GITHUB COPILOT 實戰工作坊

Agent Mode × MCP × Agentic Workflows

120 分鐘,從「會用 Copilot 補程式碼」
升級成「會指揮 AI 代理完成整件事」。

完全動手做 · 不需程式基礎 · Copilot 免費版即可完成

ABOUT ME · 講師介紹

Duran Hsieh（謝政廷）



開發 · 測試 · 效能調教　GitHub Copilot　DevOps　Azure

專長於開發、測試、效能調教、GitHub、GitHub Copilot、DevOps、Azure 開發相關技術，曾擔任技術顧問，提供國內外企業技術諮詢與數位轉型。Study4TW 社群核心成員、微軟最有價值專家、GitHub Star、書籍作家、大學業界講師、技術部落客。

Duran Blogger　https://dog0416.blogspot.com

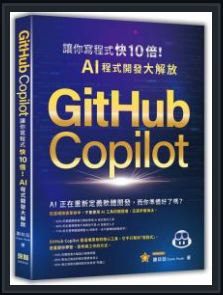
YouTube · FB　Duran 技術冶煉廠

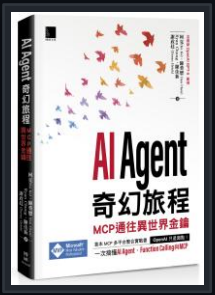
認證 · 社群

Microsoft MVP Most Valuable Professional

著作 BOOKS







下課時,你手上會有什麼

三樣留在你 GitHub 帳號上的東西

工作坊官方認證徽章

與 GitHub 官方協調發放，直接顯示在你的個人資料上，依完成登記名單發送。

GitHub Achievements

合併 PR 解鎖 YOLO，再修一個 issue 解鎖 Pull Shark。repo 必須是 Public。

一個公開的作品

GitHub Pages 網址加一份作品集頁，可以直接放進履歷。

完整條件、領取方式與疑難排解:docs/badges.md

今天怎麼跑

核心議程,190 分鐘

時間	內容	這一段要帶走的
13:10 – 13:40	演講一(30 分)	Octoverse 2026 開發者趨勢報告 三個觀念:Agent Mode、MCP、Agentic Workflow
13:40 – 15:40	實作(120 分)	Step 1 – 5 親手做一遍,含五題測驗與結業徽章
15:40 – 15:50	休息(10 分)	趁這段時間把卡住的人拉上來
15:50 – 16:20	自訂 Agentic workflow	Agentic Workflow 進階觀念
16:30 – 17:00	Q&A (爐邊會談)	討論

實作段的細部配速、每關預估時間與 agent 回合數,都寫在 AGENDA.md。

五關學習地圖

同一個專案,一路長大

STEP 1
代理模式
一段提示詞,從零生出三個檔案的待辦清單網頁。

STEP 2
多檔與還原
加深色模式與篩選;刻意弄壞,再用四種方法救回來。

STEP 3
MCP 整合
接上 Microsoft Learn 查文件,接上 GitHub 讀 issue。

STEP 4
代理工作流
寫一份劇本,讓 AI 自動讀 issue、修好、開 PR。

STEP 5
結業與獎勵
合併 PR 拿徽章、開 Pages、建立作品集頁。

每一關都有 solutions/ 對照解答:卡住超過 5 分鐘就複製,推關不中斷

章節一 · 技術一

Agent Mode

你給的是目標,不是步驟。

對應 Step 1 與 Step 2 · 預估 3 – 5 個 agent 回合

Copilot Chat 的三種模式

差別在:你把多少決定權交出去

模式	白話比喻	它會做什麼	它不會做什麼
Ask	問路人	回答問題、解釋程式碼、給你一段範例讓你自己複製	不會動你的檔案
Plan	請人幫忙規劃	通盤分析模糊的需求,拆解出多步驟的具體執行計畫	不會動你的檔案
Agent	請一組工班來施工	自己決定要看哪些檔案、建立與修改多個檔案、執行終端機指令、看到錯誤自己回頭修	—

切換方式:Chat 輸入框下方的模式下拉選單。選錯模式,後面什麼都不會發生。

參考:GitHub Docs — Copilot ask / edit / agent modes;VS Code Docs — Use chat in VS Code

