aws

上的客服人員 AI 模式和工作流程 AWS

# AWS 方案指引

# AWS 方案指引: 上的客服人員 AI 模式和工作流程 AWS

# Table of Contents

# 上的客服人員 AI 模式和工作流程 AWS

Aaron Sempf 和 Andrew Hooker，Amazon Web Services

2025 年 7 月 (文件歷史記錄)

組織正在採用大型語言模型 (LLMs) 和軟體代理程式，以使用稱為代理程式模式的新架構學科來解決動態多網域問題。代理程式模式是基礎藍圖和模組化建構，用於在許多內容中設計和協調目標導向 AI 代理程式。

## 目標對象

本指南適用於希望建置超越靜態邏輯、符號邏輯和確定性自動化之智慧型應用程式的架構師、開發人員和產品領導者。

## 目標

本指南提供 AI 代理器系統的設計架構和實作方法，可自動操作，同時保持可控制性並與目標保持一致。它將事件驅動的架構模式與各種代理程式替代方案連線，示範如何使用雲端原生架構建置生產級代理程式系統。本指南會討論下列主題：

- 客服人員模式 – 客服人員模式是可重複使用的設計範本，可描述個別客服人員的結構和行為。這包括推理代理程式、擷取授權代理程式、編碼代理程式、語音界面、工作流程協調程式和協作多代理程式系統。每個模式都說明客服人員如何感知、推理、行動和學習、進行映射 AWS 服務。
- LLM 工作流程 – 工作流程著重於客服人員如何使用 LLMs進行推理。他們探索提示策略和規劃機制，並概述如何使用 LLMs產生文字，以及在代理程式迴圈中推動結構化、可解譯和可靠的行為。
- 代理程式工作流程模式 – 工作流程模式描述多個代理程式、工具和環境如何互動以形成自主系統。這包括任務協同運作、子代理程式委派、事件型協調、可觀測性和控制的模式。這些層面可提升可擴展、可編譯且可稽核的 AI 架構。

## 關於此內容系列

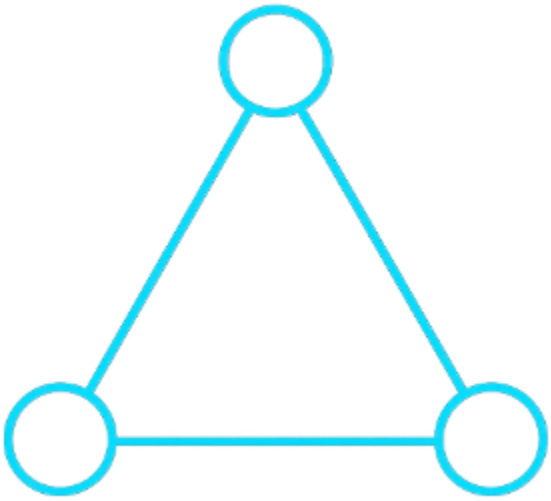
本指南是代理式 AI 相關系列的一部分 AWS。如需詳細資訊和檢視此系列中的其他指南，請參閱 AWS 規範性指南網站上的客服人員 AI。

# 客服人員模式

代理程式模式是可重複使用、可組合的建置區塊，可針對特定網域、使用案例和複雜程度量身打造。不過，代理系統與傳統應用程式不同。所有 AI 代理器設計的核心都是以下列三個基本原則為基礎的概念性模型：

- 非同步 – 代理程式在鬆耦合且事件豐富的環境中操作
- 自主性 - 代理程式獨立運作，無需人工或外部控制
- 代理 – 代理程式代表使用者或系統，以特定目標為目的行事

下圖中的三角形代表軟體代理程式的核心建置區塊：感知、原因和動作。這可讓代理系統在其環境中觀察、做出決策和採取行動。



根據設計，代理程式模式提供模組化的設計語言，用於建置 AI 系統，這表示它們可以存取、操作、可擴展和生產就緒。設計這些系統需要仔細注意以下三個相互關聯的維度，本指南稍後會進一步討論這些維度。

本節內容

- 語音和語音客服人員
- 工作流程協同運作代理程式
- 記憶體增強型代理程式
- 模擬和測試平台代理程式
- 觀察者和監控代理程式
- 多代理程式協同合作

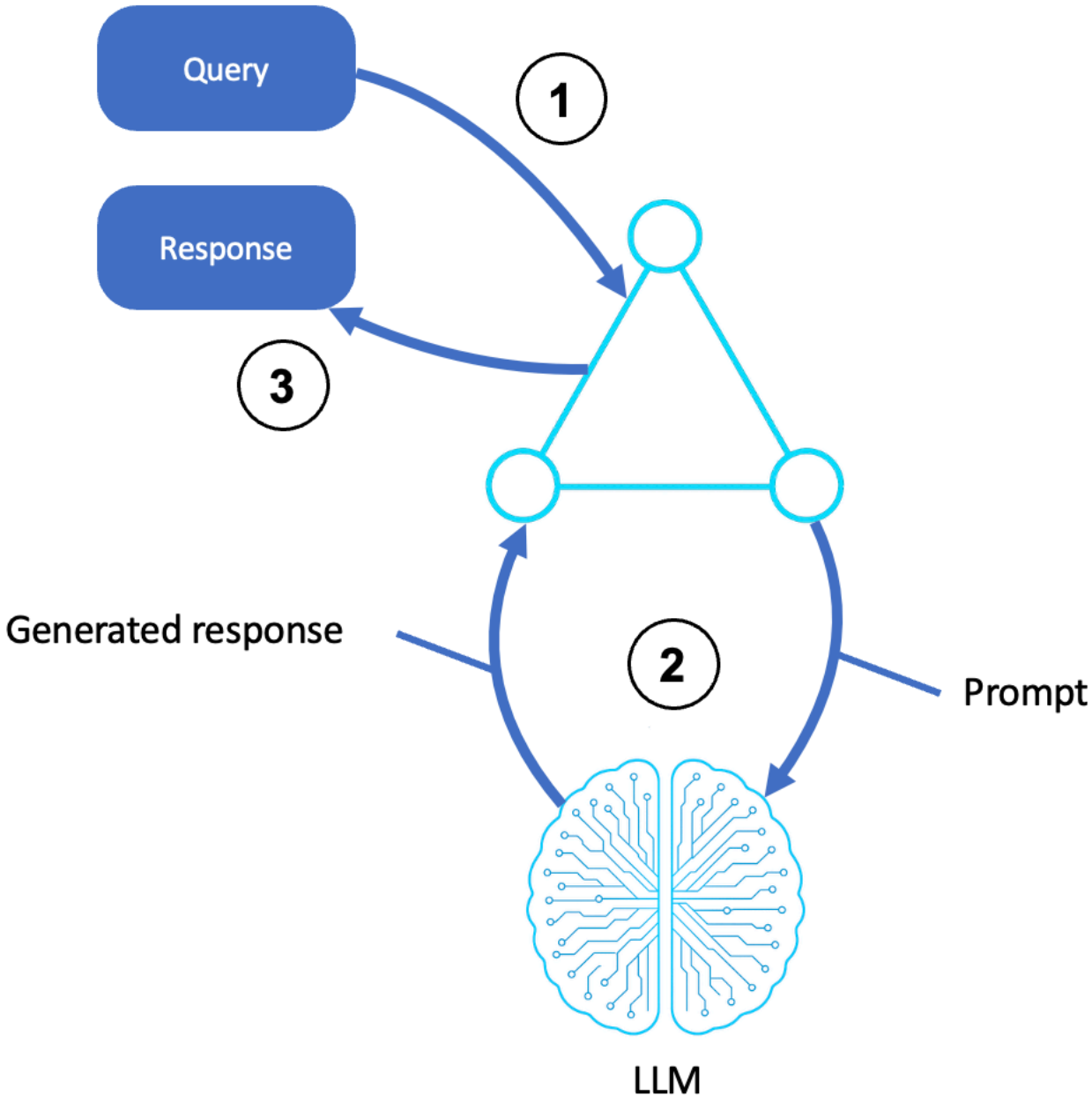
# 基本推理代理程式

基本推理代理程式是最簡單的代理式 AI 形式，可執行邏輯推論或決策以回應查詢。它接受來自使用者或系統的輸入，並使用結構化提示處理查詢和產生回應。

此模式適用於需要根據指定內容進行單一步驟推理、分類或摘要的任務。它不使用記憶體、工具或狀態管理，這使得它在大型工作流程中無狀態、輕量且高度組合。

## Architecture

下圖顯示基本推理代理程式的流程：




## 說明

1. 接收輸入
   - 使用者、系統或上游代理程式提交查詢或指示。
   - 輸入會移交給代理程式 shell 或協同運作層。
   - 此步驟包含任何預先處理、提示範本和目標識別。
2. 叫用 LLM

- 代理程式會將查詢轉換為結構化提示，並將其傳送至 LLM （例如，透過 Amazon Bedrock)。
- LLM 會根據使用預先訓練知識和內容的提示產生回應。
- 產生的輸出可能包括推理步驟 chain-of-thought)、最終答案或排名選項。

3. 傳回回應
   - 產生的輸出會轉送至客服人員的界面。
   - 這可能包括格式化、後製處理或 API 回應。

## 功能

- 支援自然語言或結構化輸入
- 使用提示詞工程來引導行為
- 無狀態且可擴展
- 可以嵌入 UI、CLI、APIs 和管道

## 限制

- 沒有記憶體或歷史意識
- 與外部工具或資料來源沒有互動
- 僅限於 LLM 在推論時知道的內容

## 常用案例

- 對話式問題和答案
- 政策說明和摘要
- 制定決策的指引
- 輕量且自動化的聊天機器人流程
- 分類、標記和評分

## 實作指引

您可以使用下列工具和服務來建立基本推理代理程式：

- Amazon Bedrock for LLM 調用 (Anthropic、AI21、Meta)

- Amazon API Gateway 或 AWS Lambda 將其公開為無狀態微服務
- 儲存在參數存放區 AWS Secrets Manager或做為程式碼的提示範本

## Summary

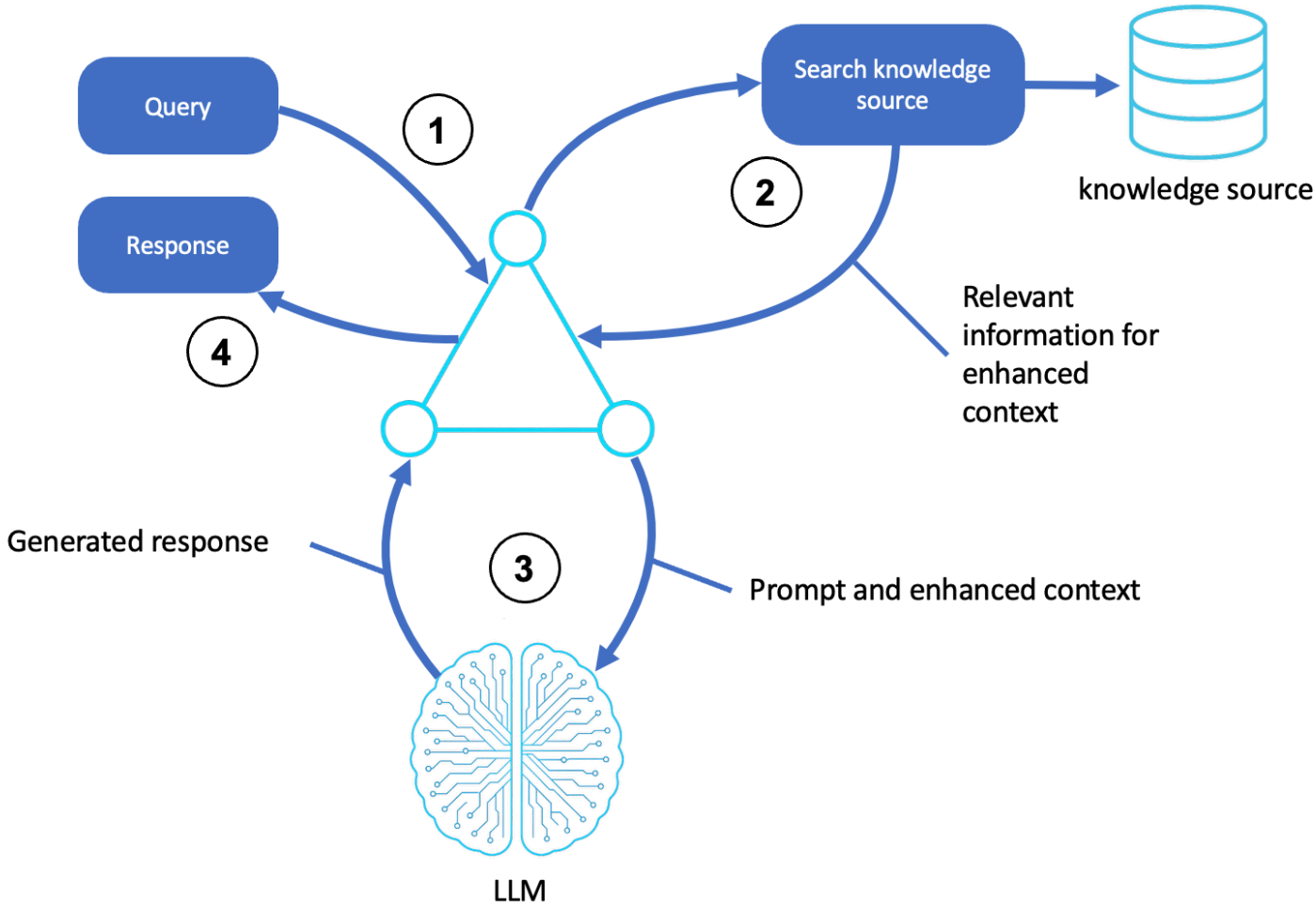
基本推理代理程式的基礎在於其簡單結構。它具有核心功能，可將目標轉換為導致智慧輸出的推理路徑。此模式通常是進階模式的起點，例如使用擷取擴增產生 (RAG) 的工具型代理程式和代理程式。它也是大型工作流程中可靠且模組化的元件。

### 代理程式 RAG

Retrieval-augmented Generation (RAG) 是一種將資訊擷取與文字產生結合的技術，以建立準確且情境式的回應。RAG 可讓客服人員先擷取相關的外部資訊，再與 LLM 互動。它將代理程式的決策以up-to-date、事實或特定網域資訊為基礎，藉此擴展代理程式的有效記憶體和推理準確性。與僅依賴預先訓練權重的無LLMs 相反，RAG 具有外部知識搜尋層，可動態增強具有內容的提示。

## Architecture

RAG 模式的邏輯如下圖所示：




## 說明

1. 接收查詢
   - 使用者或上游系統向客服人員提交查詢或目標。
   - 代理程式 shell 接受請求並將其格式化為推理提示。
2. 搜尋外部來源
   - 代理程式會從查詢中識別概念和意圖。
   - 它會使用語意搜尋或關鍵字比對來查詢知識來源，例如向量存放區、資料庫或文件索引。
   - 擷取最相關的段落、文件或實體，以供下一個步驟使用。
3. 產生內容式回應
   - 代理程式使用擷取的資訊增強提示，形成 LLM 的內容增強輸入。
   - LLM 使用生成推理（例如chain-of-thought或反射鏈）處理任何輸入，以產生準確的回應。
4. 傳回最終輸出

- 代理程式會將其包裝在任何通訊標頭或必要的格式中，然後傳回給使用者或呼叫系統，以準備輸出。
- （選用）擷取的文件和 LLM 輸出可能會記錄、計分並存放在記憶體中，以供未來查詢使用。

## 功能

- 即使在長尾或企業特定網域中也是事實基礎的輸出
- 記憶體擴充功能，無需微調模型
- 根據每個查詢和使用者狀態的動態內容
- 與向量資料庫、語意索引和中繼資料篩選完全相容

## 常用案例

- 企業知識助理
- 法規合規機器人
- 客戶支援 Copilot
- 搜尋增強型聊天機器人
- 開發人員文件代理程式

## 實作指引

使用下列工具和服務來建立使用 RAG 的代理程式：

- Amazon Bedrock for LLM 調用
- 適用於文件或結構化資料搜尋的 Amazon Kendra、OpenSearch 或 Amazon Aurora
- 文件儲存的 Amazon Simple Storage Service (Amazon S3)
- AWS Lambda 協調搜尋、提示和 LLM 推論
- 知識型整合與代理程式（使用記憶體外掛程式、語意擷取器或 Amazon Bedrock)

## Summary

代理程式 RAG 將靜態模型推理連接到動態、真實世界的智慧。它讓客服人員能夠查詢他們不知道的內容、從擷取的知識合成答案，並產生高可稽核的可靠回應。

RAG 模式是建置智慧型代理程式的基礎，無需重新訓練即可擴展知識存取。它通常是涉及工具使用、規劃和長期記憶體的更複雜協同運作模式的前綴。

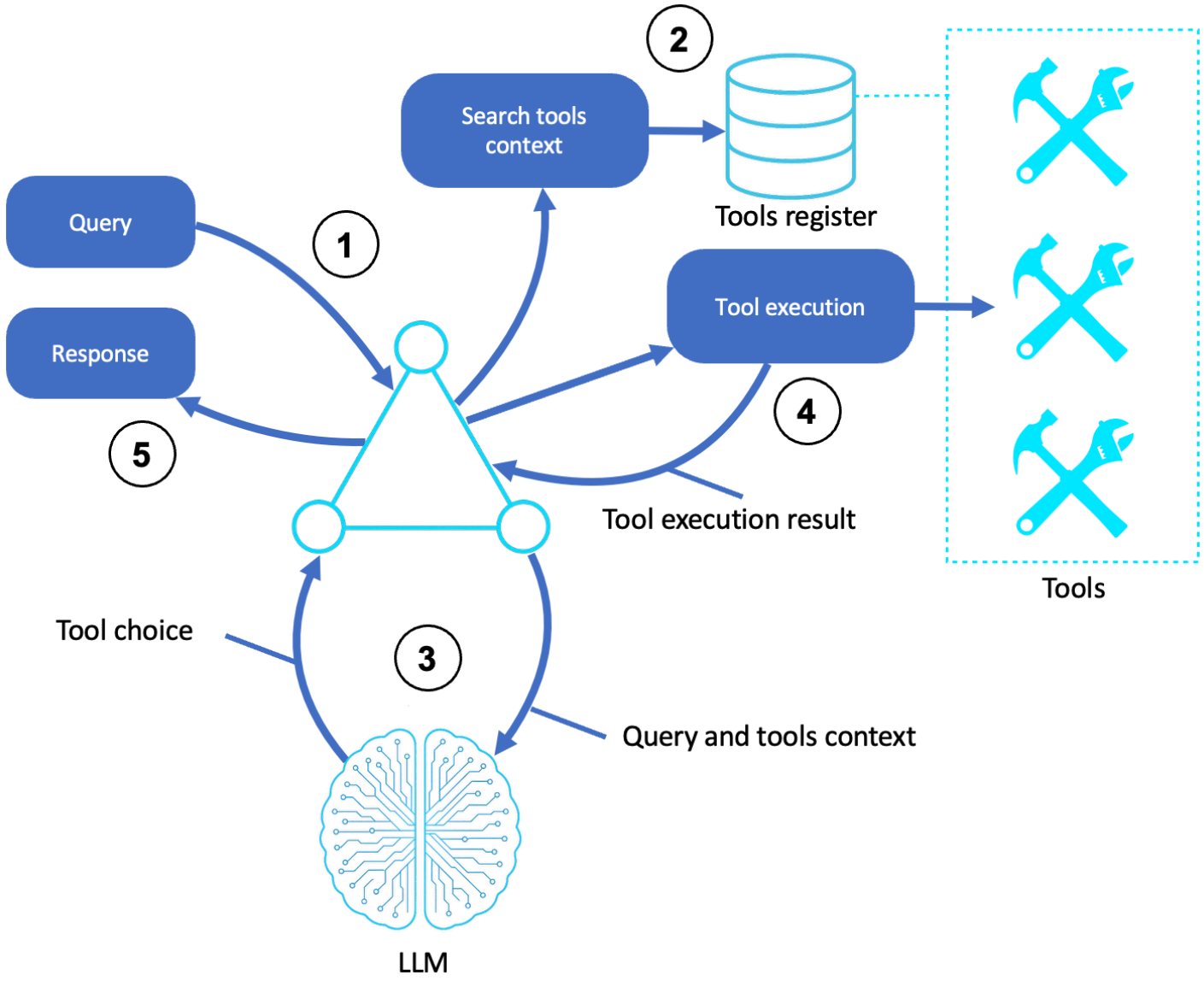
# 用於呼叫 函數的工具型代理程式

工具型代理程式透過叫用外部函數或 APIs 來完成超出僅語言推理的任務，來擴展推理代理程式的功能。此模式使用 LLM 來決定要使用哪個工具，然後產生呼叫引數，並將工具的輸出併入其推理迴圈。

此模式可讓客服人員採取行動，而不只是提供回應。工具界面代表任何可呼叫的功能，從算術計算和資料庫查詢到外部 APIs 和雲端服務。

## Architecture

下圖顯示用於呼叫 函數的工具型代理程式：




## 說明

1. 接收查詢
   - 客服人員會收到使用者或呼叫系統的自然語言查詢或任務。
2. 搜尋工具
   - 代理程式使用內部中繼資料或工具登錄檔來搜尋可用的工具、結構描述和相關功能。
3. 選取和叫用工具
   - LLM 會在提示中接收查詢和工具中繼資料（例如，函數名稱、輸入類型和描述）。
   - 它選擇最相關的工具、建構輸入引數，並傳回結構化函數呼叫。
4. 執行所選的工具

- 代理程式 shell 或工具執行器會執行選取的函數，並傳回結果（例如 API 輸出、資料庫值或運算）。

5. 傳回回應
   - LLM 會將結果直接傳遞給代理程式，或做為更新提示的一部分。然後，它會傳回自然語言結果。

## 功能

- 根據任務內容選擇動態工具
- 結構描述型提示 (OpenAPI、JSON 結構描述、 AWS 函數界面）
- 結果解譯並將輸出鏈結到推理中
- 無狀態或工作階段感知操作

## 常用案例

- 具有外部資料存取的虛擬助理
- 財務計算器和估算器
- 以 API 為基礎的知識工作者
- 叫用的 LLMs AWS Lambda、Amazon SageMaker 端點和 SaaS 服務

## 實作指引

使用下列項目建立工具型客服人員來呼叫 函數：

- 支援函式呼叫的 Amazon Bedrock (Anthropic Claude)
- AWS Lambda 做為工具執行後端
- Amazon API Gateway 或 AWS Step Functions 用於工具協同運作
- 適用於內容感知工具中繼資料的 Amazon DynamoDB 或 Amazon Relational Database Service (Amazon RDS)
- Amazon EventBridge 管道或 AWS Step Functions 映射狀態以路由輸出

## Summary

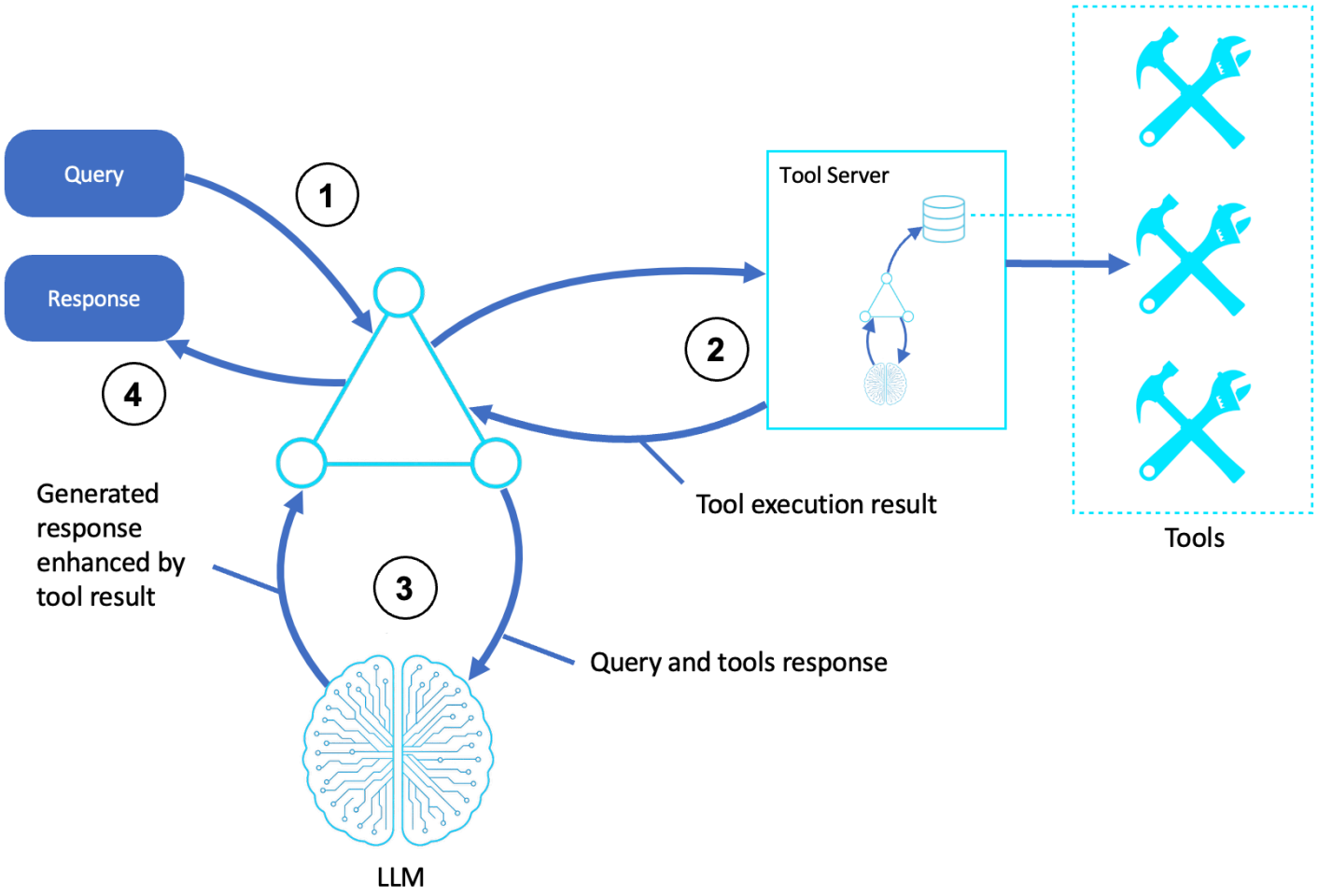
工具型函數呼叫代理程式代表從了解語言到執行動作的轉變。這些客服人員會叫用動態的內容感知工具，同時維護 LLM 推理，將被動助理轉換為完成任務、存取服務和整合業務操作的系統。此模式是代理式 AI 在企業設定中的重要元件，特別是在與宣告式結構描述、授權架構和多代理程式系統結合時。

# 伺服器的工具型代理程式

伺服器的工具型代理程式透過將工具執行委派給具有工具、指令碼和複合代理程式專用執行時間環境的外部伺服器，來增強函數呼叫代理程式。與代理程式迴圈選取和調用的內嵌函數呼叫不同，伺服器型代理程式會將邏輯和執行管道外包給其他代理程式或系統。這提供了進階功能，例如多工具鏈結、隔離執行和專業推理。工具伺服器非常適合複雜、具狀態或資源密集型的動作，其中工具本身可能涉及不同的 AI 模型、商業規則或環境。

## Architecture

以下是伺服器的工具型代理程式模式：




## 說明

1. 接收查詢
   - 使用者或系統向客服人員 shell 提交請求。
   - 代理程式會解譯查詢，並準備將其分派至工具伺服器。
2. 執行工具伺服器程序
   - 代理程式會將任務連同結構化參數傳送至工具伺服器。
   - 工具伺服器接著可能會：
     - 在專用運算系統中執行指令碼或邏輯（例如 AWS Lambda，容器或 Amazon SageMaker)
     - 使用自己的子代理程式搭配 LLM 推理來選取和執行工具
     - 管理相依性、重試或多步驟執行流程
     - 當任務完成時，將結果輸出至主要代理程式
3. 搭配工具輸出使用 LLM 推理

- 代理程式會叫用 LLM，傳遞原始查詢和工具伺服器結果做為提示的一部分。
- LLM 會合成包含新取得資訊的回應。

4. 傳回回應
   - 客服人員會傳回自然語言或結構化回應給使用者或呼叫系統。
   - （選用）結果可能會存放在記憶體或稽核日誌中。

## 功能

- 在主要代理程式執行迴圈之外叫用工具
- 工具執行可能涉及 LLM 呼叫、邏輯鏈或子代理程式
- 代理程式充當控制器或分派程式，而不只是工具包裝函式
- 啟用可編譯性、可擴展性和邏輯隔離

