在 上建置六邊形架構 AWS

AWS 規範指南

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS 規範指南: 在 上建置六邊形架構 AWS

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

Table of Contents

簡介 1
概要 2
領域驅動設計 (DDD) 2
六角形 2
目標性業務 3
改善開發週期 5
雲端內部的測試 5
本機測試 5
並行化的發展 5
產品上市時間 6
設計品質 7
本地化變更並提高可讀性 7
首先測試業務邏輯 7
可維護性 8
適應變化 9
使用連接埠和介面卡，適應新的非功能性需求 9
使用命令和命令處理常式來適應新的業務需求 9
使用服務外觀或 CQRS 模式來解耦元件 10
織檢視 10
最佳實務 12
建立商業領域的模型 12
從頭開始編寫並運行測試 12
定義域的行為 12
自動化測試和部署 13
使用微服務和 CQRS 來擴展您的產品 13
設計映射到六角形建築概念的項目結構 13
基礎設施例 15
簡單開始 15
套用 CQRS 模式 16
透過新增容器、關聯式資料庫和外部 API 來進化架構 16
新增更多網域 (縮小) 17
常見問答集 19
為什麼要使用六角形體系結構？ 19
為什麼要使用領域驅動式設計？ 19

我可以在沒有六角形體系結構的情況下練習測試驅動 19
我可以在沒有六角形架構和領域驅動設計的情況下擴展產品嗎？ 19
我應該使用哪些技術來實現六角形體系結構？ 19
我正在開發一個最低可行的產品。花時間思考軟件架構是否有意義？ 19
我正在開發一個最低限度的可行產品，沒有時間編寫測試。 19
我可以在六角形架構中使用哪些額外的設計模式？ 20
後續步驟 21
資源 22
文件歷史紀錄 24
詞彙表 25
..... 25
A 25
B 28
C 29
D 32
E 35
F 37
G 38
H 38
I 39
L 41
M 42
O 46
P 48
Q 50
R 50
S 53
T 55
U 57
V 57
W 57
Z 58
..... lix

建立六角形架構AWS

Furkan Oruc，多米尼克虎比，大流士昆斯和米哈爾普洛斯基，Amazon Web Services（AWS）

2022 年 6 月 (文件歷史記錄)

本指南介紹了用於開發軟件架構的心理模型和模式集合。隨著產品採用率的增長，這些架構很容易在整個組織中維護、擴展和擴展。像 Amazon Web Services (AWS) 這樣的雲端超大規模工具可為小型和大型企業提供建置區塊，以進行創新和建立新的軟體產品。這些新服務和功能推出的快速步伐使業務利益相關者期望他們的開發團隊能夠更快地製作新的最低可行產品 (MVP) 的原型，以便能夠盡快測試和驗證新的想法。這些微軟最有價值專家通常會被採納並成為企業軟體生態系統的一部分。在產生這些微軟最有價值專家的過程中，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。他們認為這種方法將加快開發速度並縮短上市時間。但是，如果他們無法在各個層面上創建軟件架構的基礎模型和框架，那麼為產品開發新功能將是困難甚至不可能的。缺乏確定性和不斷變化的需求也會在開發過程中減慢團隊的速度。

本指南將逐步介紹建議的軟體架構，從低階六角形架構到高階架構和組織分解，該架構使用領域驅動設計 (DDD) 來解決這些挑戰。DDD 有助於管理業務複雜性，並隨著新功能的開發擴展工程團隊。它通過使用普遍的語言使業務和技術利益相關者與業務問題（稱為域）保持一致。六角體系結構是這種方法在一個非常特定的領域（稱為有界上下文）中的技術推動者。有界的背景是業務問題的高度凝聚力和鬆散耦合的子區域。我們建議您針對所有企業軟體專案採用六角形架構，無論其複雜程度如何。

六角架構鼓勵工程團隊首先解決業務問題，而經典的分層架構將工程重點從領域轉移到解決技術問題。此外，如果軟體遵循六角形架構，則更容易採用測試驅動的開發方法，從而減少了開發人員測試業務需求所需的反饋循環。最後，使用命令和命令處理程序是一種應用 SOLID 中的單一責任和開放式原則的方法。遵循這些原則會產生程式碼庫，讓處理專案的開發人員和架構師可以輕鬆瀏覽和理解，並降低對現有功能引入重大變更的風險。

本指南適用於有興趣了解在軟體開發專案中採用六角形架構和 DDD 的好處的軟體架構師和開發人員。它包括為您的應用程序設計支持六角形架構的AWS基礎結構的示例。如需實作範例，請參閱 AWS Prescriptive Guidance 網站AWS Lambda上的使用在六角形架構中建構 Python 專案。

概要

領域驅動設計 (DDD)

在領域驅動設計（DDD）中，域是軟件系統的核心。在開發任何其他模塊之前，首先定義域模型，並且不依賴於其他低級模塊。相反，模塊（例如數據庫，表示層和外部 API）都取決於域。

在 DDD 中，架構師通過使用基於業務邏輯的分解而不是技術分解將解決方案分解為有界的上下文。本節將討論此方法的好目標性業務處。

當團隊使用六角形體系結構時，DDD 更容易實現。在六角形架構中，應用程序核心是應用程序的中心。它通過端口和適配器與其他模塊分離，並且與其他模塊沒有依賴關係。這與 DDD 完全一致，其中域是解決業務問題的應用程序的核心。本指南提出了一種方法，您可以將六角形架構的核心模型建立為有界前後關聯的領域模型。下一節將更詳細地介紹六角形架構。

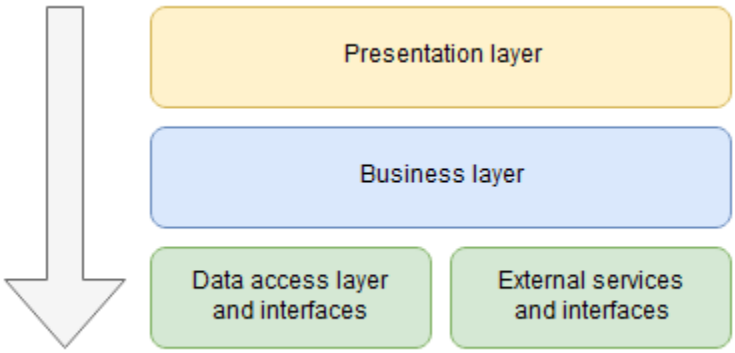
本指南不涵蓋 DDD 的所有方面，這是一個非常廣泛的主題。若要進一步瞭解，您可以檢閱網域語言網站上列出的 DDD 資源。

六角形

六角形架構，也稱為端口和適配器或洋蔥架構，是管理軟件項目中依賴反轉的原則。Hexagonal 架構在開發軟體時強調核心領域業務邏輯，並將外部整合點視為次要。Hexagonal 架構可幫助軟件工程師採用諸如測試驅動開發（TDD）之類的良好實踐，這反過來又可以促進架構演進，並幫助您長期管理複雜的域。

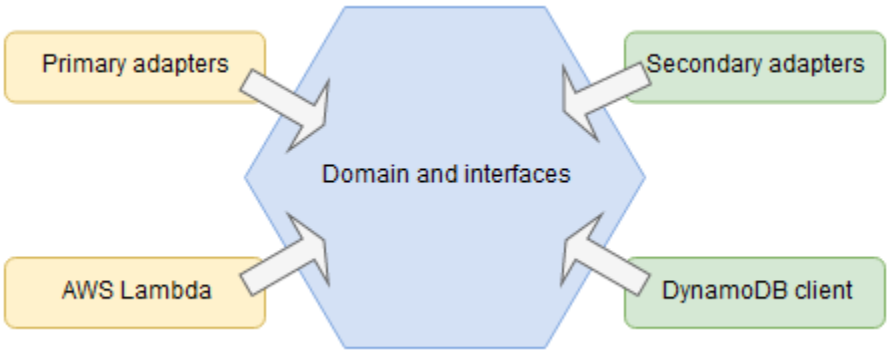
讓我們將六角形架構與經典分層架構進行比較，這是建模結構化軟件項目的最受歡迎的選擇。這兩種方法之間存在微妙但強大的差異。

