代理人工智能的基础 AWS

AWS 规范性指导

AWS 规范性指导: 代理人工智能的基础 AWS

Table of Contents

代理人工智能的基础 AWS

Aaron Sempf，亚马逊 Web Services

2025 年 7 月（文档历史记录）

在日益智能、分布式和自治系统的世界中，代理的概念——一个能够感知其环境、推理其状态并按意图行事的实体——已成为基础。代理不仅仅是执行指令的程序；它们是以目标为导向、具有上下文感知能力的实体，代表用户、系统或组织做出决策。它们的出现反映了你构建和思考软件的方式的转变：从程序逻辑和被动自动化向具有自主权和目标运行的系统的转变。

人工智能、分布式系统和软件工程的交汇处隐藏着一种被称为代理人工智能的强大范式。新一代智能系统由能够进行自适应行为、复杂协调和委托决策的软件代理组成。

本指南介绍了定义现代软件代理的原则，并概述了它们向代理人工智能的演变。为了解释这种转变，该指南提供了概念背景，然后追溯了软件代理向代理人工智能的演变：

- 软件代理简介定义了软件代理，将其与传统的软件组件进行了比较，并介绍了通过借鉴已建立的框架将代理行为与传统自动化区分开来的基本特征。
- 软件代理的目的探讨了软件代理存在的原因、它们扮演的角色、他们解决的问题以及它们如何在动态环境中实现智能委派、减少认知负荷和支持自适应行为。
- 软件代理的演变追溯了塑造软件代理的智力和技术里程碑，从早期的自治和并发概念到多代理系统和形式代理架构的出现，从而与生成式人工智能融合。
- 代理人工智能的软件代理引入了代理人工智能，这是数十年来进展的结晶，它将分布式代理模型与基础模型、无服务器计算和编排协议相结合。本节介绍这种融合如何支持新一代智能的、使用工具的代理，这些代理在大规模自治、异步和真正的代理下运行。

目标受众

本指南专为架构师、开发人员和技术领导者而设计，他们希望在将软件代理应用于现代云解决方案之前，了解软件代理向代理人工智能的历史、主要概念和演变。 AWS

目标

采用代理架构可以帮助组织：

- 缩短实现价值的时间：自动化和扩展知识工作，减少手动工作和延迟。

- 提高客户参与度：跨域提供智能助手。
- 降低运营成本：自动化以前需要人工输入或监督的决策流程。
- 推动创新和差异化：打造能够实时适应、学习和竞争的智能产品。
- 实现传统工作流程的现代化：将脚本和整体重构为模块化推理代理。

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。

软件代理简介

软件代理的概念已从1960年代的自治实体基础发展到20世纪90年代初的正式探索，已经有了长足的发展。随着数字系统变得越来越复杂（从确定性脚本到自适应的智能应用程序），软件代理已成为在计算系统中实现自主、情境感知和目标驱动行为的必不可少的基石。在云原生和人工智能增强架构的背景下，尤其是随着生成式人工智能、大型语言模型 (LLMs) 和 Amazon Bedrock 等平台的出现，软件代理正在通过新的能力和规模视角重新定义。

本导言摘自 Hyacinth S. Nwana 的开创性著作《软件代理：概述》（Nwana 1996）。它定义了软件代理，讨论了它们的概念根源，并将讨论扩展到当代框架中，定义了现代软件代理的三个总体原则：自主性、异步性和代理性。这些原则将软件代理与其他类型的服务或应用程序区分开来，使这些代理能够在分布式的实时环境中以有目的、弹性和智能的方式运行。

本节内容

从自治到分布式智能

在软件代理一词进入主流之前，早期的计算研究探讨了自主数字实体的概念，即能够独立行动、对输入做出反应并根据内部规则或目标做出决策的系统。这些早期的想法为后来成为代理范式奠定了概念基础。（有关历史时间表，请参阅本指南后面的 “软件代理的演变” 一节。）

早期的自治概念

几十年来，独立于人类操作员的机器或程序的概念一直吸引着系统设计人员。控制论、人工智能和控制系统的早期研究研究了软件如何表现出自我调节行为、动态响应变化以及在没有持续的人为监督的情况下运行。

这些想法将自主性引入了智能系统的核心属性，为软件的出现奠定了基础，这种软件可以做出决定和采取行动，而不仅仅是做出反应或执行。

actor 模型和异步执行

1970年代，《人工智能的通用模块化ACTOR形式主义》（Hewitt等人，1973年）中引入的演员模型为思考去中心化、消息驱动的计算提供了一个正式的框架。在此模型中，参与者是独立的实体，它们仅通过传递异步消息进行通信，并支持可扩展、并发和容错的系统。

演员模型强调了继续影响现代代理设计的三个关键属性：

- 状态和行为的隔离
- 实体之间的异步交互
- 动态创建和委派任务

这些属性符合分布式系统的需求，并预先配置了云原生环境中软件代理的操作特征。

分布式情报和多智能体系统

20世纪60年代后，随着计算系统的互联程度越来越高，研究人员探索了分布式人工智能（DAI）。该领域侧重于多个自治实体如何在系统中协同工作或竞争合作。DAI催生了多智能体系统的发展，在这个系统中，每个代理都有局部目标、感知和推理，但也可以在更广泛、相互关联的环境中运行。

这种分布式智能愿景仍然是构思和构建现代代理系统的核心，在这种愿景中，决策是去中心化的，而新出现的行为则源于代理的互动。

Nwana 的类型学和软件代理的兴起

1990年代中期，软件代理概念的正式化标志着智能系统演变的转折点。对这种形式化最具影响力的贡献之一是Hyacinth S. Nwana的开创性论文《软件代理：概述》（Nwana 1996），该论文提供了最早对各个维度的软件代理进行分类和理解的综合框架之一。

在这篇论文中，Nwana调查了软件代理研究的现状，并发现代理的定义和实施方式差异越来越大。该论文强调了需要一个共同的概念框架，并提出了一种根据代理的关键能力对其进行分类的类型学。它回顾了学术界和工业界的代表性代理系统，将代理与传统程序和对象区分开来，并概述了基于代理的计算所面临的挑战和机遇。

Nwana强调说，软件代理不是一个单一的概念，而是以各种复杂性和功能存在的。该类型学有助于阐明这种格局并指导未来的设计和研究。

Nwana将软件代理定义为在特定环境中持续自主运行的软件实体，该环境通常由其他代理和进程居住。该定义强调了两个关键特征：

- 连续性：该代理会随着时间的推移持续运行，无需持续的人为干预。
- 自主权：代理人有能力根据其对环境的感知做出决定并独立采取行动。

该定义与Nwana的代理类型学相结合，强调委托权限（通过自主权）和主动性是代理的基本特征。它通过突出代理和子例程或服务来区分代理和子例程或服务，代表另一个实体独立行动，以及为追求目标而发起行为的能力，而不仅仅是响应直接命令。

Nwana 的特工类型学

为了进一步区分不同类型的药物，Nwana引入了基于六个关键属性的分类系统：

- 自主权：代理无需人类或其他人的直接干预即可运行。
- 社交能力：代理人使用沟通机制与其他代理人或人类互动。
- 反应性：代理感知其环境并及时做出响应。
- 主动性：代理主动采取行动，表现出以目标为导向的行为。
- 适应性和学习能力：随着时间的推移，代理会通过经验提高其性能。
- 移动性：代理可以在不同的系统环境或网络之间移动。