Agent 的工作循環

它不是產生一段程式碼就結束



你要做的是在最後把關,而不是盯著中間每一步。

權限與把關點

它能自己做什麼,又會在哪裡停下來

它自己做

- 搜尋 workspace,自己決定要讀哪些檔案
- 建立、修改、刪除多個檔案
- 執行終端機指令、跑測試、看錯誤訊息
- 呼叫工具(內建工具與 MCP 工具)
- 發現結果不對,回頭再改一輪

它會停下來問你

- 要跑有風險的終端機指令時(Continue / Allow)
- 第一次呼叫某個 MCP 工具時的授權對話框
- 做完一輪之後的 Keep / Undo
- 你在劇本裡明寫「先問我再動手」的地方

這幾個按鈕就是你的煞車。不要習慣性連按。

VS Code 讓你設定核准層級:從每一步都要核准,到大部分自動放行

提示詞解剖

長提示詞比較省額度

【技術限制】

只用 HTML / CSS / 原生 JS,不要框架、
不要 package.json、不要 CDN

【檔案結構】

index.html / styles.css / app.js

【功能需求】

七條,每一條都可以用眼睛驗收

【介面需求】

置中、520px、卡片、RWD、CSS 變數

【其他】

註解用繁中,完成後告訴我怎麼打開

說清楚「不要做什麼」

禁用清單比功能清單更能省掉來回。

每一條都可驗收

寫得出驗收條件,agent 才知道什麼叫做完。

一次交辦清楚

與其問十次,不如一次講完。這既省額度,也是與 AI 代理協作的正確姿勢。

給它正確的上下文

Agent 不會通靈

#app.js

用井字號指名

在提示詞裡打 #檔名,明確指定要改哪一個檔。

拖曳檔案

從檔案總管直接拖進 Chat 輸入框,一次帶入多個檔案。

貼截圖

直接 Ctrl+V 貼上畫面。「長這樣不對」最快的說法。

模型怎麼選?保持預設就好

免費版會自動幫你選模型。這些任務它完全做得來,而且比較省額度 —— 不需要特地換成更貴的模型。

STEP 1 • 13:50 – 14:20 • 預估 1 – 2 回合

動手做:一段話生出一個網頁 App

操作順序

1. Ctrl + Alt + I 打開 Chat
2. 下拉選單切到 Agent
3. 模型保持預設
4. 貼上完整提示詞,按 Enter
5. 先不要動,看它跑 30 秒
6. 確認檔案都在,按 Keep
7. 對 index.html 按右鍵開來看

觀察重點

- 它先列出一份工作清單
- 檔案一個一個被建立出來
- 內容即時逐行寫入

驗收清單

新增、勾選加刪除線、刪除、未完成計數會動,按 F5 之後資料還在。

有哪一項不對?不要自己改 —— 在 Chat 裡叫 agent 修自己的 bug。

卡住超過 5 分鐘:cp solutions/step-1/*.* . 然後繼續往下走

章節二 · 技術二

Rollback

AI 一定會有做錯的時候。
敢放手的前提,是知道怎麼反悔。

對應 Step 2 · 我們會刻意把專案弄壞,再救回來

五層還原防線

由輕到重,對症下藥

層	情境	動作
1	剛改壞,還沒 commit	Chat 該回合上方的 Restore Checkpoint(VS Code 的自動快照,不動 git)
2	想丟掉所有未 commit 的變更	原始檔控制 → Discard All Changes,或 git restore . 加 git clean -fd
3	已經 commit、還沒 push	git reset --hard HEAD~1
4	已經 push 出去了	git revert <sha> 再 push。不要 force push,revert 對協作比較安全
5	全毀	從 template 重建一個 repo,複製 solutions/ 繼續