## 常用案例

- 協調模型鏈（例如，透過結合 LLM、視覺和程式碼）
- AI 驅動的自動化管道
- 具有指令碼執行器的 DevOps 助理代理程式
- 複雜的財務運算、模擬或最佳化代理程式
- 多模式工具（例如，透過結合音訊、文件和動作）

## 實作指引

您可以使用下列方式建置此模式 AWS 服務：

- Amazon Bedrock（代理程式主機和 LLM 推論）
- AWS Lambda、Amazon ECS AWS Fargate或 Amazon SageMaker 端點作為工具伺服器執行時間
- Amazon API Gateway 或 AWS App Runner 公開工具伺服器 APIs
- 用於解耦agent-to-toolAmazon EventBridge
- AWS Step Functions 或 AWS AppFabric 用於在工具伺服器上編寫多代理程式邏輯

## Summary

使用伺服器的工具型代理程式具有高度模組化和可擴展性。它們會將決策邏輯與執行分離，這可讓主要代理程式保持輕量，同時將複雜或敏感的動作卸載至其他系統。這對企業級代理式 AI 很重要，尤其是在需要控管、可觀測性、隔離、動態合成或其任何組合的環境中。

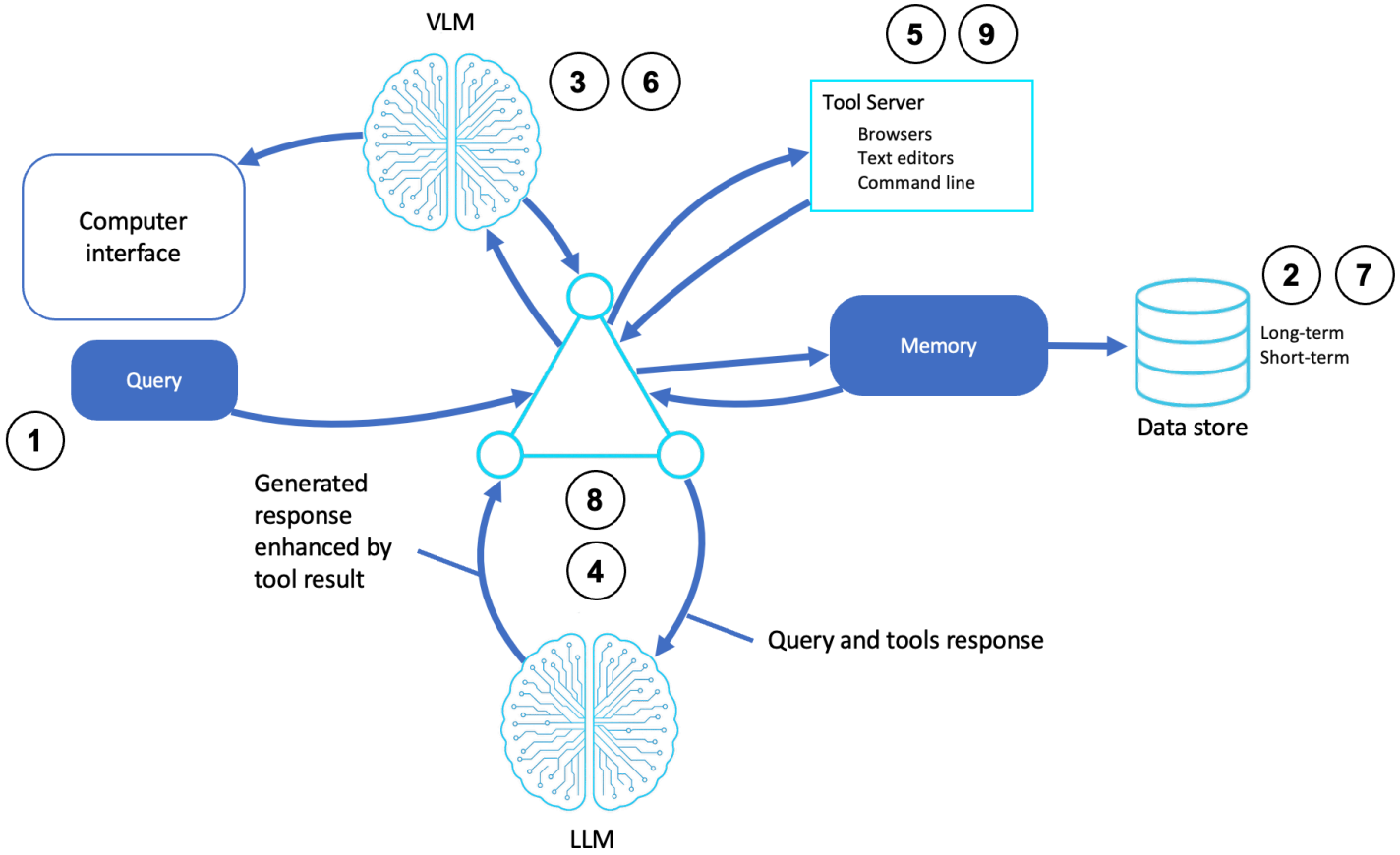
# 電腦使用代理程式

電腦使用代理程式可以模擬或控制數位環境，例如瀏覽器、終端機、檔案系統和應用程式。這些代理程式會解譯使用者意圖、與視覺和文字界面互動，並透過結合 LLM 推理、視覺語言模型 (VLMs) 和執行命令或模擬輸入事件的工具伺服器來執行目標導向動作。

此模式對於實際 AI 自動化很重要，其中代理程式不僅做為助理，也做為代理程式，通常透過使用相同的工具和環境來執行與人類相同的動作。

## Architecture

下圖顯示電腦使用代理程式模式：



## 說明

1. 接收查詢
   - 任務或請求是透過 UI、API 或自然語言界面提供。
2. 存取記憶體
   - 代理程式會擷取短期和長期記憶體，以叫用過去的命令、目標和系統狀態。
3. 分析視覺化內容
   - VLM 會觀察電腦畫面、系統狀態或 UI 元素，以了解特定內容並識別可採取動作的項目。
4. 透過 LLM 的原因
   - LLM 結合了查詢、記憶體狀態、工具和伺服器回應，以判斷下一個動作。
5. 與工具伺服器互動
   - 代理程式會叫用託管在伺服器上的工具，其中可能包括下列項目：
     - 瀏覽器（例如無頭 Chrome) 和 shell 環境
     - 文字和程式碼編輯器
     - 自訂指令碼界面
6. 更新視覺化輸入
   - 如果需要系統 UI 變更或進一步觀察，VPM 可能會重新分析畫面狀態或文字緩衝區。
7. 更新記憶體
   - 新的洞見、系統狀態或使用者意見回饋會寫入短期和長期記憶體。
8. 制定最終決策和說明
   - LLM 會根據查詢和工具輸出合成結果或建議動作。
9. 傳回回應
   - 代理程式會將結果傳回至界面（例如，已完成的任務、確認或產生的內容）。

## 功能

- 具有視覺和文字輸入的多模態推理
- 透過模擬或 API 驅動輸入控制應用程式
- 持久性狀態的記憶體管理
- 序列執行的自主性（多步驟流程）

## 常用案例

- 在 IDEs AI 開發人員
- 用於重複數位工作流程的電腦使用代理程式
- 用於軟體測試和品質保證的模擬使用者
- 透過語音或高階指示導覽 UIs 的存取代理程式
- 透過推理增強的智慧機器人程序自動化 (RPA)

## 實作指引

- 您可以使用下列方式建置此模式 AWS 服務：
- 適用於 LLM 型規劃和推理的 Amazon Bedrock
- Amazon Elastic Compute Cloud (Amazon EC2) AWS Lambda或 Amazon SageMaker 筆記本，以使用模擬 UI 環境執行工具伺服器
- 適用於記憶體持久性的 Amazon Simple Storage Service (Amazon S3) 或 Amazon DynamoDB
- 在混合式案例中用於 UI 影像分析的 Amazon Rekognition （或自訂模型）
- Amazon CloudWatch Logs 或 AWS X-Ray 可觀測性和稽核線索

## Summary

電腦使用代理程式充當自主數位運算子，彌補人類電腦互動和 AI 驅動動作之間的差距。透過整合記憶體、工具協同運作和 VLMs，這些代理程式可以適應性地與專為人類設計的系統互動、執行動作、更新檔案、導覽功能表和產生回應。

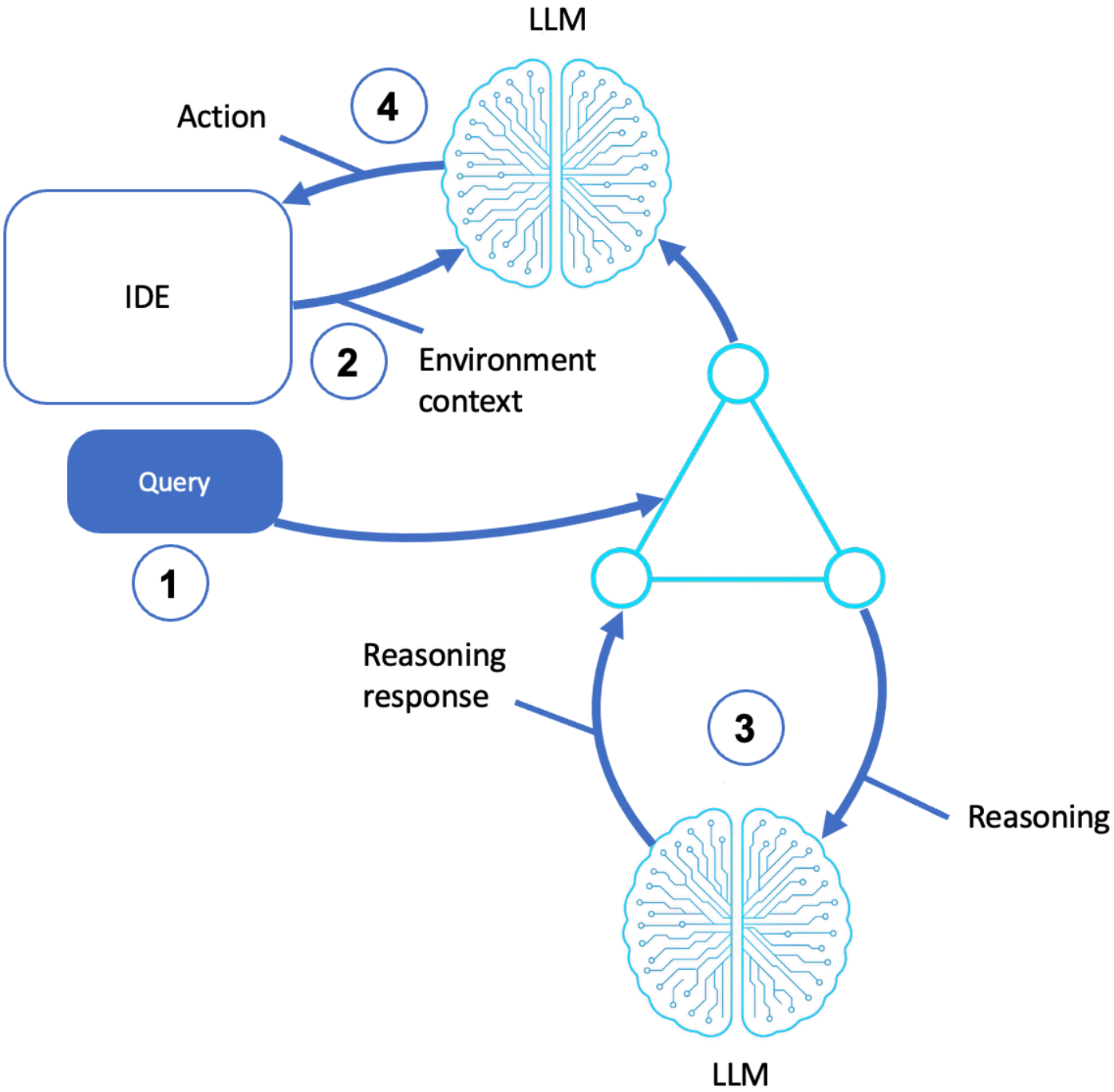
# 編碼代理程式

編碼代理程式可以說明程式設計任務、產生或修改程式碼，以及與開發人員環境互動的原因，例如 IDEs和 CLIs。這些代理程式將自然語言理解與結構化推理相結合，以協助、增強和自動化軟體開發，從函數產生到錯誤修正和測試編寫。

與自動完成工具不同，編碼代理程式會主動解譯使用者目標、查詢開發環境的內容（例如，它會開啟檔案並追蹤錯誤）、識別需求，然後提議和執行動作。

# Architecture

下圖顯示編碼代理程式模式：




## 說明

1. 接收查詢

- 使用者透過命令調色盤、聊天視窗或 CLI 提供自然語言指示（例如，「將記錄新增至此函數」或「Refactor for readability」）。

2. 擷取環境內容
   - 代理程式會從 IDE 收集內容，包括作用中檔案、游標位置、程式碼片段和符號表。
   - 它會輸出錯誤訊息、測試結果，以及來自其他代理程式的輸出。
3. LLM 推理
   - 代理程式會傳送提示給 LLM，包括查詢和環境內容。
     - LLM 會執行推理傳遞來判斷下列項目：
     - 需要變更的內容
     - 如何產生解決方案
     - 任何重構、重寫或編碼步驟
4. 執行動作
   - LLM 會將輸出傳回至代理程式，並將其匯入 IDE 或執行時間環境。
   - 這可能包括插入或修改程式碼、產生註解或文件，以及觸發下游建置、測試和固定任務。

## 功能

- 高內容感知（例如，IDE 狀態、游標和語法樹）
- 目標和意見回饋的反覆推理
- 選用程式碼規劃和動作分離（例如，第一個原因，然後採取行動）
- 適用於同步或非同步開發人員工作流程

## 常用案例

- 從任務描述產生程式碼
- 程式碼重構和最佳化
- 測試案例產生和驗證
- 錯誤說明和偵錯
- 文件助理
- 配對的程式設計 Copilot

## 實作指引

- 您可以使用下列工具和 建置此模式 AWS 服務：
- Amazon Bedrock，用於 LLM 驅動的產生和推理
- Amazon Q Developer 用於編碼建議和完成
- AWS Lambda 或 Amazon Elastic Container Service (Amazon ECS)，用於執行和測試沙盒環境
- AWS Cloud9、VS 程式碼延伸模組或自訂 IDE 整合，以託管和評估內容
- Amazon Simple Storage Service (Amazon S3) 用於儲存中繼提示、回應和修訂歷史記錄

### Summary

編碼代理程式是採用 AI 技術的新開發工具，能夠解譯自然語言、分析內容、產生多步驟程式碼變更，以及與軟體開發生命週期整合。

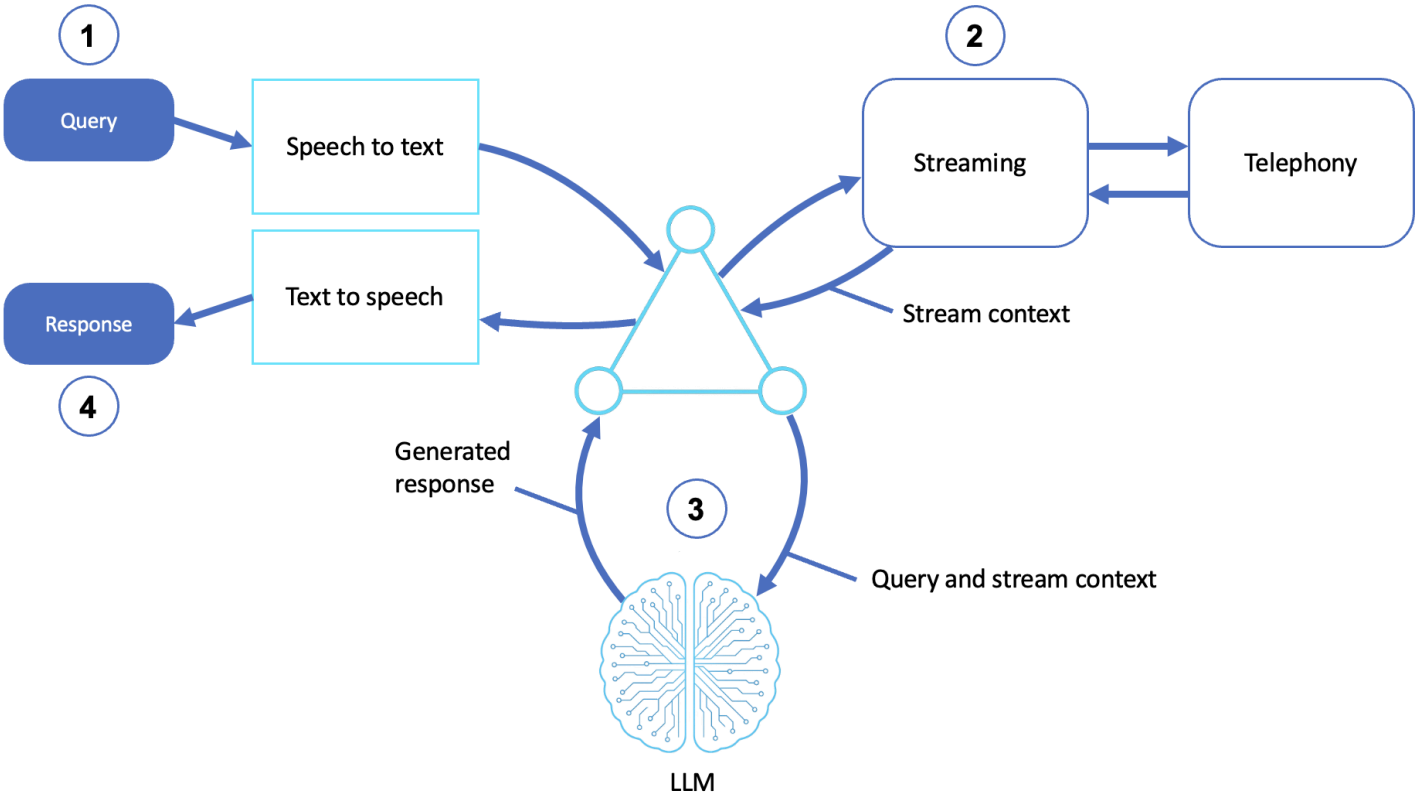
# 語音和語音客服人員

語音和語音客服人員透過語音對話與使用者互動。這些代理程式整合語音辨識、自然語言理解和語音合成，以跨電話、行動、Web 和內嵌平台啟用對話式 AI。

語音代理程式在免持、即時或可存取性驅動的環境中特別有效。透過結合串流介面與 LLM 支援的推理，它們可以促進使用者感到自然的豐富、動態互動。

### Architecture

下圖顯示語音和語音代理程式：




## 說明

1. 接收語音查詢
   - 使用者向電話、麥克風或內嵌系統發出請求。
   - speech-to-text(STT) 模組會將音訊轉換為文字。
2. 整合串流和電話內容
   - 代理程式使用串流介面來即時管理音訊 I/O。
   - 如果部署在聯絡中心或電信內容中，電話語音整合會處理工作階段路由、雙音多頻率 (DTMF) 輸入和媒體傳輸。

注意：DTMF 是指您按下電話鍵盤上的按鈕時產生的音調。在語音客服人員內串流和電話內容整合的情況下，DTMF 會在通話期間用作訊號輸入機制，尤其是在互動式語音回應 (IVR) 系統中。DTMF 輸入可讓代理程式：

- 辨識選單選擇（例如，「按 1 進行計費。 支援請按 2。」）
- 收集數值輸入（例如，帳戶號碼、PINs和確認號碼）
- 在通話流程中觸發工作流程或狀態轉換

- 必要時從語音還原為按鍵音

1. 透過 LLM 串流內容的原因
   - 查詢會傳送給代理程式，其會連同任何工作階段中繼資料（例如呼叫者 ID、先前內容）一起傳遞至 LLM。
   - LLM 會產生回應，如果互動正在進行，可能會使用chain-of-thought策略或多迴轉記憶體。
2. 傳回語音回應
   - 代理程式會使用text-to-speech(TTS) 將其回應轉換為語音。
   - 它會透過語音頻道將音訊傳回給使用者。

## 功能

- 即時語音理解和產生
- 支援 STT 和 TTS 的多語言 I/O
- 與電話語音或串流 APIs整合
- 輪換之間的工作階段意識和記憶體切換

## 常用案例

- 對話式 IVR 系統
- 虛擬接收人員和預約排程人員
- 語音驅動服務台客服人員
- 穿戴式語音助理
- 智慧家庭和可存取工具的語音界面

## 實作指引

您可以使用下列工具和 建置此模式 AWS 服務：

- Amazon Lex V2 或 Amazon Transcribe for STT
- Amazon Polly for TTS
- 用於串流和電話通訊的 Amazon Chime SDK、Amazon Connect 或 Amazon Interactive Video Service (Amazon IVS)

- Amazon Bedrock 使用 Anthropic、AI21 或其他基礎模型推理
- AWS Lambda 連接 STT、LLM、TTS 和工作階段內容

（選用）其他增強功能可能包括下列項目：

- 適用於內容感知 RAG 的 Amazon Kendra 或 OpenSearch
- 適用於工作階段記憶體的 Amazon DynamoDB
- Amazon CloudWatch Logs 和 AWS X-Ray 可追蹤性

## Summary

語音和語音客服人員是透過自然對話互動的智慧型系統。透過整合語音界面與 LLM 推理和即時串流基礎設施，語音代理器可實現無縫、可存取和可擴展的互動。

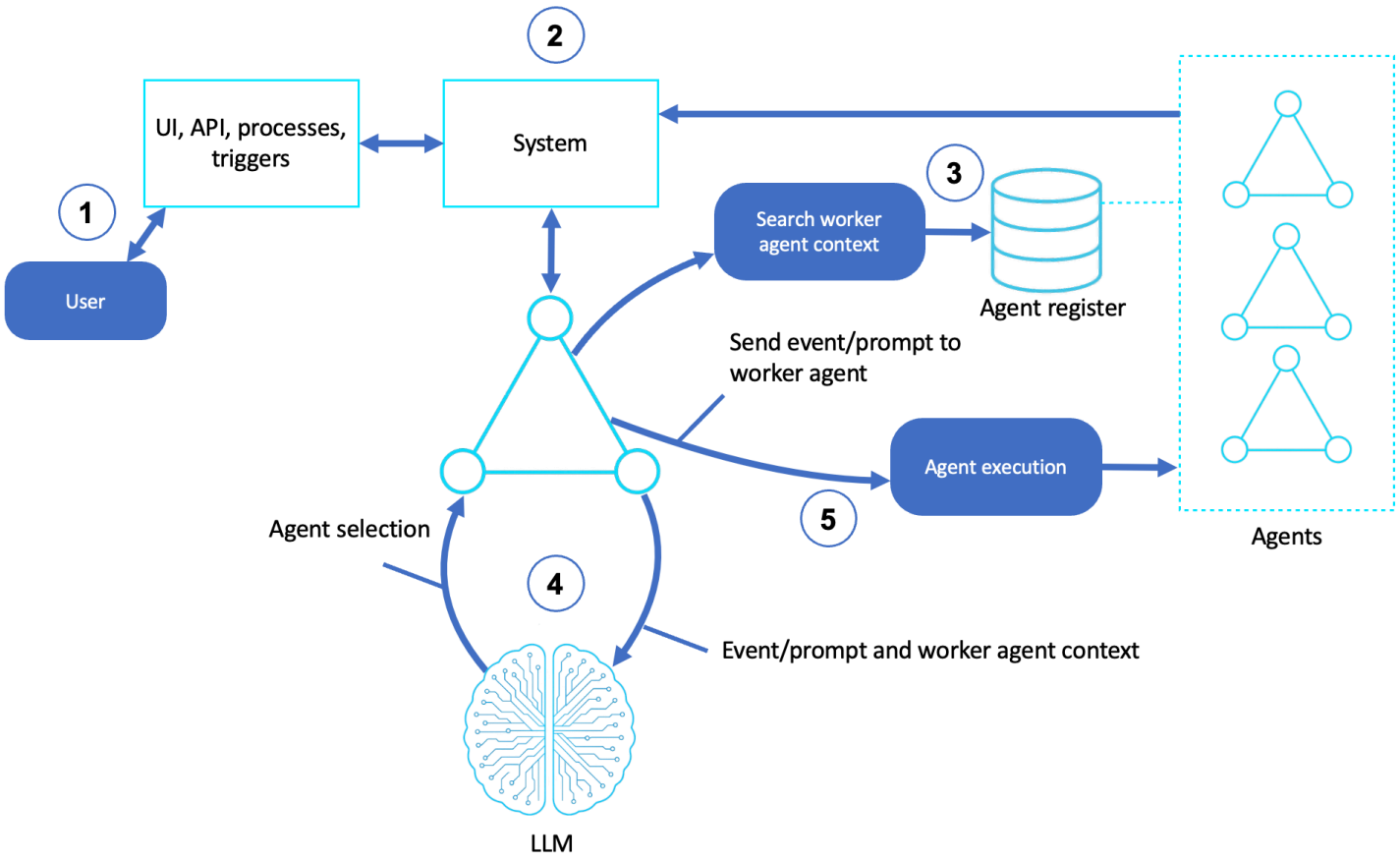
# 工作流程協同運作代理程式

工作流程協同運作代理程式會管理和協調跨分散式系統的多步驟任務、程序和服務。這些客服人員將工作委派給子代理程式或其他系統，維護執行內容，並根據中繼結果進行調整，而不是推理和獨立行動。

這些代理程式是自動化流程的基本部分。它們在處理長時間執行的任務、多重代理程式組成和跨網域整合時特別有用，其中各種代理程式和工具必須按順序或條件呼叫。

## Architecture

下圖顯示工作流程協同運作代理程式：




## 說明

1. 接收使用者輸入
   - 使用者（或外部觸發）透過 UI、API 或系統事件啟動任務。
2. 處理系統事件
   - 系統元件會收到請求並發出需要協調的事件或命令。
3. 擷取內容
   - 工作流程代理程式會根據中繼資料、網域和先前的成功率，查詢知識庫和代理程式登錄檔，為任務尋找合適的工作者代理程式。
4. 選取 LLM 代理程式
   - LLM 透過分析任務描述和可用選項，協助選擇最佳的代理程式或工作流程計劃。
   - 它也可以制定任務特定的提示，以傳送至選取的客服人員。
5. 委派和執行
   - 選擇的工作者代理程式會收到事件或提示，並開始執行命令。
   - 它可以追蹤執行狀態、在失敗時重試，並將中繼結果傳遞給序列中的下一個代理程式。

## 功能

- 客服人員組成（例如，主管、協作者客服人員和工具）
- 事件驅動或排程執行
- 隨著時間的推移，記憶體和狀態追蹤
- 階層式或平行任務協同運作（與非同步工作流程相比的同步）
- 動態代理程式選擇和鏈結

## 常用案例

- 多步驟自動化（例如，資料擷取和報告）
- 客戶服務路由和呈報（例如，agent-as-coordinator)
- AI 代理器與同一迴圈中的人類和機器人協調
- 使用 LLM 支援的邏輯自動化企業程序
- 混合系統結合了 AI 代理器和傳統協同運作工具

## 實作指引

您可以使用下列工具和 建置此模式 AWS 服務：

- Amazon Bedrock 用於推理和代理程式選擇
- AWS Step Functions 或 Amazon EventBridge 用於工作流程合成
- AWS Lambda 作為執行單位或任務執行器
- Amazon DynamoDB、Amazon Simple Storage Service (Amazon S3) 或 Amazon RDS 追蹤狀態和結果
- AWS AppFabric 或 Amazon AppFlow 進行跨系統協調
- （選用） 使用 Amazon SageMaker 執行代理程式來託管特定網域的工作者代理程式

