

# SciCover

## 部署與操作完整教學

雙語科普期刊摘要平台 — 從零開始到完整上線

項目	說明
版本	v1.0
更新日期	2026-02-28
適用對象	開發者、研究者、科技愛好者
前置需求	GitHub 帳號、Node.js 20+、Python 3.12+
部署成本	\$0（全部使用 GitHub 免費額度）

# 目錄

## 目錄

2

### 第一章：專案總覽

15

SciCover 是一個開源的科研知識轉化平台，它自動追蹤頂級科學期刊（Science、Nature、Cell）的封面文章，透過 AI 進行深度解讀，生成符合中英雙語母語習慣的科普摘要。

15

#### 1.1 架構說明

15

本專案採用「All-in-GitHub」架構，無需租用伺服器或資料庫：

15

#### 1.2 專案目錄結構

15

scicover/

15

├── .github/workflows/ # GitHub Actions 自動化工作流

15

| └── scrape-and-summarize.yml # 每週自動爬取 + AI 處理

15

| └── deploy.yml # 自動部署到 GitHub Pages

15

├── frontend/ # React + Vite 前端應用

15

| └── src/components/ # UI 元件

15

```
15 |   |—— src/hooks/           # React Hooks
15 |   |—— src/lib/             # 型別定義、常數
16 |   |—— src/styles/          # CSS 設計系統
16 |—— scripts/                 # Python 後端腳本
16 |   |—— scraper/             # 期刊爬蟲
16 |   |—— ai/                   # DeepSeek V3 摘要引擎
16 |   |—— pipeline/            # 資料管線編排器
16 |—— data/                    # 靜態 JSON 資料儲存
16 |—— images/                  # 封面圖片
```

## 第二章：環境準備

17

在開始之前，請確認您的電腦已安裝以下工具：

17

### 2.1 驗證安裝

17

在終端機（Terminal）中執行以下指令，確認各工具已正確安裝：

17

```
git --version    # 應顯示 git version 2.x.x
```

17

```
node --version    # 應顯示 v20.x.x 或以上
17
```

```
python3 --version # 應顯示 Python 3.12.x 或以上
17
```

```
npm --version     # 應顯示 10.x.x 或以上
17
```

## 2.2 註冊 GitHub 帳號

17

如果您尚未擁有 GitHub 帳號，請前往 [github.com](https://github.com) 註冊。後續所有操作（包括程式碼儲存、自動化流程、網站部署、AI API 調用）都依賴 GitHub。

17

## 第三章：本機開發設定

18

### 3.1 解壓專案

18

Step 1 解壓下載的 scicover.zip

18

```
# 將 scicover.zip 解壓到您想要的位置
18
```

```
unzip scicover.zip
18
```

```
cd scicover
18
```

### 3.2 安裝前端依賴

18

Step 2 安裝 Node.js 套件

18

```
cd frontend
18
```

```
npm install
```

18

此指令會根據 package.json 安裝所有前端依賴，包括 React、React Router、TanStack Query、Framer Motion 等。

