企業級在地化 AI 自動化叢集

MicroK8s 基礎建設部署與維運實戰指南（安全脫敏分享版）

本指南專為企業內部私有化 AI 基礎建設架構所設計，旨在解決**大模型高硬體綁定（GPU/VRAM）**、**邊緣高效能 I/O 需求**與**輕量動態運算（Agent/自動化工作流）**在 Kubernetes 環境下共同治理的衝突點。本手冊已移除所有特定金鑰、敏感憑據與環境識別碼，修改為泛用預留位置（Placeholder），以便於團隊技術交流與公開分享。

一、 AI 叢集架構拓撲與網路神經網路對接

架構設計的核心原則為**「動靜分離、硬體直通」**。將具備動態歷史記憶、頻繁變動的資料交付 K8s 託管 Persistent Volume Claim（PVC）以納入每日自動備份傘；而針對數十 GB 且高度綁定特定 GPU 硬體的靜態權重（Weights），則特許使用本機儲存（hostPath）進行極速原地掛載，免除不必要的網路儲存區轉手損耗與備份 I/O 災難。

服務名稱 (Pod)	K8s 內部網域端點 (CoreDNS)	外部存取端點 (NodePort)	儲存特性	實體掛載路徑 / 說明
pgadmin	`http://pgadmin:80`	30080	無狀態	管理主控台，死守黃金通道 30080
llama-server	`http://llama-server:8080`	30081	hostPath	`/opt/ai-models/` 直通 NVIDIA GPU (1塊)
hermes	`http://hermes:8642`	30642 (Web) 30919 (API)	動態 PVC	`hermes-data-pvc` (10Gi) 儲存對話記憶、自建套件
firecrawl-api	`http://firecrawl-api-service:3002`	無外部暴露	無狀態	內網專屬爬蟲引擎 (Playwright)

> 💡 **K8s 內網通訊核心規律：** 叢集內各組件通訊應全面摒棄 `host.docker.internal` 或外網實體 IP。Hermes 對接大模型時，API URL 請唯一指定為 `http://llama-server:8080/v1`；對接自建爬蟲時，URL 請唯一指定為 `http://firecrawl-api-service:3002`。

二、 核心組件黃金清單 (Golden Manifests)

以下為經過生產環境除錯對齊後的完全體部署藍圖，確保解決了 Entrypoint 覆蓋與 NodePort 衝突問題。

1. Llama.cpp 推理引擎部署 (llama-stack.yaml)

```
apiVersion: v1
kind: Service
metadata:
  name: llama-server
spec:
  type: NodePort
  ports:
    - port: 8080
      targetPort: 8080
      nodePort: 30081  # 避開被 pgadmin 佔用的 30080
  selector:
    app: llama-server
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: llama-server
spec:
  replicas: 1
  selector:
    matchLabels:
      app: llama-server
  template:
    metadata:
      labels:
        app: llama-server
    spec:
      containers:
        - name: llama-server
          image: ghcr.io/ggml-org/llama.cpp:server-cuda
          # 🌟 核心避坑：不可使用 command 覆蓋 entrypoint，改用 args 傳遞參數
          args: [
            "-m", "/models/<YOUR_MODEL_FILE>.gguf",
            "--mmproj", "/models/<YOUR_MMPROJ_FILE>.gguf",
            "-np", "1",
            "-t", "6",
            "--flash-attn", "on",
            "--image-min-tokens", "1024",
            "--no-mmap",
            "--port", "8080",
            "--host", "0.0.0.0",
            "--kv-unified",
            "-c", "131072",
            "--jinja",
            "--cont-batching"
          ]
          resources:
            limits:
              nvidia.com/gpu: "1" # 🚀 MicroK8s GPU 直通特權
          volumeMounts:
            - name: model-volume
              mountPath: /models
      volumes:
        - name: model-volume
          hostPath:
```

```
            path: /opt/ai-models # 🌟 獨立模型大本營，不佔用 K8s 動態儲存區
            type: Directory
```

