



上的客服人員 AI 模式和 workflows AWS

AWS 方案指引



AWS 方案指引: 上的客服人員 AI 模式和 workflows AWS

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

Table of Contents

簡介	1
目標對象	1
目標	1
關於此內容系列	1
客服人員模式	2
基本推理代理程式	3
Architecture	3
說明	4
功能	5
限制	5
常用案例	5
實作指引	5
摘要	6
Architecture	6
說明	7
功能	8
常用案例	8
實作指引	8
摘要	8
用於呼叫 函數的工具型代理程式	9
Architecture	9
說明	10
功能	11
常用案例	11
實作指引	11
摘要	12
伺服器的工具型代理程式	12
Architecture	12
說明	13
功能	14
常用案例	14
實作指引	14
摘要	15
電腦使用代理程式	15

Architecture	15
說明	16
功能	16
常用案例	17
實作指引	17
摘要	17
編碼代理程式	17
Architecture	18
說明	18
功能	19
常用案例	19
實作指引	20
摘要	20
語音和語音客服人員	20
Architecture	20
說明	21
功能	22
常用案例	22
實作指引	22
摘要	23
工作流程協同運作代理程式	23
Architecture	23
說明	24
功能	25
常用案例	25
實作指引	25
摘要	25
記憶體增強型代理程式	26
Architecture	26
說明	26
功能	27
常用案例	27
實作經記憶體驗證的代理程式	28
實作記憶體注入提示	28
摘要	29
模擬和測試平台代理程式	29

Architecture	29
說明	30
功能	31
常用案例	31
實作指引	31
摘要	32
觀察者和監控代理程式	32
Architecture	33
說明	33
功能	34
常用案例	34
實作指引	34
摘要	35
多代理程式協同合作	35
說明	37
功能	38
常用案例	38
實作指引	38
摘要	39
結論	39
要點	39
LLM 工作流程	41
LLM 擴增的認知概觀	41
提示鏈結的工作流程	42
說明	43
功能	43
常用案例	43
路由的工作流程	43
功能	44
常用案例	45
平行化的工作流程	45
功能	46
常用案例	46
協同運作的工作流程	46
功能	47
常用案例	48

評估者和反射改進迴圈的工作流程	48
常用案例	49
功能	49
結論	49
代理工作流程模式	51
從事件驅動到認知擴增系統	51
事件驅動型架構	51
Cognition 擴增工作流程	52
核心洞察	54
提示鏈結 saga 模式	54
系列事件編排	54
提示鏈結模式	55
客服人員編排	55
要點	57
路由動態分派模式	57
動態分派	58
以 LLM 為基礎的路由	59
代理程式路由器	60
要點	61
平行化和散佈收集模式	61
散佈集合	62
LLM 型平行處理（散佈加法認知）	64
客服人員平行處理	64
要點	65
Saga 協同運作模式	65
事件協同運作	66
角色型代理程式系統（協調器）	67
主管	67
要點	69
評估器反射改進迴圈模式	69
回饋控制迴圈	70
回饋控制迴圈（評估器）	71
評估者	71
要點	72
在上設計代理程式工作流程 AWS	72
結論	73

文件歷史紀錄	74
.....	lxxv

上的客服人員 AI 模式和 workflows 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

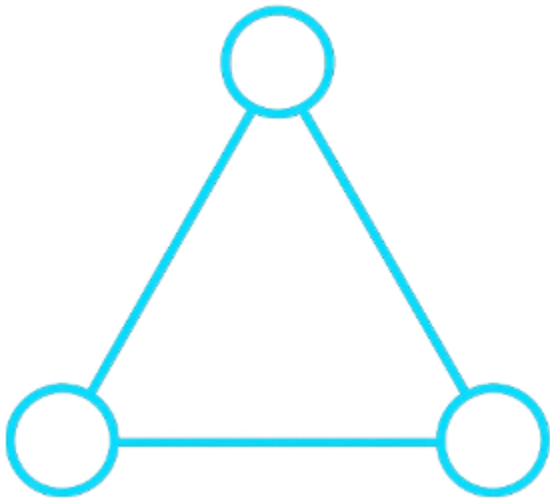
本指南是代理式 AI 相關系列的一部分 AWS。如需詳細資訊和檢視此系列中的其他指南，請參閱 AWS 規範性指南網站上的 [客服人員 AI](#)。

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 – 客服人員在沒有人工或外部控制的情況下獨立行動
- 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

- [基本推理代理程式](#)
- [用於呼叫 函數的工具型代理程式](#)
- [伺服器的工具型代理程式](#)
- [電腦使用代理程式](#)
- [編碼代理程式](#)

- [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](#)

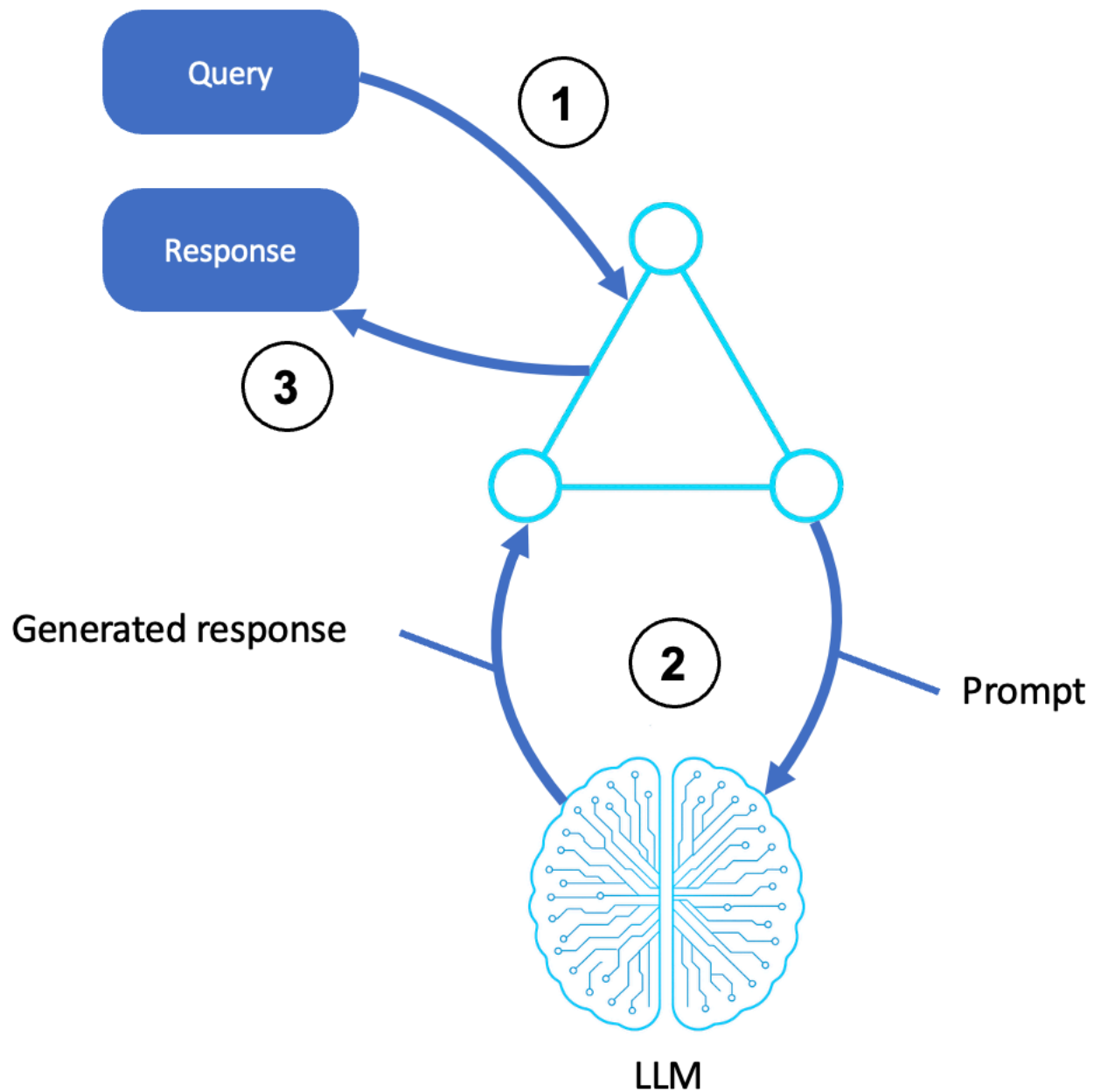
基本推理代理程式

基本推理代理程式是代理程式 AI 最簡單的形式，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代理程式的介面。
- 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

摘要

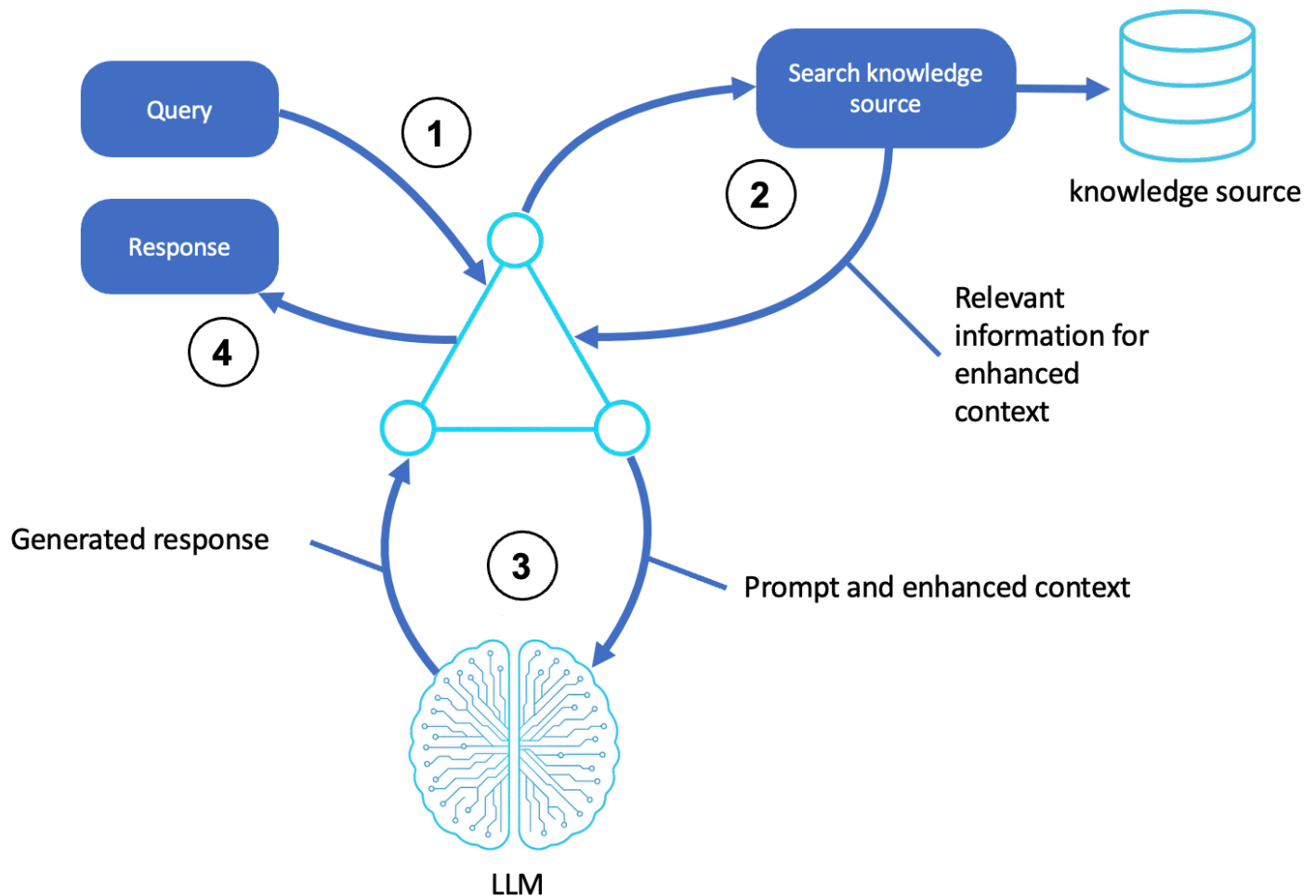
基本推理代理程式的基礎在於其簡單結構。它具有核心功能，可將目標轉換為導致智慧輸出的推理路徑。此模式通常是進階模式的起點，例如以工具為基礎的代理程式，以及使用擷取擴增產生 (RAG) 的代理程式。它也是大型工作流程中可靠且模組化的元件。

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

- 代理程式會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

- 代理程式會將輸出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
- 完全相容於向量資料庫、語意索引和中繼資料篩選

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
- 與代理程式的知識型整合（使用記憶體外掛程式、語意擷取器或 Amazon Bedrock）

摘要

代理程式 RAG 將靜態模型推理連接到動態、真實世界的智慧。它可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，並產生高度可稽核的可靠回應。

