使用容器和 Amazon API Gateway 逐步实现传统微软 ASP.NET (ASMX) 网络服务的现代化

AWS 规范性指导

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

AWS 规范性指导: 使用容器和 Amazon API Gateway 逐步实现传统微软 ASP.NET (ASMX) 网络服务的现代化

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

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

Table of Contents

简介 1
目标业务成果 1
ASP.NET 网络服务的类型 2
Strangler 无花果图案 3
REST-based ASP.NET 网络服务 4
SOAP-based ASP.NET 网络服务 (ASMX) 6
示例架构 8
其他考虑因素 9
身份验证和授权 9
领域驱动型设计 9
临时州和目标州 9
常见问题解答 11
是否 AWS 支持 Windows 容器？ 11
我能在 Linux 上运行 ASP.NET 服务吗？ 11
我可以在上面运行 Windows 容器 AWS Fargate吗？ 11
亚马逊 API Gateway 是否支持 Windows 集成的身份验证？ 它支持 OAuth 2.0 吗？ 11
后续步骤和资源 12
文档历史记录 13
..... xiv

使用容器和亚马逊 API Gateway 逐步实现传统微软 ASP.NET (ASMX) 网络服务的现代化

Dan Broadway，Amazon Web Services (AWS)

2021 年 11 月（文档历史记录）

本指南描述了在亚马逊 Web Services ()AWS上实现传统的 ASP.NET 网络服务现代化的方法和最佳实践。Strangler fig 模式是一种现代化方法，在这种方法中，传统系统的现有功能逐渐被新的应用程序和服务所取代，直到新系统最终取代遗留系统的所有旧部分或关键部分。随着功能的替换，可以移除或停用旧系统（通常是单片系统）的非活动部分。

该指南讨论了一种通过抽象方式修改的分支方法，这是一种用于逐步更改软件系统的常用技术，再加上服务委托，即服务将其行为推迟到另一个较新的服务，以帮助管理服务使用者的重大更改。

目标业务成果

本指南旨在帮助客户对基于表述性状态传输 (REST) 或简单对象访问协议 (SOAP) 标准的传统 ASP.NET 服务进行现代化改造。 AWS它侧重于需要采用渐进式现代化方法来管理交付风险的场景，而交付风险通常与解耦更大的单体应用程序有关。它还讨论了在不引入重大更改的情况下无法同步升级具有多个使用者的服务的场景。

本指南旨在帮助您：

- 在 ASP.NET 上迁移、现代化和扩展您的 ASP.NET 应用程序 AWS
- 管理与传统应用程序现代化相关的交付风险
- 解耦具有多个高可用消费者的服务

ASP.NET 网络服务的类型

当您计划对旧版 ASP.NET Web 服务进行现代化改造时 AWS，通常可以将这些服务分为两类：

- 基于 REST 的 ASP.NET 网络服务。它们是使用 ASP.NET Web API (RESTful) 或 ASP.NET Model-View-Controller (MVC) 模式构建的。
- 基于 SOAP 的 ASP.NET 网络服务。它们是使用 Windows 通信基金会 (WCF) 或 ASP.NET 网络服务 (ASMX) 文件和 ASP.NET 实现的。 WebMethods

REST 是当今网络中使用的主要建筑风格 APIs。因此，使用这种方法实现的服务与大多数现代工具和框架兼容，因此它们具有更直接的现代化路径。使用 API 网关和 API 管理器可以促进这些服务的现代化，主要问题涉及服务如何对它们所代表的资源和业务域进行建模，以及如何保护服务。但是，依赖于 SOAP 的 Web 服务通常早于基于 REST 的服务，因此目前的支持有限。这些服务的迁移涉及与基于 REST 的服务相同的问题。他们还面临着额外的挑战，那就是工具和框架支持很少甚至根本没有。

Strangler 无花果图案

在对大型整体系统进行现代化改造或改写时，马丁·福勒（Martin Fowler）引入了勒死者无花果模式，以此作为管理风险的一种方式。这种模式比喻了一种植物，这种植物的生命起初是一棵葡萄树，与一棵较老的、成熟的树木一起生长。随着葡萄树的生长，它会蔓延以完全消耗并最终取代宿主树，取而代之的是新的、更扼杀的无花果树。在实现 ASP.NET Web 服务现代化的背景下，这种模式通过在其他系统依赖于 Web 服务的情况下建立代理来逐步取代系统功能。最初，您可以将这些代理视为具有直通行为，因为它们的实现是由现有的整体应用程序服务实现的。在自然类比中，这是勒死者无花果最初向宿主树的树干送一棵葡萄树的时候。然后，创建一个与整体分离的新服务，并将代理的实现推迟到该新服务。在自然的类似物中，这是勒死无花果藤缠绕在树的一根树枝上并超越树枝的时候。这种代理然后用新服务替换代理实现的模式一直持续到旧系统的所有功能都迁移到新服务为止。此时，勒死者无花果藤完全消耗了树木，遗留系统可以退役。

