AWS Panduan keputusan

AWS Fargate atau AWS Lambda?

AWS Fargate atau AWS Lambda?: AWS Panduan keputusan

Table of Contents

AWS Fargate atau AWS Lambda?

Pahami perbedaannya dan pilih yang tepat untuk Anda

Tujuan	Untuk mengeksplorasi apakah AWS Fargate atau AWS Lambda memenuhi kebutuhan Anda akan layanan komputasi tanpa server.
Terakhir diperbarui	November 15, 2024
Layanan yang tercakup	• AWS Fargate • AWS Lambda

Pengantar

Sebelum Anda mulai menjelajahi apakah Anda memilih AWS Lambda atau AWS Fargate sebagai layanan komputasi tanpa server Anda, Anda mungkin telah mempertimbangkan rentang layanan AWS komputasi yang lebih luas (tercakup dalam panduan keputusan Memilih layanan AWS komputasi) dan mempersempitnya menjadi dua pilihan ini karena mereka menyediakan:

- Pengurangan overhead operasional: Baik Lambda dan Fargate mengabstraksikan manajemen server jauh, mengurangi kebutuhan untuk patching, pemeliharaan, dan perencanaan kapasitas.
- Pay-per-use harga: Anda hanya membayar sumber daya komputasi yang benar-benar Anda gunakan, berpotensi menurunkan biaya untuk beban kerja variabel.
- Penerapan yang lebih cepat: Biasanya menawarkan waktu penerapan yang lebih cepat dibandingkan dengan penyediaan dan konfigurasi instance. EC2
- Ketersediaan tinggi bawaan: Kedua layanan menangani redundansi infrastruktur secara otomatis.
- Kepatuhan yang disederhanakan: Permukaan serangan yang berkurang dan fitur keamanan bawaan dapat memudahkan upaya kepatuhan.
- Fokus pada kode: Pengembang dapat lebih berkonsentrasi pada penulisan kode aplikasi daripada mengelola infrastruktur.

Sementara Lambda dan Fargate keduanya merupakan opsi tanpa server, ada perbedaan yang signifikan di antara keduanya:

AWS Fargateadalah mesin komputasi tanpa server untuk kontainer, terutama digunakan dengan Amazon ECS. Ini secara otomatis mengelola infrastruktur Anda, memungkinkan Anda untuk fokus pada penerapan dan penskalaan aplikasi kontainer. Fargate sangat ideal untuk aplikasi yang berjalan lama, layanan mikro, atau pemrosesan batch — di mana Anda memerlukan kontrol halus atas alokasi sumber daya (CPU, memori) dan ingin menghindari pengelolaan server yang mendasarinya.

AWS Lambdaadalah layanan komputasi tanpa server yang secara otomatis menjalankan kode Anda sebagai respons terhadap peristiwa, dan mengelola sumber daya komputasi yang mendasarinya. Ini paling cocok untuk aplikasi berbasis peristiwa, seperti memproses file yang diunggah ke Amazon S3, menanggapi permintaan HTTP, atau menjalankan tugas terjadwal. Lambda juga cocok untuk pemrosesan aliran dan aplikasi pemrosesan data karena kemampuannya untuk menskalakan secara otomatis dalam menanggapi peristiwa dan menangani volume data yang tinggi secara real time. Lambda dapat memproses aliran data dari sumber seperti Amazon Kinesis atau Amazon DynamoDB, memungkinkan transformasi data, penyaringan, dan analitik tanpa server yang efisien tanpa mengelola infrastruktur. Lambda dirancang untuk tugas yang berumur pendek (hingga 15 menit) dan ditagih berdasarkan jumlah permintaan dan waktu eksekusi, sehingga biaya yang efektif untuk beban kerja sporadis.

Jika proyek Anda melibatkan tugas berdurasi pendek yang digerakkan oleh peristiwa, atau beban kerja yang tidak dapat diprediksi, AWS Lambda mungkin akan lebih cocok. Jika Anda perlu menjalankan aplikasi kontainer dengan kebutuhan sumber daya tertentu (atau Anda memerlukan proses persisten), AWS Fargate akan lebih tepat.