> 最好的還原,是動手前先 commit。每一關開始前先存個檔,你永遠回得去。

STEP 2 • 14:20 – 14:40 • 預估 2 – 3 回合

動手做:先蓋好,再拆掉,再蓋回來

A / B

一次改三個檔

先 commit 一個可用版本,再叫 agent 同時加上深色模式與篩選。Chat 裡會看到三個檔案的 diff。

C

故意把它弄壞

下一個很糟的指令:全部塞進單一 index.html、刪掉另外兩個檔、移除 localStorage。

D

挑一種救回來

Restore Checkpoint、Discard All、git restore、git reset。四種都能通,先試第一種。

> 加碼(E):讓 agent 讀你的 git 歷史與程式碼,自己寫出 CHANGELOG.md。

救不回來?cp solutions/step-2/*.* . 一樣可以推關

章節三 · 技術三

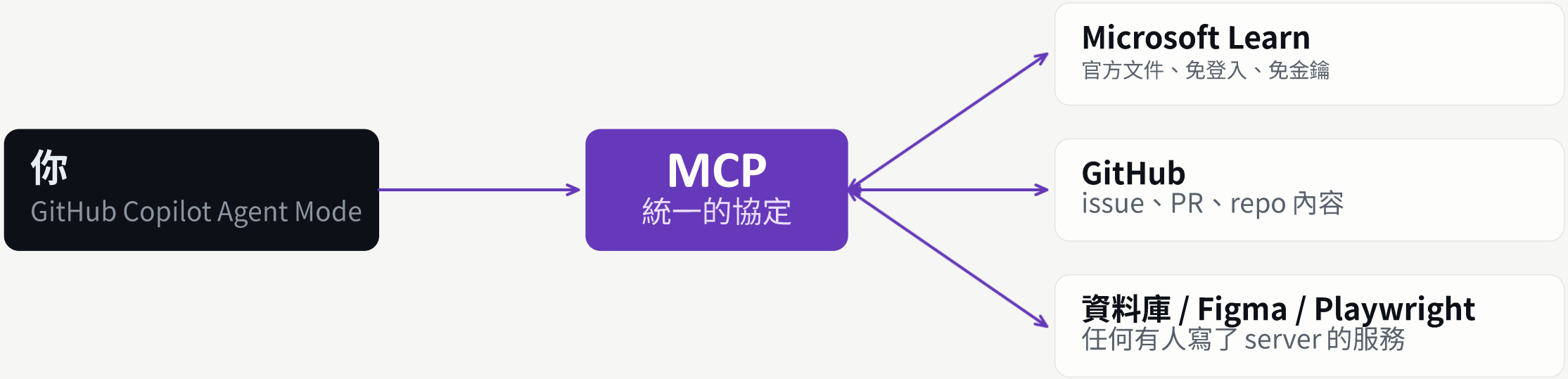
MCP

Model Context Protocol ——
讓 AI 走出這台電腦。

對應 Step 3 · 接 Microsoft Learn 與 GitHub 兩個 server

MCP 是什麼

一個通用接頭,常被形容成 AI 的 USB-C



以前每個 AI 工具接每個服務都要各寫一套整合。有了 MCP,服務方提供一個 server,所有支援 MCP 的工具都能用

三個名詞與兩種接法

聽得懂這三個字就夠用了

Server

提供能力的一方。例如 GitHub 官方的 server,讓 AI 能讀 issue、開 PR。

Client

使用能力的一方。這裡就是 VS Code 裡的 Copilot。

Tools

server 提供的一個個具體功能,例如 list_issues、create_pull_request。

接法	怎麼跑	今天用到的例子
本機 stdio	在你的電腦上啟動一個程序	常見於自架或社群 server,例如 Playwright
遠端 http	連到別人架好的網址,用 OAuth 或權杖授權	Microsoft Learn、GitHub —— 這兩個都是遠端的

每送出一則提示詞,Copilot 會把目前可用的工具清單一起交給模型,由模型決定用不用、用哪個

設定檔

整個設定就是這一個檔

.vscode/mcp.json

```
{
  "servers": {
    "microsoft-learn": {
      "type": "http",
      "url": "https://learn.microsoft.com/api/mcp"
    },
    "github": {
      "type": "http",
      "url": "https://api.githubcopilot.com/mcp/"
    }
  }
}
```

1. 存檔後按 Start

server 名稱那一行正上方會出現一排灰字。點 Start,狀態變成 Running。

2. 授權

GitHub 這個會跳出授權對話框,可能開瀏覽器確認一次

3. 確認工具載入

回 Chat 點工具圖示,看得到 server 底下的工具而且是勾選狀態。

沒看到 Start?

