AWS Guide de décision

AWS Fargate ou AWS Lambda ?

AWS Fargate ou AWS Lambda ?: AWS Guide de décision

Table of Contents

AWS Fargate ou AWS Lambda ?

Comprenez les différences et choisissez celui qui vous convient

Objectif	Pour déterminer si vous avez besoin d'un service de calcul sans serveur AWS Fargate ou si vous y AWS Lambda répondez.
Dernière mise à jour	15 novembre 2024
Services couverts	• AWS Fargate • AWS Lambda

Introduction

Avant de commencer à déterminer si vous optez pour un service de calcul sans serveur AWS Lambda ou en AWS Fargate tant que tel, vous avez probablement examiné la gamme plus large de services AWS informatiques (décrite dans le guide de décision Choisir un service de AWS calcul) et l'avez réduite à ces deux choix, car ils fournissent :

- Réduction des frais d'exploitation : Lambda et Fargate simplifient la gestion des serveurs, réduisant ainsi le besoin de correctifs, de maintenance et de planification des capacités.
- Pay-per-use tarification : vous ne payez que pour les ressources informatiques que vous utilisez réellement, ce qui peut réduire les coûts liés aux charges de travail variables.
- Déploiement plus rapide : offre généralement des délais de déploiement plus courts que le provisionnement et la configuration des EC2 instances.
- Haute disponibilité intégrée : les deux services gèrent automatiquement la redondance de l'infrastructure.
- Conformité simplifiée : la surface d'attaque réduite et les fonctionnalités de sécurité intégrées peuvent faciliter les efforts de mise en conformité.
- Concentrez-vous sur le code : les développeurs peuvent se concentrer davantage sur l'écriture du code d'application plutôt que sur la gestion de l'infrastructure.

Lambda et Fargate sont toutes deux des options sans serveur, mais elles présentent des différences importantes :

AWS Fargateest un moteur de calcul sans serveur pour les conteneurs, principalement utilisé avec Amazon ECS. Il gère automatiquement votre infrastructure, ce qui vous permet de vous concentrer sur le déploiement et le dimensionnement des applications conteneurisées. Fargate est idéal pour les applications de longue durée, les microservices ou le traitement par lots, lorsque vous avez besoin d'un contrôle précis de l'allocation des ressources (processeur, mémoire) et que vous souhaitez éviter de gérer des serveurs sous-jacents.

AWS Lambdaest un service informatique sans serveur qui exécute automatiquement votre code en réponse à des événements et gère les ressources informatiques sous-jacentes. Il convient parfaitement aux applications axées sur les événements, telles que le traitement de fichiers téléchargés sur Amazon S3, la réponse à des requêtes HTTP ou l'exécution de tâches planifiées. Lambda convient également parfaitement aux applications de traitement de flux et de traitement de données en raison de sa capacité à évoluer automatiquement en réponse aux événements et à gérer de gros volumes de données en temps réel. Lambda peut traiter des flux de données provenant de sources telles qu'Amazon Kinesis ou Amazon DynamoDB, permettant ainsi des transformations, des filtrages et des analyses de données efficaces et sans serveur sans gestion d'infrastructure. Lambda est conçu pour les tâches de courte durée (jusqu'à 15 minutes) et est facturé en fonction du nombre de demandes et du temps d'exécution, ce qui le rend rentable pour les charges de travail sporadiques.

Si votre projet implique des tâches de courte durée basées sur des événements ou des charges de travail imprévisibles, c'est AWS Lambda peut-être la solution la mieux adaptée. Si vous devez exécuter des applications conteneurisées avec des besoins spécifiques en ressources (ou si vous avez besoin de processus persistants), AWS Fargate ce serait plus approprié.

Le tableau suivant donne un aperçu plus détaillé de certaines des différences entre ces services pour vous aider à démarrer.