## Summary

工作流程代理程式在多代理程式環境中協調、調整和調整目標。這表示 AI 代理器可以協作、適應執行時間條件，並透過模組化、可解釋的工作流程提供複雜的結果。

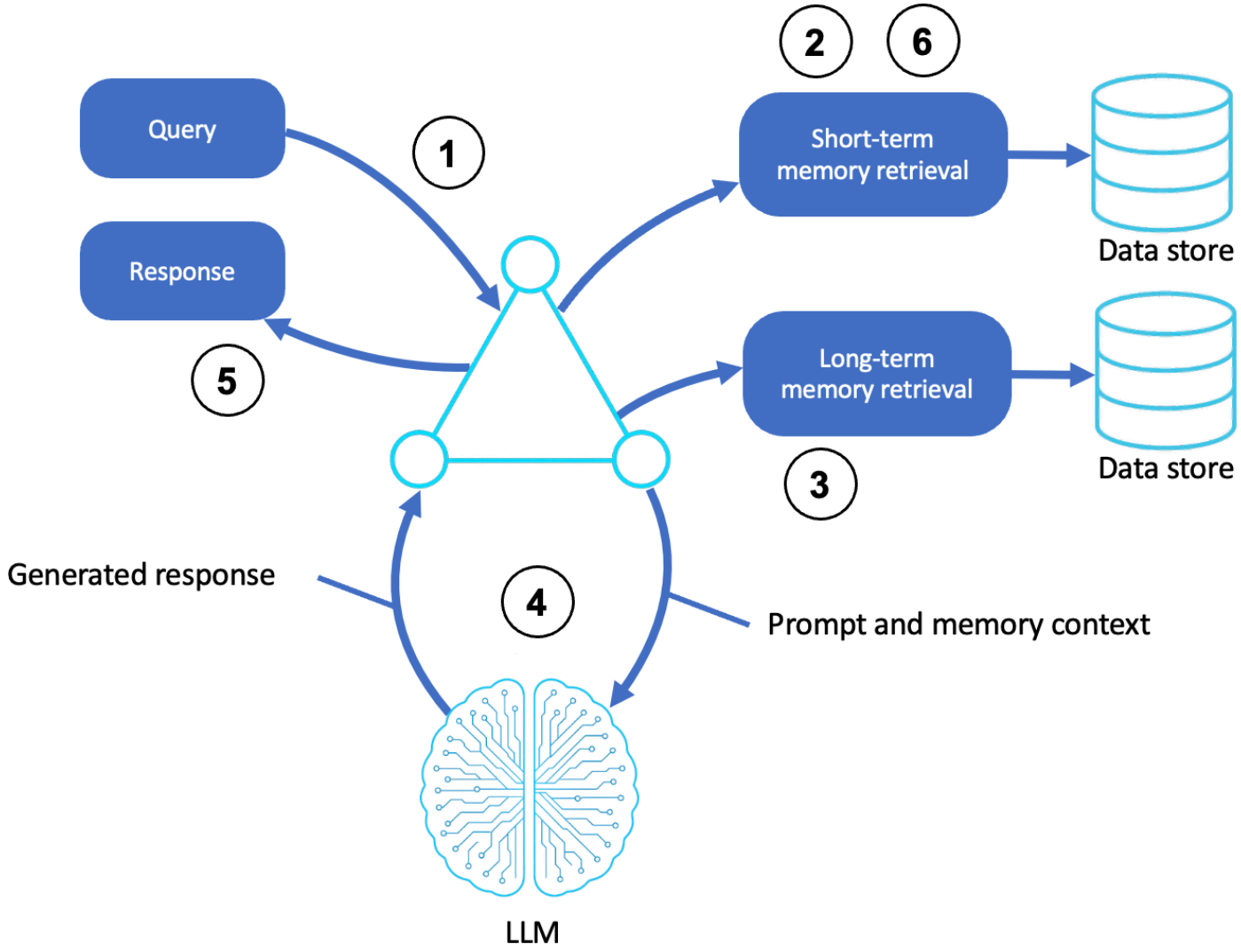
# 記憶體增強型代理程式

記憶體增強了使用短期和長期記憶體來儲存、擷取和原因的能力。這可讓他們維持跨多個任務、工作階段和互動的內容，進而產生更一致、個人化的策略回應。

與無狀態代理程式不同，記憶體增強型代理程式會透過參考歷史資料、從先前的結果中學習，以及做出符合使用者目標、偏好設定和環境的決策來進行調整。

## Architecture

下圖顯示記憶體增強型代理程式：




## 說明

1. 接收輸入或事件

- 客服人員會收到使用者查詢或系統事件。這可能是文字、API 觸發條件或環境變更。

2. 擷取短期記憶體
   - 客服人員會擷取最近的對話歷史記錄、任務內容，或與工作階段或工作流程相關的系統狀態。
3. 擷取長期記憶體
   - 代理程式會查詢長期記憶體（例如向量資料庫和鍵值存放區）以取得歷史洞見，例如：
     - 使用者偏好設定
     - 過去的決策和結果
     - 已學習的概念、摘要或體驗
4. 透過 LLM 的原因
   - 記憶體內容內嵌在 LLM 提示中，允許代理程式根據目前輸入和先前的知識進行推理。
5. 產生輸出
   - 代理程式會產生上下文感知的回應、計劃或動作，根據任務歷史記錄和使用者的輸入進行個人化。
6. 更新記憶體
   - 更新的目標、成功和失敗訊號，以及結構化回應等新資訊都會存放以供未來任務使用。

## 功能

- 對話或事件間的工作階段持續性
- 目標隨時間的持久性
- 根據不斷發展狀態的情境感知
- 先前成功和失敗通知的適應性
- 符合使用者偏好設定和歷史記錄的個人化

## 常用案例

- 記住使用者偏好設定的對話式 copilot
- 追蹤程式碼庫變更的編碼代理程式
- 根據任務歷史記錄調整的工作流程代理程式
- 從系統知識發展的數位分身
- 避免備援擷取的研究代理程式

## 實作經記憶體驗證的代理程式

將下列工具和 AWS 服務 用於記憶體增強型代理程式：

| 記憶體層 | AWS 服務 | 用途 |
| --- | --- | --- |
| 短期 | Amazon DynamoDB、Redis、Amazon Bedrock 內容 | 快速擷取最近的互動狀態 |
| 長期 （結構化） | Amazon Aurora、Amazon DynamoDB、Amazon Neptune | 事實、關係和日誌 |
| 長期 （語意） | OpenSearch、Postgre SQL、Pinecone | 內嵌型擷取 （即 RAG) |
| 儲存 | Amazon S3 | 儲存文字記錄、結構化記憶體和檔案 |
| 協同運作 | AWS Lambda 或 AWS Step Functions | 管理記憶體注入和更新生命週期 |
| 推理 | Amazon Bedrock | 具有記憶體提示的 Anthropic Claude 或 Mistral |

## 實作記憶體注入提示

若要將記憶體整合到代理程式推理中，請使用結構化狀態和擷取擴增內容注入的組合：

- 在建構語言模型的提示時，包含最新的客服人員狀態和最近的對話歷史記錄做為結構化輸入，因此它可以具有完整內容。
- 使用擷取擴增產生 (RAG) 從長期記憶體中提取相關文件或事實。
- 摘要先前的計畫、內容和互動以進行壓縮和相關性。
- 在推論期間注入外部記憶體模組，例如向量存放區或結構化日誌，以引導決策。

## Summary

記憶體增強型代理程式透過從經驗中學習並記住使用者內容來維持思維持續性。這些客服人員使用長期協作、個人化和策略推理，超越被動式智慧。在代理式 AI 方面，記憶體可讓代理程式表現得更像是適應性數位對等，更不像無狀態工具。

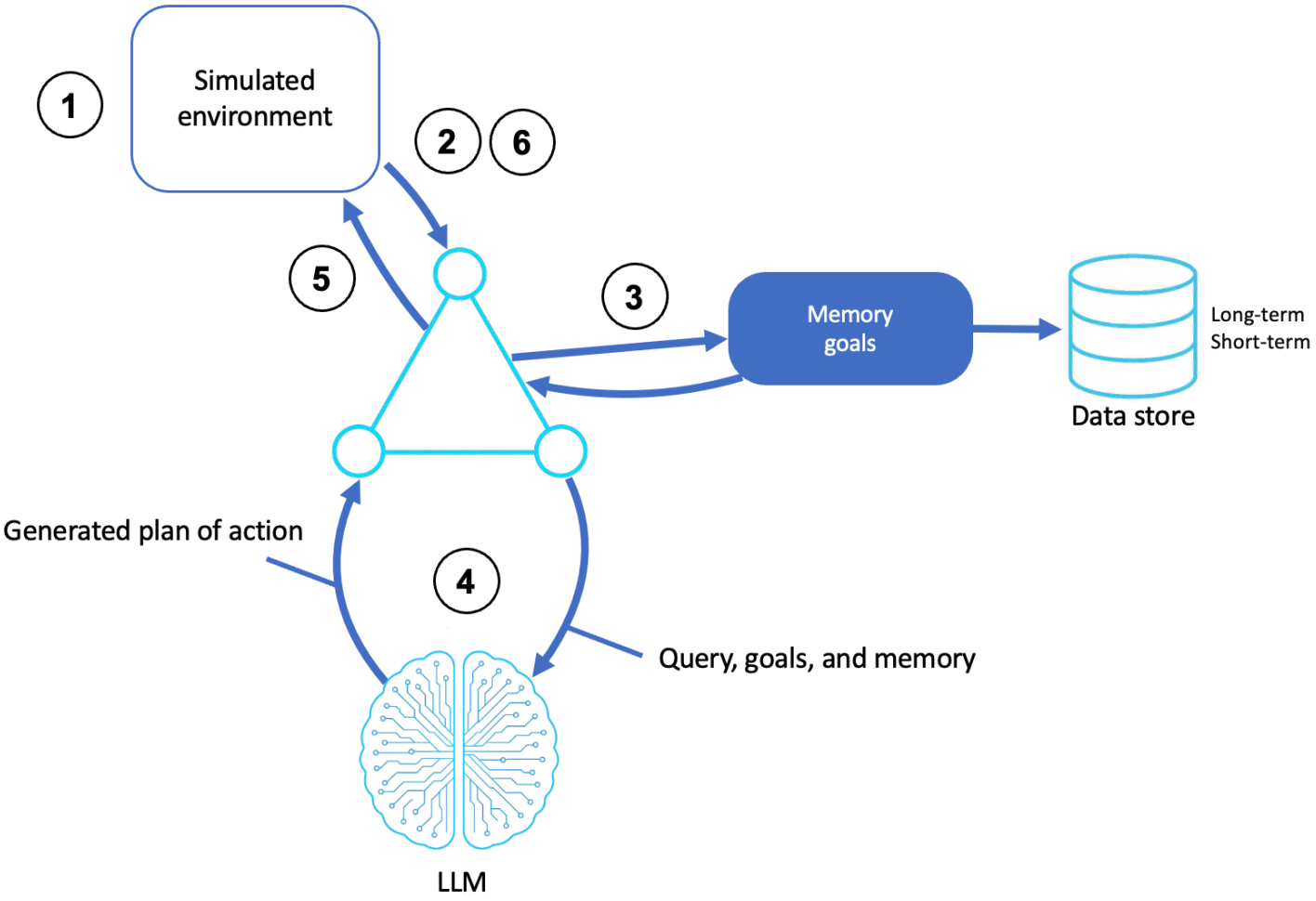
# 模擬和測試平台代理程式

模擬和測試平台代理程式在虛擬或受控制的環境中操作，它們在其中進行推理、動作和學習。這些代理程式會在可重複的設定中模擬行為、模型結果和訓練策略，然後再將其套用至真實環境。

此模式適用於反覆開發、強化學習 (RL)、自主決策評估和緊急行為測試。模擬代理程式通常在封閉迴圈中操作，接收來自其環境的意見回饋並相應地調整其行為，這使得它們對於涉及空間推理、即時控制或複雜系統動態的任務至關重要。

## Architecture

下圖顯示模擬或測試平台代理程式：




## 說明

1. 啟動環境
   - 代理程式會啟動模擬環境（例如，3D world、物理引擎、CLI 沙盒或合成資料串流）。
   - 代理程式會以初始任務、目標或政策載入環境。
2. 感知代理程式
   - 代理程式會透過模擬遙測（例如，感應器模擬、虛擬攝影機和結構化日誌）來感知目前狀態。
3. 擷取目標和記憶體
   - 代理程式會擷取其指派的目標、案例指示或內容目標。
   - 它也可能擷取先前的記憶體，包括下列項目：
     - 長期策略或政策
     - 環境地圖或已知限制條件
     - 類似模擬過去的成功或失敗

4. 原因和計劃
   - LLM 會解譯模擬狀態、任務目標和學到的知識。
   - 它會產生行動計劃或控制命令。
5. 執行模擬動作
   - 代理程式會執行計劃、修改狀態、導覽空間或與虛擬實體互動。
6. 了解
   - 客服人員評估動作結果
   - 根據代理程式的組態，它可能會執行下列動作：
     - 執行 RL
     - 記錄未來微調的結果
     - 即時調整策略

## 功能

- 在合成或虛擬環境中操作
- 支援trial-and-error學習、政策精簡和系統建模
- 低風險測試行為、故障處理和邊緣案例
- 在多代理程式設定中啟用緊急代理程式行為分析
- 支援封閉迴路控制和human-in-the-loop探勘

## 常用案例

- 機器人、無人機和遊戲的強化學習
- 虛擬道路上的自動車輛訓練
- DevOps 和測試平台案例的模擬 UIs 或 CLIs
- 社交模擬中的緊急行為實驗
- 生產前決策邏輯的安全驗證

## 實作指引

您可以使用下列工具和 建置模擬和測試平台代理程式 AWS 服務：

| 元件 | AWS 服務 | 用途 |
| --- | --- | --- |
| Environment | Amazon SageMaker Studio 實驗室中的 Amazon ECS、Amazon EC2 或自訂模擬器 | 執行虛擬世界 (Gazebo、Unity、Unreal) 或沙盒 CLIs |
| 代理程式邏輯 | Amazon Bedrock、Amazon SageMaker 或 AWS Lambda | LLM 型規劃器或 RL 代理程式 |
| 回饋迴圈 | Amazon SageMaker 強化學習、Amazon CloudWatch 或自訂日誌 | 獎勵追蹤、結果評分和行為記錄 |
| 記憶體和重播 | Amazon S3、Amazon DynamoDB 或 Amazon RDS | 持久性狀態、片段歷史記錄或案例資料 |
| 視覺效果 | Amazon CloudWatch 儀表板或 Amazon SageMaker 筆記本 | 觀察政策變更、結果和訓練指標 |

以下是其他應用程式：

- AWS SimSpace Weaver 適用於大規模空間模擬
- AWS IoT Core 用於測試陰影裝置
- 用於代理程式評估和基準測試的 Amazon SageMaker 實驗

## Summary

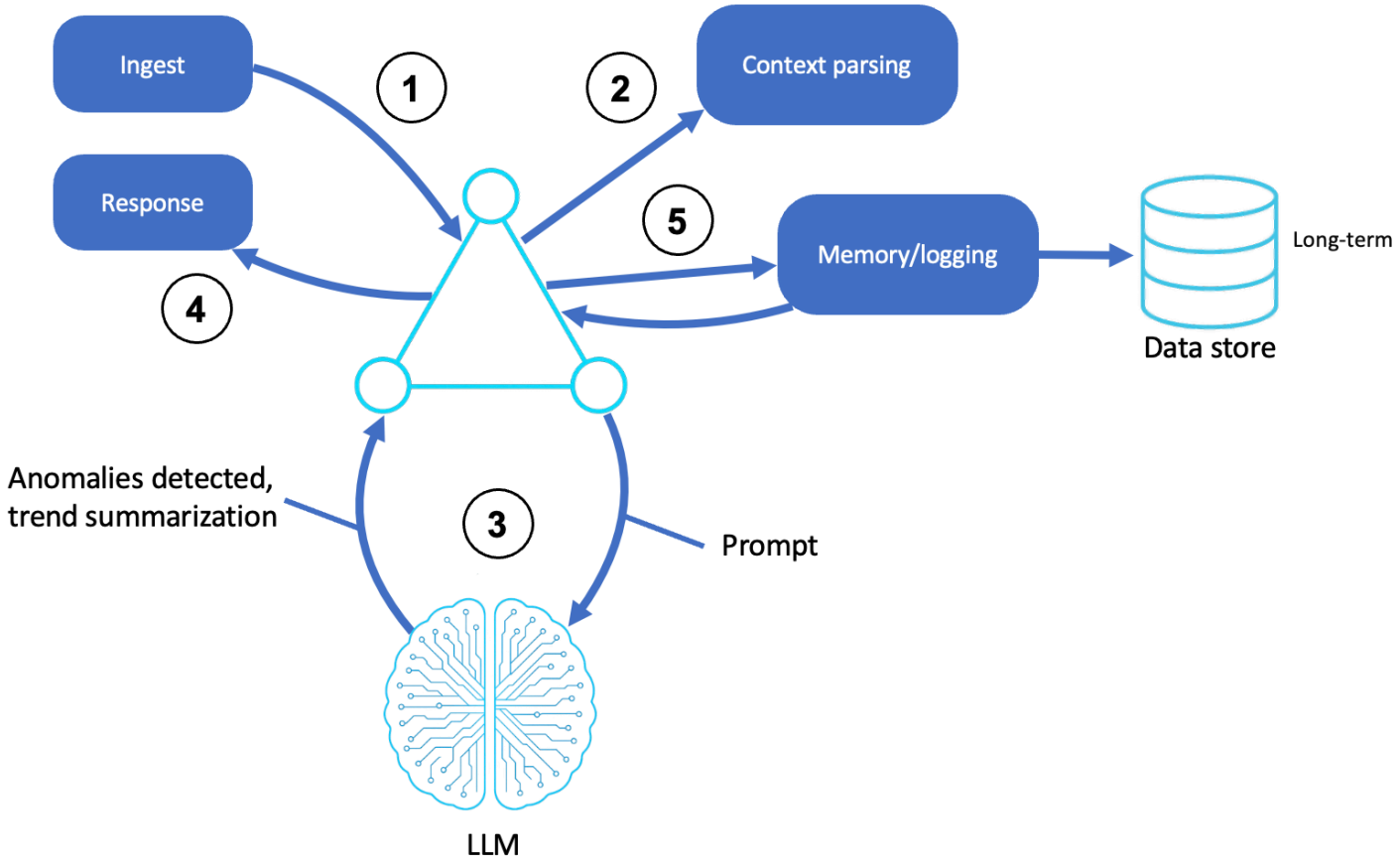
在部署到生產系統之前，模擬和測試平台代理程式用於結構化探勘。使用這些代理程式來訓練自動導覽政策、在合成環境中測試業務流程，以及評估協調模式的扭曲。

# 觀察者和監控代理程式

觀察者和監控客服人員被動觀察系統、環境和互動，以偵測模式、產生洞見和觸發動作。作為智慧監看器，它們可增強提醒、診斷和稽核，而無需直接啟動行為。

這些代理程式在傳統監控缺乏適應性或推理的情況下表現卓越，特別是在 AI-in-the-loop監控、異常偵測、合規監督和安全智慧方面。觀察程式代理程式是持續監控系統遙測和使用者互動的事件接聽程式。代理程式取決於感知、解釋和條件式呈報或報告。

## Architecture




## 說明

1. 擷取遙測
   - 代理程式會從一或多個系統來源接收輸入，例如：
     - 日誌（應用程式、基礎設施、安全性）
     - 指標（效能、延遲、用量）
     - 事件 (API 呼叫、使用者動作、感應器資料）
2. 剖析內容
   - 原始輸入會剖析、結構化並充實中繼資料，例如時間戳記、演員身分、系統狀態和追蹤 ID。
3. 使用 LLM 的原因

- 代理程式使用 LLM 或邏輯模組來解譯剖析的輸入，方法是識別異常、摘要趨勢，以及關聯分散式追蹤或時段。

4. 分類或提醒
   - 代理程式會判斷觀察到的行為是否需要下列項目：
     - 提醒或呈報
     - 報告或儀表板更新
     - 回應觸發（例如，自動修復和政策強制執行）
5. 日誌記憶體或回饋迴圈
   - 客服人員會儲存事件和決策，以供其他客服人員長期學習、稽核或未來參考。

## 功能

- 被動和非侵入性（代理程式不會直接採取行動）
- 高度可擴展和非同步
- AI 驅動的跨雜訊或分散式訊號的相互關聯
- 支援稽核、合規和即時洞見
- 可以饋送下游代理程式或人工工作流程

## 常用案例

- 微服務和 APIs 的 AI 擴增可觀測性
- 監控模型偏離、政策違規或out-of-band行為
- 客戶活動分析或互動摘要
- 監控遞交或部署的程式碼檢閱代理程式
- 使用 LLM 推理進行安全或合規日誌監控

## 實作指引

您可以使用下列工具和 來建置觀察者和監控代理程式 AWS 服務：

| 元件 | AWS 服務 | 用途 |
| --- | --- | --- |

| | | |
|---|---|---|
| 事件擷取 | Amazon EventBridge、Amazon CloudWatch Logs、Amazon Kinesis、Amazon S3 | 擷取結構化和非結構化遙測 |
| 預處理 | AWS Lambda, AWS Glue, AWS Step Functions | 將原始資料轉換為結構化提示 |
| 推理引擎 | Amazon Bedrock、Amazon SageMaker、 AWS Lambda | 分析事件、分類行為、產生洞見 |
| 儲存和記憶體 | Amazon S3、Amazon DynamoDB、OpenSearch | 持久性觀察、摘要和輸出 |
| 警示和呈報 | Amazon SNS AWS AppFabric、Amazon EventBridge | 觸發下游系統或代理程式 |

以下是其他應用程式：

- AWS Security Hub CSPM 用於安全日誌監控
- Amazon Quick Suite 視覺化代理程式輸出

## Summary

觀察者和監控客服人員可即時追蹤系統和行為。它們透過識別人類或規則可能忽略的模式來偵測異常、稽核安全性並收集操作智慧。此功能有助於建立系統，以適應不斷變化的條件，並根據全面的資料分析做出決策。

# 多代理程式協同合作

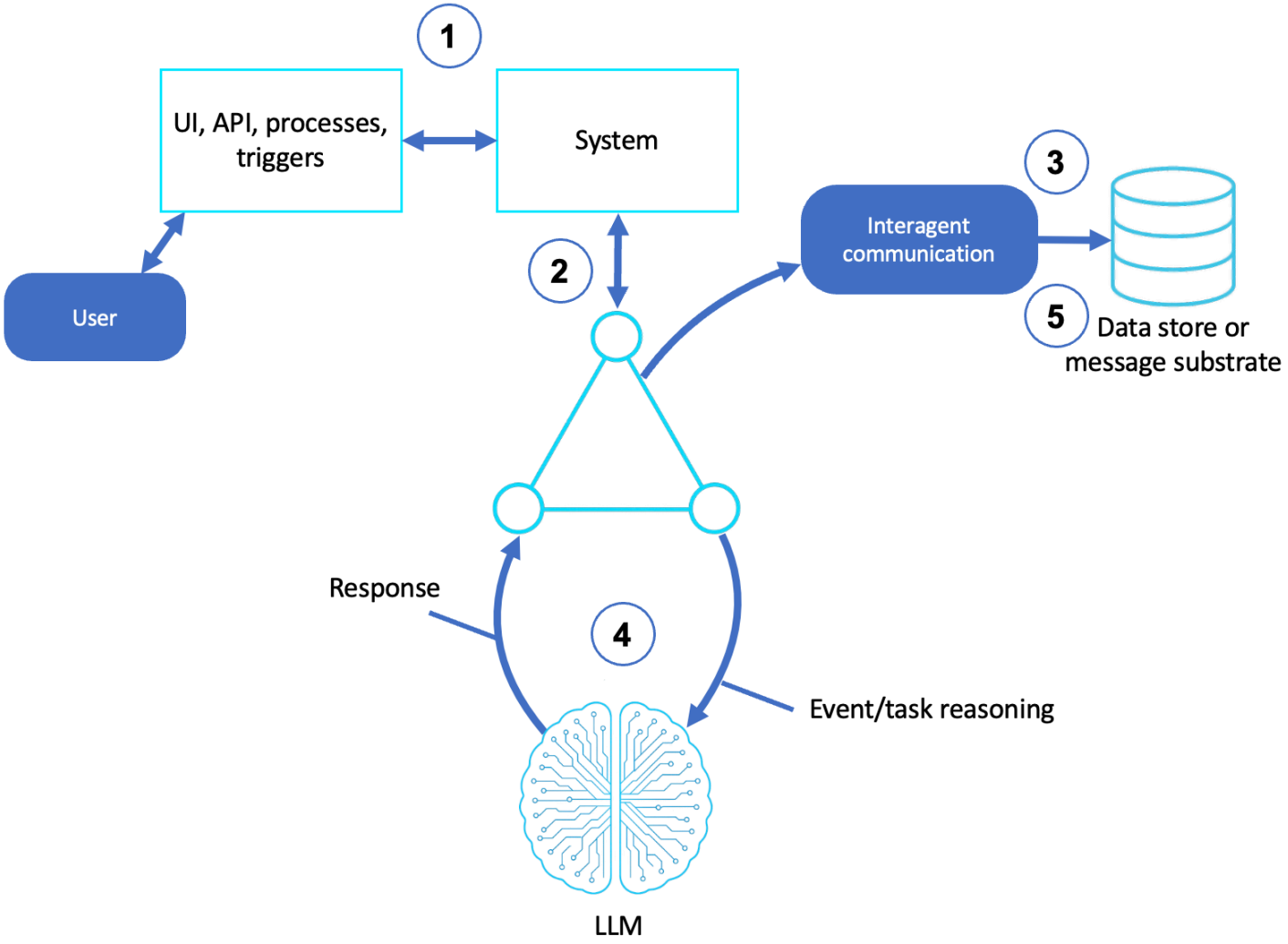
多重代理程式協同合作是指一種模式，其中多個自動代理程式，每個代理程式都有不同的角色、專業或目標，進行交涉以解決複雜的任務。這些客服人員可以獨立運作，也可以與其他客服人員一起運作，方法是共用資訊、分配責任，以及共同推理目標。

此模式與工作流程代理程式不同，工作流程代理程式會集中協調並將任務委派給結構化流程中的次級代理程式。相反地，多代理程式協同合作透過啟用認知的適應性、平行處理和劃分來強調peer-to-peer或緊急協調。下表比較多代理程式與工作流程代理程式的協同合作：

| 功能 | 工作流程代理程式 | 用途 |
| --- | --- | --- |
| 控制項 | 集中式協調器 | 分散式、分散式或角色型對等 |
| 互動 | 一個客服人員委派和追蹤執行 | 多個客服人員交涉、共用和調整 |
| 設計 | 預先定義的任務順序 | 緊急、彈性的任務分佈 |
| 協調 | 程序協同運作 | 合作或競爭互動 |
| 使用案例 | 企業程序自動化 | 複雜的推理、探勘和緊急策略 |

## Architecture

下圖顯示多代理程式協同合作：




## 說明

1. 啟動任務
   - 使用者或系統發出高階目標或問題。
   - "manager" 代理程式或啟動內容會定義目標。
2. 指派或探索角色
   - 客服人員自行指派（符號邏輯或推理）或委派（事件中介裝置）給其他角色，例如規劃人員、研究員、執行者、評論者或解釋者。
3. 與其他客服人員通訊
   - 客服人員透過共用記憶體、訊息佇列或提示鏈結進行通訊。
   - 他們可能會彼此爭論、查詢或提議子任務。
4. 使用特殊推理

- 每個代理程式都會使用自己的模型或網域邏輯來解決其部分的問題。
- 客服人員可以使用 LLMs搭配角色特定的提示和記憶體。

5. 協調輸出或目標
   - 客服人員會將貢獻合成最終答案、計劃或動作。
   - （選用）監督代理程式可能會驗證或摘要合成的輸出。

## 功能

- 具有專業角色或技能的對等層級客服人員
- 透過溝通或溝通的緊急行為
- 平行處理複雜或多面向的問題
- 支援深思熟慮、自我校正和反射反覆運算
- 建立社交動態、科學協作或企業團隊角色的模型

## 常用案例

- 自主研究團隊（搜尋代理程式、摘要程式和驗證程式）
- 軟體開發（規劃器、編碼器和測試器）
- 商業案例建模（財務、政策和合規）
- 協商、競價或多方推理
- 多模式任務（影像、文字和邏輯）

## 實作指引

您可以使用下列工具和 建置多代理程式系統 AWS 服務：

| 元件 | AWS 服務 | 用途 |
| --- | --- | --- |
| 代理程式託管 | Amazon Bedrock、Amazon SageMaker、 AWS Lambda | 託管個別 LLM 驅動代理程式 |

| | | |
|---|---|---|
| 通訊層 | Amazon SQS、Amazon EventBridge、 AWS AppFabric | 客服人員之間的傳訊和協調 |
| 共用記憶體 | Amazon DynamoDB、Amazon S3 或 OpenSearch | 多代理程式記憶體或黑板系統 |
| 協同運作層 | AWS Step Functions、 AWS Lambda 管道 | 啟動、逾時、後援和重試邏輯 |
| 客服人員識別 | Amazon Bedrock 代理程式（角色定義）AWS AppConfig 和 Amazon Bedrock converse API (Amazon Bedrock 外部的代理程式） | 角色型工具或代理程式調用和界限強制執行 |
| 緊急互動 | Amazon EventBridge 管道或代理程式登錄檔 | 啟用動態任務路由或升級 |

## Summary

多重客服人員協作會將問題解決任務分散到模組化、角色驅動的客服人員。與工作流程協同運作不同，協作模式使用緊急智慧、彈性和可擴展性，反映人類如何解決問題。它對於受惠於不同觀點的開放式網域、創意任務、多模式推理和環境特別重要。

