aws

復原力生命週期

AWS 規定指引

AWS 規定指引: 復原力生命週期

Table of Contents

彈性生命週期框架：持續改善彈性的方法

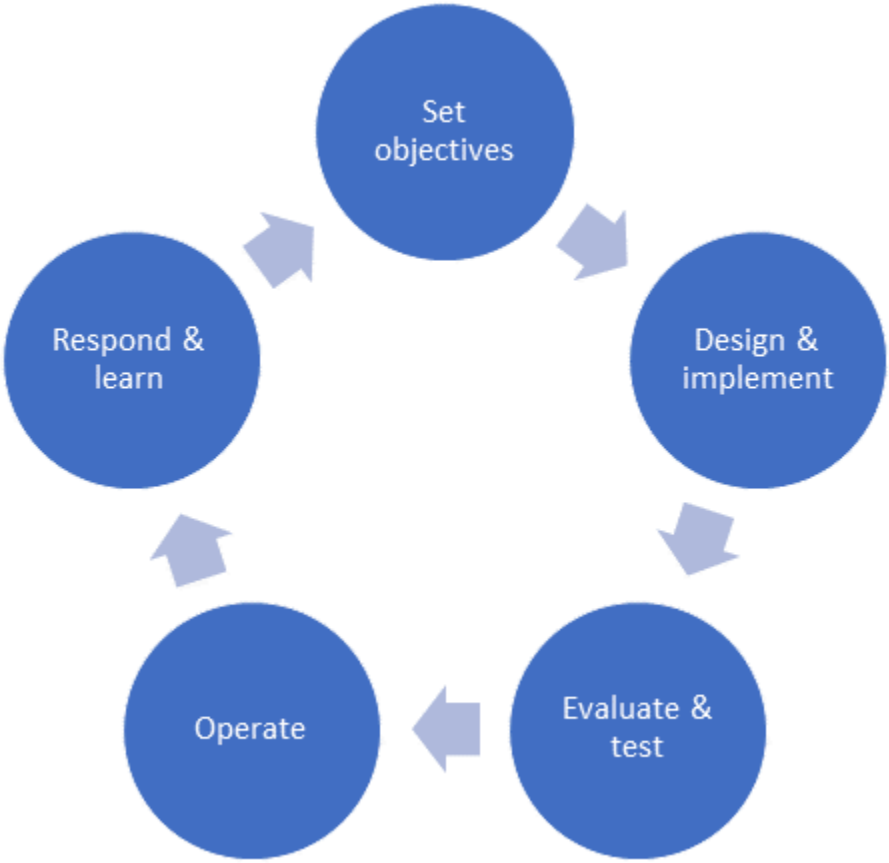
Amazon Web Services (貢獻者)

2023 年十月 (文件歷史記錄)

如今，現代組織面臨著越來越多的彈性相關挑戰，尤其是當客戶的期望轉向始終在線且始終可用的思維方式時。遠端團隊和複雜的分散式應用程式加上對頻繁發行版本的需求不斷增加。因此，組織及其應用程式必須比以往任何時候都更具彈性。

AWS 將彈性定義為應用程式能夠抵抗或從中斷中復原的能力，包括與基礎結構、相依服務、錯誤設定和暫時性網路問題相關的問題。(請參閱 AWS Well-Architected 的架構可靠性支柱文件中的彈性和可靠性元件。) 但是，為了達到所需的彈性水平，通常需要權衡。營運複雜性、工程複雜性和成本將需要進行相應的評估和調整。

基於與客戶和內部團隊多年的合作， AWS 已經開發了一個彈性生命週期框架，以捕獲彈性學習和最佳實踐。該框架概述了下圖所示的五個關鍵階段。在每個階段，您都可以使用策略，服務和機制來改善彈性姿勢。



這些階段將在本指南的以下各節中討論：

- 第 1 階段：設定目標
- 第二階段：設計及實施
- 第三階段：評估及測試
- 第 4 階段：操作
- 第 5 階段：回應和學習

術語和定義

每個階段的彈性概念應用於不同層級，從單個組件到整個系統。實施這些概念需要幾個術語的清晰定義：

- 組件是執行功能的元素，由軟件和技術資源組成。元件的範例包括程式碼設定、基礎結構 (例如網路)，甚至是伺服器、資料存放區，以及外部相依性 (例如多重要素驗證 (MFA) 裝置。
- 應用程式是可提供商業價值的元件集合，例如面向客戶的網站店面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 和區域的多個元件的集合。
- 系統是管理特定業務功能所需的應用程式、人員和程序的集合。它包含執行功能所需的應用程式、持續整合與持續交付 (CI/CD)、可觀察性、組態管理、事件回應和災難復原等作業程序，以及管理此類工作的操作員。
- 中斷是阻止您的應用程式正確交付其業務功能的事件。
- 減值是指中斷對應用程式沒有緩解的影響。如果應用程序遭受一系列中斷，則可能會受到損害。

持續彈性

彈性生命週期是一個持續的過程。即使在同一個組織中，您的應用程式團隊也可能會在每個階段執行不同的完整性層級，視應用程式的需求而定。但是，每個階段的完整性越高，您的應用程序將具有的彈性級別越高。

您應該將復原性生命週期視為組織可以運作的標準程序。 AWS 故意將彈性生命週期建模為類似於軟體開發生命週期 (SDLC)，其目標是在開發和操作應用程式時整合整個作業程序中的規劃、測試和學習。與許多敏捷開發流程一樣，彈性生命週期可以在開發過程的每次迭代中重複。 我們建議您隨著時間的推移逐步加深生命週期的每個階段中的做法。

第 1 階段：設定目標

了解需要什麼級別的彈性以及如何衡量它是設定目標階段的基礎。如果您沒有目標並且無法衡量，那麼很難改進某些東西。

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，請考慮所需的水平，以便進行正確的投資和權衡。一個很好的比喻是一輛汽車：它有四個輪胎，但只攜帶一個備用輪胎。在騎乘過程中獲得多個扁平輪胎的機會很低，而且額外的備件可能會從其他功能（例如貨物空間或燃油效率）中奪走，因此這是一個合理的權衡。

定義目標之後，您可以在稍後階段 (階段 2：設計與實作及階段 4：操作) 實作可觀察性控制項，以瞭解目標是否符合。

映射關鍵應用

定義彈性目標不應該只是一個技術對話。相反，從面向業務的重點開始，以了解應用程序應該提供什麼以及減值的後果。這種對業務目標的理解然後級聯到諸如建築，工程和運營等領域。您定義的任何彈性目標可能會套用至您的所有應用程式，但測量目標的方式通常會因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，而且如果此應用程式受損，您的組織可能會損失大量收入或遭受聲譽損害。或者，您可能有另一個不那麼重要的應用程式，可以容忍一些停機時間，而不會對組織的業務能力造成負面影響。

例如，請考慮零售公司的訂單管理應用程序。如果訂單管理應用程式的元件受損且無法正常執行，則新的銷售將無法完成。這家零售公司還在其中一座建築物中為員工設有一家咖啡店。咖啡廳有一個在線菜單，員工可以在靜態網頁上訪問。如果此網頁變得不可用，有些員工可能會抱怨，但不一定會對公司造成財務損害。根據此示例，企業可能會選擇為訂單管理應用程序設置更積極的彈性目標，但不會投入大量投資來確保 Web 應用程序的彈性。

識別最關鍵的應用程式、應用最多努力的地方，以及在何處進行權衡，與能夠測量應用程式在生產環境中的彈性一樣重要。為了更好地了解減值的影響，您可以執行業務影響分析（BIA）。BIA 提供了一種結構化和系統化的方法來識別關鍵業務應用程序並確定優先級，評估潛在風險和影響，並確定支持依賴關係。BIA 可協助您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功能，將需要多少費用。在上一個範例中，如果訂單管理應用程式受損，零售業務可能會損失大量收入。

映射用戶故事

在 BIA 程序期間，您可能會發現應用程式負責多個業務功能，或者商務功能需要多個應用程式。使用先前的零售公司範例，訂單管理功能可能需要個別的應用程式來進行結帳、促銷和訂價。如果一個應用程序失敗，企業以及與公司互動的用戶可能會感受到影響。例如，公司可能無法添加新訂單，提供促銷和折扣的訪問權限，或更新其產品的價格。訂單管理功能所需的這些不同功能可能依賴於多個應用模組。這些功能也可能具有多個外部依賴關係，這使得實現純粹以組件為中心的彈性過於復雜的過程。處理這種情況的更好方法是專注於用戶故事，其中概述了用戶在與一個應用程序或一組應用程序交互時期望的體驗。

