在 上建置六邊形架構 AWS

AWS 方案指引

AWS 方案指引: 在 上建置六邊形架構 AWS

Table of Contents

在 上建置六邊形架構 AWS

Furkan Oruc、Dominik Goby、Darius Kunce 和 Michal Ploski，Amazon Web Services (AWS)

2022 年 6 月 (文件歷史記錄)

本指南說明用於開發軟體架構的心理模型和模式集合。隨著產品採用的成長，這些架構很容易在整個組織中維護、擴展和擴展。Amazon Web Services (AWS) 等雲端超擴展器為小型和大型企業提供建置區塊，以創新和建立新的軟體產品。這些新服務和功能介紹的快速步調，可讓業務利益相關者期望其開發團隊更快地將新的可行最低產品 (MVPs) 建立原型，以便儘快測試和驗證新想法。通常，這些 MVPs會採用並成為企業軟體生態系統的一部分。在生產這些 MVPs的過程中，團隊有時會放棄軟體開發規則和最佳實務，例如 SOLID 原則和單元測試。他們假設這種方法將加速開發並縮短上市時間。不過，如果他們無法建立基礎模型和所有層級的軟體架構架構，則很難甚至不可能為產品開發新功能。缺乏確定性和不斷變化的需求也會在開發過程中拖慢團隊的速度。

本指南會逐步介紹提議的軟體架構，從低階六邊形架構到高階架構和組織分解，使用網域驅動設計 (DDD) 來解決這些挑戰。DDD 有助於管理業務複雜性，並在開發新功能時擴展工程團隊。它使用無處不在的語言，使業務和技術利益相關者與稱為網域的業務問題保持一致。六角形架構是這種方法在非常特定網域中的技術實現者，稱為邊界內容。邊界內容是業務問題高度凝聚且鬆散耦合的子區域。建議您針對所有企業軟體專案採用六邊形架構，無論其複雜性為何。

六角形架構會鼓勵工程團隊先解決業務問題，而傳統分層架構則會將工程焦點從網域轉移到先解決技術問題。此外，如果軟體遵循六邊形架構，則更容易採用測試驅動的開發方法，從而減少開發人員測試業務需求所需的意見回饋迴圈。最後，使用命令和命令處理常式是從 SOLID 套用單一責任和開放關閉原則的一種方式。遵守這些原則會產生程式碼庫，開發人員和架構師可以輕鬆導覽和了解專案，並降低將突破性變更引入現有功能的風險。

本指南適用於軟體架構師和開發人員，他們有興趣了解為其軟體開發專案採用六邊形架構和 DDD 的優勢。它包含為 AWS 支援六邊形架構的應用程式設計基礎設施的範例。如需實作範例，請參閱 AWS 規範指引網站上的使用 在六邊形架構中建構 Python 專案 AWS Lambda。

概觀

網域驅動設計 (DDD)

在網域驅動設計 (DDD) 中，網域是軟體系統的核心。網域模型會在您開發任何其他模組之前先定義，而且不依賴其他低階模組。相反地，資料庫、簡報層和外部 APIs等模組都取決於網域。

在 DDD 中，架構師會使用商業邏輯型分解而非技術分解，將解決方案分解為受限內容。此方法的優點會在 目標業務成果章節中討論。

當團隊使用六邊形架構時，DDD 更容易實作。在六邊形架構中，應用程式核心是應用程式的中心。它會透過連接埠和轉接器從其他模組解耦，並且沒有其他模組的相依性。這與 DDD 完美一致，其中網域是解決業務問題的應用程式核心。本指南提議一種方法，其中您將六邊形架構的核心建模為邊界內容的網域模型。下一節會更詳細地說明六邊形架構。

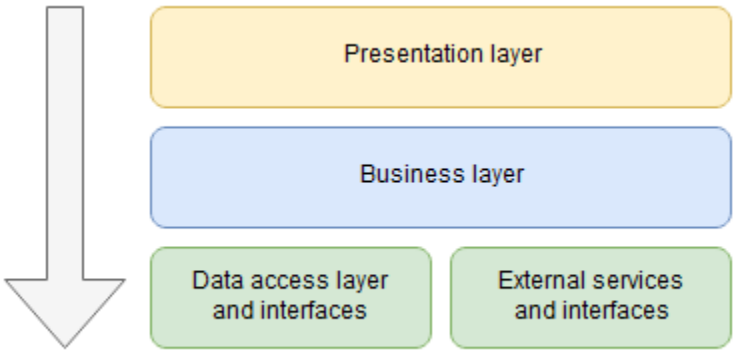
本指南未涵蓋 DDD 的所有層面，這是一個非常廣泛的主題。若要進一步了解，您可以檢閱域語言網站上列出的 DDD 資源。

六角形架構

六角型架構也稱為連接埠和轉接器或小齒輪架構，是管理軟體專案中相依性反轉的原則。六角架構在開發軟體時提升了對核心網域商業邏輯的高度關注，並將外部整合點視為次要。六角型架構可協助軟體工程師採用測試驅動開發 (TDD) 等良好實務，進而促進架構演進，並協助您長期管理複雜的網域。

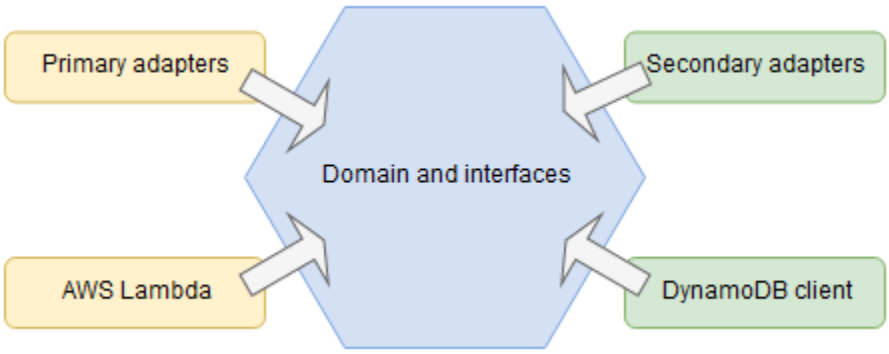
讓我們比較六邊形架構與傳統分層架構，這是建立結構化軟體專案模型時最熱門的選擇。這兩種方法之間存在細微但強大的差異。

在分層架構中，軟體專案以層結構，代表廣泛的問題，例如商業邏輯或簡報邏輯。此架構使用相依性階層，其中最上層對它們下方的層具有相依性，但不是反之亦然。在下圖中，簡報層負責使用者互動，因此包含使用者介面、APIs、命令列介面和類似的元件。呈現層與實作網域邏輯的業務層有相依性。商業層又對資料存取層和多個外部服務具有相依性。



