上的代理式 AI 基礎 AWS

AWS 方案指引

AWS 方案指引: 上的代理式 AI 基礎 AWS

Table of Contents

上的代理式 AI 基礎 AWS

Aaron Sempf，Amazon Web Services

2025 年 7 月 (文件歷史記錄)

在越來越智慧、分散式和自動化系統的世界中，代理程式的概念—可感知其環境的實體、其狀態的原因，以及有意採取行動的實體—已經成為基礎。代理程式不只是執行指示的程式；它們是目標導向、內容感知的實體，可代表使用者、系統或組織做出決策。它們的出現反映了您如何建置和思考軟體的轉變：從程序邏輯和反應式自動化轉移到以自主性和目的操作的系統。

在 AI、分散式系統和軟體工程的交集，是稱為代理式 AI 的強大範例。這種新一代的智慧型系統由軟體代理程式組成，能夠適應行為、複雜的協調和委派決策。

本指南介紹定義現代軟體代理器的原則，並概述他們朝向代理式 AI 的演變。為了解釋此轉移，本指南提供概念背景，然後追蹤軟體代理程式對客服人員 AI 的演變：

- 軟體代理程式簡介會定義軟體代理程式，將其與傳統軟體元件進行比較，並介紹基本特性，透過從已建立的架構中提取來區分代理程式行為與傳統自動化。
- 軟體代理程式的目的是檢查軟體代理程式存在的原因、他們履行的角色、他們解決的問題，以及他們如何啟用智慧型委派、減少認知負載，以及支援動態環境中的適應性行為。
- 軟體代理程式的演進追蹤塑造軟體代理程式的智慧和技術里程碑，從自主和並行的早期概念到多代理程式系統和正式代理程式架構的出現，導致與生成式 AI 的融合。
- 代理式 AI 的軟體代理程式將代理式 AI 作為幾十年來的最新進展，將分散式代理程式模型與基礎模型、無伺服器運算和協同運作通訊協定結合在一起。本節說明此收斂如何啟用新一代智慧型、工具使用型代理程式，以自主性、非同步性和真正的大規模代理操作。

目標對象

本指南專為希望先了解軟體代理程式到代理式 AI 的歷史、主要概念和演變，再採用此技術做為現代雲端解決方案的架構師、開發人員和技術領導者所設計 AWS。

目標

採用代理架構可協助組織：

- 加速實現價值的時間：自動化和擴展知識工作，並減少手動工作量和延遲。

- 改善客戶參與度：跨網域交付智慧型助理。
- 降低營運成本：自動化先前需要人工輸入或監督的決策流程。
- 推動創新和差異化：建置智慧型產品，以即時調整、學習和競爭。
- 將傳統工作流程現代化：將指令碼和整體重新建構為模組化推理代理程式。

關於此內容系列

本指南是代理式 AI 相關系列的一部分 AWS。如需詳細資訊並檢視此系列中的其他指南，請參閱 AWS 規範指引網站上的客服人員 AI。

軟體代理程式簡介

軟體代理程式的概念已從 1960 年代自主實體的基礎大幅發展到 1990 年代早期的正式探索。隨著數位系統變得越來越複雜，從確定性指令碼到適應性、智慧型應用程式，軟體代理程式已成為在運算系統中實現自動化、內容感知和目標驅動行為的重要建置區塊。在雲端原生和 AI 增強型架構的情況下，特別是隨著生成式 AI、大型語言模型 (LLMs) 和 Amazon Bedrock 等平台的出現，軟體代理程式正在透過功能和擴展的新面向重新定義。

此簡介取自精實工作軟體代理程式：Hyacinth S. Nwana (Nwana 1996) 的概觀。它定義了軟體代理程式、討論其概念根，並將討論擴展到現代架構，以定義現代軟體代理程式的三個總體原則：自主性、非同步性和代理。這些原則會將軟體代理程式與其他類型的服務或應用程式區分開來，並讓這些代理程式能夠在分散式的即時環境中使用用途、彈性和智慧運作。

本節內容

從自主到分散式智慧

在軟體代理程式進入主流一詞之前，早期運算研究探索了自動化數位實體的概念，這些實體是能夠獨立運作、對輸入做出反應，並根據內部規則或目標做出決策的系統。這些早期想法奠定了成為客服人員範例的概念基礎。（如需歷史時間軸，請參閱本指南稍後的軟體代理程式演變一節。）

自主性的早期概念

數十年來，獨立於人力運算子的機器或程式概念都吸引了系統設計人員。網路網路、人工智慧和控制系統的早期工作會檢查軟體如何展現自我調節行為、動態回應變更，以及在沒有持續人工監督的情況下運作。

這些想法將自主性引入智慧系統的核心屬性，並為可決定和行動的軟體的出現做好準備，而不是僅做出反應或執行。

演員模型和非同步執行

在 1970 年代，A Universal Modular ACTOR Formalism for Artificial Intelligence (Hewitt et al. 1973) 論文中介紹的演員模型提供了正式架構，用於考慮分散式、訊息驅動的運算。在此模型中，演員是獨立的實體，其僅透過傳遞非同步訊息進行通訊，並啟用可擴展、並行和容錯的系統。

演員模型強調了三個關鍵屬性，這些屬性會持續影響現代客服人員設計：

- 狀態和行為的隔離
- 實體之間的非同步互動
- 動態建立和委派任務

這些屬性符合分散式系統的需求，並預先建構了雲端原生環境中軟體代理程式的操作特性。

分散式智慧和多代理程式系統

隨著運算系統在 1960 年代之後變得更加相互關聯，研究人員探索了分散式人工智慧 (DAI)。此欄位著重於多個自主實體如何跨系統協作或競爭運作。DAI 促成多代理程式系統的開發，其中每個代理程式都有本機目標、感知和推理，但也在更廣泛的互連環境中運作。

這種分散式智慧的願景，其中決策是分散的，而緊急行為是由客服人員互動產生，對於如何構想和建置現代客服人員型系統仍是核心。

Nwana 的類型和軟體代理程式的崛起

1990 年代中期軟體代理程式概念的正式化，標誌了智慧型系統發展的轉折點。此正規化最有影響力的貢獻之一是 Hyacinth S. Nwana 的精算論文：軟體代理程式：概觀 (Nwana 1996)，它提供了最先全方位的架構之一，用於分類和了解各個維度的軟體代理程式。

在本白皮書中，Nwana 會調查軟體代理程式研究的狀態，並識別如何定義和實作代理程式的分歧。該白皮書重點介紹了對常見概念架構的需求，並提出了根據客服人員關鍵功能分類客服人員的類型。它會審查來自學術界和產業的代表性代理程式系統、區分代理程式與傳統程式和物件，並概述代理程式型運算的挑戰和機會。

Nwana 強調軟體代理程式不是單體概念，而是沿著複雜和功能的範圍存在。類型旨在釐清此環境，並引導未來的設計和研究。

Nwana 將軟體代理程式定義為在特定環境中持續且自主運作的軟體實體，通常由其他代理程式和程序所駐留。此定義強調兩個關鍵特性：

- 持續性：代理程式會隨著時間持續運作，而不需要持續的人工介入。
- 自主性：代理程式能夠根據其對環境的感知，獨立做出決策並對其採取行動。

此定義結合 Nwana 的代理程式類型，強調委派授權（透過自主權）和主動性作為代理程式的基礎特性。它透過強調代理程式代表另一個實體獨立採取行動的能力，並啟動行為以追求目標，而不是僅回應直接命令，來區分代理程式和子例行程序或服務。

Nwana 的代理程式類型

為了進一步區分各種類型的客服人員，Nwana 推出了基於六個關鍵屬性的分類系統：

- 自主性：代理程式在沒有人類或其他人直接介入的情況下運作。
- 社交能力：客服人員使用通訊機制與其他客服人員或人類互動。
- 回應：客服人員會感知其環境並及時回應。
- 主動性：客服人員採取主動，展現目標導向的行為。
- 適應性和學習：代理程式透過經驗隨著時間改善其效能。
- 行動性：代理程式可以在不同的系統環境或網路之間移動。