Tabel berikut memberikan tampilan yang lebih rinci pada beberapa perbedaan antara layanan ini untuk membantu Anda memulai.

Fitur	AWS Fargate	AWS Lambda
Model eksekusi	Komputasi tanpa server berbasis kontainer	Fungsi tanpa server yang digerakkan oleh peristiwa
Bahasa yang didukung	Bahasa apa pun yang dapat dijalankan dalam wadah	Bahasa yang didukung: Node.js, Python, Java, C #, Go, Ruby, dan. PowerShell Anda juga dapat membangun runtime kustom untuk mengimplementasikan AWS

		Lambda fungsi dalam bahasa pilihan Anda.
Kasus penggunaan	Aplikasi kontainer yang berjalan lama	Durasi pendek, tugas yang digerakkan oleh peristiwa
Penskalaan	Penskalaan otomatis berdasarkan jumlah tugas yang diinginkan	Penskalaan otomatis per permintaan
Awal yang dingin	35 detik hingga 2 menit	100 ms hingga 2 detik
Batas waktu eksekusi	Tidak ada batas keras	Maksimal 15 menit
Alokasi memori	Hingga 120 GiB	Hingga 10 GiB
Alokasi CPU	Hingga 16 vCPU	Proporsional dengan memori, hingga 6 vCPU
Jaringan	Berjalan di VPC, bisa digunakan ENIs	Dapat berjalan di VPC AWS terkelola atau dilampirkan ke VPC yang dikelola pelanggan menggunakan Hyperplane AWS
Manajemen Negara	Kontainer di Fargate dapat mempertahankan status di seluruh permintaan selama penampung berjalan, sehingga memungkinkan untuk menangani sesi, menyimpan data cache, atau menyimpan status dalam memori tanpa memerluka n penyimpanan eksternal . Penyimpanan eksternal direkomendasikan untuk data penting.	Stateless menurut desain (status harus dikelola secara eksternal, misalnya, Amazon S3, Amazon DynamoDB, Amazon EFS)

Dukungan kontainer	Mendukung kontainer	Dukungan kontainer terbatas (melalui penerapan gambar kontainer)
Orkestrasi	Terintegrasi dengan Amazon ECS	Tidak diperlukan orkestrasi
Model penentuan harga	Penagihan per detik untuk vCPU dan memori yang digunakan	Per pemanggilan dan durasi (GB-detik)
Batas konkurensi	Berdasarkan kapasitas cluster	1000 eksekusi bersamaan secara default (dapat ditingkat kan)
Doa yang digerakkan oleh peristiwa	Membutuhkan pengaturan tambahan	Dukungan asli untuk berbagai sumber AWS acara
Mitigasi awal dingin	Lazy loading gambar dengan Seekable OCI dapat mempercepat memulai tugas Fargate	Konkurensi yang disediakan tersedia
Batas ukuran paket	Tidak ada batas spesifik (ukuran wadah dibatasi oleh penyimpanan sementara yang dikonfigurasi, maksimum 200 GiB)	250 MB dibuka ritsleting, termasuk lapisan, 10GB untuk penerapan gambar kontainer

Perbedaan antara Fargate dan Lambda

Jelajahi perbedaan antara Fargate dan Lambda di sejumlah area utama.

Languages supported

Fargate: AWS Fargate adalah layanan orkestrasi kontainer, yang berarti mendukung bahasa pemrograman atau lingkungan runtime apa pun yang dapat dikemas ke dalam wadah Docker. Fleksibilitas ini memungkinkan pengembang untuk menggunakan hampir semua bahasa,

kerangka kerja, atau perpustakaan yang sesuai dengan kebutuhan aplikasi mereka. Apakah Anda menggunakan Python, Java, Node.js, Go, .NET, Ruby, PHP, atau bahkan bahasa dan lingkungan khusus, Fargate dapat menjalankannya selama mereka dikemas dalam wadah. Dukungan bahasa yang luas ini membuat Fargate ideal untuk menjalankan beragam aplikasi, termasuk sistem lama, layanan mikro multi-bahasa, dan aplikasi cloud-native modern.

