



Agentic AI 模式和 workflows 已开启 AWS

AWS 规范性指导



AWS 规范性指导: Agentic AI 模式和工作流程已开启 AWS

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

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆、贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

Table of Contents

简介 1

目标受众 1

目标 1

关于此内容系列 1

代理模式 2

基本推理代理 3

架构 3

说明 4

功能 5

限制 5

常见使用案例 5

实施指导 5

Summary 6

架构 6

说明 7

功能 8

常见使用案例 8

实施指导 8

Summary 8

Tool-based 用于调用函数的代理 9

架构 9

说明 10

功能 11

常见使用案例 11

实施指导 11

Summary 11

Tool-based 服务器代理 12

架构 12

说明 12

功能 13

常见使用案例 13

实施指导 14

Summary 14

Computer-use 代理人 14

架构	14
说明	15
功能	16
常见使用案例	16
实施指导	16
Summary	17
编码代理	17
架构	17
说明	18
功能	19
常见使用案例	19
实施指导	19
Summary	20
语音和语音代理	20
架构	20
说明	21
功能	22
常见使用案例	22
实施指导	22
Summary	23
工作流程编排代理	23
架构	23
说明	24
功能	25
常见使用案例	25
实施指导	25
Summary	25
Memory-augmented 代理人	26
架构	26
说明	26
功能	27
常见使用案例	27
实现内存增强代理	28
实现注入内存的提示	28
Summary	29
仿真和测试平台代理	29

架构	29
说明	30
功能	31
常见使用案例	31
实施指导	31
Summary	32
观察者和监视代理	32
架构	33
说明	33
功能	34
常见使用案例	34
实施指导	34
Summary	35
Multi-agent 合作	35
说明	37
功能	38
常见使用案例	38
实施指导	38
Summary	39
结论	39
外卖	39
法学硕士工作流程	40
LLM-augmented 认知概述	40
提示链接的工作流程	41
说明	42
功能	42
常见使用案例	42
路由工作流程	42
功能	43
常见使用案例	44
并行化工作流程	44
功能	45
常见使用案例	45
编排工作流程	45
功能	46
常见使用案例	46

赋值器的工作流程和反射优化循环	46
常见使用案例	47
功能	48
结论	48
代理工作流程模式	49
从事件驱动到认知增强系统	49
Event-driven 建筑	49
Cognition-augmented 工作流程	50
核心见解	52
提示链接传奇模式	52
saga 编配	52
提示链接模式	53
特工编舞	53
外卖	55
路由动态调度模式	55
动态调度	56
LLM-based 路由	57
代理路由器	58
外卖	59
并行化和分散-聚集模式	59
Scatter-gather	60
LLM-based 并行化 (分散-聚集认知)	61
代理并行化	61
外卖	62
传奇编排模式	62
活动编排	63
Role-based 代理系统 (协调器)	64
主管	64
外卖	66
赋值器反射精细化环路模式	66
反馈控制回路	67
反馈控制回路 (评估器)	68
评估者	68
外卖	69
在上设计代理工作流程 AWS	69
结论	70

文档历史记录 71

..... lxxii

Agentic AI 模式和 workflows 已开启 AWS

Aaron Sempf 和 Andrew Hooker , Amazon Web Services

2025 年 7 月 ([文档历史记录](#))

Organizations 正在采用大型语言模型 (LLMs) 和软件代理，使用一种名为代理模式的新架构学科来解决动态的多域问题。代理模式是基础蓝图和模块化结构，用于在许多环境中设计和编排以目标为导向的 AI 代理。

目标受众

本指南适用于想要构建超越静态逻辑、符号逻辑和确定性自动化的智能应用程序的架构师、开发人员和产品负责人。

目标

本指南提供了 AI 代理系统的设计框架和实现方法，这些系统可以自主运行，同时保持可控性并与您的目标保持一致。它将事件驱动的架构模式与各种代理替代方案联系起来，演示了如何使用云原生架构构建生产级代理系统。本指南讨论了以下主题：

- 代理模式 — 代理模式是可重复使用的设计模板，用于描述各个代理的结构和行为。这包括推理代理、检索增强代理、编码代理、语音接口、工作流程协调器和协作式多代理系统。每种模式都说明了代理如何感知、推理、行为和学习，并映射到这些 AWS 服务模式中。
- 法学硕士工作流程 — 工作流程侧重于代理如何使用推理 LLMs 。他们探讨了提示策略和计划机制，并概述了如何 LLMs 不仅用于生成文本，还用于在代理循环中推动结构化、可解释和可靠的行为。
- 代理工作流程模式 — 工作流模式描述了多个代理、工具和环境如何相互作用以形成自治系统。这包括任务编排、子代理委派、基于事件的协调、可观察性和控制的模式。这些方面促进了可扩展、可组合和可审计的 AI 架构。

关于此内容系列

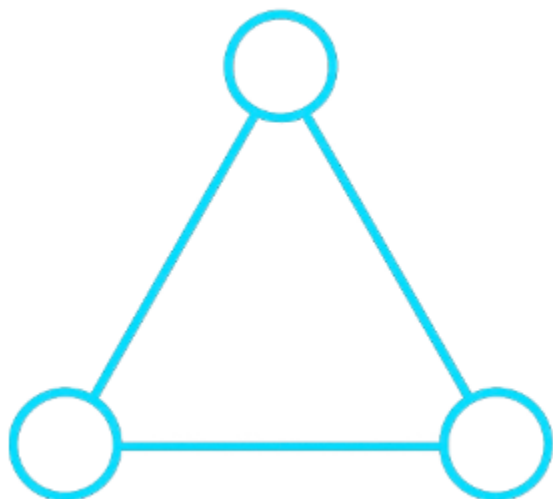
本指南是关于代理人工智能的系列文章的一部分。AWS 要了解更多信息并查看本系列中的其他指南，请参阅 AWS 规范性指导网站上的 [Agentic AI](#)。

代理模式

代理模式是可重复使用的可组合构造块，可以根据特定的领域、用例和复杂程度进行定制。但是，代理系统与传统应用不同。所有 AI 代理设计的核心是基于以下三个基本原则的概念模型：

- 异步 — 代理在松散耦合、事件丰富的环境中运行
- 自主权 — 代理独立行动，无需人为或外部控制
- 代理机构 — 代理代表用户或系统有目的地朝着特定目标行事

下图中的三角形表示软件代理的核心组成部分：感知、理性和行动。这使代理系统能够在其环境中进行观察、做出决策和采取行动。



通过设计，代理模式为构建 AI 系统提供了一种模块化设计语言，这意味着它们易于访问、可操作、可扩展并且可以投入生产。设计这些系统需要仔细注意以下三个相互关联的维度，本指南稍后将对此进行进一步讨论。

本节内容

- [基本推理代理](#)
- [Tool-based 用于调用函数的代理](#)
- [Tool-based 服务器代理](#)
- [Computer-use 代理人](#)
- [编码代理](#)
- [语音和语音代理](#)

- [工作流程编排代理](#)
- [Memory-augmented 代理人](#)
- [仿真和测试平台代理](#)
- [观察者和监视代理](#)
- [Multi-agent 合作](#)

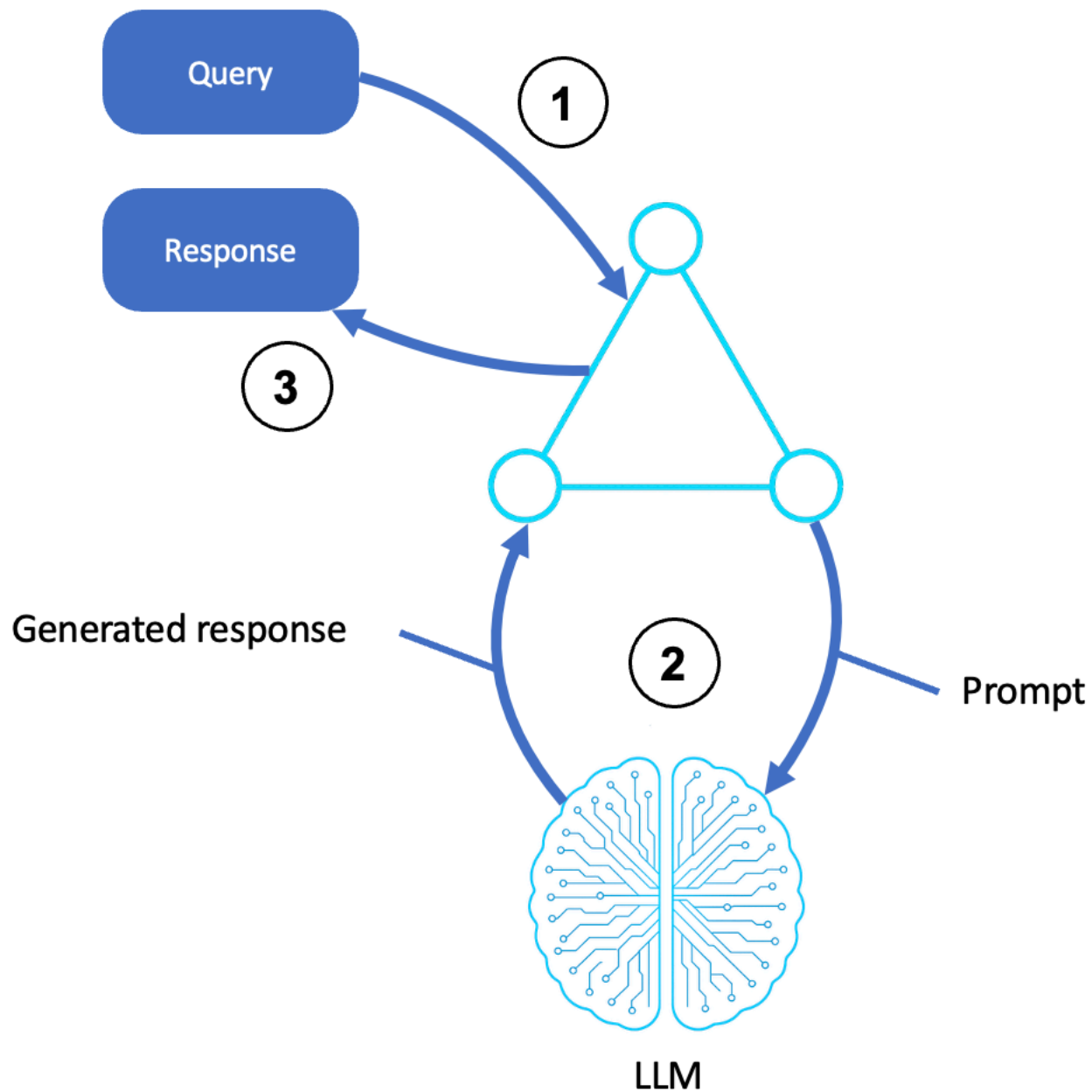
基本推理代理

基本推理代理是最简单的代理人工智能形式，它根据查询执行逻辑推理或决策。它接受用户或系统的输入，处理查询，并使用结构化提示生成响应。

这种模式对于需要根据给定上下文进行单步推理、分类或总结的任务很有用。它不使用内存、工具或状态管理，这使得它在大型工作流程中具有无状态、轻量级且高度可组合。

架构

基本推理代理的流程如下图所示：



说明

1. 接收输入

- 用户、系统或上游代理提交查询或指令。
- 输入将移交给代理 shell 或编排层。
- 此步骤包括任何预处理、提示模板和目标识别。

2. 调用 LLM

- 代理将查询转换为结构化提示并将其发送给 LLM (例如 , 通过 Amazon Bedrock) 。
- LLM 使用预先训练的知识和上下文根据提示生成响应。
- 生成的输出可能包括推理步骤 (思维链) 、最终答案或排名选项。

3. 返回响应

- 生成的输出将中继到代理的接口。
- 这可能包括格式化、后处理或 API 响应。

功能

- 支持自然语言或结构化输入
- 使用提示工程来指导行为
- 无状态且可扩展
- 可以嵌入到用户界面、CLI、API 和管道中

限制

- 没有记忆力或历史意识
- 不与外部工具或数据源交互
- 仅限于法学硕士在推理时所知道的情况

常见使用案例

- 对话问题和答案
- 政策解释和摘要
- 决策指南
- 轻量级且自动化的聊天机器人流程
- 分类、标签和评分

实施指导

您可以使用以下工具和服务来创建基本的推理代理：

- 用于调用 LLM 的 Amazon Bedrock (Anthropic、AI21、Meta)

- Amazon API Gateway 或者 AWS Lambda 将其作为无状态微服务公开
- 提示模板存储在参数存储库中 AWS Secrets Manager，或存储为代码

Summary

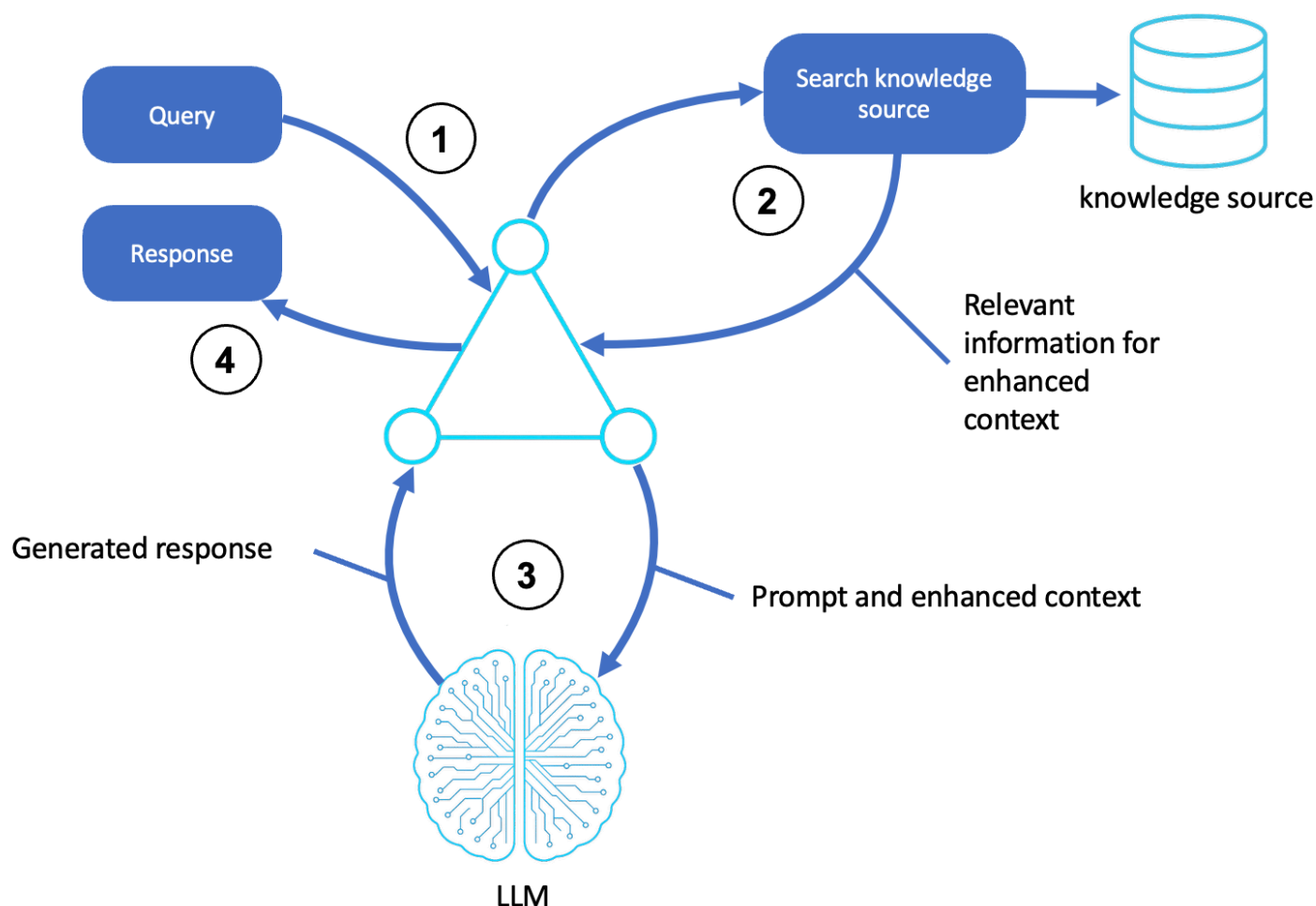
由于其结构简单，基本的推理代理是基础性的。它具有核心功能，可以将目标转化为推理路径，从而实现智能输出。这种模式通常是高级模式起点，例如基于工具的代理和使用检索增强生成 (RAG) 的代理。它也是大型工作流程的可靠模块化组件。

RAG 探员

Retrieval-augmented 生成 (RAG) 是一种将信息检索与文本生成相结合的技术，可创建准确的上下文相关响应。RAG 使代理能够在聘请法学硕士之前检索相关的外部信息。它通过将最新、事实或特定领域的信息作为决策的基础，从而扩展代理的有效记忆力和推理准确性。与仅依赖预训练权重的无状态 LLM 形成鲜明对比的是，RAG 有一个外部知识搜索层，可以根据上下文动态增强提示。

架构

RAG 模式的逻辑如下图所示：



说明

1. 收到查询

- 用户或上游系统向代理提交查询或目标。
- 代理 shell 接受请求并将其格式化为推理提示。

2. 搜索外部来源

- 代理从查询中识别概念和意图。
- 它使用语义搜索或关键字匹配来查询知识源，例如矢量存储、数据库或文档索引。
- 检索最相关的段落、文档或实体，以便在下一步中使用。

3. 生成上下文响应

- 代理使用检索到的信息对提示进行扩充，从而为 LLM 形成上下文增强输入。
- 法学硕士使用生成推理（例如思维链或反思）处理任何输入，以产生准确的响应。

4. 返回最终输出

- 代理通过将输出封装在任何通信标头或所需格式中来准备输出，然后将其返回给用户或呼叫系统。
- (可选) 检索到的文档和 LLM 输出可能会被记录、评分并存储在内存中以备将来查询。

功能

- Fact-grounded 即使在长尾域或企业特定域中也能输出
- 无需微调模型即可扩展内存
- 基于每个查询和用户状态的动态上下文
- 与矢量数据库、语义索引和元数据筛选完全兼容

常见使用案例

- 企业知识助手
- 监管合规机器人
- 客户支持副驾驶
- Search-enhanced 聊天机器人
- 开发者文档代理

实施指导

使用以下工具和服务创建使用 RAG 的代理：

- 用于调用 LLM 的 Amazon Bedrock
- 用于文档或结构化数据搜索的 Amazon Kendra 或 Amazon Aurora OpenSearch
- 用于存储文档的亚马逊简单存储服务 (Amazon S3) Service
- AWS Lambda 编排搜索、提示和 LLM 推理
- Knowledge-based 与代理集成 (通过使用内存插件、语义检索器或 Amazon Bedrock)

Summary

Agent RAG 将静态模型推理与动态的现实世界智能联系起来。它使代理能够查找他们不知道的内容，从检索到的知识中综合答案，并生成高度可信、可审计的响应。

RAG 模式是构建无需再培训即可扩展知识访问权限的智能代理的基础。它通常是更复杂的编排模式的前身，涉及工具使用、计划和长期记忆。

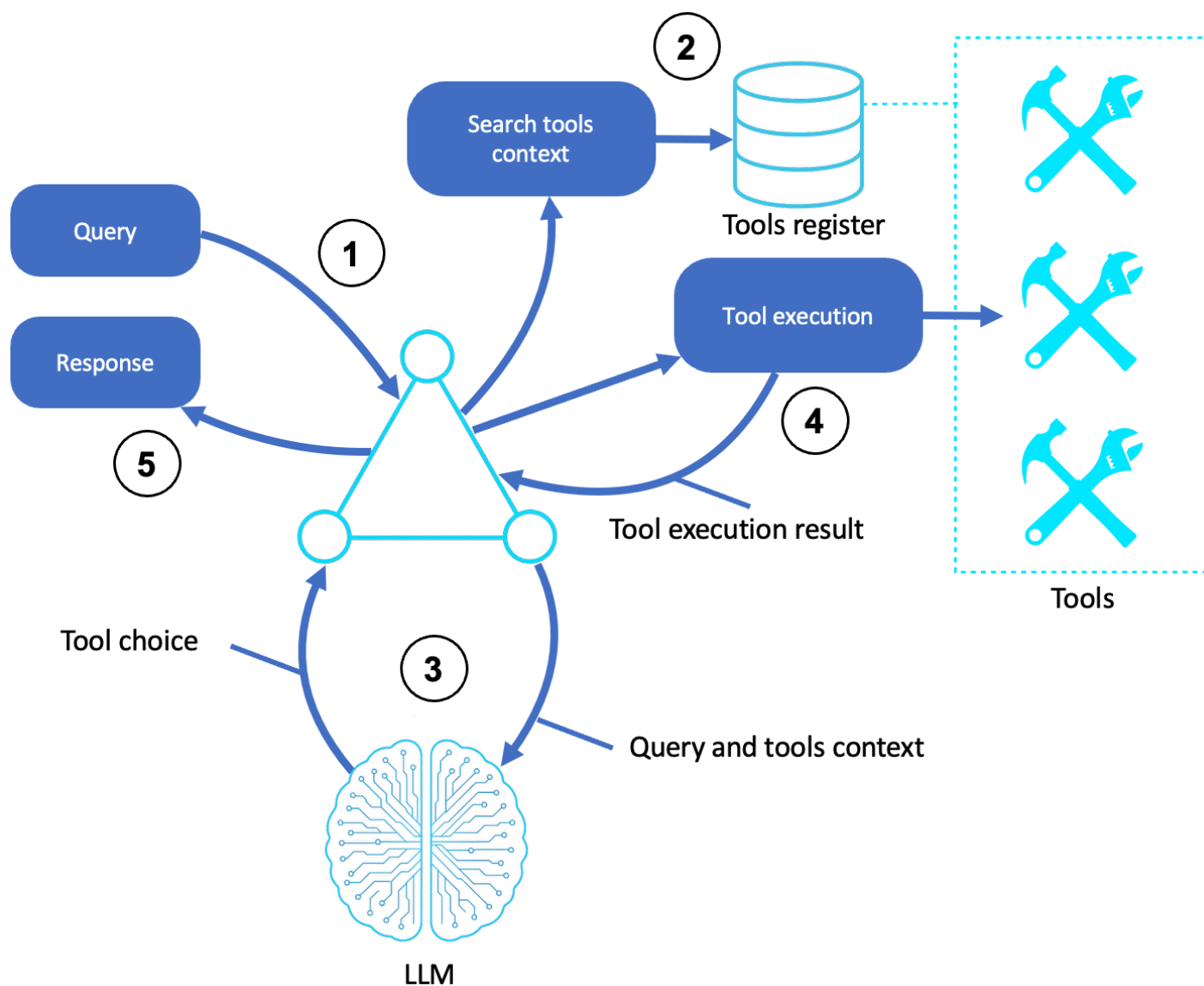
Tool-based 用于调用函数的代理

Tool-based 代理通过调用外部函数或 API 来完成超出纯语言推理范围的任务，从而扩展推理代理的功能。此模式使用 LLM 来决定使用哪个工具，然后生成调用参数并将工具的输出合并到其推理循环中。

这种模式使代理能够采取行动，而不仅仅是提供响应。工具接口代表任何可调用的功能，从算术计算和数据库查找到外部 API 和云服务。

架构

下图显示了用于调用函数的基于工具的代理：



说明

1. 接收查询

- 代理接收来自用户或呼叫系统的自然语言查询或任务。

2. 搜索工具

- 代理使用内部元数据或工具注册表来搜索可用的工具、架构和相关功能。

3. 选择和调用工具

- LLM 在其提示中接收查询和工具元数据（例如，函数名称、输入类型和描述）。
- 它选择最相关的工具，构造输入参数，然后返回结构化函数调用。

4. 运行所选工具

- 代理 shell 或工具运行器执行选定的函数并返回结果（例如，API 输出、数据库值或计算）。

5. 返回响应

- LLM 直接或作为更新提示的一部分将结果传递给代理。然后，它会返回自然语言结果。

功能

- 基于任务上下文的动态工具选择
- Schema-based 提示（OpenAPI、JSON 架构 AWS、函数接口）
- 结果解释并将输出链接到推理中
- 无状态或会话感知型操作