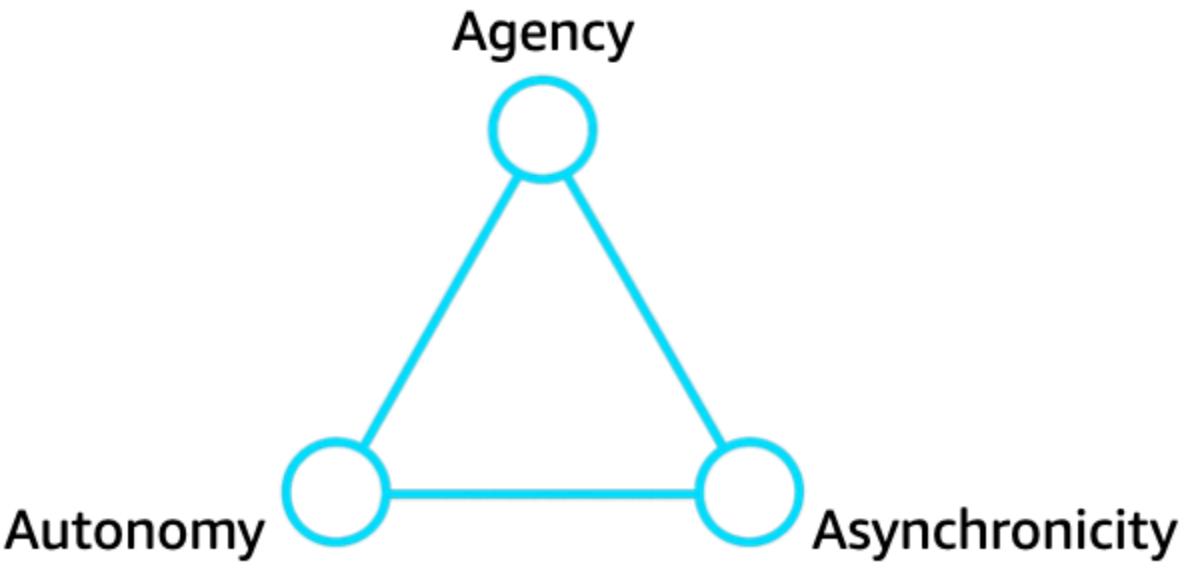
从类型学到现代代理原理

Nwana的工作既是分类法，也是基础视角，计算机界可以通过它来评估软件领域不断演变的代理形式。他强调自主权、主动性以及代表用户或系统行事的概念，为我们现在所认为的代理行为奠定了基础。

尽管技术和环境发生了变化，尤其是随着生成式人工智能、无服务器基础设施和多代理编排框架的兴起，但Nwana工作的基础见解仍然具有重要意义。它们在早期代理理论和软件代理的三大现代支柱之间架起了关键的桥梁。

现代软件代理的三大支柱

在当今人工智能驱动的平台、微服务架构和事件驱动系统的背景下，软件代理可以通过三个相互依赖的原则来定义，这些原则将它们与标准服务或自动化脚本区分开来：自治、异步性和代理。在下图和随后的图表中，三角形代表了现代软件代理的这三个支柱。



自治

现代代理独立运作。他们根据内部状态和环境背景做出决策，无需人工提示。这使他们能够实时对数据做出反应，管理自己的生命周期，并根据目标和情境输入调整自己的行为。

自主权是代理行为的基础。它允许代理在没有持续监督或硬编码控制流的情况下运行。

异步性

代理基本上是异步的。这意味着它们可以在事件、信号和刺激发生时做出反应，而不必依赖阻塞呼叫或线性工作流程。此特性支持可扩展、无阻塞的通信、分布式环境中的响应能力以及组件之间的松散耦合。

通过异步性，代理可以参与实时系统并流畅高效地与其他服务或代理进行协调。

以代理为决定性原则

自治和异步性是必要的，但仅凭这些功能不足以使系统成为真正的软件代理。关键的差异化因素是代理，它引入了：

- 以目标为导向的行为：代理人追求目标并评估实现目标的进展情况。
- 决策：代理根据规则、模型或习得的策略评估选项并选择行动。

- 委托意图：代理代表个人、系统或组织行事，具有内在的目标感。
- 情境推理：代理整合其环境的记忆或模型，以智能地指导行为。

自主和异步的系统可能仍然是一种被动服务。它之所以成为软件代理，是因为它能够有意图和目的行事，具有代理能力。

有目的的机构

自治、异步和代理原则使系统能够在分布式环境中智能、自适应和独立地运行。这些原则植根于数十年的概念和架构演变，现在是当今正在建造的许多最先进的人工智能系统的基础。

在这个生成式人工智能、以目标为导向的编排和多代理协作的新时代，了解是什么使软件代理真正具有代理作用至关重要。将代理视为决定性特征可以帮助我们超越自动化，有目的地进入自主智能领域。

软件代理的用途

随着现代系统变得越来越复杂、分布式和智能，从自主操作到用户辅助技术，软件代理的作用越来越突出。但是软件代理的根本目的是什么？ 为什么我们设计的系统超越了脚本、服务或静态模型，而是将任务委托给能够感知、推理和行动的实体？

本节探讨了软件代理的基本目的：在动态环境中实现任务的智能分配，重点是自主性、适应性和有针对性的行动。它介绍了软件代理的概念基础，追踪了他们的认知结构，并概述了它们具有独特能力解决的现实世界问题。

本节内容

从演员模特到经纪人认知

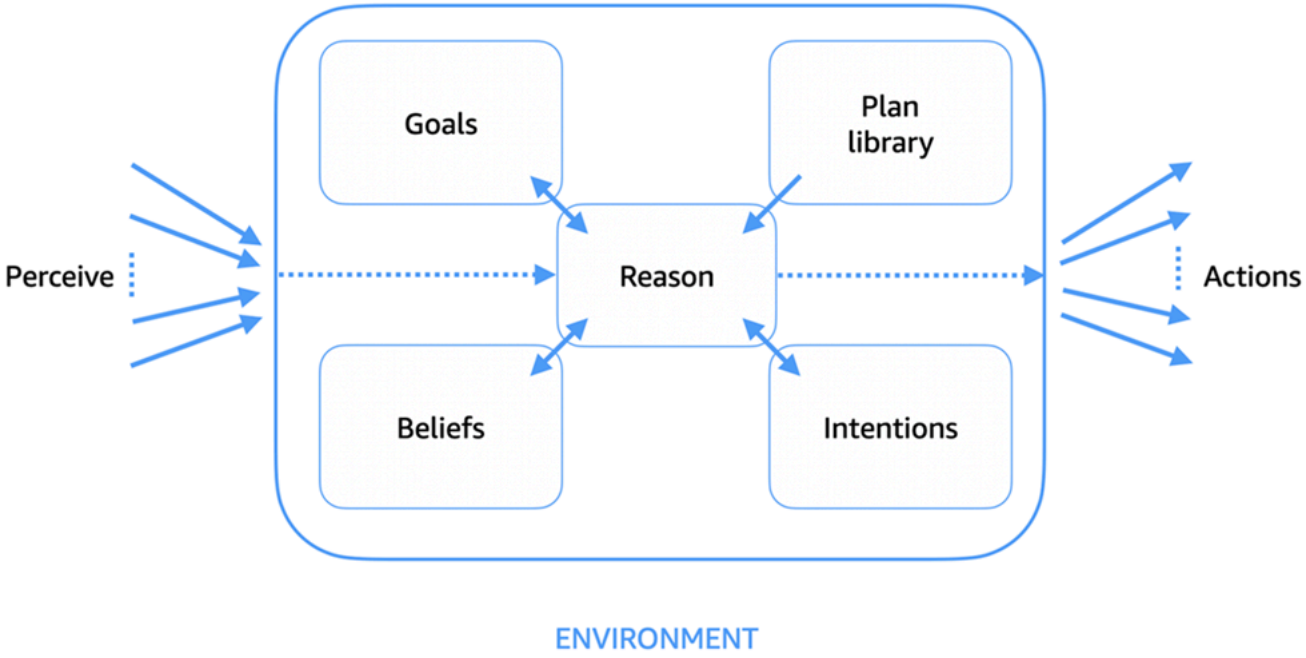
软件代理的目的和结构以早期计算模型中出现的想法为基础，尤其是卡尔·休伊特在1970年代推出的演员模型（Hewitt等人，1973年）。

actor 模型将计算视为一组独立的、同时执行的实体，称为 actor。每个参与者都封装自己的状态，仅通过异步消息传递进行交互，并且可以创建新的参与者并委托任务。

