AWS Guía de decisiones

AWS Fargate ¿o AWS Lambda?

AWS Fargate ¿o AWS Lambda?: AWS Guía de decisiones

Table of Contents

AWS Fargate ¿o AWS Lambda?

Comprenda las diferencias y elija la que sea adecuada para usted

Finalidad	Para determinar si satisface sus AWS Lambda necesidades de un servicio de cómputo sin servidor AWS Fargate o no.
Última actualización	15 de noviembre de 2024
Servicios cubiertos	• AWS Fargate • AWS Lambda

Introducción

Antes de empezar a decidir si elegir un servicio de computación sin servidor AWS Lambda o AWS Fargate como tal, es probable que haya considerado la gama más amplia de servicios de AWS computación (que se describe en la guía de decisiones sobre cómo elegir un servicio de AWS computación) y la haya reducido a estas dos opciones, ya que ofrecen:

- Reducción de la sobrecarga operativa: tanto Lambda como Fargate eliminan la administración de servidores, lo que reduce la necesidad de parches, mantenimiento y planificación de la capacidad.
- Pay-per-use precios: solo paga por los recursos informáticos que realmente utiliza, lo que podría reducir los costes de las cargas de trabajo variables.
- Implementación más rápida: normalmente ofrece tiempos de implementación más rápidos en comparación con el aprovisionamiento y la configuración de EC2 instancias.
- Alta disponibilidad integrada: ambos servicios gestionan la redundancia de la infraestructura de forma automática.
- Cumplimiento simplificado: la reducción de la superficie de ataque y las funciones de seguridad integradas pueden facilitar los esfuerzos de cumplimiento.
- Céntrese en el código: los desarrolladores pueden concentrarse más en escribir el código de la aplicación que en administrar la infraestructura.

Si bien Lambda y Fargate son opciones sin servidor, existen diferencias significativas entre ellas:

AWS Fargatees un motor de procesamiento sin servidor para contenedores, que se utiliza principalmente con Amazon ECS. Administra automáticamente su infraestructura, lo que le permite centrarse en implementar y escalar aplicaciones en contenedores. Fargate es ideal para aplicaciones de ejecución prolongada, microservicios o procesamiento por lotes, en los que necesita un control detallado de la asignación de recursos (CPU, memoria) y desea evitar la administración de los servidores subyacentes.

AWS Lambdaes un servicio informático sin servidor que ejecuta automáticamente el código en respuesta a los eventos y administra los recursos informáticos subyacentes. Es ideal para aplicaciones basadas en eventos, como procesar archivos cargados en Amazon S3, responder a solicitudes HTTP o ejecutar tareas programadas. Lambda también es adecuada para aplicaciones de procesamiento de flujos y procesamiento de datos debido a su capacidad para escalar automáticamente en respuesta a eventos y gestionar grandes volúmenes de datos en tiempo real. Lambda puede procesar flujos de datos de fuentes como Amazon Kinesis o Amazon DynamoDB, lo que permite realizar transformaciones, filtrados y análisis de datos eficientes y sin servidor sin necesidad de administrar la infraestructura. Lambda está diseñada para tareas de corta duración (hasta 15 minutos) y se factura en función del número de solicitudes y del tiempo de ejecución, lo que la hace rentable para cargas de trabajo esporádicas.

Si su proyecto incluye tareas de corta duración basadas en eventos o cargas de trabajo impredecibles, podría ser la mejor opción. AWS Lambda Si necesita ejecutar aplicaciones en contenedores con necesidades de recursos específicas (o necesita procesos persistentes), sería más adecuado. AWS Fargate

La siguiente tabla ofrece un análisis más detallado de algunas de las diferencias entre estos servicios para que pueda empezar.