Fonctionnalité	AWS Fargate	AWS Lambda
Modèle d'exécution	Calcul sans serveur basé sur des conteneurs	Fonctions sans serveur pilotées par les événements
Langues prises en charge	Toute langue pouvant être exécutée dans un conteneur	Langages pris en charge : Node.js, Python, Java, C#, Go, Ruby et PowerShell. Vous

		pouvez également créer un environnement d'exécution personnalisé pour implément er une AWS Lambda fonction dans le langage de votre choix.
Cas d’utilisation	Applications conteneurisées de longue durée	Tâches de courte durée axées sur les événements
Mise à l’échelle	Mise à l'échelle automatiq ue en fonction du nombre de tâches souhaité	Dimensionnement automatiq ue par demande
Démarrage à froid	35 secondes à 2 minutes	100 ms à 2 secondes
Délai d'exécution	Aucune limite stricte	15 minutes maximum
Allocation de mémoire	Jusqu'à 120 GiB	Jusqu'à 10 GiB
Allocation du processeur	Jusqu'à 16 vCPU	Proportionnel à la mémoire, jusqu'à 6 vCPU
Réseaux	Fonctionne en VPC, peut utiliser ENIs	Peut être exécuté dans un VPC AWS géré ou attaché à un VPC géré par le client à l'aide d'Hyperplane AWS

Gestion des états	Les conteneurs de Fargate peuvent conserver leur état d'une requête à l'autre tant que le conteneur est en cours d'exécution, ce qui permet de gérer les sessions, de mettre en cache les données ou de conserver l'état en mémoire sans avoir besoin de stockage externe. Le stockage externe est recommandé pour les données critiques.	Apatride par conception (l'état doit être géré en externe, par exemple, Amazon S3, Amazon DynamoDB, Amazon EFS)
Prise en charge du conteneur	Supporte les conteneurs	Prise en charge limitée des conteneurs (via des déploiements d'images de conteneurs)
Orchestration	Intégré à Amazon ECS	Aucune orchestration requise
Modèle de tarification	Facturation par seconde pour le vCPU et la mémoire utilisés	Par invocation et durée (Go de secondes)
Limites de simultanéité	Basé sur la capacité du cluster	1000 exécutions simultanées par défaut (peut être augmenté)
Invocation basé sur les événements	Nécessite une configuration supplémentaire	Support natif pour diverses sources AWS d'événements
Réduction des démarrages à froid	Le chargement différé d'images avec Seekable OCI peut accélérer le démarrage des tâches Fargate	Concurrence provisionnée disponible
Limite de taille du package	Aucune limite spécifique (taille du conteneur limitée par le stockage éphémère configuré, 200 GiB maximum)	250 Mo décompressés, couches comprises, 10 Go pour les déploiements d'images de conteneurs

Différences entre Fargate et Lambda

Découvrez les différences entre Fargate et Lambda dans un certain nombre de domaines clés.

Languages supported

Fargate AWS Fargate : est un service d'orchestration de conteneurs, ce qui signifie qu'il prend en charge n'importe quel langage de programmation ou environnement d'exécution pouvant être intégré dans un conteneur Docker. Cette flexibilité permet aux développeurs d'utiliser pratiquement n'importe quel langage, framework ou bibliothèque répondant aux besoins de leurs applications. Que vous utilisiez Python, Java, Node.js, Go, .NET, Ruby, PHP ou même des langages et environnements personnalisés, Fargate peut les exécuter à condition qu'ils soient encapsulés dans un conteneur. Cette prise en charge linguistique étendue fait de Fargate la solution idéale pour exécuter diverses applications, notamment des systèmes existants, des microservices multilingues et des applications cloud natives modernes.

Lambda : AWS Lambda offre un support natif pour un ensemble de langages plus limité que Fargate, spécialement conçu pour les fonctions pilotées par les événements. À l'heure actuelle, Lambda prend officiellement en charge les langues suivantes :

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

Lambda prend également en charge les environnements d'exécution personnalisés, qui vous permettent d'utiliser votre propre langage ou environnement d'exécution, mais cela nécessite davantage de configuration et de gestion que l'utilisation des options prises en charge de manière native. Si vous choisissez de déployer votre fonction Lambda à partir d'une image de conteneur, vous pouvez écrire votre fonction dans Rust en utilisant une image de base AWS réservée au système d'exploitation et en incluant le client d'exécution Rust dans votre image. Si vous utilisez un langage qui ne possède pas de client d'interface d'exécution AWS fourni, vous devez créer le vôtre.