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。

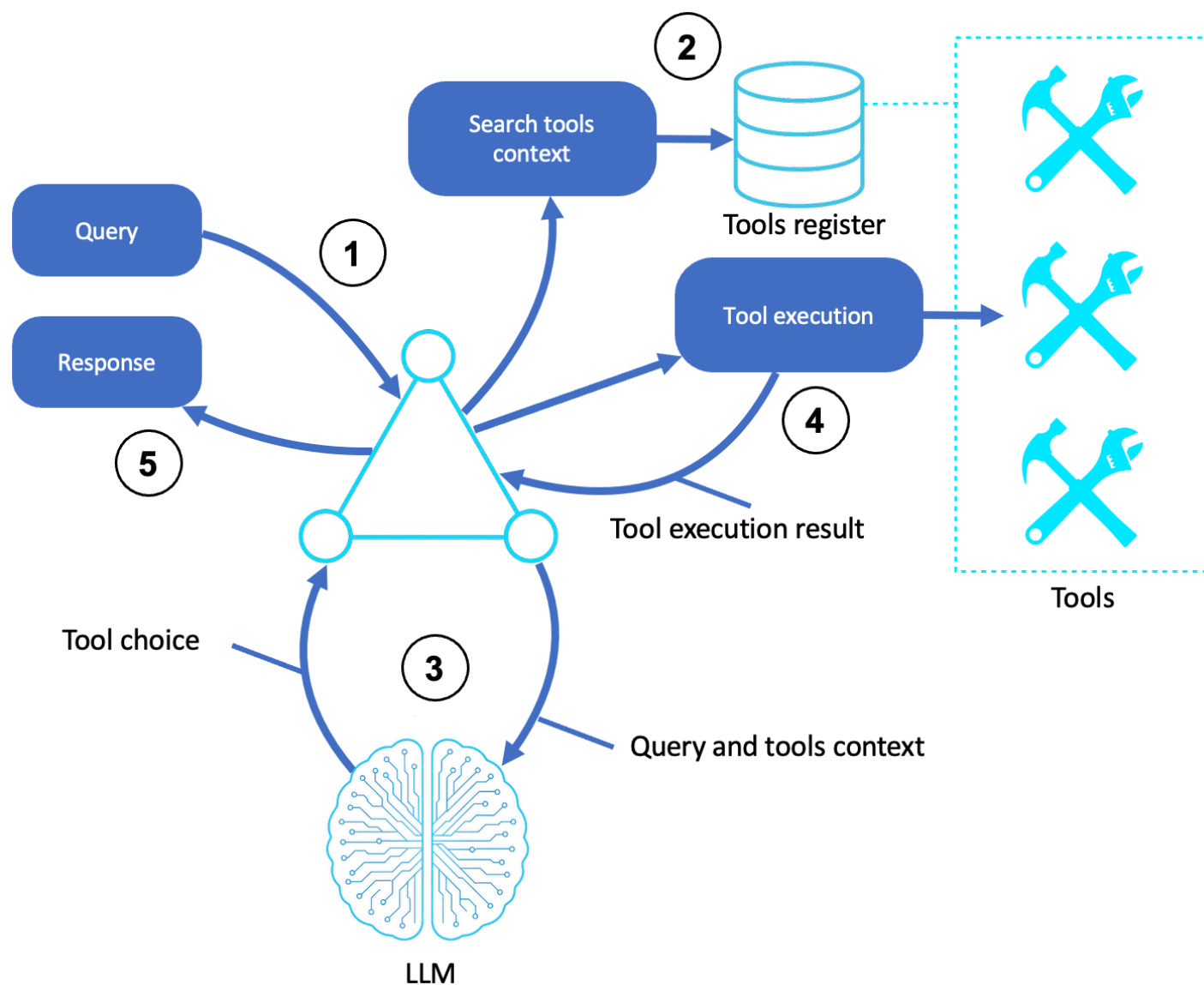
用於呼叫 函數的工具型代理程式

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，並將工具的輸出納入其推理迴圈。

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

摘要

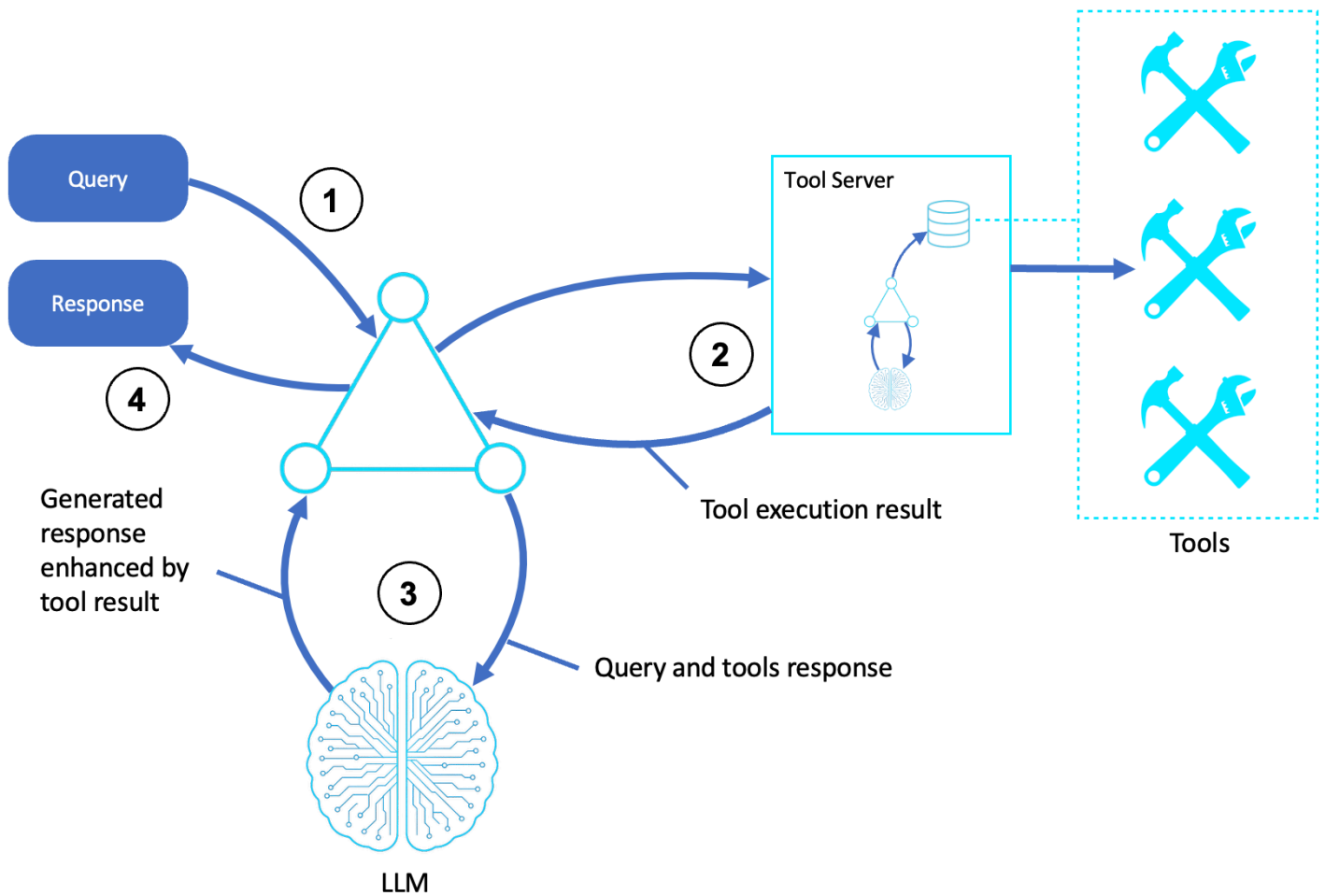
工具型函數呼叫代理程式代表從了解語言到執行動作的轉變。這些客服人員會叫用動態的內容感知工具，同時維護 LLM 推理，將被動助理轉換為完成任務、存取服務和整合業務操作的系統。此模式是企業設定中代理式 AI 的重要元件，特別是與宣告式結構描述、授權架構和多代理程式系統結合時。

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. 使用 LLM 推理搭配工具輸出

- 客服人員會叫用 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

- 代理程式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、願景和程式碼)
- 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-tool Amazon 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

摘要

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。

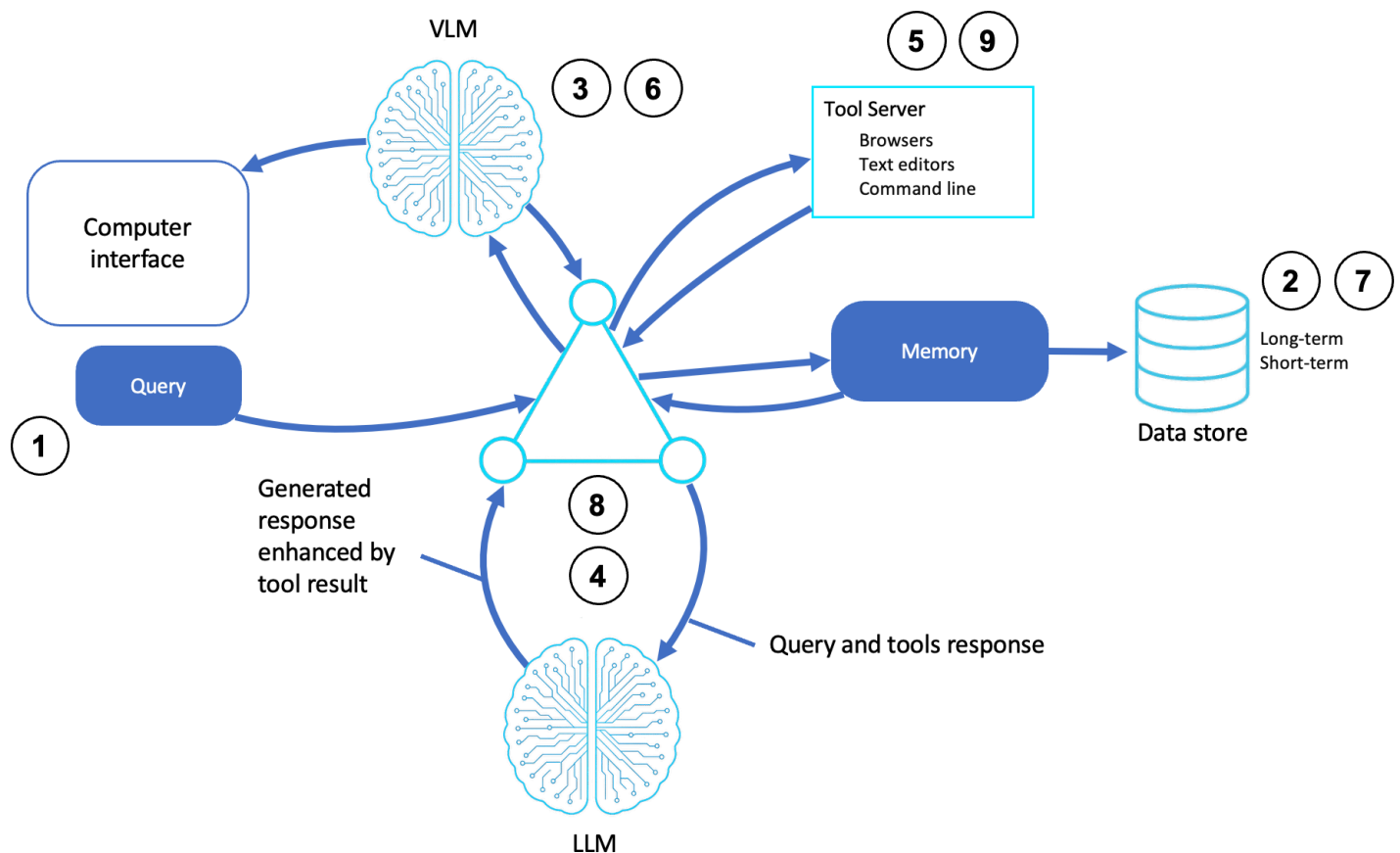
電腦使用代理程式

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。

此模式對於實際的 AI 自動化很重要，其中代理程式不僅做為助理，也做為代理程式執行動作的方式，通常使用相同的工具和環境。

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

摘要

電腦使用代理程式充當自主數位運算子，彌補人類電腦互動與 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。

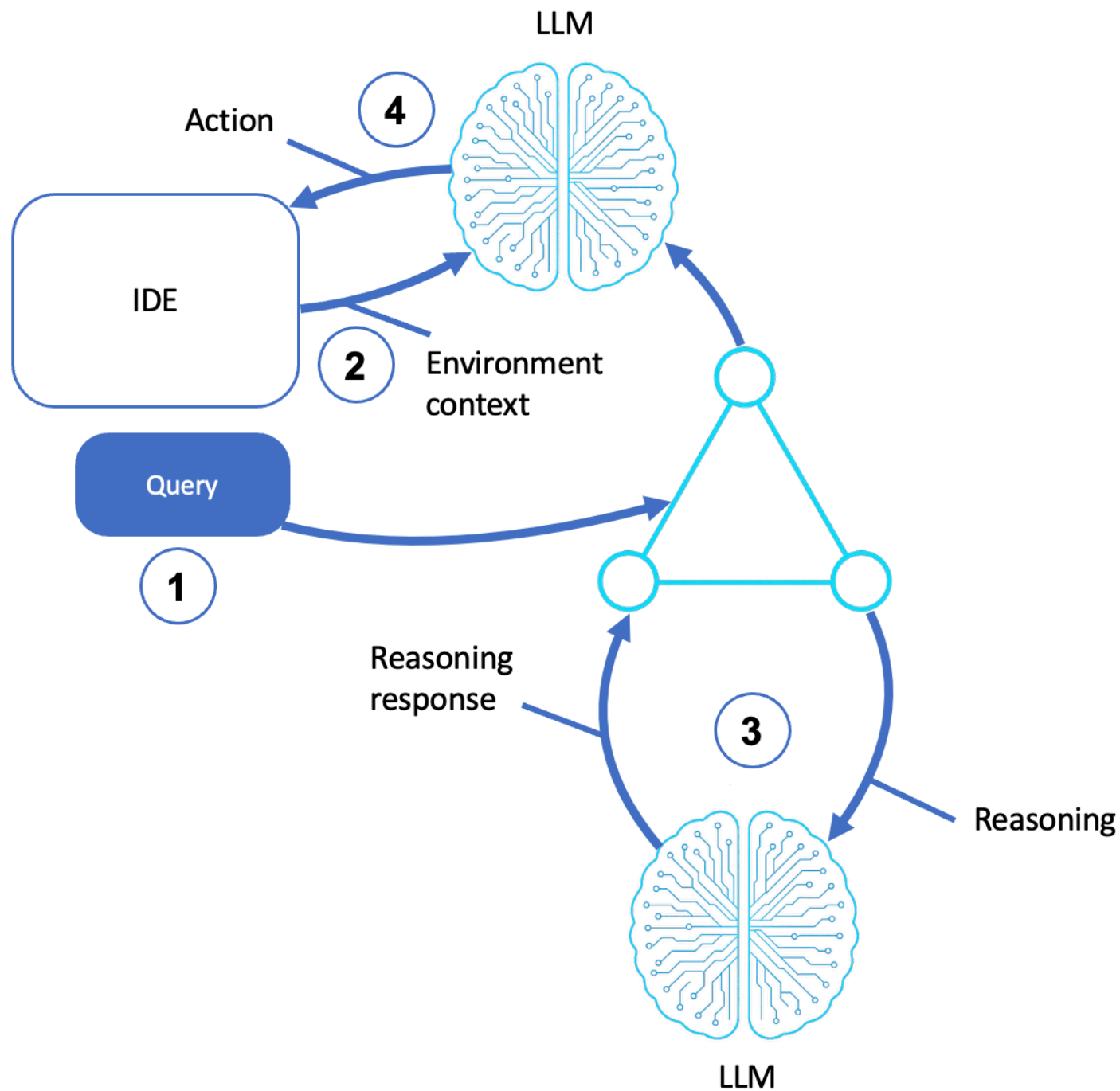
編碼代理程式

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

編碼代理程式模式如下圖所示：



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，以及來自其他客服人員的輸出。

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
- 選程式碼規劃和動作分隔（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

摘要

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。

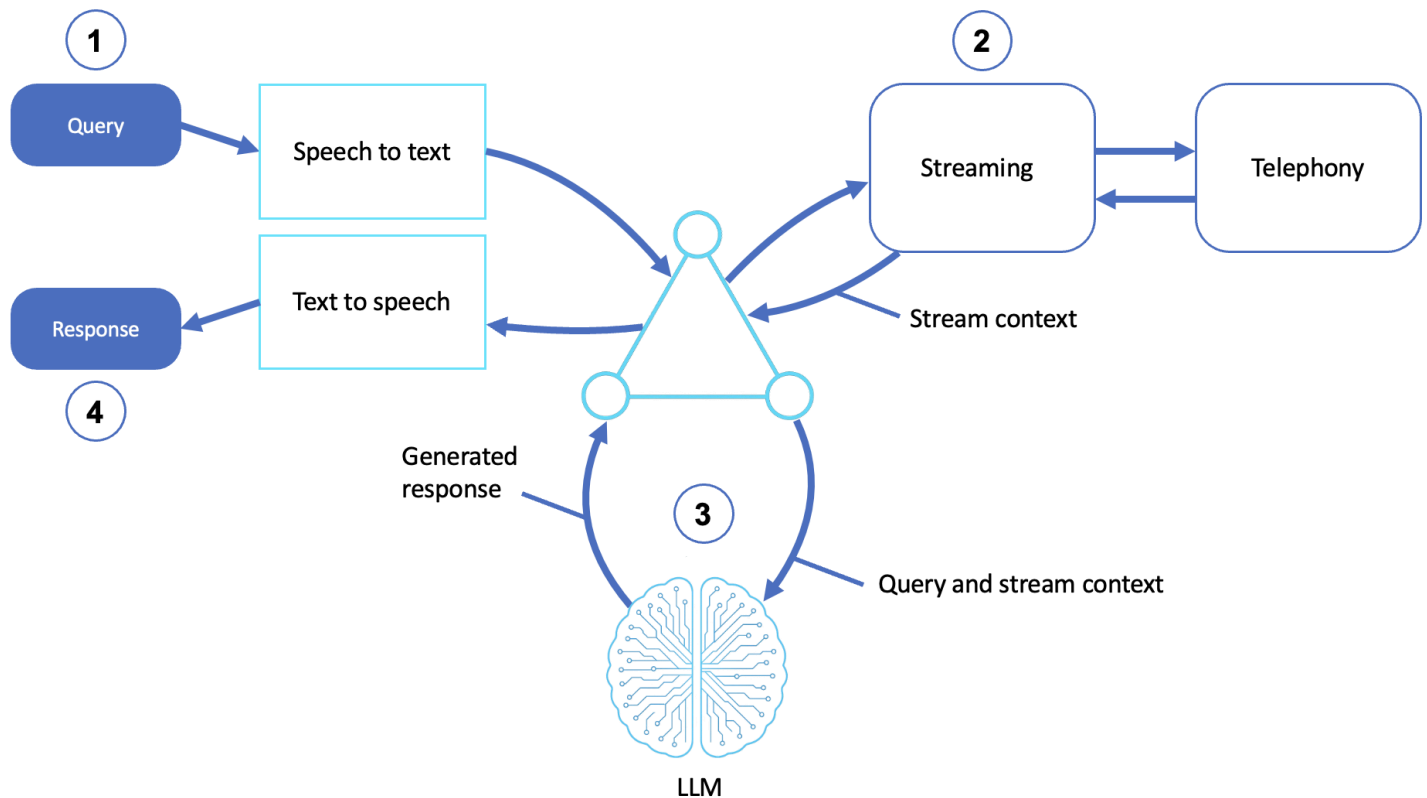
語音和語音客服人員

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。

語音客服人員在免持、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 Customer 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

