Agentic AI 框架、平台、协议和工具已启用 AWS

AWS 规范性指导

AWS 规范性指导: Agentic AI 框架、平台、协议和工具已启用 AWS

Table of Contents

Agentic AI 框架、平台、协议和工具已启用 AWS

Aaron Sempf、Ansley Verzosa 和 Amazon Web Services 的约书亚·塞缪尔 ()AWS

2026 年 1 月（文件历史记录）

Agentic AI 是人工智能、分布式系统和软件工程交汇处的强大范例。这是一类由自主的异步软件代理组成的智能系统，这些代理使用人工智能模型并与工具和资源集成。代理表现出代理权，可以感知背景，理性胜于目标，做出决定，并代表用户或系统采取有针对性的行动。这些代理在分布式环境中独立运行，通常以协作方式运行，旨在通过嵌入式智能、记忆和意图实现委托目标。

然后 AWS，组织可以利用代理人工智能来自动化复杂的工作流程，增强决策流程，并创建响应速度更快的系统。本指南提供有关构建有效的代理人工智能解决方案所必需的关键组件的信息：

- 框架概述了当前的代理人工智能框架，包括对其优势和用例的评论。了解这些框架如何减少模式、协议和工具之间无差别的繁重工作。了解关键选择标准，为您的需求选择合适的框架。
- 平台概述了代理人工智能平台（托管代理、开源编排和混合）以及选择或设计的注意事项。
- 协议探讨了用于代理交互的基本标准化通信协议。 Agent-to-agent协议正在出现，例如开源模型上下文协议 (MCP) 和 Agent2Agent (A2A)，以及其他专有实现。了解常见协议如何使不同的协议实现无缝交互。
- 工具提供有关基于协议的工具（例如 MCP）、框架原生工具和元工具的信息。Organizations 可以构建一个与其工作流程中的关键系统集成的工具包，从而实现基于终端用户和服务器的代理工作流程。

目标受众

本指南适用于寻求在现代云原生应用程序中利用人工智能驱动的软件代理功能的架构师、开发人员和技术领导者。

目标

本指南可以帮助您执行以下操作：

- 比较不同的 agentic AI 框架，为您的用例选择最合适的框架。
- 了解代理人工智能平台，这些平台提供了将单个代理转变为协调的自适应系统的功能。
- 了解开放协议在构建可持续代理人工智能架构方面的优势。
- 在构建代理系统时创建适当的工具集成策略。

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。

框架

A@@ gentic AI 的基础 AWS探讨了实现自主、以目标为导向的行为的核心模式和工作流程。实现这些模式的核心在于框架的选择。框架是预写代码的软件基础，它提供了结构化环境和通用功能，用于构建和管理、工具以及构建生产就绪型自主 AI 代理所需的编排功能。

有效的 agentic AI 框架提供了多种基本功能，可将原始的大型语言模型 (LLM) 交互转化为能够推理、协作和操作的协调智能系统：

- 代理编排可协调单个或多个代理之间的信息流和决策流，从而无需人工干预即可实现复杂的目标。
- 工具集成使代理能够与外部系统和数据源进行交互 APIs，从而将其功能扩展到语言处理之外。有关更多信息，请参阅Strands Agents文档中的工具概述。
- 内存管理提供持久或基于会话的状态，以便在交互之间维护上下文，这对于长时间运行或自适应任务至关重要。更高级的框架采用长期记忆来存储摘要和用户偏好，从而实现个性化和情境感知型代理体验。有关更多信息，请参阅LangChain博客上的 “如何思考代理框架”。
- 工作流定义支持结构化模式，例如链、路由、并行化和反射循环，从而实现复杂的自主推理。
- 通过自主系统的可观察性，部署和监控促进了从开发到生产的过渡。有关更多信息，请参阅 Amazon Bedrock AgentCore 正式上市公告。

这些功能是在整个框架环境中以不同的方法和重点实现的，每种方法和重点都为不同的自治代理用例和组织环境提供了明显的优势。

本节概述并比较了构建代理人工智能解决方案的领先框架，重点介绍了它们的优势、局限性和自主运营的理想用例：

- Strands 特工
- LangChain 和 LangGraph
- crewai
- AutoGen
- ???
- 比较代理人工智能框架

> **Note**
>
> 本节涵盖了专门支持人工智能代理的框架，不包括前端接口或没有代理的生成式人工智能。

Strands Agents

Strands Agents是一个开源 SDK，最初由发布 AWS，如AWS 开源博客中所述。 Strands Agents专为采用模型优先方法构建自主人工智能代理而设计。它提供了一个灵活、可扩展的框架，旨在与之无缝协作， AWS 服务 同时保持与第三方组件集成的开放性。Strands Agents 非常适合构建完全自主的解决方案。

的主要特点 Strands Agents

Strands Agents包括以下主要功能：

- 模型优先设计 — 围绕基础模型是代理智能核心的概念构建，可实现复杂的自主推理。有关更多信息，请参阅Strands Agents文档中的代理循环。
- 多代理协作模式 — 内置协调模型，例如 Swarm、Graph 和 Workflow 模式，可在分布式代理网络中实现可扩展的协作和治理。有关更多信息，请参阅 Strands Agents 文档中的多代理模式。
- MCP 集成 — 对模型上下文协议 (MCP) 的原生支持，支持标准化上下文配置，以实现一致 LLMs的自主操作。
- AWS 服务 集成 — 与 Amazon Bedrock、 AWS Lambda AWS Step Functions、等无缝连接， AWS 服务 实现全面的自主工作流程。如需了解更多信息，请参阅AWS 每周综述（AWS 博客）。
- 基础模型选择 — 支持各种基础模型，包括 Amazon Bedrock 上的 Anthropic Claude、Amazon Nova（Premier、Pro、Lite 和 Micro）以及其他模型，以针对不同的自主推理功能进行优化。有关更多信息，请参阅Strands Agents文档中的 Amazon Bedrock。
- LLM API 集成 — 灵活集成不同的 LLM 服务接口，包括 Amazon Bedrock、OpenAI 和其他用于生产部署的 LLM 服务接口。有关更多信息，请参阅Strands Agents文档中的 Amazon Bedrock 基本用法。
- 多模态功能 — 支持多种模式，包括文本、语音和图像处理，实现全面的自主代理交互。有关更多信息，请参阅文档中的 Amazon Bedrock Multimodal Supp ort Strands Agents。
- 工具生态系统 — 丰富的 AWS 服务 交互工具，具有扩展自主能力的自定义工具的可扩展性。有关更多信息，请参阅Strands Agents文档中的工具概述。

何时使用 Strands Agents

Strands Agents特别适合自主代理场景，包括：

- 基于 AWS 基础架构构建且希望与之进行原生集成以实现自主工作流程 AWS 服务 的组织
- 需要生产自主系统的企业级安全性、可扩展性和合规性功能的团队
- 需要在不同提供商之间灵活选择模型以完成专门的自主任务的项目
- 需要与现有 AWS 工作流程和资源紧密集成，以实现端到端自主流程的用例

的实施方法 Strands Agents

Strands Agents如其《快速入门指南》所述，为业务利益相关者提供了一种直截了当的实施方法。该框架允许各组织：

- 根据具体的业务需求，在 Amazon Bedrock 上选择基础型号，例如亚马逊 Nova（Premier、Pro、Lite 或 Micro）。
- 定义连接到企业系统和数据源的自定义工具。
- 处理多种模式，包括文本、图像和语音。
- 部署能够自主响应业务查询和执行任务的代理。

这种实现方法使业务团队无需深厚的 AI 模型开发技术专业知识即可快速开发和部署自主代理。

的真实示例 Strands Agents

AWS Transform for .NET 用于Strands Agents增强其应用程序现代化功能，如第一款AWS Transform 用于大规模实现.NET 应用程序现代化的代理 AI 服务（AWS 博客）中所述。该生产服务采用多个专门的自主代理。这些代理协同工作，分析传统的.NET 应用程序，规划现代化策略，并在无需人工干预的情况下执行云原生架构的代码转换。 AWS Transform for .NET 演示了企业自治系统的生产就绪性。Strands Agents

LangChain 和 LangGraph

LangChain是代理人工智能生态系统中最成熟的框架之一。 LangGraph扩展了其功能，以支持LangChain博客中所述的复杂、有状态的代理工作流程。它们共同为构建复杂的自主 AI 代理提供了全面的解决方案，这些代理具有丰富的编排能力，可实现独立操作。

和的主要特LangChain点 LangGraph

LangChain并LangGraph包括以下主要功能：

- 组件生态系统 — 庞大的预建组件库，用于各种自主代理功能，可实现专业代理的快速开发。有关更多信息，请参阅LangChain文档中的快速入门。
- 基础模型选择 — 支持各种基础模型，包括 Anthropic Claude、亚马逊 Bedrock 上的 Amazon Nova 模型（Premier、Pro、Lite 和 Micro），以及其他用于不同推理能力的模型。有关更多信息，请参阅 LangChain文档中的输入和输出。
- LLM API 集成 — 适用于多个大型语言模型 (LLM) 服务提供商（包括 Amazon Bedrock）和其他提供商的标准化接口OpenAI，以实现灵活部署。有关更多信息，请参阅 LangChain文档中的 LLMs。
- 多模态处理 — 内置文本、图像和音频处理支持，可实现丰富的多模态自主代理交互。有关更多信息，请参阅文档中的多模态。LangChain
- 基于图形的工作流程 — LangGraph 支持将复杂的自主代理行为定义为状态机，支持复杂的决策逻辑。有关更多信息，请参阅LangGraph平台正式发布公告。
- 内存抽象 — 短期和长期内存管理的多个选项，这对于随着时间的推移维护上下文的自主代理来说是必不可少的。有关更多信息，请参阅LangChain文档中的如何为聊天机器人添加内存。
- 工具集成 — 丰富的跨各种服务的工具集成生态系统 APIs，并扩展了自主代理功能。有关更多信息，请参阅LangChain文档中的工具。
- LangGraph 平台 — 适用于生产环境的托管部署和监控解决方案，支持长期运行的自主代理。有关更多信息，请参阅LangGraph平台正式发布公告。

何时使用LangChain和 LangGraph

LangChain特别LangGraph适合自主代理场景，包括：

- 复杂的多步骤推理工作流程，需要复杂的编排才能自主决策
- 需要访问由预先构建的组件和集成组成的庞大生态系统以实现各种自动驾驶能力的项目
- 拥有Python基于现有机器学习 (ML) 基础架构和专业知识且想要构建自主系统的团队
- 需要在长时间运行的自治代理会话中进行复杂状态管理的用例

LangChain和的实施方法 LangGraph

LangChain并为业务利益相关者LangGraph提供结构化的实施方法，详见LangGraph文档。该框架使组织能够：

- 定义代表业务流程的复杂工作流程图。
- 使用决策点和条件逻辑创建多步骤推理模式。
- 集成多模态处理功能，用于处理不同的数据类型。
- 通过内置的审查和验证机制实施质量控制。

这种基于图表的方法允许业务团队将复杂的决策过程建模为自主工作流程。团队可以清楚地了解推理过程的每个步骤，并且能够审计决策路径。

LangChain和的真实示例 LangGraph

Vodafone已使用LangChain（和LangGraph）实现了自治代理，以增强其数据工程和运营工作流程，详见其LangChain企业案例研究。他们构建了内部 AI 助手，可以自动监控性能指标，从文档系统中检索信息，并提供切实可行的见解，所有这些都通过自然语言交互完成。

该Vodafone实现使用LangChain模块化文档加载器、向量集成以及对多个 LLMs（OpenAI、LLaMA 3 和Gemini）的支持来快速对这些管道进行原型设计和基准测试。然后，他们过去通过部署模块化子代理LangGraph来构建多代理编排。这些代理执行收集、处理、汇总和推理任务。 LangGraph将这些代理集成 APIs 到他们的云系统中。

CrewAI

CrewAI是一个专门针对自主多代理编排的开源框架，可在上使用。GitHub它提供了一种结构化的方法来创建由专业自主代理组成的团队，这些团队无需人工干预即可协作解决复杂的任务。 CrewAI强调基于角色的协调和任务分配。

的主要特点 CrewAI

CrewAI提供以下主要功能：

- 基于角色的代理设计 — 自主代理通过特定的角色、目标和背景故事进行定义，以实现专业知识。有关更多信息，请参阅CrewAI文档中的制作有效代理。
- 任务委派 — 内置机制，用于根据相应代理的能力自动将任务分配给相应代理。有关更多信息，请参阅CrewAI文档中的任务。
- 代理协作 — 无需人工调解即可实现代理间自主通信和知识共享的框架。有关更多信息，请参阅CrewAI文档中的协作。
- 流程管理 — 用于顺序和并行自主执行任务的结构化工作流程。有关更多信息，请参阅CrewAI文档中的流程。

- 基础模型选择 — 支持各种基础模型，包括 Anthropic Claude、Amazon Bedrock 上的 Amazon Nova 模型（Premier、Pro、Lite 和 Micro）以及其他模型，以针对不同的自主推理任务进行优化。有关更多信息，请参阅 CrewAI文档中的 LLMs。
- LLM API 集成 — 灵活集成多个 LLM 服务接口，包括 Amazon Bedrock 和本地OpenAI模型部署。有关更多信息，请参阅CrewAI文档中的提供程序配置示例。
- 多模态支持 — 用于处理文本、图像和其他模式的新功能，以实现全面的自主代理交互。有关更多信息，请参阅CrewAI文档中的使用多式联运代理。

何时使用 CrewAI

CrewAI特别适合自主代理场景，包括：

- 复杂的问题受益于专业的、基于角色的专业知识自主工作
- 需要在多个自主代理之间进行明确协作的项目
- 基于团队的问题分解可以提高自主解决问题的用例
- 需要在不同的自主代理角色之间明确区分关注点的场景

的实施方法 CrewAI

CrewAI为业务利益相关者提供了基于角色的 AI 代理团队实施方法，如CrewAI文档中的 “入门” 中所述。该框架使组织能够：

- 定义具有特定角色、目标和专业领域的专业自治代理。
- 根据代理的专业能力为其分配任务。
- 在任务之间建立明确的依赖关系，以创建结构化的工作流程。
- 协调多个代理之间的协作以解决复杂的问题。

这种基于角色的方法反映了人类团队的结构，使企业领导者可以直观地理解和实施。Organizations 可以创建具有专业知识领域的自主团队，通过协作实现业务目标，类似于人类团队的运作方式。但是，自主团队可以在没有人为干预的情况下持续工作。

的真实示例 CrewAI

AWS 已使用与 Amazon Bedrock 集成的 Crewai 实现了自主多代理系统，详见CrewAI已发布的案例研究。 AWS 并CrewAI开发了一个安全、供应商中立的框架。CrewAI开源 “flows-and-crews” 架构与 Amazon Bedrock 基础模型、内存系统和合规护栏无缝集成。

