Auswahl eines Ansatzes für die Modernisierung von.NET-Anwendungen

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Auswahl eines Ansatzes für die Modernisierung von.NET-Anwendungen

Table of Contents

Auswahl eines Ansatzes für die Modernisierung von.NET-Anwendungen

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

August 2023 (Dokumentverlauf)

Die Wahl der richtigen Modernisierungsstrategie für ältere .NET-Anwendungen kann eine komplexe Entscheidung sein. Dieser Leitfaden bietet bewährte Methoden für technische Entscheidungsträger, die die Ansätze für die Migration ihrer veralteten NET-Anwendungen zu Amazon Web Services (AWS) und deren Modernisierung in der AWS Cloud verstehen möchten.

Bei der Modernisierung werden ältere Unternehmensanwendungen durch die Kombination moderner Infrastruktur-, Architektur- und Organisationsmuster auf eine neue Plattform oder ein Refactoring umgestaltet. Die Modernisierung trägt dazu bei, Ausfallsicherheit, technische Effizienz, geschäftliche Flexibilität und betriebliche Exzellenz zu maximieren.

Die.NET-Entwicklerplattform hat sich vom .NET Framework zu .NET Core und.NET 5 (und späteren Versionen) weiterentwickelt. Sie können Ihre älteren .NET-Anwendungen modernisieren und die Leistung, die Kosteneinsparungen und das robuste Ökosystem des Linux-Betriebssystems nutzen, oder indem Sie von.NET Framework zu .NET Core oder .NET 5 (oder höher) wechseln.

Die in diesem Dokument aufgeführten bewährten Methoden helfen Ihnen bei der Migration und Modernisierung von.NET-Anwendungen. In diesem Leitfaden werden mögliche Migrations- und Modernisierungsstrategien, Einschränkungen und AWS Dienste beschrieben, die Sie verwenden können. Zu Ihren Optionen gehören das Rehosten (Lift and Shift) Ihrer .NET-Anwendung in der Cloud sowie die Containerisierung, die Zerlegung in Microservices und die Einführung einer serverlosen Architektur.

Entscheidungsmatrix

In der folgenden Tabelle sind die Migrations- und Modernisierungsoptionen für ältere .NET-Anwendungen zusammengefasst, die auf Ihrem Anwendungsfall und Ihren Ressourcen basieren.

Anwendungsfall	Migrationsstrategie und Architektur				
	Rehosten	Als Windows-Container neu aufstellen	Re-architect als Linux-Container	Re-architect als Microservices in Linux-Containern	Re-architect als Microservices ohne Container
Sie haben Ressourcen für das Refactoring.	⊗Nein	⊗Nein	⊘Ja	⊘Ja	⊘Ja
Ihre .NET-Legacy-Anwendung wird ständig verwendet.	⊘Ja	⊘Ja	⊘Ja	⊘Ja	⊗Nein
Sie können .NET Framework-Abhängigkeiten auflösen.	⊗Nein	⊗Nein	⊘Ja	⊘Ja	⊘Ja
Sie können Windows-Abhängigkeiten entfernen.	⊗Nein	⊗Nein	⊘Ja	⊘Ja	⊘Ja
Sie möchten Ihre Anwendung	⊘Ja	⊗Nein	⊗Nein	⊗Nein	⊗Nein

als native Windows-Anwendung auf einer Amazon Elastic Compute Cloud (Amazon EC2) - Instance ausführen.					
Ihr Code kann von .NET Framework auf .NET Core oder .NET 6 portiert werden.	⊗Nein	⊗Nein	⊘Ja	⊘Ja	⊘Ja
Sie möchten Ihre monolithische Anwendung aufteilen.	⊗Nein	⊗Nein	⊗Nein	⊘Ja	⊘Ja

In den folgenden Abschnitten werden diese Optionen detailliert beschrieben:

- Rehosting
- Umplatformierung als Windows-Container
- Umplatforming als Linux-Container
- Re-architecting als Microservices in Linux-Containern
- Re-architecting als Microservices ohne Container

Hostwechsel

Beim Rehosting (Lift and Shift) wird Ihre lokale Anwendung in die Cloud migriert, ohne sie zu ändern. Diese Strategie wird hauptsächlich verwendet, um umfangreiche Anwendungen zu migrieren, um bestimmte Geschäftsziele zu erreichen, wie z. B. die Einführung eines Produkts innerhalb eines kürzeren Zeitrahmens oder das Verlassen eines lokalen Rechenzentrums. Die Anwendungen werden auf Windows-Instances von Amazon Elastic Compute Cloud (Amazon EC2) neu gehostet, die die Anforderungen der Anwendungen erfüllen, die Sie migrieren.

