彈性生命週期架構

AWS 方案指引

AWS 方案指引: 彈性生命週期架構

Table of Contents

簡介 1
術語和定義 2
持續彈性 2
階段 1：設定目標 3
映射關鍵應用程式 3
映射使用者案例 4
定義測量 4
建立其他測量 4
階段 2：設計和實作 6
AWS Well-Architected 架構 6
了解相依性 6
災難復原策略 7
定義 CI/CD 策略 7
執行 ORRs 8
了解 AWS 故障隔離界限 8
選取回應 8
彈性建模 9
安全失敗 9
階段 3：評估和測試 10
部署前活動 10
環境設計 10
整合測試 10
自動化部署管道 11
負載測試 11
部署後活動 11
執行彈性評估 11
DR 測試 12
漂移偵測 12
合成測試 12
混沌工程 12
階段 4：操作 14
可觀測性 14
事件管理 14
持續彈性 15

階段 5：回應和學習 16
建立事件分析報告 16
執行操作審查 17
檢閱警示效能 17
警示精確度 17
誤報 17
偽陰性 18
重複提醒 18
執行指標檢閱 18
提供訓練和啟用 18
建立事件知識庫 18
深度實作彈性 19
結論和資源 20
貢獻者 21
文件歷史紀錄 22
詞彙表 23
..... 23
A 23
B 26
C 27
D 30
E 33
F 35
G 36
H 37
I 38
L 40
M 41
O 45
P 47
Q 49
R 49
S 52
T 55
U 56
V 56

W 57
Z 58
..... lix

彈性生命週期架構：持續改善彈性的持續方法

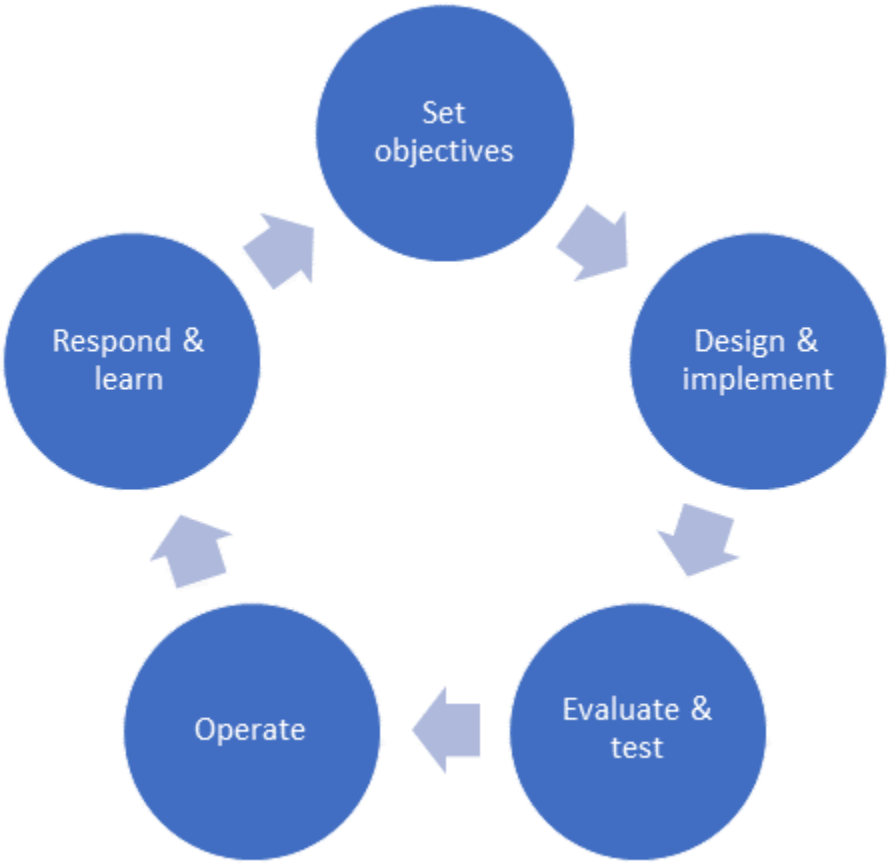
Amazon Web Services（貢獻者）

2023 年 10 月 (文件歷史記錄)

現代組織現在面臨越來越多與彈性相關的挑戰，尤其是當客戶的期望向永遠不變、永遠可用的思維轉變時。遠端團隊和複雜的分散式應用程式，與頻繁發行的需求相輔相成。因此，組織及其應用程式需要比以往更具彈性。

AWS 將彈性定義為應用程式抵禦中斷或從中斷中復原的能力，包括與基礎設施、相依服務、設定錯誤和暫時性網路問題相關的能力。（請參閱 AWS Well-Architected Framework 可靠性支柱文件中的彈性和可靠性元件。）不過，為了達到所需的彈性層級，通常需要權衡。需要相應地評估和調整操作複雜性、工程複雜性和成本。

根據與客戶和內部團隊合作多年， AWS 開發了擷取彈性學習和最佳實務的彈性生命週期架構。此架構概述了下圖所示的五個關鍵階段。在每個階段，您可以使用策略、服務和機制來改善復原狀態。



這些階段會在本指南的下列章節中討論：

- 階段 1：設定目標
- 階段 2：設計和實作
- 階段 3：評估和測試
- 階段 4：操作
- 階段 5：回應和學習

術語和定義

每個階段的彈性概念會在不同的層級套用，從個別元件到整個系統。實作這些概念需要幾個術語的明確定義：

- 元件是執行 函數的 元素，由軟體和技術資源組成。元件的範例包括程式碼組態、聯網等基礎設施，甚至是伺服器、資料存放區，以及多重要素驗證 (MFA) 裝置等外部相依性。
- 應用程式是提供商業價值的元件集合，例如面向客戶的 Web 商店或改善機器學習模型的後端程序。應用程式可能包含單一 AWS 帳戶中的元件子集，也可能是跨越多個 AWS 帳戶 和 區域的多個元件集合。
- 系統是管理指定業務職能所需的應用程式、人員和程序集合。它包含執行 函數所需的應用程式；持續整合和持續交付 (CI/CD)、可觀測性、組態管理、事件回應和災難復原等操作程序；以及管理此類任務的運算子。
- 中斷是導致應用程式無法正常交付其業務職能的事件。
- 如果中斷未緩解，則會對應用程式造成影響。如果應用程式遇到一組中斷，應用程式可能會受損。