摘要

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。

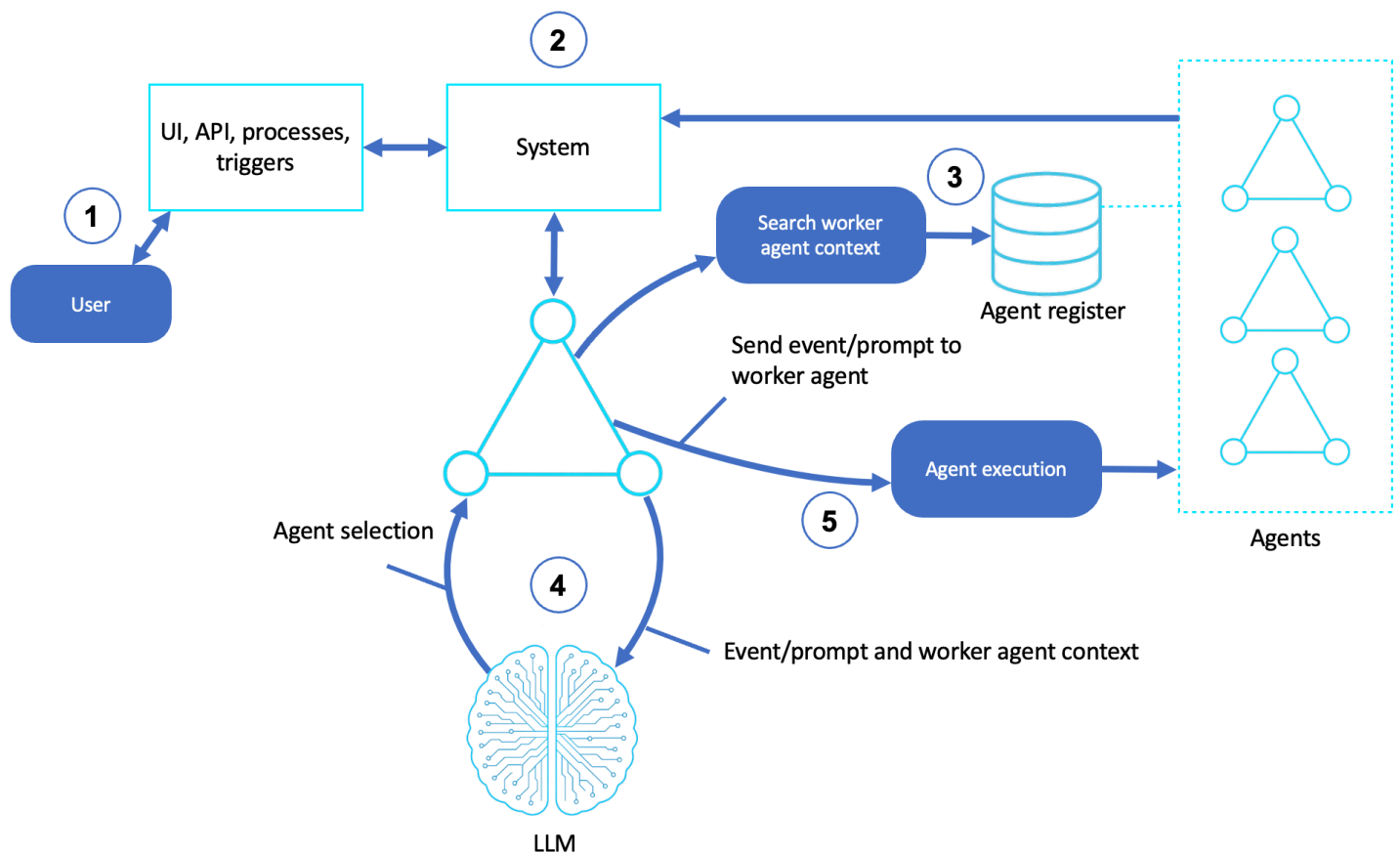
工作流程協同運作代理程式

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，為任務尋找正確的工作者代理程式。

4. 選取 LLM 代理程式

- LLM 透過分析任務描述和可用選項，協助選擇最佳的客服人員或工作流程計劃。
- 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 代理器在同一迴圈中與人類和機器人協調
- 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：

- 用於推理和代理程式選擇的 Amazon Bedrock
- 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

摘要

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。

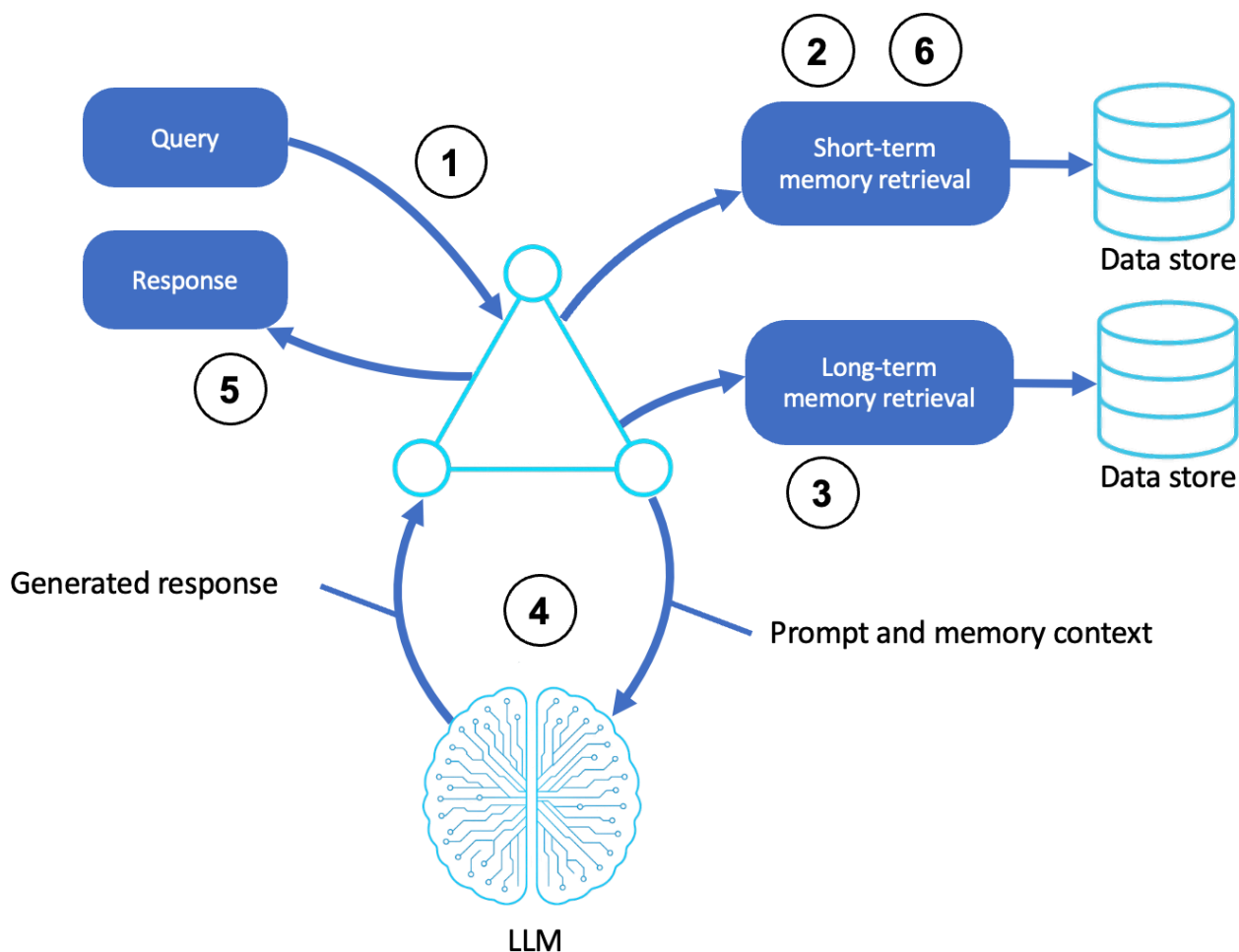
記憶體增強型代理程式

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、更個人化的策略回應。

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 workflow 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，例如下列項目：
 - 使用者偏好設定
 - 過去的決策和結果
 - 學到的概念、摘要或體驗

4. 透過 LLM 的原因

- 記憶體內容會嵌入 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供日後任務使用。

功能

- 對話或事件之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 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
協調	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，將最新的客服人員狀態和最近的對話歷史記錄納入為結構化輸入，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。

摘要

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的行為更像是適應性數位對等，更不像無狀態工具。

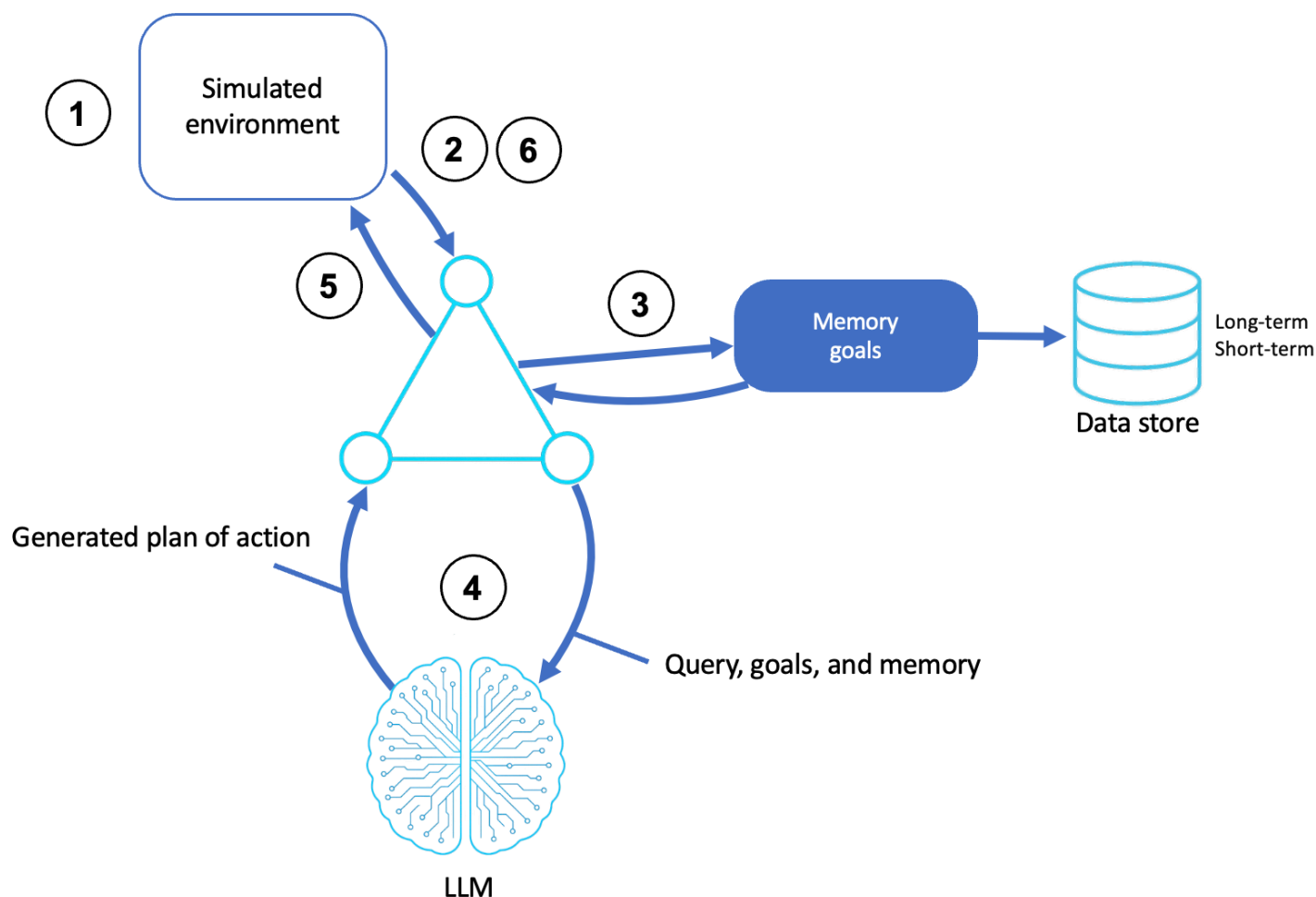
模擬和測試平台代理程式

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、行動和學習。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)、自動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](#)

摘要

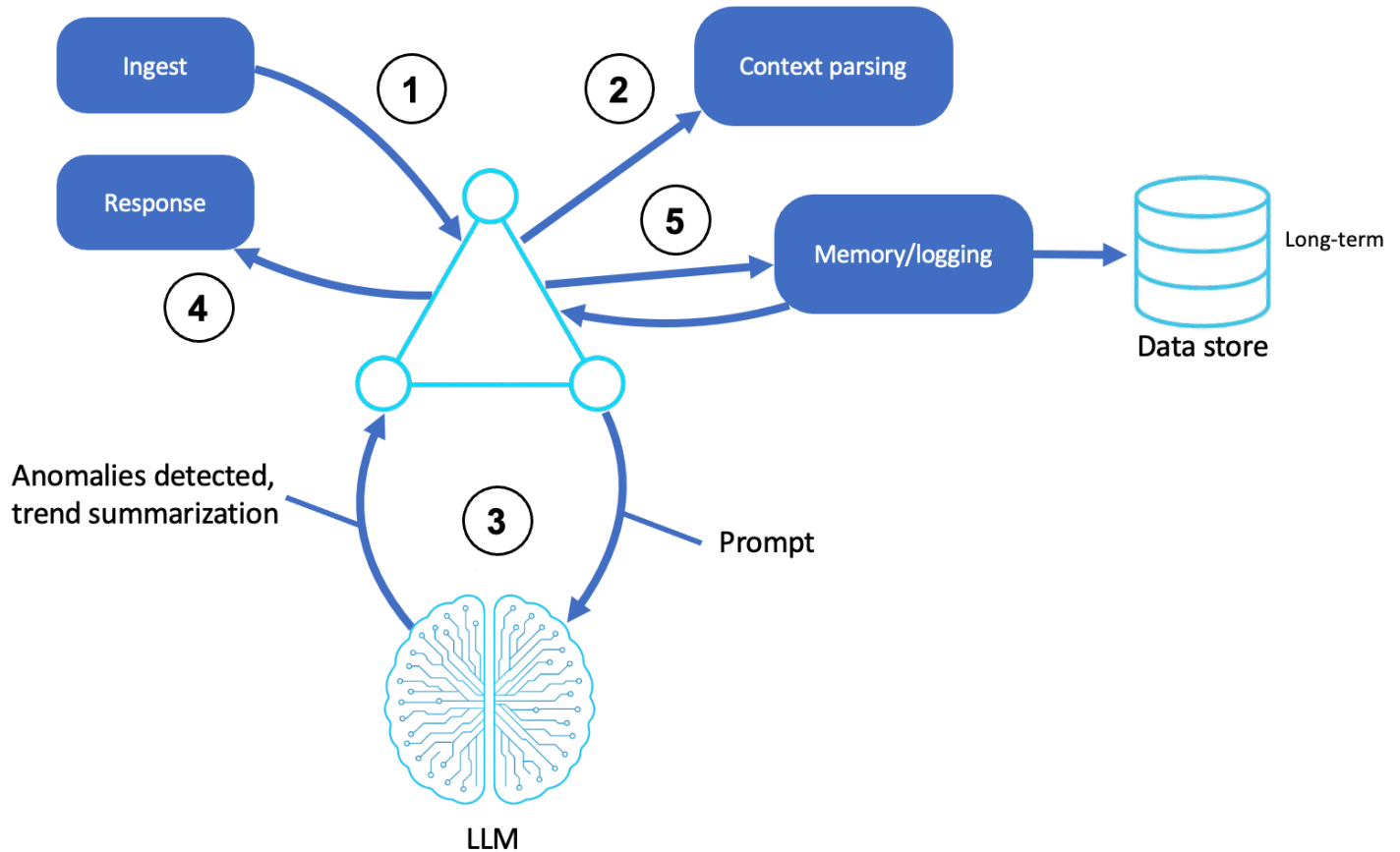
模擬和測試平台代理程式用於在部署到生產系統之前進行結構化探勘。使用這些代理程式來訓練自動導覽政策、在合成環境中測試業務流程，以及評估協調模式的扭曲。

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，它們增強了提醒、診斷和稽核，而不會直接啟動行為。

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的互動事件接聽程式。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，如下所示：
 - 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](#) 視覺化代理程式輸出

