aws

上的資料庫分解 AWS

AWS 方案指引

AWS 方案指引: 上的資料庫分解 AWS

Table of Contents

上的資料庫分解 AWS

Amazon Web Services 的 Philippe Wanner 和 Saurabh Sharma

2025 年 10 月 (文件歷史記錄)

資料庫現代化，特別是單體資料庫的分解，對於想要改善其資料管理系統中敏捷性、可擴展性和效能的組織而言，是至關重要的工作流。隨著企業的成長及其資料需求變得更加複雜，傳統的單體資料庫通常難以跟上步伐。這會導致效能瓶頸、維護挑戰，以及難以適應不斷變化的業務需求。

以下是單體資料庫的常見挑戰：

- 業務網域不一致 – 單體資料庫通常無法使技術與不同的業務網域保持一致，這可能會限制組織的成長。
- 可擴展性限制 – 系統經常達到擴展限制，這會為業務擴展造成障礙。
- 架構剛性 – 緊密耦合的結構使更新特定元件變得困難，而不會影響整個系統。
- 效能降級 – 增加資料負載和增加使用者並行通常會導致系統效能下降。

以下是資料庫分解的優點：

- 增強業務敏捷性 – 分解可快速適應不斷變化的業務需求，並支援獨立擴展。
- 最佳化效能 – 分解可協助您建立專為特定使用案例量身打造的專用資料庫解決方案，並獨立擴展每個資料庫。
- 改善成本管理 – 分解可實現更有效率的資源使用率，並降低營運成本。
- 彈性的授權選項 – 分解可創造從昂貴的專屬授權轉換到開放原始碼替代方案的機會。
- 創新支援 – 分解有助於針對特定工作負載採用專用資料庫。

目標對象

本指南可協助資料庫架構師、雲端解決方案架構師、應用程式開發團隊和企業架構師。它旨在協助您將單體資料庫分解為符合微服務的資料存放區、實作網域驅動的資料庫架構、規劃資料庫遷移策略，以及擴展資料庫操作，以滿足不斷增長的業務需求。若要了解本指南中的概念和建議，您應該熟悉關聯式和 NoSQL 資料庫原則、 AWS 受管資料庫服務和微服務架構模式。本指南旨在協助處於資料庫分解專案初始階段的組織。

目標

本指南可協助您的組織達成下列目標：

- 收集分解目標架構的需求。
- 開發評估風險和溝通的系統性方法。
- 建立分解計畫。
- 定義成功指標、關鍵績效指標 KPIs)、緩解策略和業務持續性計劃。
- 建立更好的工作負載彈性，協助您遵循業務需求。
- 了解如何針對特定使用案例採用特殊化資料庫，進而實現創新。
- 強化組織的資料安全與控管。
- 透過下列方式降低成本：
  - 降低授權費用
  - 減少廠商鎖定
  - 改善對更廣泛的社群支援和創新的存取
  - 能夠為不同的元件選擇不同的資料庫技術
  - 逐步遷移，可降低風險並隨著時間推移分散成本
  - 改善資源使用率

資料庫分解的常見挑戰和管理責任

資料庫分解是一個複雜的程序，需要仔細的規劃、執行和管理。當組織尋求現代化其資料基礎設施時，他們通常會遇到許多可能影響其專案成功的挑戰。本節說明常見障礙，並引進結構化方法來克服這些障礙。

常見挑戰

資料庫分解專案在技術、人員和業務維度方面面臨多項挑戰。在技術方面，確保分散式系統之間的資料一致性會構成重大障礙。它也可能在轉換期間產生潛在的效能和穩定性影響，而且您必須與現有系統無縫整合。與人員相關的挑戰包括與新系統相關聯的學習曲線、員工的潛在變革阻力，以及必要資源的可用性。從業務角度來看，專案必須面對時間軸超支的風險、預算限制，以及遷移過程中業務中斷的可能性。

定義角色和責任

鑑於這些跨越技術、人員和業務維度的複雜挑戰，建立清晰的角色和責任對於專案成功至關重要。負責任、負責、諮詢和知情 (RACI) 矩陣提供必要的結構來應對這些挑戰。它明確定義誰做出決策、誰執行工作、誰提供意見，以及誰需要在分解的每個階段隨時掌握資訊。這種清晰度有助於防止因模棱兩可的決策而造成的延遲、鼓勵適當的利益相關者參與，並建立關鍵交付項目的責任。如果沒有這種架構，團隊可能會遇到責任重疊、溝通遺漏和升級路徑不清楚的問題，這些問題可能會加劇現有的技術複雜性和變革管理挑戰，同時提高時間表和預算超支的風險。

以下範例 RACI 矩陣是一個起點，可協助您釐清組織中的潛在角色和責任。

任務或活動	專案經理	架構師	開發人員	利益相關者
識別業務成果和挑戰	A/R	R	C	–
定義範圍並識別需求	A	R	C	C/I
識別專案成功指標	A	R	C	I

建立並執行通訊計畫	A/R	C	C	I
定義目標架構	I	A/R	C	–
控制資料庫存取	I	A/R	R	–
建立並執行業務持續性計畫	A/R	C	I	–
分析凝聚力和耦合	I	A/R	R	I
將商業邏輯（例如預存程序）從資料庫移至應用程式層	I	A	R	–
解耦資料表關係，稱為聯結	I	A	R	–

定義資料庫分解的範圍和需求

當您定義範圍並識別資料庫分解專案的需求時，您必須從組織的需求中恢復工作。這需要系統化的方法，在技術可行性與商業價值之間取得平衡。這個初始步驟為整個程序奠定基礎，並協助您確保專案的目標與組織的目標和功能保持一致。

本節包含下列主題：

- 建立核心分析架構
- 定義資料庫分解的系統邊界
- 考慮發行週期
- 評估資料庫分解的技術限制條件
- 了解組織內容
- 評估資料庫分解的風險
- 定義資料庫分解的成功條件

建立核心分析架構

範圍定義從系統化工作流程開始，引導分析完成四個互連階段。這種全面的方法可確保資料庫分解工作以對現有系統和操作需求的透徹了解為基礎。以下是核心分析架構中的階段：

1. 演員分析 – 徹底識別與資料庫互動的所有系統和應用程式。這包括映射執行寫入操作的生產者和處理讀取操作的取用者，同時記錄其存取模式、頻率和尖峰使用時間。此以客戶為中心的檢視可協助您了解任何變更的影響，並識別在分解期間需要特別注意的關鍵路徑。
2. 活動分析 – 深入了解每個演員執行的特定操作。您可以為每個系統建立詳細的建立、讀取、更新和刪除 (CRUD) 矩陣，並識別它們存取的資料表及其方式。此分析可協助您探索分解的自然界限，並反白顯示您可以簡化目前架構的區域。
3. 相依性映射 – 記錄系統之間的直接和間接相依性，建立資料流程和關係的清晰視覺化。這有助於識別需要仔細規劃以獲得信任的潛在中斷點和領域。分析會同時考慮技術相依性，例如共用資料表和外部金鑰，以及業務流程相依性，例如工作流程序列和報告需求。
4. 一致性要求 – 使用高標準檢查每個操作的一致性需求。判斷哪些操作需要立即的一致性，例如金融交易。其他操作可以最終一致性運作，例如分析更新。此分析會直接影響整個專案中分解模式和架構決策的選擇。

定義資料庫分解的系統邊界

系統邊界是邏輯周邊，可定義一個系統結束和另一個系統開始的位置，包括資料擁有權、存取模式和整合點。定義系統界限時，請做出深思熟慮但果斷的選擇，在全面規劃與實際實作需求之間取得平衡。將資料庫視為可能跨越多個實體資料庫或結構描述的邏輯單位。此界限定義可達成下列關鍵目標：

- 識別所有外部演員及其互動模式
- 全面映射傳入和傳出相依性
- 文件技術和操作限制
- 清楚描述分解工作的範圍

考慮發行週期

了解發行週期對於規劃資料庫分解至關重要。檢閱目標系統和任何相依系統的續約時間。識別協調變更的機會。請考慮任何計劃停用的連線系統，因為這可能會影響您的分解策略。考慮現有的變更時段和部署限制條件，將業務中斷降至最低。請確定您的實作計劃符合所有連線系統的發行排程。

評估資料庫分解的技術限制條件

在繼續進行資料庫分解之前，請評估將塑造現代化方法的關鍵技術限制。檢查目前技術堆疊的功能，包括資料庫版本、架構、效能需求和服務水準協議。考慮安全和合規要求，尤其是受監管的產業。檢閱目前的資料磁碟區、成長預測和可用的遷移工具，以通知您的擴展決策。最後，請確認您對原始程式碼和系統修改的存取權，因為這些權限將決定可行的分解策略。

了解組織內容

成功的資料庫分解需要您了解系統運作所在的更廣泛的組織環境。映射跨部門相依性，並在團隊之間建立明確的溝通管道。評估您團隊的技術能力，並找出您需要解決的任何訓練需求或技能差距。考慮變革管理影響，包括如何管理轉換和維護業務連續性。評估可用資源和任何限制，例如預算或人員配置限制。最後，將您的分解策略與利益相關者的期望和優先事項保持一致，以在整個專案中促進持續支援。

評估資料庫分解的風險

全面的風險評估對於資料庫分解成功至關重要。仔細評估風險，例如遷移期間的資料完整性、潛在的系統效能降低、可能的整合失敗，以及安全漏洞。這些技術挑戰必須平衡業務風險，包括潛在的營運中

斷、資源限制、時間軸延遲和預算限制。針對每個已識別的風險，制定特定的緩解策略和應變計畫，以維持專案動能，同時保護業務營運。

建立風險矩陣，以評估潛在問題的影響和機率。與技術團隊和業務利益相關者合作，以識別風險、設定明確的介入閾值，並制定特定的緩解策略。例如，將資料遺失風險評分為高影響和低機率，而且需要強大的備份策略。輕微效能降低可能是中度影響和高機率，而且需要主動監控。

建立定期風險審查週期，以重新評估優先順序，並在專案發展時調整緩解計劃。這種系統性方法可確保資源專注於最關鍵的風險，同時為新興問題維持明確的呈報路徑。

定義資料庫分解的成功條件

資料庫分解的成功條件必須明確定義且可跨多個維度測量。從業務角度來看，建立降低成本、改善 time-to-market、系統可用性和客戶滿意度的特定目標。技術成功應透過可量化的系統效能、部署效率、資料一致性和整體可靠性改善來衡量。對於遷移程序，請定義對零資料遺失、可接受的業務中斷限制、預算合規和時間表遵循的嚴格要求。

透過維護基準和目標指標、明確的測量方法和定期檢閱排程，徹底記錄這些條件。為每個成功指標指派明確的擁有者，並在不同的指標之間映射相依性。這種衡量成功的全方位方法使技術成就與業務成果保持一致，同時在整個分解過程中保持責任制。

在分解期間控制資料庫存取

許多組織面臨一個常見案例：一個集中式資料庫，多年來已自然成長，並且由多個服務和團隊直接存取。這會產生幾個關鍵問題：

- 不受控制的成長 – 當團隊持續新增功能和修改結構描述時，資料庫變得越來越複雜且難以管理。
- 效能考量 – 即使有硬體改善，不斷增加的負載最終仍威脅要超過資料庫的功能。由於結構描述複雜性或缺乏技能而無法調整查詢。無法預測或解釋系統效能。
- 分解癱瘓 – 當多個團隊主動修改資料庫時，幾乎不可能分割或重構資料庫。

> **Note**
>
> 單體資料庫系統通常會針對應用程式或服務或管理重複使用相同的登入資料。這會導致資料庫可追蹤性不佳。設定專用角色並採用最低權限原則可協助您提高安全性和可用性。

處理變得不可靠的單體資料庫時，控制存取的最有效模式之一稱為資料庫包裝函式服務。它提供管理複雜資料庫系統的策略第一步。它建立受控制的資料庫存取，並啟用漸進式現代化，同時降低風險。此方法透過提供資料使用模式和相依性的清晰可見性，為增量改進奠定了基礎。它是一種轉換架構，可做為完全資料庫分解的步驟。包裝函式服務提供讓旅程成功所需的穩定性和控制。

本節包含下列主題：

- 使用資料庫包裝函式服務模式控制存取
- 使用 CQRS 模式控制存取

使用資料庫包裝函式服務模式控制存取

包裝函式服務是一種服務層，可做為資料庫的外觀。當您需要維護現有的功能，同時準備未來的分解時，這種方法特別有價值。此模式遵循簡單的原則 - 當某些內容過於雜亂時，請先包含雜亂。包裝函式服務會成為存取資料庫的唯一授權方式，提供受控界面，同時隱藏基礎複雜性。

當因為複雜結構描述而無法立即分解資料庫，或多個服務需要持續資料存取時，請使用此模式。它在轉換期間特別重要，因為它提供時間仔細重構，同時保持系統穩定性。當將資料擁有權合併至特定團隊，或當新應用程式需要跨多個資料表的彙總檢視時， 模式運作良好。

例如，在下列情況下套用此模式：

- 結構描述複雜性可防止立即分離
- 多個團隊需要持續的資料存取
- 建議採用漸進式現代化
- 團隊重組需要明確的資料擁有權
- 新應用程式需要合併的資料檢視

資料庫包裝函式服務模式的優點和限制

以下是資料庫包裝函式模式的優點：

- 受控成長 – 包裝函式服務可防止對資料庫結構描述進行進一步的受控新增。
- 明確界限 – 實作程序可協助您建立明確的擁有權和責任界限。
- 重構自由 – 包裝服務可讓您進行內部變更，而不會影響消費者。
- 改善可觀測性 – 包裝服務是監控和記錄的單一點。
- 簡化測試 – 包裝服務可讓您更輕鬆地使用 服務，以建立簡化的模擬版本進行測試。

以下是資料庫包裝函式模式的限制。

- 技術耦合 – 當包裝函式服務使用與耗用服務相同的技術堆疊時，效果最佳。
- 初始負荷 – 包裝函式服務需要額外可能會影響效能的基礎設施。
- 遷移工作 – 若要實作包裝函式服務，您必須跨團隊進行協調，以從直接存取轉換。
- 效能 – 如果包裝服務遇到高流量、大量使用或頻繁存取，則耗用服務可能會遇到效能不佳的情況。在資料庫上，包裝函式服務必須處理分頁、游標和資料庫連線。根據您的使用案例，它可能無法妥善擴展，而且可能不適合擷取、轉換和載入 (ETL) 工作負載。

實作資料庫包裝函式服務模式

實作資料庫包裝函式服務模式有兩個階段。首先，您要建立資料庫包裝函式服務。然後，您可以引導所有存取，並記錄存取模式。

階段 1：建立資料庫包裝函式服務

建立輕量型服務層，做為資料庫的守門員。一開始，它應該反映所有現有的功能。此包裝函式服務會成為所有資料庫操作的強制性存取點，這會將直接資料庫相依性轉換為服務層級相依性。在此層實作詳細

的記錄和監控，以追蹤使用模式、效能指標和存取頻率。維護現有的預存程序，但請確保它們只能透過這個新的服務界面存取。

階段 2：實作存取控制

透過包裝函式服務有系統地重新導向所有資料庫存取，然後從直接存取資料庫的外部系統撤銷直接資料庫許可。在服務遷移時記錄每個存取模式和相依性。此受控存取可讓資料庫元件的內部重構，而不會中斷外部取用者。例如，從低風險的唯讀操作開始，而不是複雜的交易工作流程。

階段 3：監控資料庫效能

使用包裝函式服務做為資料庫效能的集中監控點。追蹤關鍵指標，包括查詢回應時間、用量模式、錯誤率和資源使用率。設定效能閾值和異常模式的提醒。例如，監控慢速執行的查詢、連線集區使用率和交易輸送量，以主動識別潛在問題。

使用此合併檢視，透過查詢調校、資源配置調整和用量模式分析來最佳化資料庫效能。包裝函式服務的集中式性質，可讓您更輕鬆地實作改善項目，並驗證其對所有消費者的影響，同時維持一致的效能標準。

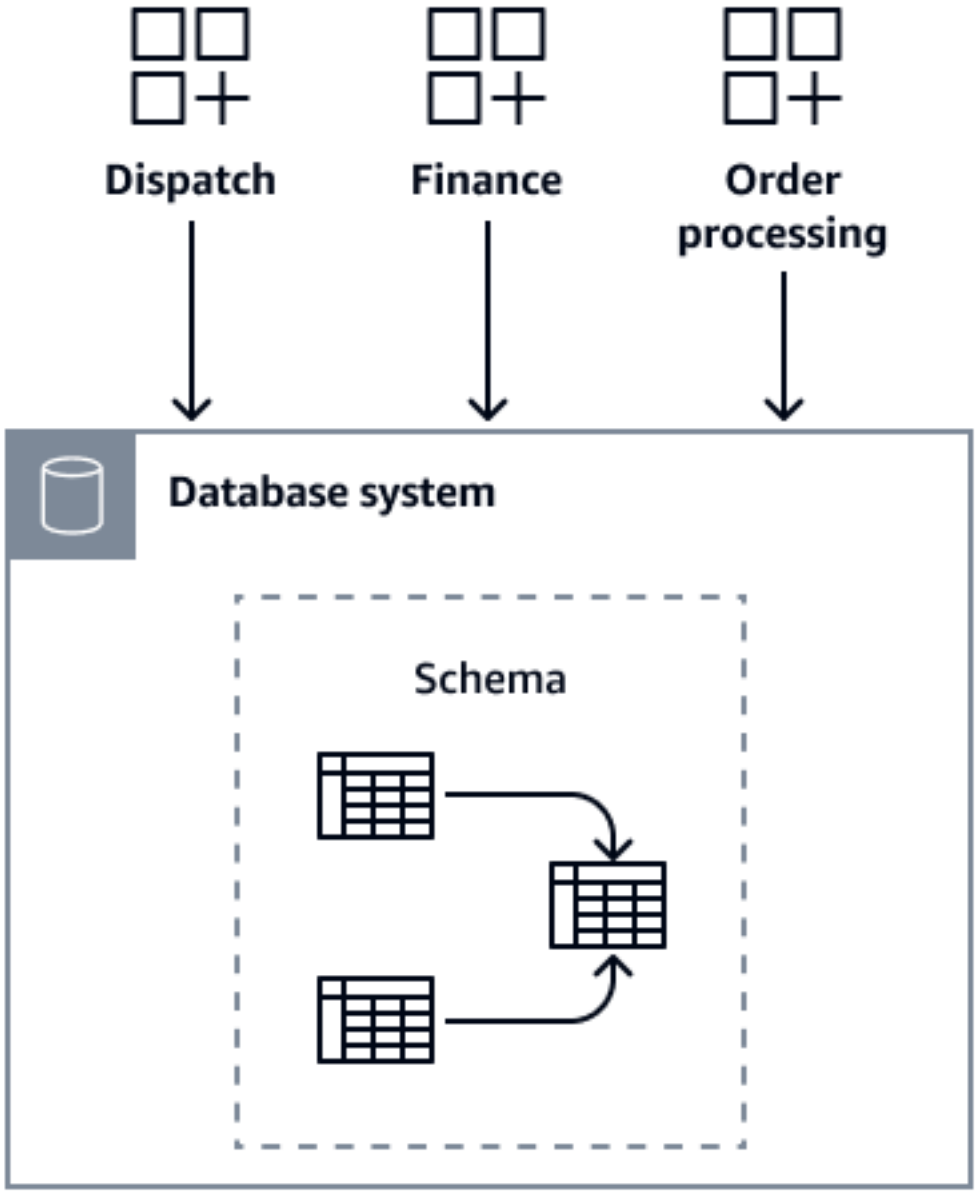
實作資料庫包裝函式服務的最佳實務

下列最佳實務可協助您實作資料庫包裝函式服務：

- 從小型開始 – 從僅代理現有功能的最小包裝函式開始
- 保持穩定性 – 保持服務界面穩定，同時進行內部改善
- 監控用量 – 實作全面監控以了解存取模式
- 明確擁有權 – 指派專用團隊來維護包裝函式和基礎結構描述
- 鼓勵本機儲存 – 鼓勵團隊將資料存放在自己的資料庫中

