AWS Leitfaden zur Entscheidungsfindung

AWS Fargate oder AWS Lambda?

AWS Fargate oder AWS Lambda?: AWS Leitfaden zur Entscheidungsfindung

Table of Contents

AWS Fargate oder AWS Lambda?

Verstehen Sie die Unterschiede und wählen Sie den für Sie richtigen aus

Zweck	Um herauszufinden, ob AWS Fargate oder ob Ihre AWS Lambda Anforderungen an einen serverlosen Rechendienst erfüllt werden.
Letzte Aktualisierung	15. November 2024
Abgedeckte Dienste	• AWS Fargate • AWS Lambda

Einführung

Bevor Sie damit beginnen, herauszufinden, ob Sie sich für AWS Lambda oder AWS Fargate als Ihren serverlosen Rechendienst entscheiden, haben Sie wahrscheinlich das breitere Spektrum an AWS Rechendiensten (die im Entscheidungsleitfaden zur Auswahl eines AWS Rechendienstes behandelt werden) in Betracht gezogen und es auf diese beiden Optionen eingegrenzt, da sie Folgendes bieten:

- Geringerer Betriebsaufwand: Sowohl Lambda als auch Fargate nehmen das Servermanagement weg und reduzieren so den Bedarf an Patches, Wartung und Kapazitätsplanung.
- Pay-per-use Preisgestaltung: Sie zahlen nur für die Rechenressourcen, die Sie tatsächlich nutzen, wodurch die Kosten für variable Workloads potenziell gesenkt werden.
- Schnellere Bereitstellung: Bietet im Vergleich zur Bereitstellung und Konfiguration EC2 von Instanzen in der Regel kürzere Bereitstellungszeiten.
- Integrierte Hochverfügbarkeit: Beide Dienste kümmern sich automatisch um die Redundanz der Infrastruktur.
- Vereinfachte Einhaltung von Vorschriften: Eine geringere Angriffsfläche und integrierte Sicherheitsfunktionen können die Einhaltung von Vorschriften erleichtern.
- Konzentrieren Sie sich auf den Code: Entwickler können sich mehr auf das Schreiben von Anwendungscode als auf die Verwaltung der Infrastruktur konzentrieren.

Lambda und Fargate sind zwar beide serverlose Optionen, es gibt jedoch erhebliche Unterschiede zwischen ihnen:

AWS Fargateist eine serverlose Rechen-Engine für Container, die hauptsächlich mit Amazon ECS verwendet wird. Sie verwaltet Ihre Infrastruktur automatisch, sodass Sie sich auf die Bereitstellung und Skalierung containerisierter Anwendungen konzentrieren können. Fargate ist ideal für Anwendungen mit langer Laufzeit, Microservices oder Batch-Verarbeitung, bei denen Sie eine genaue Kontrolle über die Ressourcenzuweisung (CPU, Speicher) benötigen und die Verwaltung der zugrunde liegenden Server vermeiden möchten.

AWS Lambdaist ein serverloser Computerdienst, der Ihren Code als Reaktion auf Ereignisse automatisch ausführt und die zugrunde liegenden Rechenressourcen verwaltet. Es eignet sich am besten für ereignisgesteuerte Anwendungen, z. B. die Verarbeitung von auf Amazon S3 hochgeladenen Dateien, die Beantwortung von HTTP-Anfragen oder die Ausführung von geplanten Aufgaben. Lambda eignet sich auch gut für Stream-Verarbeitungs- und Datenverarbeitungsanwendungen, da es automatisch als Reaktion auf Ereignisse skaliert und große Datenmengen in Echtzeit verarbeiten kann. Lambda kann Datenströme aus Quellen wie Amazon Kinesis oder Amazon DynamoDB verarbeiten und ermöglicht so effiziente, serverlose Datentransformationen, Filterung und Analysen, ohne die Infrastruktur verwalten zu müssen. Lambda ist für kurzlebige Aufgaben (bis zu 15 Minuten) konzipiert und wird auf der Grundlage der Anzahl der Anfragen und der Ausführungszeit abgerechnet, sodass es für sporadische Workloads kostengünstig ist.

