上的客服人員 AI 架構、平台、通訊協定和工具 AWS

AWS 方案指引

AWS 方案指引: 上的客服人員 AI 架構、平台、通訊協定和工具 AWS

Table of Contents

上的客服人員 AI 架構、平台、通訊協定和工具 AWS

Aaron Sempf、Ansley Verzosa 和 Joshua Samuel，Amazon Web Services (AWS)

2026 年 1 月 (文件歷史記錄)

代理式 AI 是 AI、分散式系統和軟體工程交集的強大範例。這是一個智慧型系統類別，由使用 AI 模型並與工具和資源整合的自動、非同步軟體代理程式組成。客服人員代表使用者或系統，展現代理、可以感知內容、對目標的推理、做出決策，以及採取有目的的動作。這些代理程式通常在分散式環境中獨立運作，旨在追求具有內嵌智慧、記憶體和意圖的委派目標。

在 上 AWS，組織可以利用代理式 AI 來自動化複雜的工作流程、增強決策程序，以及建立更回應的系統。本指南提供有關建置有效代理式 AI 解決方案所需的關鍵元件的資訊：

- 架構會描述目前的代理式 AI 架構，包括檢閱其優點和使用案例。了解這些架構如何減少跨模式、通訊協定和工具的繁重工作。了解金鑰選擇條件，為您的需求選擇正確的架構。
- 平台提供代理式 AI 平台（受管代理程式、開放原始碼協同運作和混合）的概觀，以及選擇或設計的考量事項。
- 通訊協定探索客服人員互動所需的基本標準化通訊協定。Agent-to-agent程式通訊協定正在出現，例如開放原始碼模型內容通訊協定 (MCP) 和 Agent2Agent (A2A)，以及其他專屬實作。探索常見的通訊協定如何讓不同的通訊協定無縫互動。
- 工具提供有關通訊協定型工具（例如 MCP)、架構原生工具和中繼工具的資訊。組織可以建置工具組，與其工作流程中的關鍵系統整合，同時啟用最終使用者和伺服器型代理程式工作流程。

目標對象

本指南適用於尋求在現代雲端原生應用程式中利用 AI 驅動軟體代理程式的架構師、開發人員和技術領導者。

目標

本指南可協助您執行以下操作：

- 比較不同的客服人員 AI 架構，為您的使用案例選擇最適合的架構。
- 了解客服人員 AI 平台，這些平台提供將個別客服人員轉換為協調、適應性系統的功能。
- 了解開放式通訊協定在建置永續性代理式 AI 架構方面的優勢。

- 在建置代理程式系統時，建立適當的工具整合策略。

關於此內容系列

本指南是代理式 AI 相關系列的一部分 AWS。如需詳細資訊和檢視此系列中的其他指南，請參閱 AWS 規範性指導網站上的 客服人員 AI 。

架構

上的代理式 AI 基礎 AWS會檢查啟用自動目標導向行為的核心模式和工作流程。實作這些模式的核心是架構的選擇。框架是預先編寫程式碼的軟體基礎，可提供結構化環境和常見功能，用於建置和管理建置生產就緒的自動 AI 代理器所需的工具和協同運作功能。

有效的代理式 AI 架構提供數種基本功能，可將原始大型語言模型 (LLM) 互動轉換為協調的智慧型系統，能夠推理、協作和採取行動：

- 客服人員協調協調跨單一或多個客服人員的資訊流程和決策，無需人為介入即可實現複雜的目標。
- 工具整合可讓客服人員與外部系統、APIs和資料來源互動，將功能延伸到語言處理之外。如需詳細資訊，請參閱 Strands Agents 文件中的工具概觀。
- 記憶體管理提供持久性或工作階段型狀態，以維護互動之間的內容，對於長時間執行或適應性任務至關重要。更進階的架構整合了長期記憶體來儲存摘要和使用者偏好設定，從而實現個人化和上下文感知的客服人員體驗。如需詳細資訊，請參閱LangChain部落格上的如何考慮客服人員架構。
- 工作流程定義支援結構化模式，例如鏈、路由、平行化和反射迴圈，可實現複雜的自主推理。
- 部署和監控透過自動化系統的可觀測性，促進從開發到生產的轉換。如需詳細資訊，請參閱 Amazon Bedrock AgentCore 一般可用性公告。

這些功能會在架構環境中使用不同的方法和階段實作，每個功能都為不同的自動代理程式使用案例和組織內容提供不同的優勢。

本節將介紹並比較建置代理式 AI 解決方案的領導架構，並著重於其優勢、限制和自動化操作的理想使用案例：

- Strands 代理程式
- LangChain 和 LangGraph
- CrewAI
- AutoGen
- ???
- 比較代理式 AI 架構

> **Note**
>
> 本節涵蓋專門支援 AI 代理程式的架構，且不涵蓋沒有 代理程式的前端界面或生成式 AI。

Strands Agents

Strands Agents 是一種開放原始碼 SDK，最初由 發行 AWS，如AWS 開放原始碼部落格中所述。Strands Agents 旨在使用模型優先的方法建置自主 AI 代理器。它提供靈活、可擴展的架構，旨在無縫搭配 使用， AWS 服務 同時保持開放與第三方元件整合。Strands Agents 非常適合建置完全自主的解決方案。

的主要功能 Strands Agents

Strands Agents 包含下列主要功能：

- 模型優先設計 – 建立的概念是基礎模型是代理程式智慧的核心，可實現複雜的自主推理。如需詳細資訊，請參閱 Strands Agents 文件中的客服人員迴圈。
- 多客服人員協作模式 – 內建協調模式，例如 Swarm、Graph 和工作流程模式，可在分散式客服人員網路之間實現可擴展的協作和管理。如需詳細資訊，請參閱 Strands Agents 文件中的多重代理程式模式。
- MCP 整合 – 原生支援模型內容通訊協定 (MCP)，為 LLMs提供標準化內容佈建，以實現一致的自動操作。
- AWS 服務 整合 – 與 Amazon Bedrock AWS Lambda、 和其他 AWS 服務 無縫連線 AWS Step Functions，以實現全面的自主工作流程。如需詳細資訊，請參閱AWS 每週總和 (AWS 部落格 ）。
- 基礎模型選擇 – 支援 Amazon Bedrock 上的各種基礎模型，包括 Anthropic Claude、Amazon Nova (Premier、Pro、Lite 和 Micro) 等，以針對不同的自動推理功能進行最佳化。如需詳細資訊，請參閱 Strands Agents 文件中的 Amazon Bedrock。
- LLM API 整合 – 與不同 LLM 服務介面的彈性整合，包括 Amazon Bedrock、OpenAI 和其他用於生產部署的介面。如需詳細資訊，請參閱 Strands Agents 文件中的 Amazon Bedrock 基本用量。
- 多模態功能 – 支援多種模態，包括文字、語音和影像處理，以進行全面的自動代理程式互動。如需詳細資訊，請參閱 Strands Agents 文件中的 Amazon Bedrock 多模式支援。
- 工具生態系統 – 豐富的 AWS 服務 互動工具集，具有擴展自動化功能的自訂工具的可擴展性。如需詳細資訊，請參閱 Strands Agents 文件中的工具概觀。

使用時機 Strands Agents

Strands Agents 特別適合自動代理程式案例，包括：

- 以希望原生與 整合 AWS 服務 以進行自動化工作流程的 AWS 基礎設施為基礎的組織
- 需要生產自動化系統企業級安全性、可擴展性和合規功能的團隊
- 需要跨不同供應商靈活選擇模型的專案，以進行專門的自動化任務
- 需要與現有 AWS 工作流程和資源緊密整合以進行端對端自主程序的使用案例

的實作方法 Strands Agents

Strands Agents 為業務利益相關者提供直接的實作方法，如 的 Quickstart 指南Python和 的 Quickstart 指南Typescript中所述。架構可讓組織：

- 根據特定業務需求，在 Amazon Bedrock 上選取基礎模型，例如 Amazon Nova (Premier、Pro、Lite 或 Micro)。
- 定義連接到企業系統和資料來源的自訂工具。
- 處理多個模態，包括文字、影像和語音。
- 部署可以自動回應業務查詢並執行任務的代理程式。