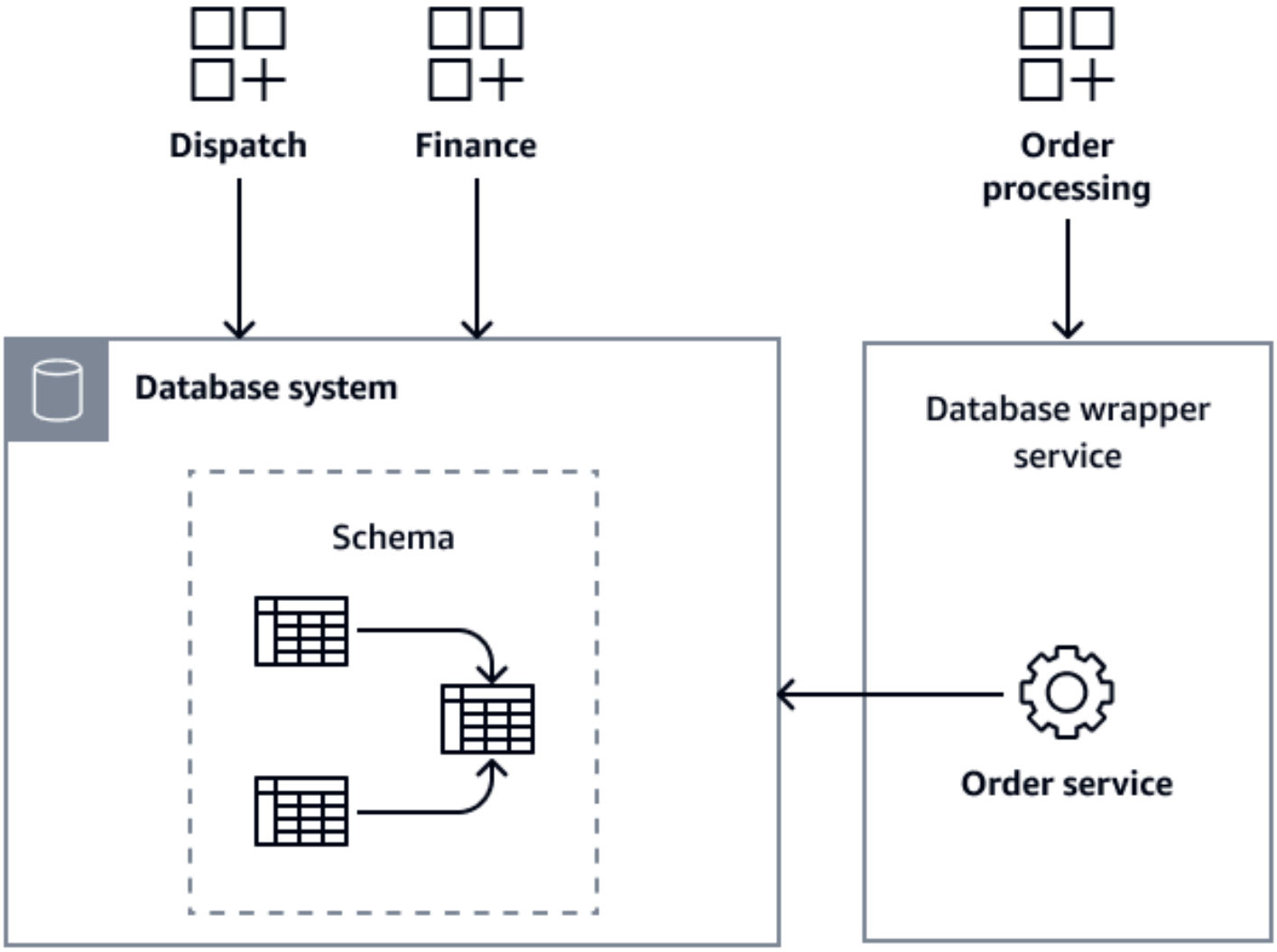
以案例為基礎的範例

本節說明名為 AnyCompany Books 的虛構公司如何使用資料庫包裝函式模式來控制存取其整體資料庫系統的範例。AnyCompany Books 有三個關鍵服務：分派、財務和訂單處理。這些服務共用中央資料庫的存取權。每個服務都由不同的團隊維護。隨著時間的推移，他們會獨立修改資料庫結構描述，以滿足其特定需求。這導致了相依性的交錯 Web 和越來越複雜的資料庫結構。



公司的應用程式或企業架構師了解需要分解此整體資料庫。他們的目標是為每個服務提供自己的專用資料庫，以改善可維護性並減少跨團隊相依性。不過，他們面臨重大挑戰 – 幾乎不可能分解資料庫，而這三個團隊都會繼續積極修改資料庫以進行中的專案。團隊之間持續的結構描述變更和缺乏協調性，使得嘗試任何重大重組具有極高風險。

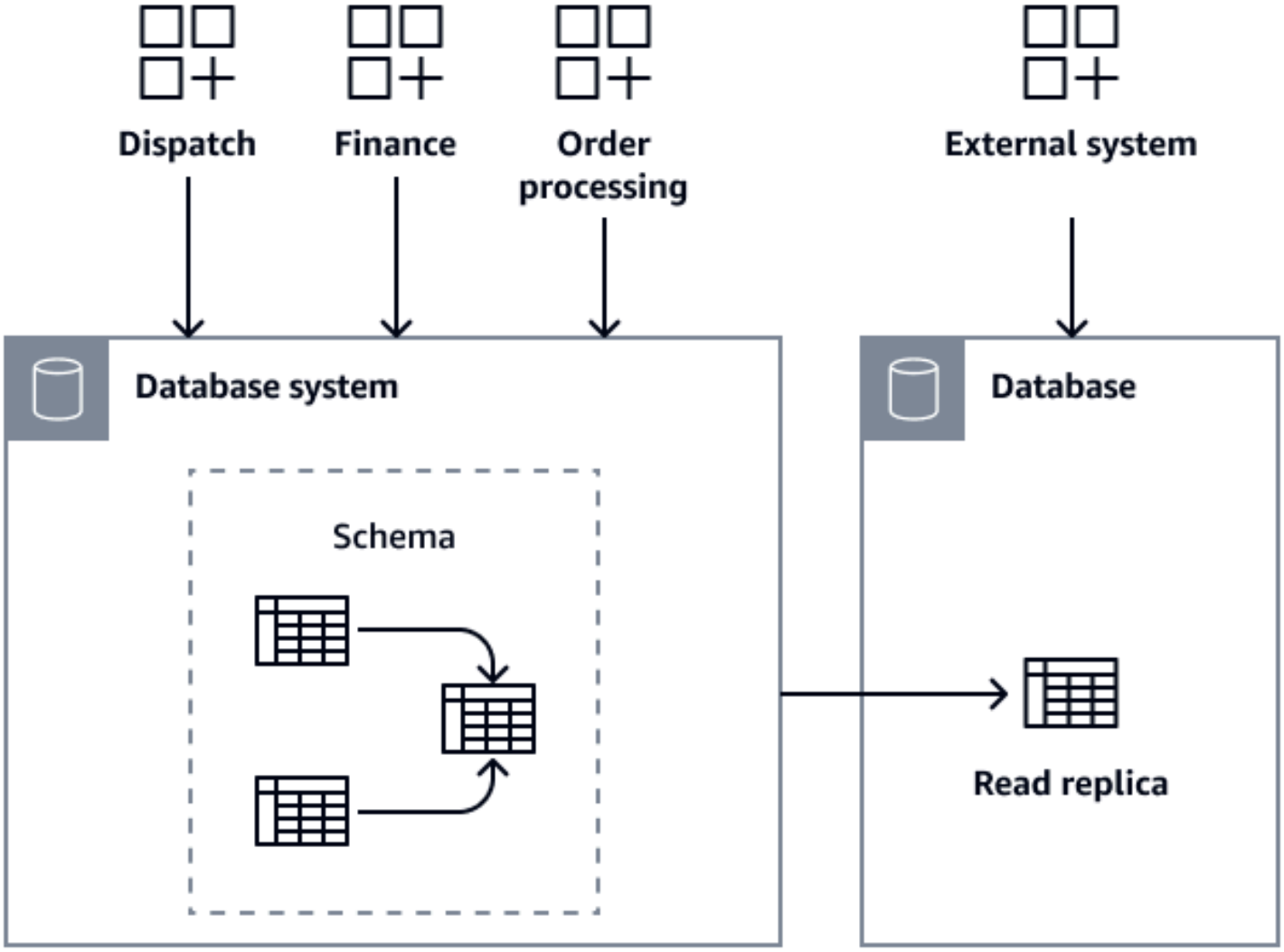
架構師使用資料庫包裝函式服務模式來開始控制對單體資料庫的存取。首先，他們為特定模組設定資料庫包裝函式服務，稱為訂單服務。然後，他們會重新導向訂單處理服務以存取包裝函式服務，而不是直接存取資料庫。下圖顯示修改後的基礎設施。



使用 CQRS 模式控制存取

另一個您可以用來隔離連接到此中央資料庫之外部系統的模式是命令查詢責任隔離 (CQRS)。如果某些外部系統主要用於讀取，例如分析、報告或其他讀取密集型操作，您可以建立個別的讀取最佳化資料存放區。

此模式可有效地隔離這些外部系統，避免資料庫分解和結構描述變更的影響。透過維護專用僅供讀取複本或特定查詢模式的專用資料存放區，團隊可以繼續其操作，而不會受到主要資料庫結構變更的影響。例如，當您分解單體資料庫時，報告系統可以繼續使用其現有的資料檢視，而分析工作負載可以透過專用分析存放區維護其目前的查詢模式。這種方法提供技術隔離並啟用組織自主權，因為不同的團隊可以獨立發展其系統，而不會與主要資料庫的轉型旅程緊密結合。



如需此模式的詳細資訊及其用於分離資料表關係的範例，請參閱本指南CQRS 模式稍後的 。

分析資料庫分解的凝聚和耦合

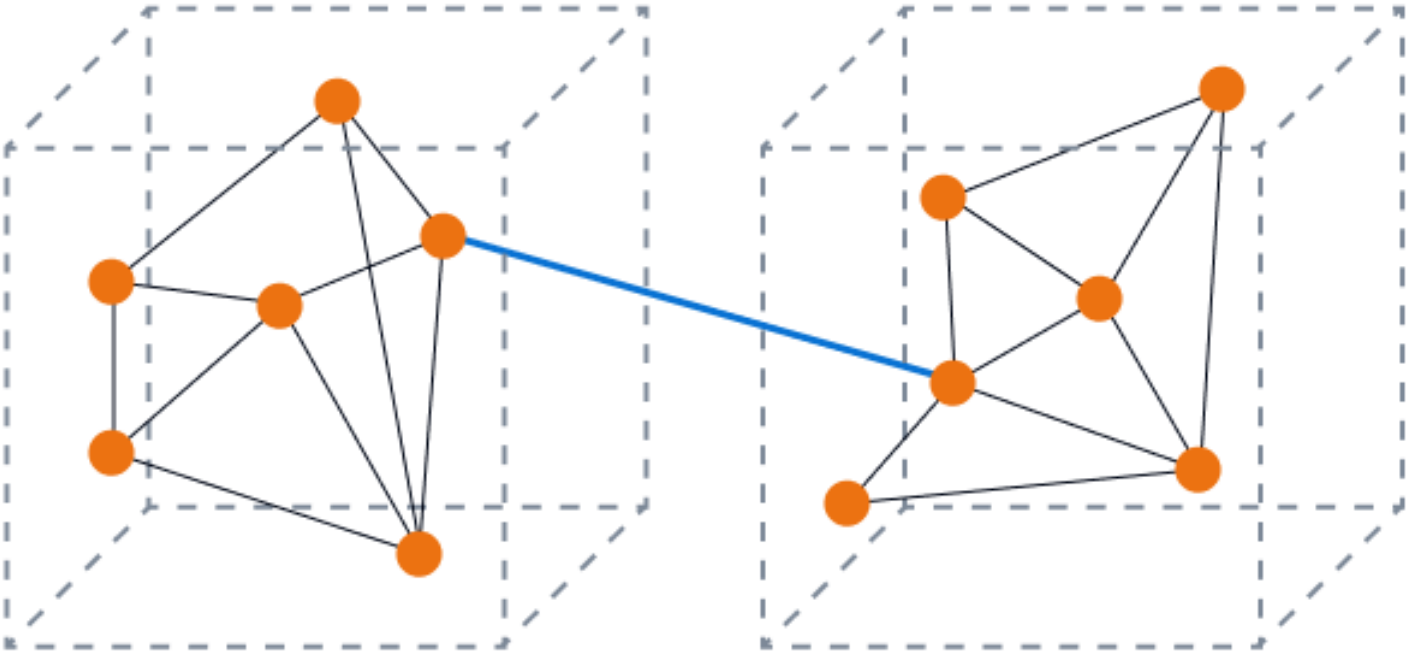
本節可協助您分析單體資料庫中的耦合和凝聚模式，以引導其分解。了解資料庫元件如何相互互動和依賴，對於識別自然中斷點、評估複雜性和規劃分階段遷移方法至關重要。此分析會顯示隱藏的相依性、反白顯示適合立即分離的區域，並協助您排定分解工作的優先順序，同時將轉換風險降至最低。透過檢查耦合和凝聚力，您可以對元件分離序列做出明智的決策，以在整個轉換過程中維持系統穩定性。

本節包含下列主題：

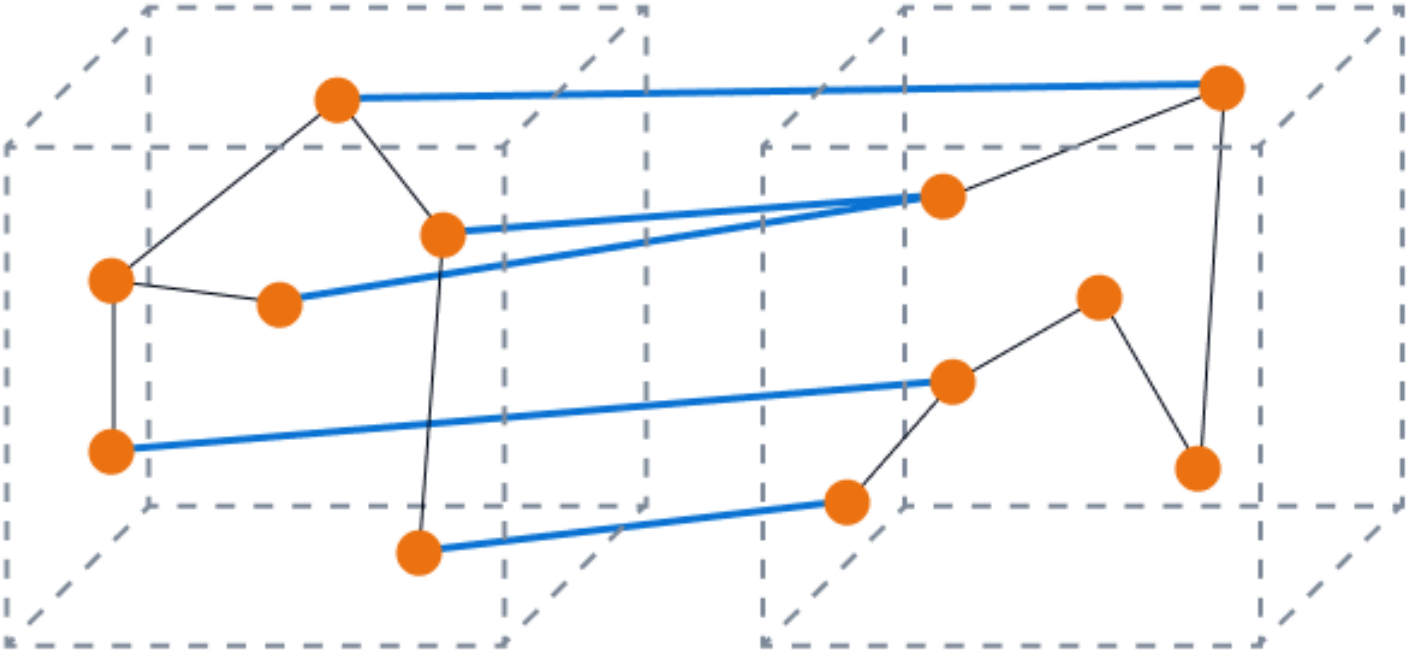
關於凝聚和耦合

耦合會測量資料庫元件之間的相互依存性。在設計良好的系統中，您想要實現鬆散耦合，其中對一個元件的變更對其他元件的影響最小。Cohesion 會測量資料庫元件中的元素搭配運作的程度，以提供單一且明確定義的用途。高度黏性表示元件的元素具有強烈關聯性，並專注於特定函數。分解單體資料庫時，您必須分析個別元件內的凝聚力，以及它們之間的聯結。此分析可協助您做出明智的決策，了解如何分解資料庫，同時維護系統完整性和效能。

下圖顯示具有高黏著性的鬆散耦合。資料庫中的元件會一起運作，以執行特定函數，而且您可以將變更對單一元件的影響降至最低。這是理想的狀態。



下圖顯示高耦合與低黏著性。資料庫元件會中斷連線，而變更很可能會影響其他元件。



單體資料庫中的常見耦合模式

將單體資料庫分解為微服務特定的資料庫時，通常會發現數種耦合模式。了解這些模式對於成功的資料庫現代化計劃至關重要。本節說明每種模式、其挑戰，以及減少耦合的最佳實務。