18

### 3.3 設定資料連結

18

#### Step 3 建立資料目錄的符號連結

18

前端開發伺服器需要能存取 data/ 和 images/ 目錄中的 JSON 檔案和圖片：

18

```
# macOS / Linux
```

18

```
ln -s ../data public/data
```

19

```
ln -s ../images public/images
```

19

```
# Windows (PowerShell，需以管理員身分執行)
```

19

```
mklink /D public\data ..\data
```

19

```
mklink /D public\images ..\images
```

19

### 3.4 啟動開發伺服器

19

#### Step 4 執行 Vite 開發伺服器

19

```
npm run dev
```

19

終端機會顯示類似訊息：Local: http://localhost:8080/，在瀏覽器中開啟即可看到 SciCover 前端。

19

### 3.5 安裝 Python 依賴

19

#### Step 5 安裝後端腳本所需套件

19

```
cd ../scripts
```

19

```
pip install -r requirements.txt
```

19

包含：requests、beautifulsoup4、lxml、openai、python-dateutil。

19

## 第四章：期刊選擇功能說明

20

### 4.1 前端期刊篩選

20

前端提供兩處期刊選擇介面：

20

### 4.2 後端 CLI 期刊選擇

20

後端爬蟲透過 `--journal` 參數指定要處理的期刊：

20

```
# 處理所有期刊（預設）
```

20

```
python -m scripts.main --journal all
```

20

```
# 只處理 Science
```

20

```
python -m scripts.main --journal Science
20
```

```
# 處理 Science 和 Nature
21
```

```
python -m scripts.main --journal Science --journal Nature
21
```

```
# 僅爬取、不執行 AI 摘要（測試用）
21
```

```
python -m scripts.main --journal Nature --dry-run
21
```

```
# 顯示詳細日誌
21
```

```
python -m scripts.main --journal Cell -v
21
```

## 第五章：GitHub 倉庫設定

22

### 5.1 建立 GitHub 倉庫

22

#### Step 1 在 GitHub 上建立新倉庫

22

### 5.2 取得 GitHub Models API 存取權

22

本專案使用 GitHub Models API 呼叫 DeepSeek V3 生成摘要。您需要建立一個具備 models:read 權限的 Personal Access Token (PAT)。

22

#### Step 3 建立 Personal Access Token

22

## 第六章：部署 GitHub Pages

24

## 6.1 啟用 GitHub Pages

24

### Step 1 設定 GitHub Pages 來源

24

## 6.2 驗證部署

24

### Step 2 觸發部署

24

推送任何變更到 main 分支即可觸發自動部署。等待幾分鐘後，您的網站會在以下位址上線：

24

<https://<your-username>.github.io/scicover/>

24

### Step 3 確認工作流狀態

24

前往倉庫頁面的 Actions 分頁，您應能看到「Deploy to GitHub Pages」工作流正在執行或已完成（綠色打勾）。

24

## 第七章：自動化爬取與 AI 摘要管線

25

### 7.1 工作流總覽

25

檔案 `.github/workflows/scrape-and-summarize.yml` 定義了每週自動執行的資料管線：

25

### 7.2 管線執行流程

25

每次執行時，管線會依序執行以下步驟：

25



## 7.3 手動觸發管線

25

除了每週自動執行外，您也可以隨時手動觸發：

25

## 第八章：前端功能操作指南

27

### 8.1 首頁

27

### 8.2 文章詳情頁

27

### 8.3 歸檔頁

27

### 8.4 深淺色模式

27

點擊右上角的太陽/月亮圖示即可切換。系統會自動偵測您的作業系統偏好，設定會儲存在 localStorage 中。

27

## 第九章：常見自訂操作

28

### 9.1 新增期刊

28

以新增 The Lancet 為例：

28

Step 1 前端：修改 constants.ts

28

```
// frontend/src/lib/constants.ts
```

28

```
// 1. 在 JournalName 型別中加入 'Lancet'
```

28

```
// (frontend/src/lib/types.ts)
28
export type JournalName = 'Science' | 'Nature' | 'Cell' | 'Lancet';
28

// 2. 在 JOURNALS 陣列中新增一筆
28
{ name: 'Lancet', color: 'var(--color-lancet)', label: 'The Lancet' },
28

// 3. 在 JOURNAL_RAW_COLORS 中新增
28
Lancet: '#004B87',
28

Step 2 前端：在 global.css 中新增顏色變數
28
/* frontend/src/styles/global.css */
28
--color-lancet: #004B87;
28

Step 3 後端：新增爬蟲
28
# scripts/scrapper/lancet_scraper.py
28
# 建立新的 LancetScraper 類別，繼承 BaseScraper
29
# 然後在 scripts/scrapper/__init__.py 中註冊
29

9.2 調整 AI 摘要風格
29

編輯 scripts/ai/prompts.py 中的 SYSTEM_PROMPT 即可調整生成風格：
29
```

### 9.3 修改更新頻率

29

編輯 .github/workflows/scrape-and-summarize.yml 中的 cron 運算式：

29

# 每週六 06:00 UTC (預設)

29

- cron: '0 6 \* \* 6'