專注於用戶故事可幫助您了解哪些客戶體驗最重要，因此您可以構建機制以抵禦特定威脅。在前面的示例中，一個用戶故事可能是 checkout，其中涉及結帳應用程序，並且依賴於定價應用程序。另一個用戶故事可能是查看促銷活動，其中涉及促銷應用程序。在映射最關鍵的應用程序及其用戶故事之後，您可以開始定義將用於衡量這些用戶故事彈性的指標。這些指標可套用至整個產品組合或個別使用者故事。

定義量測

復原點目標 (RPO)、復原時間目標 (RTO) 和服務層級目標 (SLO) 都是標準的產業測量，用來評估特定系統的彈性。RPO 是指企業在發生故障時可以容忍多少資料遺失，而 RTO 則是衡量應用程式在中斷後必須再次可用的速度。這兩個量度是以時間單位來測量：秒、分鐘和小時。您還可以測量應用程序正常運行的時間長度；也就是說，它按照設計執行其功能，並且可供用戶訪問。這些 SLO 詳細說明客戶將收到的預期服務等級，並透過指標來衡量，例如在回應時間不到一秒的回應時間內未發生錯誤地提供服務的要求百分比 (%) (例如，每月 99.99% 的要求都會收到回應)。 RPO 和 RTO 與嚴重損壞修復策略有關，假設應用程式作業和復原程序會中斷，範圍從還原備份到重新導向使用者流量。SLO 是透過實作高可用性控制來解決，這會減少應用程式的停機時間。

SLO 指標通常用於服務層級協定 (SLA) 的定義中，這些協定是服務供應商與使用者之間的合約。SLA 通常附帶財務承諾和概述處罰，如果不符合這些協議，則需要由提供商支付。但是，SLA 並不是衡量您的彈性姿勢，增加 SLA 並不會使您的應用程式更具彈性。

您可以根據 SLO、RPO 和 RTO 開始設定目標。定義彈性目標並清楚瞭解 RPO 和 RTO 目標之後，您就可以使用AWS Resilience Hub來執行架構評估，以發現與復原相關的潛在弱點。 AWS Resilience Hub 根據 AWS Well-Architected 的架構最佳實務來評估應用程式架構，並在符合您定義的 RTO 和 RPO 目標需要改善的內容中分享補救指引。

建立其他量測

RPO、RTO 和 SLO 都是彈性的良好指標，但您也可以從業務角度思考目標，並定義圍繞應用程式功能的目標。例如，您的目標可能是：如果我的前端和後端之間的延遲增加 40%，每分鐘成功訂單將保持在 98% 以上。或者：即使遺失特定元件，每秒啟動的串流仍會維持在與平均值的標準差內。您也可以建立目標，以縮短已知失敗類型的平均復原時間 (MTTR)；例如：如果發生任何這些已知問題，復原時間將縮短 x%。建立符合業務需求的目標，可協助您預測應用程式應該容忍的失敗類型。它還可以幫助您確定減少應用程序損害可能性的方法。

如果您在遺失 5% 的應用程式執行個體時，考慮繼續運作的目標，您可能會判斷應用程式應該預先調整規模，或是能夠快速擴充以支援在該事件期間造成的額外流量。或者，您可能會決定應該利用不同的架構模式，如「階段 2：設計和實作」一節中所述。

您還應該針對特定的業務目標實施可觀察性措施。例如，您可以追蹤平均訂單率、平均訂購價格、平均訂閱數量或其他指標，這些指標可根據應用程式的行為提供企業運作狀況的深入資訊。透過為應用程式實作可觀察性功能，您可以建立警示，並在這些指標超出您定義的界限時採取行動。「階段 4：操作」部分中更詳細地介紹了可觀察性。

第二階段：設計及實施

在上一階段，您可以設定彈性目標。現在，在設計和實作階段，您會嘗試預測失敗模式並找出設計選擇，並依照您在前一階段設定的目標指引。您也可以定義變更管理的策略，以及開發軟體程式碼和基礎架構組態。以下幾節重點介紹了在考慮成本，複雜性和營運開銷等權衡時應考慮的 AWS 最佳實踐。

AWS Well-Architected 的框架

當您根據所需的彈性目標架構應用程式時，您需要評估多個因素，並在最佳的架構上進行權衡。若要建置高度彈性的應用程式，您必須考慮設計、建置和部署、安全性和作業等方面。AWS Well-Architected 的架構提供一組最佳實務、設計原則和架構模式，以協助您在上設計彈性應用程式。 AWS AWS Well-Architected 的框架的六個支柱提供了設計和操作彈性，安全，高效，具成本效益和可持續性系統的最佳實踐。該框架提供了一種方法，可以根據最佳實踐始終衡量您的架構，並確定需要改進的領域。

以下是 AWS Well-Architected 的架構如何協助您設計和實作符合彈性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，即使在故障或中斷時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，使您的應用程式更小、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。您還可以通過使用節流，重試指數退回，快速故障（負載脫落），冪等性，恆定工作，斷路器和靜態穩定性來獲得構建應用程序的最佳實踐的詳細說明。
- 全面審查： AWS Well-Architected 的框架鼓勵針對最佳實踐和設計原則對您的架構進行全面審查。它提供了一種持續測量架構並識別需要改進的領域的方法。
- 風險管理： AWS Well-Architected 的架構可協助您識別和管理可能影響應用程式可靠性的風險。透過主動解決潛在故障情況，您可以降低其可能性或由此產生的損害。
- 持續改進：復原力是一個持續的過程， AWS Well-Architected 的框架強調持續改進。根據 AWS Well-Architected Framework 的指引，定期檢閱和改良您的架構和程序，您可以確保您的系統在面對不斷變化的挑戰和需求時保持彈性。

瞭解相依性

了解系統的依賴關係是恢復能力的關鍵。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 (例如協力廠商 API 和企業擁有的共用服務) 的連線。瞭解這些連接可協助您隔離和管理中斷，因為某個元件中的損壞可能會影響其他元件。這些知識可以幫助工程師評估減損的影響並相應地進行規劃，並確保資源得到有效使用。瞭解相依性可協助您建立替代策略並協調復原程序。它也可協助您判斷可以用軟相依性取代硬式相依性的案例，如此一來，您的應用程式可以在發生相依性損害時繼續提供業

務功能。相依性也會影響負載平衡和應用程式擴展的決策。當您對應用程式進行變更時，瞭解相依性非常重要，因為它可以協助您判斷潛在的風險和影響。這些知識可協助您建立穩定、彈性的應用程式，協助進行錯誤管理、影響評估、減損復原、負載平衡、擴充和變更管理。您可以手動追蹤相依性，或使用工具和服務，例如瞭AWS X-Ray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重要角色，主要是確保業務連續性。它保證了即使在災難性事件發生時，重要的業務運營也能以最少的可能損失持續存在，從而最大程度地減少停機時間和潛在的收入損失。災難復原策略對於資料保護至關重要，因為它們通常會整合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，並協助防止災難發生時完全遺失。此外，許多行業都受到政策的監管，這些政策要求企業制定適當的災難恢復策略來保護敏感數據並確保服務在災難期間仍然可用。藉由確保最小的服務損害，DR 策略也增強了客戶的信任和滿意度。實作良好且經常實作的災難復原策略可縮短災難發生後的復原時間，並協助確保應用程式能夠快速恢復上線。此外，災難可能會產生可觀的成本，不僅是因為停機而造成的收入損失，還可能會因為還原應用程式和資料所造成的費用。精心設計的 DR 策略有助於抵禦這些財務損失。

您選擇的策略取決於應用程式的特定需求、RTO 和 RPO，以及您的預算。 AWS Elastic Disaster Recovery是專門建置的復原服務，可用來協助實作內部部署和雲端式應用程式的 DR 策略。

如需詳細資訊，請參閱 AWS 網站上的「工作負載的災難復原」 AWS和「AWS 多區域基礎知識」。

定義CI/CD 策略

造成應用程式損壞的常見原因之一是程式碼或其他變更，會改變應用程式從先前已知的工作狀態。如果您不小心處理變更管理，它可能會導致頻繁的損害。變更的頻率會增加產生影響的機會。但是，較少進行變更的頻率會導致較大的變更集，因為變更集的複雜性較高，因此更有可能導致減值。持續整合和持續交付 (CI/CD) 實務的設計目的是讓變更保持小而頻繁 (從而提高生產力)，同時透過自動化讓每項變更都能達到高水準的檢驗。一些基本策略是：