该模型为去中心化推理、反应性和孤立提供了概念基础，所有这些都支撑了现代软件代理的行为架构。

代理功能：感知、推理、行动

每个软件代理的核心都是一个认知周期，通常被描述为感知、理性、行为循环。此过程如下图所示。它定义了代理如何在动态环境中自主运行。



- 感知：代理从环境中收集信息（例如事件、传感器输入或 API 信号），并更新其内部状态或信念。
- 原因：代理使用计划库或逻辑系统分析当前的信念、目标和情境知识。此过程可能涉及确定目标优先顺序、冲突解决或意图选择。
- 行动：代理选择并执行使他们更接近实现其委派目标的操作。

这种架构支持代理在严格编程之外发挥作用的能力，并支持灵活、上下文敏感和目标导向的行为。它构成了指导软件代理更广泛目的的心理框架。

自主协作和意图性

软件代理的目的是为现代计算带来自主权、情境感知和智能委派。由于代理建立在行为者模型的原则之上，体现在感知、理性、行为周期中，因此它们使系统不仅是被动的，而且是积极主动和有目的的。

代理使软件能够在复杂的环境中做出决定、调整和采取行动。它们代表用户，解释目标，并以机器速度执行任务。随着我们更深入地进入代理人工智能时代，软件代理正在成为人类意图和智能数字行动之间的操作接口。

委托意图

与传统的软件组件不同，软件代理的存在是为了代表其他事物行事：用户、另一个系统或更高级别的服务。它们带有委托意图，这意味着他们：

- 启动后独立操作。
- 做出与委托人目标一致的选择。
- 驾驭执行中的不确定性和权衡取舍。

代理弥合了指令和结果之间的差距，允许用户在更高的抽象层面上表达意图，而不必要求明确的指令。

在动态、不可预测的环境中运行

软件代理专为条件不断变化、数据实时到达、控制和上下文分散的环境而设计。

与需要精确输入或同步执行的静态程序不同，代理会适应周围环境并动态响应。这是云原生基础架构、边缘计算、物联网 (IoT) 网络和实时决策系统中的一项重要功能。

减轻人类的认知负担

软件代理的主要目的之一是减轻人类的认知和操作负担。代理可以：

- 持续监控系统和工作流程。
- 检测并应对预定义或紧急情况。
- 自动执行重复性、高容量决策。
- 以最小的延迟对环境变化做出反应。

当决策从用户转移到代理时，系统会变得更具响应能力、更具弹性和以人为本，并且可以实时适应新的信息或干扰。这可以在高度复杂或大规模的环境中加快反应周转速度并提高运营连续性。结果是人类的注意力从微观层面的决策转向了战略监督和创造性地解决问题。

启用分布式智能

软件代理能够单独或集体运行，因此可以设计出跨环境或组织进行协调的多代理系统 (MAS)。这些系统可以智能地分配任务，并朝着综合目标进行协商、合作或竞争。

例如，在全球供应链系统中，个体代理人管理工厂、运输、仓库和最后一英里交付。每个代理商都自主运营：工厂代理根据资源限制优化生产，仓库代理实时调整库存流量，配送代理根据流量和客户可用性重新安排发货路线。

这些代理可以动态通信和协调，无需集中控制即可适应港口延误或卡车故障等中断情况。该系统的整体智能源于这些交互，实现了弹性、优化的物流，超出了单个组件的能力。

在此模型中，代理充当更广泛的情报结构中的节点。它们形成了能够解决任何单一组件都无法单独处理的问题的应急系统。

有目的地行事，而不仅仅是反应

在复杂的系统中，仅靠自动化是不够的。软件代理的决定性目的是有目的地行事，评估目标，权衡背景并做出明智的选择。这意味着软件代理会追求目标，而不仅仅是响应触发因素。他们可以根据经验或反馈修改信念和意图。在这种情况下，信念是指代理根据其感知（输入和传感器）对环境的内部表示（例如，“包裹X在仓库A中”）。意图是指代理为实现目标而选择的计划（例如，“使用配送路径 B 并通知收件人”）。工程师还可以根据需要上报、推迟或调整操作。

这种意图使软件代理不只是被动执行者，而是智能系统中的自主协作者。

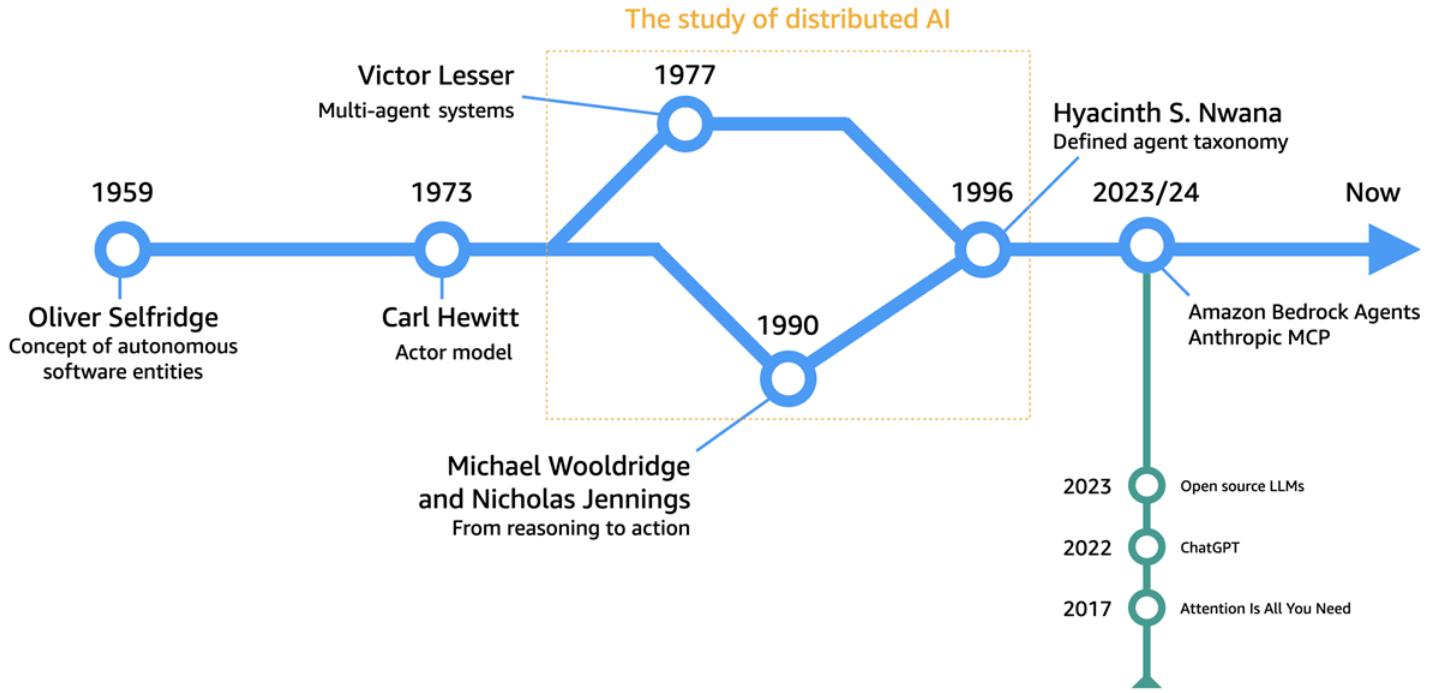
软件代理的演变

从简单的自动化系统到智能、自主和以目标为导向的软件代理的旅程反映了计算机科学、人工智能和分布式系统数十年的发展。

随之而来的是机器学习的兴起，机器学习将范式从手工制作的规则转向了统计模式识别。这些系统可以从数据中学习，推动感知、分类和决策的进步。

大型语言模型 (LLMs) 代表规模、架构和无监督学习的融合。 LLMs 只需很少或根本没有针对特定任务的培训，就可以推理、生成和调整任务。通过 LLMs 与可扩展的云原生基础设施和可组合架构相结合，我们现在正在实现代理人工智能的全部愿景：能够在企业规模上以自主权、情境感知和适应性运行的智能软件代理。

