AWS Guia de decisão

AWS Fargate ou AWS Lambda?

AWS Fargate ou AWS Lambda?: AWS Guia de decisão

Table of Contents

AWS Fargate ou AWS Lambda?

Entenda as diferenças e escolha a mais adequada para você

Finalidade	Para explorar se AWS Fargate ou AWS Lambda atender às suas necessidades de um serviço de computação sem servidor.
Última atualização	15 de novembro de 2024
Serviços cobertos	• AWS Fargate • AWS Lambda

Introdução

Antes de começar a explorar se você escolhe AWS Lambda ou AWS Fargate como seu serviço de computação sem servidor, você provavelmente já considerou a variedade mais ampla de serviços de AWS computação (abordada no guia de decisão sobre como escolher um serviço de AWS computação) e a reduziu a essas duas opções, pois elas fornecem:

- Sobrecarga operacional reduzida: tanto a Lambda quanto o Fargate eliminam o gerenciamento de servidores, reduzindo a necessidade de aplicação de patches, manutenção e planejamento de capacidade.
- Pay-per-use preços: você paga apenas pelos recursos computacionais que realmente usa, potencialmente reduzindo os custos de cargas de trabalho variáveis.
- Implantação mais rápida: normalmente oferece tempos de implantação mais rápidos em comparação com o provisionamento e a configuração de instâncias. EC2
- Alta disponibilidade integrada: ambos os serviços lidam com a redundância da infraestrutura automaticamente.
- Conformidade simplificada: a superfície de ataque reduzida e os recursos de segurança integrados podem facilitar os esforços de conformidade.
- Concentre-se no código: os desenvolvedores podem se concentrar mais em escrever o código do aplicativo do que no gerenciamento da infraestrutura.

Embora o Lambda e o Fargate sejam opções sem servidor, há diferenças significativas entre eles:

AWS Fargateé um mecanismo de computação sem servidor para contêineres, usado principalmente com o Amazon ECS. Ele gerencia automaticamente sua infraestrutura, permitindo que você se concentre na implantação e escalabilidade de aplicativos em contêineres. O Fargate é ideal para aplicativos de longa execução, microsserviços ou processamento em lote, onde você precisa de um controle refinado sobre a alocação de recursos (CPU, memória) e evita o gerenciamento de servidores subjacentes.

AWS Lambdaé um serviço de computação sem servidor que executa automaticamente seu código em resposta a eventos e gerencia os recursos computacionais subjacentes. É mais adequado para aplicativos orientados por eventos, como processamento de arquivos enviados para o Amazon S3, resposta a solicitações HTTP ou execução de tarefas agendadas. O Lambda também é adequado para aplicativos de processamento de streams e processamento de dados devido à sua capacidade de escalar automaticamente em resposta a eventos e lidar com grandes volumes de dados em tempo real. O Lambda pode processar fluxos de dados de fontes como Amazon Kinesis ou Amazon DynamoDB, permitindo transformações, filtragens e análises de dados eficientes e sem servidor, sem gerenciar a infraestrutura. O Lambda foi projetado para tarefas de curta duração (até 15 minutos) e é cobrado com base no número de solicitações e no tempo de execução, tornando-o econômico para cargas de trabalho esporádicas.

Se seu projeto envolve tarefas orientadas por eventos, de curta duração ou cargas de trabalho imprevisíveis, AWS Lambda talvez seja a melhor opção. Se você precisar executar aplicativos em contêineres com necessidades específicas de recursos (ou precisar de processos persistentes), AWS Fargate seria mais apropriado.

A tabela a seguir fornece uma visão mais detalhada de algumas das diferenças entre esses serviços para você começar.