實作耦合模式

定義：元件在程式碼和結構描述層級緊密互連。例如，修改customer資料表的結構會影響 order、inventory和 billing服務。

現代化影響：每個微服務都需要自己的專用資料庫結構描述和資料存取層。

挑戰：

- 共用資料表的變更會影響多個 服務
- 發生意外副作用的高風險
- 測試複雜性提高
- 難以修改個別元件

減少耦合的最佳實務：

- 定義元件之間的明確界面
- 使用抽象層隱藏實作詳細資訊
- 實作網域特定的結構描述

時間耦合模式

定義：操作必須以特定順序執行。例如，在庫存更新完成之前，訂單處理無法繼續。

現代化影響：每個微服務都需要自動控制資料。

挑戰：

- 中斷服務之間的同步相依性
- 效能瓶頸
- 難以最佳化
- 有限的平行處理

減少耦合的最佳實務：

- 盡可能實作非同步處理

- 使用事件驅動型架構
- 設計適當的最終一致性

部署耦合模式

定義：系統元件必須部署為單一單位。例如，付款處理邏輯的次要變更需要重新部署整個資料庫。

現代化影響：每個服務的獨立資料庫部署

挑戰：

- 高風險部署
- 有限的部署頻率
- 複雜的轉返程序

減少耦合的最佳實務：

- 分解成可獨立部署的元件
- 實作資料庫碎片策略
- 使用藍綠部署模式

網域耦合模式

定義：商業網域共用資料庫結構和邏輯。例如，`customer`、 `order`和 `inventory`網域共用資料表和預存程序。

現代化影響：網域特定資料隔離

挑戰：

- 複雜網域邊界
- 難以擴展個別網域
- 扭曲的業務規則

減少耦合的最佳實務：

- 識別明確的網域邊界

- 依網域內容分隔資料
- 實作特定網域的服務

單體資料庫中的常見凝聚模式

評估資料庫元件進行分解時，通常會發現數種凝聚模式。了解這些模式對於識別結構良好的資料庫元件至關重要。本節說明每種模式、其特性，以及強化凝聚力的最佳實務。

功能凝聚模式

定義：所有元素都直接支援並有助於執行單一、定義明確的函數。例如，付款處理模組中的所有預存程序和資料表只會處理與付款相關的操作。

現代化影響：微型服務資料庫設計的理想模式

挑戰：

- 識別明確的功能界限
- 分離混合用途元件
- 維護單一責任

強化凝聚力的最佳實務：

- 將相關函數分組在一起
- 移除不相關的功能
- 定義清晰的元件邊界

順序凝聚模式

定義：一個元素的輸出會變成另一個元素的輸入。例如，訂單饋送至訂單處理的驗證結果。

現代化影響：需要仔細的工作流程分析和資料流程映射

挑戰：

- 管理步驟之間的相依性
- 處理失敗案例

- 維護程序順序

強化凝聚力的最佳實務：

- 文件清除資料流程
- 實作適當的錯誤處理
- 在步驟之間設計清晰的界面

通訊凝聚力模式

定義：元素在相同的資料上運作。例如，客戶設定檔管理功能都可以使用客戶資料。

現代化影響：協助識別服務分離的資料邊界，以減少模組之間的耦合

挑戰：

- 判斷資料擁有權
- 管理共用資料存取
- 維持資料一致性

強化凝聚力的最佳實務：

- 定義明確的資料擁有權
- 實作適當的資料存取模式
- 設計有效的資料分割

程序性凝聚模式

定義：元素會分組在一起，因為它們必須以特定順序執行，但可能不會與功能相關。例如，在順序處理中，處理訂單驗證和使用者通知的預存程序會分組在一起，只是因為它們會依序發生，即使它們有不同的用途，並且可以由不同的服務處理。

現代化影響：需要謹慎區隔程序，同時維護流程

挑戰：

- 分解後維持正確的處理流程

- 識別與程序相依性相比的真正功能界限

強化凝聚力的最佳實務：

- 根據程序的功能用途而非執行順序來分隔程序
- 使用協同運作模式來管理程序流程
- 實作複雜序列的工作流程管理系統
- 設計事件驅動型架構，以獨立處理程序步驟

暫時凝聚力模式

定義：元素與時間要求相關。例如，下訂單時，數個操作必須一起執行：庫存檢查、付款處理、訂單確認和運送通知都必須在特定時段內發生，才能維持一致的訂單狀態。

現代化影響：可能需要在分散式系統中進行特殊處理

挑戰：

- 協調分散式服務的計時相依性
- 管理分散式交易
- 跨多個元件確認程序完成

強化凝聚力的最佳實務：

- 實作適當的排程機制和逾時
- 使用事件驅動型架構搭配明確的序列處理
- 設計與補償模式的最終一致性
- 實作分散式交易的 saga 模式

邏輯或同時的凝聚力模式

定義：元素在邏輯上會分類為執行相同動作，即使它們的關係較弱或沒有有意義的關係。其中一個範例是在相同的資料庫結構描述中存放客戶訂單資料、倉儲庫存計數和行銷電子郵件範本，因為它們都與銷售操作相關，儘管有不同的存取模式、生命週期管理和擴展需求。另一個範例是在相同的資料庫元件中結合訂單付款處理和產品目錄管理，因為它們都是電子商務系統的一部分，即使它們為具有不同營運需求的不同業務職能提供服務。

現代化影響：應重構或重組

挑戰：

- 識別更好的組織模式
- 中斷不必要的相依性
- 重組任意分組的元件

強化凝聚力的最佳實務：

- 根據真正的功能界限和業務領域進行重組
- 根據表面關係移除任意分組
- 根據業務能力實作適當的元素區隔
- 使資料庫元件符合其特定操作需求

實作低耦合和高凝聚力

最佳實務

下列最佳實務可協助您實現低耦合：

- 將資料庫元件之間的相依性降至最低
- 使用明確定義的界面進行元件互動
- 避免共用狀態和全域資料結構

下列最佳實務可協助您實現高凝聚力：

- 將相關資料和操作分組在一起
- 確保每個元件都有一個明確的責任
- 在不同業務網域之間維持明確的界限

階段 1：映射資料相依性

映射資料關係並識別自然界限。您可以使用 等工具SchemaSpy，透過在實體關係 (ER) 圖表中顯示資料表來視覺化資料庫。這可提供資料庫的靜態分析，並指出資料庫中的一些明確界限和相依性。

您也可以在圖形資料庫或Jupiter筆記本中匯出資料庫結構描述。然後，您可以套用叢集或互連元件演算法，以識別自然界限和相依性。其他 AWS Partner 工具，例如 CAST Imaging，可協助了解您的資料庫相依性。

階段 2：分析交易界限和存取模式

分析交易模式以維護原子性、一致性、隔離性、耐久性 (ACID) 屬性，並了解資料的存取和修改方式。您可以使用資料庫分析和診斷工具，例如 Oracle Automatic Workload Repository (AWR)或 PostgreSQL pg_stat_statements。此分析可協助您了解誰正在存取資料庫，以及交易界限是什麼。它也可以協助您了解執行時間資料表之間的凝聚和耦合。您也可以使用監控和分析工具來連結程式碼和資料庫執行設定檔，例如 Dynatrace AppEngine。

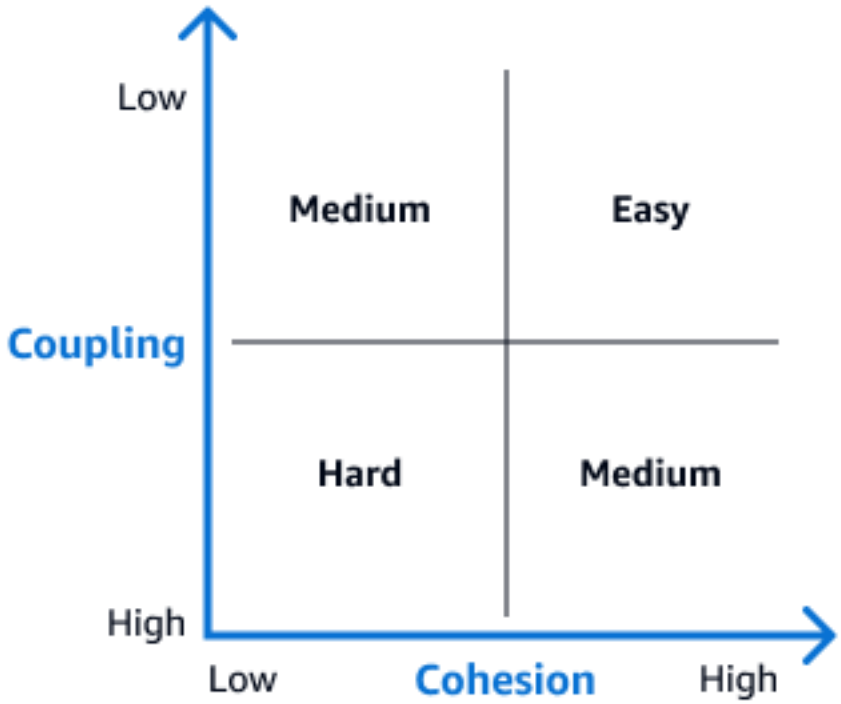
AI 工具，例如 vFunction，可以透過分析應用程式的功能和網域邊界來協助您識別網域邊界。雖然vFunction主要會分析應用程式層，但其洞見可以引導應用程式和資料庫的分解，以支援與業務網域的一致性。

階段 3：識別獨立資料表

尋找示範兩個關鍵特性的資料表：

- 高度黏著 – 資料表的內容彼此密切相關
- 低耦合 – 它們對其他資料表的相依性最低。

下列耦合黏附矩陣可協助您識別解耦每個資料表的難度。出現在此矩陣右上角的資料表是初始解耦工作的理想候選者，因為它們是最容易分離的。在 ER 圖表中，這些資料表幾乎沒有外部金鑰關係或其他相依性。解耦這些資料表之後，將進入具有更複雜關係的資料表。



> **Note**
>
> 資料庫結構通常會鏡像應用程式架構。在資料庫層級較容易解耦的資料表通常對應於在應用程式層級較容易轉換為微服務的元件。

將商業邏輯從資料庫遷移至應用程式層

從資料庫存放的程序、觸發條件和函數遷移商業邏輯到應用程式層服務，是分解單體資料庫的關鍵步驟。此轉換可改善服務自主權、簡化維護，並增強可擴展性。本節提供分析資料庫邏輯、規劃遷移策略，然後在維持業務連續性的同時實作轉換的指引。它還討論了建立有效的轉返計劃。

本節包含下列主題：

- 階段 1：分析商業邏輯
- 階段 2：分類商業邏輯
- 階段 3：遷移商業邏輯
- 商業邏輯的轉返策略

階段 1：分析商業邏輯

現代化單體資料庫時，您必須先對現有的資料庫邏輯進行全面的分析。此階段著重於三個主要類別：

- 存放程序通常包含重要的業務操作，包括資料處理邏輯、業務規則、驗證檢查和計算。作為應用程式商業邏輯的核心元件，它們需要仔細分解。例如，金融組織的預存程序可能會處理興趣計算、帳戶對帳和合規檢查。
- 觸發是處理稽核追蹤、資料驗證、計算和跨資料表一致性的關鍵資料庫元件。例如，零售組織可能會使用觸發來管理整個訂單處理系統的庫存更新，這示範了自動化資料庫操作的複雜性。
- 資料庫中的函數主要管理資料轉換、計算和查詢操作。它們通常嵌入到多個程序和應用程式中。例如，醫療保健組織可能會使用 函數來標準化患者資料或查詢醫療代碼。

每個類別代表內嵌在資料庫層中的業務邏輯的不同層面。您需要仔細評估和規劃每個項目，以便將它們遷移到應用程式層。

在此分析階段，客戶通常會面臨三個重大挑戰。首先，透過巢狀程序呼叫、跨結構描述參考和隱含資料相依性來產生複雜的相依性。其次，交易管理變得至關重要，特別是在處理多步驟交易和維護分散式系統之間的資料一致性時。第三，必須仔細評估效能考量，特別是目前受益於接近資料的批次處理操作、大量資料更新和即時計算。

若要有效地解決這些挑戰，您可以使用 AWS Schema Conversion Tool (AWS SCT) 進行初始分析，然後使用詳細的相依性映射工具。此方法可協助您了解資料庫邏輯的完整範圍，並建立全面的遷移策略，以在分解期間維持業務連續性。

透過徹底了解這些元件和挑戰，您可以更好地規劃現代化旅程，並針對遷移至微服務型架構期間要優先考慮哪些元素做出明智的決策。

分析資料庫程式碼元件時，請為每個預存程序、觸發條件和函數建立完整的文件。首先清楚地描述其目的和核心功能，包括其實作的業務規則。詳細說明所有輸入和輸出參數，並記下其資料類型和有效範圍。映射其他資料庫物件、外部系統和下游程序的相依性。明確定義交易界限和隔離要求，以維護資料完整性。記錄任何效能期望，包括回應時間需求和資源使用率模式。最後，分析用量模式，以了解尖峰負載、執行頻率和關鍵業務期間。

階段 2：分類商業邏輯

有效的資料庫分解需要跨關鍵維度對資料庫邏輯進行系統性分類：複雜性、業務影響、相依性、使用模式和遷移困難。此分類可協助您識別高風險元件、判斷測試需求，以及建立遷移優先順序。例如，具有高業務影響和頻繁使用的複雜預存程序需要仔細規劃和廣泛的測試。不過，具有最少相依性的簡單、很少使用函數可能適用於早期遷移階段。

這種結構化方法會建立平衡的遷移藍圖，將業務中斷降至最低，同時維持系統穩定性。透過了解這些相互關聯性，您可以改善分解工作的順序，並適當地配置資源。

階段 3：遷移商業邏輯

在您分析並分類商業邏輯之後，是時候遷移它了。從單體資料庫遷移商業邏輯時有兩種方法：將資料庫邏輯移至應用程式層，或將商業邏輯移至屬於微服務一部分的另一個資料庫。

如果您將商業邏輯遷移至應用程式，則資料庫資料表只會存放資料，而且資料庫不包含任何商業邏輯。這是建議的方法。您可以使用 Isquarer 或生成式 AI 工具，例如 Amazon Q Developer 或 Kiro，來轉換應用程式層的資料庫商業邏輯，例如轉換為 Java。如需詳細資訊，請參閱將商業邏輯從資料庫遷移到應用程式，以加快創新速度和靈活性 (AWS 部落格文章）。

如果您將商業邏輯遷移至另一個資料庫，您可以使用 AWS Schema Conversion Tool (AWS SCT) 將現有的資料庫結構描述和程式碼物件轉換為目標資料庫。它支援專用 AWS 資料庫服務，例如 Amazon DynamoDB、Amazon Aurora 和 Amazon Redshift。透過提供全面的評估報告和自動化轉換功能，AWS SCT 有助於簡化轉換程序，讓您專注於最佳化新的資料庫結構，以提高效能和可擴展性。隨著現代化專案的進行， AWS SCT 可以處理增量轉換以支援分階段方法，讓您能夠驗證和微調資料庫轉換的每個步驟。

商業邏輯的轉返策略

任何分解策略的兩個關鍵層面是維持回溯相容性並實作全面的轉返程序。這些元素共同運作，以協助在轉換期間保護操作。本節說明如何在分解過程中管理相容性，並建立有效的緊急復原功能，以防止潛在問題。

維持回溯相容性

在資料庫分解期間，維持回溯相容性對於順暢轉換至關重要。暫時保留現有的資料庫程序，同時逐步實作新功能。使用版本控制來追蹤所有變更，並同時管理多個資料庫版本。規劃較長的共存期間，其中來源和目標系統都必須可靠地運作。這提供了在淘汰舊版元件之前測試和驗證新系統的時間。此方法可將業務中斷降至最低，並視需要提供復原的安全網路。

緊急復原計劃

全方位的復原策略對於安全資料庫分解至關重要。在程式碼中實作功能旗標，以控制哪個商業邏輯版本處於作用中狀態。這可讓您在新實作和原始實作之間立即切換，而無需變更部署。此方法提供對轉換的精細控制，並在發生問題時協助您快速復原。將原始邏輯保留為已驗證的備份，並維護詳細的轉返程序，以指定觸發、責任和復原步驟。

在各種條件下定期測試這些復原案例，以驗證其有效性，並確保團隊熟悉緊急程序。功能旗標也會針對特定使用者群組或交易選擇性地啟用新功能，以啟用逐步推展。這在轉換期間提供額外的風險緩解層。

在資料庫分解期間解耦資料表關係

本節提供在單體資料庫分解期間分解複雜資料表關係和 JOIN 操作的指引。資料表聯結會根據資料表之間的相關資料欄，結合來自兩個或多個資料表的資料列。分隔這些關係的目標是減少資料表之間的高度耦合，同時維持微型服務之間的資料完整性。

本節包含下列主題：

- 非標準化策略
- Reference-by-key策略
- CQRS 模式
- 事件型資料同步
- 實作資料表聯結的替代方案
- 以案例為基礎的範例

非標準化策略

非標準化是一種資料庫設計策略，透過合併或複製跨資料表的資料，刻意引入備援。將大型資料庫分成小型資料庫時，跨服務複製一些資料可能很合理。例如，在行銷服務和訂單服務中存放基本客戶詳細資訊，例如姓名和電子郵件地址，無需持續進行跨服務查詢。行銷服務可能需要行銷活動目標的客戶偏好設定和聯絡資訊，而訂單服務則需要相同的資料以進行訂單處理和通知。雖然這會產生一些資料備援，但可以大幅提升服務效能和獨立性，讓行銷團隊在不依賴即時客戶服務查詢的情況下操作行銷活動。

實作非標準化時，請專注於透過仔細分析資料存取模式來識別的經常存取欄位。您可以使用工具，例如Oracle AWR報告或 pg_stat_statements，來了解通常一起擷取哪些資料。網域專家也可以提供對自然資料分組的寶貴洞見。請記住，去標準化不是all-or-nothing的方法，只是可明顯改善系統效能或降低複雜相依性的重複資料。

