HWDC 2026 • 603+604 會議室

打造跨模型 AI Agent 的 CI/CD 管線

從本地開發到雲端部署｜AI Gateway • 成本可觀測 • 雲端部署

Jimmy Liao 廖聖傑 • Google Cloud AI GDE • AI MVP

教材與手冊：`jimmyliao.pse.is/hwdc2026`

jimmyliao.pse.is/hwdc2026

Intro

Jimmy Liao

- Google Cloud AI GDE ・ Microsoft AI MVP
- Leap Core Co-founder / CTO
- Smart Manufacturing ・ Finance ・ AgentOps
- 聯絡: jimmyliao@jimmyliao.net 電子報 https://memo.jimmyliao.net/



Jimmy Liao

Intro

Step 0：取得課堂 Starter

手冊：`jimmyliao.pse.is/hwdc2026`

專案：`github.com/jimmyliao/hwdc2026-starter`

下載：`git clone` or zip (請參考手冊)

安裝：`make setup` or `uv sync`

Intro

Step 0：確認起點 0/6

- 複製設定：`cp .env.example .env`（預設 `PROVIDER=mock`）
- 環境健檢：`uv run hw doctor` 全 PASS
- 起點檢查：`make check L=L1` → 看到 `0/6` 代表環境正常
- 請參考學員手冊 📖

Intro

跨平台指令：make or uv run hw

- `make check L=L1` = `uv run hw check L1`
- `make ask Q="你好"` = `uv run hw ask "你好"`
- `claim` ／ `run` ／ `report` ／ `submit` 全部同名
- Windows 不需裝 `make`，直接用 `uv run hw`

Intro

切換 Provider 與終端機小撇步

- macOS／Linux：`PROVIDER=class make ask Q="你好"`
- PowerShell：`$env:PROVIDER="class"; uv run hw ask "你好"`
- 也可以直接改 `.env` 裡的 `PROVIDER=`
- (Windows Terminal encoding) -> `chcp 65001`

Part 1 Pain Point

開發 Agent -> 共用 -> Cloud

- Agent 在筆電上跑得很好？
- 上雲後掛過？
- 「沒報錯、結果卻不對」？

Part 1 Pain Point

When you’ d done good work (locally) before weekend

週五：Agent 在我的筆電上回答得很漂亮

```
$ PROVIDER=class make ask Q="用一句話介紹台北"
uv run hw ask "用一句話介紹台北"
台北是一座融合了現代摩天大樓、古老寺廟與多元美食文化的國際都會。
provider: remote | model: gemini-3.1-flash-lite | tokens: 16 in / 22 out | cost: $0.000000 USD
```

週五：Agent 回答漂亮的終端機畫面

一切完美

Part 1 Pain Point

Deploy to production on Friday?

No Error and log

But User sees -> “Not response” / “No answer”

同一個 Agent 上雲後：沒有答案、沒有錯誤

```
$ make models
uv run hw models
fast: gemini-3.1-flash-lite
mock: mock
class: fast
$ make ask Q="你好"
uv run hw ask "你好"

$ echo "結束碼：$?"
結束碼：0
```

Agent 上雲後回應空白

Part 1 Pain Point

Lesson Learn 1：Configuration

`OPENAI_API_KEY`

`OPENAI_KEY`

Part 1 Pain Point

Lesson Learn 2：key leaks

commit 紀錄裡出現 key（示意：假 key，已遮罩）

```
$ git log --oneline
bf7edb6 刪掉 .env（以為安全了）
8f70c55 加上設定檔
$ git log -p -S AIzaSyFAKE --format="commit %h  %s" | sed -E "s/(AIzaSyFAKE)[A-Za-z_]*/\1[redacted]/"
commit bf7edb6  刪掉 .env（以為安全了）

diff --git a/.env b/.env
deleted file mode 100644
index 93ca897..0000000
--- a/.env
+++ /dev/null
@@ -1 +0,0 @@
-GEMINI_API_KEY=AIzaSyFAKE[redacted]
commit 8f70c55  加上設定檔

diff --git a/.env b/.env
new file mode 100644
index 0000000..93ca897
--- /dev/null
+++ b/.env
@@ -0,0 +1 @@
+GEMINI_API_KEY=AIzaSyFAKE[redacted]
```

commit 紀錄中出現 key（已遮罩）

刪掉檔案，歷史還在

Part 1 Pain Point

Lesson Learn 3：handle Error/Exception

```
try:
    return call_model(prompt)
except Exception:
    return ""
```