在分層架構中，軟件項目是分層結構的，這代表了廣泛的問題，如業務邏輯或表示邏輯。這種體系結構使用依賴層次結構，其中頂層與它們下面的層具有依賴關係，但不相反。在下圖中，表示層負責用戶交互，因此它包括用戶界面，API，命令行界面和類似的組件。表示層具有業務層，它實現域邏輯的依賴關係。業務層，反過來，依賴於數據訪問層和多個外部服務。



這種配置的主要缺點是依賴結構。例如，如果用於在數據庫中存儲數據的模型發生變化，這會影響數據訪問接口。對數據模型的任何更改也會影響業務層，該業務層對數據訪問接口具有依賴性。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結構變更。反過來，這增加了回歸錯誤的可能性。

下圖所示。它集中圍繞域業務邏輯，它定義了所有接口的決策。外部組件通過稱為端口的接口與業務邏輯進行交互。端口是定義域與外部世界的交互的抽象。每個基礎結構元件都必須實作這些連接埠，因此這些元件中的變更不會再影響核心網域邏輯。



周圍的組件稱為適配器。適配器是外部世界和內部世界之間的代理，並實現在域中定義的端口。介面卡可以分為兩個群組：主要和次要。主要介面卡是軟體元件的進入點。它們允許外部參與者，用戶和服務與核心邏輯進行交互。 AWS Lambda是主要介面卡的一個很好的範例。它與多個AWS服務整合，這些服務可以叫用 Lambda 函數作為進入點。次要介面卡是處理與外部世界通訊的外部服務程式庫包裝函式。用於資料存取的 Amazon DynamoDB 用戶端就是次要介面卡的一個很好的範例。

目標性業務

本指南中討論的六角形架構可協助您達成下列目標：

- 改善開發週期，縮短上市時間

- 改善軟體品質
- 更容易適應改變

下幾節節節節節節節節詳細討論。

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，因為要在開發機器上本機複製執行階段環境非常困難。驗證軟件的一種直接方法是將其部署到雲中並在那裡進行測試。但是，這種方法涉及冗長的回饋週期，尤其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，進而大幅縮短上市時間。

雲端內部的測試

直接在雲端進行測試是確保架構元件 (例如 Amazon API Gateway 閘道中的閘道、AWS Lambda函數、Amazon DynamoDB 表格和AWS Identity and Access Management (IAM) 許可等設定正確的唯一方法。這也可能是測試組件集成的唯一可靠方法。雖然某些AWS服務 (例如 DynamoDB) 可以在本機部署，但大多數服務無法在本機設定中複寫。同時，諸如 Moto 和用於測試目的的的模擬AWS服務LocalStack之類的第三方工具可能無法準確反映真實的服務 API 合同，或者功能的數量可能會受到限制。

但是，企業軟件中最複雜的部分是業務邏輯，而不是在雲架構中。架構的變更頻率低於網域，因為這必須符合新的業務需求。因此，在雲端中測試商務邏輯會變成一個密集的程序，包括在程式碼中進行變更、啟動部署、等待環境準備就緒，以及驗證變更。如果部署只需 5 分鐘，則在商務邏輯中進行和測試 10 個變更將需要一個小時或更長時間。如果商務邏輯比較複雜，測試可能需要幾天的時間等待部署完成。如果您的團隊中有多個功能和工程師，那麼延長的時間很快就會對業務變得明顯。

本機測試

六角形架構可幫助開發人員專注於領域，而不是基礎架構技術。這種方法使用本地測試（您選擇的開發框架中的單元測試工具）來涵蓋域邏輯要求。您不必花時間解決技術整合問題，或將軟體部署到雲端來測試商務邏輯。您可以在本地運行單元測試，並將反饋循環從幾分鐘縮短為幾秒鐘。如果部署需要 5 分鐘，但單元測試在 5 秒內完成，則可大幅減少偵測錯誤所需的時間。本指南後首先測試業務邏輯面的章節將詳細介紹此方法。

並行化的發展

六角形架構方法可讓開發團隊平行化開發工作。開發人員可以單獨設計和實現服務的不同組件。通過隔離每個組件和每個組件之間定義的接口，可以實現這種並行化。

產品上市時間

本機單元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，尤其是當這些產品包含複雜的商務邏輯時，如先前所述。此外，單元測試提高程式碼涵蓋率，可大幅降低更新或重構程式碼基底時引入迴歸錯誤的風險。單元測試涵蓋範圍還使您能夠持續重構代碼庫，以保持其良好的組織，從而加快新工程師的入職過程。這將在本設計品質節中進一步討論。最後，如果業務邏輯是很好的隔離和測試，它使開發人員能夠快速適應不斷變化的功能和非功能性需求。本適應變化節將進一步說明這一點。

設計品質

採用六角形架構有助於從項目開始提高代碼庫的質量。建置可協助您從一開始就符合預期品質需求的程序，而不會拖慢開發流程是非常重要的。

本地化變更並提高可讀性

使用六角形架構方法可讓開發人員變更某個類別或組件中的程式碼，而不會影響其他類別或元件。這種設計促進了開發組件的凝聚力。透過將網域與介面卡解耦並使用已知的介面，您可以增加程式碼的可讀性。它變得更容易識別問題和角落案例。

這種方法還有助於在開發過程中進行代碼審查，並限制引入未檢測到的更改或技術債務。

首先測試業務邏輯

本地測試可以通過引入 end-to-end，集成和單元測試到項目來完成。End-to-end 測試涵蓋了整個傳入的請求生命週期。他們通常調用應用程序進入點和測試，看看它是否已經完成了業務需求。每個軟件項目應至少有一個使用已知輸入並產生預期輸出的測試場景。不過，新增更多轉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入口點傳送要求 (例如，透過 REST API 或佇列)、經過商務動作所需的所有整合點，然後宣告結果。設置測試場景的環境和斷言結果可能需要大量的開發人員的時間。

在六角形架構中，您可以隔離測試商務邏輯，並使用整合測試來測試次要介面卡。您可以在業務邏輯測試中使用模擬或偽造的適配器。您也可以將商業使用案例的測試與網域模型的單元測試結合在一起，以保持高涵蓋範圍和低耦合。建議您先測試，建議您先測試，建議您先測試。相反地，他們應該驗證次要介面卡是否正確呼叫外部服務。

理想情況下，您可以使用測試驅動開發（TDD），並在開發一開始使用適當的測試開始定義域實體或業務用例。首先編寫測試可幫助您創建域所需接口的模擬實現。當測試成功並滿足網域邏輯規則時，您可以實作實際的介面卡，並將軟體部署到測試環境。在這一點上，您的域邏輯的實現可能不理想。然後，您可以通過引入設計模式或一般重新排列代碼來重構現有體系結構以進化它。通過使用這種方法，您可以避免引入回歸錯誤，並且可以隨著項目的增長改進架構。透過將此方法與您在持續整合程序中執行的自動測試相結合，您可以在潛在錯誤上線之前減少潛在錯誤的數量。

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。完成這些實施步驟後，我們建議您在每次推送到存放庫的新變更時自動進行測試。

可維護性

可維護性是指操作和監視應用程序，以確保應用程序滿足所有需求並最大程度地減少系統故障的可能性。為了使系統可操作，您必須將其適應 future 的流量或操作需求。您還必須確保它可用且易於部署，而且對用戶端的影響最小或沒有任何影響。

要了解系統的當前和歷史狀態，您必須使其可觀察。您可以通過提供特定的指標，日誌和跟踪來實現這一點，操作員可以使用這些指標，以確保系統按預期運行並跟踪錯誤。這些機制還應該允許操作員進行根本原因分析，而無需登錄到機器並讀取代碼。

六角形架構旨在提高 Web 應用程序的可維護性，以便您的代碼整體需要更少的工作。藉由分離模組、本地化變更，以及將應用程式商務邏輯與介面卡實作解耦，您可以產生指標和記錄，協助操作員深入瞭解系統，並瞭解對主要或次要介面卡所做的特定變更範圍。