持續彈性

彈性生命週期是持續進行的程序。即使在同一組織中，您的應用程式團隊也可能在每個階段中執行不同層級的完整性，具體取決於您的應用程式需求。不過，每個階段的完成程度越高，您的應用程式會有的彈性就越高。

您應該將彈性生命週期視為組織可以操作的標準程序。 AWS 刻意將彈性生命週期建模為類似於軟體開發生命週期 (SDLC)，目標是在開發和操作應用程式時在整個操作程序中整合規劃、測試和學習。 如同許多敏捷開發程序一樣，每次反覆執行開發程序時都可以重複彈性生命週期。 我們建議您隨著時間逐漸深化生命週期的每個階段中的實務。

階段 1：設定目標

了解需要多少彈性層級，以及如何衡量它是設定目標階段的基礎。如果您沒有目標且無法測量，就很難改善。

並非所有應用程式都需要相同等級的彈性。當您設定目標時，請考慮所需的層級，以便進行正確的投資和權衡。一個很好的比喻是汽車：它有四個輪胎，但只攜帶一個備用輪胎。在乘車期間取得多個平面輪胎的機會很低，而擁有額外的備用設備可能會脫離其他功能，例如貨物空間或燃料效率，因此這是合理的取捨。

定義目標後，您會在稍後階段 (階段 2：設計和實作 和階段 4：操作) 實作可觀測性控制，以了解目標是否達成。

映射關鍵應用程式

定義彈性目標不應只是技術對話。相反地，請從業務導向的重點開始，以了解應用程式應該提供什麼，以及受損的後果。然後，對業務目標的這種理解會層疊到架構、工程和操作等領域。您定義的任何彈性目標都可能套用到所有應用程式，但目標的測量方式通常會根據應用程式的功能而有所不同。您可能正在執行對業務至關重要的應用程式，如果此應用程式受損，您的組織可能會損失重大收入或聲譽受損。或者，您可能有另一個不重要的應用程式，可以容忍一些停機時間，而不會對組織執行業務的能力產生負面影響。

例如，假設零售公司的訂單管理應用程式。如果訂單管理應用程式的元件受損且未正確執行，則新銷售將不會通過。此零售公司也為其位於其中一棟建築物的員工設有咖啡廳。咖啡廳有一個線上選單，員工可以在靜態網頁上存取。如果此網頁無法使用，有些員工可能會抱怨，但不一定會對公司造成財務損害。根據此範例，企業可能會選擇為訂單管理應用程式設定更積極的彈性目標，但不會進行重大投資以確保 Web 應用程式的彈性。

識別最關鍵的應用程式、最努力的領域，以及進行權衡的領域，與能夠測量應用程式在生產環境中的彈性一樣重要。若要進一步了解受損的影響，您可以執行業務影響分析 (BIA)。BIA 提供結構化和系統性的方法，以識別關鍵業務應用程式並排定優先順序、評估潛在風險和影響，以及識別支援的相依性。BIA 可協助量化組織最重要應用程式的停機時間成本。此指標有助於概述如果特定應用程式受損且無法完成其函數，將花費多少費用。在上述範例中，如果訂單管理應用程式受損，零售業務可能會損失重大收入。

映射使用者案例

在 BIA 過程中，您可能會發現應用程式負責多個業務職能，或業務職能需要多個應用程式。使用先前的零售公司範例，訂單管理函數可能需要單獨的應用程式來結帳、提升和定價。如果一個應用程式失敗，業務和與公司互動的使用者可能會感受到影響。例如，公司可能無法新增新訂單、提供促銷和折扣的存取權，或更新其產品的價格。訂單管理函數所需的這些不同函數可能會依賴多個應用程式。這些函數也可能有多個外部相依性，這使得實現純粹以元件為中心的彈性的過程過於複雜。處理此案例的更好方法是專注於使用者案例，其中概述了使用者在與一個應用程式或一組應用程式互動時預期的體驗。

專注於使用者案例可協助您了解哪些客戶體驗部分最重要，因此您可以建立機制來防範特定威脅。在先前的範例中，一個使用者案例可能是結帳，這涉及結帳應用程式，並且與定價應用程式有相依性。另一個使用者案例可能是檢視涉及提升應用程式的提升。映射最關鍵的應用程式及其使用者案例後，您可以開始定義用於測量這些使用者案例彈性的指標。這些指標可以套用至整個產品組合或個別使用者案例。

定義測量

復原點目標 RPOs)、復原時間目標 RTOs) 和服務層級目標 (SLOs) 是標準產業測量，用於評估指定系統的彈性。RPO 是指企業在發生故障時可容忍的資料遺失量，而 RTO 是在中斷後必須再次提供應用程式的速度。這兩個指標是以時間單位測量：秒、分鐘和小時。您也可以測量應用程式正常運作的時間量；也就是說，它會依設計執行其函數，並可供其使用者存取。這些 SLOs會詳細說明客戶將接收的預期服務水準，並透過指標進行測量，例如在不到一秒的回應時間內（例如，99.99% 的請求將每個月收到回應）未發生錯誤的服務請求百分比 (%)。 RPO 和 RTO 與災難復原策略相關，假設應用程式操作和復原程序將會中斷，範圍從還原備份到重新導向使用者流量。透過實作高可用性控制來解決 SLOs，這通常會減少應用程式的停機時間。

SLO 指標通常用於服務層級協議 (SLAs) 的定義，這是服務提供者與最終使用者之間的合約。SLAs 通常附帶財務承諾，並概述如果不符合這些協議，供應商需要支付的處罰。不過，SLA 不是彈性狀態的測量，而提高 SLA 並不會讓您的應用程式更具彈性。

