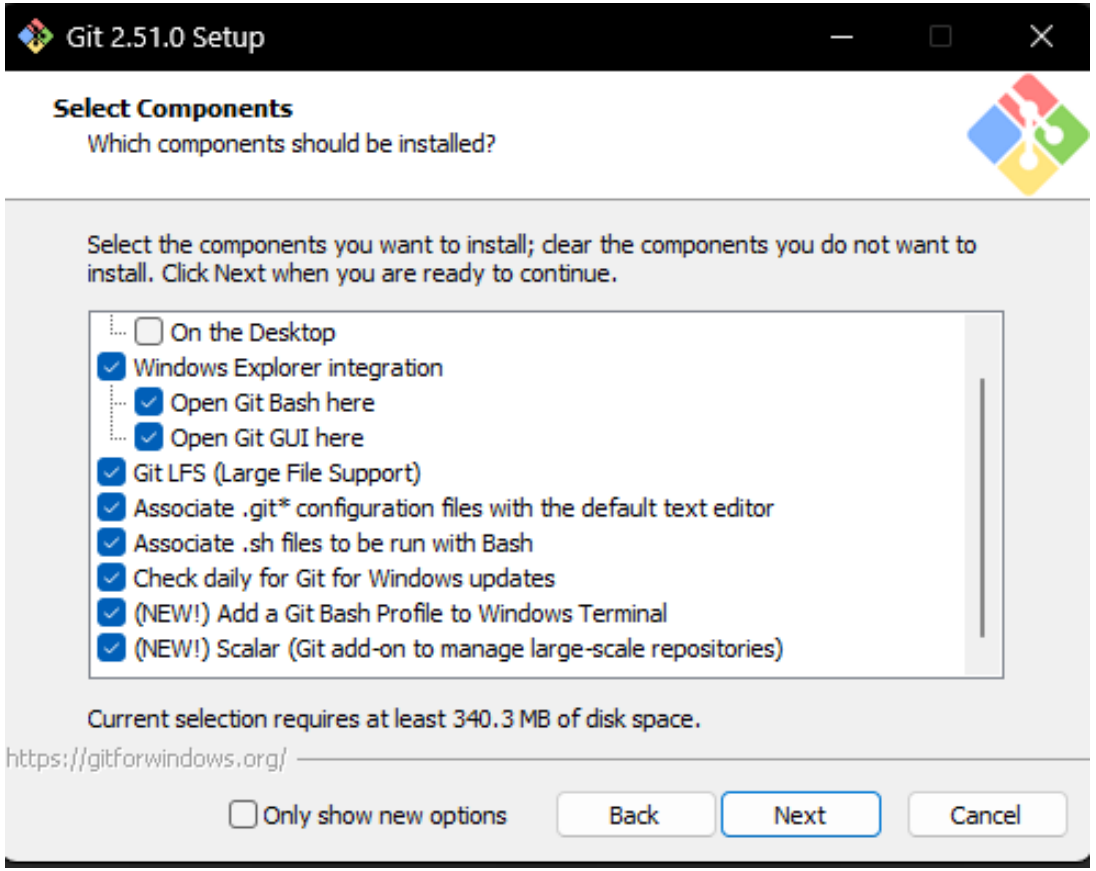
3.選擇版本

直接下一步


Git 2.51.0 Setup
Information
Please read the following important information before continuing.
When you are ready to continue with Setup, click Next.
GNU General Public License
Version 2, June 1991
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
Preamble
The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change
https://gitforwindows.org/
Only show new options
Next
Cancel

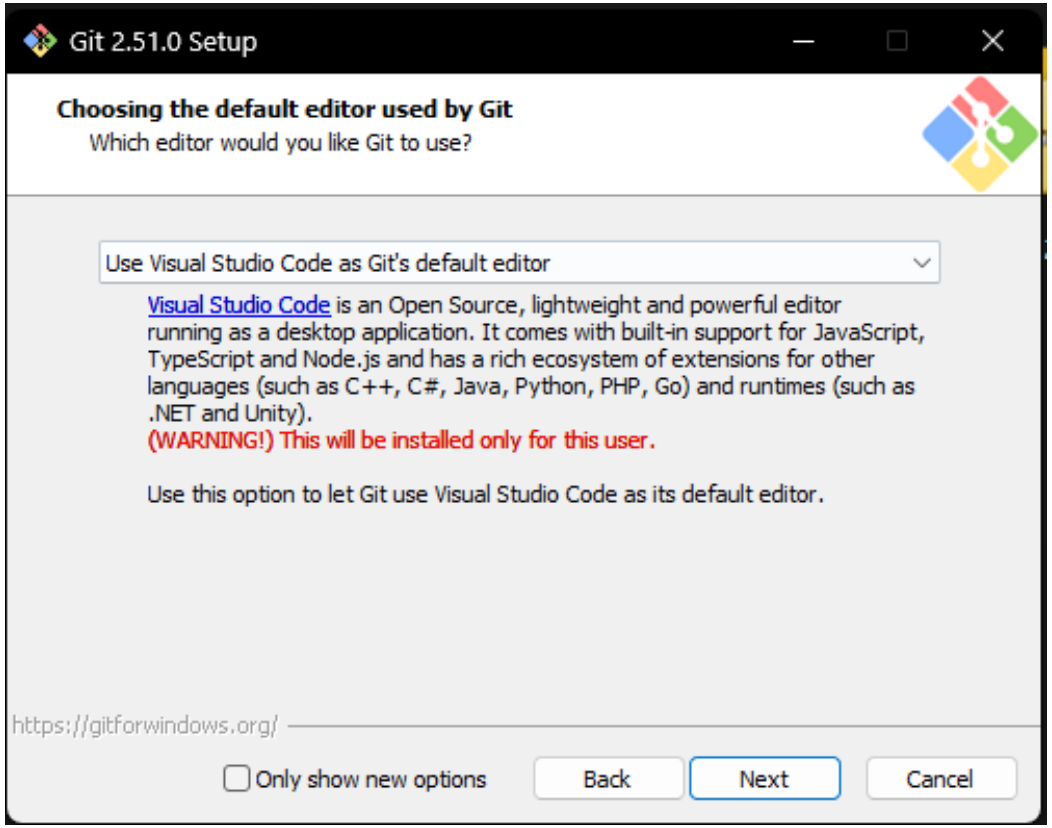
通常不用改

- 必選：Windows Explorer integration（右鍵開 Git）
- 建議：Open Git Bash here、Add Git Bash Profile to Windows Terminal
- 視需要：Open Git GUI here（圖形介面）
- 大檔案：Git LFS（之後需要時再裝也可）
- 便利：Associate .git*、Associate .sh（讓設定檔/腳本可直接用編輯器/ Bash）
- Scalar（大型倉庫才需要）



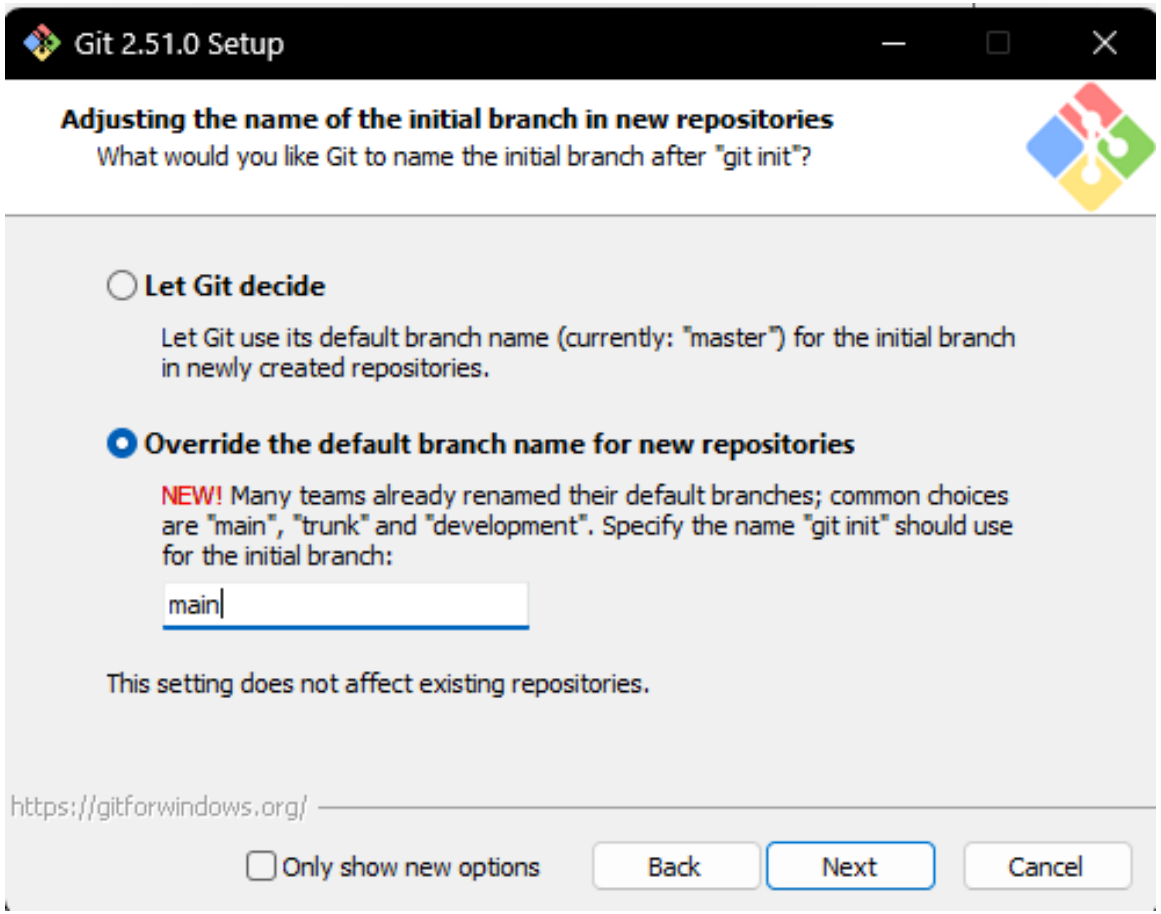
選擇常用的編輯器

- 先選：Use Visual Studio Code as Git's default editor



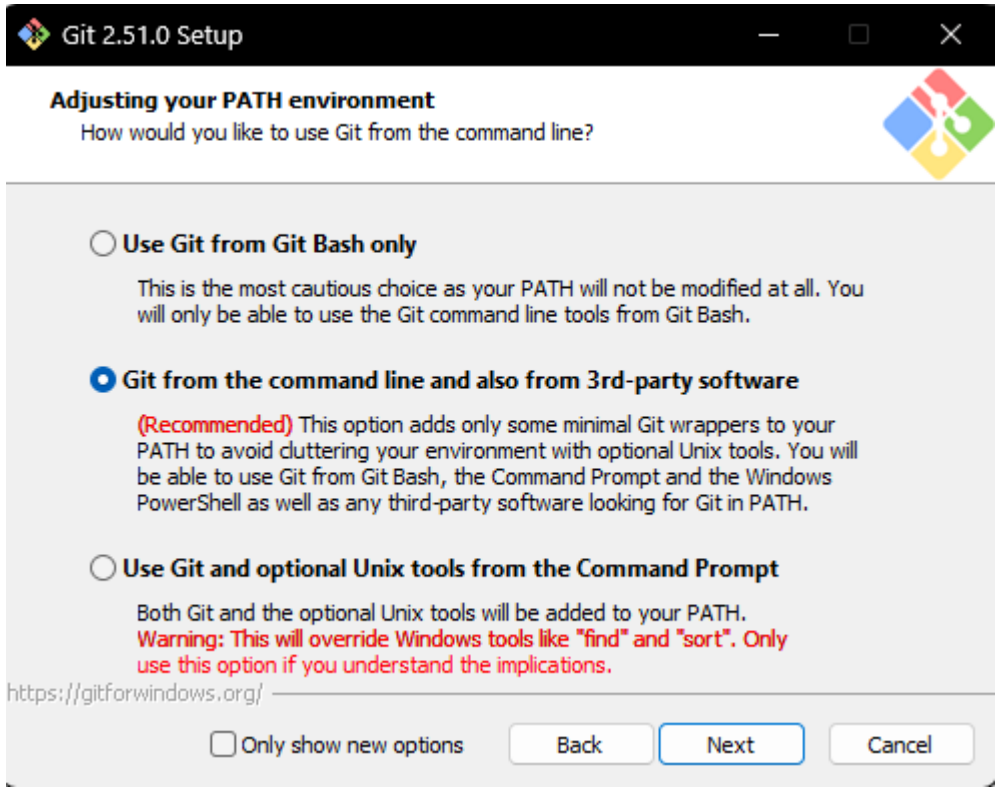
初始分支名稱

- 選下面的
- 現在主分支通常叫做 main
- 與 GitHub 預設一致，避免 master / main 混淆

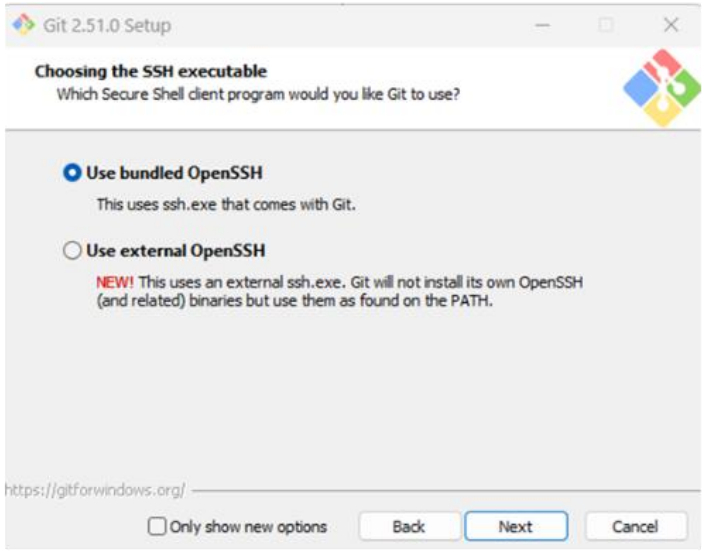


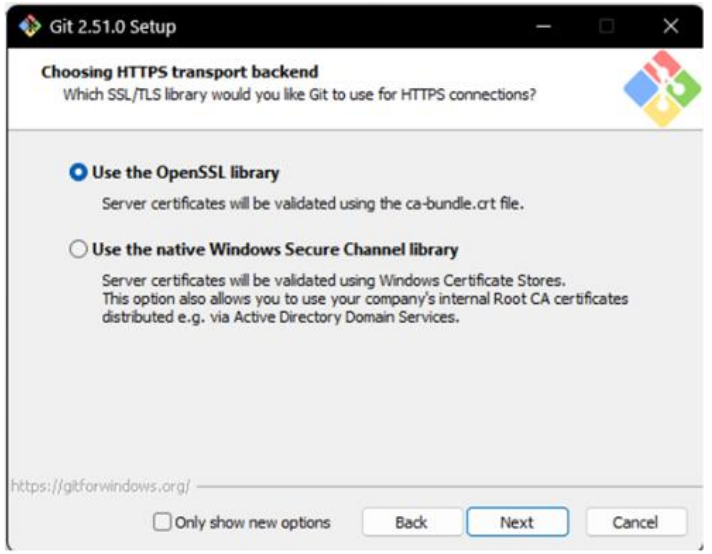
環境變數

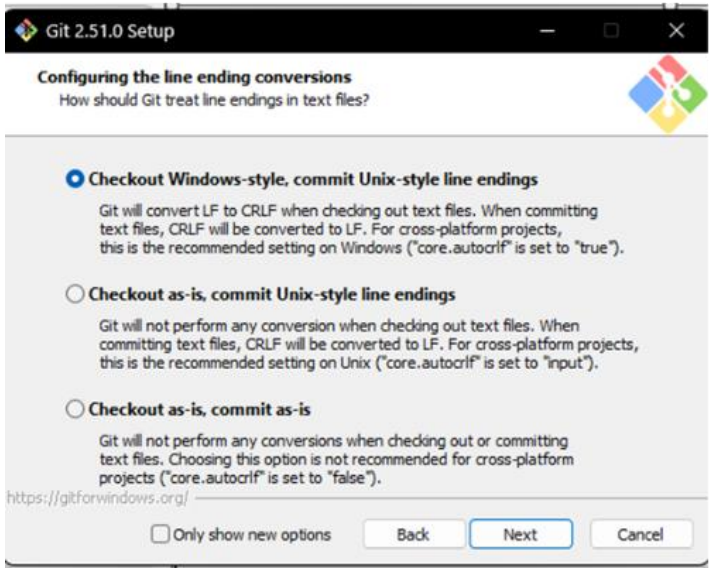
- 通常中間的就好



都使用預設即可


Git 2.51.0 Setup
Choosing the SSH executable
Which Secure Shell client program would you like Git to use?
Use bundled OpenSSH
This uses ssh.exe that comes with Git.
Use external OpenSSH
NEW! This uses an external ssh.exe. Git will not install its own OpenSSH (and related) binaries but use them as found on the PATH.
https://gitforwindows.org/
Only show new options
Back
Next
Cancel

