aws

Panduan Developer

Amazon Elastic Container Service

Amazon Elastic Container Service: Panduan Developer

Table of Contents

Apa itu Layanan Kontainer Elastis Amazon?

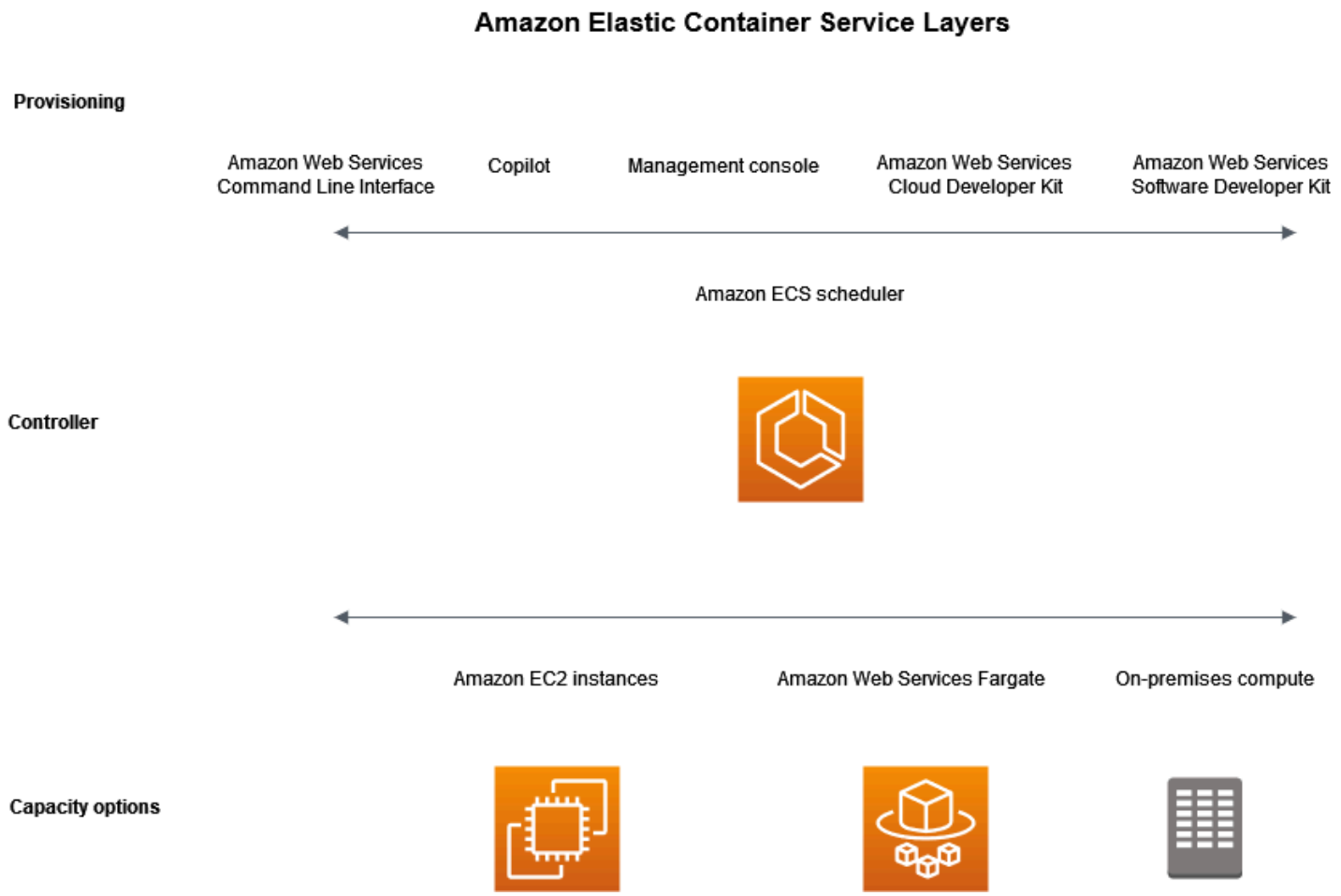
Amazon Elastic Container Service (Amazon ECS) adalah layanan orkestrasi kontainer terkelola penuh yang membantu Anda menerapkan, mengelola, dan menskalakan aplikasi kontainer dengan mudah. Sebagai layanan yang dikelola sepenuhnya, Amazon ECS hadir dengan AWS konfigurasi dan praktik terbaik operasional bawaan. Ini terintegrasi dengan alat kedua AWS dan pihak ketiga, seperti Amazon Elastic Container Registry dan Docker. Integrasi ini memudahkan tim untuk fokus membangun aplikasi, bukan lingkungan. Anda dapat menjalankan dan menskalakan beban kerja kontainer Anda Wilayah AWS di cloud, dan lokal, tanpa kerumitan mengelola bidang kontrol.

Terminologi dan komponen Amazon ECS

Ada tiga lapisan di Amazon ECS:

- Kapasitas - Infrastruktur tempat kontainer Anda berjalan
- Controller - Menyebarkan dan mengelola aplikasi Anda yang berjalan pada kontainer
- Penyediaan - Alat yang dapat Anda gunakan untuk berinteraksi dengan penjadwal untuk menyebarkan dan mengelola aplikasi dan wadah Anda

Diagram berikut menunjukkan lapisan Amazon ECS.



Kapasitas Amazon ECS

Kapasitas Amazon ECS adalah infrastruktur tempat kontainer Anda berjalan. Berikut ini adalah ikhtisar opsi kapasitas:

- Instans Amazon EC2 di cloud AWS

  Anda memilih jenis instans, jumlah instance, dan mengelola kapasitas.
- Tanpa server (AWS Fargate (Fargate)) di awan AWS

  Fargate adalah mesin komputasi tanpa server. pay-as-you-go Dengan Fargate Anda tidak perlu mengelola server, menangani perencanaan kapasitas, atau mengisolasi beban kerja kontainer untuk keamanan.
- Mesin virtual lokal (VM) atau server

  Amazon ECS Anywhere menyediakan dukungan untuk mendaftarkan instans eksternal seperti server lokal atau mesin virtual (VM), ke kluster Amazon ECS Anda.

Kapasitas dapat ditemukan di salah satu AWS sumber daya berikut:

- Zona Ketersediaan
- Zona Lokal
- Wavelength Zones
- Wilayah AWS
- AWS Outposts

Pengontrol Amazon ECS

Penjadwal Amazon ECS adalah perangkat lunak yang mengelola aplikasi Anda.

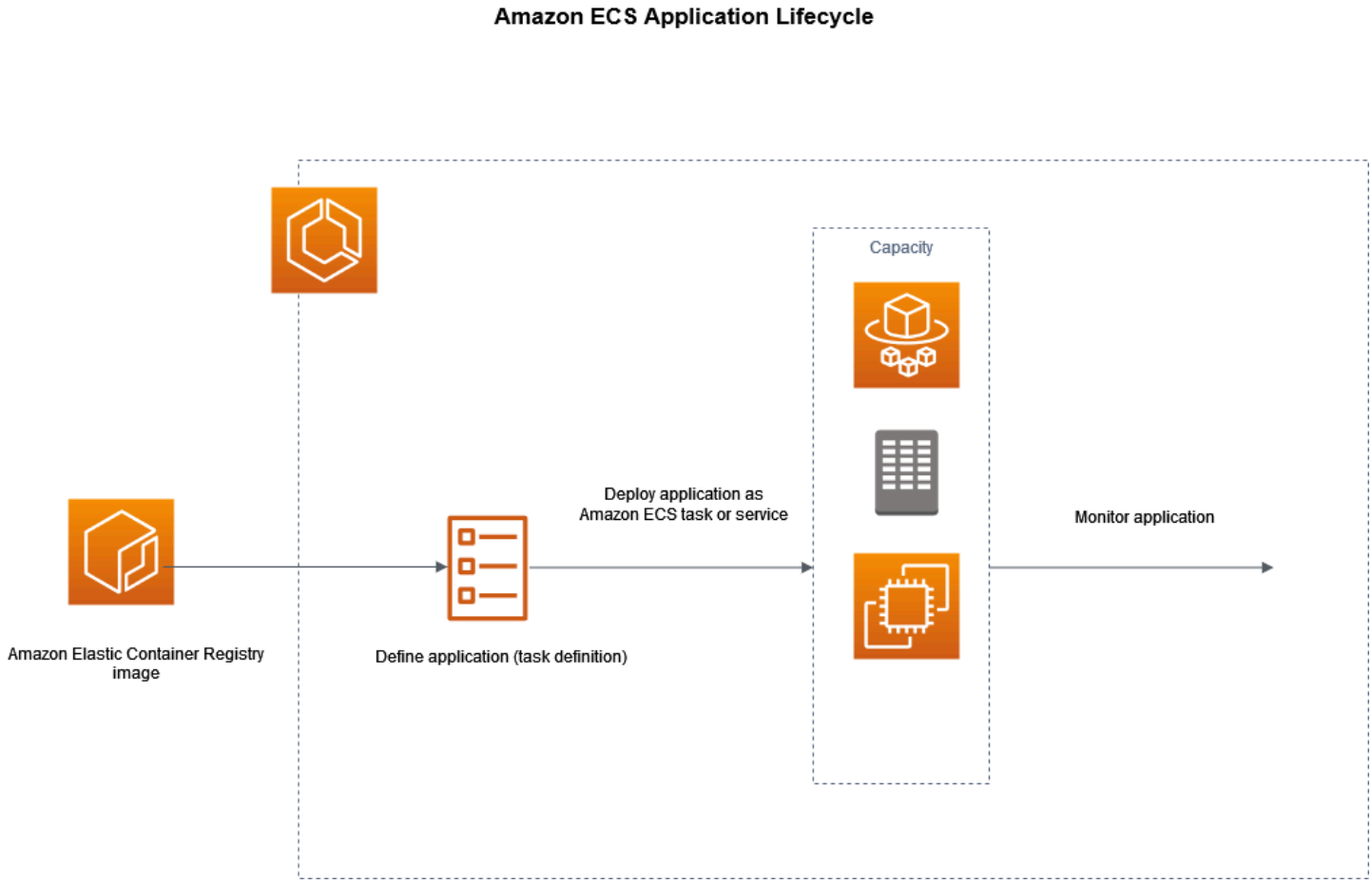
Penyediaan Amazon ECS

Ada beberapa opsi untuk penyediaan Amazon ECS:

- AWS Management Console— Menyediakan antarmuka web yang dapat Anda gunakan untuk mengakses sumber daya Amazon ECS Anda.
- AWS Command Line Interface (AWS CLI) — Menyediakan perintah untuk serangkaian AWS layanan yang luas, termasuk Amazon ECS. Ini didukung di Windows, Mac, dan Linux. Untuk informasi selengkapnya, lihat AWS Command Line Interface.
- AWS SDK — Menyediakan API khusus bahasa dan menangani banyak detail koneksi. Ini termasuk menghitung tanda tangan, menangani percobaan ulang permintaan, dan penanganan kesalahan. Untuk informasi selengkapnya, lihat AWS SDK.
- Copilot — Menyediakan alat sumber terbuka bagi pengembang untuk membangun, merilis, dan mengoperasikan aplikasi container siap produksi di Amazon ECS. Untuk informasi lebih lanjut, lihat Copilot di situs web. GitHub
- AWS CDKMenyediakan kerangka pengembangan perangkat lunak open-source yang dapat Anda gunakan untuk memodelkan dan menyediakan sumber daya aplikasi cloud Anda menggunakan bahasa pemrograman yang sudah dikenal. AWS CDK Menyediakan sumber daya Anda dengan cara yang aman dan berulang. AWS CloudFormation

Siklus hidup aplikasi

Diagram berikut menunjukkan siklus hidup aplikasi dan cara kerjanya dengan komponen Amazon ECS.



Anda harus merancang aplikasi Anda sehingga mereka dapat berjalan pada kontainer. Container adalah unit standar pengembangan perangkat lunak yang menyimpan semua yang dibutuhkan aplikasi perangkat lunak Anda untuk dijalankan. Ini termasuk kode yang relevan, runtime, alat sistem, dan pustaka sistem. Container dibuat dari template read-only yang disebut gambar. Gambar biasanya dibuat dari Dockerfile. Dockerfile adalah file plaintext yang menentukan semua komponen yang disertakan dalam wadah. Setelah dibuat, gambar-gambar ini disimpan dalam registri seperti Amazon ECR tempat mereka dapat diunduh.

Setelah Anda membuat dan menyimpan gambar Anda, Anda membuat definisi tugas Amazon ECS. Definisi tugas adalah cetak biru untuk aplikasi Anda. Ini adalah file teks dalam format JSON yang menjelaskan parameter dan satu atau lebih wadah yang membentuk aplikasi Anda. Misalnya, Anda dapat menggunakannya untuk menentukan gambar dan parameter untuk sistem operasi, wadah mana yang akan digunakan, port mana yang akan dibuka untuk aplikasi Anda, dan volume data apa yang akan digunakan dengan wadah dalam tugas. Parameter khusus telah tersedia untuk ketentuan tugas Anda tergantung pada kebutuhan aplikasi spesifik milik Anda.

Setelah menentukan definisi tugas, Anda menerapkannya sebagai layanan atau tugas di klaster. Cluster adalah pengelompokan logis tugas atau layanan yang berjalan pada infrastruktur kapasitas yang terdaftar ke cluster.

tugas adalah penunjukan hal abstrak dengan instans yang konkret dari ketentuan tugas dalam sebuah klaster. Anda dapat menjalankan tugas mandiri, atau Anda dapat menjalankan tugas sebagai bagian dari layanan. Anda dapat menggunakan layanan Amazon ECS untuk menjalankan dan mempertahankan jumlah tugas yang Anda inginkan secara bersamaan di kluster Amazon ECS. Cara kerjanya adalah, jika salah satu tugas Anda gagal atau berhenti karena alasan apa pun, penjadwal layanan Amazon ECS meluncurkan instance lain berdasarkan definisi tugas Anda. Ia melakukan ini untuk menggantikannya dan dengan demikian mempertahankan jumlah tugas yang Anda inginkan dalam layanan.

Agen kontainer berjalan pada setiap instance kontainer dalam klaster Amazon ECS. Agen mengirimkan informasi tentang tugas yang sedang berjalan dan pemanfaatan sumber daya kontainer Anda ke Amazon ECS. Ini memulai dan menghentikan tugas setiap kali menerima permintaan dari Amazon ECS.

Setelah menerapkan tugas atau layanan, Anda dapat menggunakan salah satu alat berikut untuk memantau penerapan dan aplikasi Anda:

- CloudWatch
- Pemantauan Runtime

Informasi terkait

Sumber daya terkait berikut dapat membantu Anda ketika bekerja dengan layanan ini.

- AWS Fargate— Ikhtisar fitur Fargate.
- Windows on AWS — Ikhtisar Windows tentang AWS beban kerja dan layanan.
- Linux dari AWS – Portofolio sistem operasi berbasis Linux modern dari AWS.

Tutorial untuk developer

- AWS Compute Blogs — Informasi tentang fitur baru, penyelaman mendalam ke fitur, contoh kode, dan praktik terbaik.

AWS re:Post

AWS re:Post— layanan tanya jawab AWS terkelola (Tanya Jawab) yang menawarkan jawaban yang bersumber dari banyak orang dan ditinjau oleh ahli untuk pertanyaan teknis Anda.

Harga

- Harga Amazon ECS - Informasi harga untuk Amazon ECS.
- AWS Fargate harga — Informasi harga untuk Fargate.

AWS Sumber daya umum

Sumber daya umum berikut dapat membantu Anda saat Anda bekerja dengannya AWS.

- Kelas & Lokakarya - Tautan ke kursus berbasis peran dan khusus, selain laboratorium mandiri untuk membantu mempertajam keterampilan Anda AWS dan mendapatkan pengalaman praktis.
- AWS Pusat Pengembang — Jelajahi tutorial, unduh alat, dan pelajari tentang acara AWS pengembang.
- AWS Alat Pengembang - Tautan ke alat pengembang, SDK, toolkit IDE, dan alat baris perintah untuk mengembangkan dan mengelola aplikasi. AWS
- Memulai Pusat Sumber Daya — Pelajari cara menyiapkan Akun AWS, bergabung dengan AWS komunitas, dan meluncurkan aplikasi pertama Anda.
- Hands-On Tutorial - Ikuti step-by-step tutorial untuk meluncurkan aplikasi pertama Anda. AWS
- AWS Whitepaper — Tautan ke daftar lengkap AWS whitepaper teknis, yang mencakup topik-topik seperti arsitektur, keamanan, dan ekonomi dan ditulis oleh AWS Solutions Architects atau pakar teknis lainnya.
- AWS Support Pusat — Hub untuk membuat dan mengelola AWS Support kasus Anda. Juga termasuk tautan ke sumber daya bermanfaat lainnya, seperti forum, FAQ teknis, status kesehatan layanan, dan. AWS Trusted Advisor
- AWS Support— Halaman web utama untuk informasi tentang AWS Support, saluran dukungan respons cepat untuk membantu Anda membangun dan menjalankan aplikasi di cloud. one-on-one
- Hubungi Kami – Titik kontak pusat untuk pertanyaan tentang tandaihan AWS , akun, peristiwa, penyalahgunaan, dan masalah lainnya.
- AWS Ketentuan Situs — Informasi terperinci tentang hak cipta dan merek dagang kami; akun, lisensi, dan akses situs Anda; dan topik lainnya.

Memulai dengan Amazon ECS

Panduan berikut memberikan pengantar alat yang tersedia untuk mengakses Amazon ECS dan prosedur pengantar langkah demi langkah untuk menjalankan kontainer. Dasar-dasar Docker membawa Anda melalui langkah-langkah dasar untuk membuat gambar kontainer Docker dan mengunggahnya ke repositori pribadi Amazon ECR. Panduan memulai memandu Anda menggunakan antarmuka baris perintah AWS Copilot dan AWS Management Console untuk menyelesaikan tugas-tugas umum untuk menjalankan container Anda di Amazon ECS dan. AWS Fargate

Daftar Isi

Siapkan untuk menggunakan Amazon ECS

Jika Anda sudah mendaftar ke Amazon Web Services (AWS) dan telah menggunakan Amazon Elastic Compute Cloud (Amazon EC2), Anda hampir dapat menggunakan Amazon ECS. Proses pengaturan untuk kedua layanan ini serupa. Panduan berikut mempersiapkan Anda untuk meluncurkan cluster Amazon ECS pertama Anda.

Selesaikan tugas-tugas berikut untuk menyiapkan Amazon ECS.

Mendaftar untuk Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar untuk Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk secara online.

   Anda akan diminta untuk menerima panggilan telepon dan memasukkan kode verifikasi pada keypad telepon sebagai bagian dari prosedur pendaftaran.

Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWSdibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya dalam akun. Sebagai praktik terbaik keamanan, tetapkan akses administratif ke pengguna administratif, dan hanya gunakan pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS mengirimi Anda email konfirmasi setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun saat ini dan mengelola akun dengan mengunjungi https://aws.amazon.com/ dan memilih Akun Saya.

Membuat pengguna administratif

Setelah Anda mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Amankan Anda Pengguna root akun AWS

1. Masuk ke AWS Management Consolesebagai pemilik akun dengan memilih pengguna Root dan memasukkan alamat Akun AWS email Anda. Di halaman berikutnya, masukkan kata sandi Anda.

   Untuk bantuan masuk menggunakan pengguna root, lihat Masuk sebagai pengguna root dalam Panduan Pengguna AWS Sign-In .
2. Aktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuk, lihat Mengaktifkan perangkat MFA virtual untuk pengguna Akun AWS root (konsol) Anda di Panduan Pengguna IAM.

Membuat pengguna administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center .
2. Di Pusat Identitas IAM, berikan akses administratif ke sebuah pengguna administratif.

   Untuk tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, lihat Mengkonfigurasi akses pengguna dengan default Direktori Pusat Identitas IAM di Panduan AWS IAM Identity Center Pengguna.

Masuk sebagai pengguna administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email Anda saat Anda membuat pengguna Pusat Identitas IAM.

  Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat Masuk ke portal AWS akses di Panduan AWS Sign-In Pengguna.

Buat virtual private cloud

Anda dapat menggunakan Amazon Virtual Private Cloud (Amazon VPC) untuk meluncurkan AWS sumber daya ke jaringan virtual yang telah Anda tentukan. Kami sangat menyarankan agar Anda meluncurkan instance kontainer Anda di VPC.

Jika Anda memiliki VPC default, Anda dapat melewati bagian ini dan pindah ke tugas berikutnya,. Membuat grup keamanan Untuk menentukan apakah Anda memiliki VPC default, lihat Platform yang Didukung di Konsol Amazon EC2 di Panduan Pengguna Amazon EC2 untuk Instans Linux. Jika tidak, Anda dapat membuat VPC non-default di akun Anda menggunakan langkah-langkah di bawah ini.

> **Important**
>
> Jika akun Anda mendukung Amazon EC2 Classic di suatu wilayah, maka Anda tidak memiliki VPC default di wilayah tersebut.

Untuk informasi tentang cara membuat VPC, lihat Membuat VPC hanya di Panduan Pengguna VPC Amazon, dan gunakan tabel berikut untuk menentukan opsi apa yang harus dipilih.

Opsi	Nilai
Sumber daya untuk dibuat	Hanya VPC
Nama	Secara opsional berikan nama untuk VPC Anda.
Blok CIDR IPv4	Input manual IPv4 CIDR

Opsi	Nilai
	Ukuran blok CIDR harus memiliki ukuran antara /16 dan /28.
Blok CIDR IPv6	Tidak ada blok IPv6 CIDR
Penghunian	Default

Untuk informasi lebih lanjut tentang Amazon VPC, lihat Apa itu Amazon VPC? dalam Panduan Pengguna Amazon VPC.

Membuat grup keamanan

Grup keamanan bertindak sebagai firewall untuk instans kontainer terkait, yang mengontrol lalu lintas masuk dan keluar di tingkat instans kontainer. Anda dapat menambahkan aturan ke grup keamanan yang memungkinkan Anda terhubung ke instans kontainer dari alamat IP Anda menggunakan SSH. Anda juga dapat menambahkan aturan yang mengizinkan akses HTTP dan HTTPS masuk dan keluar dari mana saja. Tambahkan aturan untuk membuka port yang diperlukan oleh tugas Anda. Instans kontainer memerlukan akses jaringan eksternal untuk berkomunikasi dengan titik akhir layanan Amazon ECS.

Jika Anda merencanakan untuk meluncurkan instans kontainer di beberapa Wilayah, Anda perlu membuat grup keamanan di setiap Wilayah. Untuk informasi lebih lanjut, lihat Wilayah dan Availability Zones dalam Panduan Pengguna Amazon EC2 untuk Instans Linux.

> **Tip**
>
> Anda memerlukan alamat IP publik komputer lokal Anda, yang bisa Anda dapatkan menggunakan layanan. Sebagai contoh, kami menyediakan layanan berikut: http://checkip.amazonaws.com/ atau https://checkip.amazonaws.com/. Untuk menemukan layanan lain yang menyediakan alamat IP Anda, gunakan frasa pencarian "apa alamat IP saya." Jika Anda terhubung melalui penyedia layanan Internet (ISP) atau dari belakang firewall tanpa alamat IP statis, Anda perlu mengetahui rentang alamat IP yang digunakan oleh komputer klien.

Untuk informasi tentang cara membuat grup keamanan, lihat Membuat grup keamanan di Panduan Pengguna Amazon EC2 untuk Instans Linux dan gunakan tabel berikut untuk menentukan opsi apa yang harus dipilih.

Opsi	Nilai
Wilayah	Wilayah yang sama di mana Anda membuat key pair Anda.
Nama	Nama yang mudah diingat, seperti ecs-instances-default-cluster.
VPC	VPC default (ditandai dengan "(default)". **Note** Jika akun Anda mendukung Amazon EC2 Classic, pilih VPC yang Anda buat di tugas sebelumnya.

Untuk informasi tentang aturan keluar yang akan ditambahkan untuk kasus penggunaan Anda, lihat Aturan grup keamanan untuk kasus penggunaan yang berbeda di Panduan Pengguna Amazon EC2 untuk Instans Linux.

Instans kontainer Amazon ECS tidak memerlukan port masuk untuk dibuka. Namun, Anda mungkin ingin menambahkan aturan SSH sehingga Anda dapat masuk ke instans kontainer dan memeriksa tugas-tugas dengan perintah Docker. Anda juga dapat menambahkan aturan untuk HTTP dan HTTPS jika Anda ingin instans kontainer Anda untuk meng-host tugas yang menjalankan web server. Instans kontainer juga memerlukan akses jaringan eksternal untuk berkomunikasi dengan titik akhir layanan Amazon ECS. Selesaikan langkah-langkah berikut ini untuk menambahkan aturan grup keamanan opsional ini.

Tambahkan tiga aturan masuk berikut ke grup keamanan Anda.Untuk informasi tentang cara membuat grup keamanan, lihat Menambahkan aturan ke grup keamanan Anda di Panduan Pengguna Amazon EC2 untuk Instans Linux.

Opsi	Nilai	
Aturan HTTP	Jenis: HTTP Sumber: Di mana saja (`0.0.0.0/0`) Opsi ini secara otomatis akan menambahkan blok CIDR 0.0.0.0/0 IPv4 sebagai sumber. Hal ini dapat diterima untuk waktu yang singkat di lingkungan pengujian, tetapi tidak aman dalam lingkungan produksi. Dalam produksi, Anda hanya dapat memberikan otorisasi pada alamat IP atau rentang alamat tertentu saja untuk mengakses instans Anda.	
Aturan HTTPS	Jenis: HTTPS Sumber: Di mana saja (`0.0.0.0/0`) Hal ini dapat diterima untuk waktu yang singkat di lingkungan pengujian, tetapi tidak aman dalam lingkungan produksi. Dalam produksi, Anda hanya dapat memberikan otorisasi pada alamat IP atau rentang alamat tertentu	

Opsi	Nilai
	saja untuk mengakses instans Anda.
Aturan SSH	Jenis: SSH Sumber: Kustom, tentukan alamat IP publik komputer atau jaringan Anda dalam notasi CIDR. Untuk menentukan alamat IP individu dalam notasi CIDR, tambahkan prefiks perutean `/32`. Misalnya, jika alamat IP Anda adalah `203.0.113.25` , sebutkan `203.0.113.25/32` . Jika perusahaan Anda mengalokasikan alamat-alamat dari sebuah rentang, tentukan keseluruhan rentang, seperti `203.0.113.0/24` . **Important** Demi alasan keamanan, kami tidak menyarankan Anda untuk mengizinkan akses SSH dari semua alamat IP (`0.0.0.0/0`) ke instans Anda, kecuali untuk tujuan pengujian dan hanya untuk waktu yang singkat.

Buat kredensi untuk terhubung ke instans EC2 Anda

Untuk Amazon ECS, key pair hanya diperlukan jika Anda berniat menggunakan tipe peluncuran EC2.

AWS menggunakan kriptografi kunci publik untuk mengamankan informasi login untuk instans Anda. Instans Linux, seperti instans penampung Amazon ECS, tidak memiliki kata sandi untuk digunakan untuk akses SSH. Anda menggunakan pasangan kunci untuk masuk dengan aman ke instans Anda. Anda menentukan nama pasangan kunci saat meluncurkan instans kontainer Anda, kemudian berikan kunci privat saat masuk menggunakan SSH.

Jika Anda belum membuat key pair, Anda dapat membuatnya menggunakan konsol Amazon EC2. Jika Anda merencanakan untuk meluncurkan instans di beberapa Wilayah, Anda perlu membuat pasangan kunci di setiap Wilayah. Untuk informasi selengkapnya tentang wilayah, lihat Wilayah dan Zona Ketersediaan di Panduan Pengguna Amazon EC2 untuk Instans Linux.

Untuk membuat pasangan kunci

- Gunakan konsol Amazon EC2 untuk membuat key pair. Untuk informasi selengkapnya tentang membuat key pair, lihat Membuat key pair di Panduan Pengguna Amazon EC2 untuk Instans Linux.

Untuk informasi tentang cara menyambung ke instans, lihat Connect ke instans Linux Anda di Panduan Pengguna Amazon EC2 untuk Instans Linux.

Instal AWS CLI

Ini AWS Management Console dapat digunakan untuk mengelola semua operasi secara manual dengan Amazon ECS. Namun, Anda dapat menginstal AWS CLI di desktop lokal atau kotak pengembang sehingga Anda dapat membuat skrip yang dapat mengotomatiskan tugas manajemen umum di Amazon ECS.

Untuk menggunakan Amazon ECS, instal AWS CLI versi terbaru. AWS CLI Untuk informasi tentang menginstal AWS CLI atau memutakhirkannya ke versi terbaru, lihat Menginstal Antarmuka Baris AWS Perintah di Panduan AWS Command Line Interface Pengguna.

Membuat gambar kontainer untuk digunakan di Amazon ECS

Amazon ECS menggunakan gambar Docker dalam definisi tugas untuk meluncurkan kontainer. Docker adalah teknologi yang menyediakan alat bagi Anda untuk membangun, menjalankan,

menguji, dan menyebarkan aplikasi terdistribusi dalam wadah. Docker menyediakan panduan tentang penerapan kontainer di Amazon ECS. Untuk informasi selengkapnya, lihat Docker Compose CLI - Amazon ECS.

Tujuan dari langkah-langkah yang diuraikan di sini adalah untuk memandu Anda membuat gambar Docker pertama Anda dan mendorong gambar itu ke Amazon ECR, yang merupakan registri penampung, untuk digunakan dalam definisi tugas Amazon ECS Anda. Panduan ini mengasumsikan bahwa Anda memiliki pemahaman dasar tentang apa itu Docker dan cara kerjanya. Untuk informasi selengkapnya tentang Docker, lihat Apa itu Docker? dan Gambaran umum Docker.

Prasyarat

Sebelum Anda mulai, pastikan prasyarat berikut terpenuhi.

- Pastikan Anda telah menyelesaikan langkah-langkah penyiapan Amazon ECR. Untuk informasi selengkapnya, lihat Menyiapkan Amazon ECR di Panduan Pengguna Amazon Elastic Container Registry.
- Pengguna Anda memiliki izin IAM yang diperlukan untuk mengakses dan menggunakan layanan Amazon ECR. Untuk informasi selengkapnya, lihat kebijakan terkelola Amazon ECR.
- Anda telah menginstal Docker. Untuk langkah-langkah instalasi Docker untuk Amazon Linux 2, lihatMenginstal Docker pada AL2023. Untuk semua sistem operasi lainnya, lihat dokumentasi Docker di ikhtisar Docker Desktop.
- Anda telah AWS CLI diinstal dan dikonfigurasi. Untuk informasi selengkapnya, lihat Menginstal AWS Command Line Interface dalam Panduan Pengguna AWS Command Line Interface .

Jika Anda tidak memiliki atau memerlukan lingkungan pengembangan lokal dan Anda lebih suka menggunakan instans Amazon EC2 untuk menggunakan Docker, kami menyediakan langkah-langkah berikut untuk meluncurkan instans Amazon EC2 menggunakan Amazon Linux 2 dan menginstal Docker Engine dan CLI Docker.

Menginstal Docker pada AL2023

Docker tersedia dalam banyak sistem operasi yang berbeda, termasuk sebagian besar distribusi Linux modern, seperti Ubuntu, dan bahkan macOS dan Windows. Untuk informasi lebih lanjut tentang cara menginstal Docker pada sistem operasi tertentu Anda, kunjungi situs webpanduan penginstalan Docker.

Anda tidak memerlukan sistem pengembangan lokal untuk menggunakan Docker. Jika Anda sudah menggunakan Amazon EC2, Anda dapat meluncurkan instans Amazon Linux 2023 dan menginstal Docker untuk memulai.

Jika Anda sudah menginstal Docker, langsung ke Membuat citra Docker.

Untuk menginstal Docker pada instans Amazon EC2 menggunakan Amazon Linux 2023 AMI

1. Luncurkan instance dengan AMI Amazon Linux 2023 terbaru. Untuk informasi selengkapnya, lihat Meluncurkan instans di Panduan Pengguna Amazon EC2 untuk Instans Linux.
2. Connect ke instans Anda. Untuk informasi lebih lanjut, lihat Connect ke Instans Linux Anda dalam Panduan Pengguna Amazon EC2 untuk Instans Linux.
3. Memperbarui paket yang diinstal dan paket cache pada instans Anda.

   ```
   sudo yum update -y
   ```

4. Instal paket Edisi Komunitas Docker terbaru.

   ```
   sudo yum install docker
   ```

5. Mulai layanan Docker.

   ```
   sudo service docker start
   ```

6. Tambahkan `ec2-user` ke grup `docker` sehingga Anda dapat menjalankan perintah Docker tanpa menggunakan `sudo`.

   ```
   sudo usermod -a -G docker ec2-user
   ```

7. Keluar dan masuk kembali untuk mengambil izin grup `docker` yang baru. Anda dapat melakukannya dengan menutup jendela terminal SSH Anda saat ini dan menghubungkan kembali ke instans Anda yang baru. Sesi SSH baru Anda akan memiliki izin grup `docker` yang sesuai.
8. Verifikasi bahwa `ec2-user` dapat menjalankan perintah Docker tanpa `sudo`.

   ```
   docker info
   ```

> **Note**
>
> Dalam beberapa kasus, Anda mungkin perlu melakukan booting ulang pada instans Anda untuk memberikan izin bagi `ec2-user` untuk mengakses daemon Docker. Coba booting ulang instans Anda jika Anda melihat kesalahan berikut ini:
>
> ```
> Cannot connect to the Docker daemon. Is the docker daemon running on this
>  host?
> ```

Membuat citra Docker

Definisi tugas Amazon ECS menggunakan citra Docker untuk meluncurkan kontainer pada contoh kontainer di klaster Anda. Di bagian ini, Anda membuat gambar Docker dari aplikasi web sederhana, dan mengujinya di sistem lokal Anda atau instans Amazon EC2, lalu mendorong gambar ke registri penampung Amazon ECR sehingga Anda dapat menggunakannya dalam definisi tugas Amazon ECS.

Untuk membuat citra Docker dari aplikasi web sederhana

1. Buat file bernama `Dockerfile`. Dockerfile adalah manifes yang menjelaskan citra dasar yang akan digunakan untuk citra Docker Anda dan apa yang ingin Anda instal dan jalankan di atasnya. Untuk informasi selengkapnya tentang Dockerfiles, buka Referensi Dockerfile.

   ```
   touch Dockerfile
   ```

2. Edit `Dockerfile` yang baru saja Anda buat dan tambahkan konten berikut.

   ```
   FROM public.ecr.aws/amazonlinux/amazonlinux:latest

   # Install dependencies
   RUN yum update -y && \
    yum install -y httpd

   # Install apache and write hello world message
   RUN echo 'Hello World!' > /var/www/html/index.html

   # Configure apache
   RUN echo 'mkdir -p /var/run/httpd' >> /root/run_apache.sh && \
   ```

```
 echo 'mkdir -p /var/lock/httpd' >> /root/run_apache.sh && \
 echo '/usr/sbin/httpd -D FOREGROUND' >> /root/run_apache.sh && \
 chmod 755 /root/run_apache.sh

EXPOSE 80

CMD /root/run_apache.sh
```

Dockerfile ini menggunakan gambar Amazon Linux 2 publik yang dihosting di Amazon ECR Public. `RUN`Instruksi memperbarui cache paket, menginstal beberapa paket perangkat lunak untuk server web, dan kemudian menulis "Hello World!" konten ke root dokumen server web. `EXPOSE`Instruksi berarti bahwa port 80 pada wadah adalah yang mendengarkan, dan `CMD` instruksi memulai server web.

3. Membangun citra Docker dari Dockerfile Anda.

> **Note**
>
> Beberapa versi Docker mungkin memerlukan jalur lengkap ke Dockerfile Anda dalam perintah berikut, bukan jalur relatif yang ditunjukkan di bawah ini.

```
docker build -t hello-world .
```

4. Buat daftar gambar kontainer Anda.

```
docker images --filter reference=hello-world
```

Output:

```
REPOSITORY          TAG                 IMAGE ID            CREATED
 SIZE
hello-world         latest              e9ffedc8c286        4 minutes ago
 194MB
```

5. Jalankan citra yang baru dibuat. Opsi `-p 80:80` memetakan port 80 yang terbuka pada kontainer ke port 80 pada sistem host. Untuk informasi lebih lanjut tentang docker run, buka Referensi menjalankan Docker.

```
docker run -t -i -p 80:80 hello-world
```

> **Note**
>
> Output dari server web Apache ditampilkan di jendela terminal. Anda dapat mengabaikan pesan "`Could not reliably determine the fully qualified domain name`".

6. Buka peramban dan arahkan ke server yang menjalankan Docker dan meng-host kontainer Anda.

   - Jika Anda menggunakan instans EC2, nilai ini adalah nilai DNS Publik untuk server, yang merupakan alamat yang sama yang Anda gunakan untuk terhubung ke instans dengan SSH. Pastikan bahwa grup keamanan untuk instans Anda mengizinkan lalu lintas masuk pada port 80.
   - Jika Anda menjalankan Docker secara lokal, arahkan peramban Anda ke http://localhost/.
   - Jika Anda menggunakan docker-machine di komputer Windows atau Mac, temukan alamat IP VirtualBox VM yang menghosting Docker dengan docker-machine ip perintah, ganti nama mesin dengan *`nama mesin`* docker yang Anda gunakan.

     ```
     docker-machine ip machine-name
     ```

   Anda akan melihat halaman web dengan pernyataan "Hello World!" .

7. Hentikan kontainer Docker dengan mengetik Ctrl + c.

Dorong gambar Anda ke Amazon Elastic Container Registry

Amazon ECR adalah layanan registri AWS Docker terkelola. Anda dapat menggunakan CLI Docker untuk mendorong, menarik, dan mengelola gambar di repositori Amazon ECR Anda. Untuk detail produk Amazon ECR, studi kasus pelanggan unggulan, dan FAQ, lihat halaman detail produk Amazon Elastic Container Registry.

Untuk menandai gambar Anda dan mendorongnya ke Amazon ECR

1. Buat repositori Amazon ECR untuk menyimpan gambar Anda. `hello-world` Perhatikan `repositoryUri` pada outputnya.

   Gantikan`region`, dengan Anda Wilayah AWS, misalnya,`us-east-1`.

   ```
   aws ecr create-repository --repository-name hello-repository --region region
   ```

   Output:

   ```
   {
       "repository": {
           "registryId": "aws_account_id",
           "repositoryName": "hello-repository",
           "repositoryArn": "arn:aws:ecr:region:aws_account_id:repository/hello-
   repository",
           "createdAt": 1505337806.0,
           "repositoryUri": "aws_account_id.dkr.ecr.region.amazonaws.com/hello-
   repository"
       }
   }
   ```

2. Tandai citra `hello-world` dengan nilai `repositoryUri` dari langkah sebelumnya.

   ```
   docker tag hello-world aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository
   ```

3. Jalankan perintah aws ecr get-login-password. Tentukan URI registri yang ingin Anda autentikasi. Untuk informasi selengkapnya, lihat Autentikasi Registri dalam Panduan Pengguna Amazon Elastic Container Registry.

   ```
   aws ecr get-login-password --region region | docker login --username AWS --
   password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
   ```

   Output:

   ```
   Login Succeeded
   ```

> **Important**
>
> Jika Anda menerima kesalahan, instal atau perbarui ke versi terbaru AWS CLI. Untuk informasi selengkapnya, lihat Menginstal AWS Command Line Interface dalam Panduan Pengguna Amazon EKS AWS Command Line Interface .

4. Dorong gambar ke Amazon ECR dengan `repositoryUri` nilai dari langkah sebelumnya.

```
docker push aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository
```

Hapus

Untuk melanjutkan dengan membuat definisi tugas Amazon ECS dan meluncurkan tugas dengan gambar penampung Anda, lewati ke file. Langkah selanjutnya Setelah selesai bereksperimen dengan gambar Amazon ECR Anda, Anda dapat menghapus repositori sehingga Anda tidak dikenakan biaya untuk penyimpanan gambar.

```
aws ecr delete-repository --repository-name hello-repository --region region --force
```

Langkah selanjutnya

Definisi tugas Anda memerlukan peran eksekusi tugas. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

Setelah Anda membuat dan mendorong gambar penampung Anda ke Amazon ECR, Anda harus mempertimbangkan langkah-langkah berikut ini.

- the section called “Menggunakan konsol dengan wadah Linux aktif AWS Fargate”
- the section called “Menggunakan wadah Windows pada AWS Fargate”
- Membuat cluster dengan tugas Fargate Linux menggunakan AWS CLI

Memulai dengan wadah Linux di AWS Fargate

Amazon Elastic Container Service (Amazon ECS) adalah layanan manajemen kontainer yang sangat skalabel, cepat, yang memudahkan untuk menjalankan, menghentikan, dan mengelola kontainer

Anda. Anda dapat meng-host kontainer Anda di infrastruktur tanpa server yang dikelola oleh Amazon ECS dengan meluncurkan layanan atau tugas Anda. AWS Fargate Untuk informasi lebih lanjut tentang Fargate, lihat. Amazon ECS aktif AWS Fargate

Mulai menggunakan Amazon ECS AWS Fargate dengan menggunakan jenis peluncuran Fargate untuk tugas Anda di Wilayah tempat Amazon ECS AWS mendukung Fargate.

Selesaikan langkah-langkah berikut untuk memulai Amazon ECS aktif. AWS Fargate

Prasyarat

Sebelum memulai, selesaikan langkah-langkah Siapkan untuk menggunakan Amazon ECS dan bahwa AWS pengguna Anda memiliki izin yang ditentukan dalam contoh kebijakan `AdministratorAccess` IAM.

Konsol mencoba untuk secara otomatis membuat peran IAM eksekusi tugas, yang diperlukan untuk tugas Fargate. Untuk memastikan bahwa konsol dapat membuat peran IAM ini, salah satu dari berikut ini harus benar:

- Pengguna Anda memiliki akses administrator. Untuk informasi selengkapnya, lihat Siapkan untuk menggunakan Amazon ECS.
- Pengguna Anda memiliki izin IAM untuk membuat peran layanan. Untuk informasi selengkapnya, lihat Membuat Peran untuk Mendelegasikan Izin ke Layanan. AWS
- Pengguna dengan akses administrator telah secara manual membuat peran eksekusi tugas sehingga peran itu tersedia pada akun yang akan digunakan. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

> **Important**
>
> Grup keamanan yang Anda pilih saat membuat layanan dengan definisi tugas Anda harus memiliki port 80 terbuka untuk lalu lintas masuk. Tambahkan aturan masuk berikut ke grup keamanan Anda. Untuk informasi tentang cara membuat grup keamanan, lihat Menambahkan aturan ke grup keamanan Anda di Panduan Pengguna Amazon EC2 untuk Instans Linux.
>
> - Jenis: HTTP
> - Protokol: TCP
> - Rentang port: 80

- Sumber: Di mana saja (`0.0.0.0/0`)

Langkah 1: Buat cluster

Buat cluster yang menggunakan VPC default.

Sebelum Anda mulai, tetapkan izin IAM yang sesuai. Untuk informasi selengkapnya, lihat the section called “Contoh klaster”.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.
3. Di panel navigasi, pilih Klaster.
4. Pada halaman Klaster, pilih Buat klaster.
5. Pada Konfigurasi klaster, untuk Nama klaster, masukkan nama unik.

   Nama tersebut dapat berisi hingga 255 huruf (huruf besar dan huruf kecil), angka, dan tanda hubung.
6. (Opsional) Untuk mengaktifkan Wawasan Kontainer, perluas Pemantauan, lalu aktifkan Gunakan Wawasan Kontainer.
7. (Opsional) Untuk membantu mengidentifikasi klaster Anda, perluas Tag, lalu konfigurasikan tag Anda.

   [Tambahkan tag] Pilih Tambah tag dan lakukan hal berikut:

   - Untuk Kunci, masukkan nama kunci.
   - Untuk Nilai, masukkan nilai kunci.

   [Hapus tag] Pilih Hapus di sebelah kanan Kunci dan Nilai tag.
8. Pilih Buat.

Langkah 2: Buat definisi tugas

Ketentuan tugas adalah seperti cetak biru untuk aplikasi Anda. Setiap kali Anda meluncurkan tugas di Amazon ECS, Anda menentukan definisi tugas. Layanan kemudian tahu citra Docker mana yang

digunakan untuk kontainer, berapa banyak kontainer untuk digunakan dalam tugas, dan alokasi sumber daya untuk setiap kontainer.

1. Di panel navigasi, pilih Ketentuan Tugas.
2. Pilih Buat Definisi Tugas baru, Buat revisi baru dengan JSON.
3. Salin dan tempel contoh ketentuan tugas berikut ke dalam kotak dan kemudian pilih Simpan.

```
{
    "family": "sample-fargate",
    "networkMode": "awsvpc",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": [
                "sh",
  "-c"
            ],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
            ]
        }
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "cpu": "256",
    "memory": "512"
}
```

4. Pilih Buat.

Langkah 3: Buat layanan

Buat layanan menggunakan definisi tugas.

1. Di panel navigasi, pilih Clusters, lalu pilih cluster yang Anda buat. Langkah 1: Buat cluster
2. Dari tab Layanan, pilih Buat.
3. Di bawah konfigurasi Deployment, tentukan cara aplikasi Anda di-deploy.
   a. Untuk definisi Tugas, pilih definisi tugas yang Anda buatLangkah 2: Buat definisi tugas.
   b. Untuk nama Layanan, masukkan nama untuk layanan Anda.
   c. Untuk tugas yang diinginkan, masukkan 1.
4. Di bawah Jaringan, Anda dapat membuat grup keamanan baru atau memilih grup keamanan yang ada untuk tugas Anda. Pastikan grup keamanan yang Anda gunakan memiliki aturan masuk yang tercantum di bawahPrasyarat.
5. Pilih Buat.

Langkah 4: Melihat layanan Anda

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pilih cluster tempat Anda menjalankan layanan.
4. Di tab Layanan, di bawah Nama layanan, pilih layanan yang Anda buatLangkah 3: Buat layanan.
5. Pilih tab Tugas, lalu pilih tugas di layanan Anda.
6. Pada halaman tugas, di bagian Konfigurasi, di bawah IP Publik, pilih Buka alamat.

Langkah 5: Bersihkan

Setelah selesai menggunakan kluster Amazon ECS, Anda harus membersihkan sumber daya yang terkait dengannya untuk menghindari biaya untuk sumber daya yang tidak Anda gunakan.

Beberapa sumber daya Amazon ECS, seperti tugas, layanan, cluster, dan instans kontainer, dibersihkan menggunakan konsol Amazon ECS. Sumber daya lain, seperti instans Amazon EC2,

penyeimbang beban Elastic Load Balancing, dan grup Auto Scaling, harus dibersihkan secara manual di konsol Amazon EC2 atau dengan menghapus tumpukan yang membuatnya. AWS CloudFormation

1. Pada panel navigasi, silakan pilih Klaster.
2. Pada halaman Clusters, pilih cluster yang Anda buat untuk tutorial ini.
3. Pilih tab Layanan.
4. Pilih layanan, lalu pilih Hapus.
5. Pada prompt konfirmasi, masukkan hapus dan kemudian pilih Hapus.

   Tunggu hingga layanan dihapus.
6. Pilih Hapus klaster. Pada prompt konfirmasi, masukkan hapus ***`nama cluster`***, lalu pilih Hapus. Menghapus cluster membersihkan sumber daya terkait yang dibuat dengan cluster, termasuk grup Auto Scaling, VPC, atau load balancer.

Memulai dengan Windows di Amazon EC2

Memulai Amazon ECS menggunakan tipe peluncuran EC2 dengan mendaftarkan definisi tugas, membuat klaster, dan membuat layanan di konsol.

Selesaikan langkah-langkah berikut untuk memulai Amazon ECS menggunakan tipe peluncuran EC2.

Prasyarat

Sebelum memulai, selesaikan langkah-langkah Siapkan untuk menggunakan Amazon ECS dan bahwa AWS pengguna Anda memiliki izin yang ditentukan dalam contoh kebijakan `AdministratorAccess` IAM.

Konsol mencoba untuk secara otomatis membuat peran IAM eksekusi tugas, yang diperlukan untuk tugas Fargate. Untuk memastikan bahwa konsol dapat membuat peran IAM ini, salah satu dari berikut ini harus benar:

- Pengguna Anda memiliki akses administrator. Untuk informasi selengkapnya, lihat Siapkan untuk menggunakan Amazon ECS.
- Pengguna Anda memiliki izin IAM untuk membuat peran layanan. Untuk informasi selengkapnya, lihat Membuat Peran untuk Mendelegasikan Izin ke Layanan. AWS

- Pengguna dengan akses administrator telah secara manual membuat peran eksekusi tugas sehingga peran itu tersedia pada akun yang akan digunakan. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

> **Important**
>
> Grup keamanan yang Anda pilih saat membuat layanan dengan definisi tugas Anda harus memiliki port 80 terbuka untuk lalu lintas masuk. Tambahkan aturan masuk berikut ke grup keamanan Anda. Untuk informasi tentang cara membuat grup keamanan, lihat Menambahkan aturan ke grup keamanan Anda di Panduan Pengguna Amazon EC2 untuk Instans Linux.
>
> - Jenis: HTTP
> - Protokol: TCP
> - Rentang port: 80
> - Sumber: Di mana saja (`0.0.0.0/0`)

Langkah 1: Buat cluster

Cluster Amazon ECS adalah pengelompokan tugas, layanan, dan instance kontainer yang logis.

Langkah-langkah berikut memandu Anda membuat klaster dengan satu instans Amazon EC2 yang terdaftar di dalamnya yang akan memungkinkan kami menjalankan tugas di dalamnya. Jika bidang tertentu tidak disebutkan, tinggalkan nilai konsol default.

Untuk membuat cluster baru (konsol Amazon ECS)

Sebelum Anda mulai, tetapkan izin IAM yang sesuai. Untuk informasi selengkapnya, lihat the section called “Contoh klaster”.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.
3. Di panel navigasi, pilih Klaster.
4. Pada halaman Klaster, pilih Buat klaster.
5. Pada Konfigurasi klaster, untuk Nama klaster, masukkan nama unik.

Nama tersebut dapat berisi hingga 255 huruf (huruf besar dan huruf kecil), angka, dan tanda hubung.

6. (Opsional) Untuk mengubah VPC dan subnet tempat tugas dan layanan Anda diluncurkan, di bawah Jaringan, lakukan salah satu operasi berikut:

   - Untuk menghapus subnet, di bawah Subnet, pilih X untuk setiap subnet yang ingin Anda hapus.
   - Untuk mengubah ke VPC selain VPC default, di bawah VPC, pilih VPC yang ada, lalu di bawah Subnet, pilih setiap subnet.

7. Untuk menambahkan instans Amazon EC2 ke klaster Anda, perluas Infrastruktur, lalu pilih instans Amazon EC2. Selanjutnya, konfigurasikan grup Auto Scaling yang bertindak sebagai penyedia kapasitas:

   a. Untuk menggunakan grup Auto Scaling yang ada, dari grup Auto Scaling (ASG), pilih grup.

   b. Untuk membuat grup Auto Scaling, dari grup Auto Scaling (ASG), pilih Buat grup baru, lalu berikan detail berikut tentang grup:

      - Untuk Sistem Operasi/Arsitektur, pilih AMI Amazon ECS yang dioptimalkan untuk instans grup Auto Scaling.
      - Untuk jenis instans EC2, pilih jenis instans untuk beban kerja Anda. Untuk informasi selengkapnya tentang berbagai jenis instans, lihat Instans Amazon EC2.

        Penskalaan terkelola berfungsi paling baik jika grup Auto Scaling Anda menggunakan jenis instans yang sama atau serupa.
      - Untuk key pair SSH, pilih pair yang membuktikan identitas Anda saat Anda terhubung ke instance.
      - Untuk Kapasitas, masukkan jumlah minimum dan jumlah maksimum instans yang akan diluncurkan di grup Auto Scaling. Instans Amazon EC2 mengeluarkan biaya saat ada di sumber daya Anda. AWS Untuk informasi selengkapnya, lihat Penetapan Harga Amazon EC2.

8. (Opsional) Untuk mengaktifkan Wawasan Kontainer, perluas Pemantauan, lalu aktifkan Gunakan Wawasan Kontainer.

9. (Opsional) Untuk mengelola tag cluster, memperluas Tag, dan kemudian melakukan salah satu operasi berikut:

   [Tambahkan tag] Pilih Tambah tag dan lakukan hal berikut:

- Untuk Kunci, masukkan nama kunci.
- Untuk Nilai, masukkan nilai kunci.

[Hapus tag] Pilih Hapus di sebelah kanan Kunci dan Nilai tag.

10. Pilih Buat.

Langkah 2: Daftarkan definisi tugas

Untuk mendaftarkan definisi tugas sampel dengan AWS Management Console

1. Di panel navigasi, pilih Ketentuan Tugas.
2. Pilih Buat definisi tugas baru, Buat definisi tugas baru dengan JSON.
3. Salin dan tempel contoh definisi tugas berikut ke dalam kotak, lalu pilih Simpan.

```
{
    "containerDefinitions": [
        {
            "command": ["New-Item -Path C:\\inetpub\\wwwroot\\index.html
 -Type file -Value '<html> <head> <title>Amazon ECS Sample App</title>
 <style>body {margin-top: 40px; background-color: #333;} </style> </head><body>
 <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p>'; C:\\ServiceMonitor.exe w3svc"],
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "essential": true,
            "cpu": 2048,
            "memory": 4096,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "name": "sample_windows_app",
            "portMappings": [
                {
                    "hostPort": 443,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
```

```
            ]
        }
    ],
    "memory": "4096",
    "cpu": "2048",
    "family": "windows-simple-iis-2019-core",
    "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
    "runtimePlatform": {"operatingSystemFamily": "WINDOWS_SERVER_2019_CORE"},
    "requiresCompatibilities": ["EC2"]
}
```

4. Verifikasi informasi Anda lalu pilih Buat.

Langkah 3: Buat Layanan

Layanan Amazon ECS membantu Anda menjalankan dan memelihara sejumlah instans definisi tugas yang ditentukan secara bersamaan di kluster Amazon ECS. Jika salah satu tugas Anda gagal atau berhenti karena alasan apa pun, penjadwal layanan Amazon ECS meluncurkan contoh lain dari definisi tugas Anda untuk menggantinya guna mempertahankan jumlah tugas yang diinginkan dalam layanan. Untuk informasi selengkapnya tentang layanan, lihat Layanan-layanan Amazon ECS.

Untuk membuat layanan

1. Pada panel navigasi, silakan pilih Klaster.
2. Pilih cluster yang Anda buatLangkah 1: Buat cluster.
3. Pada tab Layanan, pilih Buat.
4. Di bagian Lingkungan, lakukan hal berikut:
    a. Untuk opsi Komputasi, pilih Jenis peluncuran.
    b. Untuk Tipe peluncuran, pilih EC2
5. Di bagian konfigurasi Deployment, lakukan hal berikut:
    a. Untuk Keluarga, pilih definisi tugas yang Anda buatLangkah 2: Daftarkan definisi tugas.
    b. Untuk nama Layanan, masukkan nama untuk layanan Anda.
    c. Untuk tugas yang diinginkan, masukkan 1.
6. Tinjau opsi dan pilih Buat.
7. Pilih Lihat layanan untuk meninjau layanan Anda.

Langkah 4: Lihat Layanan Anda

Layanan ini adalah aplikasi berbasis web sehingga Anda dapat melihat kontainer dengan peramban web.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pilih cluster tempat Anda menjalankan layanan.
4. Di tab Layanan, di bawah Nama layanan, pilih layanan yang Anda buatLangkah 3: Buat Layanan.
5. Pilih tab Tugas, lalu pilih tugas di layanan Anda.
6. Pada halaman tugas, di bagian Konfigurasi, di bawah IP Publik, pilih Buka alamat. Screen shot di bawah ini adalah output yang diharapkan.

Langkah 5: Bersihkan

Setelah selesai menggunakan kluster Amazon ECS, Anda harus membersihkan sumber daya yang terkait dengannya untuk menghindari biaya untuk sumber daya yang tidak Anda gunakan.

Beberapa sumber daya Amazon ECS, seperti tugas, layanan, cluster, dan instans kontainer, dibersihkan menggunakan konsol Amazon ECS. Sumber daya lain, seperti instans Amazon EC2, penyeimbang beban Elastic Load Balancing, dan grup Auto Scaling, harus dibersihkan secara manual di konsol Amazon EC2 atau dengan menghapus tumpukan yang membuatnya. AWS CloudFormation

1. Pada panel navigasi, silakan pilih Klaster.
2. Pada halaman Clusters, pilih cluster cluster yang Anda buat untuk tutorial ini.

3. Pilih tab Layanan.
4. Pilih layanan, lalu pilih Hapus.
5. Pada prompt konfirmasi, masukkan hapus dan kemudian pilih Hapus.

   Tunggu hingga layanan dihapus.
6. Pilih Hapus klaster. Pada prompt konfirmasi, masukkan hapus ***`nama cluster`***, lalu pilih Hapus. Menghapus cluster membersihkan sumber daya terkait yang dibuat dengan cluster, termasuk grup Auto Scaling, VPC, atau load balancer.

Ikhtisar alat pengembang Amazon ECS

Baik Anda bagian dari perusahaan besar atau startup, Amazon ECS menawarkan berbagai alat yang dapat membantu Anda menyiapkan dan menjalankan wadah dengan cepat, terlepas dari tingkat keahlian Anda. Anda dapat bekerja dengan Amazon ECS dengan cara berikut.

- Pelajari, kembangkan, kelola, dan visualisasikan aplikasi dan layanan container Anda menggunakan AWS Management Console.
- Lakukan tindakan spesifik ke sumber daya Amazon ECS dengan penerapan otomatis melalui pemrograman atau skrip menggunakanAWS Command Line Interface, SDK AWS atau ECS API.
- Tentukan dan kelola semua sumber daya AWS di lingkungan Anda dengan deployment otomatis menggunakan AWS CloudFormation.
- Gunakan alur kerja AWSCopilot CLI end-to-end pengembang lengkap untuk membuat, merilis, dan mengoperasikan aplikasi kontainer yang sesuai dengan praktik AWS terbaik untuk infrastruktur.
- Menggunakan bahasa pemrograman pilihan Anda, tentukan infrastruktur atau arsitektur sebagai kode dengan AWS CDK.
- Kontainerisasi aplikasi yang di-host di tempat atau di instans Amazon EC2 atau keduanya dengan menggunakan App2Container AWS portabilitas terintegrasi dan ekosistem perkakas untuk kontainer.
- Menerapkan aplikasi ke Amazon ECS atau uji container lokal dengan container yang berjalan di Amazon ECS menggunakan format file Docker Compose dengan file. Amazon ECS CLI
- Luncurkan kontainer dari Integrasi Docker Desktop dengan Amazon ECS menggunakan Amazon ECS di Docker Desktop.

AWS Management Console

AWS Management ConsoleIni adalah antarmuka berbasis browser untuk mengelola sumber daya Amazon ECS. Konsol ini memberikan gambaran visual layanan, sehingga mudah untuk menjelajahi fitur dan fungsi Amazon ECS tanpa perlu menggunakan alat tambahan. Banyak tutorial dan panduan terkait tersedia yang dapat memandu Anda menggunakan konsol tersebut.

Untuk tutorial panduan menggunakan konsol tersebut, lihat Memulai dengan Amazon ECS.

Saat memulai, banyak pelanggan lebih suka menggunakan konsol karena memberikan umpan balik visual instan tentang apakah tindakan yang mereka ambil berhasil. AWSpelanggan yang akrab

denganAWS Management Console, dapat dengan mudah mengelola sumber daya terkait seperti penyeimbang beban dan instans Amazon EC2.

Mulai dengan AWS Management Console.

AWS Command Line Interface

AWS Command Line Interface (AWS CLI) adalah alat terpadu yang dapat Anda gunakan untuk mengelola layanan AWS Anda. Hanya dengan satu alat ini saja, Anda dapat mengendalikan beberapa layanan AWS dan mengotomatisasi layanan ini melalui skrip. Perintah Amazon ECS di dalamnya AWS CLI adalah cerminan dari Amazon ECS API.

AWS menyediakan dua set alat baris perintah: AWS Command Line Interface (AWS CLI) dan AWS Tools for Windows PowerShell. Untuk informasi selengkapnya, lihat Panduan Pengguna AWS Command Line Interface dan Panduan Pengguna.AWS Tools for Windows PowerShell

AWS CLIIni cocok untuk pelanggan yang lebih suka dan terbiasa membuat skrip dan berinteraksi dengan alat baris perintah dan tahu persis tindakan mana yang ingin mereka lakukan pada sumber daya Amazon ECS mereka. AWS CLIIni juga membantu pelanggan yang ingin membiasakan diri dengan Amazon ECS API. Pelanggan dapat menggunakannya AWS CLI untuk melakukan sejumlah operasi pada sumber daya Amazon ECS, termasuk operasi Buat, Baca, Perbarui, dan Hapus, langsung dari antarmuka baris perintah.

Gunakan AWS CLI jika Anda ingin atau ingin terbiasa dengan Amazon ECS API dan perintah CLI yang sesuai dan ingin menulis skrip otomatis dan melakukan tindakan spesifik pada sumber daya Amazon ECS.

AWS CloudFormation

AWS CloudFormationdan Terraform untuk Amazon ECS keduanya menyediakan cara ampuh bagi Anda untuk mendefinisikan infrastruktur Anda sebagai kode. Anda dapat dengan mudah melacak versi templat atau tumpukan AWS CloudFormation mana yang berjalan setiap saat dan melakukan rollback ke versi sebelumnya jika diperlukan. Anda dapat melakukan deployment infrastruktur dan aplikasi dengan cara otomatis yang sama. Fleksibilitas dan otomatisasi inilah yang membuat AWS CloudFormation dan Terraform dua format populer untuk menyebarkan beban kerja ke Amazon ECS dari jalur pengiriman berkelanjutan.

Untuk informasi selengkapnya tentang AWS CloudFormation, lihat Membuat sumber daya Amazon ECS dengan AWS CloudFormation.

Gunakan AWS CloudFormation atau Terraform jika Anda ingin mengotomatiskan penerapan infrastruktur dan aplikasi di Amazon ECS dan secara eksplisit menentukan serta mengelola semua sumber daya di lingkungan Anda. AWS

AWSCopilot CLI

AWSCopilot CLI (antarmuka baris perintah) adalah alat komprehensif yang memungkinkan pelanggan untuk menyebarkan dan mengoperasikan aplikasi yang dikemas dalam wadah dan lingkungan di Amazon ECS langsung dari kode sumber mereka. Saat menggunakan AWS Copilot Anda dapat melakukan operasi ini tanpa memahami dan elemen AWS Amazon ECS seperti Application Load Balancers, subnet publik, tugas, layanan, dan cluster. AWS Copilot menciptakan AWS sumber daya atas nama Anda dari pola layanan yang berpendirian, seperti layanan web yang seimbang beban atau layanan backend, menyediakan lingkungan produksi langsung untuk aplikasi kontainer. Anda dapat men-deploy melalui alur AWS CodePipeline di beberapa lingkungan, akun, atau Wilayah, yang semuanya dapat dikelola dalam CLI. Dengan menggunakan AWS Copilot Anda juga dapat melakukan tugas-tugas operator, seperti melihat log dan kesehatan layanan Anda. AWS Copilot adalah all-in-one alat yang membantu Anda mengelola sumber daya cloud dengan lebih mudah sehingga Anda dapat fokus mengembangkan dan mengelola aplikasi Anda.

Untuk informasi selengkapnya, lihat Menggunakan antarmuka baris perintah AWS Copilot.

Gunakan alur kerja end-to-end pengembang lengkap AWS Copilot untuk membuat, merilis, dan mengoperasikan aplikasi kontainer yang sesuai dengan praktik AWS terbaik untuk infrastruktur.

AWS CDK

AWS Cloud Development Kit (AWS CDK)Ini adalah kerangka pengembangan perangkat lunak open source yang dapat Anda gunakan untuk memodelkan dan menyediakan sumber daya aplikasi cloud Anda menggunakan bahasa pemrograman yang sudah dikenal. AWS CDKmenyediakan sumber daya Anda dengan cara yang aman dan berulang. AWS CloudFormation Dengan menggunakan CDK, pelanggan dapat menghasilkan lingkungan mereka dengan lebih sedikit baris kode menggunakan bahasa yang sama yang mereka gunakan untuk membangun aplikasi mereka. Amazon ECS menyediakan modul dalam CDK yang diberi nama`aecs-patterns`, yang menciptakan arsitektur umum. Pola yang tersedia adalah `ApplicationLoadBalancedFargateService()`. Pola ini membuat cluster, definisi tugas, dan sumber daya tambahan untuk menjalankan layanan Amazon ECS yang seimbang beban. AWS Fargate

Untuk informasi selengkapnya, lihat Memulai Amazon ECS menggunakan AWS CDK.

Gunakan AWS CDK jika Anda ingin menentukan infrastruktur atau arsitektur sebagai kode dalam bahasa pemrograman pilihan Anda. Misalnya, Anda dapat menggunakan bahasa yang sama yang Anda gunakan untuk menulis aplikasi Anda.

App2Container AWS

Terkadang pelanggan korporasi mungkin sudah memiliki aplikasi yang di-hosting on premise atau pada instans EC2 atau keduanya. Mereka tertarik pada ekosistem portabilitas dan perkakas kontainer khusus di Amazon ECS, dan perlu mengkontainerisasi terlebih dahulu. AWS App2Container memungkinkan Anda melakukan hal itu. App2Container (A2C) adalah alat baris perintah untuk memodernisasi aplikasi NET dan Java ke dalam aplikasi kontainer. A2C menganalisis dan membangun inventori semua aplikasi yang berjalan dalam mesin virtual, on premise atau cloud. Setelah Anda memilih aplikasi yang ingin Anda simpan, A2C mengemas artifact aplikasi dan dependensi yang teridentifikasi ke dalam citra kontainer. Kemudian mengkonfigurasi port jaringan dan menghasilkan tugas Amazon ECS. Terakhir, itu membuat CloudFormation template yang dapat Anda gunakan atau modifikasi jika diperlukan.

Untuk informasi selengkapnya, lihat Memulai dengan App2Container AWS.

Gunakan App2Container jika Anda memiliki aplikasi yang di-host di tempat atau di instans Amazon EC2 atau keduanya.

Amazon ECS CLI

Amazon ECS CLI memungkinkan Anda menjalankan aplikasi di Amazon ECS AWS Fargate dan menggunakan format file Docker Compose. Anda dapat dengan cepat menyediakan sumber daya, mendorong dan menarik gambar menggunakan Amazon ECR, dan memantau aplikasi yang sedang berjalan di Amazon ECS atau. AWS Fargate Anda juga dapat menguji kontainer yang berjalan secara lokal bersama dengan kontainer di cloud dalam CLI.

Untuk informasi selengkapnya, lihat Menggunakan antarmuka baris perintah Amazon ECS.

Gunakan CLI ECS jika Anda memiliki aplikasi Compose dan ingin menerapkannya ke Amazon ECS, atau uji container lokal dengan container yang berjalan di Amazon ECS di cloud.