您可以開始根據 SLOs、RPOs 和 RTOs 設定目標。在您定義彈性目標並清楚了解 RPO 和 RTO 目標之後，您可以使用 AWS Resilience Hub來執行架構的評估，以發現與彈性相關的潛在弱點。 AWS Resilience Hub 會根據 AWS Well-Architected Framework 最佳實務評估應用程式架構，並在具體需要改進的內容中分享修補指引，以滿足您定義的 RTO 和 RPO 目標。

建立其他測量

RPO、RTO 和 SLOs 是彈性的良好指標，但您也可以從業務角度思考目標，並定義應用程式功能的目標。例如，您的目標是：如果我的前端和後端之間的延遲增加 40%，每分鐘成功訂單將維持在 98%

以上。 或者：即使特定元件遺失，每秒啟動的串流仍會保持在與平均值的標準差內。您也可以建立目標，以縮短所有已知故障類型的平均復原時間 (MTTR)；例如：如果發生任何這些已知問題，復原時間將減少 x%。建立符合業務需求的目標可協助您預測應用程式應容忍的失敗類型。它還可協助您識別方法來降低應用程式受損的可能性。

如果您考慮在遺失為應用程式提供動力的執行個體的 5% 時繼續運作的目標，您可以判斷應用程式應預先擴展，或能夠快速擴展，以支援在該事件期間造成的額外流量。或者，您可以判斷應該利用不同的架構模式，如階段 2：設計和實作一節所述。

您也應該針對特定業務目標實作可觀測性措施。例如，您可以追蹤平均訂單率、平均訂單價格、平均訂閱數量或其他指標，這些指標可以根據您應用程式的行為，提供業務運作狀態的洞見。透過為您的應用程式實作可觀測性功能，您可以建立警示，並在這些指標超過您定義的界限時採取動作。可觀測性會在階段 4：操作區段中詳細說明。

階段 2：設計和實作

在上一個階段中，您會設定彈性目標。現在，在設計和實作階段，您會嘗試預測失敗模式，並根據您在上一個階段設定的目標來識別設計選擇。您也可以定義變更管理的策略，並開發軟體程式碼和基礎設施組態。以下各節重點介紹在考慮成本、複雜性和營運開銷等權衡時應考慮的 AWS 最佳實務。

AWS Well-Architected 架構

當您根據所需的彈性目標建構應用程式時，您需要評估多個因素，並對最佳的架構進行權衡。若要建置高彈性的應用程式，您必須考慮設計、建置和部署、安全性和操作方面的層面。AWS Well-Architected Framework 提供一組最佳實務、設計原則和架構模式，協助您設計彈性應用程式 AWS。 AWS Well-Architected Framework 的六個支柱提供設計與操作彈性、安全、有效率、經濟實惠且永續系統的最佳實務。架構提供一種方式，可讓您根據最佳實務一致地測量架構，並識別需要改進的領域。

以下是 AWS Well-Architected Framework 如何協助您設計和實作符合您恢復能力目標的應用程式的範例：

- 可靠性支柱：可靠性支柱強調建置應用程式的重要性，即使在故障或中斷期間也能正確且一致地運作。例如， AWS Well-Architected Framework 建議您使用微服務架構，讓您的應用程式更小、更簡單，因此您可以區分應用程式中不同元件的可用性需求。您也可以透過使用限流、以指數退避重試、快速失敗 （卸載）、冪等性、持續工作、斷路器和靜態穩定性，找到建置應用程式的最佳實務詳細說明。
- 全面審查： AWS Well-Architected Framework 鼓勵針對最佳實務和設計原則全面審查您的架構。它提供了一種一致測量架構並識別需要改進領域的方法。
- 風險管理： AWS Well-Architected Framework 可協助您識別和管理可能影響應用程式可靠性的風險。透過主動處理潛在的失敗案例，您可以降低它們的可能性或導致的損害。
- 持續改進：彈性是一個持續的過程，而 AWS Well-Architected Framework 強調持續改進。透過根據 AWS Well-Architected Framework 的指引定期審查和改進您的架構和程序，您可以確保您的系統在面對不斷變化的挑戰和要求時保持彈性。

了解相依性

了解系統的相依性是恢復能力的關鍵。相依性包括應用程式內元件之間的連線，以及與應用程式外部元件的連線，例如第三方 APIs和企業擁有的共享服務。了解這些連線可協助您隔離和管理中斷，因為一個元件中的受損可能會影響其他元件。此知識可協助工程師評估損害的影響，並相應地進行規劃，並確保有效使用資源。了解相依性可協助您建立替代策略並協調復原程序。它還可協助您判斷可將硬相依性

取代為軟相依性的情況，讓您的應用程式可以在相依性受損時繼續為其業務功能提供服務。相依性也會影響負載平衡和應用程式擴展的決策。當您變更應用程式時，了解相依性至關重要，因為它可協助您判斷潛在風險和影響。此知識可協助您建立穩定、有彈性的應用程式，協助進行故障管理、影響評估、損害復原、負載平衡、擴展和變更管理。您可以手動追蹤相依性，或使用 等工具和服務AWS X-Ray來了解分散式應用程式的相依性。

災難復原策略

災難復原 (DR) 策略在建置和操作彈性應用程式方面扮演關鍵角色，主要是透過確保業務連續性。它保證關鍵業務操作即使在災難性事件期間，仍能維持最不可能的損害，從而將停機時間和潛在的收入損失降至最低。DR 策略對於資料保護至關重要，因為它們通常包含跨多個位置的定期資料備份和資料複寫，這有助於保護寶貴的商業資訊，並有助於防止災難期間的總損失。此外，許多產業受到政策的監管，這些政策要求企業制定 DR 策略來保護敏感資料，並確保服務在災難期間保持可用。透過確保盡可能減少服務受損，DR 策略也會增強客戶的信任和滿意度。實作良好且經常練習的 DR 策略可減少災難後的復原時間，並有助於確保應用程式快速恢復上線。此外，災難可能會產生大量成本，不僅因為停機時間造成的收入損失，還因為還原應用程式和資料而產生。精心設計的 DR 策略有助於防止這些財務損失。