本节探讨了软件代理从基础理论到现代实践的历史，如下图所示。它重点介绍了分布式人工智能 (DAI) 和基于变压器的生成人工智能的融合，并确定了塑造代理人工智能出现的关键里程碑。



本节内容

软件代理的基础

1959 — 奥利弗·塞尔弗里奇：软件自主权的诞生

软件代理的根源可以追溯到奥利弗·塞尔弗里奇（Oliver Selfridge），他引入了自主软件实体（恶魔）的概念，即能够感知环境并独立行动的程序（Selfridge 1959）。他在机器感知和学习领域的早期工作为将来将代理视为独立、智能系统的概念奠定了哲学基础。

1973 — 卡尔·休伊特：演员模特

卡尔·休伊特的演员模型（Hewitt等人，1973年）取得了关键的进步，这是一个将代理描述为独立的并发实体的正式计算模型。在此模型中，代理可以封装自己的状态和行为，使用异步消息传递进行通信，并动态创建其他参与者并将任务委托给他们。

行为者模型为分布式、基于代理的系统提供了理论基础和架构范式。该模型预先配置了现代并发实现，例如Erlang编程语言和Akka框架。

成熟领域：从推理到行动

1977 — Victor Lesser：多代理系统

1970年代末，分布式人工智能 (DAI) 应运而生。它得到了维克多·莱瑟尔的支持，他以开创性的多代理系统（MAS）而广受认可。他的工作重点是独立软件实体如何合作、协调和谈判（参见 “资源” 部分）。这一发展催生了能够集体解决复杂问题的系统，这是构建分布式智能的重要飞跃。

1990 年代 — 迈克尔·伍尔德里奇和尼古拉斯·詹宁斯：特工频谱

到20世纪90年代，在迈克尔·伍尔德里奇和尼古拉斯·詹宁斯等研究人员的贡献下，分布式智能领域已经走向成熟。这些学者对代理进行了分类，从被动到深思熟虑，从非认知系统到目标驱动的推理代理人（Wooldridge and Jennings 1995）。他们的工作强调，代理不再是抽象的想法，而是被应用于从机器人到企业软件的广泛实践领域。

这些研究人员还引入了重点的转移：从集中推理转向分布式行动。代理人不再只是思想家，他们是在实时环境中运作的具有自主权和目标的行动者。

1996 年 — Hyacinth S. Nwana：正式确定代理概念

1996年，Hyacinth S. Nwana发表了颇具影响力的论文《软件代理：概述》，该论文提供了迄今为止最全面的代理分类。他的类型学包括自主权、社交能力、反应性、主动性、学习和移动性等属性，并区分了软件代理和传统软件结构。

Nwana还给出了一个现在被广泛接受的定义，解释如下：软件代理是一种基于软件的计算机程序，在代理关系中为用户或其他程序行事，它源于委托的概念。

这种形式化有助于将软件代理从理论结构过渡到现实世界的应用程序。它催生了一代基于代理的系统，涉及电信、工作流程自动化和智能助手等领域。

Nwana的工作是早期分布式人工智能研究和现代代理运营架构的交汇点。它是代理认知理论与其在当今系统中的实际部署之间的关键桥梁。

Parallel 时间轴：大型语言模型的兴起

在代理框架不断发展的同时，自然语言处理和机器学习领域正在发生一场并行和趋同的革命：

- 2017 — 变形金刚：论文《注意力就是你所需要的》（Vaswani等人，2017）介绍了变压器架构，它极大地改善了机器处理和生成语言的方式。
- 2022 年 — ChatGPT：OpenAI 在 GPT-3.5 上发布了一个名为 ChatGPT 的基于聊天的界面，该界面支持与通用人工智能系统进行自然的交互式对话。
- 2023 — 开源 LLMs：Llama、Falcon 和 Mistral 的发布使强大的模型可以广泛使用，并加速了开源和企业环境中代理框架的开发。

这些创新将语言模型转变为能够解析上下文、规划操作和链接响应的推理引擎，并 LLMs 成为智能软件代理的关键推动力。

时间表趋于一致：代理人工智能的出现

2023-2024 年 — 企业级代理平台

分布式软件代理架构和基于变压器的融合 LLMs 最终导致了代理人工智能的兴起。

- Amazon Bedrock Agen ts 引入了一种完全托管的方法，通过使用 Amazon Bedrock 的基础模型来构建以目标为导向、使用工具的软件代理。
- Anthropic 的模型上下文协议 (MCP) 定义了一种让大型语言模型访问外部工具、环境和内存并与之交互的方法。这是上下文、持续和自主行为的关键。

这两个里程碑代表了机构和情报的综合。代理不再局限于静态工作流程或僵化的自动化。他们现在可以跨多个步骤进行推理，与工具进行协调 APIs，保持情境状态，并随着时间的推移进行学习和适应。

2025 年 1 月至 6 月 — 扩展了企业能力

2025年上半年，随着新的企业能力，代理人工智能格局显著扩张。2025年2月，Anthropic发布了Claude 3.7 Sonnet，这是市场上第一个混合推理模型，MCP 规范获得了广泛采用。

A mazon Q Developer、Cursor 和 WindSurf 集成的 MCP 等人工智能编码助手，用于标准化代码生成、存储库分析和开发工作流程。2025 年 3 月的 MCP 版本引入了重要的企业就绪功能，包括 OAuth 2.1 安全集成、用于多样化数据访问的扩展资源类型以及通过 Streamable HTTP 增强的连接选项。在此基础上，它于2025年5月 AWS 宣布加入MCP指导委员会，为新的 agent-to-agent通信能力做出贡献。这进一步巩固了该协议作为代理人工智能互操作性的行业标准的地位。

2025年5月，通过开源 Strands Agents框架， AWS 增强了客户构建代理人工智能工作流程的选择。这种独立于提供商且与模型无关的框架使开发人员能够跨平台使用基础模型，同时保持深度服务集成。AWS 正如AWS 开源博客所强调的那样，Strands Agents遵循模型优先的设计理念，将基础模型置于代理情报的核心。这使客户可以更轻松地为其特定用例构建和部署复杂的 AI 代理。

崛起 — 代理人工智能

软件代理的演变，从早期的自主思想到支持LLM的现代编排，是漫长而分层的。从奥利弗·塞尔弗里奇（Oliver Selfridge）感知程序的愿景开始，现已发展成为一个由智能、情境感知、目标驱动的软件代理组成的强大生态系统，可以协作、适应和推理。