Integrasi Docker Desktop dengan Amazon ECS

AWSdan Docker telah berkolaborasi untuk membuat pengalaman pengembang yang disederhanakan yang dapat Anda gunakan untuk menyebarkan dan mengelola kontainer di

Amazon ECS secara langsung menggunakan alat Docker. Sekarang Anda dapat membuat dan menguji container Anda secara lokal menggunakan Docker Desktop dan Docker Compose, lalu menerapkannya ke Amazon ECS di Fargate. Untuk memulai integrasi Amazon ECS dan Docker, unduh Docker Desktop dan opsional mendaftar untuk ID Docker. Untuk informasi selengkapnya, lihat Desktop Docker dan pendaftaran ID Docker.

Pemula untuk kontainer sering mulai belajar tentang kontainer dengan menggunakan alat Docker seperti Docker CLI dan Docker Compose. Hal ini menjadikan penggunaan plugin CLI Docker Compose untuk Amazon ECS sebagai langkah alami berikutnya dalam menjalankan container setelah pengujian secara lokal. AWS Docker menyediakan panduan tentang penerapan kontainer di Amazon ECS. Untuk informasi selengkapnya, lihat Docker Compose CLI - Amazon ECS.

Anda dapat memanfaatkan fitur Amazon ECS tambahan, seperti penemuan layanan, penyeimbangan beban, dan AWS sumber daya lainnya untuk digunakan dengan aplikasi mereka dengan Docker Desktop.

Anda juga dapat mengunduh plugin Docker Compose CLI untuk Amazon ECS langsung dari. GitHub Untuk informasi selengkapnya, lihat Plugin Docker Compose CLI untuk Amazon ECS di. GitHub

SDK AWS

Anda juga dapat menggunakan AWS SDK untuk mengelola sumber daya dan operasi Amazon ECS dari berbagai bahasa pemrograman. SDK menyediakan modul untuk membantu mengurus tugas, termasuk tugas dalam daftar berikut.

- Secara kriptografi menandatangani permintaan layanan Anda
- Mencoba kembali permintaan
- Menangani respons kesalahan

Untuk informasi tentang SKD yang tersedia, lihat Alat untuk Amazon Web Services.

Ringkasan

Dengan banyaknya opsi, Anda dapat memilih opsi yang paling cocok untuk Anda. Pertimbangkan opsi berikut.

- Jika Anda berorientasi visual, Anda dapat secara visual membuat dan mengoperasikan kontainer menggunakan AWS Management Console.

- Jika Anda lebih suka CLI, pertimbangkan untuk menggunakan AWS Copilot atau. AWS CLI Atau, jika Anda lebih suka ekosistem Docker, Anda dapat memanfaatkan fungsi ECS dari dalam Docker CLI untuk men-deploy ke AWS. Setelah sumber daya ini di-deploy, Anda dapat terus mengelolanya melalui CLI atau secara visual melalui konsol.
- Jika Anda adalah seorang developer, Anda dapat menggunakan AWS CDK untuk menentukan infrastruktur Anda dalam bahasa yang sama dengan aplikasi Anda. Anda dapat menggunakan CDK dan AWS Copilot untuk mengekspor ke CloudFormation templat tempat Anda dapat mengubah pengaturan granular, menambahkan AWS sumber daya lain, dan mengotomatiskan penerapan melalui skrip atau pipa CI/CD seperti. AWS CodePipeline

API AWS CLI, SDK, atau ECS adalah alat yang berguna untuk mengotomatisasi tindakan pada sumber daya ECS, sehingga ideal untuk deployment. Untuk men-deploy aplikasi menggunakan AWS CloudFormation Anda dapat menggunakan berbagai bahasa pemrograman atau file teks sederhana untuk memodelkan dan menyediakan semua sumber daya yang diperlukan untuk aplikasi Anda. Anda kemudian dapat men-deploy aplikasi Anda di beberapa Wilayah dan akun secara otomatis dan aman. Misalnya, Anda dapat menentukan cluster ECS, layanan, definisi tugas, atau penyedia kapasitas, sebagai kode dalam file dan menyebarkan melalui perintah. AWS CLI CloudFormation

Untuk melakukan tugas operasi, Anda dapat melihat dan mengelola sumber daya pemrograman menggunakan API AWS CLI, SDK, atau ECS. Perintah seperti `describe-tasks` atau `list-services` menampilkan metadata terbaru atau daftar semua sumber daya. Hampir sama dengan deployment, pelanggan juga dapat menulis otomatisasi yang mencakup perintah seperti `update-service` untuk memberikan tindakan korektif pada deteksi sumber daya yang berhenti secara tiba-tiba. Anda juga dapat mengoperasikan layanan Anda menggunakan AWS Copilot. Perintah seperti `copilot svc logs` atau `copilot app show` memberikan detail tentang masing-masing layanan mikro, atau tentang aplikasi Anda secara keseluruhan.

Pelanggan dapat menggunakan salah satu alat yang tersedia yang disebutkan dalam dokumen ini dan menggunakannya dalam berbagai kombinasi. Alat ECS menawarkan berbagai cara agar bisa lulus menggunakan alat tertentu untuk menggunakan alat lain yang sesuai dengan kebutuhan perubahan Anda. Misalnya, Anda dapat memilih pada pengendalian terperinci daripada sumber daya atau lebih pada otomatisasi sebagaimana diperlukan. ECS juga menawarkan berbagai macam alat untuk berbagai kebutuhan dan tingkat keahlian.

Menggunakan antarmuka baris perintah AWS Copilot

Perintah AWS Copilot command line interface (CLI) menyederhanakan pembuatan, pelepasan, dan pengoperasian aplikasi kontainer siap produksi di Amazon ECS dari lingkungan pengembangan lokal. AWSCopilot CLI selaras dengan alur kerja pengembang yang mendukung praktik terbaik aplikasi modern: mulai dari menggunakan infrastruktur sebagai kode hingga membuat pipeline CI/CD yang disediakan atas nama pengguna. Gunakan AWS Copilot CLI sebagai bagian dari siklus pengembangan dan pengujian sehari-hari Anda sebagai alternatif. AWS Management Console

AWSCopilot saat ini mendukung sistem Linux, macOS, dan Windows. Untuk informasi selengkapnya tentang AWS CLI Copilot versi terbaru, lihat Rilis.

> **Note**
>
> Kode sumber untuk AWS Copilot CLI tersedia di. GitHub Dokumentasi CLI terbaru tersedia di situs web AWS Copilot. Kami merekomendasikan agar Anda mengirimkan masalah dan menarik permintaan untuk perubahan yang ingin Anda sertakan. Namun, Amazon Web Services saat ini tidak mendukung menjalankan salinan kode AWS Copilot yang dimodifikasi. Laporkan masalah dengan AWS Copilot dengan menghubungkan kami di Gitter atau di GitHubmana Anda dapat membuka masalah, memberikan umpan balik, dan melaporkan bug.

Memasang AWS Copilot CLI

AWSCopilot CLI dapat diinstal pada sistem Linux atau macOS baik dengan menggunakan Homebrew atau dengan mengunduh biner secara manual. Gunakan langkah-langkah berikut dengan metode instalasi pilihan Anda.

Menginstal AWS CLI Copilot menggunakan Homebrew

Perintah berikut digunakan untuk menginstal AWS Copilot CLI pada sistem macOS atau Linux Anda menggunakan Homebrew. Sebelum instalasi, Anda harus sudah menginstal Homebrew. Untuk informasi selengkapnya, lihat Homebrew.

```
brew install aws/tap/copilot-cli
```

Menginstal AWS CLI Copilot secara manual

Sebagai alternatif dari Homebrew, Anda dapat menginstal AWS Copilot CLI secara manual di sistem macOS atau Linux Anda. Gunakan perintah berikut untuk sistem operasi Anda untuk mengunduh biner, menerapkan izin eksekusi pada biner, serta kemudian memverifikasi biner dapat bekerja dengan daftar menu bantuan.

macOS

Untuk macOS:

```
sudo curl -Lo /usr/local/bin/copilot https://github.com/aws/copilot-cli/releases/
latest/download/copilot-darwin \
   && sudo chmod +x /usr/local/bin/copilot \
   && copilot --help
```

Untuk sistem ARM macOS:

```
sudo curl -Lo /usr/local/bin/copilot https://github.com/aws/copilot-cli/releases/
latest/download/copilot-darwin-arm64 \
   && sudo chmod +x /usr/local/bin/copilot \
   && copilot --help
```

Linux

Pada sistem Linux x86 (64-bit):

```
sudo curl -Lo /usr/local/bin/copilot https://github.com/aws/copilot-cli/releases/
latest/download/copilot-linux \
   && sudo chmod +x /usr/local/bin/copilot \
   && copilot --help
```

Pada sistem Linux ARM:

```
sudo curl -Lo /usr/local/bin/copilot https://github.com/aws/copilot-cli/releases/
latest/download/copilot-linux-arm64 \
   && sudo chmod +x /usr/local/bin/copilot \
   && copilot --help
```

Windows

Menggunakan Powershell, jalankan perintah berikut:

```
New-Item -Path 'C:\copilot' -ItemType directory; `
  Invoke-WebRequest -OutFile 'C:\copilot\copilot.exe' https://github.com/aws/
copilot-cli/releases/latest/download/copilot-windows.exe
```

(Opsional) Verifikasi CLI AWS Copilot yang diinstal secara manual menggunakan tanda tangan PGP

Executable AWS Copilot CLI ditandatangani secara kriptografis menggunakan tanda tangan PGP. Tanda tangan PGP dapat digunakan untuk memverifikasi validitas Copilot CLI yang dapat dieksekusi. AWS Gunakan langkah-langkah berikut untuk memverifikasi tanda tangan menggunakan alat GnuPG.

1. Unduh dan instal GnuPG. Untuk informasi selengkapnya, lihat situs web GnuPG.

   macOS

   Kami merekomendasikan menggunakan Homebrew. Instal Homebrew menggunakan instruksi dari situs web mereka. Untuk informasi selengkapnya, lihat Homebrew. Setelah Homebrew diinstal, gunakan perintah berikut dari terminal macOS Anda.

   ```
   brew install gnupg
   ```

   Linux

   Instal gpg menggunakan manajer paket pada rasa Linux Anda.

   Windows

   Unduh penginstal sederhana Windows dari situs web GnuPG dan instal sebagai Administrator. Setelah Anda menginstal GnuPG, tutup dan buka kembali Administrator. PowerShell

   Untuk informasi selengkapnya, lihat Unduhan GnuPG.

2. Verifikasi jalur GnuPG ditambahkan ke jalur lingkungan Anda.

   macOS

   ```
   echo $PATH
   ```

Jika Anda tidak melihat jalur GnuPG di output, jalankan perintah berikut untuk menambahkannya ke jalur.

```
PATH=$PATH:<path to GnuPG executable files>
```

Linux

```
echo $PATH
```

Jika Anda tidak melihat jalur GnuPG di output, jalankan perintah berikut untuk menambahkannya ke jalur.

```
export PATH=$PATH:<path to GnuPG executable files>
```

Windows

```
Write-Output $Env:PATH
```

Jika Anda tidak melihat jalur GnuPG di output, jalankan perintah berikut untuk menambahkannya ke jalur.

```
$Env:PATH += ";<path to GnuPG executable files>"
```

3. Buat file teks biasa lokal.

macOS

Di terminal, masukkan:

```
touch <public_key_filename.txt>
```

Buka file dengan TextEdit.

Linux

Buat file teks di editor teks seperti gedit. Simpan sebagai `public_key_filename.txt`

Windows

Buat file teks di editor teks seperti Notepad. Simpan sebagai `public_key_filename.txt`

4. Tambahkan konten berikut dari kunci publik Amazon ECS PGP dan simpan file.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2

mQINBFq1SasBEADliGcT1NVJ1ydfN8DqebYYe9ne3dt6jqKFmKowLmm6LLGJe7HU
jGtqhCWRDkN+qPpHqdArRgDZAtn2pXY5fEipHgar4CP8QgRnRMO2fl74lmavr4Vg
7K/KH8VHlq2uRw32/B94XLEgRbGTMdWFdKuxoPCttBQaMj3LGn6Pe+6xVWRkChQu
BoQAhjBQ+bEm0kNy0LjNgjNlnL3UMAG56t8E3LANIgGgEnpNsB1UwfWluPoGZoTx
N+6pHBJrKIL/1v/ETU4FXpYw2zvhWNahxeNRnoYj3uycHkeliCrw4kj0+skizBgO
2K7oVX8Oc3j5+ZilhL/qDLXmUCb2az5cMM1mOoF8EKX5HaNuq1KfwJxqXE6NNIcO
lFTrT7QwD5fMNld3FanLgv/ZnIrsSaqJOL6zRSq8O4LN1OWBVbndExk2Kr+5kFxn
5lBPgfPgRj5hQ+KTHMa9Y8Z7yUc64BJiN6F9Nl7FJuSsfqbdkvRLsQRbcBG9qxX3
rJAEhieJzVMEUNl+EgeCkxj5xuSkNU7zw2c3hQZqEcrADLV+hvFJktOz9Gm6xzbq
lTnWWCz4xrIWtuEBA2qE+MlDheVd78a3gIsEaSTfQq0osYXaQbvlnSWOoc1y/5Zb
zizHTJIhLtUyls9WisP2s0emeHZicVMfW61EgPrJAiupgc7kyZvFt4YwfwARAQAB
tCRBbWF6b24gRUNTIDxlY3Mtc2VjdXJpdHlAYW1hem9uLmNvbT6JAhwEEAECAAYF
AlrjL0YACgkQHivRXs0TaQrg1g/+JppwPqHnlVPmv7lessB8I5UqZeD6p6uVpHd7
Bs3pcPp8BV7BdRbs3sPLt5bV1+rkqOlw+0gZ4Q/ue/YbWtOAt4qY0OcEo0HgcnaX
lsB827QIfZIVtGWMhuh94xzm/SJkvngml6KB3YJNnWP61A9qJ37/VbVVLzvcmazA
McWB4HUMNrhd0JgBCo0gIpqCbpJEvUc02Bjn23eEJsS9kC7OUAHyQkVnx4d9UzXF
4OoISF6hmQKIBoLnRrAlj5Qvs3GhvHQ0ThYq0Grk/KMJJX2CSqt7tWJ8gk1n3H3Y
SReRXJRnv7DsDDBwFgT6r5Q2HW1TBUvaoZy5hF6maD09nHcNnvBjqADzeT8Tr/Qu
bBCLzkNSYqqkpgtwv7seoD2P4n1giRvDAOEfMZpVkUr+C252IaH1HZFEz+TvBVQM
Y8OWWxmIJW+J6evjo3N1eO19UHv71jvoF8zljbI4bsL2c+QTJmOv7nRqzDQgCWyp
Id/v2dUVVTk1j9omuLBBwNJzQCB+72LcIzJhYmaP1HC4LcKQG+/f41exuItenatK
lEJQhYtyVXcBlh6Yn/wzNg2NWOwb3vqY/F7m6u9ixAwgtIMgPCDE4aJ86zrrXYFz
N2HqkTSQh77Z8KPKmyGopsmN/reMuilPdINb249nA0dzoN+nj+tTFOYCIaLaFyjs
Z0r1QAOJAjkEEwECACMFAlq1SasCGwMHCwkIBwMCAQYVCAIJCgsEFgIDAQIeAQIX
gAAKCRC86dmkLVF4T9iFEACEnkm1dNXsWUx34R3c0vamHrPxvfkyI1FlEUen8D1h
uX9xy6jCEROHWEp0rjGK4QDPgM93sWJ+s1UAKg214QRVzft0y9/DdR+twApA0fzy
uavIthGd6+03jAAo6udYDE+cZC3P7XBbDiYEWk4XAF9I1JjB8hTZUgvXBL046JhG
eM17+crgUyQeetkiOQemLbsbXQ40Bd9V7zf7XJraFd8VrwNUwNb+9KFtgAsc9rk+
YIT/PEf+YOPysgcxI4sTWghtyCulVnuGoskgDv4v73PALU0ieUrvvQVqWMRvhVx1
0X90J7cC1KOyhlEQQ1aFTgmQjmXexVTwIBm8LvysFK6YXM41KjOrlz3+6xBIm/qe
bFyLUnf4WoiuOplAaJhK9pRY+XEnGNxdtN4D26Kd0F+PLkm3Tr3Hy3b1Ok34FlGr
KVHUq1TZD7cvMnnNKEELTUcKX+1mV3an16nmAg/my1JSUt6BNK2rJpY1s/kkSGSE
XQ4zuF2IGCpvBFhYAlt5Un5zwqkwwQR3/n2kwAoDzonJcehDw/C/cGos5D0aIU7I
K2X2aTD3+pA7Mx3IMe2hqmYqRt9X42yF1PIEVRneBRJ3HDezAgJrNh0GQWRQkhIx
gz6/cTR+ekr5TptVszS9few2GpI5bCgBKBisZIssT89aw7mAKWut0Gcm4qM9/yK6
1bkCDQRatUmrARAAxNPvVwreJ2yAiFcUpdRlVhsuOgnxvs1QgsIw3H7+Pacr9Hpe
8uftYZqdC82KeSKhpHq7c8gMTMucIINtH25x9BCc73E33EjCL9Lqov1TL7+QkgHe
T+JIhZwdD8Mx2K+LVVVu/aWkNrfMuNwyDUciSI4D5QHa8T+F8fgN4OTpwYjirzel
```

```
5yoICMr9hVcbzDNv/ozKCxjx+XKgnFc3wrnDfJfntfDAT7ecwbUTL+viQKJ646s+
psiqXRYtVvYInEhLVrJ0aV6zHFoigE/Bils6/g7ru1Q6CEHqEw++APs5CcE8VzJu
WAGSVHZgun5Y9N4quR/M9Vm+IPMhTxrAg7rOvyRN9cAXfeSMf77I+XTifigNna8x
t/MOdjXr1fjF4pThEi5u6WsuRdFwjY2azEv3vevodTi4HoJReH6dFRa6y8c+UDgl
2iHiOKIpQqLbHEfQmHcDd2fix+AaJKMnPGNku9qCFEMbgSRJpXz6BfwnY1QuKE+I
R6jA0frUNt2jhiGG/F8RceXzohaaC/Cx7LUCUFWc0n7z32C9/Dtj7I1PMOacdZzz
bjJzRKO/ZDv+UN/c9dwAkllzAyPMwGBkUaY68EBstnIliW34aWm6IiHhxioVPKSp
VJfyiXPO0EXqujtHLAeChfjcns3I12YshT1dv2PafG53fp33ZdzeUgsBo+EAEQEA
AYkCHwQYAQIACQUCWrVJqwIbDAAKCRC86dmkLVF4T+ZdD/9x/8APzgNJF3o3STrF
jvnV1ycyhWYGAeBJiu7wjsNWwzMFOv15tLjB7AqeVxZn+WKDD/mIOQ45OZvnYZuy
X7DR0JszaH9wrYTxZLVruAu+t6UL0y/XQ4L1GZ9QR6+r+7t1Mvbfy7BlHbvX/gYt
Rwe/uwdibI0CagEzyX+2D3kTOlHO5XThbXaNf8AN8zha91Jt2Q2UR2X5T6JcwtMz
FBvZnl3LSmZyE0EQehS2iUurU4uWOpGppuqVnbi0jbCvCHKgDGrqZ0smKNAQng54
F365W3g8AfY48s8XQwzmcliowYX9bT8PZiEi0J4QmQh0aXkpqZyFefuWeOL2R94S
XKzr+gRh3BAULoqF+qK+IUMxTip9KTPNvYDpiC66yBiT6gFDji5Ca9pGpJXrC3xe
TXiKQ8DBWDhBPVPrruLIaenTtZEOsPc4I85yt5U9RoPTStcOr34s3w5yEaJagt6S
Gc5r9ysjkfH6+6rbi1ujxMgROSqtqr+RyB+V9A5/OgtNZc8llK6u4UoOCde8jUUW
vqWKvjJB/Kz3u4zaeNu2ZyyHaOqOuH+TETcW+jsY9IhbEzqN5yQYGi4pVmDkY5vu
lXbJnbqPKpRXgM9BecV9AMbPgbDq/5LnHJJXg+G8YQOgp4lR/hC1TEFdIp5wM8AK
CWsENyt2o1rjgMXiZOMF8A5oBLkCDQRatUuSARAAr77kj7j2QR2SZeOSlFBvV7oS
mFeSNnz9xZssqrsm6bTwSHM6YLDwc7Sdf2esDdyzONETwqrVCg+FxgL8hmo9hS4c
rR6tmrP0mOmptr+xLLsKcaP7ogIXsyZnrEAEsvW8PnfayoiPCdc3cMCR/lTnHFGA
7EuR/XLBmi7Qg9tByVYQ5Yj5wB9V4B2yeCt3XtzPqeLKvaxl7PNelaHGJQY/xo+m
V0bndxf9IY+4oFJ4blD32WqvyxESo7vW6WBh7oqv3Zbm0yQrr8a6mDBpqLkvWwNI
3kpJR974tg5o5LfDu1BeeyHWPSGm4U/G4JB+JIG1ADy+RmoWEt4BqTCZ/knnoGvw
D5sTCxbKdmuOmhGyTssoG+3OOcGYHV7pWYPhazKHMPm201xKCjH1RfzRULzGKjD+
yMLT1I3AXFmLmZJXikAOlvE3/wgMqCXscbycbLjLD/bXIuFWo3rzoezeXjgi/DJx
jKBAyBTYO5nMcth1O9oaFd9d0HbsOUDkIMnsgGBE766Piro6MHo0T0rXl07Tp4pI
rwuSOsc6XzCzdImj0Wc6axS/HeUKRXWdXJwno5awTwXKRJMXGfhCvSvbcbc2Wx+L
IKvmB7EB4K3fmjFFE67yolmiw2qRcUBfygtH3eL5XZU28MiCpue8Y8GKJoBAUyvf
KeM1rO8Jm3iRAc5a/D0AEQEAAYkEPgQYAQIACQUCWrVLkgIbAgIpCRC86dmkLVF4
T8FdIAQZAQIABgUCWrVLkgAKCRDePL1hra+LjtHYD/9MucxdFe6bXO1dQR4tKhhQ
P0LRqy6zlBY9ILCLowNdGZdqorogUiUymgn3VhEhVtxTOoHcN7qOuM01PNsRnOeS
EYjf8Xrb1clzkD6xULwmOclTb9bBxnBc/4PFvHAbZW3QzusaZniNgkuxt6BTfloS
Of4inq71kjmGK+TlzQ6mUMQUg228NUQC+a84EPqYyAeY1sgvgB7hJBhYL0QAxhcW
6m20Rd8iEc6HyzJ3yCOCsKip/nRWAbf0OvfHfRBp0+m0ZwnJM8cPRFjOqqzFpKH9
HpDmTrC4wKP1+TL52LyEqNh4yZitXmZNV7giSRIkk0eDSko+bFy6VbMzKUMkUJK3
D3eHFAMkujmbfJmSMTJOPGn5SB1HyjCZNx6bhIIbQyEUB9gKCmUFaqXKwKpF6rj0
iQXAJxLR/shZ5Rk96VxzOphUl7T90m/PnUEEPwq8KsBhnMRgxa0RFidDP+n9fgtv
HLmrOqX9zBCVXh0mdWYLrWvmzQFWzG7AoE55fkf8nAEPsalrCdtaNUBHRXA0OQxG
AHMOdJQQvBsmqMvuAdjkDWpFu5y0My5ddU+hiUzUyQLjL5Hhd5LOUDdewlZgIw1j
xrEAUzDKetnemM8GkHxDgg8koev5frmShJuce7vSjKpCNg3EIJSgqMOPFjJuLWtZ
vjHeDNbJy6uNL65ckJy6WhGjEADS2WAW1D6Tfekkc21SsIXk/LqEpLMR/0g5OUif
wcEN1rS9IJXBwIy8MelN9qr5KcKQLmfdfBNEyyceBhyVl0MDyHOKC+7PofMtkGBq
```

```
13QieRHv5GJ8LB3fclqHV8pwTTo3Bc8z2g0TjmUYAN/ixETdReDoKavWJYSE9yoM
aaJu279ioVTrwpECse0XkiRyKToTjwOb73CGkBZZpJyqux/rmCV/fp4ALdSW8zbz
FJVORaivhoWwzjpfQKhwcU9lABXi2UvVm14v0AfeI7oiJPSU1zM4fEny4oiIBXlR
zhFNih1UjIu82X16mTm3BwbIga/s1fnQRGzyhqUIMii+mWra23EwjChaxpvjjcUH
5ilLc5Zq781aCYRygYQw+hu5nFkOH1R+Z50Ubxjd/aqUfnGIAX7kPMD3Lof4KldD
Q8ppQriUvxVo+4nPV6rpTy/PyqCLWDjkguHpJsEFsMkwajrAz0QNSAU5CJ0G2Zu4
yxvYlumHCEl7nbFrm0vIiA75Sa8KnywTDsyZsu3XcOcf3g+g1xWTpjJqy2bYXlqz
9uDOWtArWHOis6bq8l9RE6xr1RBVXS6uqgQIZFBGyq66b0dIq4D2JdsUvgEMaHbc
e7tBfeB1CMBdA64e9Rq7bFR7Tvt8gasCZYlNr3lydh+dFHIEkH53HzQe6l88HEic
+0jVnLkCDQRa55wJARAAyLya2Lx6gyoWoJN1a6740q3o8e9d4KggQOfGMTCflmeq
ivuzgN+3DZHN+9ty2KxXMtn0mhHBerZdbNJyjMNT1gAgrhPNB4HtXBXum2wS57WK
DNmade914L7FWTPAWBG2Wn448OEHTqsClICXXWy9IICgclAEyIq0Yq5mAdTEgRJS
Z8t4GpwtDL9gNQyFXaWQmDmkAsCygQMvhAlmu9xOIzQG5CxSnZFk7zcuL60k14Z3
Cmt49k4T/7ZU8goWi8tt+rU78/IL3J/fF9+1civ1OwuUidgfPCSvOUW1JojsdCQA
L+RZJcoXq7lfOFj/eNjeOSstCTDPfTCL+kThE6E5neDtbQHBYkEX1BRiTedsV4+M
ucgiTrdQFWKf89G72xdv8ut9AYYQ2BbEYU+JAYhUH8rYYui2dHKJIgjNvJscuUWb
+QEqJIRleJRhrO+/CHgMs4fZAkWF1VFhKBkcKmEjLn1f7EJJUUW84ZhKXjO/AUPX
1CHsNjziRceuJCJYox1cwsoq6jTE50GiNzcIxTn9xUc0UMKFeggNAFys1K+TDTm3
Bzo8H5ucjCUEmUm9lhkGwqTZgOlRX5eqPX+JBoSaObqhgqCa5IPinKRa6MgoFPHK
6sYKqroYwBGgZm6Js5chpNchvJMs/3WXNOEVg0J3z3vP0DMhxqWm+r+n9zlW8qsA
EQEAAYkEPgQYAQgACQUCWuecCQIbAgIpCRC86dmkLVF4T8FdIAQZAQgABgUCWuec
CQAKCRBQ3szEcQ5hr+ykD/4tOLRHFHXuKUcxgGaubUcVtsFrwBKma1cYjqaPms8u
6Sk0wfGRI32G/GhOrp0Ts/MOkbObq6VLTh8N5Yc/53MEl8zQFw9Y5AmRoW4PZXER
ujs5s7p4oR7xHMihMjCCBn1bvrR+34YPfgzTcgLiOEFHYT8UTxwnGmXOvNkMM7md
xD3CV5q6VAte8WKBo/220II3fcQlc9r/oWX4kXXkb0v9hoGwKbDJ1tzqTPrp/xFt
yohqnvImpnlz+Q9zXmbrWYL9/g8VCmW/NN2gju2G3Lu/TlFUWIT4v/5OPK6TdeNb
VKJO4+S8bTayqSG9CML1S57KSgCo5HUhQWeSNHI+fpe5oX6FALPT9JLDce8OZz1i
cZZ0MELP37mOOQun0AlmHm/hVzf0f311PtbzcqWaE51tJvgUR/nZFo6Ta3O5Ezhs
3VlEJNQ1Ijf/6DH87SxvAoRIARCuZd0qxBcDK0avpFzUtbJd24lRA3WJpkEiMqKv
RDVZkE4b6TW61f0o+LaVfK6E8oLpixegS4fiqC16mFrOdyRk+RJJfIUyz0WTDVmt
g0U1CO1ezokMSqkJ7724pyjr2xf/r9/sC6aOJwB/lKgZkJfC6NqL7TlxVA31dUga
LEOvEJTTE4gl+tYtfsCDvALCtqL0jduSkUo+RXcBItmXhA+tShW0pbS2Rtx/ixua
KohVD/0R4QxiSwQmICNtm9mw9ydIl1yjYXX5a9x4wMJracNY/LBybJPFnZnT4dYR
z4XjqysDwvvYZByaWoIe3QxjX84V6MlI2IdAT/xImu8gbaCI8tmyfpIrLnPKiR9D
VFYfGBXuAX7+HgPPSFtrHQONCALxxzlbNpS+zxt9r0MiLgcLyspWxSdmoYGZ6nQP
RO5Nm/ZVS+u2imPCRzNUZEMa+dlE6kHx0rS0dPiuJ4O7NtPeYDKkoQtNagspsDvh
cK7CSqAiKMq06UBTxqlTSRkm62eOCtcs3p3OeHu5GRZF1uzTET0ZxYkaPgdrQknx
ozjP5mC7X+45lcCfmcVt94TFNL5HwEUVJpmOgmzILCI8yoDTWzloo+i+fPFsXX4f
kynhE83mSEcr5VHFYrTY3mQXGmNJ3bCLuc/jq7ysGq69xiKmTlUeXFm+aojcRO5i
zyShIRJZ0GZfuzDYFDbMV9amA/YQGygLw//zP5ju5SW26dNxlf3MdFQE5JJ86rn9
MgZ4gcpazHEVUsbZsgkLizRp9imUiH8ymLqAXnfRGlU/LpNSefnvDFTtEIRcpOHc
bhayG0bk51Bd4mioOXnIsKy4j63nJXA27x5EVVHQ1sYRN8Ny4Fdr2tMAmj2O+X+J
qX2yy/UX5nSPU492e2CdZ1UhoU0SRFY3bxKHKB7SDbVeav+K5g==
=Gi5D
```

```
-----END PGP PUBLIC KEY BLOCK-----
```

Rincian kunci publik Amazon ECS PGP untuk referensi:

```
Key ID: BCE9D9A42D51784F
Type: RSA
Size: 4096/4096
Expires: Never
User ID: Amazon ECS
Key fingerprint: F34C 3DDA E729 26B0 79BE AEC6 BCE9 D9A4 2D51 784F
```

Anda dapat menutup editor teks.

5. Impor file dengan kunci publik Amazon ECS PGP dengan perintah berikut di terminal.

```
gpg --import <public_key_filename.txt>
```

6. Unduh tanda tangan AWS Copilot CLI. Tanda tangan tersebut adalah tandatangan PGP ASCII-terpisah yang tersimpan di file dengan ekstensi `.asc`. File standar memiliki nama yang sama dengan pelaksanaan yang sesuai, dengan `.asc` yang telah ditambahkan.

   macOS

   Untuk sistem macOS, jalankan perintah berikut.

   ```
   sudo curl -Lo copilot.asc https://github.com/aws/copilot-cli/releases/latest/download/copilot-darwin.asc
   ```

   Linux

   Untuk sistem Linux x86 (64-bit), jalankan perintah berikut.

   ```
   sudo curl -Lo copilot.asc https://github.com/aws/copilot-cli/releases/latest/download/copilot-linux.asc
   ```

   Untuk sistem Linux ARM, jalankan perintah berikut.

   ```
   sudo curl -Lo copilot.asc https://github.com/aws/copilot-cli/releases/latest/download/copilot-linux-arm64.asc
   ```

Windows

Menggunakan Powershell, jalankan perintah berikut.

```
Invoke-WebRequest -OutFile 'C:\copilot\copilot.asc' https://github.com/aws/copilot-cli/releases/latest/download/copilot-windows.exe.asc
```

7. Verifikasi tanda tangan dengan menggunakan perintah berikut.

   - Untuk sistem macOS dan Linux:

   ```
   gpg --verify copilot.asc /usr/local/bin/copilot
   ```

   - Untuk sistem Windows:

   ```
   gpg --verify 'C:\copilot\copilot.asc' 'C:\copilot\copilot.exe'
   ```

   Keluaran yang diharapkan

   ```
   gpg: Signature made Tue Apr  3 13:29:30 2018 PDT
   gpg:                using RSA key DE3CBD61ADAF8B8E
   gpg: Good signature from "Amazon ECS <ecs-security@amazon.com>" [unknown]
   gpg: WARNING: This key is not certified with a trusted signature!
   gpg:          There is no indication that the signature belongs to the owner.
   Primary key fingerprint: F34C 3DDA E729 26B0 79BE  AEC6 BCE9 D9A4 2D51 784F
        Subkey fingerprint: EB3D F841 E2C9 212A 2BD4  2232 DE3C BD61 ADAF 8B8E
   ```

   Important

   Peringatan dalam output diharapkan dan tidak bermasalah. Itu terjadi karena tidak ada rantai kepercayaan antara kunci PGP pribadi Anda (jika Anda memilikinya) dan kunci Amazon ECS PGP. Untuk informasi selengkapnya, lihat Web kepercayaan.

8. Untuk instalasi Windows, jalankan perintah berikut di Powershell untuk menambahkan direktori AWS Copilot ke jalur.

   ```
   $Env:PATH += ";<path to Copilot executable files>"
   ```

Langkah selanjutnya

Setelah instalasi, pelajari cara menerapkan aplikasi Amazon ECS menggunakan AWS Copilot. Untuk informasi selengkapnya, lihat Memulai Amazon ECS menggunakan AWS Copilot.

Memulai Amazon ECS menggunakan AWS Copilot

Memulai Amazon ECS menggunakan AWS Copilot dengan menerapkan aplikasi Amazon ECS.

Prasyarat

Sebelum memulai, pastikan Anda memenuhi persyaratan berikut:

- Siapkan akun AWS. Untuk informasi selengkapnya, lihat Siapkan untuk menggunakan Amazon ECS.
- Instal AWS CLI Copilot. Rilis saat ini mendukung sistem Linux dan macOS. Untuk informasi selengkapnya, lihat Memasang AWS Copilot CLI.
- Instal dan konfigurasikan AWS CLI. Untuk informasi selengkapnya, lihat AWS Command Line Interface.
- Jalankan `aws configure` untuk mengatur profil default yang akan digunakan AWS CLI Copilot untuk mengelola aplikasi dan layanan Anda.
- Instal dan jalankan Docker. Untuk informasi selengkapnya, lihat Memulai dengan Docker.

Deploy aplikasi Anda menggunakan satu perintah

Pastikan bahwa Anda sudah menginstal alat baris perintah AWS terinstal dan menjalankan `aws configure` sebelum Anda memulai.

Deploy aplikasi menggunakan perintah berikut.

```
git clone https://github.com/aws-samples/amazon-ecs-cli-sample-app.git demo-app && \
cd demo-app &&                                  \
copilot init --app demo                         \
  --name api                                    \
  --type 'Load Balanced Web Service'            \
  --dockerfile './Dockerfile'                   \
  --port 80                                     \
  --deploy
```

Deploy aplikasi Anda langkah demi langkah

Langkah 1: Konfigurasikan kredensial Anda

Jalankan `aws configure` untuk menyiapkan profil default yang digunakan AWS CLI Copilot untuk mengelola aplikasi dan layanan Anda.

```
aws configure
```

Langkah 2: Kloning aplikasi demo

Kloning aplikasi Flask sederhana dan Dockerfile.

```
git clone https://github.com/aws-samples/amazon-ecs-cli-sample-app.git demo-app
```

Langkah 3: Siapkan aplikasi Anda

1. Dari dalam direktori aplikasi demo, jalankan perintah `init`.

   Untuk pengguna Windows, jalankan `init` perintah dari folder yang berisi `copilot.exe` file yang diunduh.

   ```
   copilot init
   ```

   AWSCopilot memandu Anda melalui pengaturan aplikasi dan layanan pertama Anda dengan serangkaian petunjuk terminal, dimulai dengan langkah berikutnya. Jika Anda telah menggunakan AWS Copilot untuk menyebarkan aplikasi, Anda diminta untuk memilih salah satu dari daftar nama aplikasi.
2. Beri nama aplikasi Anda.

   ```
   What would you like to name your application? [? for help]
   ```

   Enter ***demo***.

Langkah 4: Atur Layanan ECS di Aplikasi "demo"

1. Anda diminta untuk memilih tipe layanan. Anda sedang membangun aplikasi Flask sederhana yang melayani API kecil.

```
Which service type best represents your service's architecture? [Use arrows to
 move, type to filter, ? for more help]
  > Load Balanced Web Service
    Backend Service
    Scheduled Job
```

Choose ***Load Balanced Web Service*** .

2. Berikan nama untuk layanan Anda.

```
What do you want to name this Load Balanced Web Service? [? for help]
```

Enter ***api*** untuk nama layanan Anda.

3. Pilih Dockerfile.

```
Which Dockerfile would you like to use for api? [Use arrows to move, type to
 filter, ? for more help]
  > ./Dockerfile
    Use an existing image instead
```

Choose ***Dockerfile***.

Untuk pengguna Windows, masukkan jalur ke Dockerfile di `demo-app folder` (*`... \ demo-aplikasi\ Dockerfile`*\.).

4. Tentukan port.

```
Which port do you want customer traffic sent to? [? for help] (80)
```

Enter ***80*** atau terima default.

5. Anda akan melihat log yang menunjukkan sumber daya aplikasi sedang dibuat.

```
Creating the infrastructure to manage services under application demo.
```

6. Setelah sumber daya aplikasi dibuat, deploy lingkungan pengujian.

```
Would you like to deploy a test environment? [? for help] (y/N)
```

Enter ***y***.

```
Proposing infrastructure changes for the test environment.
```

7. Anda akan melihat log menampilkan status deployment aplikasi Anda.

```
Note: It's best to run this command in the root of your Git repository.
Welcome to the Copilot CLI! We're going to walk you through some questions
to help you get set up with an application on ECS. An application is a collection
 of
containerized services that operate together.

Use existing application: No
Application name: demo
Workload type: Load Balanced Web Service
Service name: api
Dockerfile: ./Dockerfile
no EXPOSE statements in Dockerfile ./Dockerfile
Port: 80
Ok great, we'll set up a Load Balanced Web Service named api in application demo
 listening on port 80.

# Created the infrastructure to manage services under application demo.

# Wrote the manifest for service api at copilot/api/manifest.yml
Your manifest contains configurations like your container size and port (:80).

# Created ECR repositories for service api.

All right, you're all set for local development.
Deploy: Yes

# Created the infrastructure for the test environment.
- Virtual private cloud on 2 availability zones to hold your services
 [Complete]
- Virtual private cloud on 2 availability zones to hold your services
 [Complete]
  - Internet gateway to connect the network to the internet
 [Complete]
  - Public subnets for internet facing services
 [Complete]
  - Private subnets for services that can't be reached from the internet
 [Complete]
```

```
  - Routing tables for services to talk with each other
 [Complete]
- ECS Cluster to hold your services
 [Complete]
# Linked account aws_account_id and region region to application demo.

# Created environment test in region region under application demo.

Environment test is already on the latest version v1.0.0, skip upgrade.
[+] Building 0.8s (7/7) FINISHED
 => [internal] load .dockerignore
                                 0.1s
 => => transferring context: 2B
                                 0.0s
 => [internal] load build definition from Dockerfile
                                 0.0s
 => => transferring dockerfile: 37B
                                 0.0s
 => [internal] load metadata for docker.io/library/nginx:latest
                                 0.7s
 => [internal] load build context
                                 0.0s
 => => transferring context: 32B
                                 0.0s
 => [1/2] FROM docker.io/library/
nginx@sha256:aeade65e99e5d5e7ce162833636f692354c227ff438556e5f3ed0335b7cc2f1b
 0.0s
 => CACHED [2/2] COPY index.html /usr/share/nginx/html
                                 0.0s
 => exporting to image
                                 0.0s
 => => exporting layers
                                 0.0s
 => => writing image
 sha256:3ee02fd4c0f67d7bd808ed7fc73263880649834cbb05d5ca62380f539f4884c4
            0.0s
 => => naming to aws_account_id.dkr.ecr.region.amazonaws.com/demo/api:cee7709
                                 0.0s
WARNING! Your password will be stored unencrypted in /home/user/.docker/
config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

```
The push refers to repository
 [aws_account_id.dkr.ecr.region.amazonaws.com/demo/api]
592a5c0c47f1: Pushed
6c7de695ede3: Pushed
2f4accd375d9: Pushed
ffc9b21953f4: Pushed
cee7709: digest: sha_digest

# Deployed api, you can access it at http://demo-
Publi-1OQ8VMS2VC2WG-561733989.region.elb.amazonaws.com.
```

Langkah 5: Verifikasi aplikasi Anda sedang berjalan

Lihat status aplikasi Anda dengan menggunakan perintah berikut.

Daftar semua aplikasi AWS Copilot Anda.

```
copilot app ls
```

Tampilkan informasi tentang lingkungan dan layanan dalam aplikasi Anda.

```
copilot app show
```

Tampilkan informasi tentang lingkungan Anda.

```
copilot env ls
```

Tampilkan informasi tentang layanan, termasuk titik akhir, kapasitas, dan sumber daya terkait.

```
copilot svc show
```

Cantumkan semua layanan dalam aplikasi.

```
copilot svc ls
```

Tampilkan log dari layanan yang di-deploy.

```
copilot svc logs
```

Tampilkan status layanan.

```
copilot svc status
```

Cantumkan perintah dan opsi yang tersedia.

```
copilot --help
```

```
copilot init --help
```

Langkah 6. Pelajari cara membuat Alur CI/CD

Instruksi dapat ditemukan di Lokakarya ECS yang merinci cara mengotomatiskan sepenuhnya pipa CI/CD dan alur kerja git menggunakan Copilot. AWS

Langkah 7: Bersihkan

Jalankan perintah berikut untuk menghapus dan membersihkan semua sumber daya.

```
copilot app delete
```

Memulai Amazon ECS menggunakan AWS CDK

AWS Cloud Development Kit (AWS CDK)Ini adalah kerangka Infrastructure-as-Code (IAC) yang dapat Anda gunakan untuk mendefinisikan infrastruktur AWS cloud dengan menggunakan bahasa pemrograman pilihan Anda. Untuk menentukan infrastruktur cloud Anda sendiri, pertama-tama Anda menulis aplikasi (dalam salah satu bahasa yang didukung CDK) yang berisi satu atau beberapa tumpukan. Kemudian, Anda mensintesisnya ke AWS CloudFormation template dan menyebarkan sumber daya Anda ke template Anda. Akun AWS Ikuti langkah-langkah dalam topik ini untuk menerapkan server web dalam kontainer dengan Amazon Elastic Container Service (Amazon ECS) Container Service (Amazon ECS) dan di Fargate. AWS CDK

Perpustakaan AWS Konstruksi, disertakan dengan CDK, menyediakan modul yang dapat Anda gunakan untuk memodelkan sumber daya yang Layanan AWS disediakan. Untuk layanan populer, perpustakaan menyediakan konstruksi yang dikuratori dengan default cerdas dan praktik terbaik. Salah satu modul ini, khususnya`aws-ecs-patterns`, menyediakan abstraksi tingkat tinggi yang dapat Anda gunakan untuk menentukan layanan kontainer Anda dan semua sumber daya pendukung yang diperlukan dalam beberapa baris kode.

Topik ini menggunakan `ApplicationLoadBalancedFargateService`konstruksi. Konstruksi ini menyebarkan layanan Amazon ECS di Fargate di belakang penyeimbang beban aplikasi. `aws-ecs-patterns`Modul ini juga mencakup konstruksi yang menggunakan penyeimbang beban jaringan dan berjalan di Amazon EC2.

Sebelum memulai tugas ini, atur lingkungan AWS CDK pengembangan Anda, dan instal AWS CDK dengan menjalankan perintah berikut. Untuk petunjuk tentang cara mengatur lingkungan AWS CDK pengembangan Anda, lihat Memulai Dengan AWS CDK - Prasyarat.

```
npm install -g aws-cdk
```

> **Note**
>
> Instruksi ini mengasumsikan Anda menggunakan AWS CDK v2.

Topik

Langkah 1: Siapkan proyek AWS CDK Anda

Buat direktori untuk aplikasi AWS CDK baru dan inisialisasi proyek tersebut.

TypeScript

```
mkdir hello-ecs
cd hello-ecs
cdk init --language typescript
```

JavaScript

```
mkdir hello-ecs
cd hello-ecs
```

```
cdk init --language javascript
```

Python

```
mkdir hello-ecs
cd hello-ecs
cdk init --language python
```

Setelah proyek dimulai, aktifkan lingkungan virtual proyek dan instal AWS CDK dependensi dasar.

```
source .venv/bin/activate
python -m pip install -r requirements.txt
```

Java

```
mkdir hello-ecs
cd hello-ecs
cdk init --language java
```

Impor proyek Maven ini ke IDE Java Anda. Misalnya, di Eclipse, gunakan File > Import > Maven > Existing Maven Projects.

C#

```
mkdir hello-ecs
cd hello-ecs
cdk init --language csharp
```

> **Note**
>
> Templat aplikasi AWS CDK menggunakan nama direktori proyek untuk memberikan nama pada file dan kelas sumber. Dalam contoh ini, direktori diberi nama`hello-ecs`. Jika Anda menggunakan nama direktori project yang berbeda, aplikasi Anda tidak akan cocok dengan petunjuk ini.

AWS CDKv2 menyertakan konstruksi stabil untuk semua Layanan AWS dalam satu paket yang dipanggil`aws-cdk-lib`. Paket ini diinstal sebagai dependensi ketika Anda menginisialisasi proyek.

Saat bekerja dengan bahasa pemrograman tertentu, paket diinstal saat Anda membangun proyek untuk pertama kalinya. Topik ini mencakup cara menggunakan konstruksi Pola Amazon ECS, yang menyediakan abstraksi tingkat tinggi untuk bekerja dengan Amazon ECS. Modul ini bergantung pada konstruksi Amazon ECS dan konstruksi lainnya untuk menyediakan sumber daya yang dibutuhkan aplikasi Amazon ECS Anda.

Nama-nama yang Anda gunakan untuk mengimpor pustaka ini ke dalam aplikasi CDK Anda mungkin sedikit berbeda tergantung pada bahasa pemrograman yang Anda gunakan. Sebagai referensi, berikut ini adalah nama-nama yang digunakan dalam setiap bahasa pemrograman CDK yang didukung.

TypeScript

```
aws-cdk-lib/aws-ecs
aws-cdk-lib/aws-ecs-patterns
```

JavaScript

```
aws-cdk-lib/aws-ecs
aws-cdk-lib/aws-ecs-patterns
```

Python

```
aws_cdk.aws_ecs
aws_cdk.aws_ecs_patterns
```

Java

```
software.amazon.awscdk.services.ecs
software.amazon.awscdk.services.ecs.patterns
```

C#

```
Amazon.CDK.AWS.ECS
Amazon.CDK.AWS.ECS.Patterns
```

Langkah 2: Gunakan AWS CDK untuk mendefinisikan server web kontainer di Fargate

Gunakan gambar kontainer `amazon-ecs-sample`dari DockerHub. Gambar ini berisi aplikasi web PHP yang berjalan di Amazon Linux 2.

Dalam AWS CDK proyek yang Anda buat, edit file yang berisi definisi tumpukan agar menyerupai salah satu contoh berikut.

> **Note**
>
> Tumpukan adalah unit penyebaran. Semua sumber daya harus dalam tumpukan, dan semua sumber daya yang ada di tumpukan digunakan pada saat yang sama. Jika sumber daya gagal diterapkan, sumber daya lain yang sudah digunakan akan digulirkan kembali. AWS CDKAplikasi dapat berisi beberapa tumpukan, dan sumber daya dalam satu tumpukan dapat merujuk ke sumber daya di tumpukan lain.

TypeScript

Perbarui `lib/hello-ecs-stack.ts` sehingga menyerupai yang berikut ini.

```
import * as cdk from 'aws-cdk-lib';
import { Construct } from 'constructs';
import * as ecs from 'aws-cdk-lib/aws-ecs';
import * as ecsp from 'aws-cdk-lib/aws-ecs-patterns';

export class HelloEcsStack extends cdk.Stack {
  constructor(scope: Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    new ecsp.ApplicationLoadBalancedFargateService(this, 'MyWebServer', {
      taskImageOptions: {
        image: ecs.ContainerImage.fromRegistry('amazon/amazon-ecs-sample'),
      },
      publicLoadBalancer: true
    });
  }
}
```

JavaScript

Perbarui `lib/hello-ecs-stack.js` sehingga menyerupai yang berikut ini.

```
const cdk = require('aws-cdk-lib');
const { Construct } = require('constructs');
const ecs = require('aws-cdk-lib/aws-ecs');
const ecsp = require('aws-cdk-lib/aws-ecs-patterns');

class HelloEcsStack extends cdk.Stack {
  constructor(scope = Construct, id = string, props = cdk.StackProps) {
    super(scope, id, props);

    new ecsp.ApplicationLoadBalancedFargateService(this, 'MyWebServer', {
      taskImageOptions: {
        image: ecs.ContainerImage.fromRegistry('amazon/amazon-ecs-sample'),
      },
      publicLoadBalancer: true
    });
  }
}

module.exports = { HelloEcsStack }
```

Python

Perbarui `hello-ecs/hello_ecs_stack.py` sehingga menyerupai yang berikut ini.

```
import aws_cdk as cdk
from constructs import Construct

import aws_cdk.aws_ecs as ecs
import aws_cdk.aws_ecs_patterns as ecsp

class HelloEcsStack(cdk.Stack):

    def __init__(self, scope: Construct, construct_id: str, **kwargs) -> None:
        super().__init__(scope, construct_id, **kwargs)

        ecsp.ApplicationLoadBalancedFargateService(self, "MyWebServer",
            task_image_options=ecsp.ApplicationLoadBalancedTaskImageOptions(
                image=ecs.ContainerImage.from_registry("amazon/amazon-ecs-sample")),
            public_load_balancer=True
```

```
        )
```

Java

Perbarui `src/main/java/com.myorg/HelloEcsStack.java` sehingga menyerupai yang berikut ini.

```
package com.myorg;

import software.constructs.Construct;
import software.amazon.awscdk.Stack;
import software.amazon.awscdk.StackProps;

import software.amazon.awscdk.services.ecs.ContainerImage;
import
 software.amazon.awscdk.services.ecs.patterns.ApplicationLoadBalancedFargateService;
import
 software.amazon.awscdk.services.ecs.patterns.ApplicationLoadBalancedTaskImageOptions;

public class HelloEcsStack extends Stack {
    public HelloEcsStack(final Construct scope, final String id) {
        this(scope, id, null);
    }

    public HelloEcsStack(final Construct scope, final String id, final StackProps
 props) {
        super(scope, id, props);

        ApplicationLoadBalancedFargateService.Builder.create(this, "MyWebServer")
         .taskImageOptions(ApplicationLoadBalancedTaskImageOptions.builder()
           .image(ContainerImage.fromRegistry("amazon/amazon-ecs-sample"))
           .build())
         .publicLoadBalancer(true)
         .build();
    }
}
```

C#

Perbarui `src/HelloEcs/HelloEcsStack.cs` sehingga menyerupai yang berikut ini.

```
using Amazon.CDK;
using Constructs;
```

```
using Amazon.CDK.AWS.ECS;
using Amazon.CDK.AWS.ECS.Patterns;
namespace HelloEcs
{
    public class HelloEcsStack : Stack
    {
        internal HelloEcsStack(Construct scope, string id, IStackProps props =
 null) : base(scope, id, props)
        {
            new ApplicationLoadBalancedFargateService(this, "MyWebServer",
                new ApplicationLoadBalancedFargateServiceProps
                {
                    TaskImageOptions = new ApplicationLoadBalancedTaskImageOptions
                    {
                        Image = ContainerImage.FromRegistry("amazon/amazon-ecs-
sample")
                    },
                    PublicLoadBalancer = true
                });
        }
    }
}
```

Cuplikan pendek sebelumnya mencakup yang berikut:

- Nama logis layanan:`MyWebServer`.
- Gambar kontainer yang diperoleh dari DockerHub:`amazon/amazon-ecs-sample`.
- Informasi relevan lainnya, seperti fakta bahwa penyeimbang beban memiliki alamat publik dan dapat diakses dari Internet.

AWS CDKAkan membuat semua sumber daya yang diperlukan untuk menyebarkan server web termasuk sumber daya berikut. Sumber daya ini dihilangkan dalam contoh ini.

- Kluster Amazon ECS
- Instans Amazon VPC dan Amazon EC2
- Grup Auto Scaling
- Penyeimbang Beban Aplikasi
- Peran dan kebijakan IAM

Beberapa sumber daya yang disediakan secara otomatis akan dibagikan oleh semua layanan Amazon ECS yang ditentukan dalam tumpukan.

Simpan file sumber, lalu jalankan `cdk synth` perintah di direktori utama aplikasi Anda. AWS CDKMenjalankan aplikasi dan mensintesis AWS CloudFormation template darinya, dan kemudian menampilkan template. Template adalah file YAML sekitar 600 baris. Awal file ditampilkan di sini. Template Anda mungkin berbeda dari contoh ini.

```
Resources:
  MyWebServerLB3B5FD3AB:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      LoadBalancerAttributes:
        - Key: deletion_protection.enabled
          Value: "false"
      Scheme: internet-facing
      SecurityGroups:
        - Fn::GetAtt:
            - MyWebServerLBSecurityGroup01B285AA
            - GroupId
      Subnets:
        - Ref: EcsDefaultClusterMnL3mNNYNVpcPublicSubnet1Subnet3C273B99
        - Ref: EcsDefaultClusterMnL3mNNYNVpcPublicSubnet2Subnet95FF715A
      Type: application
    DependsOn:
      - EcsDefaultClusterMnL3mNNYNVpcPublicSubnet1DefaultRouteFF4E2178
      - EcsDefaultClusterMnL3mNNYNVpcPublicSubnet2DefaultRouteB1375520
    Metadata:
      aws:cdk:path: HelloEcsStack/MyWebServer/LB/Resource
  MyWebServerLBSecurityGroup01B285AA:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Automatically created Security Group for ELB
 HelloEcsStackMyWebServerLB06757F57
      SecurityGroupIngress:
        - CidrIp: 0.0.0.0/0
          Description: Allow from anyone on port 80
          FromPort: 80
          IpProtocol: tcp
          ToPort: 80
      VpcId:
        Ref: EcsDefaultClusterMnL3mNNYNVpc7788A521
```

```
    Metadata:
      aws:cdk:path: HelloEcsStack/MyWebServer/LB/SecurityGroup/Resource
# and so on for another few hundred lines
```

Untuk menyebarkan layanan di AndaAkun AWS, jalankan `cdk deploy` perintah di direktori utama aplikasi Anda. Anda diminta untuk menyetujui kebijakan IAM yang dihasilkan. AWS CDK

Penyebaran memakan waktu beberapa menit di mana AWS CDK menciptakan beberapa sumber daya. Beberapa baris terakhir dari output dari penyebaran termasuk nama host publik penyeimbang beban dan URL server web baru Anda. Mereka adalah sebagai berikut.

```
Outputs:
HelloEcsStack.MyWebServerLoadBalancerDNSXXXXXXX = Hello-MyWeb-ZZZZZZZZZZZZZ-
ZZZZZZZZZZ.us-west-2.elb.amazonaws.com
HelloEcsStack.MyWebServerServiceURLYYYYYYYY = http://Hello-MyWeb-ZZZZZZZZZZZZZ-
ZZZZZZZZZZ.us-west-2.elb.amazonaws.com
```

Langkah 3: Uji server web

Salin URL dari output penyebaran dan tempel ke browser web Anda. Pesan selamat datang berikut dari server web ditampilkan.

Langkah 4: Membersihkan

Setelah Anda selesai dengan server web, akhiri layanan menggunakan CDK dengan menjalankan `cdk destroy` perintah di direktori utama aplikasi Anda. Melakukan hal ini mencegah Anda dari menimbulkan biaya yang tidak diinginkan di masa depan.

Langkah selanjutnya

Untuk mempelajari lebih lanjut tentang cara mengembangkan AWS infrastruktur menggunakanAWS CDK, lihat Panduan AWS CDK Pengembang.

Untuk informasi tentang menulis AWS CDK aplikasi dalam bahasa pilihan Anda, lihat berikut ini:

TypeScript

Bekerja dengan AWS CDK di TypeScript

JavaScript

Bekerja dengan AWS CDK di JavaScript

Python

Bekerja dengan AWS CDK in Python

Java

Bekerja dengan AWS CDK di Jawa

C#

Bekerja dengan AWS CDK di C #

Untuk informasi selengkapnya tentang modul AWS Construct Library yang digunakan dalam topik ini, lihat ikhtisar Referensi AWS CDK API berikut.

- aws-ecs
- aws-ecs-patterns

Membuat sumber daya Amazon ECS dengan AWS CloudFormation

Amazon ECS terintegrasi denganAWS CloudFormation, layanan yang dapat Anda gunakan untuk memodelkan dan menyiapkan AWS sumber daya dengan templat yang Anda tentukan. Dengan cara ini, Anda dapat menghabiskan lebih sedikit waktu untuk membuat dan mengelola sumber daya dan infrastruktur Anda. Dengan menggunakanAWS CloudFormation, Anda dapat membuat template yang menjelaskan semua AWS sumber daya yang Anda inginkan, seperti cluster Amazon ECS tertentu.

Kemudian, AWS CloudFormation mengurus penyediaan dan konfigurasi sumber daya tersebut untuk Anda.

Bila Anda menggunakanAWS CloudFormation, Anda dapat menggunakan kembali template Anda untuk mengatur sumber daya Amazon ECS Anda secara konsisten dan berulang. Anda mendeskripsikan sumber daya Anda satu kali, dan kemudian menyediakan sumber daya yang sama lagi di beberapa Akun AWS danWilayah AWS.

Amazon ECS dan template AWS CloudFormation

Untuk menyediakan dan mengonfigurasi sumber daya untuk Amazon ECS dan layanan terkait, pastikan Anda terbiasa dengan AWS CloudFormationtemplat. AWS CloudFormationtemplate adalah file teks dalam format JSON atau YAMB yang menjelaskan sumber daya yang ingin Anda sediakan di tumpukan AndaAWS CloudFormation. Jika Anda tidak terbiasa dengan format JSON atau YAMB, atau keduanya, Anda dapat menggunakan AWS CloudFormation Designer untuk mulai menggunakan template. AWS CloudFormation Lihat informasi selengkapnya di Apa yang dimaksud dengan Desainer AWS CloudFormation? dalam Panduan Pengguna AWS CloudFormation.

Amazon ECS mendukung pembuatan cluster, definisi tugas, layanan, dan set tugas di. AWS CloudFormation Contoh berikut menunjukkan cara membuat sumber daya dengan template ini menggunakanAWS CLI. Anda juga dapat membuat sumber daya ini menggunakan AWS CloudFormation konsol. Untuk informasi selengkapnya tentang cara membuat sumber daya menggunakan AWS CloudFormation konsol, lihat Panduan AWS CloudFormation Pengguna.

Contoh template

Membuat sumber daya Amazon ECS menggunakan tumpukan terpisah

Contoh berikut menunjukkan cara membuat resource Amazon ECS dengan menggunakan tumpukan terpisah untuk setiap sumber daya.

Definisi tugas Amazon ECS

Anda dapat menggunakan template berikut untuk membuat tugas Fargate Linux.

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
       "ECSTaskDefinition": {
```

```
            "Type": "AWS::ECS::TaskDefinition",
            "Properties": {
                "ContainerDefinitions": [
                    {
                        "Command": [
                            "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon
 ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html && httpd-
foreground\""
                        ],
                        "EntryPoint": [
                            "sh",
                            "-c"
                        ],
                        "Essential": true,
                        "Image": "httpd:2.4",
                        "LogConfiguration": {
                            "LogDriver": "awslogs",
                            "Options": {
                                "awslogs-group": "/ecs/fargate-task-definition",
                                "awslogs-region": "us-east-1",
                                "awslogs-stream-prefix": "ecs"
                            }
                        },
                        "Name": "sample-fargate-app",
                        "PortMappings": [
                            {
                                "ContainerPort": 80,
                                "HostPort": 80,
                                "Protocol": "tcp"
                            }
                        ]
                    }
                ],
                "Cpu": 256,
                "ExecutionRoleArn": "arn:aws:iam::aws_account_id:role/
ecsTaskExecutionRole",
                "Family": "task-definition-cfn",
                "Memory": 512,
                "NetworkMode": "awsvpc",
                "RequiresCompatibilities": [
                    "FARGATE"
```

```
                ],
                "RuntimePlatform": {
                    "OperatingSystemFamily": "LINUX"
                }
            }

        }
        }
    }
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  ECSTaskDefinition:
    Type: 'AWS::ECS::TaskDefinition'
    Properties:
      ContainerDefinitions:
        - Command:
            - >-
              /bin/sh -c "echo '<html> <head> <title>Amazon ECS Sample
              App</title> <style>body {margin-top: 40px; background-color:
              #333;} </style> </head><body> <div
              style=color:white;text-align:center> <h1>Amazon ECS Sample
              App</h1> <h2>Congratulations!</h2> <p>Your application is now
              running on a container in Amazon ECS.</p> </div></body></html>' >
              /usr/local/apache2/htdocs/index.html && httpd-foreground"
          EntryPoint:
            - sh
            - '-c'
          Essential: true
          Image: 'httpd:2.4'
          LogConfiguration:
            LogDriver: awslogs
            Options:
              awslogs-group: /ecs/fargate-task-definition
              awslogs-region: us-east-1
              awslogs-stream-prefix: ecs
          Name: sample-fargate-app
          PortMappings:
            - ContainerPort: 80
              HostPort: 80
              Protocol: tcp
```

```
      Cpu: 256
      ExecutionRoleArn: 'arn:aws:iam::aws_account_id:role/ecsTaskExecutionRole'
      Family: task-definition-cfn
      Memory: 512
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - FARGATE
      RuntimePlatform:
        OperatingSystemFamily: LINUX
```

Klaster-klaster Amazon ECS

Anda dapat menggunakan template berikut untuk membuat cluster kosong.

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "ECSCluster": {
            "Type": "AWS::ECS::Cluster",
            "Properties": {
                "ClusterName": "MyEmptyCluster"
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  ECSCluster:
    Type: 'AWS::ECS::Cluster'
    Properties:
      ClusterName: MyEmptyCluster
```

Membuat beberapa sumber daya Amazon ECS dalam satu tumpukan

Anda dapat menggunakan contoh template berikut untuk membuat beberapa sumber daya Amazon ECS dalam satu tumpukan. Template membuat cluster Amazon ECS yang diberi `CFNCluster` nama. Cluster berisi definisi tugas Fargate Linux yang mengatur server web. Template juga membuat

layanan yang diberi nama `cfn-service` yang meluncurkan dan mempertahankan tugas yang ditentukan oleh definisi tugas. Sebelum Anda menggunakan template ini, pastikan bahwa subnet dan ID grup keamanan dalam layanan `NetworkConfiguration` semuanya milik VPC yang sama dan bahwa grup keamanan memiliki aturan yang diperlukan. Untuk informasi selengkapnya tentang aturan grup keamanan, lihat Aturan grup keamanan di panduan pengguna Amazon VPC.

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "ECSCluster": {
            "Type": "AWS::ECS::Cluster",
            "Properties": {
                "ClusterName": "CFNCluster"
            }
        },
        "ECSTaskDefinition": {
            "Type": "AWS::ECS::TaskDefinition",
            "Properties": {
                "ContainerDefinitions": [
                    {
                        "Command": [
                            "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon
 ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html && httpd-
foreground\""
                        ],
                        "EntryPoint": [
                            "sh",
                            "-c"
                        ],
                        "Essential": true,
                        "Image": "httpd:2.4",
                        "LogConfiguration": {
                            "LogDriver": "awslogs",
                            "Options": {
                                "awslogs-group": "/ecs/fargate-task-definition",
                                "awslogs-region": "us-east-1",
                                "awslogs-stream-prefix": "ecs"
                            }
```

```
                        },
                        "Name": "sample-fargate-app",
                        "PortMappings": [
                            {
                                "ContainerPort": 80,
                                "HostPort": 80,
                                "Protocol": "tcp"
                            }
                        ]
                    }
                ],
                "Cpu": 256,
                "ExecutionRoleArn": "arn:aws:iam::aws_account_id::role/
ecsTaskExecutionRole",
                "Family": "task-definition-cfn",
                "Memory": 512,
                "NetworkMode": "awsvpc",
                "RequiresCompatibilities": [
                    "FARGATE"
                ],
                "RuntimePlatform": {
                    "OperatingSystemFamily": "LINUX"
                }
            }
        },
        "ECSService": {
            "Type": "AWS::ECS::Service",
            "Properties": {
                "ServiceName": "cfn-service",
                "Cluster": {
                    "Ref": "ECSCluster"
                },
                "DesiredCount": 1,
                "LaunchType": "FARGATE",
                "NetworkConfiguration": {
                    "AwsvpcConfiguration": {
                        "AssignPublicIp": "ENABLED",
                        "SecurityGroups": [
                            "sg-abcdef01234567890"
                        ],
                        "Subnets": [
                            "subnet-abcdef01234567890"
                        ]
                    }
```

```
                },
                "TaskDefinition": {
                    "Ref": "ECSTaskDefinition"
                }
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  ECSCluster:
    Type: 'AWS::ECS::Cluster'
    Properties:
      ClusterName: CFNCluster
  ECSTaskDefinition:
    Type: 'AWS::ECS::TaskDefinition'
    Properties:
      ContainerDefinitions:
        - Command:
            - >-
              /bin/sh -c "echo '<html> <head> <title>Amazon ECS Sample
              App</title> <style>body {margin-top: 40px; background-color:
              #333;} </style> </head><body> <div
              style=color:white;text-align:center> <h1>Amazon ECS Sample
              App</h1> <h2>Congratulations!</h2> <p>Your application is now
              running on a container in Amazon ECS.</p> </div></body></html>' >
              /usr/local/apache2/htdocs/index.html && httpd-foreground"
          EntryPoint:
            - sh
            - '-c'
          Essential: true
          Image: 'httpd:2.4'
          LogConfiguration:
            LogDriver: awslogs
            Options:
              awslogs-group: /ecs/fargate-task-definition
              awslogs-region: us-east-1
              awslogs-stream-prefix: ecs
          Name: sample-fargate-app
          PortMappings:
```

```
            - ContainerPort: 80
              HostPort: 80
              Protocol: tcp
      Cpu: 256
      ExecutionRoleArn: 'arn:aws:iam::aws_account_id:role/ecsTaskExecutionRole'
      Family: task-definition-cfn
      Memory: 512
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - FARGATE
      RuntimePlatform:
        OperatingSystemFamily: LINUX
  ECSService:
    Type: 'AWS::ECS::Service'
    Properties:
      ServiceName: cfn-service
      Cluster: !Ref ECSCluster
      DesiredCount: 1
      LaunchType: FARGATE
      NetworkConfiguration:
        AwsvpcConfiguration:
          AssignPublicIp: ENABLED
          SecurityGroups:
            - sg-abcdef01234567890
          Subnets:
            - subnet-abcdef01234567890
      TaskDefinition: !Ref ECSTaskDefinition
```

Menggunakan AWS CLI untuk membuat sumber daya dari template

Perintah berikut membuat tumpukan yang diberi nama `ecs-stack` menggunakan file body template yang diberi nama`ecs-template-body.json`. Pastikan bahwa file body template dalam format JSON atau YAMG. Lokasi file ditentukan dalam `--template-body` parameter. Dalam hal ini, file body template terletak di direktori saat ini.