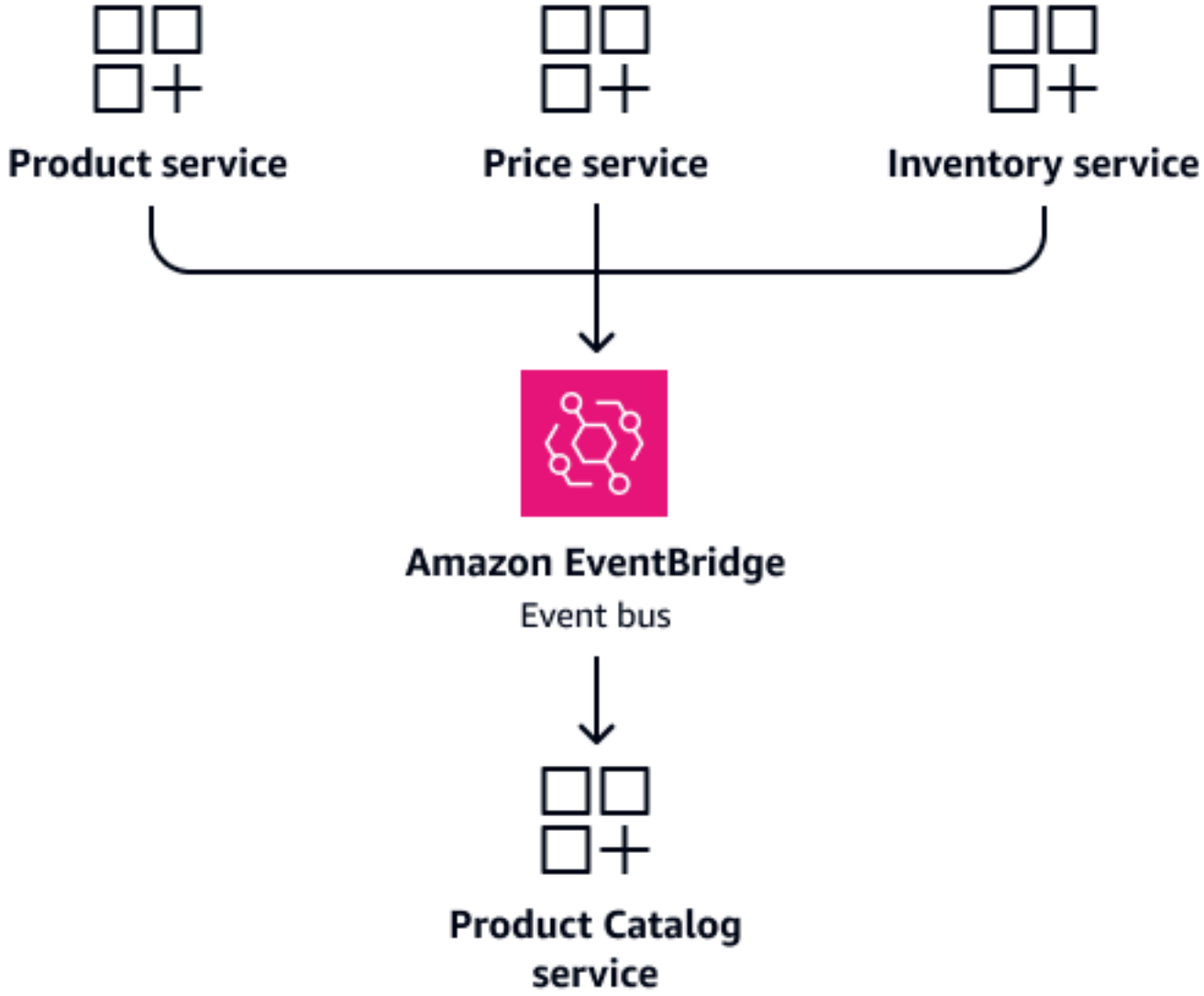
Reference-by-key策略

reference-by-key策略是一種資料庫設計模式，其中實體之間的關係是透過唯一的金鑰來維護，而不是儲存實際的相關資料。現代微服務通常只儲存相關資料的唯一識別符，而不是傳統的外部金鑰關係。例如，訂單服務不會將所有客戶詳細資訊保留在訂單資料表中，而是只會存放客戶 ID，並在需要時透過 API 呼叫擷取其他客戶資訊。此方法可維持服務獨立性，同時確保存取相關資料。

CQRS 模式

命令查詢責任隔離 (CQRS) 模式會區隔資料存放區的讀取和寫入操作。此模式在具有高效能需求的複雜系統中特別有用，尤其是具有非對稱讀取/寫入負載的系統。如果您的應用程式經常需要來自多個來源的資料組合，您可以建立專用 CQRS 模型，而不是複雜的聯結。例如，不是在每個請求上加入 `Product`、 `Pricing`和 `Inventory`資料表，而是維護包含必要資料的合併`Product Catalog`資料表。這種方法的好處可能會超過額外資料表的成本。

假設 `Product`、 `PriceInventory`和服務經常需要產品資訊的情況。建立專用服務，而不是將這些服務設定為直接存取共用資料表`Product Catalog`。此服務會維護自己的資料庫，其中包含合併的產品資訊。它可做為產品相關查詢的單一事實來源。當產品詳細資訊、價格或庫存層級變更時，個別服務可以發佈事件來更新`Product Catalog`服務。這可提供資料一致性，同時保持服務獨立性。下圖顯示此組態，其中 Amazon EventBridge 做為事件匯流排。



如下一節所述事件型資料同步，透過事件保持 CQRS 模型的更新狀態。當產品詳細資訊、價格或庫存層級變更時，個別的服務會發佈事件。Product Catalog 服務會訂閱這些事件並更新其合併檢視。這可在沒有複雜聯結的情況下提供快速讀取，並維持服務獨立性。

事件型資料同步

事件型資料同步是擷取資料變更並將其傳播為事件的模式，可讓不同的系統或元件維持同步資料狀態。當資料變更時，而不是立即更新所有相關資料庫，請發佈事件以通知訂閱的服務。例如，當客戶變更服務中的運送地址時Customer，CustomerUpdated事件會根據每個OrderDelivery服務的排程啟動服務和服務的更新。這種方法使用靈活、可擴展的事件驅動更新來取代剛性資料表聯結。有些服務可能會短暫有過時的資料，但權衡是改善系統可擴展性和服務獨立性。

實作資料表聯結的替代方案

使用讀取操作開始資料庫分解，因為它們通常更易於遷移和驗證。讀取路徑穩定後，請處理更複雜的寫入操作。對於關鍵的高效能需求，請考慮實作 CQRS 模式。針對讀取使用個別的最佳化資料庫，同時針對寫入維護另一個資料庫。

透過為跨服務呼叫新增重試邏輯並實作適當的快取層來建置彈性系統。密切監控服務互動，並設定資料一致性問題的提醒。最終目標是不是任何地方都完美的一致性，而是建立獨立服務，其效能良好，同時為您的業務需求維持可接受的資料準確性。

微服務解耦的性質在資料管理中引入了以下新的複雜性：

- 資料已分發。資料現在位於由獨立 服務管理的個別資料庫中。
- 跨服務進行即時同步通常不切實際，需要最終一致性模型。
- 先前在單一資料庫交易中發生的操作現在橫跨多個 服務。

若要解決這些挑戰，請執行下列動作：

- 實作事件驅動架構 – 使用訊息佇列和事件發佈，在 服務之間傳播資料變更。如需詳細資訊，請參閱在無伺服器土地上建置事件驅動架構。
- 採用 saga 協同運作模式 – 此模式可協助您管理分散式交易，並維護跨服務的資料完整性。如需詳細資訊，請參閱 AWS 部落格上的使用saga 協同運作模式建置無伺服器分散式應用程式。
- 故障設計 – 整合重試機制、斷路器和補償交易，以處理網路問題或服務故障。
- 使用版本戳記 – 追蹤資料版本以管理衝突，並確保套用最新的更新。
- 定期對帳 – 實作定期資料同步程序，以擷取和修正任何不一致。

以案例為基礎的範例

下列結構描述範例有兩個資料表：Customer資料表和Order資料表：

```
-- Customer table
CREATE TABLE customer (
    customer_id INT PRIMARY KEY,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    email VARCHAR(255),
    phone VARCHAR(20),
    address TEXT,
    created_at TIMESTAMP
);

-- Order table
CREATE TABLE order (
    order_id INT PRIMARY KEY,
    customer_id INT,
    order_date TIMESTAMP,
    total_amount DECIMAL(10,2),
    status VARCHAR(50),
    FOREIGN KEY (customer_id) REFERENCES customers(id)
);
```

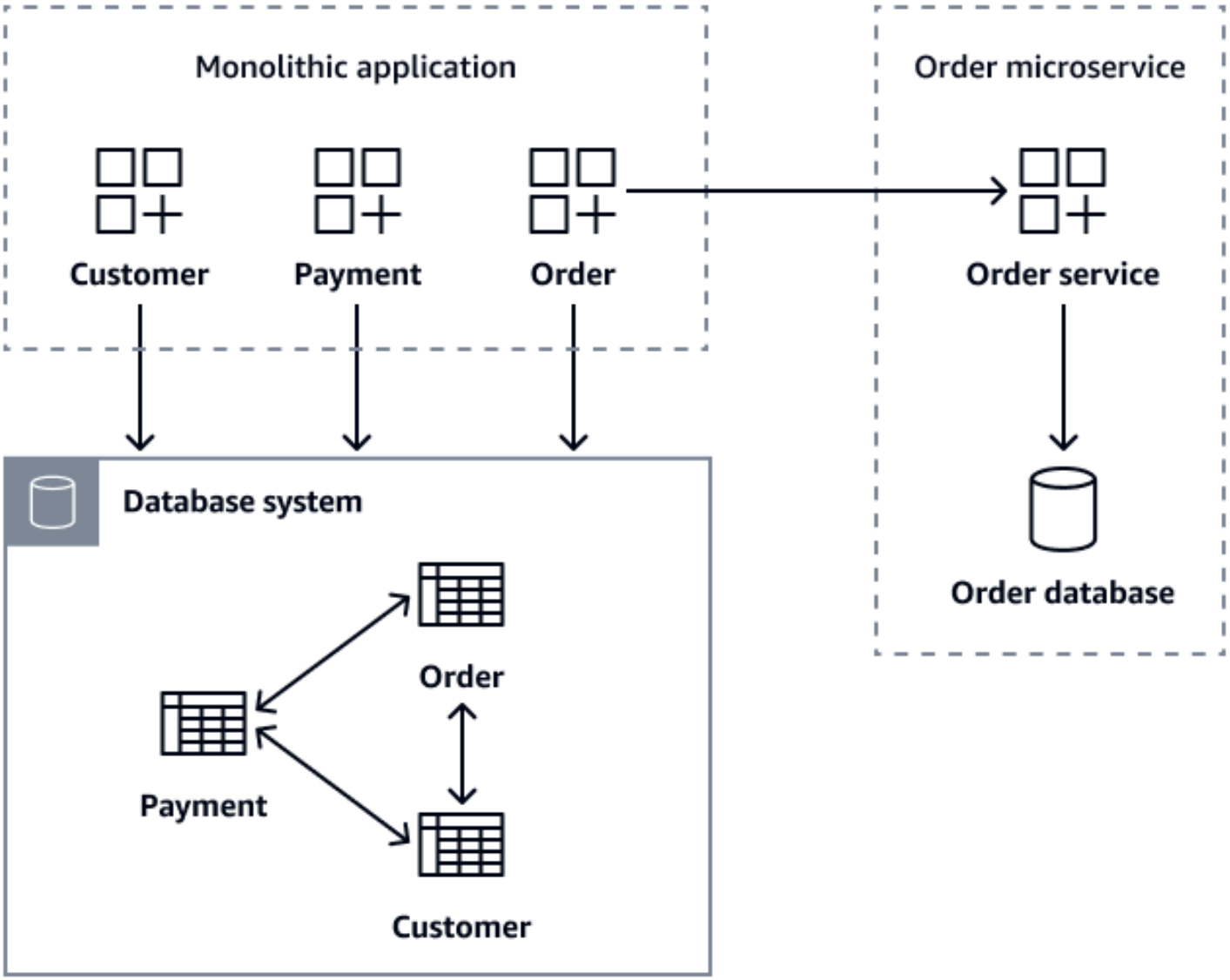
以下是如何使用非標準化方法的範例：

```
CREATE TABLE order (
    order_id INT PRIMARY KEY,
    customer_id INT,                    -- Reference only
    customer_first_name VARCHAR(100), -- Denormalized
    customer_last_name VARCHAR(100),  -- Denormalized
    customer_email VARCHAR(255),      -- Denormalized
    order_date TIMESTAMP,
    total_amount DECIMAL(10,2),
    status VARCHAR(50)
);
```

新Order資料表具有非標準化的客戶名稱和電子郵件地址。customer_id 會參考 ，而且Customer資料表沒有外部金鑰限制。以下是此非標準化方法的優點：

- Order 服務可以顯示具有客戶詳細資訊的訂單歷史記錄，而且不需要對Customer微服務進行 API 呼叫。
- 如果Customer服務關閉，Order服務仍可正常運作。
- 訂單處理和報告的查詢執行速度更快。

下圖顯示使用 getOrder(customer_id)、getOrder(order_id)getCustomerOders(customer_id)、 和 createOrder(Order order) API 呼叫微Order服務擷取訂單資料的單體應用程式。



在微服務遷移期間，您可以維護單體資料庫中的Order資料表作為轉換安全措施，以確保舊版應用程式保持正常運作。不過，所有與訂單相關的新操作都必須透過Order微服務 API 路由，此 API 會維護自己的資料庫，同時寫入舊版資料庫做為備份。此雙寫入模式提供安全網路。它允許漸進遷移，同時保

持系統穩定性。在所有客戶成功遷移到新的微服務後，您可以棄用單體資料庫中的舊版Order資料表。將單體應用程式及其資料庫分解為不同的CustomerOrder微服務之後，維護資料一致性會成為主要挑戰。

資料庫分解的最佳實務

分解單體資料庫時，組織必須建立明確的架構來追蹤進度、維護系統知識，以及解決新興的挑戰。本節提供測量分解成功、維護重要文件、實作持續改進程序，以及導覽常見挑戰的最佳實務。了解並遵循這些準則，可協助您確保資料庫分解工作可實現其預期效益，同時將營運中斷和技術負債降至最低。

本節包含下列主題：

- 衡量成功
- 文件需求
- 持續改進策略
- 克服資料庫分解中的常見挑戰

衡量成功

透過技術、營運和業務指標的混合來追蹤分解成功。在技術上，監控查詢回應時間、系統運作時間改善和部署頻率增加。以操作方式測量事件減少、問題解決速度和資源使用率改善。針對開發、追蹤功能實作速度、發行週期加速，以及減少跨團隊相依性。業務影響應可降低營運成本、加快time-to-market並改善客戶滿意度。這些指標通常在範圍階段期間定義。如需詳細資訊，請參閱本指南中的定義資料庫分解的範圍和需求。

文件需求

維護具有明確服務界限、資料流程和界面規格up-to-date系統架構文件。使用架構決策記錄 ADRs) 來擷取關鍵技術決策，包括其內容、後果和考慮的替代方案。例如，記錄為何特定服務先被分開，或如何進行特定資料一致性權衡。

排程每月架構審查，透過關鍵指標評估系統運作狀態：效能趨勢、安全合規和跨服務相依性。包含開發團隊針對整合挑戰和營運問題的意見回饋。此定期審查週期可協助您及早識別新興問題，並驗證分解工作是否與業務目標保持一致。

持續改進策略

將資料庫分解視為反覆程序，而非一次性專案。監控系統效能指標和服務互動，以識別最佳化機會。每季根據營運影響和維護成本，優先處理技術債務。例如，自動化經常執行的資料庫操作、增強監控涵蓋範圍，並根據學習的模式精簡部署程序。

克服資料庫分解中的常見挑戰

效能最佳化需要多面向的方法。在服務界限實作策略快取、根據實際用量最佳化查詢模式，並持續監控關鍵指標。透過分析趨勢並設定明確的介入閾值，主動解決效能瓶頸。

資料一致性挑戰需要謹慎的架構選擇。實作跨服務更新的事件驅動模式，並針對複雜交易使用 saga 協同運作模式。定義明確的服務界限，並在業務需求允許的情況下接受最終一致性。一致性和服務自主權之間的這種平衡對於成功分解至關重要。

卓越營運需要自動化 服務的例行任務和標準化程序。以清晰的提醒閾值維持全面監控，並投資於定期團隊培訓，以取得新的模式和工具。這種系統化的操作方法可提升可靠的服務交付，同時管理複雜性。

資料庫分解的常見問答集

這個全面的常見問答集區段解決了組織在進行資料庫分解專案時面臨的最常見問題和挑戰。從定義初始範圍和要求到遷移預存程序，這些問題提供實用的洞見和策略方法，以協助團隊成功導覽其資料庫現代化旅程。無論您是處於規劃階段或已經執行分解策略，這些答案都可協助您避免常見陷阱，並實作最佳實務以獲得最佳結果。

本節包含下列主題：

- 有關定義範圍和要求的FAQs
- 控制資料庫存取的FAQs
- 有關分析凝聚力和耦合的FAQs
- 將商業邏輯遷移至應用程式層的FAQs

有關定義範圍和要求的FAQs

本指南的 定義資料庫分解的範圍和需求章節討論如何分析互動、映射相依性，以及建立成功條件。此常見問答集區段說明有關建立和管理專案界限的關鍵問題。無論您是處理不清楚的技術限制、衝突的部門需求，還是不斷發展的業務需求，這些FAQs都會提供維持平衡方法的實際指導。

本節包含下列問題：

- 初始範圍定義應該有多詳細？
- 如果在啟動專案後發現其他相依性，該怎麼辦？
- 如何處理具有衝突需求之不同部門的利益相關者？
- 當文件不佳或過時，評估技術限制的最佳方式是什麼？
- 如何平衡即時業務需求與長期技術目標？
- 如何確保我不會遺漏靜音利益相關者的關鍵要求？
- 這些建議是否適用於單體大型主機資料庫？

初始範圍定義應該有多詳細？

從客戶的需求向後工作，以足夠的細節定義專案範圍，以識別系統界限和關鍵相依性，同時保持探索的靈活性。映射基本元素，包括系統界面、主要利益相關者和主要技術限制條件。從選取提供可測量值之系統的週框、低風險部分開始。此方法可協助團隊在處理更複雜的元件之前學習和調整策略。

記錄推動分解工作的關鍵業務需求，但避免在實作期間可能變更的過度指定詳細資訊。這種平衡方法可確保團隊可以清晰地向前邁進，同時保持適應現代化旅程中出現的新洞見和挑戰。

如果在啟動專案後發現其他相依性，該怎麼辦？

預期 會在專案進行時發現其他相依性。維護即時相依性日誌並進行定期範圍審查，以評估對時間軸和資源的影響。實作明確的變更管理程序，並在專案計畫中包含緩衝時間，以處理非預期的發現。目標是不阻止變更，而是有效管理變更。這有助於團隊快速調整，同時維持專案動能。

如何處理具有衝突需求之不同部門的利益相關者？

透過根據商業價值和系統影響的明確優先順序來處理衝突的部門需求。確保高階主管的贊助，以推動關鍵決策並快速解決衝突。安排定期利益相關者協調會議，以討論權衡和維護透明度。記錄所有決策及其理由，以促進明確的溝通並維持專案動能。專注於可量化商業利益的討論，而不是部門偏好。

當文件不佳或過時，評估技術限制的最佳方式是什麼？

面臨文件不良時，結合傳統分析與現代 AI 工具。使用大型語言模型 (LLMs) 來分析程式碼儲存庫、日誌和現有文件，以識別模式和潛在限制條件。採訪經驗豐富的開發人員和資料庫架構師，以驗證 AI 調查結果並發現未記錄的限制。部署具有增強 AI 功能的監控工具，以觀察系統行為並預測潛在問題。

建立小型技術實驗來驗證您的假設。您可以使用 AI 驅動的測試工具來加速程序。在知識庫中記錄可透過 AI 輔助更新持續增強的問題清單。考慮吸引複雜領域的主題專家，並使用 AI 配對程式設計工具來加速其分析和文件工作。

如何平衡即時業務需求與長期技術目標？

建立分階段專案藍圖，使立即業務需求與長期技術目標保持一致。識別儘早交付有形價值的快速成功，以便您可以建立利益相關者的信心。將分解分解分解為明確的里程碑。每個 都應該提供可衡量的商業利益，同時朝著架構目標邁進。透過定期藍圖審查和調整，保持靈活性以解決緊急業務需求。