分布式人工智能 (DAI) 和基于变压器的生成式人工智能的融合标志着一个新时代的开始，在这个新时代中，软件代理不再只是工具，而是智能系统中的自主参与者。

Agentic AI 代表了软件系统的下一次演变。它提供了一类智能代理，这些代理是自主的、异步的和代理的，可以按照委托的意图行事，在动态的分布式环境中有目的地运行。Agentic AI 统一了以下内容：

- 多智能体系统的架构谱系和演员模型
- 感知、理性、行为的认知模型
- 变压器的产生 LLMs 功率
- 云原生计算和无服务器计算的操作灵活性

从软件代理到代理人工智能

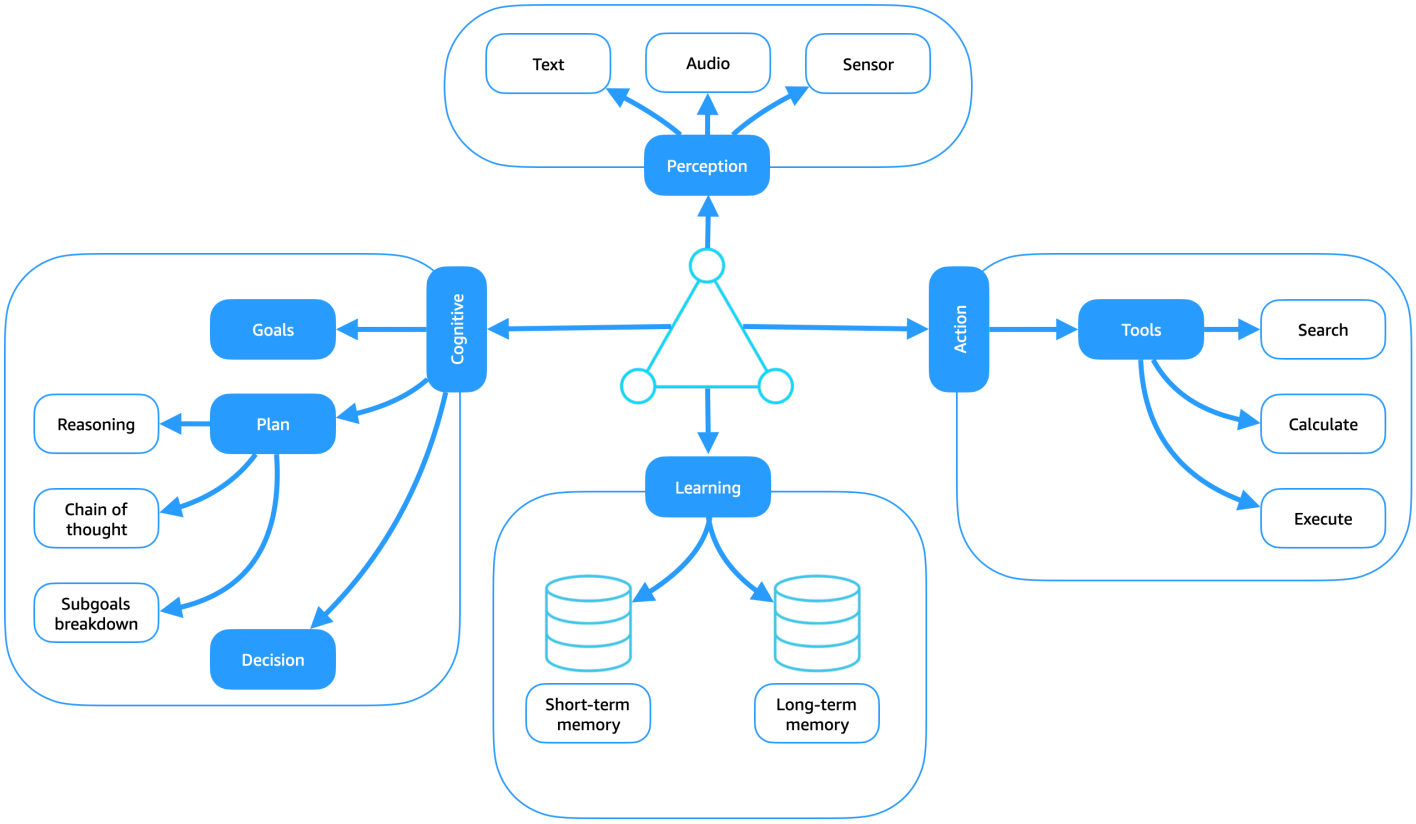
软件代理是自主的数字实体，旨在感知自己的环境，推断自己的目标，并采取相应的行动。与遵循固定逻辑的传统软件程序不同，代理会根据上下文输入和决策框架来调整其行为。这使它们非常适合动态分布式环境，例如云原生系统、机器人、智能自动化以及现在的生成式 AI 编排。

本节介绍了软件代理的核心构建块，并解释了这些组件如何在传统架构中基于感知、理性、行为模型进行交互。它讨论了生成式人工智能，尤其是大型语言模型 (LLMs)，如何改变了软件代理的推理和计划方式。这标志着代理人工智能从基于规则的系统向数据驱动的、学习到的智能发生了根本性的转变。

本节内容

软件代理的核心构建块

下图显示了大多数智能代理中的关键功能模块。每个组件都有助于代理在复杂环境中自主运行。



在感知、理性、行为循环的背景下，代理人的推理能力分布在其认知和学习模块中。通过记忆和学习的整合，代理可以根据过去的经验开发出适应性推理。当代理在其环境中行动时，它会形成一个紧急的反馈循环：每个动作都会影响未来的感知，由此产生的体验通过学习模块整合到记忆和内部模型中。这种持续的感知、推理和行动循环使代理人能够随着时间的推移而改善，并完成完整的感知、理性和行为周期。

感知模块

感知模块使代理能够通过文本、音频和传感器等多种输入模式与其环境进行交互。这些输入构成了所有推理和行动所依据的原始数据。文本输入可能包括自然语言提示、结构化命令或文档。音频输入包括语音指令或环境声音。传感器输入包括物理数据，例如视觉馈送、运动信号或 GPS 坐标。感知的核心功能是从这些原始数据中提取有意义的特征和表现形式。这使代理能够对其当前背景形成准确且可操作的理解。该过程可能涉及特征提取、物体或事件识别以及语义解释，是感知、理性、行为循环中关键的第一步。有效的感知可确保下游推理和决策以相关的 up-to-date态势感知为基础。

认知模块

认知模块是软件代理的深思熟虑的核心。它负责解释感知，形成意图，并通过以目标为导向的计划和决策来指导有目的的行为。该模块将输入转换为结构化推理过程，从而使代理能够有意而不是被动地操作。这些流程通过三个关键子模块进行管理：目标、计划和决策。

目标子模块

目标子模块定义了代理的意图和方向。目标可以是明确的（例如，“导航到某个地点” 或 “提交报告”），也可以是隐含的（例如，“最大限度地提高用户参与度” 或 “最大限度地减少延迟”）。它们是代理人推理周期的核心，为其计划和决策提供了目标状态。

代理人不断评估实现目标的进展情况，并可能根据新的认知或学习重新确定目标的优先顺序或重新制定目标。这种目标感知使代理能够适应动态环境。

规划子模块

规划子模块构造实现代理当前目标的策略。它生成操作序列，按层次分解任务，并从预定义或动态生成的计划中进行选择。

为了在不确定性或不断变化的环境中有效运作，规划不是一成不变的。现代代理可以生成 chain-of-thought序列，引入子目标作为中间步骤，并在条件变化时实时修改计划。