- 完全自動化：CI/CD 的基本概念是盡可能自動化構建和部署過程。這包括建置、測試、部署，甚至監控。自動化管道有助於減少人為錯誤的可能性，確保一致性，並使流程更加可靠和高效。
- 測試驅動開發（TDD）：在編寫應用程序代碼之前編寫測試。這項做法可確保所有程式碼都有相關的測試，進而改善程式碼的可靠性和自動化檢測的品質。這些測試在 CI 管道中運行以驗證更改。
- 經常提交和集成：鼓勵開發人員經常提交代碼並經常執行集成。較小且頻繁的變更更易於測試和除錯，從而降低發生重大問題的風險。自動化降低了每次提交和部署的成本，從而使頻繁的集成成為可能。

- 不可變基礎結構：處理您的伺服器和其他基礎結構元件，例如靜態、不可變的實體。取代基礎架構，而不是盡可能地修改它，並透過經過測試並透過管道部署的程式碼來建置新的基礎結構。
- 回滾機制：始終有一種簡單，可靠且經常測試的方法來回滾更改，如果出現問題。能夠快速返回先前已知的良好狀態對於部署安全至關重要。這可以是恢復到先前狀態的簡單按鈕，也可以完全自動化並由警報啟動。
- 版本控制：在版本控制的存儲庫中將所有應用程序代碼，配置甚至基礎結構作為代碼維護。此做法有助於確保您可以輕鬆追蹤變更，並在需要時還原變更。
- Canary 部署和藍/綠部署：首先將新版本的應用程式部署到基礎結構的子集，或維護兩個環境 (藍/綠)，可讓您驗證生產中的變更行為，並在必要時快速復原。

CI/CD 不僅與工具有關，而且與文化有關。創建一種重視自動化，測試和從失敗中學習的文化與實施正確的工具和流程同樣重要。如果在影響最小的情況下迅速完成回溯，則不應將其視為失敗，而是一種學習經驗。

執行 ORR

操作準備程度審查（ORR）有助於識別操作和程序上的差距。在 Amazon，我們建立了 ORR，透過最佳實務指導，將經營大規模服務數十年的經驗提煉成精心策劃的問題。ORR 捕獲先前學到的經驗教訓，並需要新的團隊以確保他們已經在其應用程序中考慮了這些課程。ORR 可以提供故障模式或失敗原因的清單，這些清單可以帶入以下彈性建模一節中描述的彈性建模活動中。如需詳細資訊，請參閱 AWS Well-Architected 的架構網站上的作業準備檢閱 (ORR)。

了解 AWS 故障隔離界限

AWS 提供多重故障隔離界限，協助您達成彈性目標。您可以使用這些界限來利用它們提供的可預測影響限制範圍。您應該熟悉如何使用這些界限來設計 AWS 服務，以便有意針對您為應用程式選取的相依性做出有意的選擇。若要瞭解如何在應用程式中使用邊界，請參閱 AWS 網站上的AWS 錯誤隔離邊界。

選取回應

系統可以通過多種方式對警報進行響應。某些警示可能需要作業團隊的回應，而其他警示可能會在應用程式內觸發自我修復機制。您可能會決定將可自動化的回應保留為手動作業，以控制自動化成本或管理工程限制。警報的回應類型可能會被選為執行回應成本的函數、警報的預期頻率、警報的準確度，以及完全沒有回應警示的潛在業務損失。

例如，當伺服器處理序當機時，作業系統可能會重新啟動該處理程序，或者可能會佈建新伺服器，而舊伺服器則終止，或者可能會指示操作員遠端連線至伺服器並重新啟動伺服器。這些回應的結果相同，即重新啟動應用程式伺服器處理序，但實作和維護成本的層級不同。

> **Note**
>
> 您可以選擇多個響應以採取深入的彈性方法。例如，在先前的案例中，應用程式小組可能會選擇實作所有三個回應，每個回應之間的時間延遲。如果失敗的伺服器處理序指示器在 30 秒後仍處於警告狀態，則該群組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 auto Scaling 群組來建立新的虛擬伺服器，並還原應用程式伺服器處理序。如果指示器在 300 秒後仍處於警報狀態，則可能會向操作人員發送警報以連接到原始服務器並嘗試恢復該過程。

應用團隊和業務選擇的回應應該反映企業對工程時間的前期投資來抵消營運開銷的需求。您應該仔細考慮每個響應選項的限制和預期的維護，以選擇響應-例如靜態穩定性，軟件模式（例如斷路器）或操作程序。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所管理的程式庫和模式作為此考量的輸入。

韌性建模

彈性模型記錄了應用程式將如何應對不同的預期中斷。 透過預測中斷情況，您的團隊可以實作可觀察性、自動化控制和復原程序，以減輕或預防損害，即使發生中斷。 AWS 通過使用彈性分析框架創建了開發彈性模型的指導。 此架構可協助您預測中斷情況及其對應用程式的影響。 藉由預測中斷，您可以識別建置彈性、可靠應用程式所需的緩和措施。 我們建議您使用彈性分析架構，隨著應用程式生命週期的每次迭代更新您的彈性模型。 在每次反覆運用此架構，藉由預測設計階段中斷，並在生產部署之前和之後測試應用程式，有助於減少事件。 使用此框架開發彈性模型可幫助您確保滿足彈性目標。

安全失敗

如果您無法避免中斷，請安全地失敗。請考慮使用預設的故障安全作業模式來建立應用程式，不會造成重大的業務損失。數據庫的故障安全狀態的一個例子是默認為只讀操作，其中用戶不允許創建或更改任何數據。視資料的敏感度而定，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。請考慮應用程式的故障安全狀態，並且在極端條件下預設為此操作模式。

第三階段：評估及測試

在生命週期的評估和測試階段，應用程式或對現有應用程式的變更已設計，但尚未發行到生產環境中。在這個階段中，您實作活動，以測試在先前階段已執行的做法，並評估結果。應用程式可能仍在進行中開發，或者主要開發可能已完成，而且應用程式可能會在發佈到生產環境之前進行測試。在此階段，您專注於開發和運行測試，以確認或反駁應用程序將滿足定義的彈性目標的期望。此外，您還可以開發和測試系統的操作程序。您在「階段 2：設計與實作」階段所開發的建置程序已付諸實踐，並評估結果。儘管這些測試和評估活動在生命週期的這一部分開始，但它們不會在此結束。進入「階段 4：操作」階段時，測試和評估會繼續進行。

評估和測試階段分為兩個階段：部署前活動和部署後活動。預先部署活動包含應用程式部署到任何環境之前應完成的工作，包括將軟體的新版本部署以及初始部署至測試環境。部署後的活動會在軟體部署到測試或生產環境之後進行。以下各節將更詳細地討論這些階段。

部署前活動

環境設計

您在其中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在生產環境中會發生的情況有多大的信心。您也許可以使用 Amazon DynamoDB 等服務，在開發人員電腦上在本機執行某些整合測試 (請參閱 DynamoDB 文件中的本機設定 DynamoDB)。但是，在某些時候，您需要在複製生產環境的環境中進行測試，以便在結果中獲得最高的信心。這個環境會產生成本，因此我們建議您採取分段或流水線化的方法來處理環境，其中生產環境稍後會出現在管線中。

集成測試

集成測試是測試一個應用程序的一個明確定義的組件，當它與外部依賴運行正確執行其功能的過程。這些外部相依性可能是其他自訂開發的元件、您用於應用程式的 AWS 服務、協力廠商相依性，以及內部部署相依性。 本指南著重於展示應用程式彈性的整合測試。它假設單元和集成測試已經存在，以證明軟件的功能準確性。

我們建議您設計整合測試，以專門測試您已實作的彈性模式，例如斷路器模式或負載脫落 (請參閱階段 2：設計與實作)。面向復原的整合測試通常涉及將特定負載套用至應用程式，或是使用 () 等功能，故意將中斷引入環境。AWS Fault Injection ServiceAWS FIS理想情況下，您應該將所有集成測試作為 CI/CD 管道的一部分運行，並確保每次提交代碼時都運行測試。這可協助您快速偵測並回應任何導致違反彈性目標的程式碼或組態變更。大規模的分散式應用程式很複雜，即使是微小的變更，也可能會大幅