如何確保我不會遺漏靜音利益相關者的關鍵要求？

映射整個組織的所有潛在利益相關者，包括下游系統擁有者和間接使用者。透過結構化面試、研討會和定期審查工作階段建立多個意見回饋管道。建置proof-of-concepts和原型，使需求切實並引發有意義的討論。例如，顯示系統相依性的簡單儀表板通常會顯示最初不明顯的隱藏利益相關者和需求。

與聲音和安靜的利益相關者進行定期驗證工作階段，並確保擷取所有觀點。關鍵洞見通常來自最接近日常操作的人員，而不是規劃會議中最響亮的聲音。

這些建議是否適用於單體大型主機資料庫？

本指南中所述的方法也適用於分解單體大型主機資料庫。這些資料庫的主要挑戰是管理來自各種利益相關者的需求。本指南中的技術建議可能適用於單體大型主機資料庫。如果大型主機具有關聯式資料庫，例如線上交易處理 (OLTP) 資料庫，則適用許多建議。對於線上分析處理 (OLAP) 資料庫，例如用於產生業務報告的資料庫，則僅適用部分建議。

控制資料庫存取的FAQs

本指南的 在分解期間控制資料庫存取區段討論使用資料庫包裝函式服務模式來控制資料庫存取。此常見問答集章節解決有關引入資料庫包裝函式服務的常見問題和問題，包括其對效能的潛在影響、處理現有的預存程序、管理複雜的交易，以及監督結構描述變更。

本節包含下列問題：

- 包裝函式服務不會成為新的瓶頸嗎？
- 現有預存程序會發生什麼情況？
- 如何在轉換期間管理結構描述變更？

包裝函式服務不會成為新的瓶頸嗎？

雖然資料庫包裝函式服務確實新增了額外的網路跳轉，但影響通常很小。您可以水平擴展服務，而受控存取的好處通常高於小型效能成本。將其視為效能和可維護性之間的暫時權衡。

現有預存程序會發生什麼情況？

一開始，資料庫包裝函式服務可以將預存程序公開為服務方法。隨著時間的推移，您可以逐步將邏輯移入應用程式層，從而改善測試和版本控制。逐步遷移商業邏輯以將風險降至最低。

如何在轉換期間管理結構描述變更？

透過包裝函式服務團隊集中化結構描述變更控制。此團隊負責維護所有消費者的全面可見性。此團隊會檢閱整個系統影響的提議變更、與受影響的團隊協調，並使用受控制的部署程序實作修改。例如，新增欄位時，此團隊應實作預設值或最初允許 null 來維持回溯相容性。

建立明確的變更管理程序，其中包含影響評估、測試要求和轉返程序。使用資料庫版本控制工具，並維護所有變更的清楚文件。這種集中式方法可防止結構描述修改中斷相依的服務，並維持系統穩定性。

有關分析凝聚力和耦合的FAQs

了解並有效分析資料庫耦合和凝聚力是成功分解資料庫的基礎。本指南的 分析資料庫分解的凝聚和耦合區段討論了耦合和凝聚。此常見問答集章節針對識別適當精細程度、選取正確的分析工具、記錄問題清單，以及排定耦合問題優先順序等重要問題。

本節包含下列問題：

- 在分析耦合時，如何識別適當層級的精細程度？
- 我可以使用哪些工具來分析資料庫耦合和凝聚？
- 記錄耦合和凝聚問題清單的最佳方式是什麼？
- 如何優先處理哪些耦合問題？
- 如何處理跨越多個操作的交易？

在分析耦合時，如何識別適當層級的精細程度？

從廣泛的資料庫關係分析開始，然後有系統地向下切入以識別自然分離點。使用資料庫分析工具來映射資料表層級關係、結構描述相依性和交易界限。例如，檢查 SQL 查詢中的聯結模式，以了解資料存取相依性。您也可以分析交易日誌，以識別業務流程界限。

專注於耦合自然最小的區域。這些通常與業務領域邊界一致，並代表最佳分解點。判斷適當的服務界限時，請考慮技術聯結（例如共用資料表和外部金鑰）和業務聯結（例如流程和報告需求）。

我可以使用哪些工具來分析資料庫耦合和凝聚？

您可以使用自動化工具和手動分析的組合來評估資料庫耦合和凝聚力。下列工具可協助您進行此評估：

- 結構描述視覺化工具 – 您可以使用 SchemaSpy或 等工具pgAdmin來產生 ER 圖表。這些圖表顯示資料表關係和潛在的耦合點。
- 查詢分析工具 – 您可以使用 pg_stat_statements或 SQL Server Query Store 來識別經常聯結的資料表和存取模式。
- 資料庫分析工具 – 例如 Oracle SQL Developer或 等工具，MySQL Workbench可提供查詢效能和資料相依性的洞見。
- 相依性映射工具 – AWS Schema Conversion Tool (AWS SCT) 可協助您視覺化結構描述關係，並識別緊密耦合的元件。 vFunction 可以透過分析應用程式的功能和網域邊界來協助您識別網域邊界。
- 交易監控工具 – 您可以使用資料庫特定的工具，例如 Oracle Enterprise Manager或 SQL Server Extended Events來分析交易界限。

- 商業邏輯遷移工具 – 您可以使用 Ispirer或生成式 AI 工具，例如 Amazon Q Developer 或 Kiro，將應用程式層的資料庫商業邏輯轉換為 Java。

將這些自動化分析與業務流程和網域知識的手動審查相結合，以完全了解系統耦合。這種多面向方法可確保在分解策略中同時考慮技術和業務觀點。

記錄耦合和凝聚問題清單的最佳方式是什麼？

建立可視覺化資料庫關係和使用模式的完整文件。以下是您可以用來記錄問題清單的資產類型：

- 相依性矩陣 – 映射資料表相依性並反白顯示高耦合區域。
- 關係圖 – 使用 ER 圖來顯示結構描述連線和外部金鑰關係。
- 資料表用量熱度貼圖 – 視覺化查詢頻率和資料表之間的資料存取模式。
- 交易流程圖 – 記錄多資料表交易及其界限。
- 網域界限映射 – 根據業務網域概述潛在的服務界限。

在文件中結合這些成品，並在分解進行時定期更新。對於圖表，您可以使用 draw.io或 等工具Lucidchart。請考慮實作 Wiki，以便於團隊存取和協作。這種多面向的文件方法提供清楚、共享的系統耦合和凝聚力的理解。

如何優先處理哪些耦合問題？

根據業務和技術因素的平衡評估，排定耦合問題的優先順序。針對業務影響（例如營收和客戶體驗）、技術風險（例如系統穩定性和資料完整性）、實作工作和團隊功能來評估每個問題。建立優先順序矩陣，從 1 到 5 分給這些維度的每個問題。此矩陣可協助您識別具有可管理風險的最有價值的機會。

從與現有團隊專業知識一致的高影響、低風險變更開始。這可協助您為更複雜的變更建立組織信心和動能。這種方法可促進逼真的執行，並最大化商業價值。定期審查和調整優先順序，以協助與不斷變化的業務需求和團隊容量保持一致。

如何處理跨越多個操作的交易？

透過精心設計的服務層級協調來處理多操作交易。實作複雜分散式交易的 saga 模式。將它們分成可獨立管理的較小、可逆步驟。例如，訂單處理流程可能會分割成單獨的步驟，用於清查、付款處理和建立訂單，每個步驟都有自己的補償機制。

可能的話，重新設計操作以增加原子，減少分散式交易的需求。當分散式交易無法避免時，請實作強大的追蹤和補償機制來提升資料一致性。監控交易完成率並實作明確的錯誤復原程序，以維護系統可靠性。

將商業邏輯遷移至應用程式層的FAQs

將商業邏輯從資料庫遷移至應用程式層，是資料庫現代化的關鍵和複雜層面。此商業邏輯遷移討論於本指南的 將商業邏輯從資料庫遷移至應用程式層一節。此常見問答集區段解決有關有效管理此轉換的常見問題，從選擇要遷移的初始候選者到處理複雜的預存程序和觸發程序。

本節包含下列問題：

- 如何識別要先遷移哪些預存程序？
- 將邏輯移至應用程式層有哪些風險？
- 從資料庫移開邏輯時，如何維持效能？
- 我應該如何處理涉及多個資料表的複雜預存程序？
- 如何在遷移期間處理資料庫觸發？
- 測試遷移商業邏輯的最佳方式是什麼？
- 如何管理資料庫和應用程式邏輯同時存在的轉換期間？
- 如何處理先前由資料庫管理之應用程式層中的錯誤案例？

如何識別要先遷移哪些預存程序？

首先，找出可提供低風險和高學習價值最佳組合的預存程序。專注於具有最小相依性、明確功能和非關鍵業務影響的程序。這些都是初始遷移的理想候選者，因為它們有助於團隊建立信心並建立模式。例如，選擇處理簡單資料操作的程序，而不是管理複雜交易或關鍵商業邏輯的程序。

使用資料庫監控工具來分析用量模式，並將不常存取的程序識別為早期候選項目。這種方法可將業務風險降至最低，同時提供寶貴的體驗，以便稍後處理更複雜的遷移。對每個程序的複雜性、業務關鍵性和相依性層級進行評分，以建立優先遷移序列。

將邏輯移至應用程式層有哪些風險？

將資料庫邏輯移至應用程式層會帶來幾項關鍵挑戰。系統效能可能會因為網路呼叫增加而降低，尤其是先前在資料庫中處理的資料密集型操作。交易管理變得越來越複雜，需要仔細的協調，以維護分散式操作的資料完整性。確保資料一致性變得具有挑戰性，尤其是先前依賴資料庫層級限制的操作。

遷移期間潛在的業務中斷以及開發人員的學習曲線也是重大問題。透過徹底規劃、在階段環境中進行廣泛測試，以及從較不關鍵的元件開始的逐步遷移，來減輕這些風險。實作強大的監控和復原程序，以快速識別和解決生產中的問題。

從資料庫移開邏輯時，如何維持效能？

針對經常存取的資料實作適當的快取機制、最佳化資料存取模式以將網路呼叫降至最低，以及對大量操作使用批次處理。對於non-time-critical操作，請考慮非同步處理，以改善系統回應能力。

密切監控應用程式效能指標，並視需要進行調校。例如，您可以使用大量處理取代多個單列操作、快取不常變更的參考資料，以及最佳化查詢模式以減少資料傳輸。定期效能測試和調校有助於系統維持可接受的回應時間，並改善可維護性和可擴展性。

我應該如何處理涉及多個資料表的複雜預存程序？

透過系統分解來接近複雜的多資料表預存程序。首先，將它們分成較小的邏輯一致性元件，並識別明確的交易界限和資料相依性。為每個邏輯元件建立服務介面。這可協助您逐步遷移，而不會中斷現有的功能。

從最不耦合的元件開始，step-by-step遷移。對於高度複雜的程序，請考慮在遷移更簡單的組件時暫時將其保留在資料庫中。當您朝架構目標邁進時，此混合方法可維持系統穩定性。在遷移期間持續監控效能和功能，並準備好根據結果調整您的策略。

如何在遷移期間處理資料庫觸發？

將資料庫觸發條件轉換為應用程式層級事件處理常式，同時維護系統功能。以非同步操作的訊息佇列的事件驅動模式取代同步觸發。請考慮對訊息佇列使用 Amazon Simple Notification Service (Amazon SNS) 或 Amazon Simple Queue Service (Amazon SQS)。如需稽核需求，請實作應用程式層級記錄或使用資料庫變更資料擷取 (CDC) 功能。

分析每個觸發的用途和重要性。有些觸發條件可能會由應用程式邏輯提供更好的服務，有些則可能需要事件來源模式來維持資料一致性。從簡單的觸發開始，例如稽核日誌，再處理管理業務規則或資料完整性的複雜觸發。在遷移期間仔細監控，以確保不會遺失功能或資料一致性。

測試遷移商業邏輯的最佳方式是什麼？

在部署遷移的商業邏輯之前，實作多層測試方法。從新應用程式程式碼的單元測試開始，然後新增涵蓋end-to-end業務流程的整合測試。平行執行舊實作和新實作，然後比較結果以驗證功能等效性。在各種負載條件下執行效能測試，以確認系統行為符合或超過先前的功能。

使用功能旗標來控制部署，以便在發生問題時快速復原。讓商業使用者參與驗證，特別是關鍵工作流程。在初始部署期間監控關鍵指標，並逐漸增加新實作的流量。在整個過程中，視需要維持還原至原始資料庫邏輯的能力。

如何管理資料庫和應用程式邏輯同時存在的轉換期間？

當資料庫和應用程式邏輯都使用中時，實作功能旗標來控制流量，並啟用舊實作和新實作之間的快速切換。維持嚴格的版本控制，並清楚記錄實作及其個別責任。設定兩個系統的全面監控，以快速識別任何差異或效能問題。

為每個遷移元件建立明確的轉返程序，以便您可以視需要還原至原始邏輯。定期與所有利益相關者溝通轉移狀態、潛在影響和呈報程序。此方法可協助您逐步遷移，同時維持系統穩定性和利益相關者的可信度。

如何處理先前由資料庫管理之應用程式層中的錯誤案例？

使用強大的應用程式層機制取代資料庫層級錯誤處理。實作斷路器並重試暫時性故障的邏輯。使用補償交易來維持分散式操作的資料一致性。例如，如果付款更新失敗，應用程式應該在定義的限制內自動重試，並視需要啟動補償動作。

設定全面的監控和提醒以快速識別問題，並維護詳細的稽核日誌以進行故障診斷。設計錯誤處理盡可能自動化，並為需要人工介入的案例定義明確的呈報路徑。這種多層方法提供系統彈性，同時維持資料完整性和業務流程持續性。

上的資料庫分解後續步驟 AWS

透過資料庫包裝函式服務實作初始資料庫分解策略，並將商業邏輯移至應用程式層之後，組織必須規劃下一次的演變。本節概述繼續現代化旅程的重要考量事項。

本節包含下列主題：

- 資料庫分解的增量策略
- 分散式資料庫環境的技術考量
- 支援分散式架構的組織變更

資料庫分解的增量策略

資料庫分解會遵循三個不同階段的逐步演變。團隊會先使用資料庫包裝函式服務包裝單體資料庫，以控制存取。然後，他們會開始將資料分割為服務特定的資料庫，同時維護主要資料庫以滿足舊版需求。最後，他們會完成遷移商業邏輯，以轉換到完全獨立的服務資料庫。

在整個旅程中，團隊必須實作謹慎的資料同步模式，並持續驗證服務之間的一致性。效能監控對於及早識別和解決潛在問題至關重要。隨著服務獨立發展，其結構描述應根據實際使用模式進行最佳化，而且您應該移除隨時間累積的備援結構。

這種增量方法有助於將風險降至最低，同時在整個轉型過程中維持系統穩定性。

分散式資料庫環境的技術考量

在分散式資料庫環境中，效能監控對於及早識別和解決瓶頸至關重要。團隊必須實作全面的監控系統和快取策略，以維持效能等級。讀取/寫入分割可以有效地平衡整個系統的負載。

資料一致性需要跨分散式服務仔細協調。團隊應在適當的情況下實作最終一致性模式，並建立明確的資料擁有權界限。強大的監控可提升所有服務的資料完整性。

此外，安全性必須演進以適應分散式架構。每個服務都需要精細的安全控制，而且您的存取模式需要定期檢閱。增強型監控和稽核在分散式環境中變得至關重要。

支援分散式架構的組織變更

團隊結構應與服務界限保持一致，以定義明確的擁有權和責任。組織必須建立新的通訊模式，並在團隊中建立其他技術功能。此結構應支援現有服務的維護，以及您持續的架構演變。

您必須更新操作程序，才能處理分散式架構。團隊必須修改部署程序、調整事件回應程序，並發展變更管理實務，以跨多個服務進行協調。

Resources

下列其他資源和工具可協助您的組織展開其資料庫分解之旅。

AWS 方案指引

- 將Oracle資料庫遷移至 AWS 雲端
- Oracle Database上的 Replatform 選項 AWS
- 雲端設計模式、架構和實作

AWS 部落格文章

- 將商業邏輯從資料庫遷移到應用程式，以提高創新速度和靈活性

AWS 服務

- AWS Application Migration Service
- AWS Database Migration Service (AWS DMS)
- Migration Evaluator
- AWS Schema Conversion Tool (AWS SCT)
- AWS Transform

其他工具

- AppEngine (Dynatrace 網站)
- Oracle Automatic Workload Repository (Oracle 網站)
- CAST Imaging (CAST 網站)
- Kiro (Kiro 網站)
- pgAdmin (pgAdmin 網站)
- pg_stat_statements (PostgreSQL 網站)
- SchemaSpy (SchemaSpy 網站)
- SQL Developer (Oracle 網站)

- SQLWays (Ispirer 網站)
- vFunction (vFunction 網站)

其他資源

- Monolith 到微服務 (O'Reilly 網站)

文件歷史紀錄

下表描述了本指南的重大變更。如果您想收到有關未來更新的通知，可以訂閱 RSS 摘要。

變更	描述	日期
大型主機常見問答集和 AI 工具	我們新增了這些建議是否適用於單體大型主機資料庫？ 常見問答集，我們新增了有關 AI 工具的其他資訊，您可以在資料庫分解期間使用。	2025 年 10 月 14 日
初次出版	—	2025 年 9 月 30 日

AWS 規範性指引詞彙表

以下是 AWS Prescriptive Guidance 提供的策略、指南和模式中常用的術語。若要建議項目，請使用詞彙表末尾的提供意見回饋連結。

數字

7 R

將應用程式移至雲端的七種常見遷移策略。這些策略以 Gartner 在 2011 年確定的 5 R 為基礎，包括以下內容：

- 重構/重新架構 – 充分利用雲端原生功能來移動應用程式並修改其架構，以提高敏捷性、效能和可擴展性。這通常涉及移植作業系統和資料庫。範例：將您的現場部署 Oracle 資料庫 遷移至 Amazon Aurora PostgreSQL 相容版本。
- 平台轉換 (隨即重塑) – 將應用程式移至雲端，並引入一定程度的優化以利用雲端功能。範例：將內部部署 Oracle 資料庫 遷移至 中的 Amazon Relational Database Service (Amazon RDS) for Oracle AWS 雲端。
- 重新購買 (捨棄再購買) – 切換至不同的產品，通常從傳統授權移至 SaaS 模型。範例：將您的客戶關係管理 (CRM) 系統遷移至 Salesforce.com。
- 主機轉換 (隨即轉移) – 將應用程式移至雲端，而不進行任何變更以利用雲端功能。範例：將您的現場部署 Oracle 資料庫 遷移至 中 EC2 執行個體上的 Oracle AWS 雲端。
- 重新放置 (虛擬機器監視器等級隨即轉移) – 將基礎設施移至雲端，無需購買新硬體、重寫應用程式或修改現有操作。您可以將伺服器從內部部署平台遷移到相同平台的雲端服務。範例：將 Microsoft Hyper-V應用程式 遷移至 AWS。
- 保留 (重新檢視) – 將應用程式保留在來源環境中。其中可能包括需要重要重構的應用程式，且您希望將該工作延遲到以後，以及您想要保留的舊版應用程式，因為沒有業務理由來進行遷移。
- 淘汰 – 解除委任或移除來源環境中不再需要的應用程式。