最危險的錯誤，是沒有錯誤

Part 1 Pain Point

Operator

你 → 一個號碼 → 總機 → Route A／B／C

總機：轉接、記錄、掛斷

=> AI Gateway

Component? Service? => It’s just a conceptual layer



Operator

Part 1 Pain Point

Demo (d-live)

Example => the AI gateway behind this workshop

```
PROVIDER=class make ask Q="用一句話介紹台北"
```

Part 1 Pain Point

Why mock (first)

- 零風險練習：L1／L2 全程離線
- 故障注入：`MOCK_FAIL=429`（模擬「太頻繁」）演練事故與 fallback（備援）

Part 1 Pain Point

Hands-on (by you) : preflight / env check up

Prerequisite: `uv run hw doctor` should all PASS

`make setup` 、 `make check L=L1`

L1　①〜⑥ 修完（6/6，約 15 分鐘）

→ `make ask` 回「Mock 回答」

Part 1 Pain Point

Hands-on (by you) : ready to submit / online

- L1 完成後 `make claim` → `PROVIDER=class make ask`
- `mock` ↔ `class` 互換，程式不改
- L2 `make report`、fallback（約 15 分鐘）
- L3 `make run`、`make submit`

Part 1 Pain Point

What we learn on Part 1

1. 換模型 -> 不改程式
2. 成本 -> 視覺化
3. 失敗 -> 可以觀測與自定義

Part 2 L1 簡單改

你拿到的 Agent：故意壞掉

Agent → `gateway` → mock／class

gateway：節流（限頻）、退避（等一下再試）、快取、fallback（備援）

6 個坑：基本 3 題（約 15 分鐘）＋加強 3 題（L3 送出前須完成）

先用 mock，修好再碰真實模型

Part 2 L1 簡單改

怎麼讀 `make check`

```
make check L=L1   (Windows：uv run hw check L1)
✗ 模型別名可辨識：檢查預設別名…
✓ 範例環境設定可用
L1：1/6
```

先讀 ✗ 後面的提示 → 看 log `usage.jsonl` → 才改

Windows 不支援 UTF-8 時顯示 [OK]／[X]

Part 2 L1 簡單改

基本 3 題：先看 `make models`

- 先跑 `make models` 看別名清單（`fast: gemini-3.1-flash-lite`）
- **冒號左邊** `fast` ／ `mock` ／ `class` 才是別名，勿填右邊模型全名！
- 📖 手冊 `lab-guide.html` §3 L1「基本」
- 每題都有 `▼ 卡住了才打開`

Part 2 L1 簡單改

基本 3 題

1　預設別名拼錯（`agent/config.py` 填 `"fast"`，勿填模型全名）

2　環境變數名拼錯（`.env.example` 刪掉 `PROVDER=mock`）

3　`.env` 沒從專案根目錄載入（`Path(__file__).resolve().parents[1] / ".env"`）

目標：`make check L=L1` 看到 `3/6`（📖 手冊 §3 L1 `▼ 卡住了才打開`）

Part 2 L1 簡單改

加強 3 題：為什麼要先修 ④⑤？

未修 ④ 前：一般問答會無視 `PROVIDER=mock` 硬跑 `fast`

未修 ⑤ 前：呼叫失敗會被 `except` 吞掉、只印出空行

修完 ④⑤ 後，`make ask` 才會真正讀取 `PROVIDER`！

📖 手冊 `lab-guide.html` §3 L1「加強」（含 `▼ 卡住了才打開`）

Part 2 L1 簡單改

加強 3 題（送出前必做）

4　尊重 `PROVIDER`（`agent/main.py` 拿掉 `if tenant is not None` 限制）

5　Should throw exception all the time（`sys.stderr` 並 `return 1`，不印空行吞錯）

6　資安規則（清空 `.env.example` 假 key、`.gitignore` 加上 `.env`）

目標：`make check L=L1` 達成 `6/6`（修完 ④⑤ 後 `make ask` 才會讀 `PROVIDER`）

Part 2 L1 簡單改

Moment 1：換模型，不改程式

L1 完成後領取課堂 token（自動寫入 `.env`）：

```
CLASS_CODE=brave-pearl-1023 NICKNAME=你的暱稱 make claim
```

```
PROVIDER=mock make ask Q="用一句話介紹台北"   → Mock 回答
PROVIDER=class make ask Q="用一句話介紹台北"  → 真實回應
```

