为代理人工智能构建多租户架构 AWS

AWS 规范性指导

AWS 规范性指导: 为代理人工智能构建多租户架构 AWS

Table of Contents

为代理人工智能构建多租户架构 AWS

Aaron Sempf 和 Tod Golding，Amazon Web Services

2025 年 7 月（文档历史记录）

Agentic AI 代表着一种颠覆性的范式转变，要求组织重新思考如何构建、交付和运营其系统。代理模型让团队探索新的方法，将系统分解为一个或多个代理，从而创造新的道路、可能性和价值。

代理商的许多讨论都围绕着用于构建和实施代理的工具、框架和模式。我们不仅必须采用良好的工具来创建代理，还必须采用新的集成协议、身份验证策略和可以作为代理架构基础的发现机制。

在代理工具数量不断增长的同时，团队还必须考虑其代理如何应对更传统的架构挑战。规模、邻居噪音、弹性、成本和运营效率是设计、构建和部署代理时必须评估的基本主题。无论代理多么自主和智能，我们还必须确保他们实现符合业务需求的规模经济、效率和敏捷性。

本指南的目标是探索代理足迹的各个维度。这包括审查各种代理部署和使用模式，并重点介绍创建可实现架构目标的代理的不同策略。这还意味着通过引入多租户环境中通常需要的内部结构，来研究在多租户环境中如何使用代理。

目标受众

本指南适用于想要构建人工智能驱动的多租户系统的架构师、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：

- 了解多租户代理部署，探索孤立模型和池化模型，以及租户上下文如何影响代理实施
- 探索代理管理，包括跨单供应商和多提供商环境的入职、租户隔离和资源管理
- 评估多租户代理的各个方面，包括数据所有权、监控和测试

关于此内容系列

本指南是一系列出版物的一部分，这些出版物为构建人工智能驱动的软件代理提供了架构蓝图和技术指导。 AWS AWS 规范性指导系列包括以下指南：

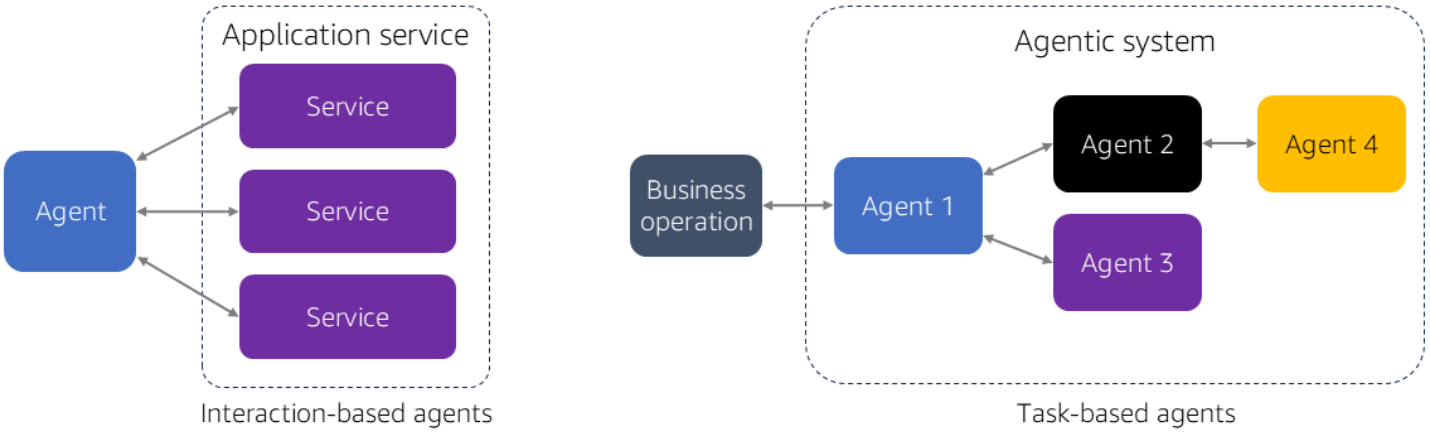
- 在 AI 上运行代理 AI AWS
- 代理人工智能的基础 AWS
- Agentic AI 模式和工作流程已开启 AWS
- Agentic AI 框架、协议和工具已启用 AWS
- 为代理人工智能构建无服务器架构 AWS
- 为代理人工智能构建多租户架构 AWS（本指南）

有关此内容系列的更多信息，请参阅 A gentic AI。

代理基础知识

在讨论架构细节之前，我们应该概述代理所扮演的不同角色，因为“代理”是一个超负荷的术语，可以应用于许多用例。让我们从一些可以帮助对它们进行分类的宽泛术语开始。

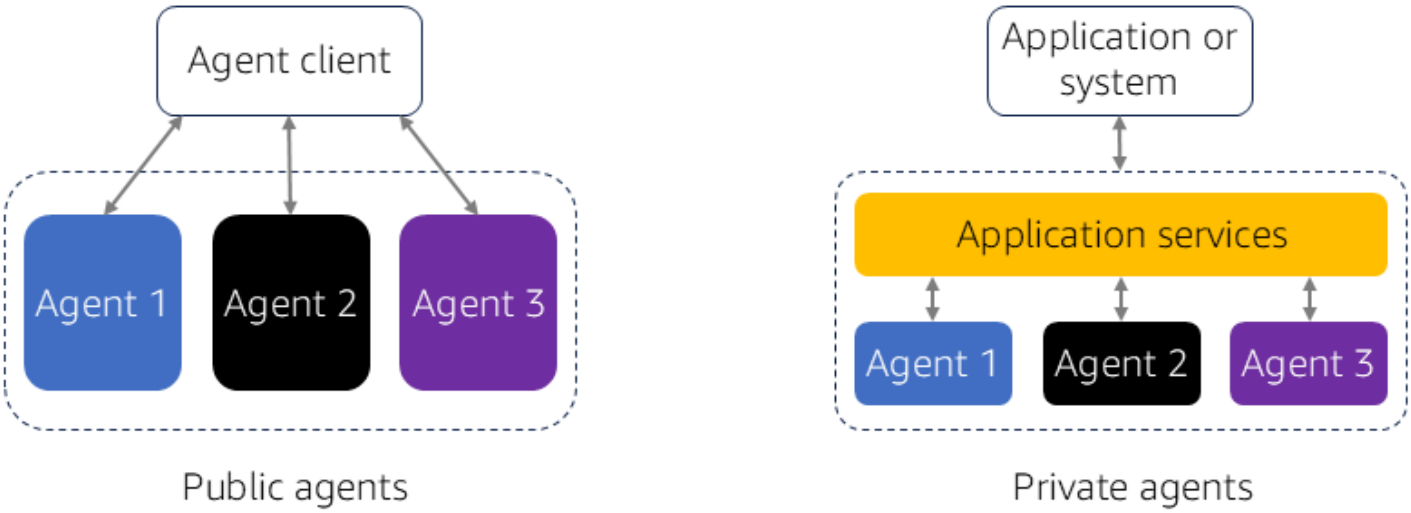
在最外层面，我们需要首先对代理的角色和性质进行分类。这具有挑战性，因为在各种各样的场景中，代理可以应用于任意数量的问题。不过，在本次讨论中，我们将重点介绍在应用程序或系统中引入代理意味着什么。在此模型中，我们强调代理如何以及在何处最好地丰富您的系统体验。您选择的选项会影响代理的构建、集成以及应用于不同域和用例的方式。下图显示了生成器使用的两种代理模式。



图的左侧是基于交互的代理。在此模式下，代理创建现有系统的视图，以协调与底层服务的交互以实现目标或结果。关键是要将代理添加到系统中，作为驱动系统特性和功能的替代方法。例如，想象一下，独立软件供应商 (ISV) 的会计系统带有用于执行操作的用户体验。基于交互的代理简化了与这些现有功能的交互。与其说是学习如何实现定义宽松的目标，不如说是提供一种编排已知途径的方法。

相比之下，图右侧的基于任务的系统代表了一种不同的方法。该系统中的代理利用他们的知识和能力来学习完成任务并推动业务成果。你可能会争辩说，这两种模式都能实现业务成果，但是基于任务的模型依赖代理本身来决定如何实现结果。这些代理人的确定性较低，而是依赖于他们的学习和进化能力。相比之下，基于交互的代理主要是为了协调一组已知功能而设计的。这些差异会影响您如何构建、确定范围和集成代理以支持您的业务。

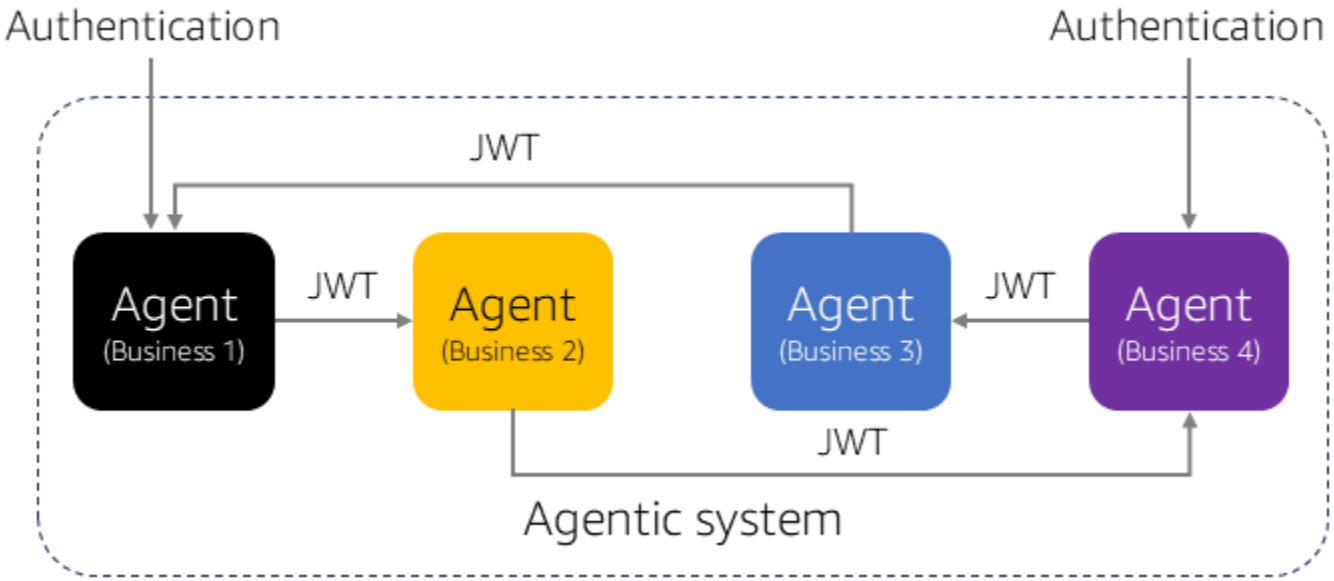
我们还需要能够描述我们部署代理的方式和地点的术语。代理在系统占地面积内的位置会影响其构建、范围和保护方式。下图概述了两个可以应用于代理的不同模型。