此組態的主要缺點是相依性結構。例如，如果在資料庫中存放資料的模型變更，這會影響資料存取界面。資料模型的任何變更也會影響業務層，而業務層依賴於資料存取界面。因此，軟體工程師無法在不影響網域邏輯的情況下進行任何基礎設施變更。這反過來會增加迴歸錯誤的可能性。

六角形架構以不同的方式定義相依性關係，如下圖所示。它會集中在定義所有界面的網域商業邏輯上進行決策。外部元件會透過稱為連接埠的界面與商業邏輯互動。連接埠是定義網域與外部世界互動的抽象。每個基礎設施元件都必須實作這些連接埠，因此這些元件的變更不會再影響核心網域邏輯。



周圍的元件稱為轉接器。轉接器是外部世界與內部世界之間的代理，並實作網域中定義的連接埠。轉接器可以分為兩個群組：主要和次要。主要轉接器是軟體元件的進入點。它們允許外部演員、使用者和服務與核心邏輯互動。 AWS Lambda 是主要轉接器的良好範例。它與可叫用 Lambda 函數做為進入點的多個 AWS 服務整合。次要轉接器是外部服務程式庫包裝函式，可處理與外部世界的通訊。次要轉接器的良好範例是用於資料存取的 Amazon DynamoDB 用戶端。

目標業務成果

本指南中討論的六邊形架構可協助您達成下列目標：

- 透過改善開發週期來縮短上市時間

- 改善軟體品質
- 更輕鬆地適應變更

以下各節會詳細討論這些程序。

改善開發週期

為雲端開發軟體為軟體工程師帶來了新的挑戰，因為在開發機器上本機複寫執行期環境非常困難。驗證軟體的簡單方法是將其部署到雲端，並在雲端進行測試。不過，這種方法涉及冗長的意見回饋週期，特別是當軟體架構包含多個無伺服器部署時。改善此意見回饋週期可縮短開發功能的時間，大幅縮短上市時間。

在雲端進行測試

直接在雲端進行測試是確保架構元件設定正確的唯一方法，例如 Amazon API Gateway、 AWS Lambda 函數、Amazon DynamoDB 資料表和 AWS Identity and Access Management (IAM) 許可中的閘道。它也可能是測試元件整合的唯一可靠方式。雖然某些 AWS 服務（例如 DynamoDB) 可以部署在本機，但大多數服務無法在本機設定中複寫。同時，用於測試目的之模擬 AWS 服務的第三方工具，例如 Moto 和 LocalStack，可能無法準確反映實際服務 API 合約，或功能數量可能受到限制。

不過，企業軟體最複雜的部分是在商業邏輯中，而不是在雲端架構中。架構變更的頻率低於網域，因此必須適應新的業務需求。因此，在雲端中測試商業邏輯會成為對程式碼進行變更、啟動部署、等待環境就緒，以及驗證變更的密集程序。如果部署只需要 5 分鐘，在商業邏輯中進行和測試 10 次變更將需要一小時或更長時間。如果商業邏輯更為複雜，測試可能需要幾天的時間等待部署完成。如果您在團隊中有多個功能和工程師，則業務很快就會注意到延長的期間。

在本機測試

六邊形架構可協助開發人員專注於網域，而非基礎設施技術。此方法使用本機測試（您所選開發架構中的單元測試工具）來涵蓋網域邏輯需求。您不需要花時間解決技術整合問題，或將軟體部署到雲端來測試商業邏輯。您可以在本機執行單元測試，並將回饋迴圈從幾分鐘縮短為幾秒。如果部署需要 5 分鐘，但單元測試在 5 秒內完成，則偵測錯誤所需的時間會大幅縮短。本指南稍後的 先測試商業邏輯章節會更詳細地介紹此方法。

平行化開發

六邊形架構方法可讓開發團隊平行化開發工作。開發人員可以個別設計和實作服務的不同元件。透過隔離每個元件和每個元件之間的定義界面，即可實現此平行化。

產品上市時間

本機單位測試可改善開發意見回饋週期，並縮短新產品或功能的上市時間，尤其是在其中包含複雜的商業邏輯時，如前所述。此外，增加單位測試的程式碼涵蓋範圍可大幅降低在更新或重構程式碼基礎時引入迴歸錯誤的風險。單元測試涵蓋範圍也可讓您持續重構程式碼基礎，使其保持井然有序，從而加快新工程師的加入程序。設計品質 本節會進一步討論。最後，如果商業邏輯經過良好隔離和測試，可讓開發人員快速適應不斷變化的功能和非功能需求。適應變更 本節會進一步說明。

設計品質

採用六邊形架構有助於從專案開始時提升程式碼基礎的品質。請務必建置程序，協助您從一開始就符合預期的品質要求，而不會拖慢開發程序。

當地語系化變更和提高可讀性

使用六邊形架構方法可讓開發人員變更一個類別或元件中的程式碼，而不會影響其他類別或元件。此設計可提升已開發元件的凝聚力。透過從轉接器解耦網域並使用眾所周知的介面，您可以提高程式碼的可讀性。識別問題和轉角案例變得更容易。

此方法也有助於在開發期間進行程式碼檢閱，並限制引進未偵測到的變更或技術債務。

先測試商業邏輯

本機測試可以透過將end-to-end、整合和單元測試引入專案來完成。End-to-end測試涵蓋整個傳入請求生命週期。他們通常會叫用應用程式進入點，並測試它是否已完成業務需求。每個軟體專案應至少有一個測試案例，使用已知輸入並產生預期輸出。不過，新增更多邊角案例案例可能會變得複雜，因為每個測試都必須設定為透過進入點（例如，透過 REST API 或佇列）傳送請求，然後完成業務動作所需的所有整合點，然後宣告結果。為測試案例設定環境並宣告結果可能需要許多開發人員的時間。

在六邊形架構中，您會獨立測試商業邏輯，並使用整合測試來測試次要轉接器。您可以在商業邏輯測試中使用模擬或仿造轉接器。您也可以將業務使用案例的測試與網域模型的單元測試結合，以維持低耦合的高涵蓋率。最佳實務是，整合測試不應驗證業務邏輯。反之，他們應該驗證次要轉接器是否正確呼叫外部服務。