## 結論

先前討論的模式說明代理式 AI 實際實作的基礎方法。從基本推理到記憶體擴增智慧，每個模式都針對以自主性、非同步和代理程式為基礎的感知、認知和動作進行唯一設定。

這些模式共用詞彙和技術藍圖，用於建置智慧型目標導向系統。無論模式是內嵌在使用者介面中、透過雲端服務協調，還是跨客服人員團隊進行協調，每個模式都是可調整且模組化的。

## 要點

- 客服人員模式可編寫 – 大多數實際客服人員混合兩種或多種模式（例如，具有工具型推理和記憶體的語音客服人員）。

- 客服人員設計是情境式 – 根據互動表面、任務複雜性、延遲容錯能力和特定領域的限制條件來選擇模式。
- AWS 可實現原生實作 – 透過 Amazon Bedrock AWS Lambda AWS Step Functions、Amazon SageMaker 和事件驅動型架構，每個代理程式模式都可以大規模交付。

# LLM 工作流程

在客服人員模式中，我們探索了常見的 AI 客服人員模式，每個模式都以一組模組化功能為基礎：感知、動作、學習和認知。在許多代理程式模式中，認知模組的核心是大型語言模型 (LLM)，能夠推理、規劃和決策。不過，單獨使用 LLM 並不足以產生智慧的目標導向行為。

為了可靠地執行複雜的任務，客服人員必須將 LLM 內嵌在結構化工作流程中，其中模型的功能透過工具、記憶體、規劃迴圈和協調邏輯增強。這些 LLM 工作流程可讓客服人員細分目標、路由子任務、呼叫外部服務、反映結果，以及與其他客服人員協調。

本章介紹建置強大、可擴展且智慧 LLM 驅動的認知模組的核心設計模式，這些模組是以可重複使用的工作流程組織而成。

本節內容

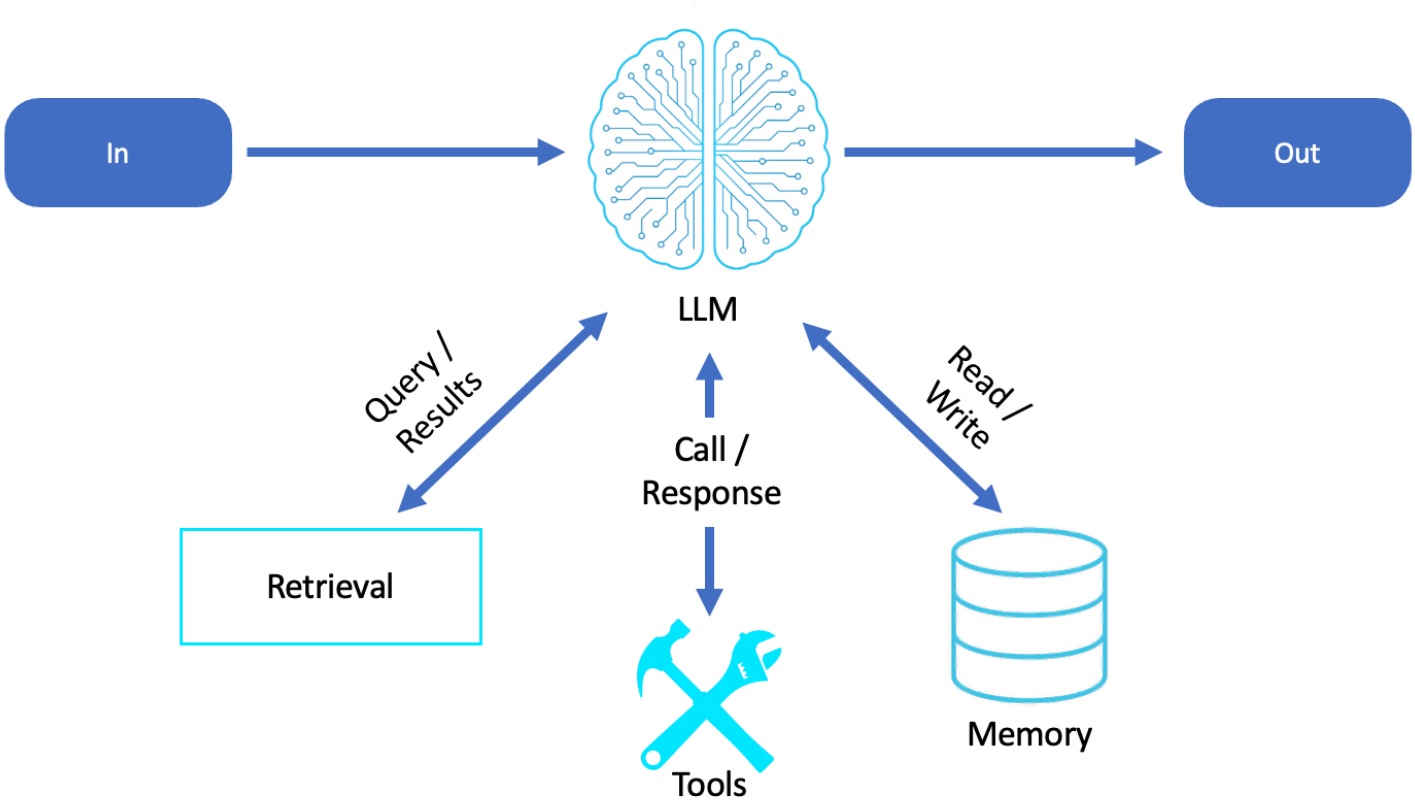


## LLM 擴增的認知概觀

在其核心上，軟體代理程式的認知模組可以檢視為包裝在擴增中的 LLM。代理程式可以使用下列建置區塊，在其環境中有效地推理原因：

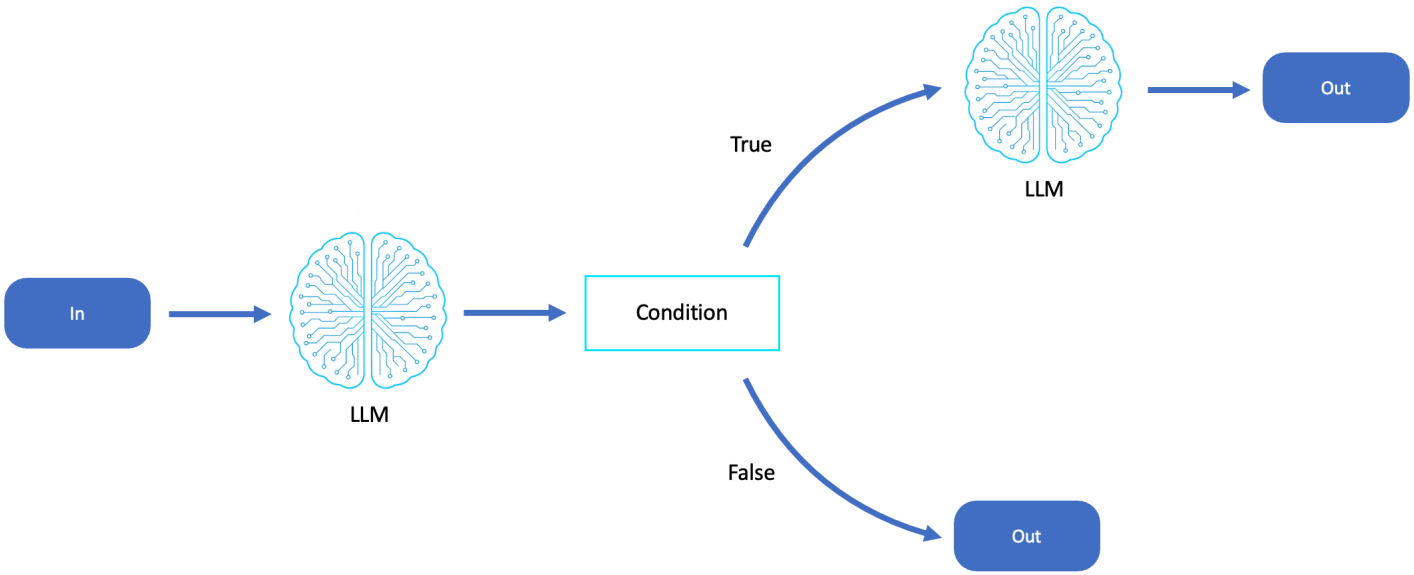
- 提示 – 使用內容、指示、範例和記憶體建立輸入架構
- 擷取 – 透過向量搜尋或語意記憶體為 LLM 提示提供up-to-date或特定領域的知識，例如透過擷取擴增產生 (RAG)
- 工具使用 – 讓 LLM 叫用 APIs或呼叫函數以擷取資訊或對資訊採取行動
- 記憶體 – 使用結構化資料庫或內容摘要，將持久性或以工作階段為基礎的狀態納入推理迴圈

這些擴增是由工作流程組成，這些工作流程會定義 LLM 如何隨時間和跨任務使用，並將其從無狀態引擎轉換為動態推理代理程式。




## 提示鏈結的工作流程

提示鏈結會將複雜的任務分解為一系列步驟，其中每個步驟都是獨立的 LLM 調用，可處理或建置在前一個步驟的輸出上。



提示鏈結工作流程適用於任務邏輯上可分為循序推理步驟，以及中繼輸出通知下一個階段的情況。它擅長於需要結構化思維、漸進式轉型或分層分析的工作流程，例如文件檢閱、程式碼產生、知識擷取和內容精簡。

## 描述

- 任務的複雜性超過單一 LLM 呼叫的內容視窗或推理深度。
- 來自一個步驟的輸出（例如，分析、摘要或規劃）會成為後續決策或產生階段的輸入。
- 您需要跨推理階段的透明度和控制（例如，可稽核的中繼結果）。
- 您想要在步驟之間插入外部驗證、篩選或擴充邏輯。
- 它非常適合在管道型推理迴圈中操作的代理程式，例如研究代理程式、編輯助理、規劃系統和多階段 copilot。

## 功能

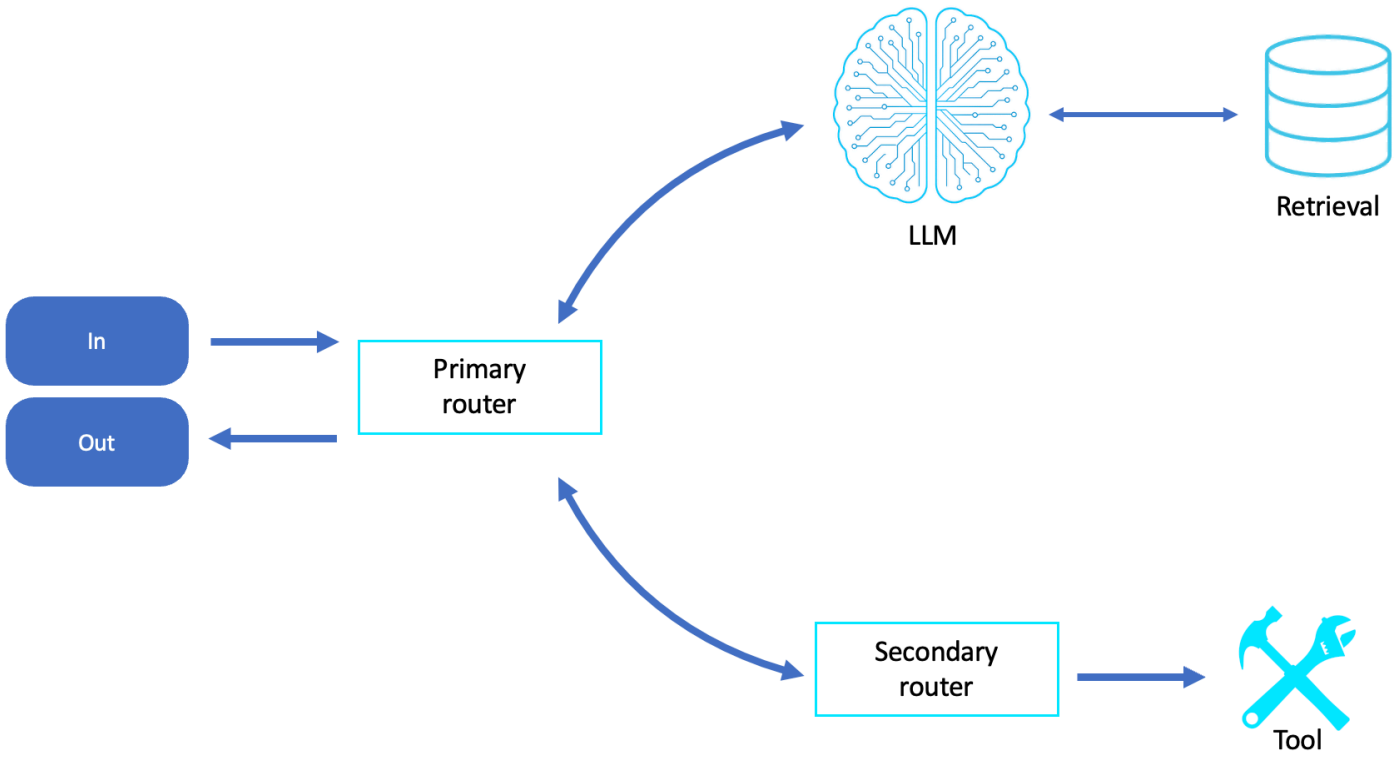
- LLM 呼叫的線性或分支鏈
- 做為結構化輸入傳遞或內嵌到後續提示中的中繼結果
- 可與 AWS Step Functions AWS Lambda或 代理程式特定的執行器協同運作

## 常用案例

- 多步驟推理任務（例如，「摘要評論重寫」）
- 研究助理合成分層輸出（例如，「搜尋擷取事實回答問題」）
- 程式碼產生管道（「產生計畫寫入程式碼測試程式碼解釋輸出」）

# 路由的工作流程

在路由模式中，分類器或路由器代理程式會使用 LLM 來解譯查詢的意圖或類別，然後將輸入路由至專門的下游任務或代理程式。




路由工作流程用於代理程式必須快速分類輸入意圖、任務類型或網域，然後將請求委派給專業子代理程式、工具或工作流程的情況。它在能力代理器中特別有用，例如作為一般助理的代理程式、企業函數的前門或跨網域面向使用者的 AI 界面。

路由在以下情況下特別有效：

- 跨各種任務分類請求（例如，搜尋、摘要、預訂、計算）。
- 輸入必須先預先處理或標準化，才能進入更專業化的工作流程。
- 不同的輸入類型（例如影像與文字、結構化與非結構化查詢）需要自訂處理。
- 客服人員扮演對話式切換板的角色，將任務委派給專業客服人員或微服務。
- 此工作流程在網域特定的 Copilot、客戶支援機器人、企業服務路由器和多模式代理程式中很常見，其中智慧分派可同時決定代理程式行為的品質和效率。

## 功能

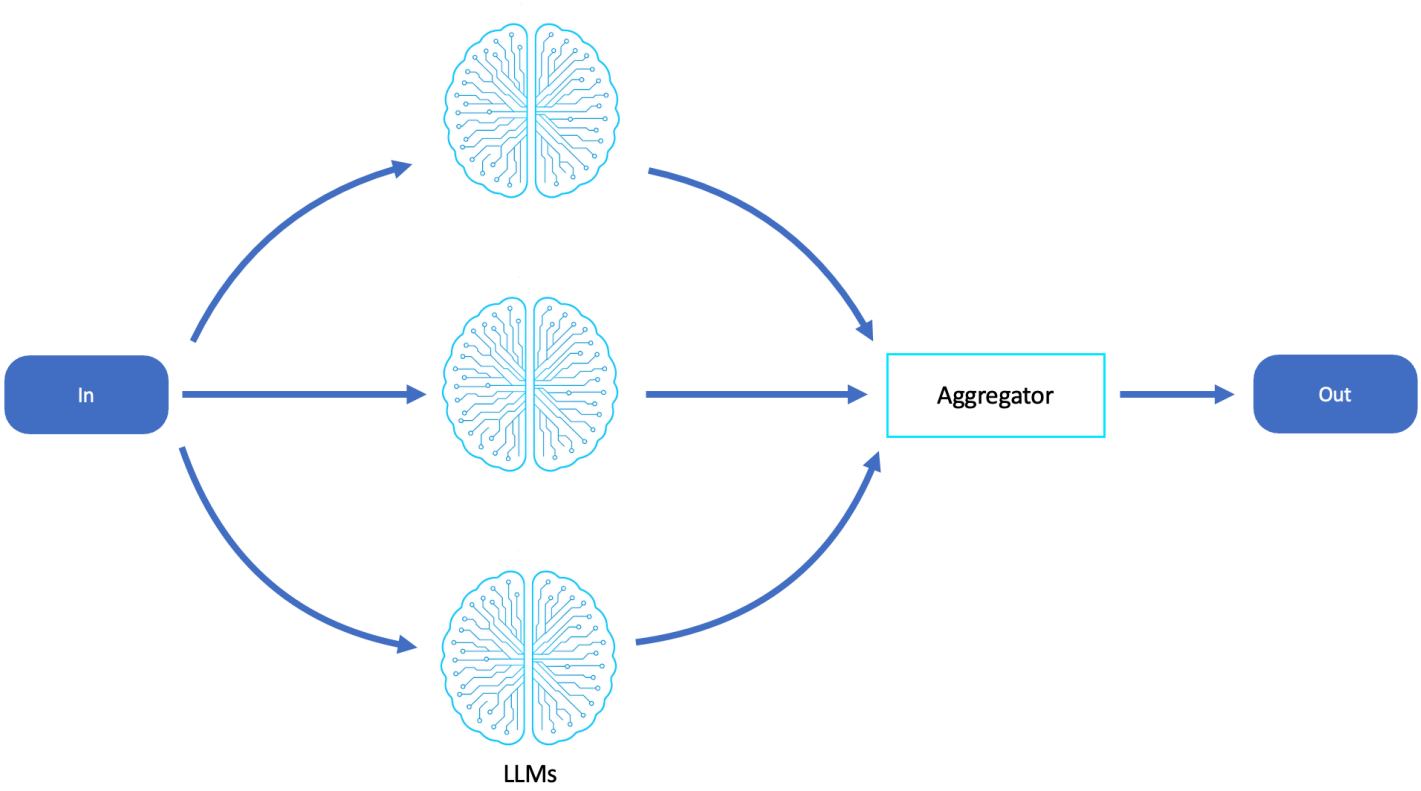
- 第一個傳遞 LLM 充當發送器
- 路由可以叫用不同的工作流程，甚至是其他代理程式模式
- 支援模組化功能擴展

## 常用案例

- 多網域助理 (「這是法律、醫療或財務問題？」)
- 使用 LLM 推理增強的決策樹
- 動態工具選擇 （例如，搜尋與程式碼產生）

## 平行化的工作流程

此工作流程涉及將任務細分為可由多個 LLM 呼叫或客服人員同時處理的獨立子任務。然後，輸出會以程式設計方式彙總並合成為結果。




當任務可分為可同時處理的獨立、非循序子任務時，會使用平行化工作流程，大幅提高效率、輸送量和可擴展性。它在大量資料、批次導向或多視角問題空間中特別強大，其中代理程式必須分析或跨多個輸入產生內容。

在下列情況下，平行化特別有效：

- 子任務不依賴彼此的中繼結果，允許它們在沒有協調的情況下平行執行。
- 任務涉及跨許多項目重複相同的推理程序 （例如，摘要多個文件或評估選項清單）。

- 並行探索多個假設或觀點，以促進多樣性、創造力或穩健性。
- 您需要透過並行 LLM 執行來減少大量或高頻率請求的延遲。
- 此工作流程通常用於文件處理代理程式、問卷或比較引擎、批次摘要器、多代理程式腦力激盪器，以及可擴展的分類或標籤任務，特別是快速、平行推理是效能優勢的情況。

### 功能

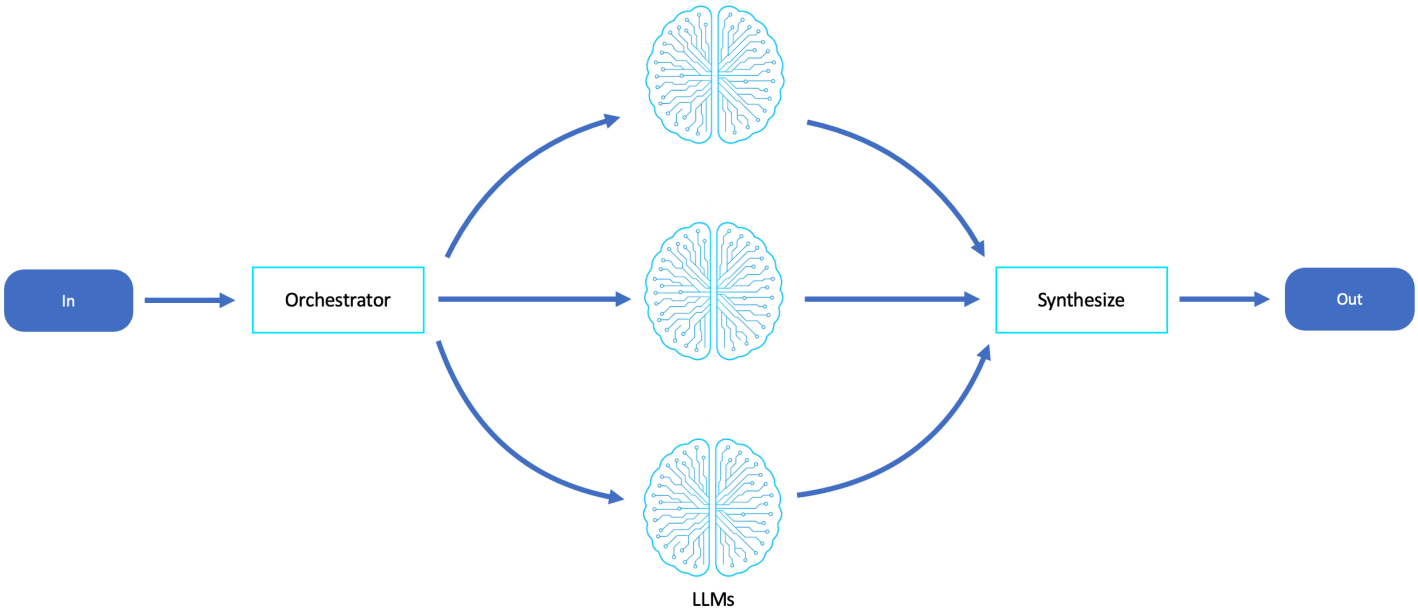
- LLM 任務的平行執行（使用 AWS Lambda AWS Fargate或 AWS Step Functions 映射狀態）
- 在合成階段需要結果對齊、驗證或重複資料刪除
- 非常適合無狀態代理程式迴圈

### 常用案例

- 平行分析多個文件或觀點
- 產生各種草案、摘要或計劃
- 加速批次任務的輸送量

## 協同運作的工作流程

中央協調器代理程式使用 LLM 來規劃、分解子任務，並將子任務委派給專門的工作者代理程式或模型，每個都具有特定的角色或網域專業知識。這反映了人工團隊結構，並支援跨多個客服人員的緊急行為。




協同運作工作流程非常適合複雜、階層式或多學科，需要結構化分解和專門執行的案例。它特別適合需要分工的任務，其中任務的不同子元件最好由具有不同功能、知識或工具集的客服人員處理。

此工作流程在下列情況下特別有效：

- 任務可以分為不同範圍、類型或推理的子任務（例如，計劃、研究、實作和測試）。
- LLM 或中繼代理程式必須協調其他代理程式、監控進度和合成結果。
- 您想要將客服人員的責任模組化，實現可擴展性、重複使用和特殊調校。
- 系統需要以角色為基礎的行為，模擬人類團隊（例如專案經理、開發人員和檢閱者）如何協同運作。

協同運作非常適合多迴轉規劃代理程式、軟體開發 Copilot、企業程序代理程式和自主專案執行器。它在實作需要集中式任務明細但分散式執行邏輯的多代理程式系統時特別有用，可跨代理程式層實現可擴展性和更可解釋的行為。

## 功能

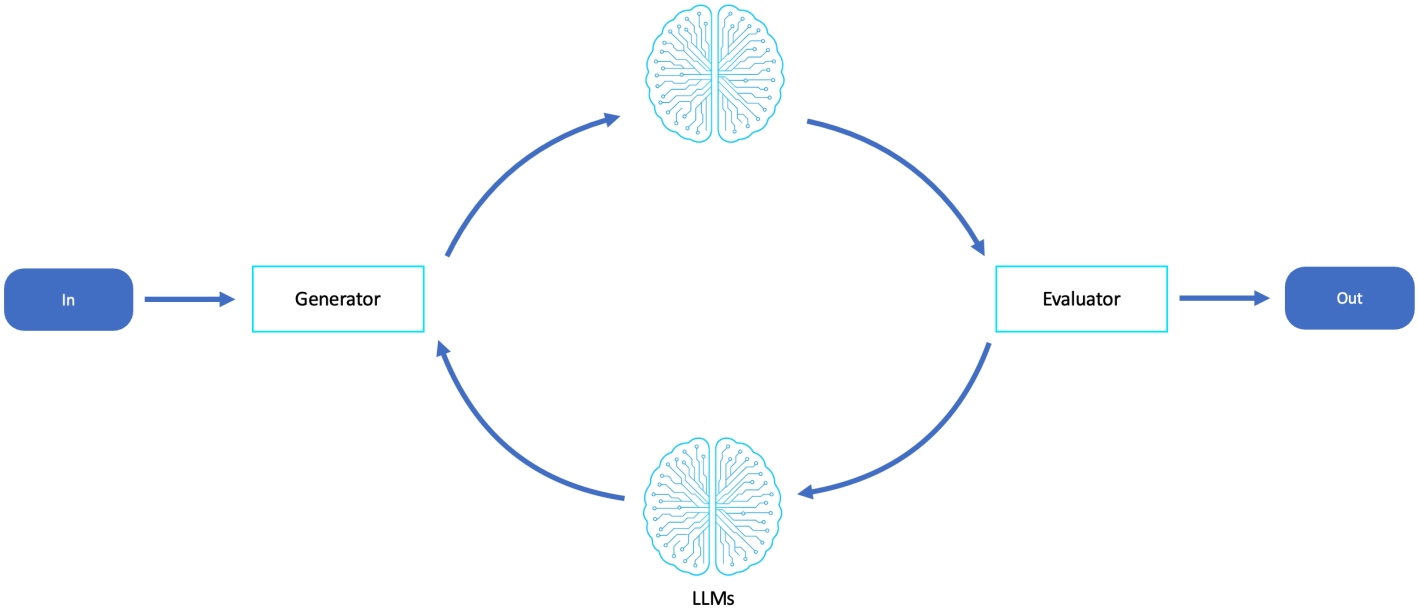
- Orchestrator 執行目標中繼關聯
- 工作者代理程式可能包含工具存取、記憶體或網域特定的提示
- 可以是階層式（即多層任務委派）

## 常用案例

- 專案經理、協調研究人員、撰寫者和品質保證代理程式
- 編碼結合規劃、執行和測試的 Copilot
- 監督工具鏈或 API 存取模式的代理程式

# 評估者和反射改進迴圈的工作流程

此工作流程提供回饋迴圈，一個 LLM 產生結果，另一個 LLM 評估或評論結果。這可提升自我反射、最佳化和反覆改進。




評估器工作流程非常適合輸出品質、準確性和一致性很重要，以及單通道產生不可靠或不足的情況。當客服人員必須自我批評、反覆運算和精簡其輸出時，此工作流程會更好，無論是符合更高的正確性標準，還是根據意見回饋探索改善的替代方案。

此工作流程在下列情況下特別有效：

- 輸出涉及主觀品質指標（例如，風格、色調和可讀性）或目標條件（例如，正確性、安全性和效能）。
- 代理程式必須透過權衡、評估限制或最佳化目標來推斷。
- 您需要內建備援和品質保證，特別是在受管制、面向客戶或創意領域。
- Human-in-the-loop審核非常昂貴或無法使用，需要自動驗證。

此工作流程用於內容產生、程式碼合成和檢閱、政策強制執行、一致性檢查、指令調校和 RAG 後製處理。它也適用於自我改善客服人員，其中持續的意見回饋有助於隨著時間的推移形成更好的回應，以建立值得信任的自動決策迴圈。

## 常用案例

- 與藍隊客服人員相比的紅隊客服人員
- 產生、評估和修訂程式碼或計劃的代理程式
- 品質保證、幻覺偵測和風格強制執行

## 功能

- 支援使用不同模型的解耦產生和評估（例如，產生 Claude 和評估 Mistral)
- 意見回饋是結構化的，用於提示修改後的輸出
- 支援多個反覆運算或收斂閾值

## 結論

LLMs提供現代軟體代理程式的認知核心，但原始模型調用不足以實現有目的、強大且可控制的智慧。若要從輸出產生移至結構化推理和目標一致行為，LLMs 必須嵌入刻意工作流程模式中，以定義模型如何處理輸入、管理內容和協調動作。