该子模块与记忆和学习紧密相连，允许代理根据过去的结果随着时间的推移完善其计划。

决策子模块

决策子模块评估可用的计划和行动，以选择最合适的下一步行动。它整合了来自感知、当前计划、代理目标和环境背景的输入。

决策是因为：

- 在相互矛盾的目标之间进行权衡
- 置信阈值（例如，感知的不确定性）
- 行动的后果
- 代理人学到的经验

根据架构的不同，代理可能会依靠符号推理、启发式方法、强化学习或语言模型 (LLMs) 来做出明智的决策。此过程使代理的行为具有情境感知能力、目标一致性和适应性。

动作模块

操作模块负责执行代理的选定决策，并与外部世界或内部系统连接以产生有意义的效果。它代表了感知、理性、行为循环的行为阶段，在这个阶段，意图被转化为行为。

当认知模块选择动作时，动作模块通过专门的子模块协调执行，其中每个子模块都与代理的集成环境保持一致：

- 物理驱动：对于嵌入在机器人系统或物联网设备中的代理，该子模块将决策转化为现实世界的物理运动或硬件级别的指令。

  示例：操纵机器人、触发阀门、打开传感器。
- 集成交互：此子模块处理非物理但外部可见的操作，例如与软件系统交互、平台或。 APIs

  示例：向云服务发送命令、更新数据库、通过调用 API 提交报告。
- 工具调用：代理通常通过使用专门的工具来扩展其能力，以完成以下子任务：
  - 搜索：查询结构化或非结构化知识来源
  - 摘要：将大型文本输入压缩为高级概览
  - 计算：执行逻辑、数值或符号计算

  工具调用通过模块化、可调用的技能实现复杂的行为组合。

学习模块

学习模块使代理能够根据经验随着时间的推移进行适应、概括和改进。它利用感知和行动的反馈不断完善代理的内部模型、策略和决策策略，从而为推理过程提供支持。

该模块与短期和长期记忆协调运行：

- 短期记忆：存储瞬态背景，例如对话状态、当前任务信息和最近的观察结果。它可以帮助代理保持交互和任务的连续性。
- 长期记忆：对过去经历中的持久知识进行编码，包括以前遇到的目标、行动结果和环境状态。长期记忆使代理能够识别模式、重复使用策略并避免重复错误。

学习模式

学习模块支持一系列范式，例如监督学习、无监督学习和强化学习，它们支持不同的环境和代理角色：

- 监督学习：根据带标签的示例（通常来自人工反馈或训练数据集）更新内部模型。

  示例：学习根据之前的对话对用户意图进行分类。

- 无监督学习：识别数据中的隐藏模式或结构，无需显式标签。

  示例：对环境信号进行聚类以检测异常。

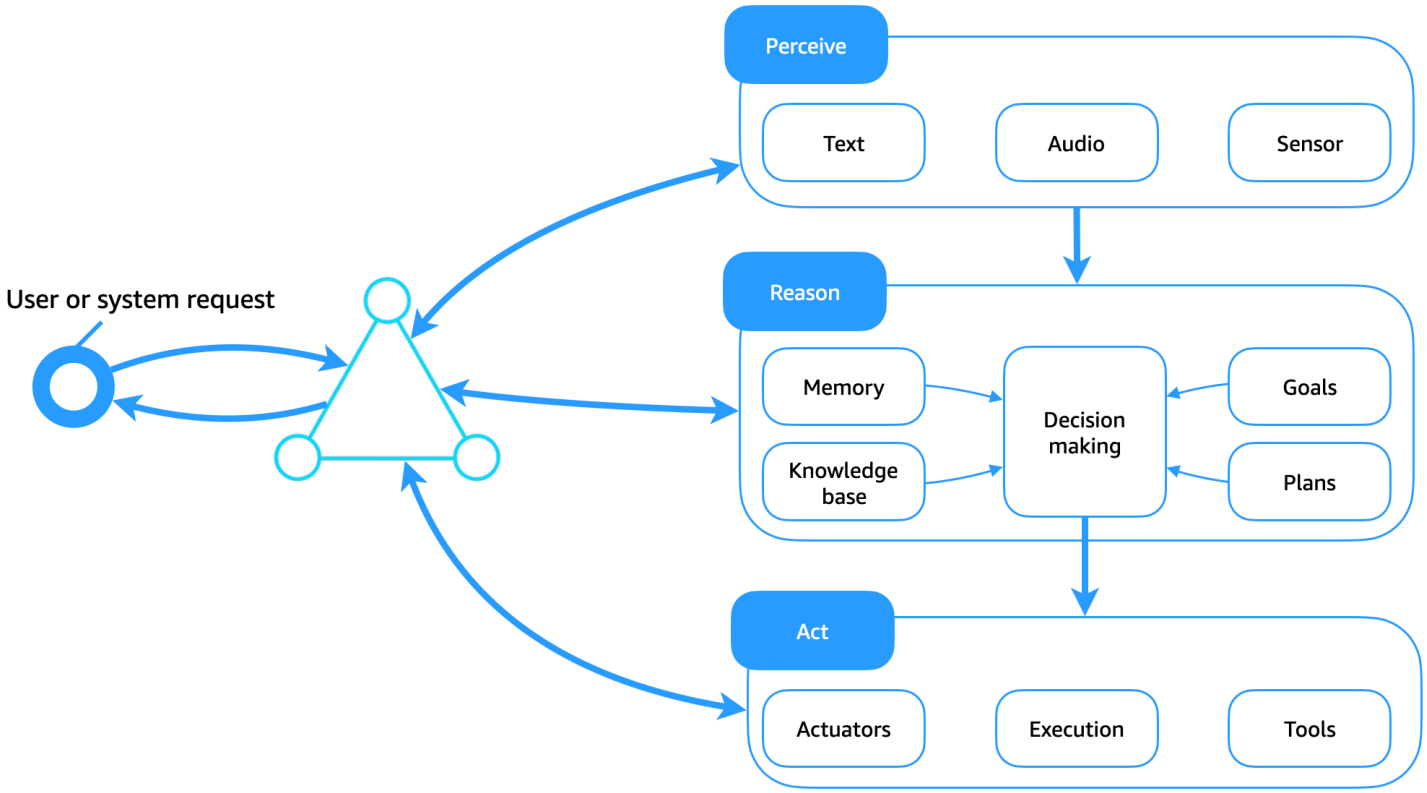
- 强化学习：通过在交互式环境中最大限度地提高累积奖励，通过反复试验优化行为。

  示例：了解哪种策略可以最快地完成任务。

学习与代理人的认知模块紧密集成。它根据过去的结果完善规划策略，通过评估历史成功来加强决策，并不断改善感知与行动之间的映射。通过这种封闭的学习和反馈循环，代理从被动执行演变为能够随着时间的推移适应新的目标、条件和背景的自我完善系统。

传统代理架构：感知、理性、行动

下图说明了上一节中讨论的构建块在感知、理性、行为周期下是如何运作的。



感知模块

感知模块充当代理与外部世界的感官接口。它将原始的环境输入转换为结构化表示，为推理提供信息。这包括处理多模态数据，例如文本、音频或传感器信号。

- 文本输入可能来自用户命令、文档或对话。
- 音频输入包括语音指示或环境声音。
- 传感器输入可捕获真实世界的信号，例如运动、视觉馈送或 GPS。

摄取原始输入后，感知过程会执行特征提取，然后进行物体或事件识别以及语义解释，以创建对当前情况的有意义的模型。这些输出为下游决策提供了结构化的背景，并将代理的推理锚定在现实世界的观测中。

原因模块

原因模块是代理的认知核心。它评估背景，制定意图并确定适当的行动。该模块通过使用所学知识和推理来编排目标驱动的行为。

原因模块由紧密集成的子模块组成：

- 记忆：以短期和长期格式保存对话状态、任务上下文和情节历史记录。
- 知识库：提供对符号规则、本体或学习模型（例如嵌入、事实和策略）的访问权限。
- 目标和计划：定义预期结果并制定实现这些结果的行动策略。可以动态更新目标，也可以根据反馈对计划进行自适应性修改。
- 决策：通过权衡选项、评估权衡和选择下一步行动，充当中央仲裁引擎。该子模块考虑了置信度阈值、目标一致性和情境约束。