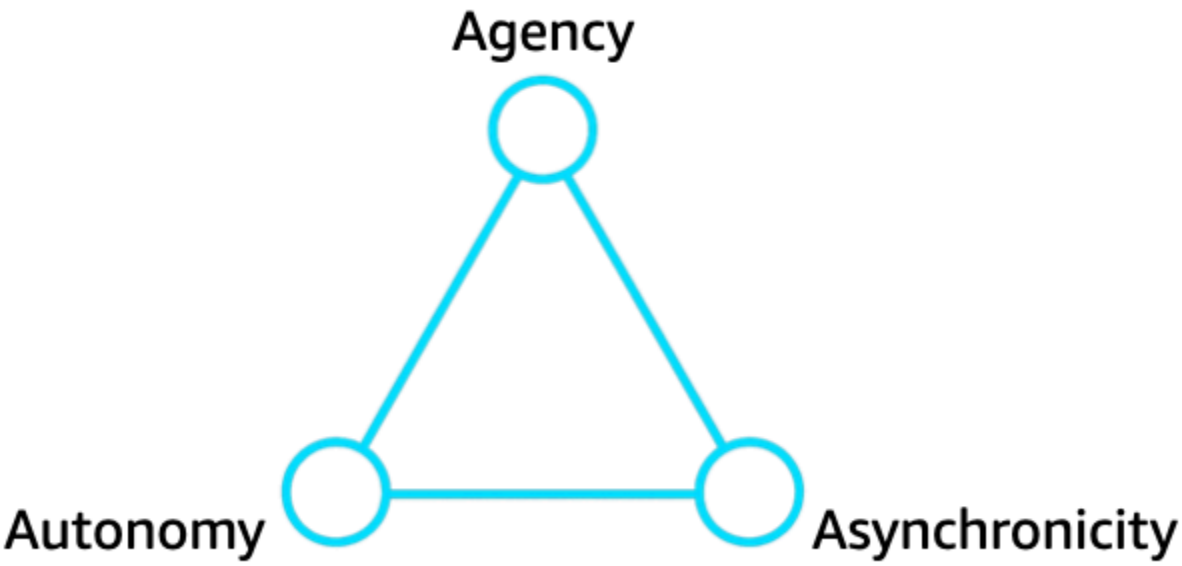
從類型學到現代代理程式原則

Nwana 的工作同時做為分類法和基礎鏡頭，運算社群可以藉此評估軟體中不斷發展的機構形式。他強調自主性、主動性和代表使用者或系統的概念，為我們現在認為客服人員的行為奠定了基礎。

雖然技術和環境已變更，尤其是隨著生成式 AI、無伺服器基礎設施和多代理程式協同運作架構的崛起，Nwana 工作的基礎洞見仍然相關。它們在早期代理程式理論和軟體代理程式的三個現代支柱之間提供關鍵橋樑。

現代軟體代理程式的三個支柱

在現今採用 AI 的平台、微服務架構和事件驅動型系統中，軟體代理程式可以透過三個相互依存的原則來定義，這些原則與標準服務或自動化指令碼區別：自主性、非同步性和代理程式。在下圖和後續圖表中，三角形代表現代軟體代理程式的這三個支柱。



自主性

現代代理程式可獨立運作。他們根據內部狀態和環境內容做出決策，而不需要人為提示。這可讓他們即時回應資料、管理自己的生命週期，並根據目標和情境輸入調整其行為。

自主性是客服人員行為的基礎。它可讓代理程式在沒有持續監督或硬式編碼控制流程的情況下運作。

非同步性

代理程式基本上是非同步的。這表示它們會在事件發生時回應事件、訊號和刺激，而不需要依賴封鎖呼叫或線性工作流程。此特性可實現可擴展、非封鎖的通訊、分散式環境中的回應能力，以及元件之間的鬆散耦合。

透過非同步性，客服人員可以參與即時系統，並與其他服務或客服人員流暢有效地協調。

代理程式作為定義原則

自主性和非同步性是必要的，但這些功能本身不足以讓系統成為真正的軟體代理程式。關鍵差異化因素是機構，其引入了：

- 目標導向行為：客服人員追求目標並評估目標進度。
- 決策：客服人員會根據規則、模型或學習到的政策來評估選項並選擇動作。

- 委派意圖：客服人員代表人員、系統或組織行事，並具有內嵌的目的感。
- 內容推理：客服人員會整合其環境的記憶體或模型，以智慧方式引導行為。

自動和非同步的系統可能仍然是被動服務。讓它成為軟體代理程式的功能在於能夠以意圖和目的行事，成為代理程式。

具有 目的的代理程式

自主性、非同步性和代理性的原則可讓系統在分散式環境中以智慧、適應性和獨立方式運作。這些原則根植於數十年的概念和架構演變，現在是許多目前建置中最進階 AI 系統的基礎。

在這個生成式 AI、目標導向協同運作和多代理程式協同合作的新時代，了解軟體代理程式真正成為代理程式的原因至關重要。將機構視為定義特性，有助於我們超越自動化，並有目的地進入自動化智慧領域。

軟體代理程式的目的

隨著現代系統變得越來越複雜、分散式和智慧，軟體代理程式的角色在從自動操作到使用者輔助技術等領域中取得了顯著的影響力。但軟體代理程式的基礎用途是什麼？ 我們為什麼設計超出指令碼、服務或靜態模型的系統，並將任務委派給能夠感知、推理和行動的實體？

本節探索軟體代理程式的基本目的：在動態環境中啟用任務的智慧委派，專注於自主性、適應性和有目的的動作。它介紹了軟體代理程式的概念基礎、追蹤其認知結構，並概述了他們唯一準備好解決的實際問題。

本節內容

從演員模型到客服人員認知

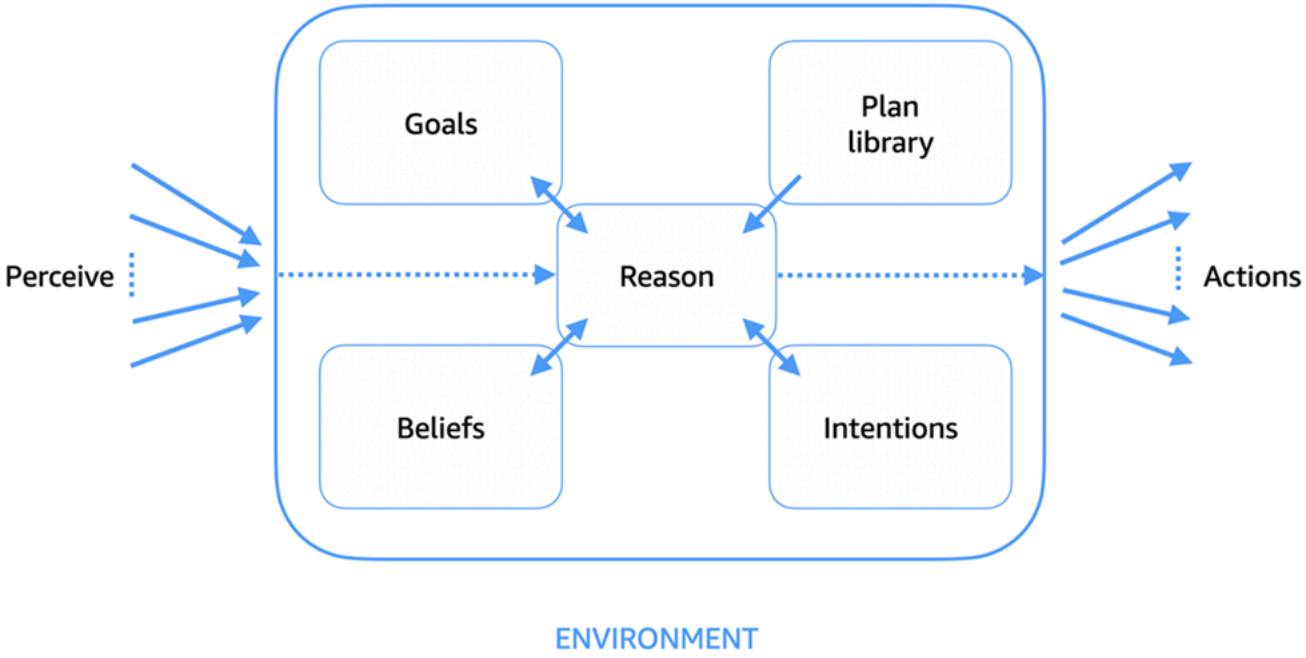
軟體代理程式的目的和結構是以早期運算模型所產生的想法為基礎，特別是 Carl Hewitt 在 1970 年代推出的演員模型 (Hewitt et al. 1973)。

演員模型會將運算視為稱為演員的獨立並行執行實體集合。每個演員都會封裝自己的狀態、僅透過非同步訊息傳遞互動，並且可以建立新的演員和委派任務。

此模型為分散式推理、反應和隔離提供了概念基礎，這些都是現代軟體代理器行為架構的基礎。

代理程式函數：感知、原因、動作

每個軟體代理程式的核心都是認知週期，通常描述為感知、原因、行為迴圈。下表說明此程序。它定義了代理程式如何在動態環境中自動運作。



- 感知：客服人員從環境收集資訊（例如事件、感應器輸入或 API 訊號），並更新其內部狀態或想法。
- 原因：客服人員使用計劃程式庫或邏輯系統來分析目前的想法、目標和情境知識。此程序可能涉及目標優先順序、衝突解決或意圖選擇。
- 動作：客服人員會選取並執行動作，讓他們更接近其委派目標。