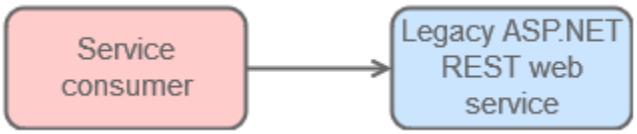
使用 strangler fig 模式时，请遵循以下最佳实践，这样您就可以更顺畅地独立扩展和部署应用程序：

- 选择具有良好测试覆盖率且与之相关的技术债务较少的组件。从这个组件开始可以让团队在现代化过程中充满信心。
- 选择具有可扩展性要求的组件，然后从其中一个组件开始。
- 选择业务需求变化频繁且部署频繁的组件。
- 要大规模实施这种模式 AWS，请在亚马逊弹性容器服务 (Amazon ECS) 中运行的 Windows 容器中部署重构后的 ASMX 服务，然后使用 Amazon API Gateway 发布现代化的 REST API。

REST-based ASP.NET 网络服务

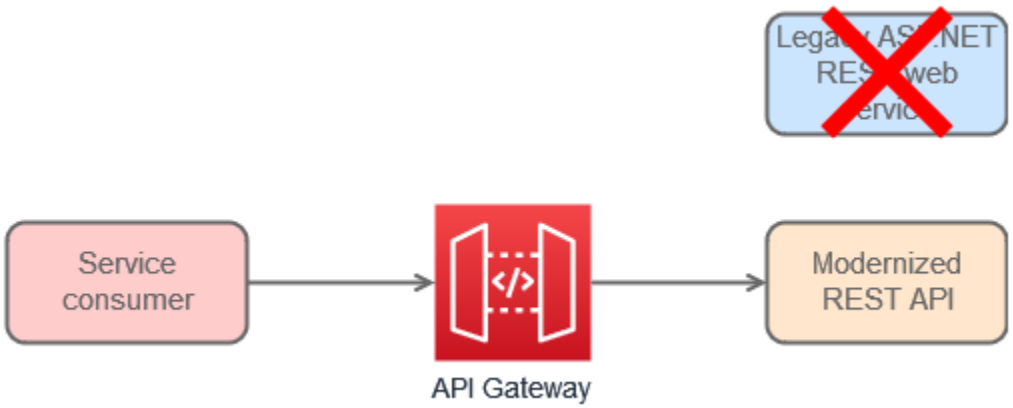
当您使用绞尽脑汁模式对 REST-based ASP.NET AWS 服务进行现代化改造时，我们建议您使用 Amazon API Gateway 来建立用于将流量转移到新服务的代理。您可以引入 API Gateway 端点作为服务使用者和正在进行现代化的传统服务之间的中介。如果旧版服务已开启 AWS，则将 API Gateway 端点配置为将请求路由到旧版 REST 服务。如果该服务尚未开启 AWS，则可以在建立新的 API Gateway 代理之前按原样迁移该服务。如果无法做到这一点，则可以采用混合云方法，使用 AWS 连接服务，例如将 API Gateway Direct Connect 连接到本地数据中心。下图描绘了引入 API Gateway 作为两者之间的代理之前和之后的 R ASP.NET EST 服务及其使用者。