图的左侧是一个包含三个不同代理的部署系统。这些代理会暴露给外部客户端，这些客户机可能是其他代理或应用程序。在此模型中，代理被称为公共代理。

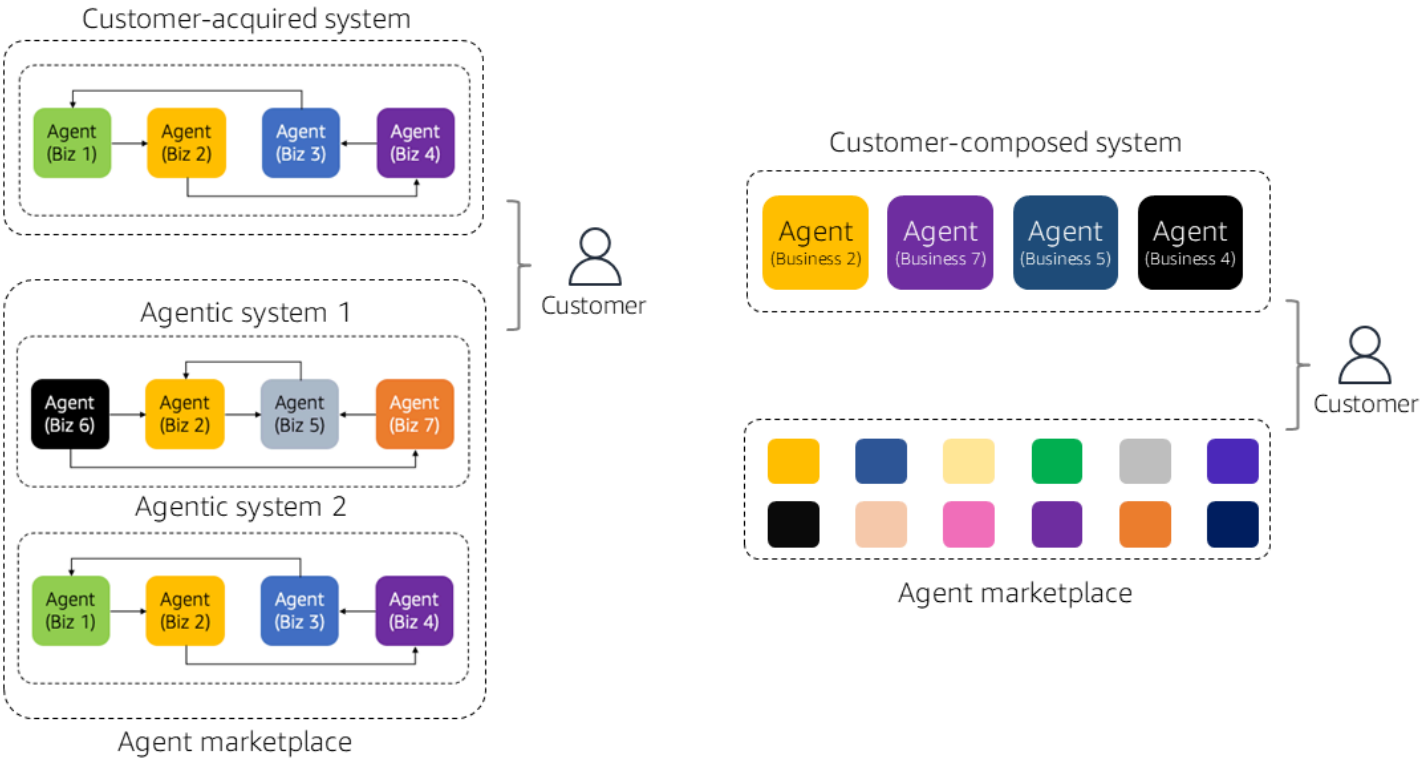
相比之下，右侧的图表显示了解决方案实施中的代理。在这种情况下，有一系列应用程序服务由用户或系统使用。这些用户在与应用程序交互时没有意识到代理是体验的一部分。然后，底层系统的服务会调用和编排代理。以这种方式部署的代理称为私有代理。

代理商的大部分价值都集中在公共模式上，在这种模式下，提供商可能会发布其代理商，目的是将其与其他第三方代理整合。然后，这些代理将成为互联服务网格或网络的一部分，这些服务共同能够解决许多用例。虽然这些代理可以用于许多领域，但 business-to-business用例却是自然而然的。下图提供了组装解决特定问题的收集代理的概念化视图。



该图显示了四个共同努力实现一系列目标的业务代理。当代理以这种方式组成时，它们代表一个代理系统，这样的系统有很多种类型。它们可能是一组预先打包的协作代理，通常作为一个单元使用。或者，系统可以由想要挑选最能满足其需求的代理组合的客户动态组装。

这两种方法都为代理集成提供了可行的途径。一些代理商的构建期望是，它们将被集成到特定的系统中，从而最大限度地提高其价值、覆盖范围和影响力。这种代理系统的概念也引发了人们对如何获取代理的疑问，可能有很多方法可以解决这个问题。下图提供了如何通过交易体验创建这些代理和系统的示例。

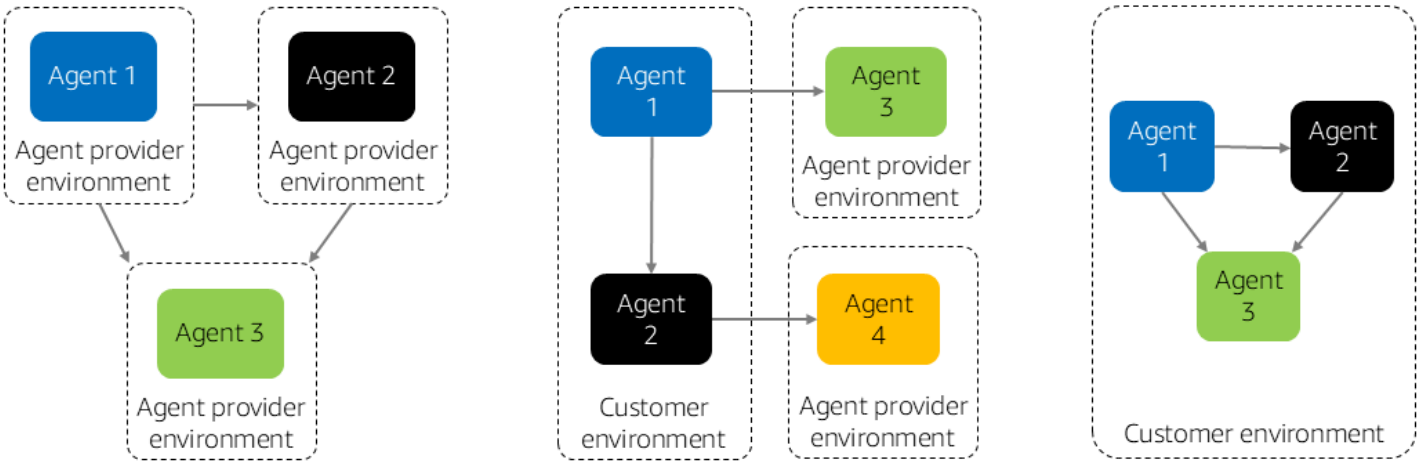


显示了两个市场体验示例。在左边，市场用于购买预打包的系统。在这种情况下，市场发现并启用了能够实现更广泛目标的系统，这些目标需要集成和协调多个代理。

右侧的示例显示了一个市场，在这个市场中，代理被发现并组成代理系统。在这种情况下，客户可以构建任何由兼容的集成代理组成的系统来满足他们的需求。以这种方式组装代理的能力取决于各个代理的兼容性模型和集成要求。

代理托管注意事项

既然你已经了解了更广泛的代理概念，那么让我们来讨论一下托管和运行这些代理意味着什么。我们必须考虑计算的运行方式和地点，计算如何扩展，如何运作，以及如何管理。同时，我们期望作为代理人看到的一些模式得到更广泛的应用和采用。下图显示了可能的排列示例。



这里代表了三种不同的策略。在图表的左侧，您可以看到一个模型，其中我们的代理在每个代理提供商的环境中托管、扩展和管理。这些代理作为服务发布和使用，在标记为代理即服务 (AaaS) 的模式下运行。右边是一个模型，其中提供商的代理都托管在专门的客户环境中。

图的中间是一个混合部署模型，它结合了这两种策略，即在客户环境中本地托管一些代理，并与一些远程托管在提供商环境中的代理进行交互。

第四个选项（未显示）可能是将代理构建为低代码服务或无代码服务，由代理基础架构服务扩展和管理。我们不会详细介绍这些内容，因为托管代理的架构和托管主要由拥有服务的组织决定。

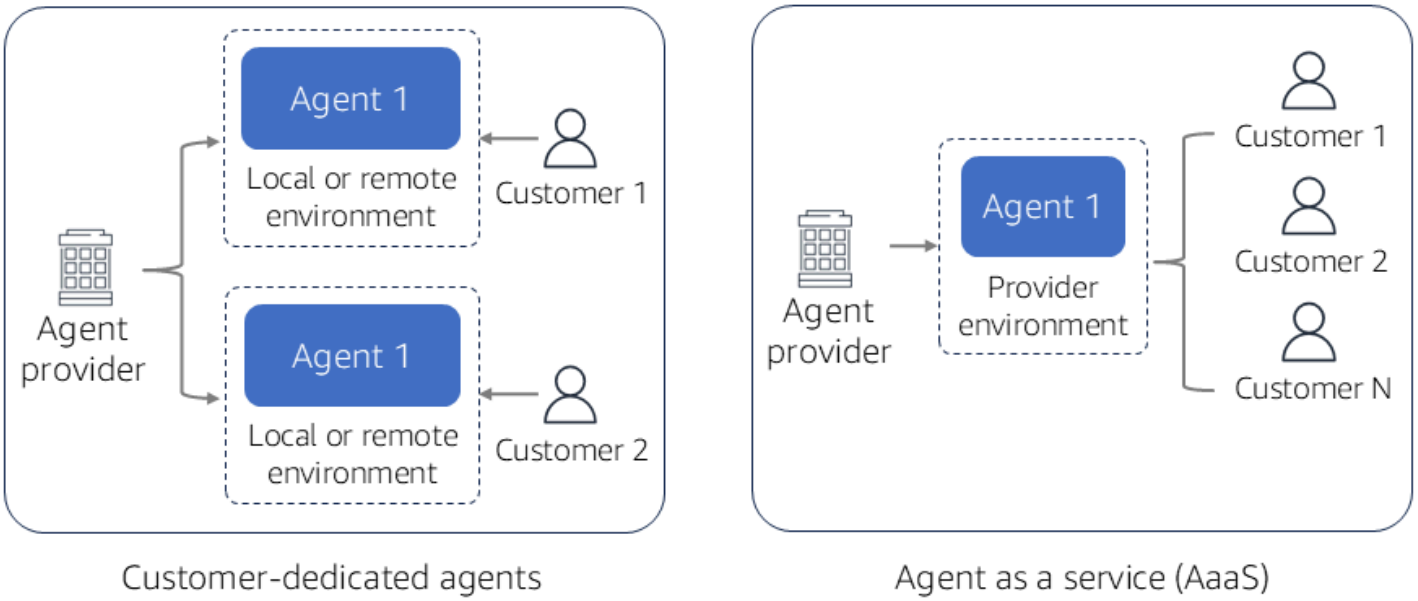
你可以想象可能影响其中一个模型采用的因素范围。例如，合规性、监管和安全限制可能会将某人推向客户托管的代理。规模、敏捷性和效率可以推动组织更多地走向 AaaS 模式。

这里的关键概念是，代理可以而且可以通过多种方式部署和托管。你的工作是确定如何最好地应用代理。占地面积、安全性和部署等因素会严重影响您对待建筑和运营代理的方式。例如，私人代理和公共代理可能有不同的设计和发布生命周期。

代理商遇见多租户

人们很容易将代理视为构建块，其中代理被视为一系列自主组件，这些组件是为了支持特定领域或业务问题的需求而组装的。更有趣的是，当我们开始考虑提供商如何打包和使用这些代理时。在许多方面，代理商成为企业的成本和收入来源。代理提供商必须考虑消费其服务的不同角色、角色的消费特征以及允许代理提供商创建与消费者一致的定价和分层模型的获利策略。