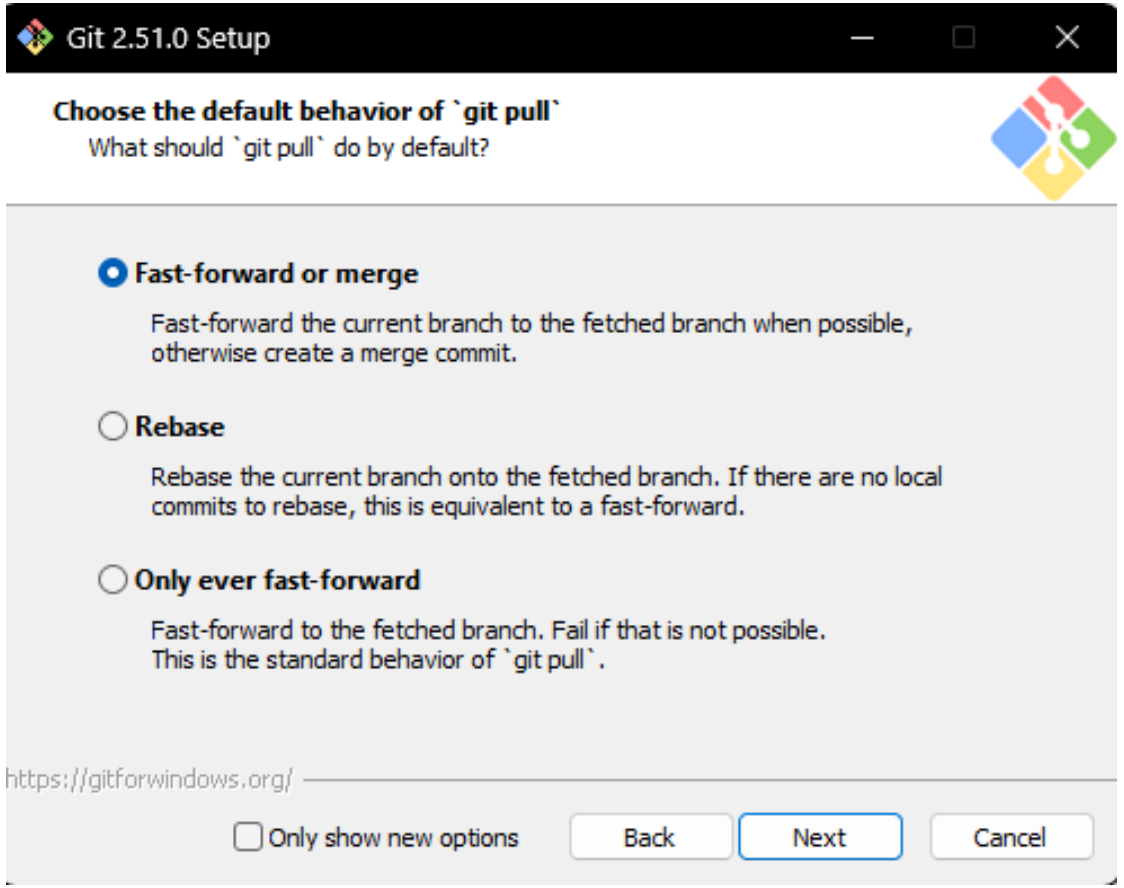

Git 2.51.0 Setup
Choosing HTTPS transport backend
Which SSL/TLS library would you like Git to use for HTTPS connections?
Use the OpenSSL library
Server certificates will be validated using the ca-bundle.crt file.
Use the native Windows Secure Channel library
Server certificates will be validated using Windows Certificate Stores. This option also allows you to use your company's internal Root CA certificates distributed e.g. via Active Directory Domain Services.
https://gitforwindows.org/
Only show new options
Back
Next
Cancel


Git 2.51.0 Setup
Configuring the line ending conversions
How should Git treat line endings in text files?
Checkout Windows-style, commit Unix-style line endings
Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").
Checkout as-is, commit Unix-style line endings
Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").
Checkout as-is, commit as-is
Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").
https://gitforwindows.org/
Only show new options
Back
Next
Cancel

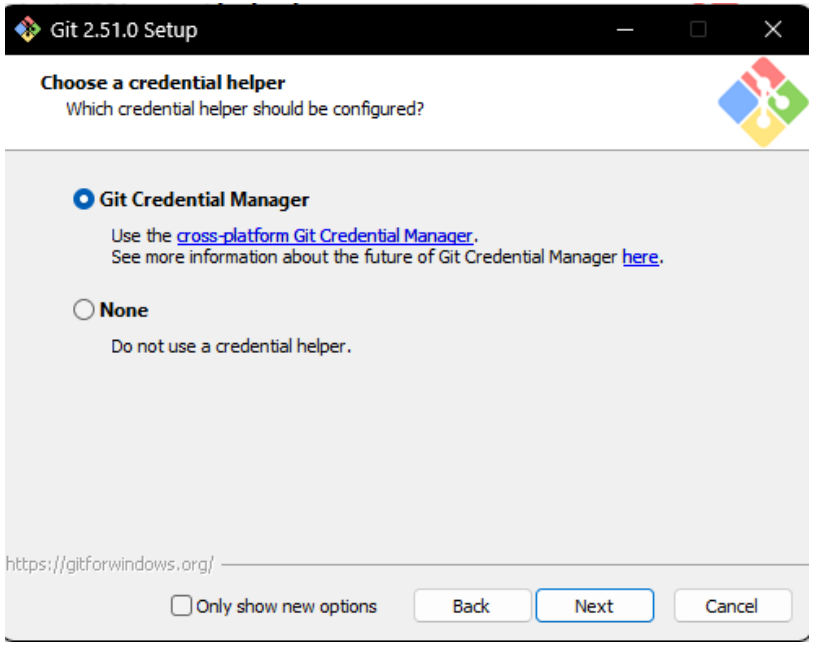

Git 2.51.0 Setup
Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?
Use MinTTY (the default terminal of MSYS2)
Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.
Use Windows' default console window
Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.
https://gitforwindows.org/
Only show new options
Back
Next
Cancel

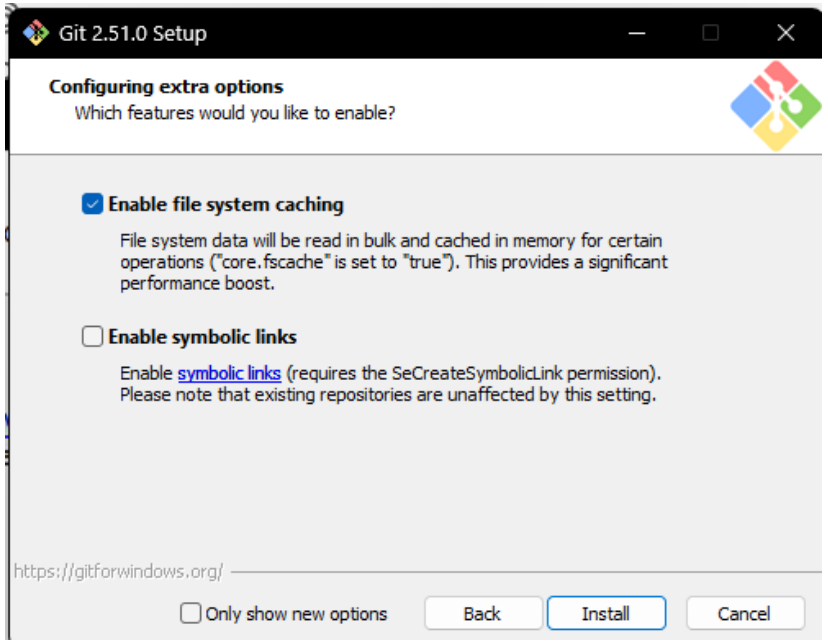
git pull 預設行為

- Beginner 推薦：Fast-forward or merge（預設；保留真實歷史）
- Rebase：歷史線性但概念較進階（適合熟悉後）
- Only ever fast-forward：若需 merge 則會失敗；新手不建議



繼續按Next


Git 2.51.0 Setup
Choose a credential helper
Which credential helper should be configured?
Git Credential Manager
Use the cross-platform Git Credential Manager.
See more information about the future of Git Credential Manager here.
None
Do not use a credential helper.
https://gitforwindows.org/
Only show new options
Back
Next
Cancel


Git 2.51.0 Setup
Configuring extra options
Which features would you like to enable?
Enable file system caching
File system data will be read in bulk and cached in memory for certain operations ("core.fscache" is set to "true"). This provides a significant performance boost.
Enable symbolic links
Enable symbolic links (requires the SeCreateSymbolicLink permission).
Please note that existing repositories are unaffected by this setting.
https://gitforwindows.org/
Only show new options
Back
Install
Cancel

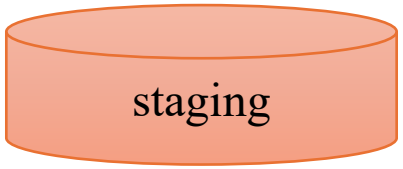
完成

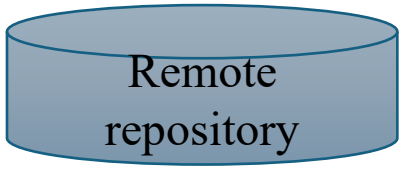

Git 2.51.0 Setup
Completing the Git Setup Wizard
Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.
Click Finish to exit Setup.
Launch Git Bash
View Release Notes
Only show new options
Finish

Implementation

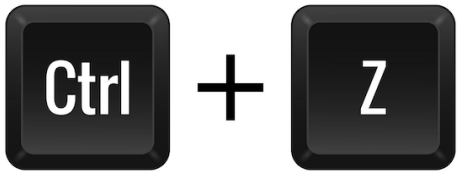
Workspace、staging、Local repository、 Remote repository

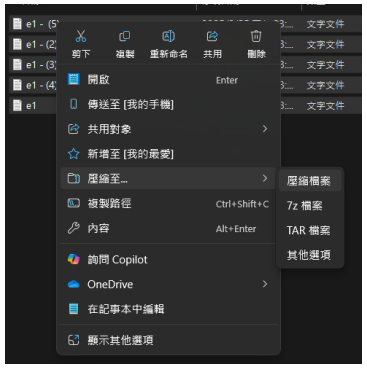
```
不用 Git：                    用 Git：
(單一資料夾亂改)               Workspace (修改區)
  ├ final_v1.c                    ↓ git add
  ├ final_v2.c                Staging (挑選要的變更)
  ├ try_fix.c                     ↓ git commit
  ├ backup_old/               Local Repo (歷史 + 可分支)
  └ 2024_backup.zip               ↓ git push
                              Remote Repo (協作 / 備份)
```

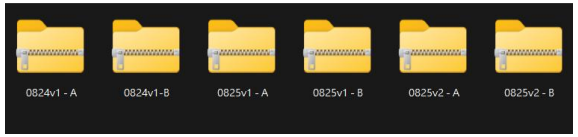




	正在編輯的檔案	準備提交的變更	已記錄的版本歷史	雲端共享的版本
After	目前正在編輯的檔案實體，尚未進入版本記錄	暫存你「這次要提交」的精準變更集合	以哈希記錄的不可變歷史（快照＋作者＋時間＋訊息）	共享同步版本歷史，支援協作、PR、Issue
Brfore	直接改原檔 靠 Ctrl+Z 回復 用一堆檔名加備份	沒概念；一次全部壓縮成壞掉的打包	手動存多個資料夾	用雲端硬碟 / USB / 即時通傳 zip；覆蓋彼此檔案；靠「誰最新」



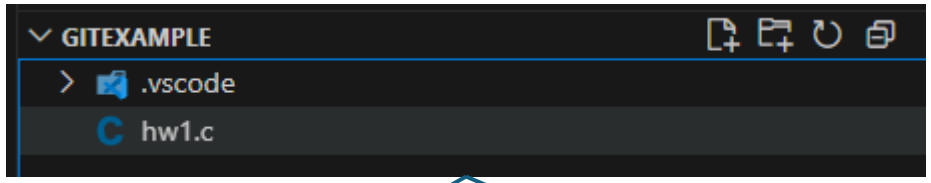




Initial

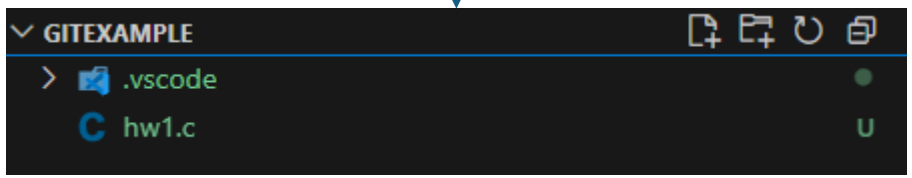
初始化儲存庫

建立檔案

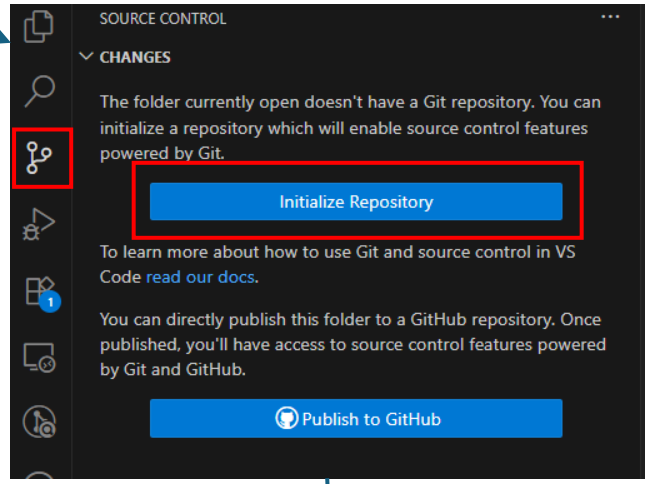


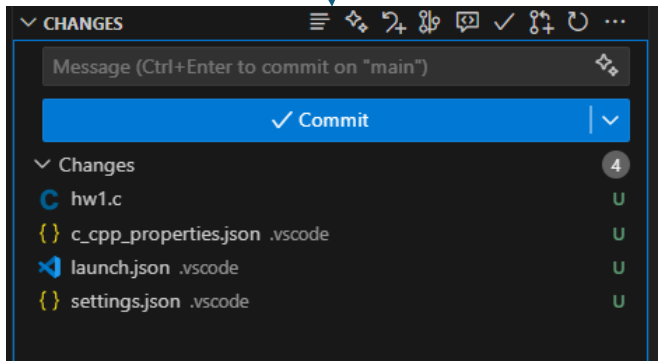
使用命令

```
E:\GitExample>git init
Initialized empty Git repository in E:/GitExample/.git/
```



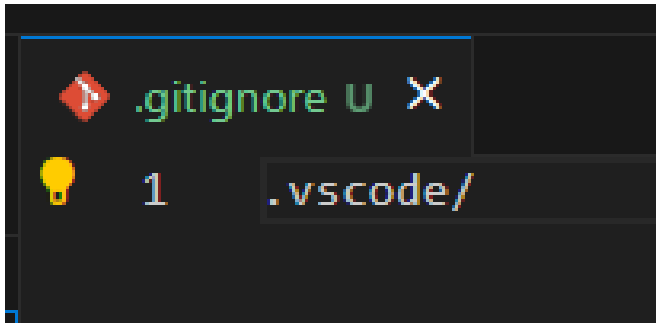
使用VS code UI

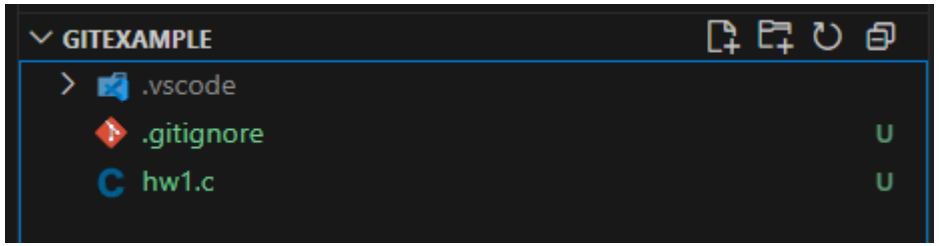


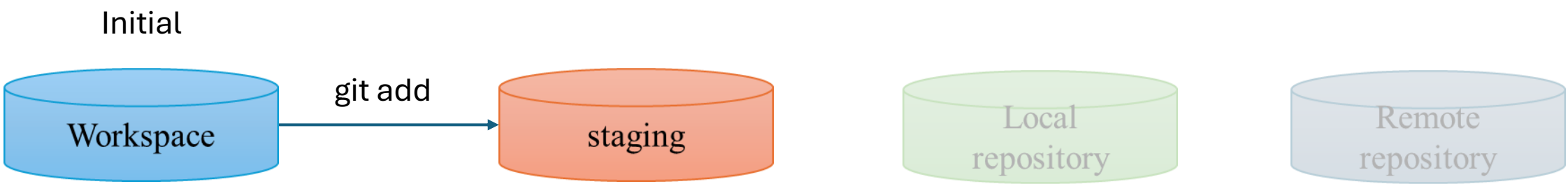


.gitignore

讓git忽略不需要追蹤的檔案或資料夾
例如:設定檔、金鑰、資料庫...



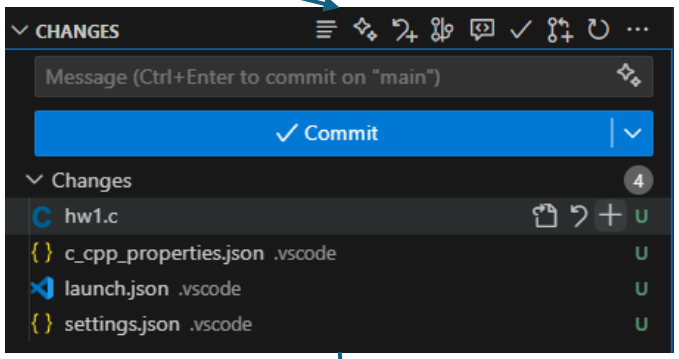



Initial
Workspace
git add
staging
Local repository
Remote repository

git add
git status

git add 檔案


```
E:\GitExample>git add hw1.c
```



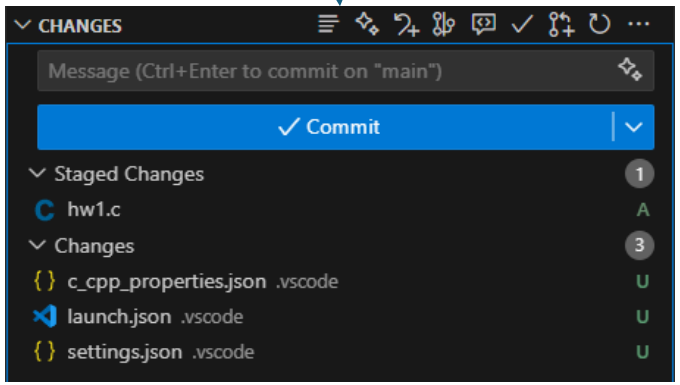
git status

```
E:\GitExample>git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   hw1.c

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



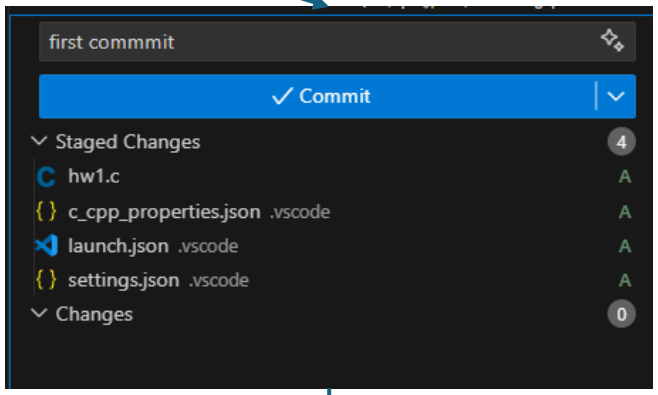

Initial
Workspace
git add
staging
git commit
Local repository
Remote repository

git commit

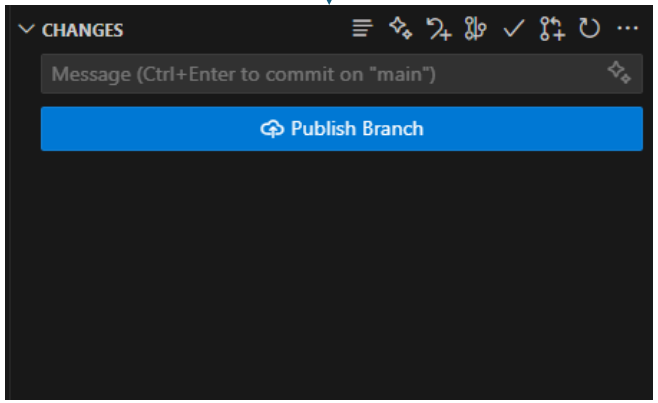
git add .
git commit –m “訊息”


```
E:\GitExample>git add .