摘要

觀察者和監控代理程式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

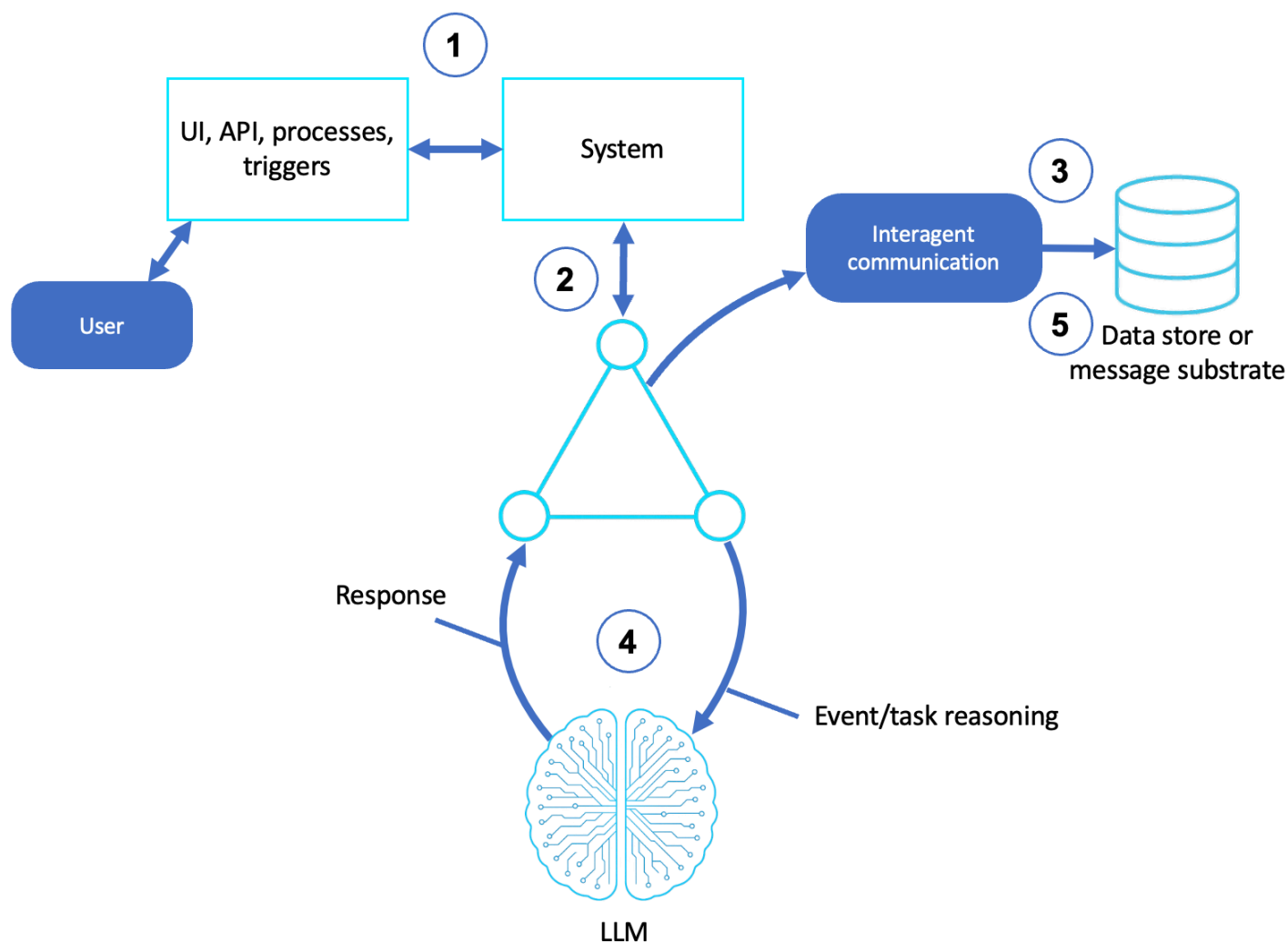
多重代理程式協同合作是指一種模式，其中多個自動代理程式各自具有不同的角色、專業或目標，進行交涉以解決複雜的任務。這些客服人員可以獨立運作，也可以與其他客服人員一起運作，方法是共用資訊、分配責任，以及共同推理目標。

此模式與 workflows 代理程式不同，workflows 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、平行處理和分割來強調 peer-to-peer 或緊急協調。下表比較多代理程式與 workflows 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、彈性的任務分發
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。
- 「管理員」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. 使用專門的推理

- 每個代理程式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、計畫或動作。
- （選用）監督代理程式可以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）
- 溝通、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

摘要

多重客服人員協同合作可將問題解決任務分散到模組化、角色驅動的客服人員。與 workflows 協同運作不同，協作模式使用緊急智慧、彈性和可擴展性，反映人類解決問題的方式。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，每個模式都是針對以自主性、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。

這些模式共用用於建置智慧型目標導向系統的詞彙和技術藍圖。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 嵌入結構化工作流程中，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

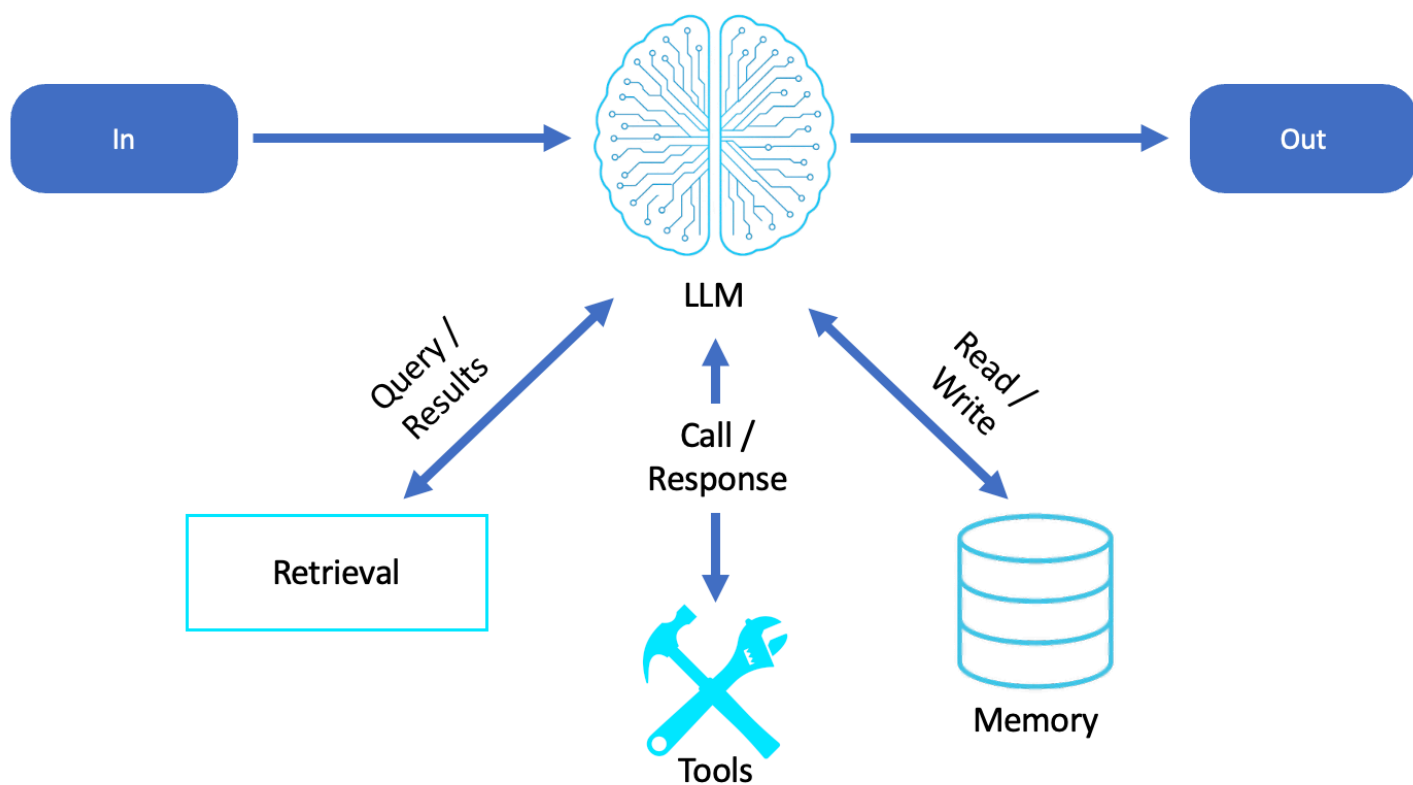
- [LLM 擴增的認知概觀](#)
- [提示鏈結的工作流程](#)
- [路由的工作流程](#)
- [平行化的工作流程](#)
- [協同運作的工作流程](#)
- [評估者和反射改進迴圈的工作流程](#)
- [結論](#)

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：

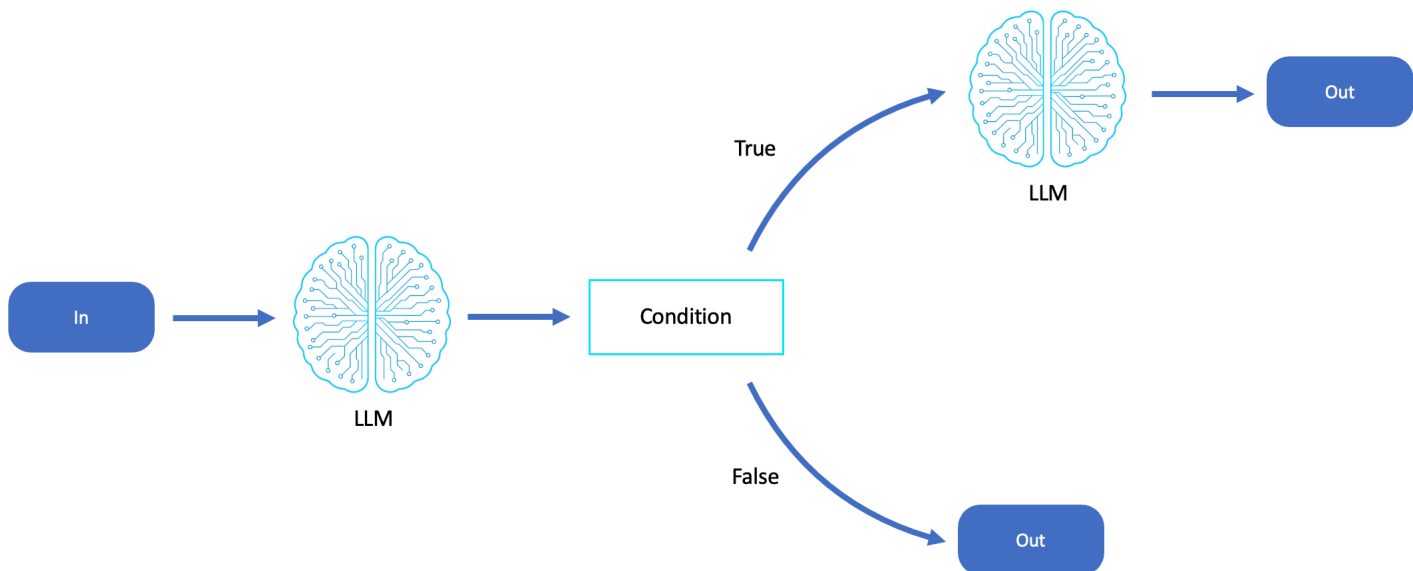
- 提示 – 使用內容、指示、範例和記憶體建立輸入架構
- 擷取 – 透過向量搜尋或語意記憶體為 LLM 提示提供 up-to-date 或特定領域的知識，例如透過擷取擴增產生 (RAG)
- 工具使用 – 讓 LLM 叫用 APIs 或呼叫函數以擷取資訊或對資訊採取行動
- 記憶體 – 使用結構化資料庫或內容摘要，將持久性或以工作階段為基礎的狀態納入推理迴圈

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 workflow 適用於在邏輯上可分為循序推理步驟的任務，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。

說明

- 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

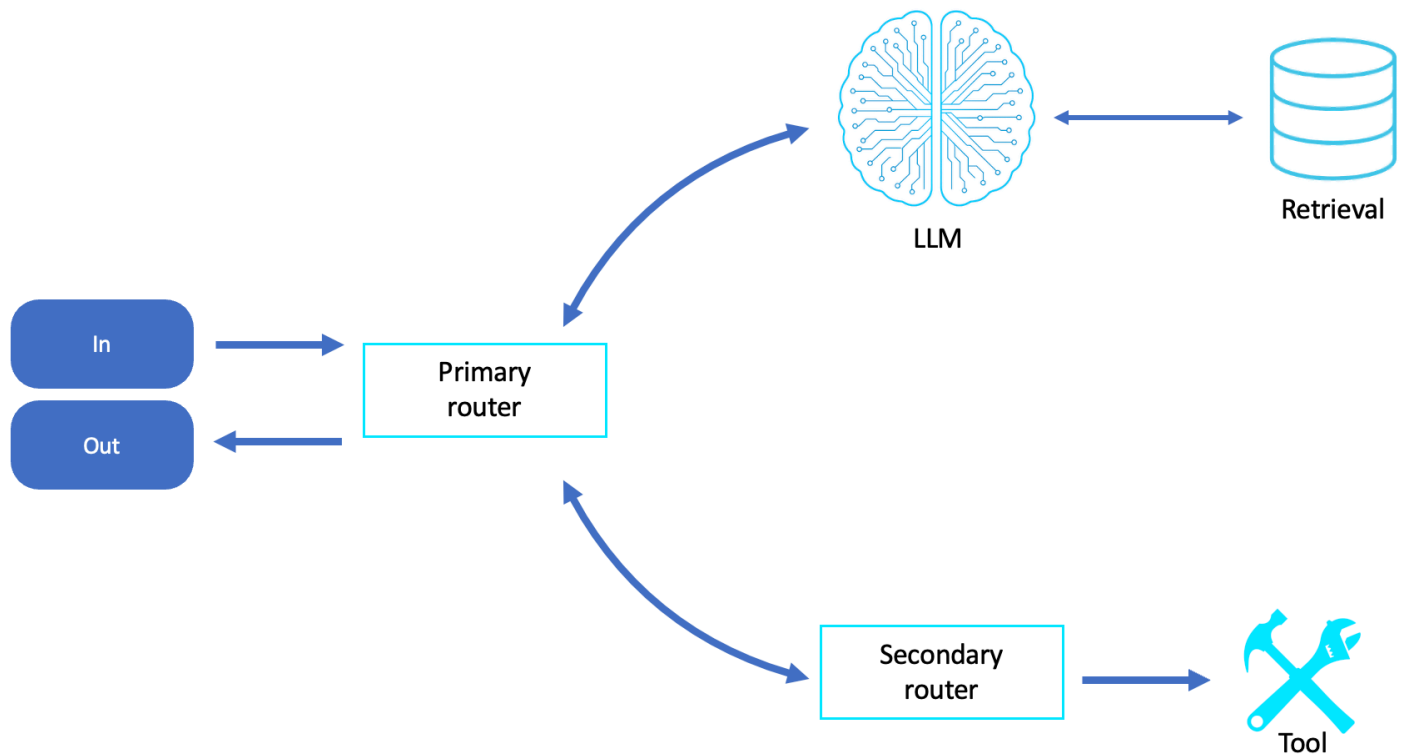
- LLM 呼叫的線性或分支鏈
- 做為結構化輸入傳遞或內嵌到後續提示中的中繼結果
- 可與 AWS Step Functions AWS Lambda 或 代理程式特定的執行器協同運作

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

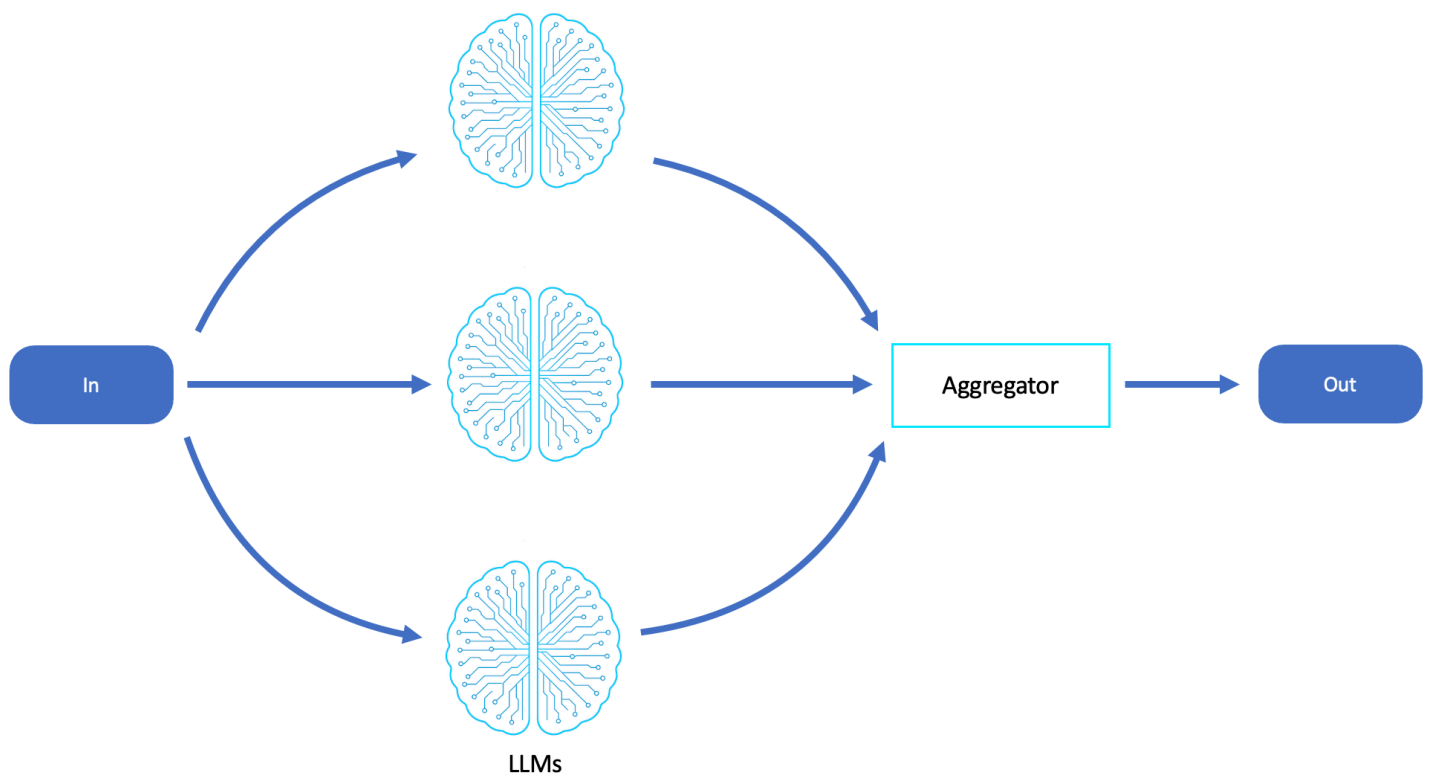
- 第一個傳遞 LLM 充當發送器
- 路由可以叫用不同的工作流程，甚至是其他代理程式模式
- 支援模組化功能擴展