這種實作方法可讓業務團隊快速開發和部署自動代理程式，而無需 AI 模型開發的深厚技術專業知識。

的實際範例 Strands Agents

AWS Transform for .NET 使用 Strands Agents 為其應用程式現代化功能提供支援，如 AWS Transform for .NET 中所述，這是第一個大規模現代化 .NET 應用程式的代理式 AI 服務 (AWS 部落格）。此生產服務採用多個專門的自主代理程式。代理程式會共同分析舊版 .NET 應用程式、規劃現代化策略，以及執行程式碼轉換至雲端原生架構，無需人工介入。 AWS Transform for .NET 示範 Strands Agents企業自動化系統的生產準備程度。

LangChain 和 LangGraph

LangChain 是代理式 AI 生態系統中最成熟的架構之一。 LangGraph擴展其功能以支援複雜且具狀態的代理程式工作流程，如 LangChain 部落格中所述。它們共同提供全方位解決方案，以建置複雜的自主 AI 代理器，並具有豐富的協同運作功能，用於獨立操作。

LangChain 和 的主要功能 LangGraph

LangChain 和 LangGraph包含下列主要功能：

- 元件生態系統 – 適用於各種自動代理程式功能的大量預先建置元件程式庫，可快速開發專用代理程式。如需詳細資訊，請參閱 LangChain 文件中的 Quickstart。
- 基礎模型選擇 – 支援 Amazon Bedrock 上的各種基礎模型，包括 Anthropic Claude、Amazon Nova 模型 (Premier、Pro、Lite 和 Micro)，以及其他具有不同推理功能的模型。如需詳細資訊，請參閱 LangChain 文件中的輸入和輸出。
- LLM API 整合 – 適用於多個大型語言模型 (LLM) 服務提供者的標準化界面，包括 Amazon BedrockOpenAI、 和其他可靈活部署的 。如需詳細資訊，請參閱 LangChain 文件中的 LLMs。
- 多模態處理 – 內建支援文字、影像和音訊處理，以啟用豐富的多模態自動代理程式互動。如需詳細資訊，請參閱 LangChain 文件中的多模態。
- 以圖形為基礎的工作流程 – LangGraph可將複雜的自主代理程式行為定義為狀態機器，支援複雜的決策邏輯。如需詳細資訊，請參閱LangGraph平台 GA 公告。
- 記憶體抽象化 – 短期和長期記憶體管理的多個選項，這對隨著時間維持內容的自動代理程式至關重要。如需詳細資訊，請參閱 LangChain 文件中的如何將記憶體新增至聊天機器人。
- 工具整合 – 跨各種服務和 APIs 的豐富工具整合生態系統，擴展自動代理程式功能。如需詳細資訊，請參閱 LangChain 文件中的工具。
- LangGraph 平台 – 生產環境的受管部署和監控解決方案，支援長時間執行的自動代理程式。如需詳細資訊，請參閱LangGraph平台 GA 公告。

何時使用 LangChain和 LangGraph

LangChain 和 LangGraph特別適合自動代理程式案例，包括：

- 複雜的多步驟推理工作流程，需要複雜的協同運作來進行自動決策
- 需要存取大型生態系統的專案，這些生態系統包含預先建置的元件和整合，以實現各種自動化功能
- 與想要建置自動化系統的現有 Python型機器學習 (ML) 基礎設施和專業知識的團隊
- 需要跨長時間執行的自動代理程式工作階段進行複雜狀態管理的使用案例

LangChain 和 的實作方法 LangGraph

LangChain 並為業務利益相關者LangGraph提供結構化實作方法，如 LangGraph 文件中所述。此架構可讓組織：

- 定義代表業務流程的複雜工作流程圖表。
- 使用決策點和條件式邏輯建立多步驟推理模式。
- 整合多模式處理功能，以處理各種資料類型。
- 透過內建的檢閱和驗證機制實作品質控制。

這種以圖形為基礎的方法可讓業務團隊將複雜的決策程序建模為自動化工作流程。團隊清楚了解推理程序的每個步驟，以及稽核決策路徑的能力。

LangChain 和 的實際範例 LangGraph

Vodafone 已使用 LangChain (和 LangGraph) 實作自動代理程式，以增強其資料工程和操作工作流程，如LangChain企業案例研究中所述。他們建立了內部 AI 助理，可透過自然語言互動自動監控效能指標、從文件系統擷取資訊，以及提供可行的洞見。

Vodafone 實作使用LangChain模組化文件載入器、向量整合和對多個 LLMs(OpenAI、LLaMA#3 和 Gemini) 的支援，來快速建立原型並對這些管道進行基準測試。然後，它們透過部署模組化子代理程式LangGraph來建構多代理程式協同運作。這些代理程式會執行收集、處理、摘要和推理任務。 透過APIs將這些代理程式LangGraph整合到其雲端系統中。

CrewAI

CrewAI 是一種開放原始碼架構，特別著重於自動多代理程式協同運作，可在 GitHub 上取得。它提供結構化方法來建立專業自主代理程式團隊，以協作解決複雜的任務，而無需人工介入。 CrewAI強調以角色為基礎的協調和任務委派。

的主要功能 CrewAI

CrewAI 提供下列主要功能：

- 以角色為基礎的代理程式設計 – 自治代理程式是以特定角色、目標和背景案例定義，以啟用專業專業知識。如需詳細資訊，請參閱 CrewAI 文件中的建立有效的代理程式。
- 任務委派 – 內建機制，可根據任務的功能自動將任務指派給適當的客服人員。如需詳細資訊，請參閱 CrewAI 文件中的任務。
- 客服人員協作 – 用於自動客服人員間通訊和知識分享的架構，無需人工調解。如需詳細資訊，請參閱 CrewAI 文件中的協同合作。
- 程序管理 – 用於循序和平行自主任務執行的結構化工作流程。如需詳細資訊，請參閱 CrewAI 文件中的處理程序。

- 基礎模型選擇 – 支援 Amazon Bedrock 上的各種基礎模型，包括 Anthropic Claude、Amazon Nova 模型 (Premier、Pro、Lite 和 Micro)，以及其他可針對不同自主推理任務進行最佳化的基礎模型。如需詳細資訊，請參閱 CrewAI 文件中的 LLMs。
- LLM API 整合 – 與多個 LLM 服務介面的彈性整合，包括 Amazon BedrockOpenAI、 和本機模型部署。如需詳細資訊，請參閱 CrewAI 文件中的提供者組態範例。
- 多模式支援 – 用於處理文字、影像和其他模態的新興功能，以實現全面的自動代理程式互動。如需詳細資訊，請參閱 CrewAI 文件中的使用多模式代理程式。

使用時機 CrewAI

CrewAI 特別適合自動代理程式案例，包括：

- 從專業、以角色為基礎的專業知識中受益的複雜問題，可自主運作
- 需要多個自主代理程式之間明確協同合作的專案
- 以團隊為基礎的問題分解改善自動解決問題的使用案例
- 需要在不同自動代理程式角色之間明確分離疑慮的情況

的實作方法 CrewAI

CrewAI 為業務利益相關者提供以角色為基礎的 AI 代理程式團隊方法實作，如 CrewAI 文件入門中所述。此架構可讓組織：

- 定義具有特定角色、目標和專業知識領域的特殊自動代理程式。
- 根據客服人員的專門功能，將任務指派給客服人員。
- 在任務之間建立明確的相依性，以建立結構化工作流程。
- 協調多個客服人員之間的協同合作，以解決複雜的問題。

這種以角色為基礎的方法反映了人類團隊結構，讓業務領導者能夠直覺地了解和實作。組織可以建立具有專業知識領域的自主團隊，以協同合作實現業務目標，類似於人類團隊的運作方式。不過，自動化團隊可以在無需人工介入的情況下持續運作。

的實際範例 CrewAI

AWS 已如CrewAI已發佈的案例研究所述，使用與 Amazon Bedrock 整合的CrewAI」實作自動多代理程式系統。 AWS 並CrewAI開發了安全的廠商中立架構。CrewAI 開放原始碼「流程和資源」架構與 Amazon Bedrock 基礎模型、記憶體系統和合規護欄無縫整合。

實作的關鍵元素包括：

- 藍圖和開放原始碼 – AWS 以及CrewAI發佈的參考設計，可將CrewAI代理程式映射至 Amazon Bedrock 模型和可觀測性工具。他們還發佈了範例系統，例如多代理程式 AWS 安全稽核人員、程式碼現代化流程和消費者包裝商品 (CPG) 後台自動化。
- 可觀測性堆疊整合 – 解決方案使用 Amazon CloudWatch、 AgentOps和 內嵌監控LangFuse，實現從概念驗證到生產的可追蹤性和偵錯。
- 已證明投資報酬率 (ROI) – 早期試行者展現了重大改進：大型程式碼現代化專案的執行速度快 70#%，以及 CPG 後台流程的處理時間縮短約 90#%。