在引入代理之前：



将 API Gateway 添加为代理后：

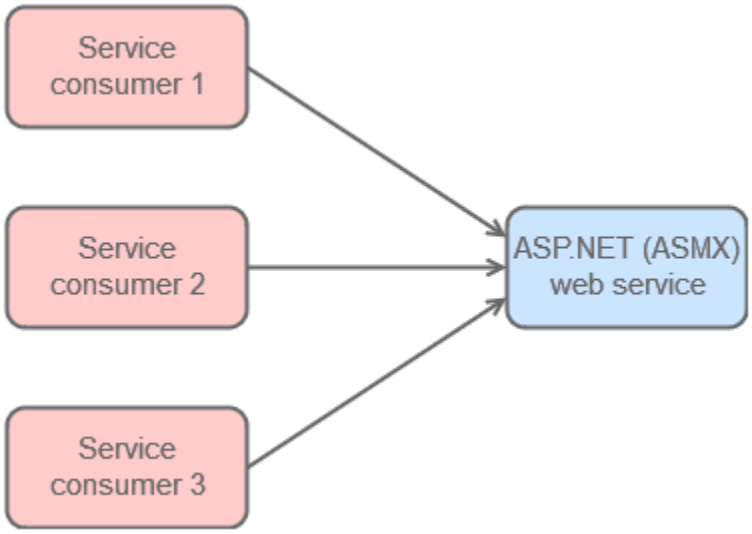
当 API Gateway 代理到位后，您可以使用 Amazon ECS 创建和部署现代化服务，从而实现高度可扩展和可用性的服务。 AWS 创建并测试代理和新近现代化的服务后，您可以重新配置 API Gateway 端点，使其指向现代化的 REST API 进行实现。



如果新近现代化的服务的 API 合同与使用系统所依赖的传统代理合同不同，则可以使用 API Gateway 的数据转换功能。使用旧系统架构构造的传入 API 请求可以映射并转换为新服务的合同。

SOAP-based ASP.NET 网络服务 (ASMX)

当你将传统 ASP.NET Web 服务 (ASMX) 现代化为 REST API 时 AWS，它可能有依赖的使用者无法同时升级，而且由于相关的风险，他们很可能不应该同时升级。下图描绘了一个 ASMX Web 服务，该服务有三个依赖的使用者。

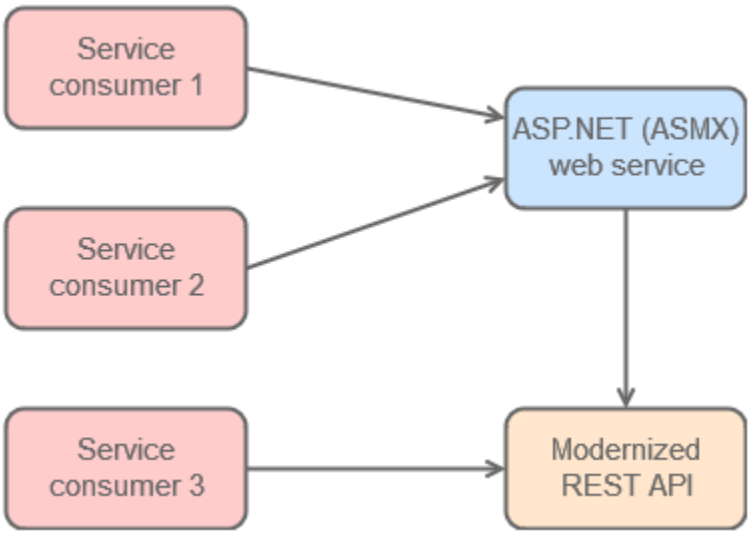


由于这些使用者使用 SOAP 协议调用 ASMX 服务，因此用 REST API 替换 ASMX 服务会导致这些使用者中断。此外，在将系统从整体架构转换为微服务的同时，将 ASMX 服务重构为 REST API 很可能涉及重塑系统行为和域模型。这可能会给消费者带来额外的突破性变化。

为了保持与旧服务的向后兼容性，从而使系统可以逐步实现现代化，你可以实施strangler fig方法的指导。在 strangler fig 方法中，与重要遗留系统现代化相关的风险在一定程度上是通过用新系统逐步取代旧系统来管理的。

