

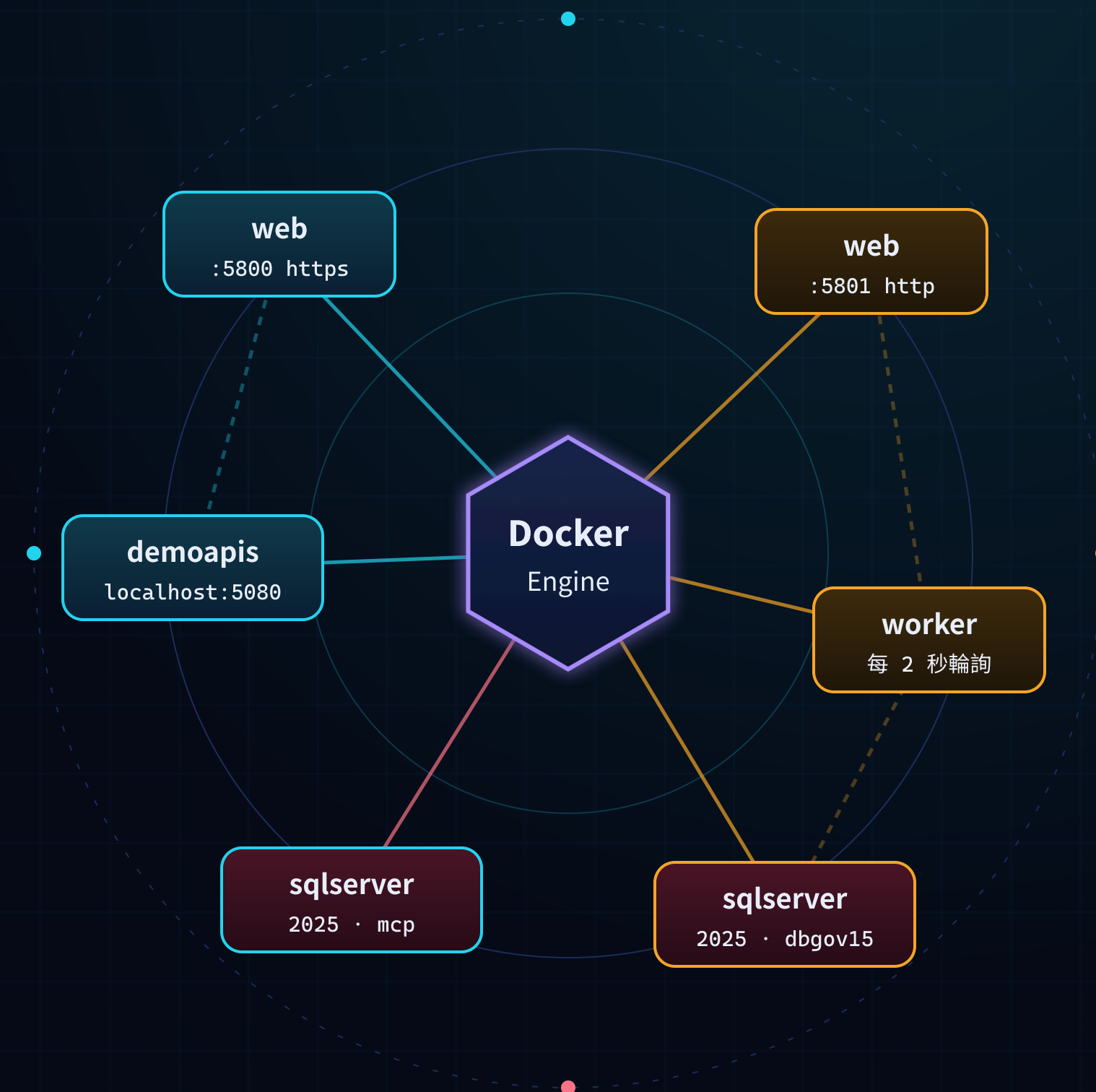
Docker 微服務 治理平台架構

兩套 .NET 10 治理平台 × 兩個 SQL Server 2025 容器
從拓樸、網路、啟動、資料到執行期流程的完整圖解

<https://127.0.0.1:5800> MCP Server API 平台 compose: mcpplatform

<http://127.0.0.1:5801> Jev SQL Server 結構治理平台 compose: dbgov15

SQL Server 2025 × 2 17.0.5005.3 Developer :14333 / :14334



簡報大綱

六個段落，由外而內：先看整體，再看每個堆疊，最後落到執行期流程與維運。

01

全景與現況

主機、兩個 Compose 專案與八個容器的全景圖；實機運行狀態與資源用量。

p.03 – p.04

02

兩個堆疊的拓樸

5800 mcpplatform 與 5801 dbgov15 的容器、網段、Volume 與掛載。

p.05 – p.08

03

網路與啟動

共用網路命名空間的兩種鏡像設計、相依條件與 db-init 初始化腳本順序。

p.09 – p.12

04

建置、資料與祕密

多階段 Dockerfile、資料持久化、SQL Server 2025 的最小權限登入、祕密流向。

p.13 – p.16

05

執行期流程

OAuth → MCP → 上游 API；提案 → 演練 → 核准 → Worker 部署；兩站環境對照。

p.17 – p.19

06

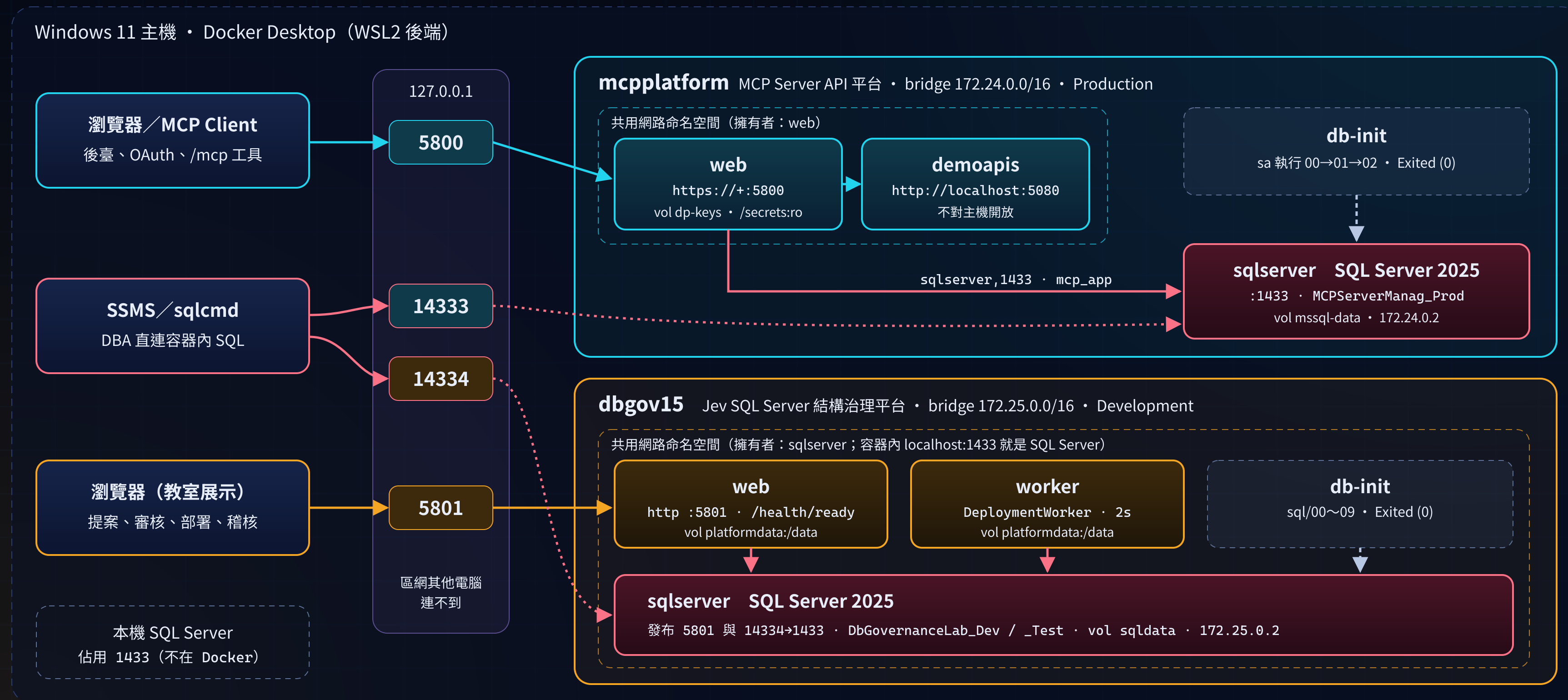
實證與維運

實機畫面、安全分層、生命週期指令、已知限制與改善建議、總結。

p.20 – p.25

全景總覽：一台主機、兩個獨立堆疊

兩套 Compose 專案各自擁有 SQL Server 2025 容器與 bridge 網路，彼此不共用任何資源；對外埠一律綁在 127.0.0.1。



實機運行狀態（2026-09-27）

六個常駐容器全部運作中，兩個 db-init 以 Exited (0) 表示初始化成功；兩個 SQL Server 與 dbgov15 web 通過健康檢查。