您選擇的策略取決於應用程式、RTO 和 RPO 以及預算的特定需求。 是一種AWS Elastic Disaster Recovery專門建置的彈性服務，可用來協助實作現場部署和雲端應用程式的 DR 策略。

如需詳細資訊，請參閱 上的工作負載災難復原 AWS和 AWS 網站上的AWS 多區域基礎。

定義 CI/CD 策略

應用程式受損的常見原因之一是程式碼或其他變更，這些變更會改變先前已知的運作狀態。如果您不小心處理變更管理，可能會導致頻繁的損害。變更的頻率會增加影響的機會。不過，變更頻率較低會導致較大的變更集，這更有可能因為變更集的複雜度高而造成損害。持續整合和持續交付 (CI/CD) 實務旨在保持少量且頻繁的變更 （導致生產力提高），同時透過自動化進行高層級的檢查。一些基礎策略包括：

- 完全自動化：CI/CD 的基本概念是盡可能自動化建置和部署程序。這包括建置、測試、部署，甚至監控。自動化管道有助於降低人為錯誤的可能性、確保一致性，並讓程序更可靠和更有效率。
- 測試驅動開發 (TDD)：在編寫應用程式程式碼之前撰寫測試。此做法可確保所有程式碼都有相關聯的測試，這可改善程式碼的可靠性和自動化檢查的品質。這些測試會在 CI 管道中執行，以驗證變更。
- 頻繁遞交和整合：鼓勵開發人員頻繁遞交程式碼並經常執行整合。小而頻繁的變更較容易測試和偵錯，可降低重大問題的風險。自動化可降低每個遞交和部署的成本，讓頻繁整合成為可能。

- 不可變基礎設施：將您的伺服器和其他基礎設施元件視為靜態、不可變的實體。取代基礎設施，而不是盡可能修改基礎設施，並透過經過測試的程式碼建立新的基礎設施，並透過管道部署。
- 轉返機制：永遠有簡單、可靠且經常測試的方法，可在發生錯誤時轉返變更。能夠快速回到先前的已知良好狀態對於部署安全至關重要。這可以是還原為先前狀態的簡單按鈕，也可以完全自動化並透過警示啟動。
- 版本控制：維護所有應用程式程式碼、組態，甚至是基礎設施，做為版本控制儲存庫中的程式碼。此實務有助於確保您可以輕鬆追蹤變更，並視需要加以還原。
- Canary 部署和藍/綠部署：首先將應用程式的新版本部署到基礎設施的子集，或維護兩個環境（藍/綠），可讓您驗證變更在生產環境中的行為，並視需要快速復原。

CI/CD 不僅與工具有關，也與文化有關。建立重視自動化、測試和學習失敗的文化與實作正確的工具和程序一樣重要。如果轉返在影響最小的情況下非常快速地完成，則不應被視為失敗，而是學習體驗。

執行 ORRs

操作整備審查 (ORR) 有助於識別操作和程序上的差距。在 Amazon，我們建立了 ORRs，透過最佳實務指引，將數十年營運高規模服務的學習精進為精選問題。ORR 會擷取先前學到的經驗教訓，並要求新團隊確保他們已考慮應用程式中的這些經驗教訓。ORRs可以提供失敗模式或失敗原因的清單，這些模式或原因可以傳遞至以下恢復能力建模一節所述的恢復能力建模活動。如需詳細資訊，請參閱 AWS Well-Architected Framework 網站上的操作準備審查 (ORRs)。

了解 AWS 故障隔離界限

AWS 提供多個故障隔離界限，協助您實現恢復能力目標。您可以使用這些界限來利用其提供的可預測影響抑制範圍。您應該熟悉如何使用這些邊界來設計 AWS 服務，以便針對您為應用程式選取的相依性進行刻意選擇。若要了解如何在應用程式中使用邊界，請參閱 AWS 網站上的AWS 故障隔離邊界 。

選取回應

系統可以透過各種方式回應警示。某些警示可能需要操作團隊的回應，而其他警示可能會觸發應用程式中的自我修復機制。您可以決定保留可自動化為手動操作的回應，以控制自動化成本或管理工程限制。可能會選擇對警示的回應類型，做為實作回應的成本、警示的預期頻率、警示的準確性，以及根本沒有回應警示的潛在業務損失的函數。

例如，當伺服器程序當機時，作業系統可能會重新啟動程序，或者可能佈建新伺服器並終止舊伺服器，或者可能指示操作員從遠端連線至伺服器並重新啟動。這些回應具有相同的結果 - 重新啟動應用程式伺服器程序 - 但實作和維護成本不同。

> **Note**
>
> 您可以選取多個回應，以採取深入的彈性方法。例如，在先前的案例中，應用程式團隊可能會選擇實作所有三個回應，每個回應之間都有時間延遲。如果失敗的伺服器程序指示器在 30 秒後仍處於警示狀態，團隊可以假設作業系統無法重新啟動應用程式伺服器。因此，他們可能會建立自動擴展群組，以建立新的虛擬伺服器並還原應用程式伺服器程序。如果指示器在 300 秒後仍處於警示狀態，則可能會傳送提醒給操作人員，以連線至原始伺服器並嘗試還原程序。

應用程式團隊和業務選擇的回應應反映業務的偏好，以預先投資工程時間來抵銷營運開銷。您應該選擇回應 – 靜態穩定性、斷路器等軟體模式或操作程序 – 透過仔細考慮每個回應選項的限制條件和預期維護。可能存在一些標準回應來引導應用程式團隊，因此您可以使用集中式架構函數管理的程式庫和模式作為此考量的輸入。

彈性建模

彈性建模會記錄應用程式如何回應不同的預期中斷。 透過預測中斷，您的團隊可以實作可觀測性、自動化控制和復原程序，以緩解或防止中斷造成的損害。 AWS 已建立使用彈性分析架構開發彈性模型的指引。 此架構可協助您預測中斷及其對應用程式的影響。 透過預測中斷，您可以識別建置彈性、可靠應用程式所需的緩解措施。我們建議您使用彈性分析架構，在每次應用程式生命週期反覆運算時更新彈性模型。 將此架構與每次反覆運算搭配使用有助於減少事件，方法是在設計階段期間預測中斷，並在生產部署前後測試應用程式。使用此架構開發彈性模型有助於確保您符合彈性目標。