影響應用程式看似不相關部分的彈性。嘗試在每次提交上運行測試。 AWS 為操作 CI/CD 管道和其他 DevOps 工具提供了一套出色的工具。如需詳細資訊，請參閱 AWS 網站 DevOps AWS上的簡介。

自動化部署管道

在生產前環境中進行部署和測試是一項重複且複雜的任務，最好留給自動化。此過程的自動化可以釋放人力資源並減少錯誤的機會。自動化此程序的機制通常稱為管線。建立管道時，建議您設定一系列越來越接近生產組態的測試環境。您可以使用這一系列環境來重複測試您的應用程式。第一個環境提供的功能組比生產環境更有限，但成本大幅降低。後續環境應新增服務並進行擴充，以更緊密地反映生產環境。

首先在第一個環境中進行測試。在您的部署通過第一個測試環境中的所有測試之後，請讓應用程式在一定程度的負載下執行一段時間，以查看是否有任何問題隨著時間的推移發生。確認您已正確設定可觀測性 (請參閱本指南後面的警示精確度)，以便您可以偵測出現的任何問題。成功完成此觀察期後，請將應用程式部署到下一個測試環境，然後重複此程序，並根據環境支援新增其他測試或負載。以這種方式對應用程式進行充分測試之後，您就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中 (請參閱本指南稍早的定義 CI/CD 策略)。文章自動化 Amazon Builders' Library 中的安全、免動手部署是一項很好的資源，描述 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，具體取決於應用程式的複雜性及其具有的相依性類型。

負載測試

從表面上看，負載測試類似於集成測試。您可以測試應用程式及其外部相依性的離散函式，以確認其如預期般運作。負載測試然後超越集成測試，專注於明確定義的負載下的應用程序的功能。負載測試需要驗證正確的功能，因此它必須在成功的集成測試後發生。重要的是要了解應用程序在預期負載下的響應以及當負載超出預期時它的行為是非常重要的。這可協助您驗證是否已實作必要的機制，以確保應用程式在極端負載下保持彈性。有關負載測試的綜合指南 AWS，請參閱 AWS 解決方案庫 AWS中的分佈式負載測試。

部署後活動

彈性是一個持續的程序，應用程式的彈性評估必須在部署應用程式之後繼續進行。部署後活動的結果 (例如持續的彈性評估) 可能需要您重新評估並更新您先前在復原生命週期中執行的一些彈性活動。

進行彈性評估

將應用程式部署到生產環境後，評估彈性並不會停止。即使您已經定義良好且自動化的部署管線，有時候變更可能會直接在生產環境中發生。此外，在部署前恢復性驗證中，可能還未考慮到某些因素。

AWS Resilience Hub提供一個集中的位置，您可以在其中評估部署的架構是否符合定義的 RPO 和 RTO 需求。您可以使用此服務執行應用程式彈性的隨選評估、自動化評估，甚至將其整合到 CI/CD 工具中，如 AWS 部落格文章中所述持續使用和評估應用 AWS Resilience Hub 程式恢復性。 AWS CodePipeline自動化這些評估是最佳做法，因為它有助於確保您持續評估生產中的彈性姿勢。

DR 測試

在第 2 階段：設計和實作中，您開發了災難復原 (DR) 策略，做為系統的一部分。在第 4 階段，您應該測試災難復原程序，以確保團隊已為事件做好充分準備，並且您的程序如預期般運作。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，以判斷是否應該更新系統程序，以獲得最佳結果。當您最初開發 DR 測試時，請預先安排測試，並確保整個團隊瞭解預期的結果、測量結果的方式，以及將使用哪些反饋機制根據結果更新程序。在您熟練執行排定的 DR 測試之後，請考慮執行未宣告的 DR 測試。真正的災難不會按計劃發生，因此您需要隨時做好計劃的準備。但是，未經宣布並不意味著計劃外。主要利益相關者仍需規劃活動，以確保進行適當的監控，並且不會對客戶和關鍵應用程式造成不利影響。

漂移偵測

即使已進行自動化和明確定義的程序，生產應用程式中的組態也可能發生意想不到的變更。要檢測應用程序配置的更改，您應該具有檢測漂移的機制，這是指與基線配置的偏差。要了解如何檢測 AWS CloudFormation 堆棧中的漂移，請參閱 AWS CloudFormation 文檔中的檢測堆棧和資源的非託管配置更改。若要偵測應用程式 AWS 環境中的漂移，請參閱 AWS Control Tower 文件AWS Control Tower 中的偵測和解決漂移。

合成測試

綜合測試是建立在生產環境中執行的可設定軟體的程序，以模擬使用者體驗的方式測試應用程式的 API。這些測試有時被稱為金絲雀，參考該術語在煤礦開採中的原始用途。綜合測試通常可以在應用程序遭受中斷時提供早期警告，即使損害是部分或間歇性的，通常是灰色故障的情況。

混沌工程

混沌工程是一個系統化的過程，涉及故意使應用程序以降低風險的方式進行破壞性事件，密切監控其響應並實施必要的改進。其目的是驗證或挑戰有關應用程序處理此類中斷能力的假設。混沌工程不會讓這些事件發生機會，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，以有效緩解措施。

混沌工程始於了解正在考慮的應用程序的正常操作條件（稱為穩定狀態）。從那裡，您制定了一個假設，詳細說明應用程序在中斷存在的情況下的成功行為。 您執行實驗，其中包括故意注入中斷，包括

但不限於網路延遲、伺服器故障、硬碟錯誤，以及外部相依性的損害。然後，您可以分析實驗結果，並根據您的學習增強應用程序的彈性。該實驗是改善應用程序的各個方面（包括其性能）的寶貴工具，並發現了可能仍然隱藏的潛在問題。此外，混沌工程有助於揭示可觀察性和警報工具的缺陷，並幫助您改進它們。它還有助於減少恢復時間並提高操作技能。混沌工程加速了最佳實踐的採用，並培養持續改進的心態。最終，它使團隊能夠通過定期練習和重複來建立和磨練他們的操作技能。

AWS 建議您在非生產環境中開始混亂的工程工作。您可以使用 AWS Fault Injection Service (AWS FIS) 執行具有一般用途錯誤的混沌工程實驗，以及特有的錯誤。 AWS這項全受管服務包括停止狀態警報和完整權限控制，因此您可以輕鬆安全地採用混亂的工程設計。

第 4 階段：操作

完成階段 3：評估和測試之後，您就可以將應用程式部署到生產環境。在「操作」階段，您可以將應用程式部署到生產環境中，並管理客戶的體驗。 您的應用程式的設計和實作決定了許多彈性結果，但是這個階段著重於系統用來維護和改善彈性的作業實務。建立卓越的營運文化有助於在這些做法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通過監控和警報。您需要檢測應用程序以了解其狀態，並且需要多樣化的觀點，這意味著您需要從服務器端和客戶端進行測量，通常使用 Canary 進行測量。您的指標應包含應用程式與其相依性的互動相關資料，以及符合錯誤隔離界限的維度。您也應該產生記錄檔，提供應用程式所執行之每個工作單元的其他詳細資料。您可以考慮使用 Amazon CloudWatch 內嵌指標格式等解決方案來合併指標和日誌。您可能會發現您總是需要更多的可觀察性，因此請考慮實施所需的儀器級別所需的成本，精力和複雜性權衡。

以下連結提供檢測應用程式和建立警示的最佳做法：

- 監控 Amazon 的生產服務（AWS 重新：發明 2020 演示文稿）
- Amazon Builders' Library：Amazon 的卓越運營（AWS RE：發明 2021 演示文稿）
- Amazon 的可觀察性最佳實踐（AWS RE：發明 2022 演示文稿）
- 檢測分散式系統以獲得操作可見性（Amazon Builders' Library 文章）
- 構建儀表板以提高操作可見性（Amazon Builders' Library 文章）

事件管理

當您的警報（或更糟的是，您的客戶）告訴您出了問題時，您應該有一個事件管理流程來處理障礙。此程序應包括聘請隨時待命的操作員、升級問題，以及建立 Runbook，以提供一致的疑難排解方法，協助消除人為錯誤。但是，損害通常不會孤立發生；單個應用程序可能會影響依賴它的多個其他應用程序。您可以透過瞭解受到影響的所有應用程式，並將來自多個團隊的操作員聚集在一次電話會議中，快速解決問題。不過，視您組織的規模和結構而定，此程序可能需要集中式作業團隊。

