选择实现.NET 应用程序现代化的方法

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: 选择实现.NET 应用程序现代化的方法

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
决策矩阵 2
重新托管 4
使用案例 4
优点 4
缺点 4
AWS 服务 4
工具 5
部署决策 5
将平台改造成一个 Windows 容器 6
使用案例 6
优点 6
缺点 6
AWS 服务 6
工具 7
部署决策 7
Re-architecting 作为 Linux 容器 9
使用案例 9
优点 9
缺点 9
AWS 服务 9
工具 10
部署决策 10
Re-architecting 作为 Linux 容器中的微服务 12
使用案例 12
优点 12
缺点 12
AWS 服务 12
工具 13
部署决策 13
Re-architecting 作为没有容器的微服务 15
使用案例 15
优点 15
缺点 15

AWS 服务 16
工具 16
部署决策 16
后续步骤和资源 17
文档历史记录 18
..... xix

选择对.NET 应用程序进行现代化改造的方法

Mathew George 和 Fabian Jahnke，Amazon Web Services ()AWS

2023 年 8 月（文档历史记录）

为.NET 传统应用程序选择正确的现代化策略可能是一个复杂的决定。本指南为想要了解将.NET 传统应用程序迁移到 Amazon Web Services (AWS) 并在 AWS 云中对其进行现代化改造的方法的技术决策者提供了最佳实践。

现代化涉及通过结合现代基础架构、架构和组织模式，对传统企业应用程序进行平台化改造或重构。现代化有助于最大限度地提高弹性、工程效率、业务灵活性和卓越运营。

.NET 开发者平台已从.NET 框架演变为.NET Core 和.NET 5（及更高版本）。您可以对.NET 传统应用程序进行现代化改造，并利用 Linux 操作系统的性能、成本节约和强大的生态系统，也可以从.NET Framework 切换到.NET Core 或.NET 5（或更高版本）。

本文档中提供的最佳实践有助于指导您的.NET 应用程序迁移和现代化工作。该指南讨论了可能的迁移和现代化策略、限制条件以及您可以使用的 AWS 服务。您的选择包括在云中重新托管（移动）.NET 应用程序，以及容器化、分解为微服务以及采用无服务器架构。

决策矩阵

下表根据您的用例和资源总结了传统.NET 应用程序的迁移和现代化选项。

使用案例	迁移策略和架构				
	重新托管	将平台重新设置为 Windows 容器	Re-architect 作为 Linux 容器	Re-architect 作为 Linux 容器中的微服务	Re-architect 作为没有容器的微服务
你有重构的资源。	⊗否	⊗否	⊘是	⊘是	⊘是
您的.NET 旧版应用程序一直在使用。	⊘是	⊘是	⊘是	⊘是	⊗否
您可以解析.NET 框架依赖关系。	⊗否	⊗否	⊘是	⊘是	⊘是
你可以移除 Windows 依赖关系。	⊗否	⊗否	⊘是	⊘是	⊘是
你想在亚马逊弹性计算云 (Amazon EC2) 实例上将应用程序作为原生 Windows 应用程序运行。	⊘是	⊗否	⊗否	⊗否	⊗否
您的代码可以从.NET 框架移植到.NET	⊗否	⊗否	⊘是	⊘是	⊘是

核心或.NET 6。					
你想拆分单片应用程序。	⊗否	⊗否	⊗否	✓是	✓是

以下各节详细描述了这些选项：

- 重新托管
- 将平台改造成一个 Windows 容器
- 将平台改造成一个 Linux 容器
- Re-architecting 作为 Linux 容器中的微服务
- Re-architecting 作为没有容器的微服务

重新托管

再托管（移动）是在不进行修改的情况下将本地应用程序迁移到云端的过程。此策略主要用于迁移大型应用程序以满足特定的业务目标，例如在更短的时间内推出产品或离开本地数据中心。这些应用程序将重新托管在满足您迁移应用程序要求的亚马逊弹性计算云 (Amazon EC2) Windows 实例上。

