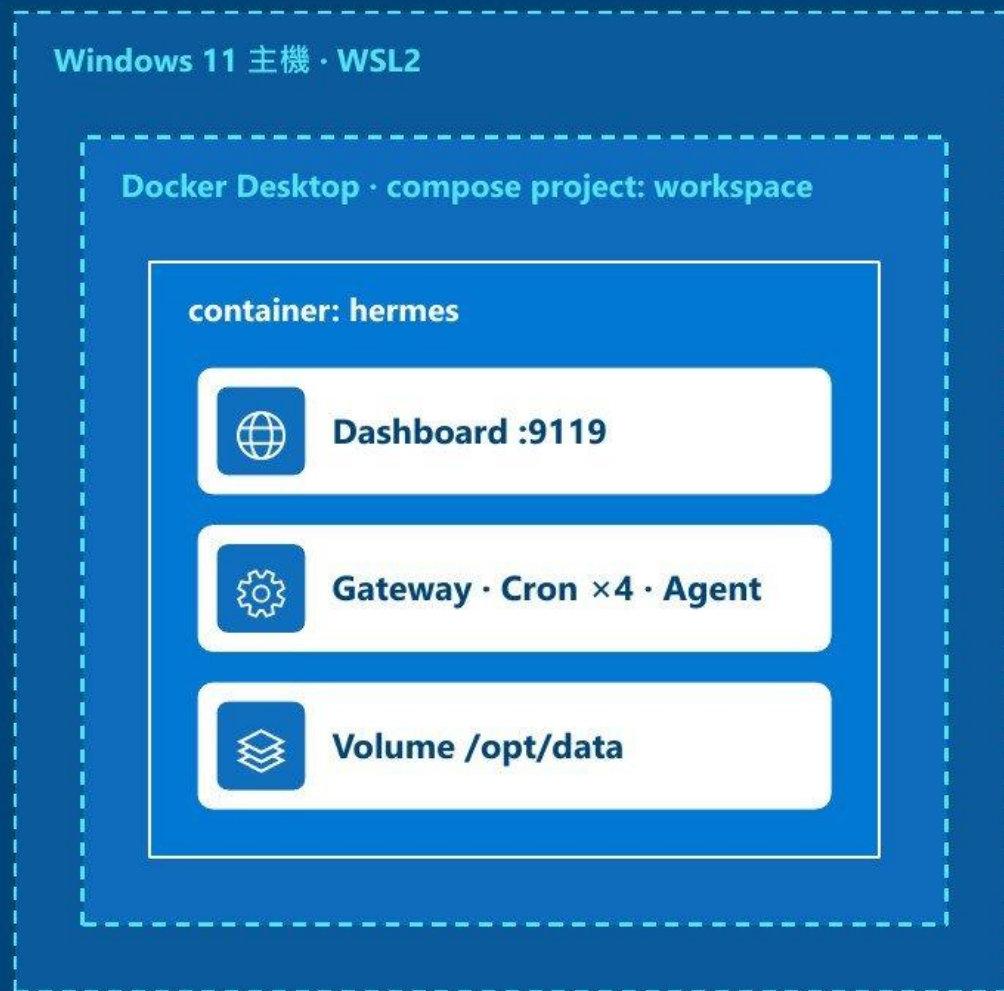


# Docker Hermes v4 Container 系統架構圖解

容器運行架構與架構完整性評估 — 以 Microsoft Azure 架構圖與 Well-Architected Framework 五大支柱呈現

nousresearch/hermes-agent:latest · Hermes Agent v0.21.5

盤點時間 2026-10-03 20:34 ( 依 docker-compose.yml 、 docker inspect / stats 與運行中容器實際檢視 ) · 修正同機容器盤點 · 評估頁改為圖文並列



# 閱讀指引：依 Azure Architecture Center 的文件結構組織

架構圖 → 工作流程 → 元件 → 考量 ( Well-Architected 五大支柱 ) → 後續步驟；v4 修正實測資料並讓每頁圖文並列

1

## Architecture 架構圖

整體架構、分層基礎設施、部署、啟動序列、程序樹、網路、資料、Gateway、LINE 互動時序

第 3、5-12 頁

2

## Workflow 工作流程

編號資料流說明，以及四個排程的共用執行模型與端到端流程

第 4、13-17 頁

3

## Components 元件

每個元件的實作方式與職責

第 4 頁

4

## Considerations 考量

安全與信任邊界、可觀測性與資源、五大支柱評估、完整性檢核矩陣

第 18-21 頁

5

## Next steps 後續步驟

改善路線圖 ( 含 compose 建議設定 ) 與維運速查

第 22-23 頁



## v4 更新重點

- 修正：同機實有 17 個運行中容器 ( v3 誤記 5 個 )，新增記憶體分配圖
- 更新：docker stats 於 20:34 重新取樣
- 圖解：支柱雷達圖、完整性環圈圖、路線圖時間軸
- 更新：每日Youtube 已手動試跑成功；排程報告共 4 份
- 改善：已知限制補上圖示，路線圖標註對應支柱

## 圖例

運算 / 容器服務

Agent · 腳本 · MCP

儲存 / volume

邊界 / 網路區域

資料流步驟

排程

外部服務 / 風險

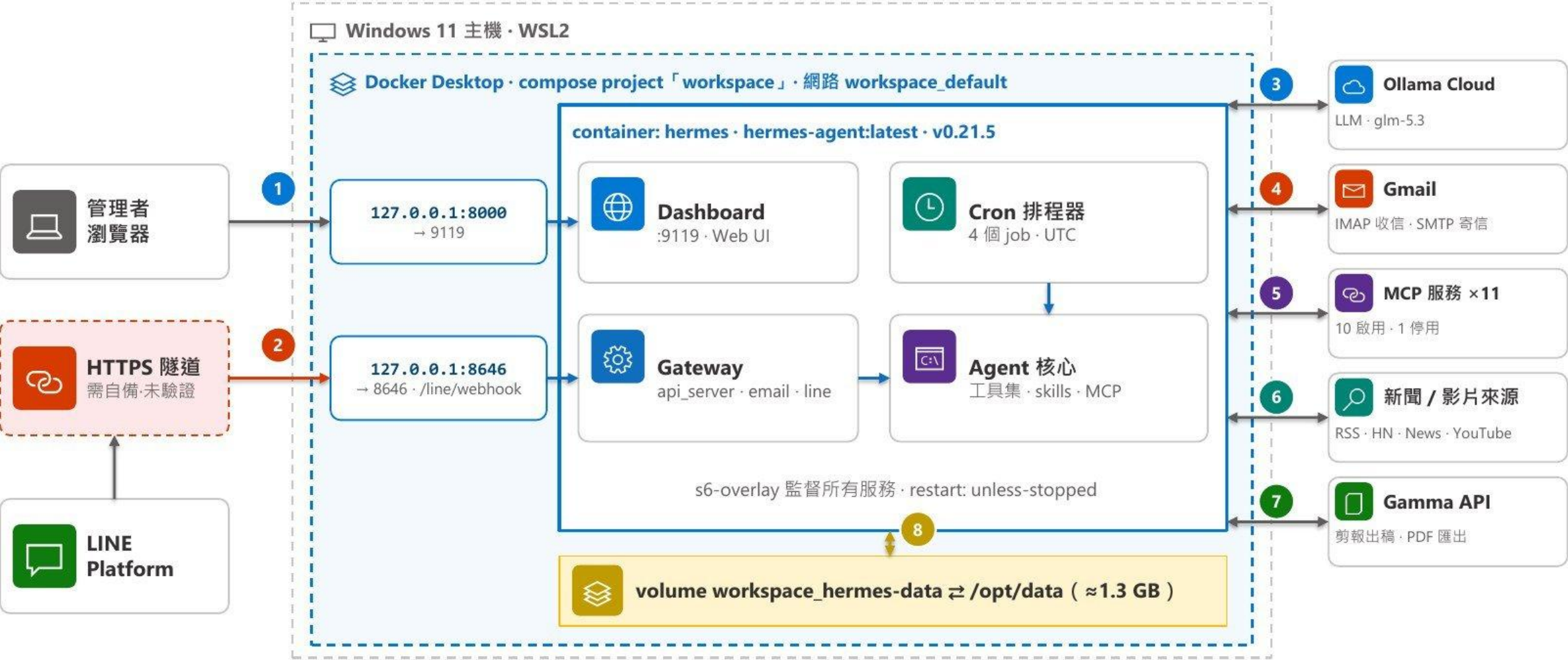
遞送 / 已落實

容器

回應 / 非同步

# 整體架構：一個容器、三個通道、四個排程、一個資料 volume

Windows 11 主機 → Docker Desktop → hermes 容器；外部服務透過本機埠或出站連線互動



① 管理者瀏覽器 → Dashboard    ② LINE webhook 經隧道 → Gateway    ③ LLM 推論    ④ Email 收寄    ⑤ MCP 工具    ⑥ 新聞與影片抓取    ⑦ Gamma 出稿    ⑧ 狀態寫入 volume