Event-driven invocation

Lambda est intrinsèquement conçu pour l'informatique pilotée par les événements. Les fonctions Lambda sont déclenchées en réponse à divers événements, notamment des modifications des données, des actions de l'utilisateur ou des tâches planifiées. Il s'intègre parfaitement à de nombreuses Services AWS applications, telles qu'Amazon S3 (par exemple, appel d'une fonction lors du chargement d'un fichier), DynamoDB (par exemple, déclenchement lors de mises à jour de données) et API Gateway (par exemple, gestion des requêtes HTTP). L'architecture Lambda pilotée par les événements est idéale pour les applications qui doivent répondre immédiatement aux événements sans avoir besoin de ressources informatiques persistantes.

Fargate n'est pas piloté nativement par les événements, mais grâce à une logique standard supplémentaire, il peut s'intégrer à des sources d'événements telles qu'Amazon SQS et Kinesis. Lambda gère l'essentiel de cette logique d'intégration pour vous, mais vous devrez implémenter cette intégration vous-même à l'aide APIs de ces services.

Runtime/use cases

Fargate est conçu pour exécuter des applications conteneurisées, fournissant un environnement d'exécution flexible dans lequel vous pouvez définir les paramètres du processeur, de la mémoire et du réseau pour vos conteneurs. Comme Fargate fonctionne sur un modèle basé sur des conteneurs, il prend en charge les processus de longue durée, les services persistants et les applications ayant des exigences d'exécution spécifiques. Les conteneurs de Fargate peuvent fonctionner indéfiniment, car le temps d'exécution n'est pas limité, ce qui le rend idéal pour les applications qui doivent être opérationnelles en permanence.

Lambda, en revanche, est optimisé pour les tâches de courte durée axées sur les événements. Les fonctions Lambda sont exécutées dans un environnement sans état où la durée d'exécution maximale est limitée à 15 minutes. Lambda convient donc parfaitement aux scénarios tels que le traitement de fichiers, le streaming de données en temps réel et le traitement des requêtes HTTP, dans lesquels les tâches sont brèves et ne nécessitent pas de processus de longue durée.

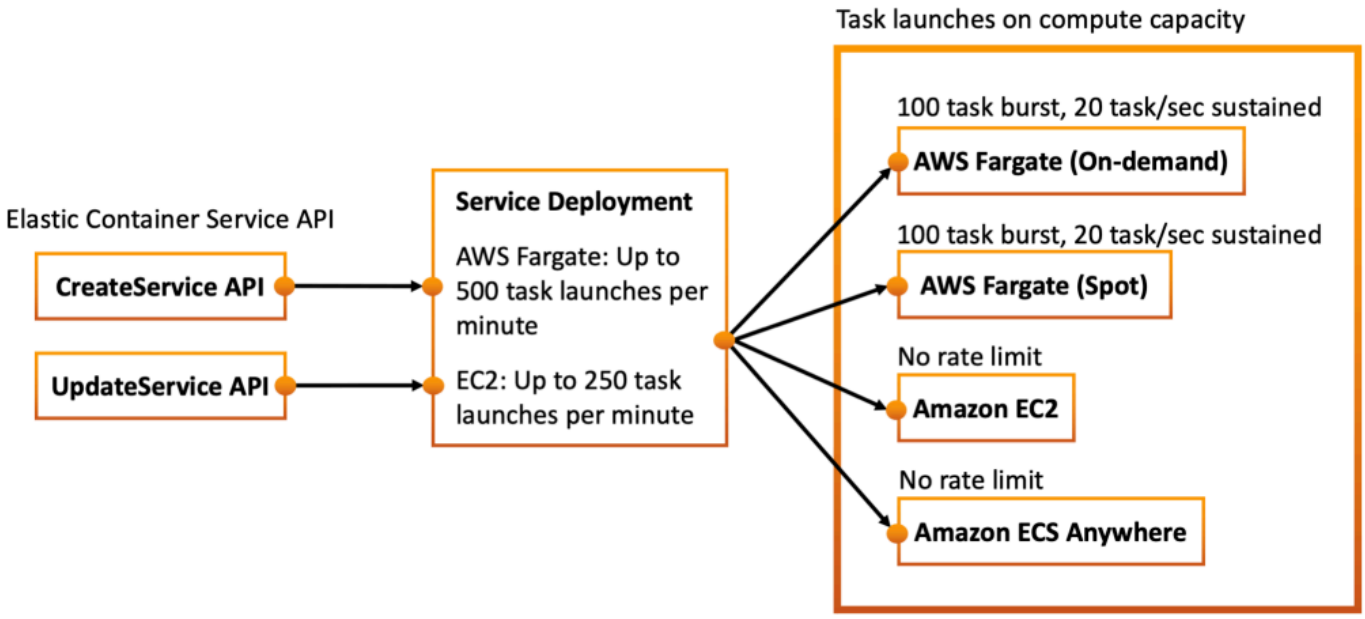
Dans Lambda, l'environnement d'exécution est plus abstrait et l'infrastructure sous-jacente est moins contrôlée. La nature apatride de Lambda signifie que chaque appel de fonction est indépendant, et que tout état ou donnée devant persister entre les invocations doit être géré en externe (par exemple, dans les bases de données ou les services de stockage).