理想情況下，您可以使用測試驅動的開發 (TDD)，並在開發開始時透過適當的測試開始定義網域實體或商業使用案例。首先撰寫測試可協助您建立網域所需界面的模擬實作。當測試成功且符合網域邏輯規則時，您可以實作實際轉接器並將軟體部署到測試環境。此時，您的網域邏輯實作可能不理想。然後，您可以引進設計模式或重新配置程式碼，以重構現有的架構來發展它。透過使用此方法，您可以避免引入迴歸錯誤，而且您可以隨著專案的成長而改善架構。透過將此方法與您在持續整合程序中執行的自動測試結合，您可以在潛在錯誤進入生產環境之前減少它們的數量。

如果您使用無伺服器部署，您可以在 AWS 帳戶中快速佈建應用程式執行個體，以進行手動整合和end-to-end測試。在這些實作步驟之後，我們建議您在推送到儲存庫的每個新變更中自動進行測試。

可維護性

可維護性是指操作和監控應用程式，以確保其符合所有要求，並將系統故障的機率降至最低。若要讓系統正常運作，您必須將其調整為符合未來的流量或操作需求。您也必須確保其可用且易於部署，且對用戶端具有最小或無影響。

若要了解系統的目前和歷史狀態，您必須讓它成為可觀察的狀態。您可以提供特定的指標、日誌和追蹤，讓運算子用來確保系統如預期般運作並追蹤錯誤，藉此達成此目的。這些機制也應該允許運算子執行根本原因分析，而不必登入機器並讀取程式碼。

六邊形架構旨在提高 Web 應用程式的可維護性，因此您的程式碼整體上需要的工作較少。透過將模組、當地語系化變更，以及取消應用程式商業邏輯與轉接器實作的耦合，您可以產生指標和日誌，協助運算子深入了解系統，並了解對主要或次要轉接器所做的特定變更範圍。

適應變更

軟體系統往往會變得複雜。其中一個原因可能是經常變更業務需求，而且很少時間相應地調整軟體架構。另一個原因是，在專案開始時設定軟體架構以適應頻繁變更的投資不足。無論原因為何，軟體系統都可能變得複雜，幾乎無法進行變更。因此，從專案開始時建置可維護的軟體架構非常重要。良好的軟體架構可以輕鬆適應變更。

本節說明如何使用易於適應非功能或業務需求的六邊形架構來設計可維護的應用程式。

使用連接埠和轉接器來適應新的非功能需求

做為應用程式的核心，網域模型會定義外部世界滿足業務需求所需的動作。這些動作是透過抽象定義，稱為連接埠。這些連接埠由不同的轉接器實作。每個轉接器都負責與另一個系統的互動。例如，您可能有一個用於資料庫儲存庫的轉接器，以及另一個用於與第三方 API 互動的轉接器。網域不知道轉接器實作，因此很容易將一個轉接器取代為另一個轉接器。例如，應用程式可能會從 SQL 資料庫切換到 NoSQL 資料庫。在此情況下，必須開發新的轉接器，以實作網域模型定義的連接埠。網域對資料庫儲存庫沒有相依性，並使用抽象進行互動，因此不需要變更網域模型中的任何內容。因此，六邊形架構可輕鬆適應非功能需求。

使用命令和命令處理常式來適應新的業務需求

在傳統分層架構中，網域取決於持久性層。如果您想要變更網域，您也必須變更持久性層。相比之下，在六邊形架構中，網域不依賴軟體中的其他模組。網域是應用程式的核心，所有其他模組（連接埠和轉接器）都取決於網域模型。網域使用相依性反轉原則，透過連接埠與外部世界通訊。相依性反轉的好處是您可以自由變更網域模型，而不會害怕破壞程式碼的其他部分。由於網域模型會反映您嘗試解決的業務問題，因此更新網域模型以適應不斷變化的業務需求並非問題。

當您開發軟體時，問題分離是必須遵循的重要原則。若要達成此區隔，您可以使用稍微修改的命令模式。這是一種行為設計模式，其中所有完成操作的必要資訊都會封裝在命令物件中。然後，命令處理常式會處理這些操作。命令處理常式是接收命令、變更網域狀態，然後將回應傳回給發起人的方法。您可以使用同步 APIs 或非同步佇列等不同的用戶端來執行命令。建議您針對網域上的每個操作使用命令和命令處理常式。透過遵循此方法，您可以引進新的命令和命令處理常式來新增功能，而無需變更現有的商業邏輯。因此，使用命令模式可讓您更輕鬆地適應新的業務需求。

使用服務外牆或 CQRS 模式來解耦元件

在六邊形架構中，主要轉接器負責將來自用戶端的傳入讀取和寫入請求鬆散地耦合到網域。有兩種方式可以實現這種鬆散耦合：使用服務外牆模式或使用命令查詢責任隔離 (CQRS) 模式。

服務外牆模式提供面向前方的界面，為呈現層或微服務等用戶端提供服務。服務外牆為用戶端提供數個讀取和寫入操作。它負責將傳入請求轉移到網域，並將從網域接收的回應映射到用戶端。對於具有多項操作的單一責任的微服務，使用服務外牆很容易。不過，使用服務外牆時，很難遵循單一責任和開放式原則。單一責任原則指出每個模組應僅負責軟體的單一功能。開放關閉原則指出程式碼應開放延伸，並關閉以進行修改。隨著服務外觀的擴展，所有操作都收集在一個界面中，將更多的相依性封裝在其中，並且更多開發人員開始修改相同的外觀。因此，我們建議只在明確表示服務在開發期間不會大幅擴展時，才使用服務外牆。

在六邊形架構中實作主要轉接器的另一種方法是使用 CQRS 模式，這會使用查詢和命令來區隔讀取和寫入操作。如前所述，命令是包含變更網域狀態所需的所有資訊的物件。命令是由命令處理常式方法執行。另一方面，查詢不會改變系統的狀態。其唯一目的是將資料傳回給用戶端。在 CQRS 模式中，命令和查詢會在不同的模組中實作。這對於遵循事件驅動架構的專案特別有利，因為命令可以實作為非同步處理的事件，而查詢可以使用 API 同步執行。查詢也可以使用針對它最佳化的不同資料庫。CQRS 模式的缺點是實作需要比服務外牆更多的時間。對於您計劃長期擴展和維護的專案，我們建議您使用 CQRS 模式。命令和查詢提供有效的機制，用於套用單一責任原則和開發鬆耦合的軟體，尤其是在大規模專案中。

CQRS 的長期優勢很大，但需要初始投資。因此，我們建議您在決定使用 CQRS 模式之前仔細評估專案。不過，您可以從一開始就使用命令和命令處理常式來建構應用程式，而不需分開讀取/寫入操作。如果您決定稍後採用該方法，這將協助您輕鬆地重構 CQRS 的專案。