# 工作流程與元件：編號資料流與每個元件的職責

Azure Architecture Center 慣例：圖上的編號對應下方文字；元件表說明實作與責任

## 工作流程（對應第 3 頁編號）

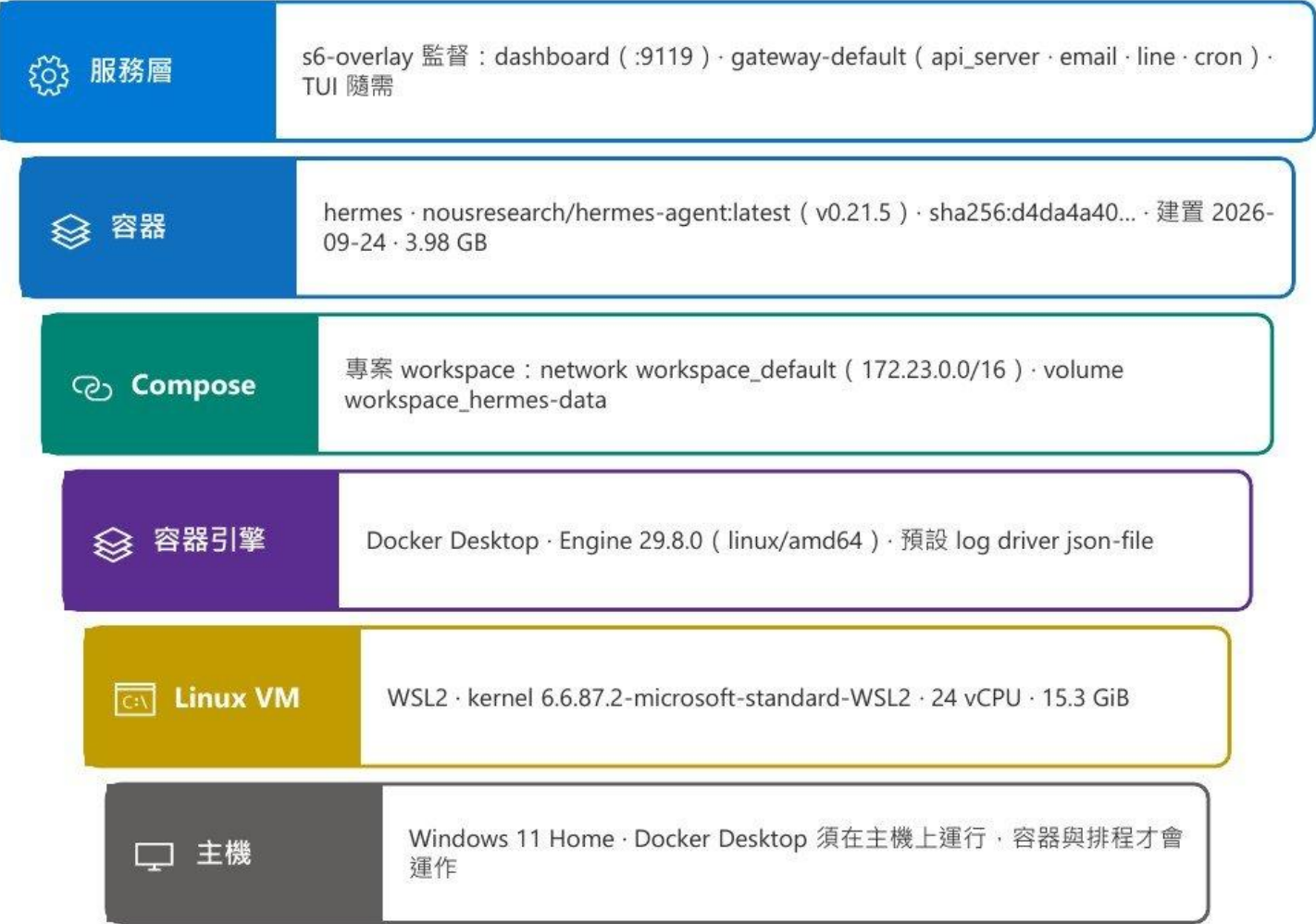
- 1 管理者以瀏覽器開啟 127.0.0.1:8000，通過 Basic Auth 後進入 Dashboard（容器 9119）。
- 2 LINE Platform 以 HTTPS 呼叫 webhook，經自備隧道到 127.0.0.1:8646；Gateway 驗簽後交給 Agent。
- 3 Agent 透過 Ollama Cloud API 以 glm-5.3 推論。
- 4 Gateway 定期以 IMAP 收信，並以 SMTP 寄出回覆與排程報告。
- 5 互動對話可呼叫 MCP 服務（OAuth）；排程以 no\_mcp 不載入。
- 6 排程前置腳本抓取新聞 RSS、HN、Google / Bing News 與 YouTube。
- 7 Meta Dots 日報經 Gamma Public API 出稿並匯出 PDF。
- 8 設定、排程、報告、對話狀態與日誌都寫入 volume /opt/data。

## 元件

| 元件         | 實作                              | 職責                           |
|------------|---------------------------------|------------------------------|
| Dashboard  | hermes dashboard : 9119         | Web 管理介面、設定、聊天（衍生 TUI）       |
| Gateway    | hermes gateway run              | 通道介接：api_server、email、line   |
| Cron 排程器   | Gateway 內建 ticker               | 每分鐘檢查到期 job；執行 4 個排程         |
| Agent 核心   | Python 3.13 · 工具集 · skills      | 推論迴圈、工具呼叫、技能載入               |
| s6-overlay | /init · s6-svscan · s6-log      | PID 1、服務監督、自動重啟、日誌輪替         |
| LLM        | Ollama Cloud · glm-5.3          | 文字推論（雲端，出站 HTTPS）            |
| Volume     | workspace_hermes-data           | HERMES_HOME：設定與狀態持久化         |
| Image      | hermes-agent:latest ( 3.98 GB ) | 唯讀程式碼、Python venv、Playwright |

# 分層基礎設施：從 Windows 主機到容器內服務

每一層的版本與規格皆為實測值；hermes 與同一 Docker Engine 上的其他專案容器共用資源



## 同一 Docker Engine：17 個運行中容器 · 7 組專案

|                 |                                                   |
|-----------------|---------------------------------------------------|
| hermes          | workspace · 127.0.0.1:8000 · 8646 · 無健康檢查         |
| rag15 ×2        | web · sqlserver · 埠 [REDACTED]                    |
| Workflow-hub ×4 | hub · worker · mcpdemo · sqlserver · 埠 [REDACTED] |
| eric12 ×2       | api · sqlserver · 埠 [REDACTED]                    |
| dbgov15 ×3      | web · worker · sqlserver · 埠 [REDACTED]           |
| mcpplatform ×3  | web · demoapis · sqlserver · 埠 [REDACTED]         |
| n8n ×2          | n8n-ubike · n8n-n8n-1 · 埠 5679 · 5678             |

## 記憶體分配 ( docker stats · 共 15.27 GiB )



⚠ hermes 未設 mem / cpu 上限 · 與另外 16 個容器共用 24 vCPU / 15.3 GiB

# 部署層：docker-compose.yml 一份檔案定義整個運行環境

單一服務 hermes；映像、啟動指令、埠、volume 與環境變數皆在此宣告

docker-compose.yml

services:

hermes:

image: nousresearch/hermes-agent:latest

container\_name: hermes

restart: unless-stopped

command: gateway run

ports:

- "127.0.0.1:8000:9119"

- "127.0.0.1:8646:8646" # LINE webhook

volumes:

- hermes-data:/opt/data

environment:

- HERMES\_DASHBOARD=1

- HERMES\_DASHBOARD\_BASIC\_AUTH\_USERNAME=\${...}

- HERMES\_DASHBOARD\_BASIC\_AUTH\_PASSWORD=\${...}

volumes:

hermes-data:

\${...} 為省略；實際值由同目錄 .env 注入，.env 已列入 .gitignore



## image

官方映像 hermes-agent:latest，運行版本 v0.21.5；未固定 digest，重新 pull 可能換版



## restart

unless-stopped：Docker Desktop 啟動後自動拉起；停機錯過的排程恢復後補跑一次



## command

gateway run：啟動 Messaging Gateway，含 cron 排程器、email、LINE 與 API server



## ports

兩個埠都只綁 127.0.0.1，僅本機可連；對外需另行建立 HTTPS 隧道



## volumes

命名 volume 掛到 /opt/data (HERMES\_HOME)，容器重建後設定與排程仍在

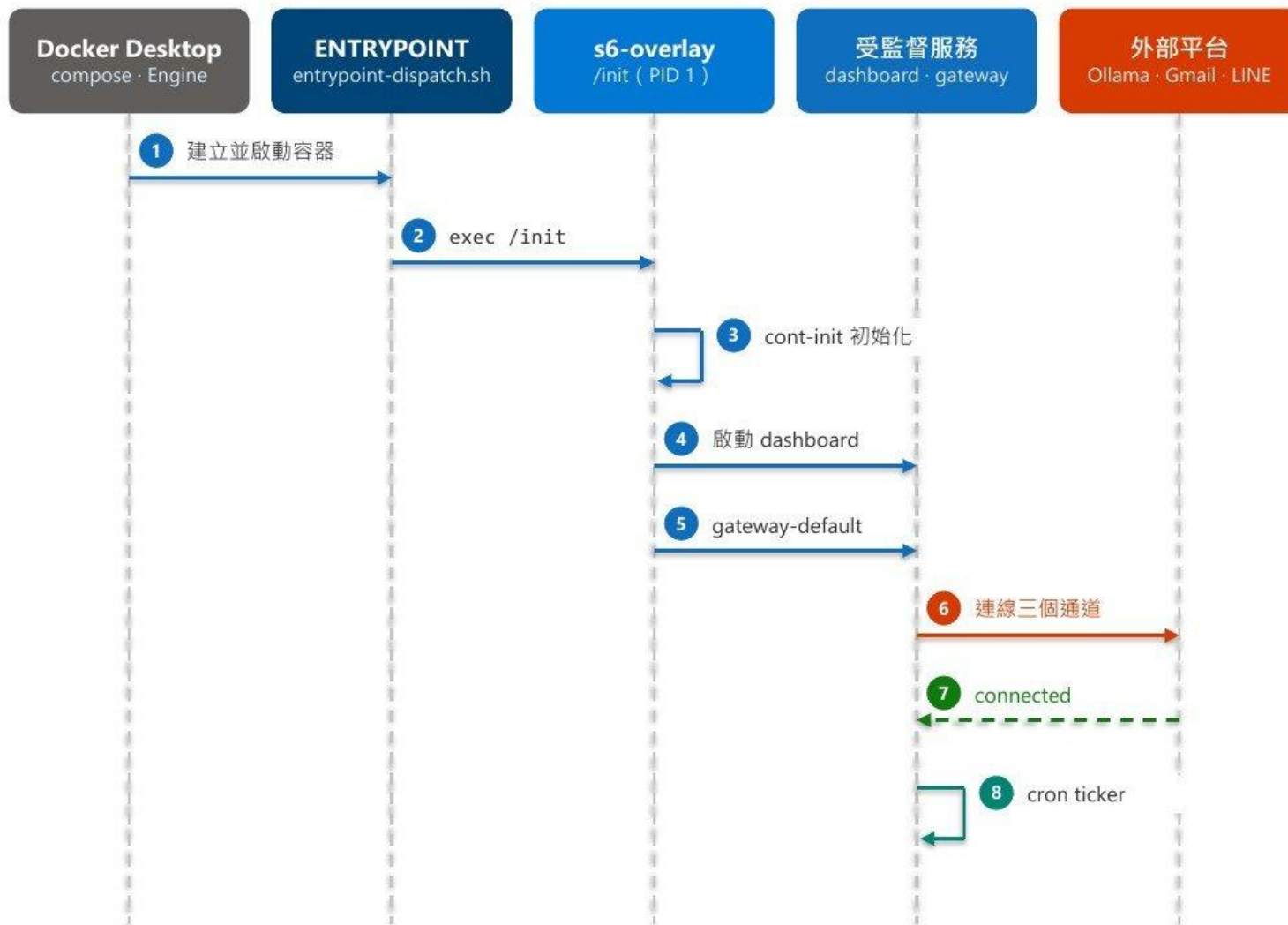


## 未宣告

healthcheck、logging 上限、mem / cpu 限制都沒有設定（見第 19、22 頁）

# 啟動序列：從 docker compose up 到排程就緒

UML 時序圖；PID 1 交給 s6-overlay，root 只做初始化，服務一律降權為 hermes 執行

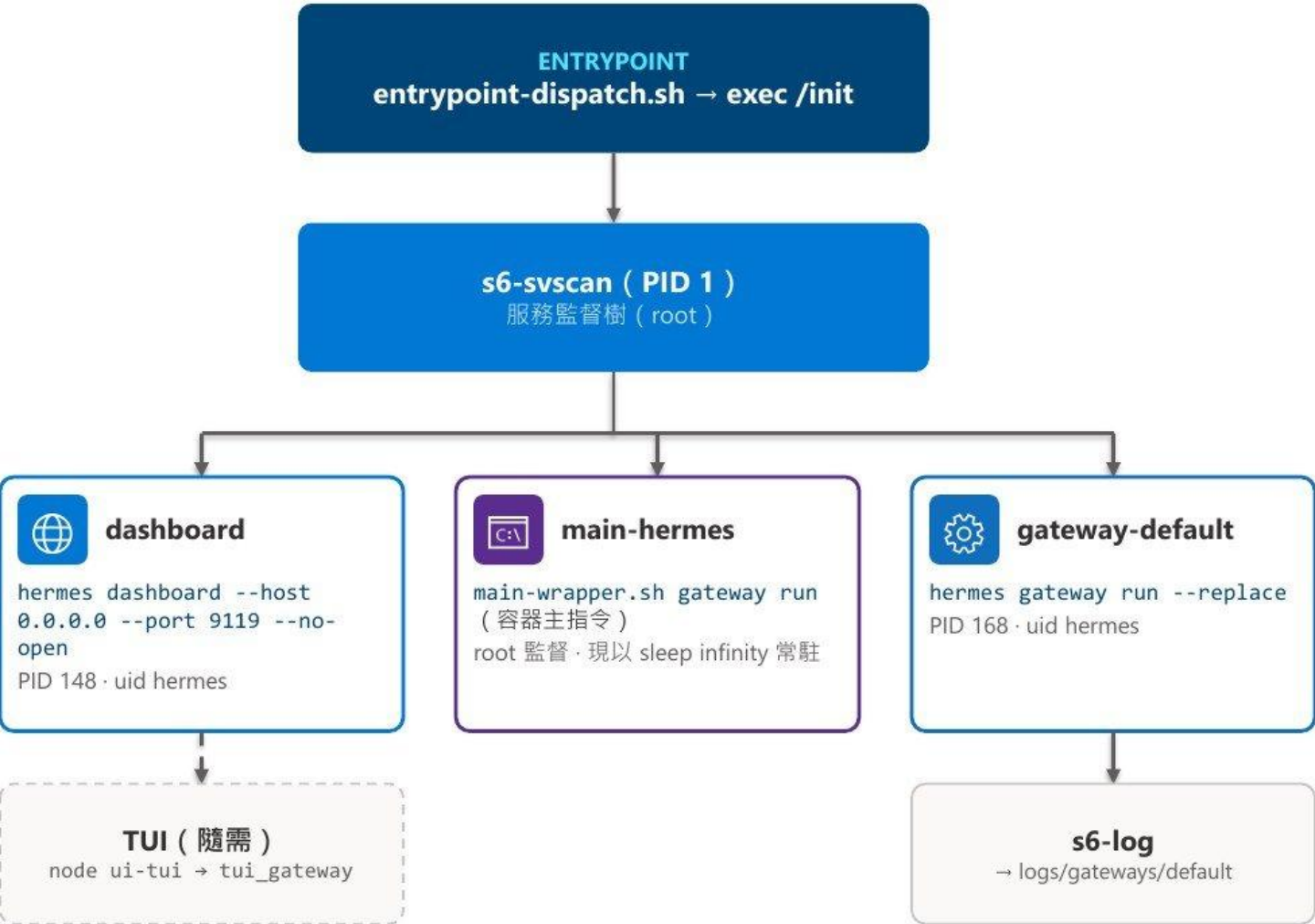


## 步驟說明

- 1 `docker compose up -d`：發布兩個 127.0.0.1 埠、掛載 volume、注入 .env 變數後啟動容器。
- 2 `entrypoint-dispatch.sh` 確認自己是 PID 1 後 `exec /init`；不是 PID 1 時改走直接啟動，沒有監督樹。
- 3 `cont-init (root)`：UID/GID 對應、volume chown、設定與 skills 同步，再校正監督權限與 profile。
- 4 `s6-rc` 以 hermes 身分啟動 Dashboard，監聽 0.0.0.0:9119。
- 5 `main-wrapper.sh` 處理 CMD「gateway run」，Gateway 由受監督服務 `gateway-default` 執行，輸出交給 `s6-log`。
- 6 Gateway 啟動 `api_server :8642`、email (IMAP / SMTP) 與 `line :8646 webhook`。
- 7 三個平台回報 `connected`，狀態寫入 `/opt/data/gateway_state.json`。
- 8 Cron ticker 每分鐘檢查到期 job；停機期間錯過的排程補跑一次。