Wenn Ihr Projekt ereignisgesteuerte, kurzzeitige Aufgaben oder unvorhersehbare Arbeitsbelastungen beinhaltet, ist dies möglicherweise die bessere Lösung. AWS Lambda Wenn Sie containerisierte Anwendungen mit bestimmten Ressourcenanforderungen ausführen müssen (oder wenn Sie persistente Prozesse benötigen), wäre dies besser geeignet. AWS Fargate

Die folgende Tabelle gibt einen detaillierteren Überblick über einige der Unterschiede zwischen diesen Diensten, um Ihnen den Einstieg zu erleichtern.

Merkmal	AWS Fargate	AWS Lambda
Ausführungsmodell	Containerbasiertes, serverlos es Computing	Ereignisgesteuerte, serverlose Funktionen
Unterstützte Sprachen	Jede Sprache, die in einem Container ausgeführt werden kann	Unterstützte Sprachen: Node.js, Python, Java, C#, Go, Ruby und PowerShell. Sie

		können auch eine benutzerdefinierte Runtime erstellen, um eine AWS Lambda Funktion in der Sprache Ihrer Wahl zu implementieren.
Anwendungsfall	Container-Anwendungen mit langer Laufzeit	Kurzfristige, ereignisgesteuerte Aufgaben
Skalierung	Automatische Skalierung auf der Grundlage der gewünschten Anzahl von Aufgaben	Automatische Skalierung pro Anfrage
Kaltstart	35 Sekunden bis 2 Minuten	100 ms bis 2 Sekunden
Zeitlimit für die Ausführung	Kein festes Limit	Maximal 15 Minuten
Speicherzuweisung	Bis zu 120 GiB	Bis zu 10 GiB
CPU-Zuweisung	Bis zu 16 vCPU	Proportional zum Arbeitsspeicher, bis zu 6 vCPU
Netzwerk	Läuft in VPC, kann verwenden ENIs	Kann in einer AWS verwalteten VPC ausgeführt oder mit Hyperplane an eine vom Kunden verwaltete VPC angeschlossen werden AWS

Statusverwaltung	Container in Fargate können den Status über Anfragen hinweg beibehalten, solange der Container läuft, wodurch es möglich ist, Sitzungen zu verarbeiten, Daten zwischenzuspeichern oder den In-Memory-Status beizubehalten, ohne externen Speicher zu benötigen. Externer Speicher wird für kritische Daten empfohlen.	Von Haus aus zustandslos (der Status muss extern verwaltet werden, z. B. Amazon S3, Amazon DynamoDB, Amazon EFS)
Container-Support	Unterstützt Container	Eingeschränkte Container-Unterstützung (über Container-Image-Bereitstellungen)
Orchestrierung	Integriert in Amazon ECS	Keine Orchestrierung erforderlich
Preismodell	Abrechnung pro Sekunde für genutzte vCPU und Arbeitsspeicher	Pro Aufruf und Dauer (GB-Sekunden)
Grenzwerte für Parallelität	Basierend auf der Clusterkapazität	Standardmäßig 1000 gleichzeitige Ausführungen (kann erhöht werden)
ereignisgesteuertes Aufrufen	Erfordert zusätzliche Einrichtung	Native Unterstützung für verschiedene AWS Ereignisquellen
Minderung des Kaltstarts	Das verzögerte Laden von Bildern mit Seekable OCI kann das Starten von Fargate-Aufgaben beschleunigen	Bereitgestellte Parallelität verfügbar

Größenbeschränkung für Package	Keine spezifische Beschränkung (Containergröße begrenzt durch konfigurierten kurzlebigen Speicher, maximal 200 GiB)	250 MB entpackt, einschließlich Ebenen, 10 GB für Container-Image-Bereitstellungen

Unterschiede zwischen Fargate und Lambda

Erkunden Sie die Unterschiede zwischen Fargate und Lambda in einer Reihe von Schlüsselbereichen.

Languages supported

Fargate: AWS Fargate ist ein Container-Orchestrierungsdienst, was bedeutet, dass er jede Programmiersprache oder Laufzeitumgebung unterstützt, die in einen Docker-Container gepackt werden kann. Diese Flexibilität ermöglicht es Entwicklern, praktisch jede Sprache, jedes Framework oder jede Bibliothek zu verwenden, die ihren Anwendungsanforderungen entspricht. Egal, ob Sie Python, Java, Node.js, Go, .NET, Ruby, PHP oder sogar benutzerdefinierte Sprachen und Umgebungen verwenden, Fargate kann sie ausführen, solange sie in einem Container gekapselt sind. Diese breite Sprachunterstützung macht Fargate ideal für die Ausführung verschiedener Anwendungen, einschließlich älterer Systeme, mehrsprachiger Microservices und moderner Cloud-nativer Anwendungen.