組織擴展

六邊形架構、網域驅動設計和（選用）CQRS 的組合可讓您的組織快速擴展產品。根據 Conway 法律，軟體架構往往會演進以反映公司的通訊結構。此觀察歷史上具有負面含義，因為大型組織通常會根據資料庫、企業服務匯流排等技術專業知識來建構團隊。這種方法的問題在於，產品和功能開發一律涉及交叉考量，例如安全性和可擴展性，這需要團隊之間持續通訊。以技術功能為基礎的結構化團隊會在組織中產生不必要的孤立，這會導致溝通不佳、缺乏擁有權，以及對大局視而不見。最後，這些組織問題會反映在軟體架構中。

另一方面，Inverse Conway Maneuver 會根據提升軟體架構的網域來定義組織結構。例如，跨職能團隊必須負責一組特定的邊界內容，這些內容是透過使用 DDD 和事件風暴來識別。這些邊界內容可能會反映產品的非常特定功能。例如，帳戶團隊可能需要負責付款內容。每個新功能都會指派給具有高度凝

聚性和鬆散耦合責任的新團隊，因此他們只能專注於該功能的交付，並減少上市時間。團隊可以根據功能的複雜性進行擴展，因此可以將複雜的功能指派給更多工程師。

最佳實務

建立商業網域的模型

從商業網域回到軟體設計，以確保您撰寫的軟體符合業務需求。

使用網域驅動型設計 (DDD) 方法，例如事件風暴，來建立商業網域的模型。事件風暴具有彈性的研討會格式。在研討會期間，網域和軟體專家會共同探索商業網域的複雜性。軟體專家使用研討會的交付項目來啟動軟體元件的設計和開發程序。

從頭開始撰寫和執行測試

使用測試驅動開發 (TDD) 驗證您正在開發之軟體的正確性。TDD 最適合在單元測試層級運作。開發人員會先撰寫測試來設計軟體元件，該測試會叫用該元件。該元件一開始沒有實作，因此測試會失敗。下一個步驟是開發人員實作元件的功能，使用測試固定裝置搭配模擬物件來模擬外部相依性或連接埠的行為。當測試成功時，開發人員可以實作真正的轉接器以繼續。這種方法可改善軟體品質並產生更易讀的程式碼，因為開發人員了解使用者如何使用元件。六角架構透過分隔應用程式核心來支援 TDD 方法。開發人員撰寫著重於網域核心行為的單元測試。他們不需要撰寫複雜的轉接器來執行測試，而是可以使用簡單的模擬物件和固定裝置。

使用行為驅動型開發 (BDD)，以確保end-to-end。在 BDD 中，開發人員會定義功能的案例，並與業務利益相關者進行驗證。BDD 測試會盡可能使用自然語言來達成此目標。六角架構支援 BDD 方法及其主要和次要轉接器的概念。開發人員可以建立可在本機執行的主要和次要轉接器，而無需呼叫外部服務。他們會將 BDD 測試套件設定為使用本機主要轉接器來執行應用程式。

在持續整合管道中自動執行每個測試，以持續評估系統的品質。

定義網域的行為

將網域分解為實體、值物件和彙總 （閱讀有關實作網域驅動的設計)，並定義其行為。實作網域的行為，讓在專案開始時寫入的測試成功。定義叫用網域物件行為的命令。定義網域物件在完成行為後發出的事件。

定義轉接器可用來與網域互動的界面。

自動化測試和部署

在初始概念驗證之後，我們建議您花時間實作 DevOps 實務。例如，持續整合和持續交付 (CI/CD) 管道和動態測試環境可協助您維持程式碼的品質，並避免在部署期間發生錯誤。

- 在 CI 程序內執行您的單元測試，並在程式碼合併之前進行測試。
- 建置 CD 程序，將您的應用程式部署到靜態開發/測試環境，或動態建立的環境，以支援自動整合和 end-to-end測試。
- 自動化專用環境的部署程序。

使用微服務和 CQRS 擴展您的產品

如果您的產品成功，請將您的軟體專案分解為微服務，以擴展您的產品。利用六邊形架構提供的可攜性來改善效能。將查詢服務和命令處理常式分割為不同的同步和非同步 APIs。考慮採用命令查詢責任隔離 (CQRS) 模式和事件驅動型架構。

如果您收到許多新功能請求，請考慮根據 DDD 模式擴展您的組織。以團隊擁有一個或多個特徵作為邊界內容的方式組織您的團隊，如先前組織擴展章節中所述。這些團隊接著可以使用六邊形架構來實作商業邏輯。

設計對應至六邊形架構概念的專案結構

基礎設施即程式碼 (IaC) 是雲端開發中廣泛採用的實務。它可讓您將基礎設施資源 （例如網路、負載平衡器、虛擬機器和閘道） 定義為原始碼，並加以維護。如此一來，您可以使用版本控制系統來追蹤架構的所有變更。此外，您可以輕鬆建立和移動基礎設施，以供測試之用。我們建議您在開發雲端應用程式時，將應用程式程式碼和基礎設施程式碼保留在相同的儲存庫中。此方法可讓您輕鬆維護應用程式的基礎設施。

我們建議您將應用程式劃分為三個對應至六邊形架構概念的資料夾或專案： `entrypoints`（主要轉接器）、 `domain`（網域和介面） 和 `adapters`（次要轉接器）。

下列專案結構提供在設計 API 時此方法的範例 AWS。專案會如先前所建議，在相同的儲存庫中維護應用程式碼 (app) 和基礎設施碼 (`infra`)。

```
app/  # application code
|--- adapters/  # implementation of the ports defined in the domain
```

```
     |--- tests/  # adapter unit tests
|--- entrypoints/  # primary adapters, entry points
     |--- api/  # api entry point
          |--- model/  # api model
          |--- tests/  # end to end api tests
|--- domain/  # domain to implement business logic using hexagonal architecture
     |--- command_handlers/  # handlers used to run commands on the domain
     |--- commands/  # commands on the domain
     |--- events/  # events emitted by the domain
     |--- exceptions/  # exceptions defined on the domain
     |--- model/  # domain model
     |--- ports/  # abstractions used for external communication
     |--- tests/  # domain tests
infra/  # infrastructure code
```

如前所述，網域是應用程式的核心，不依賴任何其他模組。我們建議您建構 domain 資料夾，以包含下列子資料夾：

- command handlers 包含在網域上執行命令的方法或類別。
- commands 包含命令物件，定義在網域上執行 操作所需的資訊。
- events 包含透過網域發出的事件，然後路由至其他微服務。
- exceptions 包含網域中定義的已知錯誤。
- model 包含網域實體、值物件和網域服務。
- ports 包含網域透過其與資料庫、APIs 或其他外部元件通訊的抽象概念。
- tests 包含在網域上執行的測試方法 （例如商業邏輯測試）。