实施的关键要素包括：

- 蓝图和开源， AWS 并CrewAI发布了将CrewAI代理映射到 Amazon Bedrock 模型和可观测性工具的参考设计。他们还发布了示例系统，例如多代理 AWS 安全审计团队、代码现代化流程和包装消费品 (CPG) 后台自动化。
- 可观察性堆栈集成 — 该解决方案嵌入了 Amazon 的监控 CloudWatch AgentOpsLangFuse，并实现了从概念验证到生产的可追溯性和调试。
- 已证明的投资回报率 (ROI) — 早期试点显示出重大改进——大型代码现代化项目的执行速度提高了 70%，CPG后台流程的处理时间缩短了约90%。

AutoGen

AutoGen是一个开源框架，最初由Microsoft。 AutoGen专注于实现对话式和协作式自主 AI 代理。它为构建多代理系统提供了一种灵活的架构，重点是代理之间针对复杂的自主工作流程进行异步、事件驱动的交互。

的主要特点 AutoGen

AutoGen提供以下主要功能：

- 对话代理 — 围绕自主代理之间的自然语言对话构建，通过对话实现复杂的推理。有关更多信息，请参阅AutoGen文档中的多代理对话框架。
- 异步架构 — 事件驱动型设计，用于非阻塞自主代理交互，支持复杂的并行工作流程。有关更多信息，请参阅AutoGen文档中的在异步聊天序列中解决多个任务。
- H uman-in-the-loop — 在需要时大力支持可选的人工参与原本自主的代理工作流程。有关更多信息，请参阅AutoGen文档中的允许在代理中进行人工反馈。
- 代码生成和执行 — 专门针对代码的自主代理的功能，可以编写和运行代码。有关更多信息，请参阅AutoGen文档中的代码执行。
- 可自定义的行为 — 灵活的自主代理配置和对话控制，适用于不同的用例。有关更多信息，请参阅文档中的 agentchat.conversable_agent。 AutoGen

- 基础模型选择 — 支持各种基础模型，包括 Anthropic Claude、亚马逊 Bedrock 上的 Amazon Nova 模型（Premier、Pro、Lite 和 Micro），以及其他用于不同自主推理功能的模型。有关更多信息，请参阅AutoGen文档中的 LLM 配置。
- LLM API 集成 — 多个 LLM 服务接口的标准化配置，包括 Amazon Bedrock、OpenAI和。Azure OpenAI有关更多信息，请参阅 API 参考中的 oai.openai_util s。AutoGen
- 多模态处理 — Support 支持文本和图像处理，以实现丰富的多模态自主代理交互。有关更多信息，请参阅AutoGen文档AutoGen中的使用多式联运模型：GPT-4V。

何时使用 AutoGen

AutoGen特别适合自主代理场景，包括：

- 需要在自主代理之间进行自然对话以进行复杂推理的应用程序
- 既需要完全自主操作又需要可选的人工监督能力的项目
- 无需人工干预即可自动生成、执行和调试代码的用例
- 需要灵活的异步自主代理通信模式的场景

的实施方法 AutoGen

AutoGen为业务利益相关者提供了一种对话式实施方法，如AutoGen文档中的入门中所述。该框架使组织能够：

- 创建通过自然语言对话进行通信的自主代理。
- 在多个代理之间实现异步、事件驱动的交互。
- 必要时将完全自主的操作与可选的人工监督相结合。
- 为不同的业务职能部门培养专业代理，通过对话进行协作。

这种对话方法使自治系统的推理变得透明，并且可供业务用户使用。决策者可以观察代理人之间的对话，以了解结论是如何得出的，并且可以选择在需要人类判断时参与对话。

的真实示例 AutoGen

Magentic-One是一个开源、通用的多代理系统，旨在在不同环境中自主解决复杂的多步骤任务，如 AI Frontiers 博客中所述。Microsoft 它的核心是 Orchestrator 代理，它使用结构化账本分解

高级目标并跟踪进度。该代理将子任务委托给专业代理（例如WebSurfer、FileSurferCoder、和ComputerTerminal），并在必要时通过重新规划进行动态调整。

该系统基于AutoGen框架构建，与模型无关，默认为 GPT-4O。它在基准测试中实现了最先进的性能，例如GAIAAssistantBench、和，所有这些都无需针对特定任务进行调整。WebArena此外，它还支持模块化可扩展性，并通过AutoGenBench建议进行严格评估。

LlamaIndex

LlamaIndex是一个数据框架，专门用于将大型语言模型 (LLMs) 与外部数据源连接起来，以实现复杂的检索增强生成 (RAG) 和代理人工智能应用程序。该框架为代理系统、自定义编排模式和系统集成提供了抽象和加速的开发工作流程，而知识驱动的人工智能解决方案则减少 time-to-production了系统集成。

的主要特点 LlamaIndex

LlamaIndex提供了一组全面的功能，使其特别适合企业代理人工智能应用程序：

- 以数据为中心的架构 — 擅长从 100 多种数据格式中提取、索引和检索信息 PDFs，包括 Microsoft Word 文档、电子表格等。该框架将企业数据转换为针对 AI 代理进行了优化的可查询知识库。有关详情，请参阅 LlamaIndex 文档。
- 生产就绪部署 — 通过LlamaIndex提供开源框架和托管服务LlamaCloud，提供企业级功能，包括安全控制、可扩展性、可观察性集成和部署灵活性。有关更多信息，请参阅LlamaIndex框架文档。
- 高级文档处理 — LlamaCloud 提供文档解析、提取、索引和检索功能，可处理复杂的布局、嵌套表格、多模式内容，甚至是手写笔记。这种复杂的解析使代理能够有效地处理包含图表、图表和复杂格式的真实企业文档。有关详情，请参阅 LlamaCloud 文档。
- 工作流程编排 — LlamaAgents 提供事件驱动、异步优先的编排引擎，用于构建多步代理系统。工作流支持复杂模式，包括循环、并行执行、条件分支和状态恢复，因此非常适合复杂的代理交互。有关更多信息，请参阅工作LlamaIndex流程文档。
- 代理检索功能 — 高级检索模式，包括混合搜索、语义搜索和自动路由，可智能地确定每个查询的最佳检索策略。该框架支持跨多个知识库的复合检索，并支持重新排名，以提高准确性。有关更多信息，请参阅 LlamaIndexRAG 文档。
- 可观察性和评估 — 与各种可观测性和评估工具LlamaIndex集成。此集成功能可帮助您跟踪和调试应用程序、评估其性能并监控成本。有关更多信息，请参阅跟踪、调试和评估LlamaIndex文档。

何时使用 LlamaIndex

LlamaIndex特别适合强调数据密集型工作流程和知识管理的代理人工智能场景：

- 文件密集型应用程序，需要代理处理、分析大量企业文档（如合同、报告、手册和监管文件）并从中提取见解
- 为生产场景快速设计原型，在这种场景中，组织希望快速构建和部署以文档为中心的代理，而无需大量的基础架构管理开销
- Rag-first 架构，优先考虑检索准确性和上下文相关性，尤其是在处理包含表格、图像和结构化数据的复杂多模式文档时
- 多代理文档工作流程，需要专门的代理来处理文档处理的不同方面，例如解析、分析、摘要和合规性检查

的实施方法 LlamaIndex

LlamaIndex 提供了适用于不同实现方法的低级构建块和高级抽象：

- 使用LlamaIndex高级 APIs功能，只需几行代码即可快速开发功能性 RAG 应用程序。这种方法为LlamaIndex刚接触代理人工智能的业务团队和开发人员提供了便利。
- 通过企业集成常见 LlamaHub 的企业系统 SharePoint，包括亚马逊简单存储服务 (Amazon S3)、数据库和。 APIs这种方法可以实现与现有数据基础架构的无缝集成。
- 灵活的部署选项包括开源自托管部署以实现最大控制力，或LlamaCloud托管服务以减少运营开销和企业功能。
- 应用程序可以从简单的查询引擎开始，然后随着需求的变化逐渐添加代理功能、多代理编排和复杂的工作流程。

的真实示例 LlamaIndex

此示例侧重于一家专门从事航空导航和运营解决方案的航空航天公司的子公司。他们需要应对日益严峻的挑战，其中包括试行不协调的人工智能聊天机器人试验。这些试验导致整个组织重复工作、漫长的开发周期、合规障碍和孤立的实施。

他们开发了一个统一的代理框架，这是一种基于LlamaIndex开源框架的可重复使用、基于模板的解决方案，可以大大提高代理创建的效率。他们比较了几个竞争框架，包括面向链的框架和基于图形的框架。最终，他们选择LlamaIndex了三个关键优势：灵活的设计、模块化组件和生产就绪的编排控制。

该平台将代理开发和部署时间缩短了 87%，从 512 小时缩短到 64 小时。这种减少是通过让团队能够使用大约 50 行代码和 JSON 配置文件来构建代理来实现的。这些团队利用了具有内置安全性、合规性和特权系统访问权限的统一框架。有关更多详细信息，请参阅LlamaIndex客户案例研究。

比较代理 AI 框架

在选择用于自主代理开发的代理 AI 框架时，请考虑每个选项如何与您的特定要求保持一致。不仅要考虑其技术能力，还要考虑其组织适应性，包括团队专业知识、现有基础设施和长期维护要求。许多组织可能会从混合方法中受益，将多个框架用于其自主人工智能生态系统的不同组件。

下表比较了每个框架在关键技术方面的成熟度级别（最强、强、足或弱）。对于每个框架，该表还包括有关生产部署选项和学习曲线复杂性的信息。

Framework	AWS 整合	自主多代理支持	自主工作流程的复杂性	多式联运能力	基础模型选择	法学硕士 API 集成	生产部署	学习曲线
AutoGen	弱	很强	很强	足够的	足够的	很强	自己动手（DIY）	陡峭
CrewAI	弱	很强	足够的	弱	足够的	足够的	自己动手	中
LangChain / LangGraph	足够的	很强	最强	最强	最强	最强	平台或 DIY	陡峭
LlamaIndex	足够的	足够的	很强	足够的	很强	很强	平台或 DIY	中
Strands Agents	最强	很强	最强	很强	很强	最强	自己动手	中

选择代理人工智能框架时的注意事项

开发自主代理时，请考虑以下关键因素：

- AWS 基础设施集成 — 投入大量资金的 Organizations AWS 将从自主工作流程的Strands Agents本机集成中受益最大。 AWS 服务 如需了解更多信息，请参阅AWS 每周综述（AWS 博客）。
- 基础模型选择 — 根据您的自主代理的推理要求，考虑哪个框架可以为您的首选基础模型（例如 Amazon Bedrock 或 C Anthropic laude 上的 Amazon Nova 模型）提供最佳支持。有关更多信息，请参阅Anthropic网站上的构建有效代理。
- LLM API 集成 — 根据框架与用于生产部署的首选大型语言模型 (LLM) 服务接口（例如 Amazon Bedrock 或OpenAI）的集成来评估框架。有关更多信息，请参阅 Strands Agents文档中的模型接口。
- 多模态要求 — 对于需要处理文本、图像和语音的自主代理，请考虑每个框架的多模态功能。有关更多信息，请参阅文档中的多模态。LangChain
- 自治工作流程的复杂性 — 具有复杂状态管理的更复杂的自主工作流程可能会偏爱高级状态机功能。of. LangGraph
- 自主团队协作 — 需要专业代理之间基于角色的明确自主协作的项目可以受益于面向团队的架构。CrewAI
- 自主开发范式 — 偏爱对话式异步模式的自主代理的团队可能更喜欢事件驱动的架构。AutoGen
- 托管或基于代码的方法 — 想要以最少的编码获得完全托管体验的组织应考虑使用 Amazon Bedrock Agents。需要更深层次定制的组织可能更喜欢Strands Agents其他具有专门功能的框架，这些框架可以更好地满足特定的自治代理要求。
- 自主系统的生产就绪性 — 考虑生产自主代理的部署选项、监控功能和企业功能。

平台

Agentic AI 平台提供了部署、扩展和管理生产级代理系统所需的基础运行时、编排和集成层。框架定义代理的构建方式以及协议控制代理的通信方式。平台为这些代理提供了安全地大规模运营、协作和发展的环境。

Agentic 平台将模型执行、上下文管理、工具集成、可观察性和治理功能整合到统一的环境中。这些平台使组织能够从实验转向企业级部署。

在本节中：

平台为何重要

Agentic AI 平台对于寻求在生产中实现自主系统的组织至关重要。它们提供以下功能：

- 为托管、扩展和协调代理提供运行时编排。
- 跨多代理工作流程管理状态、上下文和内存。
- 提供符合企业标准的安全、身份和治理控制。
- 通过标准 APIs 或协议与工具生态系统和外部系统集成。
- 实现跨代理交互和事件流的可观察性和可审计性。
- Support 支持跨模型互操作性，允许代理在单个环境中使用多个基础模型。

这些功能将单个代理转变为协调的自适应系统，可以在企业和监管范围内可靠运行。

代理人工智能平台的类型

Agentic AI 平台通常属于以下一个或多个类别：

- 托管代理 — 完全托管的平台提供内置的基础架构、内存和编排功能。它们减少了运营开销并加快了生产时间。
- 开源编排 — 开源代理平台为喜欢可自定义环境或本地部署的组织提供了灵活性和透明度。
- 混合企业 — 混合平台集成了托管和自托管组件，将云托管服务的可扩展性与企业系统的控制相结合。

平台选择注意事项

在选择或设计代理人工智能平台时，组织应考虑以下几点：

- 集成深度-评估平台与现有数据源、工具和协议的集成程度。
- 可扩展性 — 确保平台可以动态扩展，以支持自主工作负载和多代理协作。
- 安全性与合规性 — 根据组织和区域要求评估数据隐私、加密和治理功能。
- 可扩展性-选择具有模块化架构的平台，允许随着时间的推移添加新的工具、模型或代理。
- 可观察性 — 首选为代理互动提供详细遥测、可追溯性和审计日志的平台。
- 成本效益 — 考虑使用无服务器或基于使用量的模型，以优化可变工作负载的成本。

Amazon 基岩代理商

Amazon Bedrock Agents 是一项完全托管的服务，使您能够在应用程序中构建和配置自主代理。它可以协调基础模型、数据源、软件应用程序和用户对话之间的交互。其创建代理的简化方法不需要您配置容量、管理基础架构或编写自定义代码。