Scaling

Fargate évolue en ajustant le nombre de conteneurs en cours d'exécution en fonction de l'état souhaité défini dans votre service d'orchestration de conteneurs (Amazon ECS). Ce

dimensionnement peut être effectué manuellement ou automatiquement via Amazon EC2 Auto Scaling. Ce billet de blog fournit plus de détails sur la façon de procéder.

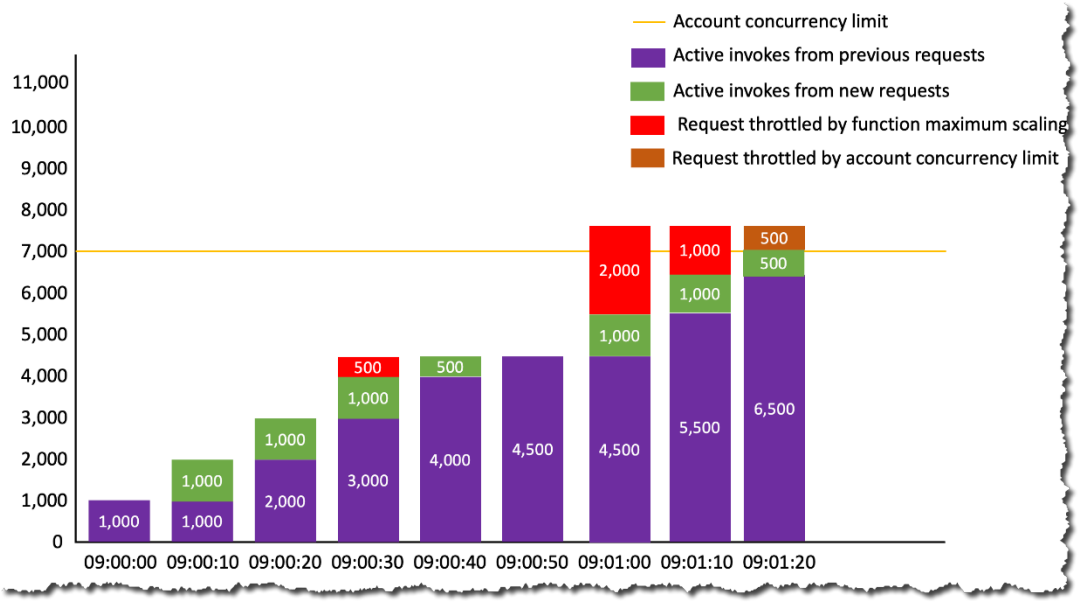
Dans Fargate, chaque conteneur fonctionne dans son environnement isolé, et le dimensionnement implique le lancement de conteneurs supplémentaires ou leur arrêt en fonction de la charge. Le planificateur de services Amazon ECS est capable de lancer jusqu'à 500 tâches en moins d'une minute par service pour le Web et d'autres services de longue durée.



Pour Lambda, la simultanéité est le nombre de requêtes en cours de vol traitées simultanément par votre AWS Lambda fonction. Cela diffère de la simultanéité dans Fargate, où chaque tâche Fargate peut gérer des demandes simultanées tant que des ressources de calcul et de réseau sont disponibles. Pour chaque demande simultanée, Lambda fournit une instance distincte de votre environnement d'exécution. Au fur et à mesure que vos fonctions reçoivent des demandes, Lambda gère automatiquement la mise à l'échelle du nombre d'environnements d'exécution jusqu'à ce que vous atteigniez la limite de simultanéité de votre compte. Par défaut, Lambda fournit à votre compte une limite de simultanéité totale de 1 000 exécutions simultanées pour toutes les fonctions d'un Région AWS, et vous pouvez demander une augmentation du quota si nécessaire.