E:\GitExample>git commit -m "first commit"
[main (root-commit) 0a3f7da] first commit
 2 files changed, 1 insertion(+)
 create mode 100644 .gitignore
 create mode 100644 hw1.c
```

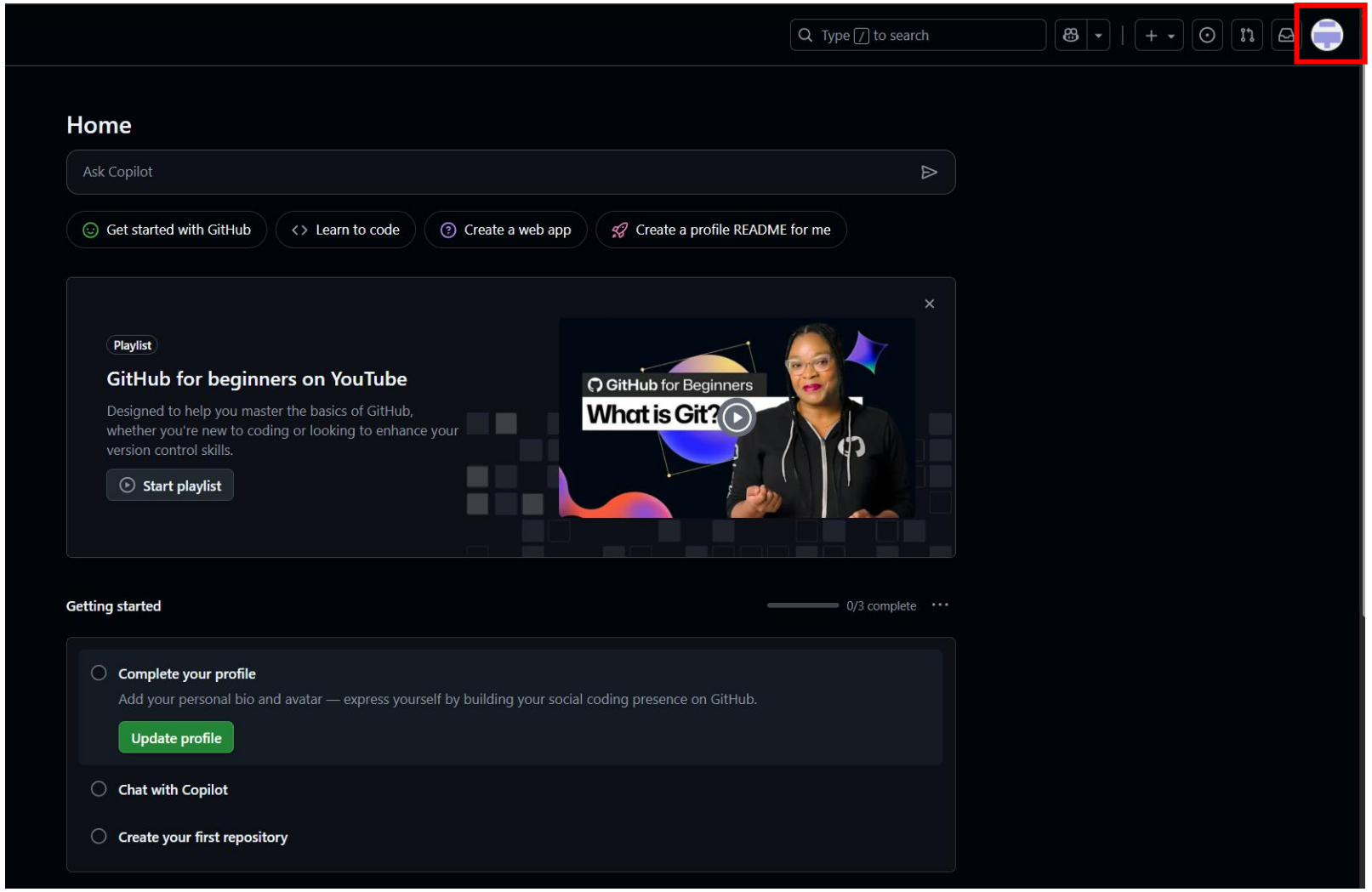


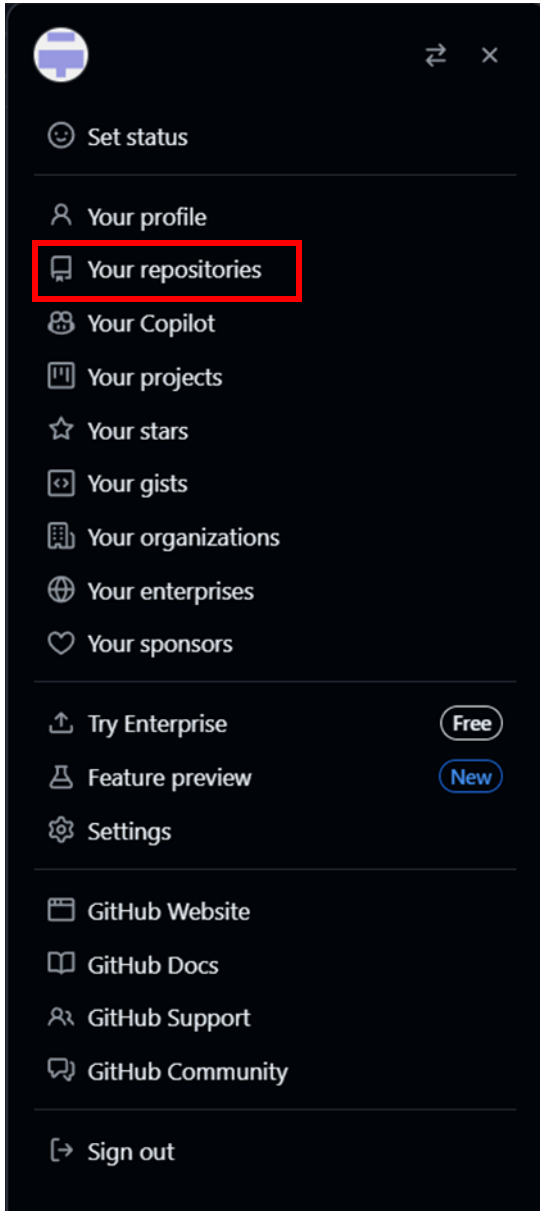