Amazon Bedrock Agents 的主要功能

Amazon Bedrock Agents 包括以下主要功能：

- 完全托管的服务 — 无需配置容量或管理底层系统，即可完成基础架构管理。有关更多信息，请参阅 Amazon Bedrock 文档中的使用 AI 代理在应用程序中自动执行任务。
- API 驱动的开发 — 通过指定模型、指令、工具和配置参数，通过简单的 API 调用定义和运行代理。有关更多信息，请参阅 Amazon Bedrock 文档中的手动创建和配置代理。
- 操作组-通过使用 API 架构创建操作组，定义您的代理可以执行的特定操作。有关更多信息，请参阅 Amazon Bedrock 文档中的使用操作组定义代理要执行的操作。
- 知识库集成 — 无缝连接到 Amazon Bedrock 知识库，使用贵组织的数据增强代理响应能力。有关更多信息，请参阅 Amazon Bedrock 文档中的利用知识库为代理生成响应。

- 高级提示模板 — 通过用于预处理、编排、知识库响应生成和后处理的提示模板自定义代理行为。有关更多信息，请参阅 Amazon Bedrock 文档中的 Amazon Bedrock 中的使用高级提示模板提高代理的准确性。
- 跟踪和可观察性-使用内置的跟踪功能跟踪代理的 step-by-step推理过程。有关更多信息，请参阅 Amazon Bedrock 文档中的使用跟踪跟踪代理的 step-by-step推理过程。
- 版本控制和别名-创建代理的多个版本，并通过别名对其进行部署，以实现受控部署。有关更多信息，请参阅 Amazon Bedrock 文档中的在应用程序中部署和使用 Amazon Bedrock 代理。

何时使用 Amazon Bedrock Agents

Amazon Bedrock Agents 特别适合自主代理场景，包括：

- 想要在不管理基础架构的情况下构建和部署代理时获得完全托管体验的组织
- 需要通过配置而不是代码快速开发和部署代理的项目
- 可从与知识库和护栏等其他 Amazon Bedrock 功能的紧密集成中受益的用例
- 没有内部资源从头开始构建代理但需要生产就绪的自主能力的团队

Amazon Bedrock Agents 的实施方法

Amazon Bedrock Agents 为业务利益相关者提供了一种基于配置的实施方法。该服务使组织能够：

- 无需编写复杂代码，即可通过 AWS 管理控制台 或 API 调用定义代理。
- 创建操作组，指定代理可以执行的 APIs 和操作。
- Connect 知识库以向代理提供特定于域的信息。
- 通过可视化界面测试和迭代代理行为。

这种托管方法允许业务团队快速开发和部署自主代理，而无需深厚的 AI 模型开发或基础设施管理方面的技术专业知识。

亚马逊 Bedrock Agents 的真实示例

本AWS 博客文章中描述的财务运营 (FinOps) 解决方案使用 Amazon Bedrock 多代理框架来创建人工智能驱动的云成本管理助手。具有成本效益的 Amazon Nova 基础模型为中央 FinOps 主管代理将任务委托给专业代理的解决方案提供支持。这些代理通过使用来获取和分析 AWS 支出数据 AWS Cost Explorer ，并使用生成节省成本的建议。 AWS Trusted Advisor

该系统包括用户通过 Amazon Cognito（托管在上的前端）进行安全访问 AWS Amplify，以及用于实时分析和预测的 AWS Lambda 操作组。财务团队可以询问自然语言问题，例如 “2025 年 2 月我的费用是多少？” 系统会以详细的故障、优化建议和预测作为响应，所有这些都是在使用部署的可扩展的无服务器架构中进行的。 AWS CloudFormation

Amazon Bedrock AgentCore

Amazon Bedrock AgentCore 是一个代理平台，可使用任何框架、模型或协议大规模安全地构建、部署和运行功能强大的代理。使用 AgentCore，您可以执行以下操作，而无需任何基础架构管理：

- 更快地构建代理。
- 使代理能够跨工具和数据采取行动。
- 借助低延迟和更长的运行时间，安全地运行代理。
- 监控生产中的代理。

AgentCore 消除了构建专业代理基础设施的无差别繁重的工作，使您可以加快代理的生产速度。它的服务可以一起使用，也可以单独使用，并且与任何框架兼容CrewAI，包括LangGraph、LlamaIndex、和Strands Agents。 AgentCore还与 Amazon Bedrock 内部或外部提供的任何基础型号兼容，提供了极大的灵活性。

AgentCore 由几个关键服务组成：

- Amazon Bedrock AgentCore Runtime — 提供安全、无服务器、可扩展的环境来托管和运行代理，无需管理部署和运行 AI 代理或工具所需的任何基础设施。
- Amazon Bedrock AgentCore Memory — 提供托管内存系统，使代理能够通过保持即时和长期的知识来保留互动中的背景信息，从而进行更加个性化和连贯的对话。
- Amazon Bedrock AgentCore Gateway — 简化了为代理创建、保护和寻找合适工具的过程。借助 AgentCore Gateway APIs，开发人员可以将 Lambda 函数和现有服务转换为与模型上下文协议 (MCP) 兼容的工具，并将其提供给代理。
- Amazon Bedrock Ident AgentCore it y — 提供安全、可扩展的代理身份和访问管理服务，可加快 AI 代理的开发。借助 AgentCore Identity，您可以为代理分配唯一的、可验证的身份，从而实现精细的访问控制，并保护代理支持的与企业系统的交互。
- Amazon Bedrock AgentCore 内置工具 — 使您能够使用内置工具来增强您的开发和测试工作流程。使用这些工具与您的应用程序进行有效的交互，使 AI 代理能够在沙盒环境中安全地编写和执行代码。使用浏览器工具让 AI 代理能够与网站进行大规模交互。

- Amazon Bedrock 可 AgentCore 观察性 — 提供日志和监控功能，让您实时了解代理的性能和行为，从而便于调试和优化。

的主要特点 AgentCore

AgentCore 包括以下主要功能：

- 完全托管且可扩展 — AgentCore 是一项完全托管的服务，这意味着它 AWS 可以处理底层基础设施和维护。它还具有可扩展性，允许您自定义和增强代理的功能。有关更多信息，请参阅 AgentCore 文档中的 AgentCoreRuntime 入门。
- 长期和短期记忆 — 通过为代理配备记忆系统来回忆当前对话和长期知识中的背景，从而提供更加个性化和相关的互动。有关更多信息，请参阅 AgentCore 文档中的 AgentCore 内存入门。
- 简化了工具开发和集成-使您的代理能够通过单个安全的端点发现和使用工具。只需几行代码，即可将现有的企业资源快速转化为代理就绪工具，让开发人员腾出时间专注于构建独特的功能。有关更多信息，请参阅 AgentCore 文档中的 AgentCore Gateway 入门。
- 安全且可扩展的基础架构 — AgentCore 为部署和操作代理提供安全且可扩展的环境。它包括身份和访问管理、数据加密和网络安全的功能。有关更多信息，请参阅 AgentCore 文档中的 AgentCore身份入门。
- 与各种工具集成-允许您将代理与各种工具集成，包括代码解释器和浏览器工具，您可以使用 AgentCore 内置工具进行构建。有关更多信息，请参阅 AgentCore 文档中的 AgentCore 代码解释器入门和 AgentCore浏览器入门。
- 全面的可观察性和监控 — 使用全面的工具来跟踪、调试和监控代理在生产中的性能，深入了解代理。可视化代理的整个执行路径，以审计其推理并解决故障。使用实时仪表板和标准化的遥测数据来跟踪关键运营指标。有关更多信息，请参阅文档中的为您的 Amazon Bedrock AgentCore 资源添加可观察性。 AgentCore

何时使用 AgentCore

AgentCore 特别适合自主代理场景，包括：

- 希望通过处理基础架构、安全、内置工具、可观察性和扩展性的完全托管服务来加快开发并降低运营开销的组织
- 需要灵活性的项目，这些服务可以协同工作或独立运行，并且与任何框架（如CrewAI或LangGraph）以及来自任何来源的任何基础模型兼容

- 需要有状态的对话代理的用例，这些代理需要维护上下文并从过去的互动中吸取教训，以提供个性化和相关的响应
- 通过与各种应用程序、数据源和数据源的简单集成，代理可以执行复杂的任务 APIs

的实施方法 AgentCore

AgentCore 专为希望将 AI 代理从使用开源或自定义代理框架构建的概念验证转移到生产环境的组织而设计。借 AgentCore助，组织可以执行以下操作：

- 在无服务器基础架构上安全地部署代理，支持任何框架和模型，并通过会话隔离以及内置的身份和访问管理来确保 end-to-end安全性和合规性。使用入门工具包为领先的代理框架快速创建 AgentCore 运行时代理。
- 通过集成用于上下文保留的持久内存来增强代理，简化工具开发和通过 AgentCore Gateway 集成。利用内置的浏览器工具和代码解释器实现高级工作流程。
- 使用由 Amazon App CloudWatch lication Insights 提供支持的可观察性仪表板跟踪OpenTelemetry、调试和监控生产中的 AI 代理，并跟踪 AgentCore 资源的关键指标（运行时间、内存、网关和工具）。
- 借助完全托管的模块化服务、组合模块化模块组合或独立组合，以及任何代理框架和模型提供商，加快部署和创新。这种灵活性可以帮助组织更快地从原型转向生产。

这种托管方法使组织能够快速、安全地构建、部署和运行任何规模的企业级 AI 代理和多代理系统。

的真实示例 AgentCore

AWS 观察到，拉丁美洲最大的银行之一多年来 AI/ML 一直提供高度个性化和安全的数字银行体验。该银行正在扩展代理人工智能服务，使用它 AgentCore 为客户提供直观的交互、增强的安全性和更高的自动化程度。首席技术官表示， AgentCore 预计将支持他们大规模履行客户承诺的努力。 AgentCore 为开发人员提供构建和管理代理的工具和灵活性，同时帮助确保遵守财务法规。

协议

AI 代理需要标准化的通信协议才能与其他代理和服务进行交互。实施代理架构的组织在互操作性、供应商独立性和投资面向未来方面面临重大挑战。

本节将帮助您浏览 agent-to-agent协议格局，重点介绍可最大限度地提高灵活性和互操作性的开放标准。（有关 agent-to-tool协议的信息，请参阅本指南后面的工具集成策略。）

本节重点介绍模型上下文协议 (MCP)，这是一项最初于 2024 年开发Anthropic的开放标准。如今，通过为协议的制定和实施做出贡献， AWS 积极支持MCP。 AWS 正在与领先的开源代理框架（包括LangGraphCrewAI、和）合作LlamaIndex，以塑造该协议上代理间通信的未来。有关更多信息，请参阅代理互操作性开放协议第 1 部分：MCP 上的代理间通信（AWS 博客）。

在本节中：

- 为什么协议选择很重要
- Agent-to-agent 协议
- 选择代理协议
- 代理协议的实施策略
- MCP 入门
- ???

为什么协议选择很重要

协议选择从根本上决定了如何构建和发展 AI 代理架构。通过选择支持代理框架之间可移植性的协议，您可以灵活地组合不同的代理系统和工作流程来满足您的特定需求。

开放协议使您能够跨多个框架集成代理。例如，LangChain用于快速原型设计和实现生产系统Strands Agents，通过通用协议（例如 MCP 或 Agent2Agent (A2A) 协议）进行通信。这种灵活性减少了对特定 AI 提供商的依赖，简化了与现有系统的集成，并使您能够随着时间的推移增强代理功能。

精心设计的协议还可以为整个代理生态系统中的身份验证和授权建立一致的安全模式。最重要的是，协议的可移植性可以让您在出现新的代理框架和功能时自由地采用它们。选择开放协议可以保护您在代理开发方面的投资，同时保持与第三方系统的互操作性。

开放协议的优势

在实现自己的扩展或构建自定义代理系统时，开放协议具有引人注目的优势：

- 文档和透明度 — 通常提供全面的文档和透明的实施
- 社区支持 — 访问更广泛的开发者社区，了解疑难解答和最佳实践
- 互操作性保证 — 更好地保证您的扩展能够在不同的实现中运行
- 未来的兼容性 — 降低了重大更改或弃用的风险
- 对@@ 开发的影响 — 为协议演变做出贡献的机会

Agent-to-agent 协议

下表概述了支持多个代理协作、委派任务和共享信息的代理协议。

协议	非常适合	注意事项
MCP 代理间通信	寻求灵活代理协作模式的组织	• 该协议提议的模型上下文协议（MCP）AWS 的扩展建立在其现有的通信基础之agent-to-agent上 • 通过 OAuth基于安全性的无缝代理协作
A2A 协议	跨平台代理生态系统	• 由 Google • 与 MCP 相比，较新的标准采用更为有限

在协议选项之间做出决定

在实现 agent-to-agent通信时，请将您的特定通信要求与相应的协议功能相匹配。不同的交互模式需要不同的协议功能。下表概述了常见的通信模式，并针对每种情况推荐了最合适的协议选择。

模式	描述	理想的协议选择
简单的请求和响应	代理之间的一次性互动	具有无状态流的 MCP
有状态的对话	正在进行的有情境的对话	带会话管理功能的 MCP
多代理协作	多个代理之间的复杂交互	MCP 代理间或 AutoGen

基于团队的工作流程	具有已定义角色的分层代理团队	MCP 间代理，或 CrewAI AutoGen

除了沟通模式之外，还有几个技术和组织因素会影响您的协议选择。下表概述了可以帮助您评估哪种协议最符合您的具体实施要求的关键注意事项。

考虑	描述	示例
安全模型	身份验证和授权要求	OAuth MCP 中的 2.0
部署环境	代理将在哪里运行和通信	分布式或单机
生态系统兼容性	与现有代理框架集成	LangChain 或 Strands Agents
可扩展性需求	代理人相互作用的预期增长	MCP 的直播功能

选择代理协议

对于大多数构建生产代理系统的组织来说，模型上下文协议 (MCP) 为通信提供了最全面、最有支持的基础 agent-to-agent。MCP 受益于开源社区 AWS 的积极开发贡献。

对于希望有效实施代理人工智能的组织来说，选择正确的代理协议非常重要。考虑因素因组织环境而异。

代理协议选择注意事项

在为代理人工智能系统选择协议时，组织应考虑以下最佳实践：

- 优先考虑开放标准 — Organizations 应采用 MCP 等开放协议，以帮助确保长期的互操作性和可扩展性，并降低供应商锁定的风险。
- 平衡速度和灵活性 — 初创企业和早期采用者可能从支持良好的专有协议开始，以实现快速开发，但随着系统的成熟，他们应该定义向开放标准的迁移路径。
- 实施抽象层 — 企业应实施协议抽象以简化迁移、实现混合采用和面向未来的集成策略。
- 强调安全性和合规性 — 受监管行业的组织应选择具有强大身份验证、加密和审计功能的协议，以满足治理和合规性要求。
- 评估生态系统成熟度 — 所有组织都应评估每项协议的健康状况、采用率和社区支持，以确保可持续性并最大限度地减少技术债务。