```
aws cloudformation create-stack \
      --stack-name ecs-stack \
      --template-body file://ecs-template-body.json
```

Untuk memastikan bahwa sumber daya dibuat dengan benar, periksa konsol Amazon ECS atau gunakan perintah berikut:

- Perintah berikut mencantumkan semua definisi tugas.

```
aws ecs list-task-definitions
```

- Perintah berikut mencantumkan semua cluster.

```
aws ecs list-clusters
```

- Perintah berikut mencantumkan semua layanan yang didefinisikan dalam cluster*CFNCluster*. Ganti *CFNCluster* dengan nama cluster tempat Anda ingin membuat layanan.

```
aws ecs list-services \
      --cluster CFNCluster
```

Pelajari selengkapnya tentang AWS CloudFormation

Untuk mempelajari selengkapnya tentang AWS CloudFormation, lihat sumber daya berikut:

- AWS CloudFormation
- Panduan Pengguna AWS CloudFormation
- Panduan Pengguna Antarmuka Baris Perintah AWS CloudFormation

Menggunakan antarmuka baris perintah Amazon ECS

Amazon ECS telah merilis AWS Copilot, alat antarmuka baris perintah (CLI) yang menyederh anakan pembuatan, pelepasan, dan pengoperasian aplikasi kontainer siap produksi di Amazon ECS dari lingkungan pengembangan lokal. Untuk informasi selengkapnya, lihat Menggunakan antarmuka baris perintah AWS Copilot.

Antarmuka baris perintah Amazon Elastic Container Service (Amazon ECS) (CLI) menyediakan perintah tingkat tinggi untuk menyederhanakan pembuatan, pembaruan, dan pemantauan cluster dan tugas dari lingkungan pengembangan lokal. Amazon ECS CLI mendukung file Docker Compose, spesifikasi open-source populer untuk mendefinisikan dan menjalankan aplikasi multi-container. Gunakan CLI ECS sebagai bagian dari siklus pengembangan dan pengujian sehari-hari Anda sebagai alternatif AWS Management Console.

Versi terbaru dari Amazon ECS CLI hanya mendukung versi utama sintaks file Docker Compose versi 1, 2, dan 3. Versi yang ditentukan dalam file compose harus berupa string `"1"`, `"1.0"`, `"2"`, `"2.0"`, `"3"`, atau `"3.0"`. Versi minor Docker Compose tidak didukung.

Kode sumber untuk Amazon ECS CLI tersedia di. GitHub Alat ini tidak lagi dikembangkan secara aktif.

Menginstal Amazon ECS CLI

> Amazon ECS telah merilis AWS Copilot, alat antarmuka baris perintah (CLI) yang menyederh anakan pembuatan, pelepasan, dan pengoperasian aplikasi kontainer siap produksi di Amazon ECS dari lingkungan pengembangan lokal. Untuk informasi selengkapnya, lihat Menggunakan antarmuka baris perintah AWS Copilot.

Langkah-langkah berikut menunjukkan cara menginstal Amazon ECS CLI di sistem macOS, Linux, atau Windows Anda.

Untuk menginstal Amazon ECS CLI

1. Unduh biner Amazon ECS CLI.

   macOS

   ```
   sudo curl -Lo /usr/local/bin/ecs-cli https://amazon-ecs-cli.s3.amazonaws.com/ecs-cli-darwin-amd64-latest
   ```

   Linux

   ```
   sudo curl -Lo /usr/local/bin/ecs-cli https://amazon-ecs-cli.s3.amazonaws.com/ecs-cli-linux-amd64-latest
   ```

   Windows

   Buka Windows PowerShell dan masukkan perintah berikut.

> **Note**
>
> Jika Anda mengalami masalah izin, pastikan Anda memiliki akses administrator di Windows dan Anda menjalankan PowerShell sebagai administrator.

```
New-Item -Path 'C:\Program Files\Amazon\ECSCLI' -ItemType Directory
Invoke-WebRequest -OutFile 'C:\Program Files\Amazon\ECSCLI\ecs-cli.exe' https://
amazon-ecs-cli.s3.amazonaws.com/ecs-cli-windows-amd64-latest.exe
```

2. Verifikasi CLI Amazon ECS menggunakan tanda tangan PGP. Executable Amazon ECS CLI ditandatangani secara kriptografis menggunakan tanda tangan PGP. Tanda tangan PGP dapat digunakan untuk memverifikasi validitas Amazon ECS CLI yang dapat dieksekusi. Gunakan langkah-langkah berikut untuk memverifikasi tanda tangan menggunakan alat GnuPG.

   a. Unduh dan instal GnuPG. Untuk informasi selengkapnya, lihat situs web GnuPG.

      macOS

      Kami merekomendasikan menggunakan Homebrew. Instal Homebrew menggunakan instruksi dari situs web mereka. Untuk informasi selengkapnya, lihat Homebrew. Setelah Homebrew diinstal, gunakan perintah berikut dari terminal macOS Anda.

      ```
      brew install gnupg
      ```

      Linux

      Instal `gpg` menggunakan manajer paket pada rasa Linux Anda.

      Windows

      Unduh penginstal sederhana Windows dari situs web GnuPG dan instal sebagai Administrator. Setelah Anda menginstal GnuPG, tutup dan buka kembali Administrator. PowerShell

      Untuk informasi selengkapnya, lihat Unduhan GnuPG.

   b. Verifikasi jalur GnuPG ditambahkan ke jalur lingkungan Anda.

macOS

```
echo $PATH
```

Jika Anda tidak melihat jalur GnuPG di output, jalankan perintah berikut untuk menambahkannya ke jalur.

```
PATH=$PATH:<path to GnuPG executable files>
```

Linux

```
echo $PATH
```

Jika Anda tidak melihat jalur GnuPG di output, jalankan perintah berikut untuk menambahkannya ke jalur.

```
export PATH=$PATH:<path to GnuPG executable files>
```

Windows

```
Write-Output $Env:PATH
```

Jika Anda tidak melihat jalur GnuPG di output, jalankan perintah berikut untuk menambahkannya ke jalur.

```
$Env:PATH += ";<path to GnuPG executable files>"
```

c. Buat file teks biasa lokal.

macOS

Di terminal, masukkan:

```
touch <public_key_filename.txt>
```

Buka file dengan TextEdit.

Linux

Buat file teks di editor teks seperti gedit. Simpan sebagai `public_key_filename.txt`

Windows

Buat file teks di editor teks seperti Notepad. Simpan sebagai `public_key_filename.txt`

d. Tambahkan konten berikut dari kunci publik Amazon ECS PGP dan simpan file.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2

mQINBFq1SasBEADliGcT1NVJ1ydfN8DqebYYe9ne3dt6jqKFmKowLmm6LLGJe7HU
jGtqhCWRDkN+qPpHqdArRgDZAtn2pXY5fEipHgar4CP8QgRnRMO2fl74lmavr4Vg
7K/KH8VHlq2uRw32/B94XLEgRbGTMdWFdKuxoPCttBQaMj3LGn6Pe+6xVWRkChQu
BoQAhjBQ+bEm0kNy0LjNgjNlnL3UMAG56t8E3LANIgGgEnpNsB1UwfWluPoGZoTx
N+6pHBJrKIL/1v/ETU4FXpYw2zvhWNahxeNRnoYj3uycHkeliCrw4kj0+skizBgO
2K7oVX8Oc3j5+ZilhL/qDLXmUCb2az5cMM1mOoF8EKX5HaNuq1KfwJxqXE6NNIcO
lFTrT7QwD5fMNld3FanLgv/ZnIrsSaqJOL6zRSq8O4LN1OWBVbndExk2Kr+5kFxn
5lBPgfPgRj5hQ+KTHMa9Y8Z7yUc64BJiN6F9Nl7FJuSsfqbdkvRLsQRbcBG9qxX3
rJAEhieJzVMEUNl+EgeCkxj5xuSkNU7zw2c3hQZqEcrADLV+hvFJktOz9Gm6xzbq
lTnWWCz4xrIWtuEBA2qE+MlDheVd78a3gIsEaSTfQq0osYXaQbvlnSWOoc1y/5Zb
zizHTJIhLtUyls9WisP2s0emeHZicVMfW61EgPrJAiupgc7kyZvFt4YwfwARAQAB
tCRBbWF6b24gRUNTIDxlY3Mtc2VjdXJpdHlAYW1hem9uLmNvbT6JAhwEEAECAAYF
AlrjL0YACgkQHivRXs0TaQrg1g/+JppwPqHnlVPmv7lessB8I5UqZeD6p6uVpHd7
Bs3pcPp8BV7BdRbs3sPLt5bV1+rkqOlw+0gZ4Q/ue/YbWtOAt4qY0OcEo0HgcnaX
lsB827QIfZIVtGWMhuh94xzm/SJkvngml6KB3YJNnWP61A9qJ37/VbVVLzvcmazA
McWB4HUMNrhd0JgBCo0gIpqCbpJEvUc02Bjn23eEJsS9kC7OUAHyQkVnx4d9UzXF
4OoISF6hmQKIBoLnRrAlj5Qvs3GhvHQ0ThYq0Grk/KMJJX2CSqt7tWJ8gk1n3H3Y
SReRXJRnv7DsDDBwFgT6r5Q2HW1TBUvaoZy5hF6maD09nHcNnvBjqADzeT8Tr/Qu
bBCLzkNSYqqkpgtwv7seoD2P4n1giRvDAOEfMZpVkUr+C252IaH1HZFEz+TvBVQM
Y8OWWxmIJW+J6evjo3N1eO19UHv71jvoF8zljbI4bsL2c+QTJmOv7nRqzDQgCWyp
Id/v2dUVVTk1j9omuLBBwNJzQCB+72LcIzJhYmaP1HC4LcKQG+/f41exuItenatK
lEJQhYtyVXcBlh6Yn/wzNg2NWOwb3vqY/F7m6u9ixAwgtIMgPCDE4aJ86zrrXYFz
N2HqkTSQh77Z8KPKmyGopsmN/reMuilPdINb249nA0dzoN+nj+tTFOYCIaLaFyjs
Z0r1QAOJAjkEEwECACMFAlq1SasCGwMHCwkIBwMCAQYVCAIJCgsEFgIDAQIeAQIX
gAAKCRC86dmkLVF4T9iFEACEnkm1dNXsWUx34R3c0vamHrPxvfkyI1FlEUen8D1h
uX9xy6jCEROHWEp0rjGK4QDPgM93sWJ+s1UAKg214QRVzft0y9/DdR+twApA0fzy
uavIthGd6+03jAAo6udYDE+cZC3P7XBbDiYEWk4XAF9I1JjB8hTZUgvXBL046JhG
eM17+crgUyQeetkiOQemLbsbXQ40Bd9V7zf7XJraFd8VrwNUwNb+9KFtgAsc9rk+
YIT/PEf+YOPysgcxI4sTWghtyCulVnuGoskgDv4v73PALU0ieUrvvQVqWMRvhVx1
0X90J7cC1KOyhlEQQ1aFTgmQjmXexVTwIBm8LvysFK6YXM41KjOrlz3+6xBIm/qe
```

```
bFyLUnf4WoiuOplAaJhK9pRY+XEnGNxdtN4D26Kd0F+PLkm3Tr3Hy3b1Ok34FlGr
KVHUq1TZD7cvMnnNKEELTUcKX+1mV3an16nmAg/my1JSUt6BNK2rJpY1s/kkSGSE
XQ4zuF2IGCpvBFhYAlt5Un5zwqkwwQR3/n2kwAoDzonJcehDw/C/cGos5D0aIU7I
K2X2aTD3+pA7Mx3IMe2hqmYqRt9X42yF1PIEVRneBRJ3HDezAgJrNh0GQWRQkhIx
gz6/cTR+ekr5TptVszS9few2GpI5bCgBKBisZIssT89aw7mAKWut0Gcm4qM9/yK6
1bkCDQRatUmrARAAxNPvVwreJ2yAiFcUpdRlVhsuOgnxvs1QgsIw3H7+Pacr9Hpe
8uftYZqdC82KeSKhpHq7c8gMTMucIINtH25x9BCc73E33EjCL9Lqov1TL7+QkgHe
T+JIhZwdD8Mx2K+LVVVu/aWkNrfMuNwyDUciSI4D5QHa8T+F8fgN4OTpwYjirzel
5yoICMr9hVcbzDNv/ozKCxjx+XKgnFc3wrnDfJfntfDAT7ecwbUTL+viQKJ646s+
psiqXRYtVvYInEhLVrJ0aV6zHFoigE/Bils6/g7ru1Q6CEHqEw++APs5CcE8VzJu
WAGSVHZgun5Y9N4quR/M9Vm+IPMhTxrAg7rOvyRN9cAXfeSMf77I+XTifigNna8x
t/MOdjXr1fjF4pThEi5u6WsuRdFwjY2azEv3vevodTi4HoJReH6dFRa6y8c+UDgl
2iHiOKIpQqLbHEfQmHcDd2fix+AaJKMnPGNku9qCFEMbgSRJpXz6BfwnY1QuKE+I
R6jA0frUNt2jhiGG/F8RceXzohaaC/Cx7LUCUFWc0n7z32C9/Dtj7I1PMOacdZzz
bjJzRKO/ZDv+UN/c9dwAkllzAyPMwGBkUaY68EBstnIliW34aWm6IiHhxioVPKSp
VJfyiXPO0EXqujtHLAeChfjcns3I12YshT1dv2PafG53fp33ZdzeUgsBo+EAEQEA
AYkCHwQYAQIACQUCWrVJqwIbDAAKCRC86dmkLVF4T+ZdD/9x/8APzgNJF3o3STrF
jvnV1ycyhWYGAeBJiu7wjsNWwzMFOv15tLjB7AqeVxZn+WKDD/mIOQ45OZvnYZuy
X7DR0JszaH9wrYTxZLVruAu+t6UL0y/XQ4L1GZ9QR6+r+7t1Mvbfy7BlHbvX/gYt
Rwe/uwdibI0CagEzyX+2D3kTOlHO5XThbXaNf8AN8zha91Jt2Q2UR2X5T6JcwtMz
FBvZnl3LSmZyE0EQehS2iUurU4uWOpGppuqVnbi0jbCvCHKgDGrqZ0smKNAQng54
F365W3g8AfY48s8XQwzmcliowYX9bT8PZiEi0J4QmQh0aXkpqZyFefuWeOL2R94S
XKzr+gRh3BAULoqF+qK+IUMxTip9KTPNvYDpiC66yBiT6gFDji5Ca9pGpJXrC3xe
TXiKQ8DBWDhBPVPrruLIaenTtZEOsPc4I85yt5U9RoPTStcOr34s3w5yEaJagt6S
Gc5r9ysjkfH6+6rbi1ujxMgROSqtqr+RyB+V9A5/OgtNZc8llK6u4UoOCde8jUUW
vqWKvjJB/Kz3u4zaeNu2ZyyHaOqOuH+TETcW+jsY9IhbEzqN5yQYGi4pVmDkY5vu
lXbJnbqPKpRXgM9BecV9AMbPgbDq/5LnHJJXg+G8YQOgp4lR/hC1TEFdIp5wM8AK
CWsENyt2o1rjgMXiZOMF8A5oBLkCDQRatUuSARAAr77kj7j2QR2SZeOSlFBvV7oS
mFeSNnz9xZssqrsm6bTwSHM6YLDwc7Sdf2esDdyzONETwqrVCg+FxgL8hmo9hS4c
rR6tmrP0mOmptr+xLLsKcaP7ogIXsyZnrEAEsvW8PnfayoiPCdc3cMCR/lTnHFGA
7EuR/XLBmi7Qg9tByVYQ5Yj5wB9V4B2yeCt3XtzPqeLKvaxl7PNelaHGJQY/xo+m
V0bndxf9IY+4oFJ4blD32WqvyxESo7vW6WBh7oqv3Zbm0yQrr8a6mDBpqLkvWwNI
3kpJR974tg5o5LfDu1BeeyHWPSGm4U/G4JB+JIG1ADy+RmoWEt4BqTCZ/knnoGvw
D5sTCxbKdmuOmhGyTssoG+3OOcGYHV7pWYPhazKHMPm201xKCjH1RfzRULzGKjD+
yMLT1I3AXFmLmZJXikAOlvE3/wgMqCXscbycbLjLD/bXIuFWo3rzoezeXjgi/DJx
jKBAyBTYO5nMcth1O9oaFd9d0HbsOUDkIMnsgGBE766Piro6MHo0T0rXl07Tp4pI
rwuSOsc6XzCzdImj0Wc6axS/HeUKRXWdXJwno5awTwXKRJMXGfhCvSvbcbc2Wx+L
IKvmB7EB4K3fmjFFE67yolmiw2qRcUBfygtH3eL5XZU28MiCpue8Y8GKJoBAUyvf
KeM1rO8Jm3iRAc5a/D0AEQEAAYkEPgQYAQIACQUCWrVLkgIbAgIpCRC86dmkLVF4
T8FdIAQZAQIABgUCWrVLkgAKCRDePL1hra+LjtHYD/9MucxdFe6bXO1dQR4tKhhQ
P0LRqy6zlBY9ILCLowNdGZdqorogUiUymgn3VhEhVtxTOoHcN7qOuM01PNsRnOeS
EYjf8Xrb1clzkD6xULwmOclTb9bBxnBc/4PFvHAbZW3QzusaZniNgkuxt6BTfloS
Of4inq71kjmGK+TlzQ6mUMQUg228NUQC+a84EPqYyAeY1sgvgB7hJBhYL0QAxhcW
6m20Rd8iEc6HyzJ3yCOCsKip/nRWAbf0OvfHfRBp0+m0ZwnJM8cPRFjOqqzFpKH9
```

```
HpDmTrC4wKP1+TL52LyEqNh4yZitXmZNV7giSRIkk0eDSko+bFy6VbMzKUMkUJK3
D3eHFAMkujmbfJmSMTJOPGn5SB1HyjCZNx6bhIIbQyEUB9gKCmUFaqXKwKpF6rj0
iQXAJxLR/shZ5Rk96VxzOphUl7T90m/PnUEEPwq8KsBhnMRgxa0RFidDP+n9fgtv
HLmrOqX9zBCVXh0mdWYLrWvmzQFWzG7AoE55fkf8nAEPsalrCdtaNUBHRXA0OQxG
AHMOdJQQvBsmqMvuAdjkDWpFu5y0My5ddU+hiUzUyQLjL5Hhd5LOUDdewlZgIw1j
xrEAUzDKetnemM8GkHxDgg8koev5frmShJuce7vSjKpCNg3EIJSgqMOPFjJuLWtZ
vjHeDNbJy6uNL65ckJy6WhGjEADS2WAW1D6Tfekkc21SsIXk/LqEpLMR/0g5OUif
wcEN1rS9IJXBwIy8MelN9qr5KcKQLmfdfBNEyyceBhyVl0MDyHOKC+7PofMtkGBq
13QieRHv5GJ8LB3fclqHV8pwTTo3Bc8z2g0TjmUYAN/ixETdReDoKavWJYSE9yoM
aaJu279ioVTrwpECse0XkiRyKToTjwOb73CGkBZZpJyqux/rmCV/fp4ALdSW8zbz
FJVORaivhoWwzjpfQKhwcU9lABXi2UvVm14v0AfeI7oiJPSU1zM4fEny4oiIBXlR
zhFNih1UjIu82X16mTm3BwbIga/s1fnQRGzyhqUIMii+mWra23EwjChaxpvjjcUH
5ilLc5Zq781aCYRygYQw+hu5nFkOH1R+Z50Ubxjd/aqUfnGIAX7kPMD3Lof4KldD
Q8ppQriUvxVo+4nPV6rpTy/PyqCLWDjkguHpJsEFsMkwajrAz0QNSAU5CJ0G2Zu4
yxvYlumHCEl7nbFrm0vIiA75Sa8KnywTDsyZsu3XcOcf3g+g1xWTpjJqy2bYXlqz
9uDOWtArWHOis6bq8l9RE6xr1RBVXS6uqgQIZFBGyq66b0dIq4D2JdsUvgEMaHbc
e7tBfeB1CMBdA64e9Rq7bFR7Tvt8gasCZYlNr3lydh+dFHIEkH53HzQe6l88HEic
+0jVnLkCDQRa55wJARAAyLya2Lx6gyoWoJN1a6740q3o8e9d4KggQOfGMTCflmeq
ivuzgN+3DZHN+9ty2KxXMtn0mhHBerZdbNJyjMNT1gAgrhPNB4HtXBXum2wS57WK
DNmade914L7FWTPAWBG2Wn448OEHTqsClICXXWy9IICgclAEyIq0Yq5mAdTEgRJS
Z8t4GpwtDL9gNQyFXaWQmDmkAsCygQMvhAlmu9xOIzQG5CxSnZFk7zcuL60k14Z3
Cmt49k4T/7ZU8goWi8tt+rU78/IL3J/fF9+1civ1OwuUidgfPCSvOUW1JojsdCQA
L+RZJcoXq7lfOFj/eNjeOSstCTDPfTCL+kThE6E5neDtbQHBYkEX1BRiTedsV4+M
ucgiTrdQFWKf89G72xdv8ut9AYYQ2BbEYU+JAYhUH8rYYui2dHKJIgjNvJscuUWb
+QEqJIRleJRhrO+/CHgMs4fZAkWF1VFhKBkcKmEjLn1f7EJJUUW84ZhKXjO/AUPX
1CHsNjziRceuJCJYox1cwsoq6jTE50GiNzcIxTn9xUc0UMKFeggNAFys1K+TDTm3
Bzo8H5ucjCUEmUm9lhkGwqTZgOlRX5eqPX+JBoSaObqhgqCa5IPinKRa6MgoFPHK
6sYKqroYwBGgZm6Js5chpNchvJMs/3WXNOEVg0J3z3vP0DMhxqWm+r+n9zlW8qsA
EQEAAYkEPgQYAQgACQUCWuecCQIbAgIpCRC86dmkLVF4T8FdIAQZAQgABgUCWuec
CQAKCRBQ3szEcQ5hr+ykD/4tOLRHFHXuKUcxgGaubUcVtsFrwBKma1cYjqaPms8u
6Sk0wfGRI32G/GhOrp0Ts/MOkbObq6VLTh8N5Yc/53MEl8zQFw9Y5AmRoW4PZXER
ujs5s7p4oR7xHMihMjCCBn1bvrR+34YPfgzTcgLiOEFHYT8UTxwnGmXOvNkMM7md
xD3CV5q6VAte8WKBo/220II3fcQlc9r/oWX4kXXkb0v9hoGwKbDJ1tzqTPrp/xFt
yohqnvImpnlz+Q9zXmbrWYL9/g8VCmW/NN2gju2G3Lu/TlFUWIT4v/5OPK6TdeNb
VKJO4+S8bTayqSG9CML1S57KSgCo5HUhQWeSNHI+fpe5oX6FALPT9JLDce8OZz1i
cZZ0MELP37mOOQun0AlmHm/hVzf0f311PtbzcqWaE51tJvgUR/nZFo6Ta3O5Ezhs
3VlEJNQ1Ijf/6DH87SxvAoRIARCuZd0qxBcDK0avpFzUtbJd24lRA3WJpkEiMqKv
RDVZkE4b6TW61f0o+LaVfK6E8oLpixegS4fiqC16mFrOdyRk+RJJfIUyz0WTDVmt
g0U1CO1ezokMSqkJ7724pyjr2xf/r9/sC6aOJwB/lKgZkJfC6NqL7TlxVA31dUga
LEOvEJTTE4gl+tYtfsCDvALCtqL0jduSkUo+RXcBItmXhA+tShW0pbS2Rtx/ixua
KohVD/0R4QxiSwQmICNtm9mw9ydIl1yjYXX5a9x4wMJracNY/LBybJPFnZnT4dYR
z4XjqysDwvvYZByaWoIe3QxjX84V6MlI2IdAT/xImu8gbaCI8tmyfpIrLnPKiR9D
VFYfGBXuAX7+HgPPSFtrHQONCALxxzlbNpS+zxt9r0MiLgcLyspWxSdmoYGZ6nQP
RO5Nm/ZVS+u2imPCRzNUZEMa+dlE6kHx0rS0dPiuJ4O7NtPeYDKkoQtNagspsDvh
```

```
cK7CSqAiKMq06UBTxqlTSRkm62eOCtcs3p3OeHu5GRZF1uzTET0ZxYkaPgdrQknx
ozjP5mC7X+45lcCfmcVt94TFNL5HwEUVJpmOgmzILCI8yoDTWzloo+i+fPFsXX4f
kynhE83mSEcr5VHFYrTY3mQXGmNJ3bCLuc/jq7ysGq69xiKmTlUeXFm+aojcRO5i
zyShIRJZ0GZfuzDYFDbMV9amA/YQGygLw//zP5ju5SW26dNxlf3MdFQE5JJ86rn9
MgZ4gcpazHEVUsbZsgkLizRp9imUiH8ymLqAXnfRGlU/LpNSefnvDFTtEIRcpOHc
bhayG0bk51Bd4mioOXnIsKy4j63nJXA27x5EVVHQ1sYRN8Ny4Fdr2tMAmj2O+X+J
qX2yy/UX5nSPU492e2CdZ1UhoU0SRFY3bxKHKB7SDbVeav+K5g==
=Gi5D
-----END PGP PUBLIC KEY BLOCK-----
```

Rincian kunci publik Amazon ECS PGP untuk referensi:

```
Key ID: BCE9D9A42D51784F
Type: RSA
Size: 4096/4096
Expires: Never
User ID: Amazon ECS
Key fingerprint: F34C 3DDA E729 26B0 79BE AEC6 BCE9 D9A4 2D51 784F
```

Anda dapat menutup editor teks.

e. Impor file dengan kunci publik Amazon ECS PGP dengan perintah berikut di terminal.

```
gpg --import <public_key_filename.txt>
```

f. Unduh tanda tangan Amazon ECS CLI. Tanda tangan tersebut adalah tandatangan PGP ASCII-terpisah yang tersimpan di file dengan ekstensi `.asc`. File tanda tangan memiliki nama yang sama dengan file yang dapat dieksekusi yang sesuai, dengan penambahan `.asc`.

macOS

```
curl -Lo ecs-cli.asc https://amazon-ecs-cli.s3.amazonaws.com/ecs-cli-darwin-amd64-latest.asc
```

Linux

```
curl -Lo ecs-cli.asc https://amazon-ecs-cli.s3.amazonaws.com/ecs-cli-linux-amd64-latest.asc
```

Windows

```
Invoke-WebRequest -OutFile ecs-cli.asc https://amazon-ecs-
cli.s3.amazonaws.com/ecs-cli-windows-amd64-latest.exe.asc
```

g. Verifikasi tanda tangan tersebut.

macOS and Linux

```
gpg --verify ecs-cli.asc /usr/local/bin/ecs-cli
```

Windows

```
gpg --verify ecs-cli.asc 'C:\Program Files\Amazon\ECSCLI\ecs-cli.exe'
```

Keluaran yang diharapkan

```
gpg: Signature made Tue Apr  3 13:29:30 2018 PDT
gpg:                using RSA key DE3CBD61ADAF8B8E
gpg: Good signature from "Amazon ECS <ecs-security@amazon.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: F34C 3DDA E729 26B0 79BE  AEC6 BCE9 D9A4 2D51 784F
     Subkey fingerprint: EB3D F841 E2C9 212A 2BD4  2232 DE3C BD61 ADAF 8B8E
```

> **Important**
>
> Peringatan dalam output diharapkan dan tidak bermasalah. Itu terjadi karena tidak ada rantai kepercayaan antara kunci PGP pribadi Anda (jika Anda memilikinya) dan kunci Amazon ECS PGP. Untuk informasi selengkapnya, lihat Web kepercayaan.

3. Menerapkan izin eksekusi ke biner.

macOS and Linux

```
sudo chmod +x /usr/local/bin/ecs-cli
```

Windows

Edit variabel lingkungan dan tambahkan `C:\Program Files\Amazon\ECSCLI` ke bidang variabel `PATH`, terpisah dari entri yang ada dengan menggunakan titik koma. Sebagai contoh:

```
setx path "%path%;C:\Program Files\Amazon\ECSCLI"
```

Mulai ulang PowerShell sehingga perubahan berlaku.

> **Note**
>
> Setelah `PATH` variabel diatur, Amazon ECS CLI dapat digunakan baik dari PowerShell Windows atau command prompt.

4. Pastikan bahwa CLI bekerja dengan benar.

```
ecs-cli --version
```

Lanjut ke Mengkonfigurasi CLI Amazon ECS.

> **Important**
>
> Anda harus mengonfigurasi CLI Amazon ECS dengan kredensil, Wilayah, AWS dan nama cluster Amazon ECS sebelum AWS Anda dapat menggunakannya. Untuk informasi selengkapnya, lihat Mengkonfigurasi CLI Amazon ECS.

Mengkonfigurasi CLI Amazon ECS

Amazon ECS telah merilis AWS Copilot, alat antarmuka baris perintah (CLI) yang menyederh anakan pembuatan, pelepasan, dan pengoperasian aplikasi kontainer siap produksi di Amazon ECS dari lingkungan pengembangan lokal. Untuk informasi selengkapnya, lihat Menggunakan antarmuka baris perintah AWS Copilot.

Amazon ECS CLI memerlukan beberapa informasi konfigurasi dasar sebelum Anda dapat menggunakannya, seperti kredensi AWS Anda, AWS Wilayah tempat membuat cluster Anda, dan

nama cluster Amazon ECS yang akan digunakan. Informasi konfigurasi disimpan dalam direktori `~/.ecs` pada sistem macOS dan Linux dan dalam `C:\Users\<username>\AppData\local\ecs` pada sistem Windows.

Untuk mengkonfigurasi Amazon ECS CLI

1. Atur profil CLI dengan perintah berikut, menggantikan *`profile_name`* dengan nama profil yang Anda inginkan, variabel lingkungan *`$AWS_ACCESS_KEY_ID`* dan *`$AWS_SECRET_ACCESS_KEY`* dengan kredensial AWS Anda.

   ```
   ecs-cli configure profile --profile-name profile_name --access-
   key $AWS_ACCESS_KEY_ID --secret-key $AWS_SECRET_ACCESS_KEY
   ```

2. Selesaikan konfigurasi dengan perintah berikut, ganti *`launch_type`*dengan tipe peluncuran tugas yang ingin Anda gunakan secara default, *`region_name`*dengan AWS Region yang Anda inginkan, *`cluster_name`*dengan nama cluster Amazon ECS yang ada atau cluster baru yang akan digunakan, dan *`configuration_name`* untuk nama yang ingin Anda berikan konfigurasi ini.

   ```
   ecs-cli configure --cluster cluster_name --default-launch-type launch_type --
   region region_name --config-name configuration_name
   ```

Menggunakan profil

Amazon ECS CLI mendukung konfigurasi beberapa set kredensil sebagai AWS profil bernama menggunakan perintah. ecs-cli configure profile Profil default dapat diatur dengan menggunakan perintah ecs-cli configure profile default. Profil ini kemudian dapat direferensikan ketika Anda menjalankan perintah Amazon ECS CLI yang memerlukan kredensil menggunakan `--ecs-profile` bendera jika tidak profil default digunakan.

Menggunakan konfigurasi cluster

Konfigurasi klaster adalah sekumpulan bidang yang menjelaskan kluster Amazon ECS termasuk nama cluster dan wilayah. Konfigurasi klaster default dapat diatur menggunakan perintah ecs-cli configure default. Amazon ECS CLI mendukung konfigurasi beberapa konfigurasi cluster bernama menggunakan opsi. `--config-name`

Memahami urutan prioritas

Ada beberapa metode untuk meneruskan kredensi dan wilayah dalam perintah Amazon ECS CLI. Berikut adalah urutan prioritas untuk masing-masing dari berikut.

Urutan prioritas untuk kredensial adalah:

1. Bendera profil Amazon ECS CLI:
   a. Profil Amazon ECS () `--ecs-profile`
   b. Profil AWS (`--aws-profile`)
2. Variabel lingkungan:
   a. `ECS_PROFILE`
   b. `AWS_PROFILE`
   c. `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, dan `AWS_SESSION_TOKEN`
3. Konfigurasi ECS berupaya mengambil kredensil dari profil ECS default.
4. AWSProfil default — Mencoba menggunakan kredensil (`aws_access_key_id`,`aws_secret_access_key`) atau `assume_role` (`role_arn`,`source_profile`) dari nama AWS profil.
   a. Variabel lingkungan `AWS_DEFAULT_PROFILE` (default ke `default`).
5. Peran instans EC2

Urutan prioritas Wilayah adalah:

1. Bendera Amazon ECS CLI:
   a. Bendera wilayah (`--region`)
   b. Bendera konfigurasi klaster (`--cluster-config`)
2. Konfigurasi ECS berupaya mengambil Wilayah dari profil ECS default.
3. Variabel lingkungan—mencoba untuk mengambil wilayah dari variabel lingkungan berikut:
   a. `AWS_REGION`
   b. `AWS_DEFAULT_REGION`
4. AWSprofil - upaya untuk menggunakan wilayah dari nama AWS profil:
   a. Variabel lingkungan `AWS_PROFILE`
   b. Variabel lingkungan `AWS_DEFAULT_PROFILE` (default ke `default`).

Amazon ECS aktif AWS Fargate

AWS Fargateadalah teknologi yang dapat Anda gunakan dengan Amazon ECS untuk menjalankan container tanpa harus mengelola server atau cluster instans Amazon EC2. Dengan AWS Fargate, Anda tidak perlu lagi menyediakan, mengonfigurasi, atau menskalakan klaster mesin virtual untuk menjalankan kontainer. Hal ini menghilangkan kebutuhan untuk memilih tipe server, memutuskan waktu untuk menskalakan klaster Anda, atau mengoptimalkan klaster kemasan.

Ketika Anda menjalankan tugas dan layanan Anda dengan tipe peluncuran Fargate, Anda mengemas aplikasi Anda dalam wadah, menentukan persyaratan CPU dan memori, menentukan kebijakan jaringan dan IAM, dan meluncurkan aplikasi. Setiap tugas Fargate memiliki batas isolasi sendiri dan tidak berbagi kernel yang mendasarinya, sumber daya CPU, sumber daya memori, atau elastic network interface dengan tugas lain. Anda mengonfigurasi definisi tugas untuk Fargate dengan menyetel parameter definisi `requiresCompatibilities` tugas ke. `FARGATE` Untuk informasi selengkapnya, lihat Jenis peluncuran.

Fargate menawarkan versi platform untuk Amazon Linux 2 dan Microsoft Windows 2019 Server edisi Lengkap dan Inti. Kecuali ditentukan lain, informasi di halaman ini berlaku untuk semua platform Fargate.

Topik ini menjelaskan berbagai komponen tugas dan layanan Fargate, dan memanggil pertimbangan khusus untuk menggunakan Fargate dengan Amazon ECS.

Untuk informasi tentang Wilayah yang mendukung kontainer Linux di Fargate, lihat. the section called “Kontainer Linux di AWS Fargate”

Untuk informasi tentang Wilayah yang mendukung kontainer Windows di Fargate, lihat. the section called “Wadah Windows di AWS Fargate”

Penelusuran Fargate

Untuk informasi tentang cara memulai menggunakan konsol, lihat:

- Memulai dengan wadah Linux di AWS Fargate
- Menggunakan wadah Windows pada AWS Fargate

Untuk informasi tentang cara mulai menggunakan AWS CLI, lihat:

- Membuat cluster dengan tugas Fargate Linux menggunakan AWS CLI
- Membuat cluster dengan tugas Fargate Windows menggunakan AWS CLI

Penyedia kapasitas

Penyedia kapasitas berikut tersedia:

- Fargate
- Fargate Spot - Jalankan tugas Amazon ECS toleran interupsi dengan tarif diskon dibandingkan dengan harga. AWS Fargate Spot Fargate menjalankan tugas pada kapasitas komputasi cadangan. Ketika AWS membutuhkan kapasitas kembali, tugas Anda akan terganggu dengan peringatan dua menit. Untuk informasi selengkapnya, lihat AWS Fargate penyedia kapasitas.

  Anda hanya dapat menggunakan Fargate Spot untuk tugas-tugas Linux yang menggunakan arsitektur X86.

Ketentuan tugas

Tugas yang menggunakan tipe peluncuran Fargate tidak mendukung semua parameter definisi tugas Amazon ECS yang tersedia. Beberapa parameter tidak didukung sama sekali, dan yang lain berperilaku berbeda untuk tugas Fargate. Untuk informasi selengkapnya, lihat CPU dan memori tugas.

Versi platform

AWS Versi platform Fargate digunakan untuk merujuk ke lingkungan runtime tertentu untuk infrastruktur tugas Fargate. Ini adalah kombinasi dari kernel dan versi runtime container. Anda memilih versi platform ketika Anda menjalankan tugas atau ketika Anda membuat layanan untuk mempertahankan sejumlah tugas yang identik.

Revisi baru versi platform dirilis saat lingkungan runtime berkembang, misalnya, jika ada pembaruan kernel atau sistem operasi, fitur baru, perbaikan bug, atau pembaruan keamanan. Versi platform Fargate diperbarui dengan membuat revisi versi platform baru. Setiap tugas berjalan pada satu revisi versi platform selama siklus hidupnya. Jika Anda ingin menggunakan revisi versi platform terbaru, maka Anda harus memulai tugas baru. Tugas baru yang berjalan di Fargate selalu berjalan pada revisi terbaru dari versi platform, memastikan bahwa tugas selalu dimulai pada infrastruktur yang aman dan ditambal.

Jika ditemukan masalah keamanan yang memengaruhi versi platform yang ada, AWS buat revisi baru yang ditambal dari versi platform dan menghentikan tugas yang berjalan pada revisi yang rentan. Dalam beberapa kasus, Anda mungkin diberi tahu bahwa tugas Anda di Fargate telah dijadwalkan untuk pensiun. Untuk informasi selengkapnya, lihat AWS FAQ pemeliharaan tugas Fargate.

Untuk informasi lebih lanjut, lihat Versi platform Fargate Linux dan Fargate versi platform Windows.

Penyeimbangan beban layanan

Layanan Amazon ECS Anda secara opsional AWS Fargate dapat dikonfigurasi untuk menggunakan Elastic Load Balancing untuk mendistribusikan lalu lintas secara merata di seluruh tugas dalam layanan Anda.

Layanan Amazon ECS AWS Fargate mendukung jenis penyeimbang beban Application Load Balancer dan Network Load Balancer. Application Load Balancers digunakan untuk merutekan lalu lintas HTTP/HTTPS (atau lapisan 7). Network Load Balancer digunakan untuk merutekan lalu lintas TCP atau UDP (atau layer 4). Untuk informasi selengkapnya, lihat Mendistribusikan lalu lintas layanan Amazon ECS menggunakan load balancing.

Ketika Anda membuat grup target untuk layanan ini, Anda harus memilih `ip` sebagai tipe target, bukan `instance`. Ini karena tugas yang menggunakan mode `awsvpc` jaringan dikaitkan dengan elastic network interface, bukan instans Amazon EC2. Untuk informasi selengkapnya, lihat Mendistribusikan lalu lintas layanan Amazon ECS menggunakan load balancing.

Menggunakan Network Load Balancer untuk merutekan lalu lintas UDP ke Amazon ECS pada AWS Fargate tugas hanya didukung saat menggunakan platform versi 1.4 atau yang lebih baru.

Metrik penggunaan

Anda dapat menggunakan metrik CloudWatch penggunaan untuk memberikan visibilitas ke akun Anda penggunaan sumber daya. Gunakan metrik ini untuk memvisualisasikan penggunaan layanan Anda saat ini pada CloudWatch grafik dan dasbor.

AWS Fargate metrik penggunaan sesuai dengan kuota AWS layanan. Anda dapat mengonfigurasi alarm yang memberitahu Anda ketika penggunaan mendekati service quotas. Untuk informasi lebih lanjut tentang service quotas AWS Fargate, lihat AWS Fargate kuota layanan.

Untuk informasi selengkapnya tentang metrik AWS Fargate penggunaan, lihat metrik AWS Fargate penggunaan di Panduan Pengguna Amazon Elastic Container Service untuk. AWS Fargate

Versi platform Fargate Linux

AWS Versi platform Fargate digunakan untuk merujuk ke lingkungan runtime tertentu untuk infrastruktur tugas Fargate. Ini adalah kombinasi dari kernel dan versi runtime container. Anda memilih versi platform ketika Anda menjalankan tugas atau ketika Anda membuat layanan untuk mempertahankan sejumlah tugas yang identik.

Revisi baru versi platform dirilis saat lingkungan runtime berkembang, misalnya, jika ada pembaruan kernel atau sistem operasi, fitur baru, perbaikan bug, atau pembaruan keamanan. Versi platform Fargate diperbarui dengan membuat revisi versi platform baru. Setiap tugas berjalan pada satu revisi versi platform selama siklus hidupnya. Jika Anda ingin menggunakan revisi versi platform terbaru, maka Anda harus memulai tugas baru. Tugas baru yang berjalan di Fargate selalu berjalan pada revisi terbaru dari versi platform, memastikan bahwa tugas selalu dimulai pada infrastruktur yang aman dan ditambal.

Jika ditemukan masalah keamanan yang memengaruhi versi platform yang ada, AWS buat revisi baru yang ditambal dari versi platform dan menghentikan tugas yang berjalan pada revisi yang rentan. Dalam beberapa kasus, Anda mungkin diberi tahu bahwa tugas Anda di Fargate telah dijadwalkan untuk pensiun. Untuk informasi selengkapnya, lihat AWS FAQ pemeliharaan tugas Fargate.

Pertimbangan

Pertimbangkan hal berikut saat menentukan versi platform:

- Saat menentukan versi platform, Anda dapat menggunakan nomor versi tertentu, misalnya`1.4.0`, atau`LATEST`.

  Ketika versi platform TERBARU dipilih, versi `1.4.0` platform digunakan.
- Jika Anda ingin memperbarui versi platform untuk suatu layanan, buat penerapan. Misalnya, asumsikan bahwa Anda memiliki layanan yang menjalankan tugas pada versi platform Linux`1.3.0`. Untuk mengubah layanan untuk menjalankan tugas pada versi platform Linux`1.4.0`, Anda dapat memperbarui layanan Anda dan menentukan versi platform baru. Tugas Anda dikerahkan kembali dengan versi platform terbaru dan revisi versi platform terbaru. Untuk informasi lebih lanjut tentang deployment, lihat Jenis Penerapan Amazon ECS.
- Jika layanan Anda ditingkatkan tanpa memperbarui versi platform, tugas tersebut menerima versi platform yang ditentukan pada penerapan layanan saat ini. Misalnya, asumsikan bahwa Anda memiliki layanan yang menjalankan tugas pada versi platform Linux`1.3.0`. Jika Anda

meningkatkan jumlah layanan yang diinginkan, penjadwal layanan memulai tugas baru menggunakan revisi versi platform terbaru dari versi platform. `1.3.0`

- Tugas baru selalu berjalan pada revisi terbaru dari versi platform, memastikan bahwa tugas selalu dimulai pada infrastruktur yang aman dan ditambal.
- Nomor versi platform untuk wadah Linux dan wadah Windows di Fargate bersifat independen. Misalnya, perilaku, fitur, dan perangkat lunak yang digunakan dalam versi platform `1.0.0` untuk wadah Windows di Fargate tidak sebanding dengan versi platform `1.0.0` untuk wadah Linux di Fargate.

Berikut ini adalah versi platform Linux yang tersedia. Untuk informasi tentang penghentian versi platform, lihat. AWS Penghentian versi platform Fargate Linux

1.4.0

Berikut ini adalah changelog untuk versi platform. `1.4.0`

- Mulai 5 November 2020, setiap tugas Amazon ECS baru yang diluncurkan di Fargate menggunakan `1.4.0` versi platform akan dapat menggunakan fitur-fitur berikut:
  - Saat menggunakan Secrets Manager untuk menyimpan data sensitif, Anda dapat menyuntikkan kunci JSON tertentu atau versi rahasia tertentu sebagai variabel lingkungan atau dalam konfigurasi log. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah.
  - Tentukan variabel lingkungan dalam jumlah besar menggunakan parameter ketentuan kontainer `environmentFiles`. Untuk informasi selengkapnya, lihat Gunakan parameter definisi tugas untuk meneruskan variabel lingkungan ke wadah.
  - Tugas yang dijalankan di VPC dan subnet yang diaktifkan untuk IPv6 akan diberikan alamat IPv4 privat dan IPv6. Untuk informasi selengkapnya, lihat Jaringan tugas Fargate di Panduan Pengguna Layanan Kontainer Elastis Amazon untuk. AWS Fargate
  - Titik akhir tugas metadata versi 4 menyediakan metadata tambahan tentang tugas dan kontainer termasuk tipe peluncuran tugas, Amazon Resource Name (ARN) kontainer, serta driver log dan opsi log driver yang digunakan. Saat melakukan kueri terhadap titik akhir `/stats` Anda juga menerima statistik tingkat jaringan untuk kontainer Anda. Untuk informasi selengkapnya, lihat Titik akhir metadata tugas versi 4.
- Mulai 30 Juli 2020, setiap tugas Amazon ECS baru yang diluncurkan di Fargate menggunakan `1.4.0` versi platform akan dapat merutekan lalu lintas UDP menggunakan Network Load Balancer ke Amazon ECS mereka pada tugas Fargate. Untuk informasi selengkapnya, lihat Mendistribusikan lalu lintas layanan Amazon ECS menggunakan load balancing.

- Mulai 28 Mei 2020, setiap tugas Amazon ECS baru yang diluncurkan di Fargate menggunakan `1.4.0` versi platform akan memiliki penyimpanan fana yang dienkripsi dengan algoritme enkripsi AES-256 menggunakan kunci enkripsi yang dimiliki. AWS Lihat informasi yang lebih lengkap di Penyimpanan sementara tugas Fargate dan Menggunakan volume data dalam tugas.
- Menambahkan dukungan untuk menggunakan volume sistem file Amazon EFS untuk penyimpanan tugas persisten. Untuk informasi selengkapnya, lihat Volume Amazon EFS.
- Penyimpanan tugas sementara telah ditingkatkan menjadi minimal 20 GB untuk setiap tugas. Untuk informasi selengkapnya, lihat Penyimpanan sementara tugas Fargate.
- Perilaku lalu lintas jaringan ke dan dari tugas telah diperbarui. Dimulai dengan platform versi 1.4.0, semua tugas Fargate menerima satu elastic network interface (disebut sebagai tugas ENI) dan semua lalu lintas jaringan mengalir melalui ENI tersebut dalam VPC Anda dan akan terlihat oleh Anda melalui log aliran VPC Anda. Untuk informasi selengkapnya tentang jaringan untuk jenis peluncuran Amazon EC2, lihat Jaringan Tugas Fargate. Untuk informasi selengkapnya tentang jaringan untuk jenis peluncuran Fargate, lihat. Jaringan tugas untuk tugas-tugas di Fargate
- ENI tugas menambahkan dukungan untuk frame jumbo. Antarmuka jaringan dikonfigurasi dengan unit transmisi maksimum (MTU), yang merupakan ukuran muatan terbesar yang muat dalam satu frame. Semakin besar MTU, semakin banyak muatan aplikasi yang termuat dalam satu frame, yang mengurangi overhead per frame dan meningkatkan efisiensi. Dengan mendukung jumbo frame akan dapat mengurangi overhead ketika jalur jaringan antara tugas dengan tujuan Anda mendukung jumbo frame, seperti semua lalu lintas yang tetap berada dalam VPC Anda.
- CloudWatch Container Insights akan menyertakan metrik kinerja jaringan untuk tugas Fargate. Untuk informasi selengkapnya, lihat Pantau kontainer Amazon ECS menggunakan Wawasan Kontainer.
- Menambahkan dukungan untuk titik akhir metadata tugas versi 4 yang memberikan informasi tambahan untuk tugas Fargate Anda, termasuk statistik jaringan untuk tugas dan Zona Ketersediaan mana tugas sedang berjalan. Untuk informasi lebih lanjut, lihat > Titik akhir metadata tugas Amazon ECS versi 4 danTitik akhir metadata tugas Amazon ECS versi 4 untuk tugas di Fargate.
- Menambahkan dukungan untuk parameter Linux `SYS_PTRACE` dalam ketentuan kontainer. Untuk informasi selengkapnya, lihat Parameter Linux.
- Agen kontainer Fargate menggantikan penggunaan agen kontainer Amazon ECS untuk semua tugas Fargate. Biasanya, perubahan ini tidak berpengaruh pada bagaimana tugas Anda berjalan.
- Waktu aktif kontainer kini menggunakan Containerd bukan Docker. Kemungkinan besar, perubahan ini tidak berpengaruh pada bagaimana tugas Anda berjalan. Anda akan melihat bahwa

beberapa pesan kesalahan yang berasal dari runtime container berubah dari menyebutkan Docker menjadi kesalahan yang lebih umum. Untuk informasi selengkapnya, lihat Kode kesalahan tugas berhenti di Panduan Pengguna Layanan Amazon Elastic Container untuk AWS Fargate.

- Berdasarkan Amazon Linux 2.

1.3.0

Berikut ini adalah changelog untuk versi platform. `1.3.0`

- Mulai 30 September 2019, setiap tugas Fargate baru yang diluncurkan mendukung driver log`awsfirelens`. FireLens Konfigurasikan Amazon ECS untuk menggunakan parameter definisi tugas untuk merutekan log ke AWS layanan atau tujuan Jaringan AWS Mitra (APN) untuk penyimpanan log dan analitik. Untuk informasi selengkapnya, lihat Menggunakan perutean log khusus.
- Menambahkan daur ulang tugas untuk tugas Fargate, yang merupakan proses menyegarkan tugas yang merupakan bagian dari layanan Amazon ECS. Untuk informasi selengkapnya, Pemeliharaan tugas di Panduan Pengguna Amazon Elastic Container Service untuk AWS Fargate.
- Dimulai pada 27 Maret 2019, tugas Fargate baru apa pun yang diluncurkan dapat menggunakan parameter definisi tugas tambahan yang Anda gunakan untuk menentukan konfigurasi proxy, dependensi untuk startup dan shutdown kontainer serta nilai batas waktu mulai dan berhenti per kontainer. Lihat informasi selengkapnya di Konfigurasi proxy, Dependensi kontainer, dan Waktu habis kontainer.
- Dimulai pada 2 April 2019, tugas Fargate baru apa pun yang diluncurkan mendukung penyuntikan data sensitif ke dalam wadah Anda dengan menyimpan data sensitif Anda baik dalam AWS Secrets Manager rahasia atau AWS Systems Manager parameter Parameter Store dan kemudian merujuknya dalam definisi penampung Anda. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah.
- Mulai 1 Mei 2019, tugas Fargate baru apa pun yang diluncurkan mendukung referensi data sensitif dalam konfigurasi log wadah menggunakan parameter definisi `secretOptions` wadah. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah.
- Mulai 1 Mei 2019, setiap tugas Fargate baru yang diluncurkan mendukung driver `splunk` log selain driver `awslogs` log. Untuk informasi selengkapnya, lihat Penyimpanan dan pencatatan.
- Mulai 9 Juli 2019, setiap tugas Fargate baru yang diluncurkan mendukung CloudWatch Wawasan Kontainer. Untuk informasi selengkapnya, lihat Pantau kontainer Amazon ECS menggunakan Wawasan Kontainer.

- Mulai 3 Desember 2019, penyedia kapasitas Fargate Spot didukung. Untuk informasi selengkapnya, lihat AWS Fargate penyedia kapasitas.
- Berdasarkan Amazon Linux 2.

Migrasi ke platform Linux versi 1.4.0

Pertimbangkan hal berikut saat memigrasikan Amazon ECS Anda pada tugas Fargate dari versi platform`1.0.0`, `1.1.01.2.0`, atau `1.3.0` ke versi platform. `1.4.0` Ini dianggap sebagai praktik terbaik untuk mengonfirmasi tugas Anda berfungsi dengan baik pada versi platform `1.4.0` sebelum memigrasikan tugas Anda.

- Perilaku lalu lintas jaringan ke dan dari tugas telah diperbarui. Dimulai dengan platform versi 1.4.0, semua Amazon ECS pada tugas Fargate menerima satu elastic network interface (disebut sebagai tugas ENI) dan semua arus lalu lintas jaringan melalui ENI tersebut dalam VPC Anda dan akan terlihat oleh Anda melalui log aliran VPC Anda. Untuk mengetahui informasi selengkapnya, lihat Jaringan tugas untuk tugas-tugas di Fargate.
- Jika Anda menggunakan titik akhir VPC antarmuka, pertimbangkan hal berikut.
  - Saat menggunakan gambar kontainer yang dihosting dengan Amazon ECR, titik akhir VPC Amazon ECR dan com.amazonaws.region.ecr.api Amazon ECR VPC serta titik akhir gateway Amazon S3 diperlukan. Untuk informasi selengkapnya, lihat Antarmuka Amazon ECR VPC endpoint AWS PrivateLink() di Panduan Pengguna Amazon Elastic Container Registry.
  - Saat menggunakan definisi tugas yang mereferensikan rahasia Secrets Manager untuk mengambil data sensitif untuk container Anda, Anda harus membuat titik akhir VPC antarmuka untuk Secrets Manager. Untuk informasi selengkapnya, lihat Menggunakan Secrets Manager dengan VPC Endpoint dalam Panduan Pengguna AWS Secrets Manager .
  - Saat menggunakan definisi tugas yang mereferensikan parameter Systems Manager Parameter Store untuk mengambil data sensitif untuk container Anda, Anda harus membuat titik akhir VPC antarmuka untuk Systems Manager. Untuk informasi selengkapnya, lihat Menggunakan Systems Manager dengan titik akhir VPC di Panduan Pengguna.AWS Systems Manager
  - Pastikan bahwa grup keamanan di Antarmuka Jaringan Elastis (ENI) yang terkait dengan tugas Anda memiliki aturan grup keamanan yang dibuat untuk memungkinkan lalu lintas antara tugas dan titik akhir VPC yang Anda gunakan.

AWS Penghentian versi platform Fargate Linux

Halaman ini mencantumkan versi platform Linux yang AWS Fargate tidak digunakan lagi atau telah dijadwalkan untuk dihentikan. Versi platform ini tetap tersedia hingga tanggal penghentian yang dipublikasikan.

Tanggal pembaruan paksa disediakan untuk setiap versi platform yang dijadwalkan untuk penghentian. Pada tanggal pembaruan paksa, layanan apa pun yang menggunakan versi `LATEST` platform yang diarahkan ke versi platform yang dijadwalkan untuk penghentian akan diperbarui menggunakan opsi force new deployment. Ketika layanan diperbarui menggunakan opsi force new deployment, semua tugas yang berjalan pada versi platform yang dijadwalkan untuk penghentian akan dihentikan dan tugas baru diluncurkan menggunakan versi platform yang ditunjuk `LATEST` tag pada saat itu. Tugas atau layanan mandiri dengan set versi platform eksplisit tidak terpengaruh oleh tanggal pembaruan paksa.

Kami menyarankan untuk memperbarui tugas mandiri layanan Anda untuk menggunakan versi platform terbaru. Untuk informasi selengkapnya tentang migrasi ke versi platform terbaru, lihatMigrasi ke platform Linux versi 1.4.0.

Setelah versi platform mencapai tanggal penghentian, versi platform tidak akan lagi tersedia untuk tugas atau layanan baru. Setiap tugas atau layanan mandiri yang secara eksplisit menggunakan versi platform yang tidak digunakan lagi akan terus menggunakan versi platform tersebut hingga tugas dihentikan. Setelah tanggal penghentian, versi platform yang tidak digunakan lagi tidak akan lagi menerima pembaruan keamanan atau perbaikan bug.

Versi platform	Tanggal pembaruan paksa	Tanggal pengusangan
1.0.0	26 Oktober 2020	14 Desember 2020
1.1.0	26 Oktober 2020	14 Desember 2020
1.2.0	26 Oktober 2020	14 Desember 2020

Untuk informasi tentang versi platform saat ini, lihat Versi platform Fargate Linux.

Changelog untuk versi Fargate Linux yang tidak digunakan lagi AWS

1.2.0

Berikut ini adalah changelog untuk versi platform. `1.2.0`

> **Note**
>
> Versi platform `1.2.0` tidak lagi tersedia. Untuk informasi tentang penghentian versi platform, lihat. AWS Penghentian versi platform Fargate Linux

- Menambahkan dukungan untuk otentikasi registri pribadi menggunakan AWS Secrets Manager. Untuk informasi selengkapnya, lihat Autentikasi registri privat untuk tugas.

1.1.0

Berikut ini adalah changelog untuk versi platform. `1.1.0`

> **Note**
>
> Versi platform `1.1.0` tidak lagi tersedia. Untuk informasi tentang penghentian versi platform, lihat. AWS Penghentian versi platform Fargate Linux

- Menambahkan dukungan untuk titik akhir metadata tugas Amazon ECS. Untuk informasi selengkapnya, lihat Metadata tugas Amazon ECS tersedia untuk tugas di Fargate.
- Menambahkan dukungan untuk pemeriksaan kondisi Docker dalam ketentuan kontainer. Untuk informasi selengkapnya, lihat Pemeriksaan kondisi.
- Menambahkan dukungan untuk penemuan layanan Amazon ECS. Untuk informasi selengkapnya, lihat Penemuan Layanan .

1.0.0

Berikut ini adalah changelog untuk versi platform. `1.0.0`

> **Note**
>
> Versi platform `1.0.0` tidak lagi tersedia. Untuk informasi tentang penghentian versi platform, lihat. AWS Penghentian versi platform Fargate Linux

- Berdasarkan Amazon Linux 2017.09.
- Pelepasan awal.

Fargate versi platform Windows

AWS Versi platform Fargate digunakan untuk merujuk ke lingkungan runtime tertentu untuk infrastruktur tugas Fargate. Ini adalah kombinasi dari kernel dan versi runtime container. Anda memilih versi platform ketika Anda menjalankan tugas atau ketika Anda membuat layanan untuk mempertahankan sejumlah tugas yang identik.

Revisi baru versi platform dirilis saat lingkungan runtime berkembang, misalnya, jika ada pembaruan kernel atau sistem operasi, fitur baru, perbaikan bug, atau pembaruan keamanan. Versi platform Fargate diperbarui dengan membuat revisi versi platform baru. Setiap tugas berjalan pada satu revisi versi platform selama siklus hidupnya. Jika Anda ingin menggunakan revisi versi platform terbaru, maka Anda harus memulai tugas baru. Tugas baru yang berjalan di Fargate selalu berjalan pada revisi terbaru dari versi platform, memastikan bahwa tugas selalu dimulai pada infrastruktur yang aman dan ditambal.

Jika ditemukan masalah keamanan yang memengaruhi versi platform yang ada, AWS buat revisi baru yang ditambal dari versi platform dan menghentikan tugas yang berjalan pada revisi yang rentan. Dalam beberapa kasus, Anda mungkin diberi tahu bahwa tugas Anda di Fargate telah dijadwalkan untuk pensiun. Untuk informasi selengkapnya, lihat AWS FAQ pemeliharaan tugas Fargate.

Pertimbangan versi platform

Pertimbangkan hal berikut saat menentukan versi platform:

- Saat menentukan versi platform, Anda dapat menggunakan nomor versi tertentu, misalnya`1.0.0`, atau`LATEST`.

  Ketika versi platform TERBARU dipilih, `1.0.0` platform digunakan.

- Tugas baru selalu berjalan pada revisi terbaru dari versi platform, memastikan bahwa tugas selalu dimulai pada infrastruktur yang aman dan ditambal.
- Gambar kontainer Microsoft Windows Server harus dibuat dari versi Windows Server tertentu. Anda harus memilih versi Windows Server yang sama `platformFamily` ketika Anda menjalankan tugas atau membuat layanan yang cocok dengan gambar wadah Windows Server. Selain itu, Anda dapat memberikan pencocokan `operatingSystemFamily` dalam definisi tugas untuk mencegah tugas dijalankan pada versi Windows yang salah. Untuk informasi selengkapnya, lihat Mencocokkan versi host kontainer dengan versi gambar kontainer di situs web Microsoft Learn.
- Nomor versi platform untuk wadah Linux dan wadah Windows di Fargate bersifat independen. Misalnya, perilaku, fitur, dan perangkat lunak yang digunakan dalam versi platform `1.0.0` untuk wadah Windows di Fargate tidak sebanding dengan versi platform `1.0.0` untuk wadah Linux di Fargate.

Berikut ini adalah versi platform yang tersedia untuk wadah Windows.

1.0.0

Berikut ini adalah changelog untuk versi platform. `1.0.0`

- Rilis awal untuk dukungan pada sistem operasi Microsoft Windows Server berikut:
  - Windows Server 2019 Full
  - Windows Server 2019 Core
  - Windows Server 2022 Lengkap
  - Windows Server 2022 Inti

Wadah Windows tentang pertimbangan Fargate

Berikut ini adalah perbedaan dan pertimbangan untuk mengetahui kapan Anda menjalankan wadah Windows di AWS Fargate.

Jika Anda perlu menjalankan tugas di wadah Linux dan Windows, maka Anda perlu membuat definisi tugas terpisah untuk setiap sistem operasi.

AWS menangani manajemen lisensi sistem operasi, sehingga Anda tidak memerlukan lisensi Microsoft Windows Server tambahan.

Kontainer Windows di AWS Fargate mendukung sistem operasi berikut:

- Windows Server 2019 Full
- Windows Server 2019 Core
- Windows Server 2022 Lengkap
- Windows Server 2022 Inti

Kontainer Windows di AWS Fargate mendukung driver awslogs. Untuk informasi selengkapnya, lihat the section called “Menggunakan driver log awslogs”.

Fitur-fitur berikut tidak didukung pada wadah Windows di Fargate:

- Akun layanan terkelola grup (GMSA)
- Amazon FSx
- ENI trunking
- Layanan App Mesh dan integrasi proxy untuk tugas
- Integrasi router log Firelens untuk tugas
- Volume EFS
- Parameter definisi tugas berikut:
  - `maxSwap`
  - `swappiness`
- Penyedia kapasitas Fargate Spot
- Volume gambar

  `volume`Opsi Dockerfile diabaikan. Sebagai gantinya, gunakan bind mount dalam definisi tugas Anda. Untuk informasi selengkapnya, lihat Pemasangan terikat.

AWS FAQ pemeliharaan tugas Fargate

Apa itu pemeliharaan tugas Fargate dan pensiun?

AWS bertanggung jawab untuk menjaga infrastruktur yang mendasari AWS Fargate. AWS menentukan kapan revisi versi platform perlu diganti dengan revisi baru. Ini dikenal sebagai pensiun tugas. AWS mengirimkan pemberitahuan pensiun tugas ketika revisi versi platform dihentikan. Kami

secara rutin memperbarui versi platform kami yang didukung untuk memperkenalkan revisi baru yang berisi pembaruan pada perangkat lunak runtime Fargate dan dependensi yang mendasarinya seperti sistem operasi dan runtime kontainer. Setelah revisi yang lebih baru tersedia, kami menghentikan revisi lama untuk memastikan semua beban kerja pelanggan berjalan pada revisi terbaru dari versi platform Fargate. Ketika revisi dihentikan, semua tugas yang berjalan pada revisi itu dihentikan.

Tugas Amazon ECS dapat dikategorikan sebagai tugas layanan atau tugas mandiri. Tugas layanan digunakan sebagai bagian dari layanan dan dikendalikan oleh jadwal Amazon ECS. Untuk informasi selengkapnya, lihat Layanan-layanan Amazon ECS. Tugas mandiri adalah tugas yang dimulai oleh Amazon ECS `RunTask` API, baik secara langsung atau oleh penjadwal eksternal seperti tugas terjadwal (yang dimulai oleh Amazon EventBridge), AWS Batch atau. AWS Step Functions

Untuk tugas layanan, Anda tidak perlu mengambil tindakan apa pun kecuali Anda ingin mengganti tugas ini sebelum AWS melakukannya. Ketika penjadwal Amazon ECS menghentikan tugas, ia menggunakan persentase sehat minimum dan meluncurkan tugas baru dalam upaya untuk mempertahankan jumlah yang diinginkan untuk layanan. Secara default, persentase kesehatan minimum dari suatu layanan adalah 100 persen, jadi tugas baru dimulai terlebih dahulu sebelum tugas dihentikan. Tugas layanan diganti secara rutin dengan cara yang sama saat Anda menskalakan layanan, menerapkan perubahan konfigurasi, atau menerapkan revisi definisi tugas. Untuk mempersiapkan proses pensiun tugas, kami sarankan Anda menguji perilaku aplikasi Anda dengan mensimulasikan skenario ini. Anda dapat melakukannya dengan menghentikan tugas individual dalam layanan Anda untuk menguji ketahanannya.

Untuk pensiun tugas mandiri, AWS hentikan tugas pada atau setelah tanggal pensiun tugas. kami tidak meluncurkan tugas pengganti ketika tugas dihentikan. Jika Anda memerlukan tugas-tugas ini untuk terus berjalan, Anda harus menghentikan tugas yang sedang berjalan dan meluncurkan tugas pengganti sebelum waktu yang ditunjukkan dalam pemberitahuan. Oleh karena itu, kami menyarankan agar pelanggan memantau keadaan tugas mandiri dan jika diperlukan, terapkan logika untuk mengganti tugas yang dihentikan.

Ketika tugas dihentikan di salah satu skenario, Anda dapat menjalankannya`describe-tasks`. Yang `stoppedReason` dalam tanggapannya adalah`ECS is performing maintenance on the underlying infrastructure hosting the task`.

Apa yang ada dalam pemberitahuan pensiun tugas?

Pemberitahuan pensiun tugas dikirim melalui AWS Health Dasbor serta melalui email ke alamat email yang terdaftar dan mencakup informasi berikut:

- Tanggal pensiun tugas - Tugas dihentikan pada atau setelah tanggal ini.
- Untuk tugas mandiri, ID tugas.
- Untuk tugas layanan, ID cluster tempat layanan berjalan dan ID layanan.
- Langkah selanjutnya yang perlu Anda ambil.

Biasanya, kami mengirim satu pemberitahuan masing-masing untuk layanan dan tugas mandiri di masing-masing Wilayah AWS. Namun, dalam kasus tertentu Anda mungkin menerima lebih dari satu peristiwa untuk setiap jenis tugas, misalnya ketika ada terlalu banyak tugas yang harus dihentikan yang akan melampaui batas dalam mekanisme pemberitahuan kami.