代理提供商可以支持多种模式来部署其代理以满足客户需求。下图显示了两种主要代理部署模型的概念视图。

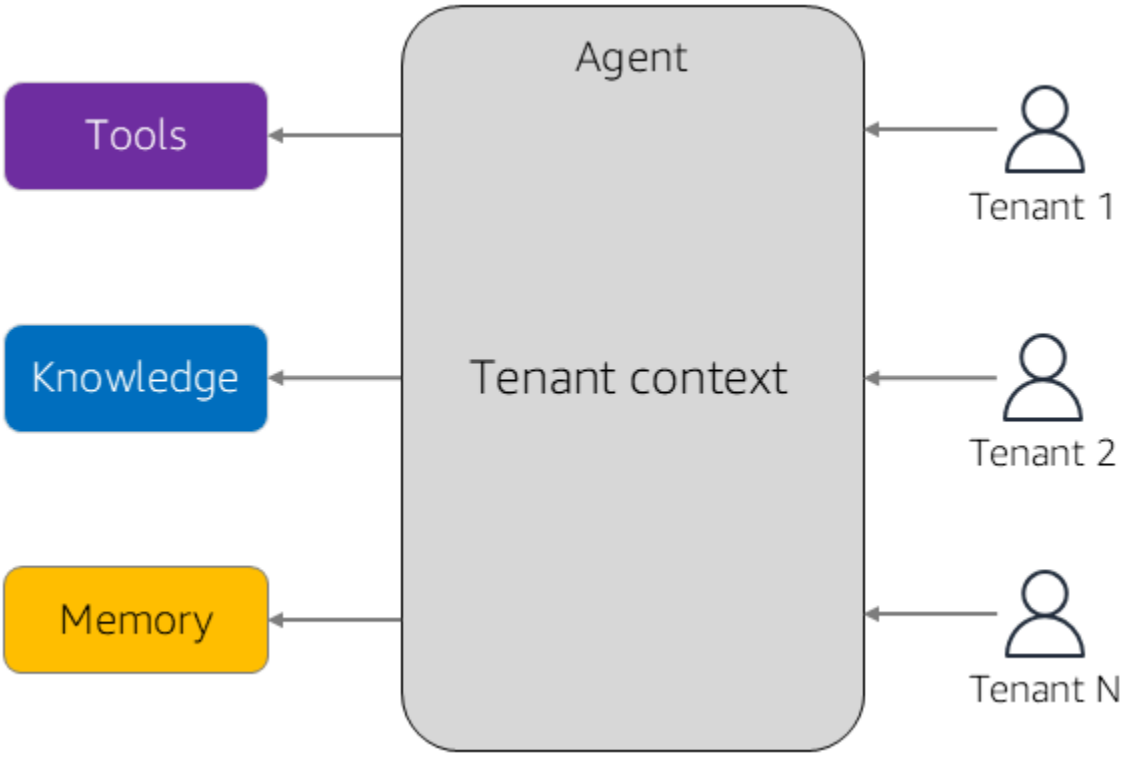


图的左侧显示了客户专用代理模型。代理提供商通过为每个已注册客户部署单独的代理实例来构建代理。通过这种方法，代理的能力及其获取知识的能力将仅限于给定客户环境的范围。这最终代表了一种以客户为单位的体验，它继承了支持专用客户环境的一些复杂性和优势。

相比之下，图右侧的图表中只有一个部署在提供商环境中的代理。代理处理来自多个客户的请求，并根据所有客户的集体经验不断发展和学习。添加的每个新客户仅代表代理的另一个有效客户。代理像代理即服务 (AaaS) 模型一样运行，使用共享结构来支持客户的需求。在这两种情况下，代理使用者都可以是应用程序、系统，甚至是其他代理。

您可以通过两种方式来查看 AaaS 模型。上面的模型为所有客户提供了相同的体验。这意味着代理的内部结构将不包括任何考虑请求客户背景的专业化级别。通常，对于这种模式，假设代理的范围、目标和价值的性质围绕着一组共同的资源、知识和成果，这些资源、知识和结果普遍适用于所有客户。

AaaS 的另一种方法是，客户环境会影响代理的体验和实施。下图提供了此上下文中 AaaS 代理足迹的概念视图。

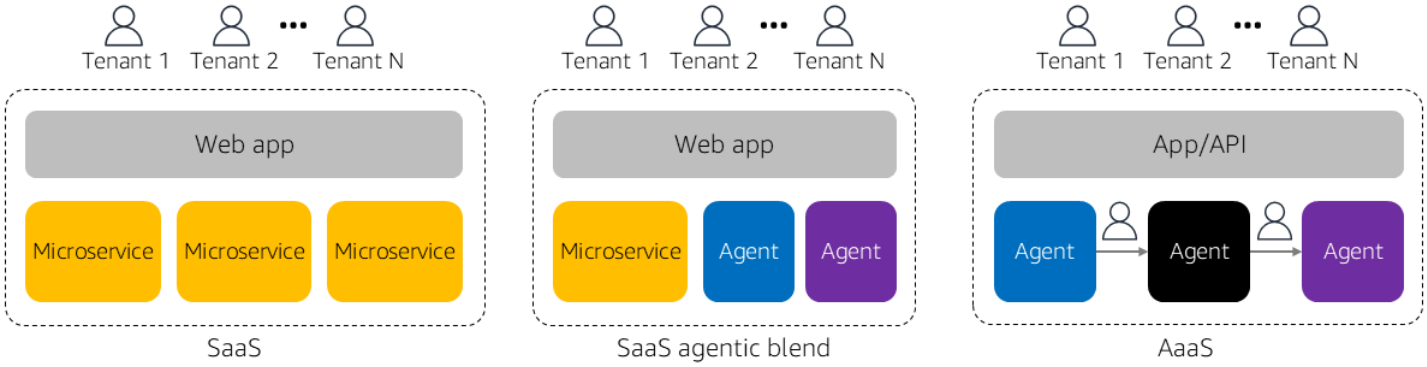


在这个 AaaS 视图中，传入请求的来源和上下文会严重影响代理的占用空间。作为代理底层实现一部分的资源、操作和工具可能因每个传入的租户请求而异。代理的价值与其利用租户上下文得出受租户状态、知识和其他因素影响的行动和结果的能力有关。有些请求可能会产生独特的租户结果，而另一些请求可能会为每个租户带来更量身定制的结果。这为代理人的学习能力增添了新的维度，其中可能包括更具情境性，以及获取和应用知识以增强目标成果。

对于提供商而言，AaaS 模式具有许多优势。由于多个客户使用单个代理，提供商有更好的机会实现规模经济、提高运营效率、控制成本和创建统一的管理体验。这有可能为代理业务带来更大的灵活性、创新性和增长。

这些品质与推动采用软件即服务 (SaaS) 模式的相同原则重叠。从本质上讲，AaaS 模型是作为一种多租户服务构建的，它继承了 SaaS 环境中许多相同的规模、弹性、隔离、入职和运营属性。在许多方面，AaaS的经验在很大程度上借鉴了SaaS提供商使用的策略和实践，但将这些术语区分开来是合理的。就我们的目的而言，重点主要放在需要多租户支持的建筑和运营代理所带来的影响上。

对于一个可以平等对待所有用户并且不需要管理持久、敏感或客户特定数据的系统来说，租赁的概念对他们的代理的影响微乎其微。对于需要在保持数据隔离、自定义和情境感知的同时为多个客户提供服务的系统，支持多个租户可能是代理设计、策略和目标的重要组成部分。下图显示了如何在代理环境中使用多租户。



此图的左侧是经典的多租户架构。它包括一个 Web 应用程序和一系列实现业务逻辑的微服务。多个租户使用此环境的共享基础架构，可以扩展以满足不断演变的租户群体不断变化的工作负载。所有租户均可通过单一管理平台运营和管理环境。

想象一下这个心理模型是如何映射到这张图右边的特工的。一个代理运行由一个或多个租户使用的 AaaS 模型。代理可能来自多个提供商，租户上下文在它们之间流动，因为一个代理的单个实例必须处理来自多个租户的请求。

此图中间的示例是一个混合模型，其中代理是整体 SaaS 体验的一部分。系统的某些部分采用更传统的模型实现，而系统的其他部分则依赖代理。这种模式在许多 SaaS 产品中可能很常见，特别是对于正在过渡到代理体验的组织而言。这种模式通常会持续下去，因为并非所有系统都以纯粹的 AaaS 形式交付。另请注意，多租户仍然适用于模型的代理。虽然代理可能嵌入在系统中，但它们仍可能处理来自多个租户的请求。

问多租户是否真的很重要是很自然的。你可能会争辩说代理处理请求，因此支持租赁可能效果不大。但是，当我们深入研究多租户代理的含义时，租赁可能会直接影响代理如何影响访问、部署和配置工具、内存、数据和其他代理部分以支持单个租户的方式。租赁还会影响扩展规模、限制、定价、分层和其他业务方面如何应用于您的代理架构。

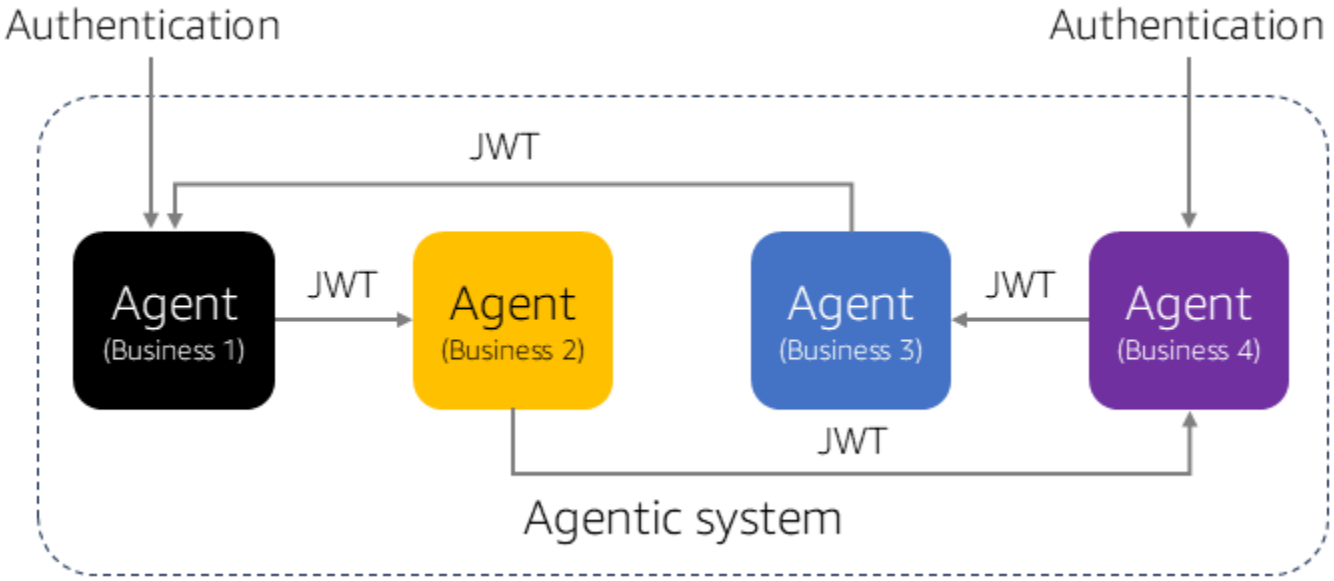
由此得出的一个结论是，有些代理用例需要多租户支持。挑战在于确定多租户如何塑造您的代理体验的整体设计和架构。对于某些代理来说，多租户支持是一种与众不同的能力，它允许代理将租户特定的上下文应用于提供目标结果的代理。

在随后的章节中，您将看到我们为描述多租户 SaaS 架构而创建的术语和设计模式将如何发挥作用。AaaS 模型可以通过借用有用的方面来采用这些概念，从而在需要时引入新的代理特定概念。

身份、租户上下文和代理系统

向单个代理添加租户上下文并不是特别困难。在许多情况下，团队可以依赖典型的机制，这些机制将用户和系统绑定到租户，并将租户感知令牌传递给代理。当我们考虑租户上下文和身份如何支持多个代理时，这很重要。在此模型中，租户必须绑定到跨越所有协作代理的身份。

总的来说，代理领域需要一个更具交叉性的身份模型，该模型要与代理系统当前和新出现的需求保持一致。代理提供商需要支持操作代理系统附带的独特安全、合规和授权模型的身份机制。在系统由客户或其他代理组成的环境中，这尤其具有挑战性。每个已加入的代理都必须将其身份和租户上下文与代理交互关联起来。下图突出显示了作为 agent-to-agent (a2a) 交互一部分的潜在身份和租户上下文挑战。



此图显示了一系列由提供商构建的代理作为我们所涵盖的代理系统的一部分进行交互。现在，它已通过身份和租户环境进行了改造。此场景是支持多个入口点的代理系统的示例。我们假设该系统中的每个代理都需要自己的身份验证机制才能将系统或用户解析为给定的租户。当这些代理交互时，租户上下文将传递给 JSON Web 令牌 (JWT)，该令牌将用于授权访问并将租户上下文注入代理。