主要轉接器是應用程式的進入點，由 entrypoints 資料夾表示。此範例使用 api 資料夾做為主要轉接器。此資料夾包含 API model，定義主要轉接器與用戶端通訊所需的界面。tests 資料夾包含 API end-to-end測試。這些是淺測試，可驗證應用程式的元件是否整合並協調運作。

次要轉接器，如 adapters 資料夾所示，實作網域連接埠所需的外部整合。資料庫儲存庫是次要轉接器的絕佳範例。當資料庫系統變更時，您可以使用網域定義的實作來撰寫新的轉接器。不需要變更網域或商業邏輯。tests 子資料夾包含每個轉接器的外部整合測試。

上的基礎設施範例 AWS

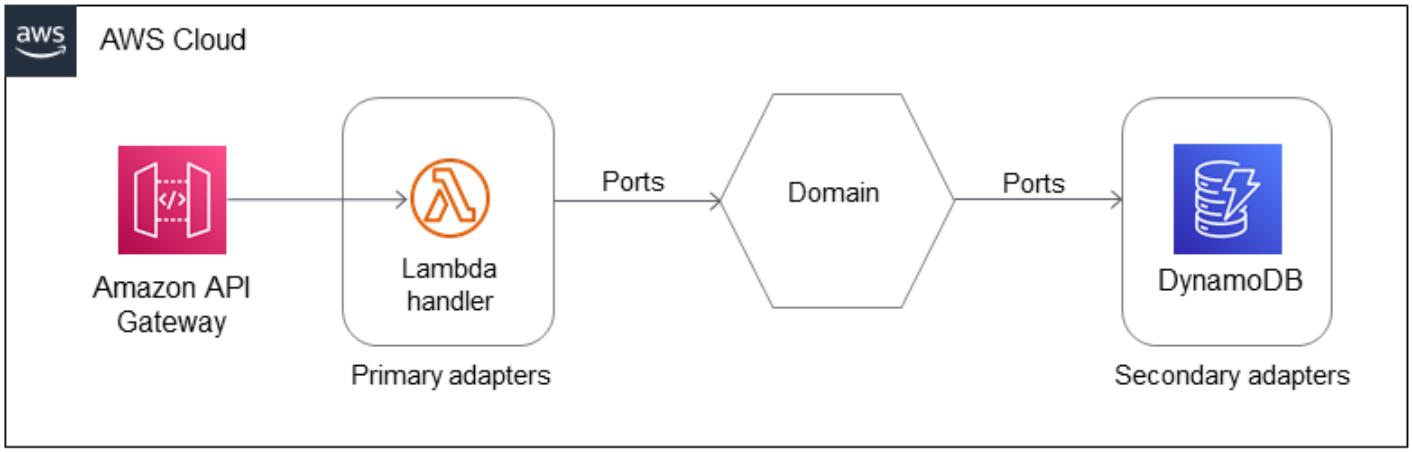
本節提供在 上 AWS 設計應用程式基礎設施的範例，可用來實作六邊形架構。我們建議您從簡單的架構開始，以建置最低可行產品 (MVP)。大多數微服務需要單一進入點來處理用戶端請求、執行程式碼的運算層，以及存放資料的持久性層。下列 AWS 服務非常適合做為六邊形架構中的用戶端、主要轉接器和次要轉接器使用：

- 用戶端：Amazon API Gateway、Amazon Simple Queue Service (Amazon SQS)、Elastic Load Balancing、Amazon EventBridge
- 主要轉接器： AWS Lambda、Amazon Elastic Container Service (Amazon ECS)、Amazon Elastic Kubernetes Service (Amazon EKS)、Amazon Elastic Compute Cloud (Amazon EC2)
- 次要轉接器：Amazon DynamoDB、Amazon Relational Database Service (Amazon RDS)、Amazon Aurora、API Gateway、Amazon SQS、Elastic Load Balancing、EventBridge、Amazon Simple Notification Service (Amazon SNS)

下列各節會更詳細地討論六邊形架構內容中的這些服務。

啟動簡單

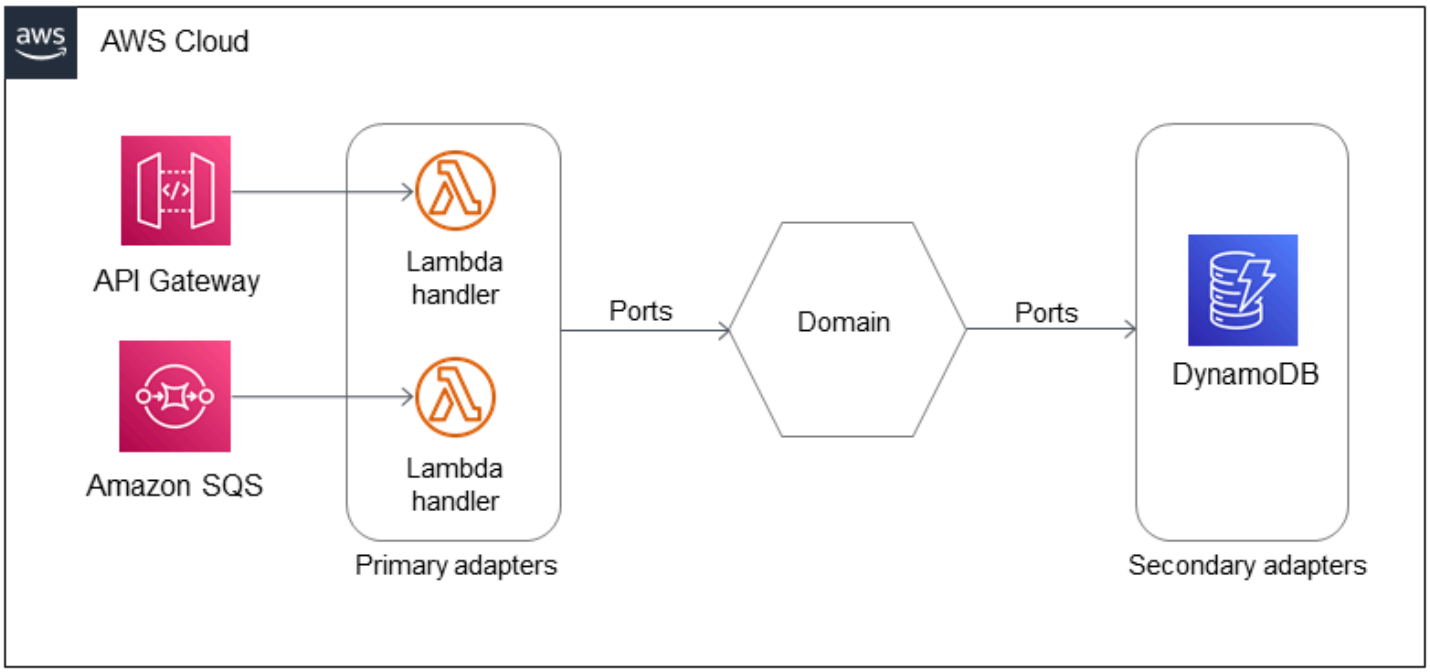
當您使用六邊形架構來建構應用程式時，建議您從簡單的 開始。在此範例中，API Gateway 用作用戶端 (REST API)，Lambda 用作主要轉接器 （運算），DynamoDB 用作次要轉接器 （持久性）。閘道用戶端會呼叫進入點，在此案例中，進入點是 Lambda 處理常式。