Windows：`$env:PROVIDER="class"; uv run hw ask "…"`（📖 手冊 §1.5、§3）

Part 2 L1 簡單改

What we learn on Part 2

1 先讀 `make check` 的 `✗` 提示

2 查 `usage.jsonl` 最後一筆 `status`

3 對照檔案與手冊 `▼ 卡住了才打開` → 還有問題再舉手

Moment 1：換模型，不改程式

Part 3 L2 進階

L2 起步：先檢查、再跑批次

1. `make check L=L2`（出廠狀態 `0/3`）
2. `make run` → 寫出 `answers.json`、`usage.jsonl`
3. `make report` → 產生本機 `report.html`

📖 手冊 §3 L2｜LAB 14:10–14:25

Part 3 L2 進階

先檢查、再跑、再看 report

- 報表直接讀取你剛跑出的 `usage.jsonl`
- 一眼看清：哪個租戶（`tenant`）、哪個模型、花多少 token
- 找出成本異常與失敗請求（📖 手冊 §3 L2）



用量與成本報表

總請求	總 Token	總成本 (USD)	快取命中 (cached)	備援使用 (fallback)
10	1,856	$0.000215	9	0

依 Tenant 分組

名稱	請求數	Token 數	成本 (USD)
default	8	1,775	$0.000215
team-a	1	45	$0.000000
team-b	1	36	$0.000000

依 Provider 分組

名稱	請求數	Token 數	成本 (USD)
mock	10	1,856	$0.000215

成本最高的前 5 筆請求

時間	Tenant	別名	Provider	模型	Token 數	成本 (USD)	狀態
2026-10-13T03:24:55.183971+00:00	default	mock	mock	mock	1,498	$0.000215	ok
2026-10-13T03:24:55.159220+00:00	team-a	mock	mock	mock	45	$0.000000	ok
2026-10-13T03:24:55.160973+00:00	team-b	mock	mock	mock	36	$0.000000	ok
2026-10-13T03:24:55.162440+00:00	default	mock	mock	mock	33	$0.000000	ok
2026-10-13T03:24:55.163997+00:00	default	mock	mock	mock	38	$0.000000	ok

錯誤請求 (0)

時間	Tenant	別名	Provider	模型	Token 數	成本 (USD)	狀態
無資料							

make report 產生的本機報告畫面

Part 3 L2 進階

瞬間 2：成本看得見

- 同一個 prompt，不同模型
- token 與價格並排
- 教學用固定價格表，參考 ai.google.dev（2026-10-09）
- 先算數字，再優化

Part 3 L2 進階

成本 spike

- 某個 prompt，成本突然 10 倍
- 找出來
- 截斷 context
- 用你剛跑出的 `usage.jsonl`（不花額度）

Part 3 L2 進階

誰花的錢？

每筆請求帶 tenant（租戶＝誰花的）標籤

客戶-專案-環境

預算上限：請求送出前就擋下

超過 → 清楚的錯誤

Part 3 L2 進階

Fallback

- `fast` → `mock`
- 每次切換都寫進 log
- 真實服務一定會遇到 429／預算上限
- 這就是為什麼

Part 3 L2 進階

事故演習：provider 掛了

14:22　講師宣告

用 mock 模擬 429

誰的 Agent 還活著？

Part 3 L2 進階

L3：送出結果

執行 `make submit`（Windows：`uv run hw submit`）

全程 `mock` ＋ 固定價格表（0 次真實呼叫）

品質過關前提下，比誰總成本最低

14:30 準時截止｜📖 手冊 §3 L3（LAB 14:25–14:30）

Part 3 L2 進階

Part 3 回扣：成本看得見、失敗會叫

2　成本，看得見

3　失敗，會叫（有 fallback）

下一站：搬上雲

Part 4 上雲

排行榜

排行榜

暱稱	關卡	通過	tokens	成本	更新時間
demo-student	L3	通過	1856	0.000000	2026-10-13T03:25:03+00:00
ada-lin	L3	通過	1612	0.000187	2026-10-13T03:25:04+00:00
ben-chen	L3	通過	1744	0.000203	2026-10-13T03:25:04+00:00
finn-dev	L3	通過	1988	0.000231	2026-10-13T03:25:04+00:00
chloe-w	L3	通過	2390	0.000281	2026-10-13T03:25:04+00:00
eva-ml	L3	通過	3105	0.000364	2026-10-13T03:25:04+00:00
dan-ops	L3	未通過	1530	0.000176	2026-10-13T03:25:04+00:00