適應變化

軟件系統往往變得複雜。造成這種情況的原因之一可能是頻繁變化的業務需求和很少的時間來相應地適應的軟件體系結構。另一個原因可能是在項目開始時建立軟件架構以適應頻繁變化的投資不足。無論是什麼原因，軟件系統都可能會變得複雜到幾乎不可能進行更改的程度。因此，從項目開始構建可維護的軟件體系結構非常重要。良好的軟件架構可以輕鬆適應變化。

本節介紹如何通過使用容易適應非功能性或業務需求的六角形架構來設計可維護的應用程序。

使用連接埠和介面卡，適應新的非功能性需求

作為應用程序的核心，域模型定義了從外部世界滿足業務需求所需的操作。這些動作是通過抽象，這被稱為端口定義。這些連接埠由不同的介面卡實作。每個介面卡都負責與另一個系統的互動。例如，您可能有一個適配器用於資料庫儲存庫，另一個適配器可與第三方 API 互動。網域並不知道介面卡的實作，因此很容易用另一個介面卡取代一個介面卡。例如，應用程式可能會從 SQL 資料庫切換到 NoSQL 資料庫。在這種情況下，必須開發一個新的適配器來實現由域模型定義的端口。該域對數據庫存庫沒有依賴關係，並使用抽象進行交互，因此不需要更改域模型中的任何內容。因此，六角形架構輕鬆適應非功能性要求。

使用命令和命令處理常式來適應新的業務需求

在傳統的分層架構中，域取決於持久層。如果您想要變更網域，您也必須變更持續性層。相比之下，在六角形體系結構中，域不依賴於軟件中的其他模塊。該域是應用程序的核心，所有其他模塊（端口和適配器）都取決於域模型。該域使用依賴反轉原則通過端口與外部世界進行通信。依賴反轉的好處是，您可以自由更改域模型，而不必害怕破壞代碼的其他部分。因為網域模型反映了您嘗試解決的商業問題，因此更新網域模型以適應不斷變化的業務需求並不是問題。

當你開發軟件時，關注點的分離是一個重要的原則要遵循。若要實現此分隔，您可以使用稍微修改的指令模式。這是一種行為設計模式，其中完成操作所需的所有信息都封裝在命令對象中。然後，命令處理常式會處理這些作業。命令處理常式是接收命令、改變網域狀態，然後將回應傳回給呼叫者的方法。您可以使用不同的用戶端 (例如同步 API 或非同步佇列) 來執行命令。建議您針對網域上的每項作業使用命令和命令處理常式。透過遵循這個方法，您可以透過引入新的命令和命令處理常式來新增功能，而不需要變更現有的商務邏輯。因此，使用命令模式可以更輕鬆地適應新的業務需求。

使用服務外觀或 CQRS 模式來解耦元件

在六角形架構中，主要介面卡負責將來自用戶端的傳入讀取和寫入要求鬆散耦合到網域。有兩種方法可以實現這種鬆散耦合：使用服務外觀模式或使用命令查詢責任隔離（CQRS）模式。

服務外觀模式提供前端介面，為客戶提供服務，例如表示層或微服務。服務外觀為用戶端提供數個讀取和寫入作業。它負責將傳入的請求傳輸到域，並將從域接收到的響應映射到客戶端。對於具有單一責任的微服務而言，使用服務外觀很容易進行多項操作。但是，在使用服務外觀時，很難遵循單一責任和開放式封閉原則。單一責任原則指出，每個模組都應該對軟體的單一功能負責。開閉原則指出，代碼應該是開放的擴展和關閉進行修改。隨著服務外觀的擴展，所有操作都收集在一個接口中，更多的依賴關係被封裝到其中，更多的開發人員開始修改相同的外觀。因此，我們建議您使用服務外觀，只有在很明顯的情況下，該服務在開發過程中不會延長很多時間。

在六角形架構中實作主要配接器的另一種方法是使用 CQRS 模式，該模式會使用查詢和命令來分隔讀取和寫入操作。如前所述，命令是包含變更網域狀態所需之所有資訊的物件。命令是由命令處理常式方法執行的。另一方面，查詢不會改變系統的狀態。他們唯一的目的是將數據返回給客戶。在 CQRS 模式中，命令和查詢在單獨的模塊中實現。這對於遵循事件驅動架構的專案尤其有利，因為命令可以實作為以非同步方式處理的事件，而查詢則可以使用 API 同步執行。查詢也可以使用針對其進行最佳化的不同資料庫。CQRS 模式的缺點是，實施比服務外觀需要更多的時間。我們建議您對計劃長期擴展和維護的專案使用 CQRS 模式。命令和查詢提供了一種有效的機制，用於應用單一責任原則和開發鬆散耦合的軟件，尤其是在大型項目中。

從長遠來看，CQRS 有很大的好處，但需要初始投資。因此，建議在決定使用 CQRS 模式之前仔細評估您的專案。但是，您可以直接從一開始就使用命令和命令處理常式來構建應用程式，而不需分隔讀取/寫入作業。如果您決定稍後採用該方法，這將幫助您輕鬆地為 CQRS 重構項目。

織檢視

六角形架構、領域導向設計和 (選擇性) CQRS 的組合，可讓您的組織快速擴充產品規模。根據康威定律，軟件架構傾向於發展以反映公司的通信結構。這種觀察歷史上有負面的含義，因為大型組織通常根據技術專業知識（例如數據庫，企業服務總線等）來構建團隊。這種方法的問題在於，產品和功能開發始終涉及交叉問題，例如安全性和可擴展性，這需要團隊之間的持續溝通。基於技術特徵的組織結構化團隊會在組織中創建不必要的孤島，從而導致溝通不良，缺乏所有權並失去了大局。最終，這些組織問題反映在軟件架構中。

另一方面，逆康威機動定義基於促進軟件體系結構的領域的組織結構。例如，跨職能專案團隊會負責一組特定的有界前後關聯，這些前後關聯是使用 DDD 和事件風暴來識別。這些有界的前後關聯可能會反映產品的非常特定的功能。例如，客戶小組可能負責付款內容。每個新功能都會分配給一個具有高度

凝聚力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，並縮短上市時間。可以根據圖徵的複雜性調整團隊的比例，因此可以將複雜的特徵指定給更多的工程師。

最佳實務

建立商業領域的模型

從業務領域回到軟件設計，以確保您正在編寫的軟件符合業務需求。

使用領域驅動設計 (DDD) 方法，例如事件風暴來建立商業領域的模型。事件風暴具有靈活的研討會形式。在研討會期間，領域和軟件專家共同探討業務領域的複雜性。軟體專家使用研討會的交付項目，開始軟體元件的設計和開發程序。

從頭開始編寫並運行測試

使用測試驅動開發（TDD）來驗證您正在開發的軟件的正確性。TDD 在單元測試級別效果最好。開發人員通過首先編寫一個測試來設計軟件組件，該測試調用該組件。該組件一開始沒有實現，因此測試失敗。作為下一步，開發人員實現組件的功能，使用帶有模擬對象的測試夾具來模擬外部依賴關係或端口的行為。當測試成功時，開發人員可以繼續實作真正的介面卡。這種方法提高了軟件質量並導致更易讀的代碼，因為開發人員了解用戶將如何使用這些組件。六角形架構通過分離應用程序核心來支持 TDD 方法。開發人員撰寫專注於網域核心行為的單元測試。他們不必編寫複雜的適配器來運行他們的測試;相反，他們可以使用簡單的模擬對象和固定裝置。

使用行為驅動開發（BDD）來確保在功能級別 end-to-end 接受。在 BDD 中，開發人員定義功能的場景，並與業務利益相關者進行驗證。BDD 測試使用盡可能多的自然語言來實現這一目標。六角形架構以其主要和次要介面卡的概念來支援 BDD 方法。開發人員可以建立可在本機執行的主要和次要介面卡，無需呼叫外部服務。他們將 BDD 測試套件設定為使用本機主要介面卡來執行應用程式。

在持續集成管道中自動運行每個測試，以不斷評估系統的質量。

定義域的行為

將網域分解為實體、值物件和彙總 (請參閱有關實作領域導向設計的資訊)，並定義其行為。實現域的行為，以便在項目開始時編寫的測試成功。定義呼叫網域物件行為的命令。定義網域物件完成行為後所發出的事件。