使用案例

此迁移策略在以下任何情况下都很有用：

- 旧版.NET 应用程序必须作为本机 Windows 应用程序运行。
- 没有时间和资源来实现应用程序现代化。
- 传统的.NET 应用程序是商用现成的 (COTS) 应用程序。

优点

与本地.NET 应用程序相比，重新托管具有以下优势：

- 工作量最少，因为它不需要更改代码或架构
- 降低成本
- 更好的合规性和安全性，因为它使用了 AWS 基础架构和安全最佳实践

缺点

- 没有充分利用云的性能、可扩展性和弹性 AWS 选项
- 难以与最先进的云服务集成

AWS 务

- Amazon EC2
- AWS Elastic Beanstalk
- AWS Transform MGN

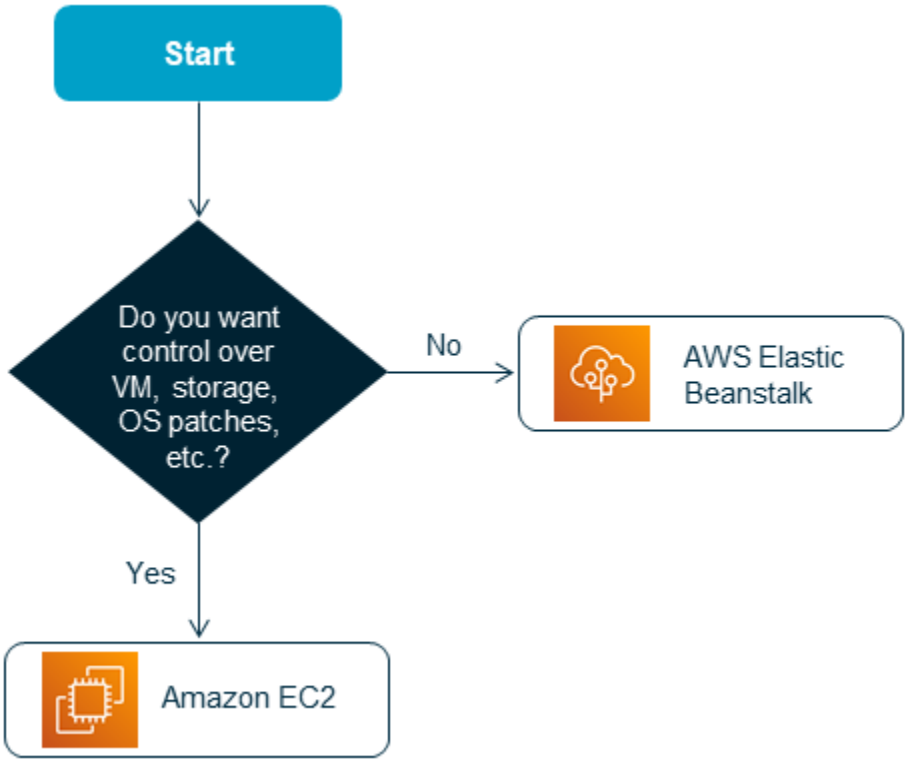
工具

工具	目的	资源
Windows Web 应用程序迁移助手	该工具是一个交互式 PowerShell 脚本，可将整个网站及其配置迁移到 Elastic Beanstalk。	将 ASP.NET 应用程序迁移到 Elastic Be AWS anstalk（博客文章）

部署决策

您可以从两个部署选项中进行选择：

- 如果您想完全控制计算环境的配置，包括内存和存储设置，并控制操作系统补丁，请将.NET 应用程序迁移到 Amazon EC2。
- 如果您不需要完全控制基础架构，请使用 Elastic Beanstalk。Elastic Beanstalk 会自动为您的应用程序设置托管环境。



将平台改造成一个 Windows 容器

将.NET 应用程序重构为 Windows 容器可以帮助您以比重构更少的精力实现业务目标。它允许您在不更改.NET 应用程序的核心架构的情况下利用容器技术。Windows 应用程序可以毫不费力地转换为容器。