此架構完全無伺服器，並為架構師提供良好的起點。我們建議您在網域中使用命令模式，因為它可以更輕鬆地維護程式碼，並適應新的業務和非功能需求。此架構可能足以透過幾個操作建置簡單的微服務。

套用 CQRS 模式

如果網域上的操作數量將擴展，我們建議您切換到 CQRS 模式。您可以使用下列範例，在 中 AWS 將 CQRS 模式套用為完全無伺服器架構。

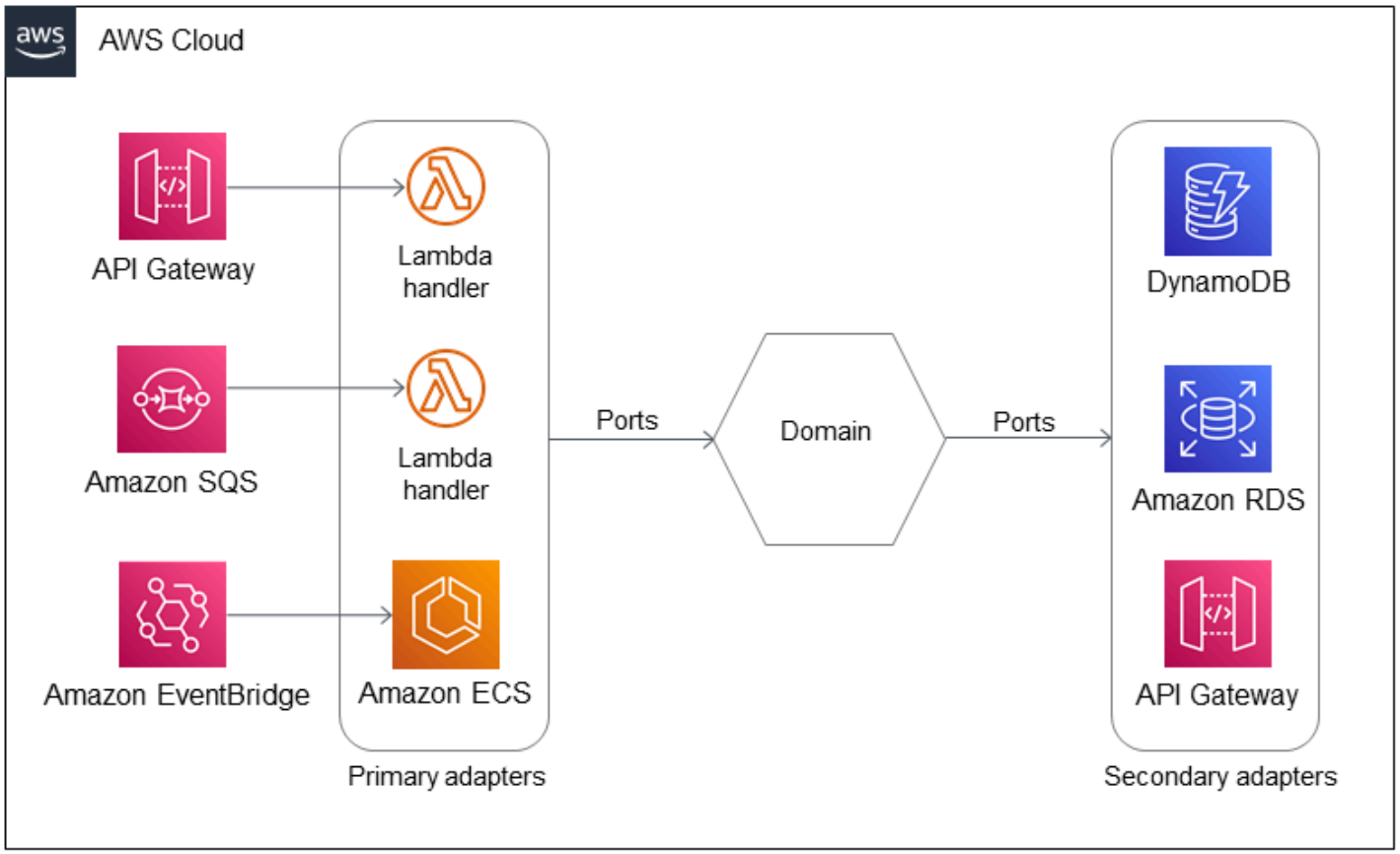


此範例使用兩個 Lambda 處理常式，一個用於查詢，另一個用於命令。使用 API 閘道做為用戶端，以同步方式執行查詢。使用 Amazon SQS 做為用戶端，以非同步方式執行命令。

此架構包含多個用戶端 (API Gateway 和 Amazon SQS) 和多個主要轉接器 (Lambda)，由其對應的進入點 (Lambda 處理常式） 呼叫。所有元件都屬於相同的邊界內容，因此它們位於相同的網域中。

透過新增容器、關聯式資料庫和外部 API 來發展架構

容器是長時間執行任務的好選項。如果您有預先定義的資料結構描述，並且想要受益於 SQL 語言的強大功能，您可能也想要使用關聯式資料庫。此外，網域必須與外部 APIs通訊。您可以發展架構範例以支援這些需求，如下圖所示。