LLM 工作流程引入了建置代理程式認知模組的基礎：

- 提示鏈結會將複雜的推理分解為模組化、可稽核的步驟。
- 路由可啟用智慧型任務分類和目標委派。
- 平行化可加速輸送量並提升多樣化推理。
- 代理程式協同運作透過任務分解和角色型執行來建構多代理程式協同合作。
- 評估器（反射/反射迴圈）可啟用自我改善、品質控制和對齊檢查。

每個工作流程都代表一個可組合的模式，可以適應客服人員的需求、任務的複雜性，以及使用者的期望。這些工作流程並非互斥。它們正在建置區塊，通常合併為支援動態推理、多代理程式協調和企業級可靠性的混合架構。

當您轉換到客服人員工作流程模式的下一章時，這些 LLM 工作流程會重新顯示為大型系統中的內嵌結構，支援目標委派、工具協同運作、決策迴圈和生命週期自主性。掌握這些 LLM 工作流程對於設計軟體代理程式至關重要，不僅可以預測文字，還可以預測原因、調整和刻意採取行動。

# 代理工作流程模式

代理程式工作流程模式將模組化軟體代理程式與結構化大型語言模型 (LLM) 工作流程整合，實現自動推理和動作。雖然受到傳統無伺服器和事件驅動型架構的啟發，但這些模式會將核心邏輯從靜態程式碼轉移到 LLM 擴增代理程式，提供增強的適應性和情境決策。這種演變將傳統雲端架構從確定性系統轉換為能夠動態解譯和智慧增強的架構，同時維持可擴展性和回應性的基本原則。

本節內容

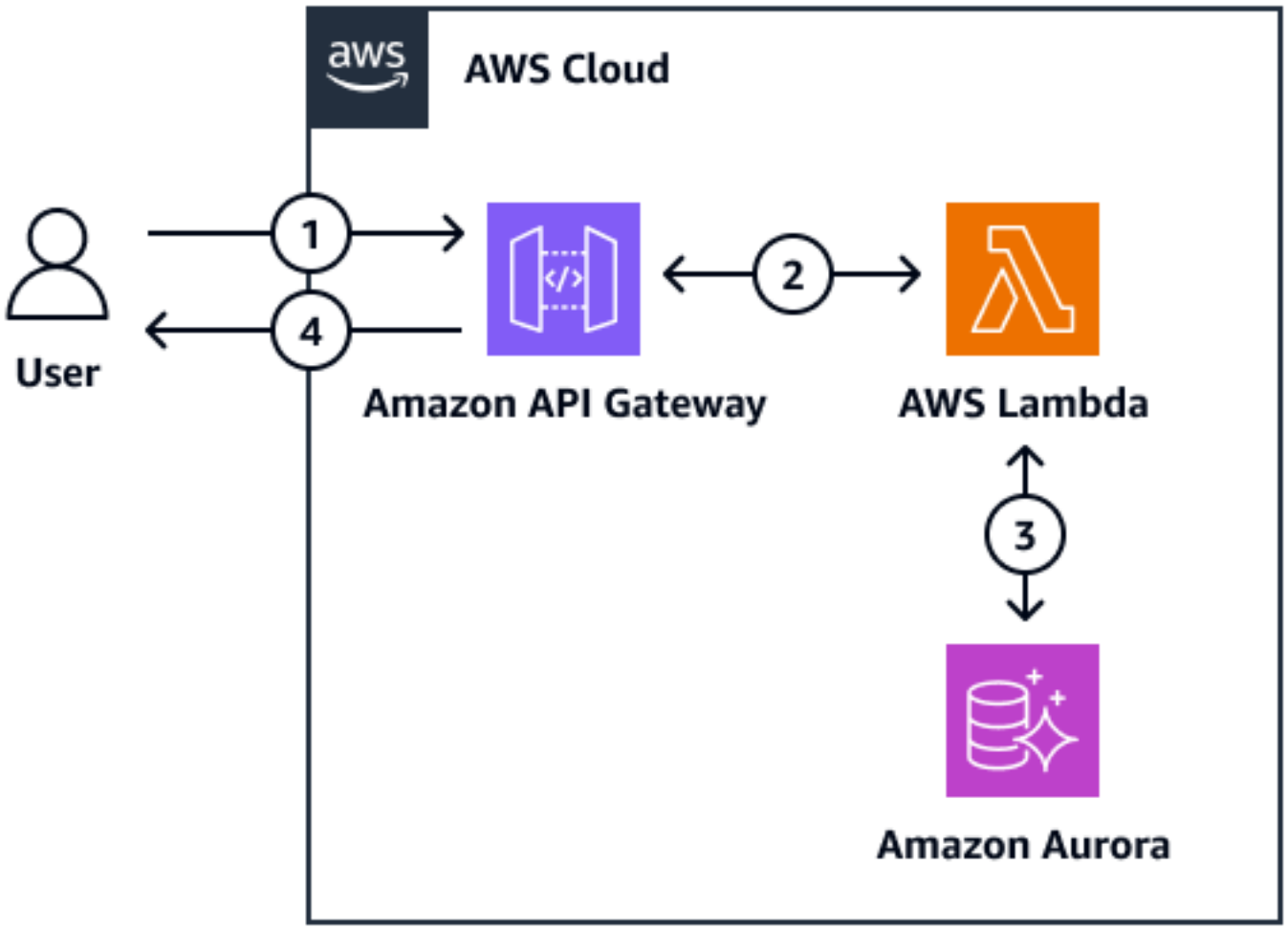


## 從事件驅動到認知擴增系統

現代雲端架構，特別是基於無伺服器和事件驅動原則的架構，傳統上依賴路由、廣發和擴充等模式來建立回應、可擴展的系統。代理式 AI 系統以這些基礎為基礎，同時圍繞 LLM 擴增推理和認知靈活性重新建構它們。這種方法允許更複雜的問題解決和自動化功能，可能會徹底改變在雲端環境中處理複雜任務的方式。

### 事件驅動型架構

下圖顯示典型的分散式系統：

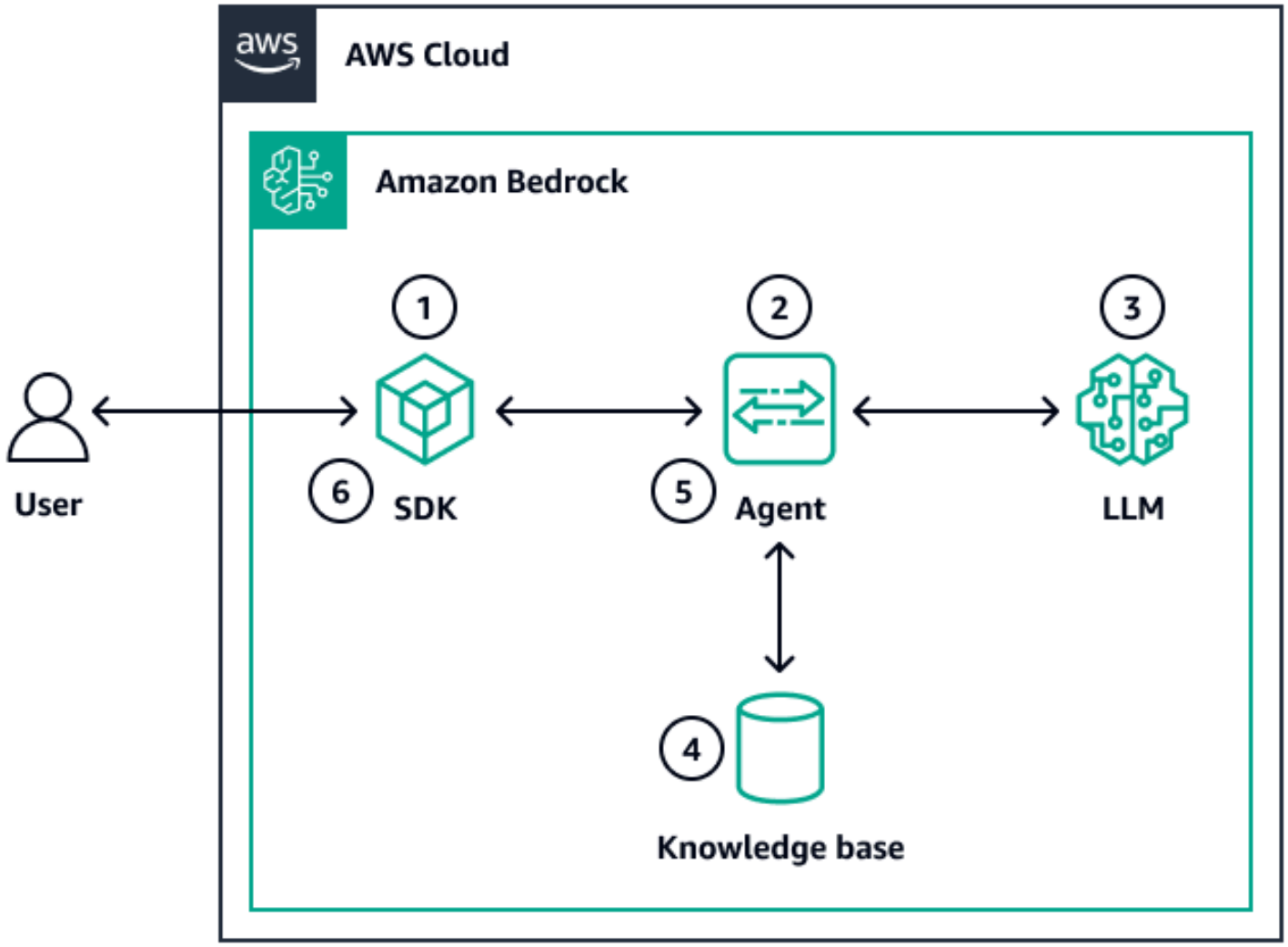



1. 使用者向 Amazon API Gateway 提交請求。
2. Amazon API Gateway 會將請求路由到 AWS Lambda 函數。
3. AWS Lambda 透過查詢 Amazon Aurora 資料庫來執行資料擴充
4. Amazon API Gateway 會將富集的承載傳回給發起人。

此結構既可靠又可擴展，但基本上是靜態的。業務規則和邏輯路徑必須明確編碼，並且適應不斷變化的內容或不完整的資訊會受到限制。

## Cognition 擴增的工作流程

代理架構會將認知增強功能新增至事件驅動型系統。下圖顯示代理程式對等項目：




1. 使用者透過 SDK 或 API 呼叫提交查詢。
2. Amazon Bedrock 代理程式會收到查詢。
3. 代理程式透過叫用 LLM 來解譯查詢
4. 代理程式透過搜尋 Amazon Bedrock 知識庫或其他外部資料來源來執行語意擴充。
5. LLM 會合成內容豐富的目標對齊回應。
6. 系統會傳回合成的回應給使用者。

在此流程中，LLM 會使用邏輯、了解意圖、擷取並結合相關內容，然後決定回應的最佳方式。此模式會反映傳統擴充模式，其中訊息會先以外部資料擴增，再進一步路由。不過，在代理系統中，此擴充不是靜態查詢。相反地，擴充是動態的、語意引導的，並由目的驅動。

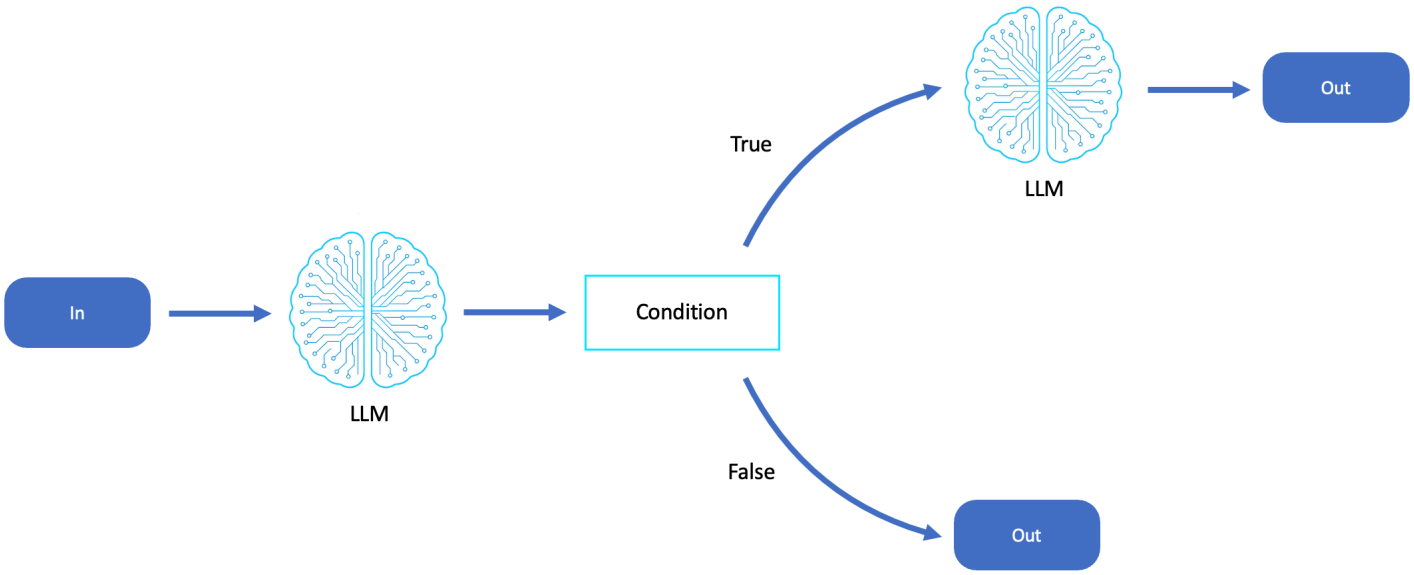
## 核心洞察

每個 LLM 工作流程都可以映射到代理工作流程模式，該模式會鏡像和發展傳統的事件驅動型架構樣式。代理程式工作流程的基本建置區塊是使用資料、工具和記憶體來增強 LLM 內容的能力。這會建立推理迴圈，以掌握資訊、適應性並與使用者意圖保持一致。當傳統系統使用查詢資料來豐富訊息時，代理系統可讓軟體運作的方式不如指令碼，更像是智慧協作者。

# 提示鏈結 saga 模式

透過將 LLM 提示鏈結重新想像為事件驅動的同義詞，我們解鎖了新的操作模型：工作流程會在自主代理程式之間進行分散式、可復原和語意協調。每個提示回應步驟都會重新建構為原子任務、發出為事件、由專用代理程式耗用，以及充實內容中繼資料。

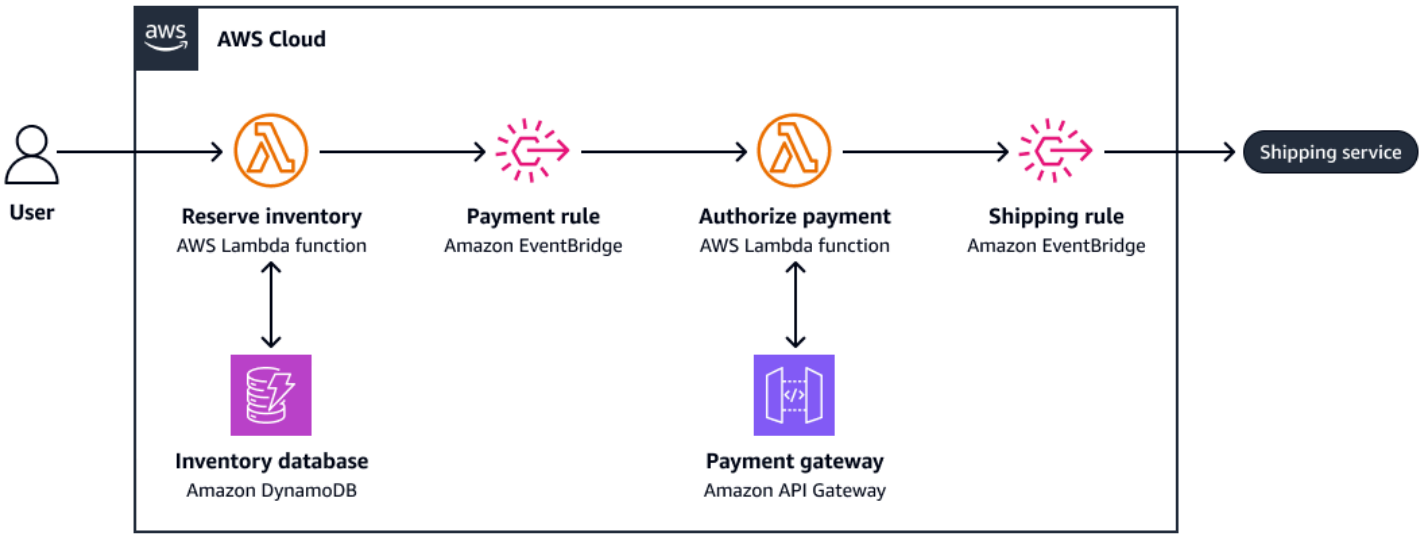
下圖是 LLM 提示鏈結的範例：




## 系列事件編排

saga 編排模式是分散式系統中沒有中央協調器的實作方法。反之，每個服務或元件都會發佈觸發下一個工作流程動作的事件。此模式廣泛用於分散式系統，以管理跨多個服務的交易。在 saga 中，系統會執行一系列協調的本機交易。如果失敗，系統會觸發補償動作以維持一致性。

下圖是 saga 編排的範例：




1. 預留庫存
2. 授權付款
3. 建立運送訂單

如果步驟 3 失敗，系統會叫用補償動作（例如取消付款或釋出庫存）。

此模式在服務鬆散耦合且狀態必須隨著時間一致地解決的事件驅動型架構中特別重要，即使存在部分故障也一樣。

## 提示鏈結模式

提示鏈結在結構和用途上都類似 saga 模式。它會執行一系列推理步驟，這些步驟會依序建置，同時保留內容並允許轉返和修訂。

## 客服人員編排

1. LLM 解譯複雜的使用者查詢並產生假設
2. LLM 詳細說明解決任務的計畫
3. LLM 執行子任務（例如，使用工具呼叫或擷取知識）
4. 如果 LLM 認為結果不滿意，則會精簡輸出或重新檢視較早的步驟

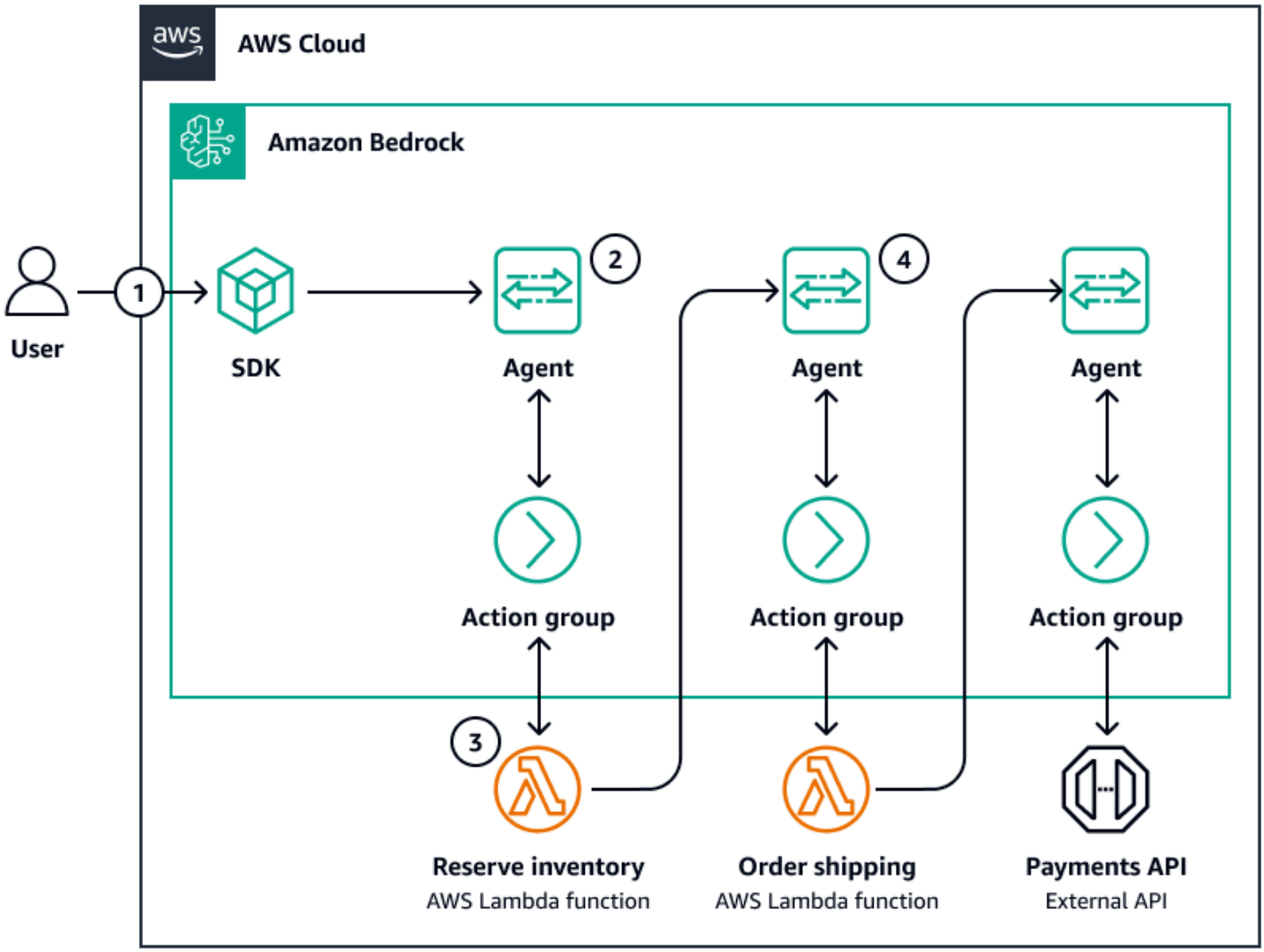
如果中繼結果有瑕疵，系統可以執行下列其中一項操作：

- 使用不同的方法重試步驟

- 還原至先前的提示並重新規劃
- 使用評估者迴圈 （例如，從評估者最佳化工具模式） 來偵測和修正失敗

如同 saga 模式，提示鏈結允許部分進度和轉返機制。透過反覆精簡和 LLM 導向的更正，而不是補償資料庫交易，就會發生這種情況。

下圖是客服人員編排的範例：




1. 使用者透過 SDK 提交查詢。
2. Amazon Bedrock 代理程式會協調推理下列各項：
   - 解譯 (LLM)
   - 規劃 (LLM)
   - 透過工具或知識庫執行

- 回應建構

3. 如果工具失敗或傳回的資料不足，代理程式可以動態重新規劃或重述任務。

4. 記憶體（例如，短期向量存放區）可以跨步驟保留其狀態

## 要點

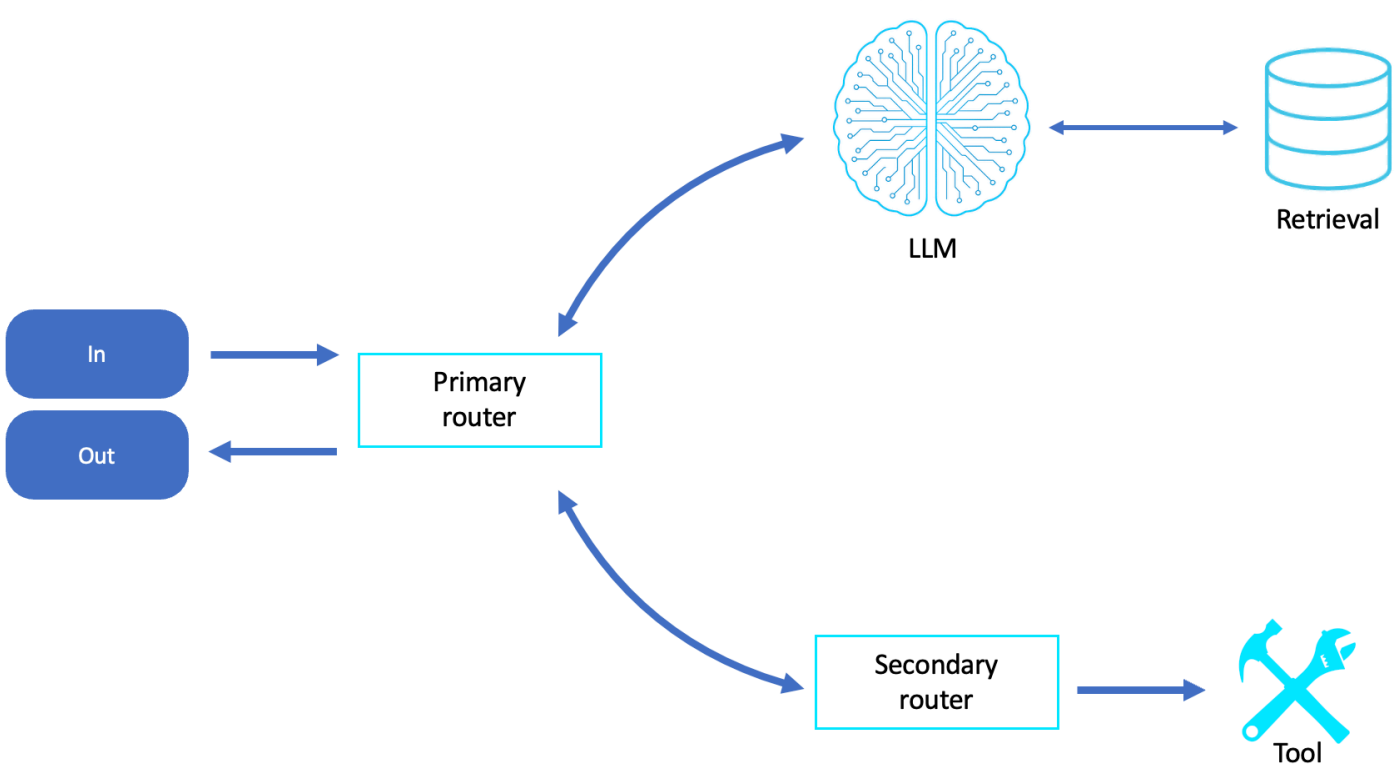
當 saga 模式使用補償邏輯管理分散式服務呼叫時，提示鏈結會使用反射排序和適應性重新規劃來管理推理任務。這兩個系統都允許累加式進度、分散式決策點和故障復原，並透過明智的推理而非嚴格復原來執行所有操作。

提示鏈結引入交易推理，這是相當於 sagas 的認知。也就是說，在更廣泛的目標導向對話中，每個「想法」都會重新評估、修訂或捨棄。

# 路由動態分派模式

在現代代理程式系統中，任務範圍從文件剖析到自動軟體產生，將請求動態路由到功能最強大的大型語言模型 (LLM) 或代理程式的能力變得至關重要。靜態路由邏輯通常內嵌在協同運作指令碼或 API 層中，缺乏即時、多模型、多功能環境所需的適應性。為了解決這個問題，LLM 路由工作流程可以轉換為事件驅動型架構，利用動態分派模式，將 LLM 呼叫轉換為智慧路由、內容感知的事件。