除了設定事件管理程序之外，您還應該透過儀表板定期檢閱指標。定期檢閱可協助您瞭解應用程式效能的客戶體驗和長期趨勢。這有助於您在問題和瓶頸造成重大生產影響之前找出問題和瓶頸。以一致的標準化方式審查指標可提供顯著的好處，但需要自上而下的買入和投資時間。

下列連結提供建立儀表板和操作指標檢閱的最佳做法：

- 構建儀表板以提高操作可見性（Amazon Builders' Library 文章）
- 亞馬遜成功失敗的方法（AWS 重新：發明 2019 演示文稿）

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，您在將應用程式部署到生產環境之前，先啟動檢閱和測試活動。在操作階段，您應該繼續在生產環境中對這些活動進行迭代。您應該透過 AWS Well-Architected 的架構檢閱、作業準備檢閱 (ORR) 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既定的基準和標準，並讓您隨時掌握最新或更新的指引。這些持續的彈性活動可協助您發現先前意想不到的中斷情況，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日和混沌工程實驗之後，您可能還想考慮在生產環境中執行這些實驗。遊戲日會模擬您已建立彈性機制以緩解的已知事件。例如，遊戲日可能會模擬區 AWS 域服務損壞，並實作多區域容錯移轉。雖然實作這些活動可能需要大量的努力，但這兩種做法都可以協助您建立自信，確保系統能夠抵禦您設計的故障模式。

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

第 5 階段：回應和學習

您的應用程式如何回應破壞性事件會影響其可靠性。從經驗中學習，以及您的應用程式過去如何回應中斷，對於提高其可靠性也至關重要。

「回應並學習」階段著重於您可以實作的實務，以便更好地回應應用程式中的顛覆性事件。它還包括幫助您從運營團隊和工程師的經驗中獲得最大程度的學習的實踐。

建立事件分析報告

當事件發生時，第一個行動是盡快防止對客戶和業務造成進一步的傷害。應用程序恢復後，下一步是了解發生了什麼並確定步驟以防止再次發生。此事件後分析通常會擷取為報告，記錄導致應用程式損害的一組事件，以及中斷對應用程式、客戶和業務的影響。這些報告成為有價值的學習成品，應該在整個企業中廣泛共享。

> **Note**
>
> 在不指派任何責任的情況下執行事件分析至關重要。假設所有運營商都採取了最好的，最合適的行動方案給出了他們所擁有的信息。請勿在報告中使用運算子或工程師的名稱。引用人為錯誤作為損害的原因可能會導致團隊成員受到保護以保護自己，從而導致捕獲不正確或不完整的信息。

良好的事件分析報告 (例如 Amazon 錯誤校正 (COE) 程序中所記載的報告，遵循標準化的格式，並嘗試盡可能詳細地擷取導致應用程式受損的情況。該報告詳細說明了一系列時間戳記的事件，並捕獲定量數據（通常是監控儀表板中的指標和屏幕截圖），這些數據描述了時間線上應用程序的可衡量狀態。該報告應該捕獲採取行動的操作員和工程師的思想過程，以及導致他們得出結論的信息。該報告還應詳細說明不同指示器的性能，例如引發了哪些警報，這些警報是否準確反映了應用程序的狀態，事件與產生的警報之間的時間滯後，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重新取得有用的狀態。時間表的這些元素可協助您的團隊瞭解自動化和操作員回應的有效性，包括解決問題的速度，以及他們在緩解中斷時的效率。

這個歷史事件的詳細圖片是一個強大的教育工具。團隊應將這些報告存儲在可供整個企業使用的中央存儲庫中，以便其他人可以查看事件並從中學習。這可以提高團隊對生產中可能出錯的事情的直覺。

詳細的事故報告儲存庫也成為操作人員訓練材料的來源。團隊可以使用事件報告來激發桌面或現場比賽日的靈感，在這一天中，團隊可以獲得回放報告中捕獲的時間表的信息。操作員可以使用時間表中的部

分資訊來逐步完成案例，並描述他們將採取的動作。然後，遊戲當天的主持人可以根據操作員的操作提供應用程序如何響應的指導。這樣可以開發操作員的故障排除技能，因此他們可以更輕鬆地預測問題並進行故障排除。

負責應用程式可靠性的集中式團隊應該在整個組織都可以存取的集中式程式庫中維護這些報告。該團隊還應負責維護報告模板和培訓團隊如何完成事件分析報告。可靠性團隊應定期檢閱報告，以偵測整個企業可透過軟體程式庫、架構模式或團隊程序變更來解決的趨勢。

進行操作審查

如第 4 階段：操作中所述，作業檢閱是檢閱最近發行的功能、事件和作業指標的機會。操作審查也是一個機會，可以與組織中更廣泛的工程社群分享功能發布和事件中的經驗。在作業檢閱期間，團隊會檢閱已復原的功能部署、發生的事件以及處理方式。這使整個組織的工程師有機會從他人的經驗中學習並提出問題。

向公司的工程社群開啟您的作業檢閱，以便他們深入瞭解執行業務的 IT 應用程式，以及可能遇到的問題類型。他們在為企業設計、實作和部署其他應用程式時，會隨身攜帶這些知識。

檢閱警示效能

如操作階段所討論的警示，可能會導致儀表板警示、建立工單、傳送電子郵件或分頁操作員。 一個應用程序將有許多警報配置為監視其操作的各個方面。隨著時間的推移，應該檢查這些警報的準確性和有效性，以提高警報的精確度，減少誤報，並合併重複的警報。

報警精度

警報應盡可能具體，以減少您必須花費在解釋或診斷導致警報的特定中斷時間。當警示因應應用程式損壞而引發時，接收並回應警示的操作員必須首先解釋警示所傳達的資訊。這些資訊可能是一個簡單的錯誤碼，可對應至復原程序之類的動作方案，或者可能包含應用程式記錄檔中的行，您必須檢閱這些行，以瞭解引發警示的原因。當您的團隊學習更有效地操作應用程序時，他們應該優化這些警報以使其盡可能清晰簡潔。

您無法預期應用程式的所有可能中斷，因此總會有一般警示需要操作員進行分析和診斷。您的團隊應該努力減少一般警報的次數，以縮短回應時間並縮短平均修復時間 (MTTR)。理想情況下，警報和自動化或人為執行的響應之間應該存在 one-to-one 關係。

誤報

運營商不需要操作員採取任何操作，但在電子郵件，頁面或工單中產生警報的警報將隨著時間的推移忽略。 定期或作為事件分析的一部分，檢閱警示，以識別那些經常被忽略或不需要操作員採取任何動作的警示 (誤報)。 您應該努力刪除警報，或改善警報，以便向操作員發出可操作的警報。

假底片

在事件發生期間，設定為在事件期間發出警示的警示可能會失敗，這可能是因為事件會以非預期的方式影響應用程式。 作為事件分析的一部分，您應該查看應該提出但沒有提出的警報。您應該努力改善這些警報，以便更好地反映事件可能產生的條件。或者，您可能必須建立額外的警示，以對應至相同的中斷，但會因為中斷的不同徵狀而引發。

重複警報

損害應用程式的中斷可能會導致多種症狀，並可能導致多個警報。 您應該定期檢閱已發出的警示和警示，或作為事件分析的一部分。 如果操作員收到重複警示，請建立彙總警示，以將其合併為單一警示訊息。

進行指標審查

您的團隊應該收集與應用程式相關的操作指標，例如按每月嚴重性分類的事件數目、偵測事件的時間、識別原因的時間、修復的時間，以及建立的工單數量、傳送警示以及引發的頁面。至少每月檢閱這些指標，以瞭解營運人員的負擔、他們處理的 signal-to-noise 比例 (例如資訊警示與可採取動作的警示)，以及團隊是否正在改善其控制下操作應用程式的能力。使用此檢閱瞭解營運團隊可衡量方面的趨勢。徵求團隊對如何提高這些指標的想法。

提供培訓和能力

很難捕獲導致事件或意外行為的應用程序及其環境的詳細描述。此外，建模應用程序的彈性以預測此類情況並不總是簡單的。您的組織應該投資訓練和支援材料，以供您的營運團隊和開發人員參與活動，例如復原力建模、事件分析、遊戲日和混沌工程實驗。這將提高團隊生成的報告的真實性以及他們捕獲的知識。這些團隊也將變得更有能力預測失敗，而無需依賴規模較小，更有經驗的工程師團隊，他們必須通過計劃的審查來提供他們的見解。