29

# 每天 06:00 UTC

29

- cron: '0 6 \* \* \*'

29

# 每週三和週六 06:00 UTC

29

- cron: '0 6 \* \* 3,6'

29

## 第十章：資料結構說明

30

### 10.1 單篇文章 JSON

30

每篇文章儲存為獨立的 JSON 檔案，路徑格式：data/articles/{year}/{month}/{id}.json

30

{

30

"id": "science-2026-02-26",

30

"journal": "Science",

30

"volume": "391",

30

```
"issue": "6788",
30
"date": "2026-02-26",
30
"coverImage": {
30
"url": "images/science/science-2026-02-26-cover.jpg",
30
"credit": "NASA, CXC, IfA"
30
},
30
"coverStory": {
30
"title": {
30
"zh": "中文標題...",
30
"en": "English Title..."
30
},
30
"summary": {
30
"zh": "中文摘要內容...",
30
"en": "English summary content..."
30
},
30
"keyArticle": { "title": "...", "authors": [...], "doi": "..." },
31
```

```
"images": [{ "url": "...", "caption": { "zh": "...", "en": "..." } }],
31
"links": { "official": "...", "doi": "..." }
31
}
31
}
31
```

## 10.2 索引檔案

31

data/index.json 包含所有文章的摘要索引，前端用它來渲染列表：

31

```
{
31
  "lastUpdated": "2026-02-28T00:00:00Z",
31
  "articles": [
    { "id": "...", "journal": "...", "date": "...",
31
      "path": "articles/2026/02/....json",
31
      "title_zh": "...", "title_en": "..." }
    ]
31
  }
31
```

## 10.3 最新文章索引

31

data/latest.json 記錄每個期刊的最新文章 ID :	
31	
{ "Science": "science-2026-02-26",	
31	
"Nature": "nature-2026-02-26",	
31	
"Cell": "cell-2026-02-19" }	
31	
第十一章：常見問題排解	
32	
第十二章：成本與免費額度總覽	
33	
總運行成本：\$0/月，完全利用 GitHub 免費額度。	
33	
附錄：快速上手檢查清單	
34	
--- 教學文件結束 ---	
34	

# 第一章：專案總覽

SciCover 是一個開源的科研知識轉化平台，它自動追蹤頂級科學期刊（Science、Nature、Cell）的封面文章，透過 AI 進行深度解讀，生成符合中英雙語母語習慣的科普摘要。

## 1.1 架構說明

本專案採用「All-in-GitHub」架構，無需租用伺服器或資料庫：

層次	技術	說明
資料獲取層	Python + BeautifulSoup	定時爬取期刊官網與預印本平台
AI 處理層	GitHub Models API + DeepSeek V3	摘要、翻譯與圖文排版建議
持久化層	靜態 JSON	處理後的數據存儲於 GitHub 倉庫
展示層	React + Vite + GitHub Pages	前端讀取 JSON 並渲染

## 1.2 專案目錄結構

```
scicover/

├── .github/workflows/          # GitHub Actions 自動化工作流

|   ├── scrape-and-summarize.yml # 每週自動爬取 + AI 處理
|   └── deploy.yml              # 自動部署到 GitHub Pages

├── frontend/                  # React + Vite 前端應用

|   ├── src/components/        # UI 元件
|   ├── src/hooks/             # React Hooks
|   └── src/lib/                # 型別定義、常數
```

```
|   └─ src/styles/           # CSS 設計系統
|
|─ scripts/                 # Python 後端腳本
|
|   └─ scraper/             # 期刊爬蟲
|
|   └─ ai/                  # DeepSeek V3 摘要引擎
|
|   └─ pipeline/            # 資料管線編排器
|
|─ data/                    # 靜態 JSON 資料儲存
|
|─ images/                  # 封面圖片
```