# 容器內部：s6-overlay 監督的程序樹

PID 1 是 s6-svscan ; Dashboard 與 Gateway 為獨立受監督服務，TUI 由 Dashboard 隨需衍生



## 1 單容器、雙服務

同一容器長駐 Dashboard ( Web UI ) 與 Gateway ( 通道 + 排程 ) ; 開啟 Dashboard 聊天時才衍生 TUI 程序

## 2 崩潰自動重啟

s6-supervise 重新拉起失效服務 ; compose 的 restart: unless-stopped 再提供第二層保護

## 3 最小權限

dashboard、gateway、TUI 皆以 hermes 執行 ; root 只跑 s6 監督與 cont-init

## 4 日誌輪替

gateway 輸出經 s6-log 保留 10 檔 × 1 MB ; 容器 stdout 則交給 Docker json-file

# 網路與連接埠：只有本機回環入口，其餘皆為出站連線

hermes 單獨接在 workspace\_default，與同機其他專案容器的網路彼此隔離

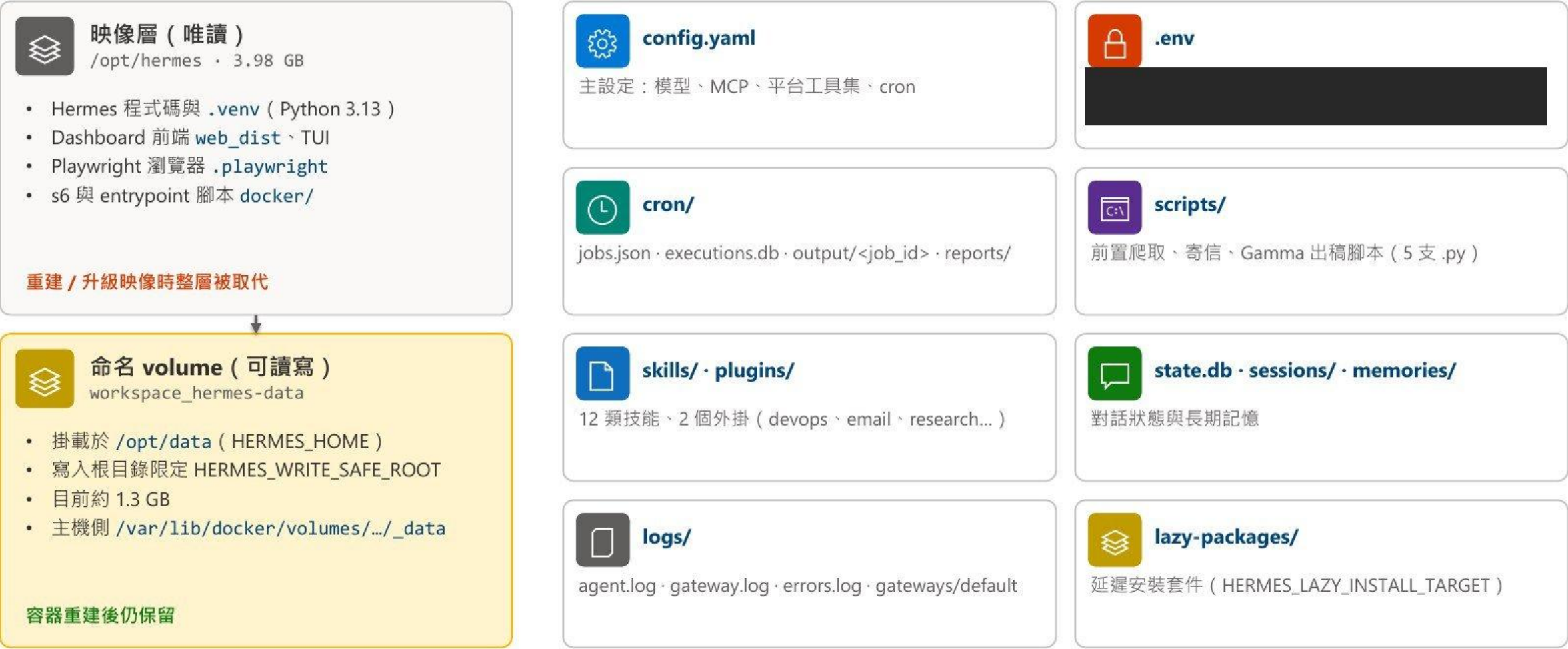


| 入口           | 主機綁定               | 認證與保護                                                 | 備註                                            |
|--------------|--------------------|-------------------------------------------------------|-----------------------------------------------|
| Dashboard    | 127.0.0.1:8000     | Basic Auth ( 帳密由 .env 注入 )                            | 只在本機瀏覽器開啟；/api/health 免認證，可作健康檢查              |
| LINE webhook | 127.0.0.1:8646     | X-Line-Signature ( HMAC-SHA256 ) + LINE_ALLOWED_USERS | LINE 需 HTTPS 回呼，必須自備隧道；本檢視未驗證隧道是否存在           |
| api_server   | 容器內 127.0.0.1:8642 | API_SERVER_KEY ( .env )                               | 未對應到主機埠，僅容器內可連                                |
| 容器出站         | — ( 無入站 )          |                                                       | LLM、Email、LINE 回覆、新聞、YouTube、Gamma、MCP 皆為主動連出 |

同機其他專案的容器 ( 資料庫、API、n8n 等 ) 位於各自的 bridge 網路，不與 hermes 互通。

# 資料持久化：映像是唯讀程式碼，`/opt/data` 才是狀態

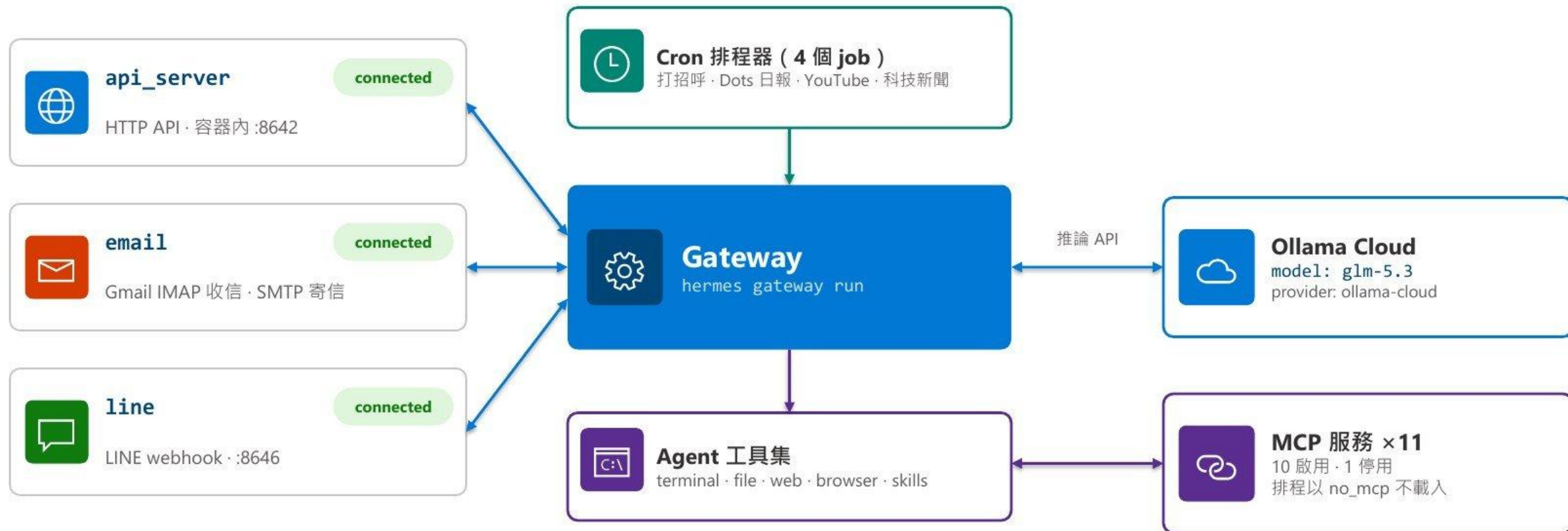
升級映像或重建容器時，只有 volume 內的設定、排程、腳本與對話記錄會被保留