AutoGen

AutoGen 是 最初發行的開放原始碼架構Microsoft。 AutoGen 專注於啟用對話式和協作式自主 AI 代理器。它提供彈性的架構，用於建置多代理程式系統，強調代理程式之間用於複雜自主工作流程的非同步、事件驅動的互動。

的主要功能 AutoGen

AutoGen 提供下列主要功能：

- 對話式客服人員 – 以自動客服人員之間的自然語言對話為基礎，透過對話實現複雜的推理。如需詳細資訊，請參閱 AutoGen 文件中的多客服人員對話架構。
- 非同步架構 – 非封鎖自動代理程式互動的事件驅動設計，支援複雜的平行工作流程。如需詳細資訊，請參閱 AutoGen 文件中的解決非同步聊天序列中的多個任務。
- Human-in-the-loop – 視需要支援選用人工參與自動代理程式工作流程。如需詳細資訊，請參閱 AutoGen 文件中的允許客服人員的人工意見回饋。
- 程式碼產生和執行 – 適用於程式碼導向自動代理程式的特殊功能，可寫入和執行程式碼。如需詳細資訊，請參閱 AutoGen 文件中的程式碼執行。
- 可自訂的行為 – 針對各種使用案例的彈性自動代理程式組態和對話控制。如需詳細資訊，請參閱 AutoGen 文件中的 agentchat.conversable_agent。

- 基礎模型選擇 – 在 Amazon Bedrock 上支援各種基礎模型，包括 Anthropic Claude、Amazon Nova 模型 (Premier、Pro、Lite 和 Micro)，以及其他具有不同自動推理功能的模型。如需詳細資訊，請參閱 AutoGen 文件中的 LLM 組態。
- LLM API 整合 – 多個 LLM 服務介面的標準化組態，包括 Amazon BedrockOpenAI、 和 Azure OpenAI。如需詳細資訊，請參閱 AutoGen API 參考中的 oai.openai_utils。
- 多模態處理 – 支援文字和影像處理，以啟用豐富的多模態自動代理程式互動。如需詳細資訊，請參閱 AutoGen 文件中的在 中使用多模態模型：GPT-4VAutoGen。

使用時機 AutoGen

AutoGen 特別適合自動代理程式案例，包括：

- 需要自動代理程式之間自然對話流程的應用程式，以進行複雜的推理
- 同時需要完全自主操作和選用人工監督功能的專案
- 涉及自動程式碼產生、執行和偵錯的使用案例，無需人工介入
- 需要彈性、非同步自主代理程式通訊模式的案例

的實作方法 AutoGen

AutoGen 為業務利益相關者提供對話式實作方法，如 AutoGen 文件入門中所述。此架構可讓組織：

- 建立自主代理程式，透過自然語言對話進行通訊。
- 在多個客服人員之間實作非同步、事件驅動的互動。
- 必要時，結合全自主操作與選用的人工監督。
- 為透過對話協作的不同業務職能開發專門的客服人員。

這種對話式方法可讓自動化系統的推理透明化，並可供商業使用者存取。決策者可以觀察客服人員之間的對話，以了解如何得出結論，並在需要人工判斷時選擇性地參與對話。

的實際範例 AutoGen

Magentic-One 是一種開放原始碼、通則多代理程式系統，旨在跨各種環境自動解決複雜的多步驟任務，如 Microsoft AI Frontiers 部落格中所述。其核心是 Orchestrator 代理程式，它會分解高階目標，並使用結構化總帳追蹤進度。此代理程式會將子任務委派給專業代理程式 （例如 WebSurfer、Coder、 和 ComputerTerminal)FileSurfer，並在必要時重新規劃以動態調整。

系統是以AutoGen架構為基礎，且與模型無關，預設為 GPT-4o。它可在 GAIA、 AssistantBench 和 等基準測試中實現最先進的效能WebArena，完全無需進行任務特定的調校。此外，它還透過 AutoGenBench建議支援模組化可擴展性和嚴格的評估。

LlamaIndex

LlamaIndex 是一種資料架構，專門用於將大型語言模型 (LLMs) 與外部資料來源連線，以啟用複雜的擷取增強生成 (RAG) 和代理式 AI 應用程式。框架為代理程式系統、自訂協同運作模式和系統整合提供抽象和加速開發工作流程，以減少知識驅動型 AI 解決方案的time-to-production。

的主要功能 LlamaIndex

LlamaIndex 提供一組完整的功能，特別適合企業代理式 AI 應用程式：

- 以資料為中心的架構 – 從超過 100 種資料格式擷取、編製索引和擷取資訊的 ExcelPDFs、MicrosoftWord 文件、試算表等。框架會將企業資料轉換為可查詢的知識庫，這些知識庫已針對 AI 代理器進行最佳化。如需詳細資訊，請參閱 LlamaIndex 文件。
- 生產就緒部署 – 透過 LlamaIndex同時提供開放原始碼架構和受管服務LlamaCloud，提供企業級功能，包括安全控制、可擴展性、可觀測性整合和部署彈性。如需詳細資訊，請參閱LlamaIndex架構文件。
- 進階文件處理 – LlamaCloud提供文件剖析、擷取、索引和擷取功能，可處理複雜的配置、巢狀資料表、多模態內容，甚至是手寫筆記。這種複雜的剖析可讓客服人員有效地處理包含圖表、圖表和複雜格式的真實世界企業文件。如需詳細資訊，請參閱 LlamaCloud 文件。
- 工作流程協同運作 – LlamaAgents提供事件驅動、非同步優先協同運作引擎，用於建置多步驟代理系統。工作流程支援複雜的模式，包括迴圈、平行執行、條件式分支和有狀態恢復，使其非常適合複雜的客服人員互動。如需詳細資訊，請參閱LlamaIndex工作流程文件。
- 代理程式擷取功能 – 進階擷取模式，包括混合搜尋、語意搜尋和自動路由，可智慧地判斷每個查詢的最佳擷取策略。此架構支援跨多個知識庫的複合擷取，並透過重新排名來提高準確性。如需詳細資訊，請參閱 LlamaIndex RAG 文件。
- 可觀測性和評估 – 與各種可觀測性和評估工具LlamaIndex整合。此整合功能可協助您追蹤和偵錯應用程式、評估其效能，以及監控成本。如需詳細資訊，請參閱追蹤、偵錯和評估LlamaIndex文件。

使用時機 LlamaIndex

LlamaIndex 特別適合強調資料密集型工作流程和知識管理的客服人員 AI 案例：

- 文件密集型應用程式，需要客服人員處理、分析和擷取大量企業文件的洞見，例如合約、報告、手冊和法規文件
- 生產案例的快速原型設計，其中組織想要快速建置和部署以文件為中心的代理程式，而無需大量的基礎設施管理開銷
- 優先考慮擷取準確性和內容相關性的 RAG 架構，特別是在使用包含資料表、影像和結構化資料的複雜多模式文件時
- 多代理程式文件工作流程，需要專門的代理程式來處理文件的不同層面，例如剖析、分析、摘要和合規檢查

的實作方法 LlamaIndex

LlamaIndex 提供低階建置區塊和高階抽象，可適應不同的實作方法：

- 使用LlamaIndex高階 APIs，在幾行程式碼中快速開發功能性 RAG 應用程式。此方法可讓初次接觸代理式 AI 的業務團隊和開發人員LlamaIndex存取。
- 透過 LlamaHub 進行企業整合，適用於熱門企業系統，包括 SharePoint、Amazon Simple Storage Service (Amazon S3)、資料庫和 APIs。此方法可讓 與現有資料基礎設施無縫整合。
- 開放原始碼自我託管部署之間的彈性部署選項，可實現最大控制，或可降低營運開銷和企業功能的LlamaCloud受管服務。
- 應用程式可以從簡單的查詢引擎開始，並隨著需求的演進逐步新增代理程式功能、多重代理程式協同運作和複雜的工作流程。

的實際範例 LlamaIndex

此範例著重於一家專門提供航空導航和操作解決方案的航太公司的子公司。他們需要解決不斷增長的挑戰，這涉及試行不協調的 AI 聊天機器人試驗。這些試驗導致整個組織的重複工作、長開發週期、合規障礙和隔離實作。

