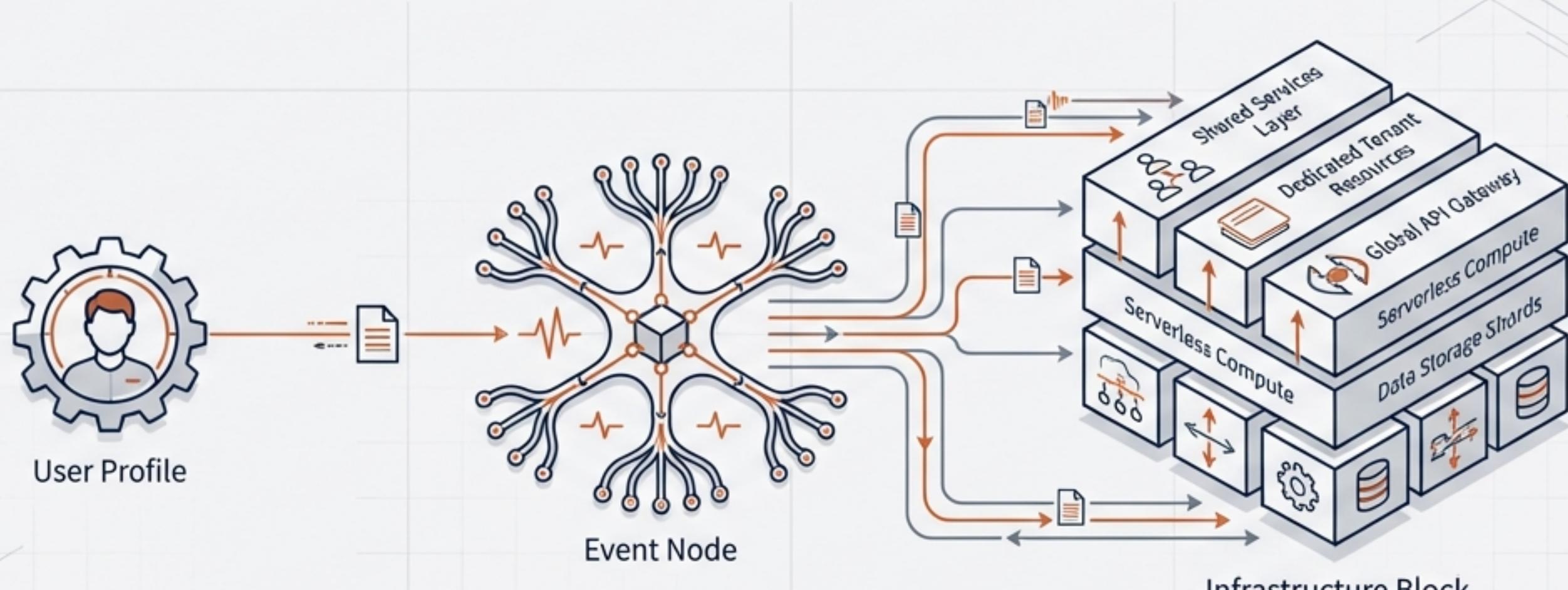


現代化 SaaS 架構藍圖：從身份驗證到多租戶隔離策略

結合 AWS Cognito、EventBridge 與 Keypiz 分層隔離架構的深度實踐



Identity (身份)

安全且可擴展的入口



Events (事件)

解耦的系統神經中樞



Isolation (隔離)