備份重點：cron/jobs.json、.env、scripts/。4 個 job 定義只存在 volume，工作區 scripts/ 只有 5 支中的 3 支，且沒有異地備份。

# Gateway 樞紐：三個通道已連線，推論交給雲端模型

狀態取自 gateway\_state.json ( 2026-10-03 12:31 UTC ) : api\_server、email、line 皆為 connected ; 4 個排程也由同一程序執行

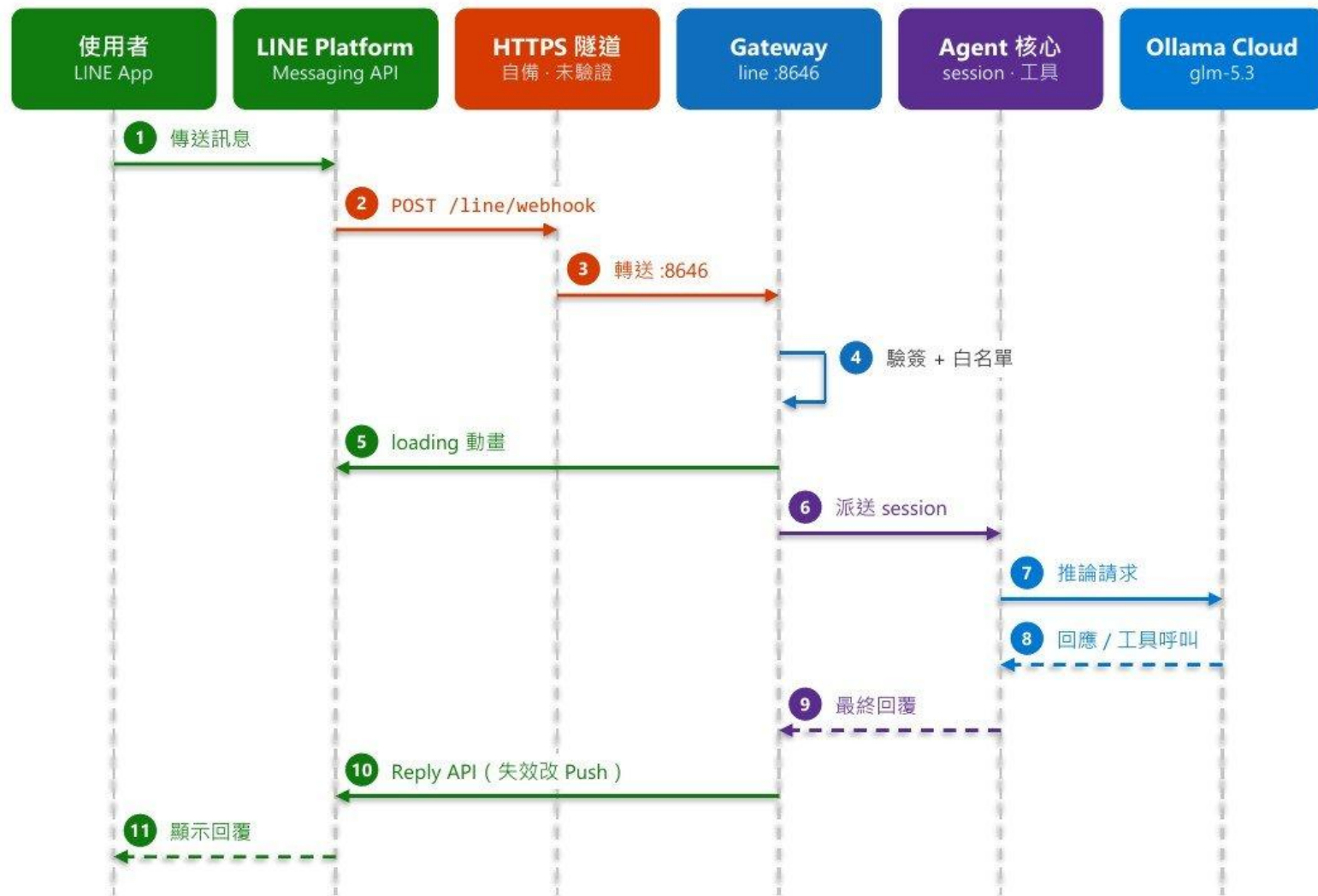


config.yaml 所列 11 個 MCP 服務；無人值守環境沒有 OAuth token 的服務會持續輸出警告，不影響排程 ( Gamma 改走 API 金鑰 )



# 互動時序：一則 LINE 訊息如何進出容器

UML 時序圖；入站只有 webhook 一條路，回覆由容器主動呼叫 LINE Messaging API 送出



## 入口

LINE 只呼叫 HTTPS webhook，需自備隧道轉到 127.0.0.1:8646；GET /line/webhook/health 可供隧道探測。



## 驗證

$X\text{-Line-Signature} = \text{base64}(\text{HMAC-SHA256}(\text{Channel Secret}, \text{body}))$   
・常數時間比對；來源須在 LINE\_ALLOWED\_USERS 白名單內。



## 回覆

優先使用免費、單次、約 60 秒有效的 reply token；失效時改用計費的 Push API。



## 慢回應

推論超過 45 秒先送按鈕訊息，使用者點擊後以新的 reply token 取回答案。



## 限制

每次最多 5 則訊息物件，文字每 4500 字切段；對話狀態寫入 /opt/data/state.db。

# 排程總覽：四個 job 在台北時間錯開執行

cron 以 UTC 計，台北 = UTC+8；全部遞送到 email，除打招呼外都以前置腳本取資料並限制工具



| 項目           | 每天打招呼             | Meta Dots 應用日報              | 每日Youtube                  | 每天科技新聞                                 |
|--------------|-------------------|-----------------------------|----------------------------|----------------------------------------|
| job id       | 4c9b336c55e5      | 7628b3aaae0b                | e53422a12c1f               | b71a65ee0d9c                           |
| 前置腳本         | 無                 | fetch_meta_dots_news.py     | fetch_youtube_dots_muse.py | fetch_ai_news.py                       |
| 資料來源         | — ( Agent 直接撰寫 )  | Google / Bing News RSS      | YouTube 搜尋 ( 今天上傳 )        | TechCrunch · Verge · MIT TR · Ars · HN |
| 工具           | 未設白名單；不需工具        | 白名單 + no_mcp                | 白名單 + no_mcp               | 白名單 + no_mcp                           |
| 產出與遞送        | 純文字早安信            | Gamma 剪報 ( 連結 + PDF ) + 純文字 | HTML 影片報告 + 純文字            | HTML 圖文信 + 純文字                         |
| 最近一次 ( UTC ) | 10-02 22:01    ok | 10-03 00:08    ok           | 10-03 09:45    ok ( 手動試跑 ) | 10-03 06:04    ok                      |
| 下次執行 ( UTC ) | 10-03 22:00       | 10-04 00:00                 | 10-04 01:00 ( 首次排程 )       | 10-04 06:00                            |

# Cron 執行模型：四個 job 共用的六個階段

到期判斷、前置腳本、Agent、產出、遞送、記錄；成功與否以回覆第一行是否為 [CRON\_FAILURE] 判定



## 到期判斷：準時、延遲、補跑 (cron/jobs.py)

| 分類       | 條件 (以每日 job 為例)         | 行為         |
|----------|-------------------------|------------|
| on_time  | 延遲 ≤ 5 分鐘 (ticker 正常誤差) | 照常執行       |
| late     | 5 分鐘 < 延遲 ≤ 寬限 2 小時     | 立即執行       |
| catch_up | 延遲 > 2 小時 (例如停機後)       | 累積的錯過只補跑一次 |