Característica	AWS Fargate	AWS Lambda
Modelo de ejecución	Computación basada en contenedores y sin servidor	Funciones sin servidor y basadas en eventos
Lenguajes admitidos	Cualquier idioma que pueda ejecutarse en un contenedor	Lenguajes compatibles: Node.js, Python, Java, C#, Go, Ruby y PowerShell. También puede crear un tiempo de ejecución personalizado para implementar una

		AWS Lambda función en el idioma que prefiera.
Caso de uso	Aplicaciones contenerizadas de larga duración	Tareas de corta duración basadas en eventos
Escalado	Escalado automático en función del número de tareas deseado	Escalado automático por solicitud
Arranque en frío	De 35 segundos a 2 minutos	De 100 ms a 2 segundos
Límite de tiempo de ejecución	Sin límite estricto	15 minutos como máximo
Asignación de memoria	Hasta 120 GiB	Hasta 10 GiB
Asignación de CPU	Hasta 16 vCPU	Proporcional a la memoria, hasta 6 vCPU
Red	Se ejecuta en VPC, puede usar ENIs	Puede ejecutarse en una VPC AWS gestionada o adjuntarse a una VPC gestionada por el cliente mediante Hyperplane AWS
Administración de estados	Los contenedores de Fargate pueden mantener el estado de todas las solicitudes mientras el contenedor esté en ejecución, lo que permite gestionar sesiones, almacenar datos en caché o mantener el estado en memoria sin necesidad de almacenamiento externo. Se recomienda el almacenamiento externo para los datos críticos.	Sin estado por diseño (el estado debe administr arse externamente, por ejemplo, Amazon S3, Amazon DynamoDB, Amazon EFS)

Compatibilidad con contenedor	Soporta contenedores	Soporte limitado de contenedores (mediante despliegues de imágenes de contenedores)
Orquestación	Integrado con Amazon ECS	No se requiere orquestación
Modelo de precios	Facturación por segundo para la vCPU y la memoria utilizadas	Por invocación y duración (GB-segundos)
Límites de simultaneidad	Basado en la capacidad del clúster	1000 ejecuciones simultáneas de forma predeterminada (se puede aumentar)
Invocación basada en eventos	Requiere configuración adicional	Soporte nativo para varias fuentes de AWS eventos
Mitigación del arranque en frío	La carga lenta de imágenes con Seekable OCI puede acelerar el inicio de las tareas de Fargate	La simultaneidad aprovisionada está disponible
Límite de tamaño de paquete	Sin límite específico (el tamaño del contenedor está limitado por el almacenamiento efímero configurado, 200 GiB como máximo)	250 MB descomprimidos, incluidas las capas, y 10 GB para despliegues de imágenes en contenedores

Diferencias entre Fargate y Lambda

Explore las diferencias entre Fargate y Lambda en una serie de áreas clave.

Languages supported

Fargate: AWS Fargate es un servicio de orquestación de contenedores, lo que significa que es compatible con cualquier lenguaje de programación o entorno de ejecución que se pueda empaquetar en un contenedor Docker. Esta flexibilidad permite a los desarrolladores utilizar prácticamente cualquier lenguaje, marco o biblioteca que se adapte a las necesidades de sus

aplicaciones. Ya sea que utilice Python, Java, Node.js, Go, .NET, Ruby, PHP o incluso lenguajes y entornos personalizados, Fargate puede ejecutarlos siempre que estén encapsulados en un contenedor. Esta amplia compatibilidad lingüística hace que Fargate sea ideal para ejecutar diversas aplicaciones, incluidos sistemas heredados, microservicios multilingües y aplicaciones modernas nativas de la nube.

Lambda: AWS Lambda ofrece soporte nativo para un conjunto de lenguajes más limitado en comparación con Fargate, diseñado específicamente para funciones basadas en eventos. A partir de ahora, Lambda admite oficialmente los siguientes lenguajes:

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

Lambda también admite tiempos de ejecución personalizados, lo que le permite crear su propio idioma o entorno de ejecución, pero esto requiere más configuración y administración en comparación con el uso de las opciones compatibles de forma nativa. Si decide implementar su función Lambda desde una imagen de contenedor, puede escribir su función en Rust utilizando una imagen base AWS exclusiva para el sistema operativo e incluyendo el cliente de tiempo de ejecución de Rust en su imagen. Si utilizas un lenguaje que no tiene un cliente de interfaz AWS de ejecución proporcionado, debes crear el tuyo propio.