从概念上讲，这种情况的主要区别在于代理独立部署和运行，这意味着每个代理必须能够解析其身份并授权访问。关键是它的身份必须具有一定的分布式能力来处理更广泛的代理系统的需求。在代理如何共享租户上下文方面也必须保持一致。

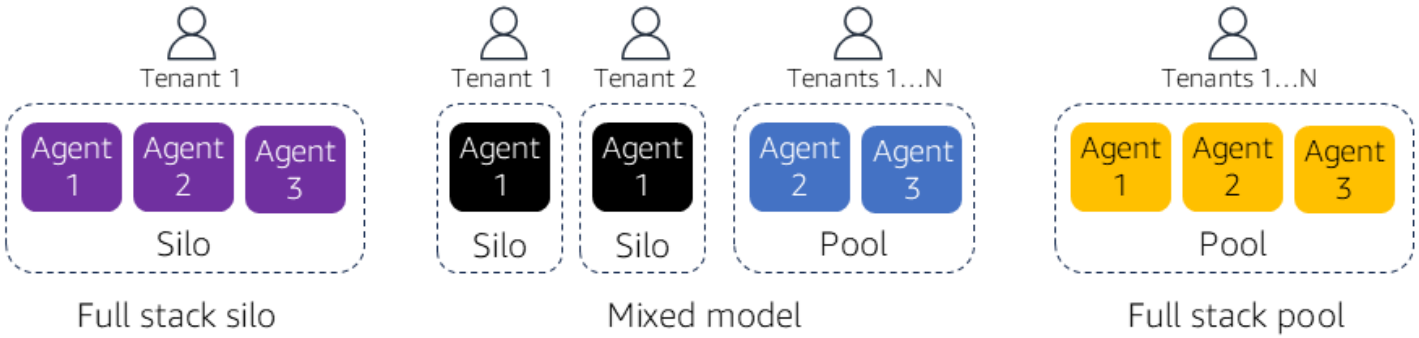
将 SaaS 的商业价值应用于 AaaS

通常，当我们考虑在 as-a-service模型中运行任何系统时，我们会考虑体验的性质以及其技术和运营足迹如何推动业务成果。例如，在采用 SaaS 时，组织利用规模经济、运营效率、成本概况和灵活性来推动增长、利润和创新。

以 AaaS 形式交付的代理可能会瞄准类似的业务成果。通过支持多个租户，代理可以使资源消耗与租户活动保持一致。这产生了传统 SaaS 环境所带来的规模经济。AaaS 还允许组织以支持频繁发布和提高代理提供商灵活性的方式管理、操作和部署代理。关键是AaaS模型不依赖于技术。它创建并推动业务战略，以促进增长、简化采用和简化运营。

代理部署模型

在基本的 AaaS 体验中，提供商可以使用各种模式部署代理。有许多因素会影响代理的部署方式，以满足客户、性能、合规性、地理位置和安全需求。不同的部署策略会影响代理的设计、实施和使用方式。在这里，我们可以引入经典的多租户术语来标记不同的部署策略。下图显示了在 AaaS 环境中部署代理的不同排列方式。

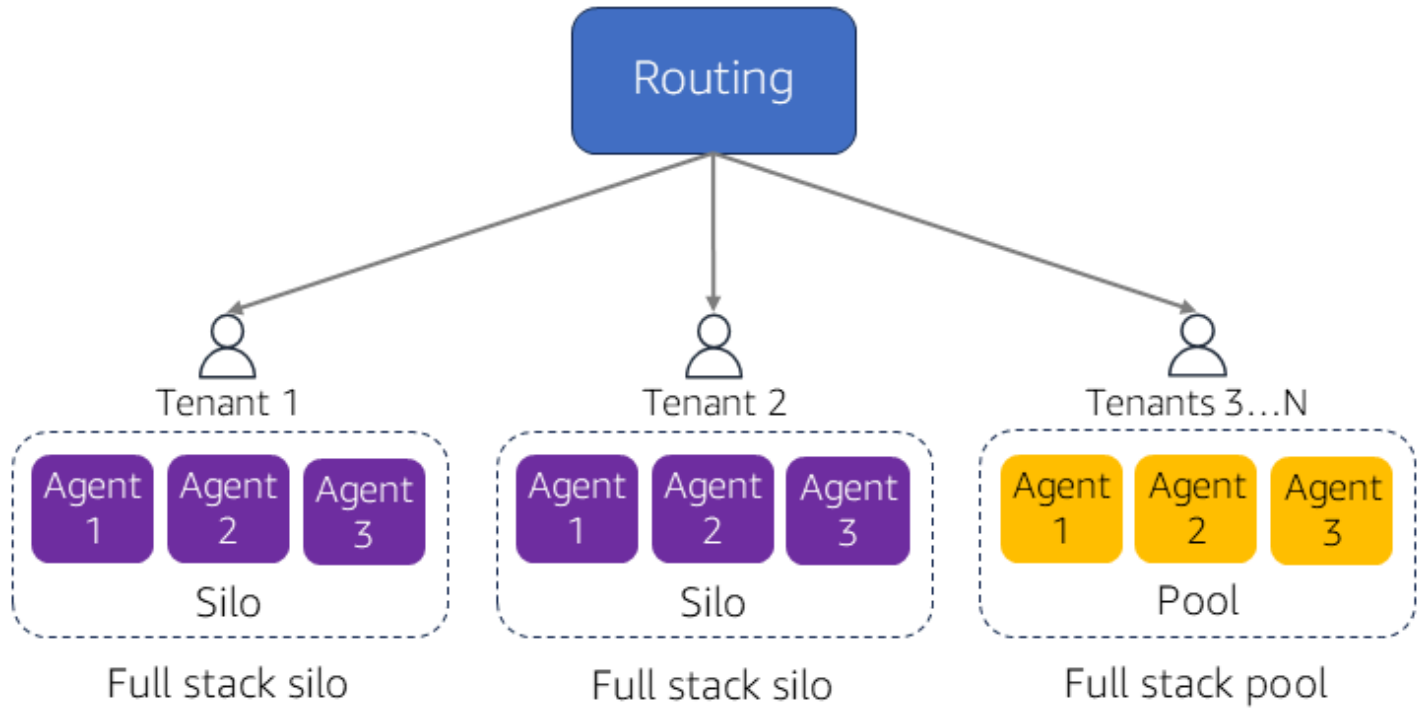


此图表示代理部署的三种模式。左侧是一个孤立的模型，其中为每个租户提供完全隔离的体验和一组专门的代理。在这种情况下，代理不会在租户之间共享计算、资源或执行环境。

中间的示例说明了一种混合模型，其中租户使用孤立代理和池化代理的组合。例如，代理 1 以孤立模式部署，每个租户都接收一个专用实例，而代理 2 和 3 则在池化模式下运行，在租户之间共享资源。

右侧是完全池化的模型，其中所有代理均在租户之间共享，提供经典的多租户部署。在这种情况下，租户利用通用的计算、内存和服务基础架构来执行代理。

这个想法是，代理可以在不同的部署模式下运行，计算和依赖资源要么专用（孤立），要么在租户之间共享（共享）。这些部署策略并不相互排斥。代理服务通常支持一系列客户需求，将两种模式结合起来以平衡性能、隔离、成本和可扩展性。下图显示了一个在同一操作环境中支持多种部署配置的代理系统。

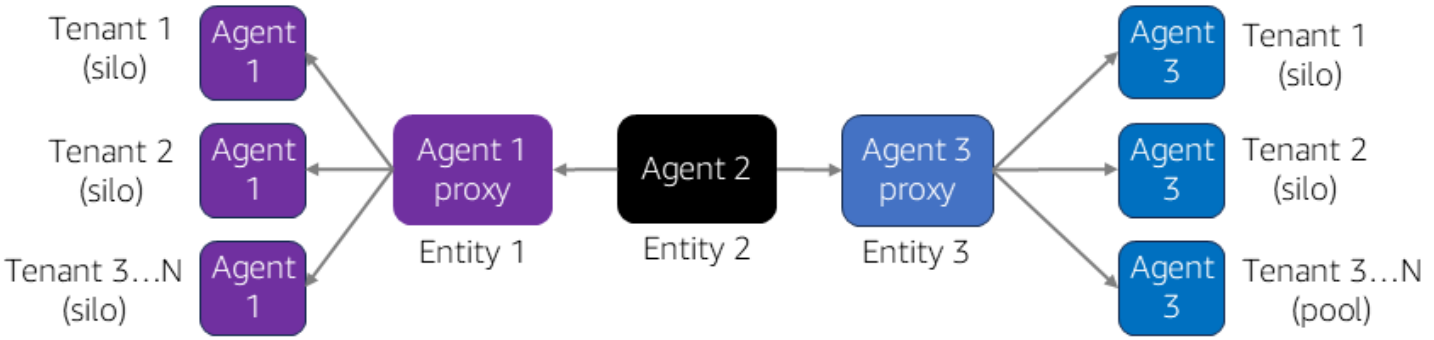


在此图中，代理提供者有三个通过代理即服务 (AaaS) 部署的代理。他们支持两种类型的租户。在左边，两个租户有合规性和性能要求，他们通过全栈孤岛模型来满足这些要求。右侧的其余租户在租户共享资源的池化模型中运行。

如果目标是敏捷性和运营效率，请尝试限制与支持按租户部署模式相关的影响。这意味着要建立路由和其他体验机制，允许通过单一控制面板管理、操作和部署代理。

如果您在低代码或无代码环境中构建代理，则不会出现孤立或池化代理的概念。相反，代理可以完全由另一个代理管理。孤立模型和池化模型更多地适用于组织控制代理结构和占地面积的环境。在这种情况下，团队应考虑支持哪种部署模式。

从表面上看，这些部署模型不会直接影响代理在更广泛的系统中的运作方式。代理可能无法直接意识到部署在孤岛或池化模型中的其他代理。相反，这些部署策略可以作为环境中路由结构的一部分来实施。下图显示了如何使用路由策略实现孤立模型和池化模型的示例。



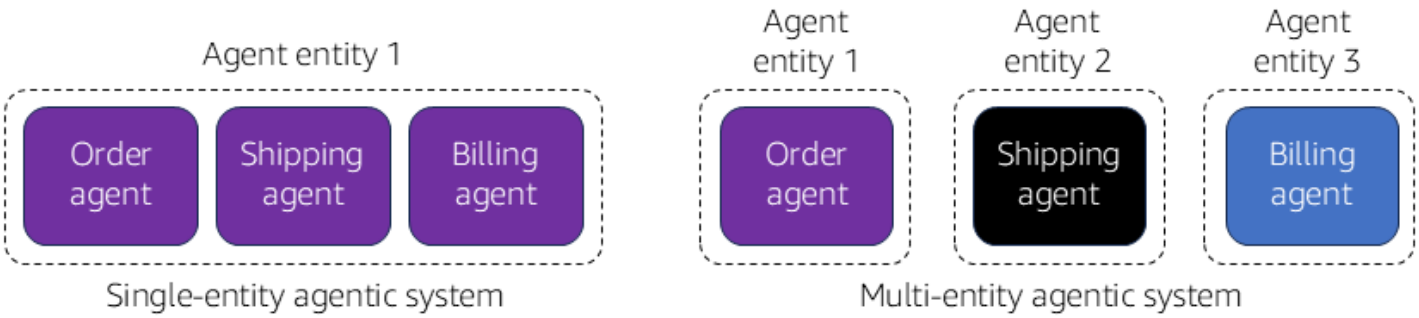
此示例包括来自三个不同提供商的三个代理。每个代理提供商都可以选择实施自己的部署策略。例如，代理 1 使用代理将入站请求分发给一组孤立的租户代理。代理 2 不需要路由，并且通过一个池化代理支持所有租户请求。Agent 3 是一种混合模型部署，其中一些租户是孤立的，而另一些则是池化的。

是否以及如何选择支持这些部署模式取决于您的解决方案的性质。您可能不需要支持任何一种型号。但是，在某些情况下，您可能必须考虑支持此策略，例如合规性、邻居噪音、性能或分层。