2

Compose 專案

8

容器（6 常駐+2 一次性）

2

SQL Server 2025 執行個體

4

具名 Volume

3

自建映像

4

對外埠（皆 127.0.0.1）

docker ps (依 compose 專案篩選)

```
$ docker ps -a --filter label=com.docker.compose.project=mcplatform
NAMES                STATUS                PORTS
mcplatform-demoapis-1 Up 6 hours
mcplatform-web-1      Up 6 hours            127.0.0.1:5800->5800/tcp
mcplatform-db-init-1  Exited (0)
mcplatform-sqlserver-1 Up 6 hours (healthy)  127.0.0.1:14333->1433/tcp

$ docker ps -a --filter label=com.docker.compose.project=dbgov15
NAMES                STATUS                PORTS
dbgov15-worker-1     Up 42 minutes
dbgov15-web-1        Up 42 minutes (healthy)
dbgov15-db-init-1    Exited (0)
dbgov15-sqlserver-1  Up 46 minutes (healthy) 127.0.0.1:5801->5801/tcp,
127.0.0.1:14334->1433/tcp
```

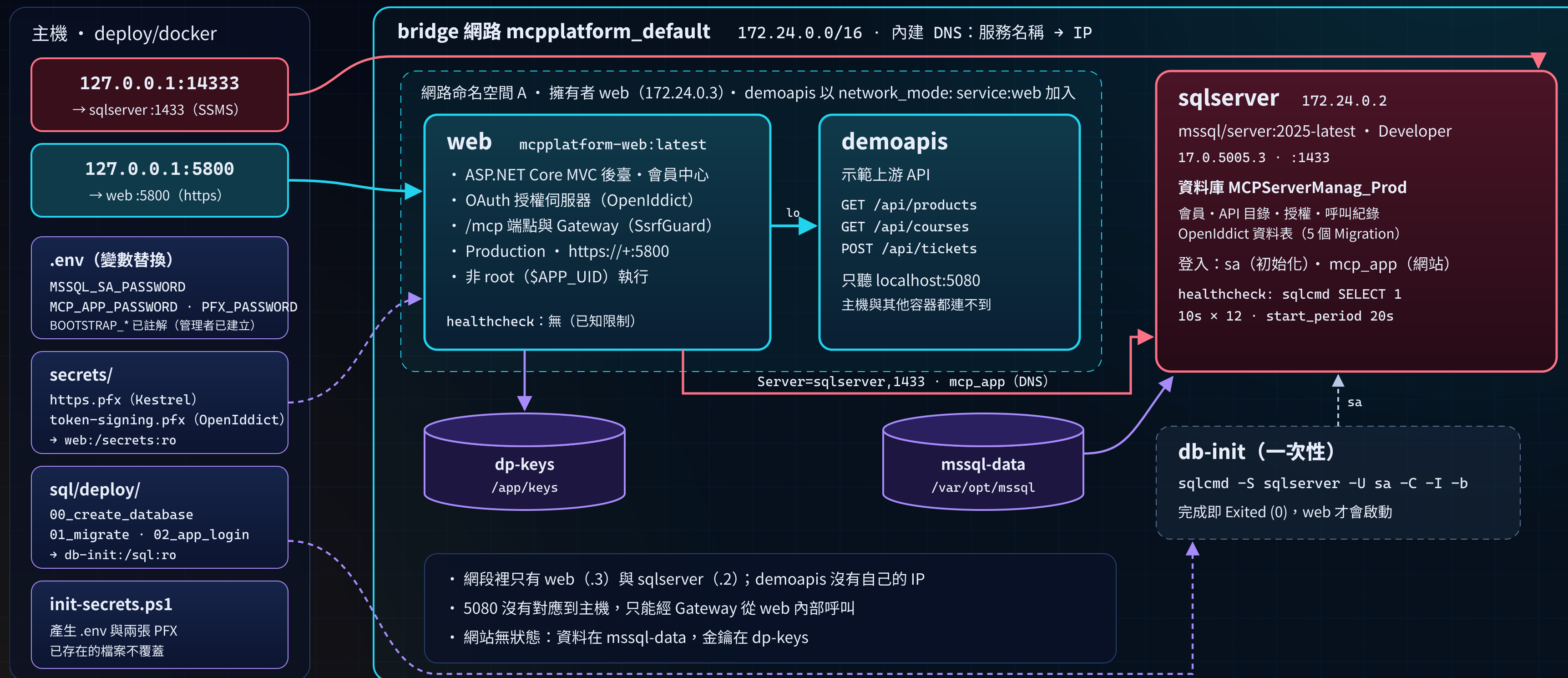
記憶體用量 (docker stats)

mcp sqlserver	1.15G
dbgov sqlserver	1018M
dbgov web	212M
mcp web	179M
dbgov worker	77M
mcp demoapis	29M