定義介面卡可用來與網域互動的介面。

自動化測試和部署

在初始概念證明之後，我們建議您投入時間實施實 DevOps踐。例如，持續整合與持續交付 (CI/CD) 管線和動態測試環境可協助您維持程式碼品質，並避免部署期間發生錯誤。

- 在 CI 進程中運行單元測試，並在合併之前測試您的代碼。
- 建置 CD 程序，將應用程式部署至靜態開發/測試環境，或是動態建立的支援自動整合與 end-to-end 測試的環境。
- 將專用環境的部署程序自動化。

使用微服務和 CQRS 來擴展您的產品

如果您的產品成功，請將您的軟體專案分解為微服務來擴展產品規模。利用六角形架構提供的可攜性來提高性能。將查詢服務和命令處理常式拆分為單獨的同步和非同步 API。考慮採用命令查詢責任隔離 (CQRS) 模式和事件驅動架構。

如果您收到許多新功能要求，請考慮根據 DDD 模式調整組織的規模。如同先前在織檢視本節中所討論的，使其擁有一或多個特徵作為有界前後關聯的方式來建構您的專案團隊。然後，這些團隊可以通過使用六角形體系結構實現業務邏輯。

設計映射到六角形建築概念的項目結構

基礎架構即程式碼 (IaC) 是雲端開發中廣泛採用的做法。它可讓您將基礎結構資源 (例如網路、負載平衡器、虛擬機器和閘道) 定義和維護為原始程式碼。如此一來，您就可以使用版本控制系統來追蹤架構的所有變更。此外，您可以建立和移動基礎結構輕鬆進行測試。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結構程式碼保存在相同的儲存庫中。此方法可讓您輕鬆維護應用程式的基礎架構。

我們建議您將應用程式分成三個資料夾或專案，這些資料夾或專案會對應到六角形架構的概念：`entrypoints`(主要介面卡)、`domain` (網域和介面) 和`adapters` (次要介面卡)。