Anwendungsfälle

Diese Migrationsstrategie ist in jedem der folgenden Szenarien nützlich:

- Die ältere .NET-Anwendung muss als native Windows-Anwendung ausgeführt werden.
- Zeit und Ressourcen für die Modernisierung der Anwendung stehen nicht zur Verfügung.
- Bei der älteren .NET-Anwendung handelt es sich um eine kommerzielle Standardanwendung (COTS).

Vorteile

Das Rehosting bietet im Vergleich zu lokalen .NET-Anwendungen die folgenden Vorteile:

- Minimaler Aufwand, da keine Code- oder Architekturänderungen erforderlich sind
- Reduzierte Kosten
- Bessere Einhaltung von Vorschriften und Sicherheit, da bewährte Methoden für AWS Infrastruktur und Sicherheit verwendet werden

Nachteile

- Nutzt die Leistungs-, Skalierbarkeits- und Ausfallsicherheitsoptionen der AWS Cloud nicht in vollem Umfang
- Es ist schwierig, sie in hochmoderne Cloud-Dienste zu integrieren

AWS service

- Amazon EC2
- AWS Elastic Beanstalk
- AWS Transform MGN

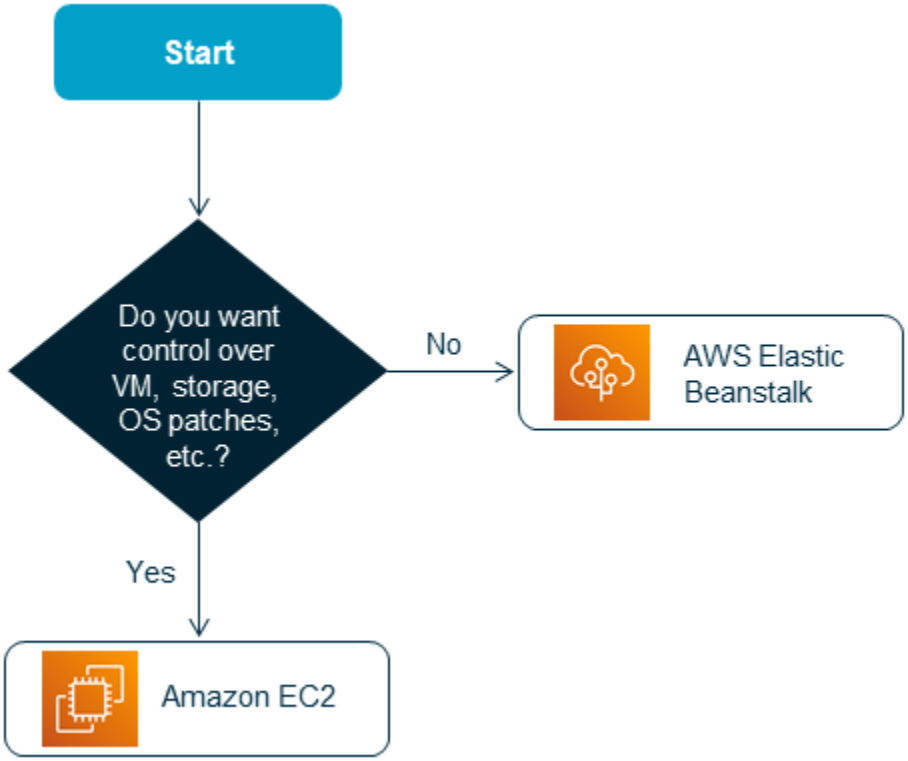
Tools

Werkzeug	Zweck	Ressource
Assistent für die Migration von Windows-Webanwendungen	Dieses Tool ist ein interaktives PowerShell Skript, das ganze Websites und ihre Konfigura tionen zu Elastic Beanstalk migriert.	ASP.NET Anwendungen zu Elastic Beanstalk migrieren (Blogbeitrag)AWS

Entscheidungen zur Bereitstellung

Sie können aus zwei Bereitstellungsoptionen wählen:

- Wenn Sie die vollständige Kontrolle über die Konfiguration Ihrer Rechenumgebung, einschließlich Speicher- und Speichereinstellungen, und Kontrolle über Betriebssystem-Patches wünschen, migrieren Sie Ihre .NET-Anwendung zu Amazon EC2.
- Wenn Sie nicht die volle Kontrolle über die Infrastruktur benötigen, verwenden Sie Elastic Beanstalk. Elastic Beanstalk richtet automatisch eine verwaltete Umgebung für Ihre Anwendung ein.