SQL Server 佔總記憶體約 **81%**（2.2 GB／2.65 GB）；兩個 .NET 網站合計不到 400 MB。

127.0.0.1:5800 MCP Server API 平台拓樸

四個服務、兩個具名 Volume、一個 bridge 網路；demoapis 掛在 web 的網路命名空間，因此網段裡只看得到兩個容器。

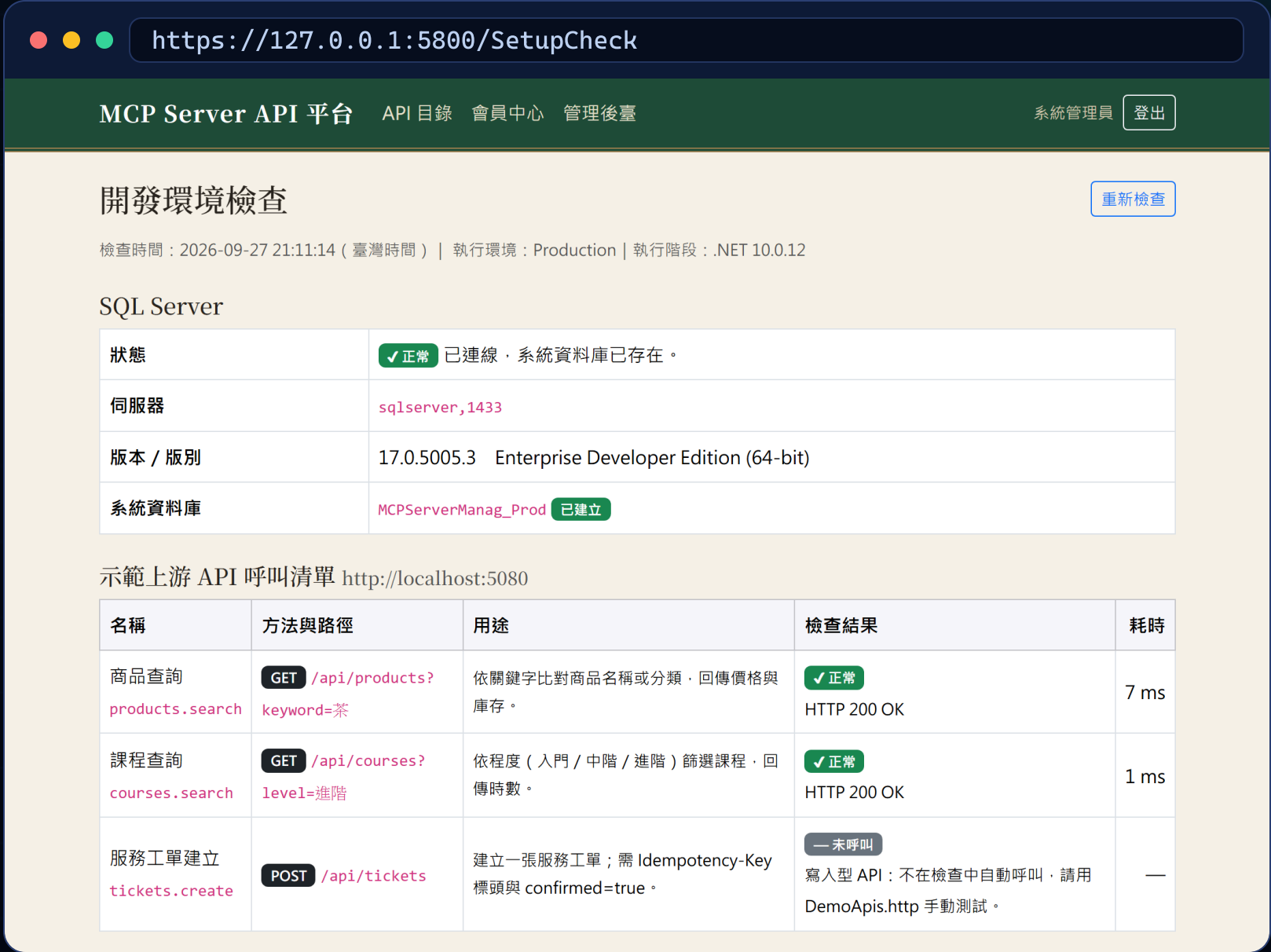


mcpplatform 元件職責與相依

一次建置產生兩個執行映像；平台自身資料全部放在 SQL Server，網站只拿到最小權限帳號 mcp_app。

服務	映像／角色	相依條件
sqlserver	mssql/server:2025-latest · Developer 平台唯一資料庫；主機 14333 供 SSMS 管理	— (healthcheck：sqlcmd SELECT 1)
db-init	同上映像，只跑 sqlcmd 建庫 → 冪等 Migration → 建立 mcp_app	sqlserver： service_healthy
web	mcpplatform-web (393 MB) MVC 後臺、OpenIddict、/mcp、Gateway	db-init: service_ completed_successfully
demoapis	mcpplatform-demoapis (340 MB) 示範上游 API，network_mode: service:web	web： service_started

右圖是平台內建的「開發環境檢查」頁：以容器內位址 `sqlserver,1433` 連線、執行環境 **Production**、.NET 10.0.12，且經由 `http://localhost:5080` 呼叫 demoapis 回 200 —— 直接證明共用命名空間的設計生效。



127.0.0.1:5801 Jev SQL Server 結構治理平台拓模

web、worker、db-init 全部加入 sqlserver 的網路命名空間，所以 5801 是發布在 sqlserver 容器上；bridge 網段裡只有 sqlserver 一個成員。



dbgov15 元件職責：平台資料與受管資料分離

平台自己的帳號、提案、審核、部署與稽核存在 SQLite；SQL Server 2025 是被探索、演練與部署的「受管目標」。

服務	映像／角色	相依條件
sqlserver	mssql/server:2025-latest 受管實驗庫 Dev／Test；網路命名空間擁有者	— (healthcheck 10s×20)
db-init	同上映像，執行 init-db.sh 依 README「從零安裝」順序跑 sql/	sqlserver: service_healthy
web	dbgov15-platform (533 MB) Migration、Bootstrap、MVC／API	db-init: service_ completed_successfully
worker	同一映像，覆寫 entrypoint 背景部署、驗證、漂移	web: service_healthy