命令面板輸入 MCP: List Servers,選該 server 再 Start Server。

請認真看這一頁

MCP Server 會在你的電腦上跑程式

只裝信任來源

官方出品,或你自己看過原始碼的。來路不明的 server 等於把電腦交出去。

授權對話框要看

第一次呼叫某個工具時 VS Code 會問你。看清楚它要做什麼再按 Continue。

寫入操作特別小心

建立 issue、開 PR、刪檔案。只開你真正需要的工具,清單越小越安全。

參考:GitHub Docs — About Model Context Protocol (MCP)

STEP 3 • 14:40 – 15:10 • 預估 2 – 3 回合

動手做:讓 AI 現場去查官方文件

A • 免登入,30 秒

Microsoft Learn

叫 agent 去查 prefers-color-scheme 與深色模式的無障礙建議,然後回頭檢視我們的 styles.css。

觀察:Chat 裡出現工具呼叫區塊,回覆裡有真實的 learn.microsoft.com 連結。

B • 需要授權

GitHub

叫 agent 列出這個 repo 所有開啟中的 issue,整理成表格。

觀察:AI 現在讀得到你 GitHub 上的東西了 —— 這是 Step 4 的前置。

> 這一關只要 .vscode/mcp.json 存在就會通過檢查。兩個 server 沒有全接成功也沒關係,先往下走。

公司網路擋住?只留 microsoft-learn 也能推關,Step 4 有不用 GitHub MCP 的替代做法

章節四 · 技術四

Agentic Workflows

把會重複做的多步驟工作,
寫成 AI 照著執行的劇本。

對應 Step 4 · 讀 issue → 改程式 → 開 PR,一路到底

這場工作坊的完整公式

三個東西組起來,才叫工作流

規矩

copilot-instructions.md

這個專案的通則,每一次對話都自動帶入。

+

劇本

fix-issue.prompt.md

一個具體任務的完整步驟,打斜線就叫得出來。

+

手腳

MCP servers

讓它真的讀得到 issue、真的開得了 PR。

= Agentic Workflow

可版控 • 可 review • 全團隊共用

可以放進 repo 的 AI 資產

三種檔案,三種生效時機

檔案	角色	什麼時候生效
.github/copilot-instructions.md	規矩	每一次對話都自動帶入
.github/prompts/*.prompt.md	劇本	你在 Chat 打斜線加檔名時
.github/agents/*.agent.md	角色	你選用那個 agent 時

為什麼要進 repo

寫在聊天記錄裡的提示詞是你一個人的;寫成檔案的提示詞是整個團隊的，而且可以 code review、可以逐版改進。

生態還在動

VS Code 近期也支援 AGENTS.md 這類跨工具的指示檔，以及把能力封裝成 skill 的寫法。名字會變，概念不變。

fix-issue.prompt.md 拆解

一份劇本裡到底寫了什麼

01
讀取 issue
用 GitHub MCP 讀 issue，摘要成中文

02
提計畫並停下
在我回覆同意前,不要動任何檔案

03
建立分支
git switch -c fix/issue-N

04
進行修改
遵守規矩檔，只改必要的檔案。

05
說明驗證方式
怎麼操作、看到什麼,才算真的修好。

06
提交並推送
commit 訊息要帶 issue 編號。

07
建立 Pull Request
內文要有 Closes #N，最後把網址貼給我。

前置設定用 YAML 標頭寫:

agent / description
argument-hint

這個檔案請手動建立並貼上 —— 裡面的參數語法是給 VS Code 讀的,agent 常會好心把它改掉

值得寫成劇本嗎

差別不在能不能做,在第二次

	一般提問	Agentic Workflow
形式	每次都重打一次提示詞	存成檔案,打斜線就叫得出來
一致性	每個人問法不同,結果不同	全團隊跑同一套流程
範圍	通常只做一個動作	讀 issue、改程式、開 PR,一路到底
可版控	在你的聊天記錄裡	在 repo 裡,可以 review、可以改進

> 判斷準則:會重複做、步驟固定、每次都該長一樣的工作,才值得寫成劇本。一次性的探索不用。

STEP 4 • 15:10 – 15:30 • 預估 2 – 3 回合

動手做:一行指令,AI 自己開 PR

```
/fix-issue issueNumber=3
```

- 先確認 GitHub MCP 的工具已勾選
- 到 Issues 頁籤挑一個 issue,記下編號
- 它會停下來問你同不同意計畫
- 回「同意,請開始」,然後放手讓它跑
- 打開它給你的 PR 網址