Lambda: AWS Lambda menawarkan dukungan asli untuk serangkaian bahasa yang lebih terbatas dibandingkan dengan Fargate, yang dirancang khusus untuk fungsi yang digerakkan oleh peristiwa. Sampai sekarang, Lambda secara resmi mendukung bahasa-bahasa berikut:

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

Lambda juga mendukung runtime kustom, yang memungkinkan Anda membawa bahasa atau lingkungan runtime Anda sendiri, tetapi ini membutuhkan lebih banyak pengaturan dan manajemen dibandingkan dengan menggunakan opsi yang didukung secara native. Jika Anda memilih untuk menerapkan fungsi Lambda Anda dari gambar kontainer, Anda dapat menulis fungsi Anda di Rust dengan menggunakan AWS gambar dasar khusus OS dan termasuk klien runtime Rust dalam gambar Anda. Jika Anda menggunakan bahasa yang tidak memiliki klien antarmuka runtime AWS yang disediakan, Anda harus membuatnya sendiri.

Event-driven invocation

Lambda secara inheren dirancang untuk komputasi berbasis peristiwa. Fungsi Lambda dipicu sebagai respons terhadap berbagai peristiwa, termasuk perubahan data, tindakan pengguna, atau tugas terjadwal. Ini terintegrasi mulus dengan banyak Layanan AWS, seperti Amazon S3 (misalnya, menjalankan fungsi saat file diunggah), DynamoDB (misalnya, memicu pembaruan data), dan API Gateway (misalnya, menangani permintaan HTTP). Arsitektur berbasis peristiwa Lambda sangat ideal untuk aplikasi yang perlu segera merespons peristiwa tanpa memerlukan sumber daya komputasi yang persisten.

Fargate tidak digerakkan oleh peristiwa asli, tetapi dengan beberapa logika boilerplate tambahan, ia dapat berintegrasi dengan sumber peristiwa seperti Amazon SQS dan Kinesis. Sementara Lambda menangani sebagian besar logika integrasi ini untuk Anda, Anda harus menerapkan integrasi ini sendiri menggunakan APIs untuk layanan ini.

Runtime/use cases

Fargate dirancang untuk menjalankan aplikasi kontainer, menyediakan lingkungan runtime yang fleksibel di mana Anda dapat menentukan pengaturan CPU, memori, dan jaringan untuk kontainer Anda. Karena Fargate beroperasi pada model berbasis container, ia mendukung proses yang berjalan lama, layanan persisten, dan aplikasi dengan persyaratan runtime tertentu. Kontainer di Fargate dapat berjalan tanpa batas waktu, karena tidak ada batasan keras pada waktu eksekusi, sehingga ideal untuk aplikasi yang perlu aktif dan berjalan terus menerus.