worker 等 web 健康才啟動：Migration 是 web 的工作，worker 不能在資料表建立前就開始輪詢 DeploymentJobs 。



平台資料 · SQLite

governance.db（16 張表、11 個 Migration） + Data Protection 金鑰

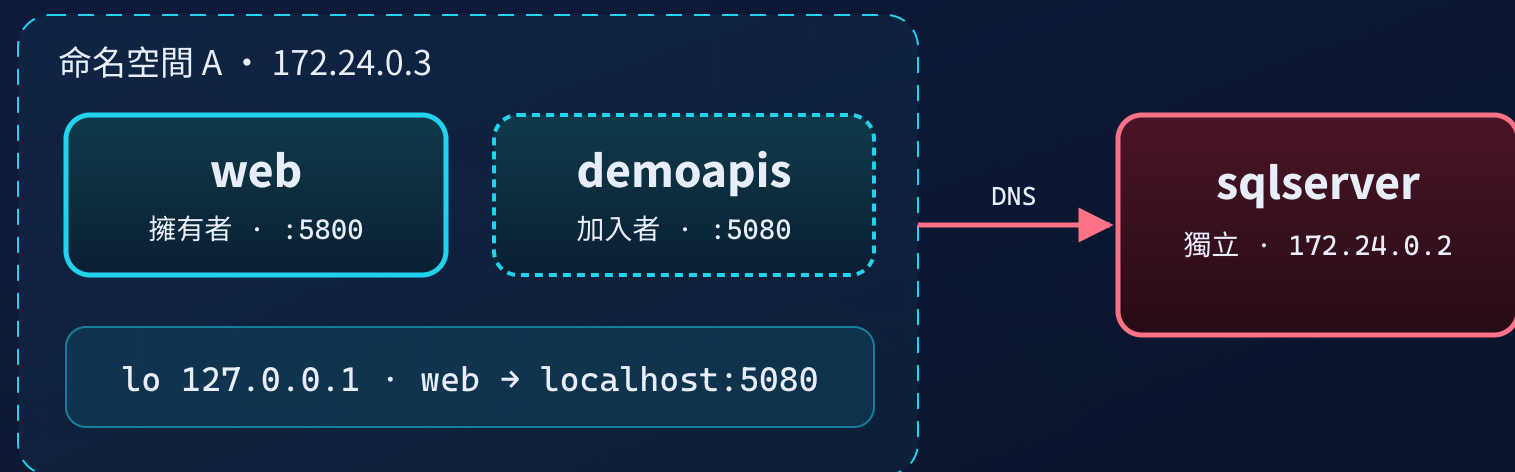
受管資料 · SQL Server

DbGovernanceLab_Dev／_Test；平台不把自己的資料寫進去

共用網路命名空間：方向相反的兩種設計

`network_mode: "service:X"` 讓容器直接使用 X 的網路介面與回送位址 —— 兩個堆疊用同一招，但「誰當擁有者」剛好相反。

5800 · 擁有者 = web



連資料庫：`Server=sqlserver,1433` (bridge DNS)

埠宣告在 **web**；網段成員：web、sqlserver

- 目的：上游呼叫必須是明寫的 **localhost** 才能通過 SsrGuard
- 副作用：只重啟 web，demoapis 會失去網路

5801 · 擁有者 = sqlserver



連資料庫：`localhost,1433` (同一回送介面)

埠宣告在 **sqlserver**；網段成員：只有 sqlserver

- 目的：種子目標 `localhost,1433` 與登錄政策不必改
- 副作用：只重啟 sqlserver，web／worker 會失去網路

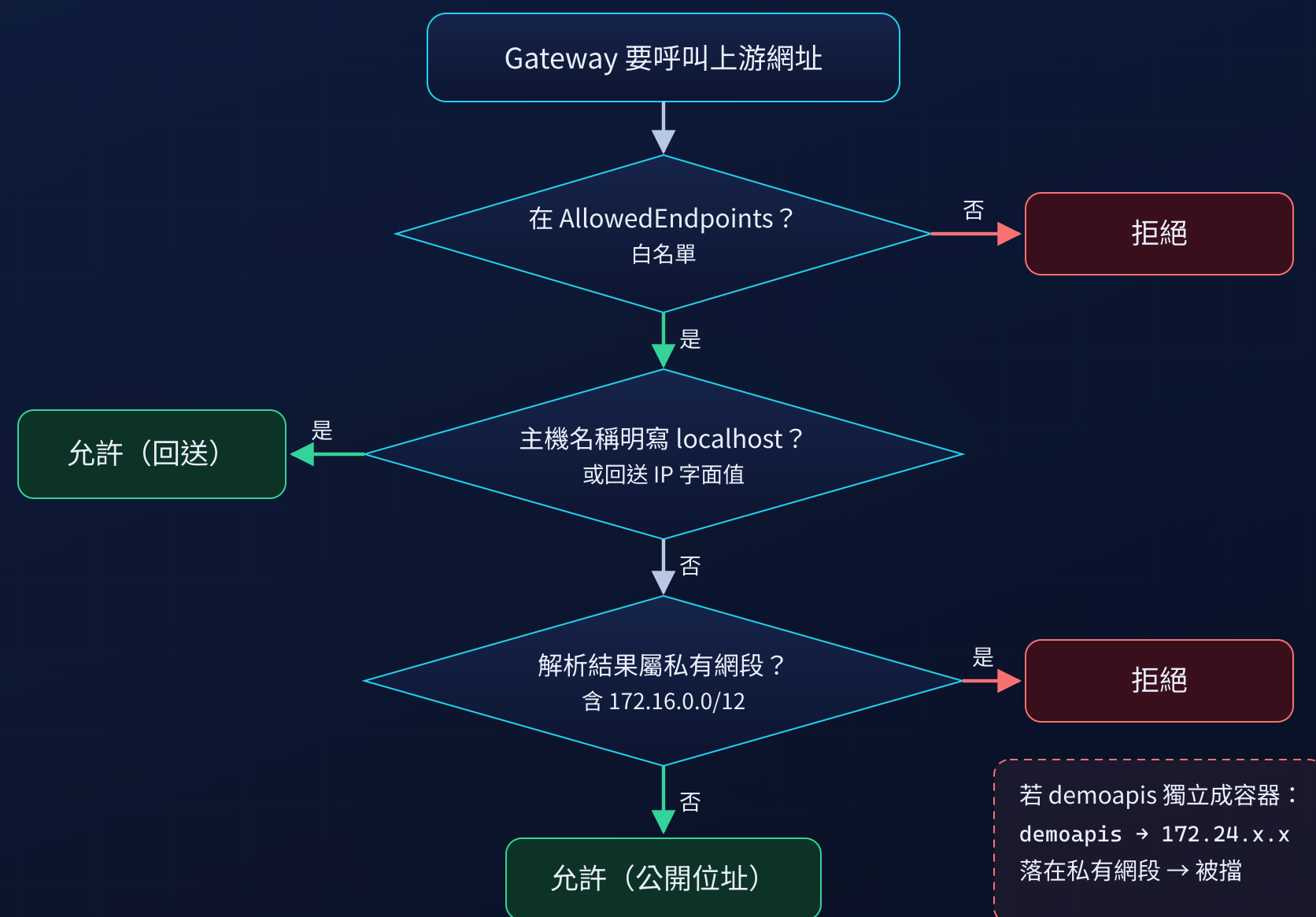
● ● ● docker network inspect (2026-09-27 實測)

```
$ docker network inspect mcplatform_default 172.24.0.0/16 mcplatform-web-1=172.24.0.3 mcplatform-sqlserver-1=172.24.0.2
$ docker network inspect dbgov15_default 172.25.0.0/16 dbgov15-sqlserver-1=172.25.0.2
# 共用命名空間的容器 (demoapis；web、worker、db-init) 沒有自己的 IP，因此不會出現在清單裡
```