安全失敗

如果您無法避免中斷， 會安全失敗。請考慮使用預設的不安全操作模式建立應用程式，其中不會產生重大業務損失。資料庫的故障安全狀態範例預設為唯讀操作，其中不允許使用者建立或變更任何資料。根據資料的敏感度，您甚至可能希望應用程式預設為關閉狀態，甚至不執行唯讀查詢。考慮應用程式的故障安全狀態，並在極端條件下預設為此操作模式。

階段 3：評估和測試

在生命週期的評估和測試階段，應用程式或現有應用程式的變更已經過設計，但尚未發佈至生產環境。在此階段，您會實作活動來測試先前階段已執行的實務，並評估結果。應用程式可能仍在作用中開發中，或主要開發可能已完成，而且應用程式在發佈至生產環境之前可能正在進行測試。在此階段，您會專注於開發和執行測試，以確認或反省應用程式將滿足定義之彈性目標的期望。此外，您可以開發和測試系統的操作程序。您在階段 2：設計和實作階段中開發的部署程序會付諸實作，並評估結果。雖然這些測試和評估活動在生命週期的這個部分開始，但不會在此結束。當您進入階段 4：操作階段時，測試和評估會持續進行。

評估和測試階段分為兩個階段：部署前活動和部署後活動。部署前活動包含在您將應用程式部署至任何環境之前應完成的任務，包括部署新版本的軟體，以及將初始部署至測試環境。部署後活動會在軟體部署到測試或生產環境之後進行。下列各節會更詳細地討論這些階段。

部署前活動

環境設計

您測試和評估應用程式的環境會影響您測試應用程式的完整程度，以及您對這些結果準確反映生產中會發生什麼的可信度。您可能可以使用 Amazon DynamoDB 等服務，在開發人員電腦上於本機執行一些整合測試（請參閱 DynamoDB 文件中的設定 DynamoDB 本機)。 DynamoDB 不過，在某個時間點，您需要在複寫生產環境的環境中進行測試，才能對結果達到最高的可信度。此環境會產生成本，因此我們建議您對環境採取暫存或管道化的方法，其中類似生產的環境稍後會出現在管道中。

整合測試

整合測試是測試應用程式定義良好的元件在使用外部相依性操作時正確執行其函數的程序。這些外部相依性可能是其他自訂開發的元件、您用於應用程式 AWS 的服務、第三方相依性和內部部署相依性。 本指南著重於展示應用程式彈性的整合測試。它假設單元和整合測試已存在，可證明軟體的功能準確性。

我們建議您設計整合測試，專門測試您已實作的彈性模式，例如斷路器模式或負載脫離（請參閱階段 2：設計和實作)。彈性導向整合測試通常涉及將特定負載套用至應用程式，或使用 AWS Fault Injection Service (AWS FIS) 等功能刻意將中斷引入環境。理想情況下，您應該在 CI/CD 管道中執行所有整合測試，並確保每次遞交程式碼時都執行測試。這可協助您快速偵測並回應任何程式碼或組態的變更，這些變更會導致違反您的彈性目標。大規模分散式應用程式非常複雜，即使微小的變更也會大幅影響應用程

式似乎不相關部分的彈性。嘗試在每個遞交上執行您的測試。 AWS 提供一組絕佳的工具，用於操作您的 CI/CD 管道和其他 DevOps 工具。如需詳細資訊，請參閱 網站上的 DevOps 簡介 AWS AWS 。

自動化部署管道

部署至生產前環境並在其中進行測試，是最適合自動化的重複且複雜的任務。此程序的自動化會釋放人力資源並減少發生錯誤的機會。自動化此程序的機制通常稱為管道。當您建立管道時，建議您設定一系列的測試環境，這些環境會越來越接近您的生產組態。您可以使用此系列環境重複測試您的應用程式。第一個環境提供的一組功能比生產環境更有限，但成本明顯較低。後續環境應新增 服務並擴展，以更緊密地反映生產環境。

首先在第一個環境中進行測試。部署在第一個測試環境中通過所有測試後，讓應用程式在一定的負載下執行一段時間，以查看是否隨著時間發生任何問題。確認您已正確設定可觀測性（請參閱本指南稍後的警示精確度)，以便您可以偵測出現的任何問題。當此觀察期間成功完成時，請將您的應用程式部署到下一個測試環境，並重複此程序，新增環境支援的其他測試或負載。以這種方式充分測試應用程式後，您可以使用先前設定的部署方法，將應用程式部署到生產環境（請參閱本指南稍早的定義 CI/CD 策略)。Amazon Builders' Library中的自動化安全、移出部署一篇文章是絕佳的資源，說明 Amazon 如何自動化程式碼部署。生產部署之前的環境數量會有所不同，取決於應用程式的複雜性及其相依性類型。

負載測試

在表面上，負載測試類似於整合測試。您可以測試應用程式及其外部相依性的離散函數，以確認其可如預期般運作。然後，負載測試會超越整合測試，以專注於應用程式在定義良好的負載下的運作方式。負載測試需要驗證正確的功能，因此必須在成功整合測試之後進行。請務必了解應用程式在預期負載下的回應能力，以及負載超過預期時的行為。這可協助您 驗證已實作必要的機制，以確保您的應用程式在極端負載下保持彈性。如需在 上載入測試的完整指南 AWS，請參閱《 AWS 解決方案程式庫》中的在上進行分散式負載測試 AWS。

部署後活動

彈性是持續進行的程序，在部署應用程式之後，必須繼續評估應用程式的彈性。部署後活動的結果，例如持續的恢復能力評估，可能需要您重新評估和更新稍早在恢復能力生命週期中執行的一些恢復能力活動。

執行彈性評估

將應用程式部署到生產環境之後，評估彈性不會停止。即使您有明確定義且自動化的部署管道，變更有時也可能直接發生在生產環境中。此外，您可能還未在部署前彈性驗證中考量到一些因素。 AWS