.NET Framework-based 容器支持 Windows Server 2016 或 2019 作为主机操作系统。

使用案例

此迁移策略在以下任何情况下都很有用：

- 您无法解析.NET 框架依赖关系。
- 你无法解析 Windows 依赖关系。
- 您没有资源将应用程序重构为.NET Core 或.NET 6。

优点

与本地.NET 应用程序相比，这种迁移方法具有以下优点：

- 最少的努力
- 提高资源利用率
- 提高了安全性
- 更好的部署选项

缺点

- 主机 Windows 操作系统的许可证成本

AWS 务

用于存储容器镜像：

- Amazon Elastic Container Registry（Amazon ECR）

要编排 Windows 容器，请执行以下操作：

- Amazon Elastic Container Service（Amazon ECS）
- Amazon Elastic Kubernetes Service(Amazon EKS)
- 亚马逊 EC2 托管带有 Windows 容器的 Docker

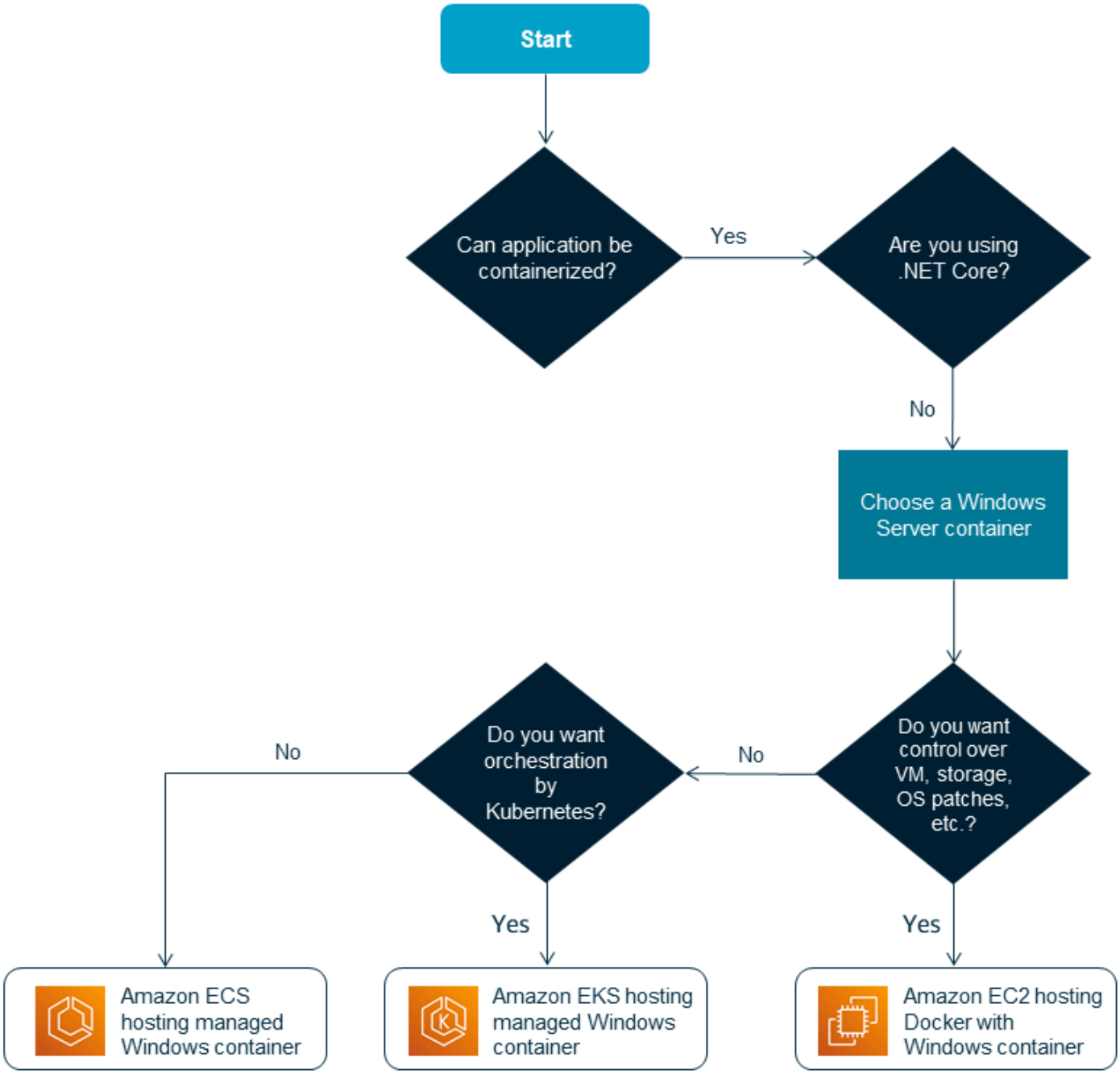
工具

工具	目的	资源
AWS App2Container (A2C)	A2C 是一款命令行工具，可轻松将.NET 和 Java 应用程序转换为容器化应用程序，从而实现这些应用程序的现代化。	• 详细信息 • 文档

部署决策

您可以从三个部署选项中进行选择：

- 如果您想完全控制计算环境的配置，包括内存和存储设置，并控制操作系统补丁，请将您的应用程序作为 Windows 容器部署在 EC2 实例上。
- 如果你想让容器由 Kubernetes 管理：在亚马逊 EKS 上将你的应用程序作为 Windows 容器部署。
- 如果您希望容器由 Amazon ECS 管理，请将您的应用程序作为 Windows 容器部署到亚马逊 ECS 上。


