Green Software Foundation

碳意識與綠色軟體導讀

Morris – 2026

Green Software Foundation

看不見的軟體，正在消耗巨大的能源

每一次 AI Prompt、API Call、資料傳輸，
背後都是：

GPU 運算 | 電力消耗 | 散熱冷卻 | 資料中心

AI 時代的效率問題，
正在變成能源問題。

資料中心耗電量佔全球用電
2-4%
相當於整個航空業規模

數位服務的碳排持續快速成長
+30%
每年成長率 (預估)

大量閒置資源仍在 24/7 運轉
20-30%
企業雲端資源浪費比例

能源成本持續攀升
Up 20%
近兩年資料中心電力成本

資料來源：Gartner 2025、IEA 2024、IDC 2024

Green Software Foundation

為什麼我們要談這個？

"軟體本身沒有形體，
但它運行的代價卻很沉重。"

我們看不見數位世界的污染，但資料中心 (Data Centers) 消耗的電力已佔全球的 2-4%，相當於整個航空業。

Gartner 2025 預測：

永續性 (Sustainability) 將成為科技戰略的前三大重點。

節費

能源成本上升

法規

歐盟 CSRD 要求

效能

高能效 = 高速度

競爭力

ESG 採購規範

什麼是綠色軟體？

這不是要做慈善，而是用 **數據驅動** 的方式，
打造更 **高效、省電、長壽** 的軟體系統。

可量化

無法量化就無法改善。我們需要知道每一行程式碼產生了多少碳

可歸因

是誰造成的？是 AI 訓練？還是無效的資料庫查詢？邊界要定義清楚

可治理

將碳排指標納入 KPI 與日常開發流程 (CI/CD) 中

Green Software Foundation

國際推動者：綠色軟體基金會 (GSF)

"致力於建構一個值得信賴的生態系，
讓綠色軟體成為業界的**預設選項 (Default Choice)**。"

🏛 組織背景

成立於 2021 年，隸屬於 **Linux Foundation** 旗下非營利組織。
匯聚微軟、GitHub、埃森哲等全球領導企業。

68+
成員組織

1100+
個人會員

1. 制定標準 (Standards)

建立產業通用的衡量規範。

SCI 規範 | ISO 21031

2. 開發工具 (Tooling)

加速開發者落實的開源工具。

Carbon Aware SDK | Impact Framework

3. 最佳實踐 (Best Practices)