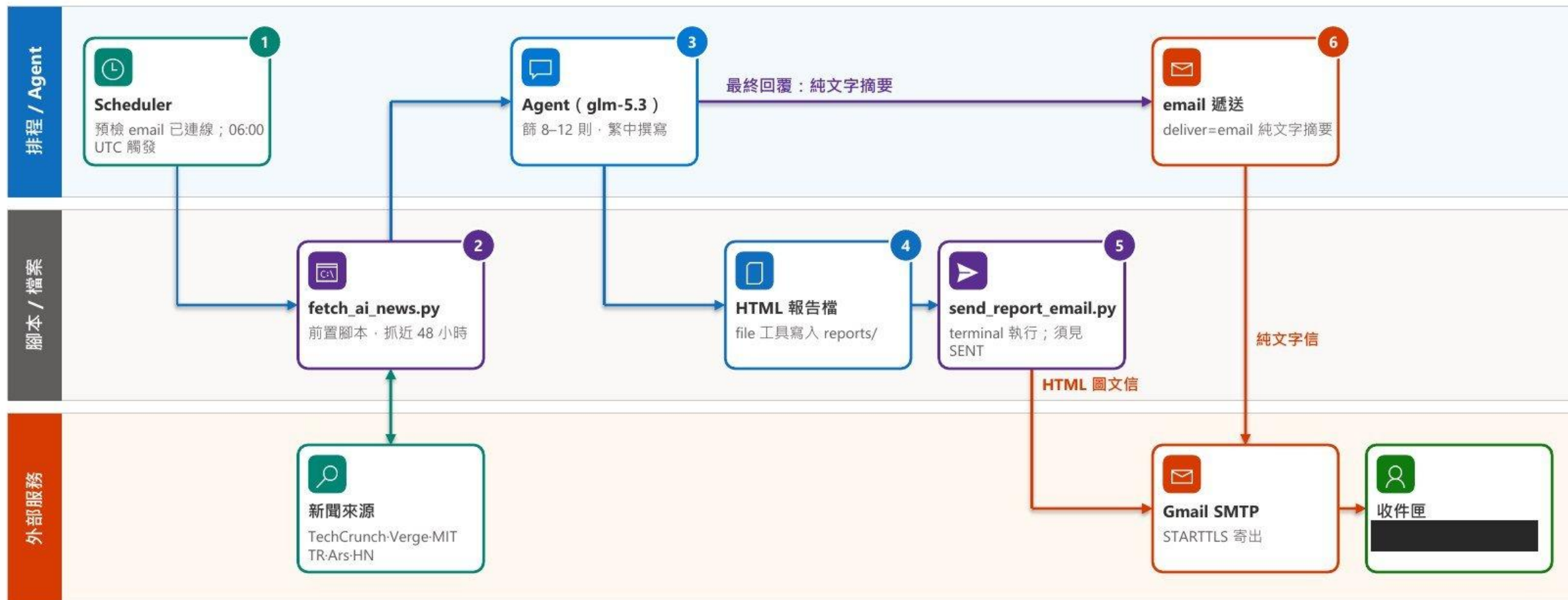
寬限 = 週期的一半，限制在 2 分鐘到 2 小時之間，所以每日 job 的寬限是 2 小時。  
cron.catch\_up\_missed: true (本機設定)：超過寬限仍補跑一次；設為 false 則略過。

## 成功與失敗的判定

- 腳本寄信的 job 須看到 SENT (Dots 為 SENT gamma clipping) 才算寄出
- 失敗時回覆第一行單獨輸出 [CRON\_FAILURE]，Hermes 只認這個標記
- 排程中需人工核准的危險指令一律拒絕 (approvals.cron\_mode 預設 deny)
- 目前 4 個 job：last\_status 皆為 ok，failure\_streak 皆為 0
- 實務以「是否收到信」為準；last\_status 只反映 Agent 回報

# 排程流程 ①：每天科技新聞 ( b71a65ee0d9c )

cron 0 6 \* \* \* ( UTC ) = 台北 14:00 → 前置爬取 → Agent 撰寫 → 寄出 HTML 圖文信，並再寄一封純文字摘要

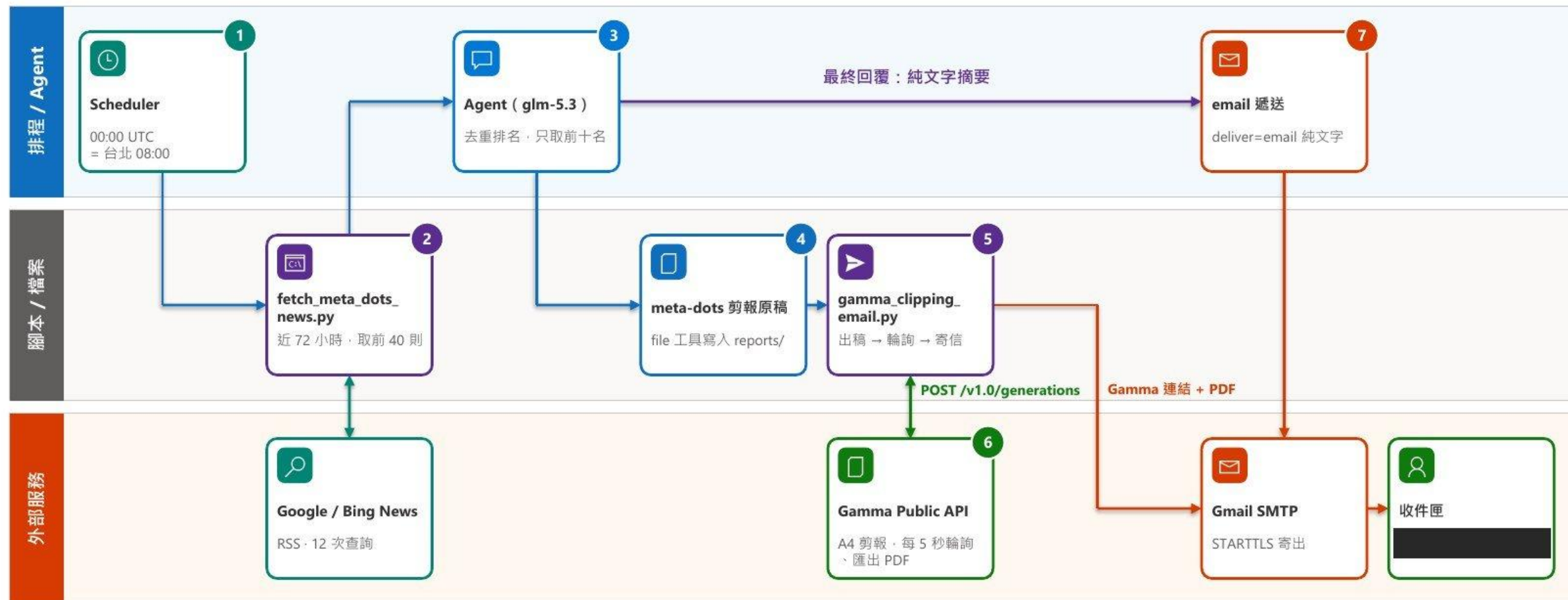


成功條件：send\_report\_email.py 輸出 SENT，否則回覆第一行必須是 [CRON\_FAILURE]。結果：每天收到兩封信（HTML 圖文信 + 純文字摘要）。

工具白名單 terminal · file · web · browser · code\_execution · skills · no\_mcp；排程中需核准的危險指令一律被拒。

## 排程流程 ②：Meta Dots 應用日報 ( 7628b3aaae0b )

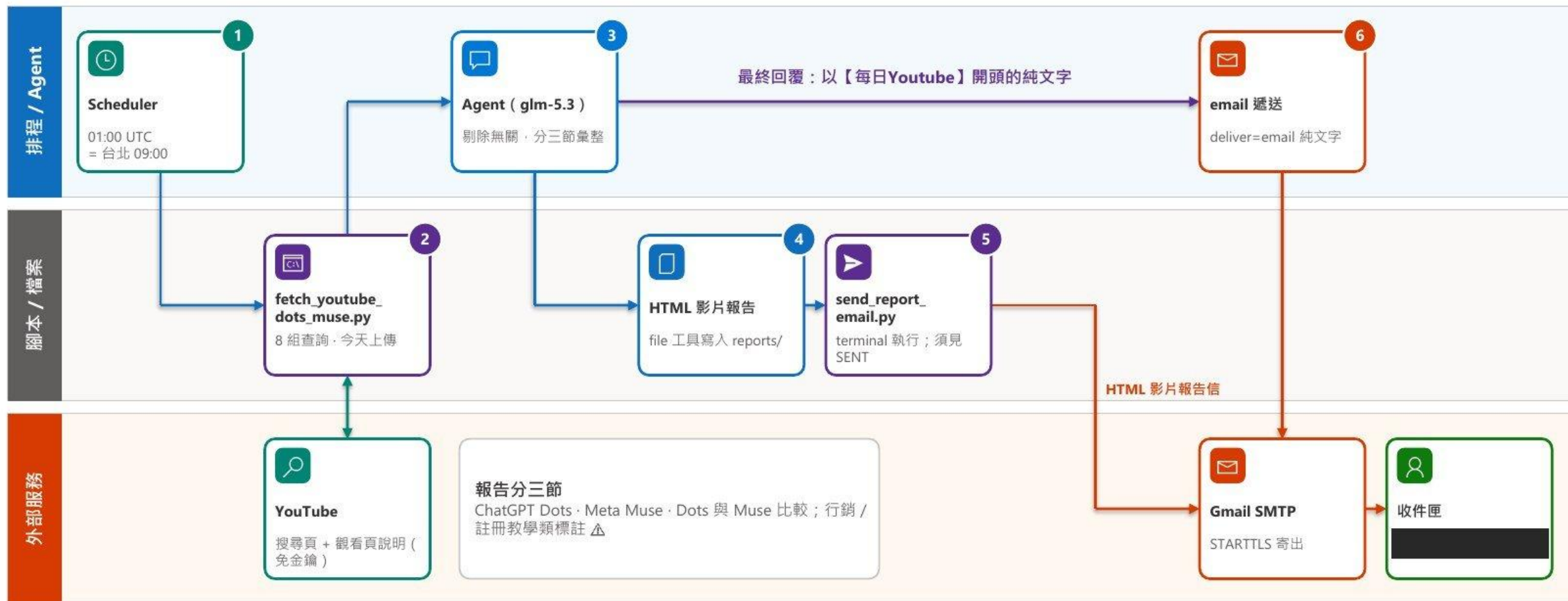
cron 0 0 \* \* \* ( UTC ) = 台北 08:00 → 前置爬取 → Agent 排名前十 → Gamma 出稿 → 寄出剪報連結與 PDF