常见使用案例

- 具有外部数据访问权限的虚拟助手
- 财务计算器和估算器
- API-based 知识工作者
- 调用 AWS Lambda Amazon SageMaker 终端节点和 SaaS 服务的 LLM

实施指导

使用以下命令创建用于调用函数的基于工具的代理：

- 支持函数调用的 Amazon Bedrock（Anthropic Claude）
- AWS Lambda 作为工具执行后端
- Amazon API Gateway 或 AWS Step Functions 用于工具编排
- 用于情境感知工具元数据的亚马逊 DynamoDB 或亚马逊关系数据库服务 (Amazon RDS)
- Amazon EventBridge 管道或 AWS Step Functions 将状态映射到路由输出的管道

Summary

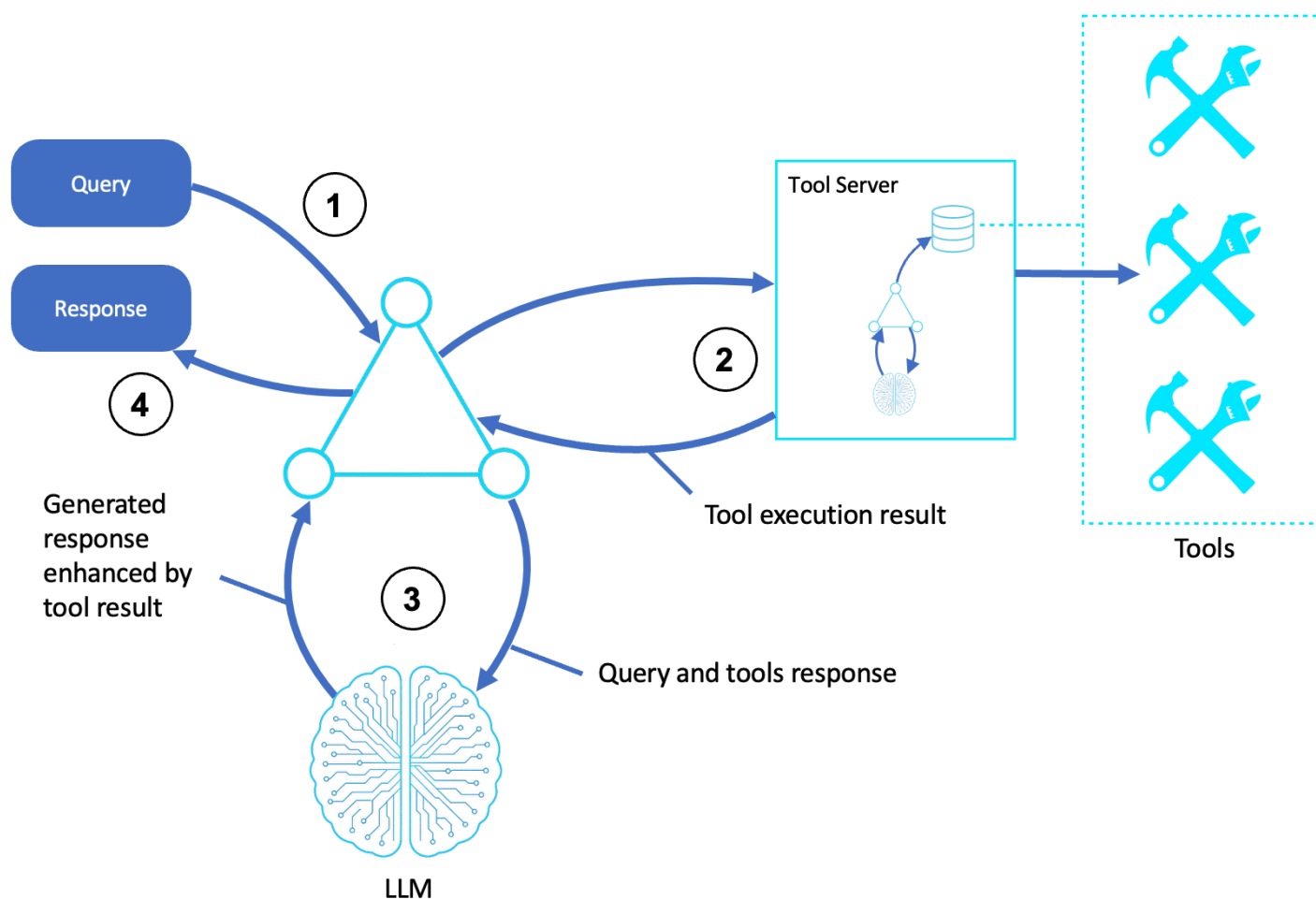
Tool-based 函数调用代理代表了从理解语言到执行操作的转变。这些代理调用动态的情境感知工具，同时保持法学硕士推理，将被动助手转变为完成任务、访问服务和整合业务运营的系统。这种模式是企业环境中代理人工智能的重要组成部分，尤其是在与声明式架构、授权框架和多代理系统结合使用时。

Tool-based 服务器代理

Tool-based 服务器代理通过将工具执行委托给具有工具、脚本和复合代理专用运行时环境的外部服务器来增强函数调用代理。与代理循环选择和调用的内联函数调用不同，基于服务器的代理将逻辑和执行管道外包给其他代理或系统。这提供了高级功能，例如多工具链接、隔离执行和专业推理。工具服务器非常适合复杂、有状态或资源密集型操作，在这些操作中，工具本身可能涉及单独的 AI 模型、业务规则或环境。

架构

以下是基于工具的服务器代理的模式：



说明

1. 接收查询

- 用户或系统向代理 shell 提交请求。

- 代理解释查询并准备将其发送到工具服务器。
2. 运行工具服务器进程
 - 代理将任务以及结构化参数发送到工具服务器。
 - 然后，工具服务器可以：
 - 在专用计算系统（例如，容器或 Amazon SageMaker）中运行脚本或逻辑 AWS Lambda
 - 使用自己的子代理和 LLM 推理来选择和运行工具
 - 管理依赖关系、重试或多步骤执行流程
 - 任务完成后将结果输出到主代理
 3. 在工具输出中使用 LLM 推理
 - 代理调用 LLM，将原始查询和工具服务器结果作为提示的一部分传递。
 - 法学硕士综合了包含新获得信息的响应。
 4. 返回响应
 - 代理向用户或呼叫系统返回自然语言或结构化响应。
 - （可选）结果可以存储在内存或审计日志中。

功能

- 工具是在主代理执行循环之外调用的
- 工具执行可能涉及 LLM 调用、逻辑链或子代理
- 代理充当控制器或调度员，而不仅仅是工具包装器
- 实现可组合性、可扩展性和逻辑隔离

常见使用案例

- 编排模型链（例如，通过组合 LLM、vision 和代码）
- AI-driven 自动化管道
- DevOps 带有脚本运行器的助理代理
- 复杂的财务计算、模拟或优化代理
- 多模态工具（例如，通过组合音频、文档和操作）

实施指导

您可以使用以下方法构建此模式 AWS 服务：

- Amazon Bedrock (代理主机和 LLM 推理)
- AWS Lambda、Amazon ECS 或亚马逊 SageMaker 终端节点作为工具服务器运行时 AWS Fargate
- Amazon API Gateway 或者 AWS App Runner 用于公开工具服务器 API
- Amazon EventBridge 用于分离代理到工具的消息传递
- AWS Step Functions 或者 AWS AppFabric 用于在工具服务器上编写多代理逻辑

Summary

Tool-based 使用服务器的代理具有高度的模块化和可扩展性。它们将决策逻辑与执行分离，从而允许主代理保持轻量级，同时将复杂或敏感的操作转移到其他系统。这对于企业级代理人工智能非常重要，尤其是在需要治理、可观察性、隔离、动态组合或其任意组合的环境中。

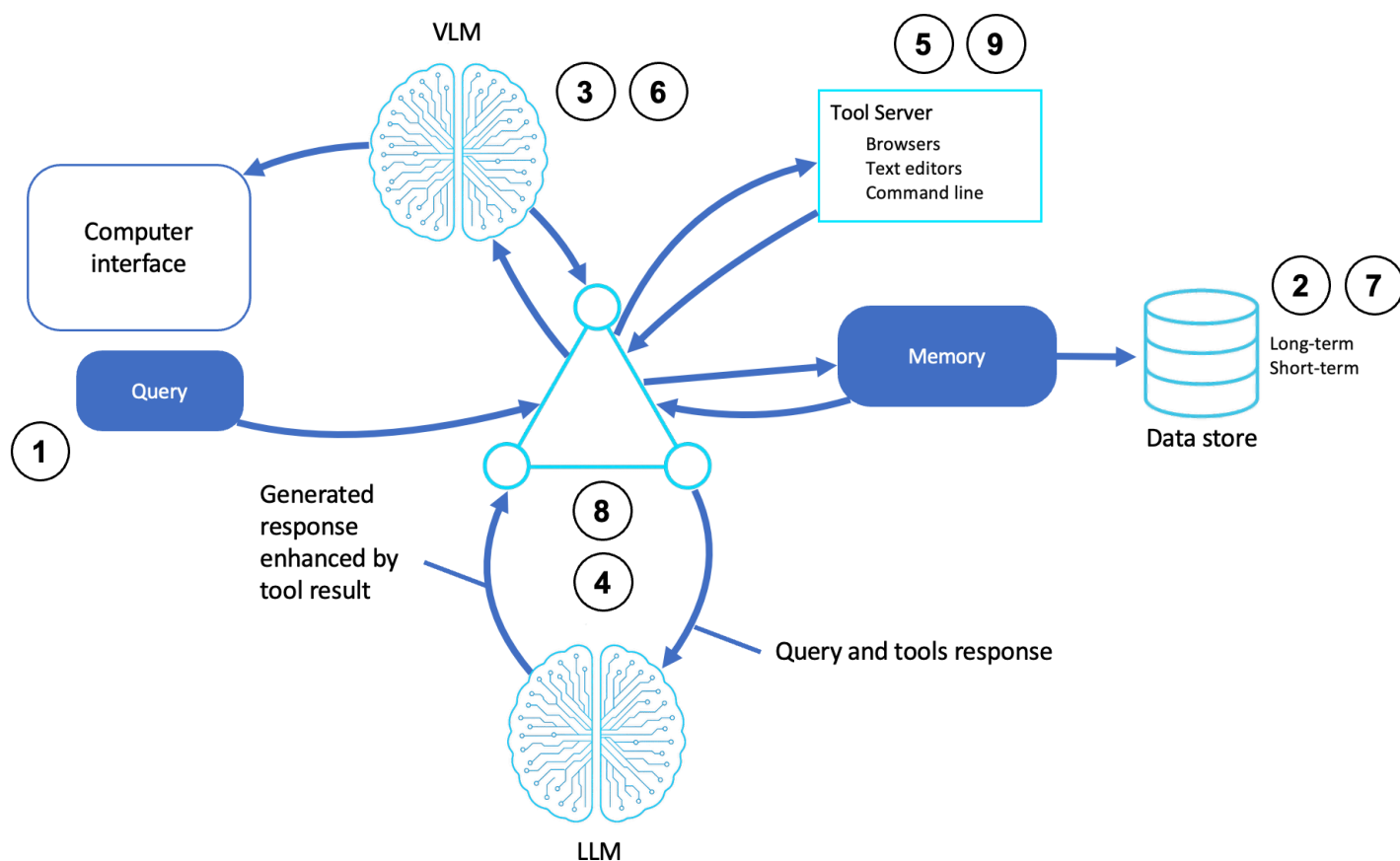
Computer-use 代理人

Computer-use 代理可以模拟或控制数字环境，例如浏览器、终端、文件系统和应用程序。这些代理通过组合 LLM 推理、视觉语言模型 (VLM) 和执行命令或模拟输入事件的工具服务器来解释用户意图，与视觉和文本界面交互，并执行以目标为导向的操作。

这种模式对于实际的人工智能自动化非常重要，在这种自动化中，代理不仅可以充当助手，还可以充当代理，像人类一样执行操作，通常使用相同的工具和环境。

架构

计算机使用的代理模式如下图所示：



说明

1. 收到查询

- 任务或请求通过 UI、API 或自然语言界面提供。

2. 访问内存

- 代理会检索短期和长期记忆，以回忆过去的命令、目标和系统状态。

3. 分析视觉背景

- VLM 观察计算机屏幕、系统状态或用户界面元素，以了解给定的上下文并识别可操作的项目。

4. 法学硕士学位的理由

- LLM 结合了查询、内存状态、工具和服务响应来确定下一个操作。

5. 与工具服务器交互

- 代理调用服务器上托管的工具，其中可能包括以下内容：
 - 浏览器（例如，无头 Chrome）和外壳环境
 - 文本和代码编辑器
 - 自定义脚本接口

6. 更新视觉输入

- 如果系统用户界面发生变化或需要进一步观察，VLM 可能会重新分析屏幕状态或文本缓冲区。

7. 更新内存

- 新的见解、系统状态或用户反馈会写入短期和长期记忆。

8. 制定最终决定和解释

- LLM 根据查询和工具输出综合结果或建议操作。

9. 返回响应

- 代理将结果返回到界面（例如，已完成的任务、确认或生成的内容）。

功能

- 带有视觉和文本输入的多模态推理
- 通过仿真或 API-driven 输入控制应用程序
- 永久状态的内存管理
- 序列执行中的自主权（多步流程）

常见使用案例

- 在 IDE 中编写和运行代码的 AI 开发人员
- Computer-use 用于重复数字化工作流程的代理
- 用于软件测试和质量保证的模拟用户
- 辅助功能代理，用于通过语音或高级指令浏览用户界面
- 通过推理增强的智能机器人流程自动化 (RPA)

实施指导

- 您可以使用以下方法构建此模式 AWS 服务：
- Amazon Bedrock 用于 LLM-based 计划和推理
- Amazon Elastic Compute Cloud (Amazon EC2) AWS Lambda，或 SageMaker 亚马逊笔记本电脑，用于在模拟 UI 环境中运行工具服务器
- 用于内存持久性的亚马逊简单存储服务 (Amazon S3) Service 或 Amazon DynamoDB
- Amazon Rekognition（或自定义模型）用于混合场景中的用户界面图像分析

- Amaz CloudWatch on Logs 或者 AWS X-Ray 用于可观察性和审计跟踪

Summary

Computer-use 代理充当自主的数字运营商，弥合了人机交互和 AI-driven 操作之间的差距。通过整合内存、工具编排和 VLM，这些代理可以自适应地与专为人设计的系统进行交互、执行操作、更新文件、浏览菜单和生成响应。

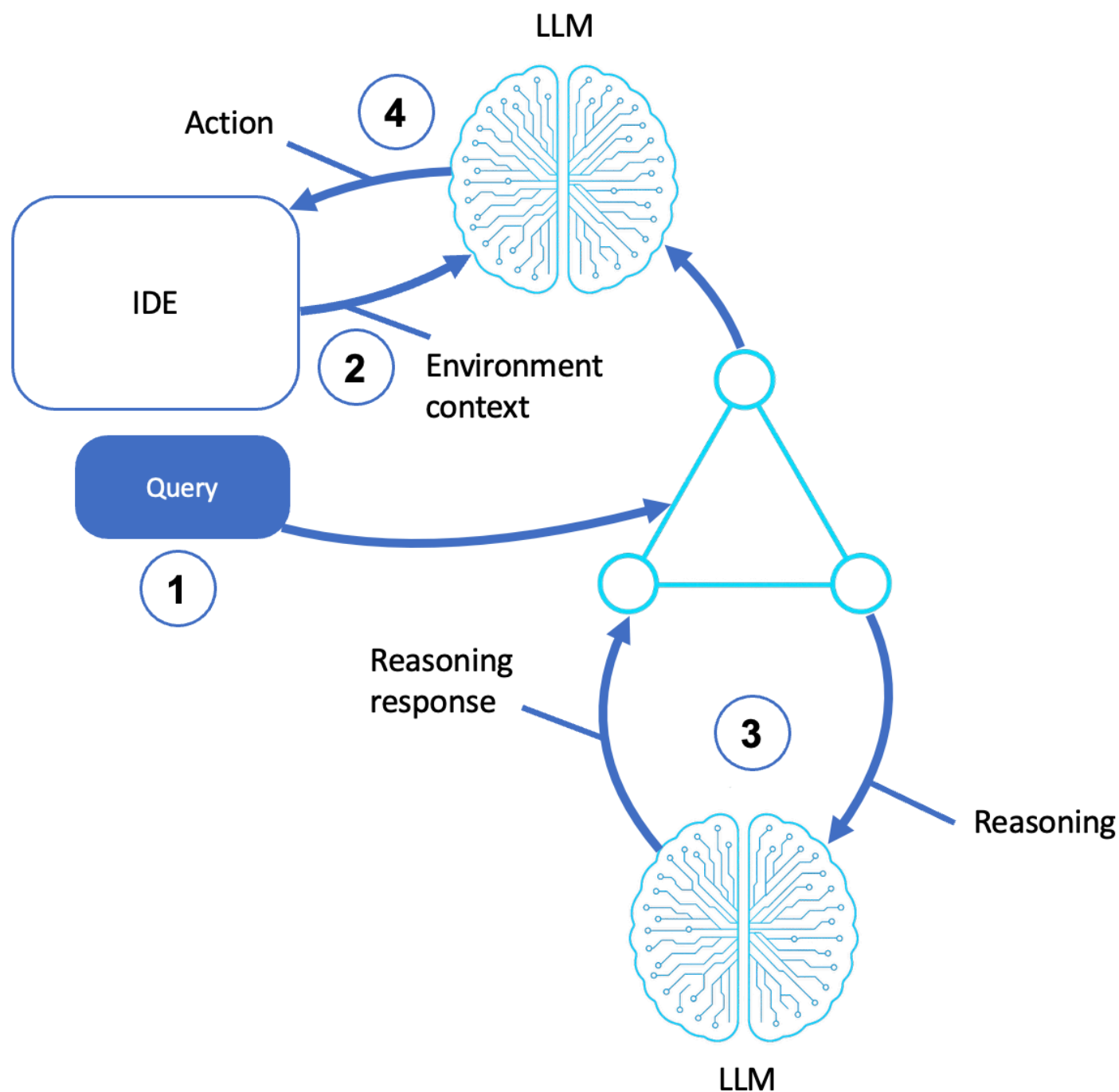
编码代理

编码代理可以推理编程任务，生成或修改代码，并与开发人员环境（例如 IDE 和 CLI）进行交互。这些代理将自然语言理解与结构化推理相结合，以协助、增强和自动化软件开发，从函数生成到错误修复和测试创作。

与自动完成工具不同，编码代理会主动解释用户目标，查询开发环境以获取上下文（例如，它打开文件并跟踪错误），识别需求，然后提出和执行操作。

架构

编码代理模式如下图所示：



说明

1. 接收查询

- 用户通过命令面板、聊天窗口或 CLI (例如,“向此功能添加日志记录”或“重构以提高可读性”) 提供自然语言指令。

2. 提取环境上下文

- 代理从 IDE 收集上下文，包括活动文件、光标位置、代码片段和符号表。
- 它输出错误消息、测试结果和其他代理的输出。

3. 法学硕士推理

- 代理向 LLM 发送提示，包括查询和环境上下文。
 - 法学硕士通过推理来确定以下内容：
 - 需要改变什么
 - 如何生成解决方案
 - 任何重构、重写或编码步骤

4. 执行动作

- LLM 将输出返回到代理并将其导入到 IDE 或运行时环境中。
- 这可能包括插入或修改代码、生成注释或文档，以及触发下游构建、测试和 linting 任务。

功能

- High-contextual 意识（例如，IDE 状态、光标和语法树）
- 目标和反馈的迭代推理
- 可选的代码规划和操作分离（例如，先推理然后行动）
- 适用于同步或异步开发者工作流程

常见使用案例

- 根据任务描述生成代码
- 代码重构和优化
- Test-case 生成和验证
- 错误解释和调试
- 文档助手
- 配对编程副驾驶

实施指导

- 您可以使用以下工具和以下工具来构建此模式 AWS 服务：

- Amazon Bedrock 用于 LLM-driven 生成和推理
- Amazon Q 开发者提供编码建议和完成信息
- AWS Lambda 或者用于运行和测试沙盒环境的亚马逊弹性容器服务 (Amazon ECS) Container Service
- AWS Cloud9、VS Code 扩展或用于托管和评估上下文的自定义 IDE 集成
- Amazon Simple Storage Service (Amazon S3) 用于存储中间提示、响应和修订历史记录

Summary

编码代理是一种新的 AI-powered 开发工具，能够解释自然语言、分析上下文、生成多步骤代码更改以及与软件开发生命周期集成。

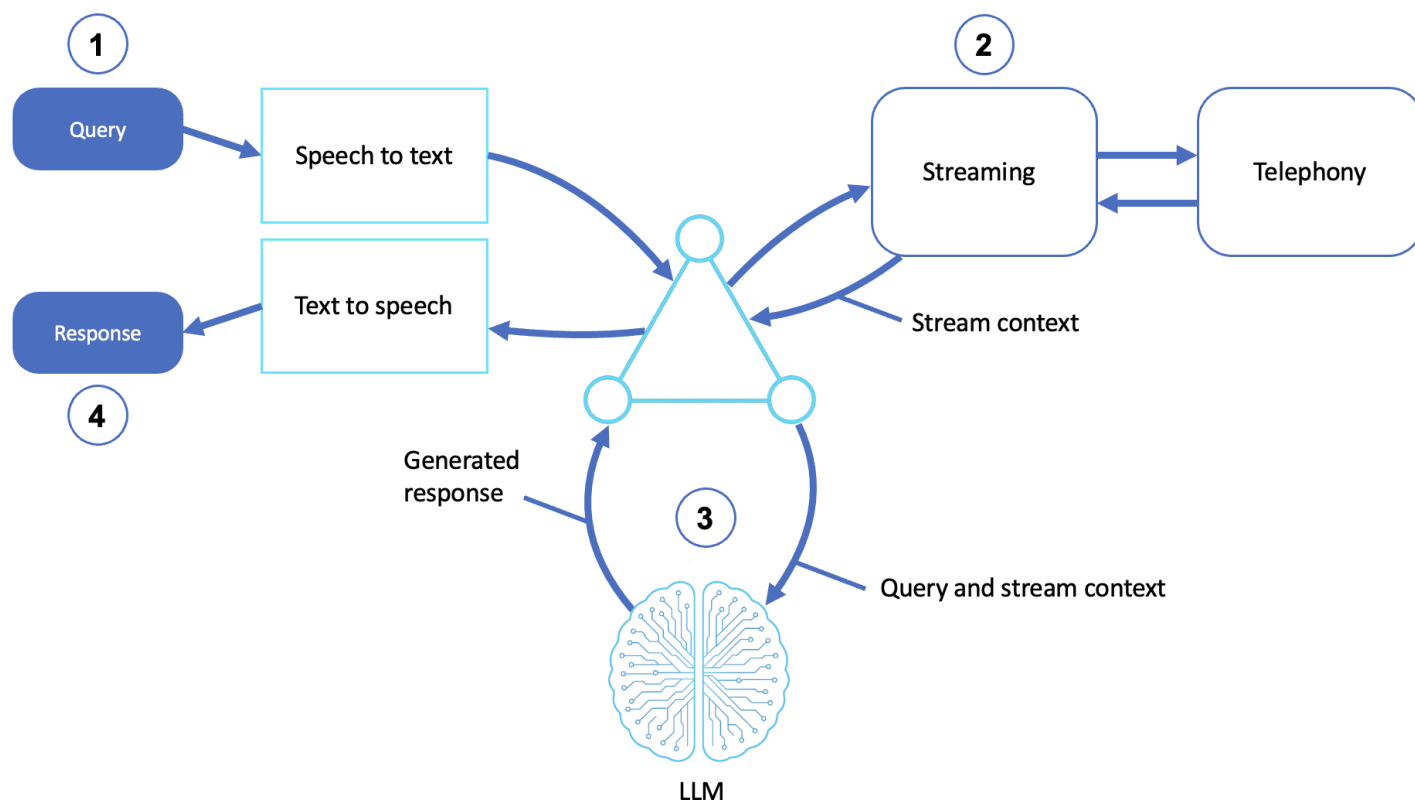
语音和语音代理

语音和语音代理通过语音对话与用户互动。这些代理集成了语音识别、自然语言理解和语音合成，可在电话、移动、网络 and 嵌入式平台上实现对话式 AI。

语音代理在免提、实时或可访问性驱动的环境中特别有效。通过将流媒体接口与 LLM-powered 推理相结合，它们促进了用户感觉自然的丰富、动态的互动。

架构

语音和语音代理如下图所示：



说明

1. 收到语音查询

- 用户向手机、麦克风或嵌入式系统发出请求。
- 语音转文本 (STT) 模块将音频转换为文本。

2. 集成了流媒体和电话环境

- 代理使用流媒体接口实时管理音频 I/O 。
- 如果部署在联络中心或电信环境中，则电话集成会处理会话路由、双音多频 (DTMF) 输入和媒体传输。

注意：DTMF 是指按下电话键盘上的按钮时产生的音调。在语音代理内部的流媒体和电话环境集成中，DTMF 用作电话呼叫期间的信号输入机制，尤其是在交互式语音应答 (IVR) 系统中。DTMF 输入使代理能够：