他們開發了統一的代理程式架構，這是以LlamaIndex開放原始碼架構為基礎的可重複使用範本型解決方案，可讓代理程式建立更有效率。它們比較了數個競爭架構，包括鏈結導向和圖形型架構。最後，他們選擇了LlamaIndex三個關鍵優勢：其彈性設計、模組化元件和生產就緒的協同運作控制。

平台可將代理程式開發和部署時間從 512 縮短 87%，縮短為 64 小時。透過讓團隊建置具有大約 50 行程式碼和 JSON 組態檔案的客服人員，即可實現此減少。團隊利用具有內建安全性、合規性和特殊權限系統存取的統一架構。如需詳細資訊，請參閱LlamaIndex客戶案例研究。

比較代理式 AI 架構

選擇適用於自動代理程式開發的代理程式 AI 架構時，請考慮每個選項如何符合您的特定需求。不僅考慮其技術能力，還考慮其組織適用性，包括團隊專業知識、現有基礎設施和長期維護需求。許多組織可能受益於混合式方法，利用多個架構來實現自動化 AI 生態系統的不同元件。

下表比較各架構在關鍵技術維度之間的成熟度等級（最強、強、適當或弱）。對於每個架構，資料表也包含生產部署選項和學習曲線複雜性的相關資訊。

架構	AWS 整合	自動多重代理程式支援	自主工作流程複雜性	多模式功能	基礎模型選擇	LLM API 整合	生產部署	學習曲線
AutoGen	脆弱	強	強	適當	適當	強	自行執行 (DIY)	陡峭
CrewAI	脆弱	強	適當	脆弱	適當	適當	DIY	適中
LangChain / LangGraph	適當	強	最強	最強	最強	最強	平台或 DIY	陡峭
LlamaIndex	適當	適當	強	適當	強	強	平台或 DIY	適中
Strands Agents	最強	強	最強	強	強	最強	DIY	適中

選擇代理式 AI 架構時的考量事項

開發自動代理程式時，請考慮下列關鍵因素：

- AWS 基礎設施整合 – 大量投資的組織 AWS 將受益於 Strands Agents 與 的原生整合 AWS 服務，以進行自動化工作流程。如需詳細資訊，請參閱AWS 每週總和 (AWS 部落格）。

- 基礎模型選擇 – 根據自動代理程式的推理需求，考慮哪個架構為您的偏好基礎模型（例如 Amazon Bedrock 或 Anthropic Claude 上的 Amazon Nova 模型）提供最佳支援。如需詳細資訊，請參閱 Anthropic 網站上的建置有效的代理程式。
- LLM API 整合 – 根據架構與您偏好的大型語言模型 (LLM) 服務介面（例如 Amazon Bedrock 或 OpenAI) 的整合來評估架構，以進行生產部署。如需詳細資訊，請參閱 文件中的 Strands Agents模型界面。
- 多模態需求 – 對於需要處理文字、影像和語音的自動代理程式，請考慮每個架構的多模態功能。如需詳細資訊，請參閱 LangChain 文件中的多模態。
- 自主工作流程複雜性 – 具有複雜狀態管理的更複雜自主工作流程可能偏好進階狀態機器功能。 的 LangGraph。
- 自主團隊協作 – 需要專業客服人員之間明確角色型自主協作的專案，可以從 的團隊導向架構中受益 CrewAI。
- 自主開發範例 – 偏好自動代理程式對話、非同步模式的團隊可能會偏好 的事件驅動型架構 AutoGen。
- 受管或程式碼型方法 – 希望以最少編碼獲得全受管體驗的組織應考慮 Amazon Bedrock 代理程式。需要更深入自訂的組織可能會偏好 Strands Agents或其他具有專門功能的架構，以更符合特定的自主代理程式需求。
- 自動系統的生產準備程度 – 考慮生產自動代理器的部署選項、監控功能和企業功能。

平台

代理式 AI 平台提供部署、擴展和管理生產級代理系統所需的基礎執行期、協同運作和整合層。架構定義客服人員的建置方式，而通訊協定則控管客服人員的通訊方式。平台提供這些代理程式大規模安全操作、協作和發展的環境。

代理程式平台將模型執行、內容管理、工具整合、可觀測性和控管功能結合到統一的環境中。這些平台可讓組織從實驗轉移到企業規模的部署。

在本節中：

為什麼平台很重要

對於尋求在生產環境中操作自動化系統的組織而言，代理式 AI 平台至關重要。它們提供下列功能：

- 提供用於託管、擴展和協調代理程式的執行時間協調。
- 跨多代理程式工作流程管理狀態、內容和記憶體。
- 提供符合企業標準的安全性、身分和控管控制。
- 透過標準 APIs 或通訊協定與工具生態系統和外部系統整合。
- 在客服人員互動和事件流程中啟用可觀測性和可稽核性。
- 支援跨模型互通性，允許客服人員在單一環境中使用多個基礎模型。

這些功能將個別代理程式轉換為協調的自適應系統，可在企業和法規界限內可靠地運作。

代理式 AI 平台的類型

代理式 AI 平台通常屬於下列一或多個類別：

- 受管代理程式 – 全受管平台提供內建的基礎設施、記憶體和協同運作功能。它們可減少營運開銷並加速生產時間。
- 開放原始碼協同運作 – 開放原始碼代理平台為偏好可自訂環境或內部部署的組織提供彈性和透明度。
- 混合企業 – 混合平台整合了受管和自我託管元件，將雲端受管服務的可擴展性與企業系統的控制相結合。

平台選擇考量事項

選取或設計代理式 AI 平台時，組織應考慮下列事項：

- 整合深度 – 評估平台與現有資料來源、工具和通訊協定的整合能力。
- 可擴展性 – 確保平台可以動態擴展以支援自主工作負載和多代理程式協同合作。
- 安全性與合規 – 根據組織和區域需求評估資料隱私權、加密和控管功能。
- 可擴展性 – 選擇具有模組化架構的平台，以允許隨時間新增新工具、模型或代理程式。
- 可觀測性 – 偏好為客服人員互動提供詳細遙測、可追蹤性和稽核日誌的平台。
- 成本效率 – 考慮無伺服器或以用量為基礎的模型，以最佳化可變工作負載的成本。

Amazon Bedrock 代理程式

Amazon Bedrock Agents 是一項全受管服務，可讓您在應用程式中建置和設定自動代理程式。它可以協調基礎模型、資料來源、軟體應用程式和使用者對話之間的互動。其建立代理程式的簡化方法不需要您佈建容量、管理基礎設施或撰寫自訂程式碼。

Amazon Bedrock 代理程式的主要功能

Amazon Bedrock 代理程式包含下列主要功能：

- 全受管服務 – 完整的基礎設施管理，無需佈建容量或管理基礎系統。如需詳細資訊，請參閱 Amazon Bedrock 文件中的使用 AI 代理器自動化應用程式中的任務。
- API 驅動的開發 – 透過簡單的 API 呼叫，透過指定模型、指示、工具和組態參數來定義和執行代理程式。如需詳細資訊，請參閱 Amazon Bedrock 文件中的手動建立和設定代理程式。
- 動作群組 – 透過使用 API 結構描述建立動作群組，定義您的代理程式可執行的特定動作。如需詳細資訊，請參閱《Amazon Bedrock 文件》中的使用動作群組來定義代理程式要執行的動作。

- 知識庫整合 – 無縫連線至 Amazon Bedrock 知識庫，以使用組織的資料增強客服人員回應。如需詳細資訊，請參閱 Amazon Bedrock 文件中的為具有知識庫的代理程式產生增強回應。
- 進階提示範本 – 透過提示範本自訂客服人員行為，以進行預先處理、協同運作、知識庫回應產生和後製處理。如需詳細資訊，請參閱《Amazon Bedrock 文件》中的使用 Amazon Bedrock 中的進階提示範本增強代理程式的準確性。
- 追蹤和可觀測性 – 使用內建的追蹤功能追蹤代理程式step-by-step推理程序。如需詳細資訊，請參閱 Amazon Bedrock 文件中的使用追蹤追蹤追蹤代理程式step-by-step推理程序。
- 版本控制和別名 – 建立多個版本的代理程式，並透過別名部署它們以進行受控推展。如需詳細資訊，請參閱 Amazon Bedrock 文件中的在您的應用程式中部署和使用 Amazon Bedrock 代理程式。