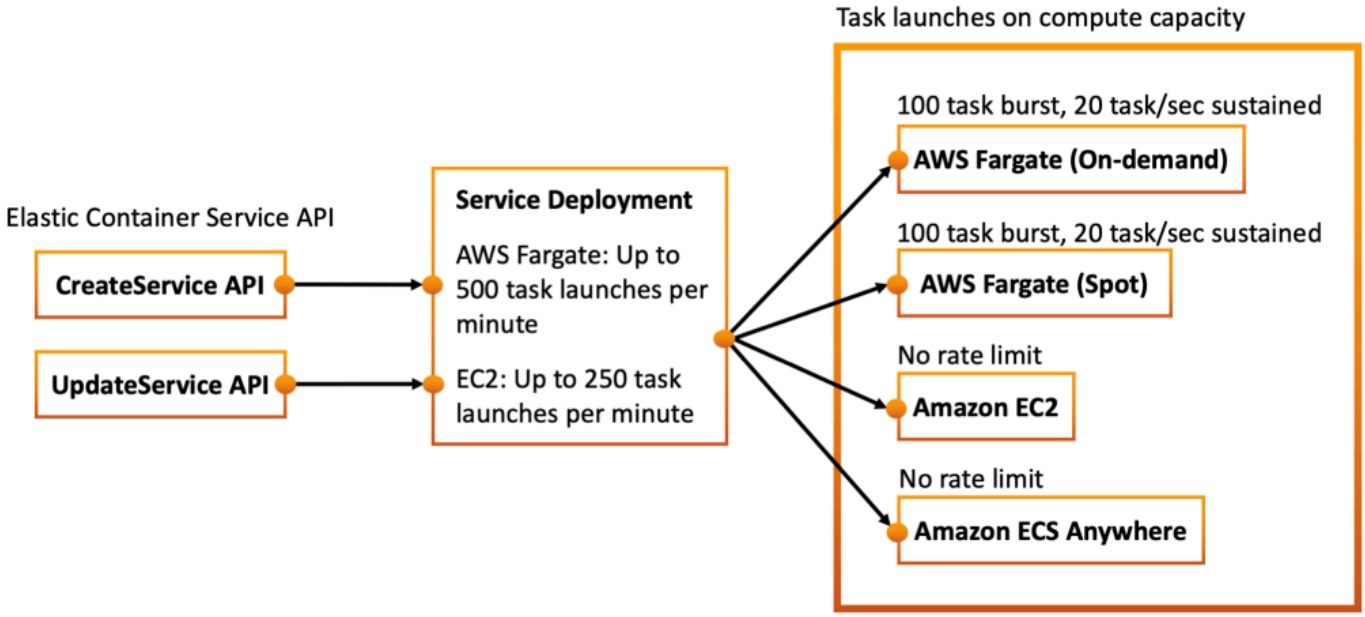
Lambda, di sisi lain, dioptimalkan untuk tugas-tugas yang berumur pendek dan didorong oleh peristiwa. Fungsi Lambda dijalankan di lingkungan stateless di mana waktu eksekusi maksimum dibatasi pada 15 menit. Hal ini membuat Lambda sangat cocok untuk skenario seperti pemrosesan file, streaming data real-time, dan penanganan permintaan HTTP, di mana tugasnya singkat dan tidak memerlukan proses yang berjalan lama.

Di Lambda, lingkungan runtime lebih abstrak, dan ada sedikit kontrol atas infrastruktur yang mendasarinya. Sifat stateless Lambda berarti bahwa setiap pemanggilan fungsi bersifat independen, dan setiap status atau data yang perlu bertahan di antara pemanggilan harus dikelola secara eksternal (misalnya, dalam database atau layanan penyimpanan).

Scaling

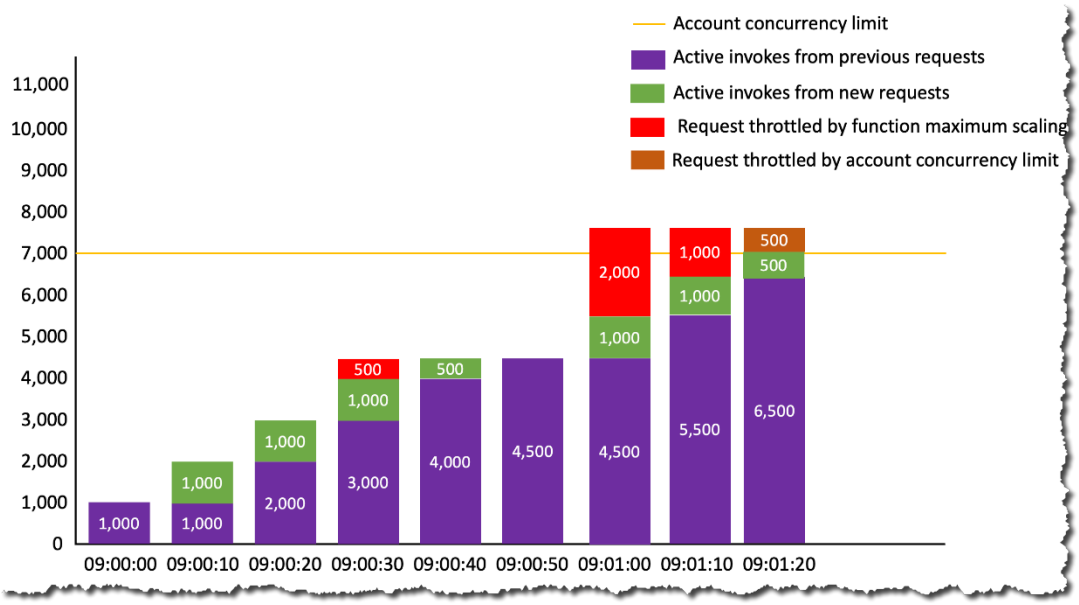
Fargate menskalakan dengan menyesuaikan jumlah kontainer yang sedang berjalan berdasarkan status yang diinginkan yang ditentukan dalam layanan orkestrasi kontainer (Amazon ECS). Penskalaan ini dapat dilakukan secara manual atau otomatis melalui Amazon EC2 Auto Scaling. Posting blog ini menawarkan rincian lebih lanjut tentang caranya.