- 参与标准开发 — Organizations 应参与标准机构或开源社区，以帮助塑造协议的演变并影响最佳实践。
- 考虑数据主权 ——政府和受监管部门应确保协议选择符合部署区域的数据驻留和主权要求。
- 利用托管服务 — 在可能的情况下，使用代理协议的托管或无服务器实施来降低运营复杂性并加快部署。

代理协议的实施策略

要在整个组织中有效地实施代理协议，请考虑以下战略步骤：

1. 从@@ 标准协调开始 — 尽可能采用既定的开放协议。
2. 创建抽象层-在您的系统和特定协议之间实现适配器。
3. 为开放标准做出贡献 — 参与协议开发社区。
4. 监控协议演变 — 随时了解新兴标准和更新。
5. 定期测试互操作性 — 验证您的实现是否保持兼容。

MCP 入门

AWS 通过为示范上下文协议 (MCP) 的制定和实施做出贡献，积极支持该协议。 AWS 正在与领先的开源代理框架（包括LangGraphCrewAI、和）合作LlamaIndex，以塑造该协议上代理间通信的未来。

要在代理架构中实现 MCP，请执行以下操作：

1. 探索 Strands AgentsSDK 等框架中的 MCP 实现。
2. 查看模型上下文协议技术文档。
3. 阅读 MCP 上的代理互操作性开放协议第 1 部分：代理间通信（AWS 博客），了解代理互操作性。
4. 加入 MCP 社区，影响协议的演变。

MCP 提供了一个通信层，使代理能够与外部数据和服务进行交互，也可以用于使代理能够与其他代理进行交互。该协议的 Streamable HTTP 传输实现为开发人员提供了一套全面的交互模式，而无需重新设计轮子。这些模式既支持无状态 request/response 流，也支持具有持久 IDs性的有状态会话管理。

通过采用 MCP 等开放协议，您可以让您的组织构建能够随着 AI 技术发展而保持灵活、互操作和适应能力强的代理系统。有关 agent-to-tool协议实现的信息，请参阅本指南后面的工具集成策略。

A2A 入门

Agent2Agent (A2A) 协议通过共享的语义层实现代理之间的去中心化协作。A2A 允许代理使用基于 JSON 的轻量级协议来发现对方、宣传自己的能力、协商任务和共享上下文，而不是通过中央协调器路由所有工作。每个代理都会发布一份能力清单。

以下示例显示了简化的 A2A 功能清单，该清单公布了代理支持的操作、所需的输入和操作元数据，以支持发现和任务协商：

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

该模型支持动态能力匹配、任务中期委派和跨组织协作。代理可以围绕任务进行自我组织，组建临时工作组，并在新功能进入或退出系统时进行调整。

A2A 支持各种交互，从简单的无状态请求到多步协商会话，包括：

- 用于低延迟 peer-to-peer协作的直接消息传递
- 语义任务协商，代理选择最合适的对等方
- 基于能力的发现，实现紧急分工
- 用于有状态的多步交互的会话锚定

通过采用 A2A 等开放的代理原生协议，组织可以创建模块化、可互操作且能够跨境协作的人工智能系统。A2A 可确保代理生态系统保持灵活性，并且可以随着新的代理、团队或外部系统的引入而发展，而无需严格的编排层或事先耦合。

要在代理架构中实现 A2A 协议，请执行以下操作：

1. 查看 A2A 协议规范 — 阅读最新版本的 A gent2Agent (A2A) 协议规范，了解能力表现、协商流程和代理握手的工作原理。
2. 探索兼容 A2A 的运行时 — 评估支持 A2A 风格的功能清单和协商的 Strands Agents SDK 等框架或自定义运行时层。 peer-to-peer
3. 为您的代理实施能力清单 — 定义每个代理的`can`、和`meta`字段`needs`，以实现发现、配对和意图级协作。

4. 试验 A2A 协商模式 — 使用请求—报价—接受循环、结构化能力查询或基于八卦的发现来了解代理如何推断谁应该处理任务。
5. 在@@ 混合基础设施环境中测试 A2A — 将 A2A 对等协商与通过 A AWS mazon 原生的事件路由相结合， EventBridge 以评估混合协调模式。
6. 加入 A2A 社区 — 加入开放工作组，及时了解扩展、安全建议和跨供应商互操作性改进的最新信息，并为协议的开发做出贡献。

工具

AI 代理通过与外部工具和数据源进行交互来执行有用的任务来创造价值。 APIs正确的工具集成策略会直接影响代理的能力、安全状况和长期灵活性。

本节将帮助您浏览工具集成格局，重点介绍可最大限度地提高自由度和灵活性的开放标准。本节重点介绍用于工具集成的模型上下文协议 (MCP)，并回顾了增强代理工作流程的特定框架工具和专门的元工具。

在本节中：

工具类别

建筑代理系统涉及三大类工具。

基于协议的工具

基于协议的工具使用标准化协议进行 agent-to-tool通信：

- MCP 工具 — 开放标准工具，可跨框架使用，具有本地和远程执行选项。
- OpenAI函数调用-特定于OpenAI模型的专有工具。
- Anthropict@@ ools — 一种OpenAI函数变体，需要特定于 Anthropic Claude 模型的专有工具。

框架原生工具

Framework-Native 工具直接内置在特定的代理框架中：

- Strands Agents t@@ ool s-轻量级，特定于Strands Agents框架的 quick-to-implement工具。

- LangChain工具 — Python 基于与LangChain生态系统紧密集成的工具。
- LlamaIndex工具-针对内部数据检索和处理进行了优化的工具LlamaIndex。

元工具

元工具无需直接采取外部操作即可增强代理工作流程：

- 工作流程工具-管理代理执行流程、分支逻辑和状态管理。
- 代理图工具-在复杂的工作流程中协调多个代理。
- 存储工具-提供跨代理会话的持久存储和信息检索。
- 反射工具-使代理能够分析和改善自己的绩效。

基于协议的工具

在考虑基于协议的工具时，模型上下文协议 (MCP) 为工具集成提供了最全面、最灵活的基础。正如关于代理互操作性的AWS 开源博客文章中所述， AWS 已将 MCP 视为一种战略协议，为其发展做出了积极贡献。

下表描述了 MCP 工具部署选项。

部署模型	描述	非常适合	实施
基于本地工作室	工具的运行过程与代理相同	开发、测试和简单工具	实施速度快，没有网络开销
基于本地服务器发送的事件 (SSE)	工具在本地运行，但通过 HTTP 进行通信	更复杂的本地工具，可以分开关注点	隔离效果更好，但延迟仍然很低
远程 HTTP 可串流	工具在远程服务器上运行	生产环境和共享工具	可扩展和集中管理

官方 MCP SDKs 可用于构建 MCP 工具：

- PythonSDK — 全面实现，提供全面的协议支持
- TypeScriptSDK —JavaScript/Web 应用程序的TypeScript实现
- Java软件开发工具包 — 适用于企业应用程序的 Java 实现

它们 SDKs 为使用您的首选语言创建兼容 MCP 的工具提供了构建模块，同时实现了协议规范的一致性。

此外， AWS 还在 Strands AgentsSDK 中实现了 MCP。S Strands Agents DK 提供了一种创建和使用 MCP 兼容工具的简单方法。Strands Agents GitHub 存储库中提供了全面的文档。对于更简单的用例或在Strands Agents框架之外工作时，官方 MCP SDKs 提供多种语言的协议直接实现。

MCP 工具的安全功能

MCP 工具的安全功能包括以下内容：

- OAuth 2.0/2.1 身份验证 — 行业标准身份验证
- 权限范围界定-工具的精细访问控制
- 工具功能发现 — 动态发现可用工具
- 结构化错误处理 — 一致的错误模式

MCP 工具入门

要为工具集成实现 MCP，请执行以下操作：

1. 探索 S Strands AgentsDK 以实现可用于生产的 MCP。
2. 查看 MCP 技术文档以了解核心概念。
3. 使用这篇AWS 开源博客文章中描述的实际示例。
4. 先从简单的本地工具开始，然后再使用远程工具。
5. 加入 MCP 社区，影响协议的演变。

探索 AgentCore 网关

Amazon Bedrock AgentCore Gatew ay 为开发人员提供了一种简单而安全的方式，让他们能够大规模构建、部署、发现和连接 MCP 工具和其他目标终端节点。借助 AgentCore Gateway，开发人员可以将 APIs、 AWS Lambda 函数和现有服务转换为兼容 MCP 的工具。然后，只需几行代码，他们就可以通过 AgentCore 网关端点向代理提供这些工具。 AgentCore Gateway 支持OpenAPISmithy、和 Lambda 作为输入类型，并且是唯一一款在完全托管的服务中同时提供全面入口身份验证和出口身份验证的解决方案。

框架原生工具

尽管模型上下文协议 (MCP) 提供了最灵活的基础，但框架原生工具为特定用例提供了优势。

Strands AgentsSDK 提供的Python基于工具的特点是其轻量级设计，只需极少的开销即可完成简单的操作。它们可以实现快速实现，并允许开发人员只需几行代码即可创建工具。此外，它们紧密集成，可在Strands Agents框架内无缝运行。