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 workflow 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

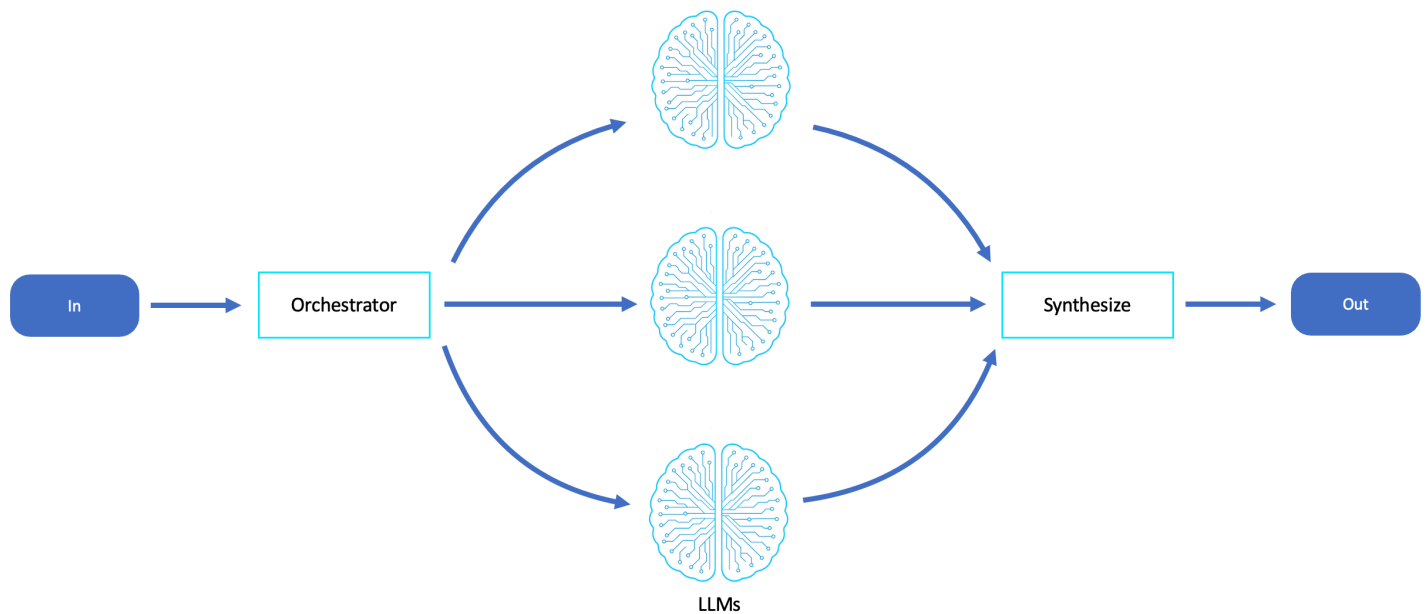
- LLM 任務的平行執行（使用 AWS Lambda AWS Fargate 或 AWS Step Functions 映射狀態）
- 在合成階段需要結果對齊、驗證或重複資料刪除
- 非常適合無狀態代理程式迴圈

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。這會反映人工團隊結構，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，規劃、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

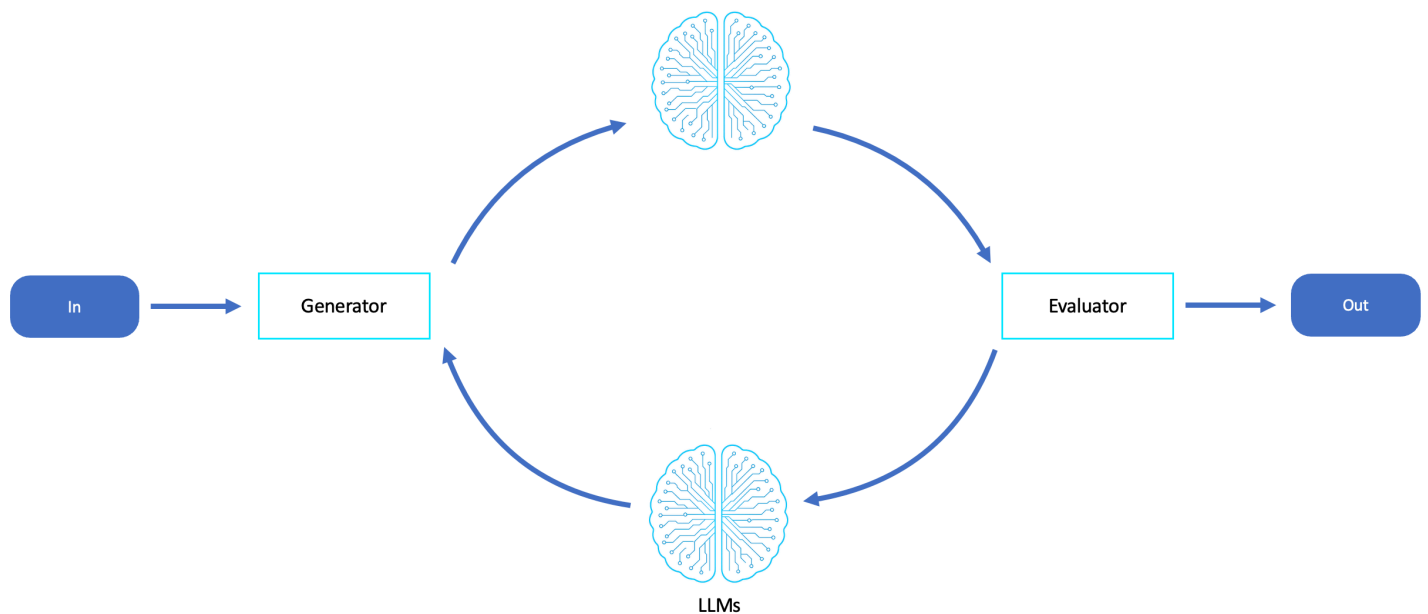
- Orchestrator 執行目標中繼關聯
- 工作者代理程式可能包含工具存取、記憶體或網域特定的提示
- 可以是階層式（亦即多層任務委派）

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，以及產生單一通道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）。
- 客服人員必須透過權衡、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 workflow 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 必須嵌入刻意的工作流程模式中，以定義模型處理輸入、管理內容和協調動作的方式。

LLM 工作流程引入了建置客服人員認知模組的基礎：

- 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 workflow 都代表一個可編譯模式，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 workflow 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 workflows 模式的下一章時，這些 LLM workflows 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 workflows 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 and 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，以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

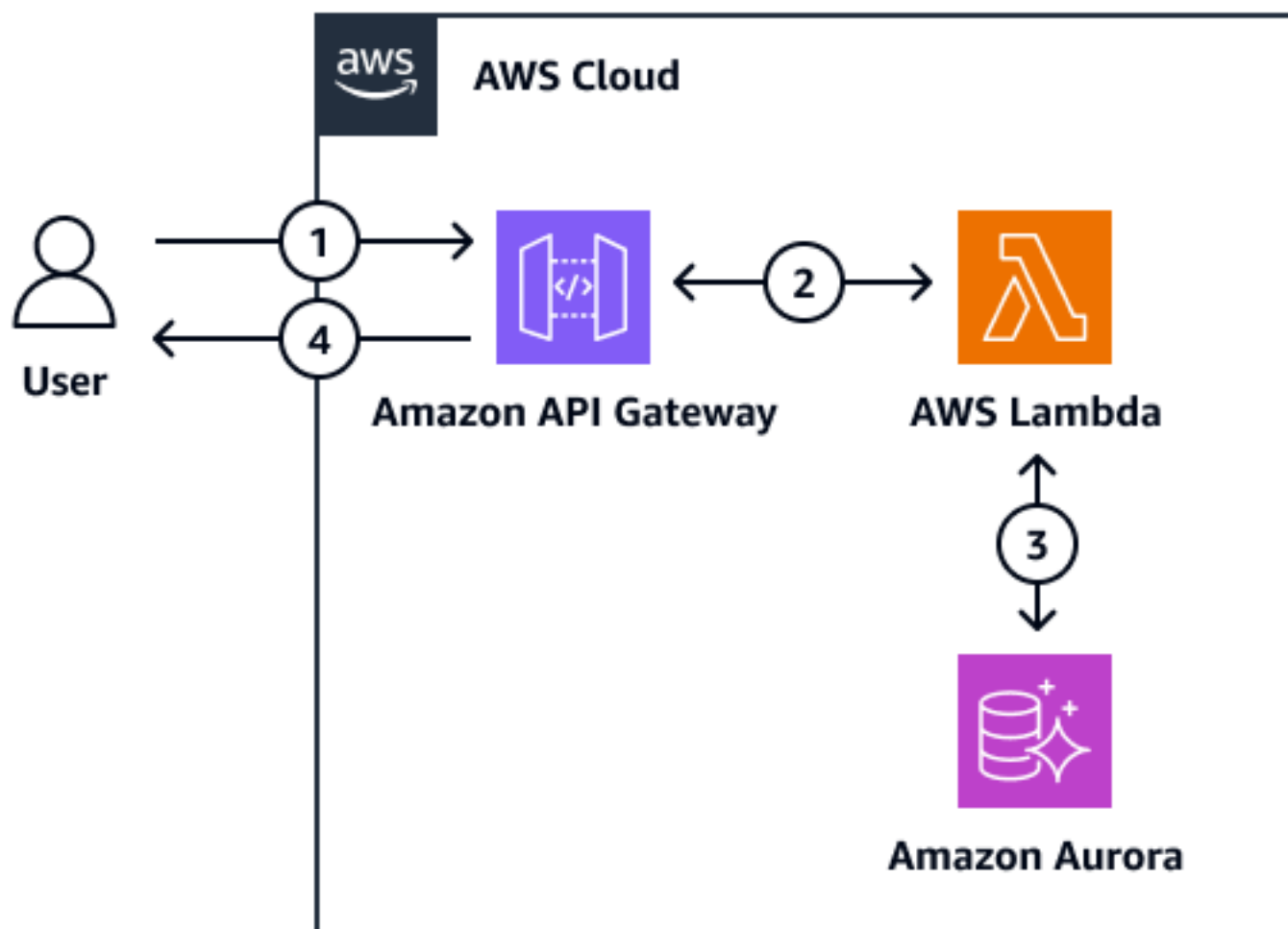
- [從事件驅動到認知擴增系統](#)
- [提示鏈結 saga 模式](#)
- [路由動態分派模式](#)
- [平行化和散佈收集模式](#)
- [Saga 協同運作模式](#)
- [評估器反射改進迴圈模式](#)
- [在 上設計代理程式工作流程 AWS](#)
- [結論](#)

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、廣播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：

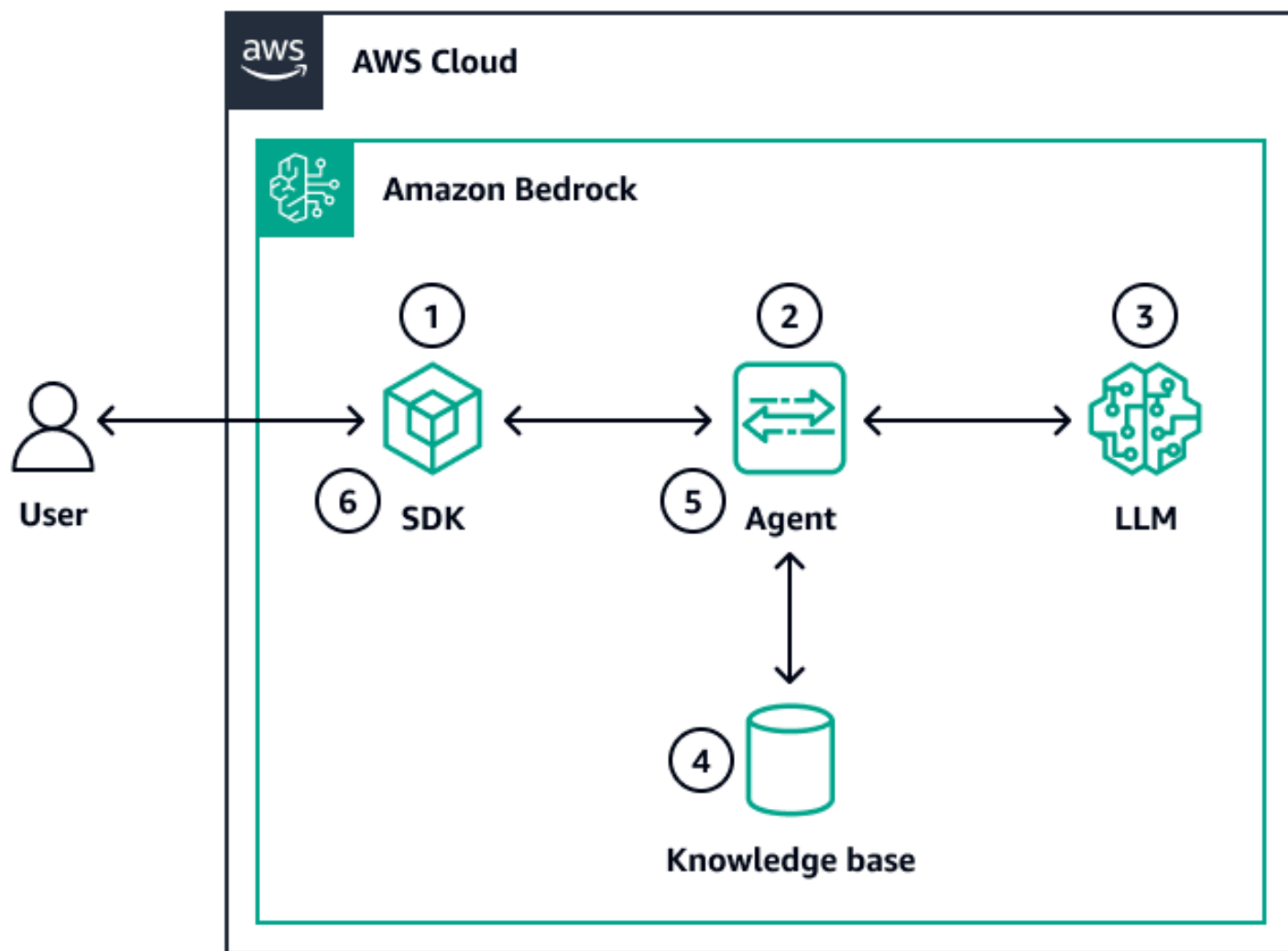


1. 使用者向 Amazon API Gateway 提交請求。
2. Amazon API Gateway 會將請求路由到 AWS Lambda 函數。
3. AWS Lambda 透過查詢 Amazon Aurora 資料庫來執行資料擴充
4. Amazon API Gateway 會將富集的承載傳回給發起人。

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，然後決定如何做出最佳回應。此模式會反映傳統擴充模式，其中訊息會先以外部資料增強，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。