- 识别菜单选项（例如，“按 1 进行计费。按 2 获得支持。”）
- 收集数字输入（例如账号、PIN 和确认号码）
- 在呼叫流中触发工作流程或状态转换

- 必要时从语音恢复为按键音

1. 通过 LLM 直播上下文了解原因

- 查询被发送到代理，代理会将其与任何会话元数据（例如，呼叫者 ID、之前的上下文）一起传递给 LLM。
- LLM 会生成响应，如果交互正在进行中，则可能使用思维链策略或多回合记忆。

2. 返回语音响应

- 代理使用文字转语音 (TTS) 将其响应转换为语音。
- 它通过语音通道将音频返回给用户。

功能

- Real-time 语音理解和生成
- 支持 ST T/O T 和 TTS 的多语言版本
- 与电话或流媒体 API 集成
- 会话感知和回合之间的记忆切换

常见使用案例

- 对话式 IVR 系统
- 虚拟接待员和预约安排员
- Voice-driven 帮助台代理
- 可穿戴语音助手
- 智能家居的语音接口和无障碍工具

实施指导

您可以使用以下工具和以下工具来构建此模式 AWS 服务：

- 适用于 STT 的 Amazon Lex V2 或 Amazon Transcribe
- 适用于 TTS 的 Amazon Polly
- 用于直播和电话的 Amazon Chime SDK、Amazon Connect Customer 或亚马逊互动视频服务 (Amazon IVS) Interactive Video Service

- Amazon Bedrock 用于使用 Anthropic、AI21 或其他基础模型进行推理
- AWS Lambda 连接 STT、LLM、TTS 和会话上下文

(可选) 其他增强功能可能包括以下内容 :

- 亚马逊 Kendra 或者 OpenSearch 用于情境感知型 RAG
- 用于会话内存的 Amazon DynamoDB
- Amazon CloudWatch 日志和可 AWS X-Ray 追溯性

Summary

语音和语音代理是通过自然对话进行交互的智能系统。通过将语音接口与 LLM 推理和实时流媒体基础设施集成, 语音代理可实现无缝、可访问和可扩展的交互。

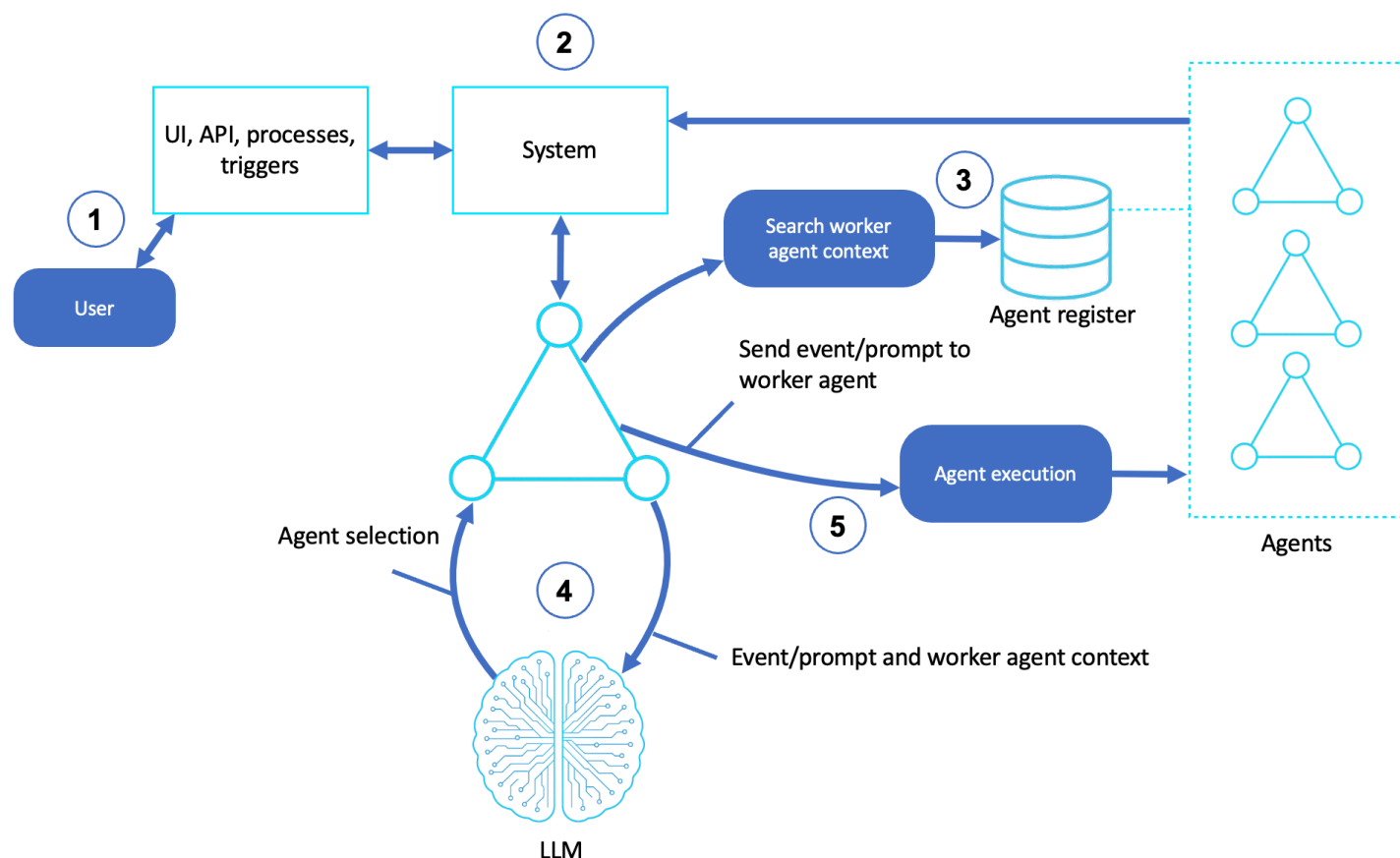
工作流程编排代理

工作流协调代理管理和协调分布式系统中的多步骤任务、流程和服务。这些代理不是孤立地推理和行动, 而是将工作委托给子代理或其他系统, 维护执行环境, 并根据中间结果进行调整。

这些代理是自动化流程的基本组成部分。它们在处理长时间运行的任务、多代理组合和跨域集成时特别有用, 在这种情况下, 必须按顺序或有条件地调用各种代理和工具。

架构

下图显示了工作流程编排代理 :



说明

1. 接收用户输入

- 用户（或外部触发器）通过 UI、API 或系统事件启动任务。

2. 处理系统事件

- 系统组件接收请求并发出需要编排的事件或命令。

3. 检索上下文

- 工作流代理会查询知识库和代理注册表，根据元数据、域名和先前的成功率为任务找到合适的工作代理。

4. 选择法学硕士代理

- 法学硕士通过分析任务描述和可用选项来帮助选择最佳代理或工作流程计划。
- 它还可以制定特定于任务的提示以发送给选定的代理。

5. 委托和执行人

- 选定的工作器代理收到事件或提示并开始运行命令。
- 它可以跟踪执行状态，在失败时重试，并将中间结果传递给序列中的下一个代理。

功能

- 代理构成 (例如, 主管、协作者代理和工具)
- Event-driven 或计划执行
- 随着时间的推移进行内存和状态跟踪
- 分层或并行任务编排 (同步与异步 workflows 相比)
- 动态代理选择和链接

常见使用案例

- 多步骤自动化 (例如, 数据摄取和报告)
- 客户服务路由和上报 (例如, 代理即协调员)
- AI 代理在同一个循环内与人类和机器人进行协调
- 使用 LLM-powered 逻辑实现企业流程自动化
- 混合系统结合了 AI 代理和传统编排工具

实施指导

您可以使用以下工具和以下工具来构建此模式 AWS 服务：

- Amazon Bedrock 用于推理和代理选择
- AWS Step Functions 或 Amazon EventBridge 用于 workflows 构成
- AWS Lambda 作为执行单元或任务运行器
- 亚马逊 DynamoDB、亚马逊简单存储服务 (Amazon S3) Simple Storage S3 或 Amazon RDS 用于跟踪状态和结果
- AWS AppFabric 或者 AppFlow 用于跨系统协调的 Amazon
- (可选) 使用 Amazon SageMaker 运行代理托管特定域的工作代理

Summary

工作流代理在多代理环境中协调、调整和调整目标。这意味着 AI 代理可以通过模块化、可解释的工作流程进行协作、适应运行条件并交付复杂的结果。

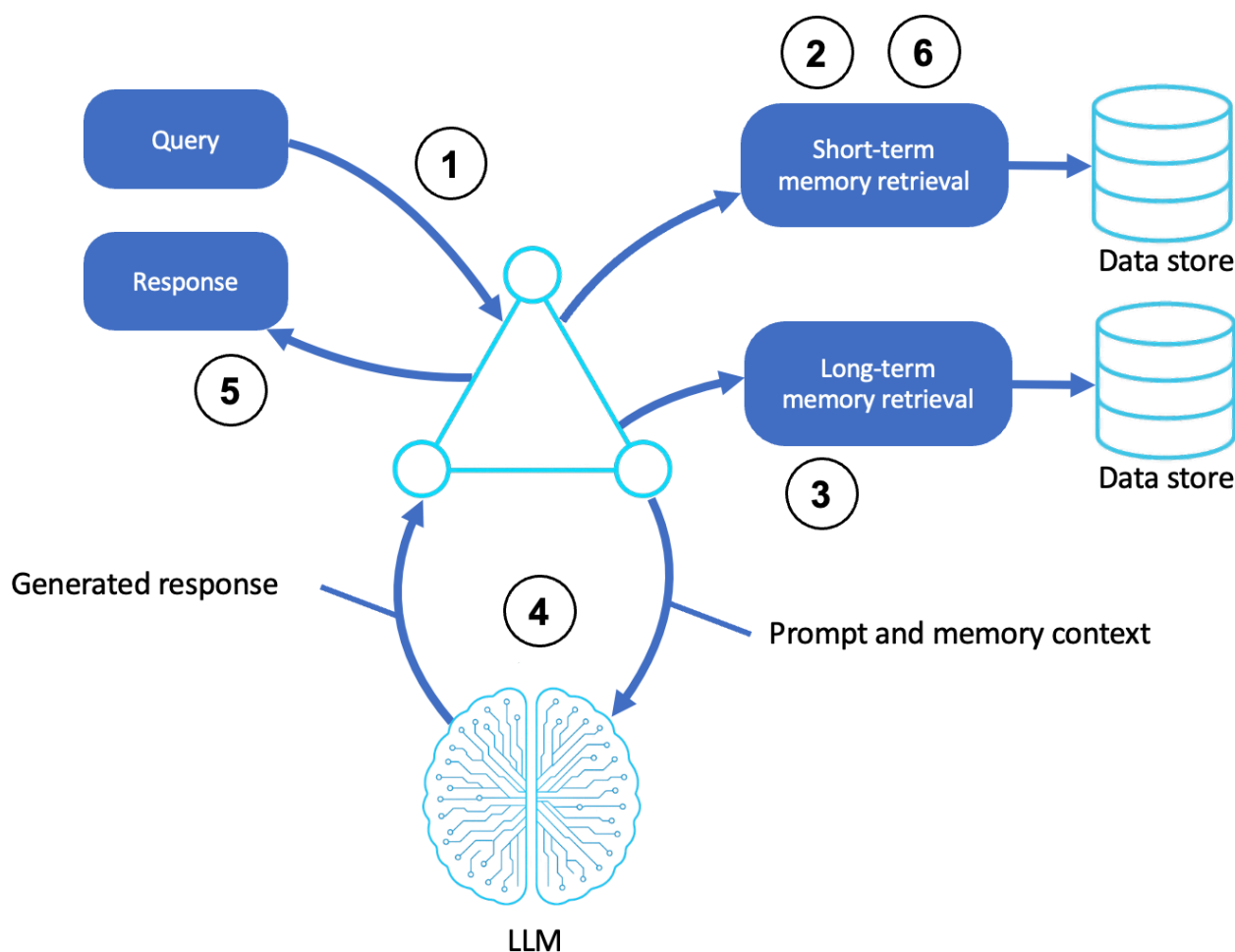
Memory-augmented 代理人

Memory-augmented 通过使用短期和长期记忆进行存储、检索和推理的能力，代理得到了增强。这使他们能够在多个任务、会话和互动中保持背景信息，从而产生更连贯、更个性化和更具战略性的响应。

与无状态代理不同，内存增强型代理通过引用历史数据进行调整，从先前的结果中学习，并做出符合用户目标、偏好和环境的决策。

架构

内存增强代理如下图所示：



说明

1. 接收输入或事件

- 代理接收用户查询或系统事件。这可能是文本、API 触发器或环境变化。

2. 检索短期记忆

- 代理检索与会话或工作流程相关的近期对话历史记录、任务上下文或系统状态。

3. 检索长期记忆

- 代理查询长期内存（例如，矢量数据库和键值存储）以获取历史见解，例如：
 - 用户偏好
 - 过去的决定和结果
 - 学到的概念、总结或经验

4. 通过法学硕士学位考试的原因

- 记忆上下文嵌入到 LLM 提示符中，允许代理根据当前输入和先验知识进行推理。

5. 生成输出

- 代理会根据任务历史记录和用户输入生成情境感知的响应、计划或操作，并对其进行个性化设置。

6. 更新内存

- 存储了新的信息，例如更新的目标、成功和失败的信号以及结构化的响应，以备将来的任务之用。

功能

- 跨对话或事件的会话连续性
- 目标随着时间的推移而保持不变
- 基于不断变化的状态的情境感知
- 以往的成功和失败为依据的适应性
- 根据用户偏好和历史记录进行个性化设置

常见使用案例

- 记住用户偏好的对话副驾驶
- 跟踪代码库变化的编码代理
- 根据任务历史进行调整的工作流代理
- 从系统知识演变而来的数字双胞胎
- 避免重复检索的研究代理

实现内存增强代理

AWS 服务 对于内存增强型代理，请使用以下工具：

内存层	AWS 服务	目的
Short-term	亚马逊 DynamoDB、Redis、 亚马逊 Bedrock 上下文	快速检索最近的交互状态
Long-term（结构化）	亚马逊 Aurora、亚马逊 DynamoDB、亚马逊 Neptune	事实、关系和日志
Long-term（语义）	OpenSearch、Postgre SQL、Pinecone	Embedding-based 检索（即 RAG）
仓储服务	Amazon S3	存储笔录、结构化记忆和文件
编排	AWS Lambda 或者 AWS Step Functions	管理内存注入和更新生命周期
推理	Amazon Bedrock	带有记忆提示的 Anthropic Claude 或 Mi

实现注入内存的提示

要将记忆整合到代理推理中，请结合使用结构化状态和检索增强上下文注入：

- 在为语言模型构建提示时，将最新的代理状态和最近的对话历史记录作为结构化输入，这样它就可以在完整的上下文中进行推理。
- 使用检索增强生成 (RAG) 从长期记忆中提取相关文档或事实。
- 总结之前的计划、背景和互动，以实现压缩和相关性。
- 在推理过程中注入外部存储器模块，例如向量存储或结构化日志，以指导决策。

Summary

Memory-augmented 代理通过从经验中学习和记住用户上下文来保持思维的连续性。这些代理通过使用长期协作、个性化和战略推理，超越了被动情报。就代理人工智能而言，内存使代理的行为更像自适应数字对应物，而不像无状态工具。

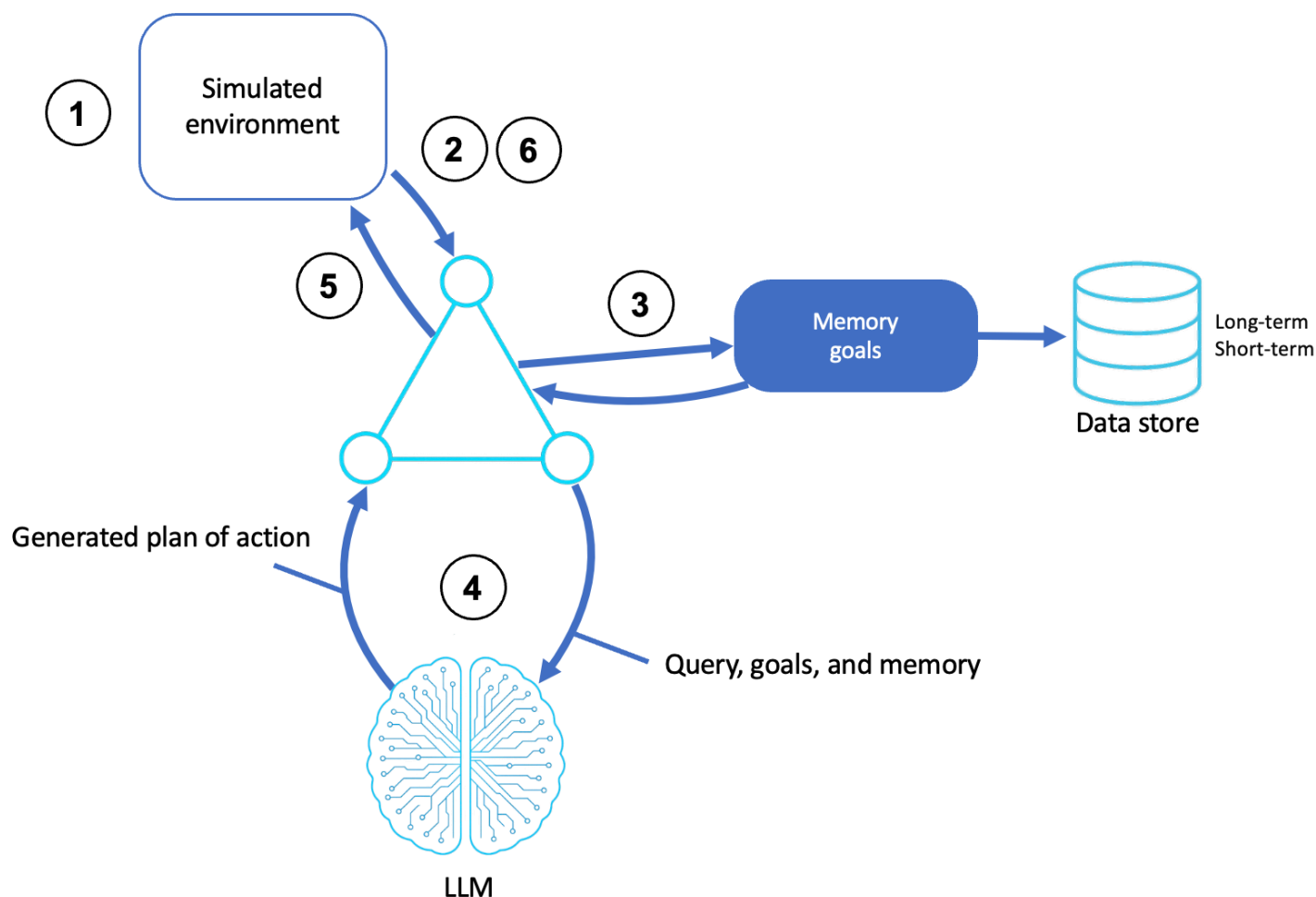
仿真和测试平台代理

仿真和测试平台代理在虚拟化或受控环境中运行，在那里他们可以推理、行动和学习。这些代理在将策略应用于现实环境之前，先在可重复的环境中模拟行为、建模结果并训练策略。

这种模式对于迭代开发、强化学习 (RL)、自主决策评估和紧急行为测试非常有用。仿真代理通常在闭环中运行，接收来自其环境的反馈并相应地调整其行为，这使得它们对于涉及空间推理、实时控制或复杂系统动力学的任务至关重要。

架构

下图显示了仿真或测试平台代理：



说明

1. 启动环境

- 代理启动模拟环境（例如，3D 世界、物理引擎、CLI 沙盒或合成数据流）。
- 将代理加载到环境中，并附有初始任务、目标或策略。

2. 感知特工

- 代理通过仿真遥测（例如，传感器仿真、虚拟摄像机和结构化日志）感知当前状态。

3. 检索目标和记忆

- 代理检索其分配的目标、场景说明或上下文目标。
- 它还可以检索以前的内存，包括以下内容：
 - Long-term 战略或政策
 - 环境地图或已知限制
 - 类似模拟中过去的成功或失败

4. 原因和计划

- 法学硕士解释模拟状态、任务目标和所学知识。
- 它生成行动计划或控制命令。

5. 执行模拟动作

- 代理执行计划、修改状态、导航空间或与虚拟实体交互。

6. 学习

- 代理评估行动结果
- 根据代理的配置，它可能会执行以下操作：
 - 执行 RL
 - 记录结果以备将来微调
 - 实时调整策略

功能

- 可在合成环境或虚拟环境中运行
- 支持试错学习、策略完善和系统建模
- Low-risk 测试行为、故障处理和边缘情况
- 支持在多智能体设置中进行紧急代理行为分析
- 支持闭环控制和人机在环探索

常见使用案例

- 机器人、无人机和游戏的强化学习
- 在虚拟道路上进行自动驾驶车辆训练
- 针对 DevOps 和测试平台场景的模拟 UI 或 CLI
- 社交模拟中的紧急行为实验
- 在生产前对决策逻辑进行安全验证

实施指导

您可以使用以下工具和以下工具构建仿真和测试平台代理：AWS 服务

组件	AWS 服务	目的
环境	亚马逊 ECS、亚马逊 EC2 或亚马逊 SageMaker 工作室实验室中的自定义模拟器	运行虚拟世界 (凉亭、Unity、虚幻) 或沙盒 CLI
代理逻辑	亚马逊 Bedrock SageMaker 、 Amazon 或 AWS Lambda	LLM-based 策划者或 RL 代理商
反馈回路	亚马逊 SageMaker 强化学习 CloudWatch、Amazon 或自定义日志	奖励跟踪、结果评分和行为记录
记忆和重播	亚马逊 S3、亚马逊 DynamoDB 或亚马逊 RDS	永久状态、剧集历史记录或场景数据
可视化	亚马逊 CloudWatch 控制面板或亚马逊 SageMaker 笔记本	观察政策变化、结果和培训指标