從共享到專屬的彈性部署

現代應用的核心挑戰：耦合、擴展與安全

- **使用者管理 (User Management)**

如何安全地處理數百萬用戶的註冊、登入和身份驗證？

- **服務通訊 (Service Communication)**

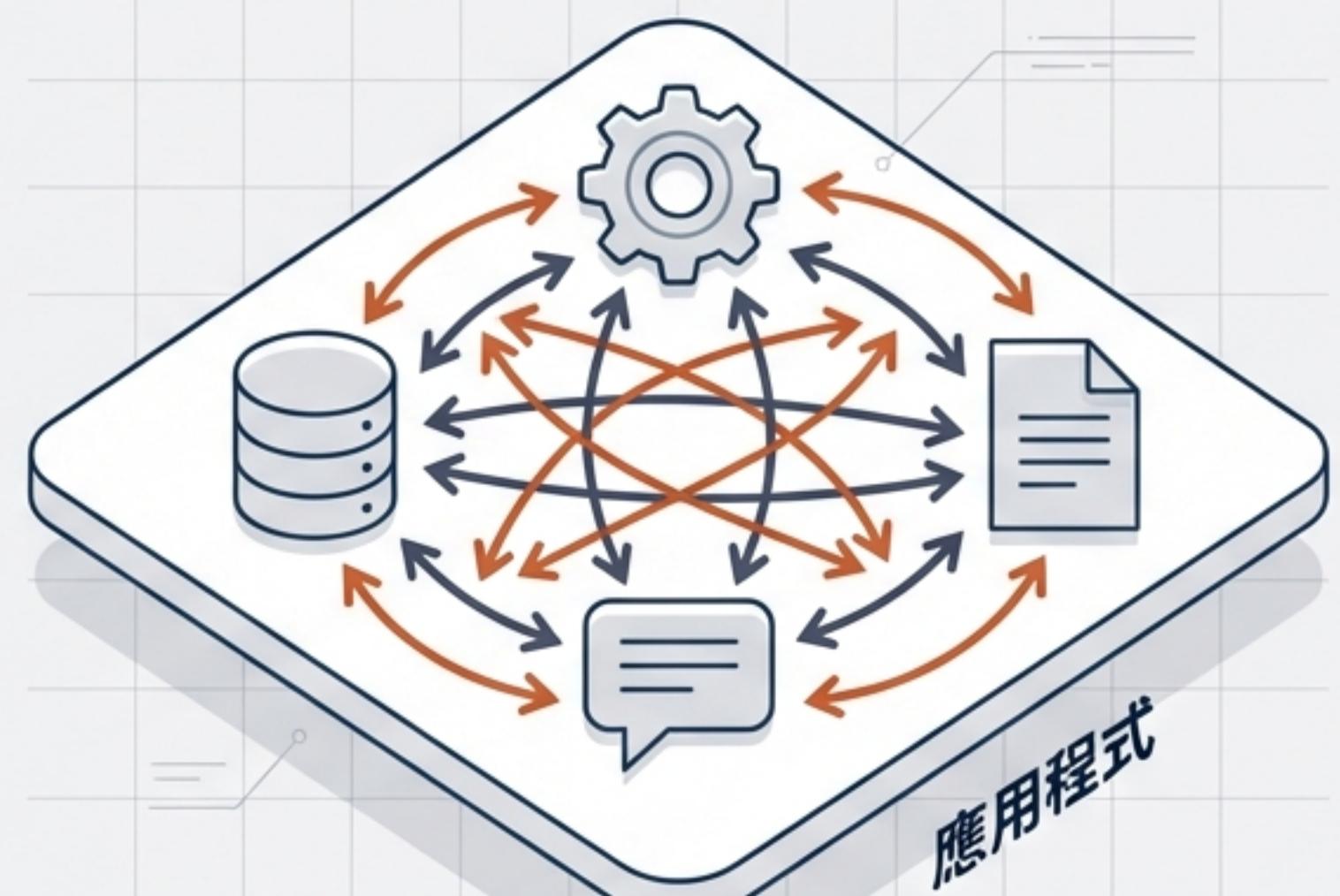
微服務之間如何高效、可靠地溝通，同時避免形成緊密耦合的「分散式單體」？

- **可擴展性 (Scalability)**

當業務需求增長時（例如從 B2C 擴展到 B2B），架構如何平滑地擴展以支持多租戶？

- **響應能力 (Responsiveness)**

系統如何對用戶行為或內部事件做出即時反應，觸發下游工作流程？



解決方案之基石：以 AWS Cognito 建立託管式身份層



核心優勢

- **核心功能：**使用者註冊、登入、忘記密碼、Token 管理 (JWT)、聯合身份驗證 (Federation)。
- **架構角色：**作為應用程式的身份驗證和授權中心，保護後端 API 和資源。

成本優化 (Cost Optimization)

對於大規模 MAU，建議在 API Gateway 層進行本地 JWT 驗證 (Local Validation)，避免頻繁調用 ` GetUser` API 以節省成本。

Cognito 多租戶身份驗證的三種常見模型

每租戶一個 User Pool

作法：

- 為每家公司建立獨立 User Pool

優點：

- 隔離度最高、各租戶可有不同的密碼/MFA 政策

缺點：

- 管理成本高、租戶多時難維運

共用 User Pool + App Client 分租戶

作法：

- 同一個 User Pool，為每個租戶建一個 App Client，並綁定各自 IdP

優點：

- 共用使用者資料，仍可對租戶客製 IdP 與 Redirect URL

缺點：

- App Client 變多、設定較複雜

共用 User Pool + 自訂屬性分租戶

作法：

- 單一 User Pool，使用 custom:tenantId 等屬性標記租戶

優點：

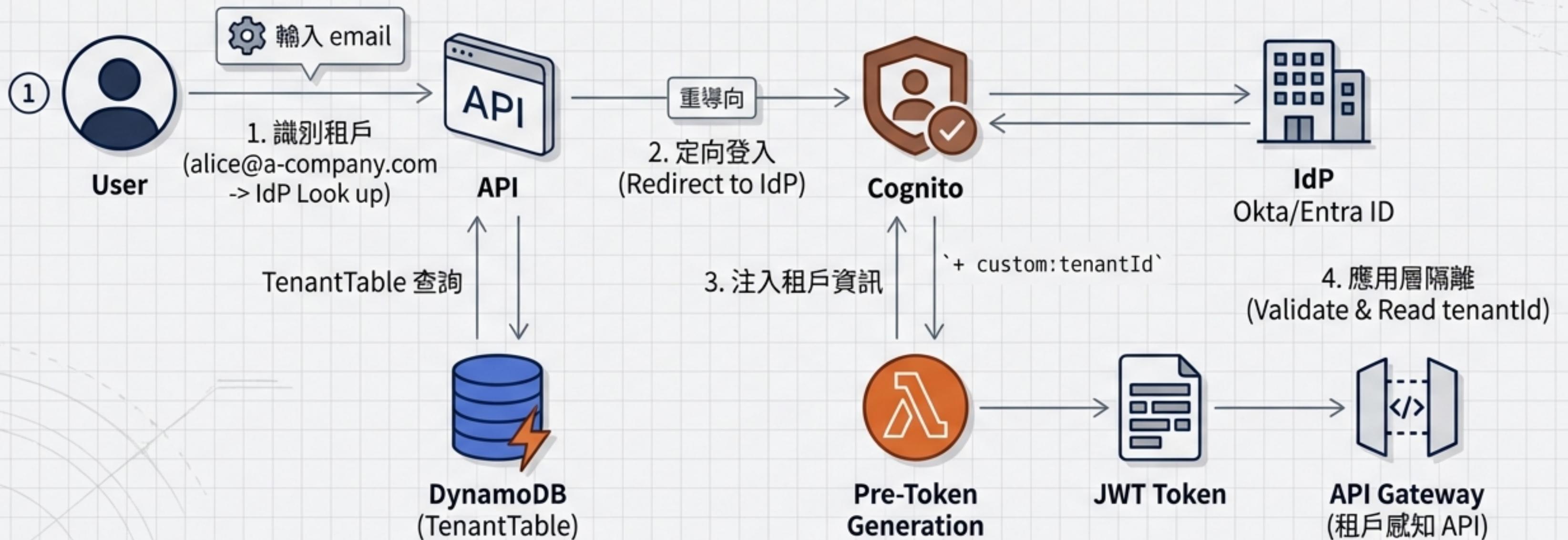
- Sign-up / Sign-in URL 完全共用，體驗一致，擴充租戶最簡單