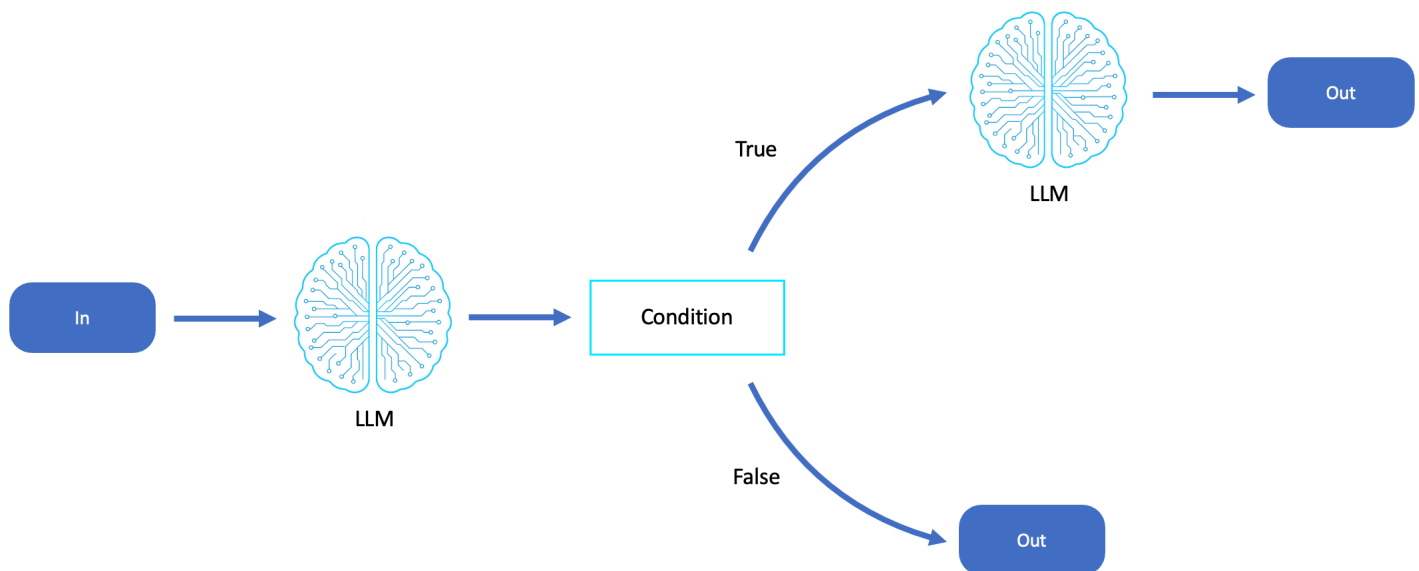
核心洞察

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、可復原和語義協調。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。

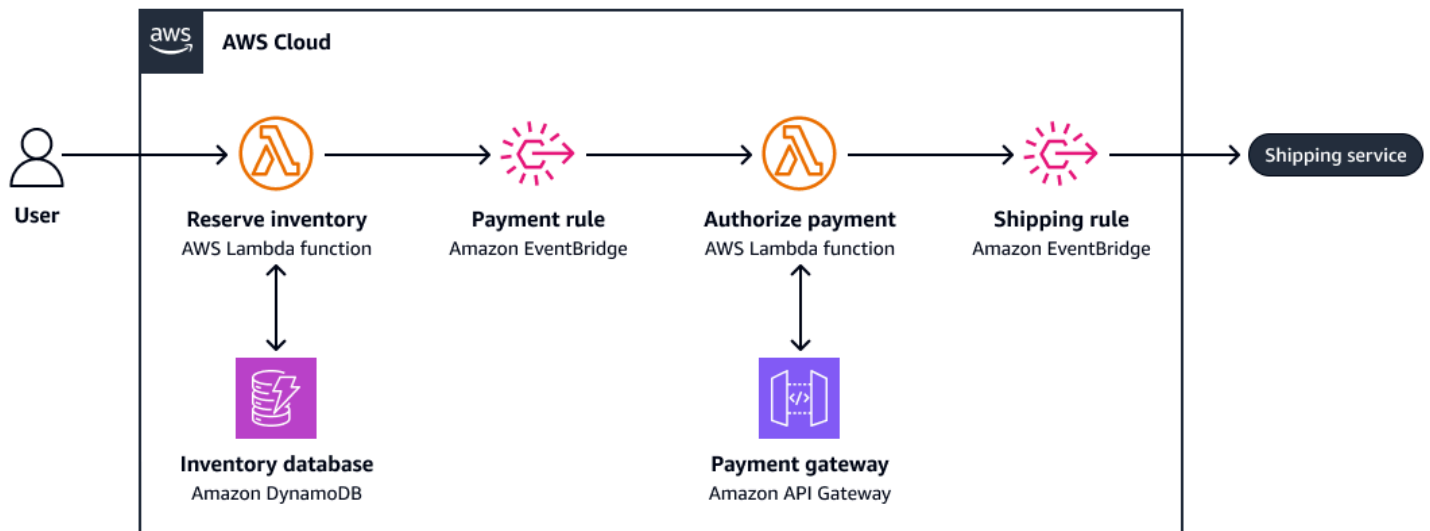
下圖是 LLM 提示鏈結的範例：



系列事件編排

saga 編排模式是沒有中央協調器的分散式系統中的實作方法。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中，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）。

此模式在事件驅動型架構中特別重要，其中服務鬆散耦合，而且即使存在部分故障，狀態仍必須隨著時間持續解決。

提示鏈結模式

提示鏈結在結構和目的上都類似 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

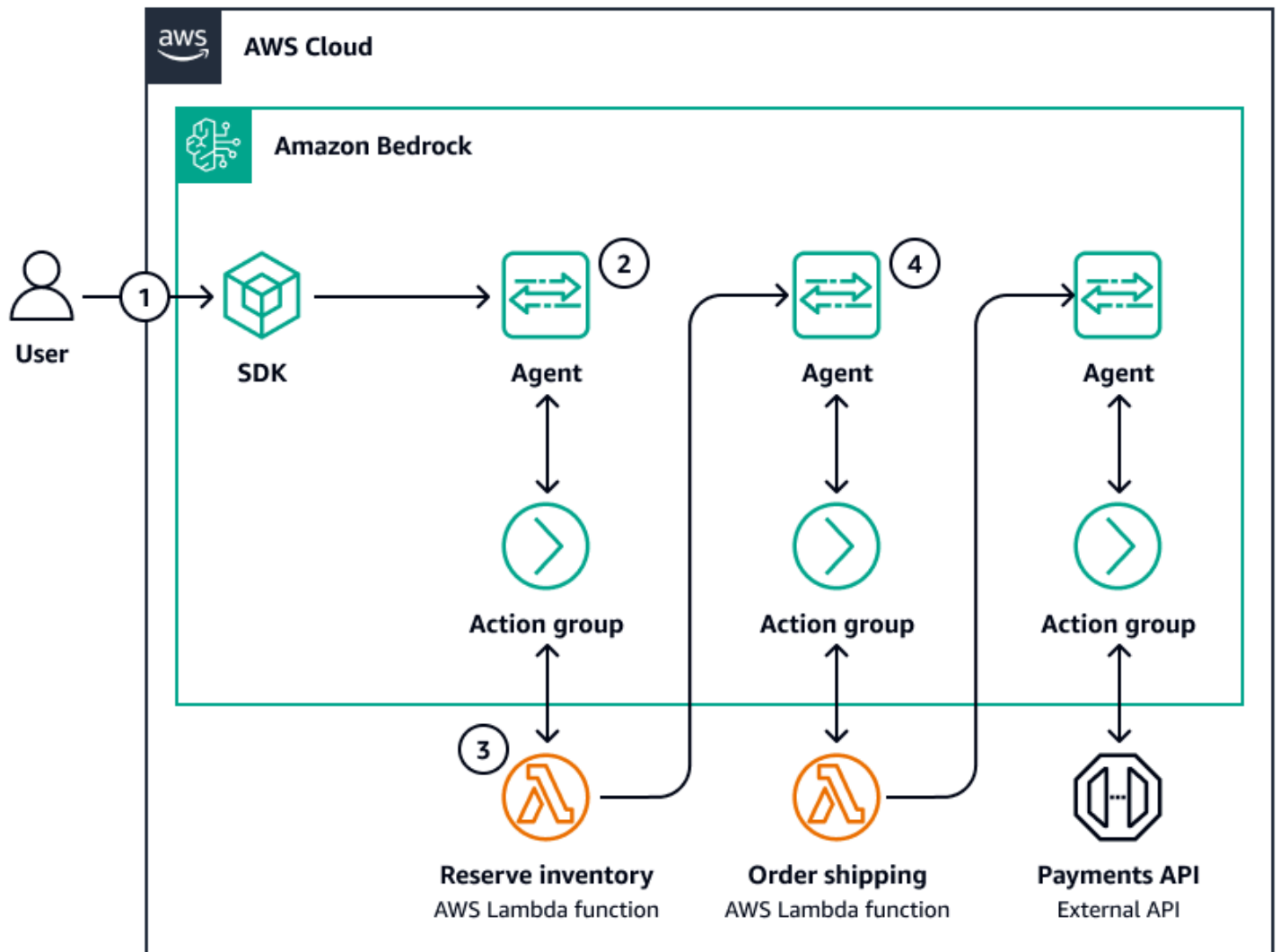
如果中繼結果有瑕疵，系統可以執行下列其中一項操作：

- 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回復機制。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

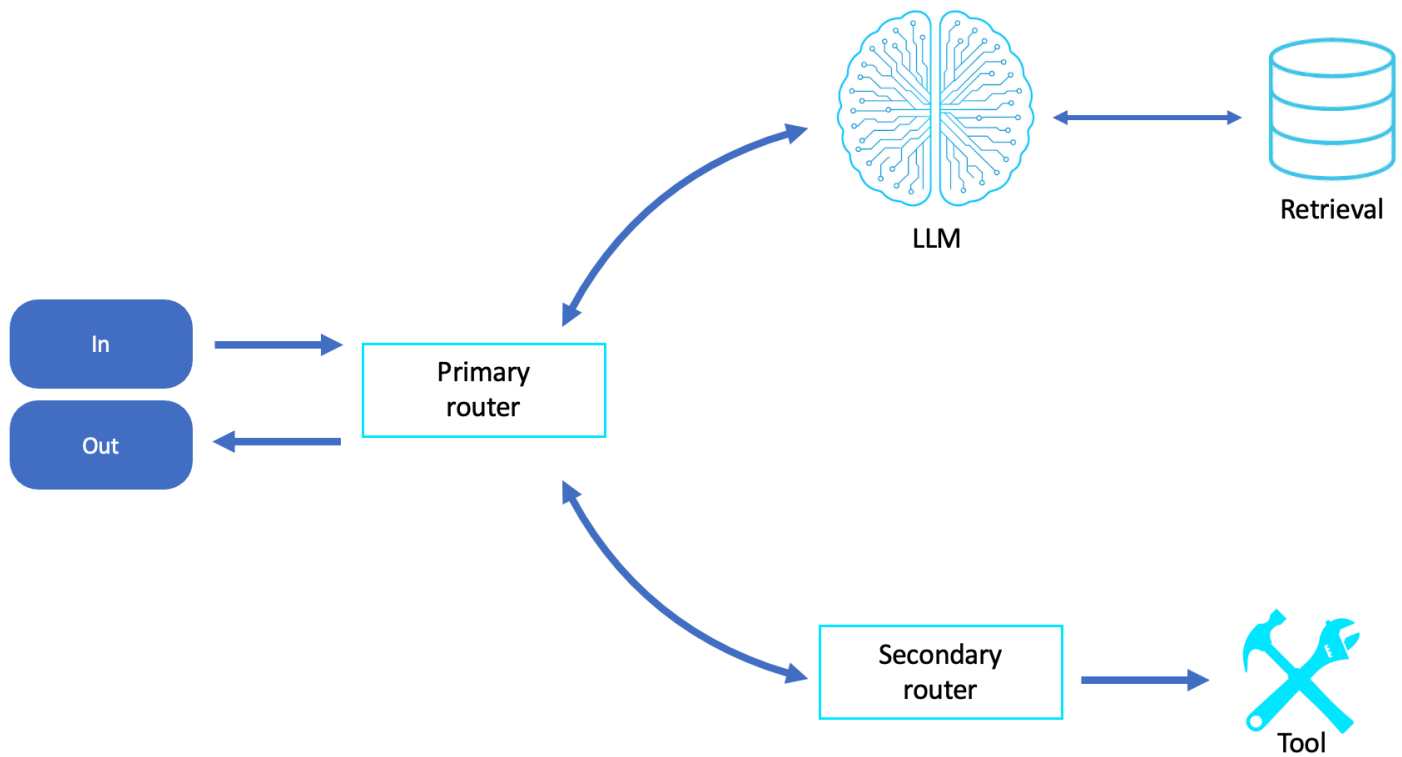
當 saga 模式使用補償邏輯管理分散式服務呼叫時，提示鏈結會使用反射排序和適應性重新規劃來管理推理任務。這兩個系統都允許累加式進度、分散式決策點和故障復原，並透過明智的推理而非嚴格復原來執行所有操作。