```
E:\GitExample>git status
On branch main
nothing to commit, working tree clean
```



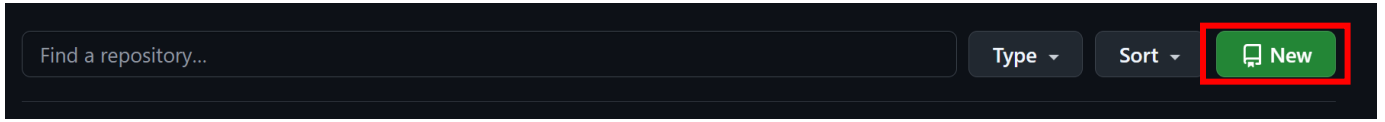

Initial
Workspace
git add
staging
git commit
Local repository
git push
Remote repository

創建遠端儲存庫

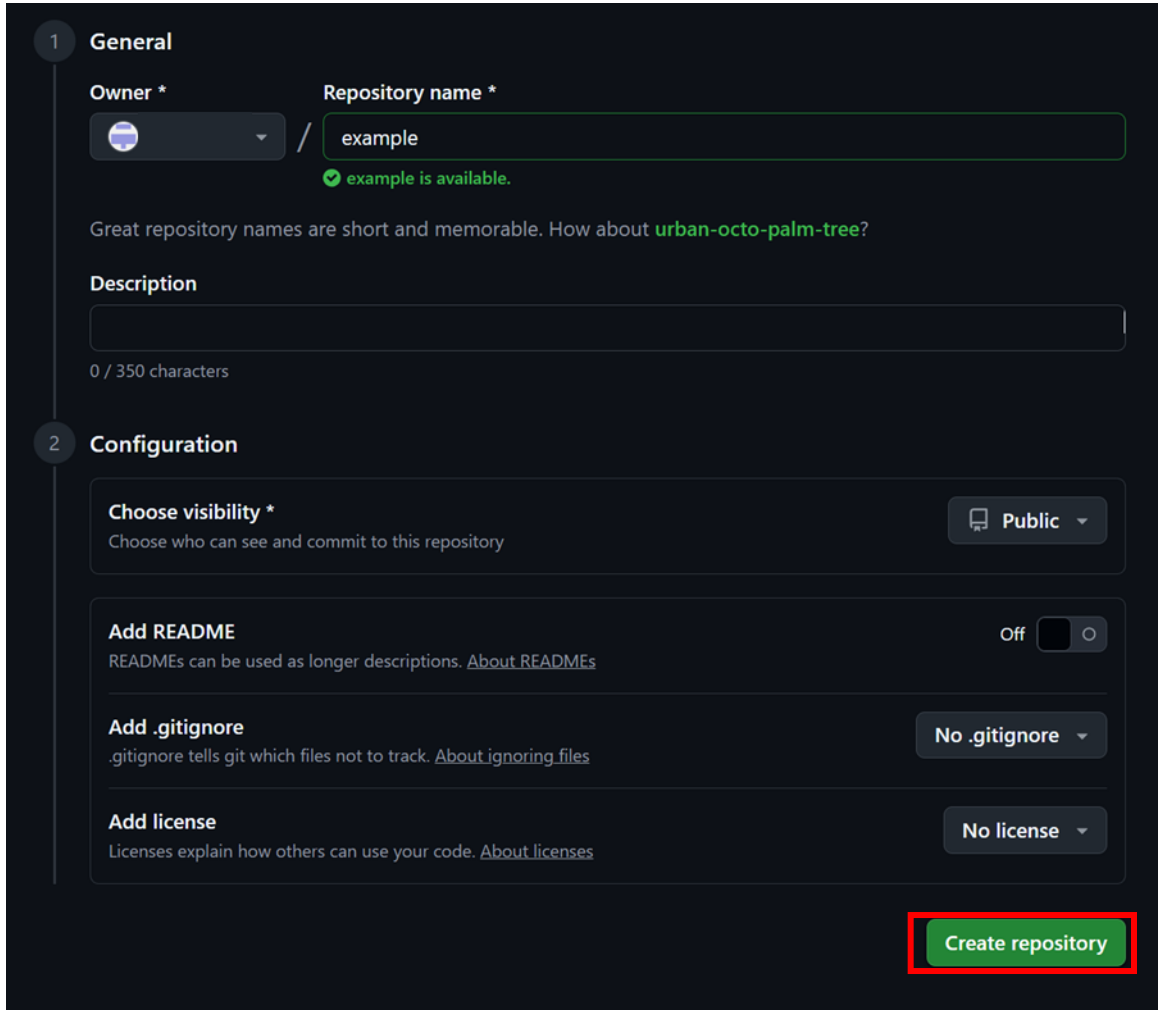

Type / to search
Home
Ask Copilot
Get started with GitHub
Learn to code
Create a web app
Create a profile README for me
Playlist
GitHub for beginners on YouTube
Designed to help you master the basics of GitHub, whether you're new to coding or looking to enhance your version control skills.
Start playlist
GitHub for Beginners
What is Git?
Getting started
0/3 complete
Complete your profile
Add your personal bio and avatar — express yourself by building your social coding presence on GitHub.
Update profile
Chat with Copilot
Create your first repository


Set status
Your profile
Your repositories
Your Copilot
Your projects
Your stars
Your gists
Your organizations
Your enterprises
Your sponsors
Try Enterprise
Free
Feature preview
New
Settings
GitHub Website
GitHub Docs
GitHub Support
GitHub Community
Sign out

創建遠端儲存庫



Repository name:盡量不要改
Description:可以不用寫
visibility :Private or public
README:在本地創建就好
.gitignore:在本地創建就好
license:在本地創建就好

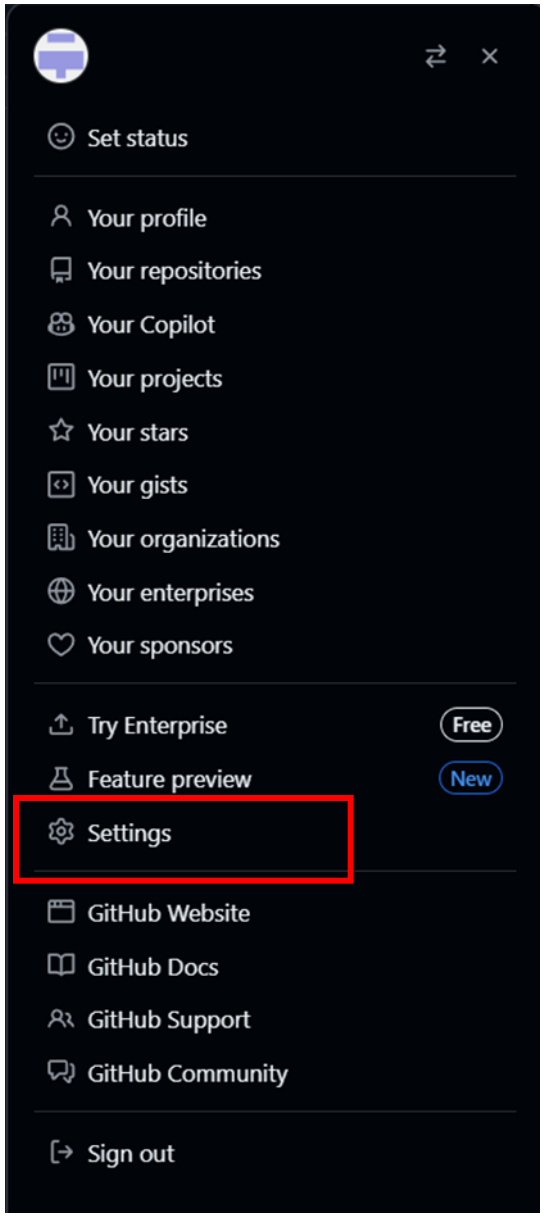


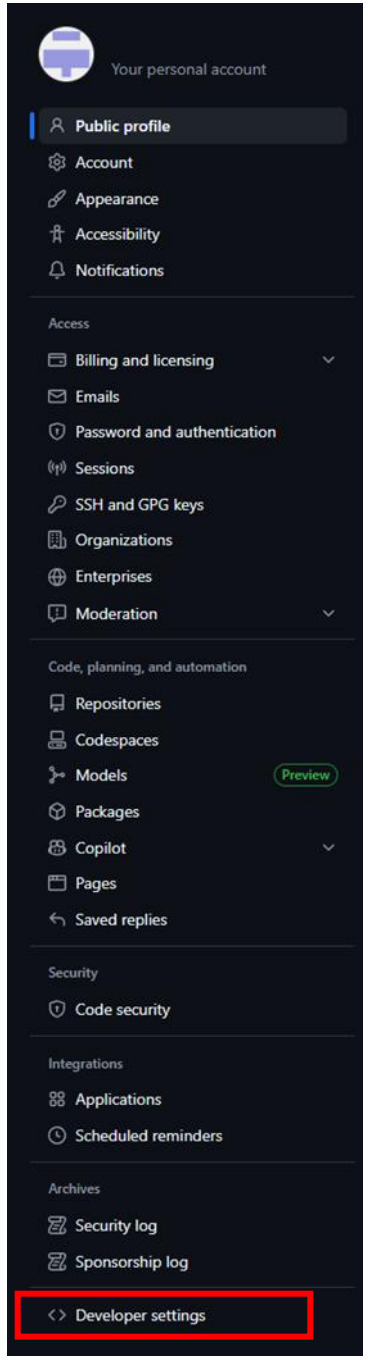
git add remote

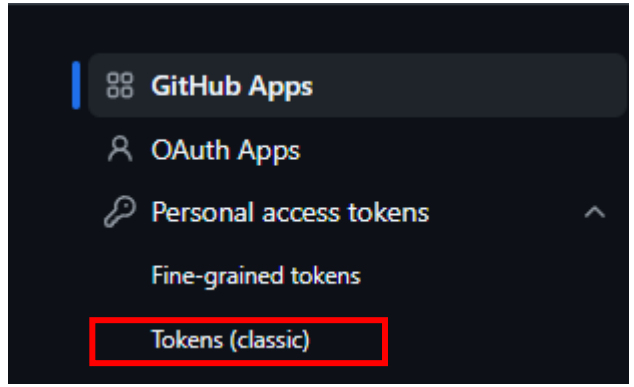
創建好後我們需要將本地與遠端連接，有兩種方式

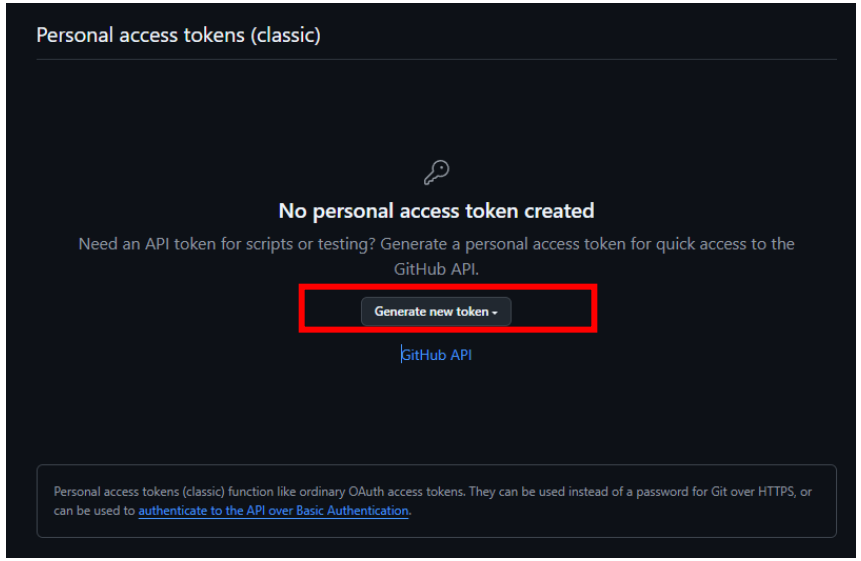
- 自己的筆電 / 長期開發： SSH
- 教室公用電腦或暫時操作：用 HTTPS

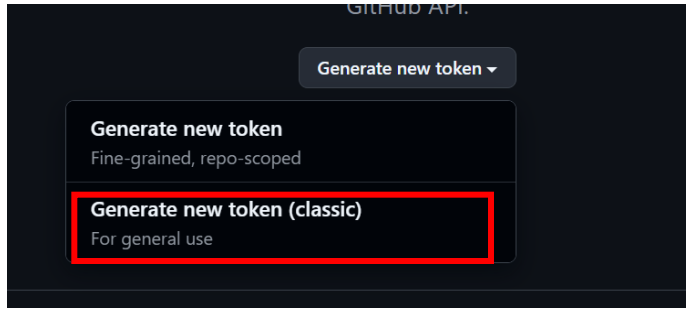
方法	HTTPS	SSH
驗證方式	Username + Personal Access Token	公私鑰（public 上傳 GitHub，private 本地）
初次設定複雜度	低（生成 PAT 貼上即可）	中（產生金鑰、加 passphrase、上傳）
後續使用便利	需憑證快取，可能偶爾再輸入	一次設定，後續幾乎免輸入
適用情境	臨時 / 公用或教室電腦（不留私鑰）	個人常用開發環境
風險點	PAT 遺失或外洩（可撤銷）	私鑰外洩（必須立刻撤除 key）
權限管理	PAT 可細緻設定 scope	單一金鑰常給全部 Repo（User 層級）
URL 形式	https://github.com/USER/REPO.git	git@github.com:USER/REPO.git


Set status
Your profile
Your repositories
Your Copilot
Your projects
Your stars
Your gists
Your organizations
Your enterprises
Your sponsors
Try Enterprise
Free
Feature preview
New
Settings
GitHub Website
GitHub Docs
GitHub Support
GitHub Community
Sign out

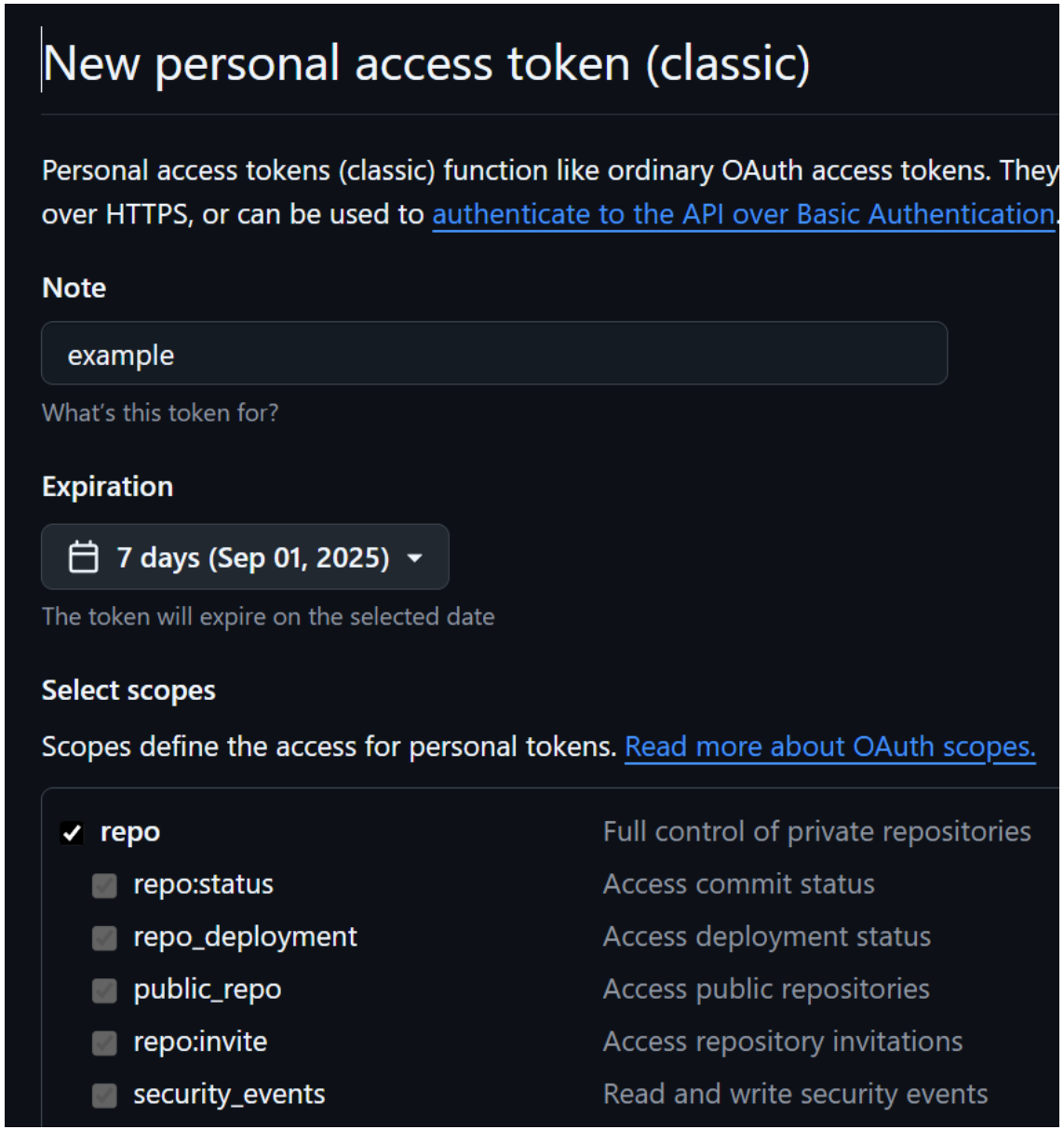

Your personal account
Public profile
Account
Appearance
Accessibility
Notifications
Access
Billing and licensing
Emails
Password and authentication
Sessions
SSH and GPG keys
Organizations
Enterprises
Moderation
Code, planning, and automation
Repositories
Codespaces
Models
Preview
Packages
Copilot
Pages
Saved replies
Security
Code security
Integrations
Applications
Scheduled reminders
Archives
Security log
Sponsorship log
Developer settings


GitHub Apps
OAuth Apps
Personal access tokens
Fine-grained tokens
Tokens (classic)

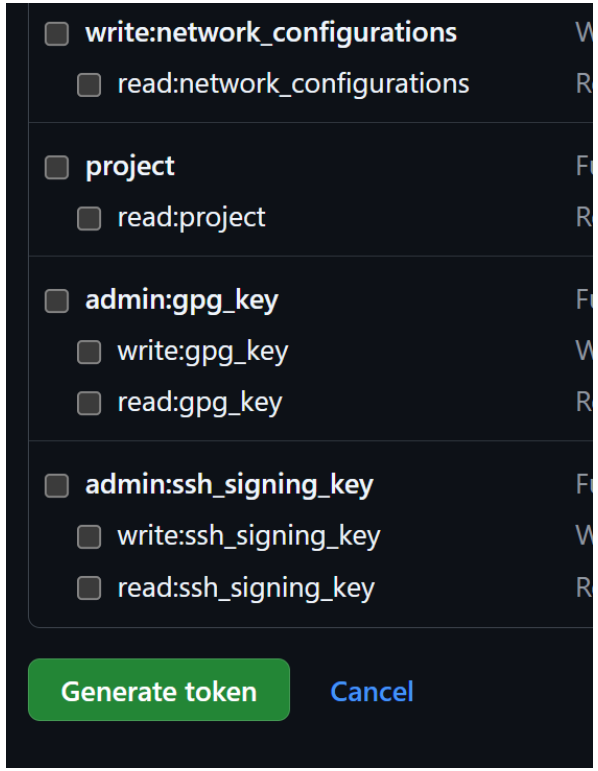

Personal access tokens (classic)
No personal access token created
Need an API token for scripts or testing? Generate a personal access token for quick access to the GitHub API.
Generate new token
GitHub API
Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.



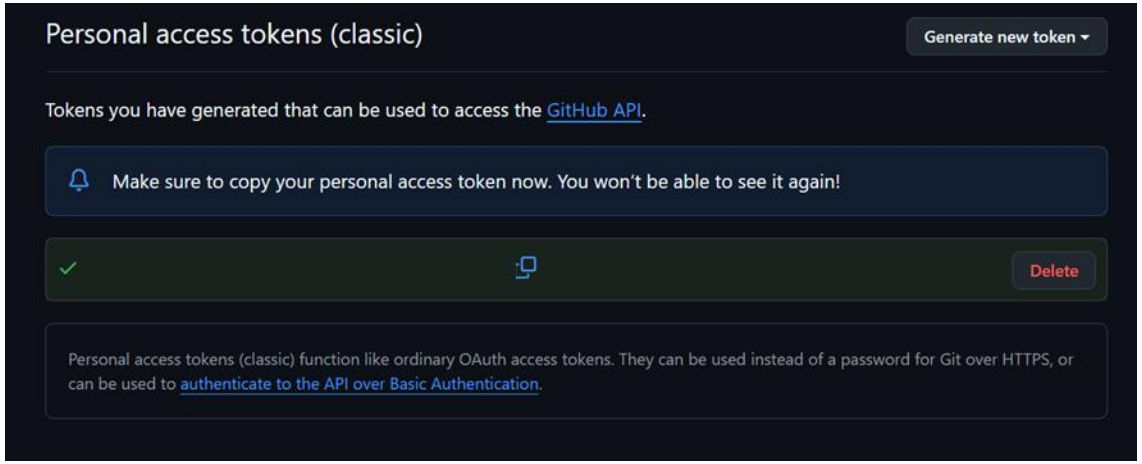
Note:寫一個給自己看的名字
權限設定先勾選repo即可。



按下 Generate token



複製生成的token，接下來會用到



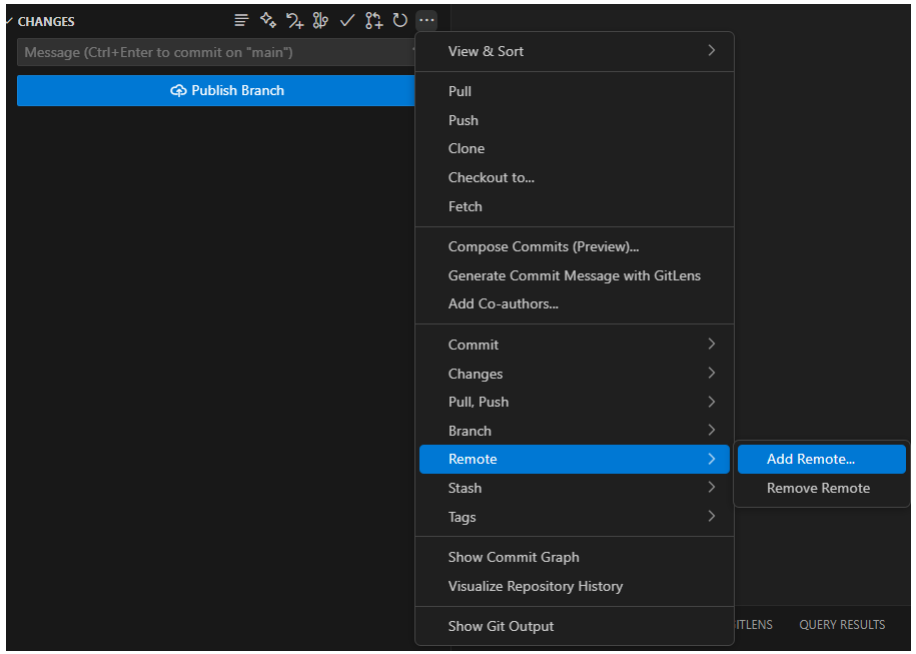
新增遠端儲存庫

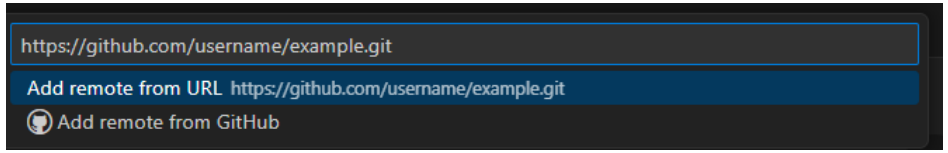
git remote add origin https://github.com/<user>/example.git
新增遠端儲存庫
git branch -M main
切換預設分支名稱（若安裝git時預設的Branch名稱就是main可以略過）

```
C:\Users\Student\example>git remote add origin https://github.com/<user>/example.git

C:\Users\Student\example>git branch -M main
```

VS Code GUI





git push

git push -u origin main
推送更新到遠端

```
C:\Users\Student\example>git push -u origin main
Username for 'https://github.com': <user>
Password for 'https://<user>@github.com':
```

成功

```
C:\Users\Student\example>git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 240 bytes | 240.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/<user>/example.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

第一次 push 時終端會要求輸入：
Username：GitHub 使用者名稱
Password：不是你的 GitHub 密碼，而是 Personal Access Token (PAT)

clone 與 pull

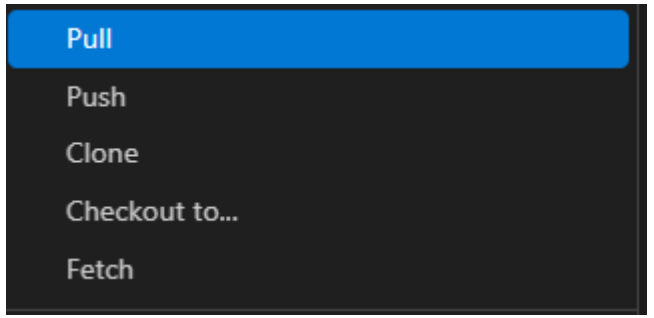
- git clone：第一次取得整個遠端版本庫。

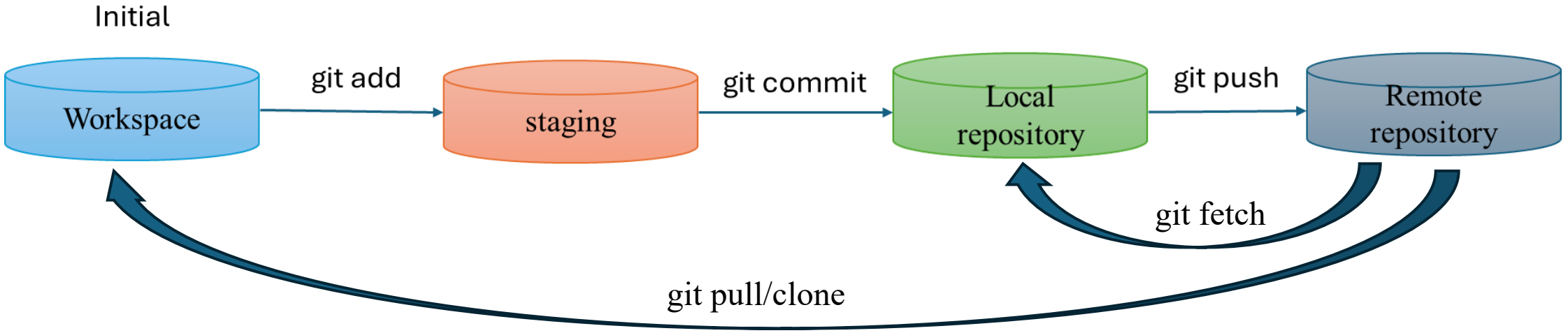
會建立專案資料夾、初始化 .git、下載歷史、設定 origin 遠端。

- git pull：已經有本地倉庫後，用來更新本地分支內容。
- clone 只需要一次；pull 是反覆多次的日常操作。

```
git clone <repository-url>
git clone https://github.com/user/project.git
```

```
git fetch
git pull
```