引入和应用租户上下文

如果我们构建支持多租户的代理，则必须首先考虑如何设置租户上下文，该上下文将用于在代理的实施中应用特定于租户的策略、策略和机制。

在最基本的层面上，您可以通过我们在经典多租户架构中使用的常用工具和机制将租户上下文引入代理。这可以通过 API 密钥或其他各种验证机制实现。 OAuth这方面的许多示例都侧重于将经过身份验证的系统或用户解析为包含租户上下文的 JSON Web 令牌 (JWT) 密钥。然后 JWT 通过系统传播。当我们考虑如何组成代理系统时，这会变得更加有趣。下图显示了两种代理环境的示例。



在此图中，左侧的模型表示一个代理系统，其中所有代理均由一个实体拥有、管理和托管。当你完全控制整个体验时，你可以使用典型的策略让租户通过每个代理。

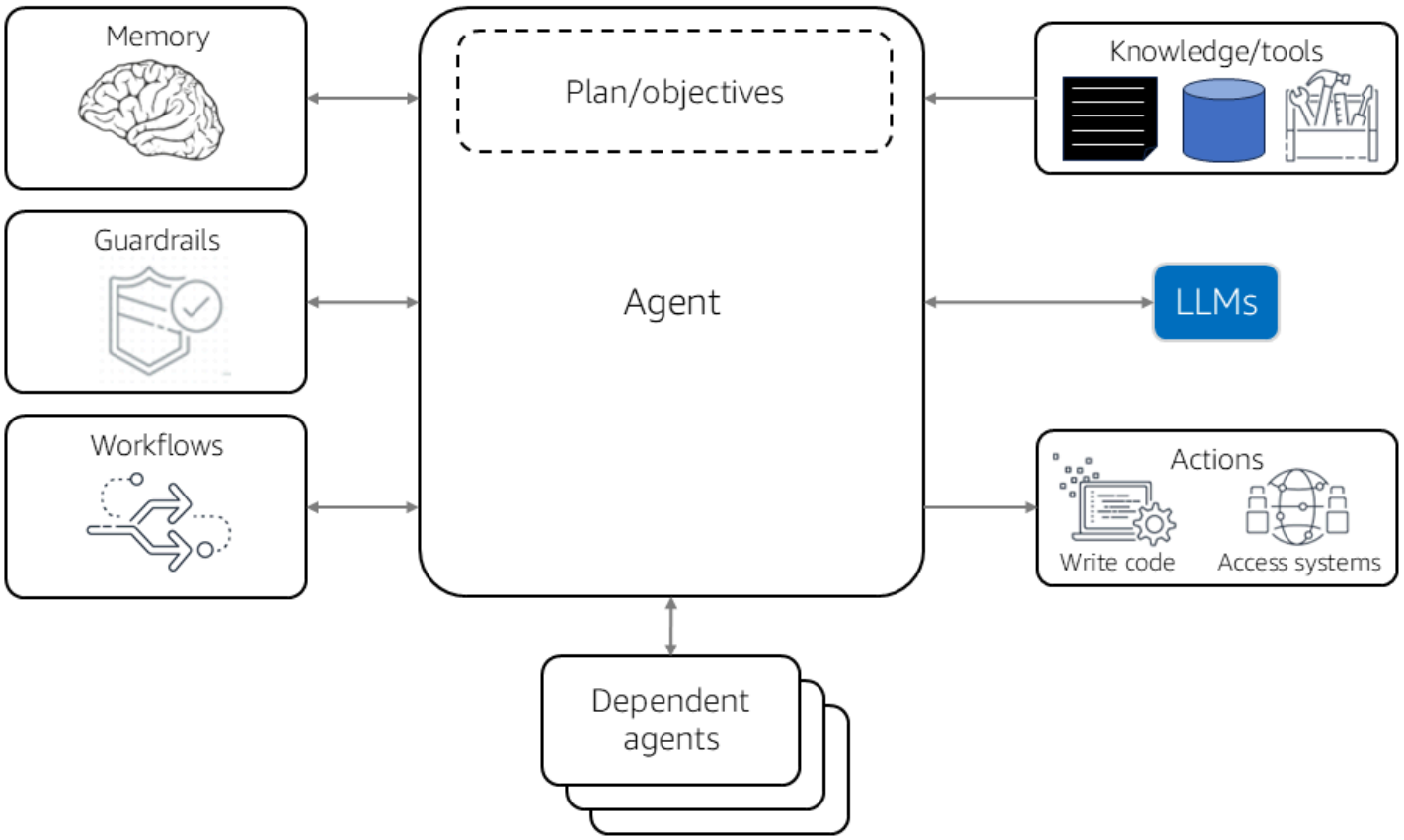
右侧的模型可能更常见，它表示跨越多个实体的代理系统。代理是独立构建、管理和操作的，因此每个代理都有自己的身份验证和授权方案。这里的挑战是，我们需要一种通用的方法来解决和共享这些代理之间的租户背景。这依赖于更加分布式的模型，在这种模型中，每个代理都必须能够对系统或用户进行身份验证，并根据所应用的机制将其解析给租户。

建造具有租户意识的代理

多租户会影响我们实现单个代理的方式。在代理处理请求时，请考虑租户上下文如何影响代理访问数据、做出决策和调用操作的方式。为了更好地了解多租户如何以及在何处影响代理的个人资料，请首先确定构造如何成为任何代理的一部分。

面临的挑战在于，代理的范围、性质和设计绝非具体，因为提供商对代理体验的设计做出自己的选择。归根结底，代理的重点在于它是一种自主学习服务，可以访问一系列工具、数据源和内存，以确定如何最好地解决任务。

确切地知道代理使用了哪些策略和模式并不重要。在多租户模型中，更重要的是确定代理的各个部分是如何配置、访问和应用的。考虑一个依赖一系列资源和机制来实现其目标的潜在代理环境。下图显示了此类代理的示例。

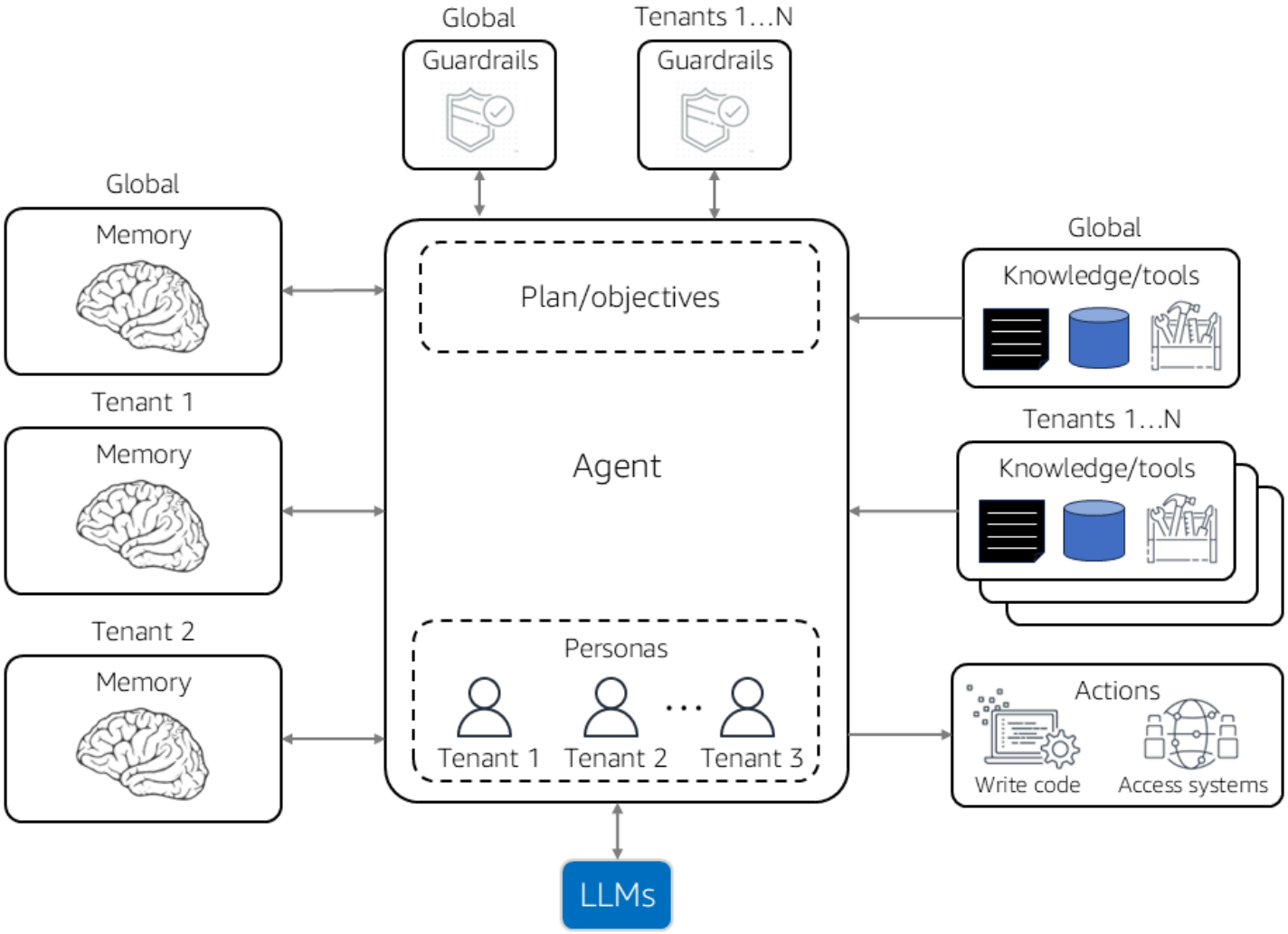


这张图代表了全面的代理可能性，展示了可以组合起来实现目标的各种工具和机制。在图表的左侧，请注意代理如何依赖内存作为其上下文的一部分、定义指导其活动的策略的护栏以及针对特定任务的工作流程。有些人可能会争辩说，工作流程不应包含在这种背景下，但可能在某些情况下，工作流程是代理体验不可或缺的一部分。

图表的右侧显示了知识和工具等输入如何提供额外的见解和背景信息，从而增强代理的能力。然后，代理会输出操作，例如编写代码或访问系统。图的底部显示了代理如何依赖一个或多个内部或第三方代理，这些代理可以作为更广泛的系统的一部分进行编排。

我们现在可以考虑引入多租户意味着什么。租赁迫使我们考虑代理如何以及在何处引入决定行为和行为的策略和机制。这为我们如何看待代理人的知识、学习、工具和记忆力增添了另一个维度。

现在让我们考虑如何修改此模型以支持多租户。下图显示了多智能体模型的示例。



在此图中，我们介绍了租户角色，这些角色旨在塑造代理如何整合租户上下文。例如，在图的左侧，代理内存被更改为支持租户特定的内存。图的右侧也是如此，代理支持租户特定的知识和工具。同样的支撑也适用于护栏。