Lambda: AWS Lambda bietet im Vergleich zu Fargate native Unterstützung für eine begrenzte Anzahl von Sprachen und wurde speziell für ereignisgesteuerte Funktionen entwickelt. Ab sofort unterstützt Lambda offiziell die folgenden Sprachen:

- Node.js
- Python
- Java
- Go
- Ruby
- C#
- PowerShell

Lambda unterstützt auch benutzerdefinierte Laufzeiten, sodass Sie Ihre eigene Sprache oder Laufzeitumgebung verwenden können. Dies erfordert jedoch mehr Einrichtung und Verwaltung als die Verwendung der nativ unterstützten Optionen. Wenn Sie Ihre Lambda-Funktion von einem Container-Image aus bereitstellen möchten, können Sie Ihre Funktion in Rust schreiben, indem Sie ein AWS reines Betriebssystem-Basisimage verwenden und den Rust-Runtime-Client in Ihr Image aufnehmen. Wenn Sie eine Sprache verwenden, die keinen AWS bereitgestellten Runtime-Interface-Client hat, müssen Sie Ihren eigenen erstellen.

Event-driven invocation

Lambda ist von Natur aus für ereignisgesteuertes Computing konzipiert. Lambda-Funktionen werden als Reaktion auf eine Vielzahl von Ereignissen ausgelöst, darunter Datenänderungen, Benutzeraktionen oder geplante Aufgaben. Es lässt sich nahtlos in viele AWS-Services Anwendungen integrieren, z. B. in Amazon S3 (zum Beispiel beim Aufrufen einer Funktion, wenn eine Datei hochgeladen wird), DynamoDB (zum Beispiel beim Auslösen von Datenaktualisierungen) und API Gateway (z. B. Verarbeitung von HTTP-Anfragen). Die ereignisgesteuerte Lambda-Architektur ist ideal für Anwendungen, die sofort auf Ereignisse reagieren müssen, ohne persistente Rechenressourcen zu benötigen.

Fargate ist nicht nativ ereignisgesteuert, kann aber mit zusätzlicher Standardlogik in Ereignisquellen wie Amazon SQS und Kinesis integriert werden. Während Lambda den Großteil dieser Integrationslogik für Sie übernimmt, müssen Sie diese Integration mithilfe von APIs for these Services selbst implementieren.

Runtime/use cases

Fargate wurde für die Ausführung von containerisierten Anwendungen entwickelt und bietet eine flexible Laufzeitumgebung, in der Sie die CPU-, Speicher- und Netzwerkeinstellungen für Ihre Container definieren können. Da Fargate auf einem containerbasierten Modell arbeitet, unterstützt es lang andauernde Prozesse, persistente Dienste und Anwendungen mit spezifischen Laufzeitanforderungen. Die Container in Fargate können unbegrenzt ausgeführt werden, da es keine feste Begrenzung der Ausführungszeit gibt, was sie ideal für Anwendungen macht, die kontinuierlich laufen müssen.