缺點：

- 授權與資料隔離需在應用和 API 層嚴格實作

深度解析推薦模型：單一 User Pool + 自訂屬性

此模型在擴展性和使用者體驗之間取得了最佳平衡



客製化使用者旅程：透過 Lambda Triggers 注入業務邏輯

presignup/preSignUp.mjs

```
export const handler = async (event) => {
  // Auto confirm users
  event.response.autoConfirmUser = true; ←

  // Auto verify email
  if (event.request.userAttributes.hasOwnProperty("email")) {
    event.response.autoVerifyEmail = true;
  }
  return event;
};
```

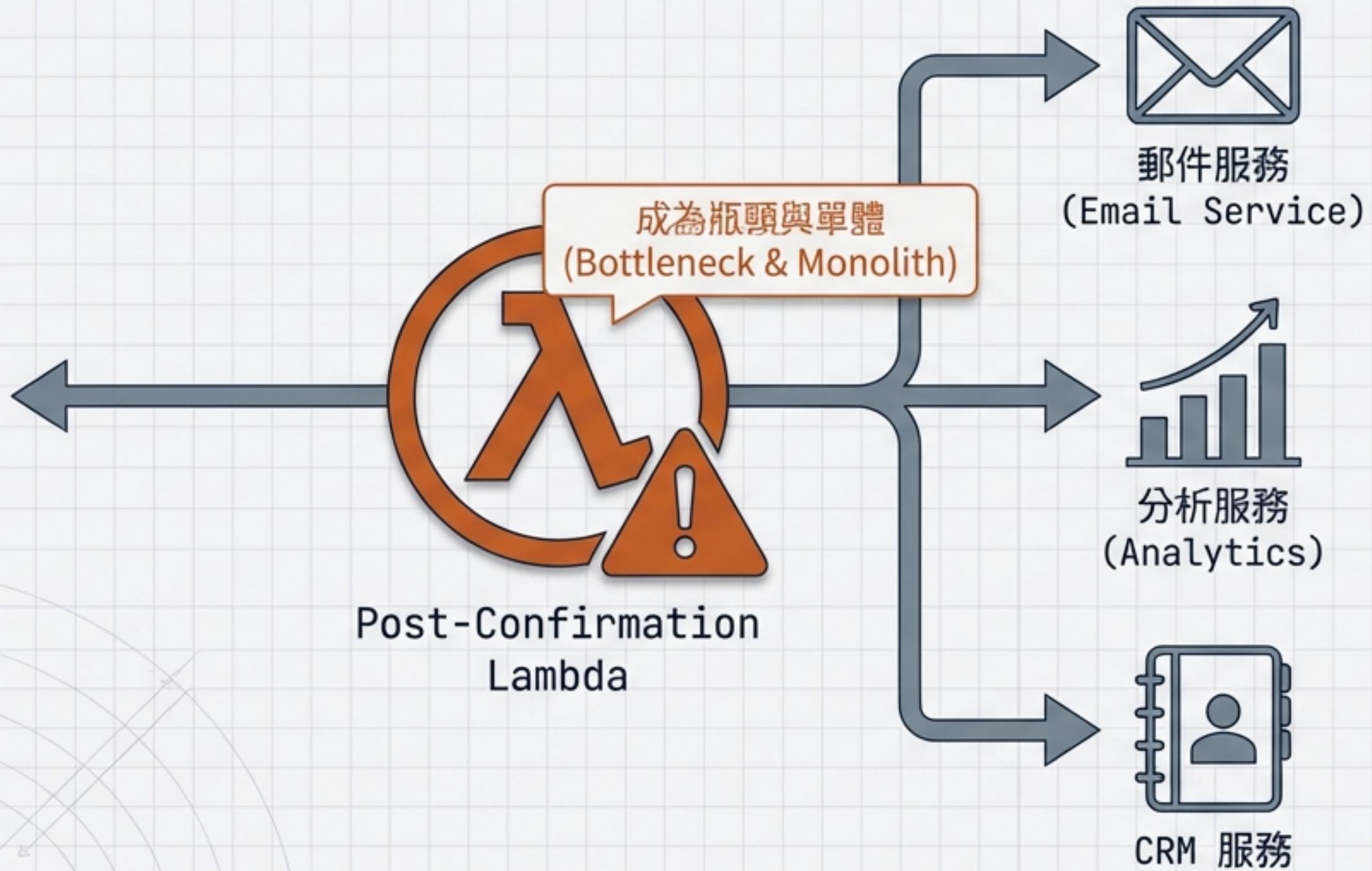
範例：註冊前自動確認使用者
(Pre-Signup Trigger)

postconfirmation/postConfirmation.mjs

```
export const handler = async (event) => {
  const { email, sub } = event.request.userAttributes;
  await docClient.send(
    new PutCommand({ ←
      TableName: "Users",
      Item: {
        userId: sub,
        email: email,
        createdAt: new Date().toISOString()
      }
    })
  );
  return event;
};
```