## 第二章：環境準備

在開始之前，請確認您的電腦已安裝以下工具：

工具	最低版本	用途	安裝方式
Git	2.30+	版本控制	git-scm.com
Node.js	20+	前端開發	nodejs.org
Python	3.12+	後端腳本	python.org
npm	隨 Node.js	套件管理	隨 Node.js 安裝

### 2.1 驗證安裝

在終端機（Terminal）中執行以下指令，確認各工具已正確安裝：

```
git --version          # 應顯示 git version 2.x.x

node --version         # 應顯示 v20.x.x 或以上

python3 --version      # 應顯示 Python 3.12.x 或以上

npm --version          # 應顯示 10.x.x 或以上
```

### 2.2 註冊 GitHub 帳號

如果您尚未擁有 GitHub 帳號，請前往 [github.com](https://github.com) 註冊。後續所有操作（包括程式碼儲存、自動化流程、網站部署、AI API 調用）都依賴 GitHub。

## 第三章：本機開發設定

### 3.1 解壓專案

#### Step 1 解壓下載的 scicover.zip

# 將 scicover.zip 解壓到您想要的位置

```
unzip scicover.zip
```

```
cd scicover
```

#### 提示

如果您使用 Windows，可以直接右鍵點擊 zip 檔案選擇「全部解壓縮」，然後在 PowerShell 或 Git Bash 中 cd 進入專案目錄。

### 3.2 安裝前端依賴

#### Step 2 安裝 Node.js 套件

```
cd frontend
```

```
npm install
```

此指令會根據 `package.json` 安裝所有前端依賴，包括 React、React Router、TanStack Query、Framer Motion 等。

### 3.3 設定資料連結

#### Step 3 建立資料目錄的符號連結

前端開發伺服器需要能存取 `data/` 和 `images/` 目錄中的 JSON 檔案和圖片：

```
# macOS / Linux
```

```
ln -s ../data public/data
```

```
ln -s ../images public/images
```

# Windows (PowerShell, 需以管理員身分執行)

```
mklink /D public\data ..\data
```

```
mklink /D public\images ..\images
```

## 3.4 啟動開發伺服器

### Step 4 執行 Vite 開發伺服器

```
npm run dev
```

終端機會顯示類似訊息：**Local: <http://localhost:8080/>**，在瀏覽器中開啟即可看到 SciCover 前端。

#### 前端功能預覽

此時您可以看到 5 篇範例文章（Science、2 篇、Nature 2 篇、Cell 1 篇），並可操作期刊篩選、文章詳情、歸檔瀏覽、深淺色模式切換等功能。

## 3.5 安裝 Python 依賴

### Step 5 安裝後端腳本所需套件

```
cd ../scripts
```

```
pip install -r requirements.txt
```

包含：requests、beautifulsoup4、lxml、openai、python-dateutil。

## 第四章：期刊選擇功能說明

### 4.1 前端期刊篩選

前端提供兩處期刊選擇介面：

- **Header 導覽列**：頁面頂部固定顯示 All / Science / Nature / Cell 分頁按鈕，點擊即可切換顯示對應期刊的文章。
- **JournalTabs 元件**：首頁主內容區上方的篩選列，功能相同但更醒目。選中時會以各期刊的品牌色（紅 / 藍 / 綠）高亮顯示。

目前支援的期刊定義在 `frontend/src/lib/constants.ts`：

```
export const JOURNALS = [  
  { name: 'Science', color: 'var(--color-science)', label: 'Science' },  
  { name: 'Nature', color: 'var(--color-nature)', label: 'Nature' },  
  { name: 'Cell', color: 'var(--color-cell)', label: 'Cell' },  
];
```

#### 如何新增期刊

如需新增期刊（例如 The Lancet），只需在 `constants.ts` 的 `JOURNALS` 陣列中加入一筆，並在 `global.css` 中定義對應的顏色變數即可。前端會自動生成篩選按鈕。