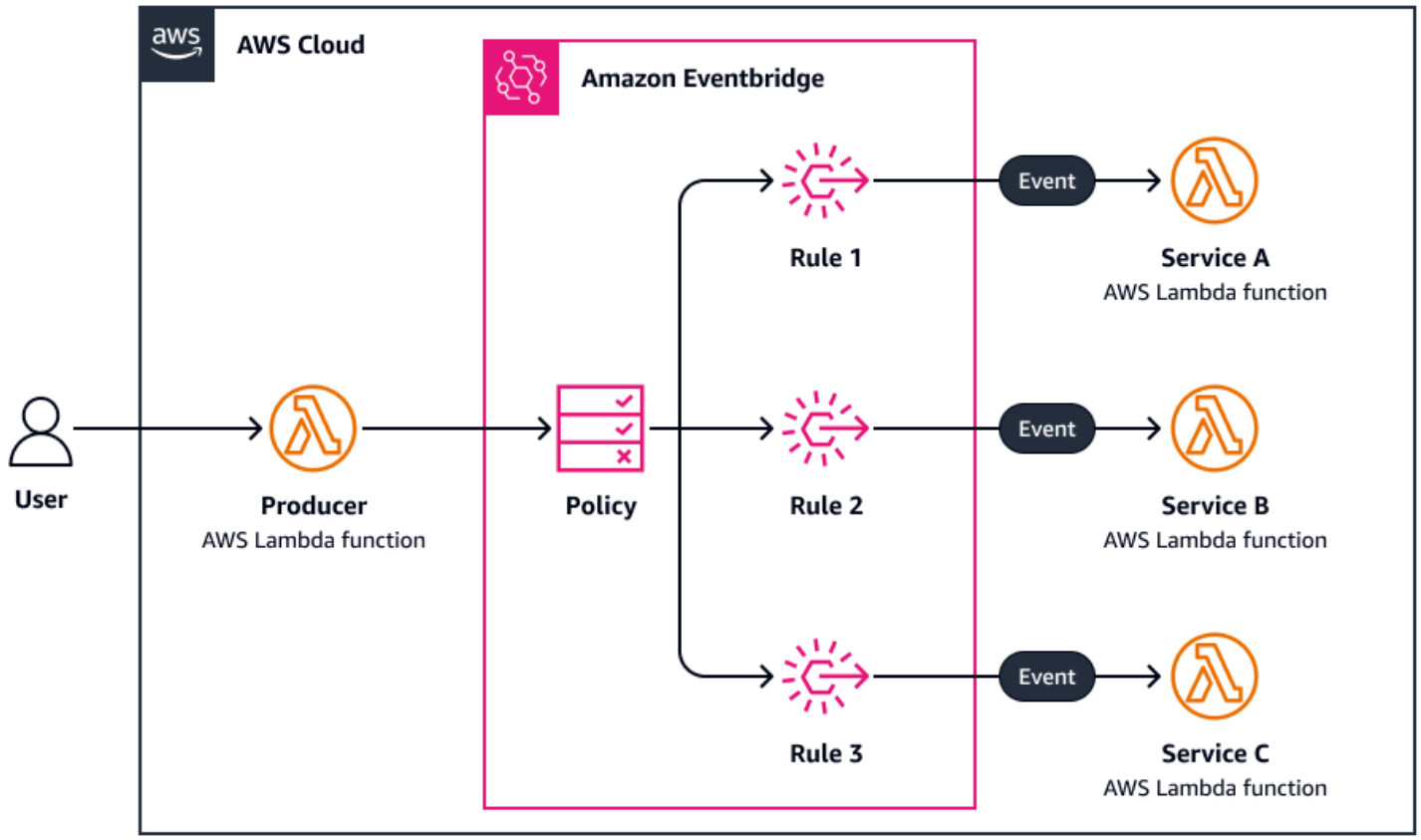
下圖是 LLM 路由的範例：




## 動態分派

在傳統分散式系統中，動態分派模式會根據傳入事件屬性，例如事件類型、來源和承載，在執行時間選取和叫用特定服務。這通常使用 Amazon EventBridge 實作，可評估傳入事件並將其路由至適當的目標（例如 AWS Lambda 函數 AWS Step Functions或 Amazon Elastic Container Service 任務）。

下圖是動態分派的範例：




1. 應用程式發出事件（例如 {"type"： "orderCreated"， "priority"： "high"}）。
2. Amazon EventBridge 會根據其路由規則評估事件。
3. 根據事件的屬性，系統會動態分派至下列項目：
   - `HighPriorityOrderProcessor`（服務 A）
   - `StandardOrderProcessor`（服務 B）
   - `UpdateOrderProcessor`（服務 C）

此模式支援鬆耦合、網域型專業化和執行時間擴充性。這可讓系統以智慧方式回應不斷變化的需求和事件語意。

## 以 LLM 為基礎的路由

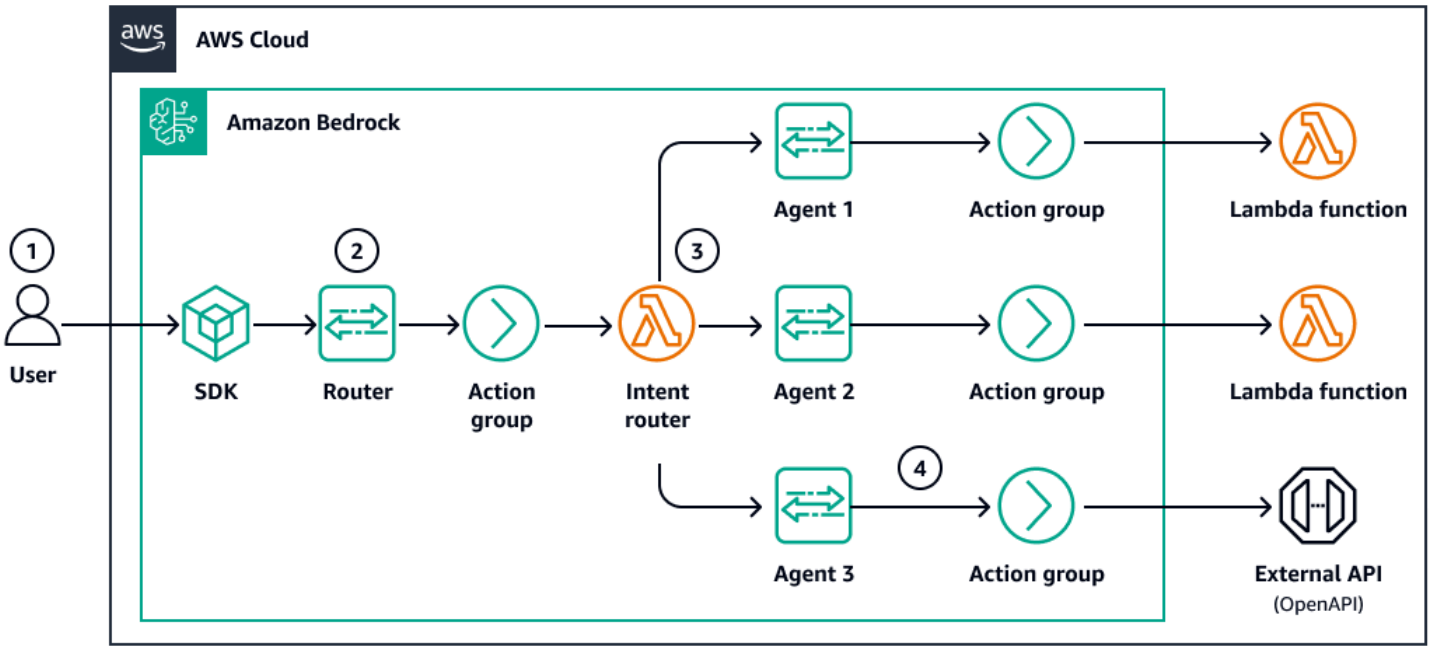
在代理系統中，路由也會執行動態任務委派，但 LLM 會透過自然語言來分類和解譯使用者的意圖，而不是 Amazon EventBridge 規則或中繼資料篩選條件。結果是靈活、語意和適應性的分派形式。

# 代理程式路由器

此架構可在沒有預先定義的結構描述或事件類型的情況下啟用豐富的意圖型分派，非常適合非結構化輸入和複雜的查詢。

1. 使用者提交請求「您可以協助我檢閱我的合約條款嗎？」
2. LLM 將此解譯為法律文件任務。
3. 代理程式會將任務路由到下列一或多個項目：
   - 合約檢閱提示範本
   - 法律推理子代理程式
   - 文件剖析工具

下圖是 代理程式路由器的範例：




1. 使用者透過 SDK 提交自然語言請求。
2. Amazon Bedrock 代理程式使用 LLM 來分類任務（例如，法律、技術或排程）。
3. 代理程式會透過動作群組動態路由任務，以叫用所需的代理程式：
   - 網域特定代理程式
   - 專用工具鏈
   - 自訂提示組態

4. 選取的處理常式會處理任務，並傳回量身打造的回應。

## 要點

當傳統動態分派使用 Amazon EventBridge 規則根據結構化事件屬性進行路由時，代理程式路由會使用 LLMs 根據意義和意圖以語意方式分類和路由任務。這可透過啟用下列項目來擴展系統的靈活性：

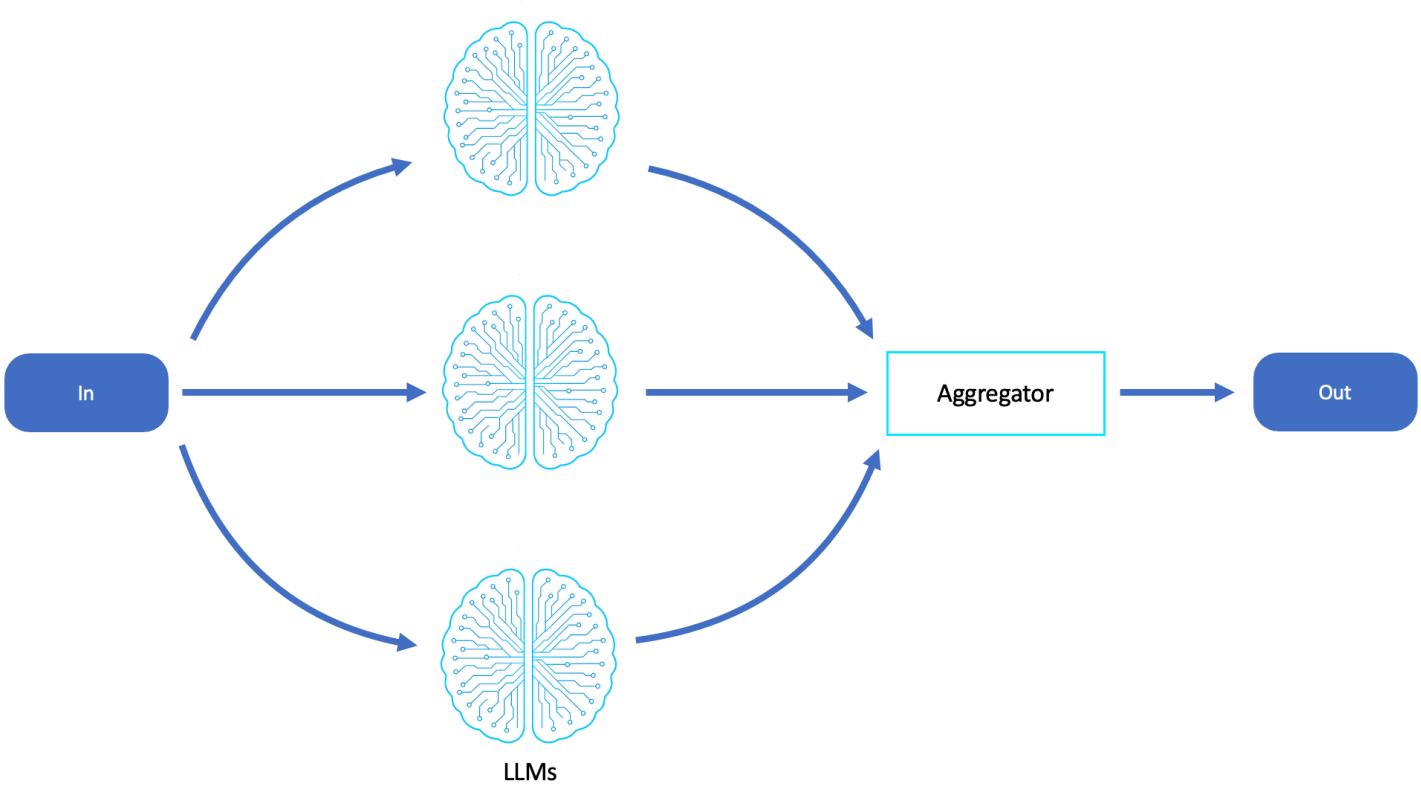
- 更廣泛的輸入理解
- 智慧型備用和工具選擇
- 透過新的客服人員角色或提示樣式的自然可擴展性

代理路由使用動態認知分派取代剛性規則，這可讓系統以語言而非程式碼演進。

# 平行化和散佈收集模式

許多進階推理和產生任務 – 例如摘要大型文件、評估多個解決方案路徑，或比較不同的觀點 – 受益於平行執行提示。需要可擴展性、回應能力和容錯能力時，傳統的循序工作流程會短暫。為了克服這個問題，可以使用事件驅動的散佈集模式來重新構想 LLM 型平行化，其中任務會動態散發到自動代理程式，並以智慧方式合成結果。

下圖是 LLM 平行化工作流程的範例：



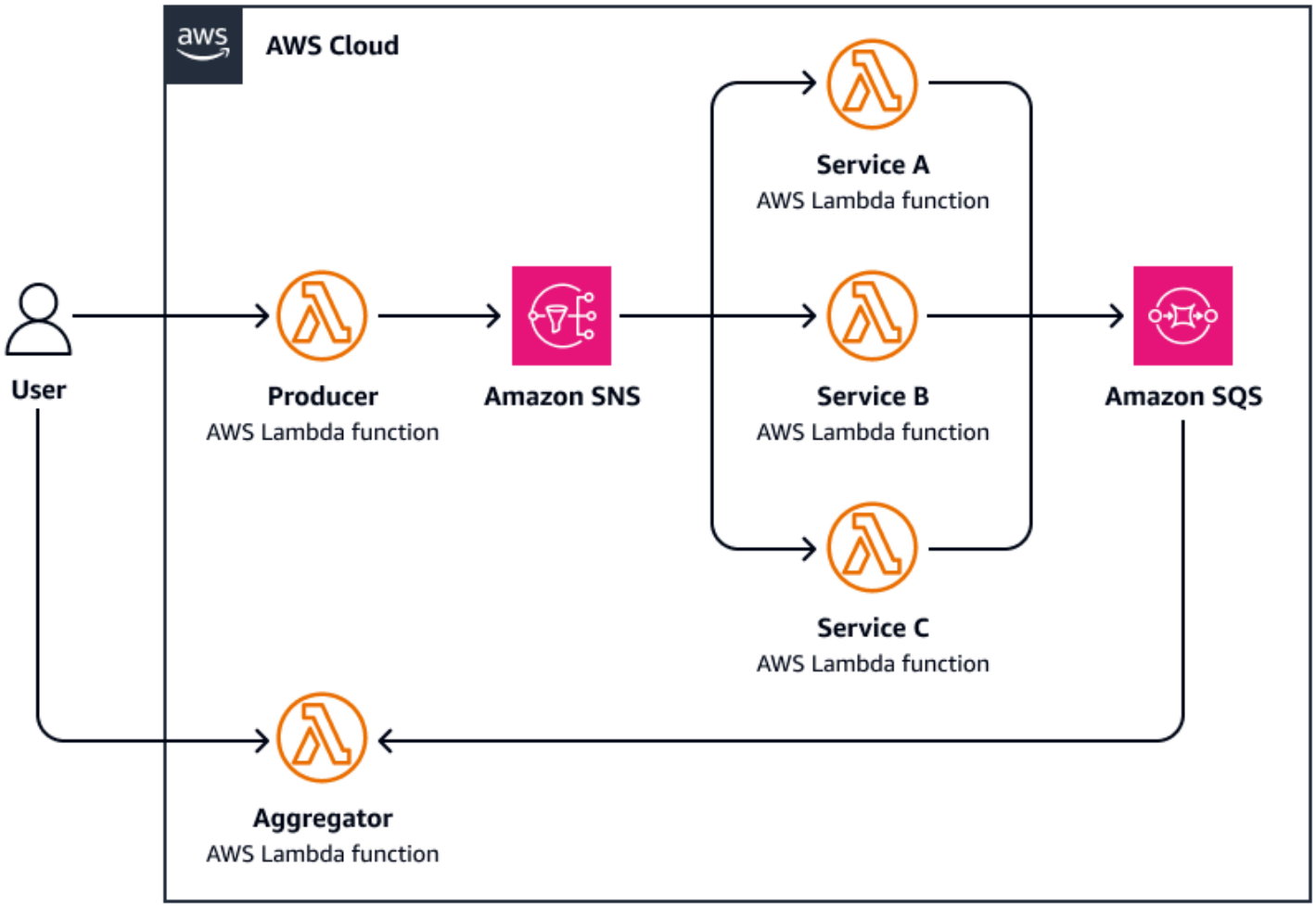

## 散佈集合

在分散式系統中，散佈收集模式會平行將任務傳送至多個服務或處理單位，等待其回應，然後將結果彙總到合併的輸出中。與廣發不同，散佈集會進行協調，因為它預期回應，通常會套用邏輯來合併、比較和選取結果。

平行化和散佈集的常見實作包括下列項目：

- AWS Step Functions 映射平行任務執行的狀態
- AWS Lambda 並行，協調多個調用函數的結果
- 具有相互關聯 IDs Amazon EventBridge
- 使用 Amazon Simple Storage Service (Amazon S3)、Amazon DynamoDB 或佇列來管理廣發並收集結果的自訂控制器模式

下圖是散佈集的範例：




1. 使用者將請求傳送至中央協調器函數，透過將平行訊息發佈至 Amazon Simple Notification Service (Amazon SNS) 主題來散佈任務。
2. 每個訊息都包含任務中繼資料，並會路由至專業工作者 AWS Lambda。
3. 每個工作者會 AWS Lambda 獨立處理其指派的子任務（例如，查詢外部 API、處理文件和分析資料）。
4. 結果會寫入常用儲存層，例如 Amazon Simple Queue Service (Amazon SQS)。
5. 彙總器函數會等待所有回應完成，然後執行下列動作：
   - 收集和彙總結果（例如，合併摘要、選取最佳相符項目）
   - 傳送最終回應或觸發下游工作流程

散佈收集模式的常見使用案例包括下列項目：

- 聯合搜尋

- 價格比較引擎
- 彙總資料分析
- 多模型推論

## LLM 型平行處理 （散佈加法認知）

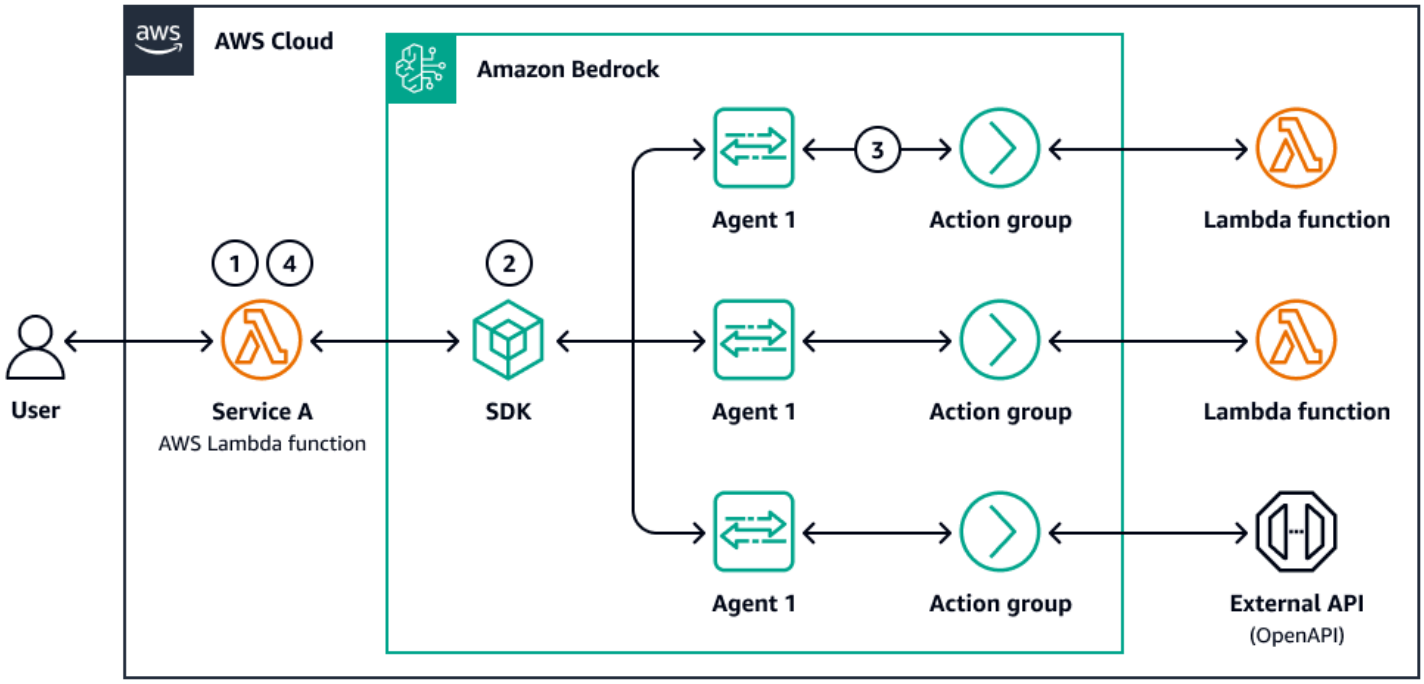
在客服人員系統中，平行處理透過將子任務分散到多個 LLM 呼叫或客服人員，緊密反映散佈集合，每個子任務都會獨立推理一部分的問題。傳回的結果會由彙總程序收集和合成，通常是另一個 LLM 或控制器代理程式。

## 代理程式平行處理

1. 客服人員提交請求「總結這 10 個報告中的洞見」。
2. 它會將報告分散至 10 個平行 LLM 摘要任務。
3. 傳回所有摘要時，代理程式會執行下列動作：
    - 將摘要彙總為統一的簡報
    - 識別主題或矛盾
    - 將合成的輸出傳送給使用者

此代理式工作流程可實現可擴展性、模組化和適應性平行推理。這非常適合需要高認知輸送量的使用案例。

下圖是代理程式平行化的範例：




1. 使用者提交分段查詢或文件集。
2. 控制器 AWS Lambda 或步驟函數會分配子任務。每個任務都會使用自己的提示叫用 Amazon Bedrock LLM 呼叫或子代理程式。
3. 當呼叫和子任務完成時，結果會儲存（例如，在 Amazon S3 或記憶體存放區中），而彙總步驟會合併、比較或篩選輸出。
4. 系統會傳回最終回應給使用者或下游代理程式。

此系統具有分散式推理迴圈，具有可追蹤性、容錯能力和選用的結果加權或選擇邏輯。

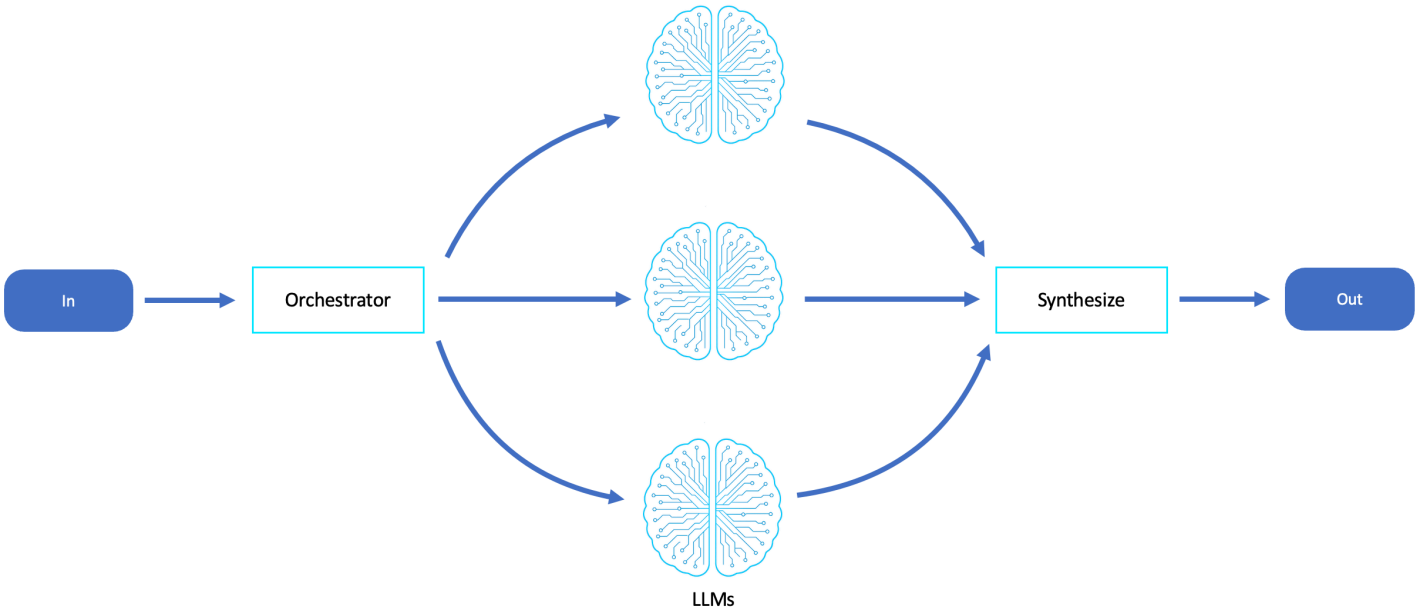
## 要點

代理程式平行處理使用散佈集合模式來分配 LLM 任務，從而實現平行處理和智慧型結果合成。

# Saga 協同運作模式

隨著 LLMs工作流程變得越來越複雜，跨越提示鏈、資料處理步驟、工具調用和客服人員協作，智慧型協同運作的需求變得至關重要。這些工作流程可以實作為事件驅動的協同運作模式，讓 LLM 型系統在自主代理程式之間動態協調、監控和調整多步驟任務，而不是依賴緊密耦合的指令碼或靜態預定執行流程。

下圖是協調器的範例：



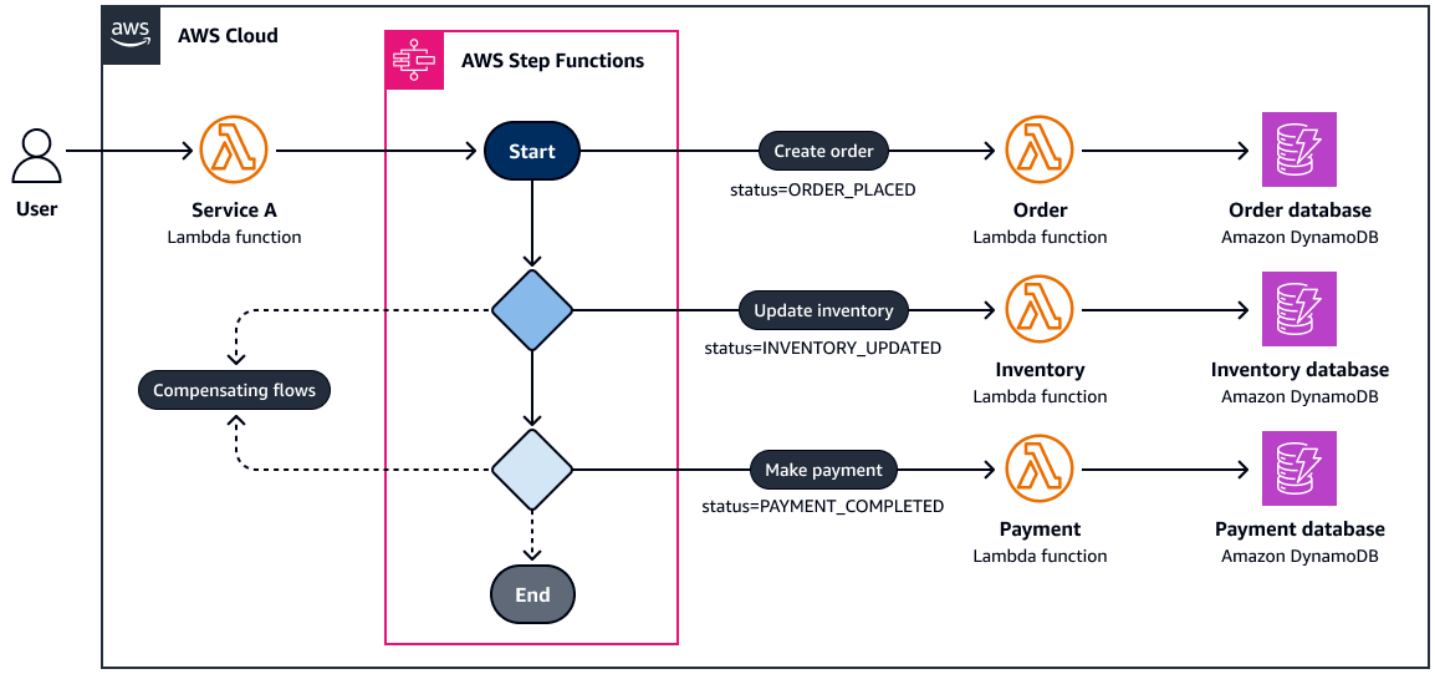

## 事件協同運作

在傳統分散式系統中，事件協同運作是指中央協調器透過明確引導多個服務或任務的控制流程來管理複雜工作流程的模式。與事件編排（每個服務都獨立反應）不同，協同運作提供集中式邏輯、可見性和對整個程序的控制。

這通常使用下列工具實作：

- AWS Step Functions – 定義和執行具狀態工作流程
- AWS Lambda – 在協調流程中執行離散任務
- Amazon SQS 或 Amazon EventBridge – 觸發非同步步驟或回應

下圖是 saga 協同運作的範例：




AWS Step Functions 工作流程會管理客戶訂單程序：

1. 建立順序 (AWS Lambda)
2. 更新庫存 (AWS Lambda)
3. 進行付款 (AWS Lambda)

協調器透過管理重試、平行分支、逾時和失敗來協調每個步驟。

## 角色型代理程式系統 （ 協調器 ）

在代理程式系統中，協調程式模式會鏡射事件協同運作，但會將邏輯分散到多個推理代理程式，每個代理程式都有定義的角色或專業。中央協調器代理程式會解譯整體任務、將其分解為子任務，並將這些任務委派給工作者代理程式，每個代理程式都針對特定網域進行最佳化 （ 例如，研究、編碼、摘要、檢閱 ）。