Resilience Hub提供一個集中位置，可讓您評估部署的架構是否符合您定義的 RPO 和 RTO 需求。您可以使用此服務執行應用程式的彈性隨需評估、自動化評估，甚至將它們整合到您的 CI/CD 工具中，如 AWS 部落格文章所述使用 AWS Resilience Hub 和 持續評估應用程式彈性 AWS CodePipeline。自動化這些評估是最佳實務，因為它有助於確保您持續評估生產環境中的彈性狀態。

DR 測試

在階段 2：設計和實作中，您開發了災難復原 (DR) 策略作為系統的一部分。在第 4 階段期間，您應該測試您的 DR 程序，以確保您的團隊為事件做好充分準備，並且您的程序可如預期般運作。您應該定期測試所有 DR 程序，包括容錯移轉和容錯回復，並檢閱每個練習的結果，以判斷是否應該以及如何更新系統的程序，以獲得最佳可能的結果。當您最初開發 DR 測試時，請提前安排測試，並確保整個團隊了解預期的內容、如何衡量結果，以及根據結果使用哪些意見回饋機制來更新程序。在您熟練執行排定的 DR 測試之後，請考慮執行無預警的 DR 測試。實際災難不會按排程發生，因此您需要隨時準備執行您的計劃。不過，未宣布並不表示未計劃。主要利益相關者仍需要規劃事件，以確保有適當的監控，並且客戶和關鍵應用程式不會受到負面影響。

漂移偵測

即使有自動化和明確定義的程序，生產應用程式中組態仍可能發生非預期的變更。若要偵測應用程式組態的變更，您應該有偵測偏離的機制，這是指與基準組態的偏差。若要了解如何偵測 AWS CloudFormation 堆疊中的偏離，請參閱 AWS CloudFormation 文件中的偵測堆疊和資源的未受管組態變更。若要偵測應用程式 AWS 環境中的偏離，請參閱 AWS Control Tower 文件中的偵測和解決偏離 AWS Control Tower。

合成測試

合成測試是建立可在生產環境中依排程執行的可設定軟體的程序，以模擬最終使用者體驗的方式測試應用程式的 APIs。這些測試有時稱為 Canary，參考該術語在煤礦中的原始用途。合成測試通常可在應用程式遭受中斷時提供早期警告，即使損害是部分或間歇性，灰色故障的情況也是如此。

混沌工程

混沌工程是一種系統化程序，涉及刻意以風險緩解的方式讓應用程式遭受破壞性事件、密切監控其回應，以及實作必要的改進。其目的是驗證或挑戰有關應用程式處理此類中斷能力的假設。混沌工程不會讓這些事件變成可能，而是讓工程師能夠在受控環境中協調實驗，通常是在低流量期間，並提供隨時可用的工程支援來有效緩解。

混沌工程從了解正在考慮的應用程式的正常操作條件開始，稱為穩定狀態。從那裡，您將制定一個假設，詳細說明應用程式在發生中斷時的成功行為。 您執行實驗，涉及刻意注入中斷，包括但不限於網

路延遲、伺服器故障、硬碟錯誤和外部相依性損害。然後，您可以分析實驗的結果，並根據經驗增強應用程式的彈性。此實驗是改善應用程式各種層面的寶貴工具，包括其效能，並發現可能一直隱藏的潛在問題。此外，混沌工程有助於顯示可觀測性和警示工具的缺陷，並協助您精簡它們。它也有助於縮短復原時間並增強操作技能。Chaos 工程可加速採用最佳實務，並培養持續改進的思維。最終，它可讓團隊透過定期練習和重複來建立和磨練他們的營運技能。

AWS 建議您在非生產環境中啟動混沌工程工作。您可以使用 AWS Fault Injection Service (AWS FIS) 執行混沌工程實驗，其中包含一般用途的故障以及獨有的故障 AWS。此全受管服務包含停止條件警示和完整許可控制，因此您可以輕鬆採用具有安全性和可信度的混沌工程。

階段 4：操作

完成階段 3：評估和測試後，您就可以將應用程式部署到生產環境。在操作階段，您將應用程式部署到生產環境，並管理客戶的體驗。 應用程式的設計和實作會決定許多彈性結果，但此階段著重於系統用來維護和改善彈性的操作實務。建立卓越營運的文化有助於在這些實務中建立標準和一致性。

可觀測性

了解客戶體驗最重要的部分是透過監控和警示。您需要檢測應用程式以了解其狀態，而且您需要不同的觀點，這表示您需要從伺服器端和用戶端測量，通常使用 Canary。您的指標應包含應用程式與其相依性和維度互動的資料，以符合您的故障隔離界限。您也應該產生日誌，提供應用程式所執行每個工作單位的其他詳細資訊。您可以考慮使用 Amazon CloudWatch 內嵌指標格式等解決方案來結合指標和日誌。您可能會發現您總是想要更多的可觀測性，因此請考慮實作所需檢測層級所需的成本、精力和複雜性權衡。

下列連結提供檢測應用程式和建立警示的最佳實務：

- 監控 Amazon 的生產服務 (AWS re：Invent 2020 簡報）
- Amazon Builders' Library：Amazon 的卓越營運 (AWS re：Invent 2021 簡報）
- Amazon 的可觀測性最佳實務 (AWS re：Invent 2022 簡報）
- 檢測分散式系統以實現操作可見性 (Amazon Builders' Library 文章）
- 建立儀表板以實現營運可見性 (Amazon Builders' Library 文章）

事件管理

當您的警示（或更糟糕的客戶）告訴您發生問題時，您應該有適當的事件管理程序來處理損害。此程序應包括與待命操作員互動、呈報問題，以及建立 Runbook 以取得一致的故障診斷方法，以協助消除人為錯誤。不過，損害通常不會單獨發生；單一應用程式可能會影響依賴它的其他多個應用程式。您可以透過了解所有受影響的應用程式，並在單一電話會議上將來自多個團隊的運算子集合在一起，快速解決問題。不過，視您組織的大小和結構而定，此程序可能需要集中式操作團隊。