以下是其他应用程序：

- [AWS SimSpace Weaver](#)用于大规模空间模拟
- [AWS IoT Core](#)用于测试影子设备
- 用于代理评估和基准测试的 [Amazon SageMaker 实验](#)

Summary

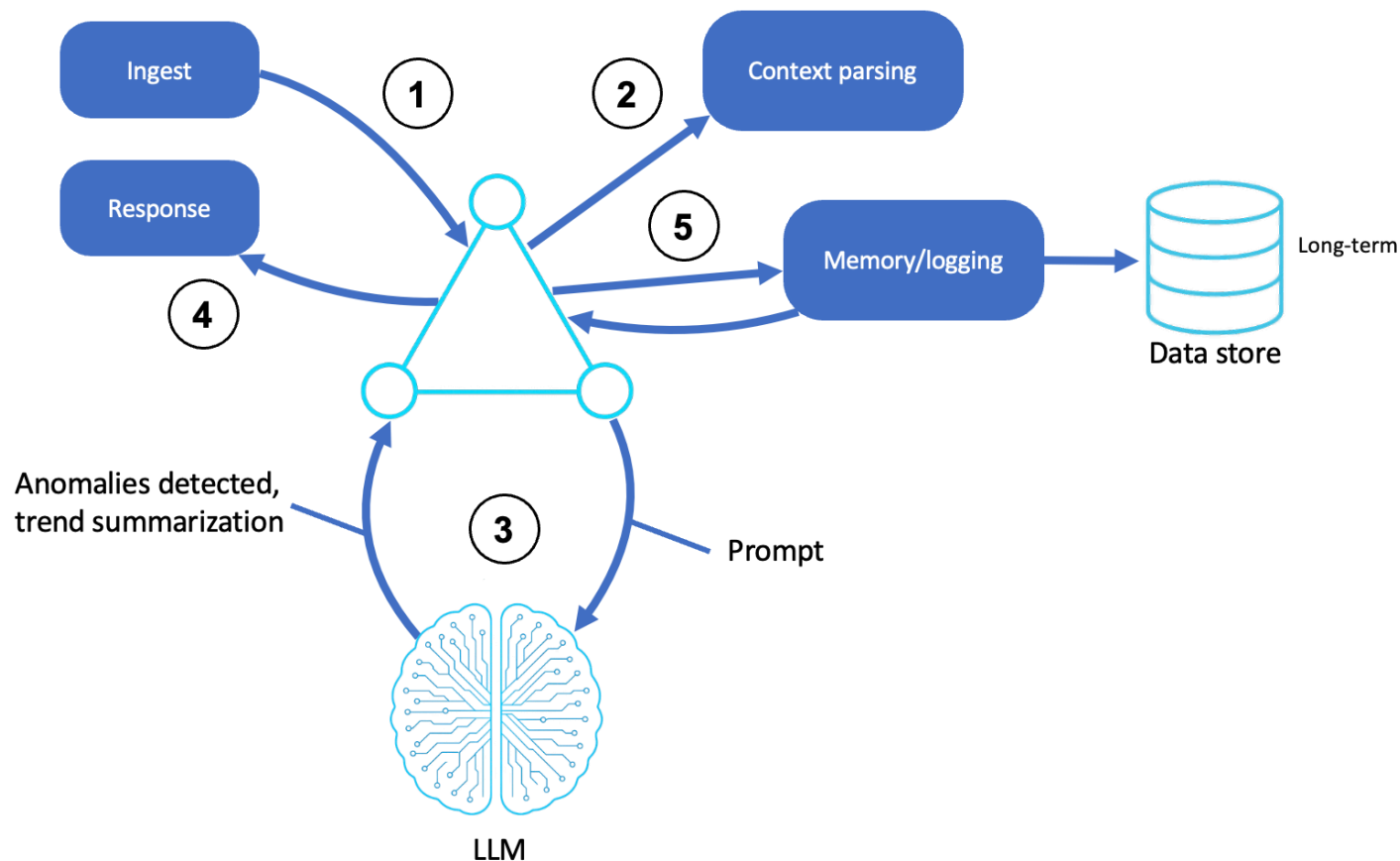
仿真和测试平台代理用于在部署到生产系统之前进行结构化探索。使用这些代理来训练自主导航策略，在合成环境中测试业务流程，并评估群体的协调模式。

观察者和监视代理

观察者和监视代理被动观察系统、环境和交互，以检测模式、生成见解并触发操作。作为智能观察者，他们无需直接启动行为即可增强警报、诊断和审计。

这些代理在传统监控缺乏适应性或推理性方面表现出色，尤其是在 AI-in-the-loop 监控、异常检测、合规监督和安全情报方面。观察者代理是持续监控系统遥测和用户交互的事件侦听器。代理取决于感知、解释以及有条件的上报或报告。

架构



说明

1. 摄取遥测数据

- 代理接收来自一个或多个系统源的输入，例如：
 - 日志（应用程序、基础架构、安全）
 - 指标（性能、延迟、使用情况）
 - 事件（API 调用、用户操作、传感器数据）

2. 解析上下文

- 原始输入经过解析、结构化并丰富元数据，例如时间戳、参与者身份、系统状态和跟踪 ID。

3. 使用法学硕士学位的原因

- 代理使用 LLM 或逻辑模块通过识别异常、汇总趋势以及跨分布式跟踪或时间窗口关联来解释已解析的输入。
4. 分类或提醒
- 代理确定所观察到的行为是否符合以下条件：
 - 警报或升级
 - 报告或控制面板更新
 - 响应触发器（例如，自动补救和策略执行）
5. 记录内存或反馈回路
- 代理存储事件和决策，以供其他代理长期学习、审计或将来参考。

功能

- 被动和非侵入性（代理不直接起作用）
- 高度可扩展和异步
- AI-driven 噪声信号或分布式信号之间的相关性
- 支持审计、合规和实时洞察
- 可以为下游代理或人工工作流程提供信息

常见使用案例

- AI-augmented 微服务和 API 的可观察性
- 监控模型偏差、策略违规行为或带外行为
- 客户活动分析或互动摘要
- 监视提交或部署的代码审查代理
- 使用 LLM 推理进行安全或合规日志监控

实施指导

您可以使用以下工具和以下工具来构建观察者和监视代理 AWS 服务：

组件	AWS 服务	目的
功能		

事件摄取	亚马逊 EventBridge、亚马逊 CloudWatch 日志、亚马逊 Kinesis、亚马逊 S3	摄取结构化和非结构化遥测
预处理	AWS Lambda, AWS Glue, AWS Step Functions	将原始数据转换为结构化提示
推理引擎	亚马逊 Bedrock、亚马逊 SageMaker、AWS Lambda	分析事件、对行为进行分类、生成见解
存储和内存	亚马逊 S3、亚马逊 DynamoDB、OpenSearch	持续的观察、总结和输出
警报和升级	亚马逊 SNS、AWS AppFabric亚马逊 EventBridge	触发下游系统或代理

以下是其他应用程序：

- [AWS Security Hub CSPM](#)用于安全日志监控
- [Amazon Quick](#) 用于可视化代理输出

Summary

观察者和监视代理可以实时跟踪系统和行为。他们通过识别人类或规则可能忽视的模式来检测异常、审计安全并收集运营情报。此功能有助于创建能够适应不断变化的条件并基于全面的数据分析做出决策的系统。

Multi-agent 合作

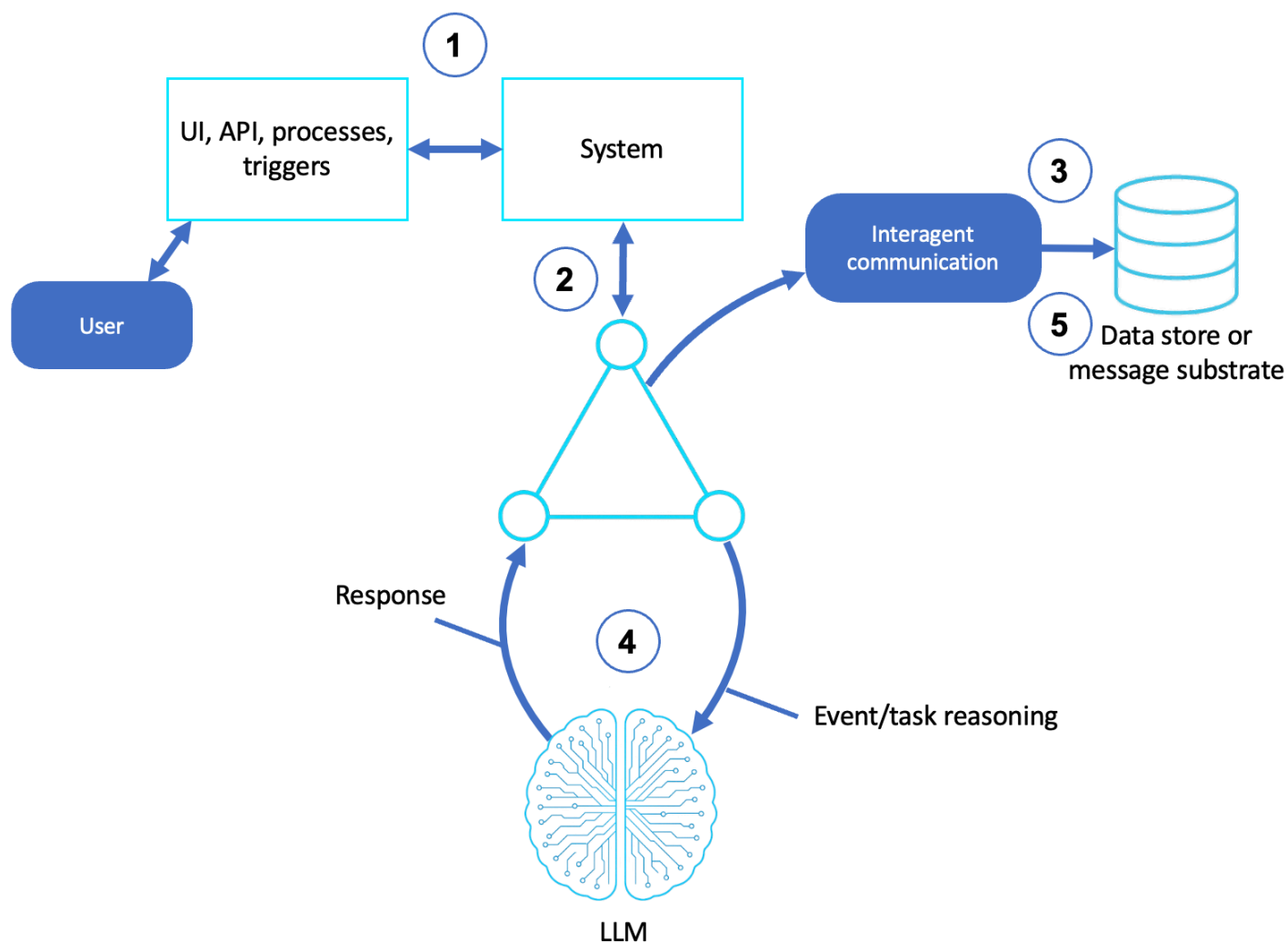
Multi-agent 协作是指一种模式，在这种模式中，多个自主代理通过协商解决复杂的任务，每个代理都有不同的角色、专业或目标。这些代理人可以独立运作，也可以通过共享信息、划分责任和集体推理来实现目标，从而与其他代理人一起行动。

这种模式不同于工作流代理，后者在结构化流程中集中协调任务并将其委派给下属代理。相比之下，多智能体协作通过实现适应性、并行性和认知分裂来强调点对点或紧急协调。下表比较了多代理与工作流代理的协作：

功能	工作流程代理	目的
控件	集中式协调员	去中心化、分布式或基于角色的 Peer 节点
相互作用	一个代理委托并跟踪执行情况	多个代理协商、共享和调整
设计	预定义的任务顺序	快速、灵活的任务分配
协调	程序编排	合作或竞争互动
使用案例	企业流程自动化	复杂的推理、探索和应急策略

架构

下图显示了多代理协作：



说明

1. 启动任务

- 用户或系统发出高级目标或问题。
- “经理”代理或启动上下文定义了目标。

2. 分配或发现角色

- 代理人自行分配（符号逻辑或推理）或被委托（事件经纪人）到其他角色，例如策划者、研究员、执行者、批评者或解释者。

3. 与其他代理沟通

- 代理通过共享内存、消息队列或提示链进行通信。
- 他们可能会互相辩论、询问或提出子任务。

4. 使用专门的推理

- 每个代理都使用自己的模型或域逻辑来解决自己的问题部分。
- 代理可以将 LLM 与特定角色的提示和内存一起使用。

5. 协调产出或目标

- 代理人将贡献综合成最终答案、计划或行动。
- （可选）监督代理可以验证或汇总合成后的输出。

功能

- Peer-level 具有特殊角色或技能的代理
- 通过沟通或谈判出现的紧急行为
- 并行处理复杂或多方面的问题
- 支持深思熟虑、自我纠正和反思性迭代
- 为社交动态、科学合作或企业团队角色建模

常见使用案例

- 自主研究团队（搜索代理、摘要器和验证者）
- 软件开发（规划员、编码员和测试员）
- 业务场景建模（财务、策略和合规）
- 谈判、竞标或多方推理
- 多模式任务（图像、文本和逻辑）

实施指导

您可以使用以下工具和 AWS 服务以下工具构建多代理系统：

组件	AWS 服务	目的
代理托管	亚马逊 Bedrock、亚马逊 SageMaker、AWS Lambda	托管个人 LLM-driven 代理
通信层	亚马逊 SQS、亚马逊、EventBridge AWS AppFabric	代理之间的消息传递和协调

共享内存	亚马逊 DynamoDB、亚马逊 S3 或 OpenSearch	Multi-agent 存储器或黑板系统
编排层	AWS Step Functions , AWS Lambda 管道	启动、超时、回退和重试逻辑
代理识别	亚马逊 Bedrock 代理 (角色定义) 和 AWS AppConfig 亚马逊 Bedrock converse API (亚马逊 Bedrock 以外的代理)	Role-based 工具或代理调用和边界强制执行
紧急互动	Amazon EventBridge 渠道或代理注册表	启用动态任务路由或升级

Summary

Multi-agent 协作将问题解决任务分配给模块化、角色驱动的代理。与 workflow 协调不同，协作模式使用的是反映人类解决问题的新兴智能、弹性和可扩展性。对于开放式领域、创造性任务、多模态推理和受益于不同视角的环境来说，它尤其有价值。

结论

前面讨论的模式说明了在现实世界中实现代理人工智能的基本方法。从基本推理到记忆增强智能，每种模式都针对基于自主性、异步性和代理性的感知、认知和行动进行了独特的配置。

这些模式共享用于构建智能、以目标为导向的系统的词汇和技术蓝图。无论模式是嵌入在用户界面中，还是通过云服务进行编排，还是跨代理团队进行协调，每种模式都具有适应性和模块化性。

外卖

- 代理模式是可组合的 — 大多数现实世界中的代理会混合两种或更多模式 (例如，具有基于工具的推理和记忆的语音代理) 。
- 代理设计是基于情境的 — 根据交互表面、任务复杂性、延迟容限和特定领域的限制来选择模式。
- AWS 原生实现是可以实现的 — 借助 Amazon Bedrock SageMaker AWS Lambda、AWS Step Functions、和事件驱动架构，每种代理模式都可以大规模交付。

法学硕士工作流程

在代理模式中，我们探索了常见的 AI 代理模式，每种模式都围绕一组模块化功能构建：感知、行动、学习和认知。在许多代理模式中，认知模块的核心是能够进行推理、计划和决策的大型语言模型 (LLM)。但是，仅仅调用法学硕士学位不足以产生明智的、以目标为导向的行为。

为了可靠地执行复杂的任务，代理必须将 LLM 嵌入结构化工作流程中，通过工具、内存、计划循环和协调逻辑增强模型的功能。这些 LLM 工作流程允许代理分解目标、路由子任务、呼叫外部服务、反思结果以及与其他代理进行协调。

本章介绍构建强大、可扩展和智能 LLM-driven 认知模块的核心设计模式，这些模块围绕可重复使用的工作流程进行组织。

本节内容

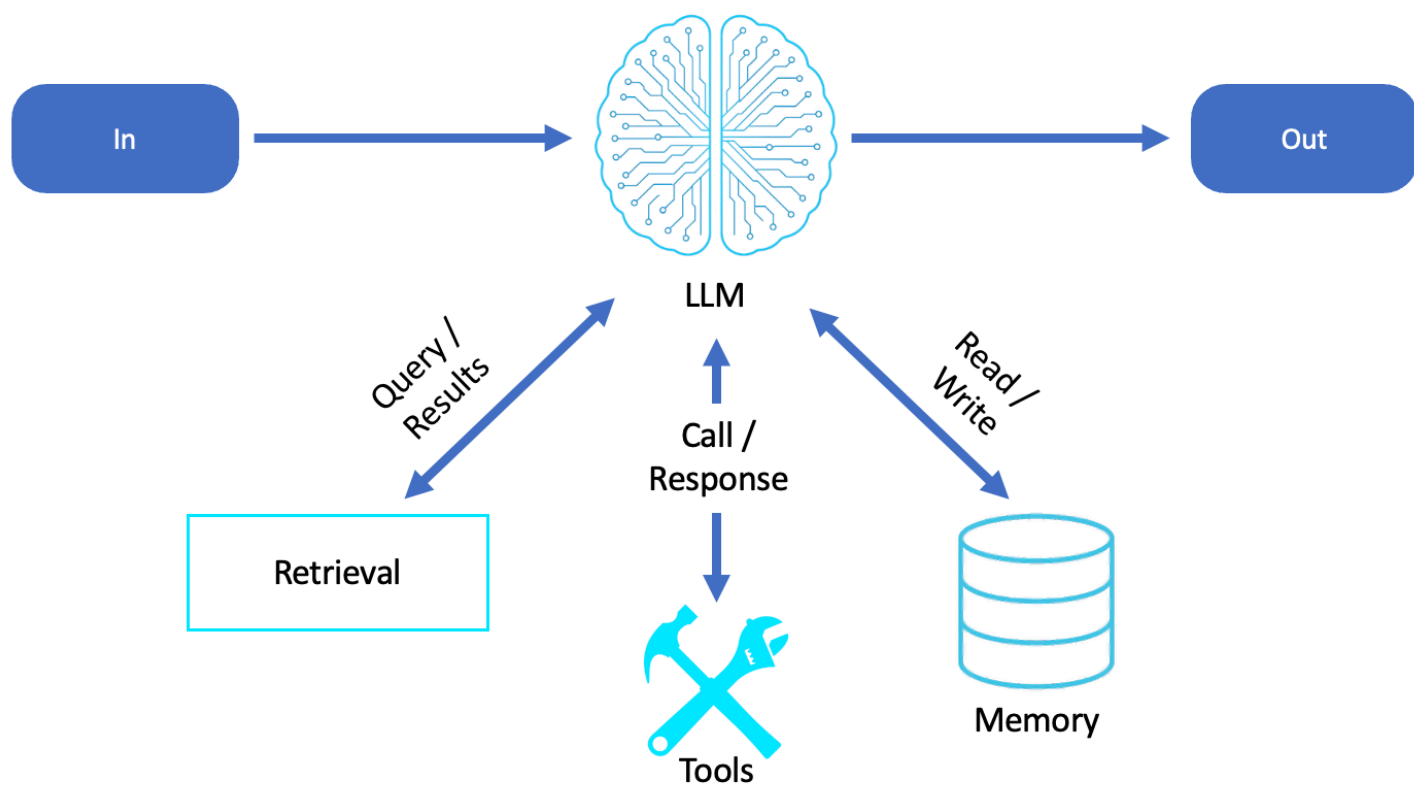
- [LLM-augmented 认知概述](#)
- [提示链接的工作流程](#)
- [路由工作流程](#)
- [并行化工作流程](#)
- [编排工作流程](#)
- [赋值器的工作流程和反射优化循环](#)
- [结论](#)

LLM-augmented 认知概述

从本质上讲，软件代理的认知模块可以看作是用增强包裹的法学硕士。代理可以使用以下构造块在其环境中进行有效的推理：

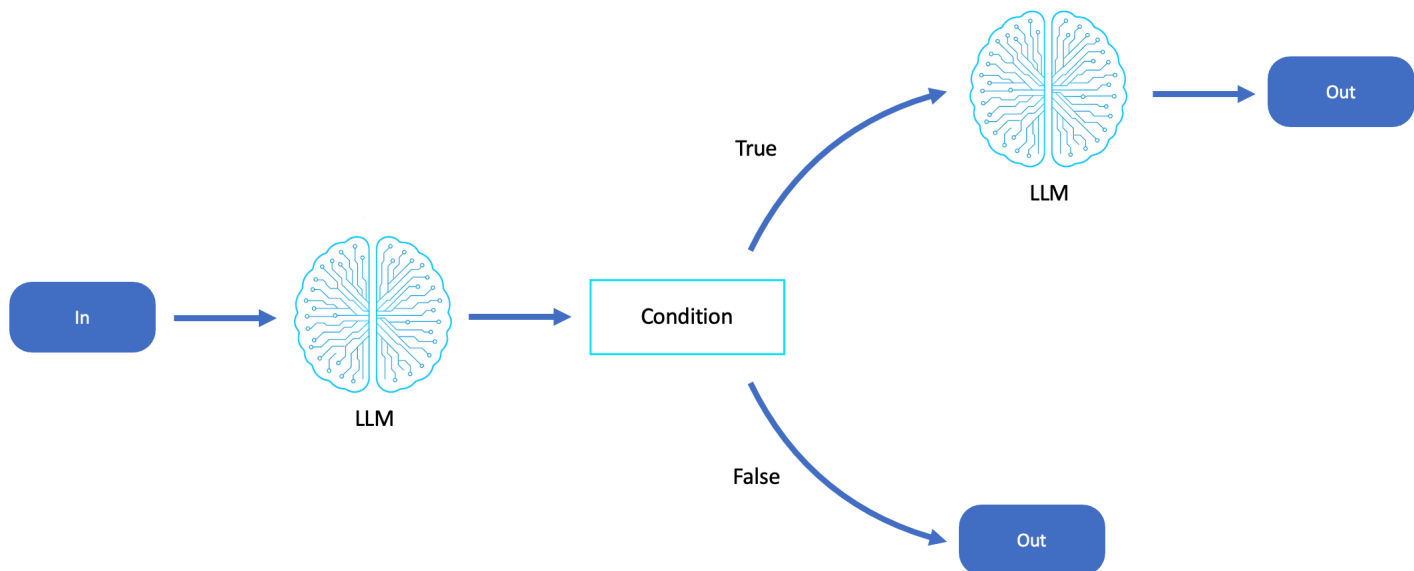
- 提示 — 使用上下文、指令、示例和内存来构图输入
- 检索 — 通过向量搜索或语义记忆（例如，通过检索增强生成 (RAG)）向 LLM 提示符提供最新或特定领域的知识
- 工具使用 — 允许 LLM 调用 API 或调用函数来检索信息或对信息采取行动
- 记忆 — 通过使用结构化数据库或情境摘要，将持续状态或基于会话的状态整合到推理循环中

这些增强功能由工作流程组成，这些工作流定义了如何随着时间的推移和跨任务使用 LLM，将其从无状态引擎转变为动态推理代理。



提示链接的工作流程

Prompt chaining 将复杂的任务分解为一系列步骤，其中每个步骤都是一个离散的 LLM 调用，用于处理或构建前一个任务的输出。



提示链接 workflow 适用于可以在逻辑上将任务划分为顺序推理步骤以及中间输出为下一阶段提供信息的场景。它在需要结构化思维、渐进式转换或分层分析的工作流程中表现出色，例如文档审查、代码生成、知识提取和内容完善。