这些组件共同使代理能够推理其环境，更新信念，选择路径，并以连贯的适应性方式行事。原因模块缩小了感知和行为之间的差距。

Act 模块

act 模块通过与数字或物理环境接口来执行任务，从而执行代理的选定决策。这就是意图变成行动的地方。

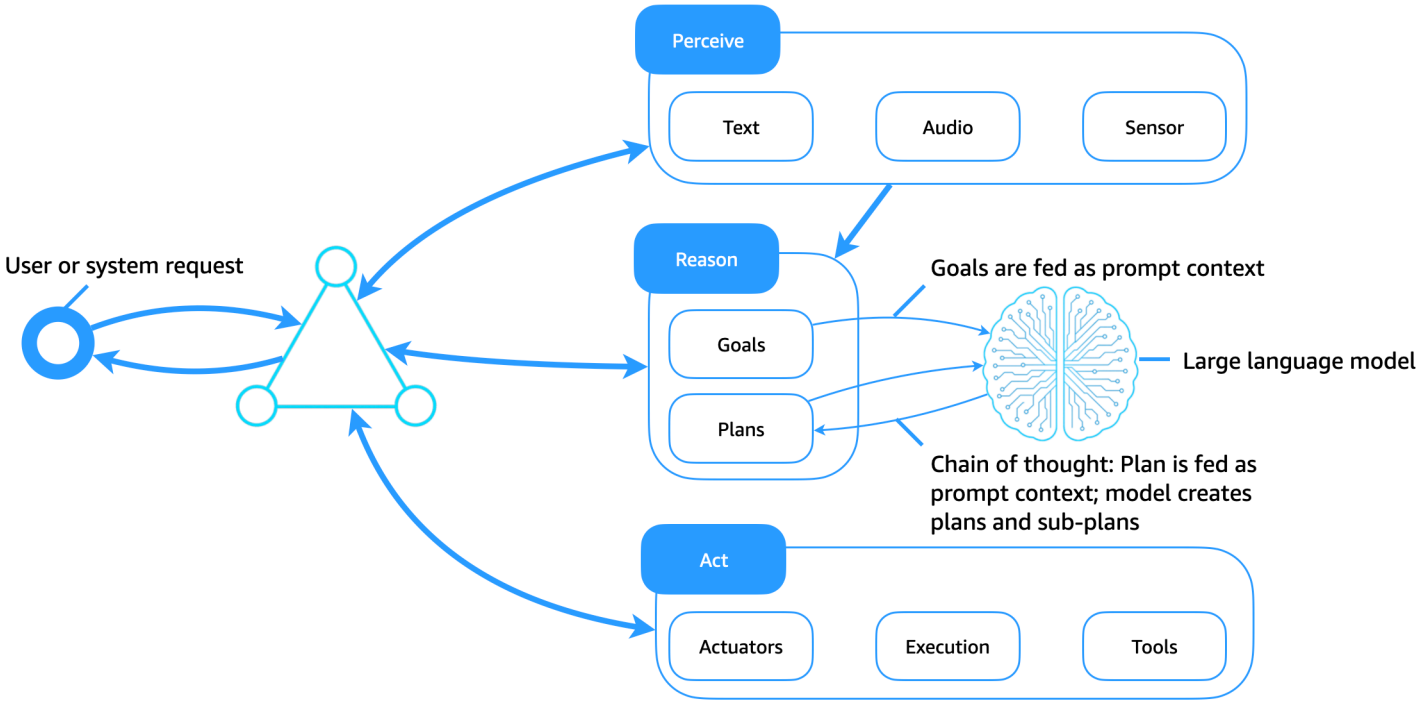
该模块包括三个功能通道：

- 执行器：对于有物理存在的代理（例如机器人和物联网设备），控制硬件级的交互，例如移动、操纵或信号。
- 执行：处理基于软件的操作，包括调用 APIs、调度命令和更新系统。
- 工具：启用诸如搜索、摘要、代码执行、计算和文档处理等功能功能。这些工具通常是动态的和上下文感知的，从而扩展了代理的实用性。

act 模块的输出反馈到环境中并关闭循环。代理会再次感知到这些结果。它们更新代理的内部状态并为未来的决策提供信息，从而完成感知、理性、行为周期。

生成式 AI 代理：将符号逻辑替换为 LLMs

下图说明了大型语言模型 (LLMs) 现在如何作为软件代理的灵活而智能的认知核心。与依赖静态计划库和手工编码规则的传统符号逻辑系统形成鲜明对比的是，它 LLMs支持自适应推理、情境规划和动态工具使用，从而改变代理的感知、推理和行为方式。



主要增强功能

此架构增强了传统的代理架构，如下所示：

- LLMs 作为认知引擎：目标、计划和查询作为即时上下文传递到模型中。法学硕士生成推理路径（例如思维链），将任务分解为子目标，并决定下一步行动。

- 通过提示使用工具： LLMs 可以通过工具使用代理或推理和行动 (ReAct) 提示进行引导，以呼叫 APIs 以及搜索、查询、计算和解释输出。
- 情境感知规划：代理根据代理的当前目标、输入环境和反馈动态生成或修改计划，无需硬编码的计划库。
- 提示上下文作为内存：代理不使用符号知识库，而是将内存、计划和目标编码为传递给模型的提示标记。
- 通过少量的、基于情境的学习来学习：通过及时的工程来 LLMs 调整行为，从而减少对明确的再训练或僵化的计划库的需求。

在基于 LLM 的代理中实现长期记忆

与在结构化知识库中存储长期记忆的传统代理不同，生成式 AI 代理必须在上下文窗口的限制下工作 LLMs。为了扩展内存并支持持久智能，生成式 AI 代理使用了几种补充技术：代理存储、检索增强生成 (RAG)、情境内学习和提示链以及预训练。

代理存储：外部长期存储器

代理状态、用户历史记录、决策和结果存储在长期代理内存存储中（例如矢量数据库、对象存储或文档存储）。按需检索相关内存，并在运行时将其注入到 LLM 提示上下文中。这会创建一个持久的内存循环，在该循环中，代理在会话、任务或交互之间保持连续性。

抹布

RAG 通过将检索到的知识与生成能力相结合，提高 LLM 的绩效。发布目标或查询时，代理会搜索检索索引（例如，通过对文档、之前的对话或结构化知识的语义搜索）。检索到的结果将附加到法学硕士提示中，该提示是在外部事实或个性化背景下生成的。这种方法扩展了代理的有效记忆力，提高了可靠性和事实正确性。

情境内学习和提示链接

代理通过使用会话内令牌上下文和结构化提示链来保持短期记忆。上下文元素（例如当前计划、先前的行动结果和座席状态）在两次通话之间传递以指导行为。

持续的预训练和微调

对于特定域的代理， LLMs 可以继续对自定义集合（例如日志、企业数据或产品文档）进行预训练。或者，通过人工反馈进行指令微调或强化学习 (RLHF) 可以将类似代理的行为直接嵌入到模型中。这会将推理模式从提示时间逻辑转移到模型的内部表示中，缩短了提示长度并提高了效率。

代理人工智能的综合优势

这些技术一起使用时，使生成式 AI 代理能够：

- 随着时间的推移保持情境意识。
- 根据用户历史记录或偏好调整行为。
- 利用 up-to-date事实知识或私人知识做出决定。
- 通过持久、合规且可解释的行为扩展到企业用例。