為什麼要共用命名空間：不改程式碼，也不放寬安全規則

兩個堆疊都是為了保留應用程式原本對「localhost」的假設，讓容器化只動部署設定。

5800 · SsrfGuard 的判斷路徑



5801 · 目標連線寫死 localhost,1433

平台種子資料的受管目標伺服器是 `localhost,1433`，目標登錄政策也只允許 `localhost` 與 `*.lab`。加入 sqlserver 的命名空間後，容器裡的 localhost 就是 SQL Server 2025：

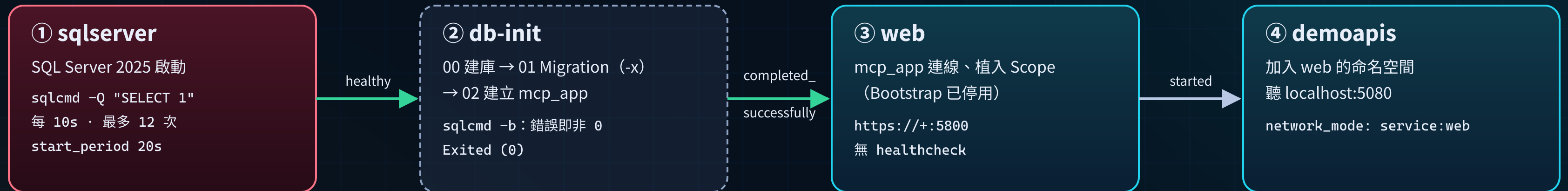


- 不必改種子資料、登錄政策或連線字串
- 與「本機直接安裝」的教材步驟完全一致
- 代價：web／worker 的生命週期綁在 sqlserver 上

啟動順序與相依條件

depends_on 的三種條件把「行程啟動」、「真的可連線」、「初始化成功」分開；任何一步失敗，後面的服務就不會起來。

mcpplatform · 5800



dbgov15 · 5801



db-init：用同一組 SQL 腳本完成可重複的初始化

兩邊都沿用「非 Docker 安裝」的同一組腳本，全部冪等；sa 密碼以環境變數 `SQLCMDPASSWORD` 傳入，不出現在命令列。

mcpplatform：3 支腳本

00_create_database.sql
-v dbName=MCPServerManag_Prod

01_migrate_idempotent.sql
EF 產生的冪等腳本 (31.6 KB)
-x 關閉 \$(...) 變數替換

02_create_app_login.sql
建立 mcp_app
db_datareader + db_datawriter

內嵌在 compose 的 entrypoint (bash -ec)。

dbgov15：init-db.sh 依相依順序執行 13 次

00 建立實驗資料庫
master · Dev + Test

01 示範領域結構
Dev、Test 各一次

04 · 05 · 06 補充物件
Dev、Test 各一次

順序關鍵：04 的跨庫 View 參照 Test.dbo.Customers，
所以兩個庫的 01 必須都先跑完

02 dbgov_reader
唯讀探索 (Dev+Test)

07 dbgov_rehearsal
Test 演練 (ddladmin)

09 dbgov_deployer
Dev 部署 (ddladmin)

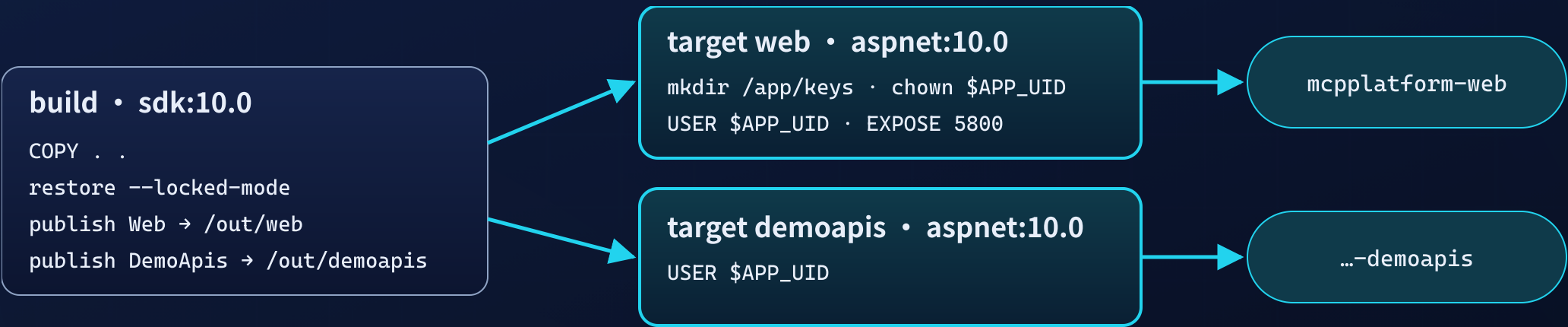
08 備份測試庫 (COPY_ONLY)
演練頁的 Backup 步驟要查得到 msdb.dbo.backupset 的備份證據