下列專案結構提供了在上設計 API 時此方法的範例AWS。該項目將應用程序代碼（ `app` ）和基礎結構代碼（ `infra` ）保存在同一個存儲庫中，如前所述。

```
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，域是應用程序的核心，不依賴於任何其他模塊。建議您建立domain資料夾的結構，使其包含下列子資料夾：

- command handlers包含在網域上執行命令的方法或類別。
- commands包含定義在網域上執行作業所需資訊的命令物件。
- events包含透過網域發出，然後路由至其他微服務的事件。
- exceptions包含網域內定義的已知錯誤。
- model包含網域實體、值物件和網域服務。
- ports包含網域與資料庫、API 或其他外部元件進行通訊的抽象概念。
- tests包含在網域上執行的測試方法 (例如商務邏輯測試)。

主要介面卡是應用程式的進入點，如entrypoints資料夾所示。此範例使用資api料夾做為主要轉接器。此資料夾包含 APImodel，用來定義主要介面卡與用戶端通訊所需的介面。該tests文件夾包含對 API 的 end-to-end 測試。這些是淺層測試，用於驗證應用程序的組件是否集成並協調工作。

次要配接器 (如adapters資料夾所表示) 會實作網域連接埠所需的外部整合。資料庫儲存庫是次要介面卡的一個很好的範例。當資料庫系統變更時，您可以使用網域所定義的實作來寫入新的配接器。不需要變更網域或業務邏輯。tests子資料夾包含每個介面卡的外部整合測試。

基礎架構範例AWS

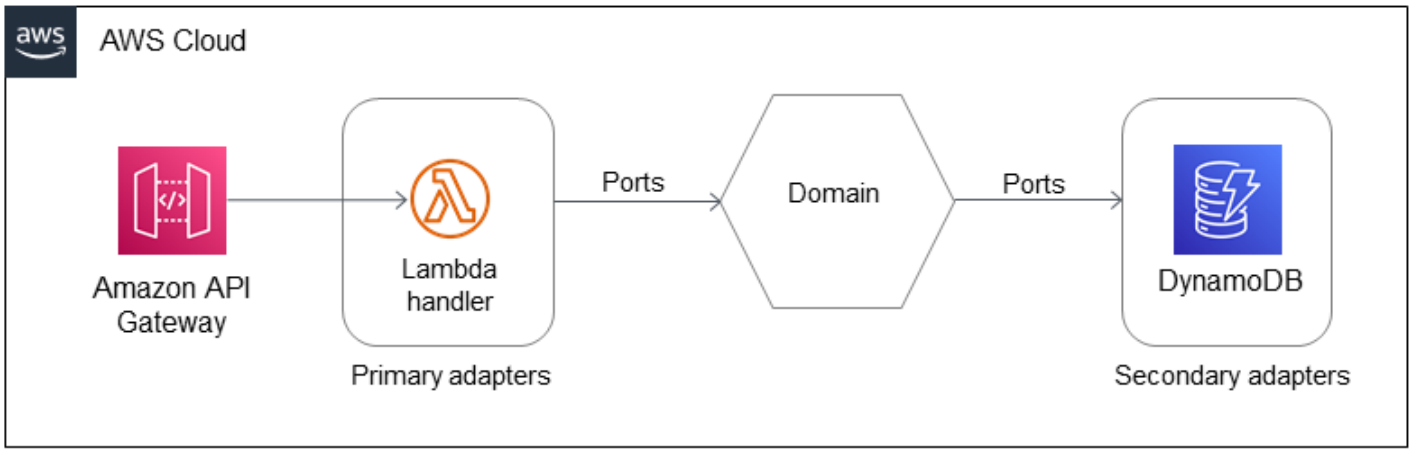
本節提供設計應用程式之基礎結構的範例，您可以使用這AWS些基礎結構來實作六角形架構。我們建議您從簡單的架構開始，以建置最低限度的可行產品 (MVP)。大多數微服務都需要單一進入點來處理用戶端要求、執行程式碼的計算層，以及用來儲存資料的持續性層。以下AWS服務是在六角形架構中用作用戶端、主要介面卡和次要配接器的絕佳選擇：

- 客戶：Amazon API Gateway、Amazon Simple Queue Service (Amazon SQS)、Elastic Load Balancing、Amazon Simple Queue Service 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)、Amazon Elastic Container Service (Amazon EC2)、Amazon Elastic Contain
- 次要轉接器:Amazon DynamoDB、Amazon Amazon 關係資料庫 Service (Amazon RDS)、Amazon Simple Notification Service (Amazon Aurora SNS)、Elastic Load Balancing、Amazon SQS imple Notification Service (Amazon SNS) EventBridge、Elastic Load Balancing、Amazon Simple Notification Service (Amazon SNS)

以下章節會帶您深入詳細了解這些服務在六邊形架構的背景下討論。

簡單開始

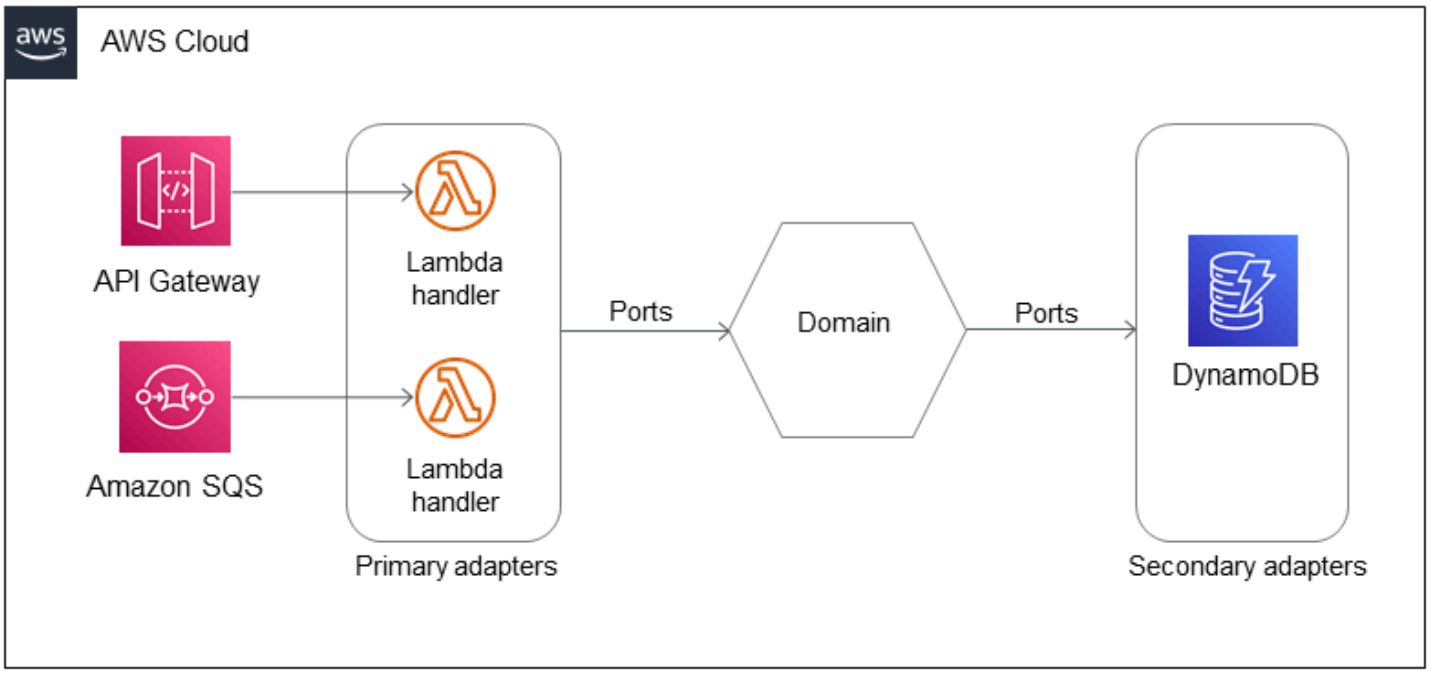
我們建議您在使用六角形架構設計應用程式時，從簡單開始。在此範例中，API Gateway 用作用戶端 (REST API)，Lambda 用作主要介面卡 (運算)，而 DynamoDB 則用作次要介面卡 (持續性)。閘道用戶端呼叫入口點，在此情況下，該入口點是 Lambda 處理常式。



這種架構是完全無伺服器的，並為建築師提供了一個很好的起點。我們建議您使用網域中的命令模式，因為它可以讓程式碼更容易維護，而且可以適應新的業務和非功能性需求。這種架構可能足以通過一些操作構建簡單的微服務。

套用 CQRS 模式

如果網域上的作業數目要擴充，建議您切換至 CQRS 模式。您可以使用下列範例，在AWS中套用 CQRS 模式做為完全無伺服器架構。

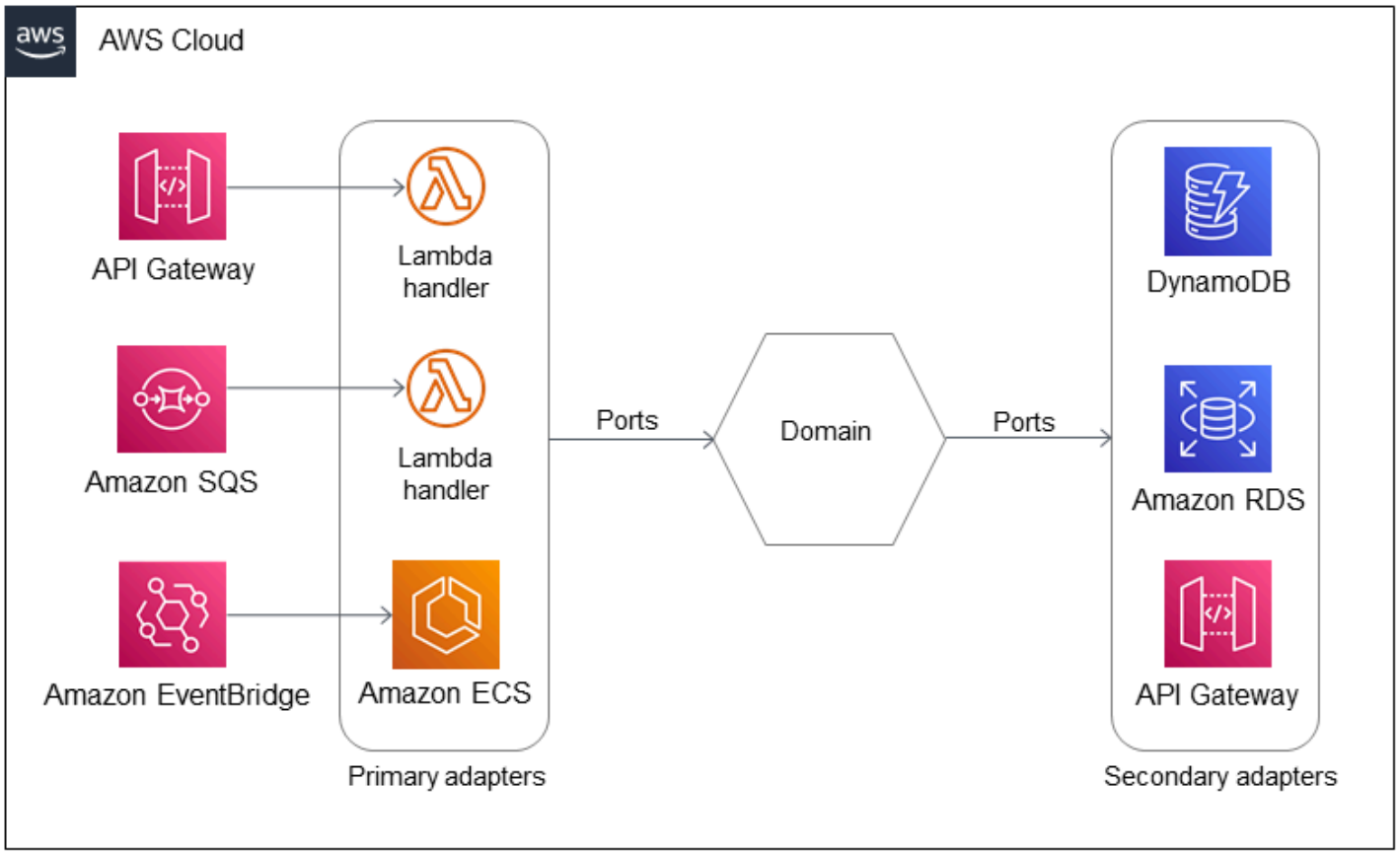


此範例使用兩個 Lambda 處理常式，一個用於查詢，另一個用於命令。使用 API 閘道做為用戶端，以同步方式執行查詢。使用 Amazon SQS 做為用戶端，以非同步方式執行命令。

此架構包括多個用戶端 (API Gateway 和 Amazon SQS) 和多個主要配接器 (Lambda)，這些用戶端由其對應的進入點 (Lambda 處理常式) 呼叫。所有元件都屬於相同的限制前後關聯，因此它們位於同一個領域內。

透過新增容器、關聯式資料庫和外部 API 來進化架構

容器是長時間執行工作的好選擇。如果您有預先定義的資料結構描述，並且想要從 SQL 語言的強大功能中受益，您可能也會想要使用關聯式資料庫。此外，網域必須與外部 API 進行通訊。您可以發展架構示例以支援這些需求，如下圖所示。

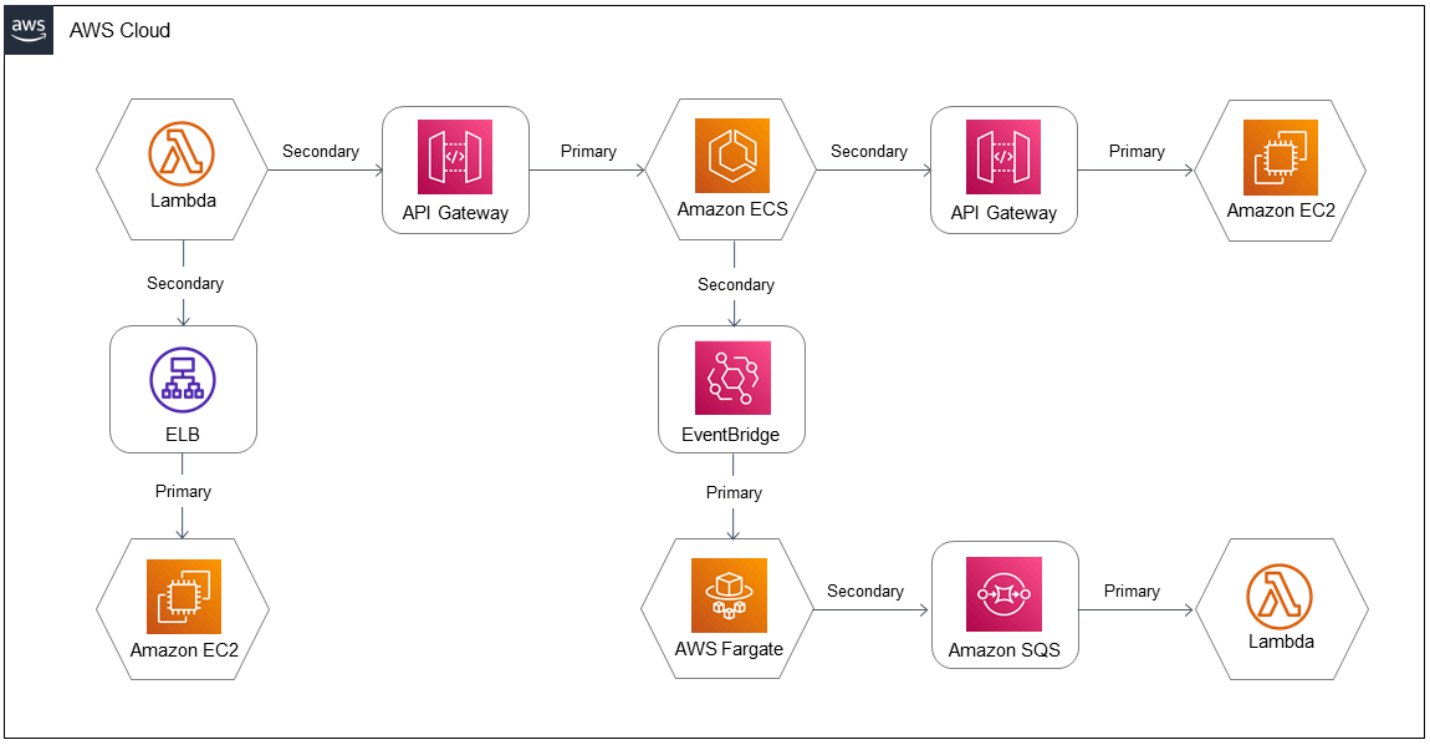


此範例使用 Amazon ECS 做為在網域中啟動長時間執行任務的主要介面卡。亞馬遜 EventBridge （客戶端）在發生特定事件時啟動 Amazon ECS 任務（入口點）。該架構包括 Amazon RDS 作為另一個用於存放關聯式資料的次要介面卡。它還添加了另一個 API 網關作為調用外部 API 調用的輔助適配器。因此，該架構使用多個主要和次要介面卡，這些介面卡依賴於單一業務網域中的不同基礎運算層。

該域始終通過稱為端口的抽象與所有主要和次要適配器鬆散耦合。該域定義了它通過使用端口從外部世界需要什麼。由於介面卡有責任實作連接埠，因此從一個介面卡切換到另一個介面卡並不會影響網域。例如，您可以透過撰寫新的介面卡從 Amazon DynamoDB 切換到 Amazon RDS，而不會影響網域。

新增更多網域 (縮小)

六角形架構與微服務架構的原則很好地對齊。到目前為止所示的架構範例包含單一網域 (或有界前後關聯)。應用程式通常包含多個網域，需要透過主要和次要介面卡進行通訊。每個域代表一個微服務，並鬆散地與其他域結合。



在此架構中，每個網域都使用不同的運算環境集。(每個網域也可能有多個運算環境，如前面的範例所示。) 每個網域都會定義透過連接埠與其他網域通訊所需的介面。連接埠是使用主要和次要介面卡來實作。如此一來，如果介面卡有變更，網域就不會受到影響。此外，網域是彼此分離的。

在上圖所示的架構範例中，Lambda、Amazon EC2、亞馬遜 ECS 和AWS Fargate用作主要配接器。API Gateway、Elastic Load Balancing 和 Amazon SQS 用作次要介面卡。 EventBridge

常見問答集

為什麼要使用六角形體系結構？

Hexagonal 架構將開發人員的注意力轉移到領域邏輯上，簡化了測試自動化，並提高了代碼質量和適應性。這些改進可加快上市時間，並更輕鬆地進行技術和組織擴展。

為什麼要使用領域驅動式設計？

領域驅動設計 (DDD) 可讓您使用業務相關者與工程師之間的共同語言來建置軟體元件和建構。DDD 可以幫助您管理軟件的複雜性，並且是長期維護軟件產品的有效策略。

我可以在沒有六角形體系結構的情況下練習測試驅動

是。測試驅動開發（TDD）不僅限於特定的軟件設計模式。但是，六角形體系結構使得練習 TDD 變得更加容易。

我可以在沒有六角形架構和領域驅動設計的情況下擴展產品嗎？

是。大多數設計模式都可以實現技術和組織產品擴展。但是，六角形架構和 DDD 使其更容易擴展，並且長期對於大型項目更有效。

我應該使用哪些技術來實現六角形體系結構？

六角形架構不僅限於特定的技術堆疊。我們建議您選擇支援相依性反轉和單元測試的技術。

我正在開發一個最低可行的產品。花時間思考軟件架構是否有意義？

是。我們建議您將模式用於 MVP 您熟悉的模式。我們鼓勵您嘗試練習六角形架構，直到您的工程師對此感到滿意為止。為新項目建立六角形架構不需要比沒有任何架構開始更大的時間投入。

我正在開發一個最低限度的可行產品，沒有時間編寫測試。

如果您的 MVP 包含商務邏輯，我們強烈建議您為其編寫自動化測試。這將減少反饋循環並節省時間。

我可以在六角形架構中使用哪些額外的設計模式？

使用 CQRS 模式來支援整個系統的縮放。使用存放庫模式來儲存和還原您的網域模型。使用工作模式單位來管理交易處理步驟。使用構成對繼承來建立領域彙總、實體和值物件的模型。請勿建立複雜的物件階層。

後續步驟

- 閱讀資源本節中收集的連結，進一步熟悉領域驅動的設計概念。
- 如果您正在實施新項目，請使用本指南中提供的項目結構模板並實現一些功能。
- 如果您正在實施現有項目的過程中，請確定可以在只讀和只寫操作之間拆分的代碼。將唯讀程式碼抽象到查詢服務中，並將唯寫程式碼放入命令處理常式中。
- 當您的基本項目結構到位時，編寫單元測試，建立具有測試自動化的持續集成（CI），並遵循測試驅動開發（TDD）實踐。

資源

參考

- 使用AWS Lambda（AWS規定指導模式）在六角形體系結構中構建 Python 項目
- 敏捷團隊（可擴展的敏捷框架網站）
- Python 的架構模式，由哈里·珀西瓦爾和鮑勃·格雷戈里（奧萊利媒體，2020 年 3 月 31 日），具體是以下章節：
  - 命令和命令處理程序
  - 命令查詢職責隔離 (CQRS)
  - 儲存庫模式
- 事件風暴:最聰明的方法合作超越筒倉邊界, 由阿爾貝托·布蘭多利尼 (事件風暴網站)
- 揭開康威定律，薩姆·紐曼（思想工程網站，2014 年 6 月 30 日）
- 與詹姆斯·貝斯威克一起AWS Lambda發展進化體系結構（AWS計算博客，2021 年 7 月 8 日）
- 網域語言：解決軟體核心的複雜性 (網域語言網站)
- 立面，從潛入亞歷山大·什韋茨的設計模式（電子書，2018 年 12 月 5 日）
- GivenWhenThen作者：馬丁·福勒（八月二十一日）
- 實施領域驅動設計，由沃恩·弗農（Addison-Wesley 專業，2013 年 2 月）
- 逆康威操縱（思想工程網站，2014 年 7 月 8 日）
- 本月模式：紅綠色重構（DZone 網站，2017 年 6 月 2 日）
- 固體設計原則解釋：依賴反轉原則與代碼示例，由索本·揚森（Stackify 網站，2018 年 5 月 7 日）
- 固體原則：解釋和例子，西蒙·霍伯格（ITNEXT 網站，2019 年 1 月 1 日）
- 敏捷開發的藝術：測試驅動開發，詹姆斯·肖爾和謝恩·華登（奧萊利媒體，2010 年 3 月 25 日）
- 面向對象編程的固體原理用簡單的英語解釋, 由 Yigit 凱末爾 Erinc (freeCodeCamp面向對象的編程帖子, 八月 20, 2020)
- 什麼是事件驅動的架構？ (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