Start
Can application be containerized?
Yes
Are you using .NET Core?
No
Choose a Windows Server container
Do you want control over VM, storage, OS patches, etc.?
No
Do you want orchestration by Kubernetes?
No
Yes
Yes
Amazon ECS hosting managed Windows container
Amazon EKS hosting managed Windows container
Amazon EC2 hosting Docker with Windows container

Re-architecting 作为 Linux 容器

通过将.NET Framework 应用程序移植到.NET Core 或.NET 6，可以在多个平台上运行应用程序，降低许可成本，提高性能并提高可扩展性。

使用案例

此迁移策略在以下任何情况下都很有用：

- 您有足够的资源和时间来重构应用程序。
- 您可以解析所有.NET 框架依赖关系。
- 你有一个长时间运行的应用程序。

优点

与本地.NET 应用程序相比，这种迁移方法具有以下优点：

- 降低总拥有成本（TCO）
- 提高了安全性和性能
- 加速创新
- 转换为云原生应用程序的好处
- Open-source

缺点

- 重构的工作量和成本

AWS 务

用于存储容器镜像：

- Amazon ECR

对于编排容器：

- 亚马逊 ECS，或带有 AWS Fargate
- 亚马逊 EKS，或者带有 Far gate 的亚马逊 EKS

AWS Fargate 是一款无服务器、按使用量付费的计算引擎，可让您专注于构建应用程序，而无需管理服务器。Fargate 与亚马逊 ECS 和亚马逊 EKS 兼容。