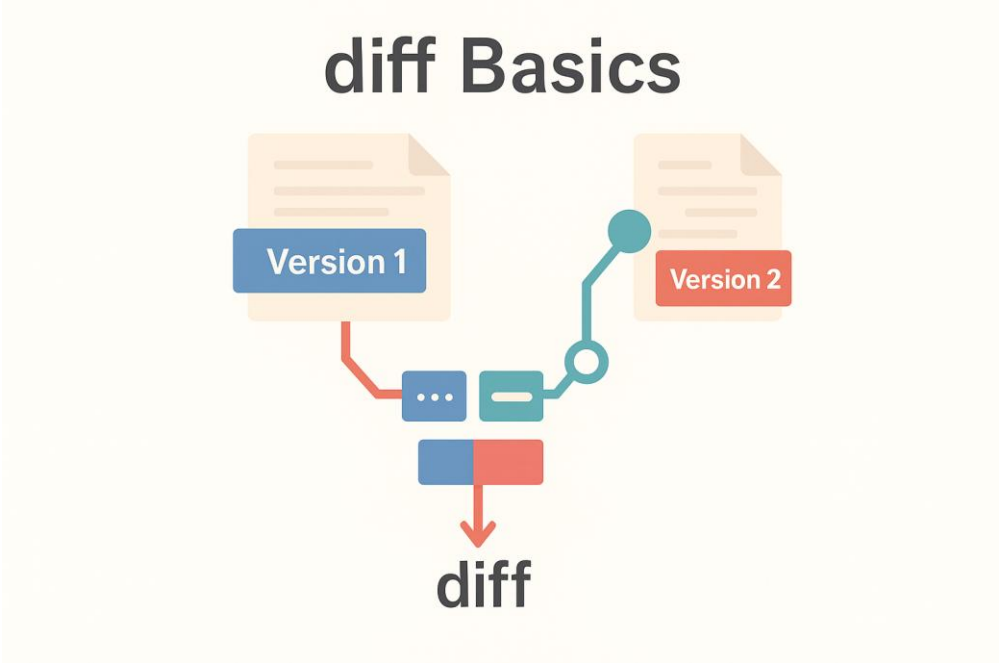




diff 基本概念

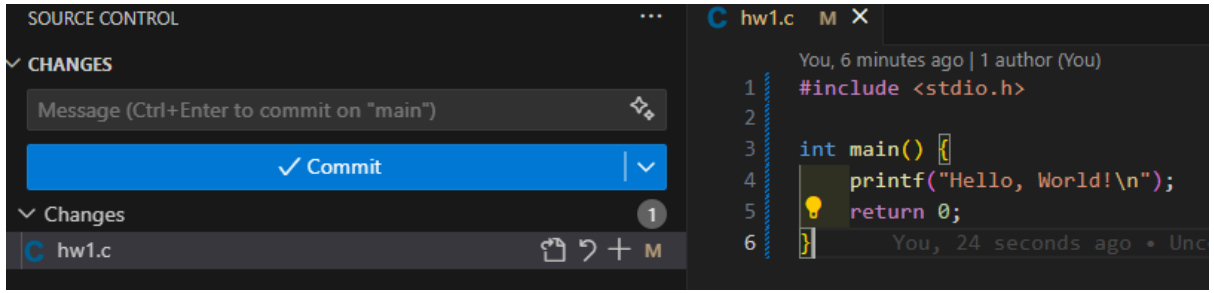
在程式更新的過程中可能會出現問題，這時可以使用diff來檢視不同版本之間的差異，快速找出問題點。

- diff：顯示兩個版本（或區域）間的差異
- 預設比較：工作區 vs 暫存區（未暫存的改動）
- 也可比較：暫存區 vs 最近一次提交 / 任意兩個 commits
- 目的：在提交前檢查是否正確、避免把 debug 內容送出



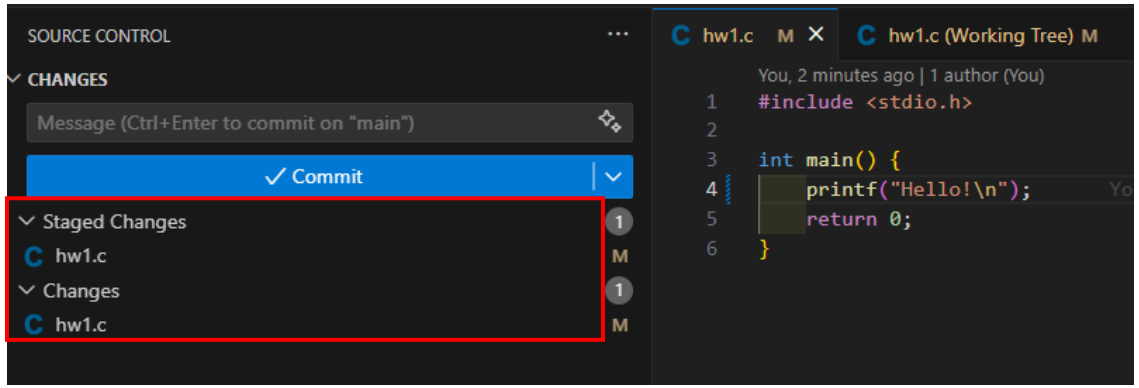
diff使用方式

先儲存一些東西到stagging



接著做一些更改

點擊檔案或是使用命令



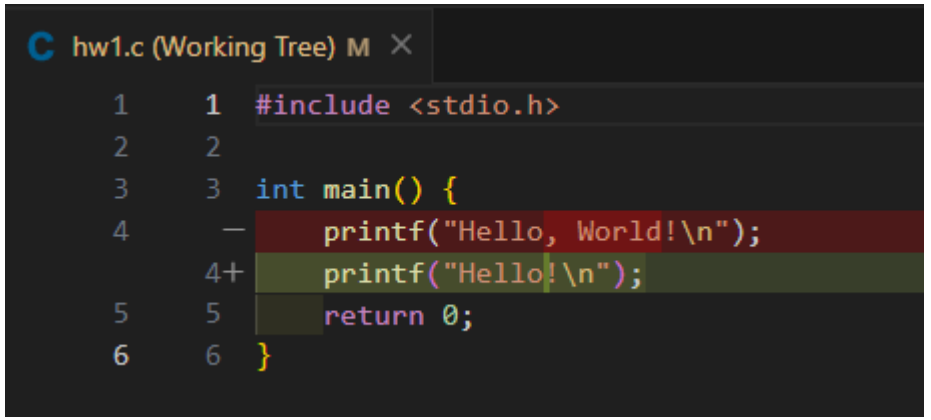
(這邊展示預設情況)

```
E:\GitExample>git diff
diff --git a/hw1.c b/hw1.c
index 8321e25..6134ec3 100644
--- a/hw1.c
+++ b/hw1.c
@@ -1,6 +1,6 @@
 #include <stdio.h>

 int main() {
-    printf("Hello, World!\n");
+    printf("Hello!\n");
     return 0;
 }
\ No newline at end of file
```

輸出符號：

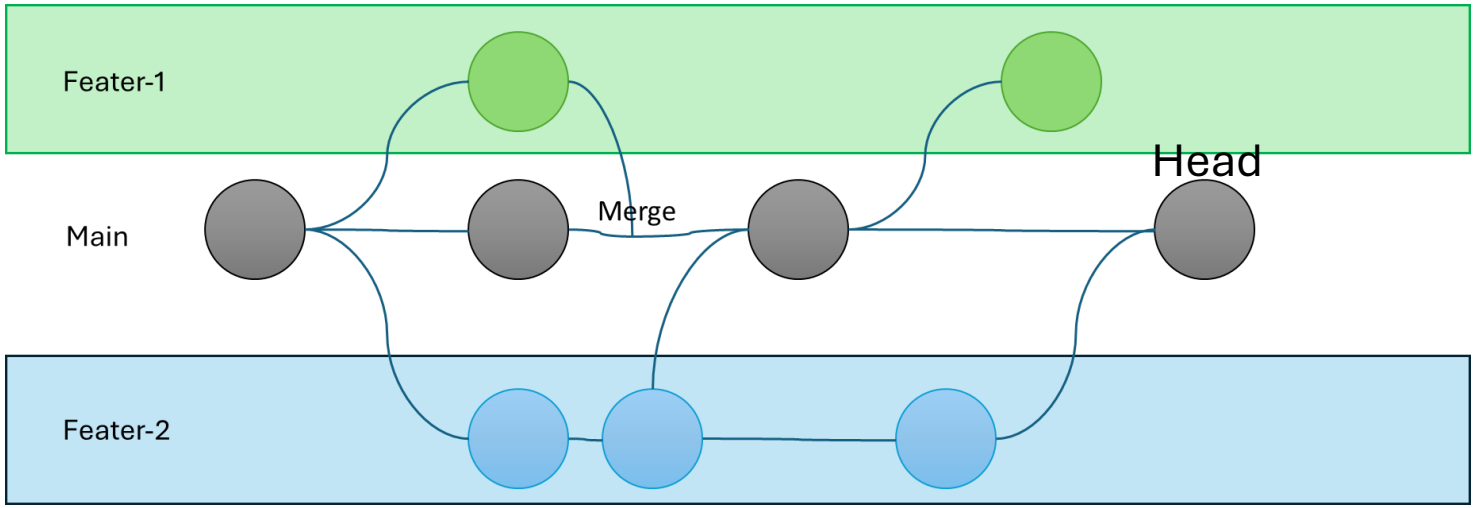
- \- 表示刪除、
- \+ 表示新增



Branch

Branch 基本概念

- 分支是指向某一串 commits 的「輕量指標」
- 讓你「平行開發 / 試驗」不破壞主線
- main穩定 / 可發布版本
- feature 分支：實作單一功能 / 修正
- 合併 (merge) 回主線後可刪除
- Head: 目前所在的位置(可以在任何地方)

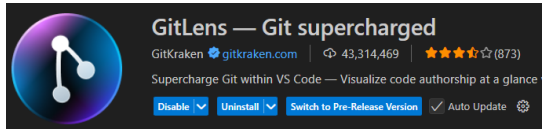


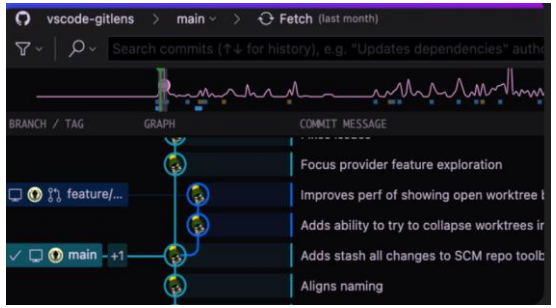
建立與查看分支

查看所有分支
git branch

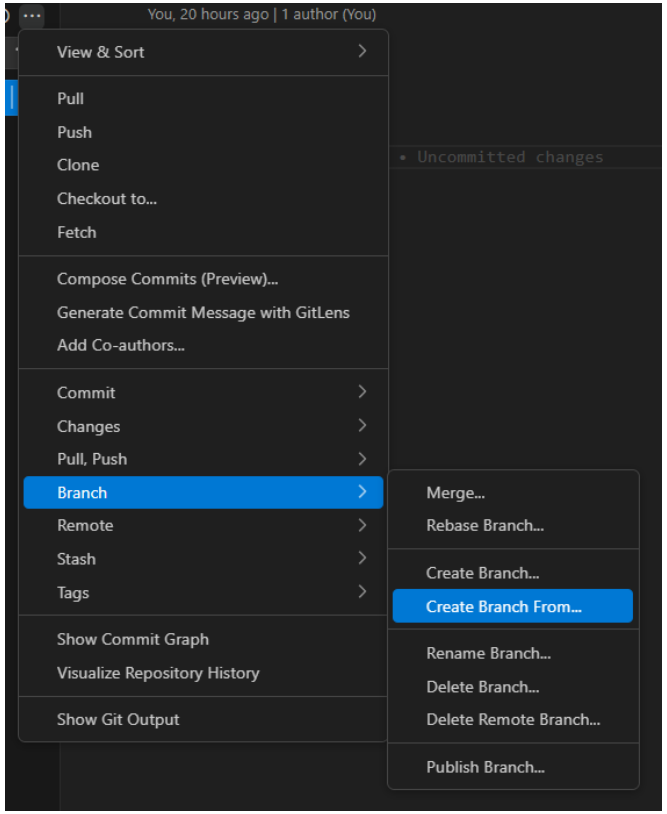
```
E:\GitExample>git branch
* main
```

推薦使用GitLens等視覺化工具來觀看



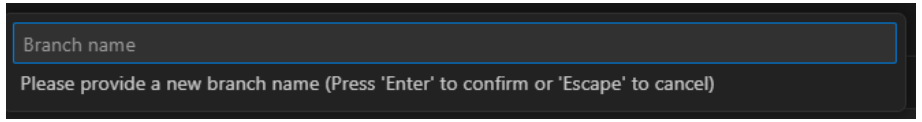


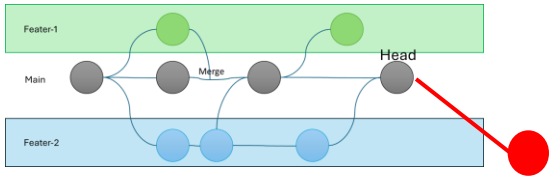
建立新分支



git branch <BranchName >
從當前的位置建立新分支

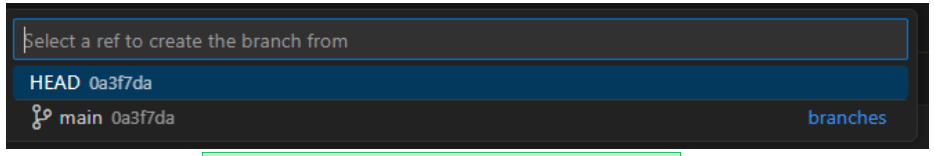
```
E:\GitExample>git branch feature-login
```

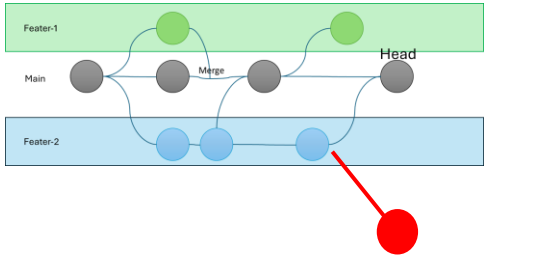




git branch <BranchName > 基底位置
從指定的位置建立新分支

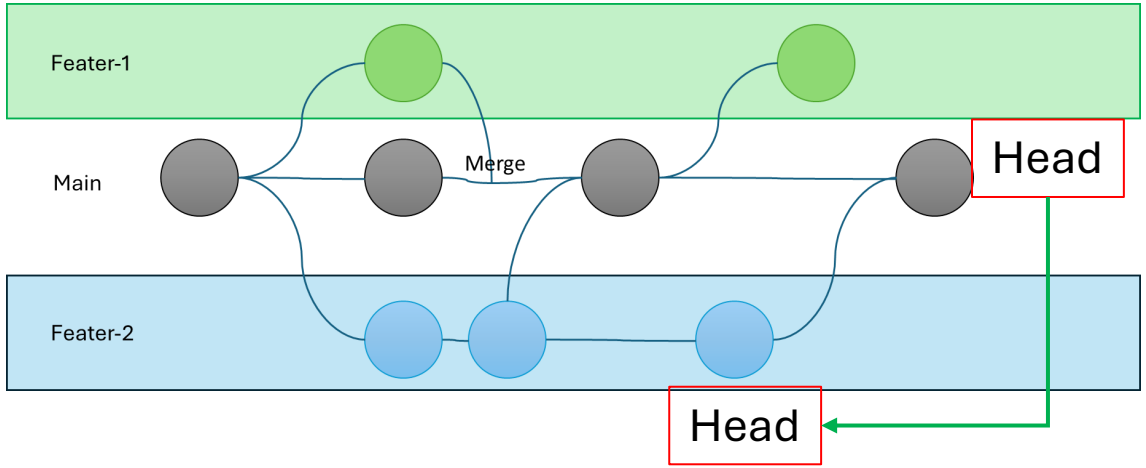
```
E:\GitExample>git branch feature-login 0a3f7dad895e65210a456a145090d631e310f91d
```





切換分支

- git switch main
- git checkout main（舊版但功能更多）

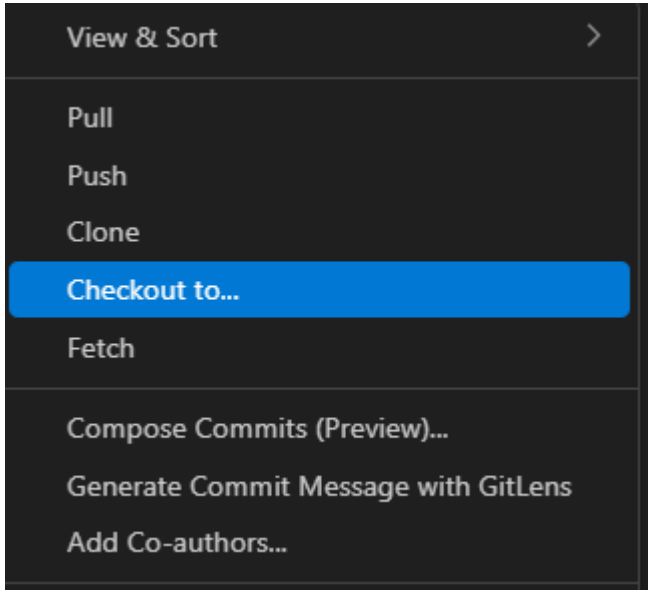


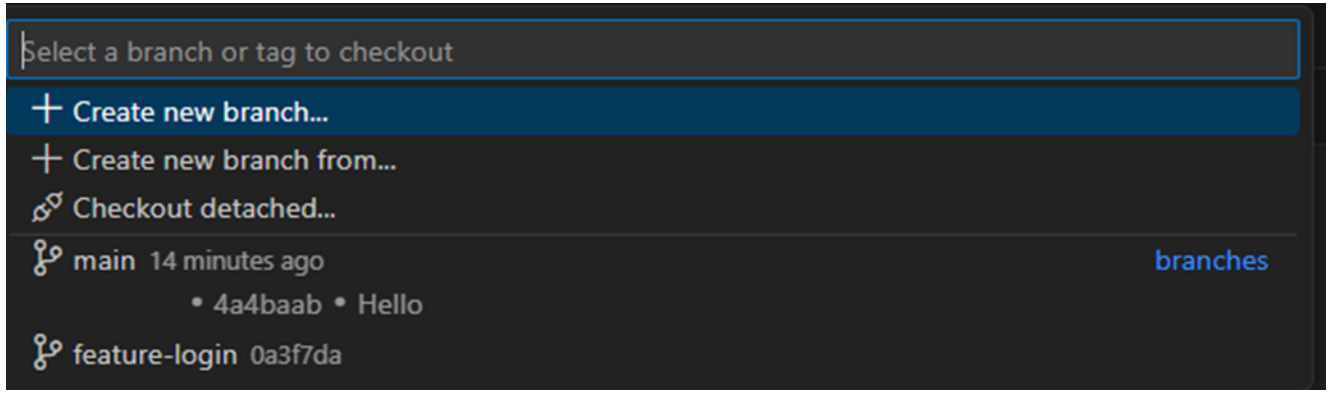
CMD

```
E:\GitExample>git checkout feature-login
Switched to branch 'feature-login'

E:\GitExample>git switch feature-login
Already on 'feature-login'
```

VS Code

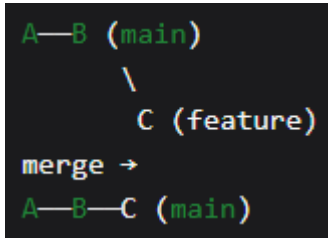




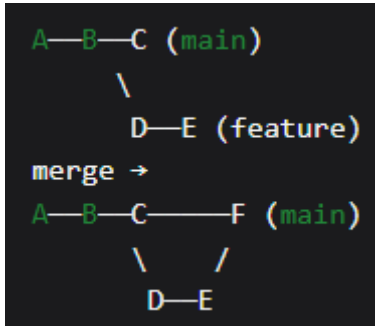
合併分支

在一個分支開發完成後需要將不同分支進行整合，這時會有兩種情況:

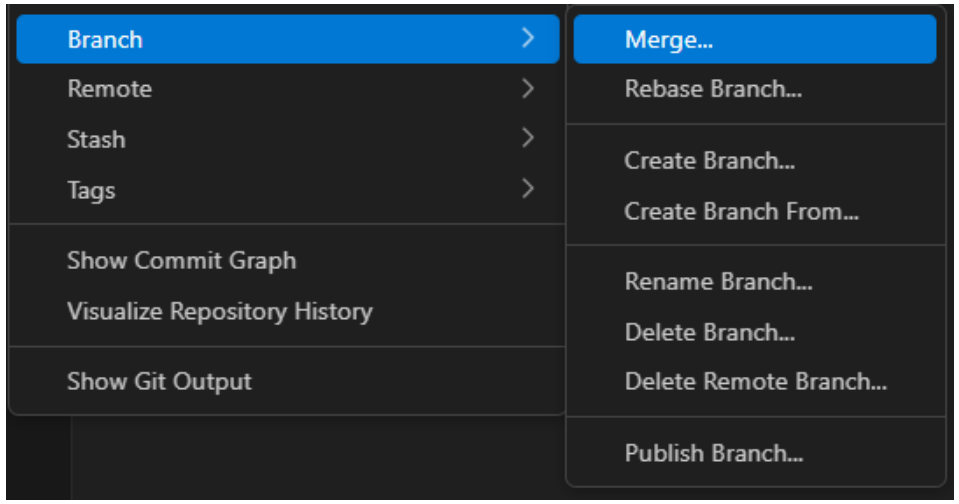
1.Fast-forward：主線未前進 → 直接指標前移

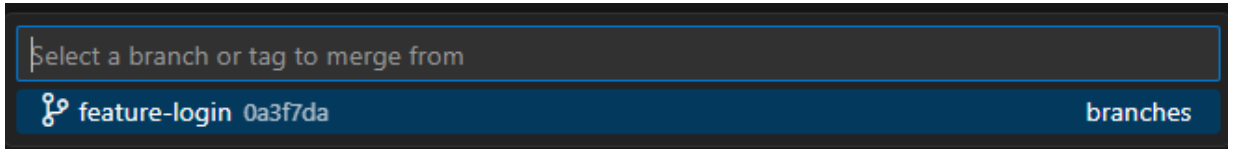


2.Merge commit：雙方各自前進需產生新節點



Rebase比較複雜這邊先跳過





合併流程

合併前：確保 main 更新

```
git switch main
git pull origin main
git merge feature-x
```

若同一行被兩邊改 → 衝突

•衝突標記：

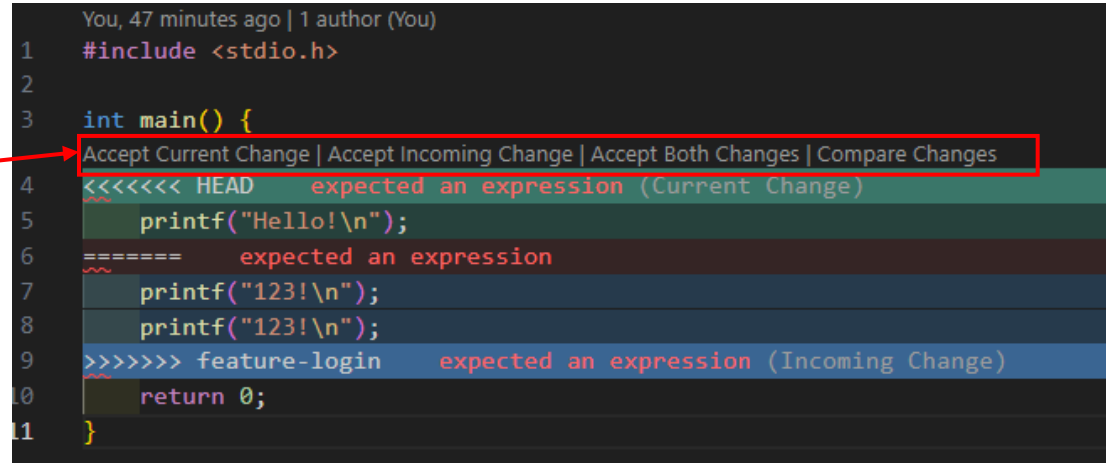
```
<<<<<<< HEAD
Line from main
=======
Line from feature
>>>>>>> feature-x
```

解決後：

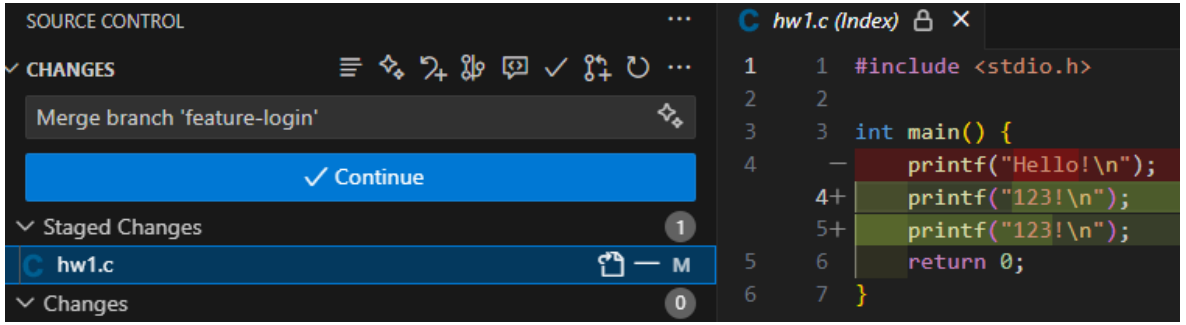
```
git add <file>
git commit
```

GUI

選擇這行要保留哪個版本



合併後查看差異並且commit更新



清理與刪除分支

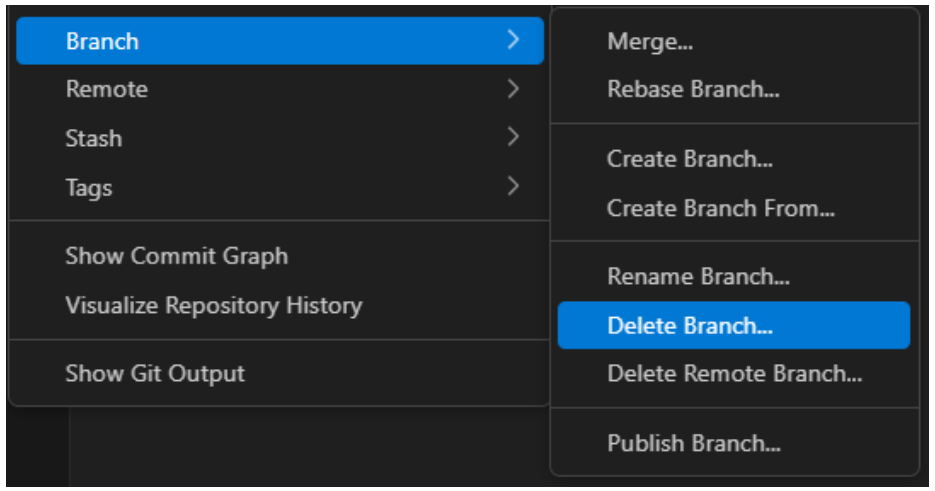
合併後可以考慮將不需要的分支刪除，避免「長期僵屍分支」

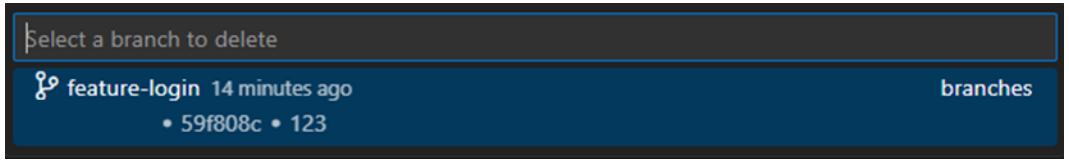
合併後本地刪除

```
git branch -d feature-x      # 安全刪除（已合併）
git branch -D feature-x      # 強制刪除（未合併，慎用）
```

遠端刪除

```
git push origin --delete feature-x
```

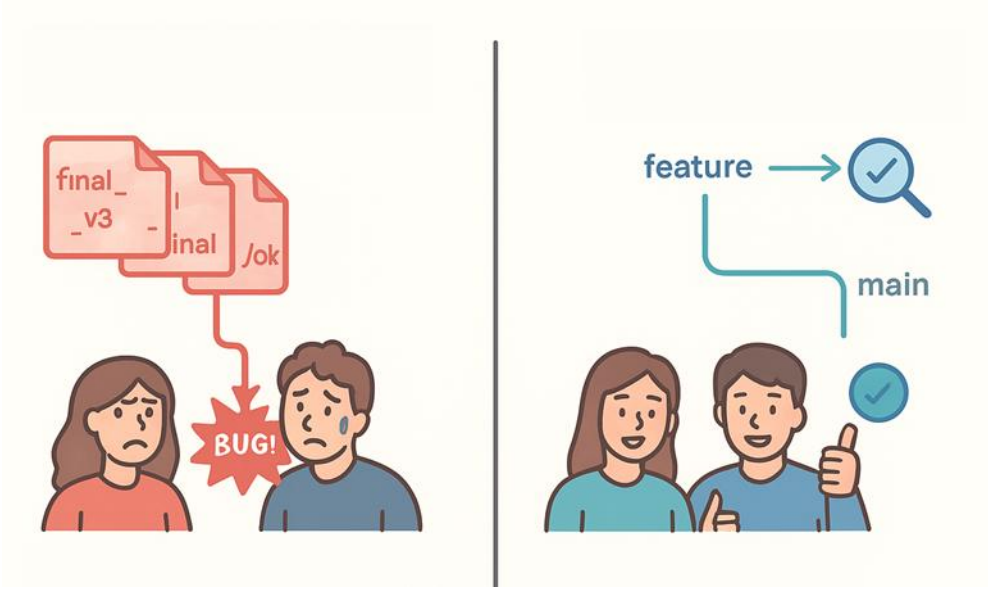




Workflow

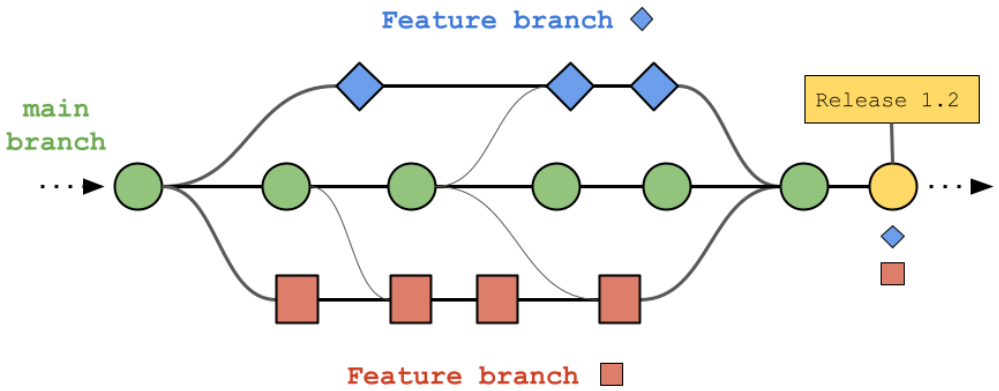
為什麼需要 Workflow?

- 降低衝突：每個功能在獨立分支中開發
- 清楚責任：誰做什麼一目了然（分支名稱與 PR）
- 便於審查：PR 讓同學/助教檢視品質
- 穩定主線：main 保持可執行 / 可發布
- 歷史乾淨：每個變更有清楚描述與紀錄
- 快速回溯：問題發生時可定位到單一功能分支
- 有助協作習慣養成：為將來實習 / 專題打基礎
- 避免「大家一直覆蓋最新檔案」的混亂

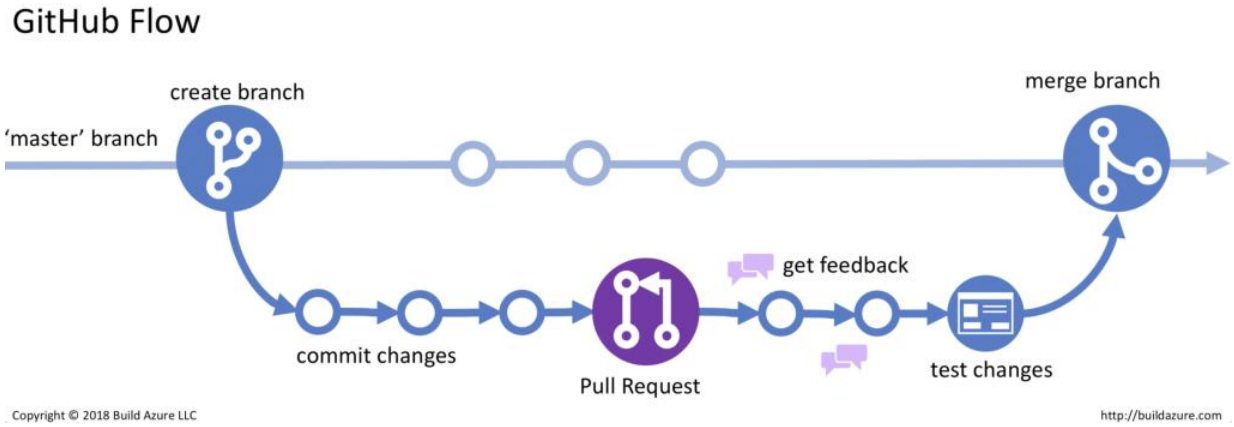


Workflow

- 單人：Main + Topic Branches
- 團隊常見：Feature Branch + Pull Request
- 其他模式：GitHub Flow / Git Flow
- 每種Workflow會有不同的Branch類型，在這之中命名通常為 Branch Type/Name



How to stop using feature branches - codefortynine GmbH

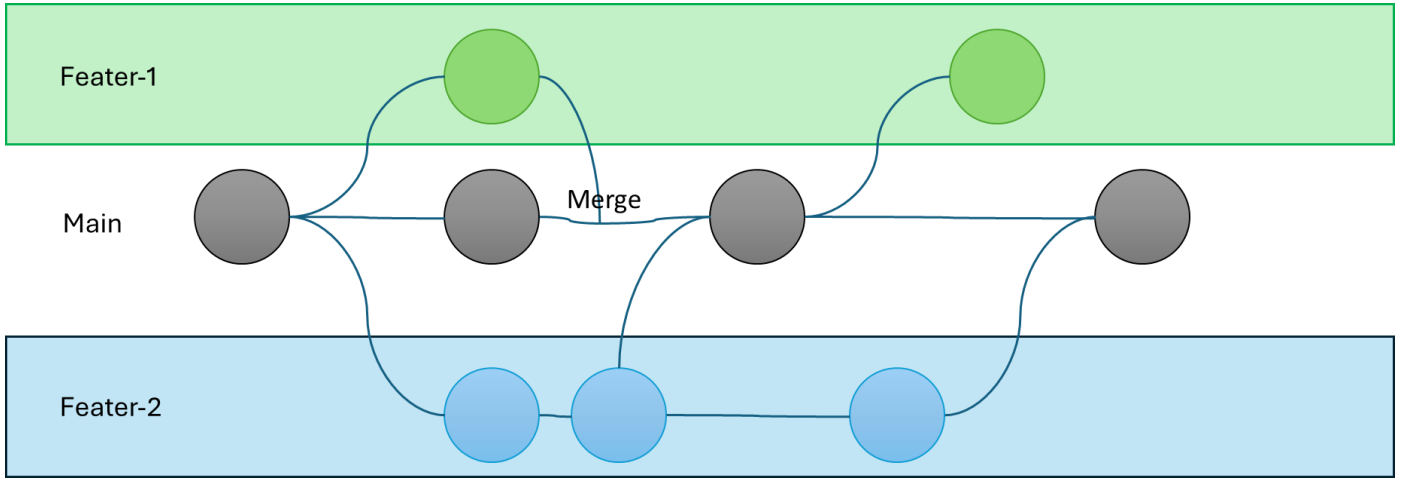


Introduction To GitHub And Git Version Control Workflow | Build5Nines

Feature Branch Workflow

- 最簡單且可擴充：main + 多個 feature 分支
- 每個功能 / 修正 = 一條短生命週期分支
- main 保持「可執行 / 可發布」狀態
- 入門 / 個人開發最佳起點

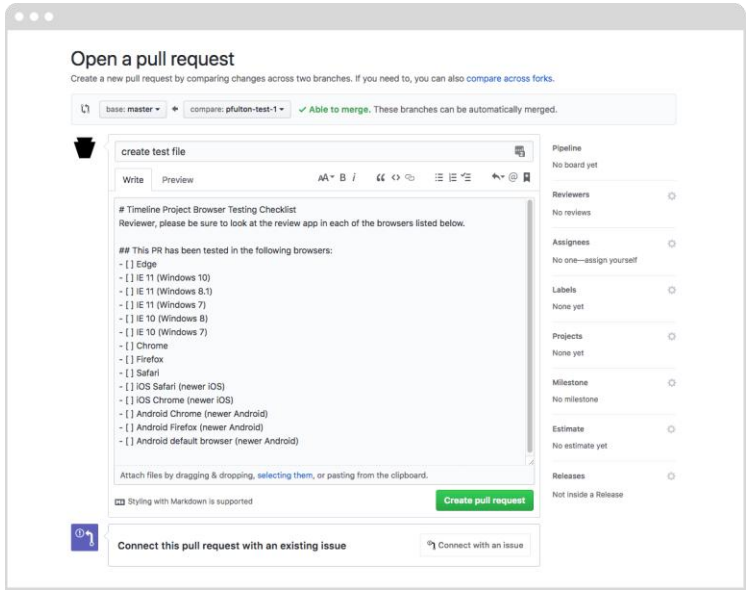
分支命名建議:
全小寫 + / 或 –
動詞或名詞組合，避免太籠統
（bad: update, good: feature/pdf-export）