- 摩托車
- LocalStack

文件歷史紀錄

下表說明本指南的重大變更。如果您想收到有關 future 更新的通知，您可以訂閱 RSS 摘要。

變更	描述	日期
初始出版	—	2022 年 6 月 15 日

AWS 規範指南詞彙表

以下是 AWS Prescriptive Guidance 所提供策略、指南和模式的常用術語。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：將您的內部部署 Oracle 資料庫遷移至 Amazon Aurora Postgre SQL-Compatible Edition。
- 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：將您的內部部署 Oracle 資料庫遷移至 中的 Oracle 的 Amazon Relational Database Service （Amazon RDS） 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：將內部部署 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至相同平台的雲端服務。範例：遷移 Microsoft Hyper-V 應用程式至 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、一致性、隔離、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，但只有來源資料庫處理來自連接應用程式的交易，同時將資料複寫至目標資料庫。目標資料庫在遷移期間不接受任何交易。

彙總函數

在一組資料列上操作並計算該群組單一傳回值的SQL函數。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

經常用於重複性問題的解決方案，其解決方案具有反效益、無效或效果不如替代方案。

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侵害。

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。如需如何在遷移策略AIOps中使用 AWS 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。

原子度、一致性、隔離、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ABAC AWS中的 。

權威性資料來源

存放主要版本資料的位置，被視為最可靠的資訊來源。您可以將資料從授權資料來源複製到其他位置，以處理或修改資料，例如匿名、編輯或假名化資料。

可用區域

中與其他可用區域中的故障 AWS 區域 隔離的不同位置，並對相同區域中的其他可用區域提供低成本、低延遲的網路連線。

AWS 雲端採用架構（AWS CAF）

來自 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，以成功地遷移至雲端。 AWS CAF 將指導方針整理成六個重點領域：業務、人員、治理、平台、安全和操作。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 提供人員開發、訓練和通訊的指引，協助組織準備好成功採用雲端。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估算的工具。 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

請參閱業務連續性規劃 。

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。您可以搭配 Amazon Detective 使用行為圖表來檢查失敗的登入嘗試、可疑API的呼叫和類似的動作。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。另請參閱永久性 。

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。此策略可協助您在影響最小的情況下快速復原。

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。某些機器人很有用或有益，例如在網際網路上為資訊編製索引的 Web 爬蟲程式。某些其他稱為壞機器人 的機器人，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由單一方控制的機器人網路，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，使用者取得其通常無權存取 AWS 帳戶 之 存取權的快速方法。如需詳細資訊，請參閱 Well-Architected 指南中的 AWS 實作碎片程序指標。

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

版本向最終使用者緩慢且增量的版本。當您有信心時，請部署新版本並完全取代目前版本。

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。您可以使用 CDC進行各種用途，例如稽核或複寫目標系統中的變更，以維持同步。

混亂工程

故意引入故障或破壞性事件，以測試系統的復原能力。您可以使用 AWS Fault Injection Service（AWS FIS）來執行實驗，以強調 AWS 工作負載並評估其回應。

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。

Cloud Center of Excellence（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您的雲端採用 （例如，建立登陸區域、定義 CCoE、建立操作模型）
- 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 在企業 AWS 雲端 策略部落格的 The Journey Toward Cloud-First 和採用階段部落格中定義。如需有關它們如何與 AWS 遷移策略關聯的資訊，請參閱遷移準備指南 。

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。 每個版本的程式碼都稱為分支 。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

很少存取且通常為歷史資料的資料。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價格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映像和影片等視覺化格式分析和擷取資訊的 AI 欄位。例如， AWS Panorama 提供將 CV 新增至內部部署攝影機網路的裝置，而 Amazon 則 SageMaker 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。您通常會CMDB在遷移的產品組合探索和分析階段使用來自 的資料。

一致性套件

您可以組合的 AWS Config 規則和修復動作集合，以自訂合規和安全檢查。您可以使用 YAML 範本，將一致性套件部署為 AWS 帳戶 和 區域中或整個組織中的單一實體。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 is commonly described as a pipeline. CI/CD 可協助您自動化程序、提高生產力、改善程式碼品質，以及更快交付。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

請參閱電腦視覺效果 。

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化。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

架構架構架構，提供分散式、分散式資料擁有權，並具有集中式管理和治理。

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 上建立資料周邊 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

在整個資料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。

資料庫操作語言 （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。

defense-in-depth

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，方法 defense-in-depth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，以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。

部署

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）

用於識別和排序限制的程序，這些限制會對軟體開發生命週期中的速度和品質產生不利影響。DVSM 延伸了最初專為精實生產實務設計的價值串流映射程序。它專注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表示法，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

在星狀結構描述 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，其行為與文字類似。這些屬性通常用於查詢限制、篩選和結果集標籤。

災難

阻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，例如意外錯誤組態或惡意軟體攻擊。

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 中的雲端中的工作負載災難復原 AWS：復原。

DML

請參閱資料庫操作語言 。

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。如需有關如何搭配 strangler fig 模式使用網域驅動設計的資訊，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。