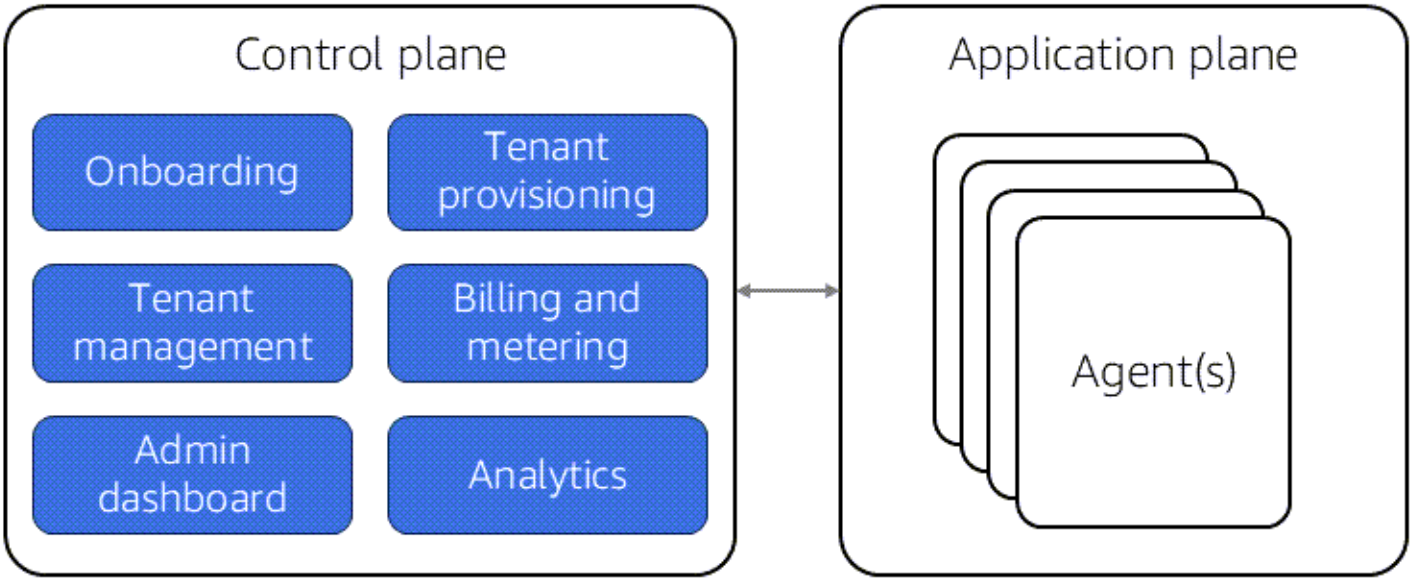
这可能是一个极端的例子，因为并非多租户代理的所有方面都需要每个租户的资源。关键是，你应该考虑如何为特定租户量身定制代理可以提高其有效性。这种方法使您的代理能够提高其影响力和价值，在响应中提供更相关的背景信息，并发展专业能力。然后，代理将能够学习、适应和执行特别适合不同角色的任务。

主要思想是，租户上下文直接影响您构建代理的方式。它还可以塑造租户与外部实体（包括其他代理）的互动。构建多租户代理会带来传统挑战，例如邻居吵闹、租户隔离、分层、限制和成本管理。您的代理的设计和架构必须解决这些基本的多租户概念，我们将在下一节中探讨这些概念。

在代理环境中使用控制平面

多租户最佳实践通常将实现分为两个不同的部分：控制平面和应用程序平面。控制平面提供了单一控制面板，用于访问跨环境租户的运营、管理和协调机制。应用程序平面是业务逻辑、特性和功能能力所在的地方。

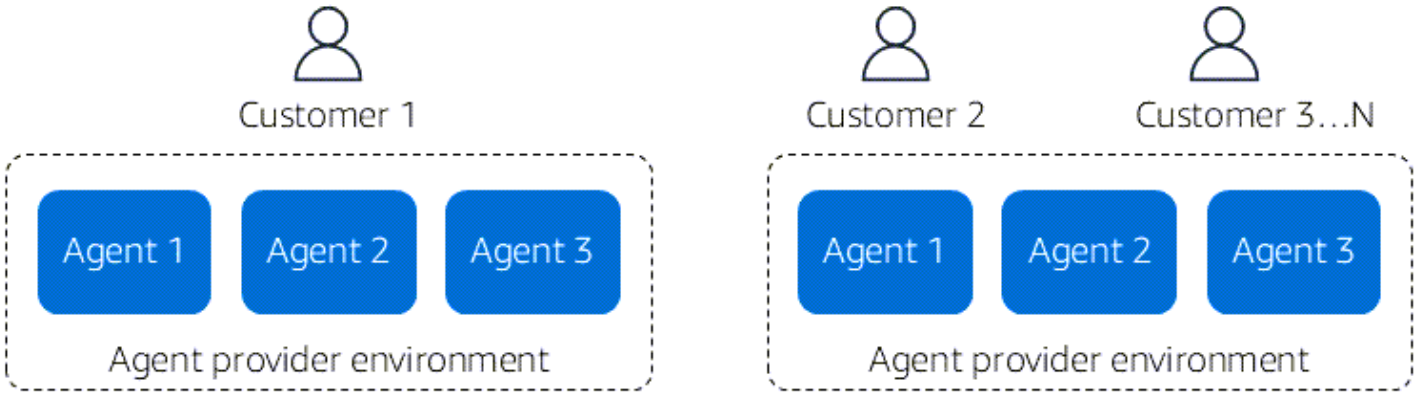
这种责任分工也适用于代理模型。多租户代理需要一定程度的集中管理、操作和洞察，通过控制平面持续满足这些需求是有意义的。下图显示了在代理即服务 (AaaS) 环境中如何划分这些平面的概念视图。



此图显示了控制平面和应用平面的传统分离。新增功能是，控制平面现在可以管理构成 AaaS 环境的代理。控制平面与所有代理交互，因为我们假设代理是由一个提供商构建、管理和部署的。

该模型引入了额外的复杂性，尤其是在代理生命周期和第三方协调方面，但保留了基本的关注点分离。通过协调代理的配置、提供租户和代理的可观察性、收集用于计费的消耗和计量数据以及管理租户策略，控制平面仍然提供相同的核心功能。

如果您考虑使用包含来自不同提供商的代理的多代理系统，则这种情况会变得更加复杂。下图显示了此类模型的示例。



此图描绘了来自不同提供商的四个代理，它们是多代理系统的一部分。第三方提供商仍在运营和部署每个代理，这些代理配置为允许来自一个或多个提供商的授权访问。但是，代理仍处于提供商的控制之下，因此每个代理都维护自己的控制平面。

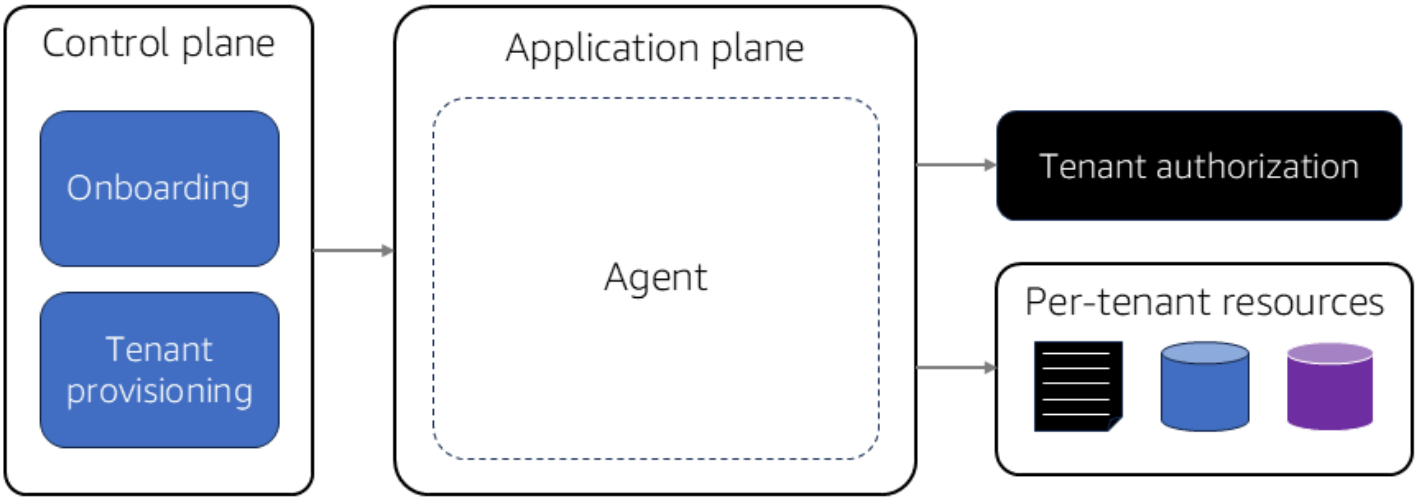
从本质上讲，这些多租户代理的行为就像与其他代理集成的第三方服务。因此，他们必须拥有自己的控制平面，以便对代理的功能进行集中操作、配置和管理。

我们假设代理是在提供商托管的体验中运行的独立服务。但是，在代理使用者对托管代理的方式和位置施加更多限制的情况下，这可能尚不清楚。

让租户加入代理商

入职通常是任何 AaaS 环境的重要组成部分。如何创建、配置和配置租户通常涉及许多活动部件、集成和工具。代理入职体验可能需要与 AaaS 控制平面相同的服务，包括租户身份、分层、按租户配置资源和配置租户策略。

您的代理入职方法受代理环境占地面积和租赁模式的影响。孤立和合并的代理都有自己的细微差别，选择使用单个代理或多个代理也会影响入职流程。下图显示了入职如何影响代理配置的概念视图。



每次您加入代理时，控制平面都必须采取必要的步骤以使租户能够访问代理。如何引入租户因代理授权模型而异，但假设您将创建一个将代理请求与单个租户关联的租户身份。此租户上下文通过将其应用于路由、作用域和控制访问来决定代理体验。

入职还可能要求您配置代理使用的每租户资源。在这里，控制平面的租户配置服务将您的代理连接到代理所查阅的租户特定数据和资源。

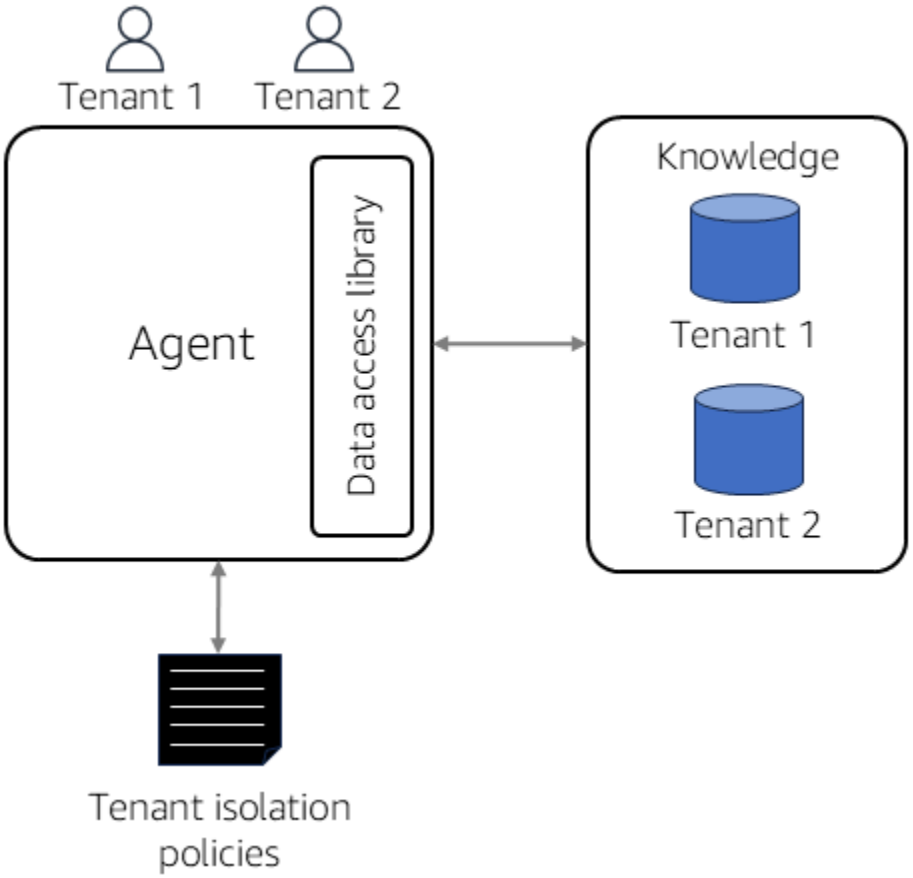
如果您的系统依赖于集成第三方代理，则还必须在入职过程中满足这些代理的需求。其工作原理取决于授权代理之间访问权限的安全和集成机制。理想情况下，编排和配置 agent-to-agent身份验证和授权所需的步骤是通过自动入门来完成的。

强制租户隔离

租户隔离是一个适用于所有多租户设置的概念。这意味着您的策略和策略可确保一个租户无法访问其他租户资源。对于多租户代理，您可能需要引入有助于强制执行和代理的租户隔离要求的结构和机制。