建立事件知識庫

事件報告是事件分析的標準輸出。您應該使用相同或類似的報表來記錄偵測到異常應用程式行為的案例，即使應用程式沒有受損也是如此。使用相同的標準化報告結構來捕捉混沌實驗和比賽日的結果。此報告代表應用程式及其環境的快照，可導致事件或其他非預期行為。您應該將這些標準化報告儲存在企業內所有工程師都可以存取的中央儲存庫中。

然後，操作團隊和開發人員可以搜索此知識庫，以了解過去哪些情況中斷了應用程序，哪些類型的案例可能導致中斷，以及防止應用程序損害的原因。該知識庫成為提高營運團隊和開發人員技能的加速器，並使他們能夠分享他們的知識和經驗。此外，您還可以使用這些報告作為比賽日或混亂實驗的訓練材料或場景，以提高操作團隊的直覺性和故障排除中斷的能力。

> **Note**
>
> 標準化的報告格式還為讀者提供了一種熟悉感，並幫助他們更快地找到所需的信息。

深入實施彈性

如前所述，進階組織將對警示實作多個回應。無法保證響應將是有效的，因此通過分層響應，應用程序將更好地裝備以優雅地失敗。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。 這些層應該以串行順序創建，以便只有在先前的響應無效時才執行連續響應。您不應該對單個警報運行多個分層響應。請改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

本指南提供的生命週期可協助您跨五個階段實作最佳實務，以協助您持續改善應用程式的彈性：設定目標、設計與實作、評估與測試、操作以及回應與學習。

如需有關本指南所討論之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 路線 53 應用程序恢復控
- AWS X-Ray

博客文章和文章：

- 可用性和超越：了解和提高分佈式系統的彈性 AWS
- AWS 故障隔離邊界
- AWS 多區域基礎
- 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
- 內部部署應用程式的災難復原 AWS
- 可靠性支柱 — AWS Well-Architected 的框架
- 彈性分析框架

貢獻者

本指南的貢獻者包括：

- 布魯諾·埃默，首席解決方案架構師， AWS
- 克拉克·里奇, 首席解決方案架構師, AWS
- 可靠性服務總經理艾琳·哈維 AWS
- 傑森·巴托，首席解決方案架構師， AWS
- 約翰·福門托，首席解決方案架構師， AWS
- 李西·劉易斯，高級產品營銷經理， AWS
- 邁克爾·哈肯，首席解決方案架構師， AWS
- 下一個庫馬爾, 首席解決方案架構師, AWS
- 萬吉多布雷, 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 規定指引詞彙

以下是 AWS 規範性指引所提供的策略、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 資料庫遷移到與 Amazon Aurora PostgreSQL 相容的版本。
- 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：將您的現場部署 Oracle 資料庫遷移至 Amazon Relational Database Service 服務 (Amazon RDS)，適用於 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 資料庫 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應用程式移轉至 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

請參閱以屬性為基礎的存取控制。

抽象的服務

請參閱受管理服務。

酸

請參閱原子性、一致性、隔離性、耐用性。

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。它比主動-被動遷移更具彈性，但需要更多的工作。

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。

聚合函數

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

AI

請參閱人工智慧。

艾奧運

請參閱人工智慧作業。

匿名化

永久刪除資料集中個人資訊的程序。匿名化可以幫助保護個人隱私。匿名資料不再被視為個人資料。

反模式

一種經常使用的解決方案，用於解決方案的生產力適得其反，效果不佳或效果低於替代方案。

應用控制

一種安全性方法，只允許使用核准的應用程式，以協助保護系統免受惡意軟體的攻擊。

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 AWS中的 ABAC。

授權資料來源

儲存資料主要版本的位置，被認為是最可靠的資訊來源。您可以將授權資料來源中的資料複製到其他位置，以便處理或修改資料，例如匿名化、編輯或將其虛擬化。

可用區域

一個獨立的位置，與其他 AWS 區域 可用區域中的故障隔離，並為相同區域中的其他可用區域提供廉價、低延遲的網路連線能力。

AWS 雲端採用架構 (AWS CAF)

指導方針和最佳做法的架構，可協 AWS 助組織制定有效率且有效的計畫，以順利移轉至雲端。AWS CAF 將指導組織到六個重點領域，稱為觀點：業務，人員，治理，平台，安全性和運營。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。針對此觀點，AWS CAF 為人員開發、訓練和通訊提供指導，以協助組織為成功採用雲端做好準備。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)

可評估資料庫移轉工作負載、建議移轉策略並提供工作預估的工具。 AWS WQF 包含在 AWS Schema Conversion Tool ()AWS SCT中。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

壞機器人

旨在破壞或對個人或組織造成傷害的機器人。

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 「位元順序」。

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

建立兩個獨立但相同環境的部署策略。您可以在一個環境中執行目前的應用程式版本 (藍色)，而在另一個環境 (綠色) 中執行新的應用程式版本。此策略可協助您以最小的影響快速回復。

機器人

透過網際網路執行自動化工作並模擬人類活動或互動的軟體應用程式。某些漫遊器是有用的或有益的，例如用於索引 Internet 上信息的網絡爬蟲。其他一些機器人（稱為不良機器人）旨在破壞或對個人或組織造成傷害。

殭屍網絡

受惡意軟件感染並受到單一方（稱為機器人牧民或機器人操作員）控制的機器人網絡。殭屍網絡是擴展機器人及其影響的最著名機制。

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)。

防碎玻璃訪問

在特殊情況下，並透過核准的程序，使用者可以快速取得他 AWS 帳戶 們通常沒有存取權限的存取權。如需詳細資訊，請參閱 AWS Well-Architected 指南中的實作防破玻璃程序指標。

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

咖啡

請參閱AWS 雲端採用架構。

金絲雀部署

向最終用戶發行版本的緩慢和增量版本。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

請參閱雲端卓越中心。

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來測試系統的彈性。您可以使用 AWS Fault Injection Service (AWS FIS) 執行實驗來 stress 您的 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。

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。雲計算通常連接到邊緣計算技術。

雲端運作模式

在 IT 組織中，這是用來建置、成熟和最佳化一或多個雲端環境的作業模型。如需詳細資訊，請參閱建立您的雲端作業模型。

採用雲端階段

組織移轉至下列四個階段時通常會經歷 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

這些階段是 Stephen Orban 在 AWS 雲端 企業策略部落格部落格文章「邁向雲端優先的旅程與採用階段」中所定義的。如需其與 AWS 移轉策略之間關聯的詳細資訊，請參閱移轉準備指南。

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。常見的雲存儲庫包括 GitHub 或 AWS CodeCommit。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。查詢此類資料時，通常可以接受緩慢的查詢。將此資料移至效能較低且成本較低的儲存層或類別可降低成本。

計算機視覺（CV）

一個 AI 領域，它使用機器學習來分析和從數字圖像和視頻等視覺格式中提取信息。例如，提 AWS Panorama 供將 CV 添加到現場部署攝像機網絡的設備，Amazon 為 CV SageMaker 提供圖像處理算法。

配置漂移

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

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

AWS Config 規則和補救動作的集合，您可以組合這些動作來自訂合規性和安全性檢查。您可以使用 YAML 範本，將一致性套件部署為 AWS 帳戶 和區域中的單一實體，或跨組織部署。如需詳細資訊，請參閱文件中的AWS Config 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 架構中安全性支柱的一個組成部分。如需詳細資訊，請參閱資料分類。

資料漂移

生產資料與用來訓練 ML 模型的資料之間有意義的變化，或輸入資料隨著時間的推移有意義的變化。資料漂移可降低機器學習模型預測中的整體品質、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

透過集中式管理和控管，提供分散式、分散式資料擁有權的架構架構。

資料最小化

僅收集和處理絕對必要的數據的原則。在中執行資料最小化 AWS 雲端 可降低隱私權風險、成本和分析碳足跡。

資料周長

您 AWS 環境中的一組預防性護欄，可協助確保只有受信任的身分正在存取來自預期網路的受信任資源。若要取得更多資訊，請參閱〈在上建立資料周長〉 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。

數據來源

在整個生命週期中追蹤資料來源和歷史記錄的程序，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 (例如分析) 的資料管理系統。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。