### 4.2 後端 CLI 期刊選擇

後端爬蟲透過 `--journal` 參數指定要處理的期刊：

# 處理所有期刊（預設）

```
python -m scripts.main --journal all
```

# 只處理 Science

```
python -m scripts.main --journal Science
```

# 處理 Science 和 Nature

```
python -m scripts.main --journal Science --journal Nature
```

# 僅爬取、不執行 AI 摘要（測試用）

```
python -m scripts.main --journal Nature --dry-run
```

# 顯示詳細日誌

```
python -m scripts.main --journal Cell -v
```

參數	說明	範例
<code>--journal NAME</code>	指定期刊，可重複使用	Science, Nature, Cell, all
<code>--dry-run</code>	僅爬取，不呼叫 AI API	節省 API 額度
<code>-v / --verbose</code>	顯示 DEBUG 級別日誌	排查問題時使用

## 第五章：GitHub 倉庫設定

### 5.1 建立 GitHub 倉庫

#### Step 1 在 GitHub 上建立新倉庫

1. 登入 GitHub，點擊右上角「+」→「New repository」。
2. 輸入倉庫名稱（例如 `scicover`）。
3. 選擇 Public（公開，才能使用免費 GitHub Pages）。
4. 不要勾選任何初始化選項（不加 README、不加 `.gitignore`）。
5. 點擊 **Create repository**。

#### Step 2 推送本機程式碼到 GitHub

```
cd scicover # 確保您在專案根目錄

git init
git add .
git commit -m "feat: initial SciCover project"

# 將下方 <your-username> 替換為您的 GitHub 帳號名稱
git remote add origin https://github.com/<your-username>/scicover.git
git branch -M main
git push -u origin main
```

### 5.2 取得 GitHub Models API 存取權

本專案使用 GitHub Models API 呼叫 DeepSeek V3 生成摘要。您需要建立一個具備 `models:read` 權限的 Personal Access Token (PAT)。

#### Step 3 建立 Personal Access Token

1. 前往 GitHub → Settings → Developer settings → Personal access tokens → Tokens (classic)。
2. 點擊 **Generate new token (classic)**。
3. 設定名稱（例如 `scicover-models`）和過期時間。

4. 在 Scopes 中勾選 `models:read`。
5. 點擊 Generate token，複製生成的 Token（僅顯示一次）。

### 重要提醒

請妥善保管此 Token，切勿將其寫入程式碼中。它僅會透過 GitHub Secrets 安全儲存。

## Step 4 將 Token 加入 Repository Secrets

1. 在您的 scicover 倉庫頁面，點擊 **Settings** → **Secrets and variables** → **Actions**。
2. 點擊 **New repository secret**。
3. 名稱輸入：`MODELS_PAT`
4. 將剛才複製的 Token 貼入值欄位。
5. 點擊 **Add secret**。

## 第六章：部署 GitHub Pages

### 6.1 啟用 GitHub Pages

#### Step 1 設定 GitHub Pages 來源

1. 在倉庫頁面點擊 **Settings** → **Pages**。
2. 在 **Source** 下拉選單中，選擇 **GitHub Actions**（而非 Branch）。
3. 儲存即可。

完成後，每次推送到 main 分支時，`deploy.yml` 工作流會自動編譯前端並部署。

### 6.2 驗證部署

#### Step 2 觸發部署

推送任何變更到 main 分支即可觸發自動部署。等待幾分鐘後，您的網站會在以下位址上線：

`https://<your-username>.github.io/scicover/`

#### Step 3 確認工作流狀態

前往倉庫頁面的 **Actions** 分頁，您應能看到「Deploy to GitHub Pages」工作流正在執行或已完成（綠色打勾）。

#### 建議

如果部署失敗，點擊失敗的工作流可以查看詳細日誌。常見問題是 GitHub Pages 尚未啟用或 Source 設定不正確。



## 第七章：自動化爬取與 AI 摘要管線

### 7.1 workflow 總覽

檔案 `.github/workflows/scrape-and-summarize.yml` 定義了每週自動執行的資料管線：

設定項	值
觸發時機	每週六 06:00 UTC，或手動觸發
執行環境	ubuntu-latest
逾時時間	15 分鐘
所需 Secret	MODELS_PAT（已在第五章設定）