Recurso	AWS Fargate	AWS Lambda
Modelo de execução	Computação sem servidor e baseada em contêineres	Funções sem servidor e orientadas por eventos
Idiomas compatíveis	Qualquer linguagem que possa ser executada em um contêiner	Linguagens suportadas: Node.js, Python, Java, C#, Go, Ruby e. PowerShell Você também pode criar um tempo de execução personalizado

		para implementar uma AWS Lambda função no idioma de sua escolha.
Caso de uso	Aplicativos em contêineres de longa duração	Tarefas de curta duração, orientadas por eventos
Escalabilidade	Dimensionamento automático com base na contagem de tarefas desejada	Escalabilidade automática por solicitação
Arranque a frio	35 segundos a 2 minutos	100 ms a 2 segundos
Limite de tempo de execução	Sem limite rígido	15 minutos no máximo
Alocação de memória	Até 120 GiB	Até 10 GiB
Alocação de CPU	Até 16 vCPUs	Proporcional à memória, até 6 vCPUs
Redes	É executado em VPC, pode usar ENIs	Pode ser executado em uma VPC AWS gerenciada ou conectada a uma VPC gerenciada pelo cliente usando o Hyperplane AWS
Gerenciamento de estados	Os contêineres no Fargate podem manter o estado em todas as solicitações enquanto o contêiner estiver em execução, possibilitando lidar com sessões, armazenar dados em cache ou manter o estado na memória sem precisar de armazenamento externo. O armazenamento externo é recomendado para dados críticos.	Sem estado por design (o estado deve ser gerenciado externamente, por exemplo, Amazon S3, Amazon DynamoDB, Amazon EFS)

Compatibilidade do contêiner	Suporta contêineres	Suporte limitado a contêineres (por meio de implantações de imagens de contêineres)
Orquestração	Integrado com o Amazon ECS	Sem necessidade de orquestração
Modelo de definição de preços	Cobrança por segundo para vCPU e memória usada	Por invocação e duração (GB-segundos)
Limites de simultaneidade	Com base na capacidade do cluster	1000 execuções simultâne as por padrão (podem ser aumentadas)
Invocação conduzida por eventos	Requer configuração adicional	Suporte nativo para várias fontes de AWS eventos
Mitigação de partida a frio	O carregamento lento de imagens com o Seekable OCI pode acelerar o início das tarefas do Fargate	Simultaneidade provisionada disponível
Limite de tamanho do pacote	Sem limite específico (tamanho do contêiner limitado pelo armazenam ento temporário configurado, máximo de 200 GiB)	250 MB descompactados, incluindo camadas, 10 GB para implantações de imagens de contêineres

Diferenças entre Fargate e Lambda

Explore as diferenças entre o Fargate e o Lambda em várias áreas importantes.

Languages supported

Fargate: AWS Fargate é um serviço de orquestração de contêineres, o que significa que ele suporta qualquer linguagem de programação ou ambiente de execução que possa ser empacotado em um contêiner Docker. Essa flexibilidade permite que os desenvolvedores usem praticamente qualquer linguagem, estrutura ou biblioteca que atenda às necessidades

de seus aplicativos. Se você estiver usando Python, Java, Node.js, Go, .NET, Ruby, PHP ou até mesmo linguagens e ambientes personalizados, o Fargate pode executá-los, desde que estejam encapsulados em um contêiner. Esse amplo suporte a idiomas torna o Fargate ideal para executar diversos aplicativos, incluindo sistemas legados, microsserviços em vários idiomas e aplicativos modernos nativos da nuvem.

Lambda: AWS Lambda oferece suporte nativo para um conjunto mais limitado de linguagens em comparação com o Fargate, projetado especificamente para funções orientadas por eventos. A partir de agora, o Lambda oferece suporte oficial aos seguintes idiomas:

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

O Lambda também oferece suporte a tempos de execução personalizados, o que permite que você traga sua própria linguagem ou ambiente de execução, mas isso requer mais configuração e gerenciamento em comparação com o uso das opções com suporte nativo. Se você optar por implantar sua função Lambda a partir de uma imagem de contêiner, poderá escrever sua função no Rust usando uma imagem base AWS somente do sistema operacional e incluindo o cliente de tempo de execução do Rust na sua imagem. Se você estiver usando uma linguagem que não tem um cliente de interface de tempo de execução AWS fornecido, você deve criar sua própria.