除了設定事件管理程序之外，您應該定期透過儀表板檢閱指標。定期審查可協助您了解客戶體驗和應用程式效能的長期趨勢。這可協助您在造成重大生產影響之前識別問題和瓶頸。以一致、標準化的方式檢閱指標可提供顯著的好處，但需要由上而下的支持和時間投資。

下列連結提供建置儀表板和操作指標檢閱的最佳實務：

- 建立儀表板以實現營運可見性 (Amazon Builders' Library 文章）
- Amazon 成功失敗的方法 (AWS re：Invent 2019 簡報）

持續彈性

在階段 2：設計和實作以及階段 3：評估和測試期間，您會在將應用程式部署到生產環境之前啟動檢閱和測試活動。在操作階段期間，您應該繼續在生產環境中反覆執行這些活動。您應該透過 AWS Well-Architected Framework Reviews、Operational Readiness Reviews (ORRs) 和彈性分析架構，定期檢閱應用程式的彈性狀態。這有助於確保您的應用程式不會偏離已建立的基準和標準，並讓您隨時掌握最新或更新的指導方針。這些持續彈性活動可協助您發現先前未預期的中斷，並協助您提出新的緩解措施。

在生產前環境中成功執行遊戲後，您可能也想要考慮在生產環境中執行遊戲日和混沌工程實驗。遊戲日模擬您已建置可緩解的彈性機制的已知事件。例如，遊戲日可能會模擬 AWS 區域服務受損，並實作多區域容錯移轉。雖然實作這些活動可能需要大量精力，但這兩個實務都可協助您建立信心，讓您的系統能夠適應您設計為承受的故障模式。

透過操作您的應用程式、遇到操作事件、檢閱指標和測試應用程式，您將遇到許多回應和學習的機會。

階段 5：回應和學習

應用程式回應中斷事件的方式會影響其可靠性。從經驗中學習，以及您的應用程式在過去如何回應中斷，對於改善其可靠性也至關重要。

回應和學習階段著重於您可以實作的實務，以更好地回應應用程式中的中斷事件。它還包括協助您從營運團隊和工程師的經驗中提取最大學習量的實務。

建立事件分析報告

當事件發生時，第一個動作是盡快避免對客戶和業務造成進一步傷害。應用程式復原後，下一個步驟是了解發生的情況，並識別防止再次發生的步驟。此事件後分析通常會擷取為報告，記錄導致應用程式受損的一組事件，以及中斷對應用程式、客戶和業務的影響。這類報告會成為寶貴的學習成品，並且應該在整個企業中廣泛共用。

> **Note**
>
> 執行事件分析而不指派任何責任至關重要。假設所有運算子都根據他們擁有的資訊採取最佳且最適當的行動。請勿在報告中使用操作員或工程師的名稱。將人為錯誤歸因於損害，可能會導致團隊成員受到保護，以保護自己，導致擷取不正確或不完整的資訊。

與 Amazon Correction of Error (COE) 程序中記錄的良好事件分析報告一樣，遵循標準化格式並嘗試盡可能詳細地擷取導致應用程式受損的條件。報告會詳細說明一系列時間戳記的事件，並擷取量化資料（通常來自監控儀表板的指標和螢幕擷取畫面），以描述應用程式在時間軸上的可測量狀態。報告應擷取採取動作的操作員和工程師的思維程序，以及導致他們做出結論的資訊。報告也應詳細說明不同指標的效能，例如引發哪些警示、這些警示是否準確反映應用程式的狀態、事件與產生的警示之間的時間延遲，以及解決事件的時間。時間軸也會擷取已啟動的 Runbook 或自動化，以及它們如何協助應用程式恢復有用的狀態。時間表的這些元素可協助您的團隊了解自動化和運算子回應的有效性，包括他們解決問題的速度，以及他們在緩解中斷方面的效率。

這個歷史事件的詳細圖片是強大的教育工具。團隊應將這些報告存放在可供整個業務使用的中央儲存庫中，以便其他人可以檢閱事件並從中學習。這可以改善團隊在生產環境中可能出錯的直覺。

詳細事件報告的儲存庫也會成為操作員的訓練材料來源。團隊可以使用事件報告來啟發桌上或現場遊戲日，其中會提供團隊資訊，播放報告中擷取的時間軸。操作員可以使用時間軸中的部分資訊演練案例，

並描述他們將採取的動作。然後，遊戲日的主持人可以提供有關應用程式如何根據操作員動作回應的指導。這可開發運算子的故障診斷技能，以便更輕鬆地預測和故障診斷問題。

負責應用程式可靠性的集中式團隊應將這些報告保存在整個組織可存取的集中式程式庫中。此團隊也應該負責維護報告範本，並訓練團隊如何完成事件分析報告。可靠性團隊應定期檢閱報告，以偵測可透過軟體程式庫、架構模式或團隊程序變更來解決的業務趨勢。

執行操作審查

如階段 4：營運所述，營運審查是檢閱最新功能版本、事件和營運指標的機會。營運審查也是與您組織中更廣泛的工程社群分享功能版本和事件學習的機會。在營運審查期間，團隊會檢閱已復原的功能部署、發生的事件，以及它們的處理方式。這讓整個組織的工程師有機會從其他人的經驗中學習並提出問題。

向貴公司的工程社群開啟營運審查，以便他們可以進一步了解執行業務的 IT 應用程式，以及他們可能遇到的問題類型。他們會在為企業設計、實作和部署其他應用程式時，攜帶這些知識。

檢閱警示效能

如操作階段所述，警示可能會導致儀表板警示、建立票證、傳送電子郵件或分頁運算子。 應用程式將設定許多警示來監控其操作的各個層面。隨著時間的推移，應檢閱這些警示的準確性和有效性，以提高警示精確度、減少誤報，以及合併重複警示。

警示精確度

警示應盡可能具體，以減少您必須花費的時間來解釋或診斷造成警示的特定中斷。當警示因應用程式受損而引發時，接收和回應警示的運算子必須先解譯警示傳遞的資訊。資訊可能是映射到復原程序等動作過程的簡單錯誤代碼，也可能包含應用程式日誌中的行，您必須檢閱這些行來了解引發警示的原因。當您的團隊學習更有效地操作應用程式時，他們應該精簡這些警示，使其盡可能清晰簡潔。