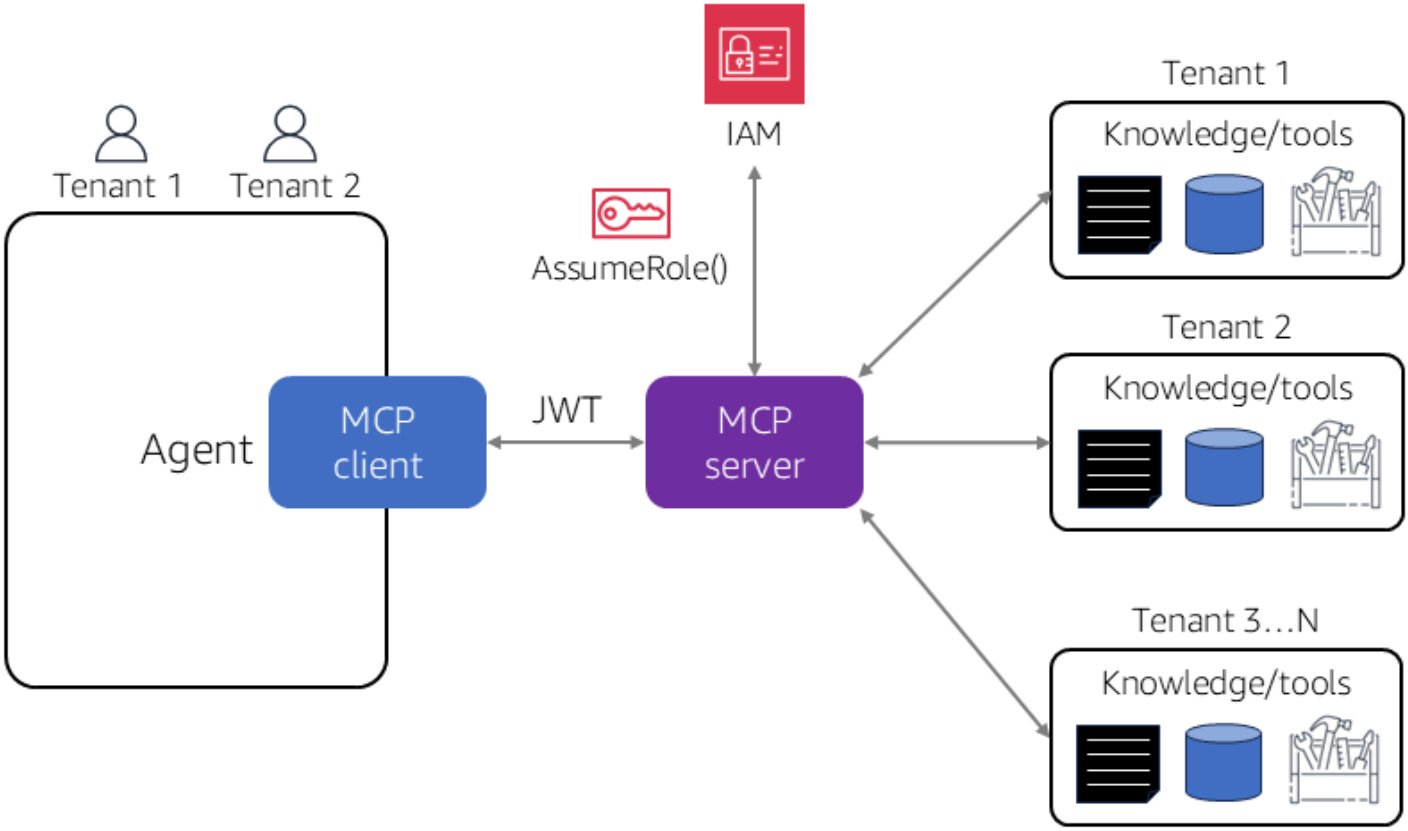
应用租户隔离就像其他使用传统多租户系统的策略一样。通常，在构建 AaaS 架构时，请确定系统中请求或操作可以访问资源的任何区域，以确定请求是否跨越任何租户边界。例如，微服务可能依赖于每个租户的专用 Amazon DynamoDB 表。这要求你引入政策，确保一个租户的表格不能被另一个租户访问。

在这种情况下，可以考虑通过代理的视角进行租户隔离，以及代理与每个租户的任何资源的交互。下图显示了一个概念示例，说明代理如何应用租户隔离策略来控制对租户资源的访问。



在此图的右侧，代理具有每个租户的知识，这些知识存储在单独的矢量数据库中。代理处理请求时，会检查提出请求的租户的上下文。基于此，代理会应用适当的隔离策略，以确保限制租户访问其指定边界之外的数据或资源。

如果您的代理使用模型上下文协议 (MCP)，它也可以实现您的租户隔离模型。下图显示了如何引入 MCP 和应用隔离策略的示例。



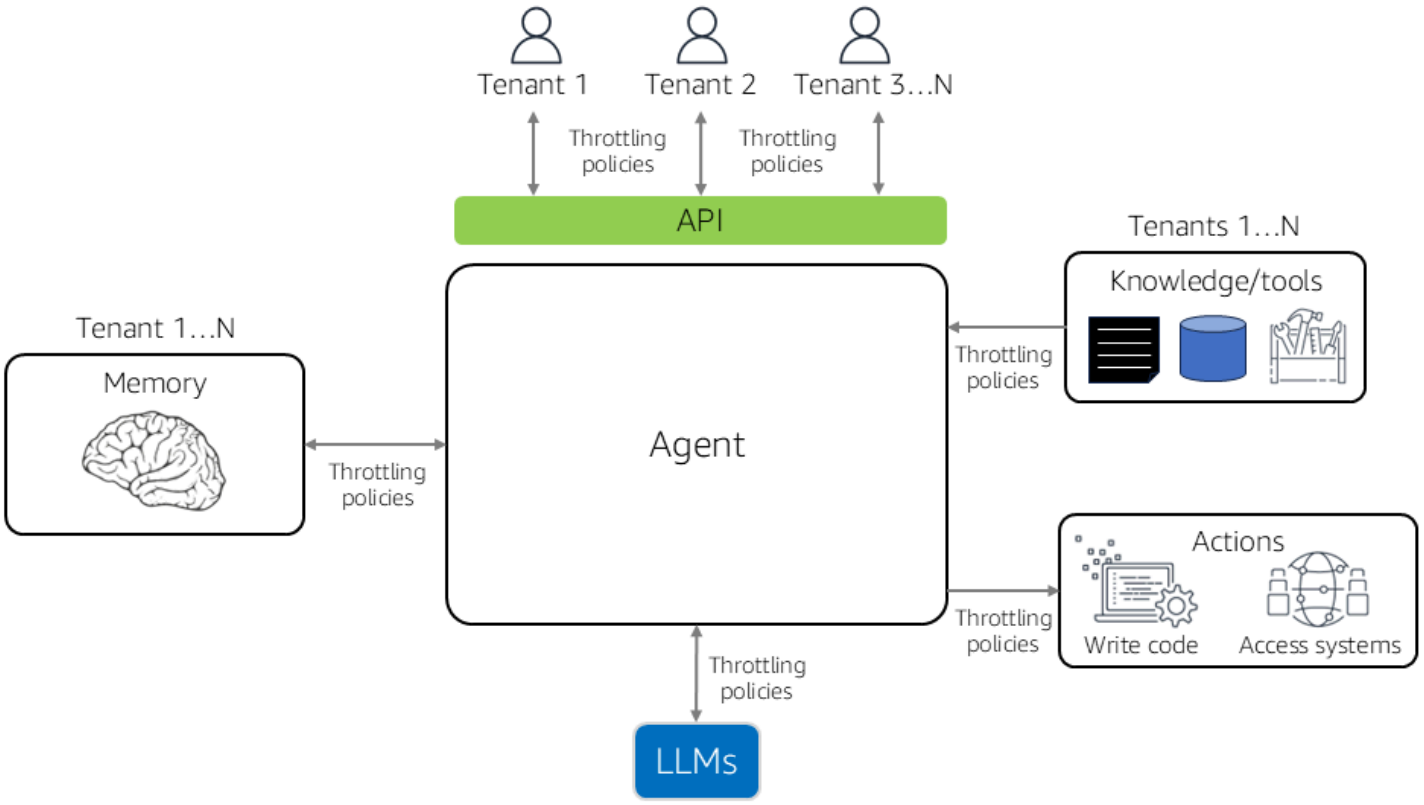
MCP 是一种标准化协议，代理使用它来与任何工具、数据和资源集成。在此示例中，MCP 客户端和 MCP 服务器与图右侧显示的租户专用知识和工具进行交互。租户上下文从一个客户端流向另一个服务器，服务器使用此上下文从 AWS Identity and Access Management (IAM) 服务获取租户范围的证书。凭证控制对每个租户资源的访问权限，确保一个租户可以访问另一个租户的资源。

由于代理采用了多租户，因此他们必须引入在处理请求时应用租户隔离策略的机制。在某些情况下，IAM 可以帮助限制对租户资源的访问。在其他情况下，您可能需要引入其他工具或框架来应用租户隔离策略。

吵闹的邻居和特工

在多租户 AaaS 环境中，多个租户共享一个代理，请考虑在哪里以及如何引入防止邻居环境嘈杂的政策。策略可以引入适用于所有消费的通用限制，也可以使用基于租户或层级的策略，根据给定角色应用限制。与对高级租户相比，您对基本级别租户的消费限制可能更大。

这种节流的概念可以应用于多个架构点。下图显示了引入噪音邻居策略的一些可能区域的示例。



在我们之前对多代理实施的审查中，我们研究了您的代理可以使用的不同资源，重点介绍了代理中每租户资源的潜力。每个接触点都是引入限制政策的潜在领域，这有助于确保租户不会超过系统或租户分层策略的消耗限制。

引入噪音邻居保护的最佳位置是架构中租户共享资源的地方。如果单个租户消耗不成比例，则这些共享或池化组件（例如计算、内存和大型语言模型）最容易受到性能下降的影响。 APIs

应用节流的一个自然位置是代理的入口点，有时被称为 “外边缘”。在这里，您可以在代理开始处理请求之前引入全局限制或 tenant-tier-based速率限制。也可以在执行路径的更深层次应用限制，例如当代理调用 LLM、访问内存或调用共享工具时。

这些政策可帮助您强制执行合理使用，在负载下保持代理弹性，并在租户之间保持一致的体验。根据您的目标，您可以专注于一般系统保护（弹性）或精细管理租户体验（例如，使用基于等级的授权）。

数据、操作和测试

代理和数据所有权

对代理实施的审查重点介绍了代理依赖给定租户数据的场景。在这种情况下，请考虑数据的生命周期，更重要的是考虑数据的存储位置。对于数据性质会影响代理访问方式的行业和用例来说，这一点尤其重要。

AaaS 提供商必须评估如何解决多租户环境中的数据问题，这些问题可能会影响代理的入职、隔离和运营。适用的细微差别和策略因您使用的工具、技术和数据而异。您可以通过多种方式来解决这个问题，这是在创建任何 AaaS 产品时需要注意的事情。

多租户代理操作

在构建代理环境时，请考虑如何操作和管理代理。作为提供商，您需要指标、数据、见解和日志，以便监控代理的运行状况、规模和活动。这在多租户代理环境中更为明显，在这种环境中，你需要了解各个租户是如何消耗代理资源的。

当您需要深入了解代理交互时，这在多代理设置中尤为重要。能够分析和跟踪代理之间的活动对于排除影响系统规模、准确性和效率的问题可能至关重要。

运营团队还可以分析法学硕士的互动，以更好地了解代理所承受的负担。 LLMs这些数据对于精炼剂的实施至关重要。它还可以让运营团队了解代理和租赁如何影响系统的总体成本状况。

培训和测试多租户代理

与建筑代理相关的一个挑战是，他们需要学习和发展。这也意味着，在将代理投入生产之前，我们必须对其进行测试、完善并提高其准确性。在许多领域，您可以检查和评估您的代理是否正确评估和分类意图，或者是否选择和调用了适当的工具和行动。变量列表很广泛，但这最终是为了确保您的代理人找到可以实现目标的结果。

检查与测试代理相关的所有活动部分和原理超出了本文档的范围，但请注意，测试策略会增加多租户 AaaS 环境的复杂性。例如，如果代理具有根据上下文应用于每个租户的数据、内存和其他结构，则代理的结果可以由每个租户的资源来决定。