停下來想一下你做了什麼

你寫了一份 Markdown 檔,打了一行指令。

AI 自己讀了 GitHub 上的 issue、理解需求、改了程式碼、建了分支、推上去、開了一個帶 Closes #N 的 Pull Request。

而這整套,是可以版控、可以 code review、整個團隊共用的。

GitHub MCP 沒接成功也能過:把 issue 內容直接貼給它,推關只檢查檔案是否存在

章節五

結業與獎勵

最後十分鐘，
把今天的努力變成帶得走的東西。

對應 Step 5 · 15:30 – 15:40 · 這一段絕對不能砍

STEP 5 • 四件事,十分鐘

把成果變成你的作品

順序	做什麼	拿到什麼
A	合併 Step 4 那個 PR	解鎖 YOLO;再修一個 issue、合併第二個 PR 解鎖 Pull Shark
B	Settings → Pages 開啟	一個任何人都打得開的公開網址,可以放進履歷
C	讓 agent 寫 PORTFOLIO.md	一份專業的作品集說明,含線上展示、功能、技術、開發方式
D	填完成登記表	工作坊官方認證徽章 —— 沒登記就拿不到

> **先確認三件事,不然徽章拿不到**
>
> repo 設為 Public(GitHub Achievements 只計算公開 repo)・
> PR 要合併進 main ・ 個人資料勾選 Show Achievements on my profile。

額度與配速

為什麼我們要數 agent 回合

12

全程 agent 回合上限

Step 1 約 1 – 2
Step 2 約 2 – 3
Step 3 約 2 – 3
Step 4 約 2 – 3
Step 5 約 1

免費版是有額度的

Copilot Free 的程式碼補全與對話用量都有上限
2026 年 6 月起所有方案改採 AI Credits 計量
課程設計成免費版全程做得完

長提示詞是刻意的

一次把需求交辦清楚，而不是來回問十次。
這既省額度，也是與 AI 代理協作的正確姿勢。

額度用完也不會卡住

每一關都有 solutions/ 對照解答，複製檔案一樣推關，學習不中斷。

學生或教師記得先申請 GitHub Education,可免費升級 Copilot Pro,額度充裕很多

現場自救

卡住了怎麼辦

狀況	去哪裡找答案
AI 把程式改壞了,想還原	docs/rollback.md —— 五層還原防線與指令速查
git push 被拒絕、issue 沒出現、關卡沒推進	docs/troubleshooting.md;也可到 Actions 手動執行 Step workflow 強制推關
Copilot 額度用完了	solutions/ —— 每一關的完整解答,複製就能過關
想確認自己學會了	docs/quiz.md —— 五題測驗與詳解
徽章沒出現、想知道怎麼領	docs/badges.md —— 徽章與成果指南

> **現場最常出問題的兩件事**
> 每次 push 前先 git pull --rebase(機器人會 commit 回 main)· 卡住超過 5 分鐘就跳關，不影響學習。

三句話帶回公司

如果今天只記得三件事

01 Agent Mode 不是自動完成的升級版。它會自己開檔案、跑指令、看結果、自我修正。你給的是目標,不是步驟。

02 MCP 讓 AI 走出這台電腦。沒有 MCP,它只看得到你的檔案。

03 會重複做的流程,寫成劇本放進 repo。規矩 + 劇本 + 手腳 = Agentic Workflow。

第四件事:動手前先 commit。敢放手的前提,是知道怎麼反悔

回去之後看這些

官方文件與教材入口

GitHub Copilot 官方文件

Copilot 的 ask / edit / agent 模式
Extending Copilot Chat with MCP servers
Setting up the GitHub MCP Server
Plans for GitHub Copilot

docs.github.com/copilot

VS Code 文件

Build with agents in VS Code
Add and manage MCP servers
MCP configuration reference
Agent customization(指示檔與劇本)

code.visualstudio.com/docs

本教材

AGENDA.md — 議程與配速
docs/00-setup.md — 環境檢查
docs/rollback.md — 五層還原
docs/quiz.md — 五題測驗
docs/badges.md — 徽章指南

MIT License · 歡迎取用改作

別忘了填完成登記表 —— 那是拿到工作坊官方認證徽章的唯一途徑