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。在上採用此策略時 AWS，您可以在 AWS

Organizations 結構的不同層加入多個控制項，以協助保護資源。例如，一 defense-in-depth 種方法可能會結合多因素驗證、網路分段和加密。

委派的管理員

在中 AWS Organizations，相容的服務可以註冊成 AWS 員帳戶，以管理組織的帳戶並管理該服務的權限。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。

發展價值流映射

用於識別限制並排定優先順序，對軟體開發生命週期中的速度和品質產生不利影響的程序。DVSM 擴展了最初為精益生產實踐而設計的價值流映射流程。它著重於創造和通過軟件開發過程中移動價值所需的步驟和團隊。

數字雙胞胎

真實世界系統的虛擬表現法，例如建築物、工廠、工業設備或生產線。數位雙胞胎支援預測性維護、遠端監控和生產最佳化。

維度表

在 star 結構描述中，較小的資料表包含事實資料表中定量資料的相關資料屬性。維度表格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、篩選和結果集標籤。

災難

防止工作負載或系統在其主要部署位置達成其業務目標的事件。這些事件可能是自然災害、技術故障或人為行為造成的結果，例如意外設定錯誤或惡意軟體攻擊。

災難復原 (DR)

您使用的策略和程序，將因災難造成的停機時間和資料遺失降到最低。如需詳細資訊，請參閱 AWS Well-Architected 的架構中的雲端中的工作負載的災難復原 AWS：雲端復原。

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。

博士

請參閱災難復原。

漂移檢測

追蹤基線組態的偏差。例如，您可以用 AWS CloudFormation 來偵測系統資源中的漂移，也可以用 AWS Control Tower 來偵測 landing zone 中可能會影響法規遵循治理要求的變更。

DVSM

請參閱開發價值流映射。

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。與雲計算相比，邊緣計算可以減少通信延遲並縮短響應時間。

加密

一種計算過程，將純文本數據（這是人類可讀的）轉換為密文。

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。您可以使用其他或 (IAM) 主體建立端點服務， AWS PrivateLink 並將權限授予其他 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 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

自動化並管理企業的關鍵業務流程（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 安全史詩包括身份和訪問管理，偵探控制，基礎結構安全性，數據保護和事件響應。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

事實表

星型架構中的中央表格。它存儲有關業務運營的定量數據。事實資料表通常包含兩種類型的資料欄：包含計量的資料欄，以及包含維度表格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減少開發生命週期的理念。這是敏捷方法的關鍵部分。

故障隔離邊界

在中 AWS 雲端，可用區域、 AWS 區域控制平面或資料平面等界限，可限制故障的影響，並協助改善工作負載的彈性。如需詳細資訊，請參閱AWS 錯誤隔離邊界。

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，請參閱機器學習模型可解釋性：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

請參閱精細的存取控制。

精細的存取控制 (FGAC)

使用多個條件來允許或拒絕訪問請求。

閃切遷移

一種資料庫移轉方法，透過變更資料擷取使用連續資料複寫，在最短的時間內移轉資料，而不是使用階段化方法。目標是將停機時間降至最低。

G

地理阻塞

請參閱地理限制。

地理限制 (地理封鎖)

在 Amazon 中 CloudFront，防止特定國家/地區的使用者存取內容分發的選項。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 工作流程被認為是遺留的，基於主幹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

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。它們是通過使用 AWS Config，Amazon AWS Security Hub GuardDuty， AWS Trusted Advisor亞馬遜檢查 Amazon Inspector 和自定義 AWS Lambda 檢查來實現的。

H

公頃

查看高可用性。

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，無需干預。HA 系統的設計可自動容錯移轉、持續提供高品質的效能，以及處理不同的負載和故障，並將效能影響降到最低。

歷史學家現代化

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。歷史學家是一種類型的數據庫，用於收集和存儲工廠中的各種來源的數據。

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。

熱數據

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的儲存層或類別，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，修補程式通常是在典型的 DevOps 發行工作流程之外進行。

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

查看基礎結構即程式碼。

身分型政策

附加至一或多個 IAM 主體的政策，用於定義其在 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。

不可變基礎設施

為生產工作負載部署新基礎結構的模型，而不是更新、修補或修改現有基礎結構。不可變的基礎架構本質上比可變基礎架構更加一致、可靠且可預測。如需詳細資訊，請參閱 Well-Architected 的架構中的使用不可變基礎結 AWS 構進行部署最佳作法。

傳入 (輸入) VPC

在 AWS 多帳戶架構中，VPC 可接受、檢查和路由來自應用程式外部的網路連線。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

Klaus Schwab 於 2016 年推出的一個術語，指的是透過連線能力、即時資料、自動化、分析和 AI/ML 的進步實現製造流程的現代化。

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 VPC（相同或不同 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 AWS 的機器學習模型可解釋性。

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

標籤式存取控制 (LBAC)

強制存取控制 (MAC) 的實作，其中每個使用者和資料本身都明確指派一個安全性標籤值。使用者安全性標籤與資料安全性標籤之間的交集決定了使用者可以看到哪些列與欄。

登陸區域

landing zone 是一個架構良好的多帳戶 AWS 環境，具有可擴展性和安全性。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

請參閱以標示為基礎的存取控制。

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

見 7 盧比

小端序系統

首先儲存最低有效位元組的系統。另請參閱 「位元順序」。

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

旨在危及計算機安全性或隱私的軟件。惡意軟件可能會破壞計算機系統，洩漏敏感信息或獲得未經授權的訪問。惡意軟體的例子包括病毒、蠕蟲、勒索軟體、特洛伊木馬程式、間諜軟體和鍵盤記錄程式。

受管理服務

AWS 服務 用於 AWS 操作基礎架構層、作業系統和平台，並且您可以存取端點以儲存和擷取資料。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

用於跟踪，監控，記錄和控制生產過程的軟件系統，可在現場將原材料轉換為成品。

MAP

請參閱 Migration Acceleration Program。

機制

一個完整的過程，您可以在其中創建工具，推動工具的採用，然後檢查結果以進行調整。機制是一個循環，它加強和改善自己，因為它運行。如需詳細資訊，請參閱 AWS Well-Architected 的架構中建置機制。

成員帳戶

屬於 AWS 帳戶 中組織的管理帳戶以外的所有帳戶 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。

郵件佇列遙測傳輸 (MQTT)

以發佈/訂閱模式為基礎的輕量型 machine-to-machine (M2M) 通訊協定，適用於資源受限 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 AWS 無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，請參閱上 AWS的實作微服務。

Migration Acceleration Program (MAP)

提供諮詢支援、訓練和服務的 AWS 計畫，協助組織為移轉至雲端建立穩固的營運基礎，並協助抵消移轉的初始成本。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。移轉工廠團隊通常包括營運、業務分析師和擁有者、移轉工程師、開發人員和 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。移轉中繼資料的範例包括目標子網路、安全性群組和 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 AWS 用應用程式遷移服務將遷移重新託管到 Amazon EC2。

遷移組合評定 (MPA)

這是一種線上工具，可提供驗證要移轉至的商業案例的 AWS 雲端資訊。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)。所有 AWS 顧問和 APN 合作夥伴顧問均可免費使用 MPA 工具 (需要登入)。

遷移準備程度評定 (MRA)

使用 AWS CAF 獲得有關組織雲端準備狀態、識別優勢和弱點，以及建立行動計劃以縮小已識別差距的過程。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

將工作負載移轉至 AWS 雲端. 如需詳細資訊，請參閱本詞彙表中的 7 Rs 項目，並參閱動員您的組織以加速大規模移轉。

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. 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，請參閱評估應用程式的現代化準備程度 AWS 雲端。

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

請參閱移轉組合評估。

MQTT

請參閱佇列遙測傳輸的郵件。

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？」

可變的基礎

一種模型，用於更新和修改生產工作負載的現有基礎結構。為了提高一致性，可靠性和可預測性，AWS Well-Architected 框架建議使用不可變的基礎結構作為最佳實踐。

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

請參閱作業整合。

OLA

請參閱作業層級協定。

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。

OPCA

請參閱開放程序通訊-統一架構。

開放程序通訊-統一架構 (OPC-UA)

用於工業自動化的 machine-to-machine (M2M) 通訊協定。OPC-UA 提供數據加密，身份驗證和授權方案的互操作性標準。

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)。

操作準備程度檢討 (ORR)