此架構支援代理程式在剛性程式設計之外運作的能力，並啟用靈活、內容敏感和目標導向的行為。它形成了指導軟體代理程式廣泛用途的心理架構。

自主協同合作和意圖

軟體代理程式的目的是為現代運算帶來自主性、內容感知和智慧委派。由於客服人員是以演員模型的原則為基礎，並體現在感知、原因、動作週期中，因此他們不僅啟用被動、主動和有目的的系統。

代理程式可讓軟體在複雜的環境中決定、調整和採取行動。它們代表使用者、解譯目標，並以機器速度實作任務。隨著我們更深入客服人員 AI 的時代，軟體代理程式正在成為人類意圖和智慧型數位動作之間的操作界面。

委派意圖

與傳統軟體元件不同，軟體代理程式會代表 其他項目：使用者、另一個系統或高階服務。它們帶有委派意圖，這表示他們：

- 啟動後獨立操作。
- 做出符合委派者目標的選擇。
- 導覽執行中的不確定性和權衡。

代理程式填補了指示和結果之間的差距，這可讓使用者以更高層級的抽象表達意圖，而不需要明確的指示。

在動態、無法預測的環境中操作

軟體代理程式是專為條件不斷變化、資料即時抵達以及控制和內容分佈的環境所設計。

與需要確切輸入或同步執行的靜態程式不同，代理程式會適應其環境並動態回應。這是雲端原生基礎設施、邊緣運算、物聯網 (IoT) 網路和即時決策系統中的重要功能。

減少人類認知負載

軟體代理程式的主要目的是減少人類的認知和操作負擔。代理程式可以：

- 持續監控系統和工作流程。
- 偵測和回應預先定義的或緊急條件。
- 自動化重複、大量決策。
- 以最小延遲回應環境變化。

當決策從使用者轉移到客服人員時，系統會變得更有回應能力、更具彈性和以人為本，並且可以即時適應新資訊或中斷。這可實現更快的反應周轉，並在高複雜性或高規模環境中實現更高的操作連續性。結果是人為焦點的轉移，從微層級決策到策略監督和創意問題解決。

啟用分散式智慧

軟體代理程式個別或共同運作的能力，可讓多代理程式系統 (MAS) 設計跨環境或組織進行協調。這些系統可以智慧地分配任務，並交涉、合作或競爭複合目標。

例如，在全球供應鏈系統中，個別客服人員會管理工廠、運送、倉儲和最後一哩運送。每個代理程式都以本機自主性運作：工廠代理程式根據資源限制最佳化生產、倉儲代理程式即時調整庫存流程，以及交付代理程式根據流量和客戶可用性重新路由貨物。

這些代理程式會動態通訊和協調，並適應中斷，例如連接埠延遲或卡車故障，無需集中控制。系統的整體智慧來自這些互動，並啟用彈性、最佳化且超出單一元件功能的物流。

在此模型中，客服人員在更廣泛的智慧結構中充當節點。它們形成的緊急系統能夠解決任何單一元件無法單獨處理的問題。

有目的的行為，而不只是反應

僅自動化在複雜系統中並不足夠。軟體代理程式的定義目的是有目的地採取行動，並評估目標、權衡內容，以及做出明智的選擇。這表示軟體代理程式追求目標，而不是僅回應觸發。他們可以根據經驗或意見回饋來修訂想法和意圖。在這種情況下，可信度是指代理程式根據其感知（輸入和感應器）的環境內部表示法（例如「套件 X 位於倉儲 A」）。意圖是指客服人員選擇實現目標的計劃（例如，「使用交付路由 B 並通知收件人」）。客服人員也可以視需要呈報、延遲或調整動作。

這種目的不僅讓軟體代理程式成為被動執行器，也讓智慧系統中的自動協作者成為可能。

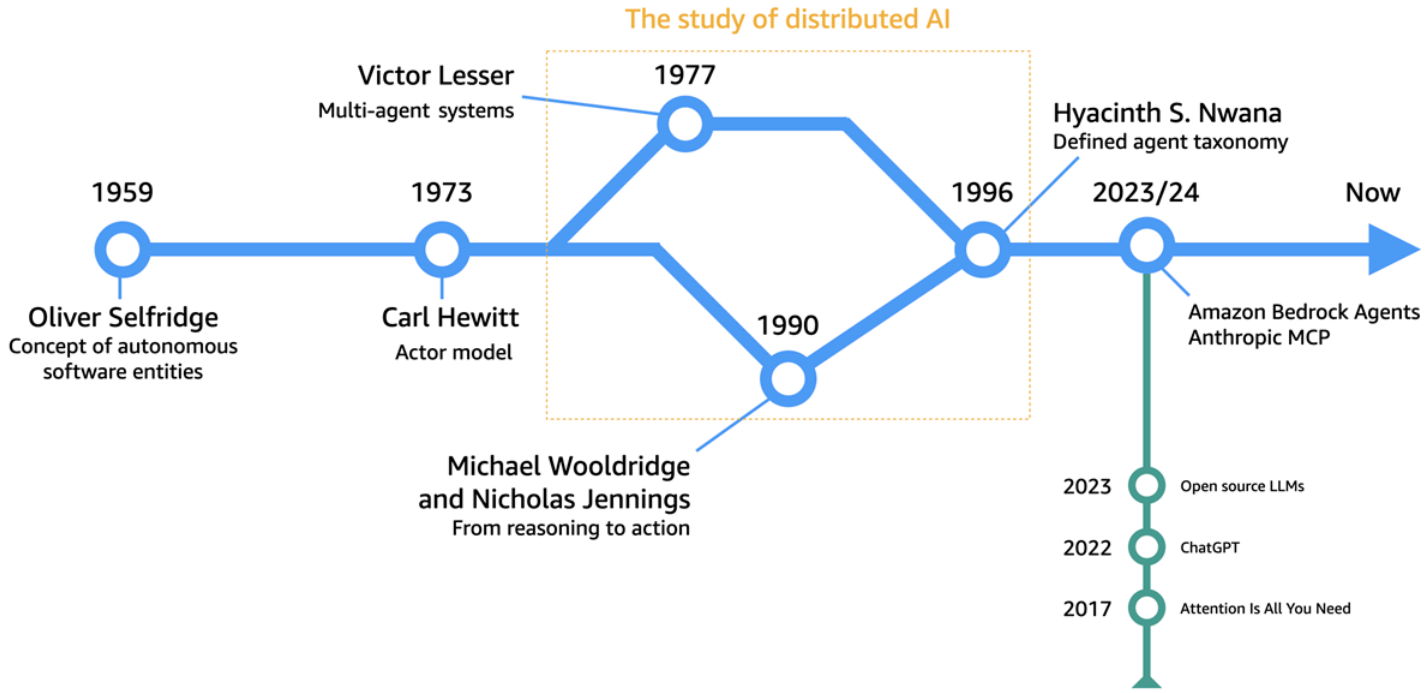
軟體代理程式的演變

從簡單的自動化系統到智慧、自主和目標導向的軟體代理程式的旅程，反映了電腦科學、人工智慧和分散式系統數十年的演進。

此演進之後是機器學習的崛起，這將範例從手動製作的規則轉移到統計模式辨識。這些系統可以從資料中學習，並實現感知、分類和決策的進展。

大型語言模型 LLMs) 代表規模、架構和非監督式學習的融合。LLMs可以在很少或沒有任務特定訓練的情況下推理、產生和調整任務。透過將 LLMs 與可擴展的雲端原生基礎設施和組合式架構結合，我們現在實現代理式 AI 的完整願景：智慧型軟體代理程式，可在企業規模下以自主性、內容感知和適應性運作。

本節探索從基礎理論到現代實務的軟體代理程式歷史記錄，如下圖所示。它強調分散式人工智慧 (DAI) 和轉換器型生成式 AI 的收斂，並識別形成代理式 AI 出現的關鍵里程碑。



本節內容

軟體代理程式的基礎

1959 年 – Oliver Selfridge：軟體自治的起源

軟體代理程式的根目錄可追溯至介紹自治軟體實體（示範）概念的 Oliver Selfridge，這些程式能夠感知其環境並獨立運作 (Selfridge 1959)。他在機器感知和學習方面的早期工作為未來將客服人員視為獨立、智慧型系統的概念奠定了哲學基礎。

1973 – Carl Hewitt：演員模型

關鍵進展隨附於 Carl Hewitt 的演員模型 (Hewitt et al. 1973)，這是一個正式的運算模型，將代理程式描述為獨立的並行實體。在此模型中，客服人員可以封裝自己的狀態和行為，使用非同步訊息傳遞進行通訊，並動態建立其他演員並將任務委派給他們。