Pour chaque fonction Lambda dans une région, le taux d'échelonnement de la simultanéité est de 1 000 instances d'exécution toutes les 10 secondes, jusqu'à la simultanéité maximale des comptes. Comme expliqué dans ce blog, si le nombre de demandes sur une période de 10 secondes dépasse 1 000, les demandes supplémentaires seront limitées. Le graphique suivant

montre comment fonctionne le dimensionnement Lambda en supposant une simultanéité de 7 000 comptes.



Cold start and cold-start mitigation

Lambda peut connaître des démarrages à froid, qui se produisent lorsqu'une fonction est invoquée après un certain temps d'inactivité. Lors d'un démarrage à froid, le service Lambda doit initialiser un nouvel environnement d'exécution, notamment en chargeant le runtime, les dépendances et le code de fonction. Ce processus peut introduire de la latence, en particulier pour les langages dont les temps d'initialisation sont plus longs (par exemple, Java ou C#). Les démarrages à froid peuvent avoir un impact sur les performances des applications, en particulier celles qui nécessitent des réponses à faible latence.

Pour atténuer les démarrages à froid dans Lambda, plusieurs stratégies peuvent être utilisées :

- Minimiser la taille des fonctions : la réduction de la taille de votre package de fonctions et de ses dépendances peut réduire le temps nécessaire à l'initialisation.
- Augmenter l'allocation de mémoire : des allocations de mémoire plus élevées augmentent la capacité du processeur, ce qui peut réduire le temps d'initialisation.
- Maintenez les fonctions au chaud : le fait d'invoquer régulièrement vos fonctions Lambda (par exemple, à CloudWatch l'aide d'événements) peut les maintenir actives et réduire le risque de démarrages à froid.
- Lambda SnapStart : utilisez Lambda SnapStart pour les fonctions Java afin de réduire le temps de démarrage.

- Concurrence provisionnée : cette fonctionnalité permet de maintenir un nombre spécifié d'instances de fonction au chaud et prêtes à répondre aux demandes, réduisant ainsi la latence de démarrage à froid. Cependant, cela augmente les coûts car vous payez pour les instances provisionnées même si elles ne traitent pas activement les demandes.

Fargate n'est généralement pas affectée par les démarrages à froid de la même manière que Lambda. Le temps nécessaire pour démarrer une tâche Fargate est directement corrélé au temps nécessaire pour extraire du registre d'images les images de conteneur définies dans la tâche. Fargate prend également en charge le chargement différé d'images de conteneurs indexées avec Seekable OCI (SOCI). Le chargement différé des images de conteneurs avec SOCI réduit le temps nécessaire au lancement des tâches Amazon ECS sur Fargate. Fargate gère des conteneurs qui restent actifs aussi longtemps que nécessaire, ce qui signifie qu'ils sont toujours prêts à traiter les demandes. Toutefois, si vous devez démarrer de nouveaux conteneurs en réponse à des événements de dimensionnement, l'initialisation des conteneurs peut prendre un certain temps, mais cela est généralement moins important par rapport aux démarrages à froid Lambda.

Memory and CPU options

Fargate fournit un contrôle granulaire de la mémoire et des ressources du processeur pour vos applications conteneurisées. Lorsque vous lancez une tâche dans Fargate, vous pouvez spécifier les exigences exactes en termes de processeur et de mémoire en fonction des besoins de votre application. Les allocations du processeur et de la mémoire sont indépendantes, ce qui vous permet de choisir les combinaisons les mieux adaptées à votre charge de travail. Par exemple, vous pouvez sélectionner des valeurs de processeur comprises entre 0,25 V CPUs et 16 V CPUs et une mémoire comprise entre 0,5 Go et 120 Go par conteneur, en fonction de votre configuration.

Cette flexibilité est idéale pour exécuter des applications qui nécessitent des caractéristiques de performance spécifiques, telles que les bases de données gourmandes en mémoire ou les tâches de calcul liées au processeur. Fargate vous permet d'optimiser l'allocation de vos ressources pour équilibrer efficacement les coûts et les performances.