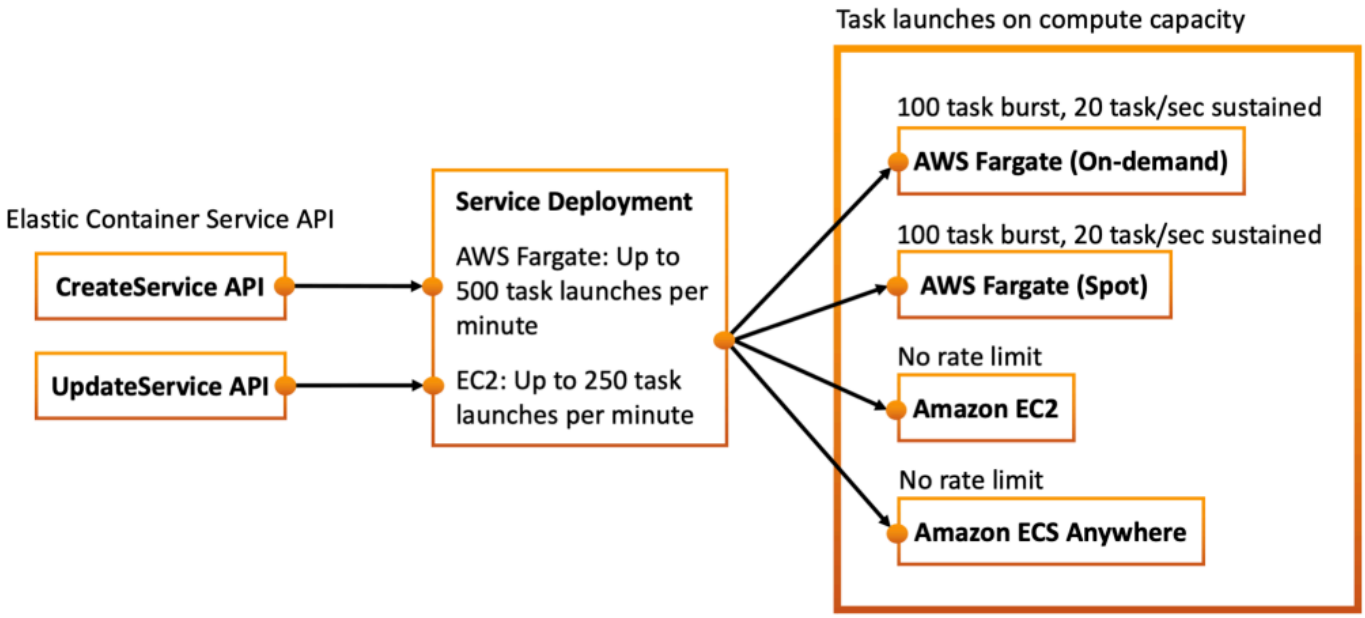
Lambda hingegen ist für kurzlebige, ereignisgesteuerte Aufgaben optimiert. Lambda-Funktionen werden in einer zustandslosen Umgebung ausgeführt, in der die maximale Ausführungszeit auf 15 Minuten begrenzt ist. Dadurch eignet sich Lambda gut für Szenarien wie Dateiverarbeitung, Echtzeit-Datenstreaming und HTTP-Anforderungsverarbeitung, bei denen die Aufgaben kurz sind und keine lang andauernden Prozesse erfordern.

In Lambda ist die Laufzeitumgebung stärker abstrahiert und es gibt weniger Kontrolle über die zugrunde liegende Infrastruktur. Lambda ist zustandslos, was bedeutet, dass jeder Funktionsaufruf unabhängig ist und dass alle Zustände oder Daten, die zwischen den Aufrufen bestehen müssen, extern verwaltet werden müssen (z. B. in Datenbanken oder Speicherdiensten).

Scaling

Fargate skaliert, indem es die Anzahl der laufenden Container an den gewünschten Status anpasst, der in Ihrem Container Orchestration Service (Amazon ECS) definiert ist. Diese Skalierung kann manuell oder automatisch über Amazon EC2 Auto Scaling erfolgen. In diesem Blogbeitrag finden Sie weitere Informationen dazu.

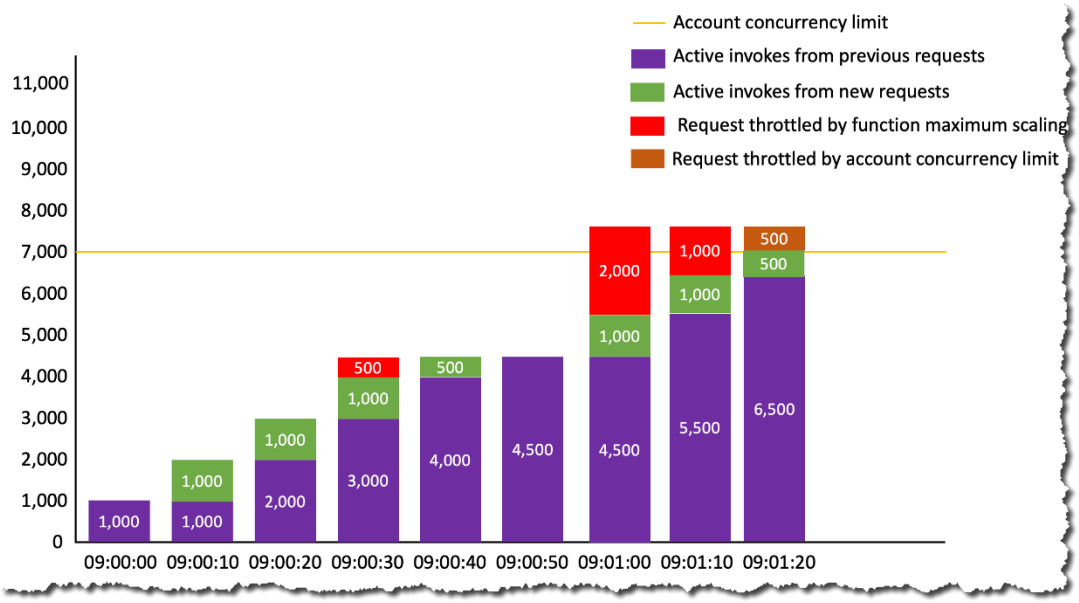
In Fargate läuft jeder Container in seiner isolierten Umgebung, und bei der Skalierung werden je nach Auslastung zusätzliche Container gestartet oder gestoppt. Der Amazon ECS Service Scheduler ist in der Lage, bis zu 500 Aufgaben in weniger als einer Minute pro Service für Web- und andere Dienste mit langer Laufzeit zu starten.