演員模型為分散式代理程式型系統提供了理論基礎和架構範例。此模型預先建構現代並行實作，例如 Erlang 程式設計語言和 Akka 架構。

使 欄位到期：從推理到動作

1977 年 – Victor Lesser：多代理程式系統

在 1970 年代後期，分散式人工智慧 (DAI) 出現。它受到 Victor Lesser 的擁護，Victor Lesser 因開拓多代理程式系統 (MAS) 而廣為人知。他的工作著重於獨立軟體實體如何合作、協調和交涉（請參閱資源一節）。此開發導致系統能夠共同解決複雜的問題，這是建置分散式智慧的重要躍進。

1990 年代 – Michael Honeydridge 和 Nicholas Jennings：代理程式頻譜

到了 1990 年代，分散式智慧領域隨著 Michael Roomselridge 和 Nicholas Jennings 等研究人員的貢獻而成熟。這些學者沿著光譜分類代理程式，從被動到深思熟慮，從非認知系統到目標驅動推理代理程式 (Wooldridge 和 Jennings 1995)。他們的工作強調客服人員不再是抽象的想法，而是應用在從機器人到企業軟體的各種實際領域。

這些研究人員也引入了焦點的轉移：從集中式推理到分散式動作。代理程式不再只是思考者，而是在具有自主性和目的的即時環境中操作的執行者。

1996 年 – Hyacinth S. Nwana：正式化客服人員概念

1996 年，Hyacinth S. Nwana 發佈了具影響力的文件軟體代理程式：概觀，提供迄今為止最全面的代理程式分類。他的類型包括自主性、社交能力、反應、主動性、學習和行動性等屬性，並區分軟體代理程式和傳統軟體建構。

Nwana 也提供現已廣泛接受的定義，並加以解釋：軟體代理程式是一種以軟體為基礎的電腦程式，其作用對象為代理關係中的使用者或其他程式，其衍生自委派概念。

此正規化有助於將軟體代理程式從理論建構轉換到真實世界應用程式。它在電信、工作流程自動化和智慧型助理等領域產生了一代以代理程式為基礎的系統。

Nwana 的工作位於早期分散式 AI 研究的收斂點，以及現代客服人員的操作架構。這是客服人員認知理論與其在現今系統中實際部署之間的重要橋樑。

平行時間軸：大型語言模型的崛起

當客服人員架構不斷發展時，自然語言處理和機器學習正在發生平行和收斂的變革：

- 2017 年 – 轉換器：紙質注意力是您需要的 (Vaswani et al. 2017) 推出了轉換器架構，大幅改善了機器處理和產生語言的方式。
- 2022 年 – ChatGPT：OpenAI 向稱為 ChatGPT 的 GPT-3.5 發佈了聊天型界面，該界面啟用了與一般用途 AI 系統的自然互動式對話。
- 2023 年 – 開放原始碼 LLMs：Llama、Falcon 和 Mistral 的版本使得強大的模型可廣泛存取，並加速開放原始碼和企業環境中代理程式架構的開發。

這些創新將語言模型轉換為推理引擎，能夠剖析內容、規劃動作和鏈結回應，而 LLMs成為智慧型軟體代理程式的關鍵推動因素。

時間軸收斂：代理式 AI 的出現

2023-2024 – 企業級代理程式平台

分散式軟體代理程式架構和以轉換器為基礎的 LLMs 的收斂，在代理程式 AI 的崛起中達到最高。

- Amazon Bedrock Agents 推出了一種全受管方式，透過使用 Amazon Bedrock 的基礎模型來建置目標驅動、工具使用的軟體代理程式。
- Anthropic 的 模型內容通訊協定 (MCP) 定義了大型語言模型存取和互動外部工具、環境和記憶體的方法。這是情境式、持久性和自主行為的關鍵。

這兩個里程碑代表機構和智慧的合成。代理程式不再受限於靜態工作流程或剛性自動化。他們現在可以跨多個步驟推理、與工具和 APIs協調、維持情境狀態，以及隨時間學習和調整。

2025 年 1 月至 6 月 – 擴展企業功能

在 2025 年上半年，代理式 AI 環境隨著新的企業功能大幅擴展。2025 年 2 月，Anthropic 發佈了 Claude 3.7 Sonnet，這是市場上第一個混合推理模型，MCP 規格獲得廣泛採用。

AI 編碼助理，例如 Amazon Q Developer、Cursor 和 WindSurf 整合 MCP，以標準化程式碼產生、儲存庫分析和開發工作流程。MCP 2025 年 3 月版本推出重要的企業就緒功能，包括 OAuth 2.1 安全整合、擴展了各種資料存取的資源類型，以及透過可串流 HTTP 增強的連線選項。在此基礎上， AWS 在 2025 年 5 月宣布加入 MCP 指導委員會，並為新的agent-to-agent之間的溝通功能做出貢獻。這進一步強化了通訊協定做為代理式 AI 互通性業界標準的地位。

在 2025 年 5 月，透過開啟採購 Strands Agents 架構， AWS 增強了建置客服人員 AI 工作流程的客戶選項。此提供者獨立且與模型無關的架構可讓開發人員跨平台使用基礎模型，同時維持深度 AWS 服務整合。如AWS 開放原始碼部落格所述，Strands Agents 遵循模型優先的設計理念，將基礎模型置於代理程式智慧的核心。這可讓客戶更輕鬆地針對特定使用案例建置和部署複雜的 AI 代理器。

Emergence – 代理式 AI

軟體代理程式的演變，從早期自主性概念到現代、啟用 LLM 的協同運作，一直很長且分層。從 Oliver Selfridge 感知程式的願景開始，它已經發展到一個強大的生態系統，包括智慧、內容感知、目標驅動的軟體代理程式，可以協作、調整和推理。

分散式人工智慧 (DAI) 和轉換器型生成式 AI 的收斂，標誌了新時代的開始，其中軟體代理程式不再只是工具，而是智慧系統中的自動執行者。

代理式 AI 代表軟體系統的下一個發展。它提供一種智慧代理程式類別，可自主、非同步和代理，並且可以採用委派意圖並在動態、分散式環境中有目的地操作。代理式 AI 統一以下項目：

- 多代理程式系統和演員模型的架構系列
- 感知、原因、行為的認知模型
- LLMs 和轉換器的生成能力
- 雲端原生和無伺服器運算的操作彈性

軟體代理程式到代理程式 AI

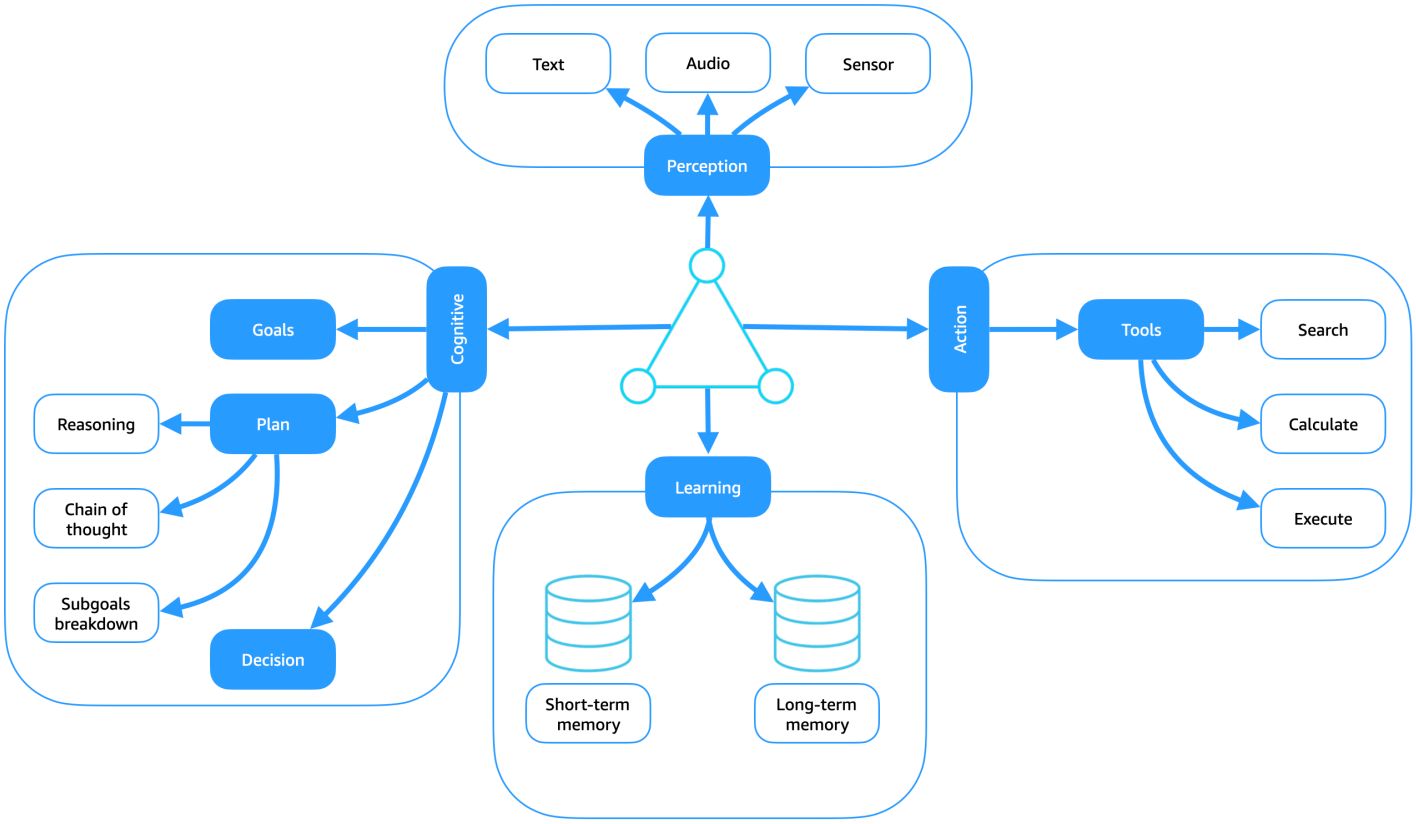
軟體代理程式是自主的數位實體，旨在感知其環境、考量其目標的原因，並採取相應行動。與遵循固定邏輯的傳統軟體程式不同，客服人員會根據內容輸入和決策架構來調整其行為。這使得它們非常適合動態的分散式環境，例如雲端原生系統、機器人、智慧型自動化，以及現在的生成式 AI 協同運作。