Anda dapat mengidentifikasi tugas yang dijadwalkan untuk pensiun dengan cara berikut:

- The AWS Health Dashboard

  AWS Health pemberitahuan dapat dikirim melalui Amazon EventBridge ke penyimpanan arsip seperti Amazon Simple Storage Service, mengambil tindakan otomatis seperti menjalankan AWS Lambda fungsi, atau sistem notifikasi lainnya seperti Amazon Simple Notification Service. Untuk informasi selengkapnya, lihat Memantau AWS Health peristiwa dengan Amazon EventBridge. Untuk konfigurasi sampel untuk mengirim notifikasi ke Amazon Chime, Slack, atau Microsoft Teams, lihat repositori AWS Health Aware aktif. GitHub

  Berikut ini adalah contoh EventBridge acara.

```
{
    "version": "0",
    "id": "3c268027-f43c-0171-7425-1d799EXAMPLE",
    "detail-type": "AWS Health Event",
    "source": "aws.health",
    "account": "123456789012",
    "time": "2023-08-16T23:18:51Z",
    "region": "us-east-1",
    "resources": [
        "cluster/service",
        "cluster/service"
    ],
    "detail": {
        "eventArn": "arn:aws:health:us-east-1::event/ECS/
AWS_ECS_TASK_PATCHING_RETIREMENT/AWS_ECS_TASK_PATCHING_RETIREMENT_test1",
        "service": "ECS",
        "eventScopeCode": "ACCOUNT_SPECIFIC",
```

```
        "communicationId":
 "7988399e2e6fb0b905ddc88e0e2de1fd17e4c9fa60349577446d95a18EXAMPLE",
        "lastUpdatedTime": "Wed, 16 Aug 2023 23:18:52 GMT",
        "eventRegion": "us-east-1",
        "eventTypeCode": "AWS_ECS_TASK_PATCHING_RETIREMENT",
        "eventTypeCategory": "scheduledChange",
        "startTime": "Wed, 16 Aug 2023 23:18:51 GMT",
        "endTime": "Fri, 18 Aug 2023 23:18:51 GMT",
        "eventDescription": [
            {
                "language": "en_US",
                "latestDescription": "\\nA software update has been deployed to
 Fargate which includes CVE patches or other critical patches. No action is required
 on your part. All new tasks launched automatically uses the latest software
 version. For existing tasks, your tasks need to be restarted in order for these
 updates to apply. Your tasks running as part of the following ECS Services will
 be automatically updated beginning Wed, 16 Aug 2023 23:18:51 GMT.\\n\\nAfter Wed,
 16 Aug 2023 23:18:51 GMT, the ECS scheduler will gradually replace these tasks,
 respecting the deployment settings for your service. Typically, services should
 see little to no interruption during the update and no action is required. When AWS
 stops tasks, AWS uses the minimum healthy percent (1) and launches a new task in
 an attempt to maintain the desired count for the service. By default, the minimum
 healthy percent of a service is 100 percent, so a new task is started first before
 a task is stopped. Service tasks are routinely replaced in the same way when
 you scale the service or deploy configuration changes or deploy task definition
 revisions. If you would like to control the timing of this restart you can update
 the service before Wed, 16 Aug 2023 23:18:51 GMT, by running the update-service
 command from the ECS command-line interface specifying force-new-deployment for
 services using Rolling update deployment type. For example:\\n\\n$ aws ecs update-
service -service service_name \\\n--cluster cluster_name -force-new-deployment\
\n\\nFor services using Blue/Green deployment type with AWS CodeDeploy:\\nPlease
 refer to create-deployment document (2) and create new deployment using same task
 definition revision.\\n\\nFor further details on ECS deployment types, please
 refer to ECS Deployment Developer Guide (1).\\nFor further details on Fargate's
 update process, please refer to the AWS Fargate User Guide (3).\\nIf you have
 any questions or concerns, please contact AWS Support (4).\\n\\n(1) https://
docs.aws.amazon.com/AmazonECS/latest/developerguide/deployment-types.html\\n(2)
 https://docs.aws.amazon.com/cli/latest/reference/deploy/create-deployment.html\\n(3)
 https://docs.aws.amazon.com/AmazonECS/latest/userguide/task-maintenance.html\\n(4)
 https://aws.amazon.com/support\\n\\nA list of your affected resources(s) can be
 found in the 'Affected resources' tab in the 'Cluster/ Service' format in the AWS
 Health Dashboard. \\n\\n"
            }
        ],
```

```
        "affectedEntities": [
            {
                "entityValue": "cluster/service"
            },
            {
                "entityValue": "cluster/service"
            }
        ]
    }
}
```

- Email

  Email dikirim ke email terdaftar untuk Akun AWS ID.

Bisakah saya mengubah tugas pensiun waktu tunggu?

Anda dapat mengkonfigurasi waktu Fargate memulai tugas pensiun. Untuk beban kerja yang memerlukan aplikasi pembaruan segera, pilih pengaturan langsung (`0`). Ketika Anda membutuhkan kontrol lebih, misalnya, ketika tugas hanya dapat dihentikan selama jendela tertentu, konfigurasikan opsi 7 hari (7), atau 14 hari (14).

Kami menyarankan Anda memilih masa tunggu yang lebih pendek untuk mengambil revisi versi platform yang lebih baru lebih cepat.

Konfigurasikan periode tunggu dengan menjalankan `put-account-setting-default` atau `put-account-setting` sebagai pengguna root atau pengguna administratif. Gunakan `fargateTaskRetirementWaitPeriod` opsi untuk `name` dan `value` opsi yang disetel ke salah satu nilai berikut:

- `0`- AWS mengirim pemberitahuan, dan segera mulai pensiun tugas yang terpengaruh.
- `7`- AWS mengirim pemberitahuan, dan menunggu 7 hari kalender sebelum mulai pensiun tugas yang terpengaruh.
- `14`- AWS mengirim pemberitahuan, dan menunggu 14 hari kalender sebelum mulai pensiun tugas yang terpengaruh.

Defaultnya adalah 7 hari.

Untuk informasi selengkapnya, lihat, put-account-setting-defaultdan put-account-settingdi Referensi API Amazon Elastic Container Service.

Untuk informasi selengkapnya, lihat AWS Fargate tugas pensiun waktu tunggu.

Bisakah saya mendapatkan pemberitahuan pensiun tugas melalui AWS layanan lain?

AWS mengirimkan pemberitahuan pensiun tugas ke AWS Health Dashboard dan ke kontak email utama di Akun AWS. AWS Health Dashboard Ini menyediakan sejumlah integrasi ke AWS layanan lain, termasuk EventBridge. Anda dapat menggunakan EventBridge untuk mengotomatiskan visibilitas pemberitahuan (Misalnya. meneruskan pesan ke alat). ChatOps Untuk informasi selengkapnya, lihat Ringkasan solusi: Menangkap pemberitahuan pensiun tugas.

Bisakah saya mengubah pensiun tugas setelah dijadwalkan?

Tidak. Jadwal didasarkan pada tugas waktu tunggu pensiun yang memiliki default 7 hari. Jika Anda membutuhkan lebih banyak waktu, Anda dapat memilih untuk mengkonfigurasi periode tunggu hingga 14 hari. Untuk informasi selengkapnya, lihat Bisakah saya mengubah tugas pensiun waktu tunggu?. Perubahan dalam konfigurasi ini berlaku untuk pensiun yang akan dijadwalkan di masa depan. Pensiun yang dijadwalkan saat ini tidak terpengaruh. Jika Anda memiliki masalah lebih lanjut, hubungi AWS Support.

Bisakah saya mengontrol waktu penggantian tugas?

Untuk layanan yang menggunakan penerapan bergulir, Anda memperbarui layanan menggunakan `update-service` `force-deployment` opsi sebelum waktu mulai pensiun.

`update-service`Contoh berikut menggunakan `force-deployment` opsi.

```
aws ecs update-service –service service_name \
    --cluster cluster_name \
     --force-new-deployment
```

Untuk layanan yang menggunakan penerapan biru/hijau, Anda perlu membuat penerapan baru di. AWS CodeDeployUntuk informasi tentang cara membuat penerapan, lihat create-deployment di Referensi.AWS Command Line Interface

Bagaimana Amazon ECS menangani tugas yang merupakan bagian dari layanan?

Amazon ECS secara bertahap menggantikan tugas yang terpengaruh dalam layanan Anda ketika masa pensiun Fargate dimulai. Saat Amazon ECS menghentikan tugas, Amazon menggunakan persen sehat minimum layanan dan meluncurkan tugas baru untuk mempertahankan jumlah tugas yang diinginkan untuk layanan. Tugas baru dimulai sebelum tugas dihentikan karena persen kesehatan minimum default adalah 100. Tugas layanan diganti secara rutin dengan cara yang sama saat Anda menskalakan layanan, menerapkan perubahan konfigurasi, atau menerapkan revisi definisi tugas. Untuk informasi lebih lanjut tentang persen sehat minimum, lihatKonfigurasi deployment.

Bisakah Amazon ECS secara otomatis menangani tugas mandiri?

Tidak. AWS tidak dapat membuat tugas pengganti untuk tugas mandiri yang dimulai oleh`RunTask`, tugas terjadwal (misalnya melalui EventBridge Scheduler) AWS Batch, atau. AWS Step Functions Amazon ECS hanya mengelola tugas yang merupakan bagian dari layanan.

Wilayah yang Didukung untuk Amazon ECS di Fargate AWS

Anda dapat menggunakan tabel berikut untuk memverifikasi dukungan Wilayah untuk kontainer Linux di AWS Fargate dan kontainer Windows di AWS Fargate.

Kontainer Linux di AWS Fargate

Kontainer Amazon ECS Linux aktif AWS Fargate didukung dalam hal berikut Wilayah AWS ini. ID Availability Zone yang didukung dicatat saat berlaku.

Nama Wilayah	Wilayah
US East (Ohio)	us-east-2
US East (N. Virginia)	us-east-1
AS Barat (California Utara)	us-barat-1 (& hanya) `usw1-az1` `usw1-az3`
AS Barat (Oregon)	as-barat-2
Afrika (Cape Town)	af-selatan-1

Nama Wilayah	Wilayah
Asia Pasifik (Hong Kong)	ap-timur-1
Asia Pasifik (Mumbai)	ap-south-1
Asia Pasifik (Tokyo)	ap-timur laut-1 (,, & hanya) `apne1-az1` `apne1-az2` `apne1-az4`
Asia Pasifik (Seoul)	ap-northeast-2
Asia Pacific (Osaka)	ap-northeast-3
Asia Pasifik (Hyderabad)	ap-south-2
Asia Pasifik (Singapura)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pasifik (Jakarta)	ap-southeast-3
Asia Pacific (Melbourne)	ap-southeast-4
Kanada (Pusat)	ca-central-1
Kanada Barat (Calgary)	ca-west-1
Tiongkok (Beijing)	cn-north-1 (& hanya) `cnn1-az1` `cnn1-az2`
Tiongkok (Ningxia)	cn-northwest-1
Europe (Frankfurt)	eu-central-1
Eropa (Zürich)	eu-central-2
Eropa (Irlandia)	eu-west-1
Europe (London)	eu-west-2
Europe (Paris)	eu-west-3
Eropa (Milan)	eu-south-1

Nama Wilayah	Wilayah
Eropa (Spanyol)	eu-south-2
Eropa (Stockholm)	eu-north-1
South America (Sao Paulo)	sa-east-1
Israel (Tel Aviv)	il-central-1
Timur Tengah (Bahrain)	me-south-1
Timur Tengah (UEA)	me-central-1
AWS GovCloud (AS-Timur)	us-gov-east-1
AWS GovCloud (AS-Barat)	us-gov-west-1

Wadah Windows di AWS Fargate

Amazon ECS Windows container AWS Fargate aktif didukung sebagai berikut Wilayah AWS. ID Availability Zone yang didukung dicatat saat berlaku.

Nama Wilayah	Wilayah
US East (Ohio)	us-east-2
AS Timur (Virginia Utara)	us-timur-1 (`use1-az1`,,, `use1-az2` `use1-az4``use1-az5`, & hanya) `use1-az6`
AS Barat (California Utara)	us-barat-1 (& hanya) `usw1-az1` `usw1-az3`
AS Barat (Oregon)	as-barat-2
Afrika (Cape Town)	af-selatan-1
Asia Pasifik (Hong Kong)	ap-timur-1
Asia Pasifik (Mumbai)	ap-south-1

Nama Wilayah	Wilayah
Asia Pasifik (Hyderabad)	ap-south-2
Asia Pacific (Osaka)	ap-northeast-3
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pasifik (Melbourne)	ap-southeast-4
Asia Pasifik (Tokyo)	ap-timur laut-1 (,, & hanya) `apne1-az1` `apne1-az2` `apne1-az4`
Kanada (Pusat)	ca-central-1 (& hanya) `cac1-az1` `cac1-az2`
Kanada Barat (Calgary)	ca-west-1
Tiongkok (Beijing)	cn-north-1 (& hanya) `cnn1-az1` `cnn1-az2`
Tiongkok (Ningxia)	cn-northwest-1
Europe (Frankfurt)	eu-central-1
Eropa (Zürich)	eu-central-2
Eropa (Irlandia)	eu-west-1
Europe (London)	eu-west-2
Europe (Paris)	eu-west-3
Eropa (Milan)	eu-south-1
Eropa (Spanyol)	eu-south-2
Eropa (Stockholm)	eu-north-1
South America (Sao Paulo)	sa-east-1

Nama Wilayah	Wilayah
Israel (Tel Aviv)	il-central-1
Timur Tengah (UEA)	me-central-1
Timur Tengah (Bahrain)	me-south-1

Merancang solusi Anda untuk Amazon ECS

Sebelum menggunakan Amazon ECS, Anda perlu membuat keputusan tentang kapasitas, jaringan, pengaturan akun, dan pencatatan sehingga Anda dapat mengonfigurasi sumber daya Amazon ECS dengan benar.

Kapasitas Amazon ECS

Kapasitasnya adalah infrastruktur tempat kontainer Anda berjalan. Berikut ini adalah opsinya:

- Instans Amazon EC2
- Tanpa Server ()AWS Fargate (Fargate)
- Mesin virtual lokal (VM) atau server

Anda menentukan infrastruktur saat membuat klaster. Anda juga menentukan jenis infrastruktur saat mendaftarkan definisi tugas. Definisi tugas mengacu pada infrastruktur sebagai “tipe peluncuran”. Anda juga menggunakan jenis peluncuran saat menjalankan tugas mandiri atau menerapkan layanan. Untuk informasi tentang opsi jenis peluncuran, lihatJenis peluncuran Amazon ECS.

Jaringan

AWS sumber daya dibuat dalam subnet. Saat Anda menggunakan instans EC2, Amazon ECS meluncurkan instance di subnet yang Anda tentukan saat membuat klaster. Tugas Anda berjalan di subnet instance. Untuk Fargate atau mesin virtual lokal, Anda menentukan subnet saat menjalankan tugas atau membuat layanan.

Tergantung pada aplikasi Anda, subnet dapat berupa subnet pribadi atau publik dan subnet dapat berada di salah satu sumber daya berikut: AWS

- Zona Ketersediaan
- Zona Lokal
- Wavelength Zones
- Wilayah AWS
- AWS Outposts

Untuk informasi selengkapnya, lihat Aplikasi Amazon ECS di subnet bersama, Local Zones, dan Wavelength Zones atau Layanan Kontainer Elastis Amazon aktif AWS Outposts.

Anda dapat membuat aplikasi Anda terhubung ke internet dengan menggunakan salah satu metode berikut:

- Subnet publik dengan gateway internet

  Gunakan subnet publik ketika Anda memiliki aplikasi publik yang membutuhkan bandwidth dalam jumlah besar atau latensi minimal. Skenario yang berlaku termasuk streaming video dan layanan game.
- Subnet pribadi dengan gateway NAT

  Gunakan subnet pribadi saat Anda ingin melindungi wadah Anda dari akses eksternal langsung. Skenario yang berlaku termasuk sistem pemrosesan pembayaran atau wadah yang menyimpan data pengguna dan kata sandi.

Mengakses fitur

Anda dapat menggunakan pengaturan akun Amazon ECS untuk mengakses fitur-fitur berikut:

- Wawasan Kontainer

  CloudWatch Container Insights mengumpulkan, mengumpulkan, dan merangkum metrik dan log dari aplikasi dan layanan mikro dalam kontainer Anda. Metrik tersebut mencakup pemanfaatan sumber daya seperti CPU, memori, disk, dan jaringan.
- `awsvpc`trunking

  Untuk jenis instans EC2 tertentu, Anda dapat memiliki antarmuka jaringan tambahan (ENI) yang tersedia pada instans kontainer yang baru diluncurkan.
- Otorisasi penandaan

  Pengguna harus memiliki izin untuk tindakan yang membuat sumber daya, seperti`ecsCreateCluster`. Jika tag ditentukan dalam tindakan pembuatan sumber daya, AWS lakukan otorisasi tambahan pada `ecs:TagResource` tindakan untuk memverifikasi apakah pengguna atau peran memiliki izin untuk membuat tag.
- Kepatuhan Fargate FIPS-140

Fargate mendukung Federal Information Processing Standard (FIPS-140) yang menetapkan persyaratan keamanan untuk modul kriptografi yang melindungi informasi sensitif. Ini adalah standar pemerintah Amerika Serikat dan Kanada saat ini, dan berlaku untuk sistem yang harus mematuhi Undang-Undang Manajemen Keamanan Informasi Federal (FISMA) atau Program Manajemen Risiko dan Otorisasi Federal (FedRAMP).

- Perubahan waktu pensiun tugas Fargate

  Anda dapat mengonfigurasi periode tunggu sebelum tugas Fargate dihentikan untuk ditambal.

- VPC tumpukan ganda

  Izinkan tugas untuk berkomunikasi melalui IPv4, IPv6, atau keduanya.

- Format Amazon Resource Name (ARN)

  Fitur tertentu, seperti otorisasi penandaan, memerlukan format Amazon Resource Name (ARN) baru.

Untuk informasi selengkapnya, lihat Mengakses fitur Amazon ECS melalui pengaturan akun.

Pencatatan log

Pencatatan dan pemantauan adalah aspek penting dalam menjaga keandalan, ketersediaan, dan kinerja beban kerja Amazon ECS. Pilihan berikut tersedia:

- Amazon CloudWatch log - rute log ke Amazon CloudWatch
- FireLensuntuk Amazon ECS - rute log ke AWS layanan atau AWS Partner Network tujuan untuk penyimpanan dan analisis log. AWS Partner Network Ini adalah komunitas mitra global yang memanfaatkan program, keahlian, dan sumber daya untuk membangun, memasarkan, dan menjual penawaran pelanggan.

Jenis peluncuran Amazon ECS

Jenis peluncuran definisi tugas mendefinisikan kapasitas apa yang dapat dijalankan tugas, misalnya AWS Fargate.

Setelah Anda memilih jenis peluncuran, Amazon ECS memverifikasi bahwa parameter definisi tugas yang Anda konfigurasikan berfungsi dengan jenis peluncuran.

Jenis peluncuran Fargate

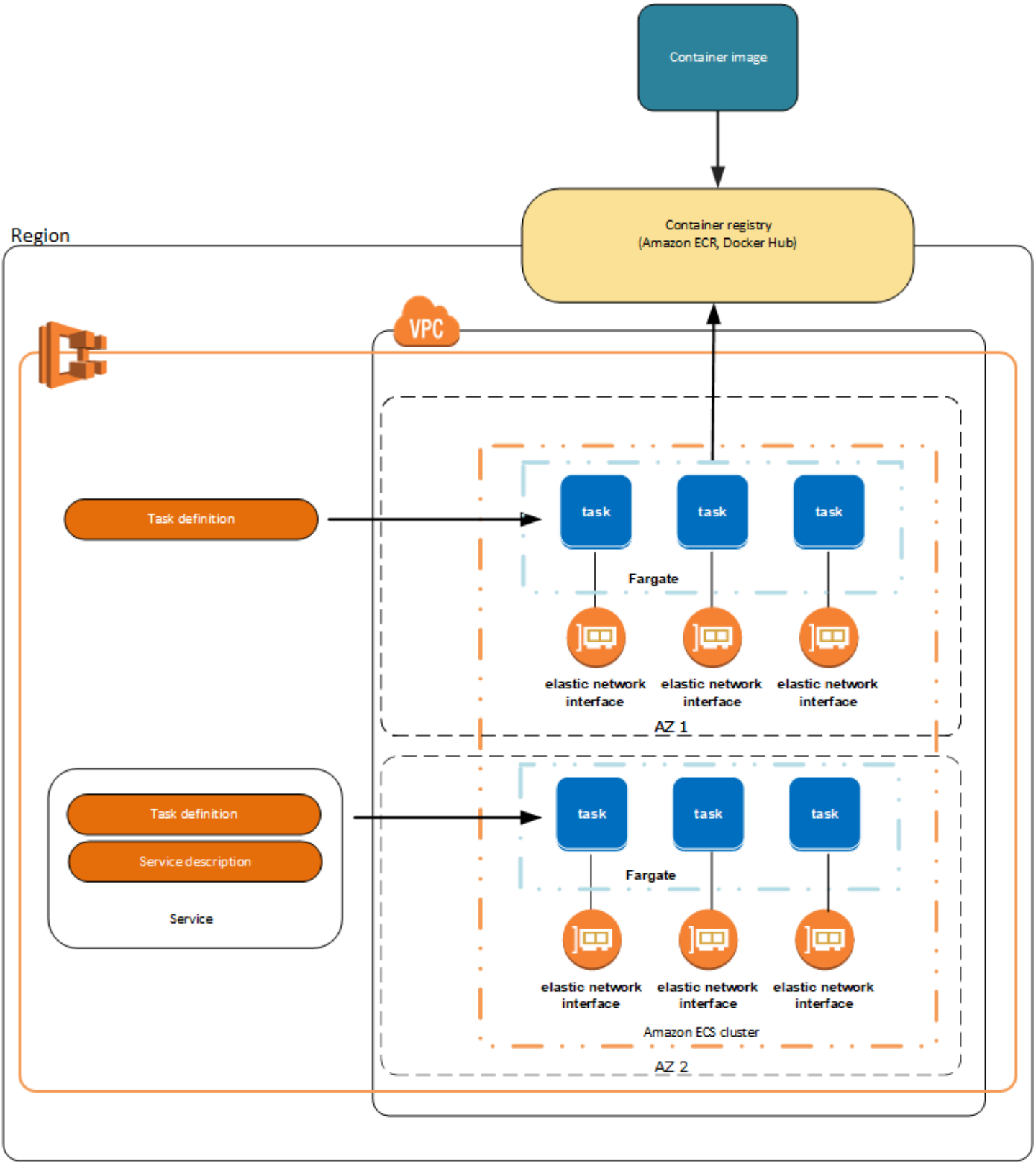
Anda dapat menggunakan tipe peluncuran Fargate untuk menjalankan aplikasi kontainer Anda tanpa perlu menyediakan dan mengelola infrastruktur yang mendasarinya. AWS Fargate adalah cara tanpa server untuk meng-host beban kerja Amazon ECS Anda.

Jenis peluncuran Fargate cocok untuk beban kerja berikut:

- Beban kerja besar yang membutuhkan overhead operasional rendah
- Beban kerja kecil yang sesekali meledak
- Beban kerja kecil
- Beban kerja batch

Untuk informasi tentang Daerah yang mendukung Fargate, lihat. the section called “AWS Wilayah Fargate”

Diagram berikut menunjukkan arsitektur umum.



Untuk informasi selengkapnya tentang Amazon ECS di Fargate, lihat. [Amazon ECS aktif AWS Fargate]

Jenis peluncuran EC2

Jenis peluncuran EC2 cocok untuk beban kerja besar yang harus dioptimalkan harga.

Saat mempertimbangkan cara memodelkan definisi tugas dan layanan menggunakan tipe peluncuran EC2, kami sarankan Anda mempertimbangkan proses apa yang harus dijalankan bersama dan bagaimana Anda dapat melakukan penskalaan setiap komponen.

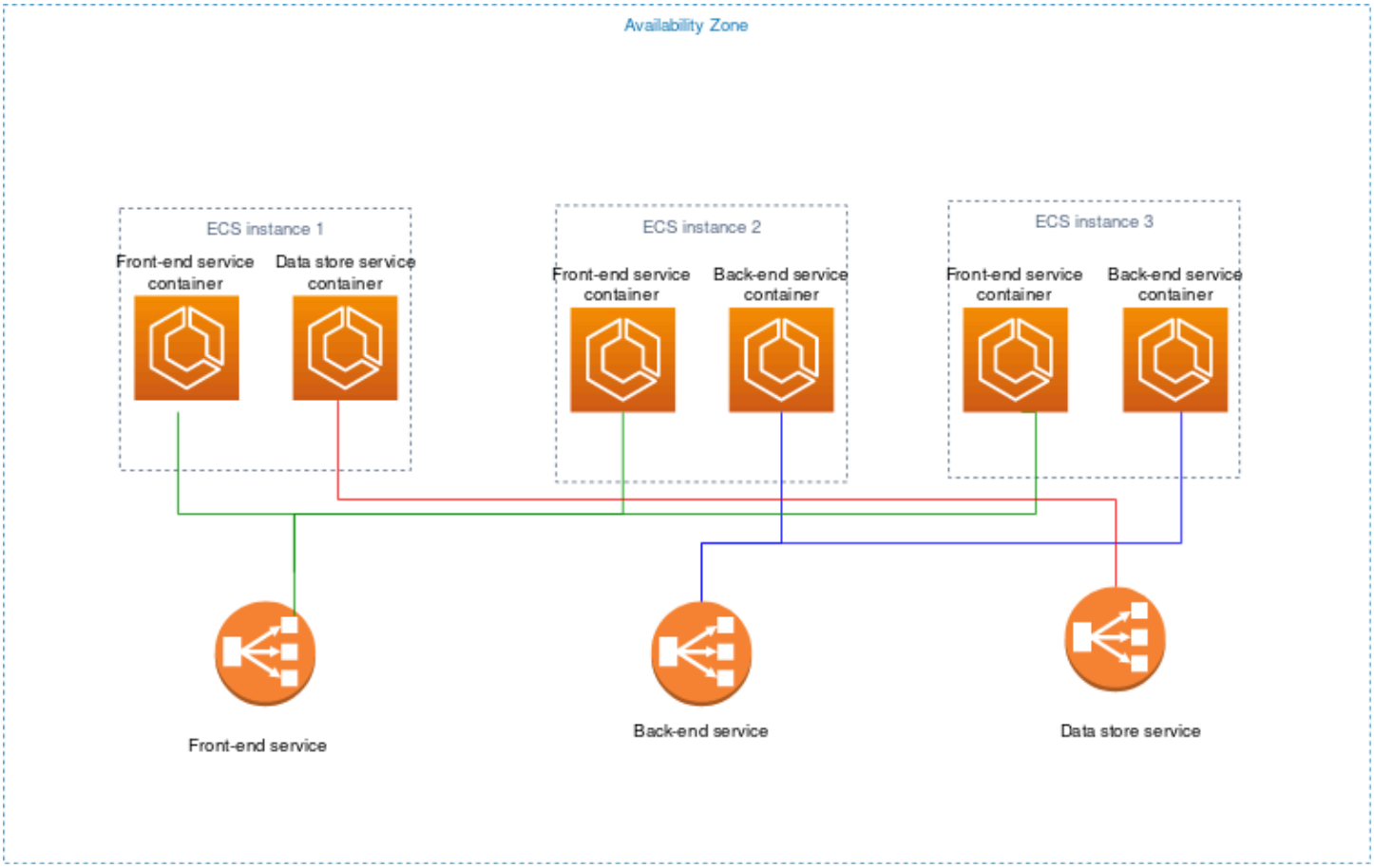
Sebagai contoh, misalkan aplikasi terdiri dari komponen-komponen berikut:

- Sebuah layanan frontend yang menampilkan informasi pada halaman web
- Sebuah layanan backend yang menyediakan API untuk layanan frontend
- Penyimpanan data

Untuk contoh ini, buat definisi tugas yang mengelompokkan kontainer yang digunakan untuk tujuan bersama. Pisahkan komponen yang berbeda menjadi beberapa definisi tugas yang terpisah. Contoh cluster berikut memiliki tiga instance kontainer yang menjalankan tiga kontainer layanan front-end, dua kontainer layanan backend, dan satu kontainer layanan penyimpanan data.

Anda dapat mengelompokkan kontainer terkait dalam ketentuan tugas, seperti kontainer terkait yang harus dijalankan bersama-sama. Misalnya, tambahkan wadah streaming log ke layanan front-end Anda dan sertakan dalam definisi tugas yang sama.

Setelah Anda memiliki ketentuan tugas, Anda dapat membuat layanan dari mereka untuk menjaga ketersediaan tugas yang Anda inginkan. Untuk informasi selengkapnya, lihat Membuat layanan menggunakan konsol. Dalam layanan Anda, Anda dapat mengaitkan kontainer dengan penyeimbang beban Elastic Load Balancing. Untuk informasi selengkapnya, lihat Mendistribusikan lalu lintas layanan Amazon ECS menggunakan load balancing. Ketika persyaratan aplikasi Anda berubah, Anda dapat memperbarui layanan Anda untuk meningkatkan atau menurunkan jumlah tugas yang diinginkan. Atau, Anda dapat memperbarui layanan untuk menerapkan versi kontainer yang lebih baru dalam tugas Anda. Untuk informasi selengkapnya, lihat Memperbarui layanan menggunakan konsol.



Jenis peluncuran eksternal

Jenis peluncuran Eksternal digunakan untuk menjalankan aplikasi kontainer di server lokal atau mesin virtual (VM) yang Anda daftarkan ke kluster Amazon ECS dan kelola dari jarak jauh. Untuk informasi selengkapnya, lihat Instans eksternal (Amazon ECS Anywhere).

Aplikasi Amazon ECS di subnet bersama, Local Zones, dan Wavelength Zones

Amazon ECS mendukung beban kerja yang menggunakan Local Zones, Wavelength Zones, dan AWS Outposts untuk saat latensi rendah atau pemrosesan data lokal merupakan persyaratan.

- Anda dapat menggunakan Local Zones sebagai perpanjangan Wilayah AWS untuk menempatkan sumber daya di beberapa lokasi yang lebih dekat dengan pengguna akhir Anda.
- Anda dapat menggunakan Wavelength Zones untuk membangun aplikasi yang memberikan latensi ultra-rendah ke perangkat 5G dan pengguna akhir. Wavelength menyebarkan layanan komputasi dan penyimpanan AWS standar ke tepi jaringan 5G operator telekomunikasi.

- AWS Outposts membawa model asli Layanan AWS, infrastruktur, dan operasi ke hampir semua pusat data, ruang co-lokasi, atau fasilitas lokal.

> **Important**
>
> Amazon ECS pada AWS Fargate beban kerja tidak didukung di Local Zones, Wavelength Zone, atau aktif saat ini. AWS Outposts

Untuk informasi tentang perbedaan antara Local Zones, Wavelength Zones, AWS Outposts dan, lihat Bagaimana saya harus memikirkan kapan harus menggunakan AWS Wavelength, AWS Local Zones, atau untuk aplikasi yang membutuhkan latensi rendah atau pemrosesan data lokal di AWS Outposts FAQ. AWS Wavelength

Subnet bersama

Anda dapat menggunakan berbagi VPC untuk berbagi subnet dengan AWS akun lain dalam hal yang sama. AWS Organizations

Anda dapat menggunakan VPC bersama untuk jenis peluncuran EC2 dengan pertimbangan berikut:

- Pemilik subnet VPC harus berbagi subnet dengan akun peserta sebelum akun tersebut dapat menggunakannya untuk sumber daya Amazon ECS.
- Anda tidak dapat menggunakan grup keamanan default VPC untuk instance container Anda karena milik pemiliknya. Selain itu, peserta tidak dapat meluncurkan instance menggunakan grup keamanan yang dimiliki oleh peserta lain atau pemilik.
- Dalam subnet bersama, peserta dan pemilik secara terpisah mengontrol grup keamanan dalam setiap akun masing-masing. Pemilik subnet dapat melihat grup keamanan yang dibuat oleh peserta tetapi tidak dapat melakukan tindakan apa pun pada mereka. Jika pemilik subnet ingin menghapus atau memodifikasi grup keamanan ini, peserta yang membuat grup keamanan harus mengambil tindakan.
- Pemilik VPC bersama tidak dapat melihat, memperbarui, atau menghapus klaster yang dibuat peserta di subnet bersama. Ini merupakan tambahan dari sumber daya VPC yang setiap akun memiliki akses berbeda. Untuk informasi selengkapnya, lihat Tanggung jawab dan izin untuk pemilik dan peserta di Panduan Pengguna Amazon VPC.

Anda dapat menggunakan VPC bersama untuk jenis peluncuran Fargate dengan pertimbangan berikut:

- Pemilik subnet VPC harus berbagi subnet dengan akun peserta sebelum akun tersebut dapat menggunakannya untuk sumber daya Amazon ECS.
- Anda tidak dapat membuat layanan atau menjalankan tugas menggunakan grup keamanan default untuk VPC karena milik pemilik. Selain itu, peserta tidak dapat membuat layanan atau menjalankan tugas menggunakan grup keamanan yang dimiliki oleh peserta lain atau pemilik.
- Dalam subnet bersama, peserta dan pemilik secara terpisah mengontrol grup keamanan dalam setiap akun masing-masing. Pemilik subnet dapat melihat grup keamanan yang dibuat oleh peserta tetapi tidak dapat melakukan tindakan apa pun pada mereka. Jika pemilik subnet ingin menghapus atau memodifikasi grup keamanan ini, peserta yang membuat grup keamanan harus mengambil tindakan.
- Pemilik VPC bersama tidak dapat melihat, memperbarui, atau menghapus klaster yang dibuat peserta di subnet bersama. Ini merupakan tambahan dari sumber daya VPC yang setiap akun memiliki akses berbeda. Untuk informasi selengkapnya, lihat Tanggung jawab dan izin untuk pemilik dan peserta di Panduan Pengguna Amazon VPC.

Untuk informasi selengkapnya tentang berbagi subnet VPC, lihat Bagikan VPC Anda dengan akun lain di Panduan Pengguna Amazon VPC.

Zona Lokal

Zona Lokal adalah perpanjangan Wilayah AWS dari jarak geografis yang dekat dengan pengguna Anda. Zona Lokal memiliki koneksinya sendiri ke internet dan mendukung AWS Direct Connect. Sumber daya yang dibuat di Zona Lokal dapat melayani pengguna lokal dengan komunikasi latensi rendah. Untuk informasi selengkapnya, lihat Zona Lokal AWS.

Zona Lokal diwakili oleh kode Wilayah diikuti oleh pengidentifikasi yang menunjukkan lokasi (misalnya,`us-west-2-lax-1a`).

Untuk menggunakan Zona Lokal, Anda harus ikut serta dalam zona tersebut. Setelah Anda ikut serta, Anda harus membuat VPC Amazon dan subnet di Zona Lokal.

Anda dapat meluncurkan instans Amazon EC2, server file Amazon FSx, dan Application Load Balancer untuk digunakan untuk klaster dan tugas Amazon ECS Anda.

Untuk informasi selengkapnya, lihat Local Zones di Panduan Pengguna Amazon EC2 untuk Instans Linux.

Wavelength Zones

Anda dapat menggunakannya AWS Wavelengthuntuk membangun aplikasi yang memberikan latensi sangat rendah ke perangkat seluler dan pengguna akhir. Wavelength menyebarkan layanan komputasi dan penyimpanan AWS standar ke tepi jaringan 5G operator telekomunikasi. Anda dapat memperluas Amazon Virtual Private Cloud ke satu atau beberapa Wavelength Zone. Kemudian, Anda dapat menggunakan AWS sumber daya seperti instans Amazon EC2 untuk menjalankan aplikasi yang memerlukan latensi sangat rendah dan koneksi ke di Wilayah. Layanan AWS

Wavelength Zone adalah Zona terisolasi di lokasi pembawa tempat infrastruktur Wavelength dikerahkan. Zona Wavelength terikat pada sebuah. Wilayah AWS Zona Panjang Gelombang adalah perpanjangan logis dari Wilayah, dan dikelola oleh bidang kontrol di Wilayah.

Zona Wavelength diwakili oleh kode Wilayah diikuti oleh pengidentifikasi yang menunjukkan Zona Wavelength (misalnya,). `us-east-1-wl1-bos-wlz-1`

Untuk menggunakan Wavelength Zone, Anda harus memilih Zona. Setelah Anda ikut serta, Anda harus membuat Amazon VPC dan subnet di Wavelength Zone. Kemudian, Anda dapat meluncurkan instans Amazon EC2 di Zona yang akan digunakan untuk klaster dan tugas Amazon ECS Anda.

Untuk informasi selengkapnya, lihat Memulai dengan AWS Wavelength di Panduan Developer AWS Wavelength .

Wavelength Zones tidak tersedia di semua. Wilayah AWS Untuk informasi tentang Wilayah yang mendukung Wavelength Zones, lihat Wavelength Zones yang Tersedia di Panduan Developer AWS Wavelength .

Layanan Kontainer Elastis Amazon aktif AWS Outposts

AWS Outposts memungkinkan AWS layanan asli, infrastruktur, dan model operasi di fasilitas lokal. Di AWS Outposts lingkungan, Anda dapat menggunakan AWS API, alat, dan infrastruktur yang sama dengan yang Anda gunakan di AWS Cloud. Amazon ECS aktif AWS Outposts sangat ideal untuk beban kerja latensi rendah yang perlu dijalankan di dekat data dan aplikasi lokal. Untuk informasi selengkapnya AWS Outposts, lihat Panduan AWS Outposts Pengguna.

Prasyarat

Berikut ini adalah prasyarat untuk menggunakan Amazon ECS pada: AWS Outposts

- Anda harus telah menginstal dan mengkonfigurasi AWS Outposts di pusat data lokal Anda.
- Anda harus memiliki koneksi jaringan yang andal antara Anda AWS Outposts dan AWS Wilayahnya.
- Anda harus memiliki kapasitas yang cukup dari tipe instans yang tersedia di AWS Outposts Anda.
- Semua instans kontainer Amazon ECS harus memiliki agen kontainer Amazon ECS 1.33.0 atau yang lebih baru.

Batasan

Berikut ini adalah batasan penggunaan Amazon ECS pada AWS Outposts:

- Amazon Elastic Container Registry AWS Identity and Access Management,, dan Network Load Balancer berjalan di AWS Wilayah, bukan di. AWS Outposts Hal ini akan meningkatkan latensi antara layanan ini dan kontainer.
- AWS Fargate tidak tersedia di AWS Outposts.

Pertimbangan Konektivitas Jaringan

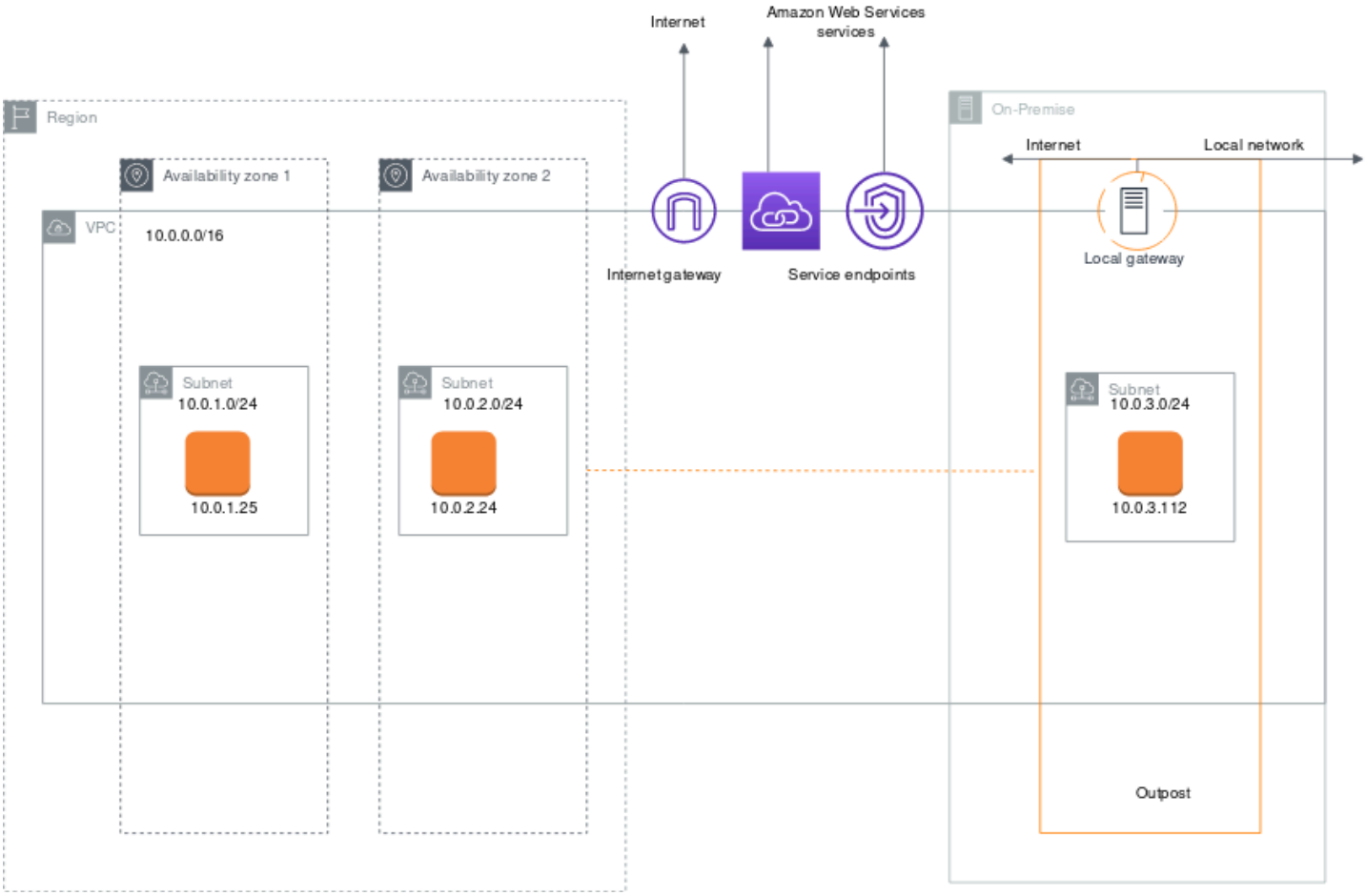
Berikut ini adalah pertimbangan konektivitas jaringan untuk AWS Outposts:

- Jika konektivitas jaringan antara Anda AWS Outposts dan AWS Wilayahnya hilang, cluster Anda akan terus berjalan. Namun, Anda tidak dapat membuat klaster baru atau mengambil tindakan baru pada klaster yang ada sampai konektivitas dipulihkan. Dalam kasus kegagalan instans, instans tidak akan diganti secara otomatis. Agen CloudWatch Log tidak akan dapat memperbarui log dan data peristiwa.
- Kami menyarankan Anda menyediakan konektivitas yang andal, sangat tersedia, dan latensi rendah antara Anda AWS Outposts dan AWS Wilayahnya.

Membuat Cluster Amazon ECS di AWS Outposts

Membuat cluster Amazon ECS pada sebuah AWS Outposts mirip dengan membuat cluster Amazon ECS di AWS Cloud. Saat Anda membuat cluster Amazon ECS di sebuah AWS Outposts, Anda harus menentukan subnet yang terkait dengan Anda. AWS Outposts

An AWS Outposts adalah perpanjangan dari AWS Wilayah, dan Anda dapat memperluas VPC Amazon di akun untuk menjangkau beberapa Availability Zone dan yang terkait. AWS Outposts Saat mengonfigurasi AWS Outposts, Anda mengaitkan subnet dengannya untuk memperluas lingkungan VPC Regional Anda ke fasilitas lokal Anda. Instans pada AWS Outposts muncul sebagai bagian dari VPC Regional Anda, mirip dengan Availability Zone dengan subnet terkait.



AWS CLI

Untuk membuat cluster Amazon ECS di AWS Outposts with AWS CLI, tentukan grup keamanan dan subnet untuk dikaitkan dengan Anda. AWS Outposts

Untuk membuat subnet yang terkait dengan Anda AWS Outposts.

```
aws ec2 create-subnet \
    --cidr-block 10.0.3.0/24 \
    --vpc-id vpc-xxxxxxxx \
    --outpost-arn arn:aws:outposts:us-west-2:123456789012:outpost/op-xxxxxxxxxxxxxxxx \
    --availability-zone-id usw2-az1
```

Contoh berikut membuat cluster Amazon ECS pada file AWS Outposts.

1. Buat peran dan kebijakan dengan hak pada AWS Outposts.

   `role-policy.json`File adalah dokumen kebijakan yang berisi efek dan tindakan untuk sumber daya. Untuk informasi tentang format file, lihat PutRolePolicydi Referensi API IAM

   ```
   aws iam create-role --role-name ecsRole \
       --assume-role-policy-document file://ecs-policy.json
   aws iam put-role-policy --role-name ecsRole --policy-name ecsRolePolicy \
       --policy-document file://role-policy.json
   ```

2. Buat profil instans IAM dengan hak pada AWS Outposts.

   ```
   aws iam create-instance-profile --instance-profile-name outpost
   aws iam add-role-to-instance-profile --instance-profile-name outpost \
       --role-name ecsRole
   ```

3. Buat sebuah VPC.

   ```
   aws ec2 create-vpc --cidr-block 10.0.0.0/16
   ```

4. Membuat grup keamanan untuk instans kontainer, menentukan kisaran CIDR yang tepat untuk AWS Outposts. (Langkah ini berbeda untuk AWS Outposts.)

   ```
   aws ec2 create-security-group --group-name MyOutpostSG
   aws ec2 authorize-security-group-ingress --group-name MyOutpostSG --protocol tcp \
       --port 22 --cidr 10.0.3.0/24
   aws ec2 authorize-security-group-ingress --group-name MyOutpostSG --protocol tcp \
       --port 80 --cidr 10.0.3.0/24
   ```

5. Buat Klaster.

6. Tentukan variabel lingkungan agen penampung Amazon ECS untuk meluncurkan instance ke dalam klaster yang dibuat pada langkah sebelumnya dan tentukan tag apa pun yang ingin Anda

tambahkan untuk membantu mengidentifikasi klaster (misalnya, `Outpost` untuk menunjukkan bahwa klaster tersebut untuk Outpost).

```
#! /bin/bash
cat << 'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_IMAGE_PULL_BEHAVIOR=prefer-cached
ECS_CONTAINER_INSTANCE_TAGS={"environment": "Outpost"}
EOF
```

> **Note**
>
> Untuk menghindari penundaan yang disebabkan oleh penarikan gambar kontainer dari Amazon ECR di Wilayah, gunakan cache gambar. Untuk melakukan ini, setiap kali tugas dijalankan, konfigurasikan agen Amazon ECS ke default untuk menggunakan gambar yang di-cache pada instance itu sendiri dengan menyetel `ECS_IMAGE_PULL_BEHAVIOR` ke. `prefer-cached`

7. Membuat instans kontainer, menentukan VPC dan subnet untuk AWS Outposts tempat instans ini harus dijalankan dan tipe instans yang tersedia di AWS Outposts. (Langkah ini berbeda untuk AWS Outposts.)

   `userdata.txt`File berisi data pengguna yang dapat digunakan instance untuk melakukan tugas konfigurasi otomatis umum dan bahkan menjalankan skrip setelah instance dimulai. Untuk informasi tentang file untuk panggilan API, lihat Menjalankan perintah pada instance Linux Anda saat diluncurkan di Panduan Pengguna Amazon EC2 untuk Instans Linux.

   ```
   aws ec2 run-instances --count 1 --image-id ami-xxxxxxxx --instance-type c5.large \
       --key-name aws-outpost-key --subnet-id subnet-xxxxxxxxxxxxxxxxx \
       --iam-instance-profile Name outpost --security-group-id sg-xxxxxx \
       --associate-public-ip-address --user-data file://userdata.txt
   ```

   > **Note**
   >
   > Perintah ini juga digunakan saat menambahkan instans tambahan ke klaster. Setiap kontainer di-deploy di klaster yang akan ditempatkan pada AWS Outposts yang spesifik.

8. Daftarkan definisi tugas Anda. Gunakan perintah berikut dan ganti `ecs-task.json` dengan nama definisi tugas Anda.

```
aws ecs register-task-definition --cli-input-json file://ecs-task.json
```

9. Jalankan tugas atau buat layanan.

   Run the task

```
aws ecs run-task --cluster mycluster --count 1 --task-definition outpost-app:1
```

   Create the service

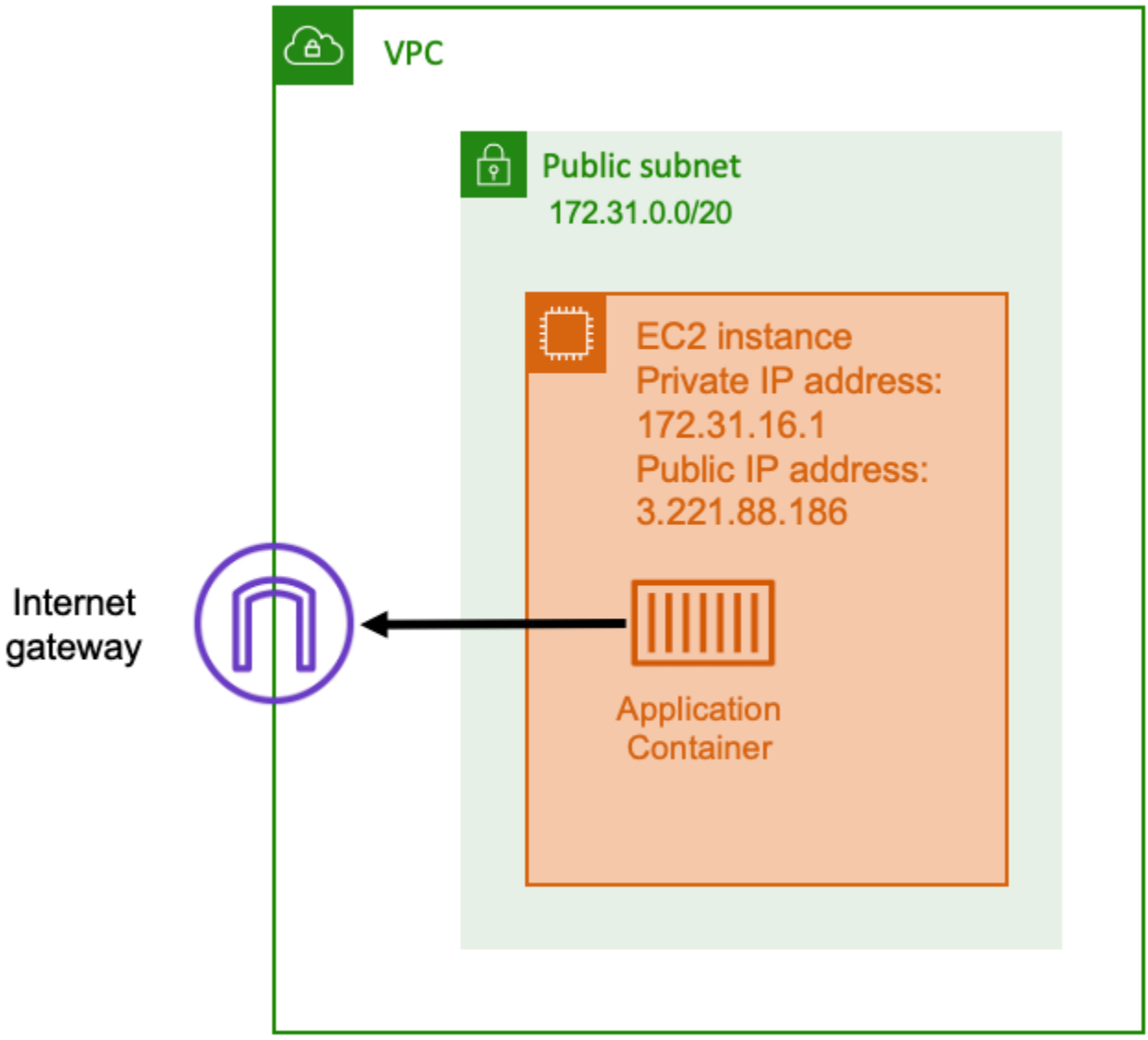
```
aws ecs create-service --cluster mycluster --service-name outpost-service \
    --task-definition outpost-app:1 --desired-count 1
```

Praktik terbaik untuk menghubungkan aplikasi Amazon ECS ke internet

Sebagian besar aplikasi kontainer memiliki setidaknya beberapa komponen yang memerlukan akses keluar ke internet. Misalnya, backend untuk aplikasi seluler memerlukan akses keluar ke pemberitahuan push.

Amazon Virtual Private Cloud memiliki dua metode utama untuk memfasilitasi komunikasi antara VPC Anda dan internet.

Subnet publik dan gateway internet



Bila Anda menggunakan subnet publik yang memiliki rute ke gateway internet, aplikasi containerized Anda dapat berjalan pada host di dalam VPC pada subnet publik. Host yang menjalankan container Anda diberi alamat IP publik. Alamat IP publik ini dapat dirutekan dari internet. Untuk informasi lebih lanjut, lihat Gateway internet di Panduan Pengguna Amazon VPC.

Arsitektur jaringan ini memfasilitasi komunikasi langsung antara host yang menjalankan aplikasi Anda dan host lain di internet. Komunikasi bersifat bi-directional. Ini berarti bahwa Anda tidak hanya dapat membuat koneksi keluar ke host lain di internet, tetapi host lain di internet mungkin juga mencoba untuk terhubung ke host Anda. Karena itu, Anda harus memperhatikan grup keamanan dan aturan

firewall Anda. Ini memastikan bahwa host lain di internet tidak dapat membuka koneksi apa pun yang tidak ingin Anda buka.

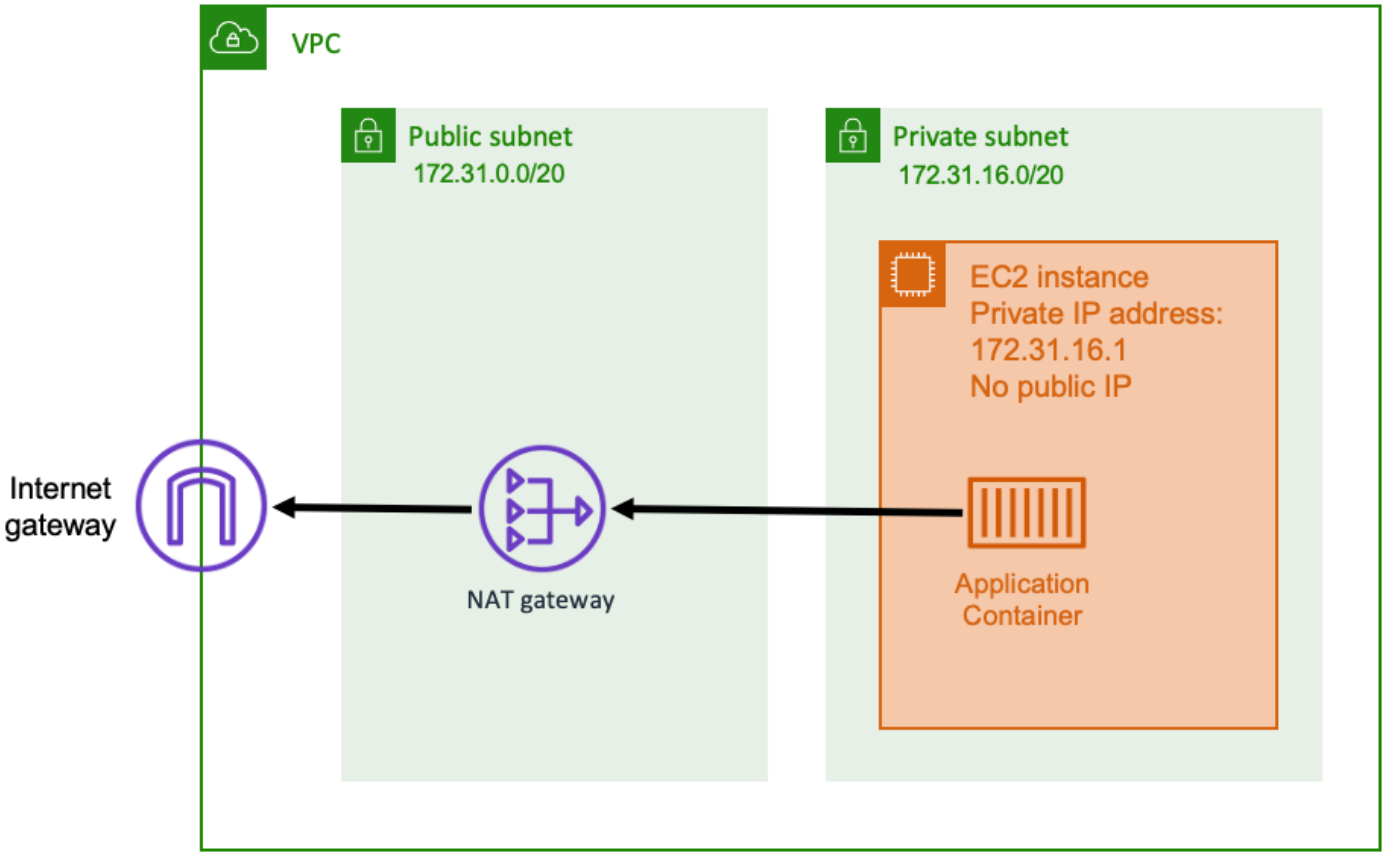
Misalnya, jika aplikasi Anda berjalan di Amazon EC2, pastikan port 22 untuk akses SSH tidak terbuka. Jika tidak, instans Anda dapat menerima upaya koneksi SSH konstan dari bot berbahaya di internet. Bot ini menjaring melalui alamat IP publik. Setelah mereka menemukan port SSH terbuka, mereka mencoba untuk memaksa kata sandi untuk mencoba mengakses instans Anda. Karena itu, banyak organisasi membatasi penggunaan subnet publik dan lebih memilih untuk memiliki sebagian besar, jika tidak semua, sumber daya mereka di dalam subnet pribadi.

Menggunakan subnet publik untuk jaringan cocok untuk aplikasi publik yang membutuhkan bandwidth dalam jumlah besar atau latensi minimal. Kasus penggunaan yang berlaku termasuk streaming video dan layanan game.

Pendekatan jaringan ini didukung baik saat Anda menggunakan Amazon ECS di Amazon EC2 dan saat Anda menggunakannya. AWS Fargate

- Amazon EC2 - Anda dapat meluncurkan instans EC2 di subnet publik. Amazon ECS menggunakan instans EC2 ini sebagai kapasitas cluster, dan kontainer apa pun yang berjalan pada instans dapat menggunakan alamat IP publik yang mendasari host untuk jaringan keluar. Ini berlaku untuk mode `host` dan `bridge` jaringan. Namun, mode `awsvpc` jaringan tidak menyediakan ENI tugas dengan alamat IP publik. Oleh karena itu, mereka tidak dapat menggunakan gateway internet secara langsung.
- Fargate - Saat Anda membuat layanan Amazon ECS, tentukan subnet publik untuk konfigurasi jaringan layanan Anda, dan gunakan opsi Tetapkan alamat IP publik. Setiap tugas Fargate berjejaring di subnet publik, dan memiliki alamat IP publik sendiri untuk komunikasi langsung dengan internet.

Subnet pribadi dan gateway NAT



Saat Anda menggunakan subnet pribadi dan gateway NAT, Anda dapat menjalankan aplikasi kontainer Anda di host yang ada di subnet pribadi. Dengan demikian, host ini memiliki alamat IP pribadi yang dapat dirutekan di dalam VPC Anda, tetapi tidak dapat dirutekan dari internet. Ini berarti bahwa host lain di dalam VPC dapat terhubung ke host menggunakan alamat IP pribadinya, tetapi host lain di internet tidak dapat melakukan komunikasi masuk ke host.

Dengan subnet pribadi, Anda dapat menggunakan gateway Network Address Translation (NAT) untuk memungkinkan host di dalam subnet pribadi terhubung ke internet. Host di internet menerima koneksi masuk yang tampaknya berasal dari alamat IP publik gateway NAT yang ada di dalam subnet publik. Gateway NAT bertanggung jawab untuk berfungsi sebagai jembatan antara internet dan VPC pribadi. Konfigurasi ini sering disukai untuk alasan keamanan karena itu berarti bahwa VPC Anda dilindungi dari akses langsung oleh penyerang di internet. Untuk informasi lebih lanjut, lihat Gateway NAT dalam Panduan Pengguna Amazon VPC.

Pendekatan jaringan pribadi ini cocok untuk skenario di mana Anda ingin melindungi wadah Anda dari akses eksternal langsung. Skenario yang berlaku termasuk sistem pemrosesan pembayaran

atau wadah yang menyimpan data pengguna dan kata sandi. Anda dikenakan biaya untuk membuat dan menggunakan gateway NAT di akun Anda. Penggunaan NAT gateway per jam dan tarif pemrosesan data juga berlaku. Untuk tujuan redundansi, Anda harus memiliki gateway NAT di setiap Availability Zone. Dengan cara ini, hilangnya ketersediaan satu Availability Zone tidak mengganggu konektivitas keluar Anda. Karena itu, jika Anda memiliki beban kerja yang kecil, mungkin lebih hemat biaya untuk menggunakan subnet pribadi dan gateway NAT.

Pendekatan jaringan ini didukung baik saat menggunakan Amazon ECS di Amazon EC2 dan saat menggunakannya. AWS Fargate

- Amazon EC2 - Anda dapat meluncurkan instans EC2 di subnet pribadi. Wadah yang berjalan pada host EC2 ini menggunakan jaringan host yang mendasarinya, dan permintaan keluar melalui gateway NAT.
- Fargate - Saat Anda membuat layanan Amazon ECS, tentukan subnet pribadi untuk konfigurasi jaringan layanan Anda, dan jangan gunakan opsi Tetapkan alamat IP publik. Setiap tugas Fargate di-host di subnet pribadi. Lalu lintas keluarnya diarahkan melalui gateway NAT apa pun yang telah Anda kaitkan dengan subnet pribadi itu.

Praktik terbaik untuk menerima koneksi masuk ke Amazon ECS dari internet

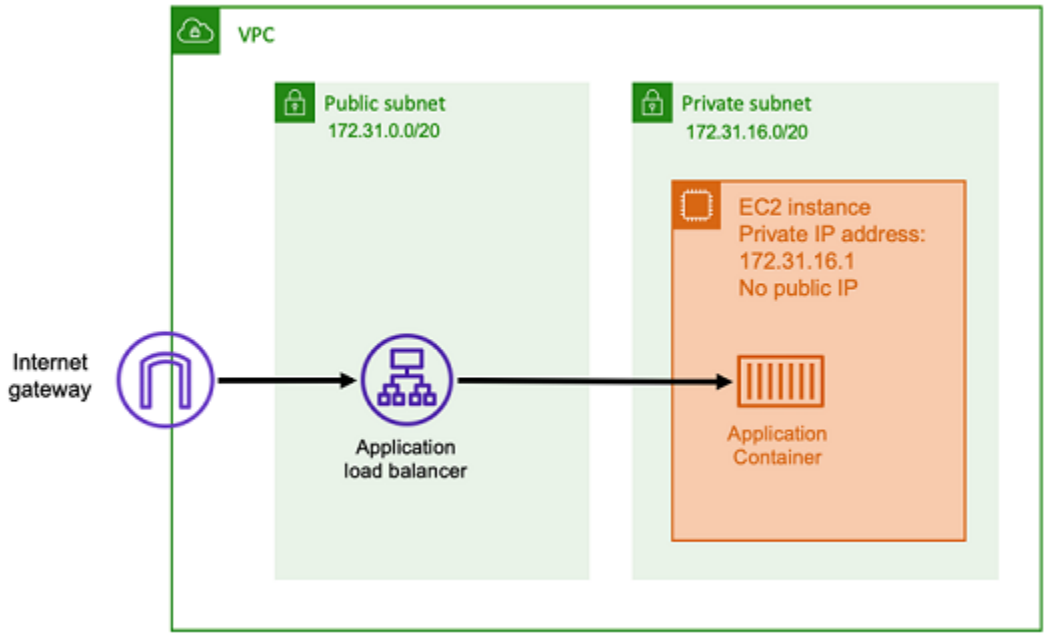
Jika Anda menjalankan layanan publik, Anda harus menerima lalu lintas masuk dari internet. Misalnya, situs web publik Anda harus menerima permintaan HTTP masuk dari browser. Dalam kasus seperti itu, host lain di internet juga harus memulai koneksi masuk ke host aplikasi Anda.

Salah satu pendekatan untuk masalah ini adalah meluncurkan container Anda pada host yang berada di subnet publik dengan alamat IP publik. Namun, kami tidak merekomendasikan ini untuk aplikasi skala besar. Untuk ini, pendekatan yang lebih baik adalah memiliki lapisan input yang dapat diskalakan yang berada di antara internet dan aplikasi Anda. Untuk pendekatan ini, Anda dapat menggunakan salah satu AWS layanan yang tercantum di bagian ini sebagai masukan.

Penyeimbang Beban Aplikasi

Application Load Balancer berfungsi pada layer aplikasi. Ini adalah lapisan ketujuh dari model Open Systems Interconnection (OSI). Hal ini membuat Application Load Balancer cocok untuk layanan HTTP publik. Jika Anda memiliki situs web atau HTTP REST API, maka Application Load Balancer

adalah penyeimbang beban yang cocok untuk beban kerja ini. Untuk informasi selengkapnya, lihat Apa itu Application Load Balancer? dalam Panduan Pengguna untuk Penyeimbang Beban Aplikasi.



Dengan arsitektur ini, Anda membuat Application Load Balancer di subnet publik sehingga memiliki alamat IP publik dan dapat menerima koneksi inbound dari internet. Ketika Application Load Balancer menerima koneksi masuk, atau lebih khusus permintaan HTTP, ia membuka koneksi ke aplikasi menggunakan alamat IP pribadinya. Kemudian, itu meneruskan permintaan melalui koneksi internal.

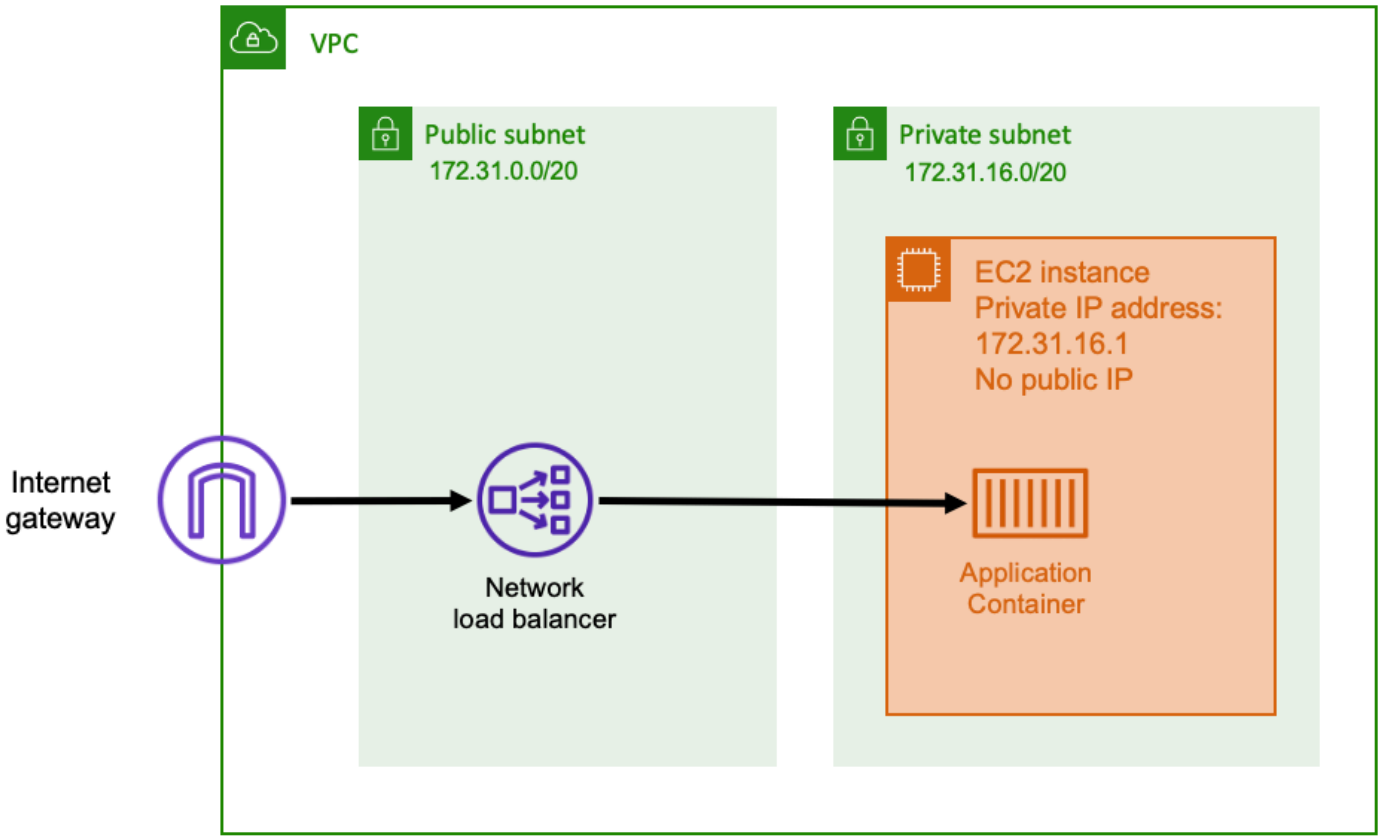
Application Load Balancer memiliki keunggulan sebagai berikut.