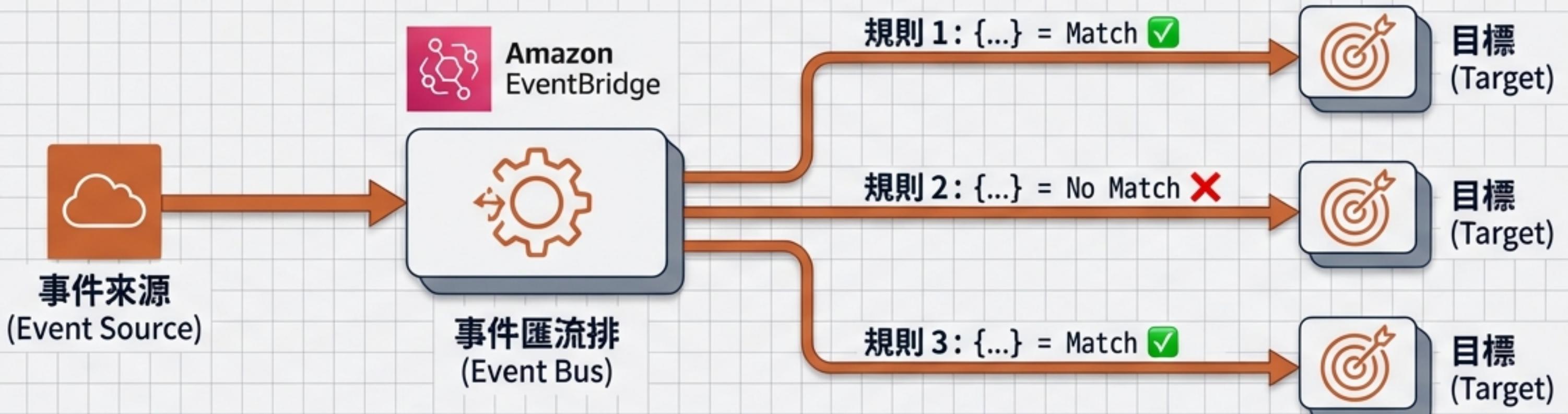
範例：在 DynamoDB 中建立使用者資料
(Post-Confirmation Trigger)

超越點對點觸發：系統級解耦的需求

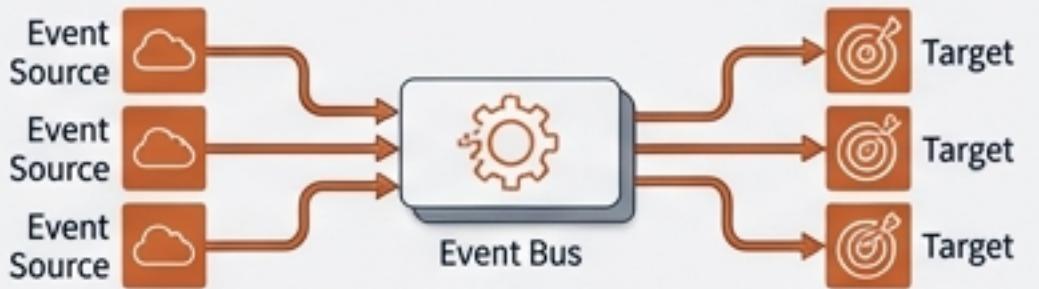


如果將所有邏輯都放入 Post-Confirmation Lambda 中，它會變得臃腫、脆弱且難以維護。我們要一個更強大的機制來廣播事件，實現真正的服務解耦。

解決方案之核心：以 Amazon EventBridge 作為系統神經中樞



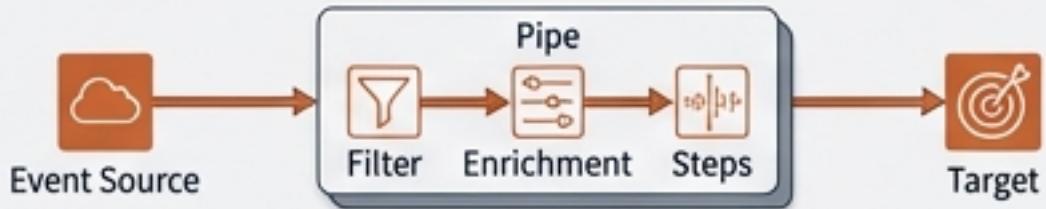
事件匯流排 (Event Buses)



多對多路由 (Many-to-Many)

- 用途：多對多路由。適用於將來自多個來源的事件路由到多個目標。
- 範例：「使用者已確認」事件發布到事件匯流排，多個服務（郵件、分析、CRM）可以訂閱並各自處理。