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。

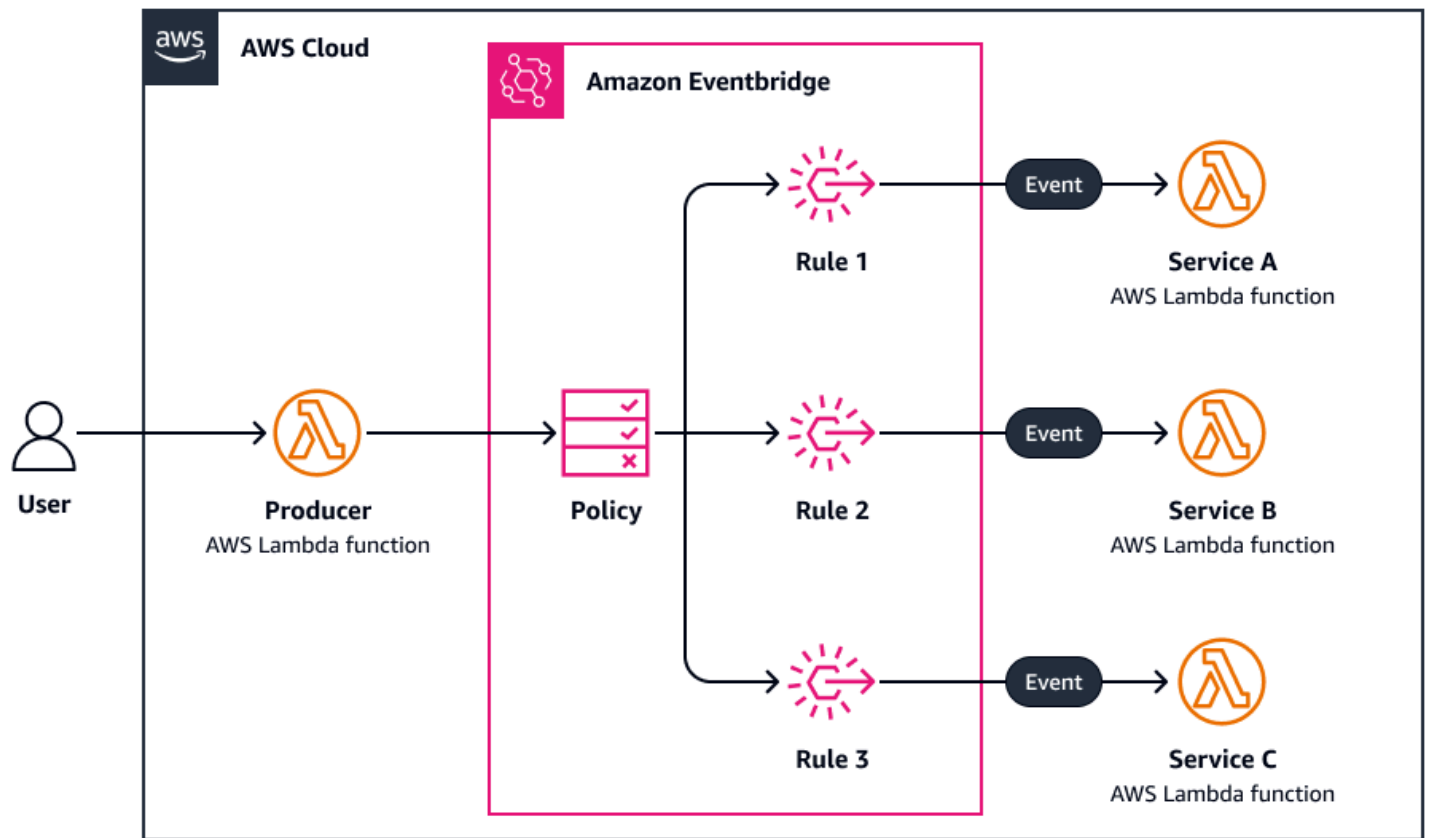
下圖是 LLM 路由的範例：



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

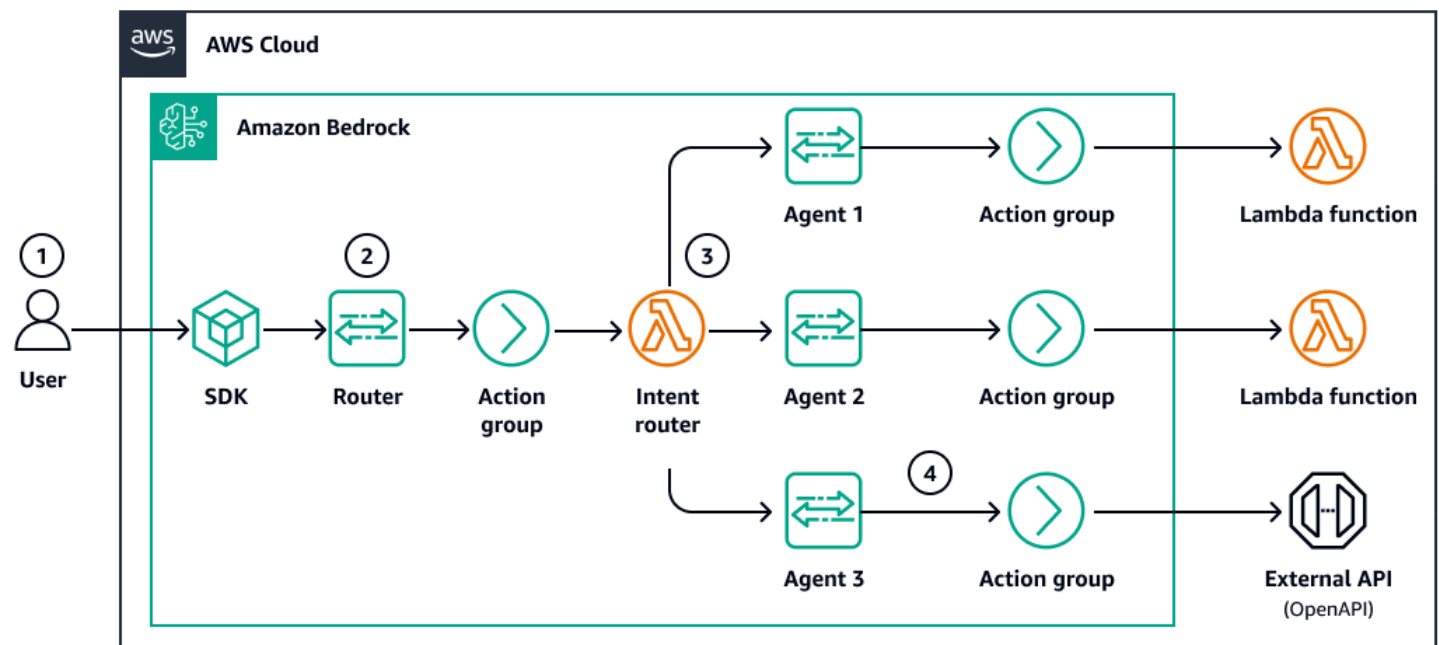
在代理系統中，路由也會執行動態任務委派，但 LLM 會透過自然語言分類和解譯使用者的意圖，而不是 Amazon EventBridge 規則或中繼資料篩選條件。結果是靈活、語意和適應性的分派形式。

代理程式路由器

此架構可啟用豐富的意圖型分派，無需預先定義的結構描述或事件類型，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：

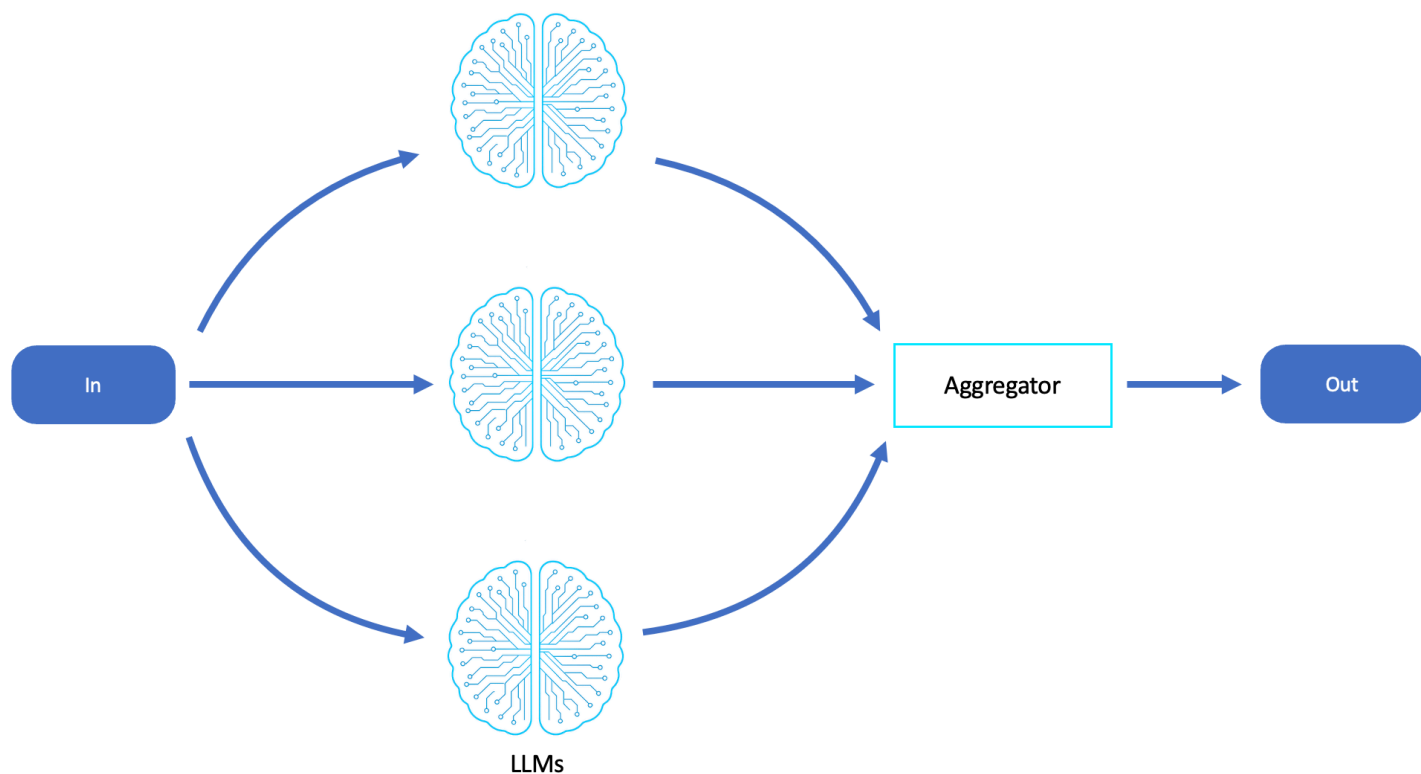
- 更廣泛的輸入理解
- 智慧型備用和工具選擇
- 透過新的客服人員角色或提示樣式的自然可擴展性

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很短。為了克服這個問題，可以使用事件驅動的散射集模式來重新構想 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：



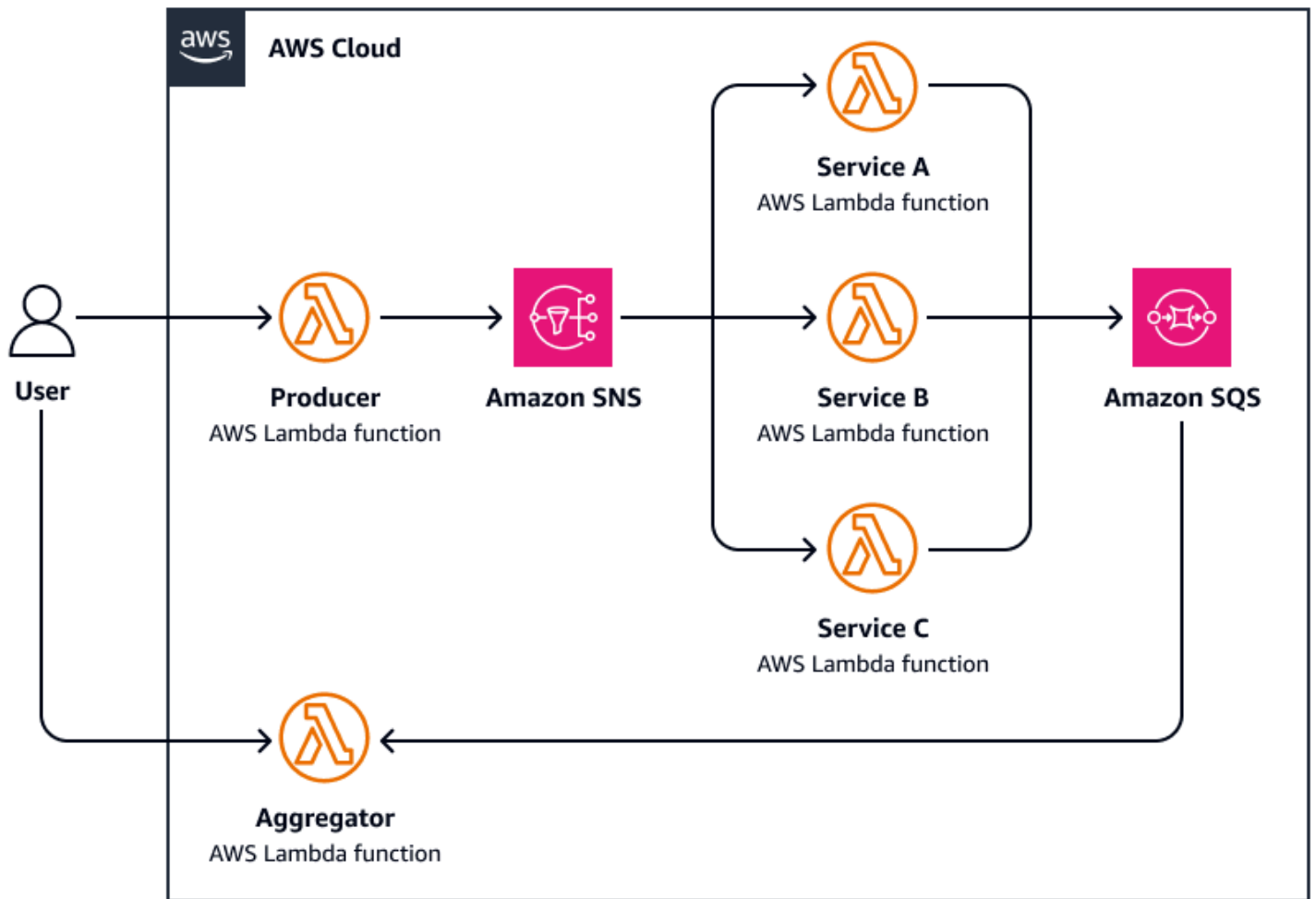
散佈集合

在分散式系統中，散佈集合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

下圖是散佈加法的範例：



1. 使用者將請求傳送到中央協調器函數，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，並會路由到專門的工作者 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. 彙總工具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）

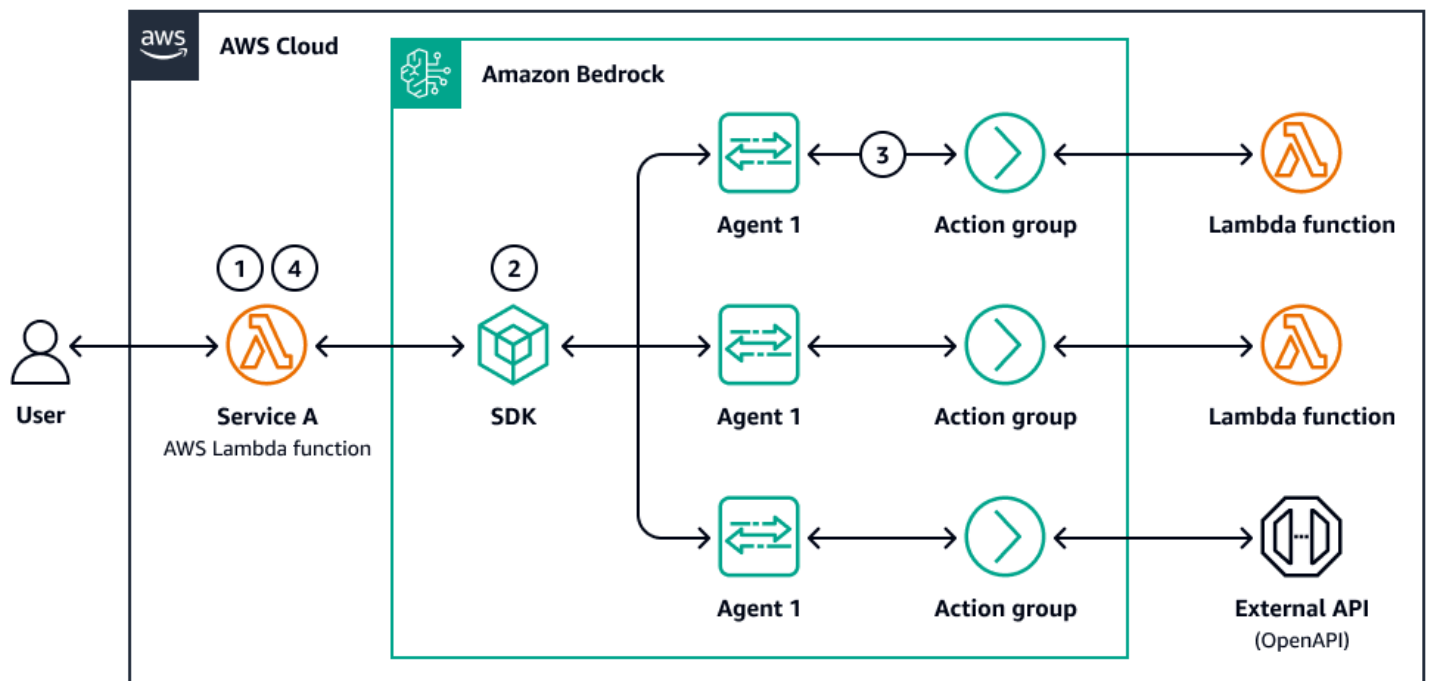
在代理程式系統中，平行處理會透過將子任務分散到多個 LLM 呼叫或代理程式來緊密反映散佈集，每個子任務都會獨立推理一部分的問題。傳回的結果會由彙總程序收集和合成，通常是另一個 LLM 或控制器代理程式。

客服人員平行處理

1. 客服人員提交請求「總結這 10 個報告的洞見」。
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. 系統會將最終回應傳回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。