登入已存在時改成 .env 的密碼並解除鎖定 —— 重跑 db-init 等於輪替密碼。

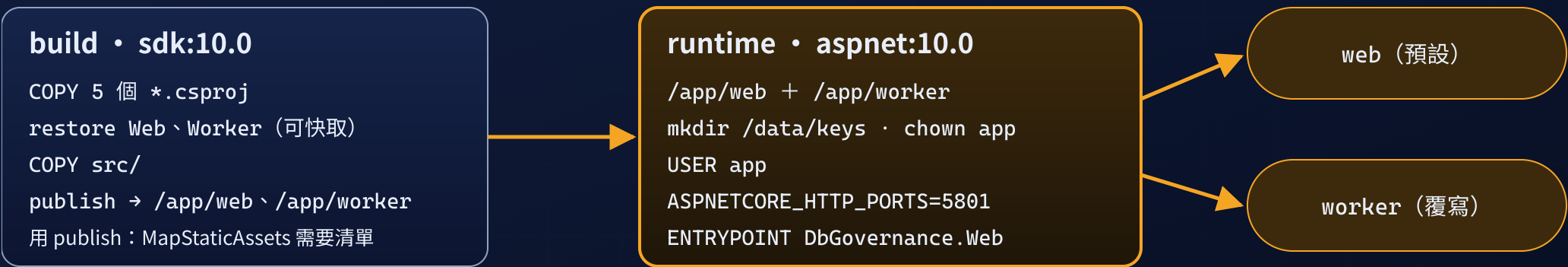
多階段 Dockerfile：SDK 只在建置時存在

兩邊都是 sdk:10.0 建置 → aspnet:10.0 執行、非 root 帳號；差別在「一次建置出兩個映像」與「一個映像放兩支程式」。

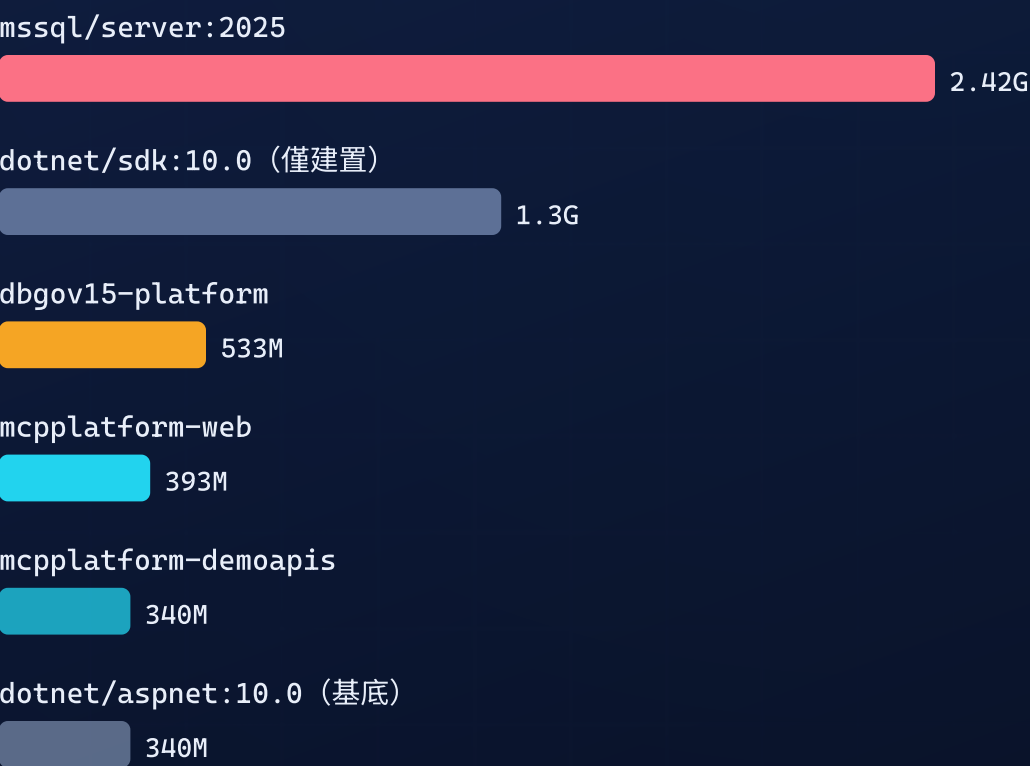
mcpplatform：1 個 build 階段 → 2 個 target



dbgov15：先還原套件（快取層）→ 一個映像含 web + worker



映像大小（docker images）



demoapis 幾乎等於 aspnet 基底；1.3 GB 的 SDK 不會進入任何執行映像。

資料持久化：4 個具名 Volume，資料庫與金鑰必須成對

容器可以隨時重建；真正要保護的是 Volume。兩邊都把 Data Protection 金鑰放進 Volume，否則重建容器就會讓所有人被登出。



兩個 SQL Server 2025 執行個體與最小權限登入

同一個映像（17.0.5005.3 Enterprise Developer Edition），不同用途：一個是平台的系統資料庫，一個是被治理的目標伺服器。sa 只給初始化使用。

mcpplatform-sqlserver

主機 127.0.0.1,14333 · 容器 sqlserver,1433

登入	誰使用	權限
sa	sqlserver、db-init	系統管理員；web 拿不到
mcp_app	web	db_datareader+db_datawriter 不能建表、改表、刪表

DB MCPServerManag_Prod
Migration 由 db-init 以 sa 套用；網站啟動時不改結構。

dbgov15-sqlserver

主機 127.0.0.1,14334 · 容器 localhost,1433

登入	使用者	DbGovernanceLab_Dev	DbGovernanceLab_Test
sa	db-init · 健康檢查	只用於執行 sql/ 腳本；平台本身不用 sa	
dbgov_reader	web · worker	CONNECT · VIEW DEFINITION 讀相依性	同左
dbgov_rehearsal	web	—	db_ddladmin · SELECT dbo msdb.backupset 唯讀
dbgov_deployer	web · worker	db_ddladmin · SELECT dbo VIEW DEFINITION	—

每個登入只拿到它那一步需要的權限：探索讀不到資料列、演練只能碰 Test、部署只能碰 Dev，三者都不是 db_owner。

映像

mcr.microsoft.com/mssql/server:2025-latest (2.42 GB)

環境變數

ACCEPT_EULA=Y · MSSQL_PID=Developer ·
TZ=Asia/Taipei

健康檢查工具

/opt/mssql-tools18/bin/sqlcmd -C（信任自簽憑證）

祕密與設定的流向

祕密只存在主機的 .env 與 secrets/，以環境變數或唯讀掛載在執行時注入；.dockerignore 排除它們，映像可以安全分享。

mcpplatform · init-secrets.ps1 產生

祕密	sqlserver	db-init	web
MSSQL_SA_PASSWORD	●	●	—
MCP_APP_PASSWORD	—	●	●
PFX_PASSWORD	—	—	●
https.pfx／token-signing.pfx	—	—	/secrets:ro
BOOTSTRAP_ADMIN_*	—	—	已註解

隨機密碼、已存在不覆蓋；token-signing.pfx 固定從檔案載入，容器重建後舊 Token 仍可驗證。

dbgov15 · docker/.env (x-app-env 錨點共用)

祕密	sqlserver	db-init	web	worker
SA_PASSWORD	●	●	—	—
READER_PASSWORD	—	●	●	●
DEPLOYER_PASSWORD	—	●	●	●
REHEARSAL_PASSWORD	—	●	●	—
ADMIN_INITIAL／USER01	—	—	●	—

admin 已存在時初始密碼不再生效；admin 首次登入必須改密碼。

●●● 環境變數 → .NET 設定 (__ 等於 :)

ConnectionStrings__DefaultConnection → ConnectionStrings:DefaultConnection
TargetSecrets__lab-deployer__Password → TargetSecrets:lab-deployer:Password
\${VAR:?訊息} 缺值時 docker compose 直接中止，不會啟動半殘的網站