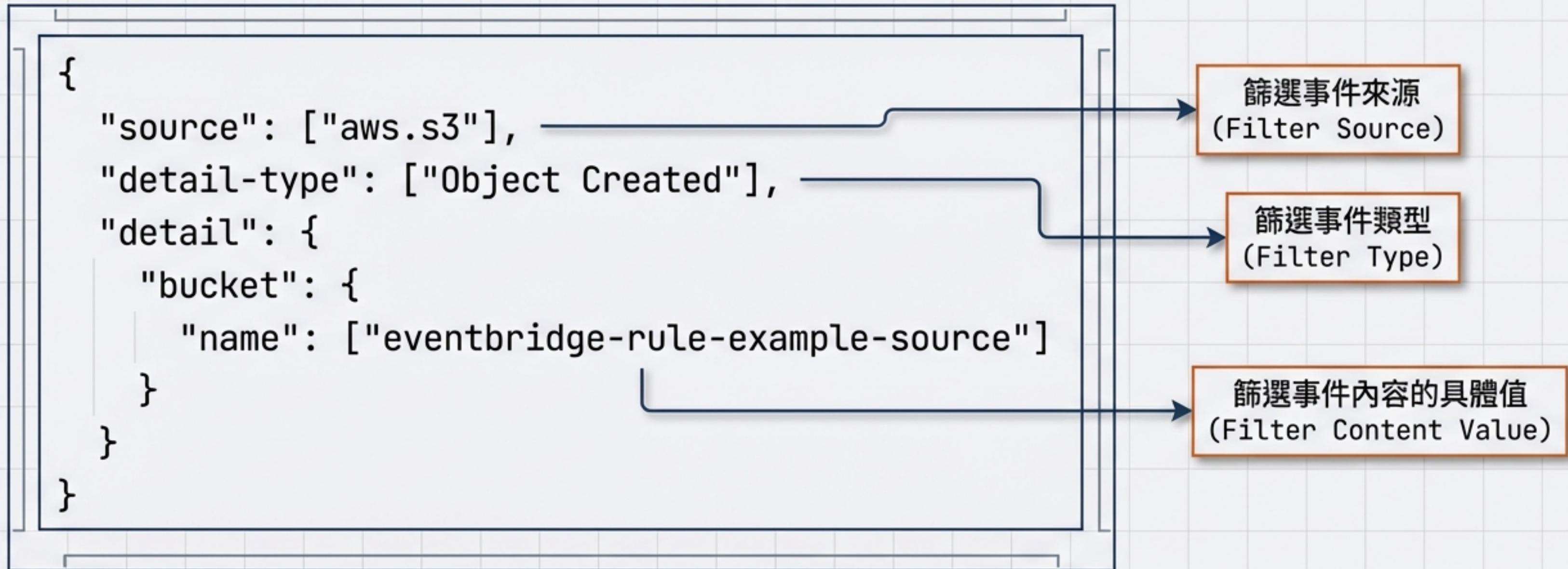
Event Buses vs. Pipes



點對點整合 (Point-to-Point)

- 用途：點對點整合。用於將單一來源的事件傳遞到單一目標，中間可選性地進行篩選、擴充和轉換。
- 範例：將 DynamoDB Stream 的變更事件直接、可靠地傳送到 SQS 佇列。