Di Fargate, setiap kontainer berjalan di lingkungan yang terisolasi, dan penskalaan melibatkan peluncuran kontainer tambahan atau menghentikannya berdasarkan beban. Penjadwal layanan Amazon ECS dapat meluncurkan hingga 500 tugas dalam waktu kurang dari satu menit per layanan untuk web dan layanan jangka panjang lainnya.



Untuk Lambda, konkurensi adalah jumlah permintaan dalam penerbangan yang ditangani oleh AWS Lambda fungsi Anda secara bersamaan. Ini berbeda dari konkurensi di Fargate, di mana setiap tugas Fargate dapat menangani permintaan bersamaan selama ada sumber daya komputasi dan jaringan yang tersedia. Untuk setiap permintaan bersamaan, Lambda menyediakan instance terpisah dari lingkungan eksekusi Anda. Saat fungsi Anda menerima lebih banyak permintaan, Lambda secara otomatis menangani penskalaan jumlah lingkungan eksekusi hingga Anda mencapai batas konkurensi akun Anda. Secara default, Lambda menyediakan akun Anda dengan batas konkurensi total 1.000 eksekusi bersamaan di semua fungsi dalam satu Wilayah AWS, dan Anda dapat meminta peningkatan kuota jika diperlukan.

Untuk setiap fungsi Lambda di Wilayah, tingkat penskalaan konkurensi adalah 1.000 instance eksekusi setiap 10 detik, hingga konkurensi akun maksimum. Seperti yang dijelaskan di blog ini, jika jumlah permintaan dalam periode 10 detik melebihi 1.000, permintaan tambahan akan dibatasi. Grafik berikut menunjukkan cara kerja penskalaan Lambda dengan asumsi konkurensi akun 7000.



Cold start and cold-start mitigation

Lambda dapat mengalami start dingin, yang terjadi ketika suatu fungsi dipanggil setelah menganggur selama beberapa waktu. Selama awal yang dingin, layanan Lambda perlu menginisialisasi lingkungan eksekusi baru, termasuk memuat runtime, dependensi, dan kode fungsi. Proses ini dapat memperkenalkan latensi, terutama untuk bahasa dengan waktu inisialisasi yang lebih lama (misalnya, Java, atau C #). Cold start dapat memengaruhi kinerja aplikasi, terutama yang membutuhkan respons latensi rendah.