以下示例演示如何使用创建简单的天气工具Strands Agents。开发人员可以用最少的代码开销快速将Python函数转换为代理可访问的工具，并从函数的文档字符串中自动生成相应的文档。

```
#Example of a simple Strands native tool

@tool

def weather(location: str) -> str:

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

Implementation here

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

对于快速原型设计或简单的用例，框架原生工具可以加快开发速度。但是，对于生产系统，与框架原生工具相比，MCP 工具提供了更好的互操作性和未来的灵活性。

下表概述了其他特定于框架的工具。

Framework	工具类型	优点	注意事项
AutoGen	函数定义	强大的多代理支持	Microsoft生态系统
LangChain	Python课堂	由预建工具组成的庞大生态系统	框架锁定
LlamaIndex	Python 函数	针对数据操作进行了优化	仅限于 LlamaIndex

元工具

元工具不直接与外部系统交互。相反，它们通过实现代理模式来增强代理能力。本节讨论工作流程、代理图和内存元工具。

工作流程元工具

工作流元工具管理代理执行流程：

- 状态管理-跨多个代理交互维护上下文
- 分支逻辑-启用条件执行路径
- 重试机制 — 使用复杂的重试策略处理失败

带有工作流程元工具的示例框架包括LangGraph和Strands Agents工作流程功能。

代理图元工具

代理图元工具可协调多个代理协同工作：

- 任务委托-将子任务分配给专业代理
- 结果聚合-合并来自多个代理的输出
- 冲突解决-解决代理人之间的分歧

像AutoGen和CrewAI专门研究代理图协调的框架一样。

内存元工具

内存元工具提供持久存储和检索：

- 对话历史记录-跨会话保持背景信息
- 知识库-存储和检索特定于域的信息
- 矢量存储-启用语义搜索功能

MCP 的资源系统提供了一种标准化的方法来实现跨不同代理框架的内存元工具。

工具集成策略

您对工具集成策略的选择直接影响您的代理可以完成的任务以及系统的发展难易程度。优先考虑模型上下文协议 (MCP) 等开放协议，同时策略性地使用框架原生工具和元工具。这样，您就可以构建一个随着 AI 技术的进步而保持灵活和强大的工具生态系统。

以下工具集成的战略方法可最大限度地提高灵活性，同时满足组织的即时需求：

1. 采用 MCP 作为基础 — MCP 提供了一种标准化的方式，可将代理连接到具有强大安全功能的工具。首先将 MCP 作为您的主要工具协议，用于：
   - 将在多个代理实施中使用的战略工具。
   - 需要强大身份验证和授权的安全敏感型工具。
   - 需要在生产环境中远程执行的工具。
2. 在@@ 适当时使用框架原生工具 — 考虑使用框架原生工具：
   - 在初始开发期间快速制作原型。
   - 简单的非关键工具，安全要求最低。
   - 利用独特功能的特定于框架的功能。
3. 为复杂的工作流程实施元工具 — 添加元工具以增强您的代理架构：
   - 从基本的工作流程模式开始，从简单开始。
   - 随着用例的成熟，会增加复杂性。
   - 标准化代理和元工具之间的接口。
4. 规划演变 — 在构建时考虑未来的灵活性：
   - 独立于实现的文档工具接口。
   - 在代理和工具之间创建抽象层。
   - 建立从专有协议到开放协议的迁移路径。

工具集成的安全最佳实践

工具集成会直接影响您的安全状况。本节概述了贵组织需要考虑的最佳实践。

身份验证和授权

使用以下强大的访问控制：

- 使用 OAuth 2.0/2.1 — 对远程工具实施行业标准身份验证。

- 实现最低权限-仅授予工具所需的权限。
- 轮换凭证-定期更新 API 密钥和访问令牌。

数据保护

为帮助保护数据，请采取以下措施：

- 验证输入和输出-对所有工具交互实施架构验证。
- 加密敏感数据-使用 TLS 进行所有远程工具通信。
- 实现数据最小化-仅将必要的信息传递给工具。

监控和审计

使用以下机制保持可见性和控制力：

- 记录所有工具调用-维护全面的审计跟踪。
- 监控异常情况-检测异常的工具使用模式。
- 实施速率限制 — 防止因过度调用工具而造成滥用。

模型上下文协议 (MCP) 安全模型全面解决了这些问题。有关更多信息，请参阅 MCP 文档中的安全注意事项。

结论

代理人工智能的格局继续快速发展，为组织提供了构建智能、自主系统的强大新方法。本指南探讨了成功实施的三个基本组成部分：提供基础的框架、提供环境的平台、支持通信的协议以及扩展功能的工具。

随着框架的成熟，可以预期互操作性将得到提高，模型上下文协议 (MCP) 等协议实现标准化，自治代理的编排功能也将更加复杂。如今，在这些框架中积累专业知识的组织将完全有能力构建越来越自主、智能的代理，从而带来可观的商业价值。

平台提供了代理系统运行的执行、治理和生命周期环境。他们处理身份、安全边界、可观察性、内存管理、会话接地以及与工具和数据的安全交互等问题。在 AWS 环境中，托管代理运行时和编排服务等平台允许组织大规模部署、监控、发展和管理自治代理和代理系统。平台将基础框架与现实世界的运营需求联系起来。

代理协议的选择代表着一项战略决策，它在眼前的开发需求与长期的灵活性和互操作性之间取得平衡。通过优先考虑开放协议并创建适当的抽象层，组织可以构建能够适应不断发展的技术的代理系统，同时满足当前的业务需求。

对于大多数组织来说，由于其开放的标准、不断增长的生态系统、对 agent-to-agent通信模式的支持以及工具集成能力，MCP是一个坚实的基础。 AWS 已将 MCP 和 Agent2Agent (A2A) 视为战略协议，积极为它们的开发做出贡献，并在 SDK 等服务中实施它们。Strands Agents通过将 MCP 或 A2A 与适当的框架原生工具和元工具一起使用，您可以构建既能提供即时价值，又能适应未来创新的代理系统。

资源

使用以下资源 AWS 和其他与自治代理开发相关的资源。

AWS 博客

- Amazon Bedrock AgentCore Memory：构建情境感知代理
- 使用 Amazon Bedrock 代理构建强大的生成式人工智能应用程序的最佳实践 – 第 1 部分
- 使用 Amazon Bedrock 代理构建强大的生成式人工智能应用程序的最佳实践 – 第 2 部分
- 使用LlamaIndex和 Amazon Bedrock 构建强大的 RAG 管道
- 使用 Amazon Bedrock AgentCore 可观测性构建值得信赖的 AI 代理
- 使用 Amazon Bedrock 和 RAGAS 评估 RAG 的回应 LlamaIndex
- Amazon Bedrock AgentCore 代码解释器简介
- Amazon Bedrock AgentCore Gateway 简介：转变企业 AI 代理工具的开发
- Amazon Bedrock AgentCore Identity 简介：大规模保护代理人工智能
- 开源 AI 代理 SDK 简Strands Agents介
- 代理互操作性开放协议第 1 部分：MCP 上的代理间通信
- 在 Amazon Bedrock AgentCore Runtime 上安全启动和扩展您的代理和工具
- AWS Transform 适用于.NET，这是第一款用于大规模实现.NET 应用程序现代化的代理 AI 服务
- AWS 每周综述：Strands Agents

AWS 规范性指导

- 在上操作代理 AI AWS
- 代理人工智能的基础 AWS
- Agentic AI 模式和工作流程已开启 AWS
- 为代理人工智能构建无服务器架构 AWS
- 为代理人工智能构建多租户架构 AWS
- 开启代理人工智能的安全性 AWS
- 检索增强生成选项和架构 AWS

AWS 资源

- Amazon Bedrock 文档
- 亚马逊 Bedrock 文档 AgentCore
- Amazon Bedrock AgentCore 入门工具包（GitHub 存储库）
- 亚马逊 Nova 文档
- AWS MCP 服务器（GitHub存储库）

其他 资源

- AutoGen文档 (Microsoft)
- 建立有效的代理人 (Anthropic)
- CrewAI GitHub存储库
- LangChain 文档
- LangGraph平台
- LlamaIndex 文档
- 模型上下文协议文档
- Strands Agents 文档
- Strands Agents工具概述
- Strands Agents快速入门指南

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
新章节	添加了 “平台” 部分	2026年1月16日
初次发布	—	2025 年 7 月 14 日

AWS 规范性指导词汇表

以下是 AWS 规范性指导提供的策略、指南和模式中的常用术语。若要推荐词条，请使用术语表末尾的提供反馈链接。

数字

7 R

将应用程序迁移到云中的 7 种常见迁移策略。这些策略以 Gartner 于 2011 年确定的 5 R 为基础，包括以下内容：

- 重构/重新架构：充分利用云原生功能来提高敏捷性、性能和可扩展性，以迁移应用程序并修改其架构。这通常涉及到移植操作系统和数据库。示例：将本地 Oracle 数据库迁移到 Amazon Aurora PostgreSQL 兼容版。
- 更换平台：将应用程序迁移到云中，并进行一定程度的优化，以利用云功能。示例：将本地 Oracle 数据库迁移到 AWS 云中的 Amazon Relational Database Service（Amazon RDS）for Oracle。
- 重新购买：转换到其他产品，通常是从传统许可转向 SaaS 模式。示例：将客户关系管理（CRM）系统迁移到 Salesforce.com。
- 重新托管（直接迁移）：将应用程序迁移到云中，无需进行任何更改即可利用云功能。示例：将本地 Oracle 数据库迁移到 AWS 云中 EC2 实例上的 Oracle。
- 重新放置（虚拟机监控器级直接迁移）：将基础设施迁移到云中，无需购买新硬件、重写应用程序或修改现有操作。您将服务器从本地平台迁移到同一平台的云服务中。示例：将Microsoft Hyper-V应用程序迁移到 AWS。
- 保留（重访）：将应用程序保留在源环境中。其中可能包括需要进行重大重构的应用程序，并且您希望将工作推迟到以后，以及您希望保留的遗留应用程序，因为迁移它们没有商业上的理由。
- 停用：停用或删除源环境中不再需要的应用程序。

A

ABAC

请参阅基于属性的访问控制。

抽象服务

请参阅托管服务。

ACID

请参阅原子性、一致性、隔离性、持久性。

主动-主动迁移

一种数据库迁移方法，在这种方法中，源数据库和目标数据库保持同步（通过使用双向复制工具或双写操作），两个数据库都在迁移期间处理来自连接应用程序的事务。这种方法支持小批量、可控的迁移，而不需要一次性割接。它比主动-被动迁移更灵活，但工作量更大。

主动-被动迁移

一种数据库迁移方法，在这种方法中，源数据库和目标数据库保持同步，但在将数据复制到目标数据库时，只有源数据库处理来自连接应用程序的事务。目标数据库在迁移期间不接受任何事务。

聚合函数

一种 SQL 函数，它对一组行进行操作并计算该组的单个返回值。聚合函数的示例包括 SUM 和 MAX。

AI

请参阅人工智能。

AIOps

请参阅人工智能运营。

匿名化

永久删除数据集中个人信息的过程。匿名化可以帮助保护个人隐私。匿名化数据不再被视为个人数据。

反模式

一种用于解决反复出现的问题的常用解决方案，而在这类问题中，此解决方案适得其反、无效或不如替代方案有效。

应用程序控制

一种安全方法，仅允许使用经批准的应用程序，以帮助保护系统免受恶意软件的侵害。

应用程序组合

有关组织使用的每个应用程序的详细信息的集合，包括构建和维护该应用程序的成本及其业务价值。这些信息是产品组合发现和分析过程的关键，有助于识别需要进行迁移、现代化和优化的应用程序并确定其优先级。

人工智能（AI）

计算机科学领域致力于使用计算技术执行通常与人类相关的认知功能，例如学习、解决问题和识别模式。有关更多信息，请参阅什么是人工智能？

人工智能操作 (AIOps)

使用机器学习技术解决运营问题、减少运营事故和人为干预以及提高服务质量的过程。有关如何在 AIOps AWS 迁移策略中使用的更多信息，请参阅操作集成指南。

非对称加密

一种加密算法，使用一对密钥，一个公钥用于加密，一个私钥用于解密。您可以共享公钥，因为它不用于解密，但对私钥的访问应受到严格限制。

原子性、一致性、隔离性、持久性（ACID）

一组软件属性，即使在出现错误、电源故障或其他问题的情况下，也能保证数据库的数据有效性和操作可靠性。

基于属性的访问权限控制（ABAC）

根据用户属性（如部门、工作角色和团队名称）创建精细访问权限的做法。有关更多信息，请参阅 AWS Identity and Access Management (I AM) 文档 AWS中的 AB AC。

权威数据来源

存储主要数据版本的位置，被认为是最可靠的信息源。您可以将数据从权威数据来源复制到其他位置，以便处理或修改数据，例如对数据进行匿名化、编辑或假名化。

可用区

中的一个不同位置 AWS 区域 ，不受其他可用区域故障的影响，并向同一区域中的其他可用区提供低成本、低延迟的网络连接。

AWS 云采用框架 (AWS CAF)

该框架包含指导方针和最佳实践 AWS ，可帮助组织制定高效且有效的计划，以成功迁移到云端。AWS CAF将指导分为六个重点领域，称为视角：业务、人员、治理、平台、安全和运营。业务、人员和治理角度侧重于业务技能和流程；平台、安全和运营角度侧重于技术技能和流程。例如，人

员角度针对的是负责人力资源（HR）、人员配置职能和人员管理的利益相关者。从这个角度来看，AWS CAF 为人员发展、培训和沟通提供了指导，以帮助组织为成功采用云做好准备。有关更多信息，请参阅 AWS CAF 网站和 AWS CAF 白皮书。

AWS 工作负载资格框架 (AWS WQF)

一种评估数据库迁移工作负载、推荐迁移策略和提供工作估算的工具。 AWS WQF 包含在 AWS Schema Conversion Tool (AWS SCT) 中。它用来分析数据库架构和代码对象、应用程序代码、依赖关系和性能特征，并提供评测报告。

B

恶意机器人

一种旨在扰乱或伤害个人或组织的机器人。

BCP

请参阅业务连续性计划。

行为图

一段时间内资源行为和交互的统一交互式视图。您可以使用 Amazon Detective 的行为图来检查失败的登录尝试、可疑的 API 调用和类似的操作。有关更多信息，请参阅 Detective 文档中的行为图中的数据。

大端序系统

一个先存储最高有效字节的系统。另请参阅字节顺序。

二进制分类

一种预测二进制结果（两个可能的类别之一）的过程。例如，您的 ML 模型可能需要预测诸如“该电子邮件是否为垃圾邮件？”或“这个产品是书还是汽车？”之类的问题

bloom 筛选条件

一种概率性、内存高效的数据结构，用于测试元素是否为集合的成员。

蓝/绿部署

一种部署策略，您可以创建两个独立但完全相同的环境。在一个环境中运行当前应用程序版本（蓝色），在另一个环境中运行新应用程序版本（绿色）。此策略可帮助您在影响最小的情况下快速回滚。

自动程序

一种通过互联网运行自动任务并模拟人类活动或交互的软件应用程序。有些机器人是有用或有益的，例如在互联网上索引信息的 Web 爬网程序。还有一些被称为恶意机器人的机器人，其目的是扰乱或伤害个人或组织。

僵尸网络

被恶意软件感染并受单方（称为僵尸网络控制者或僵尸网络操作者）控制的僵尸网络。僵尸网络是最著名的扩展机器人及其影响力的机制。

分支