Event-driven invocation

Lambda está diseñada intrínsecamente para la informática basada en eventos. Las funciones Lambda se activan en respuesta a una variedad de eventos, incluidos cambios en los datos, acciones del usuario o tareas programadas. Se integra perfectamente con muchos Servicios de AWS, como Amazon S3 (por ejemplo, al invocar una función cuando se carga un archivo), DynamoDB (por ejemplo, se activa cuando se actualizan los datos) y API Gateway (por ejemplo, al gestionar las solicitudes HTTP). La arquitectura Lambda basada en eventos es ideal para las aplicaciones que necesitan responder inmediatamente a los eventos sin requerir recursos de cómputo persistentes.

Fargate no se basa en eventos de forma nativa, pero con una lógica repetitiva adicional, puede integrarse con fuentes de eventos como Amazon SQS y Kinesis. Mientras que Lambda se encarga de la mayor parte de esta lógica de integración por usted, tendrá que implementar esta integración usted mismo utilizando estos APIs servicios.

Runtime/use cases

Fargate está diseñado para ejecutar aplicaciones en contenedores, lo que proporciona un entorno de ejecución flexible en el que puede definir la configuración de CPU, memoria y red para sus contenedores. Como Fargate funciona con un modelo basado en contenedores, admite procesos de ejecución prolongada, servicios persistentes y aplicaciones con requisitos de tiempo de ejecución específicos. Los contenedores de Fargate pueden funcionar indefinidamente, ya que no hay un límite estricto en el tiempo de ejecución, lo que los hace ideales para aplicaciones que necesitan estar en funcionamiento de forma continua.

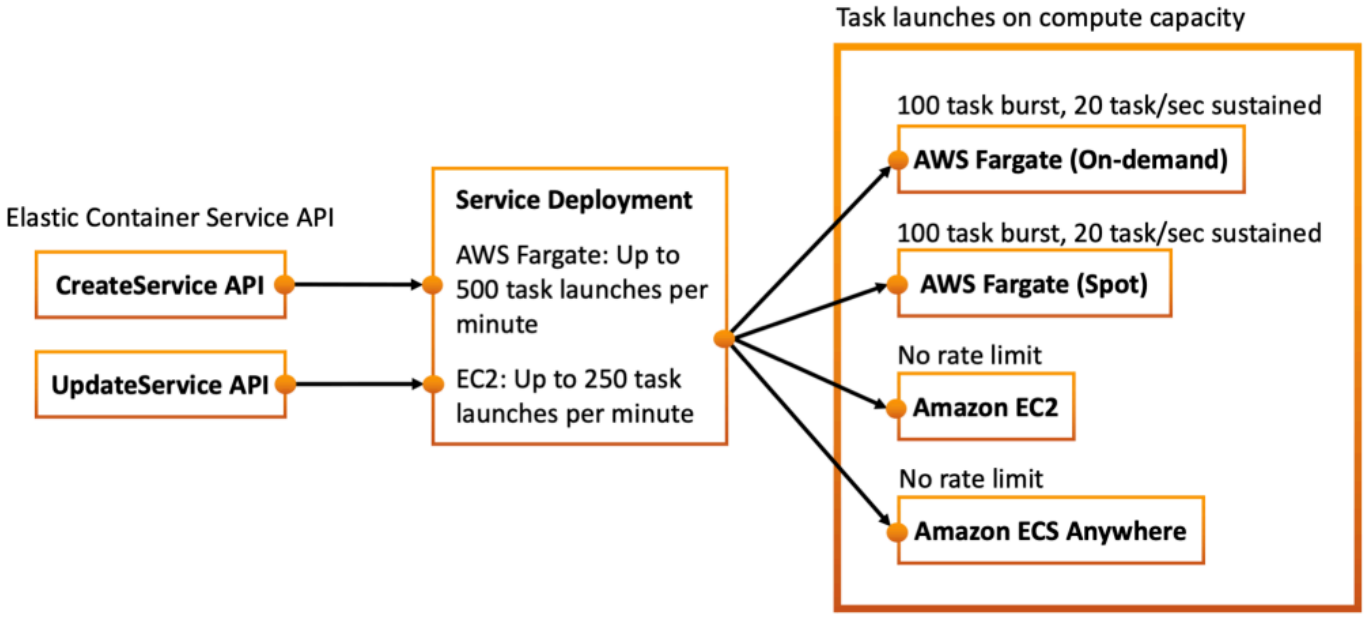
Lambda, por otro lado, está optimizada para tareas de corta duración basadas en eventos. Las funciones Lambda se ejecutan en un entorno sin estado en el que el tiempo máximo de ejecución está limitado a 15 minutos. Esto hace que Lambda sea adecuada para escenarios como el procesamiento de archivos, la transmisión de datos en tiempo real y la gestión de solicitudes HTTP, en los que las tareas son breves y no requieren procesos de ejecución prolongada.