Für Lambda ist Parallelität die Anzahl der laufenden Anfragen, die Ihre AWS Lambda Funktion gleichzeitig bearbeitet. Dies unterscheidet sich von der Parallelität in Fargate, wo jede Fargate-Aufgabe gleichzeitige Anfragen verarbeiten kann, solange Rechen- und Netzwerkressourcen verfügbar sind. Für jede gleichzeitige Anfrage stellt Lambda eine separate Instance Ihrer Ausführungsumgebung bereit. Wenn Ihre Funktionen mehr Anfragen erhalten, sorgt Lambda automatisch für die Skalierung der Anzahl der Ausführungsumgebungen, bis Sie das Gleichzeitigkeitslimit für Ihr Konto erreichen. Standardmäßig bietet Lambda Ihrem Konto ein

Gesamt-Parallelitätslimit von 1.000 gleichzeitigen Ausführungen für alle Funktionen in einem AWS-Region, und Sie können bei Bedarf eine Erhöhung des Kontingents beantragen.

Für jede Lambda-Funktion in einer Region beträgt die Skalierungsrate für Parallelität 1.000 Ausführungsinstanzen alle 10 Sekunden, bis zur maximalen Kontoparallelität. Wie in diesem Blog erklärt, werden die zusätzlichen Anfragen gedrosselt, wenn die Anzahl der Anfragen in einem Zeitraum von 10 Sekunden 1.000 übersteigt. Die folgende Grafik zeigt, wie die Lambda-Skalierung unter der Annahme einer Kontoparallelität von 7000 funktioniert.



Cold start and cold-start mitigation

Bei Lambda kann es zu Kaltstarts kommen, die auftreten, wenn eine Funktion aufgerufen wird, nachdem sie einige Zeit inaktiv war. Während eines Kaltstarts muss der Lambda-Dienst eine neue Ausführungsumgebung initialisieren, einschließlich des Ladens der Laufzeit, der Abhängigkeiten und des Funktionscodes. Dieser Prozess kann zu Latenz führen, insbesondere bei Sprachen mit längeren Initialisierungszeiten (z. B. Java oder C#). Kaltstarts können die Leistung von Anwendungen beeinträchtigen, insbesondere von Anwendungen, die Antworten mit niedriger Latenz erfordern.

Um Kaltstarts in Lambda zu verhindern, können verschiedene Strategien eingesetzt werden:

- Minimierung der Funktionsgröße: Durch die Reduzierung der Größe Ihres Funktionspakets und seiner Abhängigkeiten kann die für die Initialisierung benötigte Zeit verringert werden.
- Erhöhen Sie die Speicherzuweisung: Höhere Speicherzuweisungen erhöhen die CPU-Kapazität und reduzieren möglicherweise die Initialisierungszeit.

- Funktionen warm halten: Wenn Sie Ihre Lambda-Funktionen regelmäßig aufrufen (z. B. mithilfe von CloudWatch Events), können Sie sie aktiv halten und die Wahrscheinlichkeit von Kaltstarts verringern.
- Lambda SnapStart: Verwenden Sie Lambda SnapStart für Java-Funktionen, um die Startzeit zu reduzieren.
- Bereitgestellte Parallelität: Diese Funktion hält eine bestimmte Anzahl von Funktionsinstanzen warm und bereit, Anfragen zu bearbeiten, wodurch die Latenz beim Kaltstart reduziert wird. Es erhöht jedoch die Kosten, da Sie für die bereitgestellten Instanzen bezahlen, auch wenn diese Anfragen nicht aktiv bearbeiten.