### 7.2 管線執行流程

每次執行時，管線會依序執行以下步驟：

1. **Checkout**：拉取最新的倉庫程式碼。
2. **安裝依賴**：設定 Python 3.12，pip install 安裝套件。
3. **執行爬蟲**：用 Python 爬取 Science、Nature、Cell 官網的最新一期封面文章。
4. **下載封面圖片**：將圖片儲存到 images/ 目錄。
5. **AI 摘要**：將爬取到的文章資訊傳送給 DeepSeek V3，生成中英雙語摘要。
6. **寫入 JSON**：將結果寫入 data/articles/ 目錄，更新 index.json 和 latest.json。
7. **提交與推送**：將新數據自動 commit 並 push，觸發 GitHub Pages 重新部署。

### 7.3 手動觸發管線

除了每週自動執行外，您也可以隨時手動觸發：

4. 前往倉庫頁面 → **Actions** 分頁。
5. 在左側選擇 **Scrape & Summarise Cover Stories**。
6. 點擊 **Run workflow** → **Run workflow**。

#### 免費額度說明

GitHub Models API 免費層每天允許 50 次請求，本專案每週僅使用 3 次（每個期刊 1 次），絕對不會超過限制。GitHub Actions 免費層提供每月 2,000 分鐘，本專案每週僅使用約 2 分鐘。

## 第八章：前端功能操作指南

### 8.1 首頁

- **Hero 封面區塊**：顯示最新一篇封面文章，包含大圖、中英雙語標題、「Read More」連結。
- **期刊篩選列**：點擊 All / Science / Nature / Cell 即可篩選。
- **文章卡片網格**：回應式設計，桌面顯示 2-3 欄，手機顯示單欄。每張卡片包含封面圖、期刊標籤、日期、中英雙語標題。

### 8.2 文章詳情頁

- **期刊資訊**：顯示期刊名稱、卷期號、出版日期。
- **封面圖片**：大圖顯示，附有中英雙語圖說和圖片來源資訊。
- **雙語摘要**：中文在上、英文在下，各自有獨立標題。左側帶有期刊品牌色的裝飾邊框。
- **原文連結**：頁尾提供 Official Article、DOI Link、Open Access Preprint 三個跳轉按鈕。

### 8.3 歸檔頁

- **年份選擇器**：橫向排列的年份按鈕。
- **月份選擇器**：12 個月份按鈕網格，無資料的月份會自動變暗且不可點擊。
- **結果顯示**：選擇年月後自動顯示對應文章卡片。

### 8.4 深淺色模式

點擊右上角的太陽/月亮圖示即可切換。系統會自動偵測您的作業系統偏好，設定會儲存在 localStorage 中。

## 第九章：常見自訂操作

### 9.1 新增期刊

以新增 The Lancet 為例：

#### Step 1 前端：修改 constants.ts

```
// frontend/src/lib/constants.ts

// 1. 在 JournalName 型別中加入 'Lancet'

// (frontend/src/lib/types.ts)

export type JournalName = 'Science' | 'Nature' | 'Cell' | 'Lancet';

// 2. 在 JOURNALS 陣列中新增一筆

{ name: 'Lancet', color: 'var(--color-lancet)', label: 'The Lancet' },

// 3. 在 JOURNAL_RAW_COLORS 中新增

Lancet: '#004B87',
```

#### Step 2 前端：在 global.css 中新增顏色變數

```
/* frontend/src/styles/global.css */

--color-lancet: #004B87;
```

#### Step 3 後端：新增爬蟲

```
# scripts/scrapper/lancet_scraper.py
```

```
# 建立新的 LancetScraper 類別，繼承 BaseScraper
```

```
# 然後在 scripts/scraper/__init__.py 中註冊
```

## 9.2 調整 AI 摘要風格

編輯 `scripts/ai/prompts.py` 中的 `SYSTEM_PROMPT` 即可調整生成風格：

- 中文風格調整：目前設定為「知乎/果壳」風格，可改為學術風格或報紙風格。
- 英文風格調整：目前設定為 Quanta Magazine 風格，可改為 New Scientist 或學術摘要風格。
- 摘要長度：在 Prompt 中調整「150-250」字數範圍即可。