- Pengakhiran SSL/TLS — Application Load Balancer dapat mempertahankan komunikasi HTTPS yang aman dan sertifikat untuk komunikasi dengan klien. Ini secara opsional dapat menghentikan koneksi SSL di tingkat penyeimbang beban sehingga Anda tidak perlu menangani sertifikat dalam aplikasi Anda sendiri.
- Perutean lanjutan - Application Load Balancer dapat memiliki beberapa nama host DNS. Ini juga memiliki kemampuan routing lanjutan untuk mengirim permintaan HTTP masuk ke tujuan yang berbeda berdasarkan metrik seperti nama host atau jalur permintaan. Ini berarti Anda dapat menggunakan Application Load Balancer tunggal sebagai input untuk banyak layanan internal yang berbeda, atau bahkan layanan mikro pada jalur yang berbeda dari REST API.
- Dukungan gRPC dan soket web - Application Load Balancer dapat menangani lebih dari sekadar HTTP. Itu juga dapat memuat keseimbangan gRPC dan layanan berbasis websocket, dengan dukungan HTTP/2.

- Keamanan - Application Load Balancer membantu melindungi aplikasi Anda dari lalu lintas berbahaya. Ini mencakup fitur-fitur seperti mitigasi sinkronisasi HTTP, dan terintegrasi dengan AWS Web Application Firewall ().AWS WAF AWS WAF selanjutnya dapat menyaring lalu lintas berbahaya yang mungkin berisi pola serangan, seperti injeksi SQL atau skrip lintas situs.

Network Load Balancer

Penyeimbang Beban jaringan berfungsi pada lapisan keempat dari model Open Systems Interkoneksi (OSI). Ini cocok untuk protokol non-HTTP atau skenario di mana end-to-end enkripsi diperlukan, tetapi tidak memiliki fitur HTTP-spesifik yang sama dari Application Load Balancer. Oleh karena itu, Network Load Balancer paling cocok untuk aplikasi yang tidak menggunakan HTTP. Untuk informasi selengkapnya, lihat Apa itu Network Load Balancer? dalam Panduan Pengguna untuk Network Load Balancers.



Ketika Network Load Balancer digunakan sebagai input, fungsinya mirip dengan Application Load Balancer. Ini karena dibuat di subnet publik dan memiliki alamat IP publik yang dapat diakses di internet. Network Load Balancer kemudian membuka koneksi ke alamat IP pribadi host yang menjalankan container Anda, dan mengirimkan paket dari sisi publik ke sisi pribadi.

Fitur Network Load Balancer

Karena Network Load Balancer beroperasi pada tingkat yang lebih rendah dari tumpukan jaringan, ia tidak memiliki serangkaian fitur yang sama dengan Application Load Balancer. Namun, ia memang memiliki fitur-fitur penting berikut.

- nd-to-end Enkripsi E — Karena Network Load Balancer beroperasi pada lapisan keempat model OSI, ia tidak membaca isi paket. Ini membuatnya cocok untuk komunikasi load balancing yang membutuhkan end-to-end enkripsi.
- Enkripsi TLS — Selain end-to-end enkripsi, Network Load Balancer juga dapat menghentikan koneksi TLS. Dengan cara ini, aplikasi backend Anda tidak harus mengimplementasikan TLS mereka sendiri.
- Dukungan UDP — Karena Network Load Balancer beroperasi pada lapisan keempat model OSI, ini cocok untuk beban kerja dan protokol non HTTP selain TCP.

Menutup koneksi

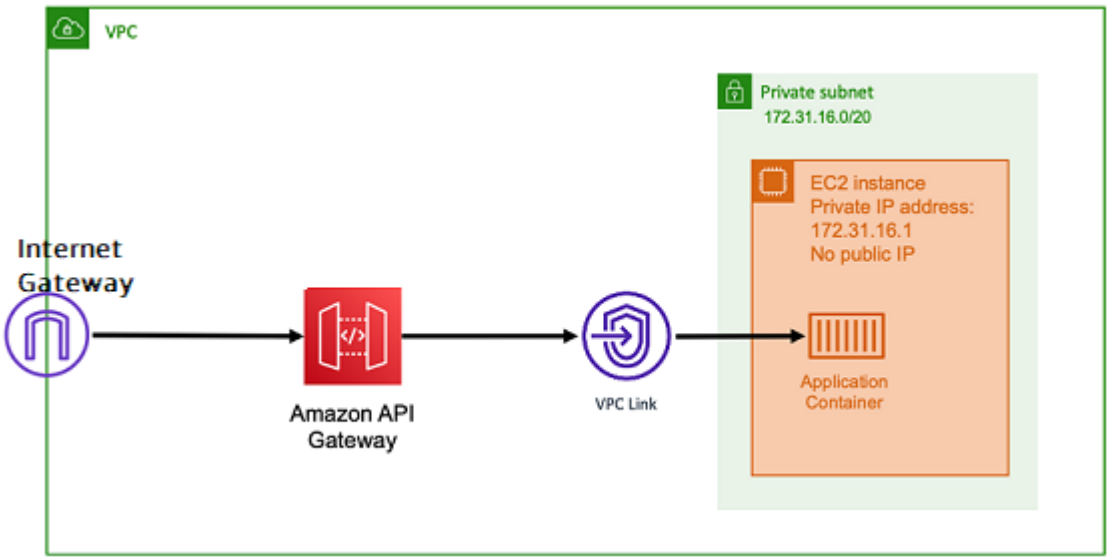
Karena Network Load Balancer tidak mengamati protokol aplikasi pada lapisan yang lebih tinggi dari model OSI, ia tidak dapat mengirim pesan penutupan ke klien dalam protokol tersebut. Berbeda dengan Application Load Balancer, koneksi tersebut harus ditutup oleh aplikasi atau Anda dapat mengkonfigurasi Network Load Balancer untuk menutup koneksi lapisan keempat ketika tugas dihentikan atau diganti. Lihat setelan penghentian sambungan untuk grup target Network Load Balancer dalam dokumentasi Network Load Balancer.

Membiarkan Network Load Balancer menutup koneksi pada lapisan keempat dapat menyebabkan klien menampilkan pesan kesalahan yang tidak diinginkan, jika klien tidak menanganinya. Lihat Builders Library untuk informasi selengkapnya tentang konfigurasi klien yang direkomendasikan di sini.

Metode untuk menutup koneksi akan bervariasi menurut aplikasi, namun salah satu caranya adalah memastikan bahwa penundaan deregistrasi target Network Load Balancer lebih lama daripada batas waktu koneksi klien. Klien akan timeout terlebih dahulu dan menyambung kembali dengan anggun melalui Network Load Balancer ke tugas berikutnya sementara tugas lama perlahan-lahan menguras semua kliennya. Untuk informasi selengkapnya tentang penundaan deregistrasi target Network Load Balancer, lihat dokumentasi Network Load Balancer.

API HTTP Gerbang Amazon API

Amazon API Gateway cocok untuk aplikasi HTTP dengan semburan mendadak dalam volume permintaan atau volume permintaan rendah. Untuk informasi selengkapnya, lihat Apa itu Amazon API Gateway? di Panduan Pengembang API Gateway.



Model harga untuk Application Load Balancer dan Network Load Balancer mencakup harga per jam untuk menjaga penyeimbang beban tersedia untuk menerima koneksi masuk setiap saat. Sebaliknya, API Gateway mengenakan biaya untuk setiap permintaan secara terpisah. Ini memiliki efek bahwa, jika tidak ada permintaan masuk, tidak ada biaya. Di bawah beban lalu lintas yang tinggi, Application Load Balancer atau Network Load Balancer dapat menangani volume permintaan yang lebih besar dengan harga per permintaan yang lebih murah daripada API Gateway. Namun, jika Anda memiliki jumlah permintaan yang rendah secara keseluruhan atau memiliki periode lalu lintas rendah, maka harga kumulatif untuk menggunakan API Gateway harus lebih hemat biaya daripada membayar biaya per jam untuk mempertahankan penyeimbang beban yang kurang dimanfaatkan. API Gateway juga dapat men-cache respons API, yang mungkin menghasilkan tingkat permintaan backend yang lebih rendah.

Fungsi API Gateway yang menggunakan tautan VPC yang memungkinkan layanan AWS terkelola terhubung ke host di dalam subnet pribadi VPC Anda, menggunakan alamat IP pribadinya. Ini dapat mendeteksi alamat IP pribadi ini dengan melihat catatan penemuan AWS Cloud Map layanan yang dikelola oleh Amazon ECS Service Discovery.

API Gateway mendukung fitur-fitur berikut.

- Operasi API Gateway mirip dengan penyeimbang beban, tetapi memiliki kemampuan tambahan yang unik untuk manajemen API
- API Gateway menyediakan kemampuan tambahan seputar otorisasi klien, tingkatan penggunaan, dan modifikasi permintaan/respons. Untuk informasi selengkapnya, lihat fitur Amazon API Gateway.
- API Gateway dapat mendukung titik akhir gateway API edge, regional, dan privat. Titik akhir tepi tersedia melalui CloudFront distribusi terkelola. Titik akhir regional dan swasta keduanya lokal untuk suatu Wilayah.
- Pengakhiran SSL/TLS
- Merutekan jalur HTTP yang berbeda ke layanan mikro backend yang berbeda

Selain fitur sebelumnya, API Gateway juga mendukung penggunaan otorisasi Lambda khusus yang dapat Anda gunakan untuk melindungi API Anda dari penggunaan yang tidak sah. Untuk informasi selengkapnya, lihat Catatan Bidang: API berbasis Kontainer Tanpa Server dengan Amazon ECS dan Amazon API Gateway.

Mengakses fitur Amazon ECS melalui pengaturan akun

Anda dapat masuk ke pengaturan akun Amazon ECS untuk memilih masuk atau keluar dari fitur tertentu. Untuk masing-masing Wilayah AWS, Anda dapat memilih untuk, atau memilih keluar dari, setiap pengaturan akun di tingkat akun atau untuk pengguna atau peran tertentu.

Anda mungkin ingin memilih masuk atau keluar dari fitur tertentu jika salah satu dari berikut ini relevan bagi Anda:

- Pengguna atau peran dapat memilih atau memilih keluar dari pengaturan akun tertentu untuk akun individu mereka.
- Pengguna atau peran dapat mengatur pengaturan opt-in atau opt-out default untuk semua pengguna di akun.
- Pengguna root atau pengguna dengan hak administrator dapat memilih, atau memilih keluar dari, peran atau pengguna tertentu apa pun di akun. Jika pengaturan akun untuk pengguna root diubah, itu menetapkan default untuk semua pengguna dan peran yang tidak dipilih oleh pengaturan akun individual.

> **Note**
>
> Pengguna federasi mengasumsikan pengaturan akun pengguna root dan tidak dapat mengatur pengaturan akun eksplisit untuk mereka secara terpisah.

Pengaturan akun berikut tersedia. Anda harus secara terpisah opt-in dan opt-out ke setiap pengaturan akun.

Amazon Resource Name (ARN) dan ID

Nama sumber daya:`serviceLongArnFormat`, `taskLongArnFormat`, dan `containerInstanceLongArnFormat`

Amazon ECS memperkenalkan format baru untuk Nama Sumber Daya Amazon (ARN) dan ID sumber daya untuk layanan, tugas, dan instans penampung Amazon ECS. Status keikutsertaan untuk setiap jenis sumber daya menentukan format Amazon Resource Name (ARN) yang digunakan sumber daya. Anda harus memilih format ARN baru untuk menggunakan fitur seperti penandaan sumber daya untuk jenis sumber daya tersebut. Untuk informasi selengkapnya, lihat Amazon Resource Name (ARN) dan ID.

Nilai default-nya `enabled`.

Sumber daya hanya akan diluncurkan setelah format ARN dan ID sumber daya baru menerima status disertakan. Semua sumber daya yang ada tidak terpengaruh. Agar layanan dan tugas Amazon ECS dapat beralih ke format ARN dan ID sumber daya baru, Anda harus membuat ulang layanan atau tugas. Untuk mentransisikan instance kontainer ke ARN baru dan format ID sumber daya, instance container harus dikeringkan dan instance container baru harus diluncurkan dan didaftarkan ke cluster.

> **Note**
>
> Tugas yang diluncurkan oleh layanan Amazon ECS hanya dapat menerima ARN baru dan format ID sumber daya jika layanan dibuat pada atau setelah 16 November 2018, dan pengguna yang membuat layanan telah memilih format baru untuk tugas.

AWSVPC trunking

Nama sumber daya: `awsvpcTrunking`

Amazon ECS mendukung peluncuran instans kontainer dengan peningkatan kepadatan elastic network interface (ENI) menggunakan jenis instans Amazon EC2 yang didukung. Ketika Anda menggunakan tipe instans ini dan memilih pada pengaturan akun `awsvpcTrunking`, ENI tambahan tersedia pada instans kontainer yang baru diluncurkan. Anda dapat menggunakan konfigurasi ini untuk menempatkan lebih banyak tugas menggunakan mode `awsvpc` jaringan pada setiap instance kontainer. Dengan menggunakan fitur ini, `c5.large` instance dengan `awsvpcTrunking` enabled memiliki peningkatan kuota ENI sepuluh. Instance container memiliki antarmuka jaringan utama, dan Amazon ECS membuat dan melampirkan antarmuka jaringan "trunk" ke instance container. Antarmuka jaringan utama dan antarmuka jaringan trunk tidak dihitung terhadap kuota ENI. Oleh karena itu, Anda dapat menggunakan konfigurasi ini untuk meluncurkan sepuluh tugas pada instance container alih-alih dua tugas saat ini. Untuk informasi selengkapnya, lihat Pembuatan torso antarmuka jaringan elastis.

Nilai default-nya `disabled`.

sumber daya hanya yang diluncurkan setelah status menyertakan menerima peningkatan batas ENI. Semua sumber daya yang ada tidak terpengaruh. Untuk mentransisikan instance kontainer ke kuota ENI yang ditingkatkan, instance kontainer harus dikeringkan dan instance kontainer baru didaftarkan ke cluster.

CloudWatch Wawasan Kontainer

Nama sumber daya: `containerInsights`

CloudWatch Container Insights mengumpulkan, mengumpulkan, dan merangkum metrik dan log dari aplikasi dan layanan mikro dalam kontainer Anda. Metrik tersebut mencakup pemanfaatan sumber daya seperti CPU, memori, disk, dan jaringan. Wawasan Kontainer juga menyediakan informasi diagnostik, seperti kegagalan mengulang kembali kontainer, untuk membantu Anda mengisolasi masalah dan mengatasinya dengan cepat. Anda juga dapat mengatur alarm CloudWatch pada metrik yang dikumpulkan oleh Wawasan Kontainer. Untuk informasi selengkapnya, lihat Pantau kontainer Amazon ECS menggunakan Wawasan Kontainer.

Ketika Anda memilih pengaturan akun `containerInsights`, semua klaster baru memiliki Wawasan Kontainer yang diaktifkan secara default. Anda dapat menonaktifkan pengaturan ini untuk klaster tertentu ketika Anda membuatnya. Anda juga dapat mengubah pengaturan ini dengan menggunakan UpdateClusterSettings API.

Untuk cluster yang berisi tugas atau layanan yang menggunakan tipe peluncuran EC2, instance container Anda harus menjalankan agen Amazon ECS versi 1.29.0 atau yang lebih baru untuk

menggunakan Container Insights. Untuk informasi selengkapnya, lihat Manajemen instance kontainer Linux.

Nilai default-nya `disabled`.

IPv6 VPC tumpukan dobel

Nama sumber daya: `dualStackIPv6`

Amazon ECS mendukung penyediaan tugas dengan alamat IPv6 selain alamat IPv4 pribadi utama.

Agar tugas menerima alamat IPv6, tugas harus menggunakan mode jaringan `awsvpc`, harus diluncurkan dalam VPC yang dikonfigurasi untuk mode tumpukan dobel, dan pengaturan akun `dualStackIPv6`harus diaktifkan. Untuk informasi selengkapnya tentang persyaratan lain, lihat Menggunakan VPC dalam mode tumpukan dobel untuk jenis peluncuran EC2 dan Menggunakan VPC dalam mode tumpukan dobel untuk jenis peluncuran Fargate.

> **Important**
>
> Pengaturan `dualStackIPv6` akun hanya dapat diubah menggunakan Amazon ECS API atau. AWS CLI Untuk informasi selengkapnya, lihat Mengubah pengaturan akun.

Jika Anda memiliki tugas yang berjalan menggunakan mode `awsvpc` jaringan di subnet yang diaktifkan IPv6 antara tanggal 1 Oktober 2020 dan 2 November 2020, pengaturan `dualStackIPv6` akun default di Wilayah tempat tugas dijalankan adalah. `disabled` Jika kondisi tersebut tidak terpenuhi, `dualStackIPv6` pengaturan default di Wilayah adalah`enabled`.

Nilai default-nya `disabled`.

Kepatuhan Fargate FIPS-140

Nama sumber daya: `fargateFIPSMode`

Fargate mendukung Federal Information Processing Standard (FIPS-140) yang menetapkan persyaratan keamanan untuk modul kriptografi yang melindungi informasi sensitif. Ini adalah standar pemerintah Amerika Serikat dan Kanada saat ini, dan berlaku untuk sistem yang harus mematuhi Undang-Undang Manajemen Keamanan Informasi Federal (FISMA) atau Program Manajemen Risiko dan Otorisasi Federal (FedRAMP).

Nilai default-nya `disabled`.

Anda harus mengaktifkan kepatuhan FIPS-140. Untuk informasi selengkapnya, lihat the section called “AWS Fargate Kepatuhan FIPS-140”.

> **Important**
>
> Pengaturan `fargateFIPSMode` akun hanya dapat diubah menggunakan Amazon ECS API atau. AWS CLI Untuk informasi selengkapnya, lihat Mengubah pengaturan akun.

Otorisasi Sumber Daya Tag

Nama sumber daya: `tagResourceAuthorization`

Beberapa tindakan Amazon ECS API memungkinkan Anda menentukan tag saat membuat sumber daya.

Amazon ECS memperkenalkan otorisasi penandaan untuk pembuatan sumber daya. Pengguna harus memiliki izin untuk tindakan yang membuat sumber daya, sepertiecsCreateCluster. Jika tag ditentukan dalam tindakan pembuatan sumber daya, AWS lakukan otorisasi tambahan pada `ecs:TagResource` tindakan untuk memverifikasi apakah pengguna atau peran memiliki izin untuk membuat tag. Oleh karena itu, Anda harus memberikan izin eksplisit untuk menggunakan tindakan. `ecs:TagResource` Untuk informasi selengkapnya, lihat the section called “Memberi tanda pada sumber daya saat sumber daya dibuat”.

Masa tunggu pensiun tugas Fargate

Nama sumber daya: `fargateTaskRetirementWaitPeriod`

AWS bertanggung jawab untuk menambal dan memelihara infrastruktur yang mendasari AWS Fargate. Saat AWS menentukan bahwa pembaruan keamanan atau infrastruktur diperlukan untuk tugas Amazon ECS yang dihosting di Fargate, tugas harus dihentikan dan tugas baru diluncurkan untuk menggantikannya. Anda dapat mengonfigurasi periode tunggu sebelum tugas dihentikan untuk ditambal. Anda memiliki opsi untuk segera menghentikan tugas, menunggu 7 hari kalender, atau menunggu 14 hari kalender.

Pengaturan ini ada di tingkat akun.

Aktivasi Runtime Monitoring

Nama sumber daya: `guardDutyActivate`

`guardDutyActivate`Parameter ini hanya-baca di Amazon ECS dan menunjukkan apakah Runtime Monitoring diaktifkan atau dimatikan oleh administrator keamanan Anda di akun Amazon ECS Anda. GuardDuty mengontrol pengaturan akun ini atas nama Anda. Untuk informasi selengkapnya, lihat Melindungi beban kerja Amazon ECS dengan Runtime Monitoring.

Topik

Amazon Resource Name (ARN) dan ID

Saat sumber daya Amazon ECS dibuat, setiap sumber daya diberi Nama Sumber Daya Amazon (ARN) dan pengenal sumber daya (ID) yang unik. Jika Anda menggunakan alat baris perintah atau Amazon ECS API untuk bekerja dengan Amazon ECS, ARN atau ID sumber daya diperlukan untuk perintah tertentu. Misalnya, jika Anda menggunakan AWS CLI perintah stop-task untuk menghentikan tugas, Anda harus menentukan tugas ARN atau ID dalam perintah.

Anda dapat memilih dan memilih keluar dari format Nama Sumber Daya Amazon (ARN) dan ID sumber daya baru berdasarkan per wilayah. Saat ini, setiap akun baru yang dibuat akan disertakan secara default.

Anda dapat menyertakan atau menolak format Amazon Resource Name (ARN) dan ID sumber daya baru kapan saja. Setelah Anda ikut serta, sumber daya baru apa pun yang Anda buat menggunakan format baru.

> **Note**
>
> ID sumber daya tidak berubah setelah dibuat. Oleh karena itu, memilih masuk atau keluar dari format baru tidak memengaruhi ID sumber daya yang ada.

Bagian berikut menjelaskan cara mengubah format ARN dan ID sumber daya. Untuk informasi selengkapnya tentang transisi ke format baru, lihat FAQ Amazon Elastic Container Service.

Format Amazon Resource Name (ARN)

Beberapa sumber daya memiliki nama yang mudah digunakan, seperti layanan bernama `production`. Dalam hal lain, Anda harus menentukan sumber daya menggunakan format Amazon Resource Name (ARN). Format ARN baru untuk tugas, layanan, dan instans penampung Amazon ECS menyertakan nama cluster. Untuk informasi lebih lanjut tentang menyertakan format baru ARN, lihat Mengubah pengaturan akun.

Tabel berikut menunjukkan format saat ini dan format baru untuk setiap jenis sumber daya.

Jenis sumber daya	ARN
Instans kontainer	Saat ini: `arn:aws:ecs:` *`region`*`:`*`aws_account_id`* `:container-instance/` *`container-instance-id`* Baru: `arn:aws:ecs:` *`region`*`:`*`aws_account_id`* `:container-instance/` *`cluster-name`* `/`*`container-instance-id`*
Layanan Amazon ECS	Saat ini: `arn:aws:ecs:` *`region`*`:`*`aws_account_id`* `:service/` *`service-name`* Baru: `arn:aws:ecs:` *`region`*`:`*`aws_account_id`* `:service/` *`cluster-name`* `/`*`service-name`*
Tugas Amazon ECS	Saat ini: `arn:aws:ecs:` *`region`*`:`*`aws_account_id`* `:task/`*`task-id`* Baru: `arn:aws:ecs:` *`region`*`:`*`aws_account_id`* `:task/`*`cluster-name`* `/`*`task-id`*

Panjang ID Sumber Daya

ID sumber daya berisi kombinasi unik dari huruf dan angka. Format ID sumber daya baru menyertakan ID yang lebih pendek untuk tugas Amazon ECS dan instans penampung. Format ID sumber daya saat ini adalah 36 karakter. ID baru dalam format 32 karakter yang tidak menyertakan tanda hubung apa pun. Untuk informasi lebih lanjut tentang memilih format ID sumber daya baru, lihat Mengubah pengaturan akun.

Lini masa format ARN dan ID sumber daya

Garis waktu untuk periode opt-in dan opt-out untuk Amazon Resource Name (ARN) baru dan format ID sumber daya untuk sumber daya Amazon ECS berakhir pada 1 April 2021. Secara default, semua akun dipilih ke format baru. Semua sumber daya baru yang dibuat menerima format baru, dan Anda tidak dapat lagi memilih keluar.

AWS Fargate Kepatuhan Standar Pemrosesan Informasi Federal (FIPS-140)

Anda harus mengaktifkan kepatuhan Standar Pemrosesan Informasi Federal (FIPS-140) di Fargate. Untuk informasi selengkapnya, lihat the section called “AWS Fargate Kepatuhan FIPS-140”.

Jalankan `put-account-setting-default` dengan `fargateFIPSMode` opsi yang disetel ke`enabled`. Untuk informasi selengkapnya, lihat, put-account-setting-defaultdi Referensi API Amazon Elastic Container Service.

- Anda dapat menggunakan perintah berikut untuk mengaktifkan kepatuhan FIPS-140.

```
aws ecs put-account-setting-default --name fargateFIPSMode --value enabled
```

Contoh Output

```
{
    "setting": {
        "name": "fargateFIPSMode",
        "value": "enabled",
        "principalArn": "arn:aws:iam::123456789012:root",
         "type": user
    }
}
```

Anda dapat menjalankan `list-account-settings` untuk melihat status kepatuhan FIPS-140 saat ini. Gunakan `effective-settings` opsi untuk melihat pengaturan tingkat akun.

```
aws ecs list-account-settings --effective-settings
```

Otorisasi penandaan

Amazon ECS memperkenalkan otorisasi penandaan untuk pembuatan sumber daya. Pengguna harus memiliki izin penandaan untuk tindakan yang membuat sumber daya, seperti. `ecsCreateCluster` Saat Anda membuat sumber daya dan menentukan tag untuk sumber daya tersebut, AWS lakukan otorisasi tambahan untuk memverifikasi bahwa ada izin untuk membuat tag. Oleh karena itu, Anda harus memberikan izin eksplisit untuk menggunakan tindakan. `ecs:TagResource` Untuk informasi selengkapnya, lihat the section called “Memberi tanda pada sumber daya saat sumber daya dibuat”.

Untuk ikut serta dalam menandai otorisasi, jalankan `put-account-setting-default` dengan `tagResourceAuthorization` opsi yang disetel ke. `enable` Untuk informasi selengkapnya, lihat, put-account-setting-defaultdi Referensi API Amazon Elastic Container Service. Anda dapat menjalankan `list-account-settings` untuk melihat status otorisasi penandaan saat ini.

- Anda dapat menggunakan perintah berikut untuk mengaktifkan otorisasi penandaan.

```
aws ecs put-account-setting-default --name tagResourceAuthorization --value on --
region region
```

Contoh Output

```
{
    "setting": {
        "name": "tagResourceAuthorization",
        "value": "on",
        "principalArn": "arn:aws:iam::123456789012:root",
        "type": user
    }
}
```

Setelah mengaktifkan otorisasi penandaan, Anda harus mengonfigurasi izin yang sesuai untuk memungkinkan pengguna menandai sumber daya saat pembuatan. Untuk informasi selengkapnya, lihat the section called “Memberi tanda pada sumber daya saat sumber daya dibuat”.

Anda dapat menjalankan `list-account-settings` untuk melihat status otorisasi penandaan saat ini. Gunakan `effective-settings` opsi untuk melihat pengaturan tingkat akun.

```
aws ecs list-account-settings --effective-settings
```

Menandai garis waktu otorisasi

Anda dapat mengonfirmasi apakah otorisasi penandaan aktif dengan menjalankan `list-account-settings` untuk melihat nilainya. `tagResourceAuthorization` Ketika nilainyaon, itu berarti otorisasi penandaan sedang digunakan. Untuk informasi selengkapnya, lihat, list-account-settingsdi Referensi API Amazon Elastic Container Service.

Berikut ini adalah tanggal-tanggal penting yang terkait dengan otorisasi penandaan.

- 18 April 2023 - Otorisasi penandaan diperkenalkan. Semua akun baru dan yang sudah ada harus ikut serta untuk menggunakan fitur ini. Anda dapat memilih untuk mulai menggunakan otorisasi penandaan. Dengan ikut serta, Anda harus memberikan izin yang sesuai.
- 9 Februari 2024 - 6 Maret 2024 - Semua akun baru dan akun yang ada yang tidak terkena dampak memiliki otorisasi penandaan sebagai default. Anda dapat mengaktifkan atau menonaktifkan pengaturan `tagResourceAuthorization` akun untuk memverifikasi kebijakan IAM Anda.

  AWS telah memberi tahu akun yang terkena dampak.

  Untuk menonaktifkan fitur, jalankan `put-account-setting-default` dengan `tagResourceAuthorization` opsi yang disetel ke`off`.
- 7 Maret 2024 - Jika Anda telah mengaktifkan otorisasi penandaan, Anda tidak dapat lagi menonaktifkan pengaturan akun.

  Kami menyarankan Anda menyelesaikan pengujian kebijakan IAM Anda sebelum tanggal ini.
- 29 Maret 2024 - Semua akun menggunakan otorisasi penandaan. Pengaturan tingkat akun tidak akan lagi tersedia di konsol Amazon ECS atau AWS CLI

AWS Fargate tugas pensiun waktu tunggu

AWS mengirimkan pemberitahuan ketika Anda memiliki tugas Fargate yang berjalan pada revisi versi platform yang ditandai untuk pensiun. Untuk informasi selengkapnya, lihat AWS FAQ pemeliharaan tugas Fargate.

Anda dapat mengkonfigurasi waktu Fargate memulai tugas pensiun. Untuk beban kerja yang memerlukan aplikasi pembaruan segera, pilih pengaturan langsung (`0`). Ketika Anda membutuhkan kontrol lebih, misalnya, ketika tugas hanya dapat dihentikan selama jendela tertentu, konfigurasikan opsi 7 hari (7), atau 14 hari (14).

Kami menyarankan Anda memilih masa tunggu yang lebih pendek untuk mengambil revisi versi platform yang lebih baru lebih cepat.

Konfigurasikan periode tunggu dengan menjalankan `put-account-setting-default` atau `put-account-setting` sebagai pengguna root atau pengguna administratif. Gunakan `fargateTaskRetirementWaitPeriod` opsi untuk `name` dan `value` opsi yang disetel ke salah satu nilai berikut:

- `0`- AWS mengirim pemberitahuan, dan segera mulai pensiun tugas yang terpengaruh.
- 7- AWS mengirim pemberitahuan, dan menunggu 7 hari kalender sebelum mulai pensiun tugas yang terpengaruh.
- 14- AWS mengirim pemberitahuan, dan menunggu 14 hari kalender sebelum mulai pensiun tugas yang terpengaruh.

Defaultnya adalah 7 hari.

Untuk informasi selengkapnya, lihat, put-account-setting-defaultdan put-account-settingdi Referensi API Amazon Elastic Container Service.

Anda dapat menjalankan perintah berikut untuk mengatur periode tunggu menjadi 14 hari.

```
aws ecs put-account-setting-default --name fargateTaskRetirementWaitPeriod --value 14
```

Contoh Output

```
{
```

```
    "setting": {
        "name": "fargateTaskRetirementWaitPeriod",
        "value": "14",
        "principalArn": "arn:aws:iam::123456789012:root",
        "type: user"
    }
}
```

Anda dapat menjalankan `list-account-settings` untuk melihat waktu tunggu pensiun tugas Fargate saat ini. Gunakan `effective-settings` opsi.

```
aws ecs list-account-settings --effective-settings
```

Pemantauan Runtime (GuardDuty integrasi Amazon)

Runtime Monitoring adalah layanan deteksi ancaman cerdas yang melindungi beban kerja yang berjalan pada instance container Fargate dan EC2 dengan terus memantau aktivitas AWS log dan jaringan untuk mengidentifikasi perilaku berbahaya atau tidak sah.

`guardDutyActivate`Parameter ini hanya-baca di Amazon ECS dan menunjukkan apakah Runtime Monitoring diaktifkan atau dimatikan oleh administrator keamanan Anda di akun Amazon ECS Anda. GuardDuty mengontrol pengaturan akun ini atas nama Anda. Untuk informasi selengkapnya, lihat Melindungi beban kerja Amazon ECS dengan Runtime Monitoring.

Anda dapat menjalankan `list-account-settings` untuk melihat pengaturan GuardDuty integrasi saat ini.

```
aws ecs list-account-settings
```

Contoh Output

```
{
    "setting": {
        "name": "guardDutyActivate",
        "value": "on",
        "principalArn": "arn:aws:iam::123456789012:doej",
        "type": aws-managed"
    }
}
```

Melihat pengaturan akun menggunakan konsol

Anda dapat menggunakan AWS Management Console untuk melihat pengaturan akun Anda.

> **Important**
>
> Pengaturan`dualStackIPv6`, `fargateFIPSMode` dan `fargateTaskRetirementWaitPeriod` akun hanya dapat dilihat atau diubah menggunakan AWS CLI.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di bilah navigasi di bagian atas layar, pilih Wilayah yang akan dilihat pengaturan akunnya.
3. Di halaman navigasi, pilih Pengaturan Akun.

Mengubah pengaturan akun

Anda dapat menggunakan AWS Management Console untuk mengubah pengaturan akun Anda.

`guardDutyActivate`Parameter ini hanya-baca di Amazon ECS dan menunjukkan apakah Runtime Monitoring diaktifkan atau dimatikan oleh administrator keamanan Anda di akun Amazon ECS Anda. GuardDuty mengontrol pengaturan akun ini atas nama Anda. Untuk informasi selengkapnya, lihat Melindungi beban kerja Amazon ECS dengan Runtime Monitoring.

> **Important**
>
> Pengaturan`dualStackIPv6`, `fargateFIPSMode` dan `fargateTaskRetirementWaitPeriod` akun hanya dapat dilihat atau diubah menggunakan AWS CLI.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di bilah navigasi di bagian atas layar, pilih Wilayah yang akan dilihat pengaturan akunnya.
3. Di halaman navigasi, pilih Pengaturan Akun.
4. Pilih Perbarui.
5. Untuk menambah atau mengurangi jumlah tugas yang dapat Anda jalankan dalam mode jaringan awsvpc untuk setiap instans EC2, di bawah AWSVPCTrunking, pilih Trunking. AWSVPC

6. Untuk menggunakan atau berhenti menggunakan CloudWatch Wawasan Kontainer secara default untuk cluster, di bawah CloudWatch Wawasan Kontainer, pilih atau hapus CloudWatch Wawasan Kontainer.
7. Untuk mengaktifkan atau menonaktifkan otorisasi penandaan, di bawah Otorisasi Penandaan Sumber Daya, pilih atau hapus Otorisasi Penandaan Sumber Daya.
8. Pilih Simpan perubahan.
9. Pada layar konfirmasi, pilih Konfirmasi untuk menyimpan pilihan.

Mengembalikan ke pengaturan akun Amazon ECS default

Anda dapat menggunakan AWS Management Console untuk mengembalikan pengaturan akun Amazon ECS Anda ke default.

Opsi Kembalikan ke akun default hanya tersedia jika pengaturan akun Anda tidak lagi menjadi pengaturan default.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di bilah navigasi di bagian atas layar, pilih Wilayah yang akan dilihat pengaturan akunnya.
3. Di halaman navigasi, pilih Pengaturan Akun.
4. Pilih Perbarui.
5. Pilih Kembalikan ke akun default.
6. Pada layar konfirmasi, pilih Konfirmasi untuk menyimpan pilihan.

Manajemen pengaturan akun menggunakan AWS CLI

Anda dapat mengelola setelan akun menggunakan Amazon ECS API, AWS CLI atau SDK. Pengaturan`dualStackIPv6`, `fargateFIPSMode` dan `fargateTaskRetirementWaitPeriod` akun hanya dapat dilihat atau diubah menggunakan alat tersebut.

Untuk informasi tentang tindakan API yang tersedia untuk definisi tugas, lihat Tindakan setelan akun di Referensi API Amazon Elastic Container Service.

Gunakan salah satu perintah berikut untuk memodifikasi pengaturan akun default untuk semua pengguna atau peran di akun Anda. Perubahan ini berlaku untuk seluruh AWS akun kecuali pengguna atau peran secara eksplisit mengesampingkan pengaturan ini untuk diri mereka sendiri.

- put-account-setting-default (AWS CLI)

```
aws ecs put-account-setting-default --name serviceLongArnFormat --value enabled --
region us-east-2
```

  Anda juga dapat menggunakan perintah ini untuk mengubah pengaturan akun lainnya. Untuk melakukan hal ini, ganti parameter name dengan pengaturan akun yang sesuai.

- Tulis-ECS (AccountSetting)AWS Tools for Windows PowerShell

```
Write-ECSAccountSettingDefault -Name serviceLongArnFormat -Value enabled -Region us-
east-1 -Force
```

Untuk mengubah pengaturan akun untuk akun pengguna Anda (AWS CLI)

Gunakan salah satu perintah berikut untuk mengubah pengaturan akun untuk pengguna Anda. Jika Anda menggunakan perintah ini sebagai pengguna root, perubahan berlaku untuk seluruh AWS akun kecuali; pengguna atau peran secara eksplisit mengesampingkan pengaturan ini untuk diri mereka sendiri.

- put-account-setting (AWS CLI)

```
aws ecs put-account-setting --name serviceLongArnFormat --value enabled --region us-
east-1
```

  Anda juga dapat menggunakan perintah ini untuk mengubah pengaturan akun lainnya. Untuk melakukan hal ini, ganti parameter name dengan pengaturan akun yang sesuai.

- Tulis-ECS (AccountSetting)AWS Tools for Windows PowerShell

```
Write-ECSAccountSetting -Name serviceLongArnFormat -Value enabled -Force
```

Untuk mengubah pengaturan akun untuk pengguna atau peran tertentu (AWS CLI)

Gunakan salah satu perintah berikut dan tentukan ARN pengguna, peran, atau pengguna root dalam permintaan untuk mengubah pengaturan akun untuk pengguna atau peran tertentu.

- put-account-setting (AWS CLI)

```
aws ecs put-account-setting --name serviceLongArnFormat --value enabled --principal-
arn arn:aws:iam::aws_account_id:user/principalName --region us-east-1
```

Anda juga dapat menggunakan perintah ini untuk mengubah pengaturan akun lainnya. Untuk melakukan hal ini, ganti parameter name dengan pengaturan akun yang sesuai.

- Tulis-ECS (AccountSetting)AWS Tools for Windows PowerShell

```
Write-ECSAccountSetting -Name serviceLongArnFormat -Value enabled -PrincipalArn
 arn:aws:iam::aws_account_id:user/principalName -Region us-east-1 -Force
```

Definisi tugas Amazon ECS

Definisi tugas adalah cetak biru untuk aplikasi Anda. Ini adalah file teks dalam format JSON yang menjelaskan parameter dan satu atau lebih wadah yang membentuk aplikasi Anda.

Berikut ini adalah beberapa parameter yang dapat Anda tentukan dalam definisi tugas:

- Jenis peluncuran yang akan digunakan, yang menentukan infrastruktur tempat tugas Anda di-host
- Gambar Docker untuk digunakan dengan setiap kontainer dalam tugas Anda
- Berapa banyak CPU dan memori untuk digunakan dengan setiap tugas atau setiap kontainer dalam tugas
- Memori dan persyaratan CPU
- Sistem operasi wadah tempat tugas berjalan
- Mode jaringan Docker digunakan untuk kontainer dalam tugas.
- Konfigurasi pencatatan digunakan untuk tugas Anda
- Apakah tugas terus berjalan jika penampung selesai atau gagal
- Perintah yang dijalankan kontainer saat dimulai
- Setiap volume data yang digunakan dengan wadah dalam tugas
- Peran IAM yang digunakan tugas Anda

Untuk daftar lengkap parameter definisi tugas, lihatParameter ketentuan tugas.

Setelah Anda membuat definisi tugas, Anda dapat menjalankan definisi tugas sebagai tugas atau layanan.

- tugas adalah penunjukan hal abstrak dengan instans yang konkret dari ketentuan tugas dalam sebuah klaster. Setelah Anda membuat definisi tugas untuk aplikasi Anda dalam Amazon ECS, Anda dapat menentukan jumlah tugas yang akan dijalankan di klaster Anda.
- Layanan Amazon ECS menjalankan dan mempertahankan jumlah tugas yang Anda inginkan secara bersamaan di kluster Amazon ECS. Cara kerjanya adalah, jika salah satu tugas Anda gagal atau berhenti karena alasan apa pun, penjadwal layanan Amazon ECS meluncurkan instance lain berdasarkan definisi tugas Anda. Ia melakukan ini untuk menggantikannya dan dengan demikian mempertahankan jumlah tugas yang Anda inginkan dalam layanan.

Topik

Status definisi tugas

Definisi tugas mengubah status saat Anda membuat, membatalkan pendaftaran, atau menghapusnya. Anda dapat melihat status definisi tugas di konsol, atau dengan menggunakan`DescribeTaskDefinition`.

Berikut ini adalah status yang mungkin untuk definisi tugas:

AKTIF

Definisi tugas adalah `ACTIVE` setelah terdaftar di Amazon ECS. Anda dapat menggunakan definisi tugas di `ACTIVE` negara bagian untuk menjalankan tugas, atau membuat layanan.

TIDAK AKTIF

Definisi tugas bertransisi dari `ACTIVE` status ke `INACTIVE` status saat Anda membatalkan pendaftaran definisi tugas. Anda dapat mengambil definisi `INACTIVE` tugas dengan meneleponDescribeTaskDefinition. Anda tidak dapat menjalankan tugas baru atau membuat layanan baru dengan definisi tugas di `INACTIVE` negara bagian. Tidak ada dampak pada layanan atau tugas yang ada.

DELETE_IN_PROGRESS

Definisi tugas transisi dari `INACTIVE` status ke `DELETE_IN_PROGRESS` status setelah Anda mengirimkan definisi tugas untuk dihapus. Setelah definisi tugas dalam `DELETE_IN_PROGRESS` status, Amazon ECS secara berkala memverifikasi bahwa definisi tugas target tidak direferensikan oleh tugas atau penerapan aktif apa pun, lalu menghapus definisi tugas secara permanen. Anda tidak dapat menjalankan tugas baru atau membuat layanan baru dengan definisi

tugas di DELETE_IN_PROGRESS negara bagian. Definisi tugas dapat dikirimkan untuk dihapus kapan saja tanpa memengaruhi tugas dan layanan yang ada.

Definisi tugas yang berada dalam DELETE_IN_PROGRESS status dapat dilihat di konsol dan Anda dapat mengambil definisi tugas dengan memanggilDescribeTaskDefinition.

Saat Anda menghapus semua revisi definisi INACTIVE tugas, nama definisi tugas tidak ditampilkan di konsol dan tidak ditampilkan di API. Jika revisi definisi tugas dalam DELETE_IN_PROGRESS status, nama definisi tugas ditampilkan di konsol dan dikembalikan di API. Nama definisi tugas dipertahankan oleh Amazon ECS dan revisi akan bertambah saat berikutnya Anda membuat definisi tugas dengan nama itu.

Jika Anda menggunakan AWS Config untuk mengelola definisi tugas Anda, AWS Config menagih Anda untuk semua pendaftaran definisi tugas. Anda hanya dikenakan biaya untuk membatalkan pendaftaran definisi tugas terbaruACTIVE. Tidak ada biaya untuk menghapus definisi tugas. Untuk informasi lebih lanjut tentang harga, lihat AWS Config Harga.

Sumber daya Amazon ECS yang dapat memblokir penghapusan

Permintaan penghapusan definisi tugas tidak akan selesai ketika ada sumber daya Amazon ECS yang bergantung pada revisi definisi tugas. Sumber daya berikut mungkin mencegah definisi tugas dihapus:

- Tugas Amazon ECS - Definisi tugas diperlukan agar tugas tetap sehat.
- Penerapan dan set tugas Amazon ECS - Definisi tugas diperlukan saat peristiwa penskalaan dimulai untuk penyebaran atau set tugas Amazon ECS.

Jika definisi tugas Anda tetap dalam DELETE_IN_PROGRESS status, Anda dapat menggunakan konsol, atau AWS CLI untuk mengidentifikasi, dan kemudian menghentikan sumber daya yang memblokir penghapusan definisi tugas.

Penghapusan definisi tugas setelah sumber daya yang diblokir dihapus

Aturan berikut berlaku setelah Anda menghapus sumber daya yang memblokir penghapusan definisi tugas:

- Tugas Amazon ECS - Penghapusan definisi tugas dapat memakan waktu hingga 1 jam untuk diselesaikan setelah tugas dihentikan.

- Penyebaran dan set tugas Amazon ECS - Penghapusan definisi tugas dapat memakan waktu hingga 24 jam untuk diselesaikan setelah penerapan atau set tugas dihapus.

Merancang aplikasi Anda

Anda merancang aplikasi Anda dengan membuat definisi tugas untuk aplikasi Anda. Definisi tugas berisi parameter yang menentukan informasi tentang aplikasi, termasuk:

- Jenis peluncuran yang akan digunakan, yang menentukan infrastruktur tempat tugas Anda di-host.

  Saat Anda menggunakan tipe peluncuran EC2, Anda juga memilih jenis instans. Untuk beberapa jenis contoh, seperti GPU, Anda perlu mengatur parameter tambahan. Untuk informasi selengkapnya, lihat Kasus penggunaan definisi tugas.
- Gambar kontainer, yang menyimpan kode aplikasi Anda dan semua dependensi yang diperlukan kode aplikasi Anda untuk dijalankan.
- Mode jaringan yang digunakan untuk wadah dalam tugas Anda

  Mode jaringan menentukan bagaimana tugas Anda berkomunikasi melalui jaringan.

  Untuk tugas yang berjalan pada instans EC2, ada beberapa opsi, tetapi kami menyarankan Anda menggunakan mode `awsvpc` jaringan. Mode `awsvpc` jaringan menyederhanakan jaringan kontainer, karena Anda memiliki kontrol lebih besar atas bagaimana aplikasi Anda berkomunikasi satu sama lain dan layanan lain dalam VPC Anda.

  Untuk tugas yang berjalan di Fargate, Anda hanya dapat menggunakan mode `awsvpc` jaringan.
- Konfigurasi logging yang akan digunakan untuk tugas Anda.
- Setiap volume data yang digunakan dengan wadah dalam tugas.

Untuk daftar lengkap parameter definisi tugas, lihatParameter ketentuan tugas.

Gunakan panduan berikut saat membuat definisi tugas:

- Gunakan setiap keluarga definisi tugas hanya untuk satu tujuan bisnis.

  Jika Anda mengelompokkan beberapa jenis wadah aplikasi bersama-sama dalam definisi tugas yang sama, Anda tidak dapat menskalakan kontainer tersebut secara independen. Misalnya, tidak mungkin situs web dan API memerlukan penskalaan pada tingkat yang sama. Ketika lalu lintas meningkat, akan ada jumlah kontainer web yang berbeda yang diperlukan dari kontainer API. Jika

kedua kontainer ini digunakan dalam definisi tugas yang sama, setiap tugas menjalankan jumlah kontainer web dan kontainer API yang sama.

- Cocokkan setiap versi aplikasi dengan revisi definisi tugas dalam keluarga definisi tugas.

  Dalam keluarga definisi tugas, pertimbangkan setiap revisi definisi tugas sebagai snapshot titik waktu dari pengaturan untuk gambar kontainer tertentu. Ini mirip dengan bagaimana wadah adalah snapshot dari semua hal yang diperlukan untuk menjalankan versi tertentu dari kode aplikasi Anda.

  Pastikan ada one-to-one pemetaan antara versi kode aplikasi, tag gambar kontainer, dan revisi definisi tugas. Proses rilis tipikal melibatkan git commit yang diubah menjadi image container yang ditandai dengan git commit SHA. Kemudian, tag gambar kontainer itu mendapatkan revisi definisi tugas Amazon ECS sendiri. Terakhir, layanan Amazon ECS diperbarui untuk memintanya menerapkan revisi definisi tugas baru.
- Gunakan peran IAM yang berbeda untuk setiap keluarga definisi tugas.

  Tentukan setiap definisi tugas dengan peran IAM-nya sendiri. Rekomendasi ini harus dilakukan bersamaan dengan rekomendasi kami untuk menyediakan setiap komponen bisnis keluarga definisi tugasnya sendiri. Dengan menerapkan kedua praktik terbaik ini, Anda dapat membatasi seberapa banyak akses yang dimiliki setiap layanan ke sumber daya di AWS akun Anda. Misalnya, Anda dapat memberikan akses layanan otentikasi Anda untuk terhubung ke database kata sandi Anda. Pada saat yang sama, Anda juga dapat memastikan bahwa hanya layanan pesanan Anda yang memiliki akses ke informasi pembayaran kartu kredit.

Praktik terbaik untuk gambar kontainer

Gambar kontainer adalah seperangkat instruksi tentang cara membangun wadah. Gambar kontainer menyimpan kode aplikasi Anda dan semua dependensi yang diperlukan kode aplikasi Anda untuk dijalankan. Dependensi aplikasi mencakup paket kode sumber yang diandalkan oleh kode aplikasi Anda, runtime bahasa untuk bahasa yang ditafsirkan, dan paket biner yang diandalkan oleh kode yang ditautkan secara dinamis.

Gunakan panduan berikut saat Anda mendesain dan membuat gambar kontainer Anda:

- Buat gambar kontainer Anda lengkap dengan menyimpan semua dependensi aplikasi sebagai file statis di dalam gambar kontainer.

  Jika Anda mengubah sesuatu di image container, buat image container baru dengan perubahannya.

- Jalankan satu proses aplikasi dalam wadah.

  Masa pakai kontainer selama proses aplikasi berjalan. Amazon ECS menggantikan proses yang macet dan menentukan di mana harus meluncurkan proses penggantian. Gambar yang lengkap membuat penerapan keseluruhan lebih tangguh.

- Membuat Anda menangani aplikasi`SIGTERM`.

  Ketika Amazon ECS menghentikan tugas, pertama-tama ia mengirimkan sinyal SIGTERM ke tugas untuk memberi tahu aplikasi bahwa ia harus menyelesaikan dan mematikan. Amazon ECS kemudian mengirim `SIGKILL` pesan. Ketika aplikasi mengabaikan`SIGTERM`, layanan Amazon ECS harus menunggu untuk mengirim `SIGKILL` sinyal untuk menghentikan proses.

  Anda perlu mengidentifikasi berapa lama waktu yang dibutuhkan aplikasi Anda untuk menyelesaikan pekerjaannya, dan memastikan bahwa aplikasi Anda menangani `SIGTERM` sinyal. Penanganan sinyal aplikasi perlu menghentikan aplikasi dari mengambil pekerjaan baru dan menyelesaikan pekerjaan yang sedang berlangsung, atau menyimpan pekerjaan yang belum selesai ke penyimpanan di luar tugas ketika pekerjaan terlalu lama untuk diselesaikan.

- Konfigurasikan aplikasi kontainer untuk menulis log ke `stdout` dan. `stderr`

  Memisahkan penanganan log dari kode aplikasi Anda memberi Anda fleksibilitas untuk menyesuaikan penanganan log di tingkat infrastruktur. Salah satu contohnya adalah mengubah sistem logging Anda. Alih-alih memodifikasi layanan Anda, dan membangun dan menerapkan gambar kontainer baru, Anda dapat menyesuaikan pengaturan.

- Gunakan tag untuk membuat versi gambar kontainer Anda.

  Gambar kontainer disimpan dalam registri kontainer. Setiap gambar dalam registri diidentifikasi oleh tag. Ada tag yang disebut`latest`. Tag ini berfungsi sebagai penunjuk ke versi terbaru dari gambar wadah aplikasi, mirip dengan `HEAD` di repositori git. Kami menyarankan Anda menggunakan `latest` tag hanya untuk tujuan pengujian. Sebagai praktik terbaik, beri tag gambar wadah dengan tag unik untuk setiap build. Kami menyarankan Anda menandai gambar Anda menggunakan git SHA untuk git commit yang digunakan untuk membangun gambar.

  Anda tidak perlu membuat image kontainer untuk setiap komit. Namun, kami menyarankan Anda membuat image kontainer baru setiap kali Anda merilis kode tertentu yang berkomitmen ke lingkungan produksi. Kami juga menyarankan Anda menandai gambar dengan tag yang sesuai dengan git commit dari kode yang ada di dalam gambar. Jika Anda menandai gambar dengan git commit, Anda dapat lebih cepat menemukan versi kode mana yang sedang dijalankan gambar.

Kami juga menyarankan Anda mengaktifkan tag gambar yang tidak dapat diubah di Amazon Elastic Container Registry. Dengan pengaturan ini, Anda tidak dapat mengubah gambar kontainer yang ditunjuk tag. Sebaliknya Amazon ECR memberlakukan bahwa gambar baru harus diunggah ke tag baru. Untuk informasi selengkapnya, lihat Mutabilitas tag gambar di Panduan Pengguna Amazon ECR.

Saat Anda merancang aplikasi untuk dijalankan AWS Fargate, Anda harus memutuskan antara menerapkan beberapa kontainer ke dalam definisi tugas yang sama dan menerapkan kontainer secara terpisah dalam beberapa definisi tugas. Jika kondisi berikut diperlukan, sebaiknya gunakan beberapa kontainer ke dalam definisi tugas yang sama:

- Kontainer Anda berbagi siklus hidup yang sama (yaitu, mereka diluncurkan dan dihentikan bersama-sama).
- Kontainer Anda harus berjalan pada host dasar yang sama (yaitu, satu kontainer mereferensikan yang lain pada port localhost).
- Kontainer Anda berbagi sumber daya.
- Kontainer Anda berbagi volume data.

Jika kondisi ini tidak diperlukan, sebaiknya gunakan kontainer secara terpisah dalam beberapa definisi tugas. Ini memungkinkan Anda untuk menskalakan, menyediakan, dan menghentikan penyediaan wadah secara terpisah.

Praktik terbaik untuk ukuran tugas Amazon ECS

Salah satu pilihan terpenting yang harus dibuat saat menerapkan kontainer di Amazon ECS adalah wadah dan ukuran tugas Anda. Ukuran wadah dan tugas Anda sangat penting untuk penskalaan dan perencanaan kapasitas. Di Amazon ECS, ada dua metrik sumber daya yang digunakan untuk kapasitas: CPU dan memori. CPU diukur dalam satuan 1/1024 dari vCPU penuh (di mana 1024 unit sama dengan 1 vCPU utuh). Memori diukur dalam megabyte. Dalam definisi tugas Anda, Anda dapat mendeklarasikan reservasi dan batasan sumber daya.

Saat Anda mendeklarasikan reservasi, Anda mendeklarasikan jumlah minimum sumber daya yang dibutuhkan tugas. Tugas Anda menerima setidaknya jumlah sumber daya yang diminta. Aplikasi Anda mungkin dapat menggunakan lebih banyak CPU atau memori daripada reservasi yang Anda deklarasikan. Namun, ini tunduk pada batasan apa pun yang juga Anda nyatakan. Menggunakan lebih dari jumlah reservasi dikenal sebagai bursting. Di Amazon ECS, reservasi dijamin. Misalnya,

jika Anda menggunakan instans Amazon EC2 untuk menyediakan kapasitas, Amazon ECS tidak menempatkan tugas pada instans yang reservasi tidak dapat dipenuhi.

Batas adalah jumlah maksimum unit CPU atau memori yang dapat digunakan oleh wadah atau tugas Anda. Setiap upaya untuk menggunakan lebih banyak CPU lebih dari batas ini menghasilkan pelambatan. Setiap upaya untuk menggunakan lebih banyak memori menghasilkan penampung Anda dihentikan.

Memilih nilai-nilai ini bisa menjadi tantangan. Ini karena nilai-nilai yang paling cocok untuk aplikasi Anda sangat bergantung pada kebutuhan sumber daya aplikasi Anda. Pengujian beban aplikasi Anda adalah kunci keberhasilan perencanaan kebutuhan sumber daya dan lebih memahami persyaratan aplikasi Anda.

Aplikasi tanpa kewarganegaraan

Untuk aplikasi stateless yang menskalakan secara horizontal, seperti aplikasi di belakang penyeimbang beban, sebaiknya Anda terlebih dahulu menentukan jumlah memori yang dikonsumsi aplikasi Anda saat melayani permintaan. Untuk melakukan ini, Anda dapat menggunakan alat tradisional seperti `ps` atau`top`, atau solusi pemantauan seperti CloudWatch Wawasan Kontainer.

Saat menentukan reservasi CPU, pertimbangkan bagaimana Anda ingin menskalakan aplikasi Anda untuk memenuhi persyaratan bisnis Anda. Anda dapat menggunakan reservasi CPU yang lebih kecil, seperti 256 unit CPU (atau 1/4 vCPU), untuk meningkatkan skala dengan cara halus yang meminimalkan biaya. Tapi, mereka mungkin tidak berskala cukup cepat untuk memenuhi lonjakan permintaan yang signifikan. Anda dapat menggunakan reservasi CPU yang lebih besar untuk skala masuk dan keluar lebih cepat dan karenanya mencocokkan lonjakan permintaan lebih cepat. Namun, pemesanan CPU yang lebih besar lebih mahal.

Aplikasi lainnya

Untuk aplikasi yang tidak menskalakan secara horizontal, seperti pekerja tunggal atau server database, kapasitas dan biaya yang tersedia mewakili pertimbangan Anda yang paling penting. Anda harus memilih jumlah memori dan CPU berdasarkan pengujian beban yang menunjukkan bahwa Anda perlu melayani lalu lintas untuk memenuhi tujuan tingkat layanan Anda. Amazon ECS memastikan bahwa aplikasi ditempatkan pada host yang memiliki kapasitas yang memadai.

Jaringan tugas untuk tugas di instans Amazon EC2

Perilaku jaringan tugas Amazon ECS yang di-host di instans Amazon EC2 bergantung pada mode jaringan yang ditentukan dalam definisi tugas. Kami menyarankan Anda menggunakan mode

`awsvpc` jaringan kecuali Anda memiliki kebutuhan khusus untuk menggunakan mode jaringan yang berbeda.

Berikut ini adalah mode jaringan yang tersedia.

Mode jaringan	Wadah Linux di EC2	Wadah Windows di EC2	Deskripsi
`awsvpc`	Ya	Ya	Tugas ini dialokasikan elastic network interface (ENI) sendiri dan alamat IPv4 pribadi utama. Ini memberi tugas properti jaringan yang sama dengan instans Amazon EC2.
`bridge`	Ya	Tidak	Tugas ini menggunakan jaringan virtual bawaan Docker di Linux, yang berjalan di dalam setiap instans Amazon EC2 yang menghosting tugas tersebut. Jaringan virtual bawaan di Linux menggunakan driver jaringan `bridge` Docker. Ini adalah mode jaringan default di Linux jika mode jaringan tidak ditentukan dalam definisi tugas.
`host`	Ya	Tidak	Tugas menggunakan jaringan host yang melewati jaringan virtual bawaan Docker dengan memetakan port kontainer langsung ke ENI dari instans Amazon EC2 yang menjadi tuan rumah tugas. Pemetaan port dinamis tidak dapat digunakan dalam mode jaringan ini. Wadah dalam definisi tugas yang menggunak an mode ini harus menentukan `hostPort` nomor tertentu. Nomor port pada host tidak dapat digunakan oleh banyak tugas. Akibatnya, Anda tidak dapat menjalankan beberapa tugas dengan definisi tugas yang sama pada satu instans Amazon EC2.

Mode jaringan	Wadah Linux di EC2	Wadah Windows di EC2	Deskripsi
`none`	Ya	Tidak	Tugas tidak memiliki konektivitas jaringan eksternal.
`default`	Tidak	Ya	Tugas ini menggunakan jaringan virtual bawaan Docker di Windows, yang berjalan di dalam setiap instans Amazon EC2 yang menghosting tugas tersebut. Jaringan virtual bawaan pada Windows menggunakan driver jaringan `nat` Docker. Ini adalah mode jaringan default pada Windows jika mode jaringan tidak ditentukan dalam definisi tugas.

Untuk informasi selengkapnya tentang jaringan Docker di Linux, lihat Ikhtisar jaringan di Dokumentasi Docker.

Untuk informasi selengkapnya tentang jaringan Docker di Windows, lihat Jaringan kontainer Windows di Microsoft Container di Dokumentasi Windows.

Topik

- awsvpcmodus jaringan
- Mode host
- Mode jembatan

`awsvpc`modus jaringan

Fitur jaringan tugas yang disediakan oleh mode `awsvpc` jaringan memberikan tugas Amazon ECS properti jaringan yang sama dengan instans Amazon EC2. Menggunakan mode `awsvpc` jaringan menyederhanakan jaringan kontainer, karena Anda memiliki kontrol lebih besar atas bagaimana aplikasi Anda berkomunikasi satu sama lain dan layanan lain dalam VPC Anda. Mode `awsvpc` jaringan juga memberikan keamanan yang lebih besar untuk wadah Anda dengan memungkinkan Anda menggunakan grup keamanan dan alat pemantauan jaringan pada tingkat yang lebih terperinci dalam tugas Anda. Anda juga dapat menggunakan fitur jaringan Amazon EC2 lainnya seperti VPC

Flow Logs untuk memantau lalu lintas ke dan dari tugas Anda. Selain itu, kontainer yang memiliki tugas yang sama dapat berkomunikasi melalui antarmuka `localhost`.

Task elastic network interface (ENI) adalah fitur Amazon ECS yang dikelola sepenuhnya. Amazon ECS membuat ENI dan menempelkannya ke instans Amazon EC2 host dengan grup keamanan yang ditentukan. Tugas mengirim dan menerima lalu lintas jaringan melalui ENI dengan cara yang sama seperti instans Amazon EC2 lakukan dengan antarmuka jaringan utama mereka. Setiap tugas ENI diberikan alamat IPv4 privat secara default. Jika VPC Anda diaktifkan untuk mode tumpukan dobel dan Anda menggunakan subnet dengan blok CIDR IPv6, tugas ENI Anda juga akan menerima alamat IPv6. Setiap tugas hanya dapat memiliki satu ENI.

ENI ini terlihat di konsol Amazon EC2 untuk akun Anda. Akun Anda tidak dapat melepaskan atau memodifikasi ENI. Hal ini untuk mencegah penghapusan ENI disengaja yang berkaitan dengan tugas yang sedang berjalan. Anda dapat melihat informasi lampiran ENI untuk tugas di konsol Amazon ECS atau dengan operasi DescribeTasksAPI. Ketika tugas berhenti atau jika layanan menurunkan skala, tugas ENI terlepas dan dihapus.

Saat Anda membutuhkan peningkatan kepadatan ENI, gunakan pengaturan `awsvpcTrunking` akun. Amazon ECS juga membuat dan melampirkan antarmuka jaringan “trunk” untuk instance container Anda. Jaringan trunk sepenuhnya dikelola oleh Amazon ECS. Trunk ENI akan dihapus saat Anda menghentikan atau membatalkan pendaftaran instance container Anda dari klaster Amazon ECS. Untuk informasi selengkapnya tentang pengaturan `awsvpcTrunking` akun, lihatPrasyarat.

Anda menentukan `awsvpc` dalam `networkMode` parameter definisi tugas. Untuk informasi selengkapnya, lihat Mode jaringan.

Kemudian, ketika Anda menjalankan tugas atau membuat layanan, gunakan `networkConfiguration` parameter yang menyertakan satu atau beberapa subnet untuk menempatkan tugas Anda dan satu atau beberapa grup keamanan untuk dilampirkan ke ENI. Untuk informasi selengkapnya, lihat Konfigurasi jaringan. Tugas ditempatkan pada instans Amazon EC2 yang kompatibel di Availability Zone yang sama dengan subnet tersebut, dan grup keamanan yang ditentukan dikaitkan dengan ENI yang disediakan untuk tugas tersebut.

Pertimbangan Linux

Pertimbangkan hal berikut saat menggunakan sistem operasi Linux.