Start
Do you want control over VM, storage, OS patches, etc.?
No
AWS Elastic Beanstalk
Yes
Amazon EC2

Umplatforming als Windows-Container

Wenn Sie Ihre .NET-Anwendung als Windows-Container auf eine Plattform umstellen, können Sie Ihre Geschäftsziele mit weniger Aufwand erreichen als mit einem Refactoring. So können Sie die Vorteile von Container-Technologien nutzen, ohne die Kernarchitektur Ihrer .NET-Anwendung zu ändern. Windows-Anwendungen können ohne großen Aufwand in Container konvertiert werden.

Framework-based .NET-Container unterstützen Windows Server 2016 oder 2019 als Host-Betriebssystem.

Anwendungsfälle

Diese Migrationsstrategie ist in jedem der folgenden Szenarien nützlich:

- Sie können .NET Framework-Abhängigkeiten nicht auflösen.
- Sie können Windows-Abhängigkeiten nicht auflösen.
- Sie verfügen nicht über die Ressourcen, um die Anwendung auf .NET Core oder .NET 6 umzustellen.

Vorteile

Dieser Migrationsansatz bietet im Vergleich zu lokalen .NET-Anwendungen die folgenden Vorteile:

- Minimaler Aufwand
- Verbesserte Ressourcennutzung
- Verbesserte Sicherheit
- Bessere Bereitstellungsoptionen

Nachteile

- Lizenzkosten für das Windows-Host-Betriebssystem

AWS service

Zum Speichern von Container-Images:

- Amazon Elastic Container Registry (Amazon ECR)

Für die Orchestrierung von Windows-Containern:

- Amazon Elastic Container Service (Amazon ECS)
- Amazon Elastic Kubernetes Service (Amazon EKS)
- Amazon EC2 hostet Docker mit Windows-Containern