成功條件：gamma\_clipping\_email.py 輸出 SENT gamma clipping，否則回覆第一行必須是 [CRON FAILURE]。結果：每天兩封信 ( Gamma 連結 + PDF、純文字摘要 )。

# 排程流程 ③：每日Youtube ( e53422a12c1f )

cron 0 1 \* \* \* ( UTC ) = 台北 09:00 → 搜尋當日影片 → Agent 篩選分組 → 寄出 HTML 影片報告與純文字摘要



成功條件：send\_report\_email.py 輸出 SENT，否則回覆第一行必須是 [CRON\_FAILURE]。結果：每天兩封信（HTML 影片報告 + 純文字摘要）。10-03 09:45 UTC 手動試跑 ok；首次排程執行 10-04 01:00 UTC。

風險：腳本解析 YouTube 頁面內嵌的 ytInitialData，頁面結構或同意頁改版時可能抓不到資料；單一查詢失敗不中斷，全部失敗才回報 [CRON\_FAILURE]。

# 安全與信任邊界：誰能碰到什麼

從網際網路到機密 volume 的五道邊界，與目前已落實及待改善的項目



# 可觀測性與資源：看得到什麼、還缺什麼

docker stats / inspect 實測 ( 2026-10-03 20:34 ) ；日誌分散在 Docker 與 volume 兩處，沒有健康檢查與告警

## 資源用量 ( docker stats )

|                                                             |                                                                     |
|-------------------------------------------------------------|---------------------------------------------------------------------|
| <div>CPU</div> <div>0.89%</div> <div>閒置時</div>              | <div>記憶體</div> <div>1.44 GiB</div> <div>/ 15.27 GiB · 曾達 1.78</div> |
| <div>PIDs</div> <div>86</div> <div>程序與執行緒 ( 曾達 113 )</div>  | <div>網路 I/O</div> <div>1.55 GB</div> <div>收 / 送 120 MB</div>        |
| <div>磁碟 I/O</div> <div>723 MB</div> <div>讀 / 寫 602 MB</div> | <div>資源上限</div> <div>未設定</div> <div>mem=0 · cpus=0</div>            |

## 日誌與狀態訊號

| 訊號         | 位置                                               | 保留 / 輪替                |
|------------|--------------------------------------------------|------------------------|
| 容器 stdout  | docker logs hermes ( json-file )                 | 未設 max-size · 持續成長     |
| Gateway 輸出 | /opt/data/logs/gateways/default                  | s6-log : 10 檔 × 1 MB   |
| Agent / 錯誤 | logs/agent.log · gateway.log · errors.log        | 程式內輪替 ( errors.log.1 ) |
| 平台狀態       | /opt/data/gateway_state.json                     | 即時覆寫                   |
| 排程執行       | cron/executions.db · output/ · usage_audit.jsonl | 累積，無清理策略               |
| 啟動記錄       | logs/container-boot.log · gateway-starts.log     | 累積                     |



### HEALTHCHECK 未設定

映像與 compose 都沒有定義；docker ps 不會顯示 healthy。Dashboard 已提供 /api/health ( 回 200 ) 可直接採用。



### 重啟記錄

RestartCount 0 ( 未發生崩潰自動重啟 )。container-boot.log 自 10-01 起共 11 次啟動；10-02 13:49 重建後的 5 次，前次結束都記為 unclean。



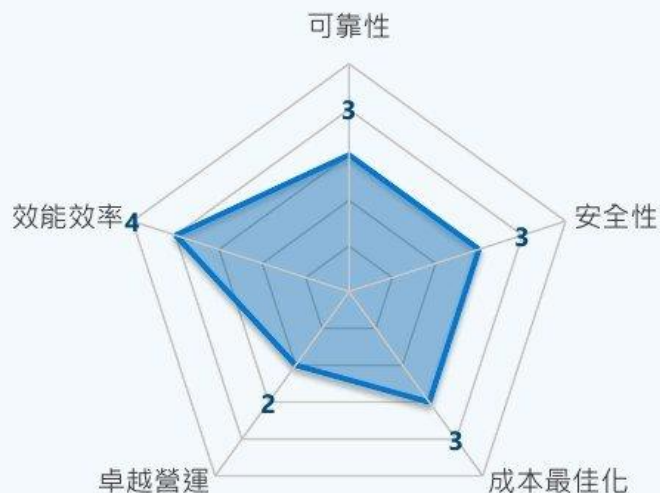
### 排程健康

4 個 job 的 last\_status 皆 ok、failure\_streak 皆 0；失敗只能從信箱或 cron runs 發現，沒有主動告警。

# 架構完整性評估：Azure Well-Architected Framework 五大支柱

以實測證據評估現況與缺口；成熟度為本次盤點的定性評分（滿分 5）

## 成熟度雷達（0-5）



### 評分依據

2 = 關鍵實務缺漏

3 = 基本控制到位，但有明確缺口

4 = 目前負載下充足

雷達中心為 0、外圈為 5；處理順序見第 22 頁

### 可靠性 Reliability

3 / 5

#### 現況

s6 監督自動重啟 + restart: unless-stopped；狀態全在命名 volume；停機錯過的排程補跑一次。

#### 缺口

沒有 HEALTHCHECK；單一容器、單一主機；volume 無異地備份；依賴主機與 Docker Desktop 持續運行。

### 安全性 Security

3 / 5

#### 現況

埠只綁 127.0.0.1；Basic Auth；LINE HMAC 驗簽 + 白名單；.env 權限 600；服務以 hermes 執行；排程 no\_mcp。

#### 缺口

### 成本最佳化 Cost Optimization

3 / 5

#### 現況

本機運行，沒有雲端主機費；前置腳本先篩資料再交給 LLM，減少 token；排程時段錯開。

#### 缺口

沒有 LLM 與 Gamma 點數的用量監控或告警；usage\_audit.jsonl 已有紀錄但未被利用。

### 卓越營運 Operational Excellence

2 / 5

#### 現況

compose 宣告式部署；cron list / runs / doctor 可查；四份排程配置報告與 v1-v4 架構圖解。

#### 缺口

job 定義與 5 支腳本只在 volume（工作區只有 3 支），未納入版控；沒有 CI、變更記錄與失敗告警。

### 效能效率 Performance Efficiency

4 / 5

#### 現況

24 vCPU / 15.3 GiB；hermes 實測 1.4-1.8 GiB、CPU < 2%；四個排程分散在 06、08、09、14 時。

#### 缺口

未設 mem / cpu 上限，與同機 16 個容器（含 5 個 SQL Server）共用；長任務受雲端 LLM 延遲影響。

# 完整性檢核矩陣：14 個架構面向的涵蓋與落實狀態

「涵蓋頁」指本簡報說明的位置；「狀態」指系統本身是否已完整落實

| 面向      | 涵蓋頁   | 證據 / 說明                                 | 狀態   | 面向      | 涵蓋頁   | 證據 / 說明                        | 狀態   |
|---------|-------|-----------------------------------------|------|---------|-------|--------------------------------|------|
| 部署定義    | 6     | docker-compose.yml、.env ( 已 gitignore ) | ✓ 完整 | 排程與作業   | 13-17 | jobs.json 4 個 job、scripts/ 5 支 | ✓ 完整 |
| 主機與執行環境 | 5     | docker info：Engine 29.8.0、WSL2、24 vCPU  | ✓ 完整 | 身分與機密   | 18    |                                | △ 部分 |
| 啟動與初始化  | 7     | entrypoint-dispatch.sh、cont-init.d      | ✓ 完整 | 可觀測性    | 19    | 有日誌與狀態檔，無集中檢視與告警               | △ 部分 |
| 程序與監督   | 8     | ps -eo pid,ppid、s6-rc.d、s6-log          | ✓ 完整 | 版本與變更管理 | 6、22  | image latest；腳本 3 / 5 在工作區     | △ 部分 |
| 網路與入口   | 9     | docker inspect 埠與 IP、network inspect    | ✓ 完整 | 健康檢查    | 19    | Config.Healthcheck = null      | ✗ 缺  |
| 資料與持久化  | 10    | /opt/data 目錄結構、du -sh 1.3 GB            | ✓ 完整 | 備份與還原   | 10、22 | 無備份排程、未做還原演練                   | ✗ 缺  |
| 通道整合    | 11-12 | gateway_state.json、LINE adapter 原始碼     | ✓ 完整 | 災難復原    | 22    | 單一主機，未定義 RTO / RPO             | ✗ 缺  |