代码存储库的一个包含区域。在存储库中创建的第一个分支是主分支。您可以从现有分支创建新分支，然后在新分支中开发功能或修复错误。为构建功能而创建的分支通常称为功能分支。当功能可以发布时，将功能分支合并回主分支。有关更多信息，请参阅关于分支（GitHub 文档）。

紧急（break-glass）访问

在特殊情况下，通过批准的流程，用户 AWS 账户 可以快速访问他们通常没有访问权限的内容。有关更多信息，请参阅 AWS Well-Architected Guidance 中的 Implement break-glass procedures 指示器。

棕地策略

您环境中的现有基础设施。在为系统架构采用棕地策略时，您需要围绕当前系统和基础设施的限制来设计架构。如果您正在扩展现有基础设施，则可以将棕地策略和全新策略混合。

缓冲区缓存

存储最常访问的数据的内存区域。

业务能力

企业如何创造价值（例如，销售、客户服务或营销）。微服务架构和开发决策可以由业务能力驱动。有关更多信息，请参阅在 AWS上运行容器化微服务白皮书中的围绕业务能力进行组织部分。

业务连续性计划（BCP）

一项计划，旨在应对大规模迁移等破坏性事件对运营的潜在影响，并使企业能够快速恢复运营。

C

CAF

请参阅 AWS 云采用框架。

金丝雀部署

缓慢而渐进地向最终用户发布版本。当您确信无误后，即可部署新版本，并完全替换当前版本。

CCoE

请参阅云卓越中心。

CDC

请参阅更改数据捕获。

更改数据捕获（CDC）

跟踪数据来源（如数据库表）的更改并记录有关更改的元数据的过程。您可以将 CDC 用于各种目的，例如审计或复制目标系统中的更改以保持同步。

混沌工程

故意引入故障或破坏性事件来测试系统的韧性。您可以使用 AWS Fault Injection Service (AWS FIS) 来执行实验，对您的 AWS 工作负载施加压力并评估其响应。

CI/CD

请参阅持续集成和持续交付。

分类

一种有助于生成预测的分类流程。分类问题的 ML 模型预测离散值。离散值始终彼此不同。例如，一个模型可能需要评估图像中是否有汽车。

客户端加密

在目标 AWS 服务 收到数据之前，对数据进行本地加密。

云卓越中心 (CCoE)

一个多学科团队，负责推动整个组织的云采用工作，包括开发云最佳实践、调动资源、制定迁移时间表、领导组织完成大规模转型。有关更多信息，请参阅 AWS 云 企业战略博客上的 CCoE 帖子。

云计算

通常用于远程数据存储和 IoT 设备管理的云技术。云计算通常连接到边缘计算技术。

云运营模型

在 IT 组织中，一种用于构建、完善和优化一个或多个云环境的运营模型。有关更多信息，请参阅构建您的云运营模型。

云采用阶段

组织迁移到 AWS 云中时通常会经历四个阶段：

- 项目 - 出于概念验证和学习目的，开展一些与云相关的项目
- 基础 — 进行基础投资以扩大云采用率（例如，创建着陆区、定义 CCo E、建立运营模型）
- 迁移 - 迁移单个应用程序
- 重塑 - 优化产品和服务，在云中创新

Stephen Orban在 AWS 云 企业战略博客的博客文章《云优先之旅和采用阶段》中定义了这些阶段。有关它们与 AWS 迁移策略的关系的信息，请参阅迁移准备指南。

CMDB

请参阅配置管理数据库。

代码存储库

通过版本控制过程存储和更新源代码和其他资产（如文档、示例和脚本）的位置。常见的云存储库包括 GitHub 或 Bitbucket Cloud。每个版本的代码都称为一个分支。在微服务结构中，每个存储库都专门用于一个功能。单个 CI/CD 管线可以使用多个存储库。

冷缓存

一种空的、填充不足或包含过时或不相关数据的缓冲区缓存。这会影响性能，因为数据库实例必须从主内存或磁盘读取，这比从缓冲区缓存读取要慢。

冷数据

很少访问的数据，且通常是历史数据。查询此类数据时，通常可以接受慢速查询。将这些数据转移到性能较低且成本更低的存储层或类别可以降低成本。

计算机视觉（CV）

一种 AI 领域，它使用机器学习来分析和提取数字图像和视频等视觉格式中的信息。例如，Amazon SageMaker AI 为 CV 提供了图像处理算法。

配置偏移

对于工作负载而言，一种偏离预期状态的配置更改。这可能会导致工作负载变得不合规，且通常是渐进的，不是故意的。

配置管理数据库（CMDB）

一种存储库，用于存储和管理有关数据库及其 IT 环境的信息，包括硬件和软件组件及其配置。您通常在迁移的产品组合发现和分析阶段使用来自 CMDB 的数据。

合规性包

一系列 AWS Config 规则和补救措施，您可以汇编这些规则和补救措施，以自定义您的合规性和安全性检查。您可以使用 YAML 模板将一致性包作为单个实体部署在 AWS 账户 和区域或整个组织中。有关更多信息，请参阅 AWS Config 文档中的一致性包。

持续集成和持续交付（CI/CD）

自动执行软件发布过程的源代码、构建、测试、暂存和生产阶段的过程。 CI/CD 通常被描述为管道。 CI/CD 可以帮助您实现流程自动化、提高生产力、提高代码质量和更快地交付。有关更多信息，请参阅持续交付的优势。CD 也可以表示持续部署。有关更多信息，请参阅持续交付与持续部署。

CV

请参阅计算机视觉。

D

静态数据

网络中静止的数据，例如存储中的数据。

数据分类

根据网络中数据的关键性和敏感性对其进行识别和分类的过程。它是任何网络安全风险管理策略的关键组成部分，因为它可以帮助您确定对数据的适当保护和保留控制。数据分类是 Well-Architecte AWS d Framework 中安全支柱的一个组成部分。有关详细信息，请参阅数据分类。

数据漂移

生产数据与用来训练机器学习模型的数据之间的有意义差异，或者输入数据随时间推移的有意义变化。数据漂移可能降低机器学习模型预测的整体质量、准确性和公平性。

传输中数据

在网络中主动移动的数据，例如在网络资源之间移动的数据。

数据网格

一种架构框架，可提供分布式、去中心化的数据所有权以及集中式管理和治理。

数据最少化

仅收集并处理绝对必要数据的原则。在中进行数据最小化 AWS 云 可以降低隐私风险、成本和分析碳足迹。

数据边界

AWS 环境中的一组预防性防护措施，可帮助确保只有可信身份才能访问来自预期网络的可信资源。有关更多信息，请参阅在上构建数据边界。 AWS

数据预处理

将原始数据转换为 ML 模型易于解析的格式。预处理数据可能意味着删除某些列或行，并处理缺失、不一致或重复的值。

数据溯源

在数据的整个生命周期跟踪其来源和历史的过程，例如数据如何生成、传输和存储。

数据主体

正在收集和处理其数据的个人。

数据仓库

一种支持商业智能（例如分析）的数据管理系统。数据仓库通常包含大量历史数据，通常用于查询和分析。

数据库定义语言（DDL）

在数据库中创建或修改表和对象结构的语句或命令。

数据库操作语言（DML）

在数据库中修改（插入、更新和删除）信息的语句或命令。

DDL

请参阅数据库定义语言。

深度融合

组合多个深度学习模型进行预测。您可以使用深度融合来获得更准确的预测或估算预测中的不确定性。

深度学习

一个 ML 子字段使用多层人工神经网络来识别输入数据和感兴趣的目标变量之间的映射。

defense-in-depth

一种信息安全方法，经过深思熟虑，在整个计算机网络中分层实施一系列安全机制和控制措施，以保护网络及其中数据的机密性、完整性和可用性。当你采用这种策略时 AWS，你会在 AWS

Organizations 结构的不同层面添加多个控件来帮助保护资源。例如，一种 defense-in-depth方法可以结合多因素身份验证、网络分段和加密。

委派管理员

在中 AWS Organizations，兼容的服务可以注册 AWS 成员帐户来管理组织的帐户并管理该服务的权限。此账户被称为该服务的委托管理员。有关更多信息和兼容服务列表，请参阅 AWS Organizations 文档中使用 AWS Organizations的服务。

部署

使应用程序、新功能或代码修复在目标环境中可用的过程。部署涉及在代码库中实现更改，然后在应用程序的环境中构建和运行该代码库。

开发环境

请参阅环境。

侦测性控制

一种安全控制，在事件发生后进行检测、记录日志和发出提醒。这些控制是第二道防线，提醒您注意绕过现有预防性控制的安全事件。有关更多信息，请参阅在 AWS上实施安全控制中的侦测性控制。

开发价值流映射（DVSM）

用于识别对软件开发生命周期中的速度和质量产生不利影响的限制因素并确定其优先级的流程。DVSM 扩展了最初为精益生产实践设计的价值流映射流程。其重点关注在软件开发过程中创造和转移价值所需的步骤和团队。

数字孪生

真实世界系统的虚拟再现，如建筑物、工厂、工业设备或生产线。数字孪生支持预测性维护、远程监控和生产优化。

维度表

星型架构中的一种较小的表，其中包含事实表中定量数据的数据属性。维度表属性通常是文本字段或行为类似于文本的离散数字。这些属性通常用于查询约束、筛选和结果集标注。

灾难

阻止工作负载或系统在其主要部署位置实现其业务目标的事件。这些事件可能是自然灾害、技术故障或人为操作的结果，例如无意的配置错误或恶意软件攻击。

灾难恢复（DR）

您用来最大程度地减少由灾难造成的停机时间和数据丢失的策略和流程。有关更多信息，请参阅 Well-Architected Fr ame AWS work 中的 “工作负载灾难恢复：云端 AWS 恢复”。

DML

请参阅数据库操作语言。

领域驱动设计

一种开发复杂软件系统的方法，通过将其组件连接到每个组件所服务的不断发展的领域或核心业务目标。Eric Evans 在其著作领域驱动设计：软件核心复杂性应对之道（Boston: Addison-Wesley Professional, 2003）中介绍了这一概念。有关如何将领域驱动设计与 strangler fig 模式结合使用的信息，请参阅使用容器和 Amazon API Gateway 逐步将原有的 Microsoft ASP.NET（ASMX）Web 服务现代化。

DR

请参阅灾难恢复。

偏差检测

跟踪与基准配置的偏差。例如，您可以使用 AWS CloudFormation 来检测系统资源中的偏差，也可以使用 AWS Control Tower 来检测着陆区中可能影响监管要求合规性的变化。

DVSM

请参阅开发价值流映射。

E

EDA

请参阅探索性数据分析。

EDI

请参阅电子数据交换。

边缘计算

该技术可提高位于 IoT 网络边缘的智能设备的计算能力。与云计算比较时，边缘计算可以减少通信延迟并缩短响应时间。

电子数据交换（EDI）

组织之间业务文件的自动交换。有关更多信息，请参阅什么是电子数据交换。

加密

一种将人类可读的纯文本数据转换为加密文字的计算流程。

加密密钥

由加密算法生成的随机位的加密字符串。密钥的长度可能有所不同，而且每个密钥都设计为不可预测且唯一。

字节顺序

字节在计算机内存中的存储顺序。大端序系统先存储最高有效字节。小端序系统先存储最低有效字节。

端点

请参阅服务端点。

端点服务

一种可以在虚拟私有云（VPC）中托管，与其他用户共享的服务。您可以使用其他 AWS 账户 或 AWS Identity and Access Management (IAM) 委托人创建终端节点服务， AWS PrivateLink 并向其授予权限。这些账户或主体可通过创建接口 VPC 端点来私密地连接到您的端点服务。有关更多信息，请参阅 Amazon Virtual Private Cloud（Amazon VPC）文档中的创建端点服务。

企业资源规划（ERP）

一种自动化和管理企业关键业务流程（例如会计、MES 和项目管理）的系统。

信封加密

用另一个加密密钥对加密密钥进行加密的过程。有关更多信息，请参阅 AWS Key Management Service (AWS KMS) 文档中的信封加密。

环境

正在运行的应用程序的实例。以下是云计算中常见的环境类型：

- 开发环境 — 正在运行的应用程序的实例，只有负责维护应用程序的核心团队才能使用。开发环境用于测试更改，然后再将其提升到上层环境。这类环境有时称为测试环境。
- 下层环境 — 应用程序的所有开发环境，比如用于初始构建和测试的环境。

- 生产环境 — 最终用户可以访问的正在运行的应用程序的实例。在 CI/CD 管道中，生产环境是最后一个部署环境。
- 上层环境 — 除核心开发团队以外的用户可以访问的所有环境。这可能包括生产环境、预生产环境和用户验收测试环境。

epic

在敏捷方法学中，有助于组织工作和确定优先级的功能类别。epics 提供了对需求和实施任务的总体描述。例如， AWS CAF 安全史诗包括身份和访问管理、侦探控制、基础设施安全、数据保护和事件响应。有关 AWS 迁移策略中 epics 的更多信息，请参阅计划实施指南。

ERP

请参阅企业资源规划。

探索性数据分析（EDA）

分析数据集以了解其主要特征的过程。您收集或汇总数据，并进行初步调查，以发现模式、检测异常并检查假定情况。EDA 通过计算汇总统计数据和创建数据可视化得以执行。

F

事实表

星型架构中的中心表。它存储有关业务运营的定量数据。通常，事实表包含两种类型的列：包含度量的列和包含维度表外键的列。

快速失效机制

一种使用频繁且增量式的测试来缩短开发生命周期的理念。这是敏捷方法的关键部分。

故障隔离边界

在中 AWS 云，诸如可用区 AWS 区域、控制平面或数据平面之类的边界，它限制了故障的影响并有助于提高工作负载的弹性。有关更多信息，请参阅 AWS 故障隔离边界。

功能分支

请参阅分支。

特征

您用来进行预测的输入数据。例如，在制造环境中，特征可能是定期从生产线捕获的图像。

特征重要性

特征对于模型预测的重要性。这通常表示为数值分数，可以通过各种技术进行计算，例如 Shapley 加法解释（SHAP）和积分梯度。有关更多信息，请参阅使用机器学习模型的可解释性 AWS。

功能转换

为 ML 流程优化数据，包括使用其他来源丰富数据、扩展值或从单个数据字段中提取多组信息。这使得 ML 模型能从数据中获益。例如，如果您将“2021-05-27 00:15:37”日期分解为“2021”、“五月”、“星期四”和“15”，则可以帮助学习与不同数据成分相关的算法学习精细模式。

少样本提示

在要求 LLM 执行类似任务之前，先向其提供少量示例，以演示任务和预期输出。此技术是上下文内学习的一种应用，其中模型可以从提示中嵌入的示例（样本）中学习。对于需要特定格式、推理或领域知识的任务，少样本提示可能非常有效。另请参阅零样本提示。

FGAC

请参阅精细访问控制。

精细访问控制（FGAC）

使用多个条件允许或拒绝访问请求。

快闪迁移

一种数据库迁移方法，通过更改数据捕获使用连续数据复制，在极短的时间内迁移数据，而非使用分阶段方法。目标是将停机时间降至最低。

FM

请参阅基础模型。

基础模型（FM）

一个大型深度学习神经网络，一直在广义和未标记数据的大量数据集上进行训练。 FMs 能够执行各种各样的一般任务，例如理解语言、生成文本和图像以及用自然语言进行对话。有关更多信息，请参阅什么是基础模型。

G

生成式人工智能

AI 模型的一个子集，这些模型已经过大量数据训练，可以使用简单的文本提示来创建新的内容和构件，例如图像、视频、文本和音频。有关更多信息，请参阅什么是生成式人工智能。

地理阻止

请参阅地理限制。

地理限制（地理阻止）

在 Amazon 中 CloudFront，一种阻止特定国家/地区的用户访问内容分发的选项。您可以使用允许列表或阻止列表来指定已批准和已禁止的国家/地区。有关更多信息，请参阅 CloudFront 文档中的限制内容的地理分布。

GitFlow 工作流程

一种方法，在这种方法中，下层和上层环境在源代码存储库中使用不同的分支。Gitflow 工作流程被认为是传统的工作流程，而基于中继的工作流程则是现代的、首选的方法。

黄金映像

系统或软件的快照，用作部署该系统或软件的新实例的模板。例如，在制造业中，黄金映像可用于在多个设备上预调配软件，并有助于提高设备制造操作的速度、可扩展性和生产效率。