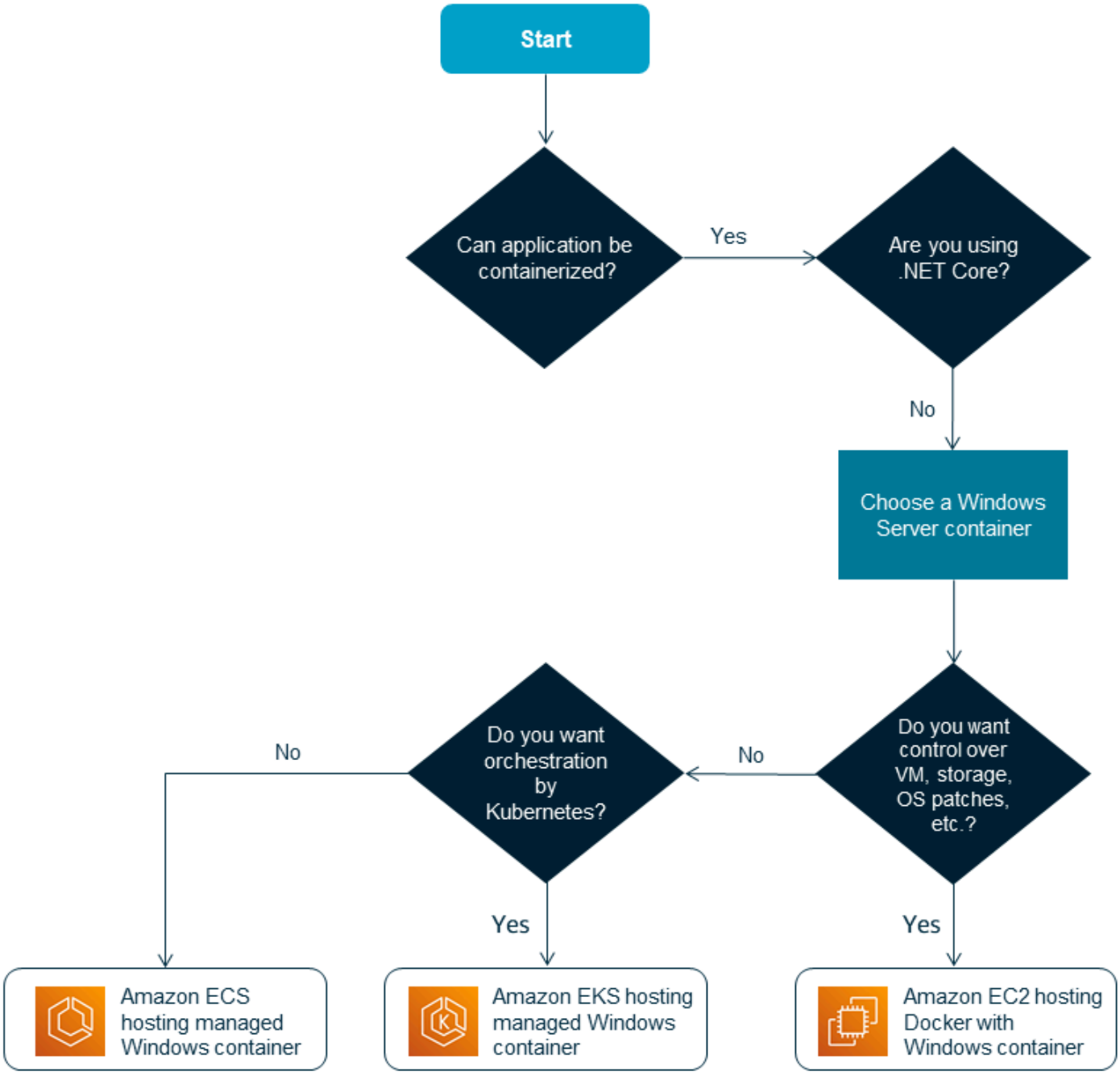
Tools

Werkzeug	Zweck	Ressource
AWS App2Container (A2C)	A2C ist ein Befehlszeilentool zur Modernisierung von.NET- und Java-Anwendungen, indem diese mit minimalem Aufwand in containerisierte Anwendungen umgewandelt werden.	• Details • Dokumentation

Entscheidungen zur Bereitstellung

Sie können aus drei Bereitstellungsoptionen wählen:

- Wenn Sie die vollständige Kontrolle über die Konfiguration Ihrer Rechenumgebung, einschließlich Speicher- und Speichereinstellungen, und Kontrolle über Betriebssystem-Patches wünschen, stellen Sie Ihre Anwendung als Windows-Container auf einer EC2-Instance bereit.
- Wenn Sie möchten, dass der Container von Kubernetes verwaltet wird: Stellen Sie Ihre Anwendung als Windows-Container auf Amazon EKS bereit.
- Wenn Sie möchten, dass der Container von Amazon ECS verwaltet wird: Stellen Sie Ihre Anwendung als Windows-Container auf Amazon ECS bereit.


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 als Linux-Container

Durch die Portierung Ihrer .NET Framework-Anwendungen auf .NET Core oder .NET 6 können Sie Ihre Anwendungen auf mehreren Plattformen ausführen, Ihre Lizenzkosten senken, die Leistung steigern und die Skalierbarkeit verbessern.

Anwendungsfälle

Diese Migrationsstrategie ist in jedem der folgenden Szenarien nützlich:

- Sie haben die Ressourcen und die Zeit zur Verfügung, um Ihre Anwendung umzugestalten.
- Sie sind in der Lage, alle .NET Framework-Abhängigkeiten aufzulösen.
- Sie haben eine lang laufende Anwendung.

Vorteile

Dieser Migrationsansatz bietet im Vergleich zu lokalen .NET-Anwendungen die folgenden Vorteile:

- Niedrigere Gesamtbetriebskosten (TCO)
- Verbesserte Sicherheit und Leistung
- Beschleunigte Innovation
- Vorteile der Umstellung auf Cloud-native Anwendungen
- Open-source

Nachteile

- Aufwand und Kosten des Refactorings

AWS service

Zum Speichern von Container-Images:

- Amazon ECR

Für die Orchestrierung von Containern:

- Amazon ECS oder Amazon ECS mit AWS Fargate
- Amazon EKS oder Amazon EKS mit Fargate

AWS Fargate ist eine serverlose, nutzungsabhängige Rechen-Engine, mit der Sie sich auf die Erstellung von Anwendungen konzentrieren können, ohne Server verwalten zu müssen. Fargate ist sowohl mit Amazon ECS als auch mit Amazon EKS kompatibel.