2. Hermes Agent 智能代理部署 (hermes-stack.yaml)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: hermes-data-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi # 給予充足空間存放動態安裝的 pip 碎檔案
---
apiVersion: v1
kind: Service
metadata:
  name: hermes-service
spec:
  type: NodePort
  ports:
    - name: port-8642
      port: 8642
      targetPort: 8642
      nodePort: 30642 # 拉回 K8s 合法 NodePort 守備範圍 (30000-32767)
    - name: port-9119
      port: 9119
      targetPort: 9119
      nodePort: 30919
  selector:
    app: hermes
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hermes
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hermes
  template:
    metadata:
      labels:
        app: hermes
    spec:
      containers:
        - name: hermes
          image: nousresearch/hermes-agent:latest
          args: ["gateway", "run"] # 修正轉譯 Docker 時代的 command 屬性
          env:
            - name: HERMES_DASHBOARD
              value: "1"
            - name: HERMES_DASHBOARD_BASIC_AUTH_USERNAME
              value: "<YOUR_USERNAME>"
```

```
            - name: HERMES_DASHBOARD_BASIC_AUTH_PASSWORD
              value: "<YOUR_PASSWORD>"
            - name: TELEGRAM_BOT_TOKEN
              value: "<YOUR_TELEGRAM_BOT_TOKEN>"
            - name: TELEGRAM_ALLOWED_USERS
              value: "<YOUR_ALLOWED_USER_IDS>"
            - name: TELEGRAM_HOME_CHANNEL
              value: "<YOUR_CHANNEL_ID>"
            - name: HERMES_AUTO_APPROVE
              value: "1"
            - name: AUTO_APPROVE_SKILLS
              value: "all"
            - name: PIP_BREAK_SYSTEM_PACKAGES
              value: "1"
            - name: EXECUTE_CODE_WITHOUT_APPROVAL
              value: "true"
            - name: HERMES_DISABLE_LAZY_INSTALLS
              value: "false"
            - name: HERMES_ALLOW_PRIVATE_URLS
              value: "true"
            # 🌟 補齊原本 Docker .env 隱藏啟用、但 Compose 漏掉的 API 變數
            - name: API_SERVER_ENABLED
              value: "true"
            - name: API_SERVER_KEY
              value: "<YOUR_SK_HERMES_API_KEY>"
          resources:
            limits:
              memory: 8Gi
              cpu: "4.0"
          volumeMounts:
            - name: hermes-data
              mountPath: /opt/data
      volumes:
        - name: hermes-data
          persistentVolumeClaim:
            claimName: hermes-data-pvc
```

三、 MIS 必看：Docker 轉 K8s 四大經典地雷與避坑指南

在將地面的 Docker 基礎建設搬遷至 Kubernetes 叢集時，由於兩者的核心網路抽象與生命週期定義不同，極易產生非預期猝死。以下歸納本次搬遷總結的四大軍規級避坑要點：

1. Command 與 Args 的「名詞轉譯陷阱」

這是最容易導致 RunContainerError 的元凶。Docker Compose 的 command 代表的是容器啟動參數，會自動追加在映像檔原廠的 ENTRYPOINT 後面。但在 K8s 中，YAML 的 command 會**直接強行覆蓋**掉原廠的 Entrypoint。如果在 K8s YAML 中錯把參數寫在 command 裡，會導致容器開機時在根目錄找不到該可執行檔（噴出 stat: no such file or directory）而猝死。**正確隊形：原廠有自帶開機指令的，K8s YAML 一律拔掉 command，只用 args 傳遞參數。**

2. NodePort 的「法定守備範圍」

Kubernetes API Server 對於邊緣網路節點對齊有極其死板的防線。NodePort 的合法分配範圍雷打不動就是 **30000 - 32767**。在編寫設計圖時，絕對不可將 Docker 時代習慣的 38642 或 8080 直接寫入 nodePort 欄位，否則會遭到 API Server 當場翻臉退貨（噴出 `provided port is not in the valid range`）。

3. Label Selector 的「強迫症語法格式」

使用 kubectl 進行標籤篩選巡邏時（例如下達 `-l` 參數），K8s 標籤篩選器**僅認同等號 = 或 ==**，絕對不允許使用冒號 `:`。若打成 `app:llama-server`，K8s 會將其誤解析為一個包含冒號的非法標籤 Key，直接退貨回報 BadRequest。

4. 應用程式 Config 的「Docker 血統殘留」

搬遷後必須全面清查應用程式內部的 `config.yaml`。諸如 `loopback_host_alias: host.docker.internal` 等 Docker 專屬網閘別名，在 K8s Pod 內完全無法被 DNS 解析。應全面與 K8s 內網域名對齊（改為 `localhost` 或特定 Service 名稱）。同時，若自建 Firecrawl 後端未配置 Tavily 等第三方搜尋 Key，應將 Hermes 的 `search_backend` 手動切換為 `browser`，特許其直接調用內建的 Playwright 進行網頁滾動搜尋，以根治搜尋不支援的報錯。

四、 歷史記憶平移（無痛大挪移腳本）

針對 Hermes 累積高達 902MB 的海量 Python 碎檔案與對話記憶，標準維運移轉流程如下（嚴禁在兩端容器同時存活時拷貝，避免檔案鎖死毀損）：