Dans Lambda, la mémoire et le processeur sont liés, le processeur étant automatiquement alloué proportionnellement à la quantité de mémoire sélectionnée. Vous pouvez choisir des allocations de mémoire entre 128 Mo et 10 Go, par incréments de 1 Mo. Le processeur évolue avec la mémoire, jusqu'à 6 vCPU, ce qui signifie que des paramètres de mémoire plus élevés se

traduisent par une augmentation de la puissance du processeur, mais vous n'avez aucun contrôle direct sur l'allocation du processeur elle-même.

Ce modèle est conçu dans un souci de simplicité, permettant aux développeurs d'ajuster rapidement les paramètres de mémoire sans avoir à gérer les configurations du processeur. Toutefois, il peut être moins flexible pour les charges de travail qui nécessitent un équilibre spécifique entre les ressources du processeur et de la mémoire. Le modèle de Lambda convient aux tâches nécessitant une mise à l'échelle directe en fonction des besoins en mémoire, mais il peut ne pas être optimal pour les applications nécessitant des ressources complexes ou très spécifiques.

Networking

Lorsque vous déployez des tâches dans Fargate, elles s'exécutent dans un Amazon VPC (Amazon Virtual Private Cloud), ce qui vous permet de contrôler totalement l'environnement réseau. Cela inclut la configuration des groupes de sécurité, des listes de contrôle d'accès au réseau (ACLs) et des tables de routage. Chaque tâche Fargate dispose de sa propre interface réseau, avec une adresse IP privée dédiée, et peut se voir attribuer une adresse IP publique si nécessaire.

Fargate prend en charge des fonctionnalités réseau avancées telles que l'équilibrage de charge (à l'aide d'Elastic Load AWS Balancing), le peering VPC et l'accès direct à d'autres fonctionnalités au sein du VPC. Services AWS Vous pouvez également l'utiliser AWS PrivateLink pour bénéficier d'une connectivité privée et sécurisée aux services pris en charge Services AWS, sans passer par Internet.

Par défaut, les fonctions Lambda sont exécutées dans un environnement réseau géré sans contrôle direct sur les interfaces réseau ou les adresses IP. Cependant, Lambda peut être rattaché à un VPC géré par le client à l' AWS aide d'Hyperplane, ce qui vous permet de contrôler l'accès aux ressources au sein de votre VPC.

Lorsque les fonctions Lambda sont associées à un VPC géré par le client, elles héritent des groupes de sécurité et des configurations de sous-réseaux du VPC, ce qui leur permet d'interagir en toute sécurité avec d'autres (comme les bases de données RDS) au sein du même VPC. Services AWS

Le service Lambda utilise une plate-forme de virtualisation des fonctions réseau pour fournir des fonctionnalités NAT du VPC Lambda au client. VPCs Cela permet de configurer les interfaces réseau élastiques (ENIs) requises au moment où les fonctions Lambda sont créées ou mises

à jour. Cela permet également ENIs de partager les données de votre compte entre plusieurs environnements d'exécution, ce qui permet à Lambda d'utiliser plus efficacement une ressource réseau limitée lorsque les fonctions évoluent.

Comme il ENIs s'agit d'une ressource épuisable et qu'il existe une limite souple de 250 ENIs par région, vous devez surveiller l'utilisation de l'interface Elastic Network si vous configurez des fonctions Lambda pour un accès VPC. Les fonctions Lambda d'une même zone de disponibilité et d'un même groupe de sécurité peuvent être partagées. ENIs En général, si vous augmentez les limites de simultanéité dans Lambda, vous devez évaluer si vous avez besoin d’une augmentation de l’interface réseau Elastic. Si la limite est atteinte, les appels de fonctions Lambda compatibles VPC sont limités.

Pricing model

La tarification Fargate est basée sur les ressources allouées à vos conteneurs, en particulier le vCPU et la mémoire que vous sélectionnez pour chaque tâche. Vous êtes facturé à la seconde, avec une charge minimale d'une minute, pour le processeur et la mémoire utilisés par vos conteneurs. Les coûts sont directement liés aux ressources consommées par votre application, ce qui signifie que vous payez pour ce que vous fournissez, que l'application traite activement les demandes ou non. Fargate convient parfaitement aux charges de travail prévisibles nécessitant des configurations de ressources spécifiques et peut optimiser les coûts en ajustant les ressources allouées. En outre, des frais supplémentaires peuvent être facturés pour les services connexes, tels que le transfert de données, le stockage et la mise en réseau (par exemple, VPC, Elastic Load Balancing).