本節介紹軟體代理程式的核心建置區塊，並根據感知、原因和行為模型，說明這些元件如何在傳統架構中互動。它討論了生成式 AI，特別是大型語言模型 (LLMs) 如何改變軟體代理程式推理和規劃的方式。這表示從規則型系統到資料驅動型的客服人員 AI 情報的基本轉移。

本節內容

軟體代理程式的核心建置區塊

下圖顯示大多數智慧型代理程式中發現的主要功能模組。每個元件都有助於代理程式在複雜環境中自動操作的能力。



在感知、原因、動作迴圈的情況下，代理程式的推理功能會分佈在其認知和學習模組中。透過整合記憶體和學習，客服人員開發以過去經驗為基礎的適應性推理。當代理程式在其環境中運作時，會建立緊急意見回饋迴圈：每個動作都會影響未來的感知，而產生的體驗會透過學習模組整合到記憶體和內部模型中。這種持續的感知、推理和動作迴圈，可讓代理程式隨著時間改善，並完成完整的感知、原因、動作週期。

感知模組

感知模組可讓代理程式透過文字、音訊和感應器等各種輸入方式與其環境互動。這些輸入會形成所有推理和動作所依據的原始資料。文字輸入可能包含自然語言提示、結構化命令或文件。音訊輸入包含語音指示或環境聲音。感應器輸入包括實體資料，例如視覺摘要、動作訊號或 GPS 座標。感知的核心功能是從此原始資料中擷取有意義的特徵和表示。這可讓代理程式建構對其目前內容的準確且可行的理解。此程序可能涉及特徵擷取、物件或事件辨識，以及語意解釋，並構成感知、原因、行為迴圈中的關鍵第一步。有效的感知可確保下游推理和決策是以相關up-to-date情境感知為基礎。

認知模組

認知模組做為軟體代理程式的刻意核心。它負責解釋感知、形成意圖，並透過目標驅動的規劃和決策來引導有目的的行為。此模組會將輸入轉換為結構化推理程序，讓代理程式能夠刻意操作，而非被動操作。這些程序是透過三個關鍵子模組進行管理：目標、規劃和決策。

目標子模組

目標子模組定義代理程式的意圖和方向。目標可以是明確（例如，「導覽至位置」或「提交報告」）或隱含（例如，「最大化使用者參與度」或「最小化延遲」）。它們是客服人員推理週期的核心，並為其規劃和決策提供目標狀態。

代理程式會持續評估其目標的進度，並根據新的感知或學習，重新排定目標的優先順序或重新產生目標。此目標意識可讓代理程式在動態環境中適應。

規劃子模組

規劃子模組會建構策略，以達成代理程式目前的目標。它會產生動作序列、以階層方式分解任務，並從預先定義或動態產生的計劃中選取。

若要在非確定性或不斷變化的環境中有效操作，規劃不是靜態的。現代客服人員可以產生chain-of-thought序列、將子目標引入中繼步驟，並在條件轉移時即時修訂計畫。

此子模組與記憶體和學習緊密連接，並允許代理程式根據過去的結果隨著時間改進其規劃。

決策子模組

決策子模組會評估可用的計劃和動作，以選取最適合的下一個步驟。它整合了感知、目前計劃、客服人員的目標和環境內容的輸入。

決策考量：

- 衝突目標之間的權衡
- 可信度閾值（例如，感知中的不確定性）
- 動作的後果
- 客服人員的學習經驗

根據架構，客服人員可能會依賴符號推理、啟發式、強化學習或語言模型 (LLMs) 做出明智的決策。此程序會保持代理程式的行為內容感知、目標一致和適應性。

動作模組

動作模組負責執行客服人員選取的決策，並與外部世界或內部系統互動，以產生有意義的效果。它代表感知、原因、動作迴圈的動作階段，其中意圖轉換為行為。

當認知模組選取動作時，動作模組會透過特殊的子模組協調執行，其中每個子模組都與代理程式的整合環境一致：

- 實體致動：對於內嵌在機器人系統或 IoT 裝置的代理程式，此子模組會將決策轉換為實際的實體移動或硬體層級指示。

  範例：轉向機器人、觸發閥、開啟感應器。
- 整合互動：此子模組會處理非實體但外部可見的動作，例如與軟體系統、平台或 APIs 互動。

  範例：將命令傳送至雲端服務、更新資料庫、呼叫 API 提交報告。
- 工具調用：客服人員通常會使用專門的工具來完成子任務，以擴展其功能，如下所示：
  - 搜尋：查詢結構化或非結構化的知識來源
  - 摘要：將大型文字輸入壓縮為高階概觀
  - 計算：執行邏輯、數值或符號運算

  工具調用可透過模組化、可呼叫的技能來實現複雜的行為合成。

學習模組

學習模組可讓客服人員根據經驗調整、一般化和改善一段時間。它使用感知和動作的意見回饋，持續精簡客服人員的內部模型、策略和決策政策，以支援推理程序。

此模組會與短期和長期記憶體協調運作：

- 短期記憶體：儲存暫時性內容，例如對話狀態、目前任務資訊和最近的觀察。它有助於客服人員在互動和任務中維持持續性。
- 長期記憶體：編碼過去經驗的持久性知識，包括先前遇到的目標、動作結果和環境狀態。長期記憶體可讓代理程式辨識模式、重複使用策略，並避免重複錯誤。