培育人才與建立模式。

綠色軟體模式 | 培訓認證

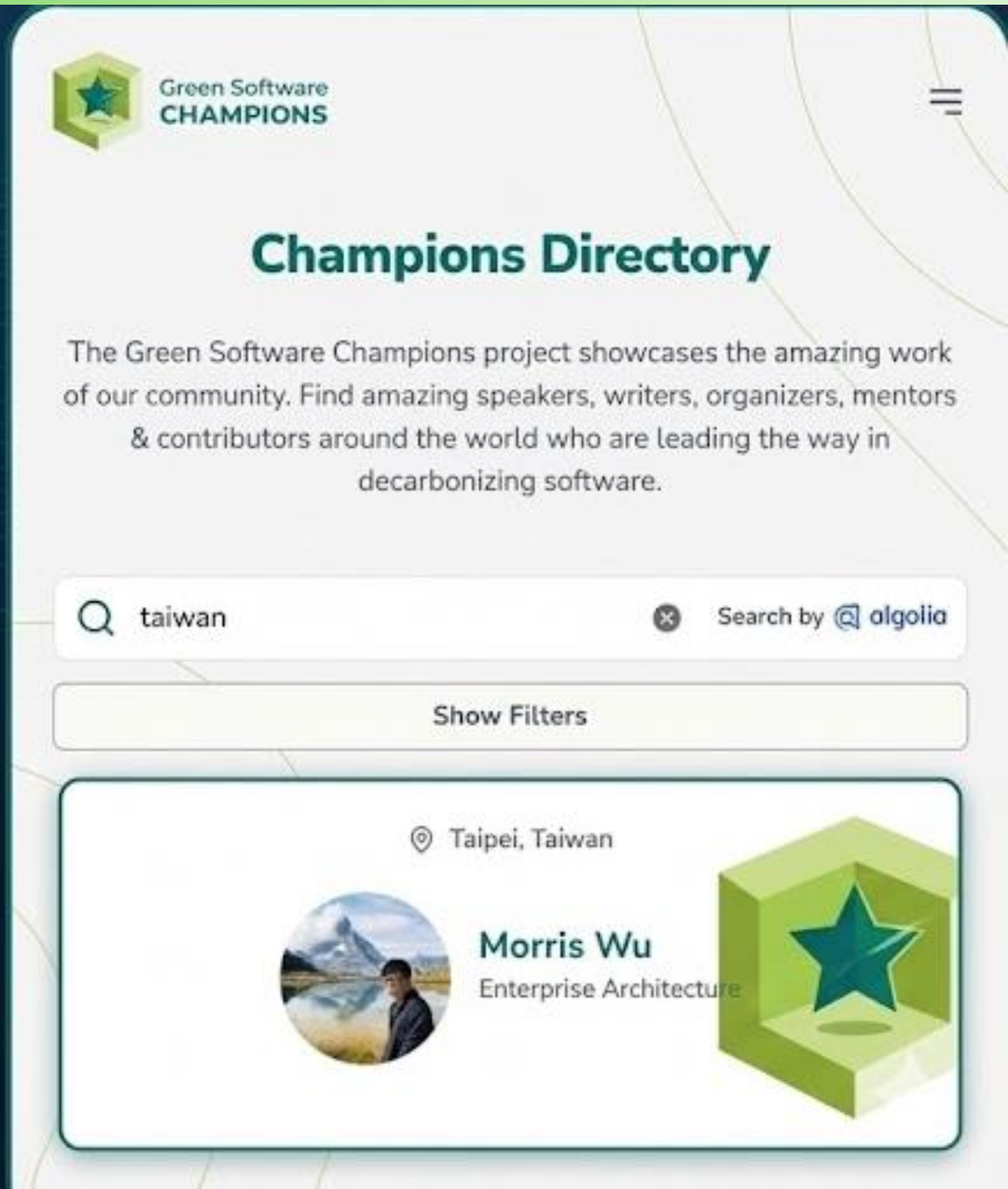
Green
Software
Foundation

Morris

CNCF TAG Environmental Sustainability

Green Software Foundation

- 職涯主要在金融產業
  - 資訊企業架構師：策略規劃、推動
  - DevOps Team Leader
  - Infer & SRE Team Leader



Green Software Foundation

綠色軟體的主流化趨勢

Gartner 戰略預測

"到 2027 年，**25% 的 CIO** 將會有與『永續技術 (Sustainable Technology)』影響力直接掛鉤的薪酬績效指標。"

永續技術已被列為 2024/2025 頂尖戰略技術趨勢的基礎框架之一，不再是選配，而是標配。

IDC 採購趨勢

全球 **85%** 的組織已將永續性指標納入 IT 採購與供應商選擇的關鍵標準。

法規強制力

歐盟 **CSRD** (企業永續發展報告指令) 於 2024 生效，強制揭露範疇三 (Scope 3) 排放，包含軟體供應鏈。

供應鏈壓力

Apple、Microsoft 要求供應鏈於 **2030 年達成淨零**。身為供應鏈一環的台灣軟硬體產業首當其衝

實踐的三大支柱

1. 能源效率

Energy Efficiency

核心：用最少的電，做最多的事。

心法：最環保的能源，是你「沒使用」的能源。

2. 硬體效率

Hardware Efficiency

核心：硬體製造過程有大量碳排 (隱含碳)。

心法：盡量延長設備壽命，榨乾每一分算力。

3. 碳意識

Carbon Awareness

核心：電網並非隨時都乾淨。

心法：在陽光普照、風力強勁時執行繁重工作。

Green Software Foundation

1. 能源效率：程式碼決定耗電

程式碼寫得越沒效率，CPU 就要跑越久，消耗的電就越多。

- ✓ **語言選擇：** 編譯式語言 (如 C, Rust) 通常比直譯式語言 (如 Python) 節能數十倍。
- ✓ **演算法最佳化：** 減少不必要的迴圈與計算。
- ✓ **無伺服器架構 (Serverless)：** 沒有請求時就不運行，徹底杜絕「閒置空轉」。

殭屍伺服器 (Zombie Servers)