Lambda a une structure tarifaire différente qui est axée sur les événements et. pay-per-execution Vous êtes facturé en fonction du nombre de demandes reçues par vos fonctions et de la durée de chaque exécution, mesurée en millisecondes. Lambda prend également en compte la quantité de mémoire que vous allouez à votre fonction, les coûts variant en fonction de la mémoire utilisée et du temps d'exécution. Le modèle de tarification inclut un niveau gratuit, offrant 1 million de requêtes gratuites et 400 000 Go de temps de calcul par mois, ce qui rend Lambda particulièrement rentable pour les charges de travail sporadiques à faible volume.

Le modèle de tarification Lambda est idéal pour les applications dont le trafic est imprévisible ou intense, car vous ne payez que pour les appels de fonctions réels et le temps d'exécution, sans qu'il soit nécessaire de fournir ou de payer pour la capacité inutilisée.

Utilisation

Maintenant que vous avez pris connaissance des critères permettant de choisir entre AWS Fargate et AWS Lambda, vous pouvez sélectionner le service qui répond à vos besoins et utiliser les informations suivantes pour vous aider à commencer à utiliser chacun d'entre eux.

AWS Fargate

- Découvrez comment créer une tâche Linux Amazon ECS pour le type de lancement Fargate

  Commencez à utiliser Amazon ECS AWS Fargate en utilisant le type de lancement Fargate pour vos tâches Linux.

  Explorez le guide
- Découvrez comment créer une tâche Windows Amazon ECS pour le type de lancement Fargate

  Commencez à utiliser Amazon ECS AWS Fargate en utilisant le type de lancement Fargate pour vos tâches Windows.

  Explorez le guide
- Commencer à utiliser Fargate et Amazon EKS

  Ce guide explique comment commencer à exécuter vos pods sur AWS Fargate votre cluster Amazon EKS.

  Explorez le guide
- AWS Fargate tarification

  Utilisez ce guide pour comprendre l'impact AWS Fargate des configurations du vCPU, de la mémoire, du stockage et du système d'exploitation sur les prix.

  Explorez le guide
- AWS Fargate questions fréquemment posées

  Obtenez des réponses aux questions les plus fréquemment posées sur AWS Fargate les fonctionnalités et les meilleures pratiques de mise en œuvre.

  Explorez le guide

AWS Lambda

- Création d'une application de traitement de fichiers sans serveur

  Présentation de step-by-step la configuration et de l'utilisation d'Amazon SNS. Il couvre des sujets tels que la création d'un sujet, l'abonnement de points de terminaison à un sujet, la publication de messages et la configuration des autorisations d'accès.

  Explorez le guide
- Guide pour développeurs sans serveur

  Ce guide vous aide à acquérir une meilleure compréhension conceptuelle du développement d'applications sans serveur et de la manière dont les différentes applications Services AWS s'intègrent pour créer des modèles d'applications qui constituent le cœur de vos applications cloud.

  Explorez le guide
- Terre sans serveur

  Ce site rassemble les dernières informations, blogs, vidéos, codes et ressources d'apprentissage pour AWS Serverless. Apprenez à utiliser et à créer des applications qui s'adaptent automatiquement à une architecture sans serveur peu coûteuse et entièrement gérée.

  Explorez le site
- AWS Lambda tarification

  Utilisez ce guide pour estimer les dépenses et optimiser les coûts en fonction de l'utilisation et de la configuration des fonctions. Il inclut un calculateur de prix pour calculer vos coûts AWS Lambda et ceux de l'architecture en une seule estimation.

  Explorez le guide
- AWS Lambda questions fréquemment posées

  Obtenez des réponses aux questions les plus fréquemment posées sur AWS Lambda les fonctionnalités et les meilleures pratiques de mise en œuvre.

  Explorez le guide

Historique du document pour AWS Fargate ou AWS Lambda ?

Le tableau suivant décrit les modifications importantes apportées à ce guide de décision. Pour recevoir des notifications concernant les mises à jour de ce guide, vous pouvez vous abonner à un flux RSS.

Modification	Description	Date
Première version	Publication initiale du guide de décision.	15 novembre 2024

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.