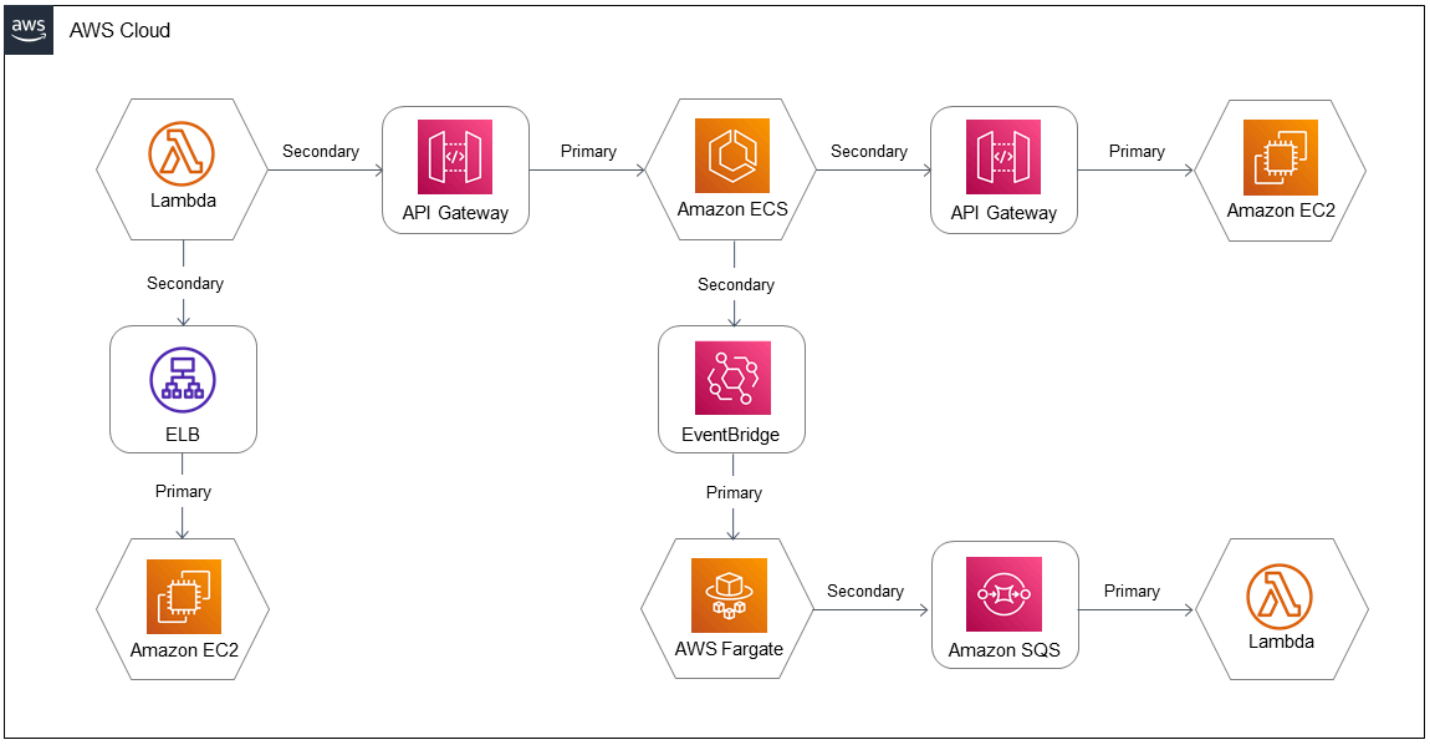


此範例使用 Amazon ECS 作為啟動網域中長時間執行任務的主要轉接器。發生特定事件時，Amazon EventBridge （用戶端） 會啟動 Amazon ECS 任務 （進入點）。架構包含 Amazon RDS 作為儲存關聯式資料的另一個次要轉接器。它也會新增另一個 API 閘道做為次要轉接器，以叫用外部 API 呼叫。因此，該架構使用多個主要和次要轉接器，這些轉接器依賴一個商業網域中的不同基礎運算層。

網域一律會透過稱為連接埠的抽象概念，鬆散地聯結所有主要和次要轉接器。網域使用連接埠定義從外部世界所需的內容。由於轉接器必須負責實作連接埠，因此從一個轉接器切換到另一個轉接器不會影響網域。例如，您可以撰寫新的轉接器，從 Amazon DynamoDB 切換到 Amazon RDS，而不會影響網域。

新增更多網域 （縮小）

六角形架構非常適合微服務架構的原則。到目前為止顯示的架構範例包含單一網域 （或邊界內容）。應用程式通常包含多個網域，這些網域需要透過主要和次要轉接器進行通訊。每個網域代表微服務，與其他網域鬆散耦合。



在此架構中，每個網域都會使用不同的一組運算環境（運算環境）。（每個網域也可能有多個運算環境，如先前範例所示。）每個網域定義了透過連接埠與其他網域通訊所需的界面。連接埠是使用主要和次要轉接器來實作。如此一來，如果轉接器發生變更，網域就不會受到影響。此外，網域會彼此分離。

在上圖中顯示的架構範例中，Lambda、Amazon EC2、Amazon ECS 和 AWS Fargate 會用作主要轉接器。API Gateway、Elastic Load Balancing、EventBridge 和 Amazon SQS 用作次要轉接器。

常見問答集

為什麼我應該使用六邊形架構？

六角形架構將開發人員的焦點轉移到網域邏輯，簡化測試自動化，並改善程式碼品質和適應性。這些改善可加快上市時間，並簡化技術和組織擴展。

為什麼我應該使用網域驅動型設計？

網域驅動設計 (DDD) 可讓您使用業務利益相關者和工程師之間的共同語言來建置軟體元件和建構。DDD 可協助您管理軟體複雜性，並且是長期維護軟體產品的有效策略。

我可以在沒有六邊形架構的情況下練習測試驅動型開發嗎？

是。測試驅動的開發 (TDD) 不限於特定軟體設計模式。不過，六邊形架構可讓您更輕鬆地練習 TDD。

我可以在沒有六邊形架構和網域驅動設計的情況下擴展產品嗎？

是。大多數的設計模式都可以實現技術和組織產品擴展。不過，六邊形架構和 DDD 可讓您更輕鬆地擴展，而且長期而言，對於大型專案更有效。

我應該使用哪些技術來實作六邊形架構？

六角形架構不限於特定技術堆疊。我們建議您選擇支援相依性反轉和單位測試的技術。

我正在開發最基本可行的產品。花時間考慮軟體架構是否合理？

是。我們建議您使用 MVPs 熟悉的設計模式。建議您嘗試並練習六邊形架構，直到工程師熟悉它為止。為新專案建立六邊形架構不需要比不使用任何架構時投入更多的時間。

我正在開發最基本可行的產品，沒有時間撰寫測試。

如果您的 MVP 包含商業邏輯，強烈建議為此撰寫自動化測試。這將減少回饋迴圈並節省時間。

我可以搭配六邊形架構使用哪些額外的設計模式？

使用 CQRS 模式來支援整體系統的擴展。使用儲存庫模式來存放和還原您的網域模型。使用工作單位模式來管理交易程序步驟。使用繼承的合成來建立網域彙總、實體和值物件的模型。請勿建置複雜的物件階層。