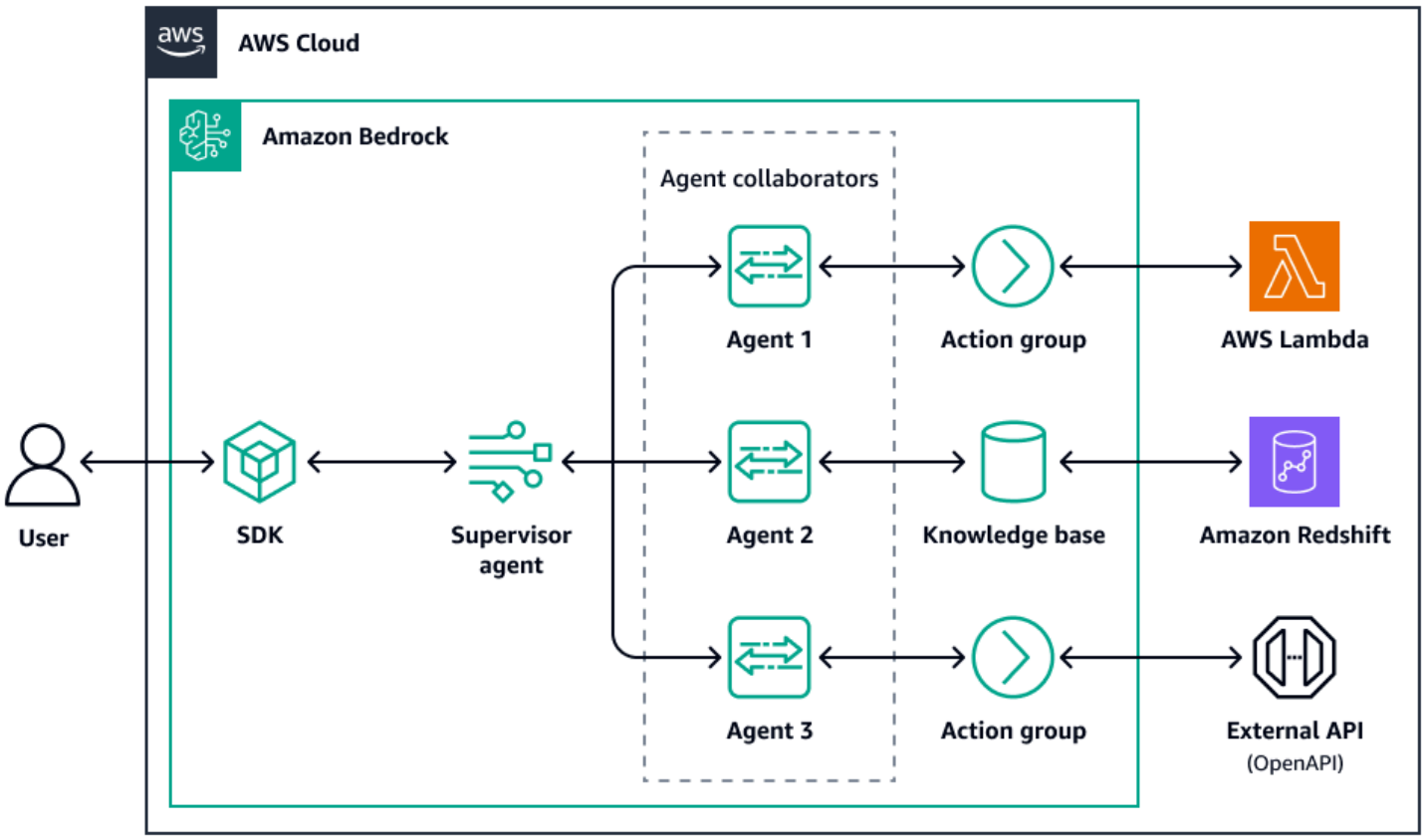
## 主管

1. 使用者提交查詢「建立專案簡介並總結前 5 個競爭對手」。
2. 協調器代理程式會執行下列動作：
   - 指派研究代理程式來尋找競爭對手資料
   - 將原始問題清單傳送至摘要代理程式

- 將結果傳遞給簡短寫入器代理程式
- 編譯使用者的最終輸出

每個代理程式都會獨立運作，但協調器會協調任務。這就像處理工作流程任務的 Lambda 函數。

下圖是主管的範例：




1. 使用者提交任務給 Amazon Bedrock 主管代理程式。
2. 主管客服人員會將請求剖析為每個客服人員協作者的子任務。
3. 每個子任務都會指派給具有角色特定提示或工具鏈的協作者代理程式。
4. 工作者代理程式會透過動作群組呼叫外部 APIs 或工具。
5. 每個工作者代理程式會以結構化格式傳回輸出。
6. 當所有工作者都傳回其結果時，主管會評估、合成並傳回最終回應。

此結構允許跨複雜的多步驟代理程式工作流程進行模組化、適應性和自我檢查。

## 要點

當事件協同運作使用集中式控制（例如， AWS Step Functions) 來引導服務執行時，角色型代理程式系統會使用 LLM 支援的協同程式代理程式來推斷目標、將子任務委派給工作者代理程式，以及合成最終輸出。

在這兩個範例中，協調器會執行下列動作：

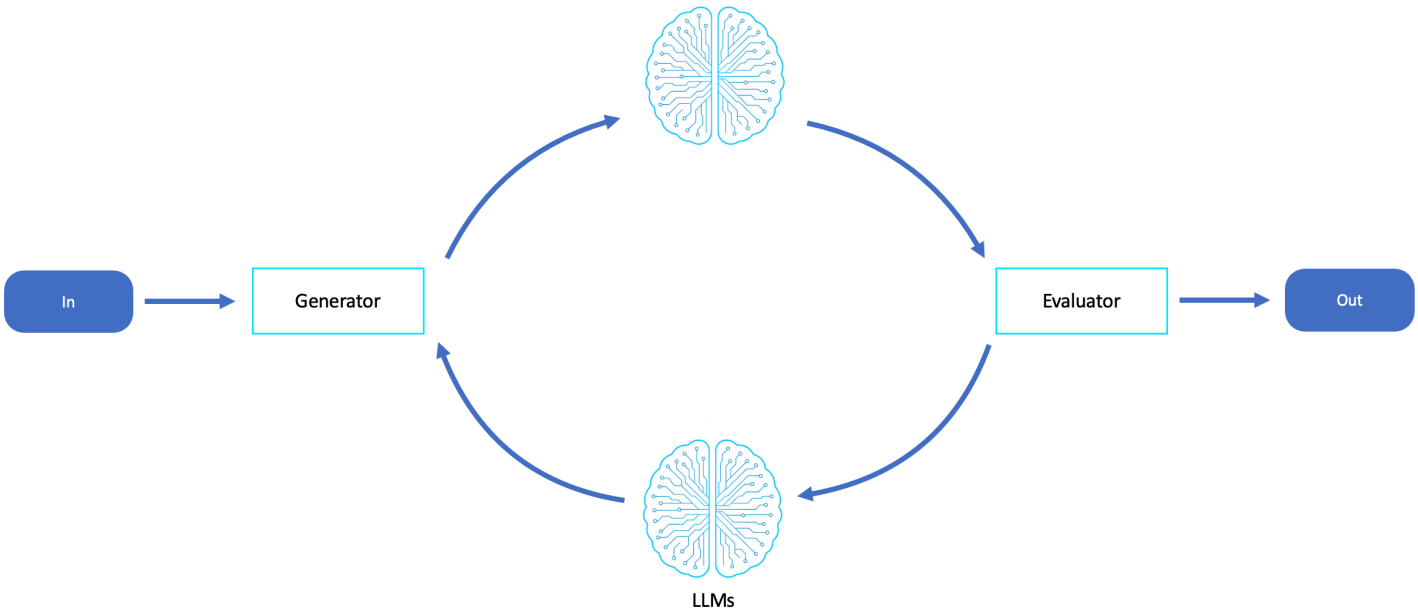
- 維護內容和執行流程
- 處理分支、排序和錯誤處理
- 從分散式元件產生統一的結果

不過，代理程式協同運作會新增推理、適應性和語意委派。這使得它非常適合開放式、模棱兩可和不斷發展的任務。

# 評估器反射-改進迴圈模式

程式碼產生、摘要或自動決策等任務從執行階段意見回饋中獲益良多，讓系統能夠透過觀察和改進進行發展。若要操作此項目，反射改善週期可以實作為事件驅動型意見回饋控制迴圈，這是受系統工程啟發的模式，適用於自動化、智慧型工作流程。

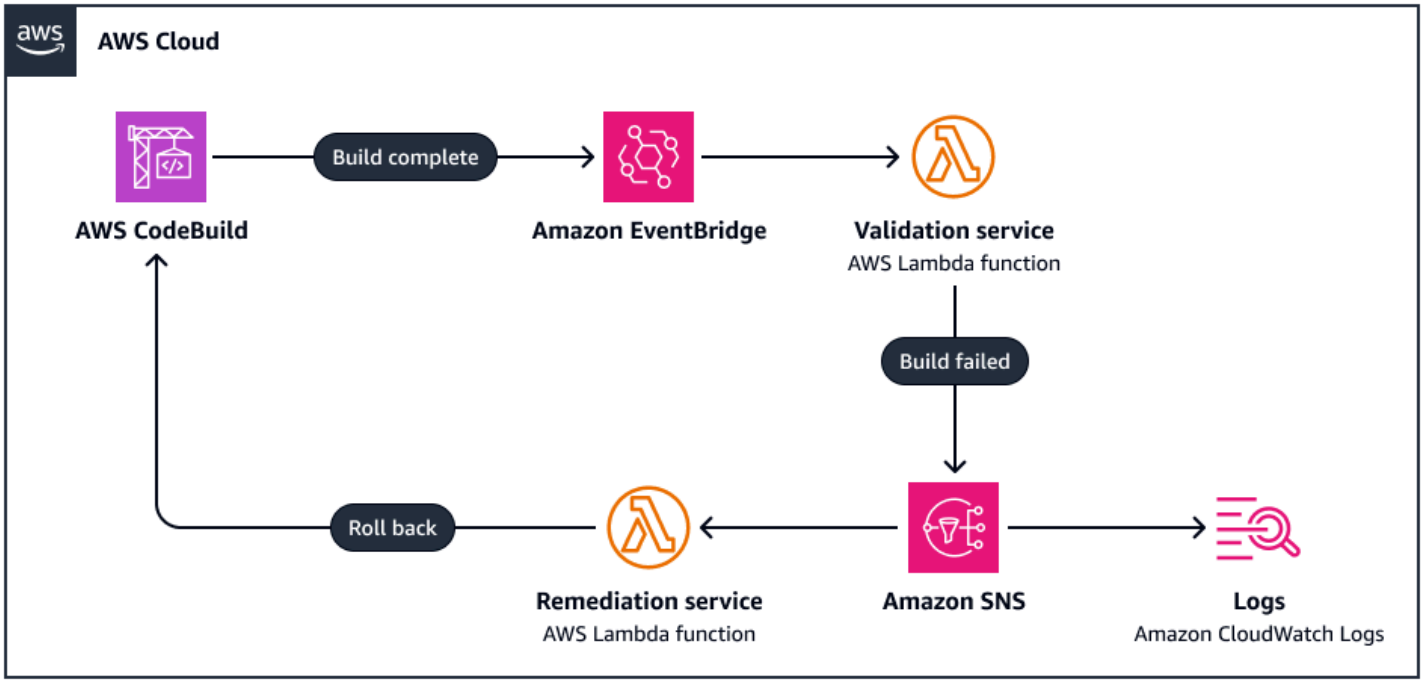
下圖是評估器反射改善意見回饋迴圈的範例：



# 回饋控制迴圈

回饋控制迴圈是一種模式，可監控自己的輸出和行為、根據定義的條件或所需狀態進行評估，然後相應地調整其動作。此架構受到控制理論的啟發，是自動化、持續整合和持續交付 (CI/CD) 管道和機器學習操作等領域的基礎。

下圖是回饋控制迴圈的範例：




1. 部署管道會發出 buildComplete 事件。
2. 事件會觸發驗證建置的自動化測試或評估任務。
3. 如果驗證失敗（例如，由於測試失敗、安全問題或違反政策），系統：
   - 發出 buildComplete 事件
   - 記錄問題或傳送通知
   - 觸發修復或修正動作，例如復原、修補或重試

迴圈會持續進行，直到產生可接受的結果或呈報，或發生逾時為止。此模式通常用於下列項目：

- 將事件路由至評估或修復任務的 Amazon EventBridge 規則
- AWS Step Functions 用於反覆重試邏輯和評估結果的分支
- 用於意見回饋觸發和警示的 Amazon Simple Notification Service (Amazon SNS) 或 Amazon CloudWatch 警示

- AWS Lambda 套用修正動作的 函數或容器化工作者

## 回饋控制迴圈 （評估器）

評估器工作流程是由 LLMs或推理代理程式提供支援的認知回饋迴圈。此程序包含下列項目：

1. 產生器代理程式或 LLM 會產生輸出 （例如，計劃、回答或草稿）。
2. 評估者代理程式會使用 critique 提示或評估 rubric 來檢閱結果。
3. 根據意見回饋，原始代理程式或新的最佳化工具代理程式會修改輸出。

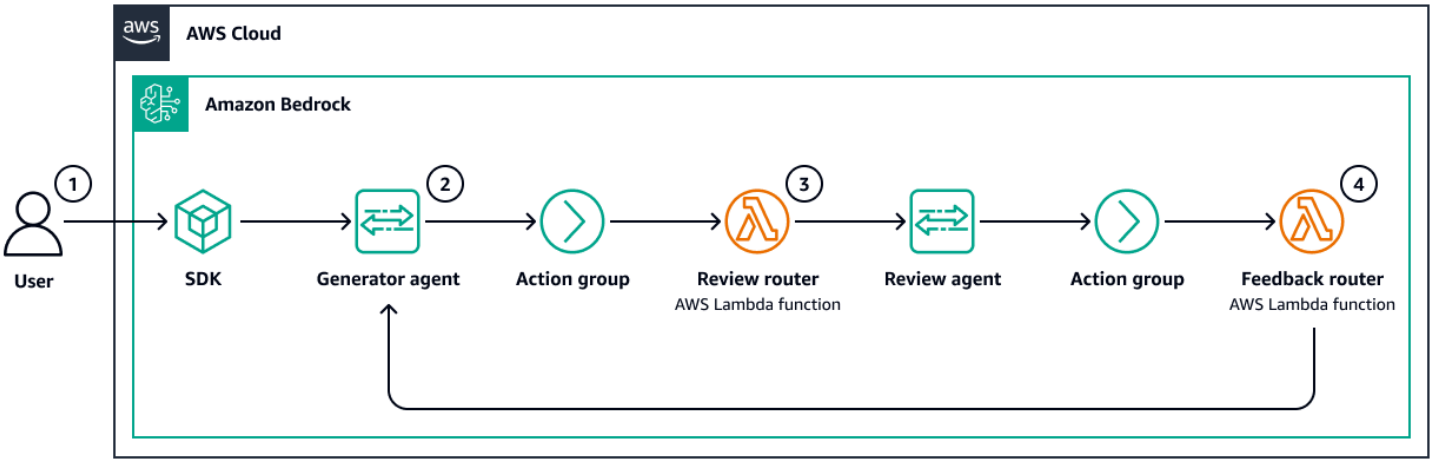
迴圈會重複，直到結果符合一組條件、獲得核准或達到重試限制為止。

## 評估者

1. 使用者要求客服人員撰寫政策摘要。
2. 產生器代理程式會草擬它。
3. 評估者代理程式會檢查涵蓋範圍、語氣和法律正確性。
4. 如果回應不足，則會進行微調並重新提交，直到回饋迴圈收斂為止。

這可啟用自我評估、反覆精簡和調適性輸出控制，無需人工輸入。

下圖是回饋控制迴圈 （評估者） 的範例：




1. 使用者發出任務 （例如，草擬業務策略）。
2. Amazon Bedrock 代理程式會使用 LLM 產生初始草稿。

3. 第二個客服人員（或後續提示）會執行結構化評估（例如，「透過清晰度、完整性和音調來評分此輸出」)。
4. 如果評分低於閾值，回應的修訂方式如下：
   - 使用內嵌的 critique 叫用產生器
   - 將意見回饋傳送至專門的精簡器代理程式
   - 反覆運算，直到達到可接受的回應為止

AWS Lambda 控制器或 等選用元件 AWS Step Functions 可以管理意見回饋閾值、重試和備用策略。

## 要點

當傳統回饋控制迴圈使用事件、指標和修復邏輯來驗證和調整系統行為時，代理程式評估程式迴圈會使用推理代理程式來動態評估、反映和修訂輸出。

在這兩個範例中：

- 輸出會在產生後進行評估
- 根據意見回饋觸發修正或精簡動作
- 系統會持續適應目標品質或目標

代理程式版本會將靜態驗證轉換為語意反射，讓自我改善的代理程式能夠評估自己的有效性。

# 在 上設計代理程式工作流程 AWS

本指南中的每個模式都可以使用 建置 AWS 服務。Amazon Bedrock 代理程式提供協同運作、資料存取和互動管道。

| 元件 | AWS 服務 | 用途 |
| --- | --- | --- |
| LLM 推理 | Amazon Bedrock | 代理程式邏輯、規劃、工具使用 |
| 工具執行 | AWS Lambda、Amazon ECS、Amazon SageMaker | 託管客服人員的外部工具 |

| | | |
|---|---|---|
| 記憶體和 RAG | Amazon Bedrock 知識庫、Amazon S3、OpenSearch | 持久性和語意記憶體 |
| 協同運作 | AWS Step Functions | 多步驟任務和客服人員協調 |
| 事件路由 | Amazon EventBridge、Amazon SQS | 解耦代理程式傳訊 |
| 使用者界面 | Amazon API Gateway、 AWS AppSync SDK | 應用程式或系統的進入點 |
| 監控 | Amazon CloudWatch， AWS X-Ray， AWS Distro for OpenTelemetry | 可觀測性和代理程式自我檢查 |

# 結論

代理工作流程模式是事件驅動架構的下一個進化階段，其中業務邏輯不是靜態定義的，而是使用大型語言模型 (LLM) 增強的認知動態推理。透過將傳統雲端原生基本概念與 LLM 工作流程和代理程式設計模式結合，組織可以建置適應性、智慧型和模組化的系統，以目標回應並從經驗中學習。

在這些模式中，Amazon Bedrock 是代理程式認知的閘道，允許 LLM 型代理程式存取事件工作流程、與工具和記憶體互動，以及提供結構化、可追蹤且一致的結果。

當您設計和部署代理程式系統時，這些工作流程模式會提供藍圖，用於建置自主、可編寫的 AI 架構。這些系統是以無伺服器最佳實務為基礎，並以智慧型基礎模型增強。

# 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，但需要比主動-被動遷移更多的工作。

主動-被動式遷移

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

彙總函數

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

AI

請參閱人工智慧。

AIOps

請參閱人工智慧操作。

匿名化

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

反模式

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

應用程式控制

一種安全方法，僅允許使用核准的應用程式，以協助保護系統免受惡意軟體攻擊。

應用程式組合

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

人工智慧 (AI)

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

人工智慧操作 (AIOps)

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

非對稱加密

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

原子性、一致性、隔離性、耐久性 (ACID)

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

屬性型存取控制 (ABAC)

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

授權資料來源

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

可用區域

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

AWS 雲端採用架構 (AWS CAF)

的指導方針和最佳實務架構 AWS ，可協助組織制定高效且有效的計劃，以成功地移至雲端。 AWS CAF 將指導方針組織到六個重點領域：業務、人員、治理、平台、安全和營運。業務、人員和控管層面著重於業務技能和程序；平台、安全和操作層面著重於技術技能和程序。例如，人員層面針對處理人力資源 (HR)、人員配備功能和人員管理的利害關係人。為此， AWS CAF 為人員開發、訓練和通訊提供指引，協助組織做好成功採用雲端的準備。如需詳細資訊，請參閱 AWS CAF 網站和 AWS CAF 白皮書。

AWS 工作負載資格架構 (AWS WQF)

一種工具，可評估資料庫遷移工作負載、建議遷移策略，並提供工作預估值。 AWS WQF 隨附於 AWS Schema Conversion Tool (AWS SCT)。它會分析資料庫結構描述和程式碼物件、應用程式程式碼、相依性和效能特性，並提供評估報告。

# B

錯誤的機器人

旨在中斷或傷害個人或組織的機器人。

BCP

請參閱業務持續性規劃。

行為圖

資源行為的統一互動式檢視，以及一段時間後的互動。您可以將行為圖與 Amazon Detective 搭配使用來檢查失敗的登入嘗試、可疑的 API 呼叫和類似動作。如需詳細資訊，請參閱偵測文件中的行為圖中的資料。

大端序系統

首先儲存最高有效位元組的系統。另請參閱 Endianness。

二進制分類

預測二進制結果的過程 (兩個可能的類別之一)。例如，ML 模型可能需要預測諸如「此電子郵件是否是垃圾郵件？」等問題 或「產品是書還是汽車？」

Bloom 篩選條件

一種機率性、記憶體高效的資料結構，用於測試元素是否為集的成員。

藍/綠部署

一種部署策略，您可以在其中建立兩個不同但相同的環境。您可以在一個環境 （藍色） 中執行目前的應用程式版本，並在另一個環境 （綠色） 中執行新的應用程式版本。此策略可協助您快速復原，並將影響降至最低。

機器人

透過網際網路執行自動化任務並模擬人類活動或互動的軟體應用程式。有些機器人有用或有益，例如在網際網路上編製資訊索引的 Web 爬蟲程式。某些其他機器人稱為惡意機器人，旨在中斷或傷害個人或組織。

殭屍網路

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

分支

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

碎片存取

在特殊情況下，以及透過核准的程序，讓使用者快速取得他們通常無權存取 AWS 帳戶 之 的存取權。如需詳細資訊，請參閱 Well-Architected 指南中的 AWS 實作打破玻璃程序指標。

棕地策略

環境中的現有基礎設施。對系統架構採用棕地策略時，可以根據目前系統和基礎設施的限制來設計架構。如果正在擴展現有基礎設施，則可能會混合棕地和綠地策略。

緩衝快取

儲存最常存取資料的記憶體區域。

業務能力

業務如何創造價值 (例如，銷售、客戶服務或營銷)。業務能力可驅動微服務架構和開發決策。如需詳細資訊，請參閱在 AWS上執行容器化微服務白皮書的圍繞業務能力進行組織部分。

業務連續性規劃 (BCP)

一種解決破壞性事件 (如大規模遷移) 對營運的潛在影響並使業務能夠快速恢復營運的計畫。

# C

CAF

請參閱AWS 雲端採用架構。

Canary 部署

版本對最終使用者的緩慢和增量版本。當您有信心時，您可以部署新版本並完全取代目前的版本。

CCoE

請參閱 Cloud Center of Excellence。

CDC

請參閱變更資料擷取。

變更資料擷取 (CDC)

追蹤對資料來源 (例如資料庫表格) 的變更並記錄有關變更的中繼資料的程序。您可以將 CDC 用於各種用途，例如稽核或複寫目標系統中的變更以保持同步。

混沌工程

故意引入故障或破壞性事件，以測試系統的彈性。您可以使用 AWS Fault Injection Service (AWS FIS) 來執行實驗，為您的 AWS 工作負載帶來壓力，並評估其回應。

CI/CD

請參閱持續整合和持續交付。

分類

有助於產生預測的分類程序。用於分類問題的 ML 模型可預測離散值。離散值永遠彼此不同。例如，模型可能需要評估影像中是否有汽車。

用戶端加密

在目標 AWS 服務 接收資料之前，在本機加密資料。

雲端卓越中心 (CCoE)

一個多學科團隊，可推動整個組織的雲端採用工作，包括開發雲端最佳實務、調動資源、制定遷移時間表以及領導組織進行大規模轉型。如需詳細資訊，請參閱 AWS 雲端 企業策略部落格上的 CCoE 文章。

雲端運算

通常用於遠端資料儲存和 IoT 裝置管理的雲端技術。雲端運算通常連接到邊緣運算技術。

雲端操作模型

在 IT 組織中，用於建置、成熟和最佳化一或多個雲端環境的操作模型。如需詳細資訊，請參閱建置您的雲端操作模型。

採用雲端階段

組織在遷移至 時通常會經歷的四個階段 AWS 雲端：

- 專案 – 執行一些與雲端相關的專案以進行概念驗證和學習用途
- 基礎 – 進行基礎投資以擴展雲端採用 (例如，建立登陸區域、定義 CCoE、建立營運模型)

- 遷移 – 遷移個別應用程式
- 重塑 – 優化產品和服務，並在雲端中創新

這些階段由 Stephen Orban 在部落格文章 The Journey Toward Cloud-First 和企業策略部落格上的採用階段中定義。 AWS 雲端 如需有關它們如何與 AWS 遷移策略關聯的資訊，請參閱遷移整備指南。

CMDB

請參閱組態管理資料庫。

程式碼儲存庫

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

冷快取

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

冷資料

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

電腦視覺 (CV)

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

組態偏離

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

組態管理資料庫 (CMDB)

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

一致性套件

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

持續整合和持續交付 (CI/CD)

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

CV

請參閱電腦視覺。

# D

靜態資料

網路中靜止的資料，例如儲存中的資料。

資料分類

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

資料偏離

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

傳輸中的資料

在您的網路中主動移動的資料，例如在網路資源之間移動。

資料網格

架構架構，提供分散式、分散式資料擁有權與集中式管理。

資料最小化

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

資料周邊

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

資料預先處理

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

資料來源

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

資料主體

正在收集和處理資料的個人。

資料倉儲

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

資料庫定義語言 (DDL)

用於建立或修改資料庫中資料表和物件之結構的陳述式或命令。

資料庫處理語言 (DML)

用於修改 (插入、更新和刪除) 資料庫中資訊的陳述式或命令。

DDL

請參閱資料庫定義語言。

深度整體

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

深度學習

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

深度防禦

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

委派的管理員

在 中 AWS Organizations，相容的服務可以註冊 AWS 成員帳戶，以管理組織的帳戶和管理該服務的許可。此帳戶稱為該服務的委派管理員。如需詳細資訊和相容服務清單，請參閱 AWS Organizations 文件中的可搭配 AWS Organizations運作的服務。

deployment

在目標環境中提供應用程式、新功能或程式碼修正的程序。部署涉及在程式碼庫中實作變更，然後在應用程式環境中建置和執行該程式碼庫。

開發環境

請參閱 環境。

偵測性控制

一種安全控制，用於在事件發生後偵測、記錄和提醒。這些控制是第二道防線，提醒您注意繞過現有預防性控制的安全事件。如需詳細資訊，請參閱在 AWS上實作安全控制中的偵測性控制。

開發值串流映射 (DVSM)

一種程序，用於識別並優先考慮對軟體開發生命週期中的速度和品質造成負面影響的限制。DVSM 延伸了原本專為精簡製造實務設計的價值串流映射程序。它著重於透過軟體開發程序建立和移動價值所需的步驟和團隊。

數位分身

真實世界系統的虛擬呈現，例如建築物、工廠、工業設備或生產線。數位分身支援預測性維護、遠端監控和生產最佳化。

維度資料表

在星星結構描述中，較小的資料表包含有關事實資料表中量化資料的資料屬性。維度資料表屬性通常是文字欄位或離散數字，其行為與文字相似。這些屬性通常用於查詢限制、篩選和結果集標記。

災難

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

災難復原 (DR)

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

DML

請參閱資料庫處理語言。

領域驅動的設計

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

DR

請參閱災難復原。

偏離偵測

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

DVSM

請參閱開發值串流映射。

# E

EDA

請參閱探索性資料分析。

EDI

請參閱電子資料交換。

邊緣運算

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

電子資料交換 (EDI)

在組織之間自動交換商業文件。如需詳細資訊，請參閱什麼是電子資料交換。

加密

將人類可讀取的純文字資料轉換為加密文字的運算程序。

加密金鑰

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

端序

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

端點

請參閱 服務端點。

端點服務

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

企業資源規劃 (ERP)

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

信封加密

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

環境

執行中應用程式的執行個體。以下是雲端運算中常見的環境類型：

- 開發環境 – 執行中應用程式的執行個體，只有負責維護應用程式的核心團隊才能使用。開發環境用來測試變更，然後再將開發環境提升到較高的環境。此類型的環境有時稱為測試環境。
- 較低的環境 – 應用程式的所有開發環境，例如用於初始建置和測試的開發環境。
- 生產環境 – 最終使用者可以存取的執行中應用程式的執行個體。在 CI/CD 管道中，生產環境是最後一個部署環境。
- 較高的環境 – 核心開發團隊以外的使用者可存取的所有環境。這可能包括生產環境、生產前環境以及用於使用者接受度測試的環境。

epic

在敏捷方法中，有助於組織工作並排定工作優先順序的功能類別。epic 提供要求和實作任務的高層級描述。例如， AWS CAF 安全概念包括身分和存取管理、偵測控制、基礎設施安全、資料保護和事件回應。如需有關 AWS 遷移策略中的 Epic 的詳細資訊，請參閱計畫實作指南。