漂移偵測

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 中的漂移，或者您可以使用 AWS Control Tower 來偵測您的登陸區域中可能會影響對治理要求合規性的變更。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。

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縮短回應時間。

加密

將純文字資料轉換為可人工讀取的運算程序。

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） 主體。這些帳戶或主體可以透過建立介面端點，私下連線至您的VPC端點服務。如需詳細資訊，請參閱 Amazon Virtual Private Cloud （AmazonVPC） 文件中的建立端點服務。

企業資源規劃 （ERP）

可自動化和管理企業關鍵業務流程 （例如會計MES、 和 專案管理 ） 的系統。

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安全特徵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。EDA 透過計算摘要統計資料和建立資料視覺化來執行。

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：包含量值的資料，以及包含維度資料表外部索引鍵的資料。

快速失敗

使用頻繁且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，邊界，例如可用區域 AWS 區域、控制平面或資料平面，這些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。這通常表示為數值分數，可透過各種技術計算，例如 Shapley 累加解釋（SHAP）和整合式漸層。如需詳細資訊，請參閱使用 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。

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。

G

地理封鎖

請參閱地理限制 。

地理限制 (地理封鎖)

在 Amazon 中 CloudFront，此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，而以中繼線為基礎的工作流程是現代的首選方法。

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

高階規則，可協助管理跨組織單位 （ ） 的資源、政策和合規性OUs。預防性防護機制會強制執行政策，以確保符合合規標準。它們是透過使用服務控制政策和IAM許可界限來實作。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。其實作方式是使用 AWS Config、 AWS Security Hub、Amazon GuardDuty、 AWS Trusted Advisor、Amazon Inspector 和自訂 AWS Lambda 檢查。

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）

工作負載在遇到挑戰或災難時持續運作的能力，無需介入。HA 系統設計為自動容錯移轉、持續提供高品質效能，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

一種用於現代化和升級操作技術 （OT） 系統的方法，以更好地滿足製造業的需求。歷史資料是一種資料庫，用於從工廠的不同來源收集和儲存資料。

異質資料庫遷移

將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 遷移至 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。

常用資料

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，修正程式通常在典型 DevOps 的發行工作流程之外建立。

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，其定義其在 AWS 雲端 環境中的許可。

閒置應用程式

在 90 天內，平均CPU和記憶體用量介於 5% 到 20% 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、修補或修改現有基礎設施。與可變基礎設施 相比，不可避免的基礎設施本質上更一致、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（傳入）VPC

在 AWS 多帳戶架構中， VPC接受、檢查和路由來自應用程式外部的網路連線。AWS Security Reference Architecture 建議設定具有傳入、傳出和檢查的網路帳戶VPCs，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 的進步，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，請參閱建置工業物聯網（IIoT）數位轉型策略。

檢查 VPC

在 AWS 多帳戶架構中，集中VPC管理 VPCs（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 Security Reference Architecture 建議設定具有傳入、傳出和檢查的網路帳戶VPCs，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，請參閱使用 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 提供 的基礎ITSM。

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。如需整合雲端操作與ITSM工具的相關資訊，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被明確指派安全標籤值。使用者安全標籤與資料安全標籤之間的交集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 Rs 。

小端序系統

首先儲存最低有效位元組的系統。另請參閱永久性 。

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、間諜軟體和鍵盤側錄程式。

受管服務

AWS 服務 可 AWS 操作基礎設施層、作業系統和平台，而且您可以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 ）

用於追蹤、監控、記錄和控制生產程序的軟體系統，可將原物料轉換為工廠的成品。

MAP

請參閱遷移加速計畫 。

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。機制是在運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 中組織一部分的管理帳戶 AWS 帳戶 之外的所有 AWS Organizations。一個帳戶一次只能是一個組織的成員。

MES

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

小型、獨立的服務，透過定義明確的方式進行通訊，APIs通常由小型、獨立的團隊擁有。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 ，透過定義明確的介面進行通訊APIs。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）

提供諮詢支援、訓練和服務，以協助組織建立強大的操作基礎以遷移至雲端，並協助抵銷遷移的初始成本的 AWS 計畫。MAP 包含以有系統方式執行舊版遷移的遷移方法，以及一組可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操作、業務分析師和擁有者、遷移工程師、開發人員和在衝刺中工作的 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：EC2使用 AWS Application Migration Service 重新託管遷移至 Amazon。

遷移產品組合評估（MPA）

線上工具，提供驗證商業案例以遷移至 的資訊 AWS 雲端。MPA 提供詳細的產品組合評估（伺服器大小調整、定價、TCO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、遷移優先順序和波規劃）。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。

遷移就緒狀態評估（MRA）

使用 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 AWS CAF。如需詳細資訊，請參閱遷移準備程度指南。MRA 是AWS 遷移策略 的第一個階段。

遷移策略

用於將工作負載遷移至 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，並請參閱將組織動員以加速大規模遷移。

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 中的應用程式現代化策略 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

請參閱原始伺服器存取身分。

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 - Unified Architecture。

開放程序通訊 - Unified Architecture （OPC-UA）

工業自動化的 machine-to-machine（M2M） 通訊協定。OPC-UA 提供與資料加密、身分驗證和授權方案的互通性標準。

操作層級協議 （OLA）

闡明哪些功能性 IT 群組承諾交付給彼此的協議，以支援服務層級協議 （SLA）。

操作預備檢閱 （ORR）

問題及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 中的操作整備檢閱 （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 Industry 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 建立的追蹤 AWS CloudTrail 會記錄 AWS 帳戶 中組織中所有 的事件 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 文件中的 CloudTrail為組織建立追蹤。