那些開著卻沒人在用的機器。

關掉它們 = 瞬間減碳 100%

很多企業有 20~30% VM 根本沒人在用，但每個月仍持續耗電、付雲端費用。

2. 硬體效率：看不見的隱含碳

手機、伺服器在「開機前」，製造過程就已經產生了巨大的碳排放。

隱含碳 (Embodied Carbon)

就像你買了一個昂貴的名牌包，你用越久，每天的使用成本就越低。

策略：

- 不要為了小升級就換硬體。
- 支援舊版手機運行 (減少使用者的換機壓力)。

提升利用率

一台伺服器只跑 10% 的效能，卻消耗 50% 的電力。

策略：

- 盡量塞滿伺服器 (虛擬化、容器化)。
- 公有雲通常比自建機房利用率更高。

Green Software Foundation

3. 碳意識：像向日葵一樣運算

電網的電力來源就像一杯綜合果汁，有時候乾淨(綠電)多，有時候髒(燃煤)多。

時間轉移

"晚點再做"

AI 訓練、報表生成、備份...這些不用馬上做的事，等到**半夜風大**或**中午太陽大**的時候再跑。

空間轉移

"去別的地方做"

現在台灣是晚上(沒太陽)，但歐洲是白天。把工作丟到那邊的雲端機房去跑。

案例：Xbox 預設在夜間綠電時段下載遊戲更新

Green Software Foundation

GreenOps：新時代的維運文化

FinOps (財務) + DevOps (開發維運) + Sustainability (永續)

看見

透過儀表板，讓開發者看見每次程式碼更新造成的碳排變化。

最佳化

自動化調整資源，沒人用的測試環境，下班自動關機。

文化

讓「減碳」成為跟「修 Bug」一樣自然的日常。

怎麼打分數？ SCI 指標

ISO 21031 標準：軟體碳強度 (Software Carbon Intensity)

SCI = (⚡ 能源 × ☁️ 碳強度 + 📦 硬體) ÷ 👤 服務單位

這個公式告訴我們，要降低分數(越低越好)，你可以：

1. **少用電** (寫好程式碼)
2. **選綠電** (碳意識)
3. **少買硬體** (硬體效率)
4. **服務更多人** (提升單位效益)

🔍 為什麼這很重要？

因為它排除了「碳權抵換」(買贖罪券)。SCI 逼著你去真正減少軟體本身的排放。

Green Software Foundation

我們在哪裡？(成熟度矩陣)

不需要一步登天，這是個循序漸進的過程。

1. 萌芽期

個人有興趣
無組織策略

2. 覺醒期

開始測量
人工優化

3. 行動期

清理殭屍資源
基礎自動化

4. 卓越期

即時碳監控
淨零排放

5. 領航期

24/7 零碳電力
業界標竿

↑ 大多數企業目前正從第 1 階段邁向第 2 階段

Green Software Foundation

綠色軟體是大家的議題

Green Software Foundation

實務案例：他們怎麼做？

Microsoft Teams

疫情期間流量暴增，為了撐住系統，他們關閉了非必要的「已讀回條」功能。

優雅降級

Xbox

成為第一台「碳意識」遊戲機。自動偵測當地電網數據，只在綠電充沛時下載大更新。

碳意識

AWS Serverless

研究顯示，無伺服器架構因為消除了閒置資源，比傳統虛擬機更節能。

能源效率

F5

透過彈性的硬體資源調度，大幅提升伺服器利用率，減少硬體採購。

硬體效率

Green Software Foundation

綠色軟體評分卡 (Scorecard)

六大關鍵指標，評估軟體「綠」的程度與商業影響。

1. 綠色影響 (Green Impact)

是否避免了「綠色缺陷」的程式碼模式？

影響：ESG 目標、營運成本

2. 軟體韌性 (Resiliency)

面對故障，系統能否自我修復？

影響：客戶滿意度、營收

3. 軟體敏捷性 (Agility)

程式碼是否易讀、易維護？

影響：維護成本、人力

4. 軟體優雅度 (Elegance)

能否以最低複雜度交付價值？

影響：創新速度

5. 雲端成熟度 (Cloud Maturity)

是否準備好遷移至 PaaS？

影響：彈性、成本最佳化

6. 開源安全性 (OSS Safety)