全新策略

在新环境中缺少现有基础设施。在对系统架构采用全新策略时，您可以选择所有新技术，而不受对现有基础设施（也称为棕地）兼容性的限制。如果您正在扩展现有基础设施，则可以将棕地策略和全新策略混合。

防护机制

帮助管理各组织单位的资源、策略和合规性的高级规则 (OUs)。预防性防护机制会执行策略以确保符合合规性标准。它们是使用服务控制策略和 IAM 权限边界实现的。侦测性护栏会检测策略违规和合规性问题，并生成提醒以进行修复。它们通过使用 AWS Config、、Amazon、 AWS Security Hub CSPM GuardDuty AWS Trusted Advisor、Amazon Inspector 和自定义 AWS Lambda 支票来实现。

H

HA

请参阅高可用性。

异构数据库迁移

将源数据库迁移到使用不同数据库引擎的目标数据库（例如，从 Oracle 迁移到 Amazon Aurora）。异构迁移通常是重新架构工作的一部分，而转换架构可能是一项复杂的任务。AWS 提供了 AWS SCT 来帮助实现架构转换。

高可用性（HA）

在遇到挑战或灾难时，工作负载无需干预即可连续运行的能力。HA 系统旨在自动进行故障转移、持续提供良好性能，并以最小的性能影响处理不同负载和故障。

历史数据库现代化

一种用于实现运营技术（OT）系统现代化和升级以更好满足制造业需求的方法。历史数据库是一种用于收集和存储工厂中各种来源数据的数据库。

保留数据

从用于训练机器学习模型的数据集中保留的一部分标注的历史数据。通过将模型预测与保留数据进行比较，您可以使用保留数据来评估模型性能。

同构数据库迁移

将源数据库迁移到共享同一数据库引擎的目标数据库（例如，从 Microsoft SQL Server 迁移到 Amazon RDS for SQL Server）。同构迁移通常是更换主机或更换平台工作的一部分。您可以使用本机数据库实用程序来迁移架构。

热数据

经常访问的数据，例如实时数据或近期的转化数据。这些数据通常需要高性能存储层或存储类别才能提供快速的查询响应。

修补程序

针对生产环境中关键问题的紧急修复。由于其紧迫性，修补程序通常是在典型的 DevOps 发布工作流程之外进行的。

hypercare 周期

割接之后，迁移团队立即管理和监控云中迁移的应用程序以解决任何问题的时间段。通常，这个周期持续 1-4 天。在 hypercare 周期结束时，迁移团队通常会将应用程序的责任移交给云运营团队。

我

IaC

请参阅基础设施即代码。

基于身份的策略

附加到一个或多个 IAM 委托人的策略，用于定义他们在 AWS 云 环境中的权限。

空闲应用程序

90 天内平均 CPU 和内存使用率在 5% 到 20% 之间的应用程序。在迁移项目中，通常会停用这些应用程序或将其保留在本地。

IIoT

请参阅工业物联网。

不可变基础设施

一种模型，可为生产工作负载部署新的基础设施，而不是更新、修补或修改现有基础设施。不可变基础设施本质上比可变基础设施更一致、更可靠、更可预测。有关更多信息，请参阅 AWS Well-Architected Framework 中的使用不可变基础设施进行部署最佳实践。

入站（入口）VPC

在 AWS 多账户架构中，一种接受、检查和路由来自应用程序外部的网络连接的 VPC。AWS 安全参考架构建议设置您的网络帐户，包括入站、出站和检查， VPCs 以保护您的应用程序与更广泛的互联网之间的双向接口。

增量迁移

一种割接策略，在这种策略中，您可以将应用程序分成小部分进行迁移，而不是一次性完整割接。例如，您最初可能只将几个微服务或用户迁移到新系统。在确认一切正常后，您可以逐步迁移其他微服务或用户，直到停用遗留系统。这种策略降低了大规模迁移带来的风险。

工业 4.0

该术语由 Klaus Schwab 在 2016 年提出，指的是通过连接、实时数据、自动化、分析和 AI/ML 的进步来实现制造流程的现代化。

基础设施

应用程序环境中包含的所有资源和资产。

基础设施即代码（IaC）

通过一组配置文件预调配和管理应用程序基础设施的过程。IaC 旨在帮助您集中管理基础设施、实现资源标准化和快速扩展，使新环境具有可重复性、可靠性和一致性。

工业物联网 (IIoT)

在工业领域使用联网的传感器和设备，例如制造业、能源、汽车、医疗保健、生命科学和农业。有关更多信息，请参阅制定工业物联网 (IIoT) 数字化转型战略。

检查 VPC

在 AWS 多账户架构中，一种集中式 VPC，用于管理对 VPCs（相同或不同 AWS 区域）、互联网和本地网络之间的网络流量的检查。AWS 安全参考架构建议设置您的网络帐户，包括入站、出站和检查， VPCs 以保护您的应用程序与更广泛的互联网之间的双向接口。

物联网（IoT）

由带有嵌入式传感器或处理器的连接物理对象组成的网络，这些传感器或处理器通过互联网或本地通信网络与其他设备和系统进行通信。有关更多信息，请参阅什么是 IoT？

可解释性

它是机器学习模型的一种特征，描述了人类可以理解模型的预测如何取决于其输入的程度。有关更多信息，请参阅使用机器学习模型的可解释性 AWS。

物联网

请参阅物联网。

IT 信息库（ITIL）

提供 IT 服务并使这些服务符合业务要求的一套最佳实践。ITIL 是 ITSM 的基础。

IT 服务管理（ITSM）

为组织设计、实施、管理和支持 IT 服务的相关活动。有关将云运营与 ITSM 工具集成的信息，请参阅运营集成指南。

ITIL

请参阅 IT 信息库。

ITSM

请参阅 IT 服务管理。

L

基于标签的访问控制（LBAC）

强制访问控制（MAC）的一种实施方式，其中明确为用户和数据本身分配了安全标签值。用户安全标签和数据安全标签之间的交集决定了用户可以看到哪些行和列。

登录区

landing zone 是一个架构精良的多账户 AWS 环境，具有可扩展性和安全性。这是一个起点，您的组织可以从这里放心地在安全和基础设施环境中快速启动和部署工作负载和应用程序。有关登录区的更多信息，请参阅设置安全且可扩展的多账户 AWS 环境。

大语言模型（LLM）

一种基于大量数据进行预训练的深度学习 AI 模型。LLM 可以执行多项任务，例如回答问题、总结文档、将文本翻译成其他语言以及完成句子。有关更多信息，请参阅什么是 LLMs。

大规模迁移

迁移 300 台或更多服务器。

LBAC

请参阅基于标签的访问控制。

最低权限

授予执行任务所需的最低权限的最佳安全实践。有关更多信息，请参阅 IAM 文档中的应用最低权限许可。

直接迁移

请参阅 7 R。

小端序系统

一个先存储最低有效字节的系统。另请参阅字节顺序。

LLM

请参阅大型语言模型。

下层环境

请参阅环境。

M

机器学习（ML）

一种使用算法和技术进行模式识别和学习的人工智能。ML 对记录的数据（例如物联网（IoT）数据）进行分析和学习，以生成基于模式的统计模型。有关更多信息，请参阅机器学习。

主分支

请参阅分支。

恶意软件

旨在危害计算机安全或隐私的软件。恶意软件可能会破坏计算机系统、泄露敏感信息或获得未经授权的访问权限。恶意软件的示例包括病毒、蠕虫、勒索软件、木马、间谍软件和键盘记录器。

托管式服务

AWS 服务 它 AWS 运行基础设施层、操作系统和平台，您可以访问端点来存储和检索数据。Amazon Simple Storage Service（Amazon S3）和 Amazon DynamoDB 就是托管服务的示例。这些服务也称为抽象服务。

制造执行系统（MES）

一种软件系统，用于跟踪、监控、记录和控制在车间将原材料转化为成品的生产过程。

MAP

请参阅迁移加速计划。

机制

一个完整的过程，您可以在其中创建工具，推动工具的采用，然后检查结果以进行调整。机制是一种在运作过程中自我强化和改善的循环。有关更多信息，请参阅在 Well-Architect AWS ed 框架中构建机制。

成员账户

AWS 账户 除属于组织中的管理账户之外的所有账户 AWS Organizations。一个账户一次只能是一个组织的成员。

MES

请参阅制造执行系统。

消息队列遥测传输（MQTT）

一种基于发布/订阅模式的轻量级 machine-to-machine (M2M) 通信协议，适用于资源受限的物联网设备。

微服务

一种小型的独立服务，通过明确的定义进行通信 APIs ，通常由小型的独立团队拥有。例如，保险系统可能包括映射到业务能力（如销售或营销）或子域（如购买、理赔或分析）的微服务。微服务

的好处包括敏捷、灵活扩展、易于部署、可重复使用的代码和恢复能力。有关更多信息，请参阅使用 AWS 无服务器服务集成微服务。

微服务架构

一种使用独立组件构建应用程序的方法，这些组件将每个应用程序进程作为微服务运行。这些微服务使用轻量级通过定义明确的接口进行通信。 APIs该架构中的每个微服务都可以更新、部署和扩展，以满足对应用程序特定功能的需求。有关更多信息，请参阅在上实现微服务。 AWS

迁移加速计划（MAP）

AWS 该计划提供咨询支持、培训和服务，以帮助组织为迁移到云奠定坚实的运营基础，并帮助抵消迁移的初始成本。MAP 提供了一种以系统的方式执行遗留迁移的迁移方法，以及一套用于自动执行和加速常见迁移场景的工具。

大规模迁移

将大部分应用程序组合分波迁移到云中的过程，在每一波中以更快的速度迁移更多应用程序。本阶段使用从早期阶段获得的最佳实践和经验教训，实施由团队、工具和流程组成的迁移工厂，通过自动化和敏捷交付简化工作负载的迁移。这是 AWS 迁移策略的第三阶段。

迁移工厂

跨职能团队，通过自动化、敏捷的方法简化工作负载迁移。迁移工厂团队通常包括运营、业务分析师和所有者、迁移工程师、开发人员和冲刺 DevOps 领域的专业人员。20% 到 50% 的企业应用程序组合由可通过工厂方法优化的重复模式组成。有关更多信息，请参阅本内容集中有关迁移工厂的讨论和云迁移工厂指南。

迁移元数据

有关完成迁移所需的应用程序和服务器的信息。每种迁移模式都需要一套不同的迁移元数据。迁移元数据的示例包括目标子网、安全组和 AWS 账户。

迁移模式

一种可重复的迁移任务，详细列出了迁移策略、迁移目标以及所使用的迁移应用程序或服务。示例：使用 AWS 应用程序迁移服务重新托管向 Amazon EC2 的迁移。

迁移组合评测（MPA）

一种在线工具，提供了用于验证迁移到 AWS 云的业务案例的信息。MPA 提供了详细的组合评测（服务器规模调整、定价、TCO 比较、迁移成本分析）以及迁移计划（应用程序数据分析和数据收集、应用程序分组、迁移优先级排序和波次规划）。所有 AWS 顾问和 APN 合作伙伴顾问均可免费使用 MPA 工具（需要登录）。

迁移准备情况评测（MRA）

使用 AWS CAF 深入了解组织的云就绪状态、确定优势和劣势以及制定行动计划以缩小已发现差距的过程。有关更多信息，请参阅迁移准备指南。MRA 是 AWS 迁移策略的第一阶段。

迁移策略

将工作负载迁移到 AWS 云的方法。有关更多信息，请参见术语表中的 7 R 词条，以及动员您的组织以加快大规模迁移。

ML

请参阅机器学习。

现代化

将过时的（原有的或单体）应用程序及其基础设施转变为云中敏捷、弹性和高度可用的系统，以降低成本、提高效率和利用创新。有关更多信息，请参阅在 AWS 云中实现应用程序现代化的策略。

现代化准备情况评估

一种评估方式，有助于确定组织应用程序的现代化准备情况；确定收益、风险和依赖关系；确定组织能够在多大程度上支持这些应用程序的未来状态。评估结果是目标架构的蓝图、详细说明现代化进程发展阶段和里程碑的路线图以及解决已发现差距的行动计划。有关更多信息，请参阅在 AWS 云中评估应用程序的现代化准备情况。

单体应用程序（单体式）

作为具有紧密耦合进程的单个服务运行的应用程序。单体应用程序有几个缺点。如果某个应用程序功能的需求激增，则必须扩展整个架构。随着代码库的增长，添加或改进单体应用程序的功能也会变得更加复杂。若要解决这些问题，可以使用微服务架构。有关更多信息，请参阅将单体分解为微服务。

MPA

请参阅迁移组合评测。

MQTT

请参阅消息队列遥测传输。

多分类器

一种帮助为多个类别生成预测（预测两个以上结果之一）的过程。例如，ML 模型可能会询问“这个产品是书、汽车还是手机？”或“此客户最感兴趣什么类别的产品？”

可变基础设施

一种用于更新和修改生产工作负载的现有基础设施的模型。为了提高一致性、可靠性和可预测性，Well-Architect AWS ed Framework 建议使用不可变基础设施作为最佳实践。

O

OAC

请参阅来源访问控制。

OAI

请参阅来源访问身份。

OCM

请参阅组织变革管理。

离线迁移

一种迁移方法，在这种方法中，源工作负载会在迁移过程中停止运行。这种方法会延长停机时间，通常用于小型非关键工作负载。

OI

请参阅运营集成。

OLA

请参阅运营级别协议。

在线迁移

一种迁移方法，在这种方法中，源工作负载无需离线即可复制到目标系统。在迁移过程中，连接工作负载的应用程序可以继续运行。这种方法的停机时间为零或最短，通常用于关键生产工作负载。

OPC-UA

请参阅开放流程通信 – 统一架构。

开放流程通信 – 统一架构（OPC-UA）

一种用于工业自动化的 machine-to-machine（M2M）通信协议。OPC-UA 提供了一个包含数据加密、身份验证和授权方案的互操作性标准。

运营级别协议（OLA）

一项协议，阐明了 IT 职能部门承诺相互交付的内容，以支持服务水平协议（SLA）。

运营准备情况审查（ORR）

一份问题核对清单和关联的最佳实践，可帮助您了解、评估、预防或缩小事件和可能的故障的范围。有关更多信息，请参阅 AWS Well-Architected Framework 中的运营准备情况审查（ORR）。

运营技术（OT）

与物理环境配合使用以控制工业运营、设备和基础设施的硬件和软件系统。在制造业中，OT 和信息技术（IT）系统的集成是工业 4.0 转型的关键重点。

运营整合（OI）

在云中实现运营现代化的过程，包括就绪计划、自动化和集成。有关更多信息，请参阅运营整合指南。

组织跟踪

由 AWS CloudTrail 此创建的跟踪记录组织 AWS 账户 中所有人的所有事件 AWS Organizations。该跟踪是在每个 AWS 账户 中创建的，属于组织的一部分，并跟踪每个账户的活动。有关更多信息，请参阅 CloudTrail文档中的为组织创建跟踪。

组织变革管理（OCM）

一个从人员、文化和领导力角度管理重大、颠覆性业务转型的框架。OCM 通过加快变革采用、解决过渡问题以及推动文化和组织变革，帮助组织为新系统和战略做好准备和过渡。在 AWS 迁移策略中，该框架被称为人员加速，因为云采用项目需要变更的速度。有关更多信息，请参阅 OCM 指南。

来源访问控制（OAC）

在中 CloudFront，一个增强的选项，用于限制访问以保护您的亚马逊简单存储服务 (Amazon S3) 内容。OAC 全部支持所有 S3 存储桶 AWS 区域、使用 AWS KMS (SSE-KMS) 进行服务器端加密，以及对 S3 存储桶的动态PUT和DELETE请求。

来源访问身份（OAI）

在中 CloudFront，一个用于限制访问权限以保护您的 Amazon S3 内容的选项。当您使用 OAI 时，CloudFront 会创建一个 Amazon S3 可以对其进行身份验证的委托人。经过身份验证的委托人只能通过特定 CloudFront 分配访问 S3 存储桶中的内容。另请参阅 OAC，其中提供了更精细和增强的访问控制。

ORR

请参阅运营准备情况审查。

OT

请参阅运营技术。

出站（出口）VPC

在 AWS 多账户架构中，一种处理从应用程序内部启动的网络连接的 VPC。AWS 安全参考架构建议设置您的网络帐户，包括入站、出站和检查， VPCs 以保护您的应用程序与更广泛的互联网之间的双向接口。

P

权限边界

附加到 IAM 主体的 IAM 管理策略，用于设置用户或角色可以拥有的最大权限。有关更多信息，请参阅 IAM 文档中的权限边界。