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 透過加速變革採用、解決轉型問題，以及推動文化和組織變革，協助組織準備和轉換新系統和策略。在 AWS 遷移策略中，由於雲端採用專案所需的變更速度，此架構稱為人員加速 。如需詳細資訊，請參閱 OCM指南。

原始存取控制（OAC）

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

原始存取身分（OAI）

在 中 CloudFront，此選項用於限制存取以保護您的 Amazon S3 內容。當您使用 時OAI， 會 CloudFront 建立 Amazon S3 可以驗證的主體。已驗證的主體只能透過特定 CloudFront 分發存取 S3 儲存貯體中的內容。另請參閱 OAC，它提供更精細和增強的存取控制。

ORR

請參閱操作預備檢閱 。

OT

請參閱操作技術 。

傳出（輸出）VPC

在 AWS 多帳戶架構中，VPC處理從應用程式內啟動之網路連線的 。AWS Security Reference Architecture 建議設定具有傳入、傳出和檢查的網路帳戶VPCs，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

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

個人身分資訊（PII）

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。的範例PII包括名稱、地址和聯絡資訊。

PII

請參閱個人識別資訊 。

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

請參閱可程式邏輯控制器 。

PLM

請參閱產品生命週期管理。

政策

可以定義許可（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。如需詳細資訊，請參閱在微服務中啟用資料持久性。

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

一種系統工程方法，在整個工程過程中將隱私權納入考量。

私有託管區域

容器，其中包含您希望 Amazon Route 53 如何回應一個或多個 內網域及其子網域的DNS查詢的資訊VPCs。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。這些控制項會先掃描資源，然後再佈建。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，並請參閱在 上實作安全控制項中的主動控制項。 AWS

產品生命週期管理 （PLM）

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

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。

擬匿名化

將資料集中的個人識別碼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。

publish/subscribe (pub/sub)

一種模式，可在微服務之間啟用非同步通訊，以提高可擴展性和回應能力。例如，在微服務型中MES，微服務可以將事件訊息發佈到其他微服務可以訂閱的頻道。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、已諮詢、知情（RACI）。

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、已諮詢、知情（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）

服務中斷和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 AWS 的資源集合。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 哪些帳戶可以使用 。

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 Rs 。

轉譯形式

請參閱 7 Rs 。

回購

請參閱 7 Rs 。

彈性

應用程式抵抗中斷或從中斷中復原的能力。在 中規劃復原能力時，高可用性和災難復原是常見的考量事項 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。

負責任、負責、已諮詢、知情（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）、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 Rs 。

淘汰

請參閱 7 Rs 。

輪換

定期更新秘密的程序，讓攻擊者更難存取憑證。

資料列和資料欄存取控制（RCAC）

使用已定義存取規則的基本靈活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（IdPs）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 Management Console 或呼叫操作， AWS API而不必IAM為您組織中的每個人建立 中的使用者。如需 SAML 2.0 型聯合的詳細資訊，請參閱IAM文件中關於 SAML 2.0 型聯合。

SCADA

請參閱 監控控制和資料擷取 。

SCP

請參閱服務控制政策 。

秘密

在 中 AWS Secrets Manager，以加密形式存放的機密或限制資訊，例如密碼或使用者憑證。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 秘密中的內容？。

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 。

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。

安全資訊和事件管理（SIEM）系統

結合安全資訊管理（SIM）和安全事件管理（SEM）系統的工具和服務。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，並產生警示。

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執行個體或輪換憑證。

伺服器端加密

由接收資料的 AWS 服務 加密其目的地的資料。

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。SCPs 定義管理員可委派給使用者或角色之動作的防護機制或設定限制。您可以使用 SCPs作為允許清單或拒絕清單，以指定允許或禁止的服務或動作。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

URL 的 進入點的 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。

服務層級協議（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

描述您共享 AWS 的雲端安全與合規責任的模型。 AWS 負責雲端的安全，而您要負責雲端的安全。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 。

單一失敗點（SPOF）

應用程式的單一關鍵元件中發生故障，可能會中斷系統。

SLA

請參閱服務層級協議 。

SLI

請參閱服務層級指示器 。

SLO

請參閱服務層級目標 。

split-and-seed 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

請參閱單一失敗點 。

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

中 IP 地址的範圍VPC。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。

T

標籤

作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。

傳輸閘道

可用來互連 VPCs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，以代表您在組織中執行任務 AWS Organizations ，並在其帳戶中執行任務。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 AWS Organizations 搭配使用其他 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

一個小型 DevOps 團隊，您可以使用兩個披薩來餵送。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

也稱為繁重提升 ，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

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

漏洞

損害系統安全性的軟體或硬體缺陷。

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，通常可接受中等慢的查詢。

視窗函數

對以某種方式與目前記錄相關聯之資料列群組執行計算的SQL函數。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）

一次性寫入資料的儲存模型，可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的。

Z

零時差漏洞

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。威脅發動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。

殭屍應用程式

平均CPU和記憶體用量低於 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。