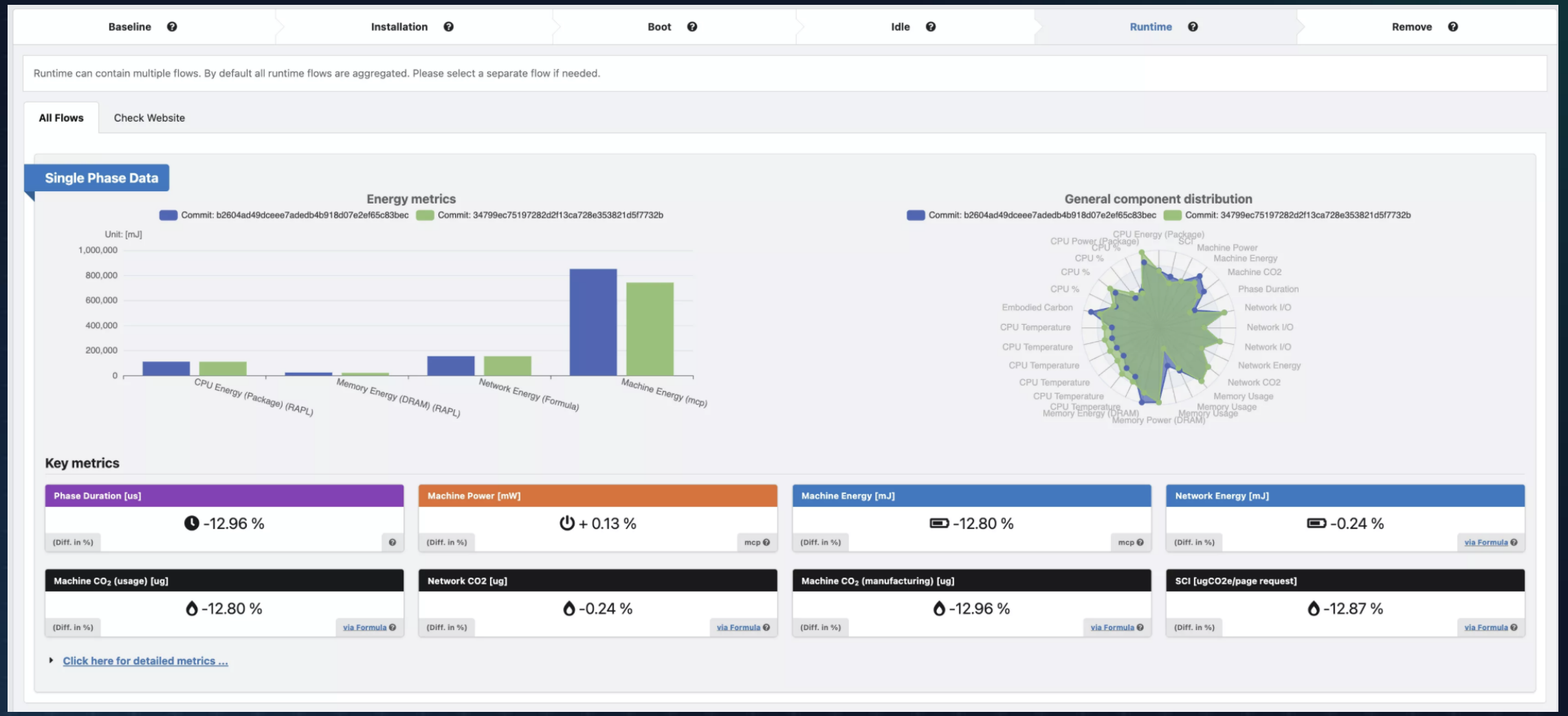
第三方元件是否安全、合規？

影響：資安風險、法律風險

Green Software Foundation

結合硬體記錄的能源應用分析

GitHub 連結