```
# 1. 強推新版設計圖上架
microk8s kubectl apply -f ~/k8s-manifests/hermes/hermes-stack.yaml

# 2. 叫醒 K8s 叢集 1 秒鐘，逼 K8s 自動在底層把長檔名實體資料夾長出來
microk8s kubectl scale deployment hermes --replicas=1

# 3. 巡邏儲存腹地，抓取熱騰騰剛出爐的 K8s 實體長檔名
ls -l /var/snap/microk8s/common/default-storage/ | grep hermes
# 🌟 畫面會彈出：default-hermes-data-pvc-pvc-<PVC_GENERATED_UUID>，請複製它

# 4. 【關鍵】令 K8s 容器安全躺平，放開底層檔案鎖；同時物理超度舊 Docker 容器
microk8s kubectl scale deployment hermes --replicas=0
cd /srv/docker/hermes && sudo docker compose down

# 5. 真槍實彈大挪移：將舊 Docker 的隱藏記憶目錄完整灌進 K8s 新家
# (請將下方路徑替換為您實體抓到的長檔名)
sudo cp -a /srv/docker/hermes/.hermes/. /var/snap/microk8s/common/default-storage/default-hermes-data-pvc-pvc-<PVC_GENERATED_UUID>/

# 6. 物理校正擁有權：全面對齊為 root 最高安全權限
sudo chown -R root:root /var/snap/microk8s/common/default-storage/default-hermes-data-pvc-pvc-<PVC_GENERATED_UUID>
```

```
# 7. 滿血復活點火，大功告成
microk8s kubectl scale deployment hermes --replicas=1
```

五、 常規運維與實戰抓包（Troubleshooting）工具箱

身為 IT 主管，當叢集狀態出現異常時，應依循標準「三聯發」外科手術式指令進行快速定位：

1. 查看 Pod 狀態與動態追蹤

不要盲等，直接動態追蹤特定標籤的 Pod 生滅狀態（注意：使用等號 =）：

```
microk8s kubectl get pod -l app=hermes -w
```

2. 拍超音波照 (Describe Pod)

當 Pod 卡在 `ContainerCreating` 或噴出 `RunContainerError` 時，直接拉到最底下的 **Events** 區塊看死因報告：

```
microk8s kubectl describe pod -l app=hermes
```

3. 直擊大腦即時日誌 (Logs)

當 Pod 順利進入 Running 綠燈，但功能不通連時，直接開火盯梢內部程式核心輸出：

```
# 查看大模型載入與 VRAM 吞噬進度
microk8s kubectl logs -l app=llama-server --tail=50 -f

# 查看 Telegram Bot 網路對接與 API 調用狀況
microk8s kubectl logs -l app=hermes --tail=50 -f

# 驗證內網連線：當 Telegram 對機器人說「幫我用 Firecrawl 爬取 https://example.com」時，盯緊此處
microk8s kubectl logs -l app=firecrawl-api --tail=20 -f
```

> ⚠️ **運維警告（ContainerCreating 假死機）：** 由於記憶平移導入了高達 902MB 的海量 PIP 與環境碎檔案，K8s 首次掛載此 PVC 時，底層 Kubelet 會花費較長時間掃描檔案系統。此時 `logs` 指令會被 BadRequest 退貨屬正常現象。請耐心調閱 `describe` 的 Events，只要確認為 `MountVolume.SetUp succeed`，靜候片刻即可順暢通車。