何時使用 Amazon Bedrock 代理程式

Amazon Bedrock Agents 特別適合自動代理程式案例，包括：

- 希望獲得完全受管體驗的組織，無需管理基礎設施即可建置和部署代理程式
- 需要透過組態而非程式碼快速開發和部署代理程式的專案
- 受益於與其他 Amazon Bedrock 功能緊密整合的使用案例，例如知識庫和護欄
- 沒有內部資源的團隊從頭開始建置代理程式，但需要生產就緒的自動化功能

Amazon Bedrock 代理程式的實作方法

Amazon Bedrock Agents 為業務利益相關者提供以組態為基礎的實作方法。服務可讓組織：

- 透過 AWS 管理主控台 或 API 呼叫定義客服人員，無需撰寫複雜的程式碼。
- 建立動作群組，指定代理程式可執行APIs 和操作。
- 連接知識庫，以提供特定網域的資訊給代理程式。
- 透過視覺化界面測試和反覆執行代理程式行為。

這種受管方法可讓業務團隊快速開發和部署自動代理程式，而無需 AI 模型開發或基礎設施管理的深厚技術專業知識。

Amazon Bedrock 代理程式的實際範例

本AWS 部落格文章所述的財務操作 (FinOps) 解決方案使用 Amazon Bedrock 多代理程式架構來建立 AI 驅動的雲端成本管理助理。經濟實惠的 Amazon Nova 基礎模型為中央 FinOps 主管代理程式將任

務委派給專業代理程式的解決方案提供支援。這些客服人員會使用 擷取和分析 AWS 支出資料， AWS Cost Explorer 並使用 產生節省成本的建議 AWS Trusted Advisor。

系統包括透過 Amazon Cognito 的安全使用者存取、託管在 上的前端 AWS Amplify，以及用於即時分析和預測 AWS Lambda 的動作群組。財務團隊可以詢問自然語言查詢，例如「2025 年 2 月我的成本是多少？」 系統會回應詳細的明細、最佳化建議和預測，全都使用 部署在可擴展的無伺服器架構內 AWS CloudFormation。

Amazon Bedrock AgentCore

Amazon Bedrock AgentCore 是一種代理程式平台，可使用任何架構、模型或通訊協定大規模安全地建置、部署和操作功能強大的代理程式。使用 AgentCore，您可以執行以下操作，無需任何基礎設施管理：

- 更快速地建置代理程式。
- 讓客服人員跨工具和資料採取動作。
- 使用低延遲和延長的執行時間安全地執行代理程式。
- 在生產環境中監控客服人員。

AgentCore 消除了建置專用代理程式基礎設施的無差別繁重工作，可讓您加速代理程式進入生產環境。其服務可以一起使用或獨立使用，並且與任何架構相容，包括 CrewAI、LlamaIndex、 LangGraph 和 Strands Agents。AgentCore 也相容於 Amazon Bedrock 內外提供的任何基礎模型，提供最大的彈性。

AgentCore 由數個關鍵服務組成：

- Amazon Bedrock AgentCore 執行期 – 提供安全、無伺服器、可擴展的環境來託管和執行您的代理程式，而無需管理部署和執行 AI 代理程式或工具所需的任何基礎設施。
- Amazon Bedrock AgentCore 記憶體 – 提供受管記憶體系統，透過保持即時和長期知識，讓客服人員保留互動的內容，以進行更個人化和一致的對話。
- Amazon Bedrock AgentCore Gateway – 簡化為客服人員建立、保護和尋找正確工具的程序。透過 AgentCore Gateway，開發人員可以將 APIs、Lambda 函數和現有服務轉換為與模型內容協定 (MCP) 相容的工具，並將其提供給客服人員。
- Amazon Bedrock AgentCore Identity – 提供安全、可擴展的代理程式身分和存取管理服務，可加速 AI 代理程式開發。使用 AgentCore Identity，您可以將唯一、可驗證的身分指派給客服人員，進而實現精細存取控制，並保護客服人員與企業系統的互動。

- Amazon Bedrock AgentCore 內建工具 – 可讓您使用內建工具來增強開發和測試工作流程。使用這些工具與您的應用程式有效互動，讓 AI 代理器能夠在沙盒環境中安全地撰寫和執行程式碼。使用瀏覽器工具讓 AI 代理器大規模與網站互動。
- Amazon Bedrock AgentCore 可觀測性 – 提供記錄和監控功能，讓您即時了解代理程式的效能和行為，以利偵錯和最佳化。

AgentCore 的主要功能

AgentCore 包含下列主要功能：

- 完全受管且可擴展 – AgentCore 是一種完全受管的服務，這表示 AWS 會處理基礎基礎設施和維護。它也是可擴展的，可讓您自訂和增強代理程式的功能。如需詳細資訊，請參閱 AgentCore 文件中的開始使用 AgentCore 執行期。 AgentCore
- 長期和短期記憶體 – 透過為客服人員配備記憶體系統，以從目前對話和長期知識中回收內容，提供更個人化的相關互動。如需詳細資訊，請參閱 AgentCore 文件中的開始使用 AgentCore 記憶體。 AgentCore
- 簡化工具開發和整合 – 讓您的代理程式能夠透過單一安全端點探索和使用工具。只需幾行程式碼，即可快速將現有的企業資源轉換為客服人員就緒工具，讓開發人員專注於建置獨特的功能。如需詳細資訊，請參閱 AgentCore 文件中的 AgentCore Gateway 入門。 AgentCore
- 安全且可擴展的基礎設施 – AgentCore 為部署和操作代理程式提供安全且可擴展的環境。它包含身分和存取管理、資料加密和網路安全的功能。如需詳細資訊，請參閱 AgentCore 文件中的開始使用 AgentCore Identity。 AgentCore
- 與各種工具整合 – 可讓您將代理程式與各種工具整合，包括程式碼解譯器，以及您可以使用 AgentCore 內建工具建置的瀏覽器工具。如需詳細資訊，請參閱 AgentCore 文件中的開始使用 AgentCore Code Interpreter 和開始使用 AgentCore 瀏覽器。 AgentCore
- 全方位的可觀測性和監控 – 使用全方位工具來追蹤、偵錯和監控其在生產環境中的效能，進而深入了解您的代理程式。視覺化代理程式的整個執行路徑，以稽核其推理並解決失敗。使用即時儀表板和標準化遙測資料來追蹤關鍵操作指標。如需詳細資訊，請參閱 AgentCore 文件中的將可觀測性新增至 Amazon Bedrock AgentCore 資源。

何時使用 AgentCore

AgentCore 特別適合自動代理程式案例，包括：

- 想要透過處理基礎設施、安全性、內建工具、可觀測性和擴展的全受管服務加速開發並降低營運開銷的組織

- 需要彈性的專案，搭配可共同運作或獨立運作的模組化服務，且與任何架構相容，例如 CrewAI或 LangGraph，以及來自任何來源的任何基礎模型
- 需要有狀態的對話客服人員的使用案例，這些客服人員需要維護內容並從過去的互動中學習，以提供個人化和相關的回應
- 透過與各種應用程式、資料來源和 APIs代理程式

AgentCore 的實作方法

AgentCore 專為希望將 AI 代理器從使用開放原始碼或自訂代理程式架構建置的概念驗證移至生產環境的組織而設計。透過 AgentCore，組織可以執行下列動作：

- 在無伺服器基礎設施上安全地部署代理程式，支援任何架構和模型，具有工作階段隔離和內建身分和存取管理，以實現end-to-end安全和合規。使用入門工具組，快速為主要代理程式架構建立 AgentCore 執行期代理程式。
- 透過整合持久性記憶體以保留內容，透過 AgentCore Gateway 簡化工具開發和整合，來增強代理程式。利用內建瀏覽器工具和程式碼解譯器進行進階工作流程。
- 使用採用 Amazon CloudWatch Application Insights 和 技術的可觀測性儀表板，追蹤生產環境中的 AI 代理器OpenTelemetry，追蹤 AgentCore 資源 （執行時間、記憶體、閘道和工具） 的關鍵指標。
- 使用任何代理程式架構和模型提供者，透過全受管、模組化服務、可組合區塊一起或獨立地加速部署和創新。這種靈活性有助於組織更快地從原型遷移到生產環境。

這種受管方法可讓組織快速安全地建置、部署和執行任何規模的企業級 AI 代理程式和多代理程式系統。

AgentCore 的實際範例

AWS 已觀察到拉丁美洲最大的銀行之一已使用 AI/ML 多年，以提供超個人化且安全的數位銀行體驗。銀行使用 AgentCore 來擴展代理式 AI 服務，為客戶提供直覺式互動、增強安全性和更高的自動化能力。根據 CTO，AgentCore 預期會支援其大規模履行客戶承諾的工作。AgentCore 為其開發人員提供工具和彈性來建置和管理代理程式，同時協助確保符合金融法規。