您無法預測應用程式的所有可能中斷，因此一律會有需要操作員分析和診斷的一般警示。您的團隊應該努力減少一般警示的數量，以改善回應時間並縮短平均修復時間 (MTTR)。理想情況下，警示與自動化或人工執行的回應之間應該有one-to-one的關係。

誤報

如果警示不需要來自運算子的動作，但會產生提醒，因為隨著時間的推移，運算子會忽略電子郵件、頁面或票證。 定期或作為事件分析的一部分，檢閱警示以識別經常忽略或不需要運算子採取動作的警示 (誤判)。 您應該努力移除警示，或改善警示，以便向運算子發出可採取動作的警示。

偽陰性

在事件期間，設定為在事件期間提醒的警示可能會失敗，可能是因為事件以非預期的方式影響應用程式。 作為事件分析的一部分，您應該檢閱應該已提出但未提出的警示。您應該努力改善這些警示，以便更好地反映事件可能產生的條件。或者，您可能必須建立其他警示，以映射至相同的中斷，但是由不同的中斷症狀所引發。

重複提醒

影響應用程式的中斷可能會導致多個症狀，並可能導致多個警示。 定期或作為事件分析的一部分，您應該檢閱發出的警示和提醒。 如果操作員收到重複的警示，請建立彙總警示，將其合併為單一警示訊息。

執行指標檢閱

您的團隊應該收集與您應用程式相關的操作指標，例如每月的事件數量、偵測事件的時間、識別原因的時間、修復的時間，以及建立的票證數量、傳送的提醒，以及提出的頁面。至少每月檢閱這些指標一次，以了解營運人員的負擔、他們處理的signal-to-noise（例如資訊警示與可行的警示），以及團隊是否改善其在其控制下操作應用程式的能力。使用此檢閱來了解營運團隊可衡量層面的趨勢。向團隊詢問如何改善這些指標的想法。

提供訓練和啟用

很難擷取導致事件或意外行為的應用程式及其環境的詳細說明。此外，建立應用程式的彈性模型來預測這類案例並不總是簡單明瞭。您的組織應該投資訓練和啟用資料，讓您的營運團隊和開發人員參與彈性建模、事件分析、遊戲日和混沌工程實驗等活動。這將提高團隊所產生報告的逼真度，以及他們所擷取的知識。這些團隊也將更有能力預測失敗，而不需要依賴更小、經驗更豐富的工程師群組，他們必須透過排定的審查來提供洞見。

建立事件知識庫

事件報告是來自事件分析的標準輸出。您應該使用相同或類似的報告來記錄偵測到異常應用程式行為的案例，即使應用程式未受損。使用相同的標準化報告結構來擷取混沌實驗和遊戲日的結果。報告代表應用程式及其環境的快照，導致事件或其他意外行為。您應該將這些標準化報告存放在中央儲存庫，讓企業中的所有工程師都能存取。

然後，營運團隊和開發人員可以搜尋此知識庫，以了解過去中斷應用程式的情況、可能導致中斷的情況類型，以及防止應用程式受損的情況。此知識庫可加速提升營運團隊和開發人員的技能，並讓他們能夠

分享其知識和經驗。此外，您可以使用報告做為遊戲日或混沌實驗的訓練材料或案例，以改善營運團隊的直覺和疑難排解中斷的能力。

> **Note**
>
> 標準化的報告格式也為讀者提供熟悉感，並協助他們更快找到他們正在尋找的資訊。

深度實作彈性

如前所述，進階組織將對警示實作多個回應。我們無法保證回應會有效，因此透過分層回應，應用程式將更能夠正常地失敗。我們建議您為每個指標實作至少兩個回應，以確保個別回應不會成為可能導致 DR 案例的單一失敗點。 這些層應以序列順序建立，因此只有在先前的回應無效時才會執行連續的回應。您不應該對單一警示執行多個分層回應。反之，請使用警示來指出回應是否失敗，如果是，則啟動下一個分層回應。

結論和資源

本指南提供生命週期，透過跨五個階段實作最佳實務，協助您持續改善應用程式的彈性：設定目標、設計和實作、評估和測試、操作和回應和學習。

如需本指南中所討論服務和概念的詳細資訊，請參閱下列資源。

AWS 服務：

- AWS Backup
- AWS Elastic Disaster Recovery
- AWS Fault Injection Service (AWS FIS)
- AWS Resilience Hub
- Amazon Application Recovery Controller (ARC)
- AWS X-Ray

部落格文章：

- 可用性及更高：了解並改善 上分散式系統的彈性 AWS
- AWS 故障隔離界限
- AWS 多區域基本概念
- 雲端中的混沌工程
- 使用 AWS Resilience Hub 和 持續評估應用程式彈性 AWS CodePipeline
- 現場部署應用程式的災難復原 AWS
- 可靠性支柱 – AWS Well-Architected Framework
- 彈性分析架構

貢獻者

本指南的貢獻者包括：

- Bruno Emer，首席解決方案架構師 AWS
- Clari Richey，首席解決方案架構師 AWS
- Elaine Harvey，可靠性服務部門總經理 AWS
- Jason Barto，首席解決方案架構師 AWS
- John Formento，首席解決方案架構師 AWS
- Lisi Lewis，資深產品行銷經理 AWS
- Michael Haken，首席解決方案架構師 AWS
- Neeraj Kumar，首席解決方案架構師 AWS
- Wangechi Doble，首席解決方案架構師 AWS

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	—	2023 年 10 月 6 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

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。

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，請參閱實作微服務 AWS。

Migration Acceleration Program (MAP)

此 AWS 計畫提供諮詢支援、訓練和服務，以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，OT 和資訊技術 (IT) 系統的整合是工業 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

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。

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)

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

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。

重新放置

請參閱 7 Rs。

Replatform

請參閱 7 Rs。

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

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。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。

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

危及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，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，多次讀取。

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。