En Lambda, el entorno de ejecución es más abstracto y hay menos control sobre la infraestructura subyacente. La naturaleza sin estado de Lambda significa que cada invocación de función es independiente y cualquier estado o dato que deba persistir entre las invocaciones debe gestionarse externamente (por ejemplo, en bases de datos o servicios de almacenamiento).

Scaling

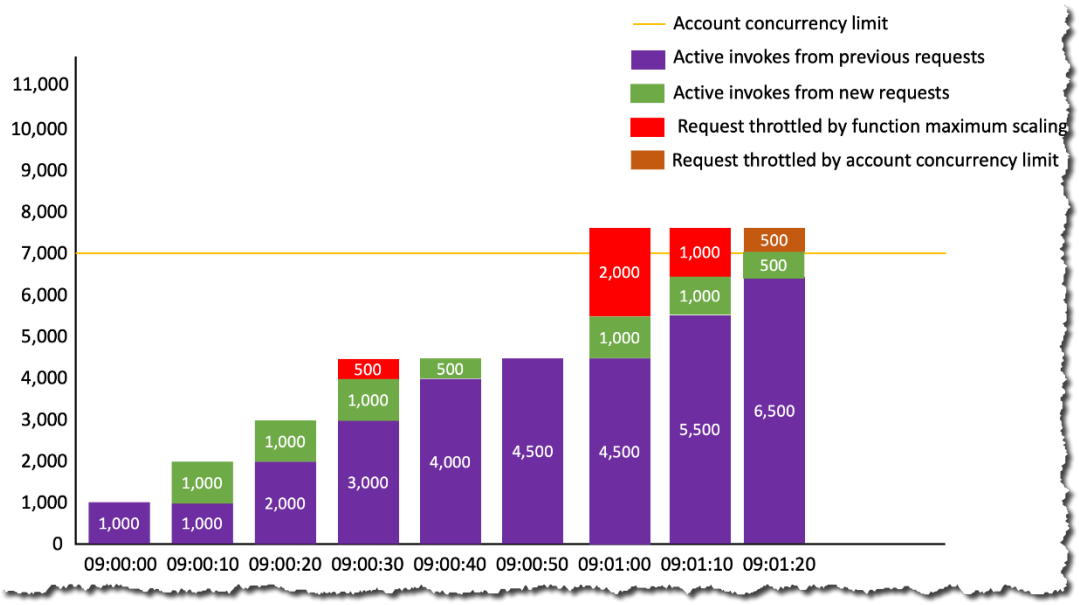
Fargate escala ajustando la cantidad de contenedores en ejecución en función del estado deseado definido en su servicio de organización de contenedores (Amazon ECS). Este escalado se puede realizar manual o automáticamente a través de Amazon EC2 Auto Scaling. Esta entrada de blog ofrece más detalles sobre cómo hacerlo.

En Fargate, cada contenedor funciona en su entorno aislado, y el escalado implica lanzar contenedores adicionales o detenerlos en función de la carga. El programador de servicios Amazon ECS puede lanzar hasta 500 tareas en menos de un minuto por servicio para servicios web y otros servicios de larga duración.



Para Lambda, la simultaneidad es el número de solicitudes en movimiento que su AWS Lambda función gestiona al mismo tiempo. Esto difiere de la simultaneidad de Fargate, donde cada tarea de Fargate puede gestionar solicitudes simultáneas siempre que haya recursos informáticos y de red disponibles. Para cada solicitud simultánea, Lambda aprovisiona una instancia independiente del entorno de ejecución. A medida que las funciones reciben más solicitudes, Lambda se encarga automáticamente de escalar la cantidad de entornos de ejecución hasta que alcance el límite de simultaneidad de la cuenta. De forma predeterminada, Lambda proporciona a su cuenta un límite total de simultaneidad de 1000 ejecuciones simultáneas en todas las funciones de una cuenta Región de AWS, y puede solicitar un aumento de cuota si es necesario.