个人身份信息（PII）

直接查看其他相关数据或与之配对时可用于合理推断个人身份的信息。PII 的示例包括姓名、地址和联系信息。

PII

请参阅个人身份信息。

playbook

一套预定义的步骤，用于捕获与迁移相关的工作，例如在云中交付核心运营功能。playbook 可以采用脚本、自动化运行手册的形式，也可以是操作现代化环境所需的流程或步骤的摘要。

PLC

请参阅可编程逻辑控制器。

PLM

请参阅产品生命周期管理。

policy

一个对象，可以定义权限（请参阅基于身份的策略）、指定访问条件（请参阅基于资源的策略）或定义 AWS Organizations 的组织中所有账户的最大权限（请参阅服务控制策略）。

多语言持久性

根据数据访问模式和其他要求，独立选择微服务的数据存储技术。如果您的微服务采用相同的数据存储技术，它们可能会遇到实现难题或性能不佳。如果微服务使用最适合其需求的数据存储，则可以更轻松地实现微服务，并获得更好的性能和可扩展性。

组合评测

一个发现、分析和确定应用程序组合优先级以规划迁移的过程。有关更多信息，请参阅评估迁移准备情况。

谓词

返回 `true` 或 `false` 的查询条件，通常位于 `WHERE` 子句中。

谓词下推

一种数据库查询优化技术，可在传输之前筛选查询中的数据。这将减少从关系数据库检索和处理的数据量，并提高查询性能。

预防性控制

一种安全控制，旨在防止事件发生。这些控制是第一道防线，帮助防止未经授权的访问或对网络的意外更改。有关更多信息，请参阅在 AWS上实施安全控制中的预防性控制。

主体

中 AWS 可以执行操作和访问资源的实体。此实体通常是 IAM 角色的根用户或用户。 AWS 账户有关更多信息，请参阅 IAM 文档中角色术语和概念中的主体。

隐私设计

一种在整个开发过程中都考虑隐私的系统工程方法。

私有托管区

一个容器，其中包含有关您希望 Amazon Route 53 如何响应针对一个或多个 VPCs域名及其子域名的 DNS 查询的信息。有关更多信息，请参阅 Route 53 文档中的私有托管区的使用。

主动控制

一种安全控制，旨在防止部署不合规资源。这些控制会在资源预置之前对其进行扫描。如果资源与控制不兼容，则不会预置它。有关更多信息，请参阅 AWS Control Tower 文档中的控制参考指南，并参见在上实施安全控制中的主动控制 AWS。

产品生命周期管理（PLM）

对产品在其整个生命周期内的数据和流程的管理，从设计、开发和发布，到增长和成熟，再到衰退和淘汰。

生产环境

请参阅环境。

可编程逻辑控制器（PLC）

在制造业中，一种高度可靠、适应性强的计算机，用于监控机器并实现制造过程自动化。

提示串接

使用一个 LLM 提示的输出作为下一个提示的输入，以生成更好的响应。该技术用于将复杂的任务分解为子任务，或者迭代地完善或扩展初步响应。它有助于提高模型响应的准确性和相关性，并允许获得更精细的个性化结果。

假名化

用占位符值替换数据集中个人标识符的过程。假名化可以帮助保护个人隐私。假名化数据仍被视为个人数据。

publish/subscribe (pub/sub)

一种支持微服务间异步通信的模式，可提高可扩展性和响应能力。例如，在基于微服务的 MES 中，微服务可以将事件消息发布到其他微服务可以订阅的频道。系统可以在不更改发布服务的情况下添加新的微服务。

Q

查询计划

一系列用于访问 SQL 关系数据库系统中的数据的步骤，类似于指令。

查询计划回归

当数据库服务优化程序选择的最佳计划不如数据库环境发生特定变化之前时。这可能是由统计数据、约束、环境设置、查询参数绑定更改和数据库引擎更新造成的。

R

RACI 矩阵

请参阅责任、问责、咨询和知情（RACI）。

RAG

请参阅检索增强生成。

勒索软件

一种恶意软件，旨在阻止对计算机系统或数据的访问，直到付款为止。

RASCI 矩阵

请参阅责任、问责、咨询和知情（RACI）。

RCAC

请参阅行列访问控制。

只读副本

用于只读目的的数据库副本。您可以将查询路由到只读副本，以减轻主数据库的负载。

重新架构

请参阅 7 R。

恢复点目标（RPO）

自上一个数据恢复点以来可接受的最长时间。这决定了从上一个恢复点到服务中断之间可接受的数据丢失情况。

恢复时间目标（RTO）

服务中断和服务恢复之间可接受的最大延迟。

重构

请参阅 7 R。

Region

地理区域内的 AWS 资源集合。每一个 AWS 区域 都相互隔离，相互独立，以提供容错、稳定性和弹性。有关更多信息，请参阅指定您的账户可以使用的 AWS 区域。

回归

一种预测数值的 ML 技术。例如，要解决“这套房子的售价是多少？”的问题 ML 模型可以使用线性回归模型，根据房屋的已知事实（如建筑面积）来预测房屋的销售价格。

重新托管

请参阅 7 R。

版本

在部署过程中，推动生产环境变更的行为。

重新放置

请参阅 7 R。

更换平台

请参阅 7 R。

重新购买

请参阅 7 R。

韧性

应用程序抵御中断或从中断中恢复的能力。在 AWS 云中规划韧性时，高可用性和灾难恢复是常见的考虑因素。有关更多信息，请参阅 AWS 云 韧性。

基于资源的策略

一种附加到资源的策略，例如 AmazonS3 存储桶、端点或加密密钥。此类策略指定了允许哪些主体访问、支持的操作以及必须满足的任何其他条件。

责任、问责、咨询和知情（RACI）矩阵

定义参与迁移活动和云运营的所有各方的角色和责任的矩阵。矩阵名称源自矩阵中定义的责任类型：负责（R）、问责（A）、咨询（C）和知情（I）。支持（S）类型是可选的。如果包括支持，则该矩阵称为 RASCI 矩阵，如果将其排除在外，则称为 RACI 矩阵。

响应性控制

一种安全控制，旨在推动对不良事件或偏离安全基线的情况进行修复。有关更多信息，请参阅在AWS上实施安全控制中的响应性控制。

保留

请参阅 7 R。

停用

请参阅 7 R。

检索增强生成（RAG）

一种生成式人工智能技术，其中 LLM 在生成响应之前引用其训练数据来源之外的权威数据来源。例如，RAG 模型可以对组织的知识库或自定义数据执行语义搜索。有关更多信息，请参阅什么是 RAG。

轮换

定期更新密钥以使攻击者更难访问凭证的过程。

行列访问控制（RCAC）

使用已定义访问规则的基本、灵活的 SQL 表达式。RCAC 由行权限和列掩码组成。

RPO

请参阅恢复点目标。

RTO

请参阅恢复时间目标。

运行手册

执行特定任务所需的一套手动或自动程序。它们通常是为了简化重复性操作或高错误率的程序而设计的。

S

SAML 2.0

许多身份提供商 (IdPs) 使用的开放标准。此功能支持联合单点登录 (SSO)，因此用户无需在 IAM 中为组织中的所有人创建用户即可登录 AWS 管理控制台 或调用 AWS API 操作。有关基于 SAML 2.0 的联合身份验证的更多信息，请参阅 IAM 文档中的关于基于 SAML 2.0 的联合身份验证。

SCADA

请参阅监督控制和数据采集。

SCP

请参阅服务控制策略。

机密密钥

在中 AWS Secrets Manager，您以加密形式存储的机密或受限信息，例如密码或用户凭证。它由密钥值及其元数据组成。密钥值可以是二进制、单个字符串或多个字符串。有关更多信息，请参阅 Secrets Manager 文档中的什么是 Amazon Secrets Manager 密钥？。

安全设计

一种在整个开发过程中都考虑安全的系统工程方法。

安全控制

一种技术或管理防护机制，可防止、检测或降低威胁行为体利用安全漏洞的能力。安全控制有以下四种类型：预防性、检测性、响应性和主动性。

安全固化

缩小攻击面，使其更能抵御攻击的过程。这可能包括删除不再需要的资源、实施授予最低权限的最佳安全实践或停用配置文件中不必要的功能等操作。

安全信息和事件管理（SIEM）系统

结合了安全信息管理（SIM）和安全事件管理（SEM）系统的工具和服务。SIEM 系统会收集、监控和分析来自服务器、网络、设备和其他来源的数据，以检测威胁和安全漏洞，并生成警报。

安全响应自动化

一种预定义的程序化操作，旨在自动响应或修复安全事件。这些自动化可作为侦探或响应式安全控制措施，帮助您实施 AWS 安全最佳实践。自动响应操作的示例包括修改 VPC 安全组、修补 Amazon EC2 实例或轮换凭证。

服务器端加密

由接收数据的人在目的地对数据 AWS 服务 进行加密。

服务控制策略（SCP）

一种策略，用于集中控制组织中所有账户的权限 AWS Organizations。 SCPs 定义防护措施或限制管理员可以委托给用户或角色的操作。您可以使用 SCPs 允许列表或拒绝列表来指定允许或禁止哪些服务或操作。有关更多信息，请参阅 AWS Organizations 文档中的服务控制策略。

服务端点

的入口点的 URL AWS 服务。您可以使用端点，通过编程方式连接到目标服务。有关更多信息，请参阅 AWS 一般参考 中的 AWS 服务 端点。

服务水平协议（SLA）

一份协议，阐明了 IT 团队承诺向客户交付的内容，比如服务正常运行时间和性能。

服务水平指示器（SLI）

对服务性能方面的衡量，例如错误率、可用性或吞吐量。

服务水平目标（SLO）

代表服务运行状况的目标指标，由服务水平指示器衡量。

责任共担模式

描述您在云安全与合规方面共同承担 AWS 的责任的模型。 AWS 负责云的安全，而您则负责云中的安全。有关更多信息，请参阅责任共担模式。

SIEM

请参阅安全信息和事件管理系统。

单点故障（SPOF）

应用程序的单个关键组件出现故障，可能会中断系统。

SLA

请参阅服务水平协议。

SLI

请参阅服务水平指示器。

SLO

请参阅服务水平目标。

split-and-seed 模型

一种扩展和加速现代化项目的模式。随着新功能和产品发布的定义，核心团队会拆分以创建新的产品团队。这有助于扩展组织的能力和服务，提高开发人员的工作效率，支持快速创新。有关更多信息，请参阅在 AWS 云中实现应用程序现代化的分阶段方法。

SPOF

请参阅单点故障。

星型架构

一种数据库组织结构，它使用一个大型事实表来存储事务数据或测量数据，并使用一个或多个较小的维度表来存储数据属性。此结构专为在数据仓库中使用或用于商业智能目的而设计。

strangler fig 模式

一种通过逐步重写和替换系统功能直至可以停用原有的系统来实现单体系统现代化的方法。这种模式用无花果藤作为类比，这种藤蔓成长为一棵树，最终战胜并取代了宿主。该模式是由 Martin Fowler 提出的，作为重写单体系统时管理风险的一种方法。有关如何应用此模式的示例，请参阅使用容器和 Amazon API Gateway 逐步将原有的 Microsoft ASP.NET（ASMX）Web 服务现代化。

子网

您的 VPC 内的一个 IP 地址范围。子网必须位于单个可用区中。

监督控制和数据采集（SCADA）

在制造业中，一种使用硬件和软件来监控实物资产和生产操作的系统。

对称加密

一种加密算法，它使用相同的密钥来加密和解密数据。

综合测试

以模拟用户交互的方式测试系统，以检测潜在问题或监控性能。你可以使用 Amazon S CloudWatch ynthetics 来创建这些测试。

系统提示

一种为 LLM 提供上下文、说明或准则以指导其行为的技术。系统提示有助于设置上下文并制定与用户交互的规则。

T

标签

键值对，用作组织资源的元数据。 AWS 标签有助于您管理、识别、组织、搜索和筛选 资源。有关更多信息，请参阅标记您的 AWS 资源。

目标变量

您在监督式 ML 中尝试预测的值。这也被称为结果变量。例如，在制造环境中，目标变量可能是产品缺陷。

任务列表

一种通过运行手册用于跟踪进度的工具。任务列表包含运行手册的概述和要完成的常规任务列表。对于每项常规任务，它包括预计所需时间、所有者和进度。

测试环境

请参阅环境。

训练

为您的 ML 模型提供学习数据。训练数据必须包含正确答案。学习算法在训练数据中查找将输入数据属性映射到目标（您希望预测的答案）的模式。然后输出捕获这些模式的 ML 模型。然后，您可以使用 ML 模型对不知道目标的新数据进行预测。

中转网关

一个网络传输中心，可用于将您的网络 VPCs 和本地网络互连。有关更多信息，请参阅 AWS Transit Gateway 文档中的什么是公交网关。

基于中继的工作流程

一种方法，开发人员在功能分支中本地构建和测试功能，然后将这些更改合并到主分支中。然后，按顺序将主分支构建到开发、预生产和生产环境。

可信访问权限

向您指定的服务授予权限，该服务可代表您在其账户中执行任务。 AWS Organizations 当需要服务相关的角色时，受信任的服务会在每个账户中创建一个角色，为您执行管理任务。有关更多信息，请参阅 AWS Organizations 文档中的AWS Organizations 与其他 AWS 服务一起使用。

优化

更改训练过程的各个方面，以提高 ML 模型的准确性。例如，您可以通过生成标签集、添加标签，并在不同的设置下多次重复这些步骤来优化模型，从而训练 ML 模型。

双披萨团队

一个小 DevOps 团队，你可以用两个披萨来喂食。双披萨团队的规模可确保在软件开发过程中充分协作。

U

不确定性

这一概念指的是不精确、不完整或未知的信息，这些信息可能会破坏预测式 ML 模型的可靠性。不确定性有两种类型：认知不确定性是由有限的、不完整的数据造成的，而偶然不确定性是由数据中固有的噪声和随机性导致的。有关更多信息，请参阅量化深度学习系统中的不确定性指南。

无差别任务

也称为繁重工作，即创建和运行应用程序所必需的工作，但不能为最终用户提供直接价值或竞争优势。无差别任务的示例包括采购、维护和容量规划。

上层环境

请参阅环境。

V

vacuum 操作

一种数据库维护操作，包括在增量更新后进行清理，以回收存储空间并提高性能。

版本控制

跟踪更改的过程和工具，例如存储库中源代码的更改。

VPC 对等连接

两者之间的连接 VPCs ，允许您使用私有 IP 地址路由流量。有关更多信息，请参阅 Amazon VPC 文档中的什么是 VPC 对等连接。

漏洞

损害系统安全的软件缺陷或硬件缺陷。

W

热缓存

一种包含经常访问的当前相关数据的缓冲区缓存。数据库实例可以从缓冲区缓存读取，这比从主内存或磁盘读取要快。

暖数据

不常访问的数据。查询此类数据时，通常可以接受中速查询。

窗口函数

一种对与当前记录有某种关联的一组行执行计算的 SQL 函数。窗口函数对于处理任务很有用，例如计算移动平均值或根据当前行的相对位置访问行的值。

工作负载

一系列资源和代码，它们可以提供商业价值，如面向客户的应用程序或后端过程。

工作流

迁移项目中负责一组特定任务的职能小组。每个工作流都是独立的，但支持项目中的其他工作流。例如，组合工作流负责确定应用程序的优先级、波次规划和收集迁移元数据。组合工作流将这些资产交付给迁移工作流，然后迁移服务器和应用程序。

WORM

请参阅一次写入多次读取。

WQF

请参阅 AWS 工作负载资格鉴定框架。

一次写入多次读取（WORM）

一种存储模型，可一次写入数据并防止数据被删除或修改。授权用户可以根据需要多次读取数据，但无法对其进行更改。此数据存储基础设施被认为不可变。

Z

零日漏洞利用

一种利用零日漏洞的攻击，通常为恶意软件。

零日漏洞

生产系统中不可避免的缺陷或漏洞。威胁主体可能利用这种类型的漏洞攻击系统。开发人员经常因攻击而意识到该漏洞。

零样本提示

为 LLM 提供执行任务的说明，但没有可以帮助指导的示例（样本）。LLM 必须使用预先训练的知识来处理任务。零样本提示的有效性取决于任务的复杂性和提示的质量。另请参阅少样本提示。

僵尸应用程序

平均 CPU 和内存使用率低于 5% 的应用程序。在迁移项目中，通常会停用这些应用程序。

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。