Fargate ist im Allgemeinen nicht in der gleichen Weise von Kaltstarts betroffen wie Lambda. Die Zeit, die benötigt wird, um eine Fargate-Aufgabe zu starten, steht in direktem Zusammenhang mit der Zeit, die benötigt wird, um die in der Aufgabe definierten Container-Images aus der Image-Registry abzurufen. Fargate unterstützt auch verzögertes Laden von Container-Images, die mit Seekable OCI (SOCI) indexiert wurden. Das verzögerte Laden von Container-Images mit SOCI reduziert die Zeit, die zum Starten von Amazon ECS-Aufgaben auf Fargate benötigt wird. Fargate betreibt Container, die so lange wie nötig aktiv bleiben, was bedeutet, dass sie immer bereit sind, Anfragen zu bearbeiten. Wenn Sie jedoch als Reaktion auf Skalierungsereignisse neue Container starten müssen, kann es bei der Initialisierung der Container zu Verzögerungen kommen, die jedoch im Vergleich zu Lambda-Kaltstarts in der Regel weniger signifikant sind.

Memory and CPU options

Fargate bietet eine detaillierte Kontrolle über Speicher- und CPU-Ressourcen für Ihre containerisierten Anwendungen. Wenn Sie eine Aufgabe in Fargate starten, können Sie die genauen CPU- und Speicheranforderungen auf der Grundlage der Anforderungen Ihrer Anwendung angeben. Die CPU- und Speicherzuweisungen sind unabhängig, sodass Sie Kombinationen auswählen können, die am besten zu Ihrer Arbeitslast passen. Sie können beispielsweise je CPUs nach Konfiguration CPU-Werte zwischen 0,25 V und 16 V CPUs und Arbeitsspeicher zwischen 0,5 GB und 120 GB pro Container auswählen.

Diese Flexibilität ist ideal für die Ausführung von Anwendungen, die bestimmte Leistungsmerkmale erfordern, wie z. B. speicherintensive Datenbanken oder CPU-gebundene Rechenaufgaben. Fargate ermöglicht es Ihnen, Ihre Ressourcenzuweisung zu optimieren, um Kosten und Leistung effektiv in Einklang zu bringen.

In Lambda sind Speicher und CPU verknüpft, wobei die CPU automatisch proportional zur ausgewählten Speichermenge zugewiesen wird. Sie können Speicherzuweisungen zwischen 128

MB und 10 GB in Schritten von 1 MB wählen. Die CPU skaliert mit dem Arbeitsspeicher auf bis zu 6 vCPU, was bedeutet, dass höhere Speichereinstellungen zu mehr CPU-Leistung führen, aber Sie haben keine direkte Kontrolle über die CPU-Zuweisung selbst.

Bei diesem Modell wurde Wert auf Einfachheit gelegt, sodass Entwickler Speichereinstellungen schnell anpassen können, ohne sich um die CPU-Konfigurationen kümmern zu müssen. Es ist jedoch möglicherweise weniger flexibel für Workloads, die ein bestimmtes Gleichgewicht zwischen CPU- und Speicherressourcen erfordern. Das Modell von Lambda eignet sich für Aufgaben, bei denen Sie eine einfache Skalierung auf der Grundlage des Speicherbedarfs wünschen. Für Anwendungen mit komplexen oder hochspezifischen Ressourcenanforderungen ist es jedoch möglicherweise nicht optimal.

Networking

Wenn Sie Aufgaben in Fargate bereitstellen, werden sie in einer Amazon VPC (Amazon Virtual Private Cloud) ausgeführt, sodass Sie die volle Kontrolle über die Netzwerkumgebung haben. Dazu gehört die Konfiguration von Sicherheitsgruppen, Netzwerkzugriffskontrolllisten (ACLs) und Routing-Tabellen. Jede Fargate-Aufgabe erhält ihre eigene Netzwerkschnittstelle mit einer dedizierten privaten IP-Adresse und kann bei Bedarf eine öffentliche IP-Adresse zugewiesen werden.

Fargate unterstützt erweiterte Netzwerkfunktionen wie Load Balancing (mit AWS Elastic Load Balancing), VPC-Peering und direkten Zugriff auf andere Funktionen AWS-Services innerhalb der VPC. Sie können auch sichere, private Verbindungen nutzen, AWS PrivateLink um sie zu unterstützen AWS-Services, ohne das Internet zu durchqueren.