如果您使用代理来模拟场景，则可能需要针对租户特定的用例扩展模拟。相应地，您必须完善验证程序，以允许每个租户的验证标准有所不同的情况。

注意事项和讨论

SaaS 适合哪里？

业内专家就代理如何影响软件即服务 (SaaS) 格局展开了激烈的辩论。虽然代理确实在为许多系统更改软件，但建议代理过时的交付模式实在是太过分了。一些SaaS提供商可能会因为采用代理而受到干扰，有些提供商可能会倾向于代理即服务（AaaS）模式，从而完全重新考虑其价值主张。其他人可能会通过有选择地引入代理来满足特定需求来取得平衡。

这个话题很有趣，因为采用最佳的SaaS原则可能代表SaaS的下一个发展。这可能意味着 SaaS 正在蓬勃发展，也可能意味着 SaaS 的基本原理正在基于代理的模型中打包和实现。决定术语最终落在何处可能不那么重要，但是SaaS作为一个概念似乎不太可能消失。代理商更有可能塑造SaaS的足迹。

最终，我们必须决定哪些策略可以应用于AaaS，这意味着使组织能够采用代理架构和业务策略，以便提供商能够最大限度地提高其代理系统的效率、价值和影响力。特工不是黑匣子。代理消耗资源、扩展运营、依赖数据并产生成本，所有这些都是提供商必须解决的因素。代理提供商必须评估多租户原则如何塑造服务产品和优化运营模式。

讨论

代理格局不断演变，设计因领域、预期用例和目标行业而异。这种演变的一部分包括进一步完善我们对建筑师在设计和建造代理时考虑的策略、模式和权衡的看法。

全面的代理策略必须与业务和技术目标保持一致。这包括定义目标市场和角色，制定定价和资源管理策略，以及确定代理如何适应更大的系统。在交付 AaaS 时，这些考虑因素尤其重要，其中，规模、成本效益和创新是主要目标。

行动能力同样重要。环境必须支持对代理活动、运行状况指标和使用模式的监控。这在多代理系统中变得更加复杂，因为在多代理系统中，操作必须跨独立代理进行协调。

总体而言，这种关于代理的讨论只是揭示了代理系统中可能包含的各种架构考虑因素。除了选择合适的工具、框架和之外 LLMs，成功还取决于创建一个满足可扩展性、效率、部署和多租户业务需求的架构。

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：将您的本地 Oracle 数据库迁移到兼容 Amazon Aurora PostgreSQL 的版本。
- 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：在中将您的本地 Oracle 数据库迁移到适用于 Oracle 的亚马逊关系数据库服务 (Amazon RDS) AWS 云。
- 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：将您的客户关系管理 (CRM) 系统迁移到 Salesforce.com。
- 更换主机（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：在中的 EC2 实例上将您的本地 Oracle 数据库迁移到 Oracle AWS 云。
- 重新定位（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。您可以将服务器从本地平台迁移到同一平台的云服务。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

参见托管服务。

ACID

参见原子性、一致性、隔离性、耐久性。

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。与主动-被动迁移相比，它更灵活，但需要更多的工作。

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

一个 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

参见人工智能。

AIOps

参见人工智能操作。

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

坏机器人

旨在破坏个人或组织或对其造成伤害的机器人。

BCP

参见业务连续性计划。

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。另请参见字节顺序。

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的应用程序版本（蓝色），在另一个环境中运行新的应用程序版本（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互动的软件应用程序。有些机器人是有用或有益的，例如在互联网上索引信息的网络爬虫。其他一些被称为恶意机器人的机器人旨在破坏个人或组织或对其造成伤害。

僵尸网络

被恶意软件感染并受单方（称为机器人牧民或机器人操作员）控制的机器人网络。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）。

破碎的玻璃通道

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 Well -Architected 指南中的 “实施破碎玻璃程序” 指示 AWS 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

参见AWS 云采用框架。

金丝雀部署

向最终用户缓慢而渐进地发布版本。当您确信时，可以部署新版本并全部替换当前版本。

CCoE

参见云卓越中心。

CDC

请参阅变更数据捕获。

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弹性。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

查看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。云计算通常与边缘计算技术相关。

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

组织迁移到以下阶段时通常会经历四个阶段 AWS 云：

- 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

参见配置管理数据库。

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 管道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)

人工智能领域，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。

配置偏差

对于工作负载，配置会从预期状态发生变化。这可能会导致工作负载变得不合规，而且通常是渐进的，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

参见计算机视觉。

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

参见数据库定义语言。

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。

委托管理员

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。

后

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

参见环境。

侦测性控制

一种安全控制，在事件发生后进行检测、记录日志和发出警报。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

在星型架构中，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)

您用来最大限度地减少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

参见数据库操作语言。

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

参见灾难恢复。

漂移检测

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

参见开发价值流映射。

E

EDA

参见探索性数据分析。

EDI

参见电子数据交换。

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。与云计算相比，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)

组织之间自动交换业务文档。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密文的计算过程。

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

参见服务端点。

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

参见企业资源规划。

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

星形架构中的中心表。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。

失败得很快

一种使用频繁和增量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

参见分支。

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。

少量提示

在要求法学硕士执行类似任务之前，向其提供少量示例，以演示该任务和所需的输出。这种技术是情境学习的应用，模型可以从提示中嵌入的示例（镜头）中学习。对于需要特定格式、推理或领域知识的任务，Few-shot 提示可能非常有效。另请参见零镜头提示。

FGAC

请参阅精细的访问控制。

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，它使用连续的数据复制，通过更改数据捕获在尽可能短的时间内迁移数据，而不是使用分阶段的方法。目标是将停机时间降至最低。

FM

参见基础模型。

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

人工智能模型的子集，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工件，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 AI。

地理封锁

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，而基于主干的工作流程是现代的首选方法。

金色影像

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配置软件，并有助于提高设备制造运营的速度、可扩展性和生产力。

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。侦测性防护机制会检测策略违规和合规性问题，并生成警报以进行修复。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

参见高可用性。

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。

抵制数据

从用于训练机器学习模型的数据集中扣留的一部分带有标签的历史数据。通过将模型预测与抵制数据进行比较，您可以使用抵制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

参见基础设施即代码。

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

参见工业物联网。

不可变的基础架构

一种为生产工作负载部署新基础架构，而不是更新、修补或修改现有基础架构的模型。不可变基础设施本质上比可变基础架构更一致、更可靠、更可预测。有关更多信息，请参阅 Well-Architected Framework 中的使用不可变基础架构 AWS 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

该术语由克劳斯·施瓦布（Klaus Schwab）于2016年推出，指的是通过连接、实时数据、自动化、分析和人工智能/机器学习的进步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置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。

IoT

参见物联网。

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。

大型语言模型 (LLM)

一种基于大量数据进行预训练的深度学习 AI 模型。法学硕士可以执行多项任务，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

见 7 R。

小端序系统

一个先存储最低有效字节的系统。另请参见字节顺序。

LLM

参见大型语言模型。

下层环境

参见环境。

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

参见分支。

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。恶意软件的示例包括病毒、蠕虫、勒索软件、特洛伊木马、间谍软件和键盘记录器。

托管服务

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。亚马逊简单存储服务 (Amazon S3) Service 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

参见迁移加速计划。

机制

一个完整的过程，在此过程中，您可以创建工具，推动工具的采用，然后检查结果以进行调整。机制是一种在运行过程中自我增强和改进的循环。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

参见制造执行系统。

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 DevOps 人员和冲刺专业人员。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： EC2 使用 AWS 应用程序迁移服务重新托管向 Amazon 的迁移。

迁移组合评测（MPA）

一种在线工具，可提供信息，用于验证迁移到的业务案例。 AWS 云 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

用于将工作负载迁移到的方法 AWS 云。有关更多信息，请参阅此词汇表中的 7 R 条目和动员组织以加快大规模迁移。

ML

参见机器学习。

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，请参阅中的应用程序现代化策略。 AWS 云

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，请参阅中的评估应用程序的现代化准备情况 AWS 云。

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

参见迁移组合评估。

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？”

可变基础架构

一种用于更新和修改现有生产工作负载基础架构的模型。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

请参阅源站访问控制。

OAI

参见源访问身份。

OCM

参见组织变更管理。

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

参见运营集成。

OLA

参见运营层协议。

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

参见开放流程通信-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)

一份问题清单和相关的最佳实践，可帮助您理解、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 Well-Architecte AWS d Frame work 中的运营准备情况评估 (ORR)。

操作技术 (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）

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

参见运营准备情况审查。

OT

参见运营技术。

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

查看个人身份信息。

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

参见可编程逻辑控制器。

PLM

参见产品生命周期管理。

policy

一个对象，可以在中定义权限（参见基于身份的策略）、指定访问条件（参见基于资源的策略）或定义组织中所有账户的最大权限 AWS Organizations （参见服务控制策略）。

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。有关更多信息，请参阅在微服务中实现数据持久性。

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`或的查询条件`false`，通常位于子`WHERE`句中。

谓词下推

一种数据库查询优化技术，可在传输前筛选查询中的数据。这减少了必须从关系数据库检索和处理的数据量，并提高了查询性能。

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。

通过设计保护隐私

一种在整个开发过程中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

一种安全控制措施，旨在防止部署不合规的资源。这些控件会在资源配置之前对其进行扫描。如果资源与控件不兼容，则不会对其进行配置。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)

在产品的整个生命周期中，从设计、开发和上市，到成长和成熟，再到衰落和移除，对产品进行数据和流程的管理。

生产环境

参见环境。

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。

提示链接

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，以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

一系列步骤，例如指令，用于访问 SQL 关系数据库系统中的数据。

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

参见 “负责任、负责、咨询、知情”（RACI）。

RAG

请参见检索增强生成。

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

参见 “负责任、负责、咨询、知情”（RACI）。

RCAC

请参阅行和列访问控制。

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。

重新架构师

见 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

见 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 AWS 区域 您的账户可以使用的账户。

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

见 7 R。

版本

在部署过程中，推动生产环境变更的行为。

搬迁

见 7 R。

更换平台

见 7 R。

回购

见 7 R。

故障恢复能力

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

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

见 7 R。

退休

见 7 R。

检索增强生成（RAG）

一种生成式人工智能技术，其中法学硕士在生成响应之前引用其训练数据源之外的权威数据源。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据的过程。

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

参见恢复点目标。

RTO

参见恢复时间目标。

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

参见监督控制和数据采集。

SCP

参见服务控制政策。

secret

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 Secret s Manager 密钥中有什么？ 在 Secrets Manager 文档中。

安全性源于设计

一种在整个开发过程中考虑安全性的系统工程方法。

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。安全控制主要有四种类型：预防性、侦测、响应式和主动式。

安全加固

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

一种预定义和编程的操作，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 实例或轮换证书。

服务器端加密

在目的地对数据进行加密，由接收方 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。

服务级别指示器 (SLI)

对服务性能方面的衡量，例如其错误率、可用性或吞吐量。

服务级别目标 (SLO)

代表服务运行状况的目标指标，由服务级别指标衡量。

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

参见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

参见服务级别协议。

SLI

参见服务级别指标。

SLO

参见服务级别目标。

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，请参阅中的分阶段实现应用程序现代化的方法。 AWS 云

恶作剧

参见单点故障。

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交易数据或测量数据，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。

监控和数据采集 (SCADA)

在制造业中，一种使用硬件和软件来监控有形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。

系统提示符

一种向法学硕士提供上下文、说明或指导方针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，充当用于组织资源的元数据。 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

参见环境。

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

参见环境。

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

一个 SQL 函数，用于对一组以某种方式与当前记录相关的行进行计算。窗口函数对于处理任务很有用，例如计算移动平均线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。

蠕虫

参见一次写入，多读。

WQF

请参阅AWS 工作负载资格框架。

一次写入，多次读取 (WORM)

一种存储模型，它可以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，但他们无法对其进行更改。这种数据存储基础架构被认为是不可变的。

Z

零日漏洞利用

一种利用未修补漏洞的攻击，通常是恶意软件。

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。

零镜头提示

向法学硕士提供执行任务的说明，但没有示例（镜头）可以帮助指导任务。法学硕士必须使用其预先训练的知识来处理任务。零镜头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 few-shot 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。