通訊協定

AI 代理器需要標準化的通訊協定，才能與其他代理程式和服務互動。實作代理程式架構的組織在互通性、廠商獨立性以及未來的投資方面面臨重大挑戰。

本節可協助您導覽agent-to-agent程式通訊協定環境，著重於將彈性和互通性最大化的開放標準。（如需agent-to-tool通訊協定的相關資訊，請參閱本指南稍後的工具整合策略。）

本節重點介紹模型內容通訊協定 (MCP)，這是 最初Anthropic在 2024 年開發的開放標準。今天，透過對通訊協定的開發和實作的貢獻， AWS 主動支援 MCP。 AWS 正在與包括 LangGraph、 CrewAI 和 等主要開放原始碼代理程式架構協作LlamaIndex，以在通訊協定上塑造代理程式間通訊的未來。如需詳細資訊，請參閱開啟客服人員互通性通訊協定第 1 部分：MCP 上的客服人員間通訊 (AWS 部落格）。

在本節中：

- 為什麼通訊協定選擇很重要
- Agent-to-agent通訊協定
- 選取代理程式通訊協定
- 代理程式通訊協定的實作策略
- MCP 入門
- ???

為什麼通訊協定選擇很重要

通訊協定選擇從根本上塑造了如何建置和發展 AI 代理器架構。透過選擇支援代理程式架構之間可攜性的通訊協定，您可以靈活地結合不同的代理程式系統和工作流程，以滿足特定需求。

開放式通訊協定可讓您跨多個架構整合客服人員。例如，使用 LangChain快速原型設計和實作生產系統與 Strands Agents，透過常見的通訊協定進行通訊，例如 MCP 或 Agent2Agent (A2A) 通訊協定。這種靈活性可降低對特定 AI 供應商的依賴性、簡化與現有系統的整合，並可讓您隨著時間增強代理程式功能。

精心設計的通訊協定也會建立一致的安全模式，以跨代理程式生態系統進行身分驗證和授權。最重要的是，通訊協定可攜性可保留您在新代理程式架構和功能出現時採用這些架構和功能的自由。選擇開放通訊協定可保護您的代理程式開發投資，同時保持與第三方系統的互通性。

開放通訊協定的優點

實作您自己的擴充功能或建置自訂代理程式系統時，開放式通訊協定會提供令人信服的優勢：

- 文件和透明度 – 通常提供全面的文件和透明實作
- 社群支援 – 存取更廣泛的開發人員社群以進行故障診斷和最佳實務
- 互通性保證 – 更佳保證您的延伸模組可跨不同的實作運作
- 未來相容性 – 降低中斷變更或棄用的風險
- 對開發的影響 – 有機會為通訊協定演變做出貢獻

Agent-to-agent通訊協定

下表提供代理程式通訊協定的概觀，可讓多個代理程式協同合作、委派任務和共用資訊。

通訊協定	適用於	考量
MCP 代理程式間通訊	尋求彈性客服人員協同合作模式的組織	• 所提議模型內容通訊協定 (MCP) 的延伸 AWS ，以其現有的agent-to-agent通訊基礎為基礎 • 啟用與 OAuth 型安全性的無縫代理程式協同合作
A2A 通訊協定	跨平台代理程式生態系統	• 後端為 Google • 較 MCP 採用更嚴格的新標準

在通訊協定選項之間決定

實作agent-to-agent通訊時，請將您的特定通訊需求與適當的通訊協定功能配對。不同的互動模式需要不同的通訊協定功能。下表概述常見的通訊模式，並針對每個案例建議最適合的通訊協定選擇。

模式	Description	理想的通訊協定選擇
簡單請求和回應	客服人員之間的一次性互動	具有無狀態流程的 MCP

狀態對話	持續對話與內容	具有工作階段管理的 MCP
多代理程式協同合作	多個客服人員之間的複雜互動	MCP 代理程式間或 AutoGen
以團隊為基礎的工作流程	具有定義角色的階層式客服人員團隊	MCP 代理程式間CrewAI、 或 AutoGen

除了通訊模式之外，數個技術和組織因素可能會影響您的通訊協定選擇。下表概述可協助您評估哪些通訊協定最符合您的特定實作需求的重要考量。

考量事項	Description	範例
安全模型	身分驗證和授權要求	MCP 中的 OAuth 2.0
部署環境	客服人員將在其中執行和通訊	分散式或單一機器
生態系統相容性	與現有代理程式架構整合	LangChain 或 Strands Agents
可擴展性需求	客服人員互動的預期成長	MCP 的串流功能

選取代理程式通訊協定

對於大多數建置生產代理程式系統的組織，模型內容通訊協定 (MCP) 為agent-to-agent之間的通訊提供了最完整且受支援的基礎。MCP 受益於來自 AWS 和開放原始碼社群的積極開發貢獻。

對於希望有效實作代理式 AI 的組織而言，選擇正確的代理程式通訊協定非常重要。考量事項因組織內容而異。

代理程式通訊協定選擇考量事項

選擇代理式 AI 系統的通訊協定時，組織應考慮下列最佳實務：

- 優先考慮開放標準 – 組織應採用 MCP 等開放通訊協定，以協助確保長期互通性、可擴展性，並降低廠商鎖定的風險。
- 平衡速度和靈活性 – 新創公司和早期採用者可以從支援良好的專屬通訊協定開始，以快速開發，但應該定義開放標準的遷移路徑，隨著系統成熟。
- 實作抽象層 – 企業應實作通訊協定抽象，以簡化遷移、啟用混合採用，以及符合未來需求的整合策略。

- 強調安全性和合規性 – 受監管產業的組織應選擇具有強大身分驗證、加密和稽核功能的通訊協定，以滿足控管和合規要求。
- 評估生態系統成熟度 – 所有組織都應評估每個通訊協定的運作狀態、採用和社群支援，以確保永續性並將技術負債降至最低。
- 參與標準開發 – Organizations 應參與標準機構或開放原始碼社群，以協助塑造通訊協定演變並影響最佳實務。
- 考慮資料主權 – 政府和受監管部門應確保通訊協定選擇符合跨部署區域的資料落地和主權要求。
- 利用受管服務 – 盡可能使用代理程式通訊協定的受管或無伺服器實作，以減少操作複雜性並加速部署。

代理程式通訊協定的實作策略

若要在整個組織中有效實作代理程式通訊協定，請考慮下列策略步驟：

1. 從標準一致性開始 – 盡可能採用已建立的開放式通訊協定。
2. 建立抽象層 – 在您的系統和特定通訊協定之間實作轉接器。
3. 致力於開放標準 – 參與通訊協定開發社群。
4. 監控通訊協定演變 – 隨時掌握新出現的標準和更新。
5. 定期測試互通性 – 驗證您的實作是否仍然相容。

MCP 入門

AWS 透過對通訊協定的開發和實作的貢獻，積極支援模型內容通訊協定 (MCP)。 AWS 正在與包括 LangGraph、 CrewAI和 在內的領導開放原始碼代理程式架構協作LlamaIndex，以在通訊協定上塑造代理程式間通訊的未來。

若要在代理程式架構中實作 MCP，請採取下列動作：

1. 探索 Strands Agents SDK 等架構中的 MCP 實作。
2. 檢閱模型內容通訊協定技術文件。
3. 閱讀客服人員互通性的開放式通訊協定第 1 部分：MCP 上的客服人員間通訊 (AWS 部落格），以了解客服人員互通性。
4. 加入 MCP 社群，以影響通訊協定的演變。

MCP 提供通訊層，可讓客服人員與外部資料和服務互動，也可以用來讓客服人員與其他客服人員互動。通訊協定的可串流 HTTP 傳輸實作為開發人員提供一組完整的互動模式，而不必重新打造輪子。這些模式支援無狀態請求/回應流程，以及具有持久 IDs的狀態工作階段管理。

透過採用 MCP 等開放式通訊協定，您可以將組織定位為建置代理程式系統，以隨著 AI 技術的演進而保持彈性、互通性和適應性。如需agent-to-tool通訊協定實作的相關資訊，請參閱本指南稍後的工具整合策略。