说明

- 任务的复杂性超过了单个 LLM 调用的上下文窗口或推理深度。
- 一个步骤（例如分析、总结或规划）的输出将成为后续决策或生成阶段的输入。
- 您需要跨推理阶段的透明度和控制力（例如，可审计的中间结果）。
- 你想在步骤之间插入外部验证、筛选或扩充逻辑。
- 它非常适合在管道式推理循环中操作的代理，例如研究代理、编辑助理、计划系统和多阶段副驾驶。

功能

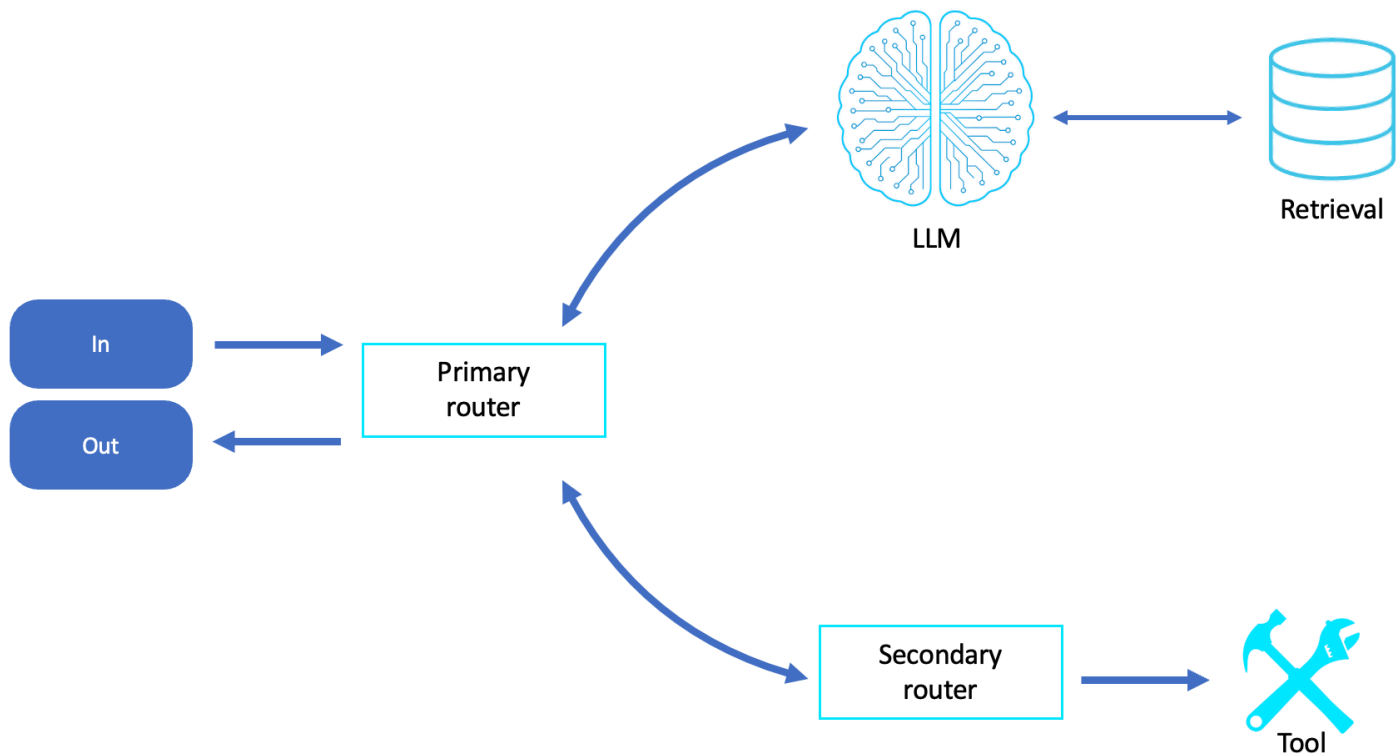
- LLM 调用的线性链或分支链
- 中间结果作为结构化输入传递或嵌入到后续提示中
- 可以与、或代理特定的运行 AWS Step Functions 器 AWS Lambda 一起编排

常见使用案例

- 多步推理任务（例如，“总结批评重写”）
- 研究助理合成分层输出（例如，“搜索摘录事实回答问题”）
- 代码生成管道（“生成计划编写代码测试代码解释输出”）

路由 workflow

在路由模式中，分类器或路由器代理使用 LLM 来解释查询的意图或类别，然后将输入路由到专门的下游任务或代理。



路由 workflow 用于代理必须快速对输入意图、任务类型或域进行分类，然后将请求委托给专门的子代理、工具或 workflow 的场景。它在能力代理中特别有用，例如充当一般助理、企业功能的前门或跨域的面向用户的 AI 接口。

在以下情况下，路由特别有效：

- 对各种任务（例如搜索、汇总、预订、计算）的请求进行分类。
- 在进入更专业的工作流程之前，必须对输入进行预处理或标准化。
- 不同的输入类型（例如，图像与文本、结构化查询与非结构化查询）需要自定义处理。
- 代理充当对话总机，将任务委托给专业代理或微服务。
- 这种 workflow 在特定领域的副驾驶、客户支持机器人、企业服务路由器和多模式代理中很常见，在这些代理中，智能调度决定了代理行为的质量和效率。

功能

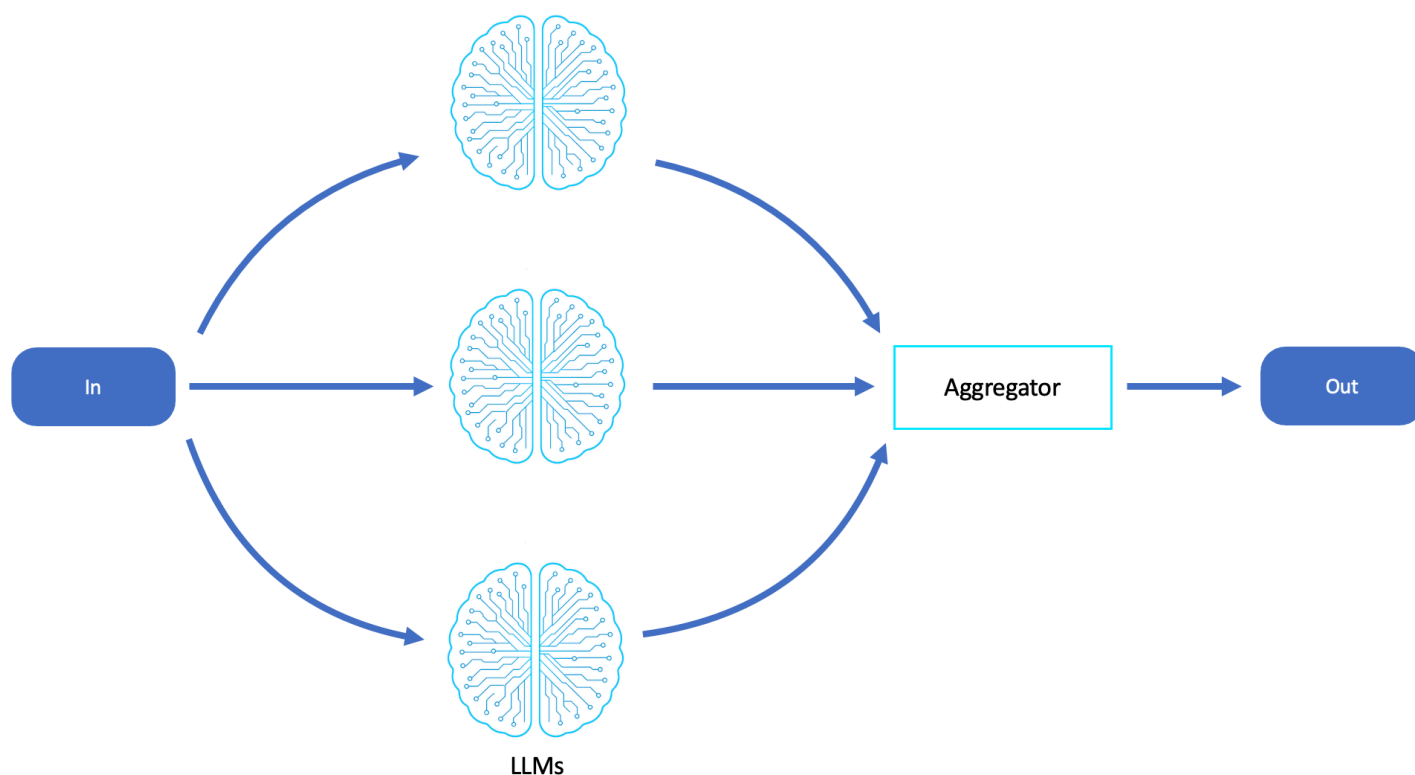
- 初级法学硕士充当调度员
- 路由可以调用不同的 workflow，甚至可以调用其他代理模式
- 支持功能的模块化扩展

常见使用案例

- 多域助手 (“这是法律、医疗还是财务问题？”)
- 通过法学硕士推理增强了决策树
- 动态工具选择 (例如，搜索与代码生成)

并行化工作流程

此工作流程包括将任务分解为独立的子任务，这些子任务可以由多个 LLM 呼叫或代理同时处理。然后以编程方式汇总输出并合成结果。



Parallelization 工作流程用于将任务划分为可以同时处理的独立、非顺序的子任务，从而显著提高效率、吞吐量和可扩展性。它在数据密集、批处理导向或多视角问题空间中特别强大，在这些空间中，代理必须通过多个输入分析或生成内容。

在以下情况下，并行化特别有效：

- 子任务不依赖彼此的中间结果，允许它们在没有协调的情况下并行运行。
- 一项任务涉及对许多项目重复相同的推理过程 (例如，总结多个文档或评估选项列表)。

- 并行探索多种假设或观点，以促进多样性、创造力或稳健性。
- 您需要通过并发 LLM 执行来减少大容量或高频请求的延迟。
- 此工作流程通常用于文档处理代理、调查或比较引擎、批量汇总器、多代理头脑风暴以及可扩展的分类或标签任务，尤其是在快速、并行推理具有性能优势的情况下。

功能

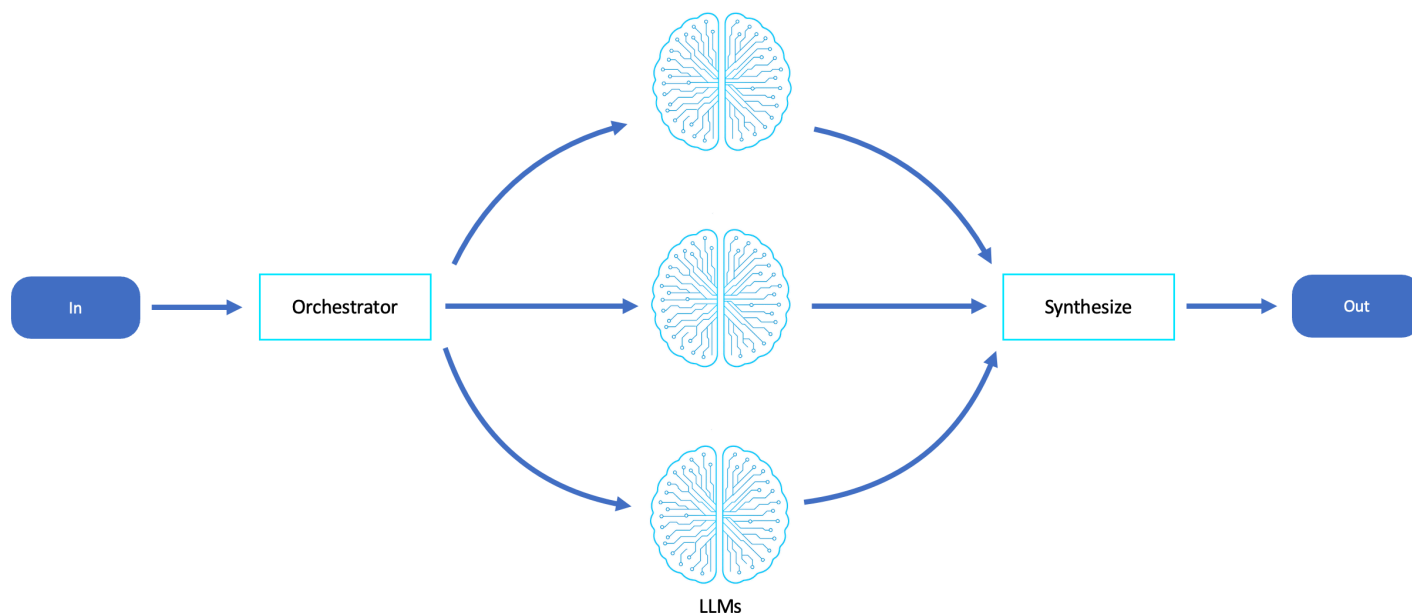
- 并行执行 LLM 任务 (通过使用 AWS Lambda AWS Fargate、或 AWS Step Functions 映射状态)
- 需要在综合阶段对结果进行校准、验证或重复数据删除
- Well-suited 用于无状态代理循环

常见使用案例

- 并行分析多个文档或视角
- 生成不同的草稿、摘要或计划
- 加快批处理作业的吞吐量

编排工作流程

中央协调器代理使用 LLM 来规划、分解子任务，并将子任务委托给专门的工作代理或模型，每个代理或模型都有特定的角色或领域专业知识。这反映了人类团队的结构，并支持多个代理的紧急行为。



编排工作流程非常适合复杂、分层或多学科的场景，需要结构化分解和专业执行。它特别适合需要分工的任务，在这些任务中，任务的不同子组件最好由具有不同能力、知识或工具集的代理来处理。

在以下情况下，此工作流程特别有效：

- 任务可以分为范围、类型或推理各不相同的子任务（例如，计划、研究、实施和测试）。
- 法学硕士或元代理必须协调其他代理，监控进度并综合结果。
- 您想对代理职责进行模块化，从而实现可扩展性、重复使用和专业调整。
- 该系统要求基于角色的行为，模仿人类团队（例如项目经理、开发人员和审阅者）的协作运作方式。

Orchestration 非常适合多回合计划代理、软件开发副驾驶、企业流程代理和自主项目执行者。在实现需要集中式任务分解但需要分布式执行逻辑的多代理系统时，它特别有用，可以跨代理层实现可扩展性和更易于解释的行为。

功能

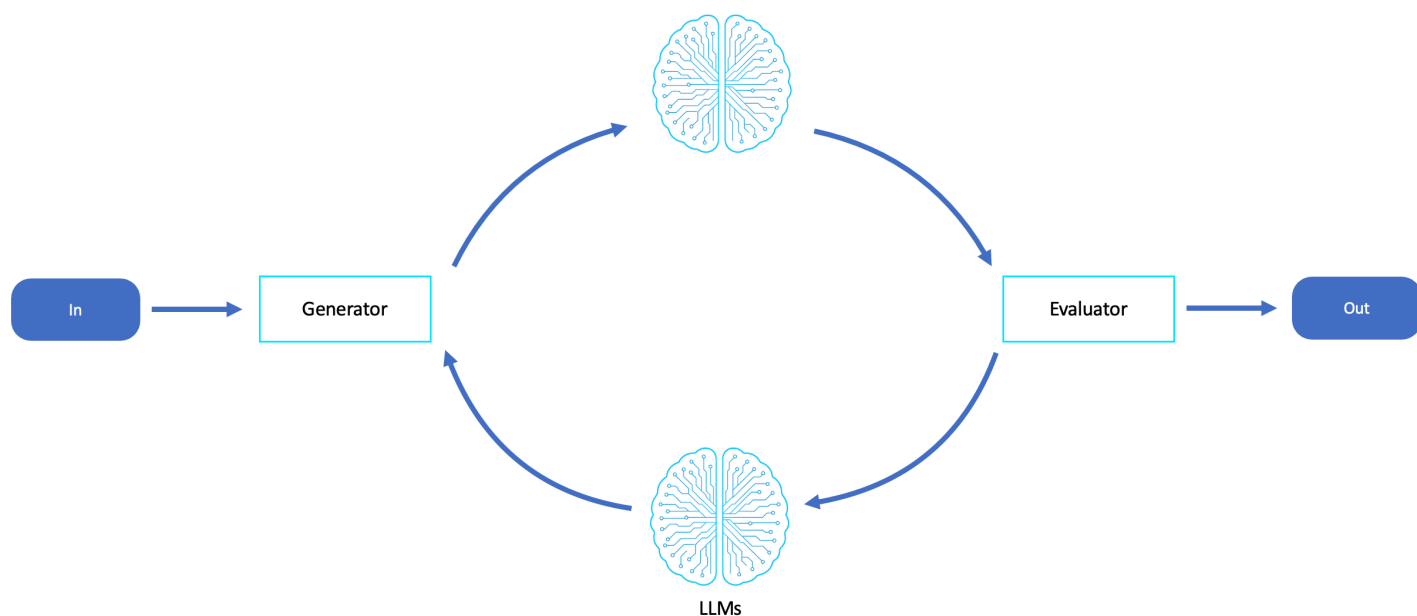
- Orchestrator 执行目标元推理
- Worker 代理可能包括工具访问权限、内存或特定域的提示
- 可以是分层的（即多级任务委派）

常见使用案例

- 项目经理、协调研究人员、作家和质量保证人员
- 将计划、执行和测试结合在一起的编程副驾驶
- 监督工具链或 API 访问模式的代理

赋值器的工作流程和反射优化循环

该工作流程提供了一个反馈循环，其中一个法学硕士生成结果，另一个法学硕士评估或批评结果。这促进了自我反思、优化和迭代改进。



评估器工作流程非常适合输出质量、准确性和对齐性很重要，以及单通道生成不可靠或不足的场景。当工程师必须自我批评、迭代和完善其输出时，这种工作流程非常出色，要么是为了达到更高的正确性标准，要么是为了根据反馈探索改进的替代方案。

在以下情况下，此工作流程特别有效：

- 输出涉及主观质量指标（例如风格、语气和可读性）或客观标准（例如正确性、安全性和性能）。
- 代理必须权衡推理、评估约束条件或针对目标进行优化。
- 您需要内置的冗余和质量保证，尤其是在受管制、面向客户或创意领域。
- Human-in-the-loop 审核费用昂贵或不可用，需要自主验证。

此工作流程用于内容生成、代码合成和审查、策略执行、对齐检查、指令调整和 RAG 后处理。它对自我完善的代理也很有用，在这些代理中，持续的反馈有助于随着时间的推移形成更好的响应，从而建立值得信赖的自主决策循环。

常见使用案例

- Red-team 特工与蓝队特工的比较
- 生成、评估和修改代码或计划的代理
- 质量保证、幻觉检测和风格强制执行

功能

- 支持使用不同的模型进行解耦生成和评估（例如，Claude 用于生成，Mistral 用于评估）
- 反馈是结构化的，用于提示修改后的产出
- 支持多次迭代或收敛阈值

结论

LLM 提供了现代软件代理的认知核心，但是原始模型调用不足以实现有目的、强大和可控的智能。要从输出生成转向结构化推理和目标一致的行为，必须将 LLM 嵌入到有意的工作流程模式中，这些模式定义了模型如何处理输入、管理上下文和协调操作。

法学硕士工作流程引入了构建代理认知模块的基础：

- 提示链接将复杂的推理分解为模块化、可审计的步骤。
- 路由支持智能任务分类和有针对性的委派。
- 并行化可加快吞吐量并促进多样化推理。
- 代理编排通过任务分解和基于角色的执行来构建多代理协作。
- Evaluator（反射精炼循环）可实现自我完善、质量控制和校准检查。

每个工作流程都代表一种可组合模式，可以根据代理的需求、任务的复杂性和用户的期望进行调整。这些工作流程并不相互排斥。它们是构建块，通常组合成支持动态推理、多代理协调和企业级可靠性的混合架构。

当你过渡到关于代理工作流程模式的下一章时，这些 LLM 工作流程将作为嵌入式结构重新出现在更大的系统中，支持目标授权、工具编排、决策循环和生命周期自主权。掌握这些法学硕士工作流程对于设计软件代理至关重要，这些代理不仅可以预测文本，还要有目的地进行推理、调整和行动。

代理工作流程模式

Agentic 工作流程模式将模块化软件代理与结构化大型语言模型 (LLM) 工作流程集成在一起，从而实现自主推理和操作。虽然受到传统的无服务器和事件驱动架构的启发，但这些模式将核心逻辑从静态代码转移到 LLM-augmented 代理，从而增强了适应性和上下文决策能力。这种演变将传统的云架构从确定性系统转变为能够进行动态解释和智能增强的架构，同时保持了可扩展性和响应能力的基本原则。

本节内容

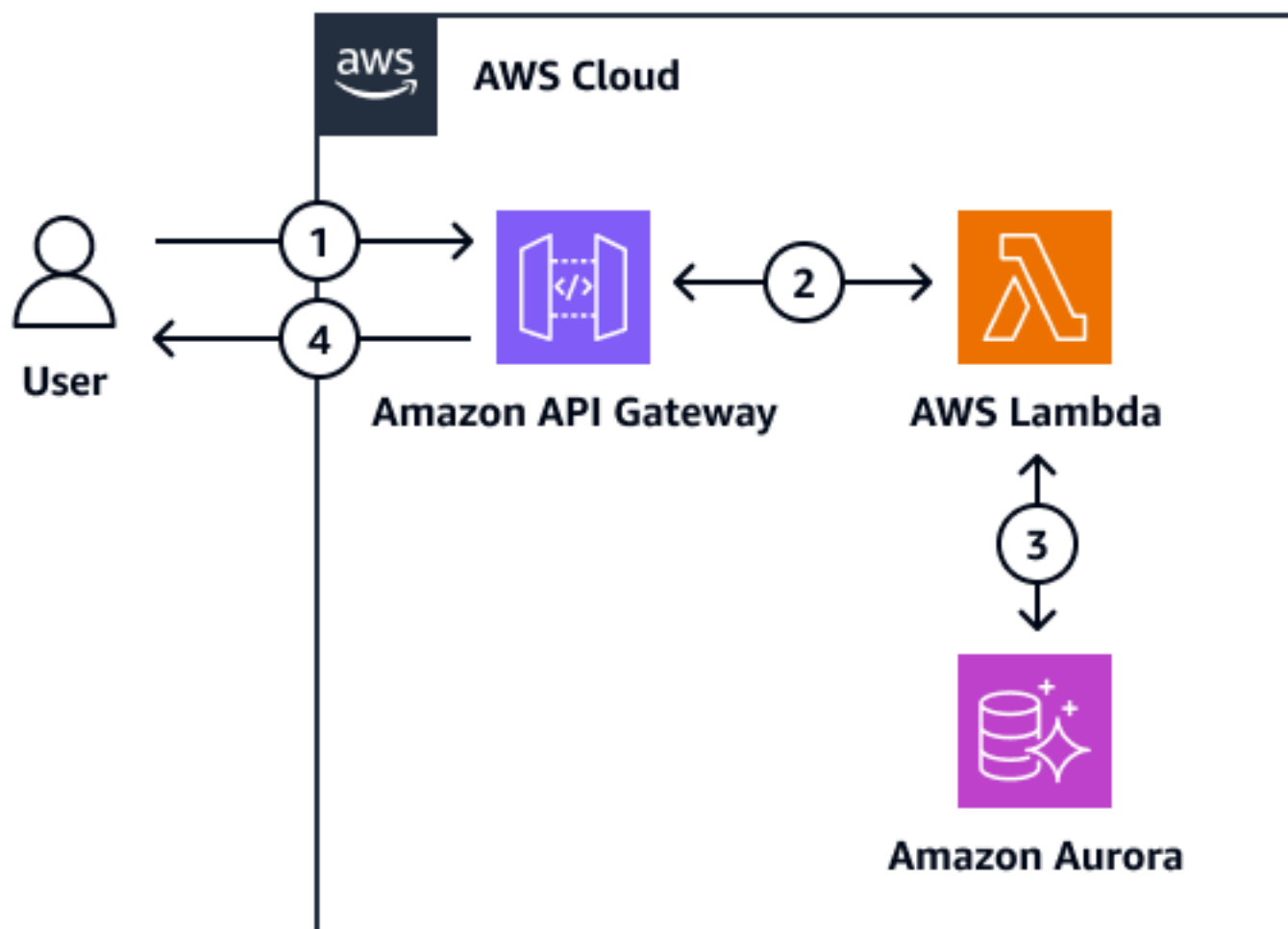
- [从事件驱动到认知增强系统](#)
- [提示链接传奇模式](#)
- [路由动态调度模式](#)
- [并行化和分散-聚集模式](#)
- [传奇编排模式](#)
- [赋值器反射精细化环路模式](#)
- [在上设计代理工作流程 AWS](#)
- [结论](#)