Github flow

- 適合分組合作時使用
- 只有一個長期分支:master (main)
- 其他分支皆為短生命週期功能分支。
- 合併即部署，保持主幹永遠可發布

Pull Request（PR）用途
- 帶有上下文的變更提案
- 提供：變更摘要（What）、理由（Why）、測試方式（How to test）
- 作為：程式碼審查入口、討論紀錄、決策與責任追蹤



Why Your GitHub Pull Requests Are Taking Forever? | Axolo Blog

流程步驟 / Flow Steps

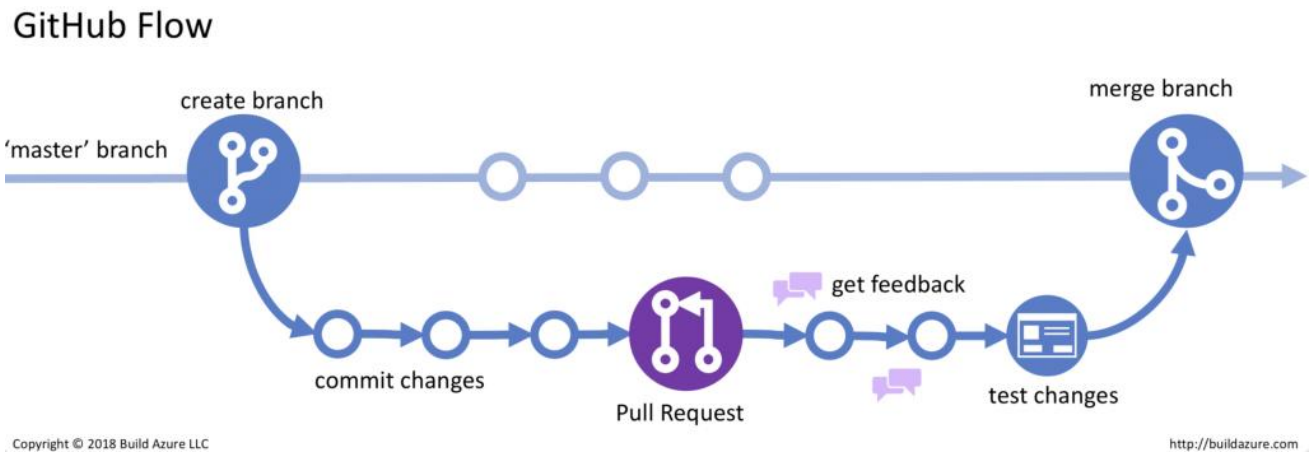
1.從 master 切出功能分支（feature/login, fix/timeout）

2.小步提交並頻繁推送；需要討論或完成主要功能時建立 PR

3.PR 進行討論 / 程式碼審查 / 自動測試（CI）

4.通過審查後合併回 master（通常保持快轉或使用 Squash）

5.部署最新 master；刪除已合併分支保持整潔

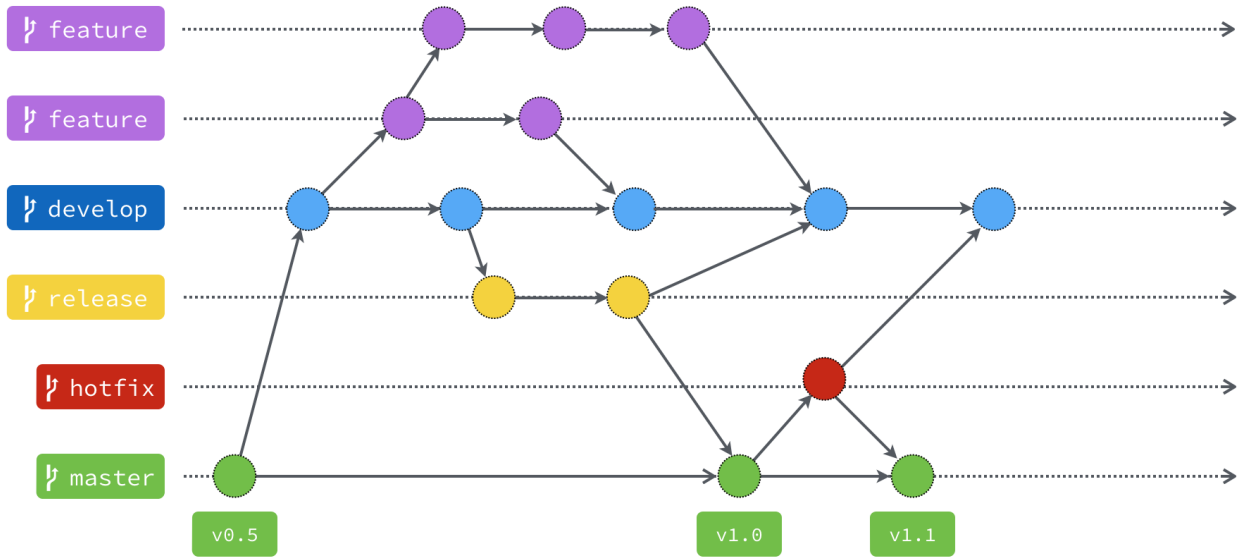


Introduction To GitHub And Git Version Control Workflow | Build5Nines

Git flow

在更大的團隊中會使用Git flow ，它提出不同的分支功能，分別有 **master、develop 、hotfix、release、feature** 五種分支。

Master:穩定、隨時可上線的版本(跟前面的main一樣)
Develop 分支:作為主要開發的分支，是所有短期分支的基礎分支。
Hotfix 分支:主要功能是用來進行**修復**
Release 分支:在 Develop 分支發布正式版本到 Master 分支之前，可以先進行一個預備發布的版本進行測試。
Feature 分支:主要用來新增功能的分支。

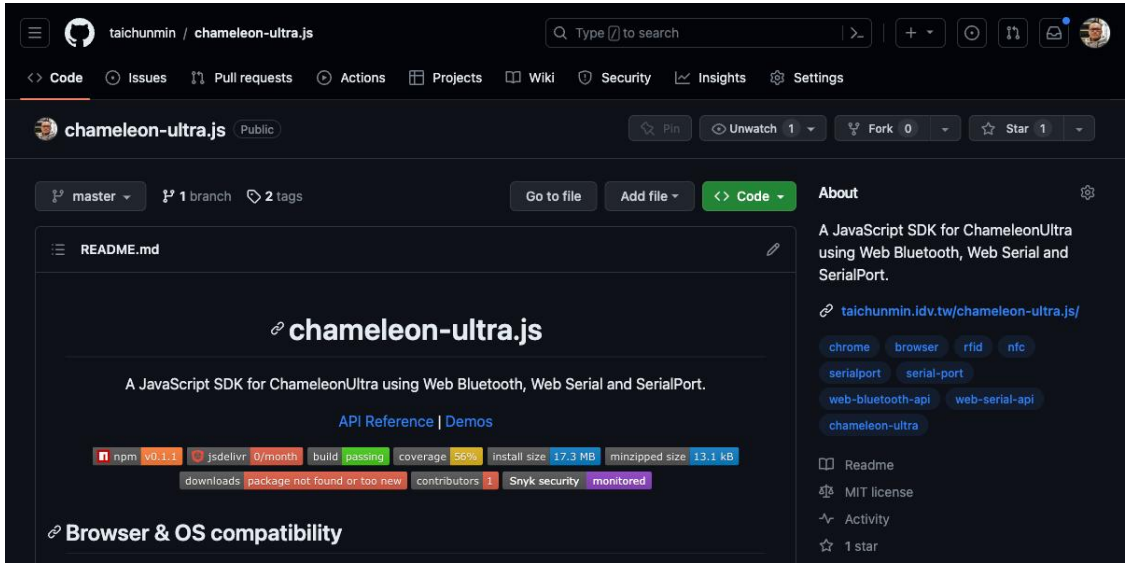


Git Flow 是什麼？為什麼需要這種東西？ - 為你自己學 Git | 高見龍

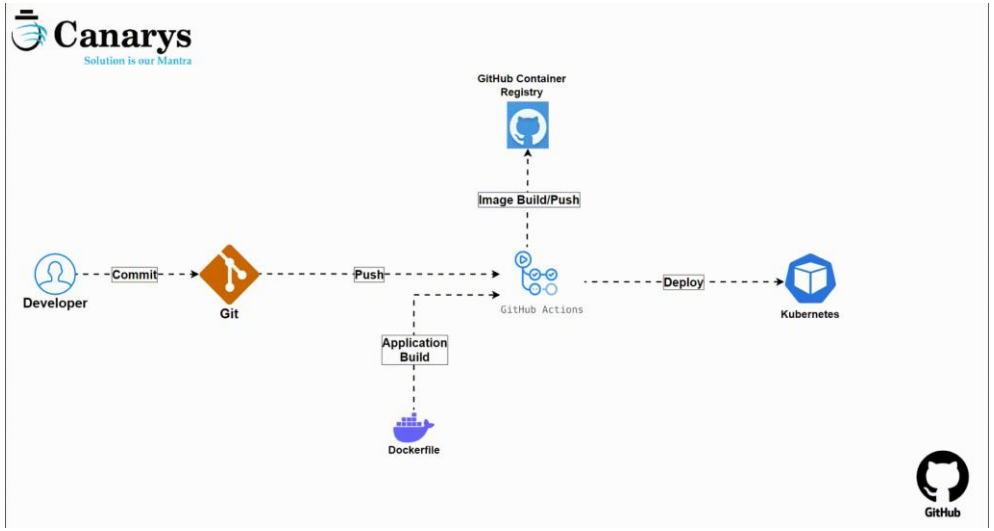
GitHub

Managing Your GitHub

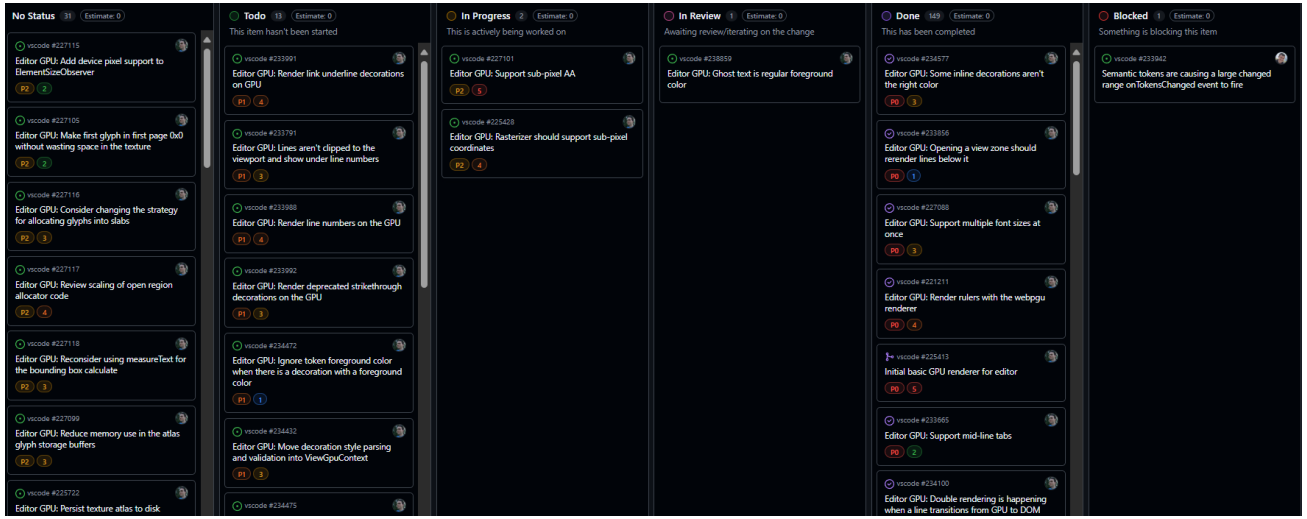
- 不只會用指令，也要「經營專案」
- 清楚的文件 = 降低溝通成本
- 規範與流程 = 減少摩擦
- 自動化 = 專注核心程式



Day 24 README.md - iT



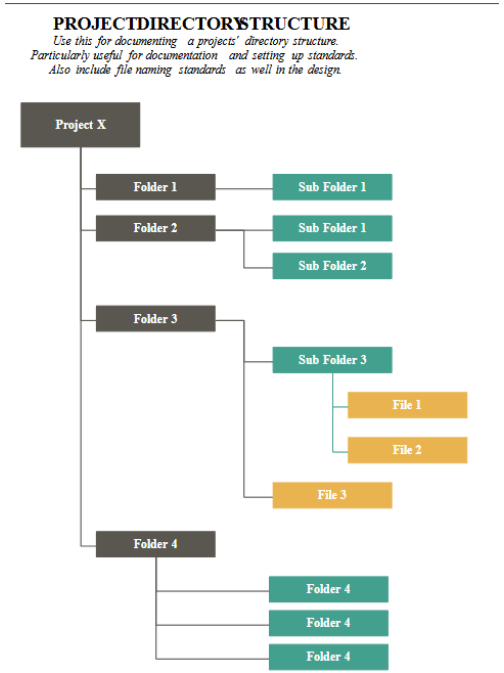
Streamline Your Docker Workflow with GitHub Actions - Canarys



Backlog · VS Code Editor GPU Renderer

儲存庫結構與命名

- 名稱短清楚
  （e.g. led-controller, signal-analyzer）
- README 放專案定位與快速開始
- 目錄有層次：src/ docs/ tests/ examples/
- 當需要不只一個檔案時就可以開始規劃
- 避免把巨大壓縮檔直接放 repo



Project Directory Structure | Creately

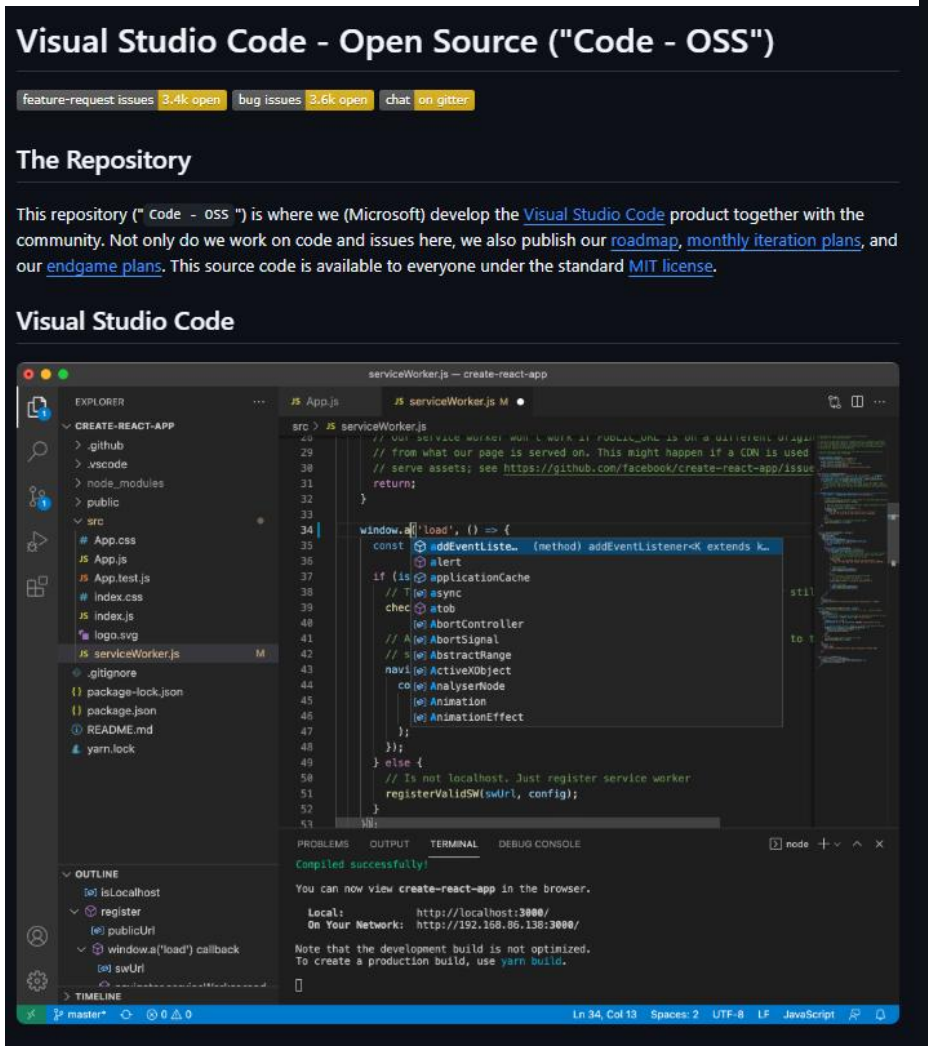
.config	chore: update baselines (#263626)	2 days ago
.devcontainer	fix: changed the node js version from 20 to 22 as per #25719...	last month
.eslint-plugin-local	Update target for build and test scripts	last month
.github	Reduce inclusion of telemetry instructions (#263685)	2 days ago
.vscode	Update notebook milestones (#263172)	5 days ago
build	Extract more types for reuse (#264032)	10 hours ago
cli	Fix --commit-id flag for code serve-web (#258904)	29 days ago
extensions	MSAL Redirect Funkiness (#264057)	8 hours ago
remote	chore: update electron@37.3.1 (#263552)	3 days ago
resources	fix: switch everyone to DEB822 (#260171)	3 weeks ago
scripts	Allow to hide Copilot via user setting and policy (fix #24961...	2 weeks ago
src	Fix disposal of chat session renderer (#264104)	2 hours ago
test	Have the lifecycle dependent on the Page since we only sup...	3 hours ago
.editorconfig	No forcing tabsize on users	7 years ago
.eslint-ignore	Fix eslint not linting our custom eslint rules	5 months ago
.git-blame-ignore-revs	eng - add ESM migration commit to `.git-blame-ignore-revs`...	last year
.gitattributes	Whitelist comments in all JSON files. For #129206	4 years ago