Event-driven invocation

O Lambda é inerentemente projetado para computação orientada a eventos. As funções Lambda são acionadas em resposta a uma variedade de eventos, incluindo alterações nos dados, ações do usuário ou tarefas agendadas. Ele se integra perfeitamente a muitos Serviços da AWS, como o Amazon S3 (por exemplo, invocando uma função quando um arquivo é carregado), o DynamoDB (por exemplo, acionando atualizações de dados) e o API Gateway (por exemplo, gerenciando solicitações HTTP). A arquitetura orientada a eventos do Lambda é ideal para aplicativos que precisam responder imediatamente aos eventos sem exigir recursos computacionais persistentes.

O Fargate não é nativamente orientado por eventos, mas com alguma lógica padronizada adicional, ele pode se integrar a fontes de eventos, como Amazon SQS e Kinesis. Embora o Lambda gerencie a maior parte dessa lógica de integração para você, você mesmo precisará implementar essa integração usando o APIs para esses serviços.

Runtime/use cases

O Fargate foi projetado para executar aplicativos em contêineres, fornecendo um ambiente de tempo de execução flexível onde você pode definir as configurações de CPU, memória e rede para seus contêineres. Como o Fargate opera em um modelo baseado em contêiner, ele oferece suporte a processos de longa execução, serviços persistentes e aplicativos com requisitos específicos de tempo de execução. Os contêineres no Fargate podem ser executados indefinidamente, pois não há limite rígido no tempo de execução, o que o torna ideal para aplicativos que precisam estar em execução contínua.

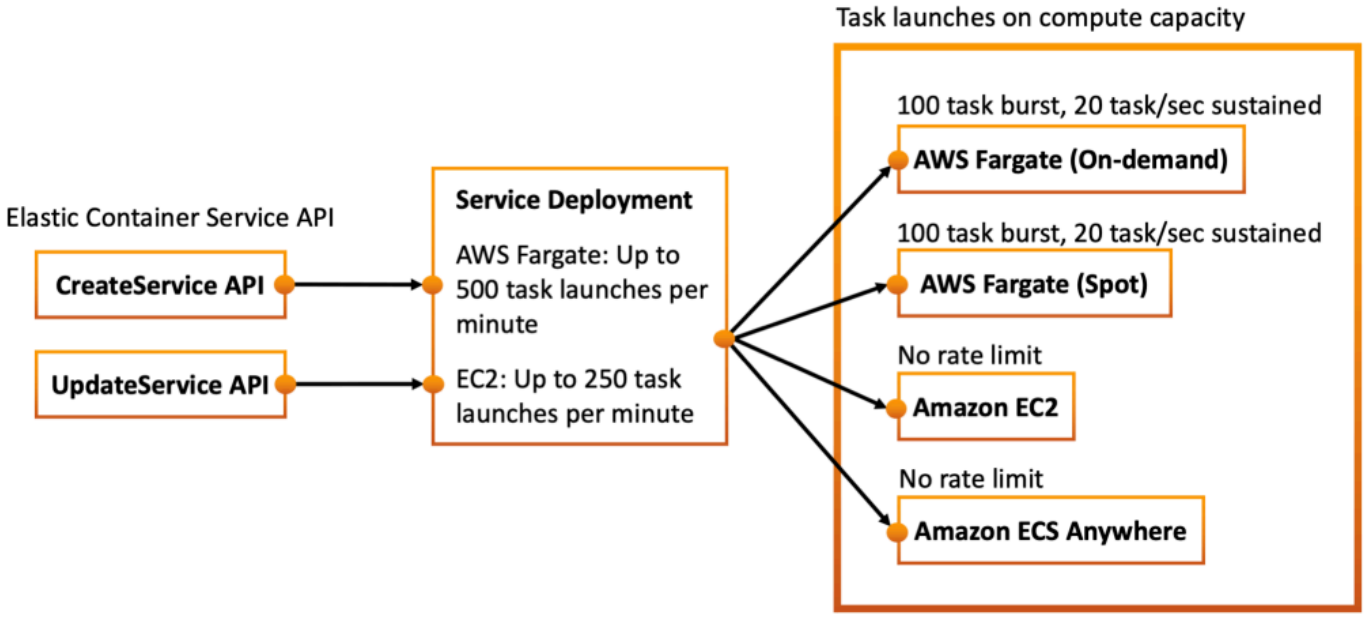
O Lambda, por outro lado, é otimizado para tarefas de curta duração e orientadas por eventos. As funções Lambda são executadas em um ambiente sem estado, onde o tempo máximo de execução é limitado a 15 minutos. Isso torna o Lambda adequado para cenários como processamento de arquivos, streaming de dados em tempo real e tratamento de solicitações HTTP, em que as tarefas são breves e não exigem processos de longa duração.