- Tugas dan layanan yang menggunakan mode `awsvpc` jaringan memerlukan peran terkait layanan Amazon ECS untuk memberi Amazon ECS izin untuk melakukan panggilan ke layanan lain AWS

atas nama Anda. Peran ini dibuat untuk Anda secara otomatis ketika Anda membuat klaster atau jika Anda membuat atau memperbarui layanan, di AWS Management Console. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS. Anda juga dapat membuat peran terkait layanan dengan perintah berikut: AWS CLI

```
aws iam create-service-linked-role --aws-service-name ecs.amazonaws.com
```

- Instans Amazon EC2 Linux Anda memerlukan versi `1.15.0` atau yang lebih baru dari agen penampung untuk menjalankan tugas yang menggunakan mode `awsvpc` jaringan. Jika Anda menggunakan AMI Amazon ECS yang dioptimalkan, instans Anda memerlukan setidaknya `1.15.0-4` versi `ecs-init` paket juga.
- Amazon ECS mengisi nama host tugas dengan nama host DNS (internal) yang disediakan Amazon saat opsi dan `enableDnsHostnames` opsi `enableDnsSupport` diaktifkan di VPC Anda. Jika opsi ini tidak diaktifkan, nama host DNS tugas disetel ke nama host acak. Untuk informasi selengkapnya tentang pengaturan DNS untuk VPC, lihat Menggunakan DNS dengan VPC Anda di Panduan Pengguna Amazon VPC.
- Setiap tugas Amazon ECS yang menggunakan mode `awsvpc` jaringan menerima elastic network interface (ENI) sendiri, yang dilampirkan ke instans Amazon EC2 yang menghostingnya. Ada kuota default untuk jumlah antarmuka jaringan yang dapat dilampirkan ke instans Amazon EC2 Linux. Antarmuka jaringan utama dihitung sebagai satu terhadap kuota itu. Misalnya, secara default, sebuah `c5.large` instance mungkin hanya memiliki hingga tiga ENI yang dapat dilampirkan padanya. Antarmuka jaringan utama untuk instance dihitung sebagai satu. Anda dapat melampirkan dua ENI tambahan ke instance. Karena setiap tugas yang menggunakan mode `awsvpc` jaringan memerlukan ENI, Anda biasanya hanya dapat menjalankan dua tugas tersebut pada jenis instance ini. Untuk informasi selengkapnya tentang batas ENI default untuk setiap jenis instans, lihat alamat IP per antarmuka jaringan per jenis instans di Panduan Pengguna Amazon EC2 untuk Instans Linux.
- Amazon ECS mendukung peluncuran instans Amazon EC2 Linux yang menggunakan tipe instans yang didukung dengan peningkatan kepadatan ENI. Saat Anda memilih setelan `awsvpcTrunking` akun dan mendaftarkan instans Amazon EC2 Linux yang menggunakan jenis instans ini ke klaster Anda, instans ini memiliki kuota ENI yang lebih tinggi. Menggunakan instans ini dengan kuota yang lebih tinggi ini berarti Anda dapat menempatkan lebih banyak tugas di setiap instans Amazon EC2 Linux. Untuk menggunakan peningkatan kepadatan ENI dengan fitur trunking, instans Amazon EC2 Anda harus menggunakan agen penampung `1.28.1` versi atau yang lebih baru. Jika Anda menggunakan AMI Amazon ECS yang dioptimalkan, instans Anda juga memerlukan setidaknya `1.28.1-2` versi paket. `ecs-init` Untuk informasi lebih lanjut tentang penyertaan pada

pengaturan akun `awsvpcTrunking`, lihat Mengakses fitur Amazon ECS melalui pengaturan akun. Untuk informasi lebih lanjut tentang trunking ENI, lihat. Pembuatan torso antarmuka jaringan elastis

- Saat menghosting tugas yang menggunakan mode `awsvpc` jaringan di instans Amazon EC2 Linux, ENI tugas Anda tidak diberikan alamat IP publik. Untuk mengakses internet, tugas harus diluncurkan di subnet pribadi yang dikonfigurasi untuk menggunakan gateway NAT. Untuk informasi lebih lanjut, lihat Gateway NAT dalam Panduan Pengguna Amazon VPC. Akses jaringan masuk harus dari dalam VPC yang menggunakan alamat IP pribadi atau dirutekan melalui penyeimbang beban dari dalam VPC. Tugas yang diluncurkan dalam subnet publik tidak memiliki akses ke internet.
- Amazon ECS hanya mengenali ENI yang dilampirkan ke instans Amazon EC2 Linux Anda. Jika Anda melampirkan ENI secara manual ke instans Anda, Amazon ECS mungkin mencoba menambahkan tugas ke instance yang tidak memiliki cukup adaptor jaringan. Hal ini dapat mengakibatkan waktu tugas habis dan pindah ke status deprovisioning dan kemudian status berhenti. Kami menyarankan agar Anda tidak melampirkan ENI ke instans Anda secara manual.
- Instans Amazon EC2 Linux harus terdaftar dengan `ecs.capability.task-eni` kemampuan yang harus dipertimbangkan untuk penempatan tugas dengan mode jaringan. `awsvpc` Instans yang menjalankan versi `1.15.0-4` atau yang lebih baru `ecs-init` terdaftar dengan atribut ini secara otomatis.
- ENI yang dibuat dan dilampirkan ke instans Amazon EC2 Linux Anda tidak dapat dilepaskan secara manual atau dimodifikasi oleh akun Anda. Hal ini untuk mencegah menghapus ENI disengaja yang berhubungan dengan tugas yang sedang berjalan. Untuk melepaskan ENI untuk suatu tugas, hentikan tugas tersebut.
- Ada batas 16 subnet dan 5 grup keamanan yang dapat ditentukan dalam `awsVpcConfiguration` saat menjalankan tugas atau membuat layanan yang menggunakan mode jaringan `awsvpc`. Untuk informasi selengkapnya, lihat AwsVpcConfigurationdi Referensi API Amazon Elastic Container Service.
- Saat tugas dimulai dengan mode `awsvpc` jaringan, agen penampung Amazon ECS membuat `pause` wadah tambahan untuk setiap tugas sebelum memulai kontainer dalam definisi tugas. Kemudian mengkonfigurasi namespace jaringan `pause` wadah dengan menjalankan plugin CNI. amazon-ecs-cni-plugins Agen kemudian memulai sisa kontainer dalam tugas sehingga mereka berbagi tumpukan jaringan kontainer `pause`. Ini berarti bahwa semua kontainer dalam tugas yang dialamatkan oleh alamat IP dari ENI, dan mereka dapat berkomunikasi satu sama lain melalui antarmuka `localhost`.
- Layanan dengan tugas yang menggunakan mode `awsvpc` jaringan hanya mendukung Application Load Balancer dan Network Load Balancer. Saat Anda membuat grup target apa pun untuk

layanan ini, Anda harus memilih `ip` sebagai jenis target. Jangan gunakan `instance`. Ini karena tugas yang menggunakan mode `awsvpc` jaringan dikaitkan dengan ENI, bukan dengan instans Amazon EC2 Linux. Untuk informasi selengkapnya, lihat Mendistribusikan lalu lintas layanan Amazon ECS menggunakan load balancing.

- Jika VPC Anda diperbarui untuk mengubah set opsi DHCP yang digunakannya, Anda tidak dapat menerapkan perubahan ini ke tugas yang ada. Mulai tugas baru dengan perubahan ini diterapkan padanya, verifikasi bahwa mereka berfungsi dengan benar, dan kemudian hentikan tugas yang ada untuk mengubah konfigurasi jaringan ini dengan aman.

Pertimbangan Windows

Berikut ini adalah pertimbangan ketika Anda menggunakan sistem operasi Windows:

- Instans kontainer menggunakan Amazon ECS yang dioptimalkan Windows Server 2016 AMI tidak dapat meng-host tugas yang menggunakan mode `awsvpc` jaringan. Jika Anda memiliki klaster yang berisi AMI Windows Server 2016 Amazon ECS yang dioptimalkan dan AMI Windows yang mendukung mode `awsvpc` `awsvpc` jaringan, tugas yang menggunakan mode jaringan tidak diluncurkan pada instance Windows 2016 Server. Sebaliknya, mereka diluncurkan pada instance yang mendukung mode `awsvpc` jaringan.
- Instans Windows Amazon EC2 Anda memerlukan versi `1.57.1` atau yang lebih baru dari agen penampung untuk menggunakan CloudWatch metrik untuk kontainer Windows yang menggunakan mode jaringan. `awsvpc`
- Tugas dan layanan yang menggunakan mode `awsvpc` jaringan memerlukan peran terkait layanan Amazon ECS untuk memberi Amazon ECS izin untuk melakukan panggilan ke layanan lain AWS atas nama Anda. Peran ini dibuat untuk Anda secara otomatis ketika Anda membuat klaster, atau jika Anda membuat atau memperbarui layanan di AWS Management Console. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS. Anda juga dapat membuat peran terkait layanan dengan perintah berikut AWS CLI .

  ```
  aws iam create-service-linked-role --aws-service-name ecs.amazonaws.com
  ```

- Instans Windows Amazon EC2 Anda memerlukan versi `1.54.0` atau yang lebih baru dari agen penampung untuk menjalankan tugas yang menggunakan mode `awsvpc` jaringan. Saat Anda mem-bootstrap instance, Anda harus mengonfigurasi opsi yang diperlukan untuk mode `awsvpc` jaringan. Untuk informasi selengkapnya, lihat the section called “Instans Kontainer Bootstrap”.
- Amazon ECS mengisi nama host tugas dengan nama host DNS (internal) yang disediakan Amazon saat `enableDnsSupport` opsi `enableDnsHostnames` dan opsi diaktifkan di VPC Anda. Jika opsi

ini tidak diaktifkan, nama host DNS tugas adalah nama host acak. Untuk informasi selengkapnya tentang pengaturan DNS untuk VPC, lihat Menggunakan DNS dengan VPC Anda di Panduan Pengguna Amazon VPC.

- Setiap tugas Amazon ECS yang menggunakan mode `awsvpc` jaringan menerima elastic network interface (ENI) sendiri, yang dilampirkan ke instans Amazon EC2 Windows yang menghostingnya. Ada kuota default untuk jumlah antarmuka jaringan yang dapat dilampirkan ke instans Windows Amazon EC2. Antarmuka jaringan utama dihitung sebagai satu terhadap kuota ini. Misalnya, secara default sebuah `c5.large` instance mungkin hanya memiliki hingga tiga ENI yang dilampirkan padanya. Antarmuka jaringan utama untuk instance dihitung sebagai salah satunya. Anda dapat melampirkan dua ENI tambahan ke instance. Karena setiap tugas menggunakan mode jaringan `awsvpc` memerlukan ENI, Anda biasanya hanya dapat menjalankan dua tugas tersebut pada Tipe instans ini. Untuk informasi selengkapnya tentang batas ENI default untuk setiap jenis instans, lihat alamat IP per antarmuka jaringan per jenis instans di Panduan Pengguna Amazon EC2 untuk Instans Windows.
- Saat menghosting tugas yang menggunakan mode `awsvpc` jaringan di instans Amazon EC2 Windows, ENI tugas Anda tidak diberikan alamat IP publik. Untuk mengakses internet, luncurkan tugas di subnet pribadi yang dikonfigurasi untuk menggunakan gateway NAT. Untuk informasi lebih lanjut, lihat Gateway NAT dalam Panduan Pengguna Amazon VPC. Akses jaringan masuk harus dari dalam VPC yang menggunakan alamat IP pribadi atau dirutekan melalui penyeimbang beban dari dalam VPC. Tugas yang diluncurkan dalam subnet publik tidak memiliki akses ke internet.
- Amazon ECS hanya mengenali ENI yang telah dilampirkan ke instans Amazon EC2 Windows Anda. Jika Anda melampirkan ENI secara manual ke instans Anda, Amazon ECS mungkin mencoba menambahkan tugas ke instance yang tidak memiliki cukup adaptor jaringan. Hal ini dapat mengakibatkan waktu tugas habis dan pindah ke status deprovisioning dan kemudian status berhenti. Kami menyarankan agar Anda tidak melampirkan ENI ke instans Anda secara manual.
- Instans Windows Amazon EC2 harus terdaftar dengan `ecs.capability.task-eni` kemampuan yang harus dipertimbangkan untuk penempatan tugas dengan mode jaringan. `awsvpc`
- Anda tidak dapat memodifikasi atau melepaskan ENI secara manual yang dibuat dan dilampirkan ke instans Windows Amazon EC2 Anda. Ini untuk mencegah Anda secara tidak sengaja menghapus ENI yang terkait dengan tugas yang sedang berjalan. Untuk melepaskan ENI untuk suatu tugas, hentikan tugas tersebut.
- Anda hanya dapat menentukan hingga 16 subnet dan 5 grup keamanan `awsVpcConfiguration` ketika Anda menjalankan tugas atau membuat layanan yang menggunakan mode `awsvpc` jaringan. Untuk informasi selengkapnya, lihat AwsVpcConfigurationdi Referensi API Amazon Elastic Container Service.

- Saat tugas dimulai dengan mode `awsvpc` jaringan, agen penampung Amazon ECS membuat `pause` wadah tambahan untuk setiap tugas sebelum memulai kontainer dalam definisi tugas. Kemudian mengkonfigurasi namespace jaringan `pause` wadah dengan menjalankan plugin CNI. amazon-ecs-cni-plugins Agen kemudian memulai sisa kontainer dalam tugas sehingga mereka berbagi tumpukan jaringan kontainer `pause`. Ini berarti bahwa semua kontainer dalam tugas yang dialamatkan oleh alamat IP dari ENI, dan mereka dapat berkomunikasi satu sama lain melalui antarmuka `localhost`.
- Layanan dengan tugas yang menggunakan mode `awsvpc` jaringan hanya mendukung Application Load Balancer dan Network Load Balancer. Saat Anda membuat grup target apa pun untuk layanan ini, Anda harus memilih `ip` sebagai jenis target, bukan`instance`. Ini karena tugas yang menggunakan mode `awsvpc` jaringan dikaitkan dengan ENI, bukan dengan instans Windows Amazon EC2. Untuk informasi selengkapnya, lihat Mendistribusikan lalu lintas layanan Amazon ECS menggunakan load balancing.
- Jika VPC Anda diperbarui untuk mengubah set opsi DHCP yang digunakannya, Anda tidak dapat menerapkan perubahan ini ke tugas yang ada. Mulai tugas baru dengan perubahan ini diterapkan padanya, verifikasi bahwa mereka berfungsi dengan benar, dan kemudian hentikan tugas yang ada untuk mengubah konfigurasi jaringan ini dengan aman.
- Berikut ini tidak didukung saat Anda menggunakan mode `awsvpc` jaringan dalam konfigurasi Windows EC2:
  - Konfigurasikan tumpukan dobel
  - IPv6
  - ENI trunking

Menggunakan VPC dalam mode tumpukan dobel

Saat menggunakan VPC dalam mode dual-stack, tugas Anda dapat berkomunikasi melalui IPv4, atau IPv6, atau keduanya. Alamat IPv4 dan IPv6 tidak bergantung satu sama lain. Oleh karena itu Anda harus mengonfigurasi perutean dan keamanan di VPC Anda secara terpisah untuk IPv4 dan IPv6. Untuk informasi selengkapnya tentang cara mengonfigurasi VPC Anda untuk mode dual-stack, lihat Memigrasi ke IPv6 di Panduan Pengguna Amazon VPC.

Jika Anda mengonfigurasi VPC Anda dengan gateway internet atau gateway internet khusus keluar, Anda dapat menggunakan VPC Anda dalam mode dual-stack. Dengan melakukan ini, tugas-tugas yang diberi alamat IPv6 dapat mengakses internet melalui gateway internet atau gateway internet khusus egres. Gateway NAT bersifat opsional. Untuk informasi selengkapnya, lihat gateway Internet dan gateway internet khusus eGress di Panduan Pengguna Amazon VPC.

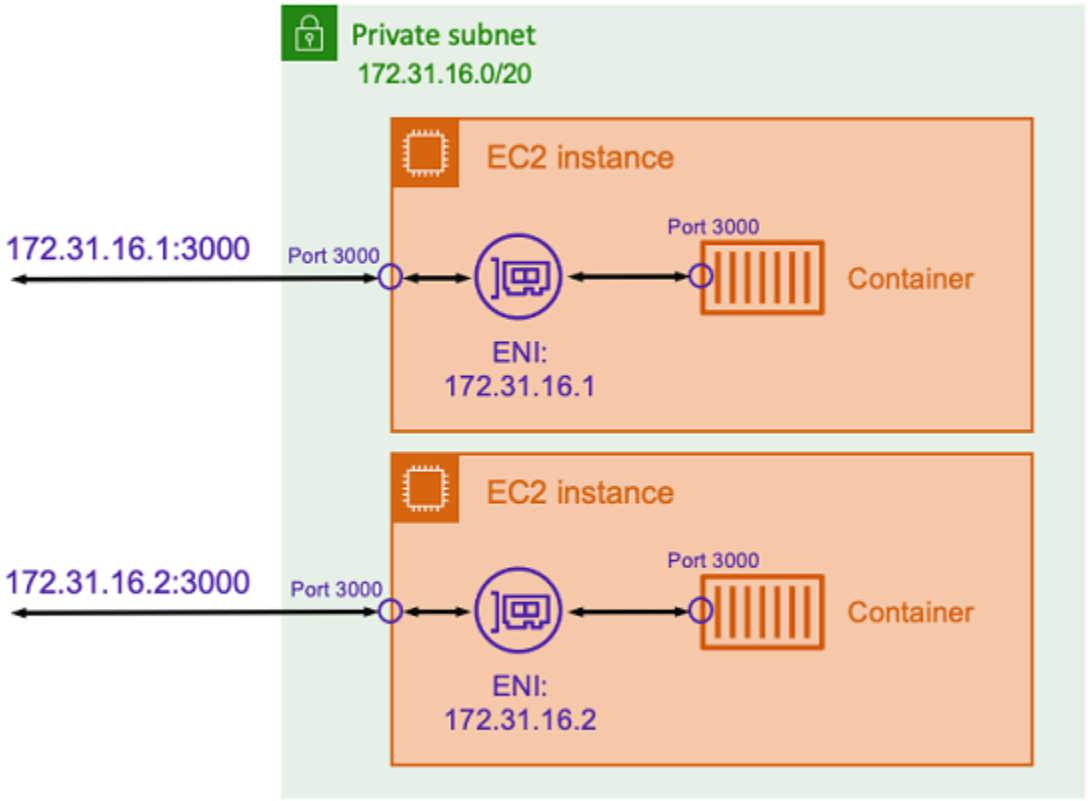
Tugas Amazon ECS diberikan alamat IPv6 jika kondisi berikut terpenuhi:

- Instans Amazon EC2 Linux yang menghosting tugas menggunakan versi `1.45.0` atau yang lebih baru dari agen penampung. Untuk informasi tentang cara memeriksa versi agen yang digunakan instans Anda, dan memperbaruinya jika diperlukan, lihatMemperbarui agen kontainer Amazon ECS.
- Pengaturan akun `dualStackIPv6` diaktifkan. Untuk informasi selengkapnya, lihat Mengakses fitur Amazon ECS melalui pengaturan akun.
- Tugas Anda adalah menggunakan mode jaringan `awsvpc`.
- VPC dan subnet Anda dikonfigurasi untuk IPv6. Konfigurasi mencakup antarmuka jaringan yang dibuat di subnet yang ditentukan. Untuk informasi selengkapnya tentang cara mengonfigurasi VPC Anda untuk mode dual-stack, lihat Memigrasi ke IPv6 dan Memodifikasi atribut pengalamatan IPv6 untuk subnet Anda di Panduan Pengguna Amazon VPC.

Mode host

Mode `host` jaringan hanya didukung untuk tugas Amazon ECS yang dihosting di instans Amazon EC2. Ini tidak didukung saat menggunakan Amazon ECS di Fargate.

Mode `host` jaringan adalah mode jaringan paling dasar yang didukung di Amazon ECS. Menggunakan mode host, jaringan kontainer terikat langsung ke host yang mendasari yang menjalankan wadah.



Asumsikan bahwa Anda menjalankan kontainer Node.js dengan aplikasi Express yang mendengarkan pada port yang `3000` mirip dengan yang diilustrasikan dalam diagram sebelumnya. Ketika mode `host` jaringan digunakan, kontainer menerima lalu lintas pada port 3000 menggunakan alamat IP dari instans Amazon EC2 host yang mendasarinya. Kami tidak menyarankan menggunakan mode ini.

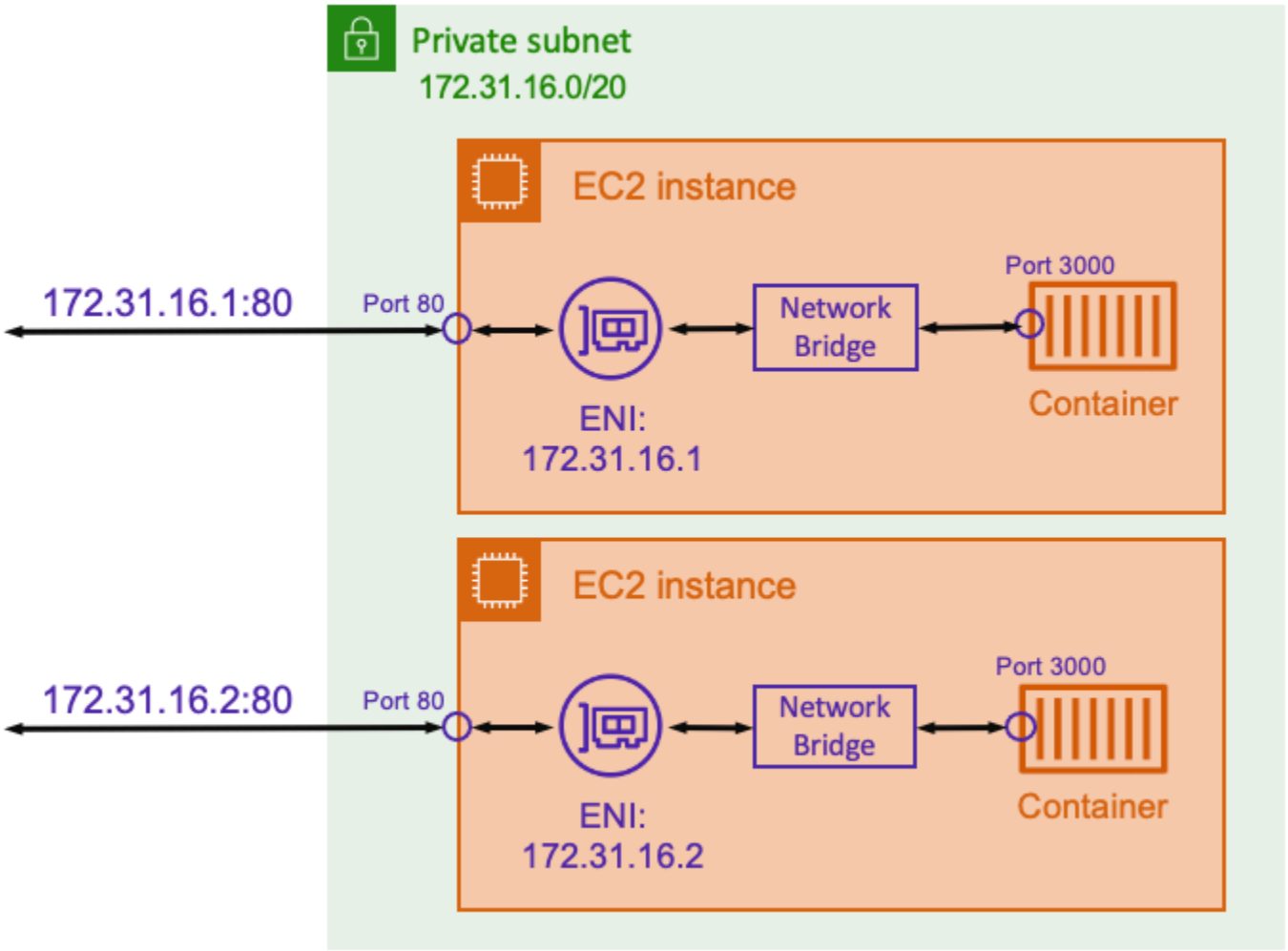
Ada kelemahan signifikan untuk menggunakan mode jaringan ini. Anda tidak dapat menjalankan lebih dari satu instantiasi tugas pada setiap host. Ini karena hanya tugas pertama yang dapat mengikat ke port yang diperlukan pada instans Amazon EC2. Juga tidak ada cara untuk memetakan ulang port kontainer saat menggunakan mode `host` jaringan. Misalnya, jika aplikasi perlu mendengarkan nomor port tertentu, Anda tidak dapat memetakan ulang nomor port secara langsung. Sebagai gantinya, Anda harus mengelola konflik port apa pun dengan mengubah konfigurasi aplikasi.

Ada juga implikasi keamanan saat menggunakan mode `host` jaringan. Mode ini memungkinkan kontainer untuk meniru host, dan memungkinkan kontainer untuk terhubung ke layanan jaringan loopback pribadi pada host.

Mode jembatan

Mode `bridge` jaringan hanya didukung untuk tugas Amazon ECS yang dihosting di instans Amazon EC2.

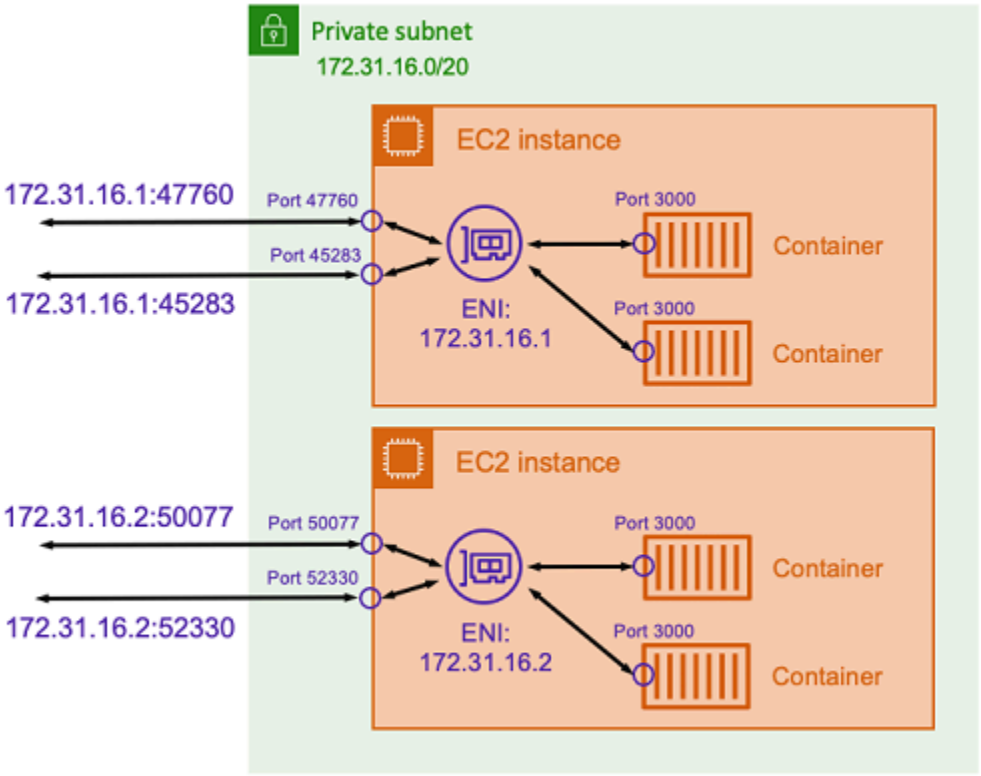
Dengan `bridge` mode, Anda menggunakan jembatan jaringan virtual untuk membuat lapisan antara host dan jaringan wadah. Dengan cara ini, Anda dapat membuat pemetaan port yang memetakan ulang port host ke port kontainer. Pemetaan dapat berupa statis atau dinamis.



Dengan pemetaan port statis, Anda dapat secara eksplisit menentukan port host mana yang ingin Anda petakan ke port kontainer. Menggunakan contoh di atas, port `80` pada host sedang dipetakan ke port `3000` pada container. Untuk berkomunikasi dengan aplikasi kontainer, Anda mengirim lalu lintas ke port `80` ke alamat IP instans Amazon EC2. Dari perspektif aplikasi kontainer, ia melihat lalu lintas masuk di port. `3000`

Jika Anda hanya ingin mengubah port lalu lintas, maka pemetaan port statis cocok. Namun, ini masih memiliki kelemahan yang sama dengan menggunakan mode `host` jaringan. Anda tidak dapat menjalankan lebih dari satu instantiasi tugas pada setiap host. Ini karena pemetaan port statis hanya memungkinkan satu kontainer untuk dipetakan ke port 80.

Untuk mengatasi masalah ini, pertimbangkan untuk menggunakan mode `bridge` jaringan dengan pemetaan port dinamis seperti yang ditunjukkan pada diagram berikut.



Dengan tidak menentukan port host dalam pemetaan port, Anda dapat meminta Docker memilih port acak yang tidak digunakan dari rentang port sementara dan menetapkannya sebagai port host publik untuk wadah. Misalnya, aplikasi Node.js yang mendengarkan pada port `3000` pada kontainer mungkin diberi port nomor tinggi acak seperti `47760` pada host Amazon EC2. Melakukan ini berarti Anda dapat menjalankan beberapa salinan wadah itu di host. Selain itu, setiap kontainer dapat diberi port sendiri di host. Setiap salinan kontainer menerima lalu lintas di pelabuhan`3000`. Namun, klien yang mengirim lalu lintas ke kontainer ini menggunakan port host yang ditetapkan secara acak.

Amazon ECS membantu Anda melacak port yang ditetapkan secara acak untuk setiap tugas. Hal ini dilakukan dengan secara otomatis memperbarui kelompok target load balancer dan penemuan AWS Cloud Map layanan untuk memiliki daftar alamat IP tugas dan port. Ini membuatnya lebih mudah untuk menggunakan layanan yang beroperasi menggunakan `bridge` mode dengan port dinamis.

Namun, salah satu kelemahan menggunakan mode `bridge` jaringan adalah sulit untuk mengunci komunikasi layanan ke layanan. Karena layanan mungkin ditetapkan ke port acak dan tidak terpakai, maka perlu untuk membuka rentang port yang luas antar host. Namun, tidak mudah untuk membuat aturan khusus sehingga layanan tertentu hanya dapat berkomunikasi dengan satu layanan spesifik lainnya. Layanan tidak memiliki port khusus untuk digunakan untuk aturan jaringan grup keamanan.

Jaringan tugas untuk tugas-tugas di Fargate

Secara default, setiap tugas Amazon ECS di Fargate disediakan elastic network interface (ENI) dengan alamat IP pribadi utama. Saat menggunakan subnet publik, Anda dapat secara opsional menetapkan alamat IP publik ke ENI tugas. Jika VPC Anda dikonfigurasi untuk mode dual-stack dan Anda menggunakan subnet dengan blok CIDR IPv6, ENI tugas Anda juga menerima alamat IPv6. Sebuah tugas hanya dapat memiliki satu ENI yang terkait dengannya pada suatu waktu. Wadah yang termasuk dalam tugas yang sama juga dapat berkomunikasi melalui `localhost` antarmuka. Untuk informasi selengkapnya tentang VPC dan subnet, lihat VPC dan subnet di Panduan Pengguna Amazon VPC.

Untuk tugas di Fargate untuk menarik gambar kontainer, tugas harus memiliki rute ke internet. Berikut ini menjelaskan bagaimana Anda dapat memverifikasi bahwa tugas Anda memiliki rute ke internet.

- Saat menggunakan subnet publik, Anda dapat menetapkan alamat IP publik ke ENI tugas.
- Saat menggunakan subnet privat, subnet bisa memiliki lampiran gateway NAT.
- Saat menggunakan gambar kontainer yang di-host di Amazon ECR, Anda dapat mengonfigurasi Amazon ECR untuk menggunakan titik akhir VPC antarmuka dan penarikan gambar terjadi di atas alamat IPv4 pribadi tugas. Untuk informasi selengkapnya, lihat Antarmuka Amazon ECR VPC endpoint AWS PrivateLink() di Panduan Pengguna Amazon Elastic Container Registry.

Karena setiap tugas mendapatkan ENI sendiri, Anda dapat menggunakan fitur jaringan seperti VPC Flow Logs, yang dapat Anda gunakan untuk memantau lalu lintas ke dan dari tugas Anda. Untuk informasi selengkapnya, lihat Log Alur VPC di Panduan Pengguna Amazon VPC.

Anda juga bisa memanfaatkannya AWS PrivateLink. Anda dapat mengonfigurasi titik akhir antarmuka VPC sehingga Anda dapat mengakses Amazon ECS API melalui alamat IP pribadi. AWS PrivateLink membatasi semua lalu lintas jaringan antara VPC Anda dan Amazon ECS ke jaringan Amazon. Anda tidak memerlukan sebuah gateway internet, perangkat NAT, atau gateway privat virtual. Untuk informasi selengkapnya, lihat AWS PrivateLinkdi Panduan Praktik Terbaik Amazon ECS.

Untuk contoh cara menggunakan `NetworkConfiguration` sumber daya AWS CloudFormation, lihatthe section called “Membuat sumber daya Amazon ECS menggunakan tumpukan terpisah”.

ENI yang dibuat sepenuhnya dikelola oleh AWS Fargate. Selain itu, ada kebijakan IAM terkait yang digunakan untuk memberikan izin untuk Fargate. Untuk tugas yang menggunakan versi platform Fargate `1.4.0` atau yang lebih baru, tugas menerima ENI tunggal (disebut sebagai tugas ENI) dan semua lalu lintas jaringan mengalir melalui ENI tersebut dalam VPC Anda. Lalu lintas ini dicatat dalam log aliran VPC Anda. Untuk tugas yang menggunakan versi platform Fargate `1.3.0` dan sebelumnya, selain tugas ENI, tugas juga menerima ENI milik Fargate terpisah, yang digunakan untuk beberapa lalu lintas jaringan yang tidak terlihat di log aliran VPC. Tabel berikut menjelaskan perilaku lalu lintas jaringan dan kebijakan IAM yang diperlukan untuk setiap versi platform.

Tindakan	Arus lalu lintas dengan versi platform Linux **`1.3.0`** dan sebelumnya	Arus lalu lintas dengan versi platform Linux **`1.4.0`**	Arus lalu lintas dengan versi platform Windows **`1.0.0`**	Izin IAM
Mengambil kredensi login Amazon ECR	Fargate dimiliki ENI	ENI tugas	ENI tugas	Eksekusi tugas peran IAM
Tarikan citra	ENI tugas	ENI tugas	ENI tugas	Eksekusi tugas peran IAM
Mengirim log melalui driver log	ENI tugas	ENI tugas	ENI tugas	Eksekusi tugas peran IAM
Mengirim log melalui FireLens Amazon ECS	ENI tugas	ENI tugas	ENI tugas	Tugas peran IAM
Mengambil rahasia dari Secrets Manager atau Systems Manager	Fargate dimiliki ENI	ENI tugas	ENI tugas	Eksekusi tugas peran IAM

Tindakan	Arus lalu lintas dengan versi platform Linux **1.3.0** dan sebelumnya	Arus lalu lintas dengan versi platform Linux **1.4.0**	Arus lalu lintas dengan versi platform Windows **1.0.0**	Izin IAM
Lalu lintas sistem file Amazon EFS	Tidak tersedia	ENI tugas	ENI tugas	Tugas peran IAM
Lalu lintas aplikasi	ENI tugas	ENI tugas	ENI tugas	Tugas peran IAM

Pertimbangan jaringan tugas Fargate

Pertimbangkan hal berikut saat menggunakan jaringan tugas.

- Peran terkait layanan Amazon ECS diperlukan untuk memberi Amazon ECS izin untuk melakukan panggilan ke layanan lain AWS atas nama Anda. Peran ini dibuat untuk Anda ketika Anda membuat klaster atau jika Anda membuat atau memperbarui layanan di AWS Management Console. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS. Anda juga dapat membuat peran terkait layanan menggunakan perintah berikut AWS CLI .

  ```
  aws iam create-service-linked-role --aws-service-name ecs.amazonaws.com
  ```

- Amazon ECS mengisi nama host tugas dengan nama host DNS yang disediakan Amazon saat `enableDnsSupport` opsi `enableDnsHostnames` dan opsi diaktifkan di VPC Anda. Jika opsi ini tidak diaktifkan, nama host DNS tugas disetel ke nama host acak. Untuk informasi selengkapnya tentang pengaturan DNS untuk VPC, lihat Menggunakan DNS dengan VPC Anda di Panduan Pengguna Amazon VPC.
- Anda hanya dapat menentukan hingga 16 subnet dan 5 grup keamanan untuk`awsVpcConfiguration`. Untuk informasi selengkapnya, lihat AwsVpcConfigurationdi Referensi API Amazon Elastic Container Service.
- Anda tidak dapat secara manual melepaskan atau memodifikasi ENI yang dibuat dan dilampirkan oleh Fargate. Ini untuk mencegah penghapusan ENI yang tidak disengaja yang terkait dengan tugas yang sedang berjalan. Untuk melepaskan ENI untuk suatu tugas, hentikan tugas tersebut.
- Jika subnet VPC diperbarui untuk mengubah set opsi DHCP yang digunakannya, Anda juga tidak dapat menerapkan perubahan ini ke tugas yang ada yang menggunakan VPC. Mulai tugas baru,

yang akan menerima pengaturan baru untuk bermigrasi dengan lancar saat menguji perubahan baru dan kemudian menghentikan yang lama, jika tidak diperlukan rollback.

- Tugas yang diluncurkan dalam subnet dengan blok IPv6 CIDR hanya menerima alamat IPv6 saat menggunakan `1.4.0` versi platform Fargate atau yang lebih baru untuk Linux atau untuk Windows. `1.0.0`
- Untuk tugas yang menggunakan versi platform `1.4.0` atau yang lebih baru untuk Linux atau `1.0.0` Windows, tugas ENIS mendukung bingkai jumbo. Antarmuka jaringan dikonfigurasi dengan unit transmisi maksimum (MTU), yang merupakan ukuran muatan terbesar yang muat dalam satu frame. Semakin besar MTU, semakin banyak muatan aplikasi yang termuat dalam satu frame, yang mengurangi overhead per frame dan meningkatkan efisiensi. Mendukung bingkai jumbo mengurangi overhead saat jalur jaringan antara tugas Anda dan tujuan mendukung bingkai jumbo.
- Layanan dengan tugas yang menggunakan tipe peluncuran Fargate hanya mendukung Application Load Balancer dan Network Load Balancer. Classic Load Balancer tidak didukung. Saat Anda membuat grup target apa pun, Anda harus memilih `ip` sebagai jenis target, bukan`instance`. Untuk informasi selengkapnya, lihat Mendistribusikan lalu lintas layanan Amazon ECS menggunakan load balancing.

Menggunakan VPC dalam mode tumpukan dobel

Ketika menggunakan VPC dalam mode tumpukan ganda, tugas Anda dapat berkomunikasi melalui IPv4 atau IPv6, atau keduanya. Alamat IPv4 dan IPv6 independen satu sama lain dan Anda harus mengonfigurasi perutean dan keamanan di VPC Anda secara terpisah untuk IPv4 dan IPv6. Untuk informasi selengkapnya tentang mengonfigurasi VPC Anda untuk mode dual-stack, lihat Memigrasi ke IPv6 di Panduan Pengguna Amazon VPC.

Jika kondisi berikut terpenuhi, tugas Amazon ECS di Fargate diberi alamat IPv6:

- Setelan `dualStackIPv6` akun Amazon ECS Anda diaktifkan (`enabled`) untuk prinsipal IAM yang meluncurkan tugas Anda di Wilayah tempat Anda meluncurkan tugas. Pengaturan ini hanya dapat dimodifikasi menggunakan API atau AWS CLI. Anda memiliki opsi untuk mengaktifkan pengaturan ini untuk prinsipal IAM tertentu di akun Anda atau untuk seluruh akun Anda dengan menetapkan pengaturan default akun Anda. Untuk informasi selengkapnya, lihat Mengakses fitur Amazon ECS melalui pengaturan akun.
- VPC dan subnet Anda diaktifkan untuk IPv6. Untuk informasi selengkapnya tentang cara mengonfigurasi VPC Anda untuk mode dual-stack, lihat Memigrasi ke IPv6 di Panduan Pengguna Amazon VPC.

- Subnet Anda diaktifkan untuk menetapkan alamat IPv6 secara otomatis. Untuk informasi selengkapnya tentang cara mengonfigurasi subnet, lihat Memodifikasi atribut pengalamatan IPv6 untuk subnet Anda di Panduan Pengguna Amazon VPC.
- Tugas atau layanan menggunakan versi platform Fargate `1.4.0` atau yang lebih baru untuk Linux.

Jika Anda mengonfigurasi VPC Anda dengan gateway internet atau gateway internet khusus keluar, tugas Amazon ECS di Fargate yang diberi alamat IPv6 dapat mengakses internet. Gateway NAT tidak diperlukan. Untuk informasi selengkapnya, lihat gateway Internet dan gateway internet khusus eGress di Panduan Pengguna Amazon VPC.

Penyimpanan sementara tugas Fargate

Saat disediakan, setiap tugas Amazon ECS yang dihosting di kontainer Linux AWS Fargate menerima penyimpanan singkat berikut untuk pemasangan bind. Ini dapat dipasang dan dibagikan di antara kontainer yang menggunakan `volumesmountPoints`,, dan `volumesFrom` parameter dalam definisi tugas. Ini tidak didukung untuk kontainer Windows aktif AWS Fargate.

Versi platform wadah Fargate Linux

Versi 1.4.0 atau yang lebih baru

Secara default, tugas Amazon ECS yang di-host di Fargate menggunakan `1.4.0` versi platform atau yang lebih baru menerima minimal 20 GiB penyimpanan sementara. Jumlah total penyimpanan fana dapat ditingkatkan, hingga maksimum 200 GiB. Anda dapat melakukan ini dengan menentukan `ephemeralStorage` parameter dalam definisi tugas Anda.

Citra kontainer yang ditarik, dikompresi, dan tidak dikompresi untuk tugas disimpan di penyimpanan sementara. Untuk menentukan jumlah total penyimpanan sementara yang harus digunakan tugas Anda, Anda harus mengurangi jumlah penyimpanan yang digunakan citra kontainer Anda dari jumlah total penyimpanan sementara yang dialokasikan tugas Anda.

Untuk tugas yang menggunakan versi platform `1.4.0` atau yang lebih baru yang diluncurkan pada 28 Mei 2020 atau lebih baru, penyimpanan sementara dienkripsi dengan algoritme enkripsi AES-256. Algoritma ini menggunakan kunci enkripsi yang AWS dimiliki.

Untuk tugas yang menggunakan versi platform `1.4.0` atau yang lebih baru yang diluncurkan pada 18 November 2022 atau lebih baru, penggunaan penyimpanan sementara dilaporkan melalui titik akhir metadata tugas. Aplikasi Anda dalam tugas Anda dapat menanyakan titik akhir metadata tugas versi 4 untuk mendapatkan ukuran cadangan penyimpanan sementara dan jumlah yang digunakan.

Selain itu, ukuran cadangan penyimpanan sementara dan jumlah yang digunakan akan dikirim ke Amazon CloudWatch Wawasan Kontainer jika Anda mengaktifkan Wawasan Kontainer.

> **Note**
>
> Fargate mencadangkan ruang pada disk. Ruang ini hanya digunakan oleh Fargate. Anda tidak akan dikenai biaya untuk ruang tersebut. Ruang ini tidak akan ditampilkan dalam metrik ini. Namun demikian, Anda dapat melihat penyimpanan tambahan ini di alat lain seperti `df`.

Versi 1.3.0 atau sebelumnya

Untuk Amazon ECS pada tugas Fargate yang menggunakan `1.3.0` versi platform atau versi sebelumnya, setiap tugas menerima penyimpanan singkat berikut.

- Penyimpanan lapisan Docker 10 GB

  > **Note**
  >
  > Jumlah ini mencakup artifact citra kontainer terkompresi dan tidak terkompresi.

- Tambahan 4 GB untuk pemasangan volume. Ini dapat dipasang dan dibagikan di antara kontainer yang menggunakan `volumesmountPoints`,, dan `volumesFrom` parameter dalam definisi tugas.

Fargate versi platform kontainer Windows

Versi 1.0.0 atau yang lebih baru

Secara default, tugas Amazon ECS yang di-host di Fargate menggunakan `1.0.0` versi platform atau yang lebih baru menerima minimal 20 GiB penyimpanan sementara. Jumlah total penyimpanan fana dapat ditingkatkan, hingga maksimum 200 GiB. Anda dapat melakukan ini dengan menentukan `ephemeralStorage` parameter dalam definisi tugas Anda.

Citra kontainer yang ditarik, dikompresi, dan tidak dikompresi untuk tugas disimpan di penyimpanan sementara. Untuk menentukan jumlah total penyimpanan sementara yang harus digunakan tugas Anda, Anda harus mengurangi jumlah penyimpanan yang digunakan gambar kontainer Anda dari jumlah total penyimpanan sementara yang dialokasikan tugas Anda.

Untuk informasi selengkapnya, lihat Pemasangan terikat.

Menggunakan volume data dalam tugas

Amazon ECS mendukung opsi volume data berikut untuk kontainer:

- Volume Amazon EBS — Volume ini menyediakan penyimpanan blok berkinerja tinggi yang hemat biaya, tahan lama, dan berkinerja tinggi untuk beban kerja kontainer intensif data. Anda dapat mengonfigurasi paling banyak 1 volume EBS untuk lampiran ke tugas ECS mandiri saat menjalankan tugas. Atau, Anda dapat mengonfigurasi 1 volume EBS per tugas untuk lampiran ke setiap tugas yang diluncurkan melalui layanan ECS saat Anda membuat atau memperbarui layanan. Volume Amazon EBS didukung untuk tugas-tugas Linux yang di-host di instans Fargate atau Amazon EC2. Untuk informasi selengkapnya, lihat Volume Amazon EBS.
- Penyimpanan tugas fana Fargate - Secara default, tugas Amazon ECS yang di-host di Fargate menggunakan versi platform `1.4.0` atau yang lebih baru menerima penyimpanan singkat minimal 20 GiB. Jumlah total penyimpanan fana dapat ditingkatkan, hingga maksimum 200 GiB. Anda dapat melakukan ini dengan menentukan `ephemeralStorage` parameter dalam definisi tugas Anda.
- Volume Amazon Elastic File System (Amazon EFS) — Volume ini menyediakan penyimpanan file yang sederhana, dapat diskalakan, dan persisten untuk digunakan dengan tugas Amazon ECS Anda. Kapasitas penyimpanan Amazon EFS bersifat elastis. Itu tumbuh dan menyusut secara otomatis saat Anda menambah dan menghapus file. Aplikasi Anda dapat memiliki penyimpanan yang mereka butuhkan saat mereka membutuhkannya. Volume Amazon EFS didukung untuk tugas yang di-host di instans Fargate atau Amazon EC2. Untuk informasi selengkapnya, lihat Volume Amazon EFS.
- Volume Amazon FSx for Windows File Server - Volume ini menyediakan server file Windows yang dikelola sepenuhnya. Server file ini didukung oleh sistem file Windows. Saat menggunakan FSx for Windows File Server bersama dengan Amazon ECS, Anda dapat menyediakan tugas Windows Anda dengan penyimpanan file persisten, terdistribusi, bersama, dan statis. Untuk informasi selengkapnya, lihat Volume FSx for Windows File Server.

  Wadah Windows di Fargate tidak mendukung opsi ini.
- Volume Docker — Volume ini adalah volume yang dikelola Docker yang dibuat di bawah instans `/var/lib/docker/volumes` Amazon EC2 host. Driver volume Docker (juga disebut sebagai plugin) digunakan untuk mengintegrasikan volume dengan sistem penyimpanan eksternal, seperti Amazon EBS. Built-in driver volume `local` atau driver volume pihak ke tiga dapat digunakan. Volume Docker hanya didukung saat menjalankan tugas di instans Amazon EC2. Wadah Windows hanya mendukung penggunaan `local` driver. Untuk menggunakan volume Docker, tentukan

`dockerVolumeConfiguration` dalam ketentuan tugas Anda. Untuk informasi selengkapnya, lihat Volume Docker.

- Bind mount — Volume ini terdiri dari file atau direktori pada host, seperti instans Amazon EC2AWS Fargate atau, yang dipasang ke dalam wadah. Volume host mount bind didukung untuk tugas yang di-host di instans Fargate atau Amazon EC2. Volume host bind mount menggunakan penyimpanan singkat di Fargate. Jumlah penyimpanan fana berbeda pada berbagai versi platform Fargate. Anda dapat meminta hingga 200 gibibytes (GiB) penyimpanan sementara pada platform Fargate Linux versi 1.4.0 atau yang lebih baru. Untuk informasi selengkapnya, lihat Pemasangan terikat.

Volume Amazon EBS

Volume Amazon Elastic Block Store (Amazon EBS) menyediakan penyimpanan blok berkinerja tinggi yang sangat tersedia, hemat biaya, tahan lama, dan berkinerja tinggi untuk beban kerja intensif data. Volume Amazon EBS dapat digunakan dengan tugas Amazon ECS untuk throughput tinggi dan aplikasi intensif transaksi. Anda dapat mengonfigurasi paling banyak 1 volume EBS untuk lampiran ke tugas ECS mandiri saat menjalankan tugas. Atau, Anda dapat mengonfigurasi 1 volume EBS per tugas untuk lampiran ke setiap tugas yang diluncurkan melalui layanan ECS saat Anda membuat atau memperbarui layanan.

Volume Amazon EBS yang dilampirkan ke tugas Amazon ECS dikelola oleh ECS atas nama Anda, dan dapat dienkripsi dengan AWS Key Management Service (AWS KMS) kunci untuk melindungi data Anda. Anda dapat mengonfigurasi volume baru yang kosong untuk lampiran, atau Anda dapat menggunakan snapshot untuk memuat data dari volume yang ada.

Untuk memantau kinerja volume, Anda juga dapat menggunakan CloudWatch metrik Amazon EBS Amazon. Untuk informasi selengkapnya tentang metrik, lihat CloudWatch Metrik Amazon untuk Amazon EBS di Panduan Pengguna Amazon EC2.

Untuk informasi selengkapnya tentang volume Amazon EBS, lihat volume Amazon EBS di Panduan Pengguna Amazon EC2.

Wilayah AWS dan Availability Zone untuk volume Amazon EBS

Volume Amazon EBS dapat dilampirkan ke tugas Amazon ECS sebagai berikut: Wilayah AWS

Nama Wilayah	Kode Wilayah
AS Timur (Virginia Utara)	us-east-1

Nama Wilayah	Kode Wilayah
US East (Ohio)	us-east-2
AS Barat (California Utara)	us-west-1
US West (Oregon)	as-barat-2
Afrika (Cape Town)	af-selatan-1
Asia Pasifik (Hong Kong)	ap-east-1
Asia Pasifik (Hyderabad)	ap-south-2
Asia Pasifik (Jakarta)	ap-southeast-3
Asia Pasifik (Mumbai)	ap-south-1
Asia Pacific (Osaka)	ap-northeast-3
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
Canada (Central)	ca-sentral-1
Europe (Frankfurt)	eu-central-1
Europe (Ireland)	eu-west-1
Europe (London)	eu-barat-2
Eropa (Milan)	eu-selatan-1
Eropa (Paris)	eu-west-3
Eropa (Spanyol)	eu-south-2

Nama Wilayah	Kode Wilayah
Eropa (Stockholm)	eu-north-1
Europe (Zurich)	eu-central-2
Timur Tengah (Bahrain)	me-south-1
Timur Tengah (UEA)	me-central-1
Amerika Selatan (Sao Paulo)	sa-east-1

> **Important**
>
> Anda tidak dapat mengonfigurasi volume Amazon EBS untuk lampiran ke tugas Fargate Amazon ECS `euc1-az2` di `use1-az3` dan Availability Zones.

Pertimbangan volume Amazon EBS

Pertimbangkan hal berikut saat menggunakan volume Amazon EBS:

- Volume Amazon EBS hanya didukung untuk tugas-tugas Linux yang dihosting di Fargate, dan tugas tipe peluncuran EC2 yang dihosting Nitro pada instans Linux berbasis dengan Amazon ECS yang dioptimalkan Amazon Machine Images (AMI).
- Untuk tugas yang di-host di Fargate, volume Amazon EBS didukung pada versi platform `1.4.0` atau yang lebih baru (Linux). Untuk informasi selengkapnya, lihat Versi platform Fargate Linux.
- Untuk tugas yang di-host di instans Amazon EC2 Linux, volume Amazon EBS didukung pada AMI yang dioptimalkan ECS atau versi lebih baru. `20231219` Untuk informasi selengkapnya, lihat Mengambil metadata AMI yang dioptimalkan Amazon ECS.
- Jenis volume Amazon EBS magnetik (`standard`) tidak didukung untuk tugas yang dihosting di Fargate. Untuk informasi selengkapnya tentang jenis volume Amazon EBS, lihat Volume Amazon EBS di Panduan Pengguna Amazon EC2.
- Peran IAM infrastruktur Amazon ECS diperlukan saat membuat layanan atau tugas mandiri yang mengonfigurasi volume saat penerapan. Anda dapat melampirkan kebijakan `AmazonECSInfrastructureRolePolicyForVolumes` IAM AWS terkelola ke peran, atau Anda dapat menggunakan kebijakan terkelola sebagai panduan untuk membuat dan melampirkan

kebijakan Anda sendiri dengan izin yang memenuhi kebutuhan spesifik Anda. Untuk informasi selengkapnya, lihat Peran IAM infrastruktur Amazon ECS.

- Anda dapat melampirkan paling banyak satu volume Amazon EBS ke setiap tugas Amazon ECS, dan itu harus berupa volume baru. Anda tidak dapat melampirkan volume Amazon EBS yang ada ke tugas. Namun, Anda dapat mengonfigurasi volume Amazon EBS baru saat penerapan menggunakan snapshot dari volume yang ada.
- Anda dapat mengonfigurasi volume Amazon EBS saat penerapan hanya untuk layanan yang menggunakan jenis penerapan pembaruan bergulir dan strategi penjadwalan Replika.
- Amazon ECS secara otomatis menambahkan tag yang dicadangkan `AmazonECSCreated` dan`AmazonECSManaged`. Jika Anda menghapus tag ini dari volume, Amazon ECS tidak akan dapat mengelola volume atas nama Anda. Untuk informasi selengkapnya tentang menandai volume Amazon EBS, lihat Menandai volume Amazon EBS. Untuk informasi selengkapnya tentang menandai sumber daya Amazon ECS, lihat Menandai sumber daya Amazon ECS Anda.
- Penyediaan volume dari snapshot volume Amazon EBS yang berisi partisi tidak didukung.
- Volume yang dilampirkan ke tugas yang dikelola oleh layanan tidak dipertahankan dan selalu dihapus pada saat pemutusan tugas.
- Anda tidak dapat melampirkan volume Amazon EBS ke tugas tipe peluncuran EC2 yang diluncurkan pada instance container berdasarkan sistemXen. Untuk informasi selengkapnya tentang jenis instans, lihat Jenis instans di Panduan Pengguna Amazon EC2.
- Anda tidak dapat mengonfigurasi volume Amazon EBS untuk lampiran ke tugas Amazon ECS yang sedang berjalan. AWS Outposts

Enkripsi data untuk volume Amazon EBS

Anda dapat menggunakan AWS Key Management Service (AWS KMS) untuk membuat dan mengelola kunci kriptografi yang melindungi data Anda. Volume Amazon EBS dienkripsi saat istirahat dengan menggunakan. AWS KMS keys Jenis data berikut dienkripsi:

- Data disimpan saat istirahat pada volume
- Disk I/O
- Snapshot dibuat dari volume
- Volume baru dibuat dari snapshot

Anda dapat mengonfigurasi enkripsi Amazon EBS secara default sehingga semua volume baru yang dibuat dan dilampirkan ke tugas dienkripsi dengan menggunakan kunci KMS yang Anda

konfigurasikan untuk akun Anda. Untuk informasi selengkapnya tentang enkripsi dan enkripsi Amazon EBS secara default, lihat enkripsi Amazon EBS di Panduan Pengguna Amazon EC2.

Volume Amazon EBS yang dilampirkan ke tugas dapat dienkripsi dengan menggunakan default Kunci yang dikelola AWS dengan aliasalias/aws/ebs, atau kunci yang dikelola pelanggan simetris. Default Kunci yang dikelola AWS unik untuk masing-masing Akun AWS per Wilayah AWS dan dibuat secara otomatis. Untuk membuat kunci terkelola pelanggan simetris, ikuti langkah-langkah dalam Membuat kunci KMS enkripsi simetris di Panduan PengembangAWS KMS .

Kebijakan kunci KMS yang dikelola pelanggan

Untuk menggunakan kunci yang dikelola pelanggan untuk mengenkripsi volume EBS yang dikonfigurasi untuk lampiran ke tugas Anda, Anda harus mengonfigurasi kebijakan kunci KMS untuk memastikan bahwa peran IAM yang Anda gunakan untuk konfigurasi volume memiliki izin yang diperlukan untuk menggunakan kunci tersebut. Kebijakan utama harus menyertakan `kms:CreateGrant` dan `kms:GenerateDataKey*` izin. `kms:ReEncryptFrom`Izin `kms:ReEncryptTo` dan diperlukan untuk mengenkripsi volume yang dibuat menggunakan snapshot. Jika Anda ingin mengkonfigurasi dan mengenkripsi hanya volume baru yang kosong untuk lampiran, Anda dapat mengecualikan `kms:ReEncryptTo` dan `kms:ReEncryptFrom` izin.

Cuplikan JSON berikut menunjukkan contoh pernyataan kebijakan kunci yang harus dilampirkan ke kebijakan kunci KMS Anda untuk memungkinkan akses ECS menggunakan kunci untuk mengenkripsi volume EBS. Untuk menggunakan contoh pernyataan kebijakan, ganti *`user input placeholders`* dengan informasi Anda sendiri. Seperti biasa, hanya konfigurasikan izin yang Anda butuhkan.

```
{
      "Sid": "ReadOnlyPermissions"
      "Effect": "Allow",
      "Principal": { "AWS": "arn:aws:iam::111122223333:role/ecsInfrastructureRole" },
      "Action": "kms:DescribeKey",
      "Resource":"*"
    },
    {
      "Sid": "DataKeyGenerationForAmazonEBS"
      "Effect": "Allow",
      "Principal": { "AWS": "arn:aws:iam::111122223333:role/ecsInfrastructureRole" },
      "Action": [
      "kms:GenerateDataKey*",
      "kms:ReEncryptTo",
```

```
      "kms:ReEncryptFrom"
      ],
      "Resource":"*",
      "Condition": {
        "StringEquals": {
          "kms:CallerAccount": "aws_account_id",
          "kms:ViaService": "ec2.region.amazonaws.com"
        },
        "ForAnyValue:StringEquals": {
          "kms:EncryptionContextKeys": "aws:ebs:id"
        }
      }
    },
    {
      "Sid": "GrantCreationForAmazonEBS"
      "Effect": "Allow",
      "Principal": { "AWS": "arn:aws:iam::111122223333:role/ecsInfrastructureRole" },
      "Action": "kms:CreateGrant",
      "Resource":"*",
      "Condition": {
        "StringEquals": {
          "kms:CallerAccount": "aws_account_id",
          "kms:ViaService": "ec2.region.amazonaws.com"
        },
        "ForAnyValue:StringEquals": {
          "kms:EncryptionContextKeys": "aws:ebs:id"
        },
        "Bool": {
          "kms:GrantIsForAWSResource": true
        }
      }
    }
```

Untuk informasi selengkapnya tentang kebijakan dan izin utama, lihat Kebijakan kunci AWS KMS dan AWS KMS izin di Panduan AWS KMS Pengembang. Untuk mengatasi masalah yang terkait dengan izin kunci, lihat Memecahkan masalah lampiran volume Amazon EBS.

Mengonfigurasi volume Amazon EBS saat penerapan

Untuk mengonfigurasi volume Amazon EBS untuk lampiran ke tugas Anda, Anda harus menentukan konfigurasi titik pemasangan dalam definisi tugas, beri nama volume, dan atur `configuredAtLaunch` ke`true`. Cuplikan JSON definisi tugas berikut menunjukkan sintaks untuk `mountPoints` dan `volumes` objek dalam definisi tugas. Untuk informasi selengkapnya tentang

parameter definisi tugas, lihatParameter ketentuan tugas. Untuk menggunakan contoh ini, ganti *user input placeholders* dengan informasi Anda sendiri.

```
{
    "family": "mytaskdef",
    "containerDefinitions": [
        {
            "name": "container-using-ebs",
            "image": "amazonlinux:2",
            "entryPoint": [
                "sh",
                "-c"
            ],
            "command": [
                "ls -la /mount/ebs"
            ],
            "mountPoints": [
                {
                    "sourceVolume": "myEbsVolume",
                    "containerPath": "/mount/ebs",
                    "readOnly": true
                }
            ]
        }
    ],
    "volumes": [
        {
            "name": "myEBSVolume",
            "configuredAtLaunch": true
        }
    ]
}
```

Untuk mendaftarkan definisi tugas dengan menggunakan AWS Command Line Interface (AWS CLI), simpan template sebagai file JSON, lalu gunakan perintah berikut. Untuk menggunakan contoh perintah ini, ganti *user input placeholders* dengan informasi Anda sendiri.

```
aws ecs register-task-definition \
    --cli-input-json file://path_to_json_file/task-definition.json
```

Untuk membuat dan mendaftarkan definisi tugas menggunakan AWS Management Console, lihatMembuat definisi tugas menggunakan konsol.

Setelah mendaftarkan definisi tugas, Anda dapat mengonfigurasi volume saat penerapan menggunakan AWS Management Console, Amazon ECS API, atau dengan meneruskan file JSON input dengan perintah berikut: AWS CLI

- `run-task`untuk menjalankan tugas ECS mandiri.
- `start-task`untuk menjalankan tugas ECS mandiri dalam instance kontainer tertentu. Perintah ini tidak berlaku untuk tugas jenis peluncuran Fargate.
- `create-service`untuk membuat layanan ECS baru.
- `update-service`untuk memperbarui layanan yang ada.

Mengonfigurasi volume saat penerapan memungkinkan Anda membuat definisi tugas yang tidak dibatasi untuk jenis volume atau pengaturan volume EBS tertentu. Anda kemudian dapat menggunakan kembali definisi tugas Anda di lingkungan eksekusi yang berbeda. Misalnya, Anda dapat memberikan lebih banyak throughput untuk beban kerja produksi Anda daripada lingkungan pra-prod Anda.

Cuplikan JSON berikut menunjukkan semua parameter volume Amazon EBS yang dapat dikonfigurasi saat penerapan. Untuk menggunakan parameter ini untuk konfigurasi volume, ganti *`user input placeholders`* dengan informasi Anda sendiri. Untuk informasi selengkapnya tentang parameter ini, lihat Konfigurasi volume.

```
"volumeConfigurations": [
        {
            "name": "ebs-volume",
            "managedEBSVolume": {
                "encrypted": true,
                "kmsKeyId": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
                "volumeType": "gp3",
                "sizeInGiB": 10,
                "snapshotId": "snap-12345",
                "iops": 3000,
                "throughput": 125,
                "tagSpecifications": [
                    {
                        "resourceType": "volume",
                        "tags": [
                            {
                                "key": "key1",
                                "value": "value1"
```

```
                            }
                        ],
                        "propagateTags": "NONE"
                    }
                ],
                "roleArn": "arn:aws::iam:1111222333:role/ecsInfrastructureRole",
                 "terminationPolicy": {
                    "deleteOnTermination": true//can't be configured for service-
managed tasks, always true
                },
                "filesystemType": "ext4"
            }
        }
    ]
```

> **Note**
>
> Agar wadah dalam tugas Anda dapat menulis ke volume Amazon EBS yang dipasang, Anda harus menjalankan penampung sebagai pengguna root.

Untuk informasi tentang cara mendiagnosis dan memperbaiki masalah yang mungkin Anda temui saat mengonfigurasi volume EBS untuk lampiran ke tugas, lihat Memecahkan masalah lampiran volume Amazon EBS. Untuk informasi tentang peran infrastruktur Amazon ECS AWS Identity and Access Management (IAM) yang diperlukan untuk lampiran volume EBS, lihat. Peran IAM infrastruktur Amazon ECS

Mengkonfigurasi volume untuk tugas mandiri

Cuplikan berikut menunjukkan sintaks untuk mengonfigurasi volume Amazon EBS untuk lampiran ke tugas mandiri. Cuplikan JSON berikut menunjukkan sintaks untuk mengkonfigurasi`volumeType`,,, `sizeInGiB` dan pengaturan. `encrypted` `kmsKeyId` Simpan cuplikan berikut sebagai file JSON. Untuk menggunakan contoh ini, ganti *`user input placeholders`* dengan informasi Anda sendiri.

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "volumeConfigurations": [
        {
            "name": "datadir",
            "managedEBSVolume": {
```

```
                "volumeType": "gp3",
                "sizeInGiB": 100,
                "roleArn":"arn:aws::iam:1111222333:role/ecsInfrastructureRole",
                "encrypted": true,
                "kmsKeyId":
 "arn:aws:kms:region:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
            }
        }
   ]
}
```

Perintah berikut dapat digunakan untuk menjalankan tugas mandiri menggunakan file input JSON. Konfigurasi yang ditentukan dalam file JSON digunakan untuk membuat dan melampirkan volume EBS ke tugas mandiri. Untuk menggunakan contoh ini, ganti *`user input placeholders`* dengan informasi Anda sendiri.

```
aws ecs run-task \
    --cli-input-json file://path_to_json_file/task.json
```

Untuk mengonfigurasi volume lampiran ke tugas mandiri menggunakan AWS Management Console, lihatJalankan aplikasi sebagai tugas Amazon ECS.

Mengkonfigurasi volume pada pembuatan layanan

Cuplikan berikut menunjukkan sintaks untuk mengonfigurasi volume Amazon EBS untuk lampiran ke tugas yang dikelola oleh layanan. Volume bersumber dari snapshot dengan menggunakan file. `snapshotId` Simpan cuplikan berikut sebagai file JSON. Untuk menggunakan contoh ini, ganti *`user input placeholders`* dengan informasi Anda sendiri.

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "serviceName": "mysvc",
   "desiredCount": 2,
   "volumeConfigurations": [
        {
            "name": "myEbsVolume",
            "managedEBSVolume": {
              "roleArn":"arn:aws::iam:1111222333:role/ecsInfrastructureRole",
              "snapshotId": "snap-12345"
            }
```

```
        }
    ]
}
```

Anda dapat menggunakan perintah berikut untuk membuat layanan dengan menggunakan file input JSON. Konfigurasi yang ditentukan dalam file JSON digunakan untuk membuat dan melampirkan volume EBS ke setiap tugas yang dikelola oleh layanan. Untuk menggunakan contoh perintah ini, ganti *user input placeholders* dengan informasi Anda sendiri.

```
aws ecs create-service \
    --cluster mycluster \
    --service-name mysvc \
    --cli-input-json file://path_to_json_file/create-service.json
```

Untuk mengonfigurasi volume lampiran ke tugas yang dikelola layanan selama pembuatan layanan menggunakan AWS Management Console, lihat. Membuat layanan menggunakan konsol

Mengkonfigurasi volume pada pembaruan layanan

Cuplikan JSON berikut menunjukkan sintaks untuk memperbarui layanan yang sebelumnya tidak memiliki volume Amazon EBS yang dikonfigurasi untuk lampiran ke tugas. Anda harus memberikan ARN definisi tugas dengan `configuredAtLaunch` set to. `true` Cuplikan JSON berikut menunjukkan sintaks untuk mengonfigurasi,, dan `volumeType` `sizeInGiBthroughput`, `iops` dan setelan volume Amazon EBS `filesystemType` yang dikonfigurasi untuk lampiran ke tugas yang dikelola oleh layanan. Satu volume dilampirkan ke setiap tugas dalam layanan. Untuk menggunakan contoh ini, ganti *user input placeholders* dengan informasi Anda sendiri.

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "serviceName": "mysvc",
   "desiredCount": 2,
  "volumeConfigurations": [
        {
            "name": "myEbsVolume",
            "managedEBSVolume": {
              "roleArn":"arn:aws::iam:1111222333:role/ecsInfrastructureRole",
               "volumeType": "gp3",
                "sizeInGiB": 100,
                 "iops": 3000,
```

```
                "throughput": 125,
                "filesystemType": "ext4"
            }
        }
    ]
}
```

Cuplikan JSON berikut menunjukkan sintaks untuk memperbarui layanan agar tidak lagi menggunakan volume Amazon EBS. Anda harus memberikan ARN definisi tugas dengan `configuredAtLaunch` set to`false`, atau definisi tugas tanpa parameter. `configuredAtLaunch` Anda juga harus menyediakan `volumeConfigurations` objek kosong. Untuk menggunakan contoh ini, ganti *`user input placeholders`* dengan informasi Anda sendiri.

```
{
    "cluster": "mycluster",
    "taskDefinition": "mytaskdef",
    "serviceName": "mysvc",
    "desiredCount": 2,
    "volumeConfigurations": []
}
```

Anda dapat menggunakan perintah berikut untuk memperbarui layanan dengan menggunakan file input JSON. Untuk menggunakan contoh perintah ini, ganti *`user input placeholders`* dengan informasi Anda sendiri.