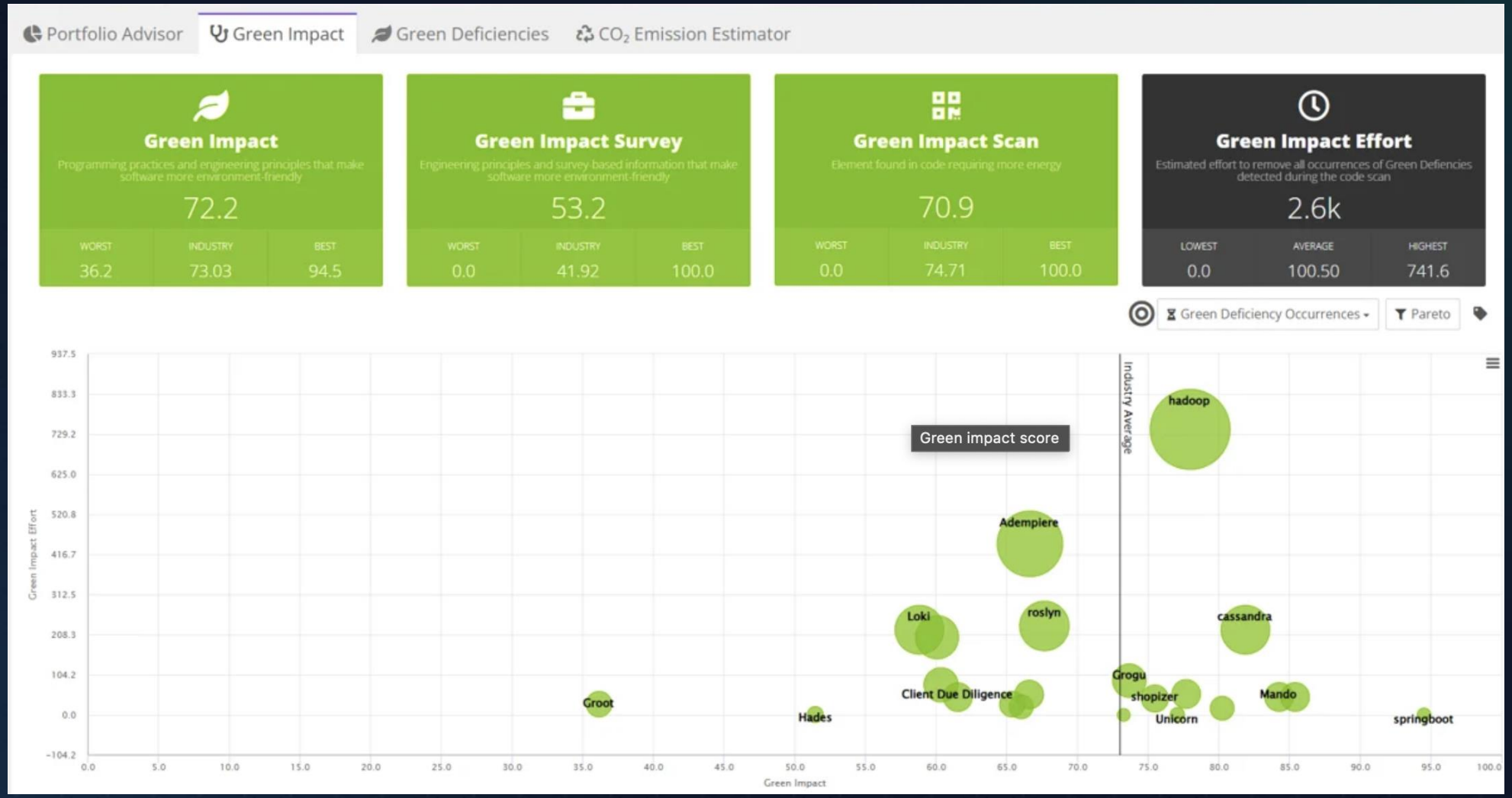


備註：圖為Lab環境產製，非正式環境

Green Software Foundation

程式碼綠色指標衝擊分析

GitHub 連結



備註：圖為Lab環境產製，非正式環境

實務應用：軟體開發生命週期 (SDLC)