即時排行榜畫面

哪些做法讓成本最低？

Part 4 上雲

同一層，換個存在形式

今天：程式裡的 `gateway`（in-process：同一支程式內）

現在：包成小服務

再來：Cloud Run

邏輯不變

Part 4 上雲

講師的 class gateway

- 你用的 `class` 就是它
- `make claim`（或 `uv run hw claim`）+ 課堂代碼 → 限時 token
- `PROVIDER=class`（只有 `make ask` 會打真實上游）
- 位址：https://hwdc-class-gateway-408858043808.asia-east1.run.app

Part 4 上雲

CI 為什麼用 mock

- 測試：不花錢
- 測試：不依賴外部 API
- 結果固定
- 才分得出：程式錯？模型錯？

Part 4 上雲

CI 長這樣

```
- uses: astral-sh/setup-uv@v10.3.0
- run: uv sync
- run: uv run hw check L1
- run: uv run hw check L2
- run: uv run hw test -q
```

內建於 `.github/workflows/ci.yml`：全部離線通過才部署

Part 4 上雲

Live demo：上 Cloud Run

1. 本機跑起同一個 `server/app.py`（`/health`、`/leaderboard`）
2. CD 範本：`.github/workflows/deploy-cloud-run.yml`
3. Cloud Run + Secret Manager（`--set-secrets=...`，key 不進映像）

📖 手冊 §4 Part 4（含完整指令，課後可自己跑）

Part 4 上雲

Moment 3：失敗會叫

啟動前環境預檢：`deploy/check_env.py` 與 `server/app.py`

```
env -u CLASS_CODE -u SERVER_SECRET -u ADMIN_TOKEN uv run --extra server uvicorn server.app:app
```

啟動瞬間 → `啟動中止：設定強度不足`（`exit 1`）

log 明確指出缺哪個，絕不靜默回空白（📖 手冊 §4.1）

Part 4 上雲

踩坑清單

1. 本機連得到、雲端連不到
2. `.env` 被打包進部署內容
3. region 與配額

Part 4 上雲

What we learn on Part 4: notification on errors

1 換模型 -> 不改程式

2 成本 -> 可觀測

3 失敗 -> 可監控

Part 5 收尾

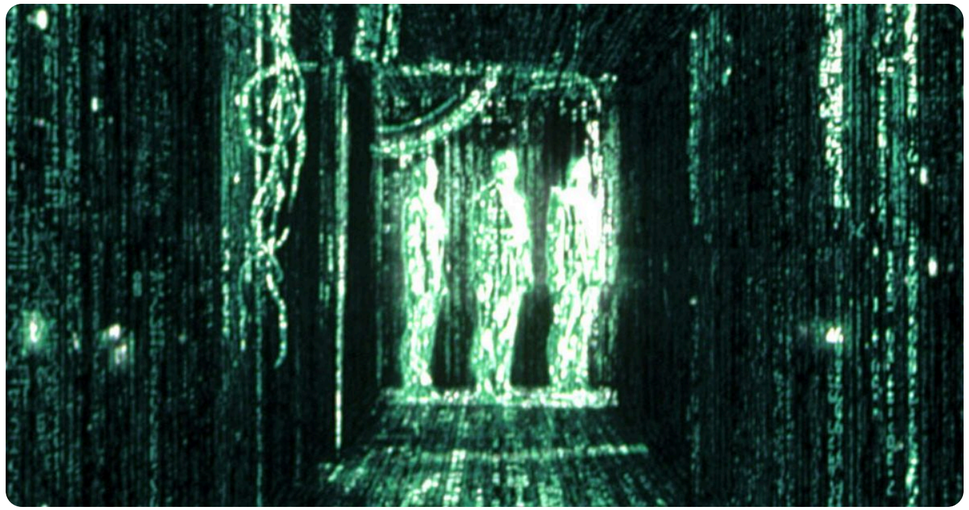
Key takeaway

1. 可離線驗證、可上雲的 `hwdc2026-starter`
2. 讀 `usage.jsonl` ／ `report.html` 的方法
3. 多租戶成本歸屬與預算規則
4. 上雲 checklist ＋「換模型、失敗會叫」的習慣

Part 5 收尾

Real world

- LiteLLM　開源，統一多家模型介面
- Cloudflare AI Gateway　託管型，快取、限流
- Portkey　閘道加可觀測性
- 選型依需求



Real world

Part 5 收尾

Next

接到自己的專案

教材與手冊：`jimmyliao.pse.is/hwdc2026`

Part 5 QnA

Q&A

Question and Feedback?

jimmyliao@jimmyliao.net