```
aws ecs update-service \
    --cli-input-json file://path_to_json_file/update-service.json
```

Untuk mengonfigurasi volume lampiran ke tugas yang dikelola layanan selama pembaruan layanan menggunakan AWS Management Console, lihat. Memperbarui layanan menggunakan konsol

Kebijakan penghentian volume Amazon EBS

Saat tugas Amazon ECS dihentikan, Amazon ECS menggunakan `deleteOnTermination` nilainya untuk menentukan apakah volume Amazon EBS yang terkait dengan tugas yang dihentikan harus dihapus. Secara default, volume EBS yang dilampirkan ke tugas dihapus saat tugas dihentikan. Untuk tugas mandiri, Anda dapat mengubah setelan ini untuk mempertahankan volume saat penghentian tugas.

> **Note**
>
> Volume yang dilampirkan ke tugas yang dikelola oleh layanan tidak dipertahankan dan selalu dihapus pada saat pemutusan tugas.

Menandai volume Amazon EBS

Anda dapat menandai volume Amazon EBS dengan menggunakan `tagSpecifications` objek. Menggunakan objek, Anda dapat memberikan tag Anda sendiri dan mengatur propagasi tag dari definisi tugas atau layanan, tergantung pada apakah volume dilampirkan ke tugas mandiri atau tugas dalam layanan. Amazon ECS secara otomatis melampirkan tag `AmazonECSCreated` dan `AmazonECSManaged` cadangan ke volume Amazon EBS. Tambahan 48 tag yang ditentukan pengguna, dikelola ECS, dan disebarkan dapat ditambahkan ke volume dengan total 50 tag maksimum per volume.

Jika Anda ingin menambahkan tag Amazon ECS yang dikelola ke volume Anda, Anda harus menyetel `enableECSManagedTags` ke `true` dalam`UpdateService`,`CreateService`, `RunTask` atau `StartTask` panggilan. Jika Anda mengaktifkan tag yang dikelola Amazon ECS, Amazon ECS akan menandai volume secara otomatis dengan informasi kluster dan layanan (`aws:ecs:`*`clusterName`*dan). `aws:ecs:`*`serviceName`* Untuk informasi selengkapnya tentang menandai sumber daya Amazon ECS, lihat Menandai sumber daya Amazon ECS Anda.

Cuplikan JSON berikut menunjukkan sintaks untuk menandai setiap volume Amazon EBS yang dilampirkan ke setiap tugas dalam layanan dengan tag yang ditentukan pengguna dan tag yang dikelola ECS. Untuk menggunakan contoh ini untuk membuat layanan, ganti *`user input placeholders`* dengan informasi Anda sendiri.

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "serviceName": "mysvc",
   "desiredCount": 2,
   "enableECSManagedTags": true,
   "volumeConfigurations": [
        {
            "name": "datadir",
            "managedEBSVolume": {
                "volumeType": "gp3",
                "sizeInGiB": 100,
```

```
                "tagSpecifications": [
                   {
                       "resourceType": "volume",
                       "tags": [
                           {
                               "key": "key1",
                               "value": "value1"
                           }
                       ],
                       "propagateTags": "NONE"
                   }
               ]
               "roleArn":"arn:aws::iam:1111222333:role/ecsInfrastructureRole",
               "encrypted": true,
               "kmsKeyId":
 "arn:aws:kms:region:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
           }
       }
   ]
}
```

> **Note**
>
> Anda dapat menentukan jenis `volume` sumber daya untuk menandai volume Amazon EBS.

Kinerja volume Amazon EBS untuk tugas sesuai permintaan Fargate

Volume Amazon EBS dasar IOPS dan throughput yang tersedia untuk tugas sesuai permintaan Fargate bergantung pada total unit CPU yang Anda minta untuk tugas tersebut. Jika Anda meminta 0,25, 0,5, atau 1 unit CPU virtual (vCPU) untuk tugas Fargate Anda, kami sarankan Anda mengonfigurasi volume SSD Tujuan Umum `gp2` (`gp3`atau) atau volume Hard Disk Drive (HDD) (atau). `st1` `sc1` Jika Anda meminta lebih dari 1 vCPU untuk tugas Fargate Anda, batas kinerja dasar berikut berlaku untuk volume Amazon EBS yang dilampirkan ke tugas. Anda mungkin untuk sementara mendapatkan kinerja EBS yang lebih tinggi daripada batas berikut. Namun, kami menyarankan Anda merencanakan beban kerja Anda berdasarkan batas-batas ini.

Unit CPU yang diminta (dalam vCPU)	Dasar Amazon EBS IOPS (16 KiB I/O)	Throughput Amazon EBS Dasar (dalam, MiBps 128 KiB I/O)	Bandwidth dasar (dalam Mbps)
2	3.000	75	360
4	5.000	120	1.150
8	10.000	250	2.300
16	15.000	500	4,500

> **Note**
>
> Saat Anda mengonfigurasi volume Amazon EBS untuk lampiran ke tugas Fargate, batas kinerja Amazon EBS untuk tugas Fargate dibagi antara penyimpanan sementara tugas dan volume terlampir.

Volume Amazon EFS

Amazon Elastic File System (Amazon EFS) menyediakan penyimpanan file yang sederhana dan dapat diskalakan untuk digunakan dengan tugas Amazon ECS Anda. Kapasitas penyimpanan Amazon EFS bersifat elastis. Itu tumbuh dan menyusut secara otomatis saat Anda menambah dan menghapus file. Aplikasi Anda dapat memiliki penyimpanan yang mereka butuhkan dan kapan mereka membutuhkannya.

Anda dapat menggunakan sistem file Amazon EFS dengan Amazon ECS untuk mengekspor data sistem file di seluruh armada instans kontainer Anda. Dengan begitu, tugas Anda memiliki akses ke penyimpanan tetap yang sama, terlepas dari instans tempat mereka mendarat. Definisi tugas Anda harus mereferensikan pemasangan volume pada instance container untuk menggunakan sistem file. Bagian berikut menjelaskan cara memulai menggunakan Amazon EFS dengan Amazon ECS.

Untuk tutorial, lihat Menggunakan sistem file Amazon EFS dengan Amazon ECS menggunakan konsol.

Pertimbangan volume Amazon EFS

Pertimbangkan hal berikut saat menggunakan volume Amazon EFS:

- Untuk tugas yang menggunakan tipe peluncuran EC2, dukungan sistem file Amazon EFS ditambahkan sebagai pratinjau publik dengan versi AMI Amazon ECS yang dioptimalkan dengan agen kontainer `20191212` versi 1.35.0. Namun, dukungan sistem file Amazon EFS memasuki ketersediaan umum dengan versi AMI Amazon ECS yang dioptimalkan `20200319` dengan agen kontainer versi 1.38.0, yang berisi jalur akses Amazon EFS dan fitur otorisasi IAM. Kami menyarankan Anda menggunakan versi AMI Amazon ECS yang dioptimalkan `20200319` atau yang lebih baru untuk menggunakan fitur-fitur ini. Untuk informasi selengkapnya, lihat AMI Amazon ECS yang dioptimalkan.

  > **Note**
  >
  > Jika Anda membuat AMI sendiri, Anda harus menggunakan agen kontainer 1.38.0 atau yang lebih baru, `ecs-init` versi 1.38.0-1 atau yang lebih baru, dan jalankan perintah berikut di instans Amazon EC2 Anda untuk mengaktifkan plugin volume Amazon ECS. Perintah akan tergantung pada apakah Anda menggunakan Amazon Linux 2 atau Amazon Linux sebagai citra dasar Anda.
  >
  > Amazon Linux 2
  >
  > ```
  > yum install amazon-efs-utils
  > systemctl enable --now amazon-ecs-volume-plugin
  > ```
  >
  > Amazon Linux
  >
  > ```
  > yum install amazon-efs-utils
  > sudo shutdown -r now
  > ```

- Untuk tugas yang di-host di Fargate, sistem file Amazon EFS didukung pada platform versi 1.4.0 atau yang lebih baru (Linux). Untuk informasi selengkapnya, lihat Versi platform Fargate Linux.
- Saat menggunakan volume Amazon EFS untuk tugas yang di-host di Fargate, Fargate membuat wadah pengawas yang bertanggung jawab untuk mengelola volume Amazon EFS. Kontainer supervisor menggunakan sejumlah kecil memori tugas. Kontainer pengawas akan terlihat saat dilakukan kueri titik akhir metadata tugas versi 4. Selain itu, ini terlihat di CloudWatch Container Insights sebagai nama `aws-fargate-supervisor` kontainer. Untuk informasi selengkapnya saat menggunakan jenis peluncuran Amazon EC2, lihat. Titik akhir metadata tugas Amazon ECS versi 4 Untuk informasi selengkapnya saat menggunakan jenis peluncuran Fargate, lihat. Titik akhir metadata tugas Amazon ECS versi 4 untuk tugas di Fargate

- Menggunakan volume Amazon EFS atau menentukan `EFSVolumeConfiguration` tidak didukung pada instans eksternal.
- Kami menyarankan Anda mengatur `ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION` parameter dalam file konfigurasi agen ke nilai yang kurang dari default (sekitar 1 jam). Perubahan ini membantu mencegah kedaluwarsa kredensi pemasangan EFS dan memungkinkan pembersihan dudukan yang tidak digunakan. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.

Menggunakan titik akses Amazon EFS

Titik akses Amazon EFS adalah titik masuk khusus aplikasi ke dalam sistem file EFS untuk mengelola akses aplikasi ke kumpulan data bersama. Untuk informasi lebih lanjut tentang titik akses Amazon EFS dan cara mengontrol akses ke sana, lihat Bekerja dengan Titik Akses Amazon EFS dalam Panduan Pengguna Amazon Elastic File System.

Titik akses dapat menerapkan identitas pengguna, termasuk grup POSIX pengguna, pada semua permintaan sistem file yang dibuat melalui titik akses. Titik akses juga dapat menerapkan direktori root yang berbeda untuk sistem file. Ini agar klien hanya dapat mengakses data di direktori yang ditentukan atau subdirektorinya.

> **Note**
>
> Saat membuat titik akses EFS, tentukan jalur pada sistem file untuk berfungsi sebagai direktori root. Saat mereferensikan sistem file EFS dengan ID titik akses dalam definisi tugas Amazon ECS Anda, direktori root harus dihilangkan atau disetel ke`/`, yang memberlakukan jalur yang disetel pada titik akses EFS.

Anda dapat menggunakan peran IAM tugas Amazon ECS untuk menegakkan bahwa aplikasi tertentu menggunakan titik akses tertentu. Dengan menggabungkan kebijakan IAM dengan titik akses, Anda dapat memberikan akses aman ke kumpulan data tertentu untuk aplikasi Anda. Untuk informasi selengkapnya tentang cara menggunakan peran IAM tugas, lihatTugas peran IAM.

Menentukan sistem file Amazon EFS dalam definisi tugas Anda

Untuk menggunakan volume sistem file Amazon EFS untuk container Anda, Anda harus menentukan konfigurasi volume dan titik mount dalam definisi tugas Anda. Cuplikan JSON definisi tugas berikut menunjukkan sintaks untuk `volumes` dan `mountPoints` objek untuk wadah.

```
{
    "containerDefinitions": [
        {
            "name": "container-using-efs",
            "image": "amazonlinux:2",
            "entryPoint": [
                "sh",
                "-c"
            ],
            "command": [
                "ls -la /mount/efs"
            ],
            "mountPoints": [
                {
                    "sourceVolume": "myEfsVolume",
                    "containerPath": "/mount/efs",
                    "readOnly": true
                }
            ]
        }
    ],
    "volumes": [
        {
            "name": "myEfsVolume",
            "efsVolumeConfiguration": {
                "fileSystemId": "fs-1234",
                "rootDirectory": "/path/to/my/data",
                "transitEncryption": "ENABLED",
                "transitEncryptionPort": integer,
                "authorizationConfig": {
                    "accessPointId": "fsap-1234",
                    "iam": "ENABLED"
                }
            }
        }
    ]
}
```

`efsVolumeConfiguration`

Tipe: Objek

Diperlukan: Tidak

Parameter ini ditentukan saat menggunakan volume Amazon EFS.

`fileSystemId`

Tipe: String

Diperlukan: Ya

ID sistem file Amazon EFS yang akan digunakan.

`rootDirectory`

Tipe: String

Wajib: Tidak

Direktori dalam sistem file Amazon EFS untuk pemasangan sebagai direktori asal di dalam host. Jika parameter ini diabaikan, asal volume Amazon EFS akan digunakan. Menentukan `/` memiliki efek yang sama seperti mengabaikan parameter ini.

> **Important**
>
> Jika titik akses EFS ditentukan dalam `authorizationConfig`, parameter direktori root harus dihilangkan atau diatur ke `/`, yang memberlakukan jalur yang diatur pada titik akses EFS.

`transitEncryption`

Jenis: String

Nilai yang valid: `ENABLED` | `DISABLED`

Diperlukan: Tidak

Menentukan apakah akan mengaktifkan enkripsi untuk data Amazon EFS dalam perjalanan antara host Amazon ECS dan server Amazon EFS. Jika otorisasi Amazon EFS IAM digunakan, enkripsi transit harus diaktifkan. Jika parameter ini diabaikan, nilai default `DISABLED` akan digunakan. Untuk informasi lebih lanjut, lihat Pengenkripsian Data Saat Transit di Panduan Pengguna Amazon Elastic File System.

`transitEncryptionPort`

Tipe: Integer

Wajib: Tidak

Port yang akan digunakan saat mengirim data terenkripsi antara host Amazon ECS dan server Amazon EFS. Jika Anda tidak menentukan port enkripsi transit, strategi pemilihan port yang digunakan oleh pembantu pemasangan Amazon EFS akan digunakan. Untuk informasi lebih lanjut, lihat Pembantu Pemasangan EFS dalam Panduan Pengguna Amazon Elastic File System.

`authorizationConfig`

Tipe: Objek

Diperlukan: Tidak

Detail konfigurasi otorisasi untuk sistem file Amazon EFS.

`accessPointId`

Tipe: String

Wajib: Tidak

ID titik akses yang akan digunakan. Jika titik akses ditentukan, nilai direktori root di dalam `efsVolumeConfiguration` harus dihilangkan atau disetel ke`/`, yang memberlakukan jalur yang ditetapkan pada titik akses EFS. Jika titik akses digunakan, enkripsi transit harus diaktifkan di `EFSVolumeConfiguration`. Untuk informasi lebih lanjut, lihat Bekerja dengan Titik Akses Amazon EFS dalam Panduan Pengguna Amazon Elastic File System.

`iam`

Tipe: String

Nilai yang valid: `ENABLED` | `DISABLED`

Diperlukan: Tidak

Menentukan apakah akan menggunakan peran IAM tugas Amazon ECS yang ditentukan dalam definisi tugas saat memasang sistem file Amazon EFS. Jika diaktifkan, enkripsi transit harus diaktifkan di `EFSVolumeConfiguration`. Jika Anda menghilangkan parameter ini, nilai default `DISABLED` akan digunakan. Untuk informasi selengkapnya, lihat IAM Role pada Tugas.

Volume FSx for Windows File Server

FSx for Windows File Server menyediakan server file Windows yang dikelola sepenuhnya, yang didukung oleh sistem file Windows. Saat menggunakan FSx for Windows File Server bersama dengan ECS, Anda dapat menyediakan tugas Windows Anda dengan penyimpanan file statis yang persisten, terdistribusi, bersama, dan statis. Untuk informasi selengkapnya, lihat Apa itu FSx for Windows File Server? .

> **Note**
>
> Instans EC2 yang menggunakan AMI Penuh Windows Server 2016 yang dioptimalkan Amazon ECS tidak mendukung volume tugas FSx for Windows File Server ECS.
> Anda tidak dapat menggunakan volume FSx for Windows File Server dalam wadah Windows pada konfigurasi Fargate.

Anda dapat menggunakan fsX for Windows File Server untuk menyebarkan beban kerja Windows yang memerlukan akses ke penyimpanan eksternal bersama, penyimpanan Regional yang sangat tersedia, atau penyimpanan throughput tinggi. Anda dapat memasang satu atau beberapa volume sistem file FSx for Windows File Server ke wadah Amazon ECS yang berjalan pada instans Amazon ECS Windows. Anda dapat membagikan volume sistem file FSx for Windows File Server antara beberapa wadah Amazon ECS dalam satu tugas Amazon ECS.

Untuk mengaktifkan penggunaan FSx for Windows File Server dengan ECS, sertakan ID sistem file FSx for Windows File Server dan informasi terkait dalam definisi tugas. Ini dalam contoh cuplikan JSON definisi tugas berikut. Sebelum Anda membuat dan menjalankan definisi tugas, Anda memerlukan yang berikut ini.

- Instans ECS Windows EC2 yang bergabung ke domain yang valid. Ini dapat di-host oleh Active Directory lokal atau Direktori Aktif yang dihosting sendiri di Amazon EC2. AWS Directory Service for Microsoft Active Directory
- Parameter AWS Secrets Manager rahasia atau Systems Manager yang berisi kredensyal yang digunakan untuk bergabung dengan domain Active Directory dan melampirkan sistem file FSx for Windows File Server. Nilai kredensial adalah kredensial nama dan kata sandi yang Anda masukkan saat membuat Direktori Aktif.

Bagian berikut menjelaskan cara memulai menggunakan FSx for Windows File Server dengan Amazon ECS.

Untuk tutorial terkait, lihatMenggunakan sistem file FSx for Windows File Server dengan Amazon ECS.

FSx for Windows File Server pertimbangan volume

Pertimbangkan hal berikut saat menggunakan volume FSx for Windows File Server:

- FSx for Windows File Server dengan Amazon ECS hanya mendukung instans Windows Amazon EC2. Instans Linux Amazon EC2 tidak didukung.
- FSx for Windows File Server dengan Amazon ECS tidak mendukung. AWS Fargate
- FSx for Windows File Server dengan Amazon ECS `awsvpc` dengan mode jaringan memerlukan `1.54.0` versi atau yang lebih baru dari agen penampung.
- Jumlah maksimum huruf drive yang dapat digunakan untuk tugas Amazon ECS adalah 23. Setiap tugas dengan volume FSx for Windows File Server mendapat huruf drive yang ditetapkan untuk itu.
- Secara default, waktu pembersihan sumber daya tugas adalah tiga jam setelah tugas berakhir. Bahkan jika tidak ada tugas yang menggunakannya, pemetaan file yang dibuat oleh tugas tetap ada selama tiga jam. Waktu pembersihan default dapat dikonfigurasi dengan menggunakan variabel lingkungan Amazon ECS. `ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION` Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.
- Tugas biasanya hanya berjalan di VPC yang sama dengan sistem file FSx for Windows File Server. Namun, dimungkinkan untuk memiliki dukungan lintas-VPC jika ada konektivitas jaringan yang mapan antara VPC cluster Amazon ECS dan sistem file fsX for Windows File Server melalui peering VPC.
- Anda mengontrol akses ke sistem file FSx for Windows File Server di tingkat jaringan dengan mengonfigurasi grup keamanan VPC. Hanya tugas yang di-host pada instans EC2 yang bergabung ke domain Direktori Aktif dengan grup keamanan Direktori Aktif yang dikonfigurasi dengan benar yang dapat mengakses berbagi file FSx for Windows File Server. Jika grup keamanan salah dikonfigurasi, ECS gagal meluncurkan tugas dengan pesan galat berikut: `unable to mount file system` *`fs-id`*
- FSx for Windows File Server terintegrasi AWS Identity and Access Management dengan (IAM) untuk mengontrol tindakan yang dapat dilakukan pengguna dan grup IAM Anda pada sumber daya FSx for Windows File Server tertentu. Dengan otorisasi klien, pelanggan dapat menentukan peran IAM yang memungkinkan atau menolak akses ke sistem file FSx for Windows File Server tertentu, secara opsional memerlukan akses hanya-baca, dan secara opsional mengizinkan atau melarang akses root ke sistem file dari klien. Untuk informasi selengkapnya, lihat Keamanan di Panduan Pengguna Amazon FSx Windows.

Menentukan sistem file FSx for Windows File Server dalam definisi tugas Anda

Untuk menggunakan volume sistem file FSx for Windows File Server untuk container Anda, tentukan konfigurasi volume dan mount point dalam definisi tugas Anda. Cuplikan JSON definisi tugas berikut menunjukkan sintaks untuk `volumes` dan `mountPoints` objek untuk wadah.

```
{
    "containerDefinitions": [
        {
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "portMappings": [],
            "command": ["New-Item -Path C:\\fsx-windows-dir\\index.html -ItemType file
 -Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
align:center> <h1>Amazon ECS Sample App</h1> <h2>It Works!</h2> <p>You are using Amazon
 FSx for Windows File Server file system for persistent container storage.</p>' -
Force"],
            "cpu": 512,
            "memory": 256,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "essential": false,
            "name": "container1",
            "mountPoints": [
                {
                    "sourceVolume": "fsx-windows-dir",
                    "containerPath": "C:\\fsx-windows-dir",
                    "readOnly": false
                }
            ]
        },
        {
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "portMappings": [
                {
                    "hostPort": 443,
                    "protocol": "tcp",
```

```
                    "containerPort": 80
                }
            ],
            "command": ["Remove-Item -Recurse C:\\inetpub\\wwwroot\\* -Force; Start-
Sleep -Seconds 120; Move-Item -Path C:\\fsx-windows-dir\\index.html -Destination C:\
\inetpub\\wwwroot\\index.html -Force; C:\\ServiceMonitor.exe w3svc"],
            "mountPoints": [
                {
                    "sourceVolume": "fsx-windows-dir",
                    "containerPath": "C:\\fsx-windows-dir",
                    "readOnly": false
                }
            ],
            "cpu": 512,
            "memory": 256,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "essential": true,
            "name": "container2"
        }
    ],
    "family": "fsx-windows",
    "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
    "volumes": [
        {
            "name": "fsx-windows-dir",
            "fsxWindowsFileServerVolumeConfiguration": {
                "fileSystemId": "fs-0eeb5730b2EXAMPLE",
                "authorizationConfig": {
                    "domain": "example.com",
                    "credentialsParameter": "arn:arn-1234"
                },
                "rootDirectory": "share"
            }
        }
    ]
}
```

FSxWindowsFileServerVolumeConfiguration

Tipe: Objek

Diperlukan: Tidak

Parameter ini ditentukan saat Anda menggunakan sistem file FSx for Windows File Server untuk penyimpanan tugas.

`fileSystemId`

Tipe: String

Diperlukan: Ya

ID sistem file FSx for Windows File Server untuk digunakan.

`rootDirectory`

Tipe: String

Diperlukan: Ya

Direktori dalam sistem file FSx for Windows File Server untuk dipasang sebagai direktori root di dalam host.

`authorizationConfig`

`credentialsParameter`

Tipe: String

Wajib: Ya

Opsi kredensial otorisasi:

- Nama Sumber Daya Amazon (ARN) dari rahasia Secrets Manager.
- Nama Sumber Daya Amazon (ARN) dari parameter Systems Manager.

`domain`

Tipe: String

Diperlukan: Ya

Nama domain yang sepenuhnya memenuhi syarat yang di-host oleh direktori AWS Directory Service for Microsoft Active Directory(AWS Managed Microsoft AD) atau Direktori Aktif EC2 yang dihosting sendiri.

Metode penyimpanan kredensial

Ada dua metode yang berbeda untuk menyimpan kredensial untuk digunakan dengan parameter kredensial.

- AWS Secrets Manager rahasia

  Kredensi ini dapat dibuat di AWS Secrets Manager konsol dengan menggunakan kategori rahasia jenis lain. Anda menambahkan baris untuk setiap pasangan kunci/nilai, nama pengguna/admin dan kata sandi/*`kata sandi`*.

- Parameter Systems Manager

  Kredensyal ini dapat dibuat di konsol parameter Systems Manager dengan memasukkan teks dalam formulir yang ada di cuplikan kode contoh berikut.

```
{
  "username": "admin",
  "password": "password"
}
```

`FSxWindowsFileServerVolumeConfiguration`Parameter `credentialsParameter` dalam definisi tugas memegang ARN rahasia atau parameter Systems Manager ARN. Untuk informasi selengkapnya, lihat Apa itu AWS Secrets Manager di Panduan Pengguna Secrets Manager dan Parameter Store Systems Manager dari Panduan Pengguna Systems Manager.

Volume Docker

Saat menggunakan volume Docker, built-in driver `local` atau driver volume pihak ke tiga dapat digunakan. Volume Docker dikelola oleh Docker dan direktori dibuat di `/var/lib/docker/volumes` pada instans kontainer yang berisi data volume.

Untuk menggunakan volume Docker, tentukan `dockerVolumeConfiguration` dalam ketentuan tugas Anda. Untuk informasi selengkapnya, lihat Menggunakan Volume.

Beberapa kasus penggunaan umum untuk volume Docker adalah sebagai berikut:

- Untuk menyediakan volume data tetap untuk digunakan dengan kontainer
- Untuk berbagi volume data yang ditetapkan di lokasi yang berbeda pada kontainer yang berbeda pada instans kontainer yang sama
- Untuk menentukan volume data kosong, tidak tetap dan memasangnya pada beberapa kontainer dalam tugas yang sama
- Untuk memberikan volume data ke tugas Anda yang dikelola oleh driver pihak ketiga

Pertimbangan volume Docker

Pertimbangkan hal berikut saat menggunakan volume Docker:

- Volume Docker hanya didukung saat menggunakan tipe peluncuran EC2 atau instance eksternal.
- Kontainer Windows hanya mendukung penggunaan driver `local`.
- Jika driver pihak ketiga digunakan, pastikan itu diinstal dan aktif pada instance kontainer sebelum agen kontainer dimulai. Jika driver pihak ketiga tidak aktif sebelum agen dimulai, Anda dapat memulai ulang agen kontainer menggunakan salah satu perintah berikut:
  - Untuk Amazon ECS yang dioptimalkan Amazon Linux 2 AMI:

    ```
    sudo systemctl restart ecs
    ```

  - Untuk Amazon ECS yang dioptimalkan Amazon Linux AMI:

    ```
    sudo stop ecs && sudo start ecs
    ```

Menentukan volume Docker dalam ketentuan tugas Anda

Sebelum kontainer Anda dapat menggunakan volume data, Anda harus menentukan konfigurasi volume dan titik pemasangan dalam ketentuan tugas Anda. Bagian ini menjelaskan konfigurasi volume untuk kontainer. Untuk tugas yang menggunakan volume Docker, tentukan `dockerVolumeConfiguration`. Untuk tugas yang menggunakan volume host pemasangan terikat, tentukan `host` dan `sourcePath` opsional.

Definisi tugas berikut JSON menunjukkan sintaks untuk `volumes` dan `mountPoints` objek untuk wadah.

```
{
    "containerDefinitions": [
        {
            "mountPoints": [
                {
                    "sourceVolume": "string",
                    "containerPath": "/path/to/mount_volume",
                    "readOnly": boolean
                }
            ]
        }
    ],
```

```
    "volumes": [
        {
            "name": "string",
            "dockerVolumeConfiguration": {
                "scope": "string",
                "autoprovision": boolean,
                "driver": "string",
                "driverOpts": {
                    "key": "value"
                },
                "labels": {
                    "key": "value"
                }
            }
        }
    ]
}
```

`name`

Tipe: String

Wajib: Tidak

Nama volume. Hingga 255 huruf (huruf besar dan kecil), angka, tanda hubung (), dan garis bawah (-) diperbolehkan. _ Nama ini direferensikan dalam `sourceVolume` parameter `mountPoints` objek definisi kontainer.

`dockerVolumeConfiguration`

Jenis: DockerVolumeConfigurationObjek

Diperlukan: Tidak

Parameter ini ditentukan saat menggunakan volume Docker. Volume Docker hanya didukung saat menjalankan tugas pada instans EC2. Wadah Windows hanya mendukung penggunaan `local` driver. Untuk menggunakan pemasangan mengikat, tentukan `host` saja.

`scope`

Tipe: String

Nilai yang Valid: `task` | `shared`

Diperlukan: Tidak

Cakupan untuk volume Docker, yang menentukan siklus hidupnya. Volume Docker yang tercakup ke `task` secara otomatis disediakan saat tugas dimulai dan dihancurkan saat tugas berhenti. Volume Docker yang tercakup sebagai `shared` dipertahankan setelah tugas berhenti.

`autoprovision`

Jenis: Boolean

Nilai default: `false`

Diperlukan: Tidak

Jika nilai ini`true`, volume Docker dibuat jika belum ada. Bidang ini hanya digunakan jika `scope` ada`shared`. Jika `scope` ya`task`, maka parameter ini harus dihilangkan atau disetel ke. `false`

`driver`

Tipe: String

Wajib: Tidak

Driver volume Docker yang digunakan. Nilai driver harus sesuai dengan nama driver yang disediakan oleh Docker karena nama ini digunakan untuk penempatan tugas. Jika driver diinstal dengan menggunakan CLI plugin Docker, `docker plugin ls` gunakan untuk mengambil nama driver dari instance container Anda. Jika driver diinstal dengan menggunakan metode lain, gunakan penemuan plugin Docker untuk mengambil nama driver. Untuk informasi lebih lanjut, lihat Penemuan plugin Docker. Parameter ini dipetakan ke `Driver` bagian Create a volume dari Docker Remote API dan `--driver` opsi untuk `docker volume create`.

`driverOpts`

Tipe: String

Wajib: Tidak

Peta opsi khusus driver Docker untuk dilewati. Parameter ini dipetakan ke `DriverOpts` bagian Create a volume dari Docker Remote API dan `--opt` opsi untuk `docker volume create`.

`labels`

Tipe: String

Wajib: Tidak

Metadata kustom untuk ditambahkan ke volume Docker. Parameter ini dipetakan ke `Labels` bagian Create a volume dari Docker Remote API dan `--label` opsi untuk `docker volume create`.

`mountPoints`

Tipe: Array objek

Diperlukan: Tidak

Titik pemasangan untuk volume data dalam penampung Anda. Parameter ini memetakan ke `Volumes` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--volume` untuk docker run.

Kontainer Windows dapat memasang seluruh direktori pada drive yang sama dengan `$env:ProgramData`. Kontainer Windows tidak dapat memasang direktori pada drive yang berbeda, dan titik pemasangan tidak dapat digunakan di seluruh drive.

`sourceVolume`

Jenis: String

Diperlukan: Ya, kapan `mountPoints` digunakan

Nama volume yang akan dipasang.

`containerPath`

Jenis: String

Diperlukan: Ya, kapan `mountPoints` digunakan

Jalur dalam wadah tempat volume akan dipasang.

`readOnly`

Tipe: Boolean

Wajib: Tidak

Jika nilai ini adalah `true`, kontainer memiliki akses hanya-baca ke volume. Jika nilai ini adalah `false`, maka kontainer dapat menulis ke volume. Nilai default-nya adalah `false`.

Contoh volume Docker

Berikut ini adalah contoh yang menunjukkan penggunaan volume Docker.

Menyediakan penyimpanan tidak tetap untuk kontainer menggunakan volume Docker

Dalam contoh ini, wadah menggunakan volume data kosong yang dibuang setelah tugas selesai. Salah satu contoh kasus penggunaan adalah Anda mungkin memiliki wadah yang perlu mengakses beberapa lokasi penyimpanan file awal selama tugas. Tugas ini dapat dicapai dengan menggunakan volume Docker.

1. Dalam ketentuan tugas bagian `volumes`, tentukan volume data dengan `name` dan nilai `DockerVolumeConfiguration`. Dalam contoh ini, kami menentukan ruang lingkup sebagai `task` jadi volume dihapus setelah tugas berhenti dan menggunakan built-in driver `local`.

```
"volumes": [
    {
        "name": "scratch",
        "dockerVolumeConfiguration" : {
            "scope": "task",
            "driver": "local",
            "labels": {
                "scratch": "space"
            }
        }
    }
]
```

2. Di bagian `containerDefinitions`, mendefiniskan sebuah kontainer dengan nilai-nilai `mountPoints` yang mereferensikan nama volume yang didefinisikan dan nilai `containerPath` untuk memasang volume pada kontainer.

```
"containerDefinitions": [
    {
        "name": "container-1",
        "mountPoints": [
            {
              "sourceVolume": "scratch",
              "containerPath": "/var/scratch"
            }
        ]
    }
```

```
]
```

Untuk menyediakan penyimpanan tetap untuk kontainer menggunakan volume Docker

Dalam contoh ini, Anda ingin volume bersama untuk beberapa kontainer untuk digunakan dan Anda ingin tetap ada setelah satu tugas yang menggunakannya dihentikan. `local`Driver bawaan sedang digunakan. Ini agar volume masih terikat dengan siklus hidup instance kontainer.

1. Dalam ketentuan tugas bagian `volumes`, tentukan volume data dengan `name` dan nilai `DockerVolumeConfiguration`. Dalam contoh ini, tentukan `shared` cakupan sehingga volume tetap ada, setel penyediaan otomatis ke. `true` Ini agar volume dibuat untuk digunakan. Kemudian, gunakan juga `local` driver bawaan.

   ```
   "volumes": [
       {
           "name": "database",
           "dockerVolumeConfiguration" : {
               "scope": "shared",
               "autoprovision": true,
               "driver": "local",
               "labels": {
                   "database": "database_name"
               }
           }
       }
   ]
   ```

2. Di bagian `containerDefinitions`, mendefiniskan sebuah kontainer dengan nilai-nilai `mountPoints` yang mereferensikan nama volume yang didefinisikan dan nilai `containerPath` untuk memasang volume pada kontainer.

   ```
   "containerDefinitions": [
       {
           "name": "container-1",
           "mountPoints": [
           {
             "sourceVolume": "database",
             "containerPath": "/var/database"
           }
         ]
       },
   ```

```
    {
      "name": "container-2",
      "mountPoints": [
        {
          "sourceVolume": "database",
          "containerPath": "/var/database"
        }
      ]
    }
  ]
```

Untuk menyediakan penyimpanan persisten NFS untuk wadah menggunakan volume Docker

Dalam contoh ini, kontainer menggunakan volume data NFS yang dipasang secara otomatis saat tugas dimulai dan dilepas saat tugas berhenti. Ini menggunakan `local` driver bawaan Docker. Salah satu contoh kasus penggunaan adalah Anda mungkin memiliki penyimpanan NFS lokal dan perlu mengaksesnya dari tugas ECS Anywhere. Ini dapat dicapai dengan menggunakan volume Docker dengan opsi driver NFS.

1. Dalam ketentuan tugas bagian `volumes`, tentukan volume data dengan `name` dan nilai `DockerVolumeConfiguration`. Dalam contoh ini, tentukan `task` cakupan sehingga volume dilepas setelah tugas berhenti. Gunakan `local` driver dan konfigurasikan `driverOpts` dengan`type`,`device`, dan o opsi yang sesuai. Ganti `NFS_SERVER` dengan endpoint server NFS.

```
"volumes": [
       {
           "name": "NFS",
           "dockerVolumeConfiguration" : {
               "scope": "task",
               "driver": "local",
               "driverOpts": {
                   "type": "nfs",
                   "device": "$NFS_SERVER:/mnt/nfs",
                   "o": "addr=$NFS_SERVER"
               }
           }
       }
   ]
```

2. Di `containerDefinitions` bagian tersebut, tentukan wadah dengan `mountPoints` nilai yang mereferensikan nama volume yang ditentukan dan `containerPath` nilai untuk memasang volume pada wadah.

```
"containerDefinitions": [
       {
           "name": "container-1",
           "mountPoints": [
               {
                 "sourceVolume": "NFS",
                 "containerPath": "/var/nfsmount"
               }
           ]
       }
   ]
```

Pemasangan terikat

Dengan pemasangan bind, file atau direktori pada host, seperti instans Amazon EC2, dipasang ke dalam wadah. Bind mount didukung untuk tugas yang di-host di instans Fargate dan Amazon EC2. Dudukan pengikat diikat ke siklus hidup wadah yang menggunakannya. Setelah semua container yang menggunakan bind mount dihentikan, seperti saat tugas dihentikan, data akan dihapus. Untuk tugas yang dihosting di instans Amazon EC2, data dapat dikaitkan dengan siklus hidup instans Amazon EC2 host dengan menentukan nilai dan opsional dalam definisi tugas Anda. `host` `sourcePath` Untuk informasi selengkapnya, lihat Menggunakan pemasangan terikat dalam dokumentasi Docker.

Berikut ini adalah kasus penggunaan umum untuk pemasangan terikat.

- Untuk menyediakan volume data kosong untuk memasang dalam satu atau lebih kontainer.
- Untuk memasang volume data host dalam satu atau lebih kontainer.
- Untuk berbagi volume data dari kontainer sumber dengan kontainer lain dalam tugas yang sama.
- Untuk mengekspos jalur dan isinya dari Dockerfile ke satu atau lebih kontainer.

Pertimbangan saat menggunakan pemasangan terikat

Saat menggunakan bind mount, pertimbangkan hal berikut.

- Untuk tugas-tugas yang di-host AWS Fargate menggunakan versi platform `1.4.0` atau yang lebih baru (Linux) `1.0.0` atau yang lebih baru (Windows), secara default mereka menerima minimal 20 GiB penyimpanan sementara untuk bind mount. Untuk tugas Linux, jumlah total penyimpanan sementara dapat ditingkatkan hingga maksimum 200 GiB dengan menentukan `ephemeralStorage` parameter dalam definisi tugas Anda. Secara default, tugas Amazon ECS Windows yang di-host di Fargate menggunakan `1.0.0` versi platform atau yang lebih baru menerima minimal 20 GiB penyimpanan sementara. Anda dapat meningkatkan jumlah total penyimpanan sementara, hingga maksimum 200 GiB dengan menentukan `ephemeralStorage` parameter dalam definisi tugas Anda.
- Untuk mengekspos file dari Dockerfile ke volume data saat tugas dijalankan, bidang data Amazon ECS mencari arahan. `VOLUME` Jika jalur absolut yang ditentukan dalam `VOLUME` direktif sama dengan yang ditentukan dalam definisi tugas, data di jalur `VOLUME` direktif akan disalin ke volume data. `containerPath` Dalam contoh Dockerfile berikut, file yang diberi nama `examplefile` dalam `/var/log/exported` direktori ditulis ke host dan kemudian dipasang di dalam wadah.

  ```
  FROM public.ecr.aws/amazonlinux/amazonlinux:latest
  RUN mkdir -p /var/log/exported
  RUN touch /var/log/exported/examplefile
  VOLUME ["/var/log/exported"]
  ```

  Secara default, izin volume diatur ke `0755` dan pemilik sebagai `root`. Anda dapat menyesuaikan izin ini di Dockerfile. Contoh berikut mendefinisikan pemilik direktori sebagai `node`.

  ```
  FROM public.ecr.aws/amazonlinux/amazonlinux:latest
  RUN yum install -y shadow-utils && yum clean all
  RUN useradd node
  RUN mkdir -p /var/log/exported && chown node:node /var/log/exported
  RUN touch /var/log/exported/examplefile
  USER node
  VOLUME ["/var/log/exported"]
  ```

- Untuk tugas yang dihosting di instans Amazon EC2, saat `sourcePath` nilai `host` dan tidak ditentukan, daemon Docker mengelola pemasangan pengikat untuk Anda. Ketika tidak ada kontainer yang mereferensikan pemasangan pengikat ini, layanan pembersihan tugas agen penampung Amazon ECS akhirnya menghapusnya. Secara default, ini terjadi tiga jam setelah wadah keluar. Namun, Anda dapat mengonfigurasi durasi ini dengan variabel `ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION` agen. Untuk informasi selengkapnya, lihat

Konfigurasi agen kontainer Amazon ECS. Jika Anda memerlukan data ini untuk bertahan di luar siklus hidup penampung, tentukan `sourcePath` nilai untuk bind mount.

Menentukan pemasangan terikat dalam ketentuan tugas Anda

Untuk tugas Amazon ECS yang di-host di instance Fargate atau Amazon EC2, cuplikan JSON definisi tugas berikut menunjukkan sintaks untuk,, dan objek untuk `volumes` definisi tugas. `mountPoints` `ephemeralStorage`

```
{
   "family": "",
   ...
   "containerDefinitions" : [
      {
         "mountPoints" : [
            {
               "containerPath" : "/path/to/mount_volume",
               "sourceVolume" : "string"
            }
          ],
          "name" : "string"
       }
    ],
    ...
    "volumes" : [
       {
          "name" : "string"
       }
    ],
    "ephemeralStorage": {
    "sizeInGiB": integer
    }
}
```

Untuk tugas Amazon ECS yang dihosting di instans Amazon EC2, Anda dapat menggunakan parameter `host` opsional dan saat menentukan `sourcePath` detail volume tugas. Ketika ditentukan, itu mengikat mount ke siklus hidup tugas daripada wadah.

```
"volumes" : [
    {
        "host" : {
```

```
            "sourcePath" : "string"
        },
        "name" : "string"
    }
]
```

Berikut ini adalah penjelasan yang lebih detail untuk setiap parameter ketentuan tugas.

`name`

Tipe: String

Wajib: Tidak

Nama volume. Hingga 255 huruf (huruf besar dan kecil), angka, tanda hubung (), dan garis bawah (-) diperbolehkan. _ Nama ini direferensikan dalam `sourceVolume` parameter `mountPoints` objek definisi kontainer.

`host`

Diperlukan: Tidak

`host`Parameter ini digunakan untuk mengikat siklus hidup bind mount ke instans Amazon EC2 host, bukan tugas, dan tempat penyimpanannya. Jika `host` parameternya kosong, maka daemon Docker menetapkan jalur host untuk volume data Anda, tetapi data tidak dijamin akan bertahan setelah wadah yang terkait dengannya berhenti berjalan.

Kontainer Windows dapat memasang seluruh direktori pada drive yang sama dengan `$env:ProgramData`.

> **Note**
>
> `sourcePath`Parameter hanya didukung saat menggunakan tugas yang di-host di instans Amazon EC2.

`sourcePath`

Tipe: String

Wajib: Tidak

Saat `host` parameter digunakan, tentukan a `sourcePath` untuk mendeklarasikan jalur pada instans Amazon EC2 host yang disajikan ke wadah. Jika parameter ini kosong, daemon Docker akan menetapkan jalur host untuk Anda. Jika `host` parameter berisi lokasi `sourcePath` file, maka volume data tetap ada di lokasi yang ditentukan pada instans Amazon EC2 host hingga Anda menghapusnya secara manual. Jika `sourcePath` nilai tidak ada pada instans Amazon EC2 host, daemon Docker membuatnya. Jika lokasinya memang ada, konten dari folder jalur sumber diekspor.

`mountPoints`

Tipe: Array objek

Diperlukan: Tidak

Titik pemasangan untuk volume data dalam penampung Anda. Parameter ini memetakan ke `Volumes` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--volume` untuk docker run.

Kontainer Windows dapat memasang seluruh direktori pada drive yang sama dengan `$env:ProgramData`. Kontainer Windows tidak dapat memasang direktori pada drive yang berbeda, dan titik pemasangan tidak dapat digunakan di seluruh drive.

`sourceVolume`

Jenis: String

Diperlukan: Ya, kapan `mountPoints` digunakan

Nama volume yang akan dipasang.

`containerPath`

Jenis: String

Diperlukan: Ya, kapan `mountPoints` digunakan

Jalur dalam wadah tempat volume akan dipasang.

`readOnly`

Tipe: Boolean

Wajib: Tidak

Jika nilai ini adalah `true`, kontainer memiliki akses hanya-baca ke volume. Jika nilai ini adalah `false`, maka kontainer dapat menulis ke volume. Nilai default-nya adalah `false`.

`ephemeralStorage`

Tipe: Objek

Wajib: Tidak

Jumlah penyimpanan sementara yang dialokasikan untuk tugas. Parameter ini digunakan untuk memperluas jumlah total penyimpanan sementara yang tersedia, di luar jumlah default, untuk tugas yang dihosting AWS Fargate menggunakan versi platform atau yang lebih baru (Linux) `1.4.0` atau `1.0.0` atau yang lebih baru (Windows).

Anda dapat menggunakan Copilot CLI CloudFormation, AWS SDK atau CLI untuk menentukan penyimpanan sementara untuk pemasangan pengikat.

Contoh pemasangan terikat

Contoh berikut mencakup kasus penggunaan yang paling umum untuk menggunakan pemasangan terikat untuk kontainer Anda.

Untuk mengalokasikan peningkatan jumlah ruang penyimpanan sementara untuk tugas Fargate

Untuk tugas Amazon ECS yang di-host di Fargate menggunakan `1.4.0` versi platform atau yang lebih baru (Linux) `1.0.0` atau (Windows), Anda dapat mengalokasikan lebih dari jumlah penyimpanan sementara default untuk wadah dalam tugas yang akan digunakan. Contoh ini dapat dimasukkan ke dalam contoh lain untuk mengalokasikan lebih banyak penyimpanan sementara untuk tugas Fargate Anda.

- Dalam ketentuan tugas, tentukan objek `ephemeralStorage`. `sizeInGiB` harus integer antara nilai-nilai `21` dan `200` dinyatakan dalam GiB.

```
"ephemeralStorage": {
    "sizeInGiB": integer
}
```

Untuk menyediakan volume data kosong untuk satu atau lebih kontainer.

Dalam beberapa kasus, Anda ingin menyediakan kontainer dalam tugas beberapa ruang scratch. Misalnya, Anda mungkin memiliki dua wadah database yang perlu mengakses lokasi penyimpanan file awal yang sama selama tugas. Hal ini dapat dicapai dengan menggunakan pemasangan terikat.

1. Dalam ketentuan tugas bagian `volumes`, tentukan pemasangan terikat dengan nama `database_scratch`.

```
  "volumes": [
    {
      "name": "database_scratch"
    }
  ]
```

2. Di `containerDefinitions` bagian ini, buat definisi wadah database. Ini agar mereka memasang volume.

```
"containerDefinitions": [
    {
      "name": "database1",
      "image": "my-repo/database",
      "cpu": 100,
      "memory": 100,
      "essential": true,
      "mountPoints": [
        {
          "sourceVolume": "database_scratch",
          "containerPath": "/var/scratch"
        }
      ]
    },
    {
      "name": "database2",
      "image": "my-repo/database",
      "cpu": 100,
      "memory": 100,
      "essential": true,
      "mountPoints": [
        {
          "sourceVolume": "database_scratch",
          "containerPath": "/var/scratch"
        }
      ]
    }
  ]
```

Untuk mengekspos jalur dan isinya dari Dockerfile ke kontainer.

Dalam contoh ini, Anda memiliki Dockerfile yang menulis data yang ingin Anda pasang di dalam kontainer. Contoh ini berfungsi untuk tugas yang di-host di instance Fargate atau Amazon EC2.

1. Buat Dockerfile. Contoh berikut menggunakan image kontainer Amazon Linux 2 publik dan membuat file yang diberi nama `examplefile` di `/var/log/exported` direktori yang ingin kita pasang di dalam wadah. Direktif `VOLUME` harus menentukan jalur absolut.

   ```
   FROM public.ecr.aws/amazonlinux/amazonlinux:latest
   RUN mkdir -p /var/log/exported
   RUN touch /var/log/exported/examplefile
   VOLUME ["/var/log/exported"]
   ```

   Secara default, izin volume diatur ke `0755` dan pemilik sebagai `root`. Izin ini dapat diubah di Dockerfile. Contoh berikut mendefinisikan pemilik direktori `/var/log/exported` diatur ke `node`.

   ```
   FROM public.ecr.aws/amazonlinux/amazonlinux:latest
   RUN yum install -y shadow-utils && yum clean all
   RUN useradd node
   RUN mkdir -p /var/log/exported && chown node:node /var/log/exported
   USER node
   RUN touch /var/log/exported/examplefile
   VOLUME ["/var/log/exported"]
   ```

2. Dalam ketentuan tugas bagian `volumes`, tentukan volume dengan nama `application_logs`.

   ```
   "volumes": [
     {
       "name": "application_logs"
     }
   ]
   ```

3. Di `containerDefinitions` bagian ini, buat definisi wadah aplikasi. Ini agar mereka memasang penyimpanan. `containerPath`Nilai harus cocok dengan jalur absolut yang ditentukan dalam `VOLUME` arahan dari Dockerfile.

   ```
   "containerDefinitions": [
     {
       "name": "application1",
   ```

```
      "image": "my-repo/application",
      "cpu": 100,
      "memory": 100,
      "essential": true,
      "mountPoints": [
        {
          "sourceVolume": "application_logs",
          "containerPath": "/var/log/exported"
        }
      ]
    },
    {
      "name": "application2",
      "image": "my-repo/application",
      "cpu": 100,
      "memory": 100,
      "essential": true,
      "mountPoints": [
        {
          "sourceVolume": "application_logs",
          "containerPath": "/var/log/exported"
        }
      ]
    }
  ]
```

Untuk menyediakan volume data kosong untuk wadah yang terkait dengan siklus hidup instans Amazon EC2 host

Untuk tugas yang di-host di instans Amazon EC2, Anda dapat menggunakan mount bind dan memiliki data yang terkait dengan siklus hidup instans Amazon EC2 host. Anda dapat melakukan ini dengan menggunakan `host` parameter dan menentukan `sourcePath` nilai. File apa pun yang ada di `sourcePath` disajikan ke wadah pada `containerPath` nilainya. File apa pun yang ditulis dengan `containerPath` nilai ditulis ke `sourcePath` nilai pada instans Amazon EC2 host.

> **Important**
>
> Amazon ECS tidak menyinkronkan penyimpanan Anda di seluruh instans Amazon EC2. Tugas yang menggunakan penyimpanan persisten dapat ditempatkan pada instans Amazon EC2 apa pun di klaster Anda yang memiliki kapasitas yang tersedia. Jika tugas Anda memerlukan penyimpanan persisten setelah berhenti dan memulai ulang, selalu tentukan

instans Amazon EC2 yang sama pada waktu peluncuran tugas dengan AWS CLI perintah start-task. Anda juga dapat menggunakan volume Amazon EFS untuk penyimpanan persisten. Untuk informasi selengkapnya, lihat Volume Amazon EFS.

1. Dalam ketentuan tugas bagian `volumes`, tentukan pemasangan terikat dengan `name` dan nilai `sourcePath`. Dalam contoh berikut, instans Amazon EC2 host berisi data `/ecs/webdata` yang ingin Anda pasang di dalam wadah.

```
"volumes": [
  {
    "name": "webdata",
    "host": {
      "sourcePath": "/ecs/webdata"
    }
  }
]
```

2. Di bagian `containerDefinitions`, mendefiniskan sebuah kontainer dengan nilai-nilai `mountPoints` yang mereferensikan nama pemasangan terikat dan nilai `containerPath` untuk memasang pemasangan terikat pada kontainer.

```
"containerDefinitions": [
  {
    "name": "web",
    "image": "nginx",
    "cpu": 99,
    "memory": 100,
    "portMappings": [
      {
        "containerPort": 80,
        "hostPort": 80
      }
    ],
    "essential": true,
    "mountPoints": [
      {
        "sourceVolume": "webdata",
        "containerPath": "/usr/share/nginx/html"
      }
    ]
```

```
    }
  ]
```

Untuk memasang volume yang ditetapkan pada beberapa kontainer di lokasi yang berbeda

Anda dapat menentukan volume data dalam ketentuan tugas dan memasang volume tersebut di lokasi yang berbeda pada kontainer yang berbeda. Misalnya, wadah host Anda memiliki folder data situs web di`/data/webroot`. Anda mungkin ingin memasang volume data tersebut sebagai read-only pada dua server web berbeda yang memiliki akar dokumen berbeda.

1. Dalam ketentuan tugas bagian `volumes`, tentukan volume data dengan nama `webroot` dan jalur sumber `/data/webroot`.

   ```
   "volumes": [
     {
       "name": "webroot",
       "host": {
         "sourcePath": "/data/webroot"
       }
     }
   ]
   ```

2. Di bagian `containerDefinitions`, tentukan kontainer untuk setiap server web dengan nilai-nilai `mountPoints` yang mengasosiasikan volume `webroot` dengan nilai `containerPath` menunjuk ke akar dokumen untuk kontainer itu.

   ```
   "containerDefinitions": [
     {
       "name": "web-server-1",
       "image": "my-repo/ubuntu-apache",
       "cpu": 100,
       "memory": 100,
       "portMappings": [
         {
           "containerPort": 80,
           "hostPort": 80
         }
       ],
       "essential": true,
       "mountPoints": [
         {
   ```

```
          "sourceVolume": "webroot",
          "containerPath": "/var/www/html",
          "readOnly": true
        }
      ]
    },
    {
      "name": "web-server-2",
      "image": "my-repo/sles11-apache",
      "cpu": 100,
      "memory": 100,
      "portMappings": [
        {
          "containerPort": 8080,
          "hostPort": 8080
        }
      ],
      "essential": true,
      "mountPoints": [
        {
          "sourceVolume": "webroot",
          "containerPath": "/srv/www/htdocs",
          "readOnly": true
        }
      ]
    }
  ]
```

Untuk memasang volume dari kontainer lain menggunakan **`volumesFrom`**.

Untuk tugas yang dihosting di instans Amazon EC2, Anda dapat menentukan satu atau beberapa volume pada penampung, lalu menggunakan `volumesFrom` parameter dalam definisi container yang berbeda dalam tugas yang sama untuk memasang semua volume dari titik pemasangan yang `sourceContainer` ditentukan semula. Parameter`volumesFrom` berlaku untuk volume yang didefinisikan dalam ketentuan tugas, dan volume yang dibangun ke dalam citra dengan Dockerfile.

1. (Opsional) Untuk berbagi volume yang dibangun ke dalam gambar, gunakan `VOLUME` instruksi di Dockerfile. Contoh berikut Dockerfile menggunakan `httpd` gambar, dan kemudian menambahkan volume dan memasangnya `dockerfile_volume` di root dokumen Apache. Ini adalah folder yang digunakan oleh server `httpd` web.

```
FROM httpd
VOLUME ["/usr/local/apache2/htdocs/dockerfile_volume"]
```

Anda dapat membangun citra dengan Dockerfile ini dan mendorongnya ke repositori, seperti Docker Hub, dan menggunakannya dalam ketentuan tugas Anda. Contoh `my-repo/httpd_dockerfile_volume` gambar yang digunakan dalam langkah-langkah berikut dibuat dengan Dockerfile sebelumnya.

2. Buat ketentuan tugas yang mendefinisikan volume lain dan memasang titik untuk kontainer. Dalam contoh bagian `volumes` ini, Anda akan membuat volume kosong yang disebut `empty`, yang dikelola oleh daemon Docker. Ada juga volume host yang ditentukan yang disebut`host_etc`. Ini mengekspor `/etc` folder pada instance wadah host.

```
{
  "family": "test-volumes-from",
  "volumes": [
    {
      "name": "empty",
      "host": {}
    },
    {
      "name": "host_etc",
      "host": {
        "sourcePath": "/etc"
      }
    }
  ],
```

Di bagian ketentuan kontainer, buat kontainer yang memasang volume yang didefinisikan sebelumnya. Dalam contoh ini, `web` wadah memasang `host_etc` volume `empty` dan. Ini adalah wadah yang menggunakan gambar yang dibangun dengan volume di Dockerfile.

```
"containerDefinitions": [
    {
      "name": "web",
      "image": "my-repo/httpd_dockerfile_volume",
      "cpu": 100,
      "memory": 500,
      "portMappings": [
        {
```

```
          "containerPort": 80,
          "hostPort": 80
        }
      ],
      "mountPoints": [
        {
          "sourceVolume": "empty",
          "containerPath": "/usr/local/apache2/htdocs/empty_volume"
        },
        {
          "sourceVolume": "host_etc",
          "containerPath": "/usr/local/apache2/htdocs/host_etc"
        }
      ],
      "essential": true
    },
```

Buat kontainer lain yang menggunakan `volumesFrom` untuk memasang semua volume yang terkait dengan kontainer `web`. Semua volume pada `web` wadah juga dipasang pada `busybox` wadah. Ini termasuk volume yang ditentukan dalam Dockerfile yang digunakan untuk membangun gambar. `my-repo/httpd_dockerfile_volume`

```
    {
      "name": "busybox",
      "image": "busybox",
      "volumesFrom": [
        {
          "sourceContainer": "web"
        }
      ],
      "cpu": 100,
      "memory": 500,
      "entryPoint": [
        "sh",
        "-c"
      ],
      "command": [
        "echo $(date) > /usr/local/apache2/htdocs/empty_volume/date && echo $(date)
 > /usr/local/apache2/htdocs/host_etc/date && echo $(date) > /usr/local/apache2/
htdocs/dockerfile_volume/date"
      ],
      "essential": false
```

```
    }
  ]
}
```

Saat tugas ini dijalankan, kedua kontainer memasang volume, dan `command` dalam `busybox` wadah menulis tanggal dan waktu ke file. File ini dipanggil `date` di setiap folder volume. Folder kemudian terlihat di situs web yang ditampilkan oleh kontainer `web`.

> **Note**
>
> Karena kontainer `busybox` menjalankan perintah cepat dan kemudian keluar, itu harus ditetapkan sebagai `"essential": false` dalam ketentuan kontainer. Jika tidak, kontainer berhenti seluruh tugas ketika keluar.

Mengelola ruang tukar kontainer

Dengan Amazon ECS, Anda dapat mengontrol penggunaan ruang memori swap pada instans Amazon EC2 berbasis Linux di tingkat penampung. Menggunakan konfigurasi swap per kontainer, setiap kontainer dalam definisi tugas dapat mengaktifkan swap atau dinonaktifkan. Bagi mereka yang mengaktifkannya, jumlah maksimum ruang swap yang digunakan dapat dibatasi. Misalnya, container latency-critical dapat menonaktifkan swap. Sebaliknya, wadah dengan permintaan memori transien yang tinggi dapat mengaktifkan swap untuk mengurangi kemungkinan out-of-memory kesalahan saat wadah sedang dimuat.

Konfigurasi swap untuk kontainer dikelola oleh parameter definisi kontainer berikut.

`maxSwap`

Jumlah total memori tukar (dalam MiB) yang dapat digunakan oleh kontainer. Parameter ini diterjemahkan ke opsi `--memory-swap` untuk docker run di mana nilai adalah jumlah memori kontainer ditambah nilai `maxSwap`.

Jika nilai `maxSwap` sebesar `0` ditentukan, kontainer tidak menggunakan swap. Nilai yang diterima adalah `0` atau bilangan bulat positif. Jika `maxSwap` parameter diabaikan, kontainer menggunakan konfigurasi swap untuk instans kontainer tempatnya berjalan. Nilai `maxSwap` harus ditetapkan untuk parameter `swappiness` yang akan digunakan.

`swappiness`

Anda dapat menggunakan ini untuk menyetel perilaku swappiness memori kontainer. `swappiness`Nilai `0` penyebab pertukaran tidak terjadi kecuali diperlukan. `swappiness`Nilai `100` penyebab halaman ditukar secara agresif. Nilai yang diterima adalah bilangan bulat antara `0` dan `100`. Jika parameter `swappiness` tidak ditentukan, nilai default sebesar `60` akan digunakan. Jika nilai tidak ditentukan untuk`maxSwap`, parameter ini diabaikan. Parameter ini sesuai dengan opsi `--memory-swappiness` untuk docker run.

Dalam contoh berikut, sintaks JSON disediakan.

```
"containerDefinitions": [{
        ...
        "linuxParameters": {
            "maxSwap": integer,
            "swappiness": integer
        },
        ...
}]
```

Pertimbangan tukar kontainer

Pertimbangkan hal berikut ketika Anda menggunakan konfigurasi tukar per kontainer.

- Ruang swap harus diaktifkan dan dialokasikan pada instans Amazon EC2 yang menghosting tugas Anda agar kontainer dapat digunakan. Secara default, AMI Amazon ECS yang dioptimalkan tidak mengaktifkan swap. Anda harus mengaktifkan swap di instans untuk menggunakan fitur ini. Untuk informasi lebih lanjut, lihat Volume Swap Penyimpanan Instans dalam Panduan Pengguna Amazon EC2 untuk Instans Linux atau Bagaimana cara mengalokasikan memori agar berfungsi sebagai ruang tukar di instans Amazon EC2 dengan menggunakan file swap?.
- Parameter definisi wadah ruang swap hanya didukung untuk definisi tugas yang menentukan jenis peluncuran EC2. Parameter ini tidak didukung untuk definisi tugas yang ditujukan hanya untuk Amazon ECS pada penggunaan Fargate.
- Fitur ini hanya didukung untuk kontainer Linux. Kontainer Windows tidak didukung saat ini.
- Jika parameter definisi `maxSwap` dan `swappiness` kontainer dihilangkan dari definisi tugas, setiap kontainer memiliki `swappiness` nilai default. `60` Selain itu, total penggunaan swap dibatasi hingga dua kali reservasi memori kontainer.
- Jika Anda menggunakan tugas di Amazon Linux 2023, `swappiness` parameter tidak didukung.

Pertimbangan Fargate

Untuk menggunakan Fargate, Anda harus mengonfigurasi definisi tugas Anda untuk menggunakan jenis peluncuran Fargate. Ada pertimbangan tambahan saat menggunakan Fargate.

Topik

Parameter ketentuan tugas

Tugas yang menggunakan tipe peluncuran Fargate tidak mendukung semua parameter definisi tugas Amazon ECS yang tersedia. Beberapa parameter tidak didukung sama sekali, dan yang lain berperilaku berbeda untuk tugas Fargate.

Parameter definisi tugas berikut tidak valid dalam tugas Fargate:

- `disableNetworking`
- `dnsSearchDomains`
- `dnsServers`
- `dockerSecurityOptions`
- `extraHosts`
- `gpu`
- `ipcMode`
- `links`
- `placementConstraints`
- `privileged`
- `maxSwap`

- `swappiness`

Parameter definisi tugas berikut valid dalam tugas Fargate, tetapi memiliki batasan yang harus diperhatikan:

- `linuxParameters`— Saat menentukan opsi khusus Linux yang diterapkan ke wadah, satu-satunya kemampuan yang dapat Anda tambahkan adalah. `capabilities` `CAP_SYS_PTRACE` Parameter `devices`, `sharedMemorySize`, dan `tmpfs` tidak didukung. Untuk informasi selengkapnya, lihat Parameter Linux.
- `volumes`— Tugas Fargate hanya mendukung volume host bind mount, sehingga `dockerVolumeConfiguration` parameternya tidak didukung. Untuk informasi selengkapnya, lihat Volume.
- `cpu`- Untuk wadah Windows aktifAWS Fargate, nilainya tidak boleh kurang dari 1 vCPU.

Untuk memastikan bahwa definisi tugas Anda memvalidasi untuk digunakan dengan Fargate, Anda dapat menentukan hal berikut saat mendaftarkan definisi tugas:

- Di bidang AWS Management Console, untuk Memerlukan Kompatibilitas, tentukan. `FARGATE`
- Di AWS CLI, tentukan `--requires-compatibilities` opsi.
- Di Amazon ECS API, tentukan `requiresCompatibilities` flag.

Sistem Operasi dan Arsitektur

Saat Anda mengonfigurasi definisi tugas dan kontainerAWS Fargate, Anda harus menentukan Sistem Operasi tempat penampung berjalan. Sistem Operasi berikut didukung untukAWS Fargate:

- Amazon Linux 2

> **Note**
>
> Kontainer Linux hanya menggunakan konfigurasi kernel dan kernel dari Sistem Operasi host. Misalnya, konfigurasi kernel mencakup kontrol `sysctl` sistem. Gambar kontainer Linux dapat dibuat dari gambar dasar yang berisi file dan program dari distribusi Linux apa pun. Jika arsitektur CPU cocok, Anda dapat menjalankan container dari image container Linux apa pun di Sistem Operasi apa pun.

- Windows Server 2019 Full
- Windows Server 2019 Core
- Windows Server 2022 Lengkap
- Windows Server 2022 Inti

Ketika Anda menjalankan wadah WindowsAWS Fargate, Anda harus memiliki arsitektur CPU X86_64.

Saat menjalankan container LinuxAWS Fargate, Anda dapat menggunakan arsitektur CPU X86_64, atau arsitektur ARM64 untuk aplikasi berbasis ARM Anda. Untuk informasi selengkapnya, lihat the section called “Bekerja dengan beban kerja ARM 64-bit di Amazon ECS”.

CPU dan memori tugas

Definisi tugas Amazon ECS untuk AWS Fargate mengharuskan Anda menentukan CPU dan memori pada tingkat tugas. Meskipun Anda juga dapat menentukan CPU dan memori pada tingkat kontainer untuk tugas Fargate, ini opsional. Sebagian besar kasus penggunaan terpenuhi dengan hanya menentukan sumber daya ini pada tingkat tugas. Tabel berikut menunjukkan kombinasi yang valid antara CPU dan memori tingkat tugas. Anda dapat menentukan nilai memori dalam file JSON di MiB atau GB. Anda dapat menentukan nilai CPU dalam file JSON sebagai bilangan bulat dalam unit CPU atau CPU virtual (vCPU).

Nilai CPU	Nilai memori	Sistem operasi yang didukung untuk AWS Fargate
256 (.25 vCPU)	512 MiB, 1 GB, 2 GB	Linux
512 (.5 vCPU)	1 GB, 2 GB, 3 GB, 4 GB	Linux
1024 (1 vCPU)	2 GB, 3 GB, 4 GB, 5 GB, 6 GB, 7 GB, 8 GB	Linux, Windows
2048 (2 vCPU)	Antara 4 GB dan 16 GB dalam peningkatan 1 GB	Linux, Windows
4096 (4 vCPU)	Antara 8 GB dan 30 GB dalam peningkatan 1 GB	Linux, Windows

Nilai CPU	Nilai memori	Sistem operasi yang didukung untuk AWS Fargate
8192 (8 vCPU) **Note** Opsi ini membutuhkan platform Linux `1.4.0` atau yang lebih baru.	Antara 16 GB dan 60 GB dalam peningkatan 4 GB	Linux
16384 (16vCPU) **Note** Opsi ini membutuhkan platform Linux `1.4.0` atau yang lebih baru.	Antara 32 GB dan 120 GB dalam peningkatan 8 GB	Linux

Jaringan tugas

Tugas Amazon ECS untuk AWS Fargate memerlukan mode `awsvpc` jaringan, yang menyediakan setiap tugas dengan antarmuka network elastis. Ketika Anda menjalankan tugas atau membuat layanan dengan mode jaringan ini, Anda harus menentukan satu subnet atau lebih untuk melampirkan antarmuka jaringan dan satu grup keamanan atau lebih untuk diterapkan ke antarmuka jaringan.

Jika Anda menggunakan subnet publik, putuskan apakah Anda ingin untuk memberikan alamat IP publik untuk antarmuka jaringan atau tidak. Untuk tugas Fargate di subnet publik untuk menarik gambar kontainer, alamat IP publik perlu ditetapkan ke antarmuka network elastis tugas, dengan rute ke internet atau gateway NAT yang dapat merutekan permintaan ke internet. Untuk tugas Fargate di subnet pribadi untuk menarik gambar kontainer, Anda memerlukan gateway NAT di subnet untuk merutekan permintaan ke internet. Saat Anda meng-host gambar penampung Anda di Amazon ECR, Anda dapat mengonfigurasi Amazon ECR untuk menggunakan titik akhir VPC antarmuka. Dalam hal ini, alamat IPv4 pribadi tugas digunakan untuk penarikan gambar. Untuk informasi selengkapnya

tentang titik akhir antarmuka Amazon ECR, lihat Titik akhir VPC antarmuka Amazon ECR ()AWS PrivateLink di Panduan Pengguna Registri Amazon Elastic Container.

Berikut ini adalah contoh `networkConfiguration` bagian untuk layanan Fargate:

```
"networkConfiguration": {
   "awsvpcConfiguration": {
      "assignPublicIp": "ENABLED",
      "securityGroups": [ "sg-12345678" ],
      "subnets": [ "subnet-12345678" ]
   }
}
```

Batas sumber daya tugas

Definisi tugas Amazon ECS untuk kontainer Linux AWS Fargate mendukung `ulimits` parameter untuk menentukan batas sumber daya yang akan ditetapkan untuk wadah.

Definisi tugas Amazon ECS untuk Windows aktif AWS Fargate tidak mendukung `ulimits` parameter untuk menentukan batas sumber daya yang akan ditetapkan untuk wadah.