ERP

請參閱企業資源規劃。

探索性資料分析 (EDA)

分析資料集以了解其主要特性的過程。您收集或彙總資料，然後執行初步調查以尋找模式、偵測異常並檢查假設。透過計算摘要統計並建立資料可視化來執行 EDA。

# F

事實資料表

星狀結構描述中的中央資料表。它存放有關業務操作的量化資料。一般而言，事實資料表包含兩種類型的資料欄：包含度量的資料，以及包含維度資料表外部索引鍵的資料欄。

快速失敗

一種使用頻繁和增量測試來縮短開發生命週期的理念。這是敏捷方法的關鍵部分。

故障隔離界限

在 中 AWS 雲端，像是可用區域 AWS 區域、控制平面或資料平面等邊界會限制故障的影響，並有助於改善工作負載的彈性。如需詳細資訊，請參閱AWS 故障隔離界限。

功能分支

請參閱分支。

特徵

用來進行預測的輸入資料。例如，在製造環境中，特徵可能是定期從製造生產線擷取的影像。

功能重要性

特徵對於模型的預測有多重要。這通常表示為可以透過各種技術來計算的數值得分，例如 Shapley Additive Explanations (SHAP) 和積分梯度。如需詳細資訊，請參閱 的機器學習模型可解譯性 AWS。

特徵轉換

優化 ML 程序的資料，包括使用其他來源豐富資料、調整值、或從單一資料欄位擷取多組資訊。這可讓 ML 模型從資料中受益。例如，如果將「2021-05-27 00:15:37」日期劃分為「2021」、「五月」、「週四」和「15」，則可以協助學習演算法學習與不同資料元件相關聯的細微模式。

少量擷取提示

在要求 LLM 執行類似的任務之前，提供少量示範任務和所需輸出的範例。此技術是內容內學習的應用程式，其中模型會從內嵌在提示中的範例 (快照) 中學習。對於需要特定格式、推理或網域知識的任務，少量的提示非常有效。另請參閱零鏡頭提示。

FGAC

請參閱精細存取控制。

精細存取控制 (FGAC)

使用多個條件來允許或拒絕存取請求。

閃切遷移

一種資料庫遷移方法，透過變更資料擷取使用連續資料複寫，以盡可能在最短的時間內遷移資料，而不是使用分階段方法。目標是將停機時間降至最低。

FM

請參閱基礎模型。

基礎模型 (FM)

大型深度學習神經網路，已針對廣義和未標記資料的大量資料集進行訓練。FMs 能夠執行各種一般任務，例如了解語言、產生文字和影像，以及以自然語言交談。如需詳細資訊，請參閱什麼是基礎模型。

# G

生成式 AI

已針對大量資料進行訓練的 AI 模型子集，可使用簡單的文字提示建立新的內容和成品，例如影像、影片、文字和音訊。如需詳細資訊，請參閱什麼是生成式 AI。

地理封鎖

請參閱地理限制。

地理限制 (地理封鎖)

Amazon CloudFront 中的選項，可防止特定國家/地區的使用者存取內容分發。您可以使用允許清單或封鎖清單來指定核准和禁止的國家/地區。如需詳細資訊，請參閱 CloudFront 文件中的限制內容的地理分佈。

Gitflow 工作流程

這是一種方法，其中較低和較高環境在原始碼儲存庫中使用不同分支。Gitflow 工作流程被視為舊版，而以幹線為基礎的工作流程是現代、偏好的方法。

黃金影像

系統或軟體的快照，做為部署該系統或軟體新執行個體的範本。例如，在製造中，黃金映像可用於在多個裝置上佈建軟體，並有助於提高裝置製造操作的速度、可擴展性和生產力。

綠地策略

新環境中缺乏現有基礎設施。對系統架構採用綠地策略時，可以選擇所有新技術，而不會限制與現有基礎設施的相容性，也稱為棕地。如果正在擴展現有基礎設施，則可能會混合棕地和綠地策略。

防護機制

有助於跨組織單位 (OU) 來管控資源、政策和合規的高層級規則。預防性防護機制會強制執行政策，以確保符合合規標準。透過使用服務控制政策和 IAM 許可界限來將其實作。偵測性防護機制可偵測政策違規和合規問題，並產生提醒以便修正。它們是透過使用 AWS Config AWS Security Hub CSPM、Amazon GuardDuty、、Amazon Inspector AWS Trusted Advisor和自訂 AWS Lambda 檢查來實作。

# H

HA

請參閱高可用性。

異質資料庫遷移

將來源資料庫遷移至使用不同資料庫引擎的目標資料庫 (例如，Oracle 至 Amazon Aurora)。異質遷移通常是重新架構工作的一部分，而轉換結構描述可能是一項複雜任務。AWS 提供有助於結構描述轉換的 AWS SCT。

高可用性 (HA)

在遇到挑戰或災難時，工作負載能夠在不介入的情況下持續運作。HA 系統的設計目的是自動容錯移轉、持續提供高品質的效能，以及處理不同的負載和故障，並將效能影響降至最低。

歷史現代化

一種方法，用於現代化和升級操作技術 (OT) 系統，以更好地滿足製造業的需求。歷史資料是一種資料庫，用於從工廠中的各種來源收集和存放資料。

保留資料

從用於訓練機器學習模型的資料集中保留的部分歷史標記資料。您可以使用保留資料，透過比較模型預測與保留資料來評估模型效能。

異質資料庫遷移

將您的來源資料庫遷移至共用相同資料庫引擎的目標資料庫 (例如，Microsoft SQL Server 至 Amazon RDS for SQL Server)。同質遷移通常是主機轉換或平台轉換工作的一部分。您可以使用原生資料庫公用程式來遷移結構描述。

熱資料

經常存取的資料，例如即時資料或最近的轉譯資料。此資料通常需要高效能儲存層或類別，才能提供快速的查詢回應。

修補程序

緊急修正生產環境中的關鍵問題。由於其緊迫性，通常會在典型 DevOps 發行工作流程之外執行修補程式。

超級護理期間

在切換後，遷移團隊在雲端管理和監控遷移的應用程式以解決任何問題的時段。通常，此期間的長度為 1-4 天。在超級護理期間結束時，遷移團隊通常會將應用程式的責任轉移給雲端營運團隊。

## I

IaC

將基礎設施視為程式碼。

身分型政策

連接至一或多個 IAM 主體的政策，可定義其在 AWS 雲端 環境中的許可。

閒置應用程式

90 天期間 CPU 和記憶體平均使用率在 5% 至 20% 之間的應用程式。在遷移專案中，通常會淘汰這些應用程式或將其保留在內部部署。

IIoT

請參閱工業物聯網。

不可變的基礎設施

為生產工作負載部署新基礎設施的模型，而不是更新、修補或修改現有的基礎設施。不可變基礎設施本質上比可變基礎設施更一致、可靠且可預測。如需詳細資訊，請參閱 AWS Well-Architected Framework 中的使用不可變基礎設施的部署最佳實務。

傳入 (輸入) VPC

在 AWS 多帳戶架構中，接受、檢查和路由來自應用程式外部之網路連線的 VPC。AWS 安全參考架構建議您使用傳入、傳出和檢查 VPC 來設定網路帳戶，以保護應用程式與更廣泛的網際網路之間的雙向介面。

增量遷移

一種切換策略，您可以在其中將應用程式分成小部分遷移，而不是執行單一、完整的切換。例如，您最初可能只將一些微服務或使用者移至新系統。確認所有項目都正常運作之後，您可以逐步移動其他微服務或使用者，直到可以解除委任舊式系統。此策略可降低與大型遷移關聯的風險。

工業 4.0

由 Klaus Schwab 於 2016 年推出的術語，透過連線能力、即時資料、自動化、分析和 AI/ML 的進展，指製造程序的現代化。

基礎設施

應用程式環境中包含的所有資源和資產。

基礎設施即程式碼 (IaC)

透過一組組態檔案來佈建和管理應用程式基礎設施的程序。IaC 旨在協助您集中管理基礎設施，標準化資源並快速擴展，以便新環境可重複、可靠且一致。

工業物聯網 (IIoT)

在製造業、能源、汽車、醫療保健、生命科學和農業等產業領域使用網際網路連線的感測器和裝置。如需詳細資訊，請參閱建立工業物聯網 (IIoT) 數位轉型策略。

檢查 VPC

在 AWS 多帳戶架構中，集中式 VPC 可管理 VPCs 之間 （在相同或不同的 中 AWS 區域)、網際網路和內部部署網路之間的網路流量檢查。AWS 安全參考架構建議您使用傳入、傳出和檢查 VPC 來設定網路帳戶，以保護應用程式與更廣泛的網際網路之間的雙向介面。

物聯網 (IoT)

具有內嵌式感測器或處理器的相連實體物體網路，其透過網際網路或本地通訊網路與其他裝置和系統進行通訊。如需詳細資訊，請參閱什麼是 IoT？

可解釋性

機器學習模型的一個特徵，描述了人類能夠理解模型的預測如何依賴於其輸入的程度。如需詳細資訊，請參閱 的機器學習模型可解譯性 AWS。

IoT

請參閱物聯網。

IT 資訊庫 (ITIL)

一組用於交付 IT 服務並使這些服務與業務需求保持一致的最佳實務。ITIL 為 ITSM 提供了基礎。

IT 服務管理 (ITSM)

與組織的設計、實作、管理和支援 IT 服務關聯的活動。如需有關將雲端操作與 ITSM 工具整合的資訊，請參閱操作整合指南。

ITIL

請參閱 IT 資訊庫。

ITSM

請參閱 IT 服務管理。

# L

標籤型存取控制 (LBAC)

強制存取控制 (MAC) 的實作，其中使用者和資料本身都會獲得明確指派的安全標籤值。使用者安全標籤和資料安全標籤之間的交集會決定使用者可以看到哪些資料列和資料欄。

登陸區域

登陸區域是架構良好的多帳戶 AWS 環境，可擴展且安全。這是一個起點，您的組織可以從此起點快速啟動和部署工作負載與應用程式，並對其安全和基礎設施環境充滿信心。如需有關登陸區域的詳細資訊，請參閱設定安全且可擴展的多帳戶 AWS 環境。

大型語言模型 (LLM)

預先訓練大量資料的深度學習 AI 模型。LLM 可以執行多個任務，例如回答問題、摘要文件、將文字翻譯成其他語言，以及完成句子。如需詳細資訊，請參閱什麼是 LLMs。

大型遷移

遷移 300 部或更多伺服器。

LBAC

請參閱標籤型存取控制。

最低權限

授予執行任務所需之最低許可的安全最佳實務。如需詳細資訊，請參閱 IAM 文件中的套用最低權限許可。

隨即轉移

請參閱 7 個 R。

小端序系統

首先儲存最低有效位元組的系統。另請參閱 Endianness。

LLM

請參閱大型語言模型。

較低的環境

請參閱 環境。

# M

機器學習 (ML)

一種使用演算法和技術進行模式識別和學習的人工智慧。機器學習會進行分析並從記錄的資料 (例如物聯網 (IoT) 資料) 中學習，以根據模式產生統計模型。如需詳細資訊，請參閱機器學習。

主要分支

請參閱分支。

惡意軟體

旨在危及電腦安全或隱私權的軟體。惡意軟體可能會中斷電腦系統、洩露敏感資訊，或取得未經授權的存取。惡意軟體的範例包括病毒、蠕蟲、勒索軟體、特洛伊木馬程式、間諜軟體和鍵盤記錄器。

受管服務

AWS 服務 會 AWS 操作基礎設施層、作業系統和平台，而您會存取端點來存放和擷取資料。Amazon Simple Storage Service (Amazon S3) 和 Amazon DynamoDB 是受管服務的範例。這些也稱為抽象服務。

製造執行系統 (MES)

一種軟體系統，用於追蹤、監控、記錄和控制生產程序，將原物料轉換為現場成品。

MAP

請參閱遷移加速計劃。

機制

建立工具、推動工具採用，然後檢查結果以進行調整的完整程序。機制是在操作時強化和改善自身的循環。如需詳細資訊，請參閱 AWS Well-Architected Framework 中的建置機制。

成員帳戶

除了屬於組織一部分的管理帳戶 AWS 帳戶 之外的所有 AWS Organizations。帳戶一次只能是一個組織的成員。

製造執行系統

請參閱製造執行系統。

訊息佇列遙測傳輸 (MQTT)

根據發佈/訂閱模式的輕量型machine-to-machine(M2M) 通訊協定，適用於資源受限的 IoT 裝置。

微服務

一種小型的獨立服務，它可透過定義明確的 API 進行通訊，通常由小型獨立團隊擁有。例如，保險系統可能包含對應至業務能力 (例如銷售或行銷) 或子領域 (例如購買、索賠或分析) 的微服務。微服務的優點包括靈活性、彈性擴展、輕鬆部署、可重複使用的程式碼和適應力。如需詳細資訊，請參閱使用無 AWS 伺服器服務整合微服務。

微服務架構

一種使用獨立元件來建置應用程式的方法，這些元件會以微服務形式執行每個應用程式程序。這些微服務會使用輕量型 API，透過明確定義的介面進行通訊。此架構中的每個微服務都可以進行更新、部署和擴展，以滿足應用程式特定功能的需求。如需詳細資訊，請參閱實作微服務 AWS。

Migration Acceleration Program (MAP)

此 AWS 計畫提供諮詢支援、訓練和服務，以協助組織建立強大的營運基礎，以移至雲端，並協助抵銷遷移的初始成本。MAP 包括用於有條不紊地執行舊式遷移的遷移方法以及一組用於自動化和加速常見遷移案例的工具。

大規模遷移

將大部分應用程式組合依波次移至雲端的程序，在每個波次中，都會以更快的速度移動更多應用程式。此階段使用從早期階段學到的最佳實務和經驗教訓來實作團隊、工具和流程的遷移工廠，以透過自動化和敏捷交付簡化工作負載的遷移。這是 AWS 遷移策略的第三階段。

遷移工廠

可透過自動化、敏捷的方法簡化工作負載遷移的跨職能團隊。遷移工廠團隊通常包括營運、業務分析師和擁有者、遷移工程師、開發人員以及從事 Sprint 工作的 DevOps 專業人員。20% 至 50% 之間的企業應用程式組合包含可透過工廠方法優化的重複模式。如需詳細資訊，請參閱此內容集中的遷移工廠的討論和雲端遷移工廠指南。

遷移中繼資料

有關完成遷移所需的應用程式和伺服器的資訊。每種遷移模式都需要一組不同的遷移中繼資料。遷移中繼資料的範例包括目標子網路、安全群組和 AWS 帳戶。

遷移模式

可重複的遷移任務，詳細描述遷移策略、遷移目的地以及所使用的遷移應用程式或服務。範例：使用 AWS Application Migration Service 重新託管遷移至 Amazon EC2。

遷移組合評定 (MPA)

線上工具，提供驗證商業案例以遷移至 的資訊 AWS 雲端。MPA 提供詳細的組合評定 (伺服器適當規模、定價、總體擁有成本比較、遷移成本分析) 以及遷移規劃 (應用程式資料分析和資料收集、應用程式分組、遷移優先順序，以及波次規劃)。MPA 工具 （需要登入） 可供所有 AWS 顧問和 APN 合作夥伴顧問免費使用。

遷移準備程度評定 (MRA)

使用 AWS CAF 取得組織雲端整備狀態的洞見、識別優缺點，以及建立行動計劃以消除已識別差距的程序。如需詳細資訊，請參閱遷移準備程度指南。MRA 是 AWS 遷移策略的第一階段。

遷移策略

用來將工作負載遷移至 的方法 AWS 雲端。如需詳細資訊，請參閱本詞彙表中的 7 個 Rs 項目，並請參閱動員您的組織以加速大規模遷移。

機器學習 (ML)

請參閱機器學習。

現代化

將過時的 (舊版或單一) 應用程式及其基礎架構轉換為雲端中靈活、富有彈性且高度可用的系統，以降低成本、提高效率並充分利用創新。如需詳細資訊，請參閱《》中的現代化應用程式的策略 AWS 雲端。

現代化準備程度評定

這項評估可協助判斷組織應用程式的現代化準備程度；識別優點、風險和相依性；並確定組織能夠在多大程度上支援這些應用程式的未來狀態。評定的結果就是目標架構的藍圖、詳細說明現代化程序的開發階段和里程碑的路線圖、以及解決已發現的差距之行動計畫。如需詳細資訊，請參閱《》中的評估應用程式的現代化準備 AWS 雲端程度。

單一應用程式 (單一)

透過緊密結合的程序作為單一服務執行的應用程式。單一應用程式有幾個缺點。如果一個應用程式功能遇到需求激增，則必須擴展整個架構。當程式碼庫增長時，新增或改進單一應用程式的功能也會變得更加複雜。若要解決這些問題，可以使用微服務架構。如需詳細資訊，請參閱將單一體系分解為微服務。

MPA

請參閱遷移產品組合評估。

MQTT

請參閱訊息佇列遙測傳輸。

多類別分類

一個有助於產生多類別預測的過程 (預測兩個以上的結果之一)。例如，機器學習模型可能會詢問「此產品是書籍、汽車還是電話？」 或者「這個客戶對哪種產品類別最感興趣？」

可變基礎設施

更新和修改生產工作負載現有基礎設施的模型。為了提高一致性、可靠性和可預測性， AWS Well-Architected Framework 建議使用不可變基礎設施做為最佳實務。

# O

OAC

請參閱原始存取控制。

OAI

請參閱原始存取身分。

OCM

請參閱組織變更管理。

離線遷移

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

OI

請參閱 操作整合。

OLA

請參閱操作層級協議。

線上遷移

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

OPC-UA

請參閱開放程序通訊 - 統一架構。

開放程序通訊 - 統一架構 (OPC-UA)

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

操作水準協議 (OLA)

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

操作整備審查 (ORR)

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

操作技術 (OT)

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

操作整合 (OI)

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

組織追蹤

建立的線索 AWS CloudTrail 會記錄 AWS 帳戶 組織中所有 的所有事件 AWS Organizations。在屬於組織的每個 AWS 帳戶 中建立此追蹤，它會跟蹤每個帳戶中的活動。如需詳細資訊，請參閱 CloudTrail 文件中的建立組織追蹤。

組織變更管理 (OCM)

用於從人員、文化和領導力層面管理重大、顛覆性業務轉型的架構。OCM 透過加速變更採用、解決過渡問題，以及推動文化和組織變更，協助組織為新系統和策略做好準備，並轉移至新系統和策略。在 AWS 遷移策略中，此架構稱為人員加速，因為雲端採用專案所需的變更速度。如需詳細資訊，請參閱 OCM 指南。

原始存取控制 (OAC)

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

原始存取身分 (OAI)

CloudFront 中的一個選項，用於限制存取以保護 Amazon S3 內容。當您使用 OAI 時，CloudFront 會建立一個可供 Amazon S3 進行驗證的主體。經驗證的主體只能透過特定 CloudFront 分發來存取 S3 儲存貯體中的內容。另請參閱 OAC，它可提供更精細且增強的存取控制。

ORR

請參閱操作整備審核。

OT

請參閱操作技術。

傳出 (輸出) VPC

在 AWS 多帳戶架構中，處理從應用程式內啟動之網路連線的 VPC。AWS 安全參考架構建議您使用傳入、傳出和檢查 VPC 來設定網路帳戶，以保護應用程式與更廣泛的網際網路之間的雙向介面。

# P

許可界限

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

個人身分識別資訊 (PII)

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

PII

請參閱個人身分識別資訊。

手冊

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

PLC

請參閱可程式設計邏輯控制器。

PLM

請參閱產品生命週期管理。

政策

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

混合持久性

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

組合評定

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

述詞

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

述詞下推

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

預防性控制

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

委託人

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

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

以預留位置值取代資料集中個人識別符的程序。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

請參閱擷取增強產生。

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 Rs。

復原點目標 (RPO)

自上次資料復原點以來可接受的時間上限。這會決定最後一個復原點與服務中斷之間可接受的資料遺失。

復原時間目標 (RTO)

服務中斷與服務還原之間的可接受延遲上限。

重構

請參閱 7 Rs。

區域

地理區域中的 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 Rs。

版本

在部署程序中，它是將變更提升至生產環境的動作。

重新定位

請參閱 7 個 R。

Replatform

請參閱 7 個 R。

回購

請參閱 7 Rs。

彈性

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

資源型政策

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

負責者、當責者、事先諮詢者和事後告知者 (RACI) 矩陣

定義所有涉及遷移活動和雲端操作之各方的角色和責任的矩陣。矩陣名稱衍生自矩陣中定義的責任類型：負責人 (R)、責任 (A)、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 Rs。

檢索增強生成 (RAG)

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

輪換

定期更新秘密的程序，讓攻擊者更難存取登入資料。

資料列和資料欄存取控制 (RCAC)

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

RPO

請參閱復原點目標。

RTO

請參閱復原時間目標。

執行手冊

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

# S

SAML 2.0

許多身分提供者 (IdP) 使用的開放標準。此功能會啟用聯合單一登入 (SSO)，讓使用者可以登入 AWS 管理主控台 或呼叫 AWS API 操作，而不必為您組織中的每個人在 IAM 中建立使用者。如需有關以 SAML 2.0 為基礎的聯合詳細資訊，請參閱 IAM 文件中的關於以 SAML 2.0 為基礎的聯合。

SCADA

請參閱監督控制和資料擷取。

SCP

請參閱服務控制政策。

秘密

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

依設計的安全性

透過整個開發程序將安全性納入考量的系統工程方法。

安全控制

一種技術或管理防護機制，它可預防、偵測或降低威脅行為者利用安全漏洞的能力。安全控制有四種主要類型：預防性、偵測性、回應性和主動性。

安全強化

減少受攻擊面以使其更能抵抗攻擊的過程。這可能包括一些動作，例如移除不再需要的資源、實作授予最低權限的安全最佳實務、或停用組態檔案中不必要的功能。

安全資訊與事件管理 (SIEM) 系統

結合安全資訊管理 (SIM) 和安全事件管理 (SEM) 系統的工具與服務。SIEM 系統會收集、監控和分析來自伺服器、網路、裝置和其他來源的資料，以偵測威脅和安全漏洞，並產生提醒。

安全回應自動化

預先定義和程式設計的動作，旨在自動回應或修復安全事件。這些自動化可做為偵測或回應式安全控制，協助您實作 AWS 安全最佳實務。自動化回應動作的範例包括修改 VPC 安全群組、修補 Amazon EC2 執行個體或輪換登入資料。

伺服器端加密

由接收資料的 AWS 服務 在其目的地加密資料。

服務控制政策 (SCP)

為 AWS Organizations中的組織的所有帳戶提供集中控制許可的政策。SCP 會定義防護機制或設定管理員可委派給使用者或角色的動作限制。您可以使用 SCP 作為允許清單或拒絕清單，以指定允許或禁止哪些服務或動作。如需詳細資訊，請參閱 AWS Organizations 文件中的服務控制政策。

服務端點

的進入點 URL AWS 服務。您可以使用端點，透過程式設計方式連接至目標服務。如需詳細資訊，請參閱 AWS 一般參考 中的 AWS 服務 端點。

服務水準協議 (SLA)

一份協議，闡明 IT 團隊承諾向客戶提供的服務，例如服務正常執行時間和效能。

服務層級指標 (SLI)

服務效能方面的測量，例如其錯誤率、可用性或輸送量。

服務層級目標 (SLO)

代表服務運作狀態的目標指標，由服務層級指標測量。

共同責任模式

描述您與 共同 AWS 承擔雲端安全與合規責任的模型。 AWS 負責雲端的安全，而 負責雲端的安全。如需詳細資訊，請參閱共同責任模式。

SIEM

請參閱安全資訊和事件管理系統。

單一故障點 (SPOF)

應用程式的單一關鍵元件故障，可能會中斷系統。

SLA

請參閱服務層級協議。

SLI

請參閱服務層級指標。

SLO

請參閱服務層級目標。

先拆分後播種模型

擴展和加速現代化專案的模式。定義新功能和產品版本時，核心團隊會進行拆分以建立新的產品團隊。這有助於擴展組織的能力和服務，提高開發人員生產力，並支援快速創新。如需詳細資訊，請參閱 中的階段式應用程式現代化方法 AWS 雲端。

SPOF

請參閱單一故障點。

星狀結構描述

使用一個大型事實資料表來存放交易或測量資料的資料庫組織結構，並使用一或多個較小的維度資料表來存放資料屬性。此結構旨在用於資料倉儲或商業智慧用途。

Strangler Fig 模式

一種現代化單一系統的方法，它會逐步重寫和取代系統功能，直到舊式系統停止使用為止。此模式源自無花果藤，它長成一棵馴化樹並最終戰勝且取代了其宿主。該模式由 Martin Fowler 引入，作為重寫單一系統時管理風險的方式。如需有關如何套用此模式的範例，請參閱使用容器和 Amazon API Gateway 逐步現代化舊版 Microsoft ASP.NET (ASMX) Web 服務。

子網

您 VPC 中的 IP 地址範圍。子網必須位於單一可用區域。

監控控制和資料擷取 (SCADA)

在製造中，使用硬體和軟體來監控實體資產和生產操作的系統。

對稱加密

使用相同金鑰來加密及解密資料的加密演算法。

合成測試

以模擬使用者互動的方式測試系統，以偵測潛在問題或監控效能。您可以使用 Amazon CloudWatch Synthetics 來建立這些測試。

系統提示

一種向 LLM 提供內容、指示或指導方針以指示其行為的技術。系統提示有助於設定內容，並建立與使用者互動的規則。

# T

標籤

做為中繼資料以組織 AWS 資源的鍵/值對。標籤可協助您管理、識別、組織、搜尋及篩選資源。如需詳細資訊，請參閱標記您的 AWS 資源。

目標變數

您嘗試在受監督的 ML 中預測的值。這也被稱為結果變數。例如，在製造設定中，目標變數可能是產品瑕疵。

任務清單

用於透過執行手冊追蹤進度的工具。任務清單包含執行手冊的概觀以及要完成的一般任務清單。對於每個一般任務，它包括所需的預估時間量、擁有者和進度。

測試環境

請參閱 環境。

訓練

為 ML 模型提供資料以供學習。訓練資料必須包含正確答案。學習演算法會在訓練資料中尋找將輸入資料屬性映射至目標的模式 (您想要預測的答案)。它會輸出擷取這些模式的 ML 模型。可以使用 ML 模型，來預測您不知道的目標新資料。

傳輸閘道

可以用於互連 VPC 和內部部署網路的網路傳輸中樞。如需詳細資訊，請參閱 AWS Transit Gateway 文件中的什麼是傳輸閘道。

主幹型工作流程

這是一種方法，開發人員可在功能分支中本地建置和測試功能，然後將這些變更合併到主要分支中。然後，主要分支會依序建置到開發環境、生產前環境和生產環境中。

受信任的存取權

將許可授予您指定的服務，以代表您在組織中 AWS Organizations 及其帳戶中執行任務。受信任的服務會在需要該角色時，在每個帳戶中建立服務連結角色，以便為您執行管理工作。如需詳細資訊，請參閱 文件中的 AWS Organizations 搭配使用 AWS Organizations 與其他 AWS 服務。

調校

變更訓練程序的各個層面，以提高 ML 模型的準確性。例如，可以透過產生標籤集、新增標籤、然後在不同的設定下多次重複這些步驟來訓練 ML 模型，以優化模型。

雙比薩團隊

兩個比薩就能吃飽的小型 DevOps 團隊。雙披薩團隊規模可確保軟體開發中的最佳協作。

# U

不確定性

這是一個概念，指的是不精確、不完整或未知的資訊，其可能會破壞預測性 ML 模型的可靠性。有兩種類型的不確定性：認知不確定性是由有限的、不完整的資料引起的，而隨機不確定性是由資料中固有的噪聲和隨機性引起的。如需詳細資訊，請參閱量化深度學習系統的不確定性指南。

未區分的任務

也稱為繁重工作，這是建立和操作應用程式的必要工作，但不為最終使用者提供直接價值或提供競爭優勢。未區分任務的範例包括採購、維護和容量規劃。

較高的環境

請參閱 環境。

# V

清空

一種資料庫維護操作，涉及增量更新後的清理工作，以回收儲存並提升效能。

版本控制

追蹤變更的程序和工具，例如儲存庫中原始程式碼的變更。

VPC 對等互連

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

漏洞

危害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缺陷或漏洞。威脅行為者可以使用這種類型的漏洞來攻擊系統。開發人員經常因為攻擊而意識到漏洞。

零鏡頭提示

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

殭屍應用程式

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

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。