从事件驱动到认知增强系统

现代云架构，尤其是那些基于无服务器和事件驱动原则的云架构，传统上依赖路由、扇出和丰富等模式来创建响应迅速、可扩展的系统。Agentic AI 系统建立在这些基础之上，同时围绕 LLM-augmented 推理和认知灵活性对其进行重新构建。这种方法允许更复杂的问题解决和自动化功能，有可能彻底改变云环境中处理复杂任务的方式。

Event-driven 建筑

下图显示了一个典型的分布式系统：

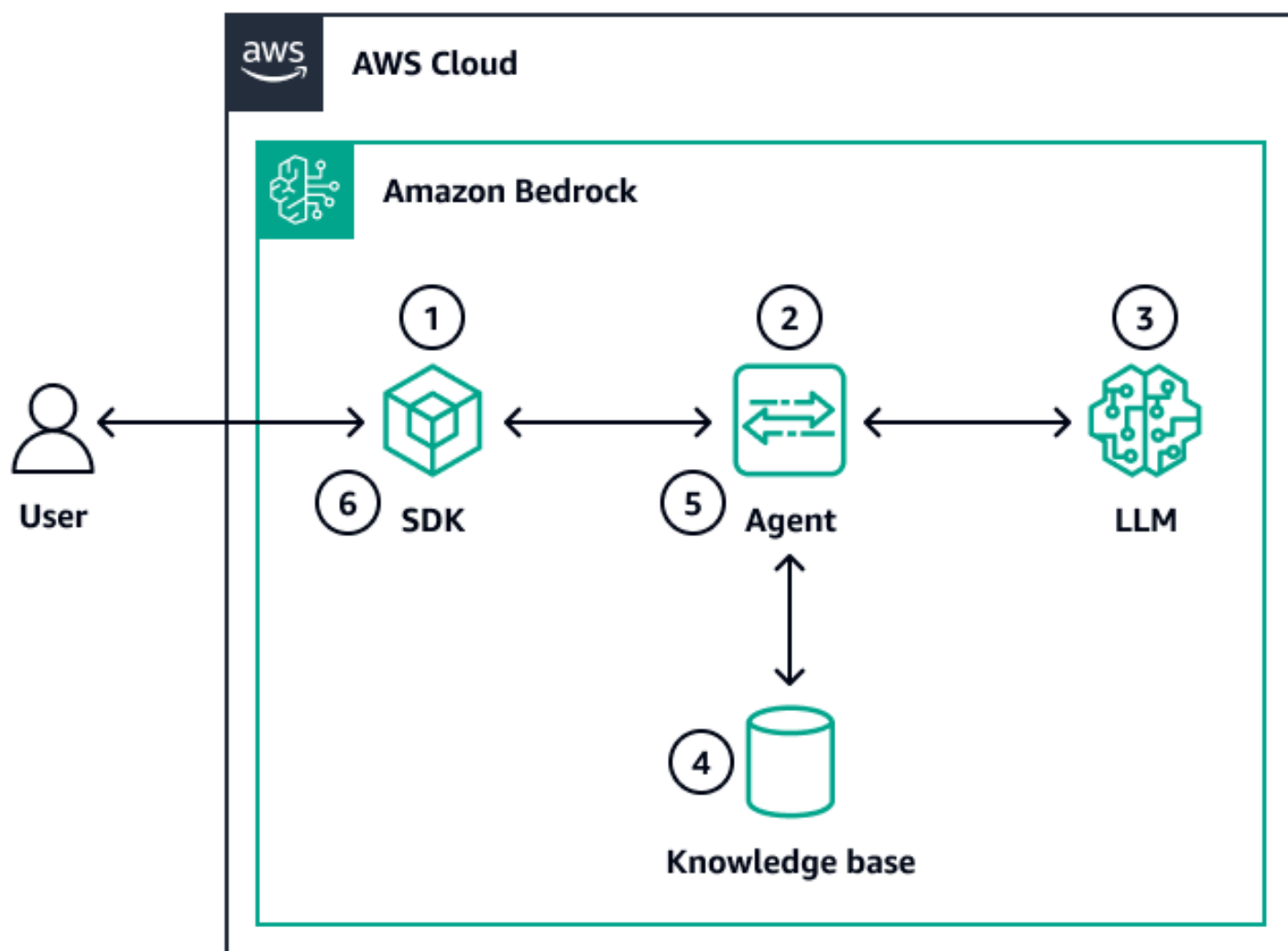


1. 用户向 Amazon API Gateway 提交请求。
2. Amazon API Gateway 会将请求路由到一个 AWS Lambda 函数。
3. AWS Lambda 通过查询 Amazon Aurora 数据库来进行数据扩充
4. Amazon API Gateway 会将充实的有效负载返回给调用方。

这种结构既可靠又可扩展，但它基本上是静态的。必须对业务规则和逻辑路径进行明确编码，适应不断变化的上下文或不完整信息是有限的。

Cognition-augmented 工作流程

Agentic 架构为事件驱动的系统增加了认知增强功能。下图显示了代理等效物：



1. 用户通过 SDK 或 API 调用提交查询。
2. Amazon Bedrock 代理收到查询。
3. 代理通过调用 LLM 来解释查询
4. 代理通过搜索 Amazon Bedrock 知识库或其他外部数据源来进行语义扩充。
5. 法学硕士综合了情境丰富、目标一致的回应。
6. 系统向用户返回合成响应。

在此流程中，法学硕士使用逻辑，理解意图，检索并组合相关上下文，然后决定如何最好地做出响应。这种模式反映了传统的扩充模式，即消息在进一步路由之前先用外部数据进行扩充。但是，在代理系统中，这种浓缩并不是一种静态的查找。相反，丰富是动态的、语义指导的，由目的驱动。

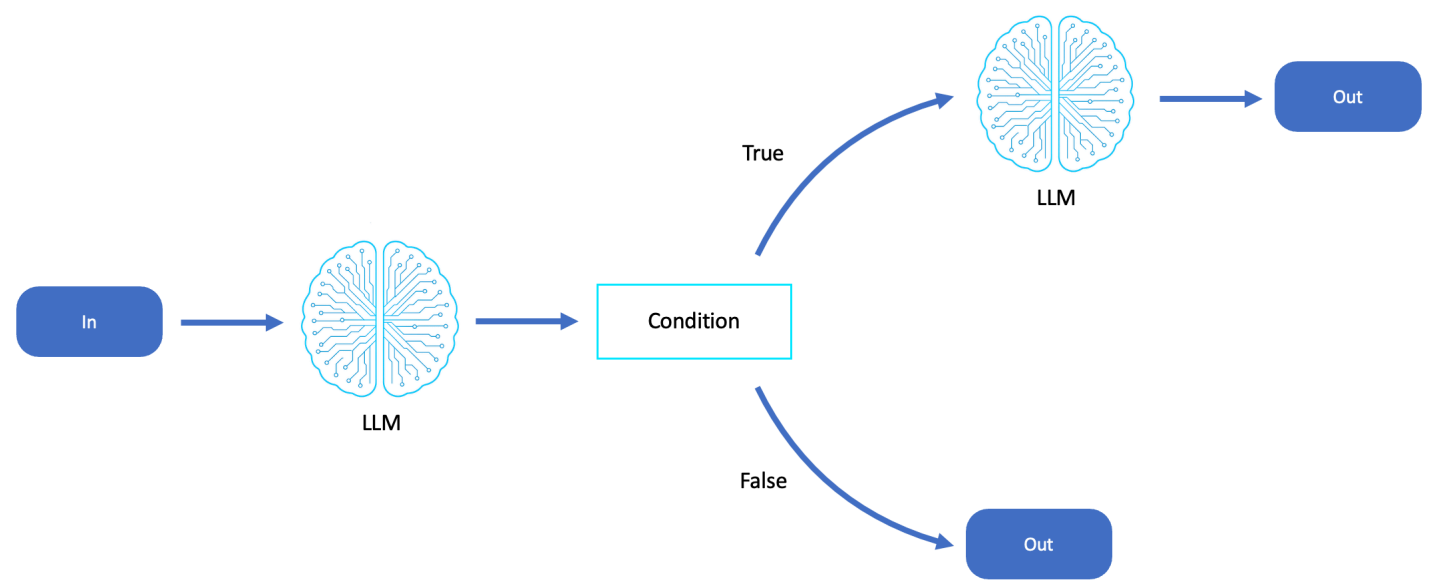
核心见解

每个 LLM 工作流程都可以映射到代理工作流程模式，该模式反映并演变了传统的事件驱动架构风格。代理工作流程的一个基本组成部分是能够使用数据、工具和内存来增强法学硕士的上下文。这会创建一个明智、自适应且符合用户意图的推理循环。传统系统通过查找数据来丰富消息，而代理系统使软件的行为与其说是脚本，不如说是智能协作者。

提示链接传奇模式

通过将 LLM 提示链重新构想为一个事件驱动的传奇，我们解锁了一种新的运营模式：工作流程变得分散、可恢复，并且可以跨自治代理进行语义协调。每个提示响应步骤都被重新定义为原子任务，作为事件发出，由专用代理使用，并富含上下文元数据。

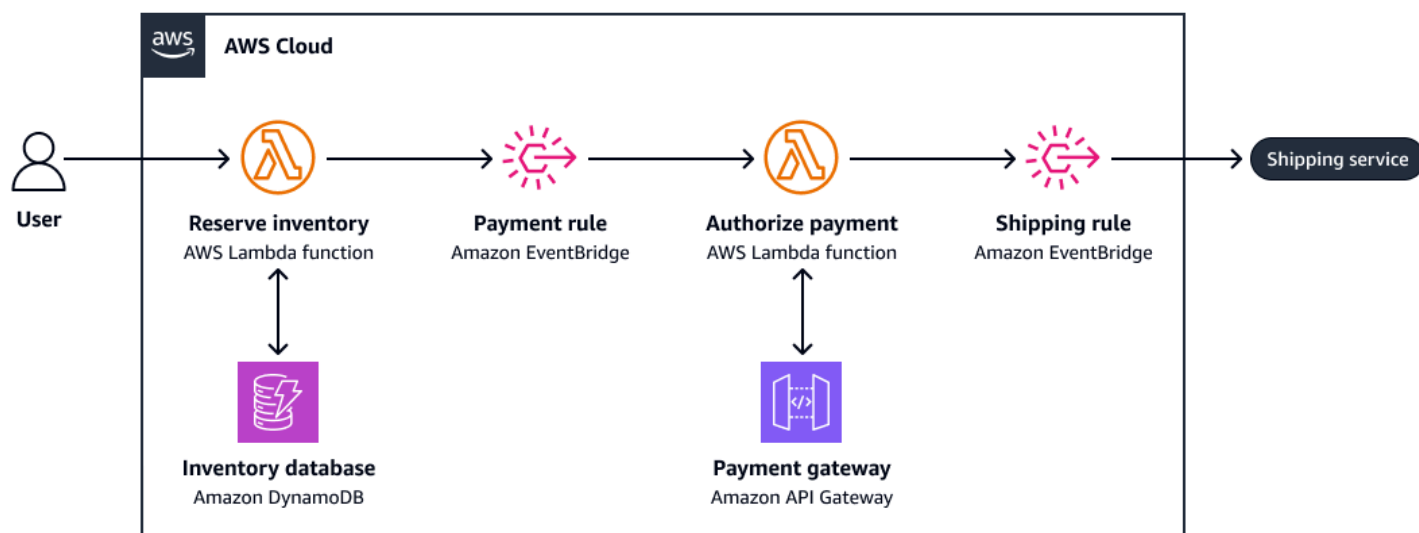
下图是 LLM 提示符链接的示例：



saga 编配

传奇编舞模式是一种在没有中央协调器的分布式系统中的实现方法。相反，每个服务或组件都会发布触发下一个工作流程操作的事件。这种模式广泛用于分布式系统中，用于管理跨多个服务的交易。在传奇中，该系统运行了一系列协调的本地交易。如果一个失败，系统会触发补偿操作以保持一致性。

下图是传奇编舞的示例：



1. 预留库存
2. 授权付款
3. 创建配送订单

如果步骤 3 失败，系统将调用补偿操作（例如，取消付款或发放库存）。

这种模式在事件驱动的架构中特别有价值，在这种架构中，服务是松散耦合的，即使存在部分故障，也必须随着时间的推移一致地解析状态。

提示链接模式

提示链接在结构和目的上都类似于传奇模式。它执行一系列推理步骤，这些步骤按顺序构建，同时保留上下文并允许回滚和修订。

特工编舞

1. LLM 解释复杂的用户查询并生成假设
2. 法学硕士详细制定了解决任务的计划
3. LLM 执行子任务（例如，通过使用工具调用或检索知识）
4. 如果 LLM 认为结果不令人满意，则会完善输出或重新审视之前的步骤

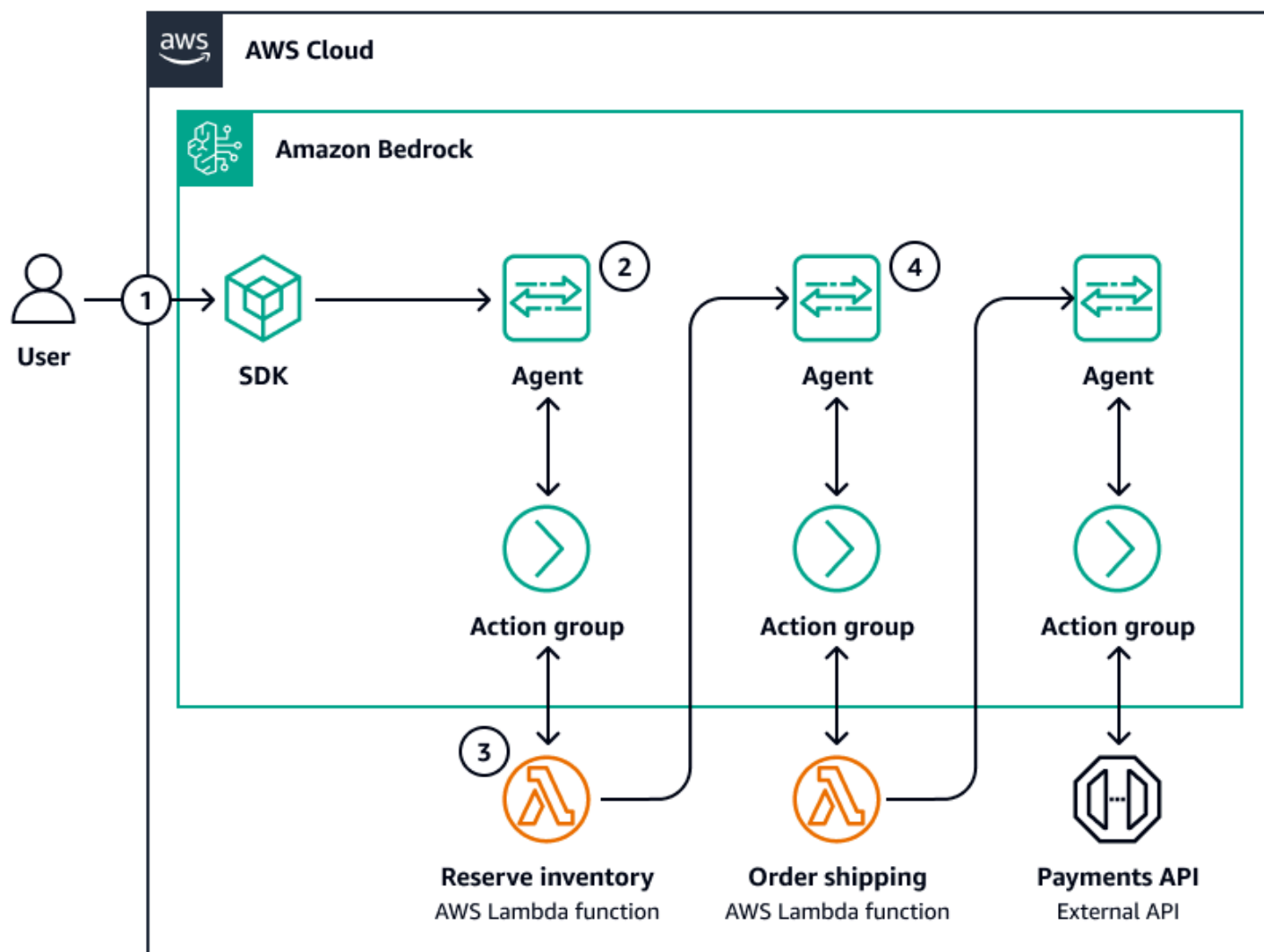
如果中间结果存在缺陷，则系统可以执行以下操作之一：

- 使用其他方法重试这些步骤

- 恢复到之前的提示并重新计划
- 使用赋值器循环（例如，来自赋值器-优化器模式）来检测和纠正故障

与传奇模式一样，提示链接允许部分进度和回滚机制。这是通过迭代细化和 LLM-directed 更正而不是通过补偿数据库事务来实现的。

下图是特工编舞的示例：



1. 用户通过 SDK 提交查询。
2. Amazon Bedrock 代理通过以下方式精心策划推理：
 - 口译（法学硕士）
 - 规划（法学硕士）
 - 通过工具或知识库执行

- 响应构建

3. 如果工具出现故障或返回的数据不足，代理可以动态地重新计划或重新表述任务。

4. 内存（例如，短期向量存储）可以跨步骤保持其状态

外卖

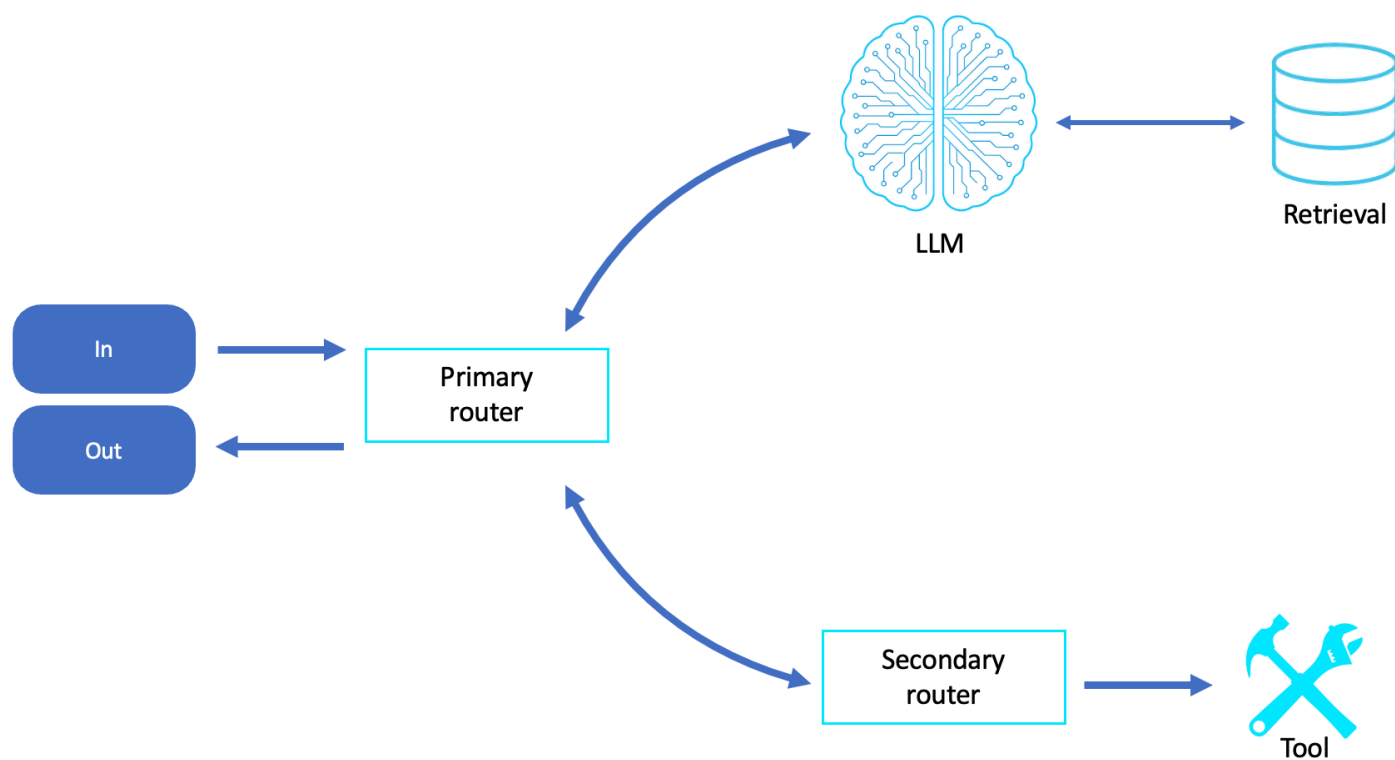
传奇模式使用补偿逻辑管理分布式服务调用，而提示链则通过反射式排序和自适应重新规划来管理推理任务。这两个系统都允许渐进式进展、分散的决策点和故障恢复，所有这些都是通过明智的推理而不是僵硬的回滚来实现的。

提示链接引入了交易推理，这在认知上等同于传奇。也就是说，作为更广泛的目标导向对话的一部分，每个“想法”都会被重新评估、修改或放弃。

路由动态调度模式

在现代代理系统中，任务范围从文档解析到自主软件生成，将请求动态路由到功能最强的大型语言模型 (LLM) 或代理的能力变得至关重要。静态路由逻辑通常嵌入在编排脚本或 API 层中，缺乏实时、多模型、多功能环境所需的适应性。为了解决这个问题，可以将 LLM 路由工作流转换为事件驱动的架构，该架构利用动态调度模式，将 LLM 调用转换为智能路由、上下文感知的事件。