## 9.3 修改更新頻率

編輯 `.github/workflows/scrape-and-summarize.yml` 中的 cron 運算式：

```
# 每週六 06:00 UTC (預設)
```

```
- cron: '0 6 * * 6'
```

```
# 每天 06:00 UTC
```

```
- cron: '0 6 * * *'
```

```
# 每週三和週六 06:00 UTC
```

```
- cron: '0 6 * * 3,6'
```

## 第十章：資料結構說明

### 10.1 單篇文章 JSON

每篇文章儲存為獨立的 JSON 檔案，路徑格式：`data/articles/{year}/{month}/{id}.json`

```
{  
  
  "id": "science-2026-02-26",  
  
  "journal": "Science",  
  
  "volume": "391",  
  
  "issue": "6788",  
  
  "date": "2026-02-26",  
  
  "coverImage": {  
  
    "url": "images/science/science-2026-02-26-cover.jpg",  
  
    "credit": "NASA, CXC, IfA"  
  },  
  
  "coverStory": {  
  
    "title": {  
  
      "zh": "中文標題...",  
  
      "en": "English Title..."  
    },  
  
    "summary": {  
  
      "zh": "中文摘要內容...",  
  
      "en": "English summary content..."  
    },  
  },  
}
```

```
    "keyArticle": { "title": "...", "authors": [...], "doi": "..." },

    "images": [{ "url": "...", "caption": { "zh": "...", "en": "..." } }],

    "links": { "official": "...", "doi": "..." }

  }

}
```

## 10.2 索引檔案

`data/index.json` 包含所有文章的摘要索引，前端用它來渲染列表：

```
{

  "lastUpdated": "2026-02-28T00:00:00Z",

  "articles": [

    { "id": "...", "journal": "...", "date": "...",

      "path": "articles/2026/02/....json",

      "title_zh": "...", "title_en": "..." }

  ]

}
```

## 10.3 最新文章索引

`data/latest.json` 記錄每個期刊的最新文章 ID：

```
{ "Science": "science-2026-02-26",

  "Nature":  "nature-2026-02-26",

  "Cell":    "cell-2026-02-19" }
```

## 第十一章：常見問題排解

問題	可能原因	解決方法
前端看不到文章	data/ 路徑未正確連結	確認 public/data 和 public/images 符號連結存在
Actions 失敗： MODELS_PAT	Secret 未設定或 Token 過期	重新建立 PAT 並更新 Repository Secret
爬蟲失敗	期刊網站結構變更	檢查 scraper 中的 CSS 選擇器是否過時
AI 輸出格式錯誤	DeepSeek V3 回傳非 JSON	管線已內建重試邏輯，檢查日誌確認
GitHub Pages 404	SPA 路由問題	確認 deploy.yml 已正確設定
npm install 失敗	Node.js 版本過低	升級到 Node.js 20+



## 第十二章：成本與免費額度總覽

服務	免費額度	本專案用量	是否足夠
GitHub Actions	2,000 分鐘/月	約 8 分鐘/月	綁綁有餘
GitHub Models API	50 次/天	3 次/週	綁綁有餘
GitHub Pages	公開倉庫免費	靈活使用	完全免費
GitHub 儲存空間	1 GB	預估 < 50 MB	綁綁有餘

總運行成本：\$0/月，完全利用 GitHub 免費額度。

## 附錄：快速上手檢查清單

步驟	動作	狀態
1	解壓 scicover.zip	
2	cd frontend && npm install	
3	建立 data/images 符號連結	
4	npm run dev 確認前端正常	
5	pip install -r scripts/requirements.txt	
6	在 GitHub 建立新倉庫	
7	git init && git push	
8	建立 PAT 並加入 Secrets	
9	啟用 GitHub Pages (Source: GitHub Actions)	
10	手動觸發 Actions 驗證	
11	確認網站上線	

--- 教學文件結束 ---