No Lambda, o ambiente de execução é mais abstrato e há menos controle sobre a infraestrutura subjacente. A natureza sem estado do Lambda significa que cada invocação de função é independente, e qualquer estado ou dado que precise persistir entre as invocações deve ser gerenciado externamente (por exemplo, em bancos de dados ou serviços de armazenamento).

Scaling

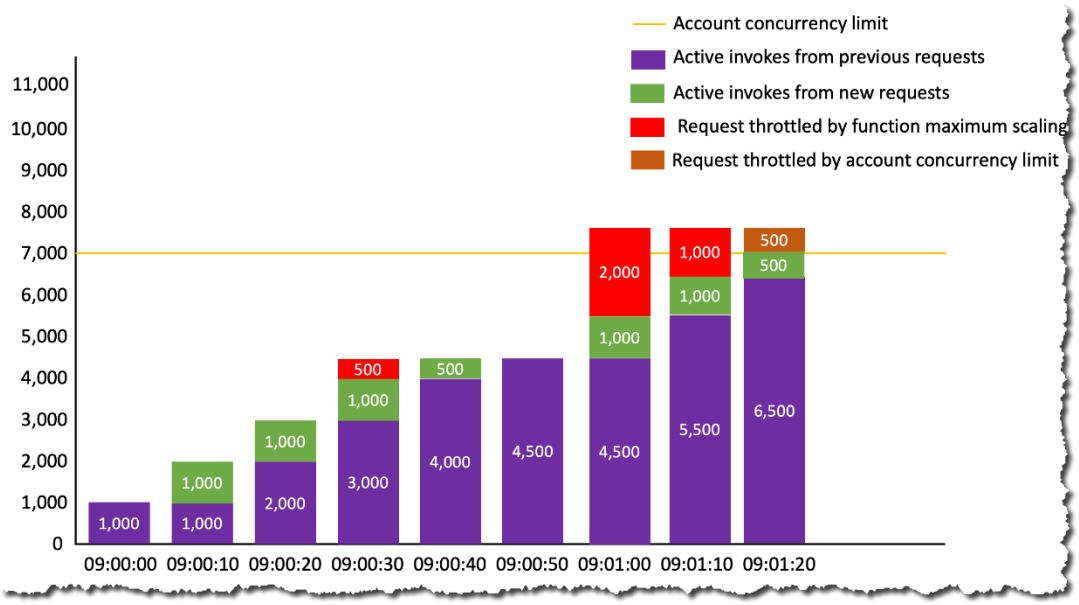
O Fargate escala ajustando o número de contêineres em execução com base no estado desejado definido em seu serviço de orquestração de contêineres (Amazon ECS). Essa escalabilidade pode ser feita manual ou automaticamente por meio do Amazon EC2 Auto Scaling. Esta postagem do blog oferece mais detalhes sobre como.

No Fargate, cada contêiner é executado em seu ambiente isolado, e o escalonamento envolve o lançamento de contêineres adicionais ou a parada deles com base na carga. O agendador de serviços do Amazon ECS é capaz de iniciar até 500 tarefas em menos de um minuto por serviço para a web e outros serviços de longa duração.