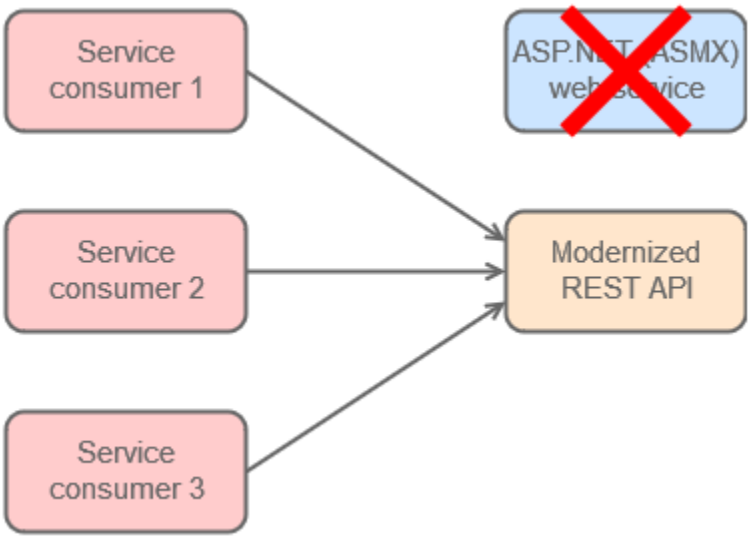
抽象分支是马丁·福勒（Martin Fowler）引入的另一种方法，它是实现勒死者无花果模式的有效技术。与 strangler fig 方法一样，当必须更换主要功能或对其进行现代化改造时，逐个抽象分支可以帮助工程师对系统进行渐进和渐进的更改。在这种方法中，在实现特定功能的代码和使用该功能的代码（即客户端代码）之间引入了一个抽象层。这个抽象层不一定是可以由具体类继承的抽象类型。相反，它可以是任何捕获消费者和提供者之间功能契约的实现。

对于传统的 SOAP-based ASP.NET Web 服务，您可以通过一种受抽象技术分支启发的方法，通过保持兼容性，实现服务使用者从旧到新的逐步迁移。但是， SOAP-based ASP.NETWeb 服务的方法使用委托而不是继承。通过重构传统的 ASMX 服务，将其实现委托给新的 REST API，可以实现与 ASMX 使用者的兼容。下图描述了将其实现委托给新的 REST API 的旧版 ASMX 服务。此委派允许服务使用者 3 升级到新的 REST API，而服务使用者 1 和 2 则独立升级。



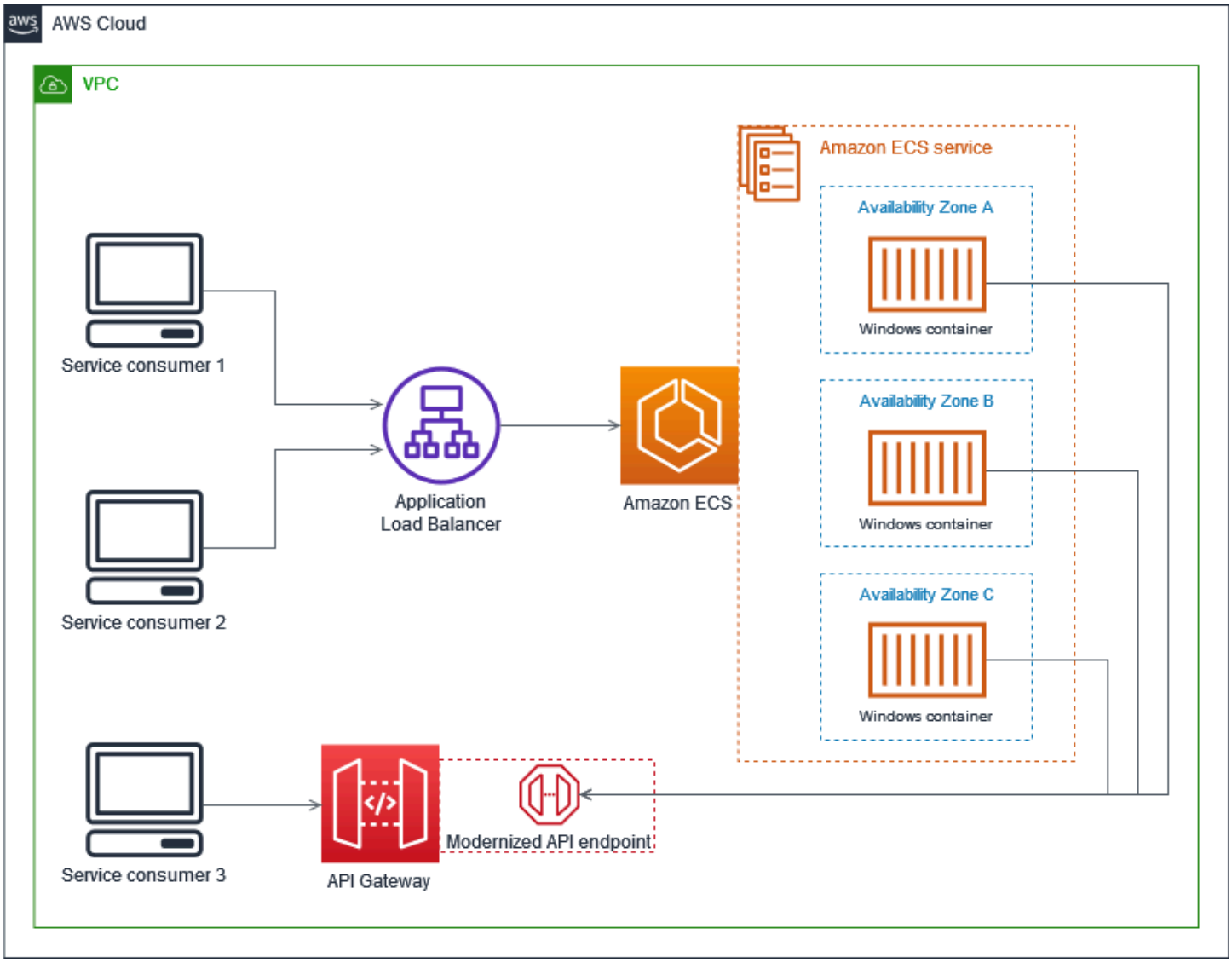
ASMX 服务重构的性质取决于该服务的功能和非功能（跨功能）要求。在某些情况下，重构可能涉及从旧的身份验证和授权机制转换为新的身份验证和授权机制，将服务 ASMX 有效负载从 XML 转换为 JSON，然后使用转换后的有效负载调用 REST API。在更复杂的场景中，重构后的 ASMX 服务可能需要协调对多个 REST API 的调用并保持状态。