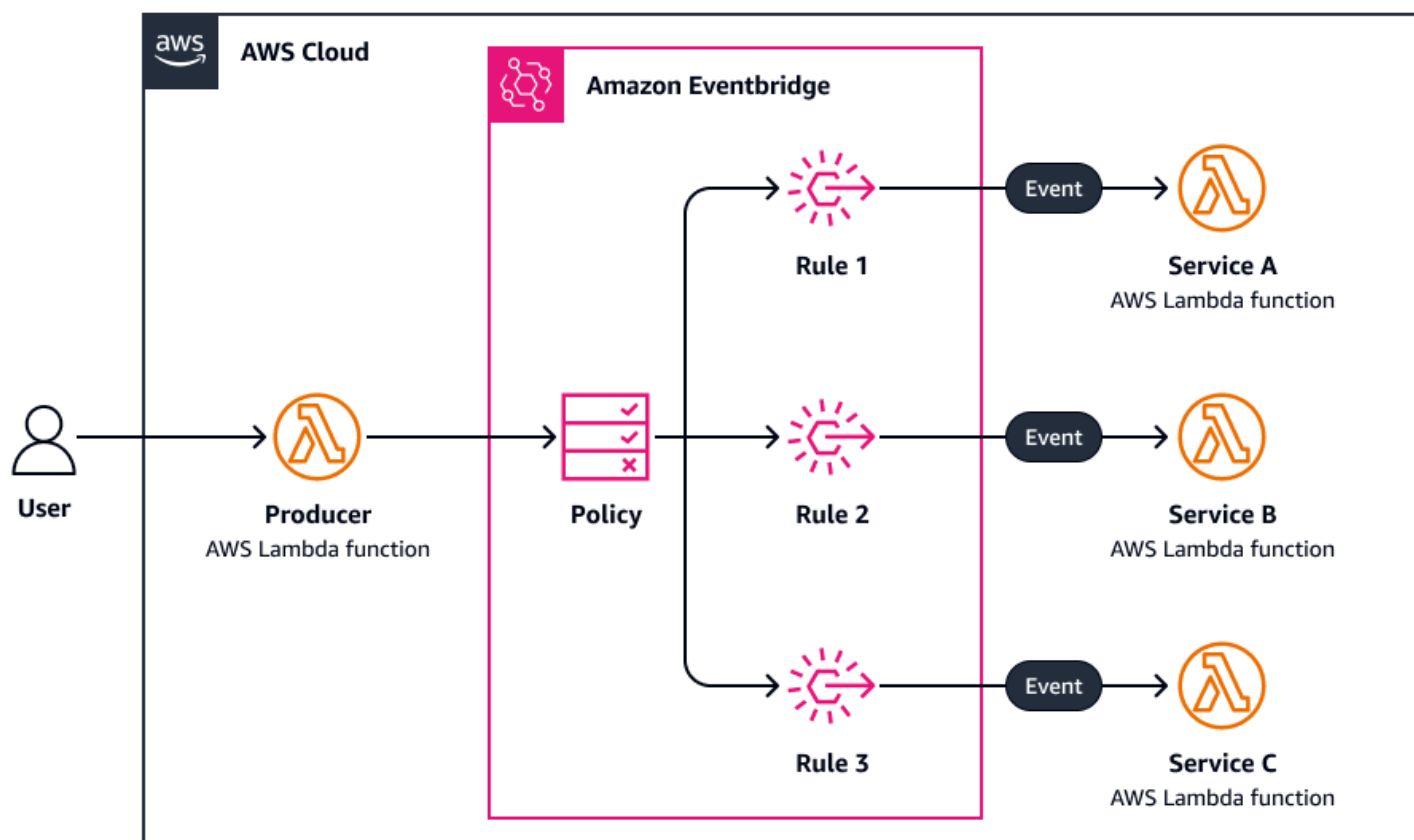
下图是 LLM 路由的示例：



动态调度

在传统的分布式系统中，动态调度模式根据传入的事件属性（例如事件类型、源和负载）在运行时选择和调用特定的服务。这通常使用 Amazon 实现 EventBridge，亚马逊可以评估传入的事件并将其路由到相应的目标（例如 AWS Step Functions，AWS Lambda 函数或亚马逊弹性容器服务任务）。

下图是动态调度的示例：



1. 应用程序会发出一个事件（例如，{"type": "orderCreated", "优先级": "高"}）。
2. Amazon EventBridge 会根据其路由规则评估事件。
3. 根据事件的属性，系统会动态调度到以下内容：
 - HighPriorityOrderProcessor（服务 A）
 - StandardOrderProcessor（服务 B）
 - UpdateOrderProcessor（服务 C）

此模式支持松散耦合、基于域的专业化和运行时可扩展性。这使系统能够智能地响应不断变化的需求和事件语义。

LLM-based 路由

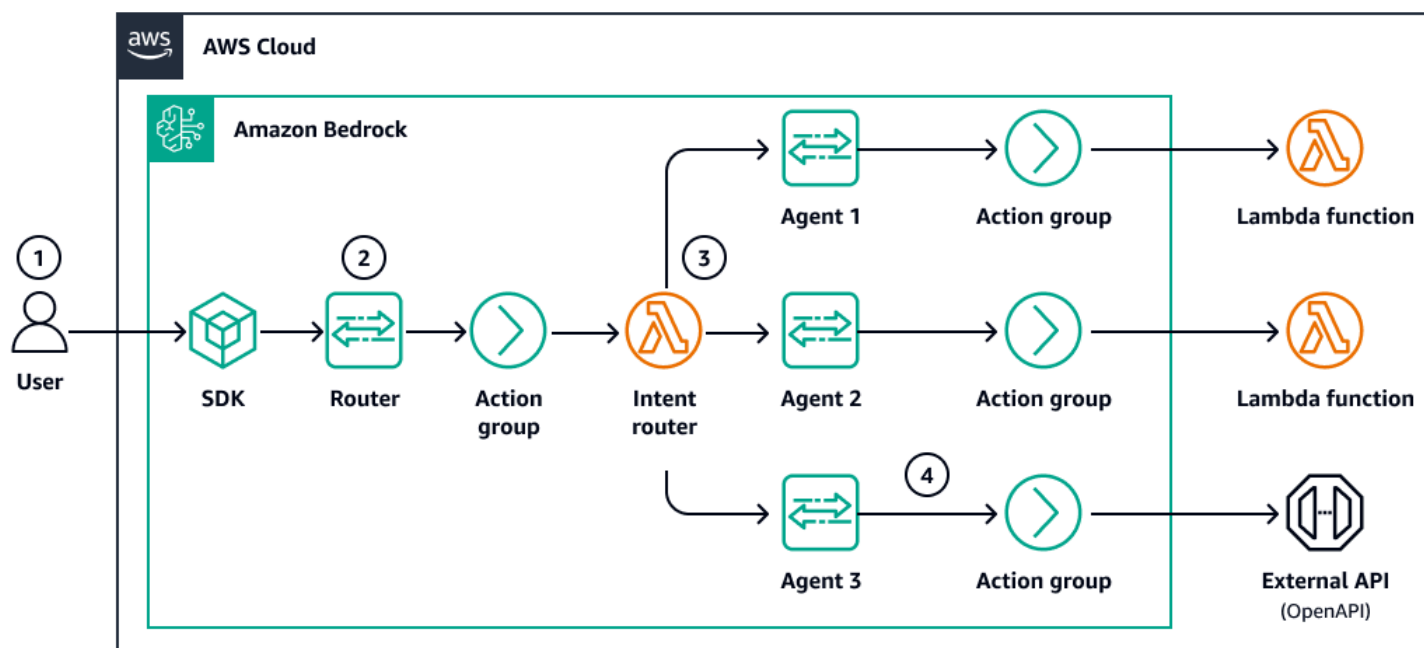
在代理系统中，路由还会执行动态任务委派，但是 LLM 不是亚马逊 EventBridge 规则或元数据筛选器，而是通过自然语言对用户的意图进行分类和解释。结果是一种灵活、语义化和自适应的调度形式。

代理路由器

这种架构支持基于意图的丰富调度，无需预定义架构或事件类型，非常适合非结构化输入和复杂查询。

1. 用户提交了请求“你能帮我查看合同条款吗？”
2. 法学硕士将其解释为一项法律文件任务。
3. 代理将任务路由到以下一项或多项：
 - 合同审查提示模板
 - 法律推理子代理
 - 文档解析工具

下图是代理路由器的示例：



1. 用户通过 SDK 提交自然语言请求。
2. Amazon Bedrock 代理使用 LLM 对任务进行分类（例如，法律、技术或日程安排）。
3. 代理通过操作组动态路由任务以调用所需的代理：
 - Domain-specific 代理人
 - 专业工具链
 - 自定义提示符配置
4. 选定的处理器处理任务并返回定制响应。

外卖

传统动态调度使用 Amazon EventBridge 规则根据结构化事件属性进行路由，而代理路由则使用 LLM 根据含义和意图对任务进行语义分类和路由。这通过启用以下功能扩展了系统的灵活性：

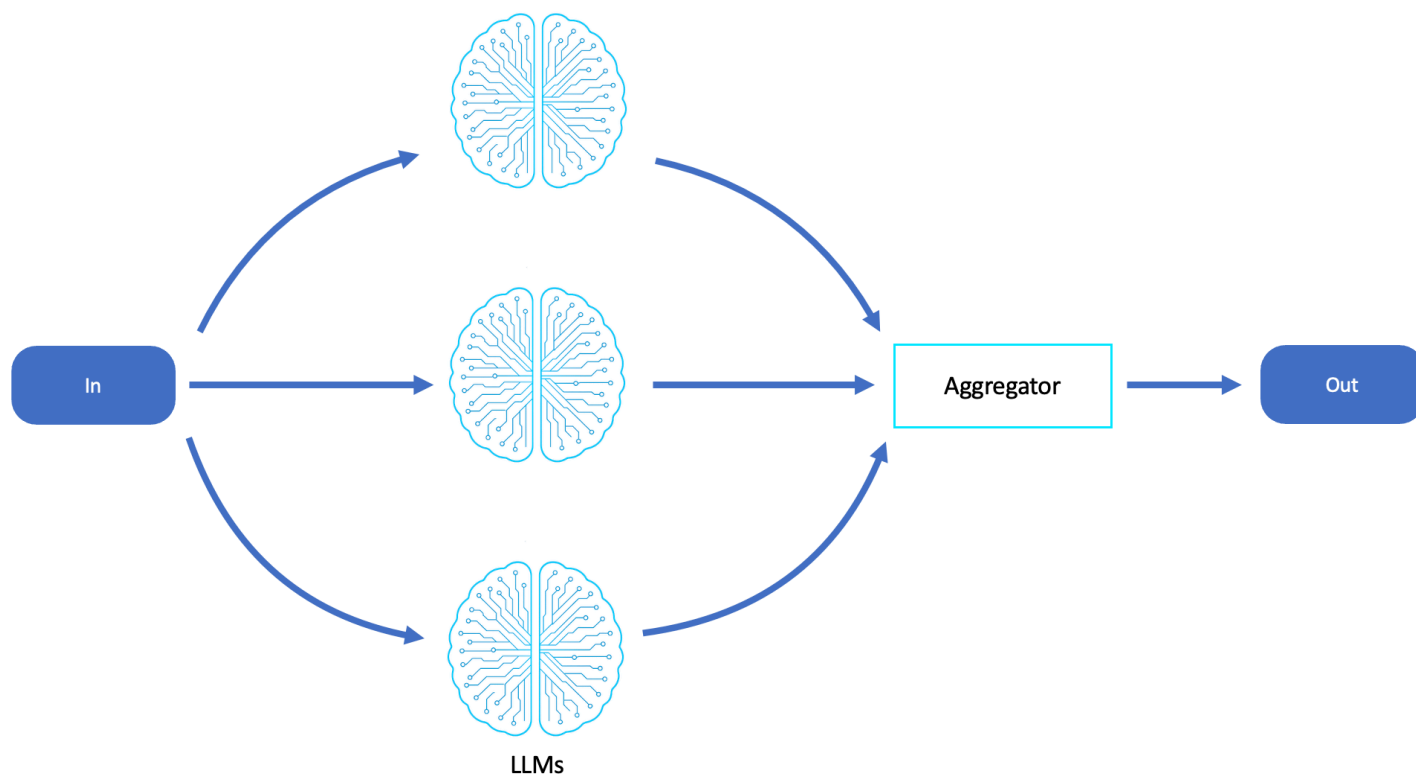
- 更广泛的输入理解
- 智能备用和刀具选择
- 通过新的代理角色或提示样式实现自然的可扩展性

代理路由用动态认知调度取代了严格的规则，这允许系统随着语言而不是代码的变化而发展。

并行化和分散-聚集模式

许多高级推理和生成任务（例如总结大型文档、评估多个解决方案路径或比较不同的视角）都受益于提示的并行执行。当需要可扩展性、响应能力和容错能力时，传统的顺序工作流程是不够的。为了克服这个问题，可以使用事件驱动的分散聚集模式重新构想 LLM-based 并行化，在这种模式下，任务被动态分散到自主代理，然后智能地合成结果。

下图是 LLM 并行化工作流程的示例：



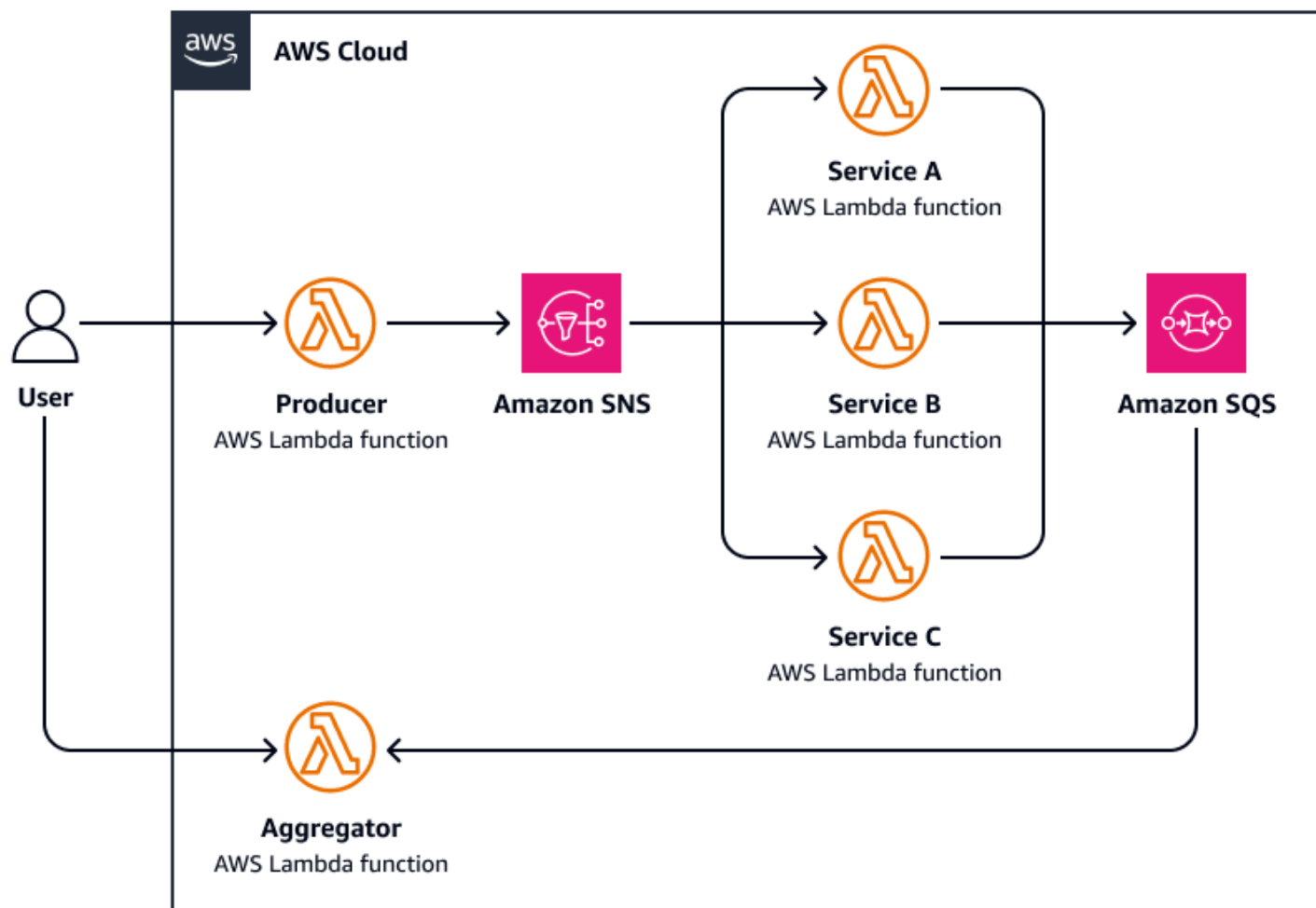
Scatter-gather

在分布式系统中，分散聚集模式将任务并行发送到多个服务或处理单元，等待它们的响应，然后将结果聚合到合并的输出中。与扇出不同，scatter-gather 是协调的，因为它期望得到响应，并且通常会应用逻辑来合并、比较和选择结果。

并行化和分散收集的常见实现包括以下内容：

- AWS Step Functions 映射并行任务执行的状态
- AWS Lambda 通过并发，协调来自多个调用函数的结果
- EventBridge 带有关联 ID 和聚合 workflows 的 Amazon
- 使用亚马逊简单存储服务 (Amazon S3)、Amazon DynamoDB 或队列管理扇出和收集结果的自定义控制器模式

下图是分散聚集的示例：



1. 用户向中央协调器功能发送请求，该功能通过向亚马逊简单通知服务 (Amazon SNS) Simple Notification Service 主题发布并行消息来分散任务。
2. 每条消息都包含任务元数据，并发送给专业工作人员 AWS Lambda。
3. 每个工作人员都 AWS Lambda 独立处理其分配的子任务（例如，查询外部 API、处理文档和分析数据）。
4. 结果会写入公共存储层，例如亚马逊简单队列服务 (Amazon SQS) Simple Queue Service。
5. 聚合器函数等待所有响应完成，然后执行以下操作：
 - 收集和汇总结果（例如，合并摘要、选择最佳匹配项）
 - 发送最终响应或触发下游工作流程

分散聚集模式的常见用例包括：

- 联合搜索
- 价格比较引擎
- 汇总数据分析
- 多模型推理

LLM-based 并行化（分散-聚集认知）

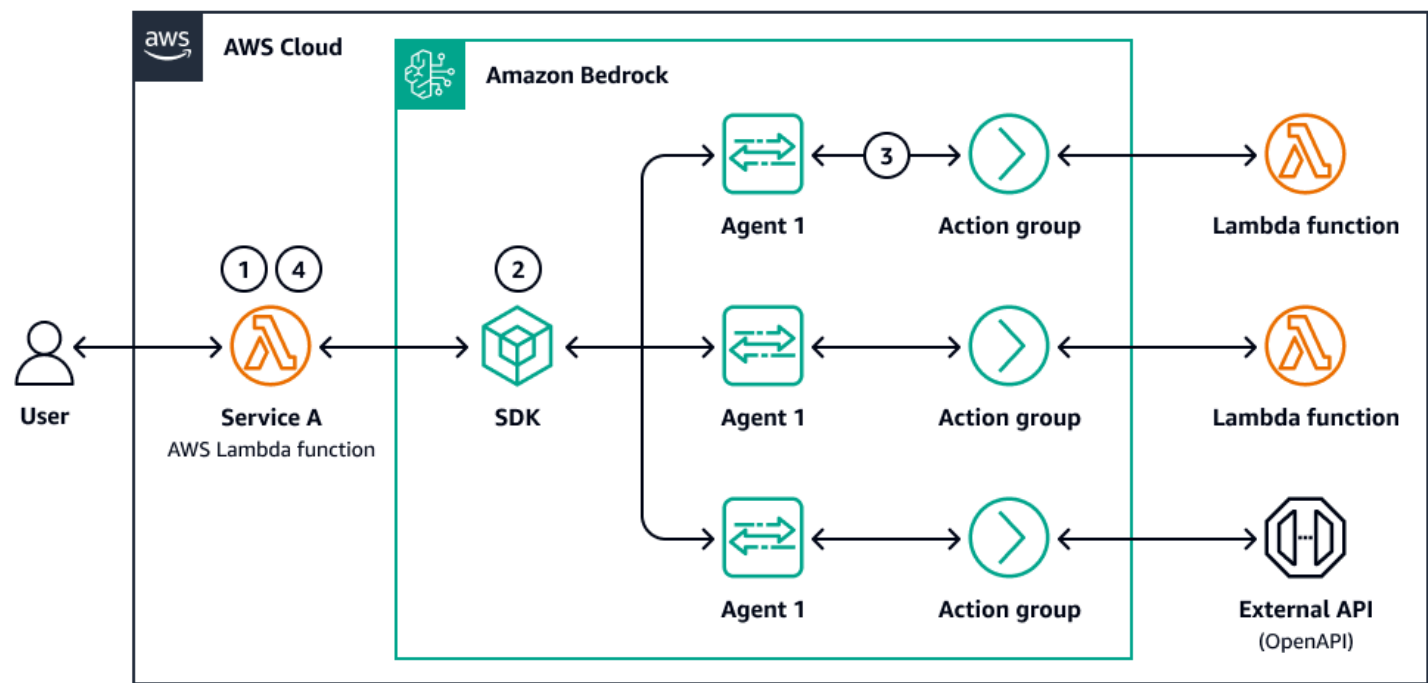
在代理系统中，并行化通过在多个 LLM 调用或代理之间分配子任务来密切反映分散聚集，每个调用或代理独立推理问题的一部分。返回的结果由聚合过程收集和合成，聚合过程通常是另一个 LLM 或控制器代理。

代理并行化

1. 代理提交了“汇总这 10 份报告的意见”请求。
2. 它将报告分散到 10 个并行的 LLM 摘要任务中。
3. 当它返回所有摘要时，代理会执行以下操作：
 - 将摘要汇总到统一的简报中
 - 识别主题或矛盾
 - 将合成后的输出发送给用户

这种代理工作流程支持可扩展、模块化和自适应的并行推理。这非常适合需要高认知吞吐量的用例。

下图是代理并行化的示例：



1. 用户提交分段查询或文档集。
2. 控制器 AWS Lambda 或步进函数分配子任务。每个任务都会使用自己的提示调用 Amazon Bedrock LLM 调用或子代理。
3. 调用和子任务完成后，结果将被存储（例如，在 Amazon S3 或内存存储中），然后聚合步骤会合并、比较或筛选输出。
4. 系统将最终响应返回给用户或下游代理。

该系统具有分布式推理循环，具有可追溯性、容错能力以及可选的结果加权或选择逻辑。

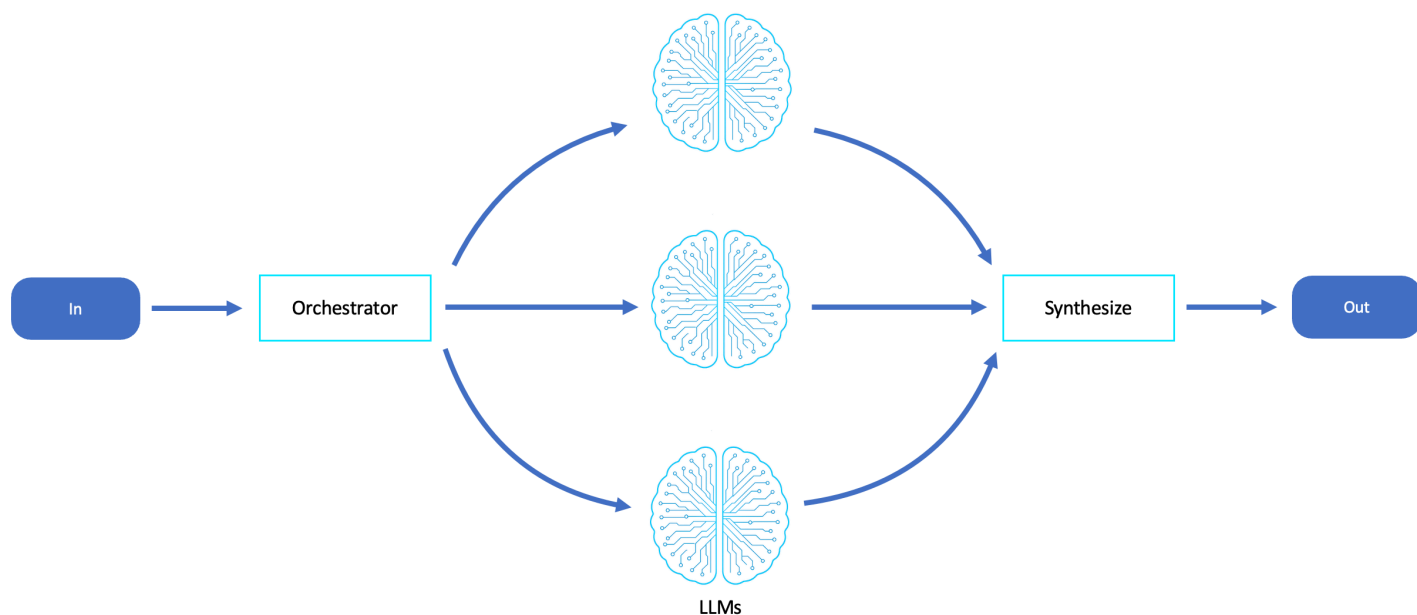
外卖

代理并行化使用分散聚集模式来分发 LLM 任务，从而实现并行处理和智能结果合成。

传奇编排模式

随着由 LLM 驱动的工作流程变得越来越复杂，跨越提示链、数据处理步骤、工具调用和代理协作，对智能编排的需求变得至关重要。这些工作流程可以作为事件驱动的编排模式来实现，而不是依赖紧密耦合的脚本或静态的预先确定的执行流程，从而使 LLM-based 系统能够动态地协调、监控和调整自主代理之间的多步骤任务。