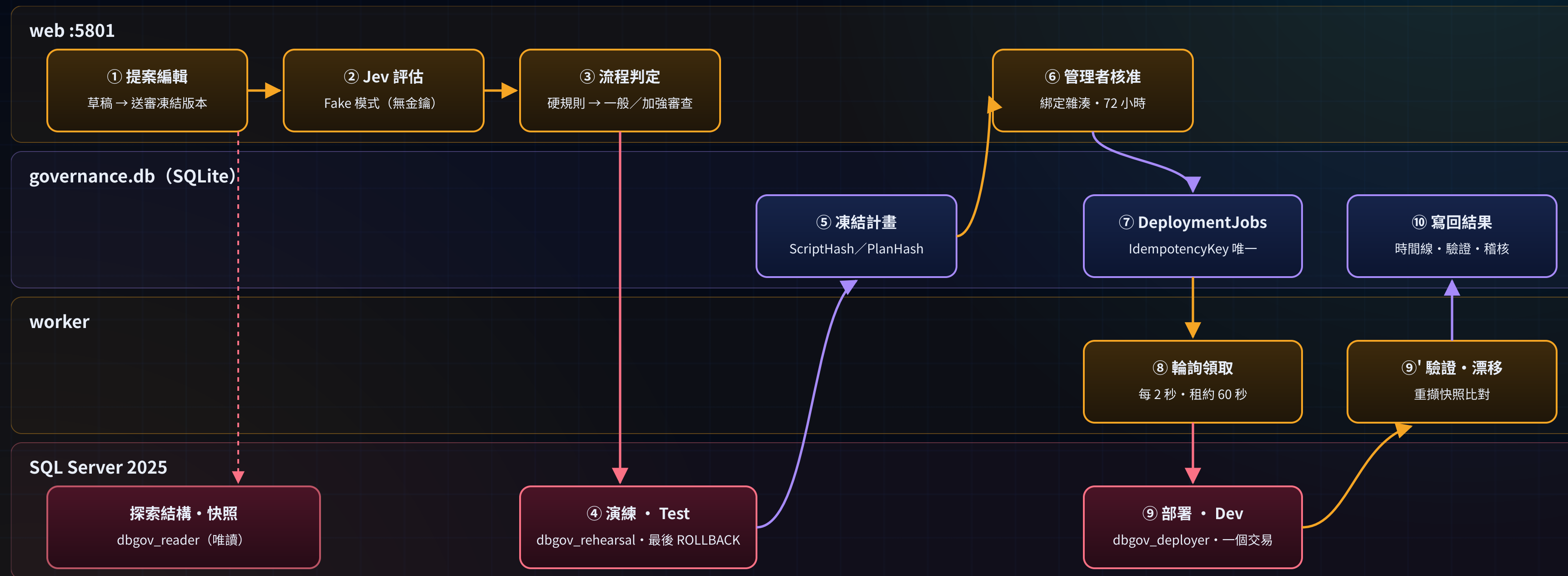
5800 執行期：OAuth → MCP 工具 → 上游 API

MCP Client 以授權碼+PKCE 取得 Token，再呼叫 /mcp；web 在同一個命名空間裡以 localhost:5080 呼叫 demoapis。



5801 執行期：提案 → 演練 → 核准 → Worker 部署

web 與 worker 透過共用的 governance.db 交接工作；對 SQL Server 的每一步都換成權限剛好夠用的登入。



- 同一個目標同時只允許一筆「執行中」或「結果不明」；worker 在 COMMIT 後、寫回前中斷時標記 NeedsReconciliation，由管理者對帳，不自動補跑。
- 執行前再檢查：核准有效、綁定雜湊一致、以唯讀帳號重讀結構比對基準（漂移即中止）。回復工作執行凍結的回復批次。

兩套部署的差異一覽

同樣是「.NET 10 網站+SQL Server 2025+一次性 db-init」，但定位不同：5800 以正式環境設定運行，5801 是教室展示設定。

面向	5800 · mcpplatform	5801 · dbgov15
對外網址／傳輸	https://127.0.0.1:5800（開發憑證 PFX）	http://127.0.0.1:5801
執行環境	Production ：OpenIddict 強制 https、無示範資料	Development ：允許 TrustServerCertificate、http Cookie、示範目標與 user01
應用程式資料	SQL Server MCPServerManag_Prod	SQLite /data/governance.db
SQL Server 的角色	平台的系統資料庫	被治理的受管目標（Dev／Test）
命名空間擁有者	web（demoapis 加入）	sqlserver（web、worker、db-init 加入）
容器網段	172.24.0.0/16 · 2 個成員	172.25.0.0/16 · 1 個成員
主機 SQL 埠	14333	14334
程式映像	2 個（web 393 MB、demoapis 340 MB）	1 個（web+worker，533 MB）
網站健康檢查	無（demoapis 只等 started）	/health/ready（worker 等 healthy）
祕密產生	init-secrets.ps1（隨機＋兩張 PFX）	手動維護 docker/.env（6 個值）
外部依賴	無（上游 API 就在同一命名空間）	Jev AI 決策：無金鑰時 Fake 模式

5800 實機畫面：後臺儀表板與資料庫概況

以系統管理員登入 <https://127.0.0.1:5800>；資料庫概況頁讀出容器內 SQL Server 的資料表與已套用的 Migration。



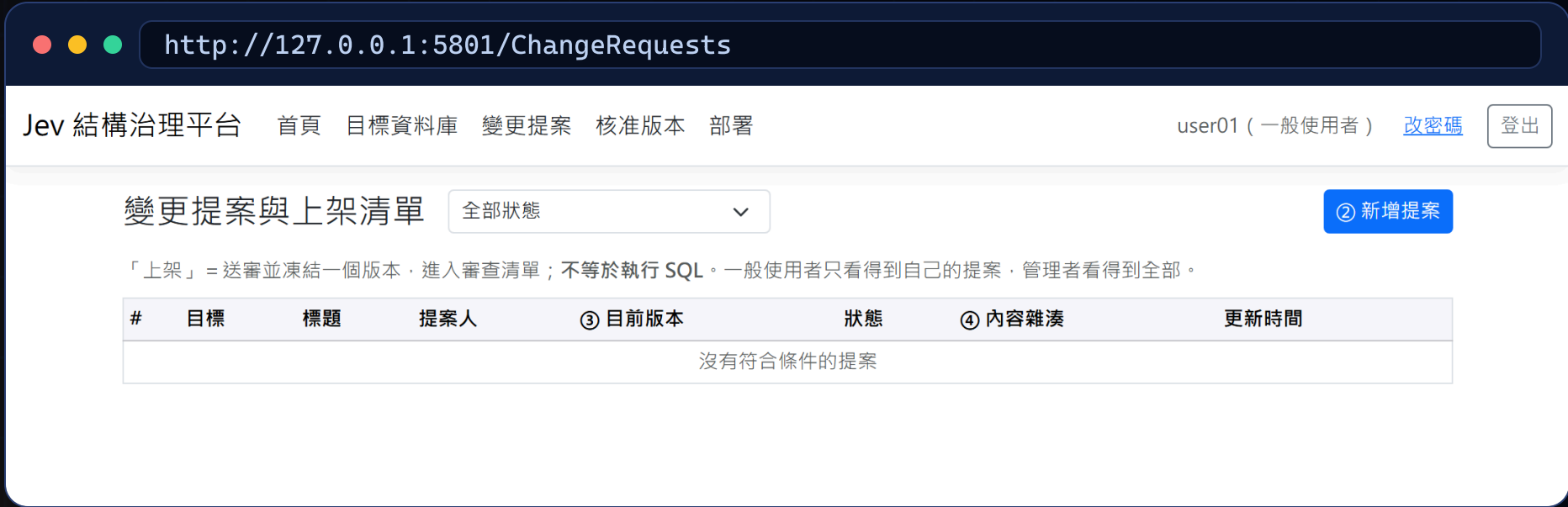
儀表板以 **臺灣時間** 計算區間（映像內建 tzdata／ICU，`TZ=Asia/Taipei`）；新部署的環境尚無呼叫紀錄。