需求/設計

- 碳影響評估
- 選擇高效架構
- 刪減冗餘功能

開發

- 綠色程式碼規範
- 演算法最佳化
- **AI 輔助節能**

測試

- 能耗基準測試
- 碳排放模擬
- CI 碳差異報告

維護/維運

- 即時碳監控
- 碳感知排程
- 定期綠色審核

AI Coding 時代的永續心態

Prompt 工程： 在提示詞中加入「請以最低能耗、最少記憶體佔用為原則」的約束。

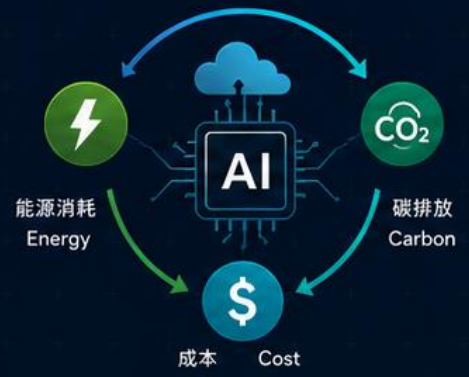
CI 守門員： 自動偵測並拒絕高耗能的程式碼變更 (Carbon Diff)。

Green Software Foundation

AI Token Economics

能源與碳成本，將重新定義 AI 設計

Token = AI 世界的成本貨幣



NEW HTML vs Markdown：AI 讀取更容易，但更耗 Token

部分網路訊息指出：

- HTML 結構化，AI 解析更精準
- 適合複雜文件、長篇內容
- 利於資訊擷取與理解

但 Token 使用量較高：

Markdown → HTML

+1.87 倍

Token 使用量增加

* 根據網路實測與社群分享平均值，實際數值依內容而異

AI 成本正在快速轉變

過去 AI 時代	新 AI 時代
比模型大小	比 Token 效率
重視參數量	重視能源成本
一次性訓練成本	長期推論成本
單純追求準確率	平衡成本 × ×碳排 × 效能

為何 Token 很重要？

每一次 AI 回應都會產生

GPU 運算 · 電力消耗 · 雲端成本 · CO_2 碳排放

當 AI Agent 大量運行後

推論成本將超越訓練成本

尤其在：AI Copilot、AI Agent、多代理系統、長上下文模型情境下

Lean AI：用更少 Token，創造更高價值

不是最強的 AI，而是最有效率的 AI

- 使用適合大小的模型
- 減少不必要推理
- 控制 Token 長度
- 降低 Agent 呼叫次數
- 優化記憶與上下文

精實設計
節能減碳
創造更大價值

未來企業的 AI KPI

未來企業不只看準確率與功能數量，
而是衡量新一代 AI 指標

- 每次任務 Token 成本
- 每次推論碳排
- 每美元創造價值
- 每瓦特產生效能

一句話總結

未來 AI 的競爭力，不只是「更聰明」，而是：

更節能、更低碳、更高 Token 效率！

關鍵字

Token Flow · AI Energy Loop · Carbon-aware AI · Lean Agentic AI · Sustainable AI · Cost vs Intelligence