8 ✓ 完整

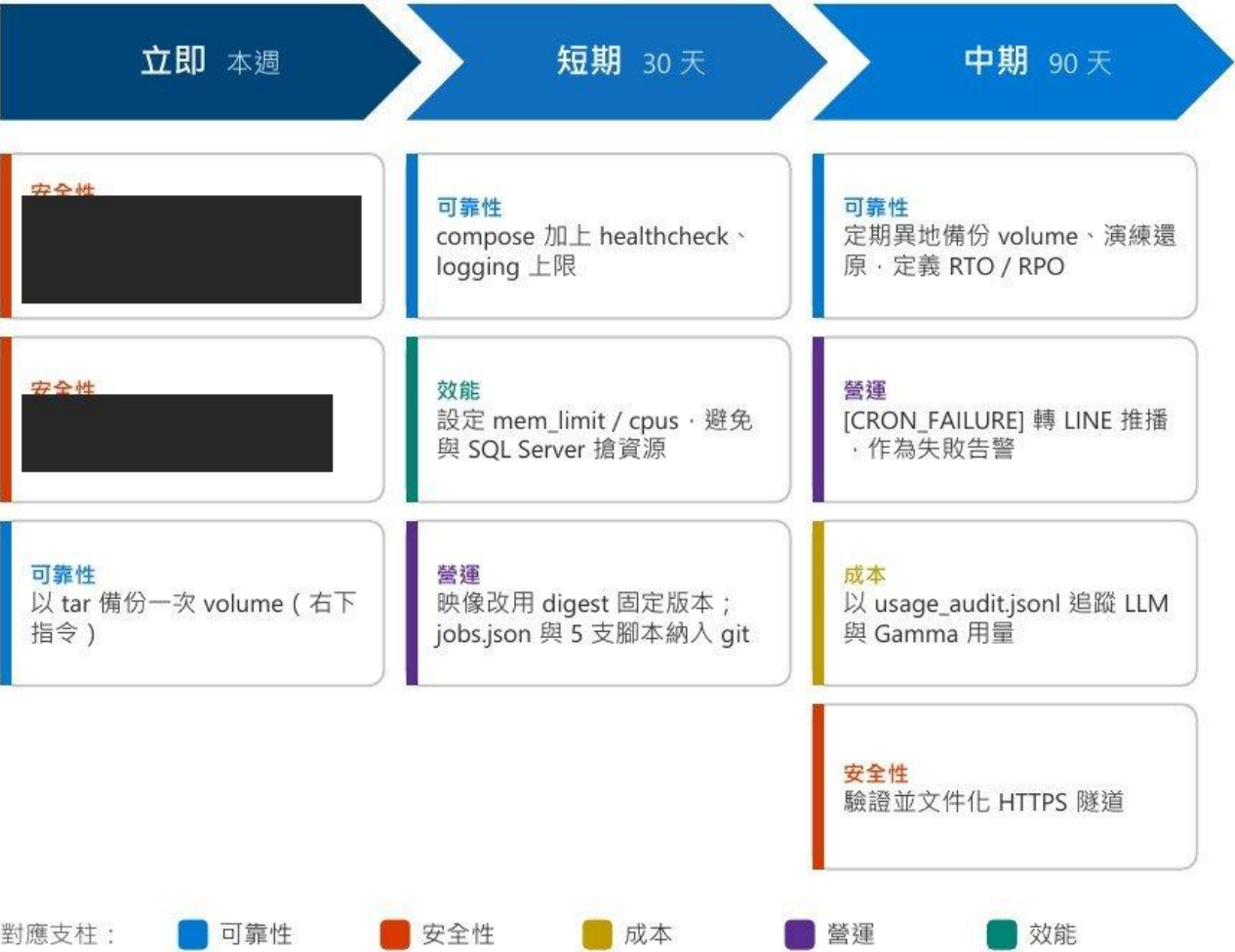
3 △ 部分

3 ✗ 缺

文件涵蓋全部 14 項；系統本身 8 項完整。  
部分：身分與機密、可觀測性、版本與變更管理  
缺：健康檢查、備份與還原、災難復原 ( 改善順序見第 22 頁 )

# 後續步驟：改善路線圖與建議設定

依五大支柱缺口排序，卡片色條標示對應支柱；右側為可加入 docker-compose.yml 的建議片段



docker-compose.yml ( 建議新增 )

```
services:
  hermes:
    image: nousresearch/hermes-agent@sha256:d4da4a40...
    healthcheck:
      test: ["CMD", "curl", "-fsS",
        "http://127.0.0.1:9119/api/health"]
      interval: 60s
      timeout: 5s
      retries: 3
    logging:
      options: { max-size: "10m", max-file: "3" }
    mem_limit: 4g # 實測 1.4-1.8 GiB
    cpus: 4
```

PowerShell：備份 volume ( 唯讀掛載 )

```
docker run --rm -v workspace_hermes-data:/data:ro `
-v ${PWD}\backup:/backup alpine `
tar czf /backup/hermes-data.tgz -C /data .
```

# 維運速查與已知限制

PowerShell 指令；操作排程一律加 -u hermes

| 需求         | 指令                                                                                 |
|------------|------------------------------------------------------------------------------------|
| 啟動 / 更新    | <code>docker compose up -d</code>                                                  |
| 查看容器 log   | <code>docker logs -f hermes</code>                                                 |
| 改 .env 後重啟 | <code>docker restart hermes</code>                                                 |
| 健康狀態       | <code>curl http://127.0.0.1:8000/api/health</code>                                 |
| 資源用量       | <code>docker stats hermes --no-stream</code>                                       |
| 列出排程       | <code>docker exec -u hermes hermes hermes cron list</code>                         |
| 立即執行一次     | <code>docker exec -u hermes hermes hermes cron run &lt;job_id&gt;</code>           |
| 執行記錄       | <code>docker exec -u hermes hermes hermes cron runs &lt;job_id&gt;</code>          |
| 排程診斷       | <code>docker exec -u hermes hermes hermes cron doctor</code>                       |
| 暫停 / 恢復    | <code>docker exec -u hermes hermes hermes cron pause resume &lt;job_id&gt;</code>  |
| 更新腳本 (複製)  | <code>docker cp scripts\x.py hermes:/opt/data/scripts/</code>                      |
| 更新腳本 (改權限) | <code>docker exec -u root hermes chown hermes:hermes /opt/data/scripts/x.py</code> |
| Dashboard  | <code>http://127.0.0.1:8000</code>                                                 |

## 已知限制

- **-u hermes**  
避免以 root 寫入 jobs.json 造成權限錯亂
- **.env 變更**  
需重啟容器才會載入
- **失敗偵測**  
Hermes 只認回覆第一行 [CRON\_FAILURE]；以「是否收到信」為準，別只看 last\_status
- **時區**  
cron 以 UTC 計算；台北 = UTC+8
- **停機補跑**  
容器停止期間不執行；恢復後錯過的排程補跑一次 (多次錯過只補一次)
- **外部相依**  
Gamma 點數用盡時 Dots 日報失敗；YouTube 改版可能讓影片抓取失效

# 重點回顧

Docker Hermes Container 運行架構一頁帶走

1

個容器

hermes · 由 s6-overlay 監督 Dashboard 與 Gateway

2

個本機埠

8000 Dashboard、8646 LINE，皆綁 127.0.0.1

1

個 volume

/opt/data 保存設定、排程、腳本與記憶 ( ≈1.3 GB )

3

個通道連線

api\_server · email · line

4

個排程 job

打招呼、Dots 日報、每日 Youtube、科技新聞，皆遞送到 email

運行重點：容器常駐，排程才準時（停機錯過的補跑一次）；狀態全在 /opt/data；對外只有 LINE webhook 需要 HTTPS 隧道。

完整性：14 個面向中 8 項完整、3 項部分、3

資料來源：docker-compose.yml、docker inspect / stats / info、容器內程序與原始碼、config.yaml、jobs.json、gateway\_state.json、docs/ 四份排程配置報告；盤點時間 2026-10-03 20:34。