将服务使用者从传统的 ASMX 服务迁移到现代化的 REST API 的过程会逐步进行，直到所有服务使用者都已更新为引用新 API 为止。下图描绘了完全现代化的状态，即所有服务使用者都引用 REST API，而 ASMX 服务由于不再需要而已停用。



示例架构

SOAP-based ASMX 服务的最终状态架构显示所有使用现代化 REST API 的服务使用者，从而导致传统 ASMX Web 服务停用。但是，这种模式的重点不是最终状态架构，而是在消费者升级期间迁移期间所需的临时状态架构。下图说明了这种临时状态。在这个临时架构中，服务使用者 1 和 2 仍在使用传统的 ASMX 服务，该服务已经过重构，将其实现委托给现代化的 REST API。重构后的 Windows 服务作为 Windows 容器部署在 Amazon ECS 中，跨三个可用区进行配置以实现高可用性，并通过应用程序负载均衡器进行访问。服务使用者 3 已更新，可以直接通过 API Gateway 使用新的 REST API。



其他考虑因素

到目前为止，我们已经讨论了使用 API Gateway 和 Windows 容器对旧版 ASP.NET 网络服务进行 AWS现代化改造。以下是需要考虑的其他一些注意事项：

- 安全性
- 重塑服务域
- 当有许多服务需要现代化时，排序 Web 服务升级

以下各节将讨论这些内容。

身份验证和授权

现代网络服务 APIs 通常使用基于令牌（JSON Web Token 或 JWT）的身份验证和授权标准，例如 2.0 OAuth 和 OpenID Connect，而传统的 ASP.NET 网络服务传统上依赖于 Windows Active Directory 域的基本身份验证或 Windows 集成的身份验证。作为最佳实践，如果新旧身份验证和授权方法不兼容，我们建议您在对 Web 服务进行现代化改造之前，先升级这些安全机制 AWS。尝试映射身份或尝试在新旧系统之间安全地传输安全状态可能不是一件容易的事，但可能会带来安全漏洞。

领域驱动型设计

在将整体分解为单个服务时，域驱动设计 (DDD) 是一种最佳实践，通常用于将系统建模为有凝聚力的业务领域。DDD 是一种通过将实现与不断演变的核心业务概念模型联系起来来开发满足复杂需求的软件的方法。其前提是将项目的主要重点放在核心领域和领域逻辑上，并将系统设计建立在反映业务的模型之上。如果您在对现有的单体应用程序进行现代化改造时使用 DDD，则需要从整体式应用程序的功能和用户流程向后推进。您可以将 DDD 与 Strangler 模式结合使用，以指导打破整体结构并确定要扼杀哪些服务的过程，因此被称为 “域驱动设计”。