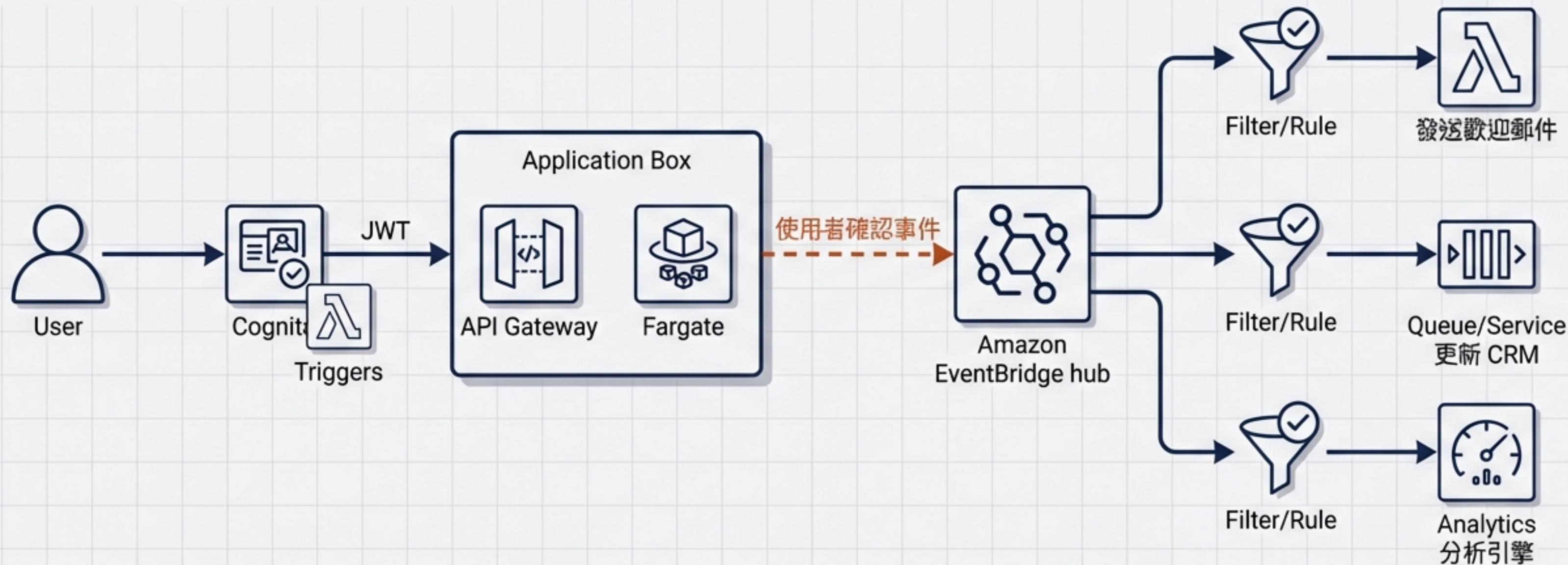
精準路由的藝術：使用事件模式 (Event Patterns) 進行過濾



透過定義「事件模式」，目標可以只訂閱它們感興趣的事件，而忽略所有其他流量。

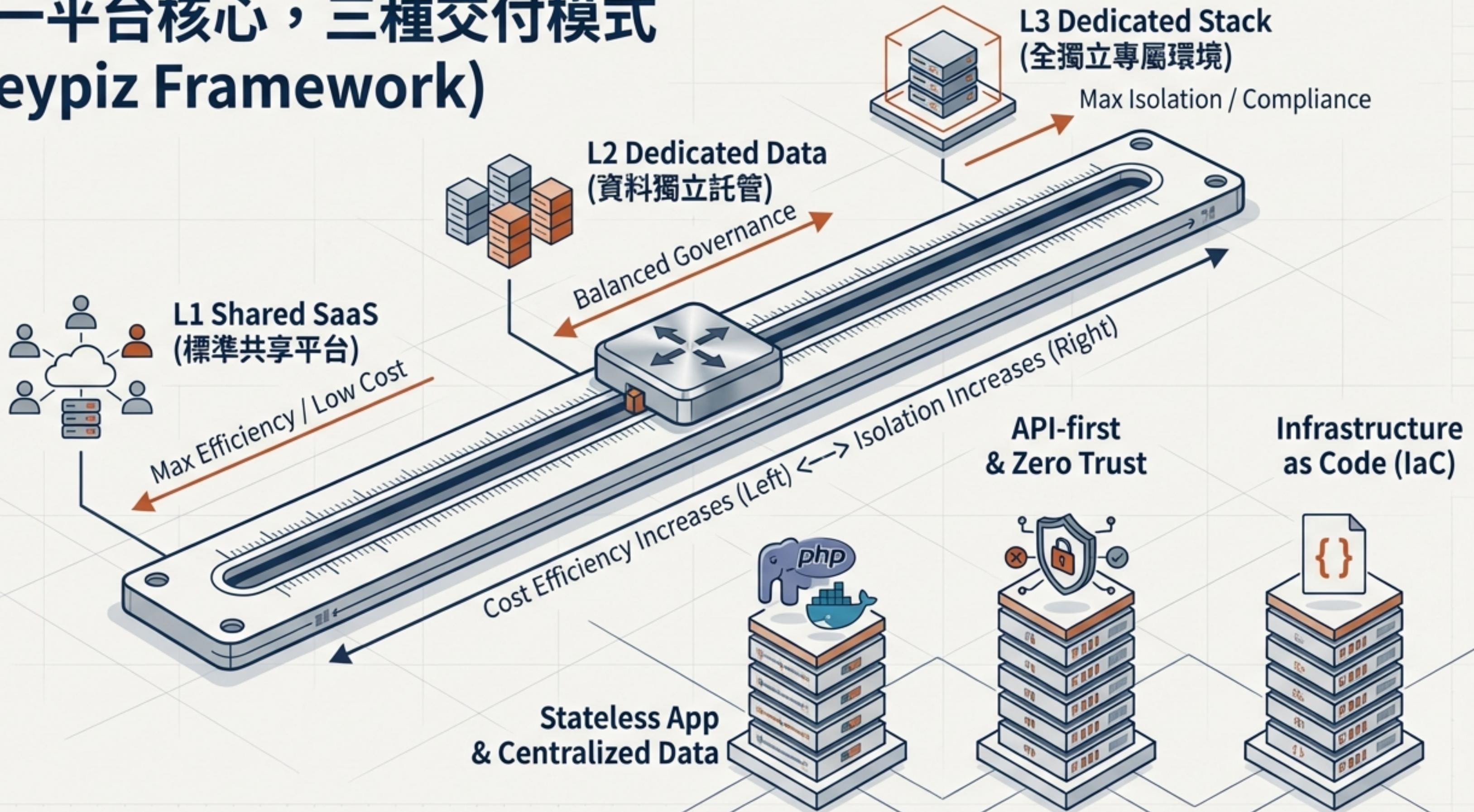
完整架構圖：從使用者到事件驅動的全景

這張圖展示了我們沿途構建的完整、現代化的應用程式架構。它將安全的身分驗證與可擴展的事件驅動設計無縫結合。



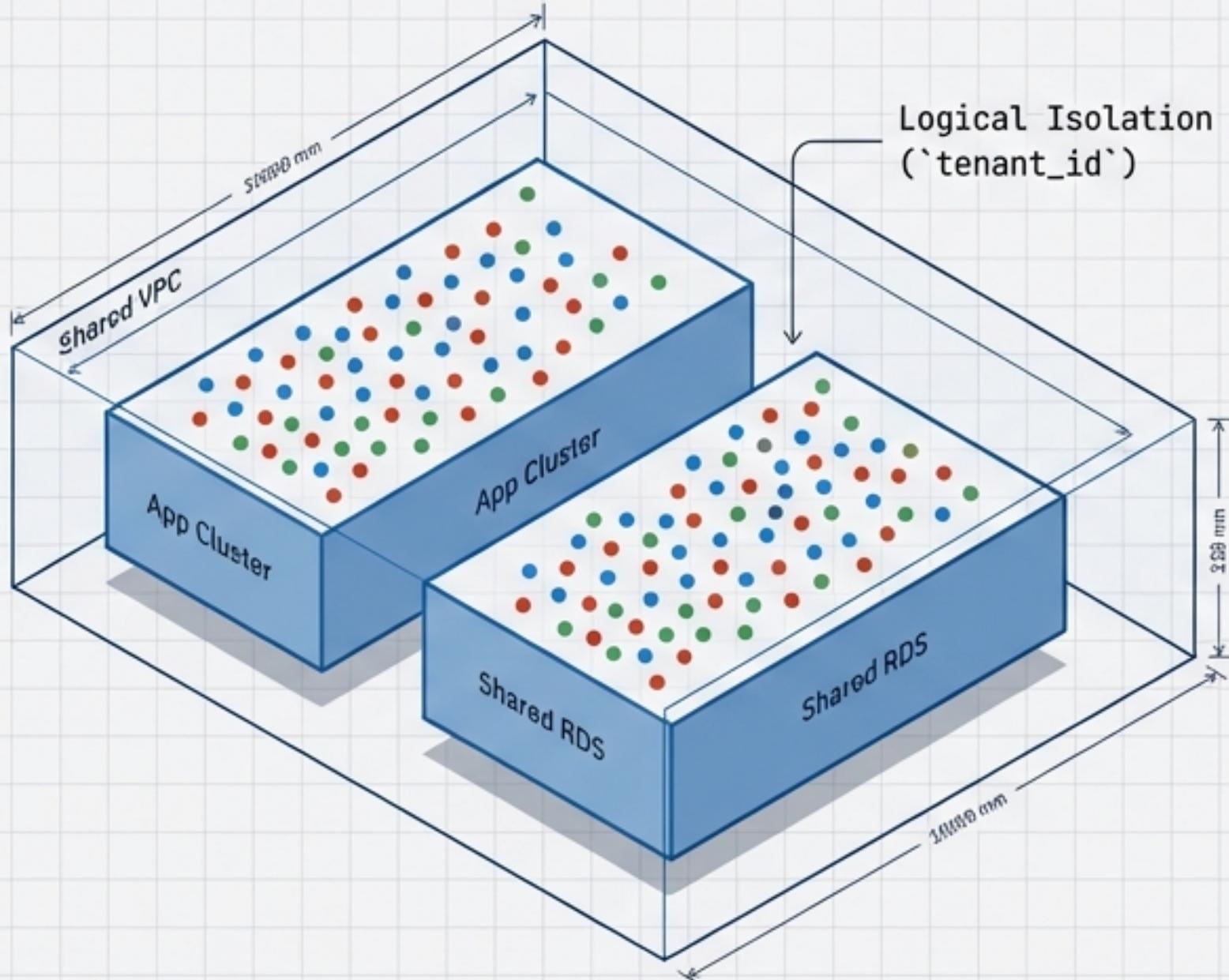
使用者登入 (Cognito) -> 觸發應用邏輯 -> 發送事件 (EventBridge) -> 非同步驅動下游服務