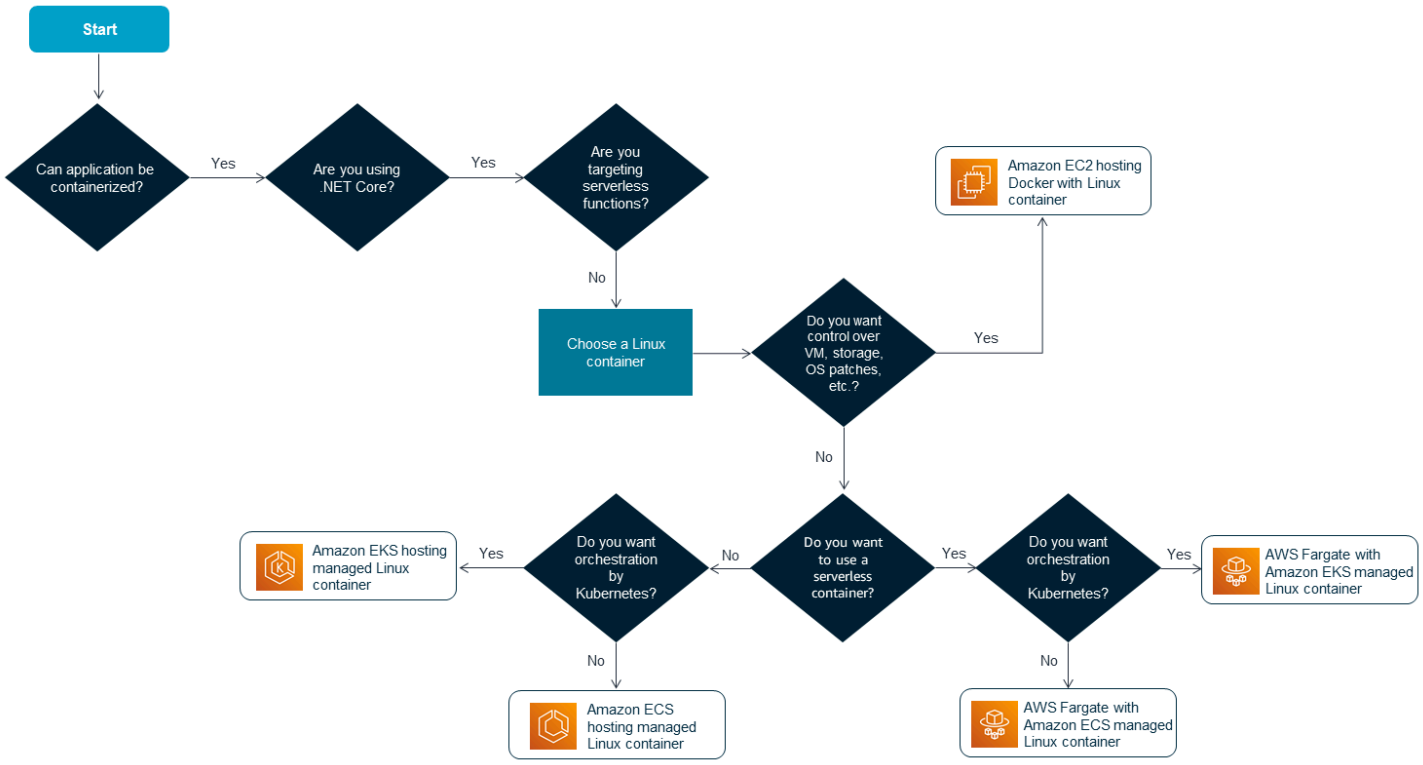
Tools

Werkzeug	Zweck	Ressource
Porting Assistant für .NET	Dieses Analysetool scannt .NET Framework-Anwendungen und generiert eine Bewertung der Kompatibilität mit .NET Core. Die Bewertung hilft Ihnen, Ihre Anwendungen schneller auf Linux zu portieren.	• Details • Dokumentation
AWS App2Container (A2C)	A2C ist ein Befehlszeilentool zur Modernisierung von.NET- und Java-Anwendungen, indem diese mit minimalem Aufwand in containerisierte Anwendungen umgewandelt werden.	• Details • Dokumentation

Entscheidungen zur Bereitstellung

Sie können aus fünf Bereitstellungsoptionen wählen:

- Wenn Sie die vollständige Kontrolle über die Konfiguration Ihrer Rechenumgebung, einschließlich Speicher- und Speichereinstellungen, und Kontrolle über Betriebssystem-Patches wünschen, stellen Sie Ihre Anwendung als Linux-Container auf einer EC2-Instance bereit.
- Wenn Sie möchten, dass der Container von Kubernetes verwaltet und als serverloser Container ausgeführt wird, stellen Sie Ihre Anwendung mit Fargate als Linux-Container auf Amazon EKS bereit.
- Wenn Sie möchten, dass der Container von Amazon ECS verwaltet und als serverloser Container ausgeführt wird, stellen Sie Ihre Anwendung mit Fargate als Linux-Container auf Amazon ECS bereit.
- Wenn Sie möchten, dass der Container von Kubernetes verwaltet wird, Sie aber die Rechenressourcen des Containers selbst verwalten möchten: Stellen Sie Ihre Anwendung als Linux-Container auf Amazon EKS bereit.
- Wenn Sie möchten, dass der Container von Amazon ECS verwaltet wird, Sie aber die Rechenressourcen des Containers selbst verwalten möchten, stellen Sie Ihre Anwendung als Linux-Container auf Amazon ECS bereit.