通过增强 LLMs 外部记忆、检索层和持续训练，代理可以实现以前仅通过符号系统无法实现的认知连续性和目标水平。

将传统 AI 与软件代理和代理人工智能进行比较

下表详细比较了传统 AI、软件代理和代理 AI。

特征	传统人工智能	软件代理	代理式人工智能
示例	垃圾邮件过滤器、图片分类器、推荐引擎	聊天机器人、任务调度器、监控代理	AI 助手、自主开发者代理、多代理 LLM 编排
执行模型	Batch 或同步	事件驱动或定时	异步、事件驱动和目标驱动
自治	有限；通常需要人工或外部编排	中等；在预定义的范围内独立运行	高；使用自适应策略独立行动
反应性	对输入数据有反应	对环境和事件做出反应	被动和主动；预测并启动行动
积极主动	稀有	存在于某些系统中	核心属性；推动以目标为导向的行为
Communication	最小；通常是独立的或 API 绑定的	代理间或代理人与人之间的消息传递	丰富的多代理和 human-in-the-loop交互功能

特征	传统人工智能	软件代理	代理式人工智能
决策	仅限模型推断（分类、预测等）	符号推理，或基于规则的决策或脚本式决策	情境化、基于目标的动态推理（通常是 LLM 增强型）
委托意图	否；执行用户直接定义的任务	部分；代表范围有限的用户或系统行事	是；根据委派的目标行事，通常是跨服务、用户或系统
学习和适应	通常以模型为中心（例如，机器学习训练）	有时是自适应的	嵌入式学习、记忆或推理（例如，反馈、自我纠正）
中介机构	无；人类工具	隐式或基本	明确；有目的、目标和自我指导的运作
情境感知	低；无状态或基于快照	中等；某些状态跟踪	高；使用内存、情境上下文和环境模型
基础设施角色	嵌入在应用程序或分析管道中	中间件或服务层组件	与云端、无服务器或边缘系统集成的可组合代理网格

总而言之：

- 传统的人工智能以工具为中心，功能狭窄。它侧重于预测或分类。
- 传统的软件代理引入了自主权和基本的通信，但它们通常是受规则约束或静态的。
- Agentic AI 汇集了自主权、异步性和代理性。它使智能的、以目标为导向的实体能够在复杂的系统中进行推理、行动和适应。这使得代理人工智能成为云原生、人工智能驱动的未来的理想之选。

后续步骤

本指南讨论了代理人工智能的历史和基础，代理人工智能代表了传统软件代理向由生成式人工智能提供支持的自主智能系统的演变。它描述了早期的软件代理如何遵循预定义的规则和逻辑在固定边界内自动执行任务，并解释了代理人工智能如何通过整合大型语言模型在此基础上构建，使代理能够在开放式环境中动态推理、学习和适应。

您可以通过查看本系列中的以下出版物来深入探索 agentic AI：

- 在上面实施代理人工智能 AWS提供了一种组织策略，可以将代理人工智能从孤立的实验转变为企业规模的创造价值的基础架构。
- 上的 Agentic AI 模式和工作流程 AWS讨论了用于设计、撰写和编排以目标为导向的 AI 代理的基础蓝图和模块化结构。
- 上 AWS面的 Agentic AI 框架、协议和工具涵盖了在构建 Agentic AI 解决方案时需要考虑的软件基础、工具包和协议。
- 为代理人工智能构建无服务器架构 AWS讨论了作为现代 AI 工作负载自然基础的无服务器架构，并描述了如何在中构建 AI 原生无服务器架构。 AWS 云
- 为代理人工智能构建多租户架构 AWS描述了在多租户环境中使用人工智能代理的情况，包括托管注意事项、部署模型和控制平面。

资源

有关本指南中讨论的概念的更多信息，请参阅以下指南和文章。

AWS 参考文献

- Amazon Bedrock 代理
- Amazon Q 开发者版
- Strands Agent

其他参考资料

- 休伊特、卡尔、彼得·毕晓普和理查德·斯泰格。“适用于人工智能的通用模块化ACTOR形式主义。”第三届国际人工智能联席会议论文集（1973）：235-245。 https://www.ijcai。 org/Proceedings/73/Papers/027B.pdf
- Lesser，Victor R.，相关出版物（见完整清单）：
  - Lesser、Victor R. 和 Daniel D. Cor “功能精确的协作分布式系统。” IEEE 关于系统、人和控制论的交易 11，第 1 号（1981）：81-96。 https://ieeexplore.ieee。 org/abstract/document/4308581
  - Decker、Keith S. 和 Victor R. Lesser “协调服务中的沟通。” AAAI 代理间通信规划研讨会（1994 年）。 https://www.researchgate。 net/profile/Victor-Lesser/publication/2768884_Communication_in_the_Service_of_Coordination/links/00b7d51cc2a0750cb4000000/Communication-in-the-Service-of-Coordination.pdf
  - Durfee、Edmund H.、Victor R. Lesser 和 Daniel D. Corkill “合作式分布式问题解决的趋势。” IEEE 知识与数据工程学会刊（1989）。 http://mas.cs.umass。 edu/Documents/ieee-tkde89.pdf
  - Durfee、Edmund H.、V.R. Lesser 和 D.D. Corkill，“分布式人工智能”。分布式问题解决网络中通过沟通进行合作（1987）：29- 58。 https://www.academia。 edu/download/79885643/durf94_1.pdf
  - Láasri、Brigitte、Hassan Láasri、Susan Lander 和 Victor Lesser。“智能谈判代理的通用模型。”《国际合作信息系统杂志》 01，第02期（1992）：291-317。 https://doi.org/10.1142/S0218215792000210
  - 兰德、苏珊 E. 和 Victor R. Lesser “了解协商在异构代理之间的分布式搜索中的作用。” IJCAI'93：第十三届国际人工智能联席会议论文集（1993）：438-444。 https://www.ijcai。 org/Proceedings/93-1/Papers/062.pdf

  - 兰德、苏珊、Victor R. Lesser 和玛格丽特 ·E· 康奈尔。“合作专家代理人的冲突解决策略” CKBS'90：合作知识型系统国际工作会议论文集（1990 年 10 月）：183-200。 https://doi.org/10.1007/978-1-4471-1831-2_10
  - Prasad、M.V. Nagendra、Victor Lesser 和 Susan E. Lander。“异构多智能体系统中的学习实验。” IJCAI-95 多智能体系统适应与学习研讨会（1995）：59-64。 https://www.researchgate.net/publication/2784280_Learning_Experiments_in_a_Heterogeneous_Multi-agent_System
- Nwana，Hyacinth S. “软件代理：概述。”《知识工程评论》 11，no. 3（1996 年 10 月/11 月）：205-244。 https://teaching.shu.ac。 uk/aces/rh1/elearning/multiagents/introduction/nwana.pdf
- Selfridge，Oliver G. “Pandemonium：学习范式。” 思维过程的机械化：在国家物理实验室举行的研讨会论文集 1（1959）：511—529。 https://aitopics。 org/download/classics:504 E1BAC
- Vaswani、Ashish、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N. Gomez、Lukasz Kaiser 和 Illia Polosukhin。“注意力就是你所需要的。” 第31届神经信息处理系统会议（NIPS）论文集。《神经信息处理系统进展》30（2017）：5998-6008。 https://proceedings.neurips。 cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
- 伍尔德里奇、迈克尔和尼古拉斯 ·R· 詹宁斯。“智能代理：理论与实践。”《知识工程评论》 10，不是。2（1995 年 1 月）：115-152。 https://www.cs.cmu.edu/ ~ motionplanning/papers/sbp_papers/integrated1/woodridge _intelligent_agents.pdf

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

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，彼此独立，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。

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。