A

ABAC

請參閱屬性型存取控制。

抽象服務

請參閱 受管服務。

ACID

請參閱原子性、一致性、隔離性、耐久性。

主動-主動式遷移

一種資料庫遷移方法，其中來源和目標資料庫保持同步 (透過使用雙向複寫工具或雙重寫入操作)，且兩個資料庫都在遷移期間處理來自連接應用程式的交易。此方法支援小型、受控制批次的遷移，而不需要一次性切換。它更靈活，但需要比主動-被動遷移更多的工作。

主動-被動式遷移

一種資料庫遷移方法，其中來源和目標資料庫會保持同步，但只有來源資料庫會在資料複寫至目標資料庫時處理來自連線應用程式的交易。目標資料庫在遷移期間不接受任何交易。

彙總函數

在一組資料列上運作的 SQL 函數，會計算群組的單一傳回值。彙總函數的範例包括 SUM和 MAX。

AI

請參閱人工智慧。

AIOps

請參閱人工智慧操作。

匿名化

在資料集中永久刪除個人資訊的程序。匿名化有助於保護個人隱私權。匿名資料不再被視為個人資料。

反模式

經常用於經常性問題的解決方案，其中解決方案具有反生產力、無效或比替代解決方案更有效。

應用程式控制

一種安全方法，僅允許使用核准的應用程式，以協助保護系統免受惡意軟體攻擊。

應用程式組合

有關組織使用的每個應用程式的詳細資訊的集合，包括建置和維護應用程式的成本及其商業價值。此資訊是產品組合探索和分析程序的關鍵，有助於識別要遷移、現代化和優化的應用程式並排定其優先順序。

人工智慧 (AI)

電腦科學領域，致力於使用運算技術來執行通常與人類相關的認知功能，例如學習、解決問題和識別模式。如需詳細資訊，請參閱什麼是人工智慧？

人工智慧操作 (AIOps)

使用機器學習技術解決操作問題、減少操作事件和人工干預以及提高服務品質的程序。如需有關如何在 AWS 遷移策略中使用 AIOps 的詳細資訊，請參閱操作整合指南。

非對稱加密

一種加密演算法，它使用一對金鑰：一個用於加密的公有金鑰和一個用於解密的私有金鑰。您可以共用公有金鑰，因為它不用於解密，但對私有金鑰存取應受到高度限制。

原子性、一致性、隔離性、耐久性 (ACID)

一組軟體屬性，即使在出現錯誤、電源故障或其他問題的情況下，也能確保資料庫的資料有效性和操作可靠性。

屬性型存取控制 (ABAC)

根據使用者屬性 (例如部門、工作職責和團隊名稱) 建立精細許可的實務。如需詳細資訊，請參閱《 AWS Identity and Access Management (IAM) 文件》中的 ABAC for AWS。

授權資料來源

您存放主要版本資料的位置，被視為最可靠的資訊來源。您可以將授權資料來源中的資料複製到其他位置，以處理或修改資料，例如匿名、修訂或假名化資料。

可用區域

中的不同位置 AWS 區域 ，可隔離其他可用區域中的故障，並提供相同區域中其他可用區域的低成本、低延遲網路連線能力。

AWS 雲端採用架構 (AWS CAF)

的指導方針和最佳實務架構 AWS ，可協助組織制定高效且有效的計劃，以成功地移至雲端。 AWS CAF 將指導方針組織到六個重點領域：業務、人員、治理、平台、安全和營運。業務、人員和控管層面著重於業務技能和程序；平台、安全和操作層面著重於技術技能和程序。例如，人員層面針對處理人力資源 (HR)、人員配備功能和人員管理的利害關係人。為此， AWS CAF 為人員開發、訓練和通訊提供指引，協助組織做好成功採用雲端的準備。如需詳細資訊，請參閱 AWS CAF 網站和 AWS CAF 白皮書。

AWS 工作負載資格架構 (AWS WQF)

一種工具，可評估資料庫遷移工作負載、建議遷移策略，並提供工作預估值。 AWS WQF 隨附於 AWS Schema Conversion Tool (AWS SCT)。它會分析資料庫結構描述和程式碼物件、應用程式程式碼、相依性和效能特性，並提供評估報告。

B

錯誤的機器人

旨在中斷或傷害個人或組織的機器人。

BCP

請參閱業務持續性規劃。

行為圖

資源行為的統一互動式檢視，以及一段時間後的互動。您可以將行為圖與 Amazon Detective 搭配使用來檢查失敗的登入嘗試、可疑的 API 呼叫和類似動作。如需詳細資訊，請參閱偵測文件中的行為圖中的資料。

大端序系統

首先儲存最高有效位元組的系統。另請參閱 Endianness。

二進制分類

預測二進制結果的過程 (兩個可能的類別之一)。例如，ML 模型可能需要預測諸如「此電子郵件是否是垃圾郵件？」等問題 或「產品是書還是汽車？」

Bloom 篩選條件

一種機率性、記憶體高效的資料結構，用於測試元素是否為集的成員。

藍/綠部署

一種部署策略，您可以在其中建立兩個不同但相同的環境。您可以在一個環境 （藍色） 中執行目前的應用程式版本，並在另一個環境 （綠色） 中執行新的應用程式版本。此策略可協助您快速復原，並將影響降至最低。

機器人

透過網際網路執行自動化任務並模擬人類活動或互動的軟體應用程式。有些機器人有用或有益，例如在網際網路上編製資訊索引的 Web 爬蟲程式。某些其他機器人稱為惡意機器人，旨在中斷或傷害個人或組織。

殭屍網路

受到惡意軟體感染且受單一方控制之機器人的網路，稱為機器人繼承器或機器人運算子。殭屍網路是擴展機器人及其影響的最佳已知機制。

分支

程式碼儲存庫包含的區域。儲存庫中建立的第一個分支是主要分支。您可以從現有分支建立新分支，然後在新分支中開發功能或修正錯誤。您建立用來建立功能的分支通常稱為功能分支。當準備好發佈功能時，可以將功能分支合併回主要分支。如需詳細資訊，請參閱關於分支 (GitHub 文件)。

碎片存取

在特殊情況下，以及透過核准的程序，讓使用者快速取得他們通常無權存取 AWS 帳戶 之 的存取權。如需詳細資訊，請參閱 Well-Architected 指南中的 AWS 實作打破玻璃程序指標。

棕地策略

環境中的現有基礎設施。對系統架構採用棕地策略時，可以根據目前系統和基礎設施的限制來設計架構。如果正在擴展現有基礎設施，則可能會混合棕地和綠地策略。

緩衝快取

儲存最常存取資料的記憶體區域。

業務能力

業務如何創造價值 (例如，銷售、客戶服務或營銷)。業務能力可驅動微服務架構和開發決策。如需詳細資訊，請參閱在 AWS上執行容器化微服務白皮書的圍繞業務能力進行組織部分。

業務連續性規劃 (BCP)

一種解決破壞性事件 (如大規模遷移) 對營運的潛在影響並使業務能夠快速恢復營運的計畫。

C

CAF

請參閱AWS 雲端採用架構。

Canary 部署

版本對最終使用者的緩慢和增量版本。當您有信心時，您可以部署新版本並完全取代目前的版本。

CCoE

請參閱 Cloud Center of Excellence。

CDC

請參閱變更資料擷取。

變更資料擷取 (CDC)

追蹤對資料來源 (例如資料庫表格) 的變更並記錄有關變更的中繼資料的程序。您可以將 CDC 用於各種用途，例如稽核或複寫目標系統中的變更以保持同步。

混沌工程

故意引入故障或破壞性事件，以測試系統的彈性。您可以使用 AWS Fault Injection Service (AWS FIS) 來執行實驗，為您的 AWS 工作負載帶來壓力，並評估其回應。

CI/CD

請參閱持續整合和持續交付。

分類

有助於產生預測的分類程序。用於分類問題的 ML 模型可預測離散值。離散值永遠彼此不同。例如，模型可能需要評估影像中是否有汽車。

用戶端加密

在目標 AWS 服務 接收資料之前，在本機加密資料。

雲端卓越中心 (CCoE)

一個多學科團隊，可推動整個組織的雲端採用工作，包括開發雲端最佳實務、調動資源、制定遷移時間表以及領導組織進行大規模轉型。如需詳細資訊，請參閱 AWS 雲端 企業策略部落格上的 CCoE 文章。

雲端運算

通常用於遠端資料儲存和 IoT 裝置管理的雲端技術。雲端運算通常連接到邊緣運算技術。

雲端操作模型

在 IT 組織中，用於建置、成熟和最佳化一或多個雲端環境的操作模型。如需詳細資訊，請參閱建置您的雲端操作模型。

採用雲端階段

組織在遷移至 時通常會經歷的四個階段 AWS 雲端：

- 專案 – 執行一些與雲端相關的專案以進行概念驗證和學習用途
- 基礎 – 進行基礎投資以擴展雲端採用 (例如，建立登陸區域、定義 CCoE、建立營運模型)

- 遷移 – 遷移個別應用程式
- 重塑 – 優化產品和服務，並在雲端中創新

這些階段由 Stephen Orban 在部落格文章 The Journey Toward Cloud-First 和企業策略部落格上的採用階段中定義。 AWS 雲端 如需有關它們如何與 AWS 遷移策略關聯的資訊，請參閱遷移整備指南。

CMDB

請參閱組態管理資料庫。

程式碼儲存庫

透過版本控制程序來儲存及更新原始程式碼和其他資產 (例如文件、範例和指令碼) 的位置。常見的雲端儲存庫包括 GitHub或 Bitbucket Cloud。程式碼的每個版本都稱為分支。在微服務結構中，每個儲存庫都專用於單個功能。單一 CI/CD 管道可以使用多個儲存庫。

冷快取

一種緩衝快取，它是空的、未填充的，或者包含過時或不相關的資料。這會影響效能，因為資料庫執行個體必須從主記憶體或磁碟讀取，這比從緩衝快取讀取更慢。

冷資料

很少存取且通常是歷史資料的資料。查詢這類資料時，通常可接受慢查詢。將此資料移至效能較低且成本較低的儲存層或類別，可以降低成本。

電腦視覺 (CV)

使用機器學習從數位影像和影片等視覺化格式分析和擷取資訊的 AI 欄位。例如，Amazon SageMaker AI 提供 CV 的影像處理演算法。

組態偏離

對於工作負載，組態會從預期狀態變更。這可能會導致工作負載變得不合規，而且通常是漸進和無意的。

組態管理資料庫 (CMDB)

儲存和管理有關資料庫及其 IT 環境的資訊的儲存庫，同時包括硬體和軟體元件及其組態。您通常在遷移的產品組合探索和分析階段使用 CMDB 中的資料。

一致性套件

您可以組合的 AWS Config 規則和修補動作集合，以自訂您的合規和安全檢查。您可以使用 YAML 範本，將一致性套件部署為 AWS 帳戶 和 區域中或整個組織的單一實體。如需詳細資訊，請參閱 AWS Config 文件中的一致性套件。

持續整合和持續交付 (CI/CD)

自動化軟體發行程序的來源、建置、測試、暫存和生產階段的程序。CI/CD 通常被描述為管道。CI/CD 可協助您將程序自動化、提升生產力、改善程式碼品質以及加快交付速度。如需詳細資訊，請參閱持續交付的優點。CD 也可表示持續部署。如需詳細資訊，請參閱持續交付與持續部署。

CV

請參閱電腦視覺。

D

靜態資料

網路中靜止的資料，例如儲存中的資料。

資料分類

根據重要性和敏感性來識別和分類網路資料的程序。它是所有網路安全風險管理策略的關鍵組成部分，因為它可以協助您確定適當的資料保護和保留控制。資料分類是 AWS Well-Architected Framework 中安全支柱的元件。如需詳細資訊，請參閱資料分類。

資料偏離

生產資料與用於訓練 ML 模型的資料之間有意義的變化，或輸入資料隨時間有意義的變更。資料偏離可以降低 ML 模型預測的整體品質、準確性和公平性。

傳輸中的資料

在您的網路中主動移動的資料，例如在網路資源之間移動。

資料網格

架構架構，提供分散式、分散式資料擁有權與集中式管理。

資料最小化

僅收集和處理嚴格必要資料的原則。在 中實作資料最小化 AWS 雲端 可以降低隱私權風險、成本和分析碳足跡。

資料周邊

AWS 環境中的一組預防性防護機制，可協助確保只有信任的身分才能從預期的網路存取信任的資源。如需詳細資訊，請參閱在 上建置資料周邊 AWS。

資料預先處理

將原始資料轉換成 ML 模型可輕鬆剖析的格式。預處理資料可能意味著移除某些欄或列，並解決遺失、不一致或重複的值。

資料來源

在整個生命週期中追蹤資料的原始伺服器和歷史記錄的程序，例如資料的產生、傳輸和儲存方式。

資料主體

正在收集和處理資料的個人。

資料倉儲

支援商業智慧的資料管理系統，例如 分析。資料倉儲通常包含大量歷史資料，通常用於查詢和分析。

資料庫定義語言 (DDL)

用於建立或修改資料庫中資料表和物件之結構的陳述式或命令。

資料庫處理語言 (DML)

用於修改 (插入、更新和刪除) 資料庫中資訊的陳述式或命令。

DDL

請參閱資料庫定義語言。

深度整體

結合多個深度學習模型進行預測。可以使用深度整體來獲得更準確的預測或估計預測中的不確定性。

深度學習

一個機器學習子領域，它使用多層人工神經網路來識別感興趣的輸入資料與目標變數之間的對應關係。

深度防禦

這是一種資訊安全方法，其中一系列的安全機制和控制項會在整個電腦網路中精心分層，以保護網路和其中資料的機密性、完整性和可用性。當您在 上採用此策略時 AWS，您可以在 AWS Organizations 結構的不同層新增多個控制項，以協助保護資源。例如，defense-in-depth方法可能會結合多重驗證、網路分割和加密。

委派的管理員

在 中 AWS Organizations，相容的服務可以註冊 AWS 成員帳戶，以管理組織的帳戶和管理該服務的許可。此帳戶稱為該服務的委派管理員。如需詳細資訊和相容服務清單，請參閱 AWS Organizations 文件中的可搭配 AWS Organizations運作的服務。

deployment

在目標環境中提供應用程式、新功能或程式碼修正的程序。部署涉及在程式碼庫中實作變更，然後在應用程式環境中建置和執行該程式碼庫。

開發環境

請參閱 環境。

偵測性控制

一種安全控制，用於在事件發生後偵測、記錄和提醒。這些控制是第二道防線，提醒您注意繞過現有預防性控制的安全事件。如需詳細資訊，請參閱在 AWS上實作安全控制中的偵測性控制。

開發值串流映射 (DVSM)

一種程序，用於識別並優先考慮對軟體開發生命週期中的速度和品質造成負面影響的限制。DVSM 延伸了原本專為精簡製造實務設計的價值串流映射程序。它著重於透過軟體開發程序建立和移動價值所需的步驟和團隊。

數位分身

真實世界系統的虛擬呈現，例如建築物、工廠、工業設備或生產線。數位分身支援預測性維護、遠端監控和生產最佳化。

維度資料表

在星星結構描述中，較小的資料表包含有關事實資料表中量化資料的資料屬性。維度資料表屬性通常是文字欄位或離散數字，其行為與文字相似。這些屬性通常用於查詢限制、篩選和結果集標記。

災難

防止工作負載或系統在其主要部署位置實現其業務目標的事件。這些事件可能是自然災難、技術故障或人為動作的結果，例如意外設定錯誤或惡意軟體攻擊。

災難復原 (DR)

您用來將災難造成的停機時間和資料遺失降至最低的策略和程序。如需詳細資訊，請參閱 AWS Well-Architected Framework 中的 上工作負載的災難復原 AWS：雲端中的復原。

DML

請參閱資料庫處理語言。

領域驅動的設計

一種開發複雜軟體系統的方法，它會將其元件與每個元件所服務的不斷發展的領域或核心業務目標相關聯。Eric Evans 在其著作 Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston: Addison-Wesley Professional, 2003) 中介紹了這一概念。如需有關如何將領域驅動的設計與 strangler fig 模式搭配使用的資訊，請參閱使用容器和 Amazon API Gateway 逐步現代化舊版 Microsoft ASP.NET (ASMX) Web 服務。

DR

請參閱災難復原。

偏離偵測

追蹤與基準組態的偏差。例如，您可以使用 AWS CloudFormation 來偵測系統資源中的偏離，也可以使用 AWS Control Tower 來偵測登陸區域中可能影響控管要求合規性的變更。 https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-stack-drift.html

DVSM

請參閱開發值串流映射。

E

EDA

請參閱探索性資料分析。

EDI

請參閱電子資料交換。

邊緣運算

提升 IoT 網路邊緣智慧型裝置運算能力的技術。與雲端運算相比，邊緣運算可以減少通訊延遲並改善回應時間。

電子資料交換 (EDI)

在組織之間自動交換商業文件。如需詳細資訊，請參閱什麼是電子資料交換。

加密

將人類可讀取的純文字資料轉換為加密文字的運算程序。

加密金鑰

由加密演算法產生的隨機位元的加密字串。金鑰長度可能有所不同，每個金鑰的設計都是不可預測且唯一的。

端序

位元組在電腦記憶體中的儲存順序。大端序系統首先儲存最高有效位元組。小端序系統首先儲存最低有效位元組。

端點

請參閱 服務端點。

端點服務

您可以在虛擬私有雲端 (VPC) 中託管以與其他使用者共用的服務。您可以使用 建立端點服務，AWS PrivateLink 並將許可授予其他 AWS 帳戶 或 AWS Identity and Access Management (IAM) 委託人。這些帳戶或主體可以透過建立介面 VPC 端點私下連接至您的端點服務。如需詳細資訊，請參閱 Amazon Virtual Private Cloud (Amazon VPC) 文件中的建立端點服務。

企業資源規劃 (ERP)

一種系統，可自動化和管理企業的關鍵業務流程 （例如會計、MES 和專案管理）。

信封加密

使用另一個加密金鑰對某個加密金鑰進行加密的程序。如需詳細資訊，請參閱 AWS Key Management Service (AWS KMS) 文件中的信封加密。

環境

執行中應用程式的執行個體。以下是雲端運算中常見的環境類型：

- 開發環境 – 執行中應用程式的執行個體，只有負責維護應用程式的核心團隊才能使用。開發環境用來測試變更，然後再將開發環境提升到較高的環境。此類型的環境有時稱為測試環境。
- 較低的環境 – 應用程式的所有開發環境，例如用於初始建置和測試的開發環境。
- 生產環境 – 最終使用者可以存取的執行中應用程式的執行個體。在 CI/CD 管道中，生產環境是最後一個部署環境。
- 較高的環境 – 核心開發團隊以外的使用者可存取的所有環境。這可能包括生產環境、生產前環境以及用於使用者接受度測試的環境。

epic

在敏捷方法中，有助於組織工作並排定工作優先順序的功能類別。epic 提供要求和實作任務的高層級描述。例如， AWS CAF 安全概念包括身分和存取管理、偵測控制、基礎設施安全、資料保護和事件回應。如需有關 AWS 遷移策略中的 Epic 的詳細資訊，請參閱計畫實作指南。