下图是协调器的示例：



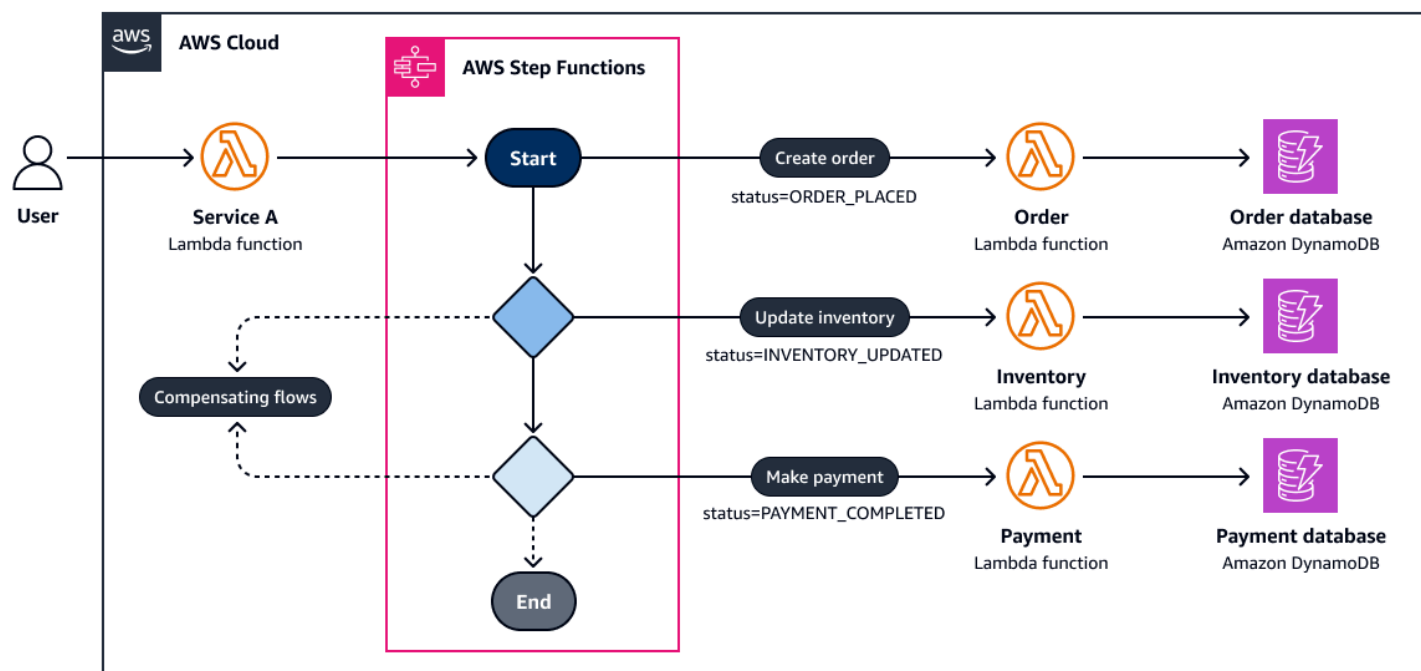
活动编排

在传统的分布式系统中，事件编排是指一种模式，在这种模式中，中央协调员通过明确指挥跨多个服务或任务的控制流来管理复杂的工作流程。与活动编排（每项服务独立做出反应）不同，编排提供了对整个流程的集中逻辑、可见性和控制。

这通常使用以下工具实现：

- AWS Step Functions— 定义和执行有状态的工作流程
- AWS Lambda— 在精心策划的流程中执行离散的任务
- 亚马逊 SQS 或亚马逊 EventBridge — 触发异步步骤或响应

下图是传奇编排的示例：



AWS Step Functions workflow管理客户订单流程：

1. 创建订单 (AWS Lambda)
2. 更新库存 (AWS Lambda)
3. 付款 (AWS Lambda)

协调器通过管理重试、并行分支、超时和失败来协调每个步骤。

Role-based 代理系统 (协调器)

在代理系统中，协调器模式反映了事件编排，但将逻辑分配给多个推理代理，每个推理代理都有明确的角色或专长。中央协调器代理解释整体任务，将其分解为子任务，然后将这些任务委托给工作代理，每个代理都针对特定领域（例如研究、编码、摘要、审查）进行了优化。

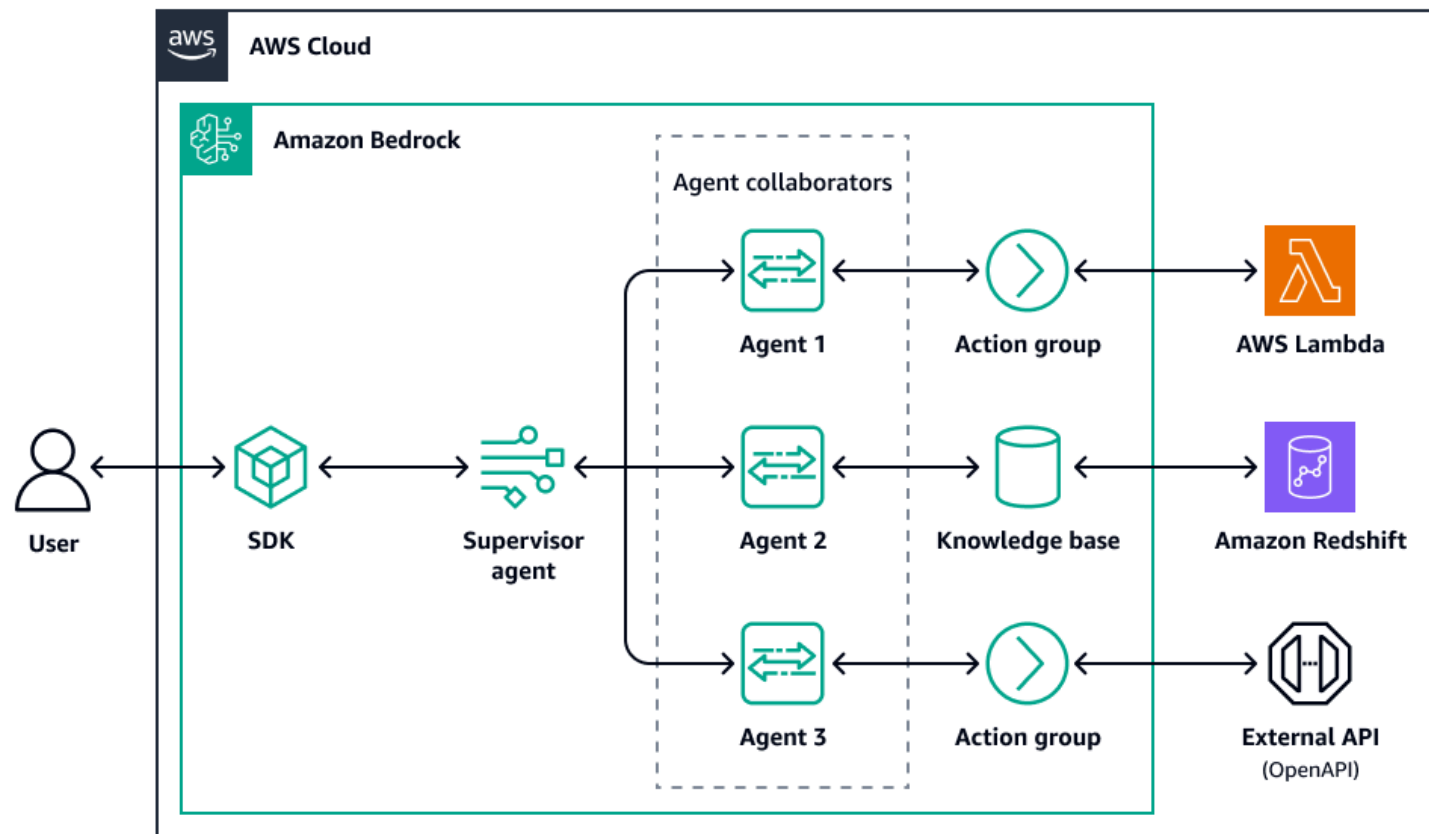
主管

1. 用户提交了“创建项目简介并总结排名前五的竞争对手”的查询。
2. 协调器代理执行以下操作：
 - 指派研究代理人寻找竞争对手的数据
 - 将原始发现结果发送给汇总代理
 - 将结果传递给简报撰稿人代理

- 为用户编译最终输出

每个代理独立运行，但协调器负责协调任务。这就像处理工作流程任务的 Lambda 函数。

下图是主管的示例：



1. 用户向 Amazon Bedrock 主管代理提交任务。
2. 主管代理将请求解析为每个代理协作者的子任务。
3. 每个子任务都分配给带有特定角色提示或工具链的协作者代理。
4. 工作人员代理通过操作组调用外部 API 或工具。
5. 每个工作代理都以结构化格式返回输出。
6. 当所有工作人员返回结果时，主管会评估、综合并返回最终响应。

这种结构允许在复杂的多步骤代理工作流程中实现模块化、适应性和内省。

外卖

事件编排使用集中控制（例如 AWS Step Functions）来指导服务执行，而基于角色的代理系统则使用 LLM-powered 协调器代理来推理目标，将子任务委托给工作代理并合成最终输出。

在这两种范式中，协调器都会执行以下操作：

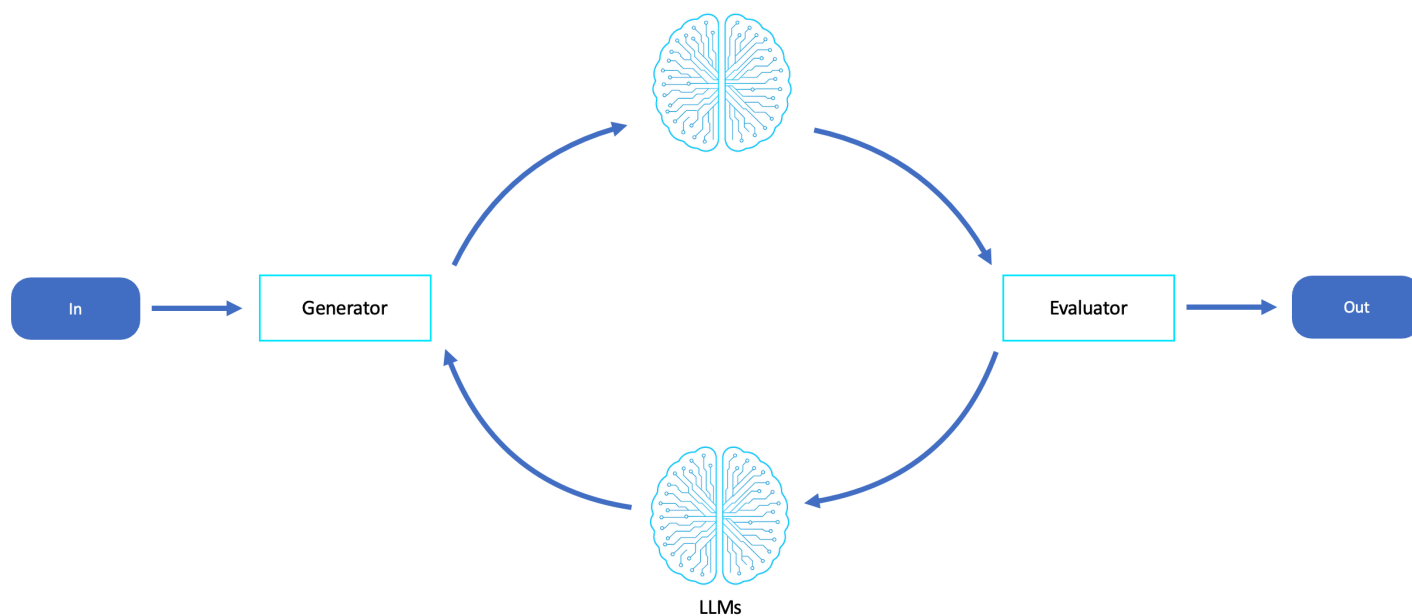
- 维护上下文和执行流程
- 处理分支、排序和错误处理
- 通过分布式组件生成统一的结果

但是，代理编排增加了推理、适应性和语义委托。这使得它非常适合开放式、模棱两可和不断演变的任务。

赋值器反射精细化环路模式

诸如代码生成、摘要或自主决策之类的任务可以从运行时反馈中受益匪浅，从而使系统能够通过观察和完善来发展。为了实现这一点，可以将反射-精炼周期作为事件驱动的反馈控制回路来实现，这种模式的灵感来自系统工程，适用于自主、智能的工作流程。

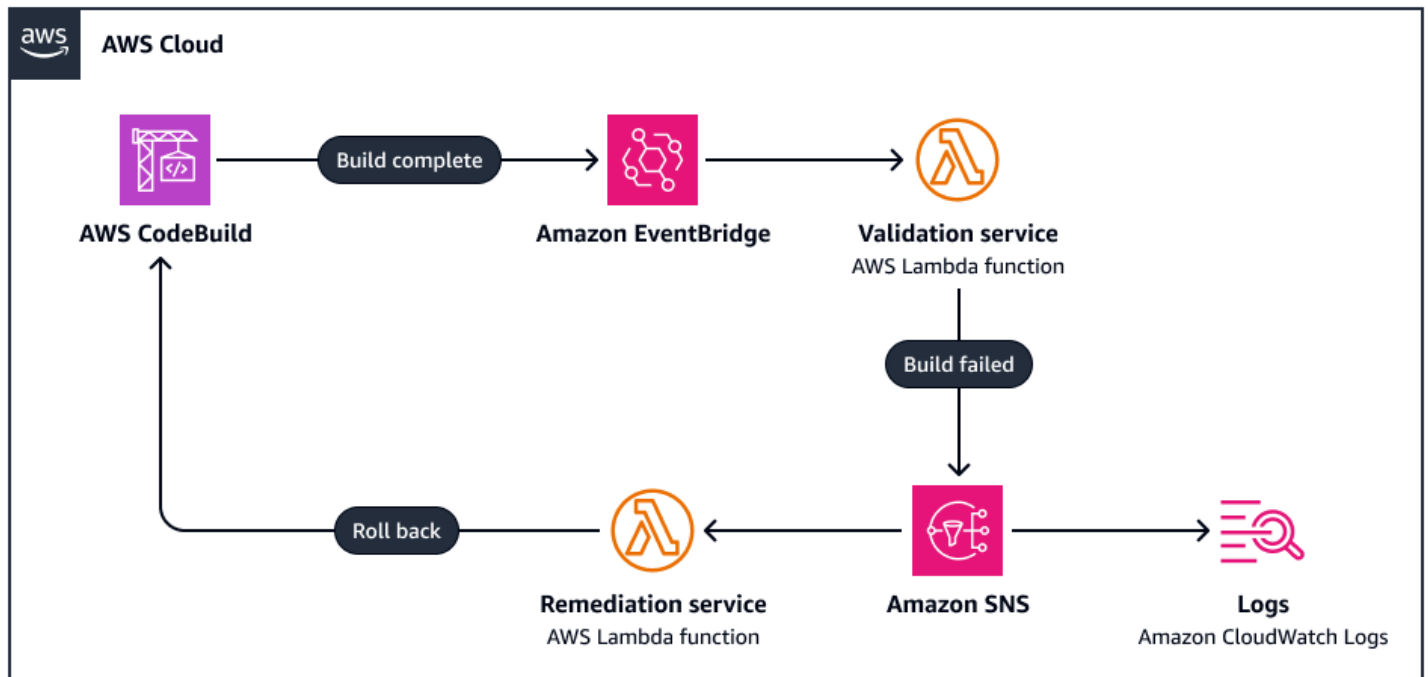
下图是评估器反射精简反馈回路的示例：



反馈控制回路

反馈控制回路是一种模式，它监视自己的输出和行为，根据定义的标准或所需状态对其进行评估，然后相应地调整其动作。该架构的灵感来自控制理论，是自动化、持续集成和持续交付 (CI/CD) 管道以及机器学习操作等领域的基础。

下图是反馈控制回路的示例：



1. 部署管道会发出 buildComplete 事件。
2. 该事件会触发自动测试或评估作业，以验证构建。
3. 如果验证失败（例如，由于测试失败、安全问题或违反策略），则系统：
 - 发出 BuildComplete 事件
 - 记录问题或发送通知
 - 触发补救或更正操作，例如回滚、修补或重试

循环一直持续到产生可接受的结果或升级，或者出现超时。此模式通常用于以下用途：

- Amazon 将事件路由到评估或补救任务的 EventBridge 规则
- AWS Step Functions 用于迭代重试逻辑和对评估结果进行分支
- 亚马逊简单通知服务 (Amazon SNS) Simple Notification Service 或 Amazon CloudWatch mazon 警报，用于发送反馈触发器和警报

- AWS Lambda 职能或容器化工作人员采取纠正措施

反馈控制回路（评估器）

评估者工作流程是一个由法学硕士或推理代理提供支持的认知反馈循环。该过程包括以下内容：

1. 生成器代理或 LLM 生成输出（例如，计划、答案或草稿）。
2. 评估人员使用批评提示或评估标准来审查结果。
3. 根据反馈，原始代理或新的优化器代理修改输出。

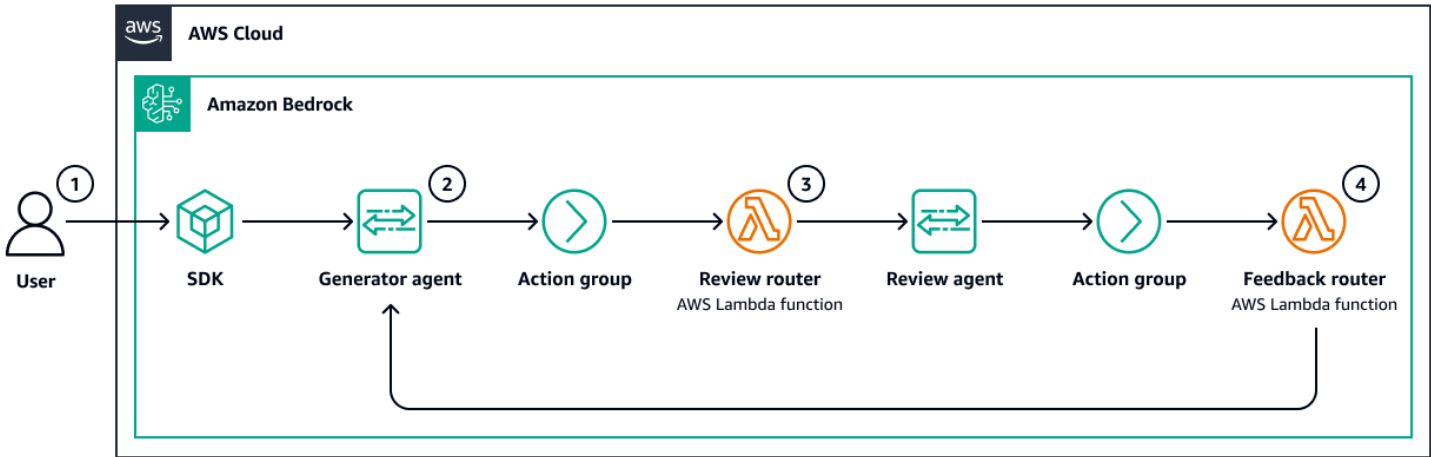
循环会重复，直到结果符合一组标准、获得批准或达到重试限制。

评估者

1. 用户要求代理撰写策略摘要。
2. 生成器代理起草它。
3. 评估人员检查覆盖范围、语气和法律正确性。
4. 如果响应不充分，则会对其进行完善并重新提交，直到反馈回路趋于一致。

这可以实现自我评估、迭代完善和自适应输出控制，所有这些都无需人工输入。

下图是反馈控制回路（赋值器）的示例：



1. 用户下达任务（例如，起草业务策略）。
2. Amazon Bedrock 代理使用 LLM 生成初始草稿。

3. 第二个代理（或后续提示）执行结构化评估（例如，“根据清晰度、完整性和语气对输出进行评分”）。
4. 如果评级低于阈值，则通过以下方式修改响应：
 - 使用嵌入式评论重新调用生成器
 - 将反馈发送给专业的炼油厂代理商
 - 迭代直到得到可接受的响应

可选组件，例如 AWS Lambda 控制器或 AWS Step Functions 可以管理反馈阈值、重试和回退策略。

外卖

传统的反馈控制回路使用事件、指标和补救逻辑来验证和调整系统行为，而代理评估器循环则使用推理代理来动态评估、反映和修改输出。

在这两种范式中：

- 输出在生成后进行评估
- 根据反馈触发纠正或完善措施
- 系统不断适应目标质量或目标

代理版本将静态验证转换为语义反射，从而使自我完善的代理能够评估自己的有效性。

在上设计代理工作流程 AWS

本指南中的每种模式都可以使用构建 AWS 服务。Amazon Bedrock 代理提供编排、数据访问和互动渠道。

组件	AWS 服务	目的
法学硕士推理	Amazon Bedrock	代理逻辑、规划、工具使用
工具执行	AWS Lambda、亚马逊 ECS、亚马逊 SageMaker	为代理提供外部工具
内存和 RAG	亚马逊 Bedrock 知识库，亚马逊 S3，OpenSearch	持久记忆和语义记忆

编排	AWS Step Functions	多步骤任务和代理协调
事件路由	亚马逊 EventBridge、亚马逊 SQS	解耦代理间消息
用户界面	亚马逊 API Gateway、AWS AppSync、SDK	应用程序或系统的入口点
监控	亚马逊 CloudWatch、AWS X-Ray、AWS Distro for OpenTelemetry	可观察性和代理内省

结论

代理 workflows 模式是事件驱动架构的下一个演变阶段，其中业务逻辑不是静态定义的，而是通过使用大型语言模型 (LLM) (增强认知能力) 来动态推理。通过将传统的云原生原生基元与法学硕士 workflows 和代理设计模式相结合，组织可以构建自适应、智能和模块化的系统，这些系统可以有针对性地响应并从经验中学习。

在这些模式中，Amazon Bedrock 是通往代理认知的门户，它允许 LLM-based 代理访问事件 workflows，与工具和内存进行交互，并提供结构化、可追溯和一致的结果。

在您设计和部署代理系统时，这些 workflows 模式为构建自主、可组合的 AI 架构提供了蓝图。这些系统以无服务器最佳实践为基础，并通过智能基础模型进行了增强。

文档历史记录

下表介绍了本指南的一些重要更改。如果您希望收到有关未来更新的通知，可以订阅 [RSS 源](#)。

变更	说明	日期
初次发布	—	2025 年 7 月 14 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。