Re-architecting als Microservices in Linux-Containern

Eine Microservices-Architektur ist ein Ansatz zur Entwicklung einer einzelnen Anwendung als Suite kleiner Dienste. Jeder Dienst läuft in seinem eigenen Prozess und kommuniziert mit anderen Diensten über einfache Mechanismen. Dieser Ansatz unterteilt eine monolithische Anwendung in kleinere Dienste, wobei jeder Dienst einem einzigen Zweck dient und als Container bereitgestellt wird.

Anwendungsfälle

Diese Migrationsstrategie ist nützlich, wenn:

- Sie möchten Ihr monolithisches System in Microservices aufteilen.
- Sie haben die Ressourcen und die Zeit für das Refactoring zur Verfügung.
- Sie können alle .NET Framework-Abhängigkeiten auflösen.
- Sie haben eine lang laufende Anwendung.

Vorteile

Dieser Migrationsansatz bietet im Vergleich zu lokalen .NET-Anwendungen die folgenden Vorteile:

- Schnellere Innovation, da es einfacher ist, einer Microservices-Architektur neue Funktionen hinzuzufügen
- Hohe Verfügbarkeit und Zuverlässigkeit
- Höhere Agilität und Skalierbarkeit auf Abruf
- Unabhängige Bereitstellung und moderne Pipelines für kontinuierliche Integration und kontinuierliche Bereitstellung (CI/CD)
- Starke Modulgrenzen und technische Vielfalt

Nachteile

- Aufwand und Kosten des Refactorings
- Potenzielle betriebliche Komplexität

AWS service

Sie können die folgenden AWS Dienste verwenden, um ein auf Microservices basierendes System zu entwickeln:

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

Tools

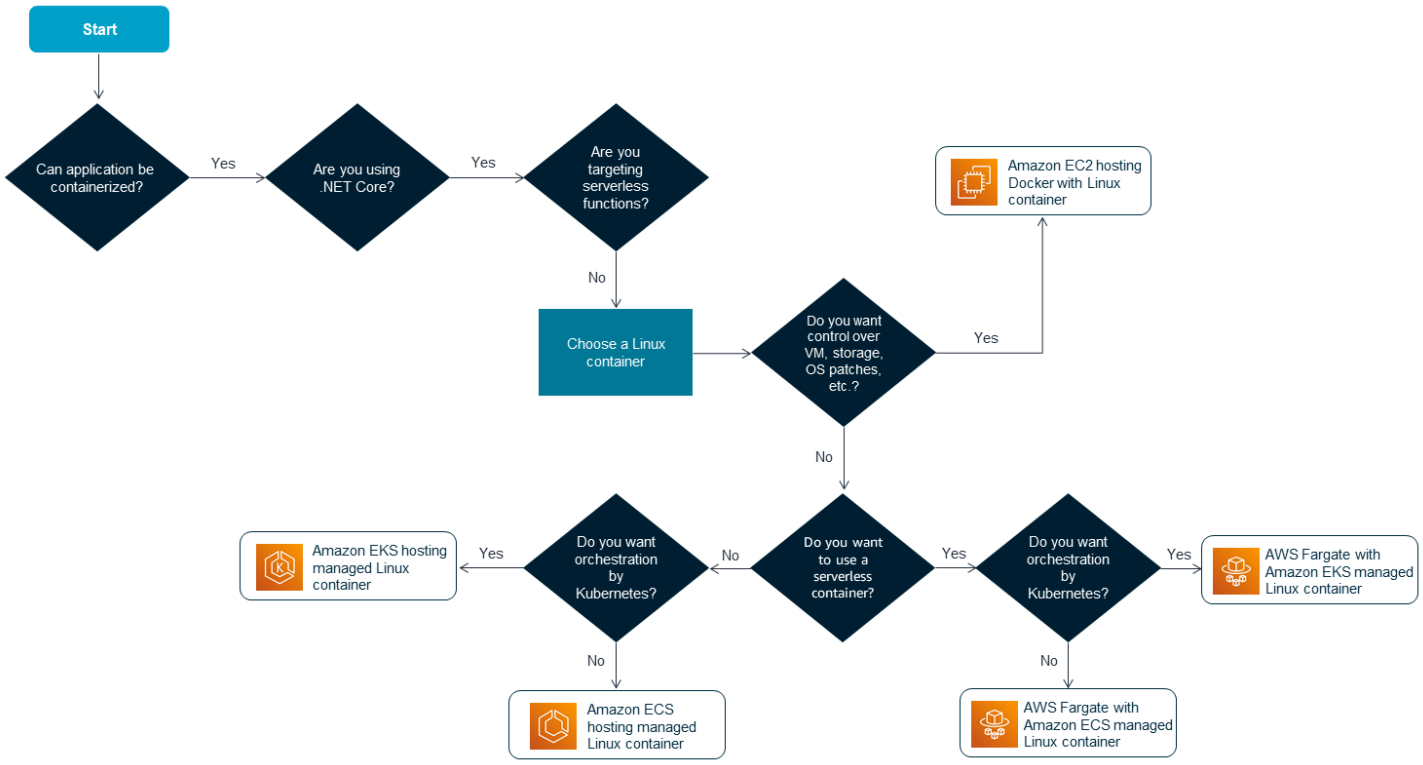
AWS Professional Services bietet maßgeschneiderte Tools und Services, mit denen Sie Ihre monolithischen Anwendungen in Microservices umwandeln können.