Para o Lambda, simultaneidade é o número de solicitações em andamento que sua AWS Lambda função está processando ao mesmo tempo. Isso difere da simultaneidade no Fargate, onde cada tarefa do Fargate pode lidar com solicitações simultâneas, desde que haja recursos de computação e rede disponíveis. Para cada solicitação simultânea, o Lambda provisiona uma instância separada do seu ambiente de execução. À medida que suas funções recebem mais solicitações, o Lambda gerencia a escalabilidade do número de ambientes de execução de forma automática até atingir o limite de simultaneidade da sua conta. Por padrão, o Lambda fornece à sua conta um limite total de simultaneidade de 1.000 execuções simultâneas em todas as funções em um Região da AWS, e você pode solicitar um aumento de cota, se necessário.

Para cada função Lambda em uma região, a taxa de escalabilidade de simultaneidade é de 1.000 instâncias de execução a cada 10 segundos, até a simultaneidade máxima da conta. Conforme explicado neste blog, se o número de solicitações em um período de 10 segundos exceder 1.000, as solicitações adicionais serão limitadas. O gráfico a seguir demonstra como o escalonamento do Lambda funciona assumindo uma simultaneidade de conta de 7000.



Cold start and cold-start mitigation

O Lambda pode experimentar arranques a frio, que ocorrem quando uma função é invocada depois de ficar inativa por algum tempo. Durante uma inicialização a frio, o serviço Lambda precisa inicializar um novo ambiente de execução, incluindo o carregamento do tempo de execução, das dependências e do código da função. Esse processo pode introduzir latência, especialmente para linguagens com tempos de inicialização mais longos (por exemplo, Java ou C#). As partidas a frio podem afetar o desempenho dos aplicativos, especialmente aqueles que exigem respostas de baixa latência.

Para mitigar arranques frios no Lambda, várias estratégias podem ser empregadas:

- Minimize o tamanho da função: reduzir o tamanho do pacote de funções e suas dependências pode diminuir o tempo necessário para a inicialização.
- Aumente a alocação de memória: maiores alocações de memória aumentam a capacidade da CPU, reduzindo potencialmente o tempo de inicialização.
- Mantenha as funções aquecidas: invocar periodicamente suas funções do Lambda (por exemplo, CloudWatch usando Eventos) pode mantê-las ativas e reduzir a probabilidade de arranques a frio.
- Lambda SnapStart: use o Lambda SnapStart para funções Java para reduzir o tempo de inicialização.
- Simultaneidade provisionada: esse recurso mantém um número específico de instâncias de função aquecidas e prontas para atender às solicitações, reduzindo a latência de inicialização

a frio. No entanto, isso aumenta os custos à medida que você paga pelas instâncias provisionadas, mesmo que elas não estejam lidando ativamente com as solicitações.

O Fargate geralmente não é afetado por partidas a frio da mesma forma que o Lambda. O tempo necessário para iniciar uma tarefa do Fargate está diretamente relacionado ao tempo necessário para extrair as imagens do contêiner definidas na tarefa do registro de imagens. O Fargate também suporta o carregamento lento de imagens de contêiner que foram indexadas com o Seekable OCI (SOCI). O carregamento lento de imagens de contêiner com o SOCI reduz o tempo necessário para iniciar tarefas do Amazon ECS no Fargate. O Fargate executa contêineres que permanecem ativos pelo tempo que for necessário, o que significa que eles estão sempre prontos para atender às solicitações. No entanto, se você precisar iniciar novos contêineres em resposta a eventos de escalabilidade, pode haver algum atraso na inicialização dos contêineres, mas isso geralmente é menos significativo em comparação com as partidas a frio do Lambda.

Memory and CPU options

O Fargate fornece controle granular sobre os recursos de memória e CPU para seus aplicativos em contêineres. Ao iniciar uma tarefa no Fargate, você pode especificar os requisitos exatos de CPU e memória com base nas necessidades do seu aplicativo. As alocações de CPU e memória são independentes, permitindo que você escolha as combinações que melhor se adequam à sua carga de trabalho. Por exemplo, você pode selecionar valores de CPU que variam de 0,25 v CPUs a 16 v CPUs e memória de 0,5 GB a 120 GB por contêiner, dependendo da sua configuração.