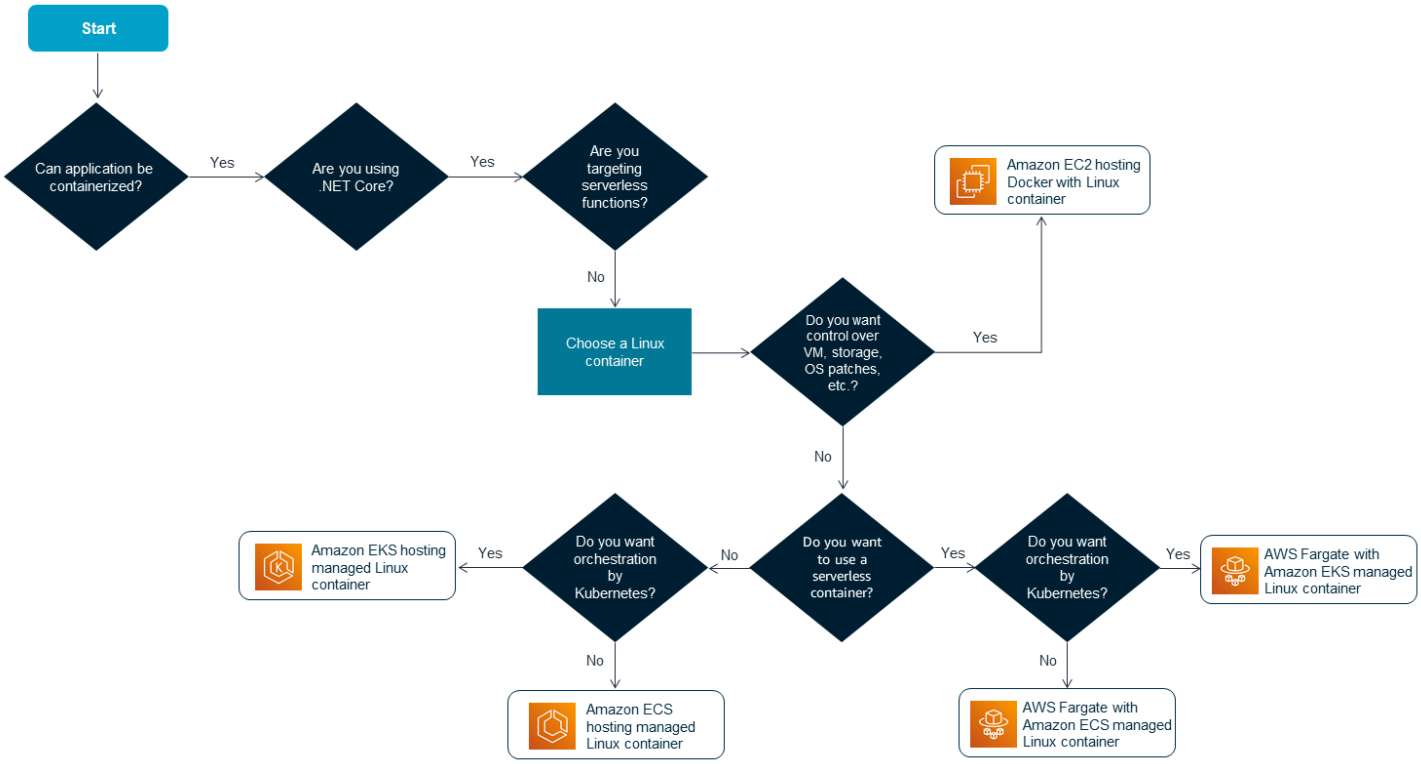
工具

工具	目的	资源
Porting Assistant for .NET	此分析工具扫描.NET Framework 应用程序并生成.NET Core 兼容性评估。该评估可帮助您更快地将应用程序移植到 Linux。	• 详细信息 • 文档
AWS App2Container (A2C)	A2C 是一款命令行工具，可轻松将.NET 和 Java 应用程序转换为容器化应用程序，从而实现这些应用程序的现代化。	• 详细信息 • 文档

部署决策

您可以从五个部署选项中进行选择：

- 如果您想完全控制计算环境的配置，包括内存和存储设置，并控制操作系统补丁，请将您的应用程序作为 Linux 容器部署在 EC2 实例上。
- 如果您希望容器由 Kubernetes 管理并作为无服务器容器运行，请使用 Fargate 将您的应用程序作为 Linux 容器部署到亚马逊 EKS 上。
- 如果您希望容器由 Amazon ECS 管理并作为无服务器容器运行，请使用 Fargate 将您的应用程序作为 Linux 容器部署到亚马逊 ECS 上。
- 如果您希望容器由 Kubernetes 管理，但又想自己管理容器的计算资源，请将您的应用程序作为 Linux 容器部署在 Amazon EKS 上。
- 如果您希望容器由 Amazon ECS 管理，但又想自己管理容器的计算资源，请将您的应用程序作为 Linux 容器部署在 Amazon ECS 上。