Entscheidungen zur Bereitstellung

Sie können aus fünf Bereitstellungsoptionen wählen:

- Wenn Sie die vollständige Kontrolle über die Konfiguration Ihrer Rechenumgebung, einschließlich Speicher- und Speichereinstellungen, und Kontrolle über Betriebssystem-Patches wünschen, stellen Sie Ihre Anwendung als Linux-Container auf einer EC2-Instance bereit.
- Wenn Sie möchten, dass der Container von Kubernetes verwaltet und als serverloser Container ausgeführt wird, stellen Sie Ihre Anwendung mit Fargate als Linux-Container auf Amazon EKS bereit.
- Wenn Sie möchten, dass der Container von Amazon ECS verwaltet und als serverloser Container ausgeführt wird, stellen Sie Ihre Anwendung mit Fargate als Linux-Container auf Amazon ECS bereit.

- Wenn Sie möchten, dass der Container von Kubernetes verwaltet wird, Sie aber die Rechenressourcen des Containers selbst verwalten möchten: Stellen Sie Ihre Anwendung als Linux-Container auf Amazon EKS bereit.
- Wenn Sie möchten, dass der Container von Amazon ECS verwaltet wird, Sie aber die Rechenressourcen des Containers selbst verwalten möchten, stellen Sie Ihre Anwendung als Linux-Container auf Amazon ECS bereit.



Re-architecting als Microservices ohne Container

AWS Lambda ist ein serverloser Rechendienst, mit dem Sie Code ausführen können, ohne Server bereitzustellen oder zu verwalten, eine auslastungsabhängige Cluster-Skalierungslogik zu erstellen, Eventintegrationen zu verwalten oder Laufzeiten zu verwalten. Lambda führt Ihre Funktion nur bei Bedarf aus und skaliert automatisch – von einigen Anforderungen pro Tag bis zu Tausenden pro Sekunde. Sie zahlen nur für die Rechenzeit, die Sie tatsächlich verbrauchen — es fallen keine Gebühren an, wenn Ihr Code nicht ausgeführt wird. Bei diesem Ansatz wird eine monolithische Anwendung in kleinere Dienste unterteilt, wobei jeder Dienst einem einzigen Zweck dient. Wenn der Dienst nicht ständig läuft, kann er als Lambda-Funktion implementiert werden. Andernfalls sollte der Dienst in einem Container ausgeführt werden.

Anwendungsfälle

Sie können diese Migrationsstrategie in den folgenden Szenarien verwenden:

- Sie möchten Ihr monolithisches System in Microservices aufteilen.
- Sie haben die Ressourcen und die Zeit für das Refactoring zur Verfügung.
- Sie können alle .NET Framework-Abhängigkeiten auflösen.
- Ihre Anwendungen werden nicht ständig ausgeführt, sondern nur für einen sehr kurzen Zeitraum.