A2A 入門

Agent2Agent (A2A) 通訊協定可讓客服人員透過共用語意層進行分散式協同合作。A2A 不會透過中央協調器路由所有工作，而是允許客服人員彼此探索、公告其功能、交涉任務，以及使用輕量型 JSON 型通訊協定共用內容。每個客服人員都會發佈功能資訊清單。

下列範例顯示簡化的 A2A 功能資訊清單，公告代理程式支援的動作、所需的輸入和操作中繼資料，以啟用探索和任務交涉：

```
{
  "can": ["summarize.text", "extract.keywords"],
  "needs": ["document.input"],
  "meta": { "version": "1.0.3", "latencyMs": 120 }
}
```

此模型可啟用動態功能比對、中任務委派和跨組織協同合作。客服人員可以自行整理任務、形成臨時工作群組，並在新功能進入或退出系統時進行調整。

A2A 支援從簡單的無狀態請求到多步驟交涉工作階段的互動，包括：

- 用於低延遲協同合作的直接peer-to-peer傳訊
- 語意任務交涉，其中客服人員會選取最適合的對等
- 以能力為基礎的探索，實現緊急的人力分配
- 具狀態多步驟互動的工作階段錨定

透過採用開放的客服人員原生通訊協定，例如 A2A，組織可建立模組化、可互通且能夠跨邊界協作的 AI 系統。A2A 可確保客服人員生態系統保持彈性，並可隨著引進新的客服人員、團隊或外部系統而發展，而不需要剛性協同運作層或先前的耦合。

若要在代理程式架構中實作 A2A 通訊協定，請採取下列動作：

1. 檢閱 A2A 通訊協定規格 – 讀取 Agent2Agent (A2A) 通訊協定規格的最新版本，以了解功能資訊清單、交涉流程和客服人員交握的運作方式。
2. 探索 A2A-compatible執行時間 – 評估架構，例如 Strands Agents SDK 或支援 A2A-style功能資訊清單和peer-to-peer交涉的自訂執行時間層。
3. 為您的客服人員實作功能資訊清單 – 定義每個客服人員的 `can`、 `needs`和 `meta` 欄位，以啟用探索、配對和意圖層級協同合作。
4. 實驗 A2A 交涉模式 – 使用 request-offer-accept 迴圈、結構化功能查詢或 gossip 型探索，以了解客服人員應處理任務的人員原因。
5. 在混合基礎設施環境中測試 A2A – 結合 A2A 對等交涉與 AWS 透過 Amazon EventBridge 原生 的事件路由，以評估混合協調模式。
6. 加入 A2A 社群 – 與開放的工作群組互動，以掌握最新的延伸模組、安全建議和跨供應商互通性改進，並有助於通訊協定的開發。

工具

AI 代理器透過與外部工具、APIs和資料來源互動來提供價值，以執行有用的任務。正確的工具整合策略會直接影響代理程式的功能、安全狀態和長期彈性。

本節協助您導覽工具整合環境，專注於將自由和靈活性最大化的開放標準。本節重點介紹工具整合的模型內容通訊協定 (MCP)，並檢閱架構特定的工具和特殊的中繼工具，以增強代理程式工作流程。

在本節中：

工具類別

建置代理程式系統包含三個主要類別的工具。

通訊協定型工具

通訊協定型工具使用標準化通訊協定進行agent-to-tool的通訊：

- MCP 工具 – 使用本機和遠端執行選項，開啟跨架構運作的標準工具。
- OpenAI 函數呼叫 – 專屬於OpenAI模型的專屬工具。
- Anthropic 工具 – 函數OpenAI呼叫 Claude Anthropic 模型專屬工具的變化。

架構原生工具

架構原生工具直接建置在特定代理程式架構中：

- Strands Agents 工具 – 架構特有的輕量、quick-to-implement工具Strands Agents。
- LangChain 工具 – 與LangChain生態系統緊密整合的 Python型工具。
- LlamaIndex 工具 – 已針對 中的資料擷取和處理進行最佳化的工具LlamaIndex。

中繼工具

Meta-tools 可增強代理程式工作流程，無需直接採取外部動作：

- 工作流程工具 – 管理代理程式執行流程、分支邏輯和狀態管理。
- 客服人員圖形工具 – 在複雜的工作流程中協調多個客服人員。
- 記憶體工具 – 提供跨客服人員工作階段的持久性儲存和擷取資訊。
- 反射工具 – 可讓客服人員分析和改善自己的效能。

通訊協定型工具

考慮通訊協定型工具時，模型內容通訊協定 (MCP) 為工具整合提供了最全面且靈活的基礎。如AWS 開放原始碼部落格的客服人員互通性文章所述， AWS 採用 MCP 作為策略通訊協定，積極為其開發做出貢獻。

下表說明 MCP 工具部署的選項。

部署模型	Description	適用於	實作
以本機 stdio 為基礎的	工具在與代理程式相同的程序中執行	開發、測試和簡單工具	快速實作，無需網路額外負荷
本機伺服器傳送事件 (SSE) 型	工具會在本機執行，但透過 HTTP 進行通訊	具有分離疑慮的更複雜本機工具	更好的隔離，但仍低延遲
遠端 HTTP 可串流	在遠端伺服器上執行的工具	生產環境和共用工具	可擴展且集中管理

官方 MCP SDKs可用於建置 MCP 工具：

- Python SDK – 完整通訊協定支援的全面實作
- TypeScript SDK – 適用於 Web 應用程式的 JavaScript/TypeScript 實作
- Java SDK – 企業應用程式的 Java 實作

這些SDKs提供以您慣用語言建立 MCP 相容工具的建置區塊，以及一致的通訊協定規格實作。

此外， AWS 已在 Strands Agents SDK 中實作 MCP。Strands Agents 開發套件提供直接的方式來建立和使用與 MCP 相容的工具。Strands Agents GitHub 儲存庫提供完整的文件。對於更簡單的使用案例或在Strands Agents框架之外工作時，官方 MCP SDKs以多種語言直接實作通訊協定。

MCP 工具的安全功能

MCP 工具的安全功能包括下列項目：

- OAuth 2.0/2.1 身分驗證 – 產業標準身分驗證
- 許可範圍 – 工具的精細存取控制
- 工具功能探索 – 可用工具的動態探索
- 結構化錯誤處理 – 一致的錯誤模式

MCP 工具入門

若要實作 MCP 進行工具整合，請採取下列動作：

1. 探索適用於生產就緒 MCP 實作的 Strands Agents 開發套件。
2. 檢閱 MCP 技術文件以了解核心概念。
3. 使用此AWS 開放原始碼部落格文章中所述的實際範例。
4. 從簡單的本機工具開始，然後再繼續進行遠端工具。
5. 加入 MCP 社群，以影響通訊協定的演變。

探索 AgentCore Gateway

Amazon Bedrock AgentCore Gateway 為開發人員提供簡單且安全的方式來大規模建置、部署、探索和連線至 MCP 工具和其他目標端點。透過 AgentCore Gateway，開發人員可以將 APIs、 AWS Lambda 函數和現有服務轉換為與 MCP 相容的工具。然後，只要幾行程式碼，他們就可以透過 AgentCore Gateway 端點將這些工具提供給客服人員。AgentCore Gateway 支援 OpenAPI、 Smithy 和 Lambda 作為輸入類型，並且是唯一在全受管服務中提供完整輸入身分驗證和輸出身分驗證的解決方案。

架構原生工具

雖然模型內容通訊協定 (MCP) 提供最靈活的基礎，但架構原生工具可為特定使用案例提供優勢。

Strands Agents 開發套件提供以 Python為基礎的工具，其輕量型設計為特徵，只需最少的負荷即可進行簡單的操作。它們可以快速實作，並允許開發人員使用幾行程式碼來建立工具。此外，它們已緊密整合，可在Strands Agents架構內順暢運作。

下列範例示範如何使用 建立簡單的天氣工具Strands Agents。開發人員可以將Python函數快速轉換為客服人員可存取的工具，並將程式碼額外負荷降至最低，並自動從函數的 Docstring 產生適當的文件。

```
#Example of a simple Strands native tool

@tool

def weather(location: str) -> str:

"""Get the current weather for a location""" #

Implementation here

return f"The weather in {location} is sunny."
```

對於快速原型或簡單的使用案例，架構原生工具可以加速開發。不過，對於生產系統，MCP 工具提供比架構原生工具更好的互通性和未來彈性。

下表提供其他架構特定工具的概觀。

架構	工具類型	優點	考量
AutoGen	函數定義	強大的多代理程式支援	Microsoft 生態系統
LangChain	Python 類別	預先建置工具的大型生態系統	架構鎖定
LlamaIndex	Python 函式	針對資料操作進行最佳化	限制為 LlamaIndex