Para cada función Lambda de una región, la tasa de escalado simultáneo es de 1000 instancias de ejecución cada 10 segundos, hasta la simultaneidad máxima de la cuenta. Como se explica en este blog, si el número de solicitudes en un período de 10 segundos supera las 1000, las solicitudes adicionales se reducirán. El siguiente gráfico muestra cómo funciona el escalado Lambda suponiendo una simultaneidad de cuentas de 7000.



Cold start and cold-start mitigation

Lambda puede experimentar arranques en frío, que se producen cuando se invoca una función después de haber estado inactiva durante algún tiempo. Durante un arranque en frío, el servicio Lambda necesita inicializar un nuevo entorno de ejecución, incluida la carga del tiempo de ejecución, las dependencias y el código de la función. Este proceso puede introducir latencia, especialmente en lenguajes con tiempos de inicialización más largos (por ejemplo, Java o C#). Los arranques en frío pueden afectar al rendimiento de las aplicaciones, especialmente de las que requieren respuestas de baja latencia.

Para mitigar los arranques en frío en Lambda, se pueden emplear varias estrategias:

- Minimice el tamaño de la función: reducir el tamaño del paquete de funciones y sus dependencias puede reducir el tiempo necesario para la inicialización.
- Aumente la asignación de memoria: las asignaciones de memoria más altas aumentan la capacidad de la CPU, lo que podría reducir el tiempo de inicialización.
- Mantenga las funciones calientes: si invoca periódicamente las funciones de Lambda (por ejemplo, CloudWatch mediante Events), puede mantenerlas activas y reducir la probabilidad de arranques en frío.
- Lambda SnapStart: utilice Lambda SnapStart para funciones de Java para reducir el tiempo de inicio.
- Simultaneidad aprovisionada: esta función mantiene un número específico de instancias de funciones activas y listas para atender las solicitudes, lo que reduce la latencia de arranque en

frío. Sin embargo, aumenta los costes a medida que se pagan las instancias aprovisionadas, incluso si no gestionan las solicitudes de forma activa.

Por lo general, Fargate no se ve afectado por los arranques en frío de la misma manera que Lambda. El tiempo que se tarda en iniciar una tarea de Fargate está directamente relacionado con el tiempo que se tarda en extraer del registro de imágenes las imágenes del contenedor definidas en la tarea. Fargate también admite la carga diferida de imágenes de contenedores que se han indexado con Seekable OCI (SOCI). La carga diferida de imágenes de contenedores con SOCI reduce el tiempo necesario para lanzar las tareas de Amazon ECS en Fargate. Fargate gestiona contenedores que permanecen activos durante el tiempo que sea necesario, lo que significa que siempre están listos para gestionar las solicitudes. Sin embargo, si necesita iniciar nuevos contenedores en respuesta a eventos de escalado, es posible que se produzca algún retraso mientras se inicializan los contenedores, pero esto suele ser menos significativo en comparación con los arranques en frío de Lambda.

Memory and CPU options

Fargate proporciona un control detallado de los recursos de memoria y CPU para sus aplicaciones en contenedores. Al iniciar una tarea en Fargate, puede especificar los requisitos exactos de CPU y memoria en función de las necesidades de la aplicación. Las asignaciones de CPU y memoria son independientes, lo que le permite elegir las combinaciones que mejor se adapten a su carga de trabajo. Por ejemplo, puede seleccionar valores de CPU que oscilan entre 0,25 V CPUs y 16 V CPUs y memoria entre 0,5 GB y 120 GB por contenedor, según la configuración.

Esta flexibilidad es ideal para ejecutar aplicaciones que requieren características de rendimiento específicas, como bases de datos con un uso intensivo de memoria o tareas de computación vinculadas a la CPU. Fargate le permite optimizar la asignación de recursos para equilibrar el costo y el rendimiento de manera efectiva.