Green Software Foundation

你可以立即採取的行動

ACTIONS YOU CAN TAKE TODAY

從小改變開始，每一步都能帶來巨大的影響。

01 關掉 不在用的資源

- 關閉閒置的 VM / 容器
- 刪除未使用的儲存與備份
- 停用測試環境（下班關機、設定排程）

難度：低　影響：高

5 分鐘就能開始！立刻省電資錢

02 減少 不必要的傳輸

- 最佳化圖片 / 影片大小
- 減少不必要的 API 呼叫
- 善用快取與 CDN

難度：低　影響：高

小優化，大效益！立刻降低浪費

03 量測 與建立可視化

- 導入 Carbon Aware 工具
- 建立基礎的能耗指標
- 設定 GreenOps 儀表板

難度：低　影響：高

看見數據，才能持續改善

04 設定目標 持續改善

- 設定減碳或能耗目標
- 追蹤改善進度
- 定期回顧與優化

難度：低　影響：高

每一次改善，累積大改變

05 影響團隊 一起行動

- 分享最佳實踐
- 建立節能文化
- 讓永續成為團隊習慣

難度：低　影響：高

一個人走得快，一群人走得遠

改變，不需要很難。

從今天的一個小行動，讓我們一起打造**更高效、更永續**的未來！

從今天開始，選擇一件你最容易做到的事，**立即行動！**

選一件小事 → 現在就去做 → 感受改變

Green Software Foundation

3 大高效行動 × 4 步執行框架

用更少能源，完成同樣甚至更多的數位價值

01 追蹤與監控 能源使用及成本

- 關注 PUE ≤ 1.3
- 運用工具掌握碳排與耗能來源
- 導入 GreenOps 即時監控與警報

減少 **20~40%** 運算浪費
降低 **6~12%** 雲端成本

02 設計更精簡的軟體

- 閒置時間自動縮減或關閉 (Dev/Test 環境)
- 簡化 CI/CD 流程
- 善用 AI 輔助產出綠色程式碼

縮短 **20~50%** 建置時間
同步降低能源消耗

03 選擇合適大小的 AI 模型

- 採用適用的模型 (SLM)
- 聚焦推論 (Inference) 效率
- 優化資料與提示詞

每次推論能耗
降低 **30~70%**

1 評估 Assess

建立能源、成本與使用率基準，找出高耗能工作負載與無效率之處

2 最佳化 Optimise

執行三大要事：
低 PUE 架構、精簡軟體、AI 模型與提示詞最佳化

3 驗證 Validate

重新衡量能源使用、碳排及成本的改變，確認改善措施確實發揮成效

4 擴展 Scale

將效率檢查機制嵌入日常 DevOps/GreenOps 流程，並推廣至全公司團隊

政策與資源支持 (IMDA)

- 全球第一個加入綠色軟體基金會 (GSF) 的政府組織
- 推出「雲端碳計算器」與「綠色軟體指南」，提供實用工具與指引
- 實證可降低 ≥20% 能源與成本 (經實際案例驗證)
- 制定標準，打造高效能數位基礎設施 SS697:2023 · SS715:2025
- 提供補助與資源支持協助企業對齊標準，加速數位永續轉型

Green Software Foundation

今天帶走的三件事

THREE KEY TAKEAWAYS

01 Green is **Performance**

02 Start with **OFF**

03 Measure to **Manage**

我們正在進入
AI 大規模生成的時代。

下一代優秀的工程師，
不只是寫得快，
而是能用**更少資源**，
創造**更大價值**的人。

Green Software Foundation

綠色軟體 - 七大實踐框架

參考資料：Accenture: Uniting technology and sustainability

Green Software Foundation

- 設計高效且使用體驗良好的介面。
- 簡化操作以減少螢幕使用時間，降低碳排放。
- 優化效能的方法：
  - 選擇節能的螢幕色彩設計。
  - 評估與提升系統處理能力。
  - 壓縮內容與圖像以降低資源耗用。

- **平台與語言選擇**：優先選擇具備高能源效率的方案。
- **軟體架構設計**：注重永續性，減少資源消耗。
- **開發與維運**：在程式開發、測試、部署及運維中推行環保方法。

通過夜間模式設定，可將使用者介面的碳排放減少60%

解釋型語言的能耗比半編譯型語言高出10倍，且比編譯型語言高出48倍

Green Software Foundation

如何在開發週期中增加永續性呢?

應用程式運作流程優化

- 流程應盡可能自動化，即使需要人工介入，也應以電子化方式進行。
- 投資於 DevOps，以提升流程和運作的彈性與效率。
- 最大限度減少開發過程中不必要的流程，包括使用者體驗相關的流程。
- 優先選擇具有綠色標準的產品供應鏈。
- 善用 APM 工具，精確掌握應用程式的資源使用狀態。

數據管理選擇

•實現精簡高效的資料收集與儲存，避免冗餘。
•根據資料使用頻率區分為熱、溫、冷凍級別，減少存取資源的消耗。

基礎架構調整

•善用端點技術（如 CDN）以減少長距離傳輸耗能。
•網路架構設計應盡量避免跨區域（如跨雲環境）及不必要的往返傳輸。
•資料儲存應採用重複數據刪除技術，以降低資源消耗並提升效能。

- 推動資料中心向雲端遷移，採用適當的託管服務及綠色雲端軟體開發，減少硬體耗損並提升運行效能。

  案例：亞洲東>歐洲北，降低66%碳排

- 評估邊緣計算的部署方案，將數據儲存於接近使用者端的位置，降低傳輸能耗。

- 選用具有綠能標章的產品，減輕基礎設施（包括終端設備、網路、伺服器等）的環境負擔。

（金融業多數已制定相關採購規範。）

案例：全球約17%的電子廢棄物妥善回收，並多數企業的硬體回收率不到10%