Start
Can application be containerized?
Yes
Are you using .NET Core?
Yes
Are you targeting serverless functions?
No
Choose a Linux container
Do you want control over VM, storage, OS patches, etc.?
Yes
Amazon EC2 hosting Docker with Linux container
No
Do you want to use a serverless container?
Yes
Do you want orchestration by Kubernetes?
Yes
AWS Fargate with Amazon EKS managed Linux container
No
AWS Fargate with Amazon ECS managed Linux container
No
Do you want orchestration by Kubernetes?
Yes
Amazon EKS hosting managed Linux container
No
Amazon ECS hosting managed Linux container

Re-architecting 作为 Linux 容器中的微服务

微服务架构是一种将单个应用程序开发为一套小型服务的方法。每项服务都在自己的进程中运行，并通过轻量级机制与其他服务进行通信。这种方法将单体应用程序分解为较小的服务，其中每项服务仅用于单一目的并作为容器部署。

使用案例

在以下情况下，此迁移策略非常有用：

- 您想将您的整体系统分解为微服务。
- 您有足够的资源和时间进行重构。
- 您可以解析所有.NET 框架依赖关系。
- 你有一个长时间运行的应用程序。

优点

与本地.NET 应用程序相比，这种迁移方法具有以下优点：

- 更快的创新，因为在微服务架构中添加新功能更容易
- 高可用性和可靠性
- 提高敏捷性和按需扩展性
- 独立部署、现代持续集成和持续部署 (CI/CD) 管道
- 强大的模块界限和技术多样性

缺点

- 重构的工作量和成本
- 潜在的操作复杂性

AWS 务

您可以使用以下 AWS 服务来开发基于微服务的系统：

- Amazon API Gateway
- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Queue Service (Amazon SQS)
- Amazon ECS
- Amazon EKS
- AWS Lambda
- AWS Fargate
- CloudFormation 或 AWS Cloud Development Kit (AWS CDK)
- AWS Identity and Access Management (IAM)
- Amazon Simple Storage Service（Amazon S3）
- Amazon ECR