Essa flexibilidade é ideal para executar aplicativos que exigem características específicas de desempenho, como bancos de dados com uso intenso de memória ou tarefas de computação vinculadas à CPU. O Fargate permite que você otimize sua alocação de recursos para equilibrar custo e desempenho de forma eficaz.

No Lambda, a memória e a CPU são vinculadas, com a CPU alocada automaticamente em proporção à quantidade de memória selecionada. Você pode escolher alocações de memória entre 128 MB e 10 GB, em incrementos de 1 MB. A CPU é dimensionada com a memória, até 6 vCPU, o que significa que configurações de memória mais altas resultam em mais potência da CPU, mas você não tem controle direto sobre a alocação da CPU em si.

Esse modelo foi projetado para ser simples, permitindo que os desenvolvedores ajustem rapidamente as configurações de memória sem precisar gerenciar as configurações da CPU. No entanto, ele pode ser menos flexível para cargas de trabalho que exigem um equilíbrio específico entre recursos de CPU e memória. O modelo do Lambda é adequado para tarefas em que você

deseja escalabilidade direta com base nas necessidades de memória, mas pode não ser ideal para aplicativos com demandas de recursos complexas ou altamente específicas.

Networking

Quando você implanta tarefas no Fargate, elas são executadas em uma Amazon VPC (Amazon Virtual Private Cloud), oferecendo controle total sobre o ambiente de rede. Isso inclui a configuração de grupos de segurança, listas de controle de acesso à rede (ACLs) e tabelas de roteamento. Cada tarefa do Fargate tem sua própria interface de rede, com um endereço IP privado dedicado, e pode ser atribuído a um endereço IP público, se necessário.

O Fargate oferece suporte a recursos avançados de rede, como balanceamento de carga (usando AWS ELB), emparelhamento de VPC e acesso direto a outras pessoas dentro da VPC. Serviços da AWS Você também pode usar AWS PrivateLink para obter conectividade segura e privada com o suporte Serviços da AWS, sem atravessar a Internet.

Por padrão, as funções Lambda são executadas em um ambiente de rede gerenciado sem controle direto sobre interfaces de rede ou endereços IP. No entanto, o Lambda pode ser anexado a uma VPC gerenciada pelo cliente AWS usando o Hyperplane, permitindo que você controle o acesso aos recursos dentro da sua VPC.

Quando as funções Lambda são anexadas a uma VPC gerenciada pelo cliente, elas herdam os grupos de segurança e as configurações de sub-rede da VPC, permitindo que interajam com segurança com outros (como bancos de dados do RDS) dentro da mesma VPC. Serviços da AWS

O serviço Lambda usa uma plataforma de virtualização de funções de rede para fornecer recursos de NAT do Lambda VPC ao cliente. VPCs Isso configura as interfaces de rede elástica necessárias (ENIs) no ponto em que as funções Lambda são criadas ou atualizadas. Ele também permite que sua conta seja compartilhada em vários ambientes ENIs de execução, o que permite que a Lambda faça um uso mais eficiente de um recurso de rede limitado quando as funções se expandem.

Como ENIs são um recurso esgotável e há um limite flexível de 250 ENIs por região, você deve monitorar o uso da interface de elastic network se estiver configurando funções Lambda para acesso à VPC. As funções do Lambda na mesma AZ e no mesmo grupo de segurança podem ser compartilhadas. ENIs Geralmente, se você aumentar os limites de simultaneidade no Lambda, deverá avaliar se precisa de um aumento na interface de rede elástica. Se o limite for atingido, isso fará com que as invocações de funções do Lambda habilitadas para VPC sejam restringidas.

Pricing model

O preço do Fargate é baseado nos recursos alocados aos seus contêineres, especificamente na vCPU e na memória que você seleciona para cada tarefa. Você é cobrado por segundo, com uma carga mínima de um minuto, pela CPU e pela memória que seus contêineres usam. Os custos estão diretamente vinculados aos recursos que seu aplicativo consome, o que significa que você paga pelo que provisiona, independentemente de o aplicativo estar processando ativamente as solicitações. O Fargate é adequado para cargas de trabalho previsíveis nas quais você precisa de configurações de recursos específicas e pode otimizar os custos ajustando os recursos alocados. Além disso, pode haver cobranças adicionais por serviços relacionados, como transferência de dados, armazenamento e rede (por exemplo, VPC, ELB).