ERP

請參閱企業資源規劃。

探索性資料分析 (EDA)

分析資料集以了解其主要特性的過程。您收集或彙總資料，然後執行初步調查以尋找模式、偵測異常並檢查假設。透過計算摘要統計並建立資料可視化來執行 EDA。

F

事實資料表

星狀結構描述中的中央資料表。它存放有關業務操作的量化資料。一般而言，事實資料表包含兩種類型的資料欄：包含度量的資料，以及包含維度資料表外部索引鍵的資料欄。

快速失敗

一種使用頻繁和增量測試來縮短開發生命週期的理念。這是敏捷方法的關鍵部分。

故障隔離界限

在 中 AWS 雲端，像是可用區域 AWS 區域、控制平面或資料平面等邊界會限制故障的影響，並有助於改善工作負載的彈性。如需詳細資訊，請參閱AWS 故障隔離界限。

功能分支

請參閱分支。

特徵

用來進行預測的輸入資料。例如，在製造環境中，特徵可能是定期從製造生產線擷取的影像。

功能重要性

特徵對於模型的預測有多重要。這通常表示為可以透過各種技術來計算的數值得分，例如 Shapley Additive Explanations (SHAP) 和積分梯度。如需詳細資訊，請參閱 的機器學習模型可解譯性 AWS。

特徵轉換

優化 ML 程序的資料，包括使用其他來源豐富資料、調整值、或從單一資料欄位擷取多組資訊。這可讓 ML 模型從資料中受益。例如，如果將「2021-05-27 00:15:37」日期劃分為「2021」、「五月」、「週四」和「15」，則可以協助學習演算法學習與不同資料元件相關聯的細微模式。

少量擷取提示

在要求 LLM 執行類似的任務之前，提供少量示範任務和所需輸出的範例。此技術是內容內學習的應用程式，其中模型會從內嵌在提示中的範例 (快照) 中學習。對於需要特定格式、推理或網域知識的任務，少量的提示非常有效。另請參閱零鏡頭提示。

FGAC

請參閱精細存取控制。

精細存取控制 (FGAC)

使用多個條件來允許或拒絕存取請求。

閃切遷移

一種資料庫遷移方法，透過變更資料擷取使用連續資料複寫，以盡可能在最短的時間內遷移資料，而不是使用分階段方法。目標是將停機時間降至最低。

FM

請參閱基礎模型。

基礎模型 (FM)

大型深度學習神經網路，已針對廣義和未標記資料的大量資料集進行訓練。FMs 能夠執行各種一般任務，例如了解語言、產生文字和影像，以及以自然語言交談。如需詳細資訊，請參閱什麼是基礎模型。

G

生成式 AI

已針對大量資料進行訓練的 AI 模型子集，可使用簡單的文字提示建立新的內容和成品，例如影像、影片、文字和音訊。如需詳細資訊，請參閱什麼是生成式 AI。

地理封鎖

請參閱地理限制。

地理限制 (地理封鎖)

Amazon CloudFront 中的選項，可防止特定國家/地區的使用者存取內容分發。您可以使用允許清單或封鎖清單來指定核准和禁止的國家/地區。如需詳細資訊，請參閱 CloudFront 文件中的限制內容的地理分佈。

Gitflow 工作流程

這是一種方法，其中較低和較高環境在原始碼儲存庫中使用不同分支。Gitflow 工作流程被視為舊版，而以幹線為基礎的工作流程是現代、偏好的方法。

黃金影像

系統或軟體的快照，做為部署該系統或軟體新執行個體的範本。例如，在製造中，黃金映像可用於在多個裝置上佈建軟體，並有助於提高裝置製造操作的速度、可擴展性和生產力。

綠地策略

新環境中缺乏現有基礎設施。對系統架構採用綠地策略時，可以選擇所有新技術，而不會限制與現有基礎設施的相容性，也稱為棕地。如果正在擴展現有基礎設施，則可能會混合棕地和綠地策略。

防護機制

有助於跨組織單位 (OU) 來管控資源、政策和合規的高層級規則。預防性防護機制會強制執行政策，以確保符合合規標準。透過使用服務控制政策和 IAM 許可界限來將其實作。偵測性防護機制可偵測政策違規和合規問題，並產生提醒以便修正。它們是透過使用 AWS Config AWS Security Hub CSPM、Amazon GuardDuty、、Amazon Inspector AWS Trusted Advisor和自訂 AWS Lambda 檢查來實作。

H

HA

請參閱高可用性。

異質資料庫遷移

將來源資料庫遷移至使用不同資料庫引擎的目標資料庫 (例如，Oracle 至 Amazon Aurora)。異質遷移通常是重新架構工作的一部分，而轉換結構描述可能是一項複雜任務。AWS 提供有助於結構描述轉換的 AWS SCT。

高可用性 (HA)

在遇到挑戰或災難時，工作負載能夠在不介入的情況下持續運作。HA 系統的設計目的是自動容錯移轉、持續提供高品質的效能，以及處理不同的負載和故障，並將效能影響降至最低。

歷史現代化

一種方法，用於現代化和升級操作技術 (OT) 系統，以更好地滿足製造業的需求。歷史資料是一種資料庫，用於從工廠中的各種來源收集和存放資料。

保留資料

從用於訓練機器學習模型的資料集中保留的部分歷史標記資料。您可以使用保留資料，透過比較模型預測與保留資料來評估模型效能。

異質資料庫遷移

將您的來源資料庫遷移至共用相同資料庫引擎的目標資料庫 (例如，Microsoft SQL Server 至 Amazon RDS for SQL Server)。同質遷移通常是主機轉換或平台轉換工作的一部分。您可以使用原生資料庫公用程式來遷移結構描述。

熱資料

經常存取的資料，例如即時資料或最近的轉譯資料。此資料通常需要高效能儲存層或類別，才能提供快速的查詢回應。

修補程序

緊急修正生產環境中的關鍵問題。由於其緊迫性，通常會在典型 DevOps 發行工作流程之外執行修補程式。

超級護理期間

在切換後，遷移團隊在雲端管理和監控遷移的應用程式以解決任何問題的時段。通常，此期間的長度為 1-4 天。在超級護理期間結束時，遷移團隊通常會將應用程式的責任轉移給雲端營運團隊。

I

IaC

請參閱基礎設施即程式碼。

身分型政策

連接至一或多個 IAM 主體的政策，可定義其在 AWS 雲端 環境中的許可。

閒置應用程式

90 天期間 CPU 和記憶體平均使用率在 5% 至 20% 之間的應用程式。在遷移專案中，通常會淘汰這些應用程式或將其保留在內部部署。

IIoT

請參閱工業物聯網。

不可變的基礎設施

為生產工作負載部署新基礎設施的模型，而不是更新、修補或修改現有的基礎設施。不可變基礎設施本質上比可變基礎設施更一致、可靠且可預測。如需詳細資訊，請參閱 AWS Well-Architected Framework 中的使用不可變基礎設施部署最佳實務。

傳入 (輸入) VPC

在 AWS 多帳戶架構中，接受、檢查和路由來自應用程式外部之網路連線的 VPC。AWS 安全參考架構建議您使用傳入、傳出和檢查 VPC 來設定網路帳戶，以保護應用程式與更廣泛的網際網路之間的雙向介面。

增量遷移

一種切換策略，您可以在其中將應用程式分成小部分遷移，而不是執行單一、完整的切換。例如，您最初可能只將一些微服務或使用者移至新系統。確認所有項目都正常運作之後，您可以逐步移動其他微服務或使用者，直到可以解除委任舊式系統。此策略可降低與大型遷移關聯的風險。

工業 4.0

2016 年 Klaus Schwab 推出的術語，透過連線能力、即時資料、自動化、分析和 AI/ML 的進展，指製造程序的現代化。

基礎設施

應用程式環境中包含的所有資源和資產。

基礎設施即程式碼 (IaC)

透過一組組態檔案來佈建和管理應用程式基礎設施的程序。IaC 旨在協助您集中管理基礎設施，標準化資源並快速擴展，以便新環境可重複、可靠且一致。

工業物聯網 (IIoT)

在製造業、能源、汽車、醫療保健、生命科學和農業等產業領域使用網際網路連線的感測器和裝置。如需詳細資訊，請參閱建立工業物聯網 (IIoT) 數位轉型策略。

檢查 VPC

在 AWS 多帳戶架構中，集中式 VPC 可管理 VPCs 之間 （在相同或不同的 中 AWS 區域)、網際網路和內部部署網路之間的網路流量檢查。AWS 安全參考架構建議您使用傳入、傳出和檢查 VPC 來設定網路帳戶，以保護應用程式與更廣泛的網際網路之間的雙向介面。

物聯網 (IoT)

具有內嵌式感測器或處理器的相連實體物體網路，其透過網際網路或本地通訊網路與其他裝置和系統進行通訊。如需詳細資訊，請參閱什麼是 IoT？

可解釋性

機器學習模型的一個特徵，描述了人類能夠理解模型的預測如何依賴於其輸入的程度。如需詳細資訊，請參閱 的機器學習模型可解譯性 AWS。

IoT

請參閱物聯網。

IT 資訊庫 (ITIL)

一組用於交付 IT 服務並使這些服務與業務需求保持一致的最佳實務。ITIL 為 ITSM 提供了基礎。

IT 服務管理 (ITSM)

與組織的設計、實作、管理和支援 IT 服務關聯的活動。如需有關將雲端操作與 ITSM 工具整合的資訊，請參閱操作整合指南。

ITIL

請參閱 IT 資訊庫。

ITSM

請參閱 IT 服務管理。

L

標籤型存取控制 (LBAC)

強制存取控制 (MAC) 的實作，其中使用者和資料本身都會獲得明確指派的安全標籤值。使用者安全標籤和資料安全標籤之間的交集會決定使用者可以看到哪些資料列和資料欄。

登陸區域

登陸區域是架構良好的多帳戶 AWS 環境，可擴展且安全。這是一個起點，您的組織可以從此起點快速啟動和部署工作負載與應用程式，並對其安全和基礎設施環境充滿信心。如需有關登陸區域的詳細資訊，請參閱設定安全且可擴展的多帳戶 AWS 環境。

大型語言模型 (LLM)

預先訓練大量資料的深度學習 AI 模型。LLM 可以執行多個任務，例如回答問題、摘要文件、將文字翻譯成其他語言，以及完成句子。如需詳細資訊，請參閱什麼是 LLMs。

大型遷移

遷移 300 部或更多伺服器。

LBAC

請參閱標籤型存取控制。

最低權限

授予執行任務所需之最低許可的安全最佳實務。如需詳細資訊，請參閱 IAM 文件中的套用最低權限許可。

隨即轉移

請參閱 7 個 R。

小端序系統

首先儲存最低有效位元組的系統。另請參閱 Endianness。

LLM

請參閱大型語言模型。

較低的環境

請參閱 環境。

M

機器學習 (ML)

一種使用演算法和技術進行模式識別和學習的人工智慧。機器學習會進行分析並從記錄的資料 (例如物聯網 (IoT) 資料) 中學習，以根據模式產生統計模型。如需詳細資訊，請參閱機器學習。

主要分支

請參閱分支。

惡意軟體

旨在危及電腦安全或隱私權的軟體。惡意軟體可能會中斷電腦系統、洩露敏感資訊，或取得未經授權的存取。惡意軟體的範例包括病毒、蠕蟲、勒索軟體、特洛伊木馬程式、間諜軟體和鍵盤記錄器。

受管服務

AWS 服務 會 AWS 操作基礎設施層、作業系統和平台，而您會存取端點來存放和擷取資料。Amazon Simple Storage Service (Amazon S3) 和 Amazon DynamoDB 是受管服務的範例。這些也稱為抽象服務。

製造執行系統 (MES)

一種軟體系統，用於追蹤、監控、記錄和控制生產程序，將原物料轉換為現場成品。

MAP

請參閱遷移加速計劃。

機制

建立工具、推動工具採用，然後檢查結果以進行調整的完整程序。機制是在操作時強化和改善自身的循環。如需詳細資訊，請參閱 AWS Well-Architected Framework 中的建置機制。

成員帳戶

除了屬於組織一部分的管理帳戶 AWS 帳戶 之外的所有 AWS Organizations。帳戶一次只能是一個組織的成員。

製造執行系統

請參閱製造執行系統。

訊息佇列遙測傳輸 (MQTT)

根據發佈/訂閱模式的輕量型machine-to-machine(M2M) 通訊協定，適用於資源受限的 IoT 裝置。

微服務

一種小型的獨立服務，它可透過定義明確的 API 進行通訊，通常由小型獨立團隊擁有。例如，保險系統可能包含對應至業務能力 (例如銷售或行銷) 或子領域 (例如購買、索賠或分析) 的微服務。微服務的優點包括靈活性、彈性擴展、輕鬆部署、可重複使用的程式碼和適應力。如需詳細資訊，請參閱使用無 AWS 伺服器服務整合微服務。

微服務架構

一種使用獨立元件來建置應用程式的方法，這些元件會以微服務形式執行每個應用程式程序。這些微服務會使用輕量型 API，透過明確定義的介面進行通訊。此架構中的每個微服務都可以進行更新、部署和擴展，以滿足應用程式特定功能的需求。如需詳細資訊，請參閱在 上實作微服務 AWS。

Migration Acceleration Program (MAP)

一種 AWS 計畫，提供諮詢支援、訓練和服務，協助組織建立強大的營運基礎，以移至雲端，並協助抵銷遷移的初始成本。MAP 包括用於有條不紊地執行舊式遷移的遷移方法以及一組用於自動化和加速常見遷移案例的工具。

大規模遷移

將大部分應用程式組合依波次移至雲端的程序，在每個波次中，都會以更快的速度移動更多應用程式。此階段使用從早期階段學到的最佳實務和經驗教訓來實作團隊、工具和流程的遷移工廠，以透過自動化和敏捷交付簡化工作負載的遷移。這是 AWS 遷移策略的第三階段。

遷移工廠

可透過自動化、敏捷的方法簡化工作負載遷移的跨職能團隊。遷移工廠團隊通常包括營運、業務分析師和擁有者、遷移工程師、開發人員以及從事 Sprint 工作的 DevOps 專業人員。20% 至 50% 之間的企業應用程式組合包含可透過工廠方法優化的重複模式。如需詳細資訊，請參閱此內容集中的遷移工廠的討論和雲端遷移工廠指南。

遷移中繼資料

有關完成遷移所需的應用程式和伺服器的資訊。每種遷移模式都需要一組不同的遷移中繼資料。遷移中繼資料的範例包括目標子網路、安全群組和 AWS 帳戶。

遷移模式

可重複的遷移任務，詳細描述遷移策略、遷移目的地以及所使用的遷移應用程式或服務。範例：使用 AWS Application Migration Service 重新託管遷移至 Amazon EC2。

遷移組合評定 (MPA)

線上工具，提供驗證商業案例以遷移至 的資訊 AWS 雲端。MPA 提供詳細的組合評定 (伺服器適當規模、定價、總體擁有成本比較、遷移成本分析) 以及遷移規劃 (應用程式資料分析和資料收集、應用程式分組、遷移優先順序，以及波次規劃)。MPA 工具 （需要登入） 可供所有 AWS 顧問和 APN 合作夥伴顧問免費使用。

遷移準備程度評定 (MRA)

使用 AWS CAF 取得組織雲端整備狀態的洞見、識別優缺點，以及建立行動計劃以消除已識別差距的程序。如需詳細資訊，請參閱遷移準備程度指南。MRA 是 AWS 遷移策略的第一階段。

遷移策略

用來將工作負載遷移至 的方法 AWS 雲端。如需詳細資訊，請參閱此詞彙表中的 7 個 Rs 項目，並請參閱調動您的組織以加速大規模遷移。

機器學習 (ML)

請參閱機器學習。

現代化

將過時的 (舊版或單一) 應用程式及其基礎架構轉換為雲端中靈活、富有彈性且高度可用的系統，以降低成本、提高效率並充分利用創新。如需詳細資訊，請參閱《》中的現代化應用程式的策略 AWS 雲端。

現代化準備程度評定

這項評估可協助判斷組織應用程式的現代化準備程度；識別優點、風險和相依性；並確定組織能夠在多大程度上支援這些應用程式的未來狀態。評定的結果就是目標架構的藍圖、詳細說明現代化程序的開發階段和里程碑的路線圖、以及解決已發現的差距之行動計畫。如需詳細資訊，請參閱《》中的評估應用程式的現代化準備 AWS 雲端程度。

單一應用程式 (單一)

透過緊密結合的程序作為單一服務執行的應用程式。單一應用程式有幾個缺點。如果一個應用程式功能遇到需求激增，則必須擴展整個架構。當程式碼庫增長時，新增或改進單一應用程式的功能也會變得更加複雜。若要解決這些問題，可以使用微服務架構。如需詳細資訊，請參閱將單一體系分解為微服務。

MPA

請參閱遷移產品組合評估。

MQTT

請參閱訊息佇列遙測傳輸。

多類別分類

一個有助於產生多類別預測的過程 (預測兩個以上的結果之一)。例如，機器學習模型可能會詢問「此產品是書籍、汽車還是電話？」 或者「這個客戶對哪種產品類別最感興趣？」

可變基礎設施

更新和修改生產工作負載現有基礎設施的模型。為了提高一致性、可靠性和可預測性， AWS Well-Architected Framework 建議使用不可變基礎設施做為最佳實務。

O

OAC

請參閱原始存取控制。

OAI

請參閱原始存取身分。

OCM

請參閱組織變更管理。

離線遷移

一種遷移方法，可在遷移過程中刪除來源工作負載。此方法涉及延長停機時間，通常用於小型非關鍵工作負載。

OI

請參閱 操作整合。

OLA

請參閱操作層級協議。

線上遷移

一種遷移方法，無需離線即可將來源工作負載複製到目標系統。連接至工作負載的應用程式可在遷移期間繼續運作。此方法涉及零至最短停機時間，通常用於關鍵的生產工作負載。

OPC-UA

請參閱開放程序通訊 - 統一架構。

開放程序通訊 - 統一架構 (OPC-UA)

用於工業自動化的machine-to-machine(M2M) 通訊協定。OPC-UA 提供資料加密、身分驗證和授權機制的互通性標準。

操作水準協議 (OLA)

一份協議，闡明 IT 職能群組承諾向彼此提供的內容，以支援服務水準協議 (SLA)。

操作整備審查 (ORR)

問題和相關最佳實務的檢查清單，可協助您了解、評估、預防或減少事件和可能失敗的範圍。如需詳細資訊，請參閱 AWS Well-Architected Framework 中的操作整備審查 (ORR)。

操作技術 (OT)

使用實體環境控制工業操作、設備和基礎設施的硬體和軟體系統。在製造中，整合 OT 和資訊技術 (IT) 系統是工業 4.0 轉型的關鍵重點。

操作整合 (OI)

在雲端中將操作現代化的程序，其中包括準備程度規劃、自動化和整合。如需詳細資訊，請參閱操作整合指南。

組織追蹤

由 建立的線索 AWS CloudTrail ，會記錄 AWS 帳戶 組織中所有 的所有事件 AWS Organizations。在屬於組織的每個 AWS 帳戶 中建立此追蹤，它會跟蹤每個帳戶中的活動。如需詳細資訊，請參閱 CloudTrail 文件中的建立組織追蹤。