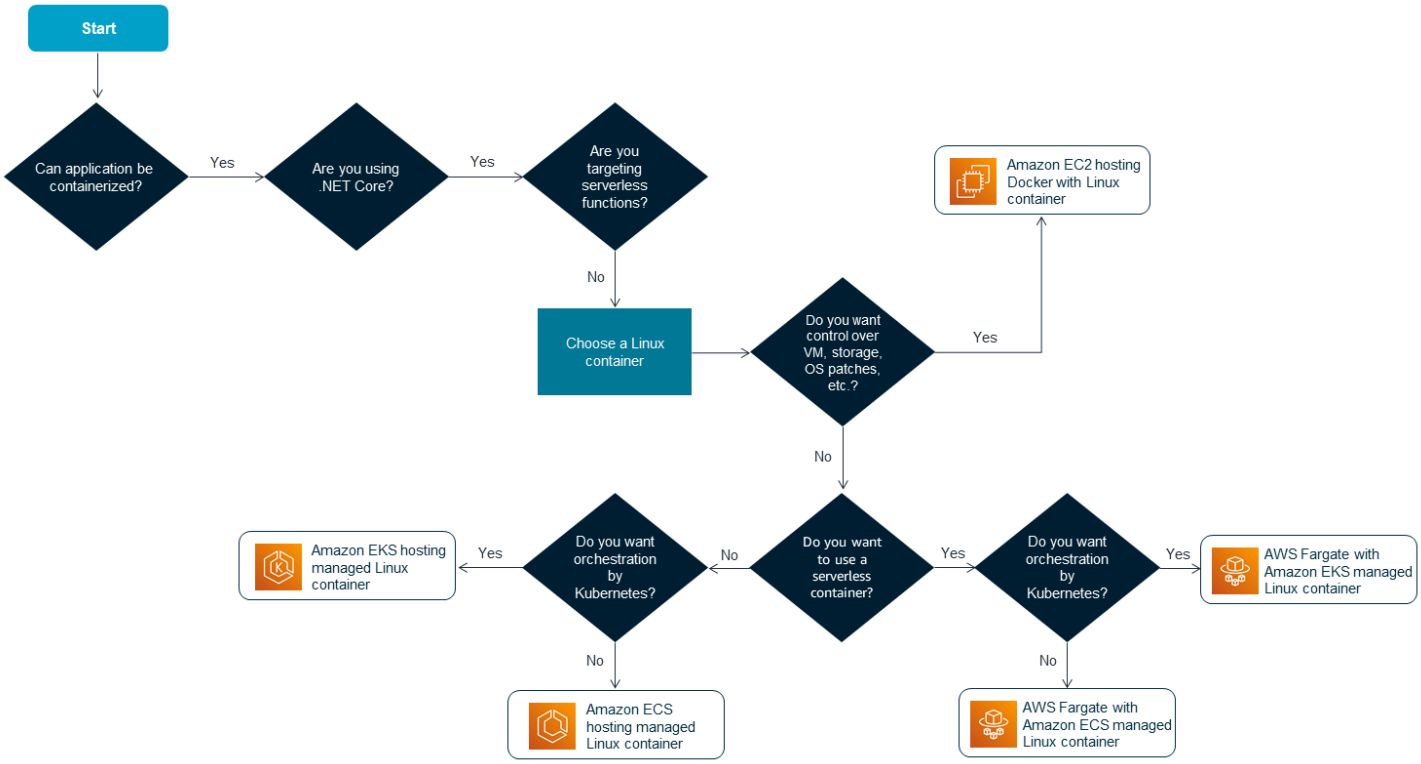
工具

AWS Professional Services 提供自定义工具和服务，帮助您将整体应用程序重构为微服务。

部署决策

您可以从五个部署选项中进行选择：

- 如果您想完全控制计算环境的配置，包括内存和存储设置，并控制操作系统补丁，请将您的应用程序作为 Linux 容器部署在 EC2 实例上。
- 如果您希望容器由 Kubernetes 管理并作为无服务器容器运行，请使用 Fargate 将您的应用程序作为 Linux 容器部署到亚马逊 EKS 上。
- 如果您希望容器由 Amazon ECS 管理并作为无服务器容器运行，请使用 Fargate 将您的应用程序作为 Linux 容器部署到亚马逊 ECS 上。
- 如果您希望容器由 Kubernetes 管理，但又想自己管理容器的计算资源，请将您的应用程序作为 Linux 容器部署在 Amazon EKS 上。
- 如果您希望容器由 Amazon ECS 管理，但又想自己管理容器的计算资源，请将您的应用程序作为 Linux 容器部署在 Amazon ECS 上。


Start
Can application be containerized?
Yes
Are you using .NET Core?
Yes
Are you targeting serverless functions?
No
Choose a Linux container
Do you want control over VM, storage, OS patches, etc.?
Yes
Amazon EC2 hosting Docker with Linux container
No
Do you want to use a serverless container?
Yes
Do you want orchestration by Kubernetes?
Yes
AWS Fargate with Amazon EKS managed Linux container
No
AWS Fargate with Amazon ECS managed Linux container
No
Do you want orchestration by Kubernetes?
Yes
Amazon EKS hosting managed Linux container
No
Amazon ECS hosting managed Linux container