O Lambda tem uma estrutura de preços diferente, orientada por eventos e. pay-per-execution Você é cobrado com base no número de solicitações que suas funções recebem e na duração de cada execução, medida em milissegundos. O Lambda também leva em consideração a quantidade de memória que você aloca para sua função, com custos escaláveis com base na memória usada e no tempo de execução. O modelo de preços inclui um nível gratuito, oferecendo 1 milhão de solicitações gratuitas e 400.000 GB por segundo de tempo de computação por mês, o que torna o Lambda particularmente econômico para cargas de trabalho esporádicas de baixo volume.

O modelo de preços Lambda é ideal para aplicativos com padrões de tráfego imprevisíveis ou intermitentes, pois você paga apenas pelas invocações reais da função e pelo tempo de execução, sem a necessidade de provisionar ou pagar pela capacidade ociosa.

Use

Agora que você leu sobre os critérios para escolher entre AWS Fargate e AWS Lambda, você pode selecionar o serviço que atende às suas necessidades e usar as informações a seguir para ajudá-lo a começar a usar cada um deles.

AWS Fargate

- Saiba como criar uma tarefa Linux do Amazon ECS para o tipo de lançamento Fargate

  Comece a usar o Amazon ECS AWS Fargate usando o tipo de lançamento Fargate para suas tarefas Linux.

  Explore o guia

- Saiba como criar uma tarefa Windows do Amazon ECS para o tipo de lançamento Fargate

  Comece a usar o Amazon ECS AWS Fargate usando o tipo de lançamento Fargate para suas tarefas do Windows.

  Explore o guia

- Introdução ao Fargate e ao Amazon EKS

  Este guia descreve como começar a executar seus pods AWS Fargate com seu cluster Amazon EKS.

  Explore o guia

- AWS Fargate preços

  Use este guia para entender como as configurações de vCPU, memória, armazenamento e sistema operacional afetam os preços. AWS Fargate

  Explore o guia

- AWS Fargate perguntas frequentes

  Obtenha respostas para perguntas comuns sobre AWS Fargate recursos e melhores práticas de implementação.

  Explore o guia

AWS Lambda

- Crie um aplicativo de processamento de arquivos sem servidor

  Um step-by-step passo a passo sobre como configurar e usar o Amazon SNS. Ele abrange tópicos como criar um tópico, inscrever endpoints em um tópico, publicar mensagens e configurar permissões de acesso.

  Explore o guia

- Serverless Developer Guide

  Este guia ajuda você a desenvolver uma melhor compreensão conceitual do desenvolvimento de aplicativos sem servidor e de como vários Serviços da AWS se encaixam para criar padrões de aplicativos que formam o núcleo de seus aplicativos em nuvem.

Explore o guia

- Terra sem servidor

  Este site reúne as informações, blogs, vídeos, códigos e recursos de aprendizado mais recentes sobre o AWS Serverless. Aprenda a usar e criar aplicativos que escalam automaticamente em uma arquitetura sem servidor totalmente gerenciada e de baixo custo.

  Explore o site

- AWS Lambda preços

  Use este guia para estimar despesas e otimizar custos com base no uso e na configuração da função. Ele inclui uma calculadora de preços para calcular seu custo AWS Lambda e o custo de arquitetura em uma única estimativa.

  Explore o guia

- AWS Lambda perguntas frequentes

  Obtenha respostas para perguntas comuns sobre AWS Lambda recursos e melhores práticas de implementação.

  Explore o guia

Histórico do documento para AWS Fargate ou AWS Lambda?

A tabela a seguir descreve as mudanças importantes nesse guia de decisão. Para receber notificações sobre atualizações deste guia, você pode assinar um feed RSS.

Alteração	Descrição	Data
Lançamento inicial	Lançamento inicial do guia de decisão.	15 de novembro de 2024

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.