Vorteile

Dieser Migrationsansatz bietet im Vergleich zu lokalen .NET-Anwendungen die folgenden Vorteile:

- Schnellere Innovation, da es einfacher ist, einer Microservices-Architektur neue Funktionen hinzuzufügen
- Hohe Verfügbarkeit und Zuverlässigkeit
- Höhere Agilität und Skalierbarkeit auf Abruf
- Unabhängige Bereitstellung und moderne CI/CD Pipelines
- Starke Modulgrenzen und technische Vielfalt
- Kosteneinsparungen
- Geringerer Aufwand für die Bereitstellung der Infrastruktur

Nachteile

- Aufwand und Kosten des Refactorings
- Mögliche betriebliche Komplexität
- Keine Unterstützung für lang laufende Anwendungen

AWS service

Dies sind einige der wichtigen AWS Dienste, die Sie verwenden können, um eine Microservices-Architektur mit AWS Lambda folgenden Komponenten zu entwickeln:

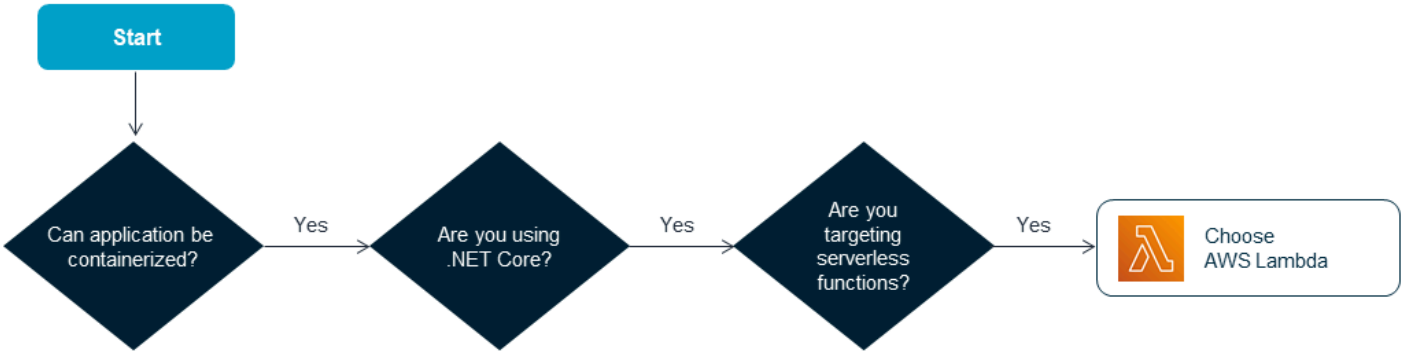
- Amazon API Gateway
- Amazon-Simple-Notification-Service (Amazon-SNS)
- Amazon-Simple-Queue-Service (Amazon SQS)
- AWS Lambda
- CloudFormation oder AWS CDK
- IAM
- Amazon S3

Tools

AWS Professional Services bietet maßgeschneiderte Tools und Services, mit denen Sie Ihre monolithischen Anwendungen in Microservices umwandeln können.

Entscheidungen zur Bereitstellung

Dieser Migrations- und Modernisierungsansatz wird unterstützt von AWS Lambda


Start
Can application be containerized?
Yes
Are you using .NET Core?
Yes
Are you targeting serverless functions?
Yes
Choose AWS Lambda

Nächste Schritte und Ressourcen

Nachdem Sie eine Migrations- und Modernisierungsstrategie für Ihre .NET-Anwendung ausgewählt haben, nutzen Sie die folgenden Ressourcen, um Ihr Unternehmen und Ihre Anwendungen auf ihre Eignung hin zu bewerten und mit dem Migrationsprozess zu beginnen:

- Strategie zur Modernisierung von Anwendungen in der Cloud AWS
- Bewertung der Migrationsbereitschaft
- Bewertung der Modernisierungsbereitschaft von Anwendungen in der AWS Cloud
- AWS Migration Acceleration Program (MAP)

Tools für die Migration

- Assistent für die Migration von Windows-Webanwendungen
- AWS App2Container (A2C)
- Portierungsassistent für .NET

Dokumentverlauf

In der folgenden Tabelle werden wichtige Änderungen in diesem Leitfaden beschrieben. Um Benachrichtigungen über zukünftige Aktualisierungen zu erhalten, können Sie einen RSS-Feed abonnieren.

Änderung	Beschreibung	Datum
Dienste neu hosten	Zum AWS Transform MGN Bereich Rehosting hinzugefügt.	7. August 2023
.NET	.NET-Verweise wurden aktualisiert, sodass sie die neueste Version wiedergeben.	9. Dezember 2021
Erste Veröffentlichung	—	3. November 2021

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.