Tugas Amazon ECS yang dihosting di Fargate menggunakan nilai batas sumber daya default yang ditetapkan oleh sistem operasi dengan pengecualian parameter batas sumber daya. `nofile` Batas `nofile` sumber daya menetapkan batasan pada jumlah file terbuka yang dapat digunakan wadah. Di Fargate, batas `nofile` lunak default adalah `1024` dan batas keras adalah. `65535` Anda dapat mengatur nilai dari kedua batas hingga`1048576`.

Berikut ini adalah contoh potongan ketentuan tugas yang menunjukkan cara untuk menentukan batas `nofile` kustom yang telah dilipatgandakan:

```
"ulimits": [
    {
       "name": "nofile",
       "softLimit": 2048,
       "hardLimit": 8192
    }
]
```

Untuk informasi lebih lanjut tentang batas sumber daya lain yang dapat disesuaikan, lihat Batas sumber daya.

Pencatatan log

Pencatatan peristiwa

Amazon ECS mencatat tindakan yang diperlukan. EventBridge Anda dapat menggunakan acara Amazon ECS EventBridge untuk menerima notifikasi mendekati waktu nyata mengenai status klaster, layanan, dan tugas Amazon ECS Anda saat ini. Selain itu, Anda dapat mengotomatiskan tindakan untuk menanggapi peristiwa ini. Untuk informasi selengkapnya, lihat Otomatiskan tanggapan terhadap kesalahan Amazon ECS menggunakan EventBridge.

Pencatatan siklus hidup tugas

Tugas yang berjalan di Fargate menerbitkan stempel waktu untuk melacak tugas melalui status siklus hidup tugas. Anda dapat melihat stempel waktu dalam detail tugas di AWS Management Console dan dengan menjelaskan tugas di dan SDK AWS CLI . Misalnya, Anda dapat menggunakan stempel waktu untuk mengevaluasi berapa banyak waktu yang dihabiskan tugas untuk mengunduh gambar kontainer dan memutuskan apakah Anda harus mengoptimalkan ukuran gambar kontainer, atau menggunakan indeks OCI Seekable. Untuk informasi selengkapnya tentang praktik gambar kontainer, lihatPraktik terbaik untuk gambar kontainer.

Pencatatan aplikasi

Definisi tugas Amazon ECS untuk AWS Fargate mendukung`awslogs`,`splunk`, dan driver `awsfirelens` log untuk konfigurasi log.

Driver `awslogs` log mengonfigurasi tugas Fargate Anda untuk mengirim informasi log ke Amazon CloudWatch Logs. Berikut ini menunjukkan potongan ketentuan tugas di tempat driver log `awslogs` dikonfigurasi:

```
"logConfiguration": {
   "logDriver": "awslogs",
   "options": {
      "awslogs-group" : "/ecs/fargate-task-definition",
      "awslogs-region": "us-east-1",
      "awslogs-stream-prefix": "ecs"
   }
}
```

Untuk informasi selengkapnya tentang penggunaan driver `awslogs` log dalam definisi tugas untuk mengirim log kontainer Anda ke CloudWatch Log, lihatMenggunakan driver log awslogs.

Untuk informasi selengkapnya tentang driver log `awsfirelens` dalam ketentuan tugas, lihat Menggunakan perutean log khusus.

Untuk informasi selengkapnya tentang penggunaan driver log `splunk` dalam ketentuan tugas, lihat `splunk`driver log.

Penyimpanan tugas

Untuk tugas Amazon ECS yang dihosting di Fargate, jenis penyimpanan berikut didukung:

- Volume Amazon EBS menyediakan penyimpanan blok yang hemat biaya, tahan lama, dan berkinerja tinggi untuk beban kerja kontainer intensif data. Untuk informasi selengkapnya, lihat Volume Amazon EBS.
- Volume Amazon EFS untuk penyimpanan persisten. Untuk informasi selengkapnya, lihat Volume Amazon EFS.
- Bind mount untuk penyimpanan sementara. Untuk informasi selengkapnya, lihat Pemasangan terikat.

Lazy loading gambar kontainer menggunakan Seekable OCI (SOCI)

Tugas Amazon ECS di Fargate yang menggunakan `1.4.0` versi platform Linux dapat menggunakan Seekable OCI (SOCI) untuk membantu memulai tugas lebih cepat. Dengan SOCI, kontainer hanya menghabiskan beberapa detik pada penarikan gambar sebelum mereka dapat memulai, menyediakan waktu untuk pengaturan lingkungan dan instantiasi aplikasi saat gambar diunduh di latar belakang. Ini disebut lazy loading. Saat Fargate memulai tugas Amazon ECS, Fargate secara otomatis mendeteksi apakah indeks SOCI ada untuk gambar dalam tugas dan memulai penampung tanpa menunggu seluruh gambar diunduh.

Untuk kontainer yang berjalan tanpa indeks SOCI, gambar kontainer diunduh sepenuhnya sebelum penampung dimulai. Perilaku ini sama pada semua versi platform Fargate lainnya dan pada AMI yang dioptimalkan Amazon ECS di instans Amazon EC2.

Indeks OCI yang dapat dicari

Seekable OCI (SOCI) adalah teknologi open source yang dikembangkan oleh AWS yang dapat meluncurkan kontainer lebih cepat dengan malas memuat gambar kontainer. SOCI bekerja dengan membuat indeks (Indeks SOCI) dari file dalam gambar kontainer yang ada. Indeks ini membantu meluncurkan kontainer lebih cepat, memberikan kemampuan untuk mengekstrak file individual

dari gambar kontainer sebelum mengunduh seluruh gambar. Indeks SOCI harus disimpan sebagai artefak di repositori yang sama dengan gambar dalam registri kontainer. Anda hanya boleh menggunakan indeks SOCI dari sumber tepercaya, karena indeks adalah sumber otoritatif untuk konten gambar. Untuk informasi selengkapnya, lihat Memperkenalkan OCI yang Dapat Dicari untuk memuat gambar kontainer yang lambat.

Pertimbangan

Jika Anda ingin Fargate menggunakan indeks SOCI untuk memuat gambar kontainer dengan malas dalam suatu tugas, pertimbangkan hal berikut:

- Hanya tugas yang berjalan pada versi platform Linux yang `1.4.0` dapat menggunakan indeks SOCI. Tugas yang menjalankan kontainer Windows di Fargate tidak didukung.
- Tugas yang berjalan pada X86_64 atau arsitektur ARM64 CPU didukung. Tugas Linux dengan arsitektur ARM64 tidak mendukung penyedia kapasitas Fargate Spot.
- Gambar kontainer dalam definisi tugas harus memiliki indeks SOCI di registri kontainer yang sama dengan gambar.
- Gambar kontainer dalam definisi tugas harus disimpan dalam registri gambar yang kompatibel. Berikut daftar daftar yang kompatibel:
  - Pendaftar pribadi Amazon ECR.
- Hanya gambar kontainer yang menggunakan gzip kompresi atau tidak dikompresi yang didukung. Gambar kontainer yang menggunakan zstd kompresi tidak didukung.
- Kami menyarankan Anda mencoba lazy loading dengan gambar kontainer yang lebih besar dari ukuran 250 MiB terkompresi. Anda cenderung tidak melihat pengurangan waktu untuk memuat gambar yang lebih kecil.
- Karena lazy loading dapat mengubah berapa lama tugas Anda dimulai, Anda mungkin perlu mengubah berbagai timeout seperti masa tenggang pemeriksaan kesehatan untuk Elastic Load Balancing.
- Jika Anda ingin mencegah gambar kontainer dimuat lambat, hapus indeks SOCI dari registri kontainer. Jika gambar kontainer dalam tugas tidak memenuhi salah satu pertimbangan, gambar kontainer tersebut diunduh dengan metode default.

Membuat indeks OCI Seekable

Agar gambar kontainer dimuat lambat, diperlukan indeks SOCI (file metadata) yang dibuat dan disimpan di repositori gambar kontainer di sepanjang sisi gambar kontainer. Untuk membuat

dan mendorong indeks SOCI, Anda dapat menggunakan alat CLI soci-snapshotter open source. GitHub Atau, Anda dapat menerapkan CloudFormation AWS SOCI Index Builder. Ini adalah solusi tanpa server yang secara otomatis membuat dan mendorong indeks SOCI saat gambar kontainer didorong ke Amazon ECR. Untuk informasi selengkapnya tentang solusi dan langkah-langkah penginstalan, lihat Pembuat Indeks CloudFormation AWS SOCI di GitHub. CloudFormation AWS SOCI Index Builder adalah cara untuk mengotomatisasi memulai dengan SOCI, sementara alat soci open source memiliki lebih banyak fleksibilitas seputar pembuatan indeks dan kemampuan untuk mengintegrasikan pembuatan indeks dalam pipeline integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD) Anda.

> **Note**
>
> Agar indeks SOCI dibuat untuk gambar, gambar harus ada di penyimpanan containerd gambar di komputer yang sedang berjalan`soci-snapshotter`. Jika gambar ada di toko Docker gambar, gambar tidak dapat ditemukan.

Memverifikasi bahwa tugas menggunakan pemuatan lambat

Untuk memverifikasi bahwa tugas dimuat dengan malas menggunakan SOCI, periksa titik akhir metadata tugas dari dalam tugas. Saat Anda menanyakan titik akhir metadata tugas versi 4, ada `Snapshotter` bidang di jalur default untuk wadah yang Anda kueri. Selain itu, ada `Snapshotter` bidang untuk setiap kontainer di `/task` jalur. Nilai default untuk bidang ini adalah`overlayfs`, dan bidang ini diatur ke `soci` jika SOCI digunakan. Untuk informasi selengkapnya, lihat Titik akhir metadata tugas versi 4 di Panduan Pengguna Amazon Elastic Container Service untuk. AWS Fargate

Pertimbangan EC2 Windows

Tugas yang berjalan pada instans EC2 Windows tidak mendukung semua parameter definisi tugas Amazon ECS yang tersedia. Beberapa parameter tidak didukung sama sekali, dan yang lain berperilaku berbeda.

Parameter definisi tugas berikut tidak didukung untuk definisi tugas Amazon EC2 Windows:

- `containerDefinitions`
  - `disableNetworking`
  - `dnsServers`
  - `dnsSearchDomains`

  - `extraHosts`
  - `links`
  - `linuxParameters`
  - `privileged`
  - `readonlyRootFilesystem`
  - `user`
  - `ulimits`
- `volumes`
  - `dockerVolumeConfiguration`
- `cpu`

  Kami merekomendasikan agar CPU tingkat-kontainer ditentukan untuk kontainer Windows.
- `memory`

  Kami merekomendasikan agar CPU tingkat-kontainer ditentukan untuk kontainer Windows.
- `proxyConfiguration`
- `ipcMode`
- `pidMode`
- `taskRoleArn`

  Peran IAM untuk tugas pada fitur instans Windows EC2 memerlukan konfigurasi tambahan, tetapi sebagian besar konfigurasi ini mirip dengan mengonfigurasi peran IAM untuk tugas pada instance container Linux. Untuk mengetahui informasi selengkapnya, lihat the section called “Konfigurasi tambahan untuk peran tugas Windows”.

Membuat definisi tugas menggunakan konsol

Untuk membuat definisi tugas semudah mungkin, konsol Amazon ECS memiliki pilihan default untuk banyak pilihan.

Anda dapat membuat definisi tugas dengan menggunakan konsol, atau dengan mengedit file JSON.

Validasi JSON

Editor JSON konsol Amazon ECS memvalidasi hal berikut dalam file JSON:

- File tersebut adalah file JSON yang valid.
- File tidak berisi kunci asing.
- File berisi `familyName` parameter.
- Setidaknya ada satu entri di bawah`containerDefinitions`.

AWS CloudFormation tumpukan

Perilaku berikut berlaku untuk definisi tugas yang dibuat di konsol Amazon ECS baru sebelum 12 Januari 2023.

Saat Anda membuat definisi tugas, konsol Amazon ECS secara otomatis membuat CloudFormation tumpukan yang memiliki nama yang dimulai`ECS-Console-V2-TaskDefinition-`. Jika Anda menggunakan AWS CLI atau AWS SDK untuk membatalkan pendaftaran definisi tugas, Anda harus menghapus tumpukan definisi tugas secara manual. Untuk informasi selengkapnya, lihat Menghapus tumpukan di Panduan AWS CloudFormation Pengguna.

Definisi tugas yang dibuat setelah 12 Januari 2023, tidak memiliki CloudFormation tumpukan yang dibuat secara otomatis untuk mereka.

Prosedur

Amazon ECS console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Definisi tugas.
3. Pada menu Create new task definition, pilih Create new task definition.
4. Untuk Keluarga penetapan tugas, tentukan nama unik untuk penetapan tugas tersebut.
5. Untuk jenis Peluncuran, pilih lingkungan aplikasi. Default konsol adalah AWS Fargate(yang tanpa server). Amazon ECS menggunakan nilai ini untuk melakukan validasi guna memastikan bahwa parameter definisi tugas valid untuk jenis infrastruktur.
6. Untuk Sistem Operasi/Arsitektur, pilih sistem operasi dan arsitektur CPU untuk tugas tersebut.

   Untuk menjalankan tugas Anda pada arsitektur ARM 64-bit, pilih Linux/ARM64. Untuk informasi selengkapnya, lihat the section called “Platform runtime”.

Untuk menjalankan AWS Fargatetugas Anda di wadah Windows, pilih sistem operasi Windows yang didukung. Untuk informasi selengkapnya, lihat the section called “Sistem Operasi dan Arsitektur”.

7. Untuk ukuran Tugas, pilih CPU dan nilai memori yang akan dicadangkan untuk tugas tersebut. Nilai CPU ditentukan sebagai vCPU dan memori ditentukan sebagai GB.

   Untuk tugas yang dihosting di Fargate, tabel berikut menunjukkan kombinasi CPU dan memori yang valid.

Nilai CPU	Nilai memori	Sistem operasi yang didukung untuk AWS Fargate
256 (.25 vCPU)	512 MiB, 1 GB, 2 GB	Linux
512 (.5 vCPU)	1 GB, 2 GB, 3 GB, 4 GB	Linux
1024 (1 vCPU)	2 GB, 3 GB, 4 GB, 5 GB, 6 GB, 7 GB, 8 GB	Linux, Windows
2048 (2 vCPU)	Antara 4 GB dan 16 GB dalam peningkatan 1 GB	Linux, Windows
4096 (4 vCPU)	Antara 8 GB dan 30 GB dalam peningkatan 1 GB	Linux, Windows
8192 (8 vCPU) **Note** Opsi ini membutuhkan platform Linux `1.4.0` atau yang lebih baru.	Antara 16 GB dan 60 GB dalam peningkatan 4 GB	Linux
16384 (16vCPU)	Antara 32 GB dan 120 GB dalam peningkatan 8 GB	Linux

Nilai CPU	Nilai memori	Sistem operasi yang didukung untuk AWS Fargate
Note Opsi ini membutuhkan platform Linux `1.4.0` atau yang lebih baru.		

Untuk tugas yang dihosting di Amazon EC2, nilai CPU tugas yang didukung adalah antara 128 unit CPU (0,125 vCPU) dan 10240 unit CPU (10 vCPU). Untuk menentukan nilai memori dalam GB, masukkan GB setelah nilai. Misalnya, untuk mengatur nilai Memori ke 3GB, masukkan 3GB.

> **Note**
>
> Tingkat tugas CPU dan memori parameter diabaikan untuk Windows kontainer.

8. Untuk mode Jaringan, pilih mode jaringan yang akan digunakan. Defaultnya adalah mode awsvpc. Untuk informasi selengkapnya, lihat jaringan tugas Amazon ECS.

   Jika Anda memilih jembatan, di bawah pemetaan Port, untuk port Host, masukkan nomor port pada instance kontainer untuk memesan kontainer Anda.

9. (Opsional) Perluas bagian Peran tugas untuk mengonfigurasi peran AWS Identity and Access Management (IAM) untuk tugas:

   a. Untuk peran Tugas, pilih peran IAM yang akan ditetapkan ke tugas. Peran IAM tugas memberikan izin untuk kontainer dalam tugas untuk memanggil operasi AWS API.

   b. Untuk peran eksekusi tugas, pilih peran.

      Untuk informasi tentang kapan menggunakan peran eksekusi tugas, lihatthe section called “Eksekusi tugas peran IAM”. Jika Anda tidak membutuhkan peran tersebut, pilih Tidak Ada.

10. Untuk setiap kontainer untuk menentukan definisi tugas Anda, selesaikan langkah-langkah berikut.

    a. Untuk Nama, masukkan nama untuk wadah.

    b. Untuk URI Gambar, masukkan gambar yang akan digunakan untuk memulai wadah. Gambar dalam registri Galeri Publik Amazon ECR dapat ditentukan dengan menggunakan nama registri Publik Amazon ECR saja. Misalnya, jika `public.ecr.aws/ecs/amazon-ecs-agent:latest` ditentukan, wadah Amazon Linux yang dihosting di Galeri Publik Amazon ECR digunakan. Untuk semua repositori lainnya, tentukan repositori dengan menggunakan format atau. `repository-url/image:tag` `repository-url/image@digest`

    c. Jika gambar Anda berada di registri pribadi di luar Amazon ECR, di bawah Registri pribadi, aktifkan otentikasi registri pribadi. Kemudian, di Secrets Manager ARN atau nama, masukkan Amazon Resource Name (ARN) rahasia.

    d. Untuk wadah Essential, jika definisi tugas Anda memiliki dua atau lebih kontainer yang ditentukan, Anda dapat menentukan apakah penampung harus dianggap penting. Ketika sebuah wadah ditandai sebagai Essential, jika kontainer itu berhenti, maka tugas dihentikan. Setiap definisi tugas harus berisi setidaknya satu wadah penting.

    e. Pemetaan port memungkinkan kontainer untuk mengakses port pada host untuk mengirim atau menerima lalu lintas. Di bawah pemetaan Port, lakukan salah satu hal berikut:

       - Saat Anda menggunakan mode jaringan awsvpc, untuk port Container dan Protocol, pilih pemetaan port yang akan digunakan untuk wadah.
       - Saat Anda menggunakan mode jaringan jembatan, untuk port Container dan Protocol, pilih pemetaan port yang akan digunakan untuk wadah.

       Pilih Tambahkan lebih banyak pemetaan port untuk menentukan pemetaan port kontainer tambahan.

    f. Untuk memberikan akses read-only container ke sistem file root, untuk Read only root file system, pilih Read only.

    g. (Opsional) Untuk menentukan batas CPU, GPU, dan memori tingkat kontainer yang berbeda dari nilai tingkat tugas, di bawah batas alokasi sumber daya, lakukan hal berikut:

       - Untuk CPU, masukkan jumlah unit CPU yang dicadangkan agen kontainer Amazon ECS untuk kontainer.

- Untuk GPU, masukkan jumlah unit GPU untuk instance kontainer.

  Instans Amazon EC2 dengan dukungan GPU memiliki 1 unit GPU untuk setiap GPU. Untuk informasi selengkapnya, lihat the section called “Bekerja dengan GPU di Amazon ECS”.
- Untuk batas keras Memori, masukkan jumlah memori, dalam GB, untuk disajikan ke wadah. Jika wadah mencoba melampaui batas keras, wadah berhenti.
- Daemon Docker 20.10.0 atau yang lebih baru menyimpan minimal 6 mebibytes (MiB) memori untuk wadah, jadi jangan tentukan kurang dari 6 MiB memori untuk wadah Anda.

  Daemon Docker 19.03.13-ce atau sebelumnya menyimpan minimal 4 MiB memori untuk wadah, jadi jangan tentukan kurang dari 4 MiB memori untuk kontainer Anda.
- Untuk batas lunak Memori, masukkan batas lunak (dalam GB) memori untuk cadangan wadah.

  Ketika memori sistem sedang diperdebatkan, Docker upaya untuk menjaga memori kontainer ke batas lunak ini. Jika Anda tidak menentukan memori tingkat tugas, Anda harus menentukan bilangan bulat bukan nol untuk salah satu atau kedua batas keras Memori dan batas lunak Memori. Jika Anda menentukan keduanya, batas keras Memori harus lebih besar dari batas lunak Memori.

  Fitur ini tidak didukung pada wadah Windows.

h. (Opsional) Perluas bagian variabel Lingkungan untuk menentukan variabel lingkungan untuk disuntikkan ke dalam wadah. Anda dapat menentukan variabel lingkungan baik secara individual dengan menggunakan pasangan nilai kunci atau secara massal dengan menentukan file variabel lingkungan yang dihosting di bucket Amazon S3. Untuk informasi tentang cara memformat file variabel lingkungan, lihatGunakan parameter definisi tugas untuk meneruskan variabel lingkungan ke wadah.

i. (Opsional) Pilih opsi Gunakan koleksi log untuk menentukan konfigurasi log. Untuk setiap driver log yang tersedia, ada opsi driver log untuk ditentukan. Opsi default mengirimkan log kontainer ke Amazon CloudWatch Logs. Opsi driver log lainnya dikonfigurasi dengan menggunakan AWS FireLens. Untuk informasi selengkapnya, lihat Menggunakan perutean log khusus.

   Berikut ini menjelaskan setiap tujuan log kontainer secara lebih rinci.

- Amazon CloudWatch — Konfigurasikan tugas untuk mengirim log kontainer ke CloudWatch Log. Opsi driver log default disediakan, yang membuat grup CloudWatch log atas nama Anda. Untuk menentukan nama grup log yang berbeda, ubah nilai opsi driver.
- Ekspor log ke Splunk — Konfigurasikan tugas untuk mengirim log kontainer ke Splunk driver yang mengirim log ke layanan jarak jauh. Anda harus memasukkan URL ke layanan Splunk web Anda. SplunkToken ditetapkan sebagai opsi rahasia karena dapat diperlakukan sebagai data sensitif.
- Ekspor log ke Amazon Data Firehose — Konfigurasikan tugas untuk mengirim log kontainer ke Firehose. Opsi driver log default disediakan, yang mengirimkan log ke aliran pengiriman Firehose. Untuk menentukan nama aliran pengiriman yang berbeda, ubah nilai opsi driver.
- Ekspor log ke Amazon Kinesis Data Streams — Konfigurasikan tugas untuk mengirim log kontainer ke Kinesis Data Streams. Opsi driver log default disediakan, yang mengirim log ke aliran Kinesis Data Streams. Untuk menentukan nama aliran yang berbeda, ubah nilai opsi driver.
- Ekspor log ke Amazon OpenSearch Service — Konfigurasikan tugas untuk mengirim log kontainer ke domain OpenSearch Layanan. Opsi driver log harus disediakan. Untuk informasi selengkapnya, lihat Meneruskan log ke domain Layanan Amazon OpenSearch .
- Ekspor log ke Amazon S3 — Konfigurasikan tugas untuk mengirim log kontainer ke bucket Amazon S3. Opsi driver log default disediakan, tetapi Anda harus menentukan nama bucket Amazon S3 yang valid.

j. (Opsional) Konfigurasikan parameter wadah tambahan.

Untuk mengkonfigurasi opsi ini	Lakukan hal berikut	
Healthcheck Ini adalah perintah yang menentukan apakah sebuah wadah sehat. Untuk informasi selengkapnya, lihat Tentukan kesehatan tugas Amazon ECS menggunakan pemeriksaan kesehatan kontainer.	Perluas HealthCheck, lalu konfigurasikan item berikut: • Untuk Command, masukkan daftar perintah yang dipisahkan koma. Anda dapat memulai perintah dengan `CMD` untuk menjalankan argumen perintah secara langsung, atau `CMD-SHELL` menjalankan perintah dengan shell default container. Jika tidak ada yang `CMD` ditentukan, digunakan. • Untuk Interval, masukkan jumlah detik antara setiap pemeriksaan kesehatan. Nilai yang valid adalah antara 5 dan 30. • Untuk Timeout, masukkan periode waktu (dalam detik) untuk menunggu pemeriksaan kesehatan berhasil sebelum	

Untuk mengkonfigurasi opsi ini	Lakukan hal berikut	
	dianggap gagal. Nilai yang valid adalah antara 2 dan 60. • Untuk periode Mulai, masukkan periode waktu (dalam detik) untuk menunggu wadah untuk bootstrap sebelum perintah pemeriksaan kesehatan berjalan. Nilai yang valid adalah antara 0 dan 300. • Untuk Mencoba lagi, masukkan berapa kali untuk mencoba kembali perintah pemeriksaan kesehatan ketika ada kegagalan. Nilai yang valid adalah antara 1 dan 10.	

Untuk mengkonfigurasi opsi ini	Lakukan hal berikut	
Batas waktu kontainer Opsi ini menentukan kapan harus memulai dan menghentikan wadah.	Perluas batas waktu Kontainer, lalu konfigura sikan yang berikut ini: • Untuk mengonfigurasi waktu menunggu sebelum menyerah pada penyelesaian dependensi untuk wadah, untuk Batas waktu mulai, masukkan jumlah detik. • Untuk mengonfigurasi waktu menunggu sebelum penampung dihentikan jika tidak keluar secara normal dengan sendirinya, untuk Stop timeout, masukkan jumlah detik.	

Untuk mengkonfigurasi opsi ini	Lakukan hal berikut	
Pengaturan jaringan kontainer Opsi ini menentukan apakah akan menggunakan jaringan dalam wadah.	Perluas pengaturan jaringan Container, lalu konfigurasikan yang berikut ini: • Untuk menonaktifkan jaringan kontainer, pilih Matikan jaringan. • Untuk mengkonfigurasi alamat IP server DNS yang disajikan ke wadah, di server DNS, masukkan alamat IP setiap server pada baris terpisah. • Untuk mengonfigurasi domain DNS untuk mencari nama non-fully-qualified host yang disajikan ke wadah, di domain pencarian DNS, masukkan setiap domain pada baris terpisah. Polanya adalah **^[a-zA-Z0-9-.]{0,253}[a-zA-Z0-9]$** . • Untuk mengonfigurasi nama host kontainer, dalam nama Host,	

Untuk mengkonfigurasi opsi ini	Lakukan hal berikut	
	masukkan nama kambing kontainer. • Untuk menambahkan nama host dan pemetaan alamat IP yang ditambahkan ke `/etc/hosts` file pada wadah, pilih Tambahkan host tambahan, lalu untuk Hostname dan alamat IP, masukkan nama host dan alamat IP.	

Untuk mengkonfigurasi opsi ini	Lakukan hal berikut	
Dockerkonfigurasi Ini mengesampingkan nilai-nilai di. Dockerfile	Perluas Dockerkon figurasi, lalu konfigura sikan item berikut: • Untuk Command, masukkan perintah yang dapat dieksekusi untuk sebuah wadah. Parameter ini dipetakan ke `Cmd` bagian Create a container dari Docker Remote API dan `COMMAND` opsi untuk`docker run`. Parameter ini mengesampingkan `CMD` instruksi dalam file. Dockerfile • Untuk titik Masuk, masukkan Docker ENTRYPOINT yang diteruskan ke wadah. Parameter ini dipetakan ke `Entrypoint` bagian Create a container dari Docker Remote API dan `--entrypoint` opsi untuk`docker run`. Parameter ini	

Untuk mengkonfigurasi opsi ini	Lakukan hal berikut	
	mengesampingkan `ENTRYPOINT` instruksi dalam file. Dockerfile • Untuk direktori Kerja, masukkan direktori bahwa wadah akan menjalankan titik masuk dan instruksi perintah yang disediakan. Parameter ini dipetakan ke `WorkingDir` bagian Create a container dari Docker Remote API dan `--workdir` opsi untuk`docker run`. Parameter ini mengesampingkan `WORKDIR` instruksi dalam file. Dockerfile	

Untuk mengkonfigurasi opsi ini	Lakukan hal berikut	
Ulimits Nilai-nilai ini menimpa pengaturan kuota sumber daya default untuk sistem operasi. Parameter ini memetakan ke `Ulimits` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--ulimit` untuk docker run.	Perluas batas Sumber daya (ulimits), lalu pilih Tambah ulimit. Untuk nama Limit, pilih limit. Kemudian, untuk Soft limit dan Hard limit, masukkan nilainya. Untuk menambahkan tambahanulimits, pilih Tambah ulimit.	
Dockerlabel Opsi ini menambahkan metadata ke wadah Anda. Parameter ini memetakan ke `Labels` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--label` untuk docker run.	Perluas Dockerlabel, pilih Tambahkan pasangan nilai kunci, lalu masukkan Kunci dan Nilai. Untuk menambahkan Docker label tambahan, pilih Tambah pasangan nilai kunci.	

Untuk mengkonfigurasi opsi ini	Lakukan hal berikut	
Pesanan startup kontainer Opsi ini mendefinisikan dependensi untuk startup dan shutdown container . Sebuah kontainer dapat berisi beberapa dependensi.	Perluas urutan ketergantungan Startup, lalu konfigurasikan yang berikut ini: a. Pilih Tambahkan ketergantungan kontainer. b. Untuk Kontainer, pilih wadahnya. c. Untuk Kondisi, pilih kondisi ketergantungan startup. Untuk menambahkan dependensi tambahan, pilih Tambahkan ketergantungan kontainer.	

k. (Opsional) Pilih Tambahkan lebih banyak kontainer untuk menambahkan wadah tambahan ke definisi tugas.

11. (Opsional) Bagian Penyimpanan digunakan untuk memperluas jumlah penyimpanan sementara untuk tugas yang dihosting di Fargate. Anda juga dapat menggunakan bagian ini untuk menambahkan konfigurasi volume data untuk tugas tersebut.

    - Untuk memperluas penyimpanan sementara yang tersedia di luar nilai default 20 gibibytes (GiB) untuk tugas Fargate Anda, untuk Jumlah, masukkan nilai hingga. 200 GiB

12. (Opsional) Untuk menambahkan konfigurasi volume data untuk definisi tugas, pilih Tambahkan volume, lalu ikuti langkah-langkah ini.

    a. Untuk nama Volume, masukkan nama untuk volume data. Nama volume data digunakan saat membuat titik mount container.

b. Untuk konfigurasi Volume, pilih apakah Anda ingin mengonfigurasi volume saat membuat definisi tugas atau selama penerapan.

> **Note**
>
> Volume yang dapat dikonfigurasi saat membuat definisi tugas termasuk Bind mount, Docker Amazon EFS, dan Amazon FSx for Windows File Server. Volume yang dapat dikonfigurasi saat penerapan saat menjalankan tugas, atau saat membuat atau memperbarui layanan menyertakan Amazon EBS.

c. Untuk tipe Volume, pilih tipe volume yang kompatibel dengan tipe konfigurasi yang Anda pilih, lalu konfigurasikan jenis volume.

Tipe volume	Langkah-langkah	
Bind mount	a. Pilih Tambahkan titik pemasangan, lalu konfigurasikan yang berikut ini: • Untuk Container, pilih wadah untuk titik pemasangan. • Untuk volume Sumber, pilih volume data yang akan dipasang ke wadah. • Untuk jalur Kontainer , masukkan jalur pada wadah untuk memasang volume. • Untuk Read only, pilih apakah container memiliki akses read-only ke volume. b. Untuk menambahk an titik pemasangan tambahan, Tambahkan titik pemasangan.	

Tipe volume	Langkah-langkah	
EFS	a. Untuk ID sistem File, pilih ID sistem file Amazon EFS. b. (Opsional) Untuk Root directory (Direktor i asal), masukkan direktori dalam sistem file Amazon EFS untuk pemasangan sebagai direktori asal di dalam host. Jika parameter ini diabaikan, asal volume Amazon EFS akan digunakan. Jika Anda berencana untuk menggunakan titik akses EFS, biarkan bidang ini kosong. c. (Opsional) Untuk titik Akses, pilih ID titik akses yang akan digunakan. d. (Opsional) Untuk mengenkripsi data antara sistem file Amazon EFS dan host Amazon ECS atau untuk menggunak an peran eksekusi tugas saat memasang volume, pilih Konfigurasi lanjutan,	

Tipe volume	Langkah-langkah	
	lalu konfigurasikan yang berikut ini: • Untuk mengenkripsi data antara sistem file Amazon EFS dan host Amazon ECS, pilih Enkripsi Transit, lalu untuk Port, masukkan port yang akan digunakan saat mengirim data terenkripsi antara host Amazon ECS dan server Amazon EFS. Jika Anda tidak menentukan port enkripsi transit, strategi pemilihan port yang digunakan oleh pembantu pemasangan Amazon EFS akan digunakan. Untuk informasi lebih lanjut, lihat Pembantu Pemasangan EFS di Panduan Pengguna Amazon Elastic File System. • Untuk menggunakan peran IAM tugas Amazon ECS yang ditentukan dalam definisi tugas saat	

Tipe volume	Langkah-langkah	
	memasang sistem file Amazon EFS, pilih otorisasi IAM. e. Pilih Tambahkan titik pemasangan, lalu konfigurasikan yang berikut ini: • Untuk Container, pilih wadah untuk titik pemasangan. • Untuk volume Sumber, pilih volume data yang akan dipasang ke wadah. • Untuk jalur Kontainer , masukkan jalur pada wadah untuk memasang volume. • Untuk Read only, pilih apakah container memiliki akses read-only ke volume. f. Untuk menambahk an titik pemasangan tambahan, Tambahkan titik pemasangan.	

Tipe volume	Langkah-langkah	
Docker	a. Untuk Driver, masukkan konfigurasi Docker volume. Wadah Windows hanya mendukung penggunaan driver lokal. Untuk menggunakan bind mount, tentukan host. b. Untuk Lingkup, pilih siklus hidup volume. • Untuk memiliki siklus hidup terakhir ketika tugas dimulai dan berhenti, pilih Tugas. • Agar volume tetap ada setelah tugas berhenti, pilih Dibagikan. c. Pilih Tambahkan titik pemasangan, lalu konfigurasikan yang berikut ini: • Untuk Container, pilih wadah untuk titik pemasangan. • Untuk volume Sumber, pilih volume data yang akan dipasang ke wadah.	

Tipe volume	Langkah-langkah	
	• Untuk jalur Kontainer , masukkan jalur pada wadah untuk memasang volume. • Untuk Read only, pilih apakah container memiliki akses read-only ke volume. d. Untuk menambahkan titik pemasangan tambahan, Tambahkan titik pemasangan.	

Tipe volume	Langkah-langkah	
fsX for Windows File Server	a. Untuk ID sistem File, pilih ID sistem file FSx for Windows File Server. b. Untuk direktori Root, masukkan direktori , masukkan direktori dalam sistem file FSx for Windows File Server untuk dipasang sebagai direktori root di dalam host. c. Untuk parameter Credential, pilih bagaimana kredensyal disimpan. • Untuk menggunak an AWS Secrets Manager, masukkan Nama Sumber Daya Amazon (ARN) dari rahasia Secrets Manager. • Untuk menggunak an AWS Systems Manager, masukkan Amazon Resource Name (ARN) dari parameter Systems Manager.	

Tipe volume	Langkah-langkah	
	d. Untuk Domain, masukkan nama domain yang memenuhi syarat yang di-host oleh direktori AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD) atau Direktori Aktif EC2 yang dihosting sendiri. e. Pilih Tambahkan titik pemasangan, lalu konfigurasikan yang berikut ini: • Untuk Container, pilih wadah untuk titik pemasangan. • Untuk volume Sumber, pilih volume data yang akan dipasang ke wadah. • Untuk jalur Kontainer , masukkan jalur pada wadah untuk memasang volume. • Untuk Read only, pilih apakah container memiliki akses read-only ke volume.	

Tipe volume	Langkah-langkah	
	f. Untuk menambahkan titik pemasangan tambahan, Tambahkan titik pemasangan.	

Tipe volume	Langkah-langkah	
Volume Amazon EBS	a. Pilih Tambahkan titik pemasangan, lalu konfigurasikan yang berikut ini: • Untuk Container, pilih wadah untuk titik pemasangan. • Untuk volume Sumber, pilih volume data yang akan dipasang ke wadah. • Untuk jalur Kontainer , masukkan jalur pada wadah untuk memasang volume. • Untuk Read only, pilih apakah container memiliki akses read-only ke volume. b. Untuk menambahk an titik pemasangan tambahan, Tambahkan titik pemasangan.	

13. Untuk menambahkan volume dari wadah lain, pilih Tambahkan volume dari, lalu konfigurasikan yang berikut ini:

    • Untuk Kontainer, pilih wadahnya.

- Untuk Sumber, pilih wadah yang memiliki volume yang ingin Anda pasang.
- Untuk Read only, pilih apakah container memiliki akses read-only ke volume.

14. (Opsional) Untuk mengonfigurasi pengaturan penelusuran aplikasi dan pengumpulan metrik dengan menggunakan AWS Distro for OpenTelemetry integrasi, perluas Pemantauan, lalu pilih Gunakan koleksi metrik untuk mengumpulkan dan mengirim metrik tugas Anda ke Amazon CloudWatch atau Amazon Managed Service for Prometheus. Saat opsi ini dipilih, Amazon ECS membuat sespan AWS Distro for OpenTelemetry kontainer yang telah dikonfigurasi sebelumnya untuk mengirim metrik aplikasi. Untuk informasi selengkapnya, lihat Korelasikan kinerja aplikasi Amazon ECS menggunakan metrik aplikasi.

    a. Saat Amazon CloudWatch dipilih, metrik aplikasi kustom Anda dirutekan ke CloudWatch metrik khusus. Untuk informasi selengkapnya, lihat Mengekspor metrik aplikasi ke Amazon CloudWatch.

    > **Important**
    >
    > Saat mengekspor metrik aplikasi ke Amazon CloudWatch, definisi tugas Anda memerlukan peran IAM tugas dengan izin yang diperlukan. Untuk informasi selengkapnya, lihat Izin IAM yang diperlukan untuk AWS Distro untuk OpenTelemetry integrasi dengan Amazon CloudWatch.

    b. Saat Anda memilih Amazon Managed Service for Prometheus (instrumentasi pustaka Prometheus), CPU tingkat tugas, memori, jaringan, dan metrik penyimpanan serta metrik aplikasi kustom Anda dirutekan ke Amazon Managed Service untuk Prometheus. Untuk titik akhir penulisan jarak jauh Workspace, masukkan URL titik akhir penulisan jarak jauh untuk ruang kerja Anda. Prometheus Untuk target Scraping, masukkan host dan port yang dapat digunakan AWS Distro for OpenTelemetry kolektor untuk mengikis data metrik. Untuk informasi selengkapnya, lihat Mengekspor metrik aplikasi ke Amazon Managed Service untuk Prometheus.

    > **Important**
    >
    > Saat mengekspor metrik aplikasi ke Amazon Managed Service untuk Prometheus, definisi tugas Anda memerlukan peran IAM tugas dengan izin yang diperlukan. Untuk informasi selengkapnya, lihat Izin IAM yang diperlukan untuk

AWS Distro untuk integrasi OpenTelemetry dengan Amazon Managed Service untuk Prometheus.

c. Saat Anda memilih Amazon Managed Service for Prometheus OpenTelemetry (instrumentasi), CPU tingkat tugas, memori, jaringan, dan metrik penyimpanan serta metrik aplikasi kustom Anda dirutekan ke Amazon Managed Service for Prometheus. Untuk titik akhir penulisan jarak jauh Workspace, masukkan URL titik akhir penulisan jarak jauh untuk ruang kerja Anda. Prometheus Untuk informasi selengkapnya, lihat Mengekspor metrik aplikasi ke Amazon Managed Service untuk Prometheus.

> **Important**
>
> Saat mengekspor metrik aplikasi ke Amazon Managed Service untuk Prometheus, definisi tugas Anda memerlukan peran IAM tugas dengan izin yang diperlukan. Untuk informasi selengkapnya, lihat Izin IAM yang diperlukan untuk AWS Distro untuk integrasi OpenTelemetry dengan Amazon Managed Service untuk Prometheus.

15. (Opsional) Perluas bagian Tag untuk menambahkan tag, sebagai pasangan nilai kunci, ke definisi tugas.

    - [Tambahkan tag] Pilih Tambah tag, lalu lakukan hal berikut:
      - Untuk Kunci, masukkan nama kunci.
      - Untuk Nilai, masukkan nilai kunci.
    - [Menghapus tanda] Di samping tanda, pilih Hapus tanda.

16. Pilih Buat untuk mendaftarkan definisi tugas.

Amazon ECS console JSON editor

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Definisi tugas.
3. Pada menu Create new task definition, pilih Create new task definition with JSON.
4. Di kotak editor JSON, edit file JSON Anda,

   JSON harus lulus pemeriksaan validasi yang ditentukan dalam. the section called “Validasi JSON”

5. Pilih Buat.

Memperbarui definisi tugas menggunakan konsol

Revisi definisi tugas adalah salinan dari definisi tugas saat ini dengan nilai parameter baru menggantikan yang sudah ada. Semua parameter yang tidak Anda modifikasi ada dalam revisi baru.

Untuk memperbarui ketentuan tugas, membuat revisi ketentuan tugas. Jika ketentuan tugas yang digunakan dalam layanan, Anda harus memperbarui layanan tersebut untuk menggunakan ketentuan tugas yang diperbarui.

Saat Anda membuat revisi, Anda dapat memodifikasi properti kontainer berikut dan properti lingkungan.

- URI gambar kontainer
- Pemetaan pelabuhan
- Variabel-variabel lingkungan
- Ukuran tugas
- Ukuran kontainer
- role tugas
- Peran pelaksanaan tugas
- Volume dan titik pemasangan kontainer
- Registri pribadi

Validasi JSON

Editor JSON konsol Amazon ECS memvalidasi hal berikut dalam file JSON:

- File tersebut adalah file JSON yang valid
- File tidak berisi kunci asing
- File berisi `familyName` parameter
- Setidaknya ada satu entri di bawah `containerDefinitions`

Amazon ECS console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah yang berisi ketentuan tugas Anda.
3. Di panel navigasi, pilih Definisi tugas.
4. Pilih definisi tugas.
5. Pilih revisi definisi tugas, lalu pilih Buat revisi baru, Buat revisi baru.
6. Pada halaman Buat revisi definisi tugas baru, buat perubahan. Misalnya, untuk mengubah definisi kontainer yang ada (seperti gambar kontainer, batas memori, atau pemetaan port), pilih wadah, lalu buat perubahan.
7. Verifikasi informasi, lalu pilih Perbarui.
8. Jika ketentuan tugas Anda digunakan dalam layanan, perbarui layanan Anda dengan ketentuan tugas yang diperbarui. Untuk informasi selengkapnya, lihat Memperbarui layanan menggunakan konsol.

Amazon ECS console JSON editor

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Definisi tugas.
3. Pilih Buat revisi baru, Buat revisi baru dengan JSON.
4. Di kotak editor JSON, edit file JSON Anda,

   JSON harus lulus pemeriksaan validasi yang ditentukan dalam. the section called “Validasi JSON”
5. Pilih Buat.

Membatalkan pendaftaran revisi definisi tugas menggunakan konsol

Bila Anda tidak lagi memerlukan revisi definisi tugas tertentu di Amazon ECS, Anda dapat membatalkan pendaftaran revisi definisi tugas sehingga tidak lagi ditampilkan dalam panggilan `ListTaskDefinition` API atau di konsol saat Anda ingin menjalankan tugas atau memperbarui layanan.

Ketika Anda membatalkan pendaftaran revisi ketentuan tugas, segera ditandai sebagai `INACTIVE`. Tugas dan layanan yang ada yang mereferensikan tugas revisi definisi `INACTIVE` terus berjalan tanpa gangguan. Layanan yang ada yang mereferensikan revisi ketentuan tugas `INACTIVE` masih dapat menaikkan atau menurunkan skala dengan mengubah jumlah layanan yang diinginkan.

Anda tidak dapat menggunakan revisi ketentuan tugas `INACTIVE` untuk menjalankan tugas baru atau membuat layanan baru. Anda juga tidak dapat memperbarui layanan yang sudah ada untuk mereferensikan revisi ketentuan tugas `INACTIVE` (meskipun mungkin ada jendela hingga 10 menit setelah pencabutan pendaftaran tempat pembatasan ini belum berlaku).

> **Note**
>
> Saat Anda membatalkan pendaftaran semua revisi dalam keluarga tugas, keluarga definisi tugas dipindahkan ke daftar. `INACTIVE` Menambahkan revisi baru definisi `INACTIVE` tugas memindahkan keluarga definisi tugas kembali ke `ACTIVE` daftar.
> Pada saat ini, revisi ketentuan tugas `INACTIVE` tetap dapat ditemukan di akun Anda tanpa batas waktu. Namun, perilaku ini dapat berubah di masa mendatang. Oleh karena itu, Anda tidak boleh mengandalkan revisi ketentuan tugas `INACTIVE` yang bertahan di luar siklus hidup tugas dan layanan terkait apa pun.

AWS CloudFormation tumpukan

Perilaku berikut berlaku untuk definisi tugas yang dibuat di konsol Amazon ECS baru sebelum 12 Januari 2023.

Saat Anda membuat definisi tugas, konsol Amazon ECS secara otomatis membuat CloudFormation tumpukan yang memiliki nama yang dimulai`ECS-Console-V2-TaskDefinition-`. Jika Anda menggunakan AWS CLI atau AWS SDK untuk membatalkan pendaftaran definisi tugas, Anda harus menghapus tumpukan definisi tugas secara manual. Untuk informasi selengkapnya, lihat Menghapus tumpukan di Panduan AWS CloudFormation Pengguna.

Definisi tugas yang dibuat setelah 12 Januari 2023, tidak memiliki CloudFormation tumpukan yang dibuat secara otomatis untuk mereka.

Untuk membatalkan pendaftaran definisi tugas baru (konsol Amazon ECS)

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih wilayah yang berisi ketentuan tugas Anda.

3. Di panel navigasi, pilih Definisi tugas.
4. Pada halaman Definisi tugas, pilih keluarga definisi tugas yang berisi satu atau beberapa revisi yang ingin Anda deregister.
5. Pada halaman Nama definisi tugas, pilih revisi yang akan dihapus, lalu pilih Tindakan, Deregister.
6. Verifikasi informasi di jendela Deregister, lalu pilih Deregister untuk menyelesaikan.

Menghapus revisi definisi tugas menggunakan konsol

Saat tidak lagi memerlukan revisi definisi tugas tertentu di Amazon ECS, Anda dapat menghapus revisi definisi tugas.

Saat Anda menghapus revisi definisi tugas, itu segera bertransisi dari ke. `INACTIVE` `DELETE_IN_PROGRESS` Tugas dan layanan yang ada yang mereferensikan revisi definisi `DELETE_IN_PROGRESS` tugas terus berjalan tanpa gangguan.

Anda tidak dapat menggunakan revisi definisi `DELETE_IN_PROGRESS` tugas untuk menjalankan tugas baru atau membuat layanan baru. Anda juga tidak dapat memperbarui layanan yang ada untuk mereferensikan revisi definisi `DELETE_IN_PROGRESS` tugas.

Saat Anda menghapus semua revisi definisi `INACTIVE` tugas, nama definisi tugas tidak ditampilkan di konsol dan tidak ditampilkan di API. Jika revisi definisi tugas dalam `DELETE_IN_PROGRESS` status, nama definisi tugas ditampilkan di konsol dan dikembalikan di API. Nama definisi tugas dipertahankan oleh Amazon ECS dan revisi akan bertambah saat berikutnya Anda membuat definisi tugas dengan nama itu.

Sumber daya Amazon ECS yang dapat memblokir penghapusan

Permintaan penghapusan definisi tugas tidak akan selesai ketika ada sumber daya Amazon ECS yang bergantung pada revisi definisi tugas. Sumber daya berikut mungkin mencegah definisi tugas dihapus:

- Tugas Amazon ECS - Definisi tugas diperlukan agar tugas tetap sehat.
- Penerapan dan set tugas Amazon ECS - Definisi tugas diperlukan saat peristiwa penskalaan dimulai untuk penyebaran atau set tugas Amazon ECS.

Jika definisi tugas Anda tetap dalam `DELETE_IN_PROGRESS` status, Anda dapat menggunakan konsol, atau AWS CLI untuk mengidentifikasi, dan kemudian menghentikan sumber daya yang memblokir penghapusan definisi tugas.

Penghapusan definisi tugas setelah sumber daya yang diblokir dihapus

Aturan berikut berlaku setelah Anda menghapus sumber daya yang memblokir penghapusan definisi tugas:

- Tugas Amazon ECS - Penghapusan definisi tugas dapat memakan waktu hingga 1 jam untuk diselesaikan setelah tugas dihentikan.
- Penyebaran dan set tugas Amazon ECS - Penghapusan definisi tugas dapat memakan waktu hingga 24 jam untuk diselesaikan setelah penerapan atau set tugas dihapus.

Untuk menghapus definisi tugas (konsol Amazon ECS)

Anda harus membatalkan pendaftaran revisi definisi tugas sebelum Anda menghapusnya. Untuk informasi selengkapnya, lihat the section called “Membatalkan pendaftaran revisi definisi tugas menggunakan konsol”.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih wilayah yang berisi ketentuan tugas Anda.
3. Di panel navigasi, pilih Definisi tugas.
4. Pada halaman Definisi tugas, pilih keluarga definisi tugas yang berisi satu atau beberapa revisi yang ingin Anda hapus.
5. Pada halaman Nama definisi tugas, pilih revisi yang akan dihapus, lalu pilih Tindakan, Hapus.
6. Verifikasi informasi di kotak Hapus konfirmasi, lalu pilih Hapus untuk menyelesaikan.

Kasus penggunaan definisi tugas

Pelajari lebih lanjut tentang cara menulis definisi tugas untuk berbagai AWS layanan dan fitur.

Bergantung pada beban kerja Anda, ada parameter definisi tugas tertentu yang perlu diatur. Juga untuk jenis peluncuran EC2, Anda harus memilih instance tertentu yang bermesin untuk beban kerja.

Topik

- Bekerja dengan GPU di Amazon ECS

- Menggunakan transcoding video di Amazon ECS
- Menggunakan AWS Neuron di Amazon Linux 2 di Amazon ECS
- Menggunakan instans DL1 pembelajaran mendalam di Amazon ECS
- Bekerja dengan beban kerja ARM 64-bit di Amazon ECS
- Menggunakan driver log awslogs
- Menggunakan perutean log khusus
- Autentikasi registri privat untuk tugas
- Gunakan parameter definisi tugas untuk meneruskan variabel lingkungan ke wadah
- Gunakan file untuk meneruskan variabel lingkungan ke wadah
- Meneruskan data sensitif ke wadah

Bekerja dengan GPU di Amazon ECS

Amazon ECS mendukung beban kerja yang menggunakan GPU, saat Anda membuat cluster dengan instance container yang mendukung GPU. Instans container berbasis GPU Amazon EC2 yang menggunakan tipe instans p2, p3, p5, g3, g4, dan g5 menyediakan akses ke GPU NVIDIA. Untuk informasi selengkapnya, lihat Instans Komputasi Akselerasi Linux di Panduan Pengguna Amazon EC2 untuk Instans Linux.

Amazon ECS menyediakan AMI yang dioptimalkan untuk GPU yang dilengkapi dengan driver kernel NVIDIA yang telah dikonfigurasi sebelumnya dan runtime GPU Docker. Untuk informasi selengkapnya, lihat AMI Amazon ECS yang dioptimalkan.

Anda dapat menunjuk sejumlah GPU dalam definisi tugas Anda untuk pertimbangan penempatan tugas di tingkat kontainer. Amazon ECS menjadwalkan instans kontainer yang tersedia yang mendukung GPU dan menyematkan GPU fisik ke wadah yang tepat untuk kinerja optimal.

Berikut tipe instans Amazon EC2 berbasis GPU yang didukung. Untuk informasi selengkapnya, lihat Instans Amazon EC2 P2, Instans P3 AmazonEC2, Instans P4d Amazon EC2, InstansAmazon EC2 P5, Instans Amazon EC2 G3, Instans Amazon EC2 G4, dan Instans Amazon EC2 G5.

Jenis instans	GPU	Memori GPU (GiB)	vCPU	Memori (GiB)
p3.2xlarge	1	16	8	61

Jenis instans	GPU	Memori GPU (GiB)	vCPU	Memori (GiB)
p3.8xlarge	4	64	32	244
p3.16xlarge	8	128	64	488
p3dn.24xlarge	8	256	96	768
p4d.24xlarge	8	320	96	1152
p5.48xlarge	8	640	192	2048
g3s.xlarge	1	8	4	30,5
g3.4xlarge	1	8	16	122
g3.8xlarge	2	16	32	244
g3.16xlarge	4	32	64	488
g4dn.xlarge	1	16	4	16
g4dn.2xlarge	1	16	8	32
g4dn.4xlarge	1	16	16	64
g4dn.8xlarge	1	16	32	128
g4dn.12xlarge	4	64	48	192
g4dn.16xlarge	1	16	64	256
g5.xlarge	1	24	4	16
g5.2xlarge	1	24	8	32
g5.4xlarge	1	24	16	64
g5.8xlarge	1	24	32	128
g5.16xlarge	1	24	64	256

Jenis instans	GPU	Memori GPU (GiB)	vCPU	Memori (GiB)
g5.12xlarge	4	96	48	192
g5.24xlarge	4	96	96	384
g5.48xlarge	8	192	192	768

Anda dapat mengambil ID Amazon Machine Image (AMI) untuk AMI yang dioptimalkan Amazon ECS dengan menanyakan API Parameter Store. AWS Systems Manager Dengan menggunakan parameter ini, Anda tidak perlu mencari ID AMI yang dioptimalkan Amazon ECS secara manual. Untuk informasi selengkapnya tentang Systems Manager Parameter Store API, lihat GetParameter. Pengguna yang Anda gunakan harus memiliki izin `ssm:GetParameter` IAM untuk mengambil metadata AMI Amazon ECS yang dioptimalkan.

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/gpu/
recommended --region us-east-1
```

Pertimbangan

> **Note**
>
> Dukungan untuk tipe keluarga instans g2 telah usang. Jenis keluarga instans g2 hanya didukung pada versi yang lebih awal dari AMI yang `20230906` dioptimalkan untuk GPU Amazon ECS.
> Jenis keluarga instans p2 hanya didukung pada versi yang lebih awal dari AMI yang `20230912` dioptimalkan untuk GPU Amazon ECS. Jika Anda perlu terus menggunakan instance p2, lihat. Apa yang harus dilakukan jika Anda membutuhkan instance P2
> Pembaruan driver NVIDIA/CUDA di tempat pada kedua jenis keluarga instance ini akan menyebabkan potensi kegagalan beban kerja GPU.

Kami menyarankan Anda mempertimbangkan hal berikut sebelum Anda mulai bekerja dengan GPU di Amazon ECS.

- Klaster Anda dapat terdiri dari campuran GPU dan instans kontainer non-GPU.

- Anda dapat menjalankan beban kerja GPU pada instance eksternal. Saat mendaftarkan instance eksternal dengan cluster Anda, pastikan `--enable-gpu` flag disertakan pada skrip instalasi. Untuk informasi selengkapnya, lihat Pendaftaran instans eksternal ke sebuah klaster.
- Anda harus mengatur `ECS_ENABLE_GPU_SUPPORT` ke `true` dalam file konfigurasi agen Anda. Untuk informasi selengkapnya, lihat the section called “Konfigurasi agen kontainer”.
- Saat menjalankan tugas atau membuat layanan, Anda dapat menggunakan atribut tipe instance saat mengonfigurasi batasan penempatan tugas untuk menentukan instance kontainer tempat tugas akan diluncurkan. Dengan melakukan hal tersebut, Anda bisa lebih efektif menggunakan sumber daya Anda. Untuk informasi selengkapnya, lihat Bagaimana Amazon ECS menempatkan tugas pada instans kontainer.

  Contoh berikut meluncurkan tugas pada instans kontainer `g4dn.xlarge` di klaster default Anda.

  ```
  aws ecs run-task --cluster default --task-definition ecs-gpu-task-def \
       --placement-constraints type=memberOf,expression="attribute:ecs.instance-type ==
    g4dn.xlarge" --region us-east-2
  ```

- Untuk setiap kontainer yang memiliki persyaratan sumber daya GPU yang ditentukan dalam definisi kontainer, Amazon ECS menetapkan runtime kontainer menjadi runtime kontainer NVIDIA.
- Runtime kontainer NVIDIA membutuhkan beberapa variabel lingkungan untuk disetel dalam wadah agar berfungsi dengan baik. Untuk daftar variabel lingkungan ini, lihat Konfigurasi Khusus dengan Docker. Amazon ECS menetapkan nilai variabel `NVIDIA_VISIBLE_DEVICES` lingkungan menjadi daftar ID perangkat GPU yang ditetapkan Amazon ECS ke penampung. Untuk variabel lingkungan lain yang diperlukan, Amazon ECS tidak mengaturnya. Jadi, pastikan image container Anda menyetelnya atau disetel dalam definisi container.
- Keluarga tipe instans p5 didukung pada versi `20230929` dan yang lebih baru dari AMI yang dioptimalkan untuk GPU Amazon ECS.
- Keluarga tipe instans g4 didukung pada versi `20230913` dan yang lebih baru dari AMI yang dioptimalkan untuk GPU Amazon ECS. Untuk informasi selengkapnya, lihat AMI Amazon ECS yang dioptimalkan. Ini tidak didukung dalam alur kerja Create Cluster di konsol Amazon ECS. Untuk menggunakan jenis instans ini, Anda harus menggunakan konsol Amazon EC2 AWS CLI, atau API dan mendaftarkan instans secara manual ke cluster Anda.
- Tipe instans p4d.24xlarge hanya bekerja dengan CUDA 11 atau yang lebih baru.
- AMI yang dioptimalkan untuk GPU Amazon ECS mengaktifkan IPv6, yang menyebabkan masalah saat menggunakan. `yum` Hal ini dapat diatasi dengan mengonfigurasi `yum` untuk menggunakan IPv4 dengan perintah berikut.

```
echo "ip_resolve=4" >> /etc/yum.conf
```

- Saat Anda membuat gambar kontainer yang tidak menggunakan gambar dasar NVIDIA/CUDA, Anda harus menyetel variabel runtime `NVIDIA_DRIVER_CAPABILITIES` kontainer ke salah satu nilai berikut:
  - `utility,compute`
  - `all`

  Untuk informasi tentang cara mengatur variabel, lihat Mengontrol Runtime Kontainer NVIDIA di situs web NVIDIA.
- GPU tidak didukung pada wadah Windows.

Luncurkan instance kontainer GPU

Untuk menggunakan instance GPU di Amazon ECS, Anda perlu membuat template peluncuran, file data pengguna, dan meluncurkan instance.

Anda kemudian dapat menjalankan tugas yang menggunakan definisi tugas yang dikonfigurasi untuk GPU.

Menggunakan templat peluncuran

Anda dapat membuat template peluncuran.

- Buat template peluncuran yang menggunakan ID AMI GPU Amazon ECS yang dioptimalkan Untuk AMI. Untuk informasi tentang cara membuat template peluncuran, lihat Membuat template peluncuran baru menggunakan parameter yang Anda tentukan di Panduan Pengguna Amazon EC2 untuk Instans Linux.

  Gunakan ID AMI dari langkah sebelumnya untuk image Amazon Machine. Untuk informasi tentang cara menentukan ID AMI dengan parameter Systems Manager, lihat Menentukan parameter Systems Manager dalam template peluncuran di Panduan Pengguna Amazon EC2 untuk Instans Linux.

  Tambahkan yang berikut ini ke data Pengguna di template peluncuran. Ganti *`nama cluster dengan nama`* cluster Anda.

  ```
  #!/bin/bash
  ```

```
echo ECS_CLUSTER=cluster-name >> /etc/ecs/ecs.config;
echo ECS_ENABLE_GPU_SUPPORT=true >> /etc/ecs/ecs.config
```

Gunakan AWS CLI

Anda dapat menggunakan AWS CLI untuk meluncurkan instance container.

1. Buat file yang dipanggil`userdata.toml`. File ini digunakan untuk data pengguna contoh. Ganti *nama cluster dengan nama* cluster Anda.

   ```
   #!/bin/bash
   echo ECS_CLUSTER=cluster-name >> /etc/ecs/ecs.config;
   echo ECS_ENABLE_GPU_SUPPORT=true >> /etc/ecs/ecs.config
   ```

2. Jalankan perintah berikut untuk mendapatkan ID AMI GPU. Anda menggunakan ini pada langkah berikut.

   ```
   aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/gpu/
   recommended --region us-east-1
   ```

3. Jalankan perintah berikut untuk meluncurkan instance GPU. Ingatlah untuk mengganti parameter berikut:

   - Ganti *subnet* dengan ID subnet pribadi atau publik tempat instans Anda akan diluncurkan.
   - *Ganti gpu_ami* dengan ID AMI dari langkah sebelumnya.
   - Ganti *t3.large* dengan tipe instance yang ingin Anda gunakan.
   - Ganti *wilayah* dengan kode Wilayah.

   ```
   aws ec2 run-instances --key-name ecs-gpu-example \
      --subnet-id subnet \
      --image-id gpu_ami \
      --instance-type t3.large \
      --region region \
      --tag-specifications 'ResourceType=instance,Tags=[{Key=GPU,Value=example}]' \
      --user-data file://userdata.toml \
      --iam-instance-profile Name=ecsInstanceRole
   ```

4. Jalankan perintah berikut untuk memverifikasi bahwa instance container terdaftar ke cluster. Saat Anda menjalankan perintah ini, ingatlah untuk mengganti parameter berikut:

- Ganti *cluster* dengan nama cluster Anda.
- Ganti *wilayah* dengan kode Wilayah Anda.

```
aws ecs list-container-instances --cluster cluster-name --region region
```

Menentukan GPU dalam ketentuan tugas Anda

Untuk menggunakan GPU pada instance container dan runtime GPU Docker, pastikan Anda menetapkan jumlah GPU yang dibutuhkan container Anda dalam definisi tugas. Saat wadah yang mendukung GPU ditempatkan, agen kontainer Amazon ECS menyematkan jumlah GPU fisik yang diinginkan ke wadah yang sesuai. Jumlah GPU yang dicadangkan untuk semua kontainer dalam tugas tidak dapat melebihi jumlah GPU yang tersedia pada instance container tempat tugas diluncurkan. Untuk informasi selengkapnya, lihat Membuat definisi tugas menggunakan konsol.

> **Important**
>
> Jika persyaratan GPU Anda tidak ditentukan dalam ketentuan tugas, tugas akan menggunakan waktu aktif Docker default.

Berikut ini menunjukkan format JSON untuk persyaratan GPU dalam ketentuan tugas:

```
{
  "containerDefinitions": [
     {
        ...
        "resourceRequirements" : [
            {
               "type" : "GPU",
               "value" : "2"
            }
        ],
     },
...
}
```

Contoh berikut menunjukkan sintaksis untuk kontainer Docker yang menentukan persyaratan GPU. Wadah ini menggunakan dua GPU, menjalankan `nvidia-smi` utilitas, dan kemudian keluar.

```
{
  "containerDefinitions": [
    {
      "memory": 80,
      "essential": true,
      "name": "gpu",
      "image": "nvidia/cuda:11.0.3-base",
      "resourceRequirements": [
         {
           "type":"GPU",
           "value": "2"
         }
      ],
      "command": [
        "sh",
        "-c",
        "nvidia-smi"
      ],
      "cpu": 100
    }
  ],
  "family": "example-ecs-gpu"
}
```

Apa yang harus dilakukan jika Anda membutuhkan instance P2

Jika Anda perlu menggunakan instance P2, Anda dapat menggunakan salah satu opsi berikut untuk terus menggunakan instance.

Anda harus memodifikasi data pengguna instance untuk kedua opsi. Untuk informasi selengkapnya, lihat Bekerja dengan data pengguna instans di Panduan Pengguna Amazon EC2 untuk Instans Linux.

Gunakan AMI yang dioptimalkan GPU terakhir yang didukung

Anda dapat menggunakan `20230906` versi AMI yang dioptimalkan untuk GPU, dan menambahkan yang berikut ini ke data pengguna instance.

Ganti nama cluster dengan nama cluster Anda.

```
#!/bin/bash
echo "exclude=*nvidia* *cuda*" >> /etc/yum.conf
echo "ECS_CLUSTER=cluster-name" >> /etc/ecs/ecs.config
```

Gunakan AMI terbaru yang dioptimalkan untuk GPU, dan perbarui data pengguna

Anda dapat menambahkan berikut ini ke data pengguna instance. Ini menghapus instalasi driver Nvidia 535/Cuda12.2, dan kemudian menginstal driver Nvidia 470/Cuda11.4 dan memperbaiki versi.

```
#!/bin/bash
yum remove -y cuda-toolkit* nvidia-driver-latest-dkms*
tmpfile=$(mktemp)
cat >$tmpfile <<EOF
[amzn2-nvidia]
name=Amazon Linux 2 Nvidia repository
mirrorlist=\$awsproto://\$amazonlinux.\$awsregion.\$awsdomain/\$releasever/amzn2-
nvidia/latest/\$basearch/mirror.list
priority=20
gpgcheck=1
gpgkey=https://developer.download.nvidia.com/compute/cuda/repos/rhel7/
x86_64/7fa2af80.pub
enabled=1
exclude=libglvnd-*
EOF

mv $tmpfile /etc/yum.repos.d/amzn2-nvidia-tmp.repo
yum install -y system-release-nvidia cuda-toolkit-11-4 nvidia-driver-latest-
dkms-470.182.03
yum install -y libnvidia-container-1.4.0 libnvidia-container-tools-1.4.0 nvidia-
container-runtime-hook-1.4.0 docker-runtime-nvidia-1

echo "exclude=*nvidia* *cuda*" >> /etc/yum.conf
nvidia-smi
```

Buat AMI yang dioptimalkan dengan GPU yang kompatibel dengan P2 Anda sendiri

Anda dapat membuat AMI yang dioptimalkan untuk GPU Amazon ECS kustom Anda sendiri yang kompatibel dengan instans P2, lalu meluncurkan instans P2 menggunakan AMI.

1. Jalankan perintah berikut untuk mengkloning file. `amazon-ecs-ami repo`

   ```
   git clone https://github.com/aws/amazon-ecs-ami
   ```

2. Tetapkan agen Amazon ECS yang diperlukan dan sumber versi AMI Amazon Linux di `release.auto.pkrvars.hcl` atau`overrides.auto.pkrvars.hcl`.
3. Jalankan perintah berikut untuk membangun AMI EC2 yang kompatibel dengan P2 pribadi.

   Ganti region dengan Region dengan instance Region.

   ```
   REGION=region make al2keplergpu
   ```

4. Gunakan AMI dengan data pengguna instans berikut untuk menyambung ke klaster Amazon ECS.

   Ganti nama cluster dengan nama cluster Anda.

   ```
   #!/bin/bash
   echo "ECS_CLUSTER=cluster-name" >> /etc/ecs/ecs.config
   ```

Menggunakan transcoding video di Amazon ECS

Untuk menggunakan beban kerja transcoding video di Amazon ECS, daftarkan instans Amazon EC2 VT1. Setelah mendaftarkan instans ini, Anda dapat menjalankan beban kerja transcoding video langsung dan pra-render sebagai tugas di Amazon ECS. Instans Amazon EC2 VT1 menggunakan kartu transcoding media Xilinx U30 untuk mempercepat beban kerja transcoding video langsung dan pra-render.

> **Note**
>
> Untuk petunjuk tentang cara menjalankan beban kerja transcoding video dalam wadah selain Amazon ECS, lihat dokumentasi Xilinx.

Pertimbangan

Sebelum Anda mulai menerapkan VT1 di Amazon ECS, pertimbangkan hal berikut:

- Cluster Anda dapat berisi campuran instance VT1 dan non-VT1.
- Anda memerlukan aplikasi Linux yang menggunakan kartu transcoding media Xilinx U30 dengan codec AVC (H.264) dan HEVC (