學習模式

學習模組支援一系列範例，例如監督式、非監督式和強化式學習，其支援不同的環境和客服人員角色：

- 監督式學習：根據標記的範例更新內部模型，通常是來自人工意見回饋或訓練資料集。

  範例：學習根據先前的對話分類使用者意圖。

- 非監督式學習：在沒有明確標籤的情況下識別資料中的隱藏模式或結構。

  範例：叢集環境訊號以偵測異常。

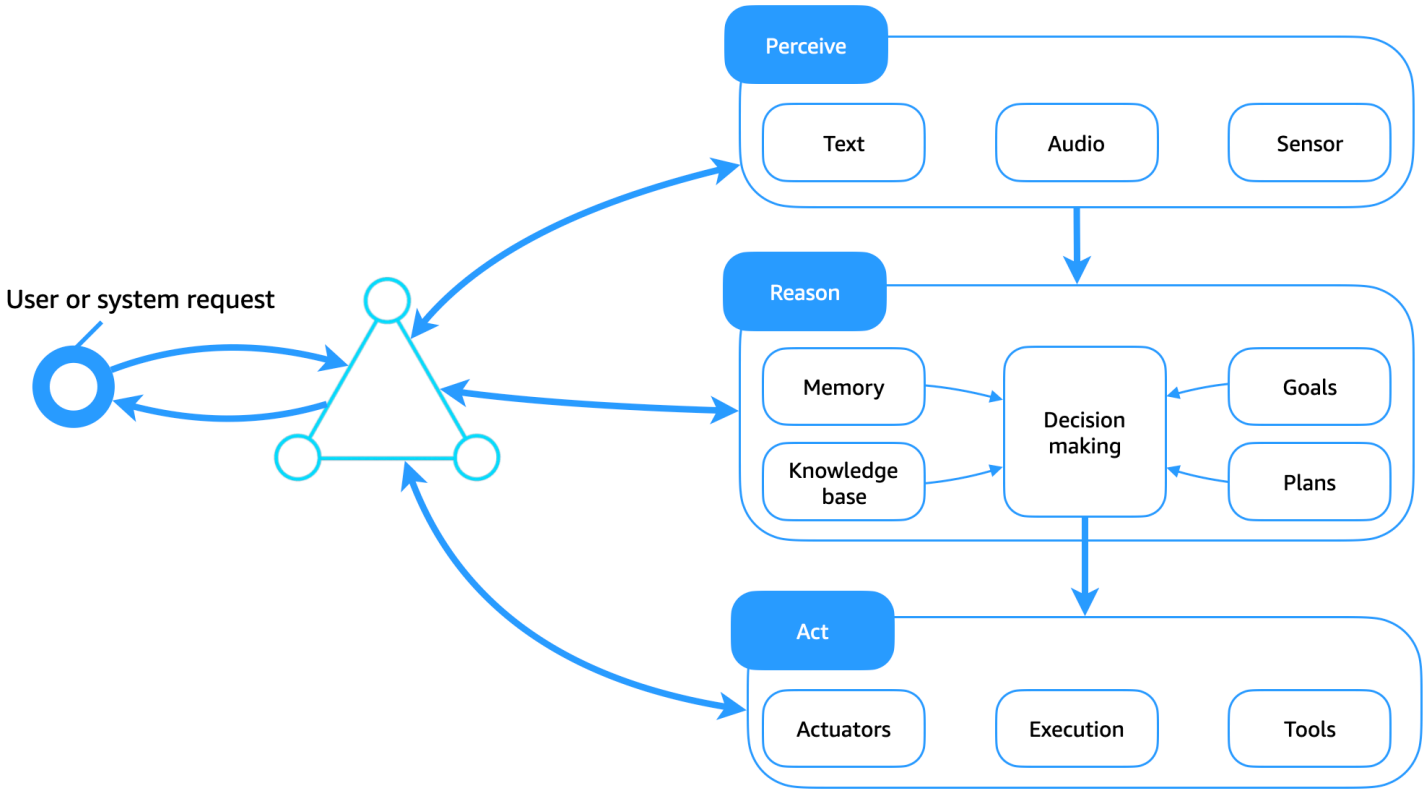
- 強化學習：透過在互動式環境中最大化累積獎勵，透過試驗和錯誤來最佳化行為。

  範例：了解哪些策略導致最快的任務完成。

Learning 與代理程式的認知模組緊密整合。它根據過去的結果精簡規劃策略，透過評估歷史成功來增強決策能力，並持續改善感知和動作之間的映射。透過此封閉式學習和意見回饋迴圈，客服人員會超越被動執行，成為能夠隨著時間適應新目標、條件和內容的自我改善系統。

傳統代理程式架構：感知、原因、行為

下圖說明上一節討論的建置區塊如何在感知、原因、動作週期下運作。



感知模組

感知模組做為代理程式與外部世界的感官界面。它將原始環境輸入轉換為告知推理的結構化表示法。這包括處理多模式資料，例如文字、音訊或感應器訊號。

- 文字輸入可能來自使用者命令、文件或對話。
- 音訊輸入包括語音指示或環境聲音。
- 感應器輸入會擷取動作、視覺摘要或 GPS 等真實訊號。

擷取原始輸入後，感知程序會執行特徵擷取，接著進行物件或事件辨識和語意解釋，以建立目前情況的有意義模型。這些輸出提供下游決策的結構化內容，並將代理程式的推理錨定在實際觀察中。

原因模組

原因模組是代理程式的認知核心。它評估內容、制定意圖，並確定適當的動作。本單元使用學習到的知識和推理來協調目標驅動的行為。

原因模組由緊密整合的子模組組成：

- 記憶體：維持短期和長期格式的對話狀態、任務內容和偶發歷史記錄。
- 知識庫：提供對符號規則、拓撲或學習模型（例如內嵌、事實和政策）的存取權。
- 目標和計劃：定義所需的成果，並建構動作策略以達成這些成果。目標可以動態更新，而且計劃可以根據意見回饋進行調整。
- 決策：透過權衡選項、評估權衡和選取下一個動作，充當中央仲裁引擎。此子模組會考量可信度閾值、目標對齊和內容限制。

這些元件共同允許代理程式推理其環境、更新想法、選取路徑，並以一致、適應性的方式行事。原因模組會縮小感知和行為之間的差距。

動作模組

動作模組會透過與數位或實體環境互動來執行任務，來執行代理程式選取的決策。這是意圖成為動作的地方。

本單元包含三個功能頻道：

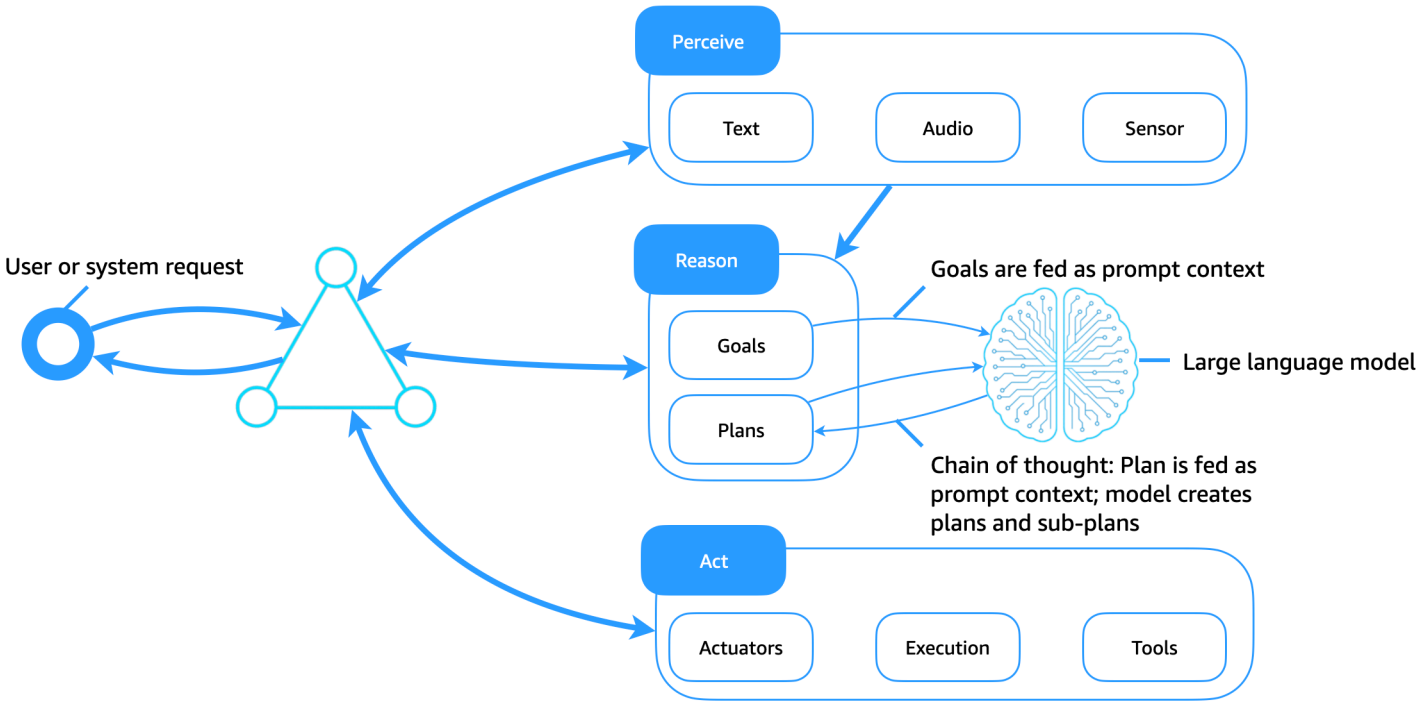
- 傳動器：對於具有實體存在的代理程式（例如機器人和 IoT 裝置），會控制硬體層級的互動，例如移動、操作或訊號。

- 執行：處理以軟體為基礎的動作，包括叫用 APIs、分派命令和更新系統。
- 工具：啟用搜尋、摘要、程式碼執行、計算和文件處理等功能。這些工具通常具有動態和內容感知，可擴展代理程式的公用程式。

動作模組的輸出會回饋環境並關閉迴圈。代理程式會再次感知這些結果。它們會更新客服人員的內部狀態並通知未來的決策，因此完成感知、原因、動作週期。

生成式 AI 代理器：以 LLMs 取代符號邏輯

下圖說明大型語言模型 (LLMs) 現在如何做為軟體代理程式的彈性和智慧型認知核心。與依賴靜態計劃程式庫和手動編碼規則的傳統符號邏輯系統相比，LLMs可實現適應性推理、內容規劃和動態工具使用，從而改變客服人員如何感知、推理和行動。



金鑰增強功能

此架構增強了傳統代理程式架構，如下所示：

- LLMs做為認知引擎：目標、計劃和查詢會做為 提示內容傳遞至模型。LLM 會產生推理路徑（例如思考鏈）、將任務分解為子目標，並決定下一個動作。
- 透過提示使用工具：LLMs可以透過工具使用代理程式或推理和動作 (ReAct) 提示來呼叫 APIs，以及搜尋、查詢、計算和解譯輸出。