Untuk mengurangi awal dingin di Lambda, beberapa strategi dapat digunakan:

- Minimalkan ukuran fungsi: Mengurangi ukuran paket fungsi Anda dan dependensinya dapat mengurangi waktu yang dibutuhkan untuk inisialisasi.
- Meningkatkan alokasi memori: Alokasi memori yang lebih tinggi meningkatkan kapasitas CPU, berpotensi mengurangi waktu inisialisasi.
- Menjaga fungsi tetap hangat: Memanggil fungsi Lambda Anda secara berkala (misalnya, CloudWatch menggunakan Acara) dapat membuatnya tetap aktif dan mengurangi kemungkinan start dingin.
- Lambda SnapStart: Gunakan Lambda SnapStart untuk fungsi Java untuk mengurangi waktu startup.
- Konkurensi yang disediakan: Fitur ini membuat sejumlah instance fungsi tetap hangat dan siap untuk melayani permintaan, mengurangi latensi start dingin. Namun, ini meningkatkan biaya karena Anda membayar untuk instance yang disediakan meskipun mereka tidak secara aktif menangani permintaan.

Fargate umumnya tidak terpengaruh oleh start dingin dengan cara yang sama seperti Lambda. Waktu yang diperlukan untuk memulai tugas Fargate secara langsung berkorelasi dengan waktu yang diperlukan untuk menarik gambar kontainer yang ditentukan dalam tugas dari registri gambar. Fargate juga mendukung pemuatan lambat gambar kontainer yang telah diindeks dengan Seekable OCI (SOCI). Lazy loading gambar kontainer dengan SOCI mengurangi waktu yang dibutuhkan untuk meluncurkan tugas Amazon ECS di Fargate. Fargate menjalankan kontainer yang tetap aktif selama diperlukan, artinya mereka selalu siap menangani permintaan. Namun, jika Anda perlu memulai kontainer baru sebagai respons terhadap peristiwa penskalaan, mungkin ada beberapa penundaan saat kontainer diinisialisasi, tetapi ini biasanya kurang signifikan dibandingkan dengan Lambda cold start.

Memory and CPU options

Fargate menyediakan kontrol granular atas memori dan sumber daya CPU untuk aplikasi kontainer Anda. Saat Anda meluncurkan tugas di Fargate, Anda dapat menentukan persyaratan CPU dan memori yang tepat berdasarkan kebutuhan aplikasi Anda. Alokasi CPU dan memori bersifat independen, memungkinkan Anda memilih kombinasi yang paling sesuai dengan beban kerja Anda. Misalnya, Anda dapat memilih nilai CPU mulai dari 0,25 v CPUs hingga 16 v CPUs dan memori dari 0,5 GB hingga 120 GB per kontainer, tergantung pada konfigurasi Anda.

Fleksibilitas ini sangat ideal untuk menjalankan aplikasi yang memerlukan karakteristik kinerja tertentu, seperti database intensif memori atau tugas komputasi terikat CPU. Fargate memungkinkan Anda mengoptimalkan alokasi sumber daya untuk menyeimbangkan biaya dan kinerja secara efektif.

Di Lambda, memori dan CPU ditautkan, dengan CPU secara otomatis dialokasikan secara proporsional dengan jumlah memori yang Anda pilih. Anda dapat memilih alokasi memori antara 128 MB dan 10 GB, dengan peningkatan 1 MB. Skala CPU dengan memori, hingga 6 vCPU, yang berarti pengaturan memori yang lebih tinggi menghasilkan lebih banyak daya CPU, tetapi Anda tidak memiliki kontrol langsung atas alokasi CPU itu sendiri.

Model ini dirancang untuk kesederhanaan, memungkinkan pengembang untuk dengan cepat menyesuaikan pengaturan memori tanpa perlu mengelola konfigurasi CPU. Namun, mungkin kurang fleksibel untuk beban kerja yang membutuhkan keseimbangan tertentu antara CPU dan sumber daya memori. Model Lambda cocok untuk tugas-tugas di mana Anda ingin penskalaan langsung berdasarkan kebutuhan memori, tetapi mungkin tidak optimal untuk aplikasi dengan permintaan sumber daya yang kompleks atau sangat spesifik.

Networking

Saat Anda menerapkan tugas di Fargate, tugas tersebut berjalan di Amazon VPC (Amazon Virtual Private Cloud), memberi Anda kendali penuh atas lingkungan jaringan. Ini termasuk mengonfigurasi grup keamanan, daftar kontrol akses jaringan (ACLs), dan tabel perutean. Setiap tugas Fargate mendapatkan antarmuka jaringannya sendiri, dengan alamat IP pribadi khusus, dan dapat diberikan alamat IP publik jika diperlukan.