Re-architecting 作为没有容器的微服务

AWS Lambda 是一项无服务器计算服务，无需预置或管理服务器、创建可感知工作负载的集群扩展逻辑、维护事件集成或管理运行时，即可运行代码。只有在需要时 Lambda 才运行您的函数，并且能自动扩展，从每天几个请求扩展到每秒数千个请求。您只需按消耗的计算时间付费 - 代码未运行时不产生费用。在这种方法中，将单体应用程序分解为较小的服务，其中每项服务都用于单一用途。如果服务不是持续运行，则可以将其实现为 Lambda 函数；否则，该服务应在容器中运行。

使用案例

您可以在以下场景中使用此迁移策略：

- 您想将您的整体系统分解为微服务。
- 您有足够的资源和时间进行重构。
- 您可以解析所有.NET 框架依赖关系。
- 您的应用程序不会持续运行；它们运行的时间很短。

优点

与本地.NET 应用程序相比，这种迁移方法具有以下优点：

- 更快的创新，因为在微服务架构中添加新功能更容易
- 高可用性和可靠性
- 提高灵活性和按需扩展性
- 独立部署和现代化 CI/CD 管道
- 强大的模块界限和技术多样性
- 节省成本
- 减少基础设施配置工作

缺点

- 重构的工作量和成本
- 潜在的操作复杂性

- 不支持长时间运行的应用程序

AWS 务

以下是您可以用来开发微 AWS 服务架构的一些重要服务，包括： AWS Lambda

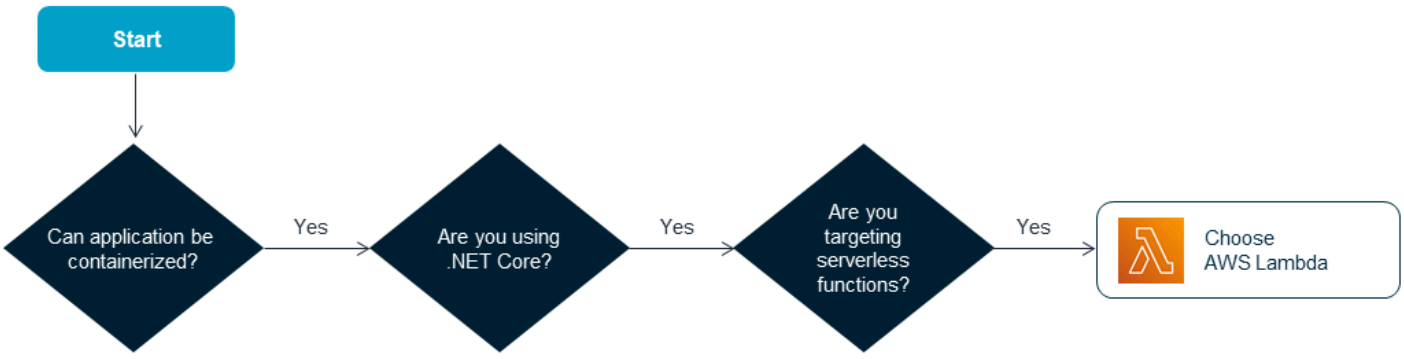
- Amazon API Gateway
- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Queue Service (Amazon SQS)
- AWS Lambda
- CloudFormation 或 AWS CDK
- IAM
- Amazon S3

工具

AWS Professional Services 提供自定义工具和服务，帮助您将整体应用程序重构为微服务。

部署决策

这种迁移和现代化方法得到以下方面的支持 AWS Lambda



后续步骤和资源

为.NET 应用程序选择迁移和现代化策略后，请使用以下资源评估您的企业和应用程序的准备情况，并开始迁移过程：

- 实现云端应用程序现代化的策略 AWS
- 评估迁移准备情况
- 评估 AWS 云端应用程序的现代化准备情况
- AWS 移民加速计划 (MAP)

迁移工具

- Windows Web 应用程序迁移助手
- AWS App2Container (A2C)
- Porting Assistant for .NET

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
再托管服务	已 AWS Transform MGN 添加到重新托管部分。	2023 年 8 月 7 日
.NET	更新了.NET 引用以反映最新版本。	2021 年 12 月 9 日
初次发布	—	2021 年 11 月 3 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。