MCPServerManag_Prod 已套用 5 個 Migration（含 AddOpenIddict）；AspNetUsers = 1，即 Bootstrap 建立的首位管理者。

5801 實機畫面與健康檢查

以示範使用者 user01 登入；健康檢查端點同時證明「平台資料」與「受管資料庫」兩條連線都正常。



部署頁說明：部署在背景由 Worker 執行，頁面每 2 秒重新整理。



/health/ready 是 compose 的 healthcheck 目標（worker 依它啟動）；**/health/lab** 經 localhost,1433 連到 SQL Server 2025。

安全設計：由外而內四層

兩個堆疊採用相同的分層思路；表格右兩欄是各自的具體做法。

① 對外暴露面 只讓本機、只經正門	5800 · mcpplatform 5800、14333 只綁 127.0.0.1 - demoapis 5080 完全不對主機開放 - 全程 https (Production)	5801 · dbgov15 5801、14334 只綁 127.0.0.1 - 平台只連 localhost 受管目標 - http 僅限教室 (Development)
② 身分與權限 每個程序只拿需要的權限	- sa 只在 sqlserver、db-init - web 用 mcp_app (讀寫，不能改結構) - 容器以非 root \$APP_UID 執行	- reader／rehearsal／deployer 分工 - 演練只碰 Test、部署只碰 Dev - 容器以非 root app 執行
③ 祕密保護 祕密不進映像、不上命令列	.dockerignore 排除 .env、secrets/ - PFX 以唯讀掛載 /secrets - SQLCMDPASSWORD 傳遞 sa 密碼	.dockerignore 排除 docker/.env \${VAR:?} 缺值即中止 - 平台以 secret:// 參照取得帳密
④ 應用層防護 程式本身的防線	- OAuth 授權碼＋PKCE、驗 iss／aud - 每位會員每分鐘 60 次限制 - AllowedEndpoints + SsrfGuard	- 禁止自我核准、核准綁定雜湊 - ScriptDom AST 檢查、硬規則攔截 - 稽核遮罩、部署冪等鍵與租約

生命週期與日常維運

Volume 決定資料的生死；共用命名空間決定「不能單獨重啟誰」。

指令	容器	映像	Volume
<code>docker compose stop / start</code>	停止／啟動	保留	保留
<code>docker compose up -d</code>	依設定差異重建	沿用	保留
<code>up -d --build</code>	重建程式容器，db-init 重跑	重建	保留
<code>up -d --force-recreate</code>	全部重建	沿用	保留
<code>docker compose down</code>	移除（含網路）	保留	保留
<code>docker compose down -v</code>	移除	保留	刪除

●●● 常用指令

```
# 5800：在 McpPlatform/deploy/docker
pwsh ./init-secrets.ps1 ; docker compose up -d --build
# 5801：在 範例/15_... (Git Bash 需 MSYS_NO_PATHCONV=1)
docker compose -f docker/docker-compose.yml up -d --build
curl http://127.0.0.1:5801/health/ready
```

5800 注意事項

- 單獨 `restart web` 會讓 demoapis 斷網 → 改用 `up -d`
- 首位管理者建立後把 .env 的 BOOTSTRAP 兩行註解掉
- 換網域或埠時，三個 `OAuth__*` 要一起改

5801 注意事項

- 不要只重啟 sqlserver → 整組 `down` 再 `up -d`
- 改 .env 密碼後 `up -d --force-recreate db-init web worker` 輪替
- 示範部署後以 `sql/10` 重設開發庫

已知限制與改善建議

目前的設定適合本機展示與教學；走向正式環境前，下列項目需要調整。

範圍	現況	建議
5800	web 沒有 healthcheck；demoapis 只等 started	加入 https 健康檢查，demoapis 改用 service_healthy
5800	HTTPS 使用本機開發憑證（僅本機信任）	換成網域正式憑證，同步修改 OAuth__Issuer 與埠對應
5801	以 Development 執行：http、TrustServerCertificate、Fake Jev	依 deployment-runbook 改 Production：受信任憑證、HTTPS、Jev 金鑰
5801	web／worker 生命週期綁在 sqlserver 命名空間	正式環境把目標改成可解析的主機名稱，讓服務各自獨立網路
兩者	Data Protection 金鑰在 Linux 為明文 XML	ProtectKeysWithCertificate，或放入金鑰管理服務
兩者	映像標籤 latest（含 mssql/server:2025-latest）	改用版本號或 digest 固定，便於回復
兩者	SQL Server 為 Developer 版	正式環境改授權版本（MSSQL_PID）或外部 SQL Server
兩者	日誌只在容器 stdout，容器移除即消失	設定日誌驅動或集中式日誌收集

總結：同一套容器化手法，兩種取捨

01

完全隔離的兩個堆疊

各自的 SQL Server 2025、bridge 網路與 Volume；埠只綁 127.0.0.1，並避開主機 1433。

02

命名空間共用，方向相反

5800 讓上游 API 加入 web（滿足 SSRF 規則）；5801 讓應用加入 sqlserver（保留 localhost,1433）。

03

資料角色不同

5800 的 SQL Server 是平台系統庫；5801 的平台資料在 SQLite，SQL Server 是被治理的目標。

04

失敗即停、最小權限

healthy → completed → started / healthy 的相依鏈；sa 只給初始化，網站與 worker 各拿剛好夠用的登入。

<https://127.0.0.1:5800>

<http://127.0.0.1:5801>

SQL Server 2025 · 14333 / 14334