中繼工具

中繼工具不會直接與外部系統互動。反之，他們會實作代理程式模式來增強代理程式功能。本節討論工作流程、代理程式圖形和記憶體中繼工具。

工作流程中繼工具

工作流程中繼工具可管理代理程式執行的流程：

- 狀態管理 – 維護多個客服人員互動的內容
- 分支邏輯 – 啟用條件式執行路徑
- 重試機制 – 使用複雜的重試策略處理失敗

具有工作流程中繼工具的範例架構包括 LangGraph和 Strands Agents工作流程功能。

代理程式圖形中繼工具

代理程式圖形中繼工具會協調多個同時運作的代理程式：

- 任務委派 – 將子任務指派給專業客服人員
- 結果彙總 – 合併來自多個客服人員的輸出
- 衝突解決 – 解決客服人員之間的分歧

像 AutoGen和 之類的架構CrewAI專門處理客服人員圖形協調。

記憶體中繼工具

記憶體中繼工具提供持久性儲存和擷取：

- 對話歷史記錄 – 維護跨工作階段的內容
- 知識庫 – 存放和擷取特定網域的資訊
- 向量存放區 – 啟用語意搜尋功能

MCP 的資源系統提供標準化的方式來實作跨不同代理程式架構運作的記憶體中繼工具。

工具整合策略

您選擇的工具整合策略會直接影響代理程式可以完成的內容，以及系統可以輕鬆發展的程度。優先考慮開放式通訊協定，例如模型內容通訊協定 (MCP)，同時策略性地使用架構原生工具和中繼工具。如此一來，您就能建置工具生態系統，隨著 AI 技術發展而保持彈性和強大。

下列工具整合的策略方法可最大限度地提高靈活性，同時滿足組織的立即需求：

1. 採用 MCP 作為您的基礎 – MCP 提供標準化方法，將代理程式連接到具有強大安全功能的工具。從 MCP 開始，做為下列項目的主要工具通訊協定：
   - 將用於多個代理程式實作的策略工具。
   - 需要強大身分驗證和授權的安全敏感工具。
   - 在生產環境中需要遠端執行的工具。
2. 適當時使用架構原生工具 – 考慮架構原生工具：
   - 初始開發期間的快速原型設計。
   - 具有最低安全需求的簡單非關鍵工具。
   - 利用唯一功能的架構特定功能。
3. 實作複雜工作流程的中繼工具 – 新增中繼工具以增強您的代理程式架構：
   - 從基本工作流程模式開始簡單。
   - 隨著使用案例成熟，增加複雜性。
   - 標準化代理程式和中繼工具之間的界面。
4. 規劃進化 – 以未來靈活性為重心建置：
   - 獨立於實作的文件工具界面。
   - 在客服人員和工具之間建立抽象層。
   - 建立從專屬通訊協定到開放通訊協定的遷移路徑。

工具整合的安全最佳實務

工具整合會直接影響您的安全狀態。本節概述要為您的組織考慮的最佳實務。

身分驗證和授權

利用下列強大的存取控制：

- 使用 OAuth 2.0/2.1 – 實作遠端工具的產業標準身分驗證。
- 實作最低權限 – 僅授予工具所需的許可。
- 輪換登入資料 – 定期更新 API 金鑰和存取權杖。

資料保護

為了協助保護資料，請採用下列措施：

- 驗證輸入和輸出 – 為所有工具互動實作結構描述驗證。
- 加密敏感資料 – 對所有遠端工具通訊使用 TLS。
- 實作資料最小化 – 僅將必要資訊傳遞至工具。

監控和稽核

使用以下機制來維持可見性和控制：

- 記錄所有工具叫用 – 維護完整的稽核線索。
- 監控異常 – 偵測不尋常的工具使用模式。
- 實作速率限制 – 透過過多的工具呼叫防止濫用。

模型內容通訊協定 (MCP) 安全模型可全面解決這些問題。如需詳細資訊，請參閱 MCP 文件中的安全考量。

結論

代理式 AI 的前景持續快速發展，為組織提供強大的新方法來建置智慧、自動化系統。本指南探索了成功實作的三個基本元件：提供基礎的架構、提供環境的平台、啟用通訊的通訊協定，以及擴展功能的工具。

隨著架構的成熟，您可以預期互通性、模型內容通訊協定 (MCP) 等通訊協定的標準化，以及自動化代理程式更複雜的協同運作功能。目前使用這些架構建立專業知識的組織，將能夠妥善地建置越來越自主的智慧型代理程式，以提供顯著的商業價值。

平台提供代理系統運作所在的執行、控管和生命週期環境。它們處理諸如身分、安全界限、可觀測性、記憶體管理、工作階段基礎以及與工具和資料的安全互動等問題。在 AWS 環境中，受管代理程式執行時間和協同運作服務等平台可讓組織大規模部署、監控、發展和管理自動代理程式和代理程式系統。平台會橋接基礎架構與實際操作需求。

客服人員通訊協定的選擇代表策略決策，在即時開發需求與長期彈性和互通性之間取得平衡。透過優先考慮開放式通訊協定並建立適當的抽象層，組織可以建置代理程式系統，以適應不斷發展的技術，同時滿足目前的業務需求。

對於大多數組織而言，MCP 代表堅實的基礎，因為它具有開放的標準、不斷成長的生態系統、agent-to-agent通訊模式的支援，以及工具整合功能。 AWS 採用 MCP 和 Agent2Agent (A2A) 作為策略通訊協定，積極為其開發做出貢獻，並在 Strands Agents SDK 等服務之間實作它們。透過使用 MCP 或 A2A 搭配適當的架構原生工具和中繼工具，您可以建置代理程式系統，以提供立即值，同時保持適應未來的創新。

Resources

使用下列 AWS 和其他與自動代理程式開發相關的資源。

AWS 部落格

- Amazon Bedrock AgentCore 記憶體：建置內容感知代理程式
- 使用 Amazon Bedrock Agents 建置強大生成式 AI 應用程式的最佳實務 – 第 1 部分
- 使用 Amazon Bedrock Agents 建置強大生成式 AI 應用程式的最佳實務 – 第 2 部分
- 使用 LlamaIndex和 Amazon Bedrock 建置強大的 RAG 管道
- 使用 Amazon Bedrock AgentCore 可觀測性建置值得信賴的 AI 代理器
- 使用 Amazon Bedrock LlamaIndex和 RAGAS 評估 RAG 回應
- Amazon Bedrock AgentCore 程式碼解譯器簡介
- Amazon Bedrock AgentCore Gateway 簡介：轉換企業 AI 代理程式工具開發
- Amazon Bedrock AgentCore Identity：大規模保護代理式 AI
- 簡介 Strands Agents，開放原始碼 AI 代理程式 SDK
- 客服人員互通性的開放式通訊協定第 1 部分：MCP 上的客服人員間通訊
- 在 Amazon Bedrock AgentCore 執行期上安全地啟動和擴展您的代理程式和工具
- AWS Transform for .NET 是第一個大規模現代化 .NET 應用程式的代理式 AI 服務
- AWS 每週總和： Strands Agents

AWS 方案指引

- 在 上操作代理式 AI AWS
- 上的代理式 AI 基礎 AWS
- 上的客服人員 AI 模式和工作流程 AWS
- 在 上建置代理式 AI 的無伺服器架構 AWS
- 在 上建置代理式 AI 的多租戶架構 AWS
- 上的代理式 AI 安全性 AWS
- 在 上擷取增強產生選項和架構 AWS

AWS 資源

- Amazon Bedrock 文件
- Amazon Bedrock AgentCore 文件
- Amazon Bedrock AgentCore Starter Toolkit (GitHub 儲存庫)
- Amazon Nova 文件
- AWS MCP 伺服器 (GitHub 儲存庫)

其他資源

- AutoGen 文件 (Microsoft)
- 建置有效的代理程式 (Anthropic)
- CrewAI GitHub 儲存庫
- LangChain 文件
- LangGraph 平台
- LlamaIndex 文件
- 模型內容通訊協定文件
- Strands Agents 文件
- Strands Agents 工具概觀
- Strands Agents 快速入門指南

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
新增章節	新增的平台區段	2026 年 1 月 16 日
初次出版	—	2025 年 7 月 14 日

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。