組織變更管理 (OCM)

用於從人員、文化和領導力層面管理重大、顛覆性業務轉型的架構。OCM 透過加速變更採用、解決過渡問題，以及推動文化和組織變更，協助組織為新系統和策略做好準備，並轉移至新系統和策略。在 AWS 遷移策略中，此架構稱為人員加速，因為雲端採用專案所需的變更速度。如需詳細資訊，請參閱 OCM 指南。

原始存取控制 (OAC)

CloudFront 中的增強型選項，用於限制存取以保護 Amazon Simple Storage Service (Amazon S3) 內容。OAC 支援所有 S3 儲存貯體中的所有伺服器端加密 AWS KMS (SSE-KMS) AWS 區域，以及對 S3 儲存貯體的動態PUT和DELETE請求。

原始存取身分 (OAI)

CloudFront 中的一個選項，用於限制存取以保護 Amazon S3 內容。當您使用 OAI 時，CloudFront 會建立一個可供 Amazon S3 進行驗證的主體。經驗證的主體只能透過特定 CloudFront 分發來存取 S3 儲存貯體中的內容。另請參閱 OAC，它可提供更精細且增強的存取控制。

ORR

請參閱操作整備審核。

OT

請參閱操作技術。

傳出 (輸出) VPC

在 AWS 多帳戶架構中，處理從應用程式內啟動之網路連線的 VPC。AWS 安全參考架構建議您使用傳入、傳出和檢查 VPC 來設定網路帳戶，以保護應用程式與更廣泛的網際網路之間的雙向介面。

P

許可界限

附接至 IAM 主體的 IAM 管理政策，可設定使用者或角色擁有的最大許可。如需詳細資訊，請參閱 IAM 文件中的許可界限。

個人身分識別資訊 (PII)

當直接檢視或與其他相關資料配對時，可用來合理推斷個人身分的資訊。PII 的範例包括名稱、地址和聯絡資訊。

PII

請參閱個人身分識別資訊。

手冊

一組預先定義的步驟，可擷取與遷移關聯的工作，例如在雲端中提供核心操作功能。手冊可以採用指令碼、自動化執行手冊或操作現代化環境所需的程序或步驟摘要的形式。

PLC

請參閱可程式設計邏輯控制器。

PLM

請參閱產品生命週期管理。

政策

可定義許可的物件（請參閱身分型政策）、指定存取條件（請參閱資源型政策），或定義組織中所有帳戶的最大許可 AWS Organizations（請參閱服務控制政策）。

混合持久性

根據資料存取模式和其他需求，獨立選擇微服務的資料儲存技術。如果您的微服務具有相同的資料儲存技術，則其可能會遇到實作挑戰或效能不佳。如果微服務使用最適合其需求的資料儲存，則可以更輕鬆地實作並達到更好的效能和可擴展性。

組合評定

探索、分析應用程式組合並排定其優先順序以規劃遷移的程序。如需詳細資訊，請參閱評估遷移準備程度。

述詞

傳回 `true`或 的查詢條件`false`，通常位於 `WHERE`子句中。

述詞下推

一種資料庫查詢最佳化技術，可在傳輸前篩選查詢中的資料。這可減少必須從關聯式資料庫擷取和處理的資料量，並改善查詢效能。

預防性控制

旨在防止事件發生的安全控制。這些控制是第一道防線，可協助防止對網路的未經授權存取或不必要變更。如需詳細資訊，請參閱在 AWS上實作安全控制中的預防性控制。

委託人

中可執行動作和存取資源 AWS 的實體。此實體通常是 AWS 帳戶、IAM 角色或使用者的根使用者。如需詳細資訊，請參閱 IAM 文件中角色術語和概念中的主體。

依設計的隱私權

透過整個開發程序將隱私權納入考量的系統工程方法。

私有託管區域

一種容器，它包含有關您希望 Amazon Route 53 如何回應一個或多個 VPC 內的域及其子域之 DNS 查詢的資訊。如需詳細資訊，請參閱 Route 53 文件中的使用私有託管區域。

主動控制

旨在防止部署不合規資源的安全控制。這些控制項會在佈建資源之前對其進行掃描。如果資源不符合控制項，則不會佈建。如需詳細資訊，請參閱 AWS Control Tower 文件中的控制項參考指南，並參閱實作安全控制項中的主動控制項。 AWS

產品生命週期管理 (PLM)

管理產品整個生命週期的資料和程序，從設計、開發和啟動，到成長和成熟，再到拒絕和移除。

生產環境

請參閱 環境。

可程式設計邏輯控制器 (PLC)

在製造中，高度可靠、可調整的電腦，可監控機器並自動化製造程序。

提示鏈結

使用一個 LLM 提示的輸出做為下一個提示的輸入，以產生更好的回應。此技術用於將複雜任務分解為子任務，或反覆精簡或展開初步回應。它有助於提高模型回應的準確性和相關性，並允許更精細、個人化的結果。

擬匿名化

以預留位置值取代資料集中個人識別符的程序。假名化有助於保護個人隱私權。假名化資料仍被視為個人資料。

發佈/訂閱 (pub/sub)

一種模式，可啟用微服務之間的非同步通訊，以提高可擴展性和回應能力。例如，在微服務型 MES 中，微服務可以將事件訊息發佈到其他微服務可訂閱的頻道。系統可以新增新的微服務，而無需變更發佈服務。

Q

查詢計劃

一系列步驟，如指示，用於存取 SQL 關聯式資料庫系統中的資料。

查詢計劃迴歸

在資料庫服務優化工具選擇的計畫比對資料庫環境進行指定的變更之前的計畫不太理想時。這可能因為對統計資料、限制條件、環境設定、查詢參數繫結的變更以及資料庫引擎的更新所導致。

R

RACI 矩陣

請參閱負責、負責、諮詢、告知 (RACI)。

RAG

請參閱擷取增強產生。

勒索軟體

一種惡意軟體，旨在阻止對計算機系統或資料的存取，直到付款為止。

RASCI 矩陣

請參閱負責、負責、諮詢、告知 (RACI)。

RCAC

請參閱資料列和資料欄存取控制。

僅供讀取複本

用於唯讀用途的資料庫複本。您可以將查詢路由至僅供讀取複本以減少主資料庫的負載。

重新架構師

請參閱 7 Rs。

復原點目標 (RPO)

自上次資料復原點以來可接受的時間上限。這會決定最後一個復原點與服務中斷之間可接受的資料遺失。

復原時間目標 (RTO)

服務中斷與服務還原之間的可接受延遲上限。

重構

請參閱 7 Rs。

區域

地理區域中的 AWS 資源集合。每個 AWS 區域 都獨立於其他 ，以提供容錯能力、穩定性和彈性。如需詳細資訊，請參閱指定 AWS 區域 您的帳戶可以使用哪些。

迴歸

預測數值的 ML 技術。例如，為了解決「這房子會賣什麼價格？」的問題 ML 模型可以使用線性迴歸模型，根據已知的房屋事實 (例如，平方英尺) 來預測房屋的銷售價格。

重新託管

請參閱 7 Rs。

版本

在部署程序中，它是將變更提升至生產環境的動作。

重新定位

請參閱 7 個 R。

Replatform

請參閱 7 個 R。

回購

請參閱 7 Rs。

彈性

應用程式抵禦中斷或從中斷中復原的能力。在 中規劃彈性時，高可用性和災難復原是常見的考量 AWS 雲端。如需詳細資訊，請參閱AWS 雲端 彈性。

資源型政策

附接至資源的政策，例如 Amazon S3 儲存貯體、端點或加密金鑰。這種類型的政策會指定允許存取哪些主體、支援的動作以及必須滿足的任何其他條件。

負責者、當責者、事先諮詢者和事後告知者 (RACI) 矩陣

定義所有涉及遷移活動和雲端操作之各方的角色和責任的矩陣。矩陣名稱衍生自矩陣中定義的責任類型：負責人 (R)、責任 (A)、諮詢 (C) 和知情 (I)。支援 (S) 類型為選用。如果您包含支援，則矩陣稱為 RASCI 矩陣，如果您排除它，則稱為 RACI 矩陣。

回應性控制

一種安全控制，旨在驅動不良事件或偏離安全基準的補救措施。如需詳細資訊，請參閱在 AWS上實作安全控制中的回應性控制。

保留

請參閱 7 個 R。

淘汰

請參閱 7 Rs。

檢索增強生成 (RAG)

一種生成式 AI 技術，其中 LLM 會在產生回應之前參考訓練資料來源以外的授權資料來源。例如，RAG 模型可能會對組織的知識庫或自訂資料執行語意搜尋。如需詳細資訊，請參閱什麼是 RAG。

輪換

定期更新秘密的程序，讓攻擊者更難存取登入資料。

資料列和資料欄存取控制 (RCAC)

使用已定義存取規則的基本、彈性 SQL 表達式。RCAC 包含資料列許可和資料欄遮罩。

RPO

請參閱復原點目標。

RTO

請參閱復原時間目標。

執行手冊

執行特定任務所需的一組手動或自動程序。這些通常是為了簡化重複性操作或錯誤率較高的程序而建置。

S

SAML 2.0

許多身分提供者 (IdP) 使用的開放標準。此功能會啟用聯合單一登入 (SSO)，讓使用者可以登入 AWS 管理主控台 或呼叫 AWS API 操作，而不必為您組織中的每個人在 IAM 中建立使用者。如需有關以 SAML 2.0 為基礎的聯合詳細資訊，請參閱 IAM 文件中的關於以 SAML 2.0 為基礎的聯合。

SCADA

請參閱監督控制和資料擷取。

SCP

請參閱服務控制政策。

秘密

您以加密形式存放的 AWS Secrets Manager機密或限制資訊，例如密碼或使用者登入資料。它由秘密值及其中繼資料組成。秘密值可以是二進位、單一字串或多個字串。如需詳細資訊，請參閱 Secrets Manager 文件中的 Secrets Manager 秘密中的什麼內容？。

依設計的安全性

透過整個開發程序將安全性納入考量的系統工程方法。

安全控制

一種技術或管理防護機制，它可預防、偵測或降低威脅行為者利用安全漏洞的能力。安全控制有四種主要類型：預防性、偵測性、回應性和主動性。

安全強化

減少受攻擊面以使其更能抵抗攻擊的過程。這可能包括一些動作，例如移除不再需要的資源、實作授予最低權限的安全最佳實務、或停用組態檔案中不必要的功能。

安全資訊與事件管理 (SIEM) 系統

結合安全資訊管理 (SIM) 和安全事件管理 (SEM) 系統的工具與服務。SIEM 系統會收集、監控和分析來自伺服器、網路、裝置和其他來源的資料，以偵測威脅和安全漏洞，並產生提醒。

安全回應自動化

預先定義和程式設計的動作，旨在自動回應或修復安全事件。這些自動化可做為偵測或回應式安全控制，協助您實作 AWS 安全最佳實務。自動化回應動作的範例包括修改 VPC 安全群組、修補 Amazon EC2 執行個體或輪換登入資料。

伺服器端加密

由接收資料的 AWS 服務 在其目的地加密資料。

服務控制政策 (SCP)

為 AWS Organizations中的組織的所有帳戶提供集中控制許可的政策。SCP 會定義防護機制或設定管理員可委派給使用者或角色的動作限制。您可以使用 SCP 作為允許清單或拒絕清單，以指定允許或禁止哪些服務或動作。如需詳細資訊，請參閱 AWS Organizations 文件中的服務控制政策。

服務端點

的進入點 URL AWS 服務。您可以使用端點，透過程式設計方式連接至目標服務。如需詳細資訊，請參閱 AWS 一般參考 中的 AWS 服務 端點。

服務水準協議 (SLA)

一份協議，闡明 IT 團隊承諾向客戶提供的服務，例如服務正常執行時間和效能。

服務層級指標 (SLI)

服務效能方面的測量，例如其錯誤率、可用性或輸送量。

服務層級目標 (SLO)

代表服務運作狀態的目標指標，由服務層級指標測量。

共同責任模式

描述您與 共同 AWS 承擔雲端安全與合規責任的模型。 AWS 負責雲端的安全，而 負責雲端的安全。如需詳細資訊，請參閱共同責任模式。

SIEM

請參閱安全資訊和事件管理系統。

單一故障點 (SPOF)

應用程式的單一關鍵元件故障，可能會中斷系統。

SLA

請參閱服務層級協議。

SLI

請參閱服務層級指標。

SLO

請參閱服務層級目標。

先拆分後播種模型

擴展和加速現代化專案的模式。定義新功能和產品版本時，核心團隊會進行拆分以建立新的產品團隊。這有助於擴展組織的能力和服務，提高開發人員生產力，並支援快速創新。如需詳細資訊，請參閱 中的階段式應用程式現代化方法 AWS 雲端。

SPOF

請參閱單一故障點。

星狀結構描述

使用一個大型事實資料表來存放交易或測量資料的資料庫組織結構，並使用一或多個較小的維度資料表來存放資料屬性。此結構旨在用於資料倉儲或商業智慧用途。

Strangler Fig 模式

一種現代化單一系統的方法，它會逐步重寫和取代系統功能，直到舊式系統停止使用為止。此模式源自無花果藤，它長成一棵馴化樹並最終戰勝且取代了其宿主。該模式由 Martin Fowler 引入，作為重寫單一系統時管理風險的方式。如需有關如何套用此模式的範例，請參閱使用容器和 Amazon API Gateway 逐步現代化舊版 Microsoft ASP.NET (ASMX) Web 服務。

子網

您 VPC 中的 IP 地址範圍。子網必須位於單一可用區域。

監控控制和資料擷取 (SCADA)

在製造中，使用硬體和軟體來監控實體資產和生產操作的系統。

對稱加密

使用相同金鑰來加密及解密資料的加密演算法。

合成測試

以模擬使用者互動的方式測試系統，以偵測潛在問題或監控效能。您可以使用 Amazon CloudWatch Synthetics 來建立這些測試。

系統提示

一種向 LLM 提供內容、指示或指導方針以指示其行為的技術。系統提示有助於設定內容，並建立與使用者互動的規則。

T

標籤

做為中繼資料以組織 AWS 資源的鍵/值對。標籤可協助您管理、識別、組織、搜尋及篩選資源。如需詳細資訊，請參閱標記您的 AWS 資源。

目標變數

您嘗試在受監督的 ML 中預測的值。這也被稱為結果變數。例如，在製造設定中，目標變數可能是產品瑕疵。

任務清單

用於透過執行手冊追蹤進度的工具。任務清單包含執行手冊的概觀以及要完成的一般任務清單。對於每個一般任務，它包括所需的預估時間量、擁有者和進度。

測試環境

請參閱 環境。

訓練

為 ML 模型提供資料以供學習。訓練資料必須包含正確答案。學習演算法會在訓練資料中尋找將輸入資料屬性映射至目標的模式 (您想要預測的答案)。它會輸出擷取這些模式的 ML 模型。可以使用 ML 模型，來預測您不知道的目標新資料。

傳輸閘道

可以用於互連 VPC 和內部部署網路的網路傳輸中樞。如需詳細資訊，請參閱 AWS Transit Gateway 文件中的什麼是傳輸閘道。

主幹型工作流程

這是一種方法，開發人員可在功能分支中本地建置和測試功能，然後將這些變更合併到主要分支中。然後，主要分支會依序建置到開發環境、生產前環境和生產環境中。

受信任的存取權

將許可授予您指定的服務，以代表您在組織中 AWS Organizations 及其帳戶中執行任務。受信任的服務會在需要該角色時，在每個帳戶中建立服務連結角色，以便為您執行管理工作。如需詳細資訊，請參閱 文件中的 AWS Organizations 搭配使用 AWS Organizations 與其他 AWS 服務。

調校

變更訓練程序的各個層面，以提高 ML 模型的準確性。例如，可以透過產生標籤集、新增標籤、然後在不同的設定下多次重複這些步驟來訓練 ML 模型，以優化模型。

雙比薩團隊

兩個比薩就能吃飽的小型 DevOps 團隊。雙披薩團隊規模可確保軟體開發中的最佳協作。

U

不確定性

這是一個概念，指的是不精確、不完整或未知的資訊，其可能會破壞預測性 ML 模型的可靠性。有兩種類型的不確定性：認知不確定性是由有限的、不完整的資料引起的，而隨機不確定性是由資料中固有的噪聲和隨機性引起的。如需詳細資訊，請參閱量化深度學習系統的不確定性指南。

未區分的任務

也稱為繁重工作，這是建立和操作應用程式的必要工作，但不為最終使用者提供直接價值或提供競爭優勢。未區分任務的範例包括採購、維護和容量規劃。

較高的環境

請參閱 環境。

V

清空

一種資料庫維護操作，涉及增量更新後的清理工作，以回收儲存並提升效能。

版本控制

追蹤變更的程序和工具，例如儲存庫中原始程式碼的變更。

VPC 對等互連

兩個 VPC 之間的連線，可讓您使用私有 IP 地址路由流量。如需詳細資訊，請參閱 Amazon VPC 文件中的什麼是 VPC 對等互連。

漏洞

危害系統安全性的軟體或硬體瑕疵。

W

暖快取

包含經常存取的目前相關資料的緩衝快取。資料庫執行個體可以從緩衝快取讀取，這比從主記憶體或磁碟讀取更快。

暖資料

不常存取的資料。查詢這類資料時，通常可接受中等速度的查詢。

視窗函數

SQL 函數，對與目前記錄在某種程度上相關的資料列群組執行計算。視窗函數適用於處理任務，例如根據目前資料列的相對位置計算移動平均值或存取資料列的值。

工作負載

提供商業價值的資源和程式碼集合，例如面向客戶的應用程式或後端流程。

工作串流

遷移專案中負責一組特定任務的功能群組。每個工作串流都是獨立的，但支援專案中的其他工作串流。例如，組合工作串流負責排定應用程式、波次規劃和收集遷移中繼資料的優先順序。組合工作串流將這些資產交付至遷移工作串流，然後再遷移伺服器和應用程式。

WORM

請參閱寫入一次，讀取許多。

WQF

請參閱AWS 工作負載資格架構。

寫入一次，讀取許多 (WORM)

儲存模型，可一次性寫入資料，並防止刪除或修改資料。授權使用者可以視需要多次讀取資料，但無法變更資料。此資料儲存基礎設施被視為不可變。

Z

零時差入侵

利用零時差漏洞的攻擊，通常是惡意軟體。

零時差漏洞

生產系統中未緩解的缺陷或漏洞。威脅行為者可以使用這種類型的漏洞來攻擊系統。開發人員經常因為攻擊而意識到漏洞。

零鏡頭提示

提供 LLM 執行任務的指示，但沒有可協助引導任務的範例 (快照)。LLM 必須使用其預先訓練的知識來處理任務。零鏡頭提示的有效性取決於任務的複雜性和提示的品質。另請參閱少量擷取提示。

殭屍應用程式

CPU 和記憶體平均使用率低於 5% 的應用程式。在遷移專案中，通常會淘汰這些應用程式。

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。