單一平台核心，三種交付模式 (Keypiz Framework)



L1 標準共享 vs. L2 資料獨立託管

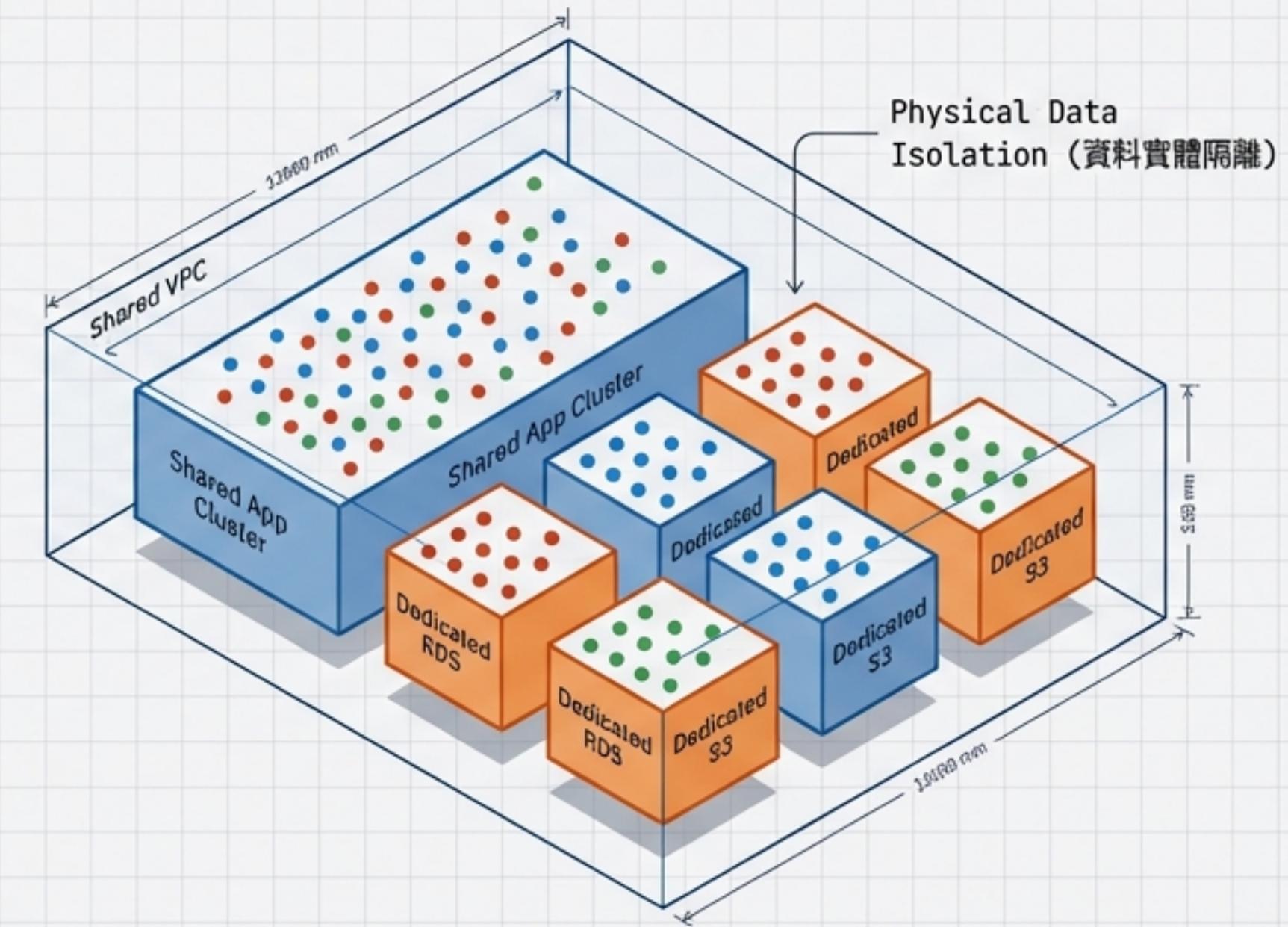
Level 1 (Shared SaaS)



SLA: 99.5%

Target: MVP / SMB

Level 2 (Dedicated Data)



SLA: 99.7%

Target: Mid-Market / Compliance

Level 3 | Dedicated Stack (全獨立專屬環境)