En Lambda, la memoria y la CPU están vinculadas, y la CPU se asigna automáticamente en proporción a la cantidad de memoria que seleccione. Puede elegir asignaciones de memoria de entre 128 MB y 10 GB, en incrementos de 1 MB. La CPU se amplía con la memoria, hasta 6 vCPU, lo que significa que una configuración de memoria más alta da como resultado una mayor potencia de la CPU, pero no tienes control directo sobre la asignación de la CPU en sí.

Este modelo está diseñado pensando en la simplicidad, lo que permite a los desarrolladores ajustar rápidamente la configuración de la memoria sin necesidad de administrar las configuraciones de la CPU. Sin embargo, puede ser menos flexible para las cargas de trabajo que

requieren un equilibrio específico entre los recursos de CPU y memoria. El modelo de Lambda es adecuado para tareas en las que se desea un escalado sencillo en función de las necesidades de memoria, pero puede que no sea óptimo para aplicaciones con demandas de recursos complejas o muy específicas.

Networking

Cuando despliega tareas en Fargate, estas se ejecutan en una Amazon VPC (Amazon Virtual Private Cloud), lo que le proporciona un control total sobre el entorno de red. Esto incluye la configuración de grupos de seguridad, listas de control de acceso a la red (ACLs) y tablas de enrutamiento. Cada tarea de Fargate tiene su propia interfaz de red, con una dirección IP privada dedicada, y se le puede asignar una dirección IP pública si es necesario.

Fargate admite funciones de red avanzadas, como el equilibrio de carga (mediante AWS Elastic Load Balancing), la interconexión de VPC y el acceso directo a otras funciones de la VPC. Servicios de AWS También se puede utilizar AWS PrivateLink como conexión segura y privada a la compatible Servicios de AWS, sin necesidad de tener que atravesar Internet.

De forma predeterminada, las funciones de Lambda se ejecutan en un entorno de red gestionado sin control directo sobre las interfaces de red o las direcciones IP. Sin embargo, Lambda se puede conectar a una VPC gestionada por el cliente AWS mediante Hyperplane, lo que le permite controlar el acceso a los recursos de la VPC.

Cuando las funciones de Lambda se adjuntan a una VPC gestionada por el cliente, heredan los grupos de seguridad y las configuraciones de subred de la VPC, lo que les permite interactuar de forma segura con otras (como las bases de datos de RDS) dentro de la misma VPC. Servicios de AWS

El servicio Lambda utiliza una plataforma de virtualización de funciones de red para proporcionar capacidades de NAT desde la VPC de Lambda al cliente. VPCs Esto configura las interfaces de red elásticas requeridas (ENIs) en el punto en el que se crean o actualizan las funciones Lambda. También permite que su cuenta se comparta en varios entornos ENIs de ejecución, lo que permite a Lambda hacer un uso más eficiente de un recurso de red limitado cuando las funciones se escalan.

Dado que ENIs se trata de un recurso agotable y hay un límite flexible de 250 ENIs por región, debe supervisar el uso de la interfaz de red elástica si va a configurar las funciones de Lambda para el acceso a la VPC. Las funciones Lambda de la misma zona de disponibilidad y el mismo grupo de seguridad pueden compartir. ENIs Por lo general, si aumenta los límites de

simultaneidad en Lambda, debe evaluar si necesita un aumento de la interfaz de red elástica. Si se alcanza el límite, se limitan las invocaciones de funciones de Lambda habilitadas para VPC.

Pricing model

Los precios de Fargate se basan en los recursos asignados a sus contenedores, específicamente la vCPU y la memoria que seleccione para cada tarea. Se le facturará por segundo, con un cargo mínimo de un minuto, por la CPU y la memoria que utilicen sus contenedores. Los costos están directamente relacionados con los recursos que consume su aplicación, lo que significa que usted paga por lo que aprovisiona, independientemente de si la aplicación procesa activamente las solicitudes. Fargate es ideal para cargas de trabajo predecibles en las que se necesitan configuraciones de recursos específicas y puede optimizar los costos ajustando los recursos asignados. Además, es posible que se apliquen cargos adicionales por los servicios relacionados, como la transferencia de datos, el almacenamiento y las redes (por ejemplo, VPC o Elastic Load Balancing).