問題和相關最佳做法的檢查清單，可協助您瞭解、評估、預防或減少事件和可能的故障範圍。如需詳細資訊，請參閱 AWS Well-Architected 的架構中的作業準備檢閱 (ORR)。

操作技術

可與實體環境搭配使用的硬體和軟體系統，以控制工業作業、設備和基礎設施。在製造業中，整合 OT 和資訊技術 (IT) 系統是工業 4.0 轉型的關鍵焦點。

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 記錄中組織 AWS 帳戶 中所有人的所有事件 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 移轉策略中，這個架構稱為人員加速，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)

在中 CloudFront，限制存取權限以保護 Amazon Simple Storage Service (Amazon S3) 內容的增強選項。OAC 支援所有 S3 儲存貯體 AWS 區域、伺服器端加密 AWS KMS (SSE-KMS)，以及 S3 儲存貯體的動態PUT和DELETE請求。

原始存取身分 (OAI)

在中 CloudFront，用於限制存取以保護 Amazon S3 內容的選項。當您使用 OAI 時， CloudFront 會建立 Amazon S3 可用來進行驗證的主體。經驗證的主體只能透過特定散發存取 S3 儲存 CloudFront 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

請參閱作業整備檢閱。

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的網路連線的 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。

個人識別資訊 (PII)

直接查看或與其他相關數據配對時，可用於合理推斷個人身份的信息。PII 的範例包括姓名、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。

公司

請參閱可編程邏輯控制器

PLM

查看產品生命週期管理。

政策

可以定義權限 (請參閱以身分識別為基礎的策略)、指定存取條件 (請參閱以資源為基礎的策略) 或定義組織中所有帳戶的最大權限的物件 AWS Organizations (請參閱服務控制策略)。

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`，通常位於子`WHERE`句中。

謂詞下推

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

隱私設計

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

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

一種安全控制項，旨在防止部署不符合規範的資源。這些控制項會在資源佈建之前進行掃描。如果資源不符合控制項，則不會佈建該資源。如需詳細資訊，請參閱 AWS Control Tower 文件中的控制項參考指南，並參閱實作安全性控制中的主動控制 AWS。

產品生命週期管理 (PLM)

在產品的整個生命週期中管理資料和流程，從設計、開發、上市到成長與成熟度，再到下降和移除。

生產環境

請參閱環境。

可編程邏輯控制器 (PLC)

在製造業中，一台高度可靠且適應性強的計算機，可監控機器並自動化製造過程。

化名化

以預留位置值取代資料集中的個人識別碼的程序。化名化有助於保護個人隱私。假名化數據仍被認為是個人數據。

發布/訂閱 (發布/訂閱)

一種模式，可在微服務之間實現非同步通訊，以提高延展性和回應能力 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通道。系統可以在不變更發佈服務的情況下新增微服務。

Q

查詢計劃

一系列步驟，如指示，用來存取 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

拉齊矩陣

請參閱負責任，負責，諮詢，通知（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。

拉西矩陣

請參閱負責任，負責，諮詢，通知（RACI）。

RCAC

請參閱列與欄存取控制。

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。

重新建築師

見 7 盧比

復原點目標 (RPO)

自上次資料復原點以來可接受的時間上限。這決定了最後一個恢復點和服務中斷之間可接受的數據丟失。

復原時間目標 (RTO)

服務中斷與恢復服務之間的最大可接受延遲。

重構

見 7 盧比

區域

地理區域中的 AWS 資源集合。每個都 AWS 區域 是隔離和獨立於其他的，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 您的帳戶可以使用的項目。

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。

重新主持

見 7 盧比

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

見 7 盧比

再平台

見 7 盧比

買回

見 7 盧比

彈性

應用程式抵抗或從中斷中復原的能力。在規劃備援時，高可用性和災難復原是常見的考量因素。AWS 雲端如需詳細資訊，請參閱AWS 雲端 復原力。

資源型政策

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

負責者、當責者、事先諮詢者和事後告知者 (RACI) 矩陣

定義移轉活動和雲端作業所涉及之所有各方的角色與責任的矩陣。矩陣名稱衍生自矩陣中定義的責任型別：負責 (R)、負責 (A)、諮詢 (C) 及通知 (I)。支撐（S）類型是可選的。如果您包含支援，則該矩陣稱為 RASCI 矩陣，如果您將其排除，則稱為 R ACI 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

見 7 盧比

退休

見 7 盧比

旋轉

定期更新密碼以使攻擊者更難以存取認證的程序。

資料列與資料行存取控制 (RCAC)

使用已定義存取規則的基本、彈性 SQL 運算式。RCAC 由資料列權限和資料行遮罩所組成。

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

許多身份提供者（IdPs）使用的開放標準。此功能可啟用聯合單一登入 (SSO)，因此使用者可以登入 AWS Management Console 或呼叫 AWS API 作業，而不必為組織中的每個人在 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

請參閱服務控制策略。

秘密

您以加密形式儲存的機密或受限制資訊，例如密碼或使用者認證。 AWS Secrets Manager它由秘密值及其中繼資料組成。密碼值可以是二進位、單一字串或多個字串。如需詳細資訊，請參閱「Secrets Manager 碼中有什麼內容？ 在 Secrets Manager 文檔中。

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。安全性控制有四種主要類型：預防性、偵測、回應式和主動式。

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

預先定義且程式化的動作，其設計用來自動回應或修復安全性事件。這些自動化作業可做為偵探或回應式安全控制項，協助您實作 AWS 安全性最佳實務。自動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

在其目的地的數據加密，通 AWS 服務 過接收它。

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 文件中的服務控制原則。

服務端點

的進入點的 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。

服務等級指示器 (SLI)

對服務效能層面的測量，例如錯誤率、可用性或輸送量。

服務等級目標 (SLO)

代表服務狀況的目標測量結果，由服務層次指示器測量。

共同責任模式

描述您在雲端安全性和合規方面共享的責任的模型。 AWS AWS 負責雲端的安全性，而您則負責雲端的安全性。如需詳細資訊，請參閱共同責任模式。

暹

請參閱安全性資訊和事件管理系統。

單點故障 (SPF)

應用程式的單一重要元件發生故障，可能會中斷系統。

SLA

請參閱服務等級協議。

SLI

請參閱服務層級指示器。

SLO

請參閱服務等級目標。

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，請參閱中的應用程式現代化的階段化方法。 AWS 雲端

痙攣

請參閱單一故障點。

星型綱要

使用一個大型事實資料表來儲存交易或測量資料，並使用一或多個較小的維度表格來儲存資料屬性的資料庫組織結構。這種結構是專為在數據倉庫中使用或用於商業智能目的。

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。

監督控制與資料擷取 (SCADA)

在製造業中，使用硬體與軟體來監控實體資產與生產作業的系統。

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

以模擬使用者互動以偵測潛在問題或監控效能的方式測試系統。您可以使用 Amazon CloudWatch Synthetics 來創建這些測試。

T

標籤

作為組織 AWS 資源的中繼資料的索引鍵值配對。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 文件中的傳輸閘道是什麼。

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

授與權限給您指定的服務，以代表您在組織內 AWS Organizations 及其帳戶中執行工作。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 AWS Organizations 文件中的AWS Organizations 與其他 AWS 服務搭配使用。

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 DevOps 團隊，你可以餵兩個比薩餅。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。

無差別的任務

也稱為繁重工作，是創建和操作應用程序所必需的工作，但不能為最終用戶提供直接價值或提供競爭優勢。無差異化作業的範例包括採購、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

會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。

溫暖的數據

不常存取的資料。查詢此類資料時，通常可以接受中度緩慢的查詢。

視窗功能

一種 SQL 函數，可對以某種方式與當前記錄相關的一組行執行計算。視窗函數對於處理工作非常有用，例如計算移動平均值或根據目前列的相對位置存取列的值。

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。

蠕蟲

看到寫一次，多讀。

WQF

請參閱 AWS 工作負載資格架構。

寫一次，多讀（WORM）

一種儲存模型，可單次寫入資料並防止資料遭到刪除或修改。授權用戶可以根據需要多次讀取數據，但無法更改數據。這種數據存儲基礎設施被認為是不可變的。

Z

零日漏洞

一種利用零時差漏洞的攻擊，通常是惡意軟件。

零時差漏洞

生產系統中未緩解的瑕疵或弱點。威脅參與者可以利用這種類型的漏洞攻擊系統。由於攻擊，開發人員經常意識到該漏洞。

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。