临时州和目标州

在对多个 ASP.NET Web 服务进行现代化改造时 AWS，最好先定义所有服务现代化后的目标状态架构。但是，目标状态架构不一定是最终状态或最终状态，因为业务环境经常变化，系统目标状态应根据需要进行更新，以与业务目标保持一致。定义目标状态后，您可以从那里向后移动，定义逐步实现目标状态愿景的临时状态架构。你可以把这些临时状态架构想象成扼杀无花果树在遇到宿主树的大部分并吞没宿主树的大部分时向树上移动。临时状态架构通常会引入临时或重叠的结构，这些结构不会成为最终

状态架构的一部分，以促进向目标状态的演变。基于 SOAP 的 ASP.NET Web 服务的现代化就是一个例子：暂时引入了基于 Windows 的容器，以弥合依赖传统服务的调用系统之间的差距，直到它们可以升级到新的 REST API。

常见问题解答

本节提供了有关对旧版 ASP.NET Web 服务进行现代化改造的常见问题的答案。

是否 AWS 支持 Windows 容器？

是的，亚马逊弹性容器服务 (Amazon ECS) 和亚马逊 Elastic Kubernetes Service（亚马逊 EKS）都支持 Windows 容器。要开始使用 Windows 容器使用这些 AWS 服务，请参阅以下内容：

- 使用亚马逊 EC2 Windows 容器开始使用亚马逊 ECS 控制台（亚马逊 ECS 文档）
- 亚马逊 EKS Windows Container Support 现已正式上AWS 线（新闻博客）

我能在 Linux 上运行 ASP.NET 服务吗？

是的，用.NET Core 编写的 ASP.NET 服务 (APIs) 可以在 Linux 上运行。要使用在 Linux 上运行.NET 核心应用程序 AWS Elastic Beanstalk，请参阅 Elastic Be anstalk 文档中的在 Linux 上使用.NET Core。

我可以在上面运行 Windows 容器 AWS Fargate吗？

在撰写本指南时，只有亚马逊 EC2 启动类型支持 Windows 容器。对于 Windows 容器，Fargate 启动类型目前不受支持。有关更多信息，请参阅亚马逊 ECS 文档中的亚马逊 EC2 Windows 容器。

亚马逊 API Gateway 是否支持 Windows 集成的身份验证？ 它支持 OAuth 2.0 吗？

默认情况下不是。但是，API Gateway 通过使用 Amazon Cognito 用户池支持 OAuth 2.0，并通过使用自定义授权方支持其他身份验证和授权提供商。要开始使用这些选项，请参阅以下内容：

- 在 Amazon API Gateway 中引入自定义授权方（AWS 计算博客）
- 使用 Amazon Cognito 用户池作为授权方控制对 REST API 的访问（API Gateway 文档）

后续步骤和资源

要开始旧版 ASP.NET 应用程序的迁移和现代化过程，请执行以下操作：

1. 确定要迁移到的 ASP.NET 网络服务的类型。 AWS从现代化的角度来看，ASP.NET Web 服务可以分为三类。本指南中讨论的前两个类别是使用 ASMX 文件或使用 Windows 通信基金会 (WCF) 编写的基于 SOAP 的 ASP.NET Web 服务，以及使用 ASP.NET Web 或 ASP.NET (MVC) 模式编写的基于 REST 的 RESTful ASP.NET 服务。 APIs Model-View-Controller第三种类型的 ASP.NET Web 服务是使用 ASP.NET MVC 和.NET Core 或.NET 5 编写的。这是当前推荐的编写 ASP.NET REST 的方法，迁移到 ASP.NET RES APIs T 不需要大量的现代化改造。 AWS
2. 确定如何保护这些服务的身份验证和授权，确定您的方法，并确定现代化工作的这一方面的优先顺序。
3. 熟悉 Windows 容器、亚马逊 ECS 和 Amazon API Gateway。
4. 阅读有关领域驱动的设计的信息。这将确保现代化服务围绕有凝聚力的功能域而不是传统系统的设计进行建模，从而帮助您充分利用现代化工作。
5. 建立目标状态架构，然后向后推进，以确定实现该结果所需的临时状态架构。

有关本指南中讨论的模式的更多信息，请参阅 Martin Fowler 的以下文章：

- Strangler fig应用程序
- 按抽象分支
- 领域驱动型设计

有关本指南中讨论的 AWS 服务的更多信息，请参阅以下参考资料：

- Amazon Elastic Container Service（Amazon ECS）
- Amazon API Gateway

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	—	2021 年 11 月 15 日

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