Fargate mendukung fitur-fitur jaringan canggih seperti load balancing (menggunakan Elastic Load AWS Balancing), VPC peering, dan akses langsung ke yang lain di dalam VPC. Layanan AWS Anda juga dapat menggunakan konektivitas pribadi yang aman AWS PrivateLink untuk didukung Layanan AWS, tanpa melintasi internet.

Secara default, fungsi Lambda dijalankan di lingkungan jaringan yang dikelola tanpa kontrol langsung atas antarmuka jaringan atau alamat IP. Namun, Lambda dapat dilampirkan ke VPC yang dikelola pelanggan AWS menggunakan Hyperplane, memungkinkan Anda untuk mengontrol akses ke sumber daya di dalam VPC Anda.

Ketika fungsi Lambda dilampirkan ke VPC yang dikelola pelanggan, mereka mewarisi grup keamanan VPC dan konfigurasi subnet, memungkinkan mereka untuk berinteraksi secara aman dengan yang lain (seperti database RDS) dalam VPC yang sama. Layanan AWS

Layanan Lambda menggunakan platform Virtualisasi Fungsi Jaringan untuk menyediakan kemampuan NAT dari VPC Lambda kepada pelanggan. VPCs Ini mengkonfigurasi antarmuka jaringan elastis yang diperlukan (ENIs) pada titik di mana fungsi Lambda dibuat atau diperbarui. Ini juga memungkinkan ENIs dari akun Anda untuk dibagikan di beberapa lingkungan eksekusi, yang memungkinkan Lambda membuat penggunaan sumber daya jaringan terbatas secara lebih efisien saat skala fungsi.

Karena ENIs merupakan sumber daya yang dapat habis dan ada batas lunak 250 ENIs per Wilayah, Anda harus memantau penggunaan elastic network interface jika Anda mengonfigurasi fungsi Lambda untuk akses VPC. Fungsi Lambda di AZ yang sama dan grup keamanan yang sama dapat dibagikan. ENIs Umumnya, jika Anda meningkatkan batas konkurensi di Lambda, Anda harus mengevaluasi apakah Anda memerlukan peningkatan antarmuka network elastis. Jika batas tercapai, ini menyebabkan pemanggilan fungsi Lambda berkemampuan VPC dibatasi.

Pricing model

Harga Fargate didasarkan pada sumber daya yang dialokasikan ke wadah Anda, khususnya vCPU dan memori yang Anda pilih untuk setiap tugas. Anda ditagih per detik, dengan biaya minimum satu menit, untuk CPU dan memori yang digunakan kontainer Anda. Biaya terkait

langsung dengan sumber daya yang dikonsumsi aplikasi Anda, artinya Anda membayar untuk apa yang Anda berikan, terlepas dari apakah aplikasi tersebut secara aktif memproses permintaan. Fargate sangat cocok untuk beban kerja yang dapat diprediksi di mana Anda memerlukan konfigurasi sumber daya tertentu dan dapat mengoptimalkan biaya dengan menyesuaikan sumber daya yang dialokasikan. Selain itu, mungkin ada biaya tambahan untuk layanan terkait, seperti transfer data, penyimpanan, dan jaringan (misalnya, VPC, Elastic Load Balancing).

Lambda memiliki struktur harga yang berbeda yang didorong oleh peristiwa dan. pay-per-execution Anda dikenakan biaya berdasarkan jumlah permintaan yang diterima fungsi Anda dan durasi setiap eksekusi, diukur dalam milidetik. Lambda juga memperhitungkan jumlah memori yang Anda alokasikan ke fungsi Anda, dengan penskalaan biaya berdasarkan memori yang digunakan dan waktu eksekusi. Model penetapan harga mencakup tingkat gratis, menawarkan 1 juta permintaan gratis dan 400.000 GB-detik waktu komputasi per bulan, yang membuat Lambda sangat hemat biaya untuk volume rendah, beban kerja sporadis.