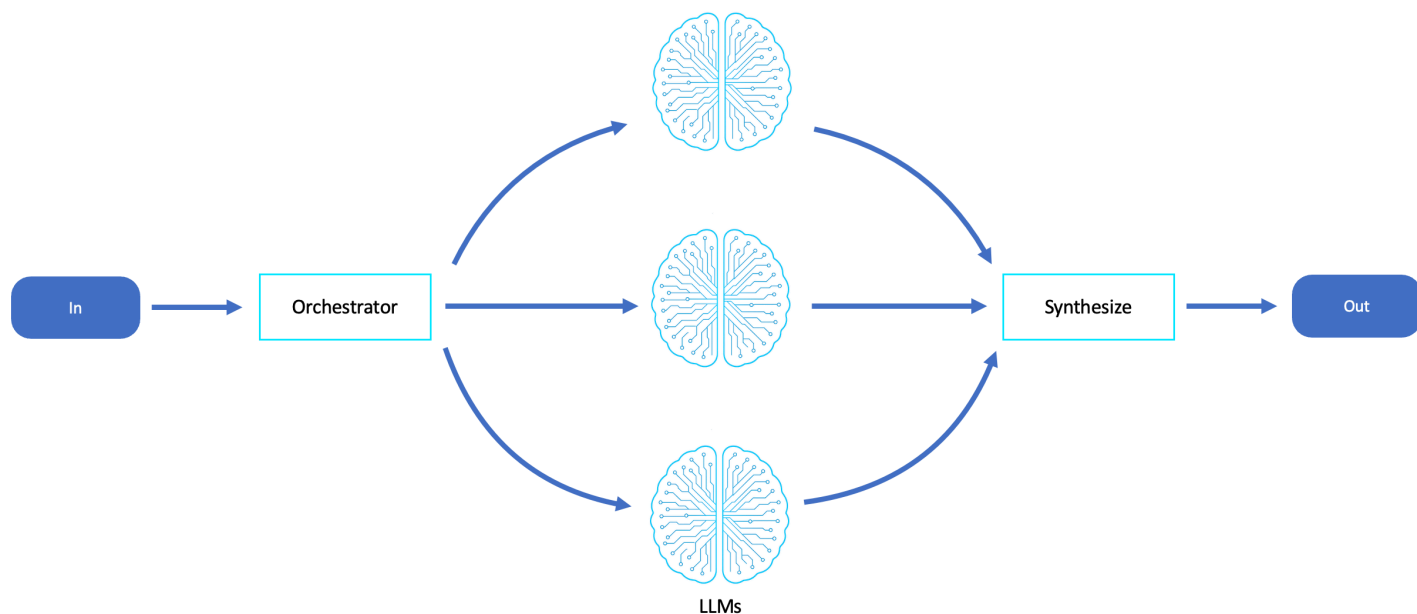
要點

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：



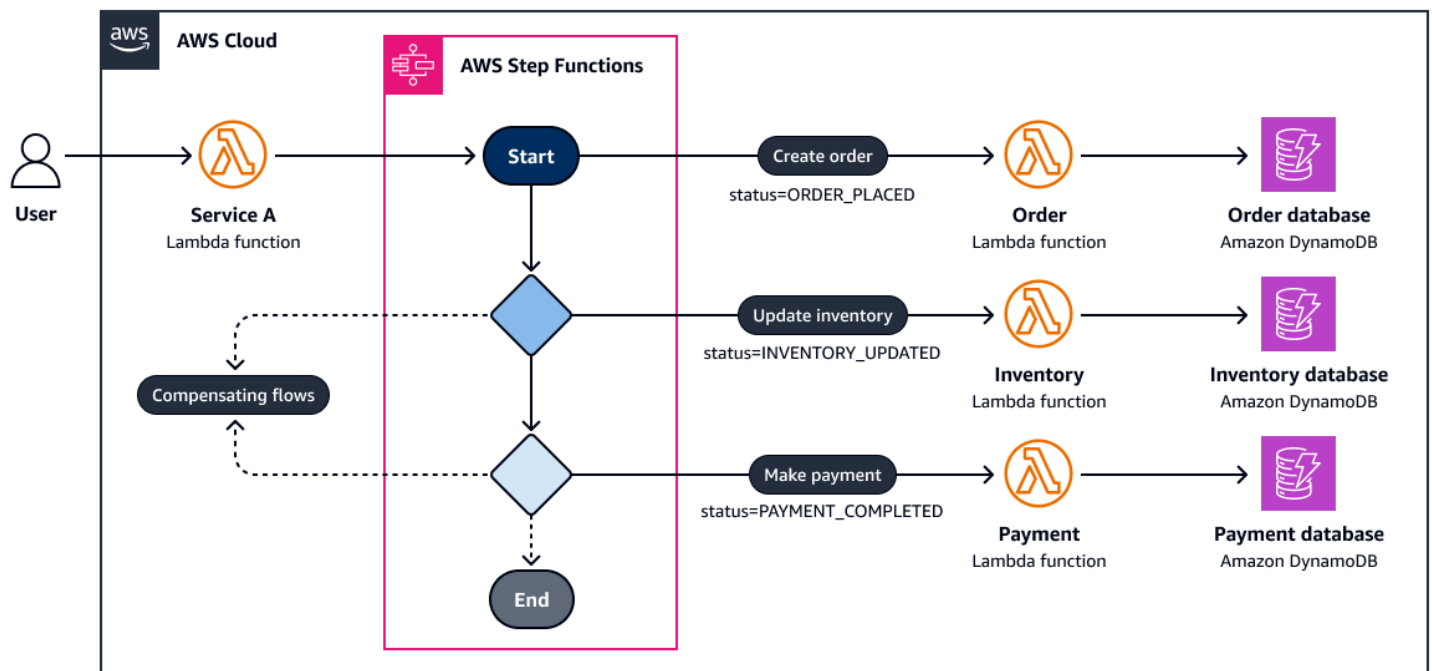
事件協同運作

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）。

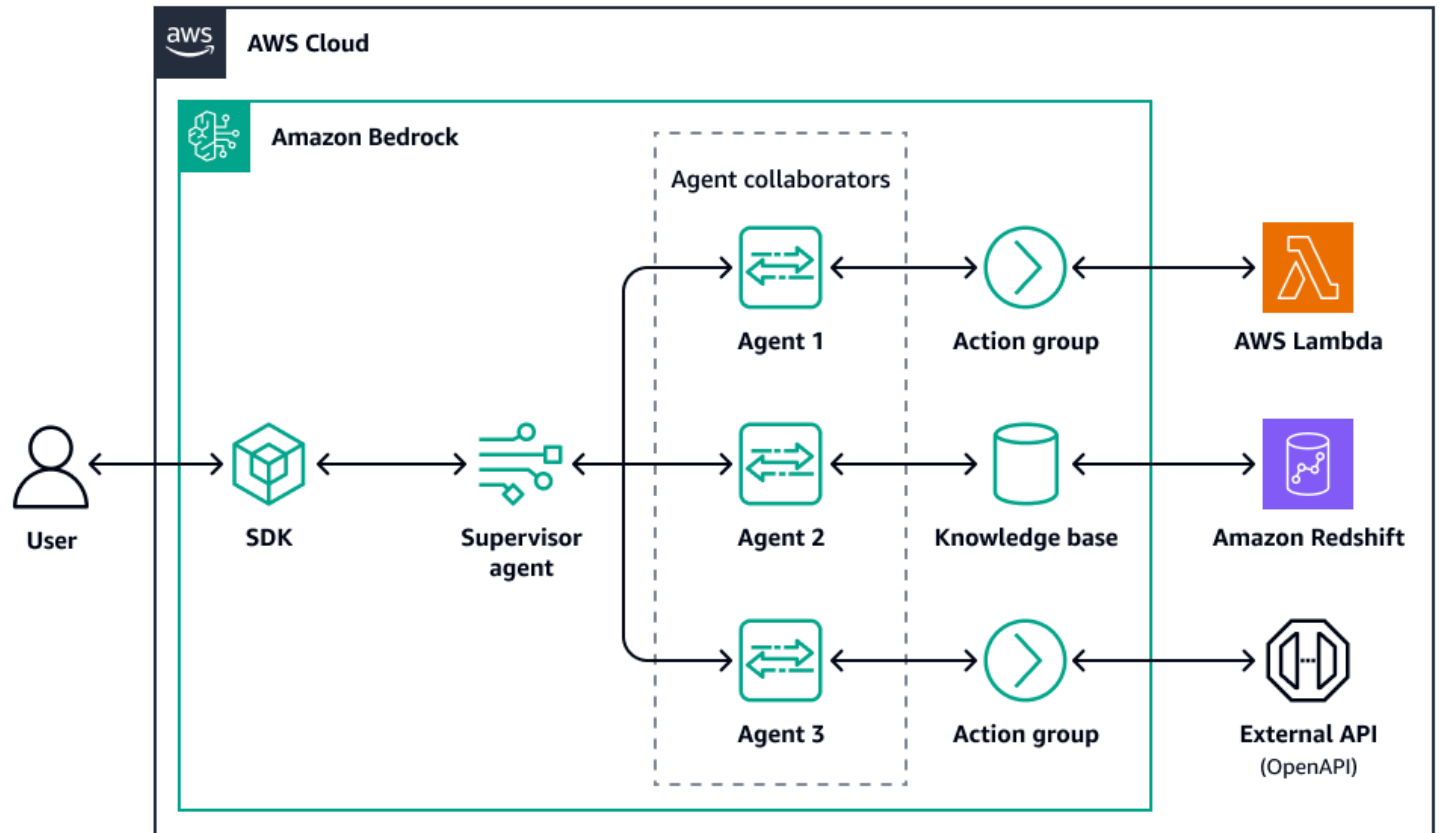
主管

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：

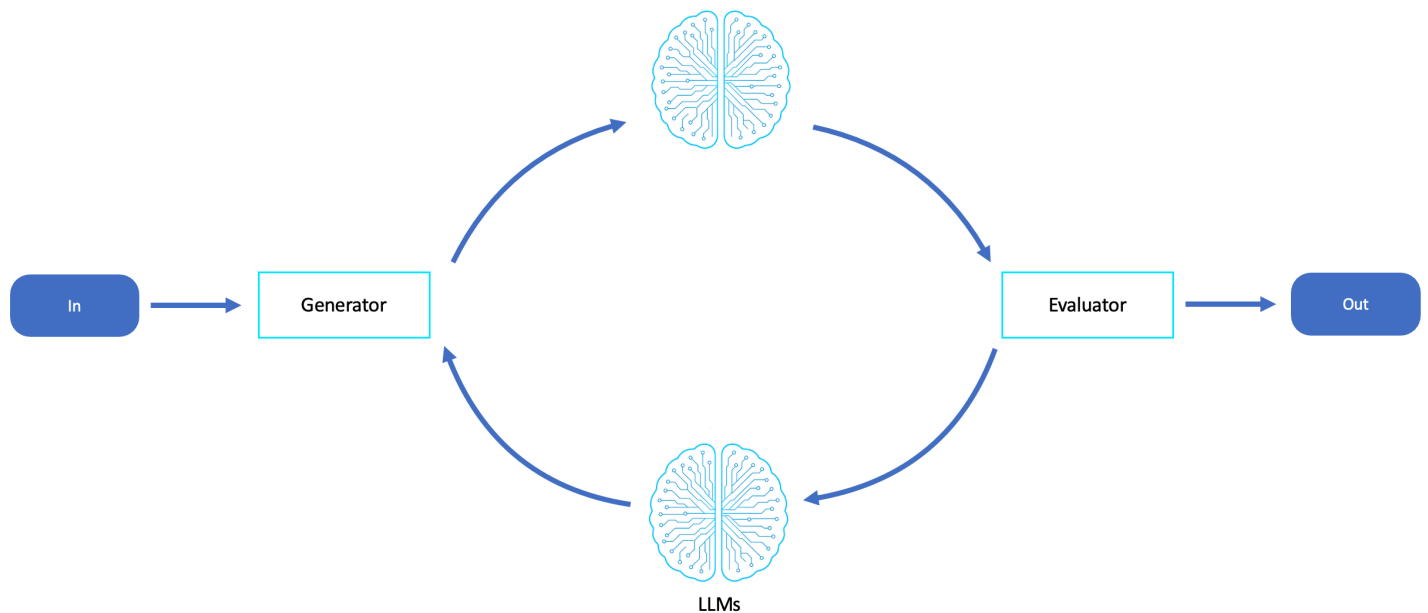
- 維護內容和執行流程
- 處理分支、排序和錯誤處理
- 從分散式元件產生統一的結果

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、模稜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時間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。

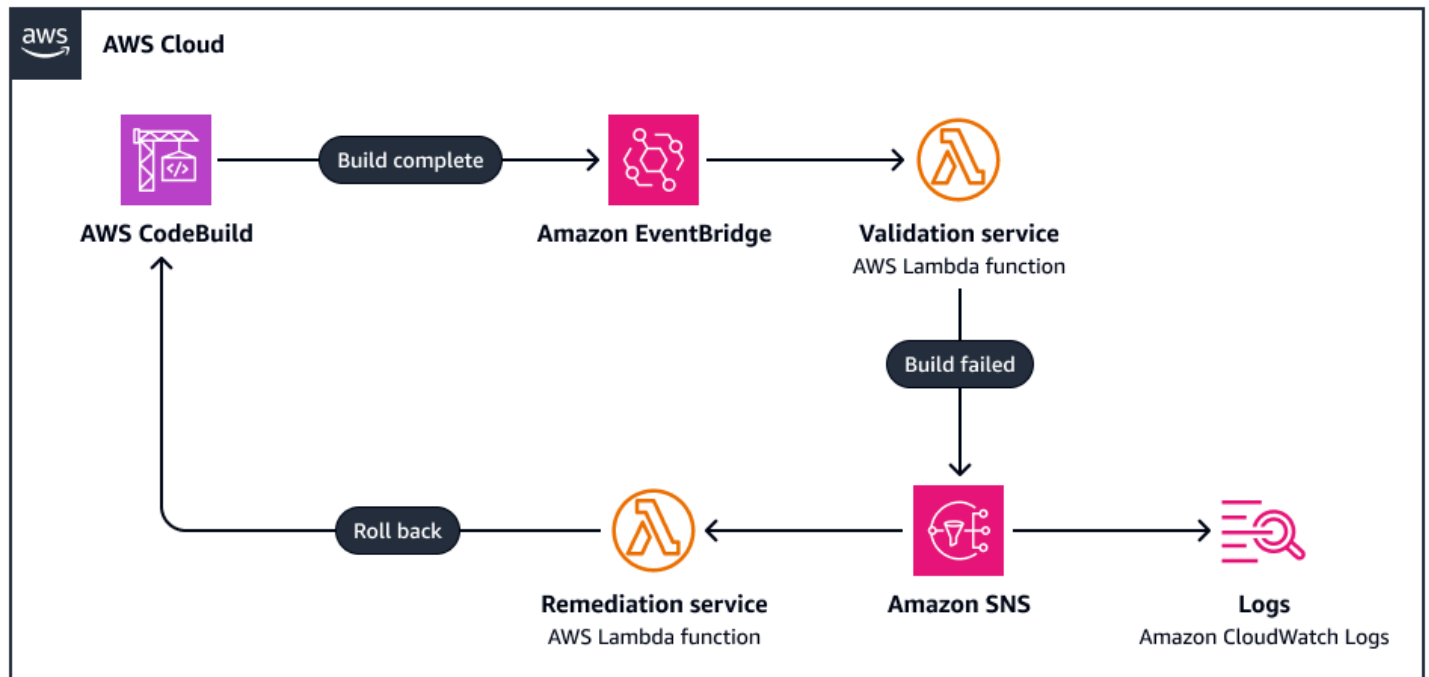
下圖是評估器反射改善意見回饋迴圈的範例：



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。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。

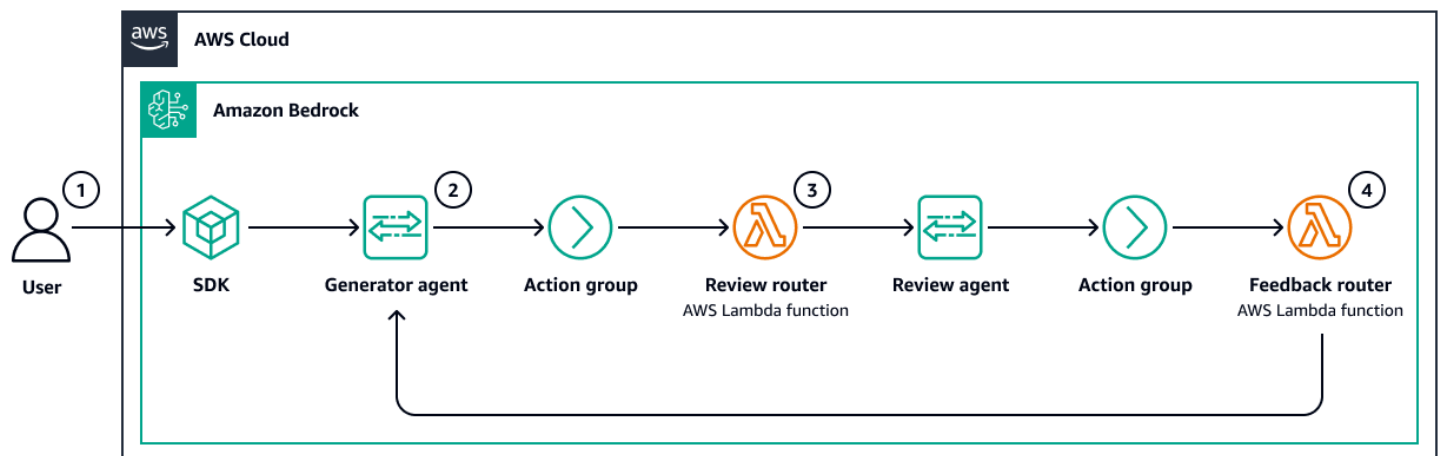
迴圈會重複，直到結果符合一組條件、獲得核准或達到重試限制為止。

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、反覆精簡和適應性輸出控制，無需人工輸入。

下圖是意見回饋控制迴圈（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. 第二個代理程式（或後續提示）會執行結構化評估（例如，「透過清晰度、完整性和音調對此輸出進行評分」）。
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 workflows 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
協調	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 軟體開發套件	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演變階段，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，以目的回應並從經驗中學習。

在這些模式中，Amazon Bedrock 是代理程式認知的開道，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

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。