Standardmäßig werden Lambda-Funktionen in einer verwalteten Netzwerkumgebung ohne direkte Kontrolle über Netzwerkschnittstellen oder IP-Adressen ausgeführt. Lambda kann jedoch mithilfe von AWS Hyperplane an eine vom Kunden verwaltete VPC angehängt werden, sodass Sie den Zugriff auf Ressourcen innerhalb Ihrer VPC kontrollieren können.

Wenn Lambda-Funktionen an eine vom Kunden verwaltete VPC angehängt werden, übernehmen sie die Sicherheitsgruppen und Subnetzkonfigurationen der VPC, sodass sie sicher mit anderen AWS-Services (wie RDS-Datenbanken) innerhalb derselben VPC interagieren können.

Der Lambda-Service verwendet eine Plattform zur Virtualisierung von Netzwerkfunktionen, um dem Kunden NAT-Funktionen von der Lambda-VPC aus bereitzustellen. VPCs Dadurch werden die erforderlichen elastischen Netzwerkschnittstellen (ENIs) an dem Punkt konfiguriert, an dem Lambda-Funktionen erstellt oder aktualisiert werden. Es ermöglicht auch die gemeinsame

Nutzung ENIs von Ihrem Konto in mehreren Ausführungsumgebungen, sodass Lambda eine begrenzte Netzwerkressource effizienter nutzen kann, wenn Funktionen skaliert werden.

Da es ENIs sich um eine erschöpfbare Ressource handelt und es ein Soft-Limit von 250 ENIs pro Region gibt, sollten Sie die Nutzung von elastic network interface überwachen, wenn Sie Lambda-Funktionen für den VPC-Zugriff konfigurieren. Lambda-Funktionen in derselben AZ und derselben Sicherheitsgruppe können gemeinsam ENIs genutzt werden. Wenn Sie die Gleichzeitigkeitsgrenzen in Lambda erhöhen, sollten Sie generell prüfen, ob Sie eine Erhöhung der elastischen Netzwerkschnittstelle benötigen. Wenn das Limit erreicht wird, führt dies dazu, dass Aufrufe von VPC-fähigen Lambda-Funktionen gedrosselt werden.

Pricing model

Die Fargate-Preise basieren auf den Ressourcen, die Ihren Containern zugewiesen sind, insbesondere auf der vCPU und dem Arbeitsspeicher, den Sie für jede Aufgabe auswählen. Die CPU und der Arbeitsspeicher, die Ihre Container verwenden, werden Ihnen pro Sekunde mit einer Mindestgebühr von einer Minute in Rechnung gestellt. Die Kosten hängen direkt von den Ressourcen ab, die Ihre Anwendung verbraucht, d. h. Sie zahlen für das, was Sie bereitstellen, unabhängig davon, ob die Anwendung aktiv Anfragen verarbeitet. Fargate eignet sich gut für vorhersehbare Workloads, bei denen Sie spezifische Ressourcenkonfigurationen benötigen, und Sie können die Kosten optimieren, indem Sie die zugewiesenen Ressourcen anpassen. Darüber hinaus können zusätzliche Gebühren für verwandte Dienste wie Datenübertragung, Speicherung und Netzwerke (z. B. VPC, Elastic Load Balancing) anfallen.

Lambda hat eine andere Preisstruktur, die ereignisgesteuert ist und. pay-per-execution Die Gebühren richten sich nach der Anzahl der Anfragen, die Ihre Funktionen erhalten, und nach der Dauer jeder Ausführung, gemessen in Millisekunden. Lambda berücksichtigt auch die Speichermenge, die Sie Ihrer Funktion zuweisen, wobei die Kosten auf der Grundlage des verwendeten Speichers und der Ausführungszeit skalieren. Das Preismodell beinhaltet ein kostenloses Kontingent mit 1 Million kostenlosen Anfragen und 400.000 GB-Sekunden Rechenzeit pro Monat, was Lambda besonders kostengünstig für sporadische Workloads mit geringem Volumen macht.

Das Lambda-Preismodell ist ideal für Anwendungen mit unvorhersehbaren oder stark frequentierten Datenverkehrsmustern, da Sie nur für die tatsächlichen Funktionsaufrufen und die Ausführungszeit zahlen, ohne dass ungenutzte Kapazität bereitgestellt oder dafür bezahlt werden muss.