Model penetapan harga Lambda sangat ideal untuk aplikasi dengan pola lalu lintas yang tidak dapat diprediksi atau meledak, karena Anda hanya membayar untuk pemanggilan fungsi aktual dan waktu eksekusi, tanpa perlu menyediakan atau membayar kapasitas idle.

Gunakan

Sekarang setelah Anda membaca tentang kriteria untuk memilih antara AWS Fargate dan AWS Lambda, Anda dapat memilih layanan yang memenuhi kebutuhan Anda, dan menggunakan informasi berikut untuk membantu Anda mulai menggunakan masing-masing.

AWS Fargate

- Pelajari cara membuat tugas Amazon ECS Linux untuk jenis peluncuran Fargate

  Mulai menggunakan Amazon ECS AWS Fargate dengan menggunakan jenis peluncuran Fargate untuk tugas Linux Anda.

  Jelajahi panduannya
- Pelajari cara membuat tugas Amazon ECS Windows untuk jenis peluncuran Fargate

  Mulailah menggunakan Amazon ECS AWS Fargate dengan menggunakan jenis peluncuran Fargate untuk tugas Windows Anda.

Jelajahi panduannya

- Memulai dengan Fargate dan Amazon EKS

  Panduan ini menjelaskan cara memulai menjalankan pod AWS Fargate dengan klaster Amazon EKS Anda.

  Jelajahi panduannya

- AWS Fargate harga

  Gunakan panduan ini untuk memahami bagaimana vCPU, memori, penyimpanan, dan konfigurasi sistem operasi memengaruhi harga. AWS Fargate

  Jelajahi panduannya

- AWS Fargate pertanyaan yang sering diajukan

  Dapatkan jawaban atas pertanyaan umum tentang AWS Fargate kemampuan, dan praktik terbaik untuk implementasi.

  Jelajahi panduannya

AWS Lambda

- Membuat aplikasi pemrosesan file tanpa server

  step-by-stepPanduan pengaturan dan penggunaan Amazon SNS. Ini mencakup topik seperti membuat topik, berlangganan titik akhir ke topik, menerbitkan pesan, dan mengonfigurasi izin akses.

  Jelajahi panduannya

- Panduan Developer untuk Nirserver

  Panduan ini membantu Anda mengembangkan pemahaman konseptual yang lebih baik tentang pengembangan aplikasi tanpa server, dan bagaimana berbagai Layanan AWS kesesuaian untuk membuat pola aplikasi yang membentuk inti dari aplikasi cloud Anda.

  Jelajahi panduannya

- Tanah Tanpa Server

Situs ini menyatukan informasi terbaru, blog, video, kode, dan sumber pembelajaran untuk Tanpa AWS Server. Pelajari cara menggunakan dan membangun aplikasi yang menskalakan secara otomatis pada arsitektur tanpa server berbiaya rendah dan terkelola sepenuhnya.

Jelajahi situs

- AWS Lambda harga

  Gunakan panduan ini untuk memperkirakan pengeluaran dan mengoptimalkan biaya berdasarkan penggunaan dan konfigurasi fungsi. Ini termasuk kalkulator harga untuk menghitung biaya arsitektur AWS Lambda dan Anda dalam satu perkiraan.

  Jelajahi panduannya

- AWS Lambda pertanyaan yang sering diajukan

  Dapatkan jawaban atas pertanyaan umum tentang AWS Lambda kemampuan, dan praktik terbaik untuk implementasi.

  Jelajahi panduannya

Riwayat dokumen untuk AWS Fargate atau AWS Lambda?

Tabel berikut menjelaskan perubahan penting pada panduan keputusan ini. Untuk pemberitahuan tentang pembaruan panduan ini, Anda dapat berlangganan umpan RSS.

Perubahan	Deskripsi	Tanggal
Rilis awal	Rilis awal panduan keputusan.	November 15, 2024

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.