- 內容感知規劃：客服人員會根據客服人員的目前目標、輸入環境和意見回饋動態產生或修改計劃，而不需要硬式編碼的計劃程式庫。
- 提示內容做為記憶體：客服人員不會使用符號知識庫，而是將記憶體、計劃和目標編碼為傳遞給模型的提示字符。
- 透過少量擷取內容學習進行學習：LLMs 透過提示工程來調整行為，從而減少明確重新訓練或嚴格計劃程式庫的需求。

在 LLM 型代理程式中實現長期記憶體

與將長期記憶體存放在結構化知識庫中的傳統代理程式不同，生成式 AI 代理程式必須在 LLMs的內容時段限制內運作。為了擴展記憶體並支援持久性智慧，生成式 AI 代理器使用數種補充技術：代理程式存放區、擷取增強生成 (RAG)、內容內學習和提示鏈結，以及預先訓練。

代理程式存放區：外部長期記憶體

客服人員狀態、使用者歷史記錄、決策和結果會存放在長期客服人員記憶體存放區 （例如向量資料庫、物件存放區或文件存放區）。相關記憶體會隨需擷取，並在執行時間插入 LLM 提示內容。這會建立持久性記憶體迴圈，其中代理程式會保留跨工作階段、任務或互動的連續性。

RAG

RAG 透過結合擷取的知識與生成功能來增強 LLM 效能。發出目標或查詢時，客服人員會搜尋擷取索引（例如，透過對文件的語意搜尋、先前的對話或結構化知識）。擷取的結果會附加到 LLM 提示中，這將導致產生外部事實或個人化內容。此方法可擴展代理程式的有效記憶體，並改善可靠性和事實正確性。

內容內學習和提示鏈結

客服人員會使用工作階段內字符內容和結構化提示鏈結來維護短期記憶體。內容元素，例如目前的計劃、先前的動作結果和客服人員狀態，會在呼叫之間傳遞，以引導行為。

持續預先訓練和微調

對於特定網域的代理程式，LLMs 可以在自訂集合上繼續預先訓練，例如日誌、企業資料或產品文件。或者，從人類意見回饋 (RLHF) 進行指令微調或強化學習，可以將類似客服人員的行為直接嵌入模型中。這會將推理模式從提示時間邏輯轉移到模型的內部表示法、減少提示長度並提高效率。

代理式 AI 的合併優勢

這些技術一起使用時，可讓生成式 AI 代理器：

- 隨著時間的推移保持情境感知。
- 根據使用者歷史記錄或偏好設定調整行為。
- 使用up-to-date、事實或私有知識做出決策。
- 使用持久性、合規且可解釋的行為擴展到企業使用案例。

透過使用外部記憶體、擷取層和持續訓練增強 LLMs，客服人員可以達成先前無法單獨透過符號系統達成的認知持續性和目的層級。

比較傳統 AI 與軟體代理程式和代理程式 AI

下表提供傳統 AI、軟體代理程式和代理程式 AI 的詳細比較。

特性	傳統 AI	軟體代理程式	代理式 AI
範例	垃圾郵件篩選條件、影像分類器、建議引擎	Chatbot、任務排程器、監控代理程式	AI 助理、自主開發人員代理程式、多代理程式 LLM 協調
執行模型	批次或同步	事件驅動或排程	非同步、事件驅動和目標驅動
自主性	有限；通常需要人工或外部協同運作	中；在預先定義的邊界內獨立運作	高；獨立執行自適應策略
反應	對輸入資料有反應	對環境和事件有反應	被動和主動；預測並啟動動作
主動性	罕見	存在於某些系統中	核心屬性；推動目標導向行為
Communication	最小；通常為獨立或 API 繫結	客服人員間或客服人員-人類傳訊	豐富的多代理程式和human-in-the-loop互動
決策	僅限模型推論（分類、預測等）	符號推理，或規則式或指令碼式決策	內容式、目標型、動態推理（通常是 LLM 增強型）

特性	傳統 AI	軟體代理程式	代理式 AI
委派的意圖	否；執行使用者直接定義的任務	部分；代表範圍有限的使用者或系統	是；通常跨服務、使用者或系統使用委派的目標
學習和適應	通常以模型為中心（例如 ML 訓練）	有時適應性	內嵌學習、記憶體或推理（例如，意見回饋、自我修正）
代理程式	無；人類的工具	隱含或基本	明確；以目的、目標和自我導向運作
內容感知	低；無狀態或以快照為基礎的	中度；某些狀態追蹤	高；使用記憶體、情境內容和環境模型
基礎結構角色	內嵌在應用程式或分析管道中	中介軟體或服務層元件	與雲端、無伺服器或邊緣系統整合的可編譯代理程式網格

綜上所述：

- 傳統 AI 以工具為中心，功能更窄。它著重於預測或分類。
- 傳統軟體代理程式引入自主權和基本通訊，但它們通常受到規則限制或靜態。
- 代理式 AI 將自主權、非同步和代理整合在一起。它可讓智慧、目標驅動的實體在複雜的系統中進行推理、行動和調整。這使得代理式 AI 非常適合雲端原生 AI 驅動的未來。

後續步驟

本指南討論代理程式 AI 的歷史和基礎，這代表傳統軟體代理程式演變為由生成式 AI 提供支援的自動化智慧型系統。它描述了早期軟體代理程式如何遵循預先定義的規則和邏輯來自動化固定界限內的任務，並說明代理程式 AI 如何透過整合大型語言模型來在此基礎上建置，讓代理程式能夠在開放式環境中進行推理、學習和動態調整。

您可以檢閱此系列中的下列出版物，以深入探索代理式 AI：

- 在 上操作代理式 AI AWS 提供組織策略，將代理式 AI 從隔離實驗轉換為企業規模、創造價值的基礎設施。
- 上的代理式 AI 模式和工作流程 AWS討論了用於設計、編寫和協調目標導向 AI 代理器的基礎藍圖和模組化建構。
- 上的客服人員 AI 架構、通訊協定和工具 AWS涵蓋了建置客服人員 AI 解決方案時要考慮的軟體基礎、工具組和通訊協定。
- 在 上建置代理式 AI 的無伺服器架構 AWS會討論無伺服器架構作為現代 AI 工作負載的自然基礎，並說明如何在 中建置 AI 原生無伺服器架構 AWS 雲端。
- 在 上建置代理式 AI 的多租用戶架構 AWS描述在多租用戶設定中使用 AI 代理器，包括託管考量、部署模型和控制平面。

資源

如需本指南所討論概念的詳細資訊，請參閱下列指南和文章。

AWS 參考

- Amazon Bedrock 代理程式
- Amazon Q Developer
- Strands Agents SDK

其他參考

- Hewitt、Carl、Peter Bishop 和 Richard Steiger。「通用模組化 ACTOR 人工智慧形式。」 第三次國際人工智慧聯合會議 (1973) 的進展：235-245。https：//https://www.ijcai.org/Proceedings/73/Papers/027B.pdf
- Lesser、Victor R.、相關出版物 (請參閱完整清單)：
  - Lesser、Victor R. 和 Daniel D. Corkill。「功能精確、合作分散式系統」。Systems、Man 和 Cybernetics 11 的 IEEE 交易，編號 1 (1981)：81-96。https：//https://ieeexplore.ieee.org/abstract/document/4308581
  - Decker、Keith S. 和 Victor R. Lesser。「協調服務中的通訊。」 AAAI 客服人員通訊規劃研討會 (1994)。https：//https://www.researchgate.net/profile/Victor-Lesser/publication/2768884_Communication_in_the_Service_of_Coordination/links/00b7d51cc2a0750cb4000000/Communication-in-the-Service-of-Coordination.pdf
  - Durfee、Edmund H.、Victor R. Lesser 和 Daniel D. Corkill。「合作分散式問題解決趨勢」。有關知識和資料工程的 IEEE 交易 (1989)。https：//http://mas.cs.umass.edu/Documents/ieee-tkde89.pdf
  - Durfee、Edmund H.、V.R. Lesser 和 D.D. Corkill，「分散式人工智慧」。透過分散式問題解決網路 (1987) 中的通訊進行合作：29-58。https：//https://www.academia.edu/download/79885643/durf94_1.pdf
  - Lâasri、Brigitte、Hassan Lâasri、Susan Lander 和 Victor Lesser。「智慧型溝通代理程式的一般模型。」 International Journal of Cooperative Information Systems 01，編號 02 (1992)：291-317。https：//https://doi.org/10.1142/S0218215792000210