Lambda tiene una estructura de precios diferente que se basa en eventos y. pay-per-execution Se le cobrará en función del número de solicitudes que reciban sus funciones y de la duración de cada ejecución, medida en milisegundos. Lambda también tiene en cuenta la cantidad de memoria que se asigna a la función, y los costos se escalan en función de la memoria utilizada y del tiempo de ejecución. El modelo de precios incluye una capa gratuita, que ofrece 1 millón de solicitudes gratuitas y 400 000 GB por segundo de tiempo de procesamiento al mes, lo que hace que Lambda sea especialmente rentable para cargas de trabajo esporádicas y de bajo volumen.

El modelo de precios de Lambda es ideal para aplicaciones con patrones de tráfico impredecibles o en ráfagas, ya que solo paga por las invocaciones de funciones reales y el tiempo de ejecución, sin necesidad de aprovisionar o pagar por la capacidad inactiva.

Uso

Ahora que ha leído los criterios para elegir entre AWS Fargate y AWS Lambda, puede seleccionar el servicio que mejor se adapte a sus necesidades y utilizar la siguiente información para empezar a utilizar cada uno de ellos.

AWS Fargate

- Aprenda a crear una tarea de Amazon ECS Linux para el tipo de lanzamiento de Fargate

Comience a utilizar Amazon ECS AWS Fargate utilizando el tipo de lanzamiento Fargate para sus tareas de Linux.

Explore la guía

- Aprenda a crear una tarea de Amazon ECS para Windows para el tipo de lanzamiento de Fargate

  Comience a utilizar Amazon ECS AWS Fargate utilizando el tipo de lanzamiento Fargate para sus tareas de Windows.

  Explore la guía
- Cómo empezar a usar Fargate y Amazon EKS

  En esta guía, se describe cómo empezar a ejecutar los pods en el AWS Fargate clúster de Amazon EKS.

  Consulta la guía
- AWS Fargate precios

  Utilice esta guía para entender cómo afectan AWS Fargate a los precios las configuraciones de vCPU, memoria, almacenamiento y sistema operativo.

  Explore la guía
- AWS Fargate preguntas frecuentes

  Obtenga respuestas a preguntas frecuentes sobre AWS Fargate las capacidades y las mejores prácticas de implementación.

  Explore la guía

AWS Lambda

- Cree una aplicación de procesamiento de archivos sin servidor

  Un step-by-step recorrido por la configuración y el uso de Amazon SNS. Abarca temas como la creación de un tema, la suscripción de puntos de enlace a un tema, la publicación de mensajes y la configuración de los permisos de acceso.

  Explore la guía

- Guía para desarrolladores sin servidor

  Esta guía le ayuda a comprender mejor desde el punto de vista conceptual el desarrollo de aplicaciones sin servidor y cómo Servicios de AWS se combinan para crear patrones de aplicaciones que constituyen el núcleo de sus aplicaciones en la nube.

  Explore la guía
- Tierra sin servidores

  Este sitio reúne la información, los blogs, los videos, el código y los recursos de aprendizaje más recientes sobre AWS Serverless. Aprenda a usar y crear aplicaciones que se escalen automáticamente en una arquitectura sin servidor totalmente gestionada y de bajo coste.

  Explore el sitio
- AWS Lambda precios

  Utilice esta guía para estimar los gastos y optimizar los costes en función del uso y la configuración de las funciones. Incluye una calculadora de precios para calcular sus costes AWS Lambda y los de la arquitectura en una sola estimación.

  Explore la guía
- AWS Lambda preguntas frecuentes

  Obtenga respuestas a preguntas frecuentes sobre AWS Lambda las capacidades y las mejores prácticas de implementación.

  Explore la guía

¿Historial de documentos para AWS Fargate o AWS Lambda?

En la siguiente tabla se describen los cambios importantes en esta guía de decisiones. Para recibir notificaciones sobre las actualizaciones de esta guía, puede suscribirse a una fuente RSS.

Cambio	Descripción	Fecha
Versión inicial	Versión inicial de la guía de decisiones.	15 de noviembre de 2024

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.