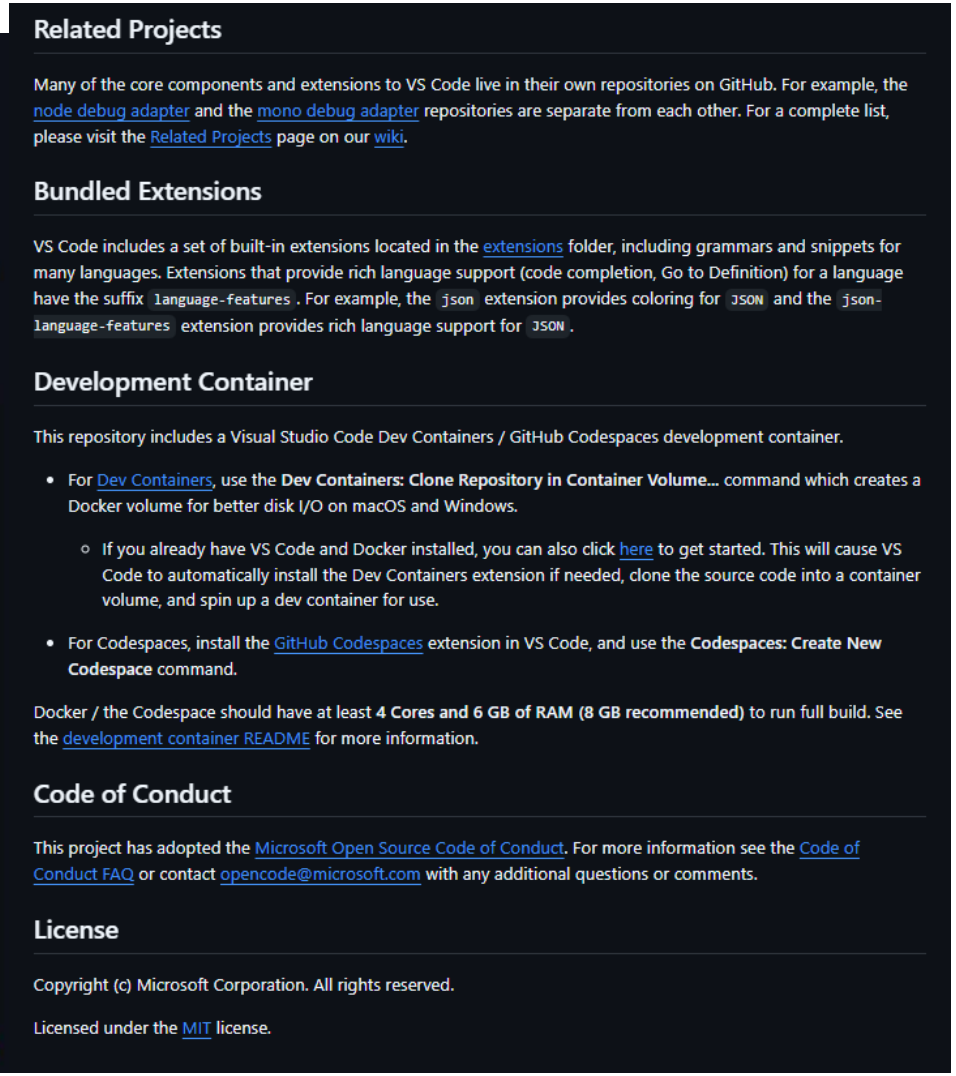
microsoft/vscode: Visual Studio Code

README 核心內容

這是一個專案的入口，在此應該要讓他人能夠快速理解這個專案
「在做什麼」、「如何使用」、「授權如何」

- 專案簡介
- 安裝與執行步驟
- 範例
- 目錄結構
- 貢獻方式
- 授權





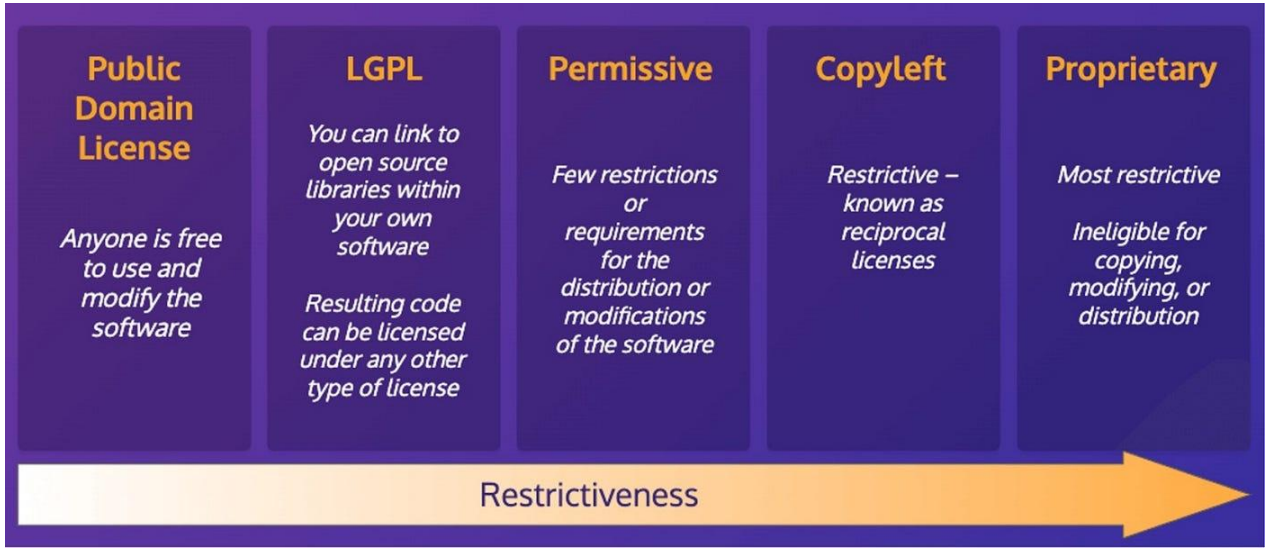
microsoft/vscode: Visual Studio Code

關於 LICENSE

在建構專案時也需要注意法律，以保障自身的權益，
在儲存庫添加LICENSE檔案，讓他人了解你希望如何授權。

- 沒有授權 = 預設「別人不可自由用」
- 常見：MIT（寬鬆）、Apache 2.0（含專利條款）、GPL（傳染性）
- 課堂 / 學習專案可用 MIT
- 放在根目錄檔名：LICENSE

6種常見的開源軟體授權條款解析



Software Licensing. What are Software Licenses? | by Udesh Indumina | Medium

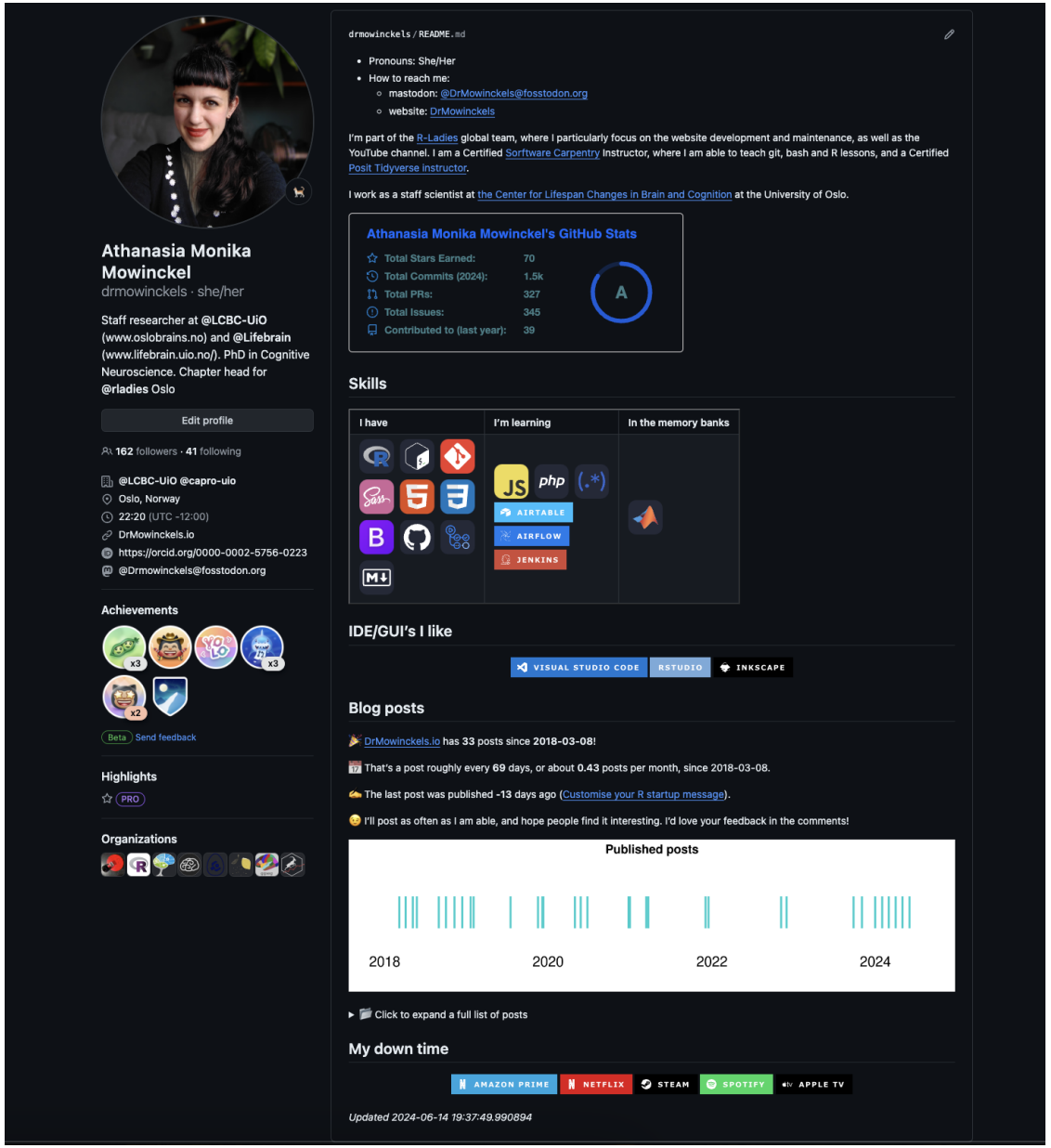
snyk	Copyleft					Permissive			
	GPL	AGPL	LGPL		MPL	Apache	MIT	BSD	Unlicense
Permissions in addition to commercial use, distribution, modification:									
Patent use	●	●	●	●	●	●	●	●	●
Patent use	●	●	●	●	●	●	●	●	●
Conditions									
Disclose source	●	●	●	●	●	●	●	●	●
License & copyright notice	●	●	●	●	●	●	●	Source	●
Network use is distribution	●	●	●	●	●	●	●	●	●
Same license	●	●	Library	●	File	●	●	●	●
State changes	●	●	●	Some	●	●	●	●	●
Limitations/Disclaimers									
Liability	●	●	●	●	●	●	●	●	●
Warranty	●	●	●	●	●	●	●	●	●
Trademark use	No explicit limitation				●	●	●	●	●

Open Source Licenses: Types and Comparison | Snyk

個人檔案經營

為何經營個人檔案

- 第一印象：如技術履歷首頁
- 幫助招生、實習、合作機會
- 展示程式實作深度與學習節奏
- 良好結構節省他人理解時間
- 不是炫酷堆砌，而是「清晰 + 可信」



lowlighter/metrics: 📊 An infographics generator with 30+ plugins and 300+ options to display stats about your GitHub account and render them as SVG, Markdown, PDF or JSON!

Improving your GitHub Profile - Dr. Mowinckel's

建立 Profile README

- 建立特別倉庫：名稱 = 你的 GitHub 使用者名稱
- 勾選 Initialize with README
- README.md 內容即成為個人首頁
- 可以使用模板快速建立

你好，我是 [你的名字 YOUR_NAME] 👋

一句定位（Tagline）

> [一句話定位你的技術焦點與價值，例如：專注低功耗 MCU 與可靠度測試的學生開發者]

👤 關於我（About Me）

- 目前專注：[例如：感測器整合 / RTOS 任務排程 / 前端可視化]
- 過去經驗亮點：[例如：課程專題建立自動測試流程，降低 30% 手動驗證時間]
- 動機 / 興趣：[例如：喜歡把硬體行為用程式模擬驗證]
- 正在尋找：[例如：尋找嵌入式實習 / 想參與開源驅動程式]

🧱 技術堆疊（Tech Stack）

語言：C（熟悉）、Python（熟悉）、Rust（入門）
平台 / 框架：STM32 HAL、ESP-IDF、FreeRTOS
工具：Git、GitHub Actions、PlatformIO、Docker
測試 / 品質：Unity C Test、pytest、clang-format
正在學習：Rust 嵌入式、硬體迴圈測試（HIL）

🚀 精選專案（Featured Projects）

1. [PROJECT_NAME_1]

一句價值：[例：模組化 I2C 感測器聚合：自動偵測 + 統一資料格式]
技術：C, FreeRTOS, I2C, PlatformIO
我的角色：[例：系統架構 + 裝置抽象層設計]
亮點：[例：抽象層降低新增新感測器所需程式變動行數 50%]
下一步：[例：加入離線快取 + 測試覆蓋報告]

2. [PROJECT_NAME_2]

一句價值：
技術：
我的角色：
亮點：
下一步：

3. [PROJECT_NAME_3]

一句價值：
技術：
我的角色：
亮點：

📫 目前學習與成長焦點（Learning Focus）

- 主題：[例：低功耗模式切換 & 電流分析]
- 方法：[例：使用電流探棒量測 + 腳本紀錄 + 比較不同策略]
- 成果（更新）：[例：已達成睡眠電流 < 15μA（原本 42μA）]

🗓️ 最近進展（Recent Updates）

- 2025-08：完成感測器自動註冊機制（Hash + 裝置表）
- 2025-07：導入 GitHub Actions 自動單元測試
- 2025-07：撰寫部落格：RTOS Task 排程抖動分析

🧪 實驗 / 探索（Experiments）（可選）

- [實驗名稱]：假設 → 方法 → 觀察
- 例：中斷優先權調整對資料延遲的影響：降低最高優先權頻率 → 平均延遲 -18%

🌱 開源參與（Open Source Contributions）（可選）

- PR：#1234 到 repoX 修正記憶體釋放
- Issue：#56 回報文件錯誤並提供補丁
- 計畫：尋找 first-timers-only 標籤練習

✍️ 文章 / 筆記（Posts / Notes）（可選）

- RTOS Task 排程量測方法 – 使用計數器 + GPIO 標記
- I2C 感測器抽象層設計筆記

🛠️ 常用工作流程（Workflows）（可選）

- Commit 策略：小步 + 類型前綴（feat:, fix:, refactor:, test:）
- 測試：push 觸發 CI、硬體迴圈測試手動標記
- 代碼品質：clang-format / 靜態分析（cppcheck）

📫 聯絡方式（Contact）

- Email：yourname (at) example.com
- LinkedIn：[...]/in/your-id
- 其他（可選）：個人網站 / 部落格

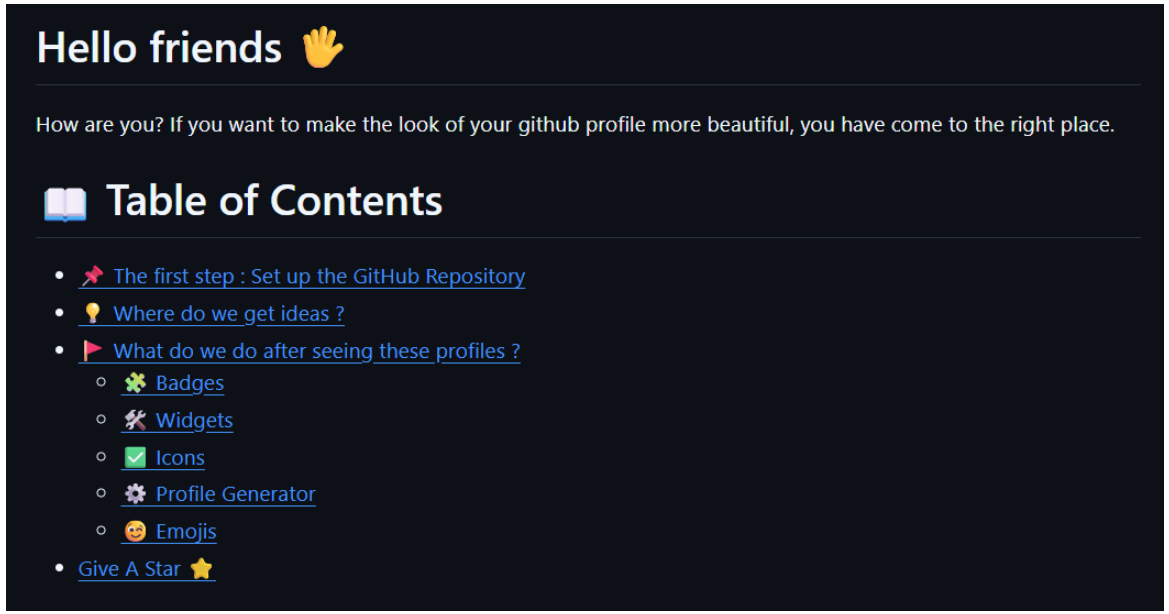
🧩 其他（Optional Extras）

build repo or workflow not found license repo not found

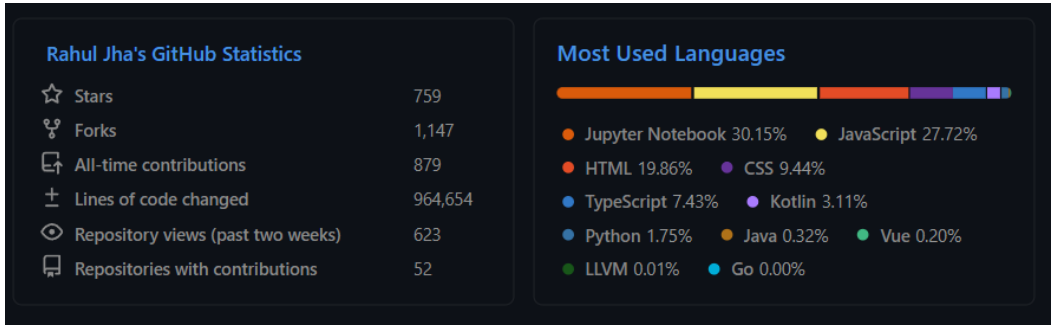
🔄 （可選）動態區塊（Dynamic Sections）

動態統計與小工具

beautify-github-profile 這是一個別人整理好的profile撰寫資源，裡面有很多可以使用的工具與模板，讓個人頁面更生動。



顯示使用的語言與一些數據等等



顯示活耀度