  - Lander、Susan E. 和 Victor R. Lesser。「了解異質代理程式之間分散式搜尋中的溝通角色。」 IJCAI'93：第 13 屆人工智慧國際聯合會議 (1993) 的程序：438-444。https：//https://www.ijcai.org/Proceedings/93-1/Papers/062.pdf
  - Lander、Susan、Victor R. Lesser 和 Margaret E. Connell。「合作專家代理程式的衝突解決策略」CKBS'90：合作知識型系統的國際工作會議程序 (1990 年 10 月）：183-200。https：//https://doi.org/10.1007/978-1-4471-1831-2_10
  - Prasad、M.V. Nagendra、Victor Lesser 和 Susan E. Lander。「在異質多代理程式系統中學習實驗」。IJCAI-95 多代理程式系統調適和學習研討會 (1995)：59-64. https：//https://www.researchgate.net/publication/2784280_Learning_Experiments_in_a_Heterogeneous_Multi-agent_System
- Nwana、Hyacinth S.「軟體代理程式：概觀」。知識工程審查 11，否 3 (1996 年 10 月/11 月）：205-244。https：//https://teaching.shu.ac.uk/aces/rh1/elearning/multiagents/introduction/nwana.pdf
- Selfridge、Oliver G. "Pandemonium：A Paradigm for Learning"。思維程序的機制：在國家物理實驗室 1 (1959) 持有的研討會程序：511–529。https：//https://aitopics.org/download/classics:504E1BAC
- Vaswani、Ashish、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Acidian N. Gomez、Lukasz Kaiser 和 Illiabelsukhin。「您只需要注意。」 神經資訊處理系統 (NIPS) 第 31 屆會議的進展。神經資訊處理系統 30 (2017) 的進展：5998-6008。https：//https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
- Honeydridge、Michael 和 Nicholas R. Jennings。「Intelligent Agents：理論和實務。」 知識工程審查 10，否。2 (1995 年 1 月）：115-152。https：//https://www.cs.cmu.edu/~motionplanning/papers/sbp_papers/integrated1/woodridge_intelligent_agents.pdf

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	—	2025 年 7 月 14 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，但比主動-被動遷移需要更多的工作。

主動-被動式遷移

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

彙總函數

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

AI

請參閱人工智慧。

AIOps

請參閱人工智慧操作。

匿名化

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

反模式

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

應用程式控制

一種安全方法，僅允許使用核准的應用程式，以協助保護系統免受惡意軟體攻擊。

應用程式組合

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

人工智慧 (AI)

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

人工智慧操作 (AIOps)

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

非對稱加密

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

原子性、一致性、隔離性、耐久性 (ACID)

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

屬性型存取控制 (ABAC)

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

授權資料來源

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

可用區域

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

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。因此， AWS CAF 為人員開發、訓練和通訊提供指引，協助組織做好成功採用雲端的準備。如需詳細資訊，請參閱 AWS CAF 網站和 AWS CAF 白皮書。

AWS 工作負載資格架構 (AWS WQF)

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。有些其他機器人稱為惡意機器人，旨在中斷或傷害個人或組織。

殭屍網路

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

分支

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

碎片存取

在特殊情況下，並透過核准的程序，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) 執行試驗，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

部落格文章中的 Stephen Orban 定義了這些階段： AWS 雲端 企業策略部落格上的邁向雲端優先之旅和採用階段。如需有關它們如何與 AWS 遷移策略相關的詳細資訊，請參閱遷移整備指南。

CMDB

請參閱組態管理資料庫。

程式碼儲存庫

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

冷快取

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

冷資料

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

電腦視覺 (CV)

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

組態偏離

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

組態管理資料庫 (CMDB)

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

一致性套件

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

持續整合和持續交付 (CI/CD)

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

CV

請參閱電腦視覺。

D

靜態資料

網路中靜止的資料，例如儲存中的資料。

資料分類

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

資料偏離

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

傳輸中的資料

在您的網路中主動移動的資料，例如在網路資源之間移動。

資料網格

架構架構，提供分散式、分散式資料擁有權與集中式管理。

資料最小化

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

資料周邊

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

資料預先處理

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

資料來源

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

資料主體

正在收集和處理其資料的個人。

資料倉儲

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

資料庫定義語言 (DDL)

用於建立或修改資料庫中資料表和物件之結構的陳述式或命令。

資料庫處理語言 (DML)

用於修改 (插入、更新和刪除) 資料庫中資訊的陳述式或命令。

DDL

請參閱資料庫定義語言。

深度整體

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

深度學習

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

深度防禦

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

委派的管理員

在 中 AWS Organizations，相容的服務可以註冊 AWS 成員帳戶，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並排定優先順序。DVSM 擴展了最初專為精簡製造實務設計的價值串流映射程序。它著重於透過軟體開發程序建立和移動價值所需的步驟和團隊。

數位分身

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，其行為類似於文字。這些屬性通常用於查詢限制、篩選和結果集標記。

災難

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

災難復原 (DR)

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

DML

請參閱資料庫處理語言。

領域驅動的設計

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

DR

請參閱災難復原。

偏離偵測

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

DVSM

請參閱開發值串流映射。

E

EDA

請參閱探索性資料分析。

EDI

請參閱電子資料交換。

邊緣運算

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

電子資料交換 (EDI)

在組織之間自動交換商業文件。如需詳細資訊，請參閱什麼是電子資料交換。

加密

將人類可讀取的純文字資料轉換為加密文字的運算程序。

加密金鑰

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

端序

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

端點

請參閱 服務端點。

端點服務

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

企業資源規劃 (ERP)

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

信封加密

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

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給 LLM。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，已在廣義和未標記資料的大量資料集上進行訓練。FMs 能夠執行各種一般任務，例如了解語言、產生文字和影像，以及以自然語言交談。如需詳細資訊，請參閱什麼是基礎模型。

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 系統旨在自動容錯移轉、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，請參閱在 上實作微服務 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 建議使用不可變的基礎設施作為最佳實務。

O

OAC

請參閱原始存取控制。

OAI

請參閱原始存取身分。

OCM

請參閱組織變更管理。

離線遷移

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

OI

請參閱 操作整合。

OLA

請參閱操作層級協議。

線上遷移

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

OPC-UA

請參閱開啟程序通訊 - 統一架構。

開放程序通訊 - 統一架構 (OPC-UA)

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

操作水準協議 (OLA)

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

操作整備審查 (ORR)

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

操作技術 (OT)

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

操作整合 (OI)

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

組織追蹤

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 支援使用 S3 AWS KMS (SSE-KMS) 的所有伺服器端加密中的所有 S3 儲存貯體 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)

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

PII

請參閱個人身分識別資訊。

手冊

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

PLC

請參閱可程式設計邏輯控制器。

PLM

請參閱產品生命週期管理。

政策

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

混合持久性

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

組合評定

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

述詞

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

述詞下推

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

預防性控制

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

委託人

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

依設計的隱私權

透過整個開發程序將隱私權納入考量的系統工程方法。

私有託管區域

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

主動控制

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

產品生命週期管理 (PLM)

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

生產環境

請參閱 環境。

可程式設計邏輯控制器 (PLC)

在製造中，高度可靠、可調整的電腦，可監控機器並自動化製造程序。

提示鏈結

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

擬匿名化

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，以提高可擴展性和回應能力。例如，在微服務型 MES 中，微服務可以將事件訊息發佈到其他微服務可訂閱的頻道。系統可以新增新的微服務，而無需變更發佈服務。

Q

查詢計劃

一系列步驟，如指示，用於存取 SQL 關聯式資料庫系統中的資料。

查詢計劃迴歸

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

R

RACI 矩陣

請參閱負責、負責、諮詢、告知 (RACI)。

RAG

請參閱擷取增強生成。

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 個 R。

Replatform

請參閱 7 個 R。

回購

請參閱 7 個 R。

彈性

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

資源型政策

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

負責者、當責者、事先諮詢者和事後告知者 (RACI) 矩陣

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

回應性控制

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

保留

請參閱 7 個 R。

淘汰

請參閱 7 個 R。

檢索增強生成 (RAG)

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

輪換

定期更新秘密的程序，讓攻擊者更難存取登入資料。

資料列和資料欄存取控制 (RCAC)

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

RPO

請參閱復原點目標。

RTO

請參閱復原時間目標。

執行手冊

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

S

SAML 2.0

許多身分提供者 (IdP) 使用的開放標準。此功能會啟用聯合單一登入 (SSO)，AWS 管理主控台 讓使用者可以登入 或呼叫 AWS API 操作，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

請參閱服務控制政策。

秘密

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

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。

未區分的任務

也稱為繁重工作，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，通常可接受中等速度的查詢。

視窗函數

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

工作負載

提供商業價值的資源和程式碼集合，例如面向客戶的應用程式或後端流程。

工作串流

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

WORM

請參閱寫入一次，讀取許多。

WQF

請參閱AWS 工作負載資格架構。

寫入一次，讀取許多 (WORM)

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

Z

零時差入侵

利用零時差漏洞的攻擊，通常是惡意軟體。

零時差漏洞

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

零鏡頭提示

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

殭屍應用程式

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

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。