Target Audience

政府、金融、集團級部署

SLA

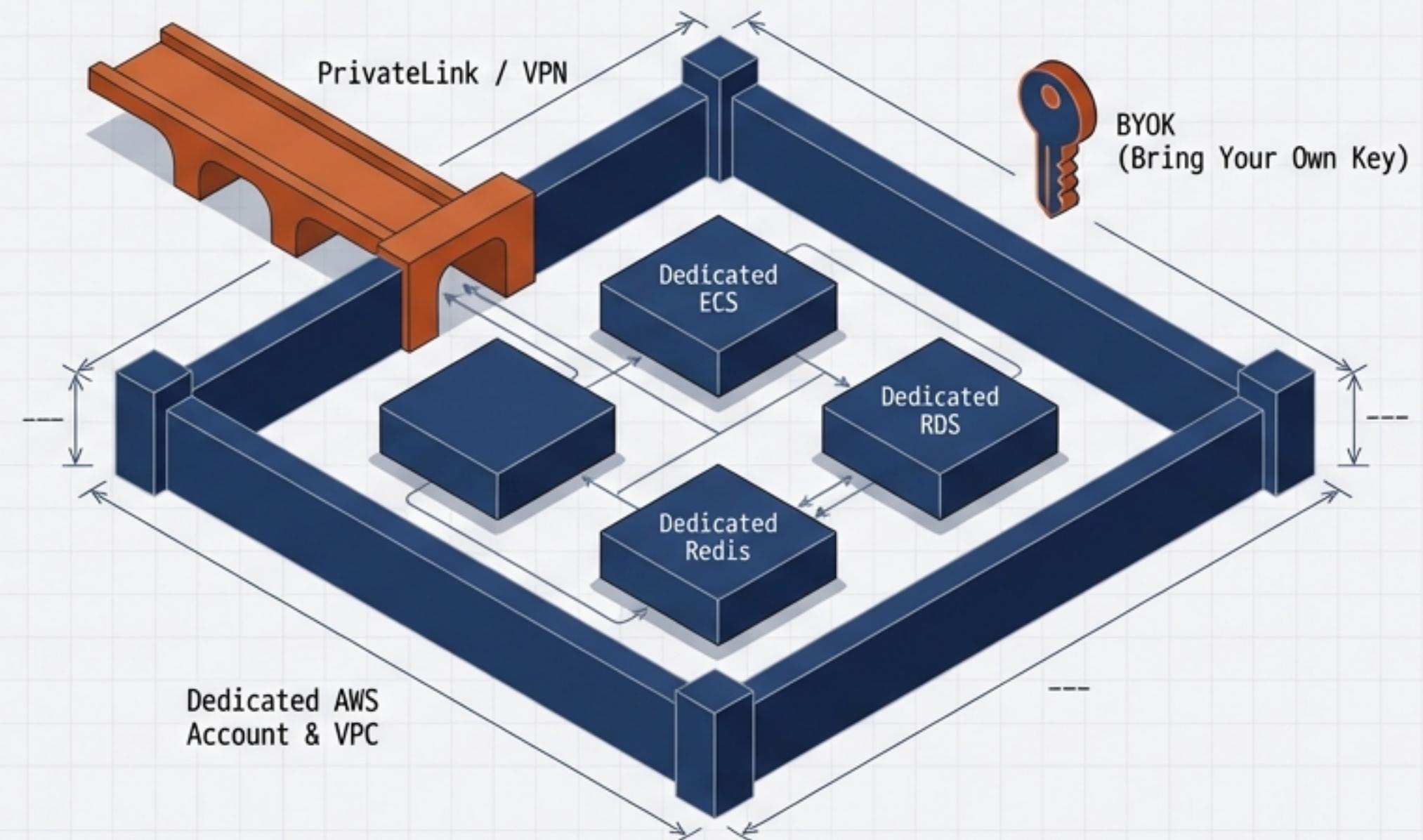
99.9%

Isolation Type

全環境實體隔離

Pricing

NT\$ 80,000 - 200,000+ / 月



高度隔離與合規，為關鍵業務與政府專案而生

功能差異與責任邊界矩陣

項目	Level 1 Shared	Level 2 Dedicated Data	Level 3 Dedicated Stack
租戶隔離 (Tenant Isolation)	邏輯隔離	資料獨立	環境獨立
應用程式 (App)	共用	共用	獨立
資料庫 (Database)	共用	獨立 (Dedicated)	獨立 (Dedicated)
備份策略 (Backup)	平台統一 (Daily)	獨立策略 (Custom)	專屬 (Dedicated)
升級節奏 (Upgrade)	平台統一	平台統一	專屬節奏 (Tenant Control)
法遵彈性 (Compliance)	基本	中高	最高 (BYOK)

Responsibility Note:

L3: 平台負責全端管理，但客戶擁有變更核准權 (Change Approval)。

您的企業級雲端夥伴：成長路徑與核心總結

Growth Arrow

Phase 1: Start Fast (L1/L2 導入)



Level 1/2 導入 (60天內可交付)

Growth Arrow

Phase 2: Scale Securely



Upgrade to Level 3

Focus: Enterprise / Risk Management

Growth Arrow

Phase 3: Future (Global/Event-Driven)



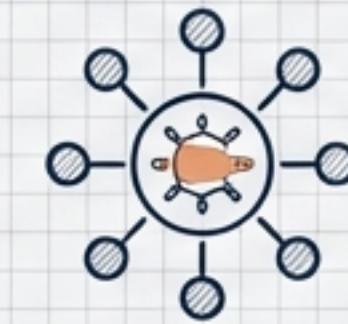
Multi-region / Event-driven



Identity First

從安全的身份層開始 (Cognito)

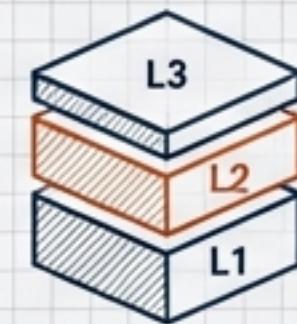
使用 AWS Cognito 作為起點，建立一個安全、可客製化的身份驗證基礎。不要自己動手造輪子。



Event Driven

用事件匯流排解耦一切 (EventBridge)

引入 Amazon EventBridge 作為系統的神經中樞，讓您的服務能獨立演進、部署和擴展，實現模塊的靈活性。



Flexible Isolation

為 B2B 擴展設計分層架構 (Keypiz L1-L3)

採用『單一 User Pool + 自訂屬性』的多租戶策略，以應對未來業務增長，同時保持一致的使用者體驗。

Technical Promise: 單一程式碼庫 (Single Codebase)，確保升級穩定性。