Verwenden Sie

Nachdem Sie sich mit den Kriterien für die Wahl zwischen AWS Fargate und vertraut gemacht haben AWS Lambda, können Sie den Service auswählen, der Ihren Anforderungen entspricht. Anhand der folgenden Informationen können Sie sich bei den ersten Schritten mit den einzelnen Diensten vertraut machen.

AWS Fargate

- Erfahren Sie, wie Sie eine Amazon ECS-Linux-Aufgabe für den Starttyp Fargate erstellen

  Beginnen Sie mit der Aktivierung AWS Fargate von Amazon ECS, indem Sie den Starttyp Fargate für Ihre Linux-Aufgaben verwenden.

  Erkunden Sie den Leitfaden
- Erfahren Sie, wie Sie eine Amazon ECS-Windows-Aufgabe für den Starttyp Fargate erstellen

  Beginnen Sie mit der Aktivierung AWS Fargate von Amazon ECS, indem Sie den Fargate-Starttyp für Ihre Windows-Aufgaben verwenden.

  Erkunden Sie den Leitfaden
- Erste Schritte mit Fargate und Amazon EKS

  In diesem Handbuch werden die ersten Schritte zum Ausführen Ihrer Pods AWS Fargate mit Ihrem Amazon EKS-Cluster beschrieben.

  Erkunden Sie den Leitfaden
- AWS Fargate Preisgestaltung

  In diesem Leitfaden erfahren Sie, wie sich vCPU-, Arbeitsspeicher-, Speicher- und Betriebssystemkonfigurationen auf die AWS Fargate Preisgestaltung auswirken.

  Erkunden Sie den Leitfaden
- AWS Fargate häufig gestellte Fragen

  Hier erhalten Sie Antworten auf häufig gestellte Fragen zu AWS Fargate Funktionen und bewährten Methoden für die Implementierung.

  Erkunden Sie den Leitfaden

AWS Lambda

- Erstellen Sie eine serverlose Dateiverarbeitungs-App

  Eine step-by-step schrittweise Anleitung zur Einrichtung und Verwendung von Amazon SNS. Es behandelt Themen wie das Erstellen eines Themas, das Abonnieren eines Themas für Endpunkte, das Veröffentlichen von Nachrichten und das Konfigurieren von Zugriffsberechtigungen.

  Erkunden Sie den Leitfaden
- Serverless-Entwicklerhandbuch

  Dieser Leitfaden hilft Ihnen dabei, ein besseres konzeptionelles Verständnis für die Entwicklung serverloser Anwendungen zu entwickeln und zu erfahren, wie verschiedene Methoden AWS-Services zusammenpassen, um Anwendungsmuster zu erstellen, die den Kern Ihrer Cloud-Anwendungen bilden.

  Erkunden Sie den Leitfaden
- Serverloses Land

  Diese Website enthält die neuesten Informationen, Blogs, Videos, Code und Lernressourcen für AWS Serverless. Erfahren Sie, wie Sie Apps verwenden und erstellen, die sich auf einer kostengünstigen, vollständig verwalteten serverlosen Architektur automatisch skalieren lassen.

  Erkunden Sie die Website
- AWS Lambda Preisgestaltung

  Verwenden Sie diesen Leitfaden, um die Kosten abzuschätzen und die Kosten auf der Grundlage der Nutzung und Konfiguration der Funktionen zu optimieren. Es enthält einen Preisrechner, mit dem Sie Ihre Kosten AWS Lambda und Ihre Architekturkosten in einer einzigen Schätzung berechnen können.

  Erkunden Sie den Leitfaden
- AWS Lambda häufig gestellte Fragen

  Hier erhalten Sie Antworten auf häufig gestellte Fragen zu AWS Lambda Funktionen und bewährten Methoden für die Implementierung.

  Erkunden Sie den Leitfaden

Dokumentenhistorie für AWS Fargate oder AWS Lambda?

In der folgenden Tabelle werden die wichtigen Änderungen an diesem Entscheidungsleitfaden beschrieben. Für Benachrichtigungen über Aktualisierungen dieses Handbuchs können Sie einen RSS-Feed abonnieren.

Änderung	Beschreibung	Datum
Erstversion	Erste Veröffentlichung des Entscheidungsleitfadens.	15. November 2024

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.