後續步驟

- 閱讀 Resources 區段中收集的連結，以進一步了解網域驅動的設計概念。
- 如果您要實作新專案，請使用本指南提供的專案結構範本，並實作一些功能。
- 如果您正在實作現有專案，請識別可在唯讀和唯讀操作之間分割的程式碼。將唯讀程式碼抽象至查詢服務，並將唯讀程式碼放入命令處理常式。
- 當您的基本專案結構到位時，請撰寫單位測試、建立與測試自動化的持續整合 (CI)，並遵循測試驅動的開發 (TDD) 實務。

Resources

參考

- 使用 （方案指引模式）在六邊形架構中建構 Python 專案 AWS LambdaAWS
- 敏捷團隊 （擴展敏捷架構網站）
- Python 的架構模式，作者為 Harry Percival 和 Bob Gregory (O'Reilly Media，2020 年 3 月 31 日），特別是下列章節：
  - 命令和命令處理常式
  - 命令查詢責任區隔 (CQRS)
  - 儲存庫模式
- 事件風暴：超越孤立界限的最智慧協作方法，作者為Alberto Brandolini (Event Storming 網站）
- 解開 Conway's Law，作者為 Sam Newman (Thoughtworks 網站，2014 年 6 月 30 日）
- 使用 開發演化架構 AWS Lambda，作者：James Beswick (AWS 運算部落格，2021 年 7 月 8 日）
- 域語言：在軟體核心中處理複雜性 （域語言網站）
- 外觀，從深入探討 Alexander Shvets 的設計模式 （電子書，2018 年 12 月 5 日）
- GivenWhenThen，作者：Martin Fowler (2013 年 8 月 21 日）
- 實作網域驅動設計，作者為 Vaughn Vernon (Addison-Wesley Professional，2013 年 2 月）
- Inverse Conway Maneuver (Thoughtworks 網站，2014 年 7 月 8 日）
- 本月模式：Red Green Refactor (DZone 網站，2017 年 6 月 2 日）
- SOLID 設計原則說明：程式碼範例的相依性反轉原則，作者為 Thorben Janssen (Stackify 網站，2018 年 5 月 7 日）
- SOLID 原則：說明和範例，作者：Simon Hoiberg (ITNEXT 網站，2019 年 1 月 1 日）
- 敏捷開發的藝術：測試驅動開發，作者：James Shore 和 Shane Warden (O'Reilly Media，2010 年 3 月 25 日）
- 純英文物件導向程式設計的 SOLID 原則，作者為 Yigit Kemal Erinc (freeCodeCamp 物件導向程式設計文章，2020 年 8 月 20 日）
- 什麼是事件驅動架構？(AWS 網站）

AWS 服務

- Amazon API Gateway

- Amazon Aurora
- Amazon DynamoDB
- Amazon Elastic Compute Cloud (Amazon EC2)
- Amazon Elastic Container Service (Amazon ECS)
- Amazon Elastic Kubernetes Service (Amazon EKS)
- Elastic Load Balancing
- Amazon EventBridge
- AWS Fargate
- AWS Lambda
- Amazon Relational Database Service (Amazon RDS)
- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Queue Service (Amazon SQS)

其他工具

- 摩托
- LocalStack

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
初次出版	—	2022 年 6 月 15 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

A2A Agent-to-Agent)

支援任務委派和狀態轉移的agent-to-agent協同合作的狀態通訊協定。

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，但比主動-被動遷移需要更多的工作。

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。

客服人員

一種 AI 系統，可以使用工具自動推理、規劃和採取動作來實現目標。

客服人員操作

在生產環境中大規模建置、測試、部署和執行 AI 代理器的操作實務。

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

永久刪除資料集中個人資訊的程序。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

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。

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。因此， 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)

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的工具。 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。有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。

公民開發人員

在沒有專業技術技能的情況下，使用無程式碼/低程式碼平台建立 AI 應用程式的商業使用者。

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 Enterprise Strategy 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 遷移策略相關的詳細資訊，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

正在收集和處理其資料的個人。

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要素驗證、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 成員帳戶來管理組織的帳戶，並管理該服務的許可。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 擴展了最初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

虛擬呈現真實世界的系統，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，其行為類似於文字。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。少量的提示對於需要特定格式、推理或網域知識的任務來說非常有效。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，已在廣義和未標記資料的大量資料集上進行訓練。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。

FM 閘道

集中式中介，可控制和標準化對基礎模型的存取。也稱為 LLM 閘道。

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。

護欄 (AI)

安全機制可篩選、驗證和限制代理程式輸入和輸出，以協助確保負責任且安全的 AI 行為。

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。

human-in-the-loop (HitL)

一種工作流程模式，其中代理程式執行會在關鍵決策點暫停進行人工審核和核准。

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

將基礎設施視為程式碼。

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的部署最佳實務。

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

由 Klaus Schwab 於 2016 年推出的術語，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。

MCP

請參閱模型內容通訊協定。

模型內容通訊協定 (MCP)

適用於代理程式對工具通訊的無狀態通訊協定。

MCP 伺服器

透過模型內容通訊協定公開一或多個工具的服務。

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，請參閱本詞彙表中的 7 個 Rs 項目，並請參閱動員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 建議使用不可變的基礎設施作為最佳實務。

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

請參閱開啟程序通訊 - 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 中的操作準備度審查 (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、使用 AWS KMS (SSE-KMS) 的伺服器端加密 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。

設計隱私權

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)

產品整個生命週期的資料和程序管理，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 提示的輸出作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

將資料集中的個人識別符取代為預留位置值的程序。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，以改善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

請參閱擷取增強生成。

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 個 R。

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 個 R。

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 個 R。

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 個 R。

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

矩陣，定義所有參與遷移活動和雲端操作之各方的角色和責任。矩陣名稱衍生自矩陣中定義的責任類型：負責人 (R)、責任 (A)、已諮詢 (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 個 R。

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。

斯卡達

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 秘密中的內容？ 在 Secrets Manager 文件中。

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 AWS 服務 接收資料的 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)

服務效能層面的測量，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。

陰影 AI

在組織內受管管道之外建置或使用的未授權 AI 應用程式。

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

做為中繼資料的鍵值對，用於組織您的 AWS 資源。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。

tool

代理程式可以叫用以在外部系統中執行操作的函數或 API。

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。

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，通常可接受中等緩慢的查詢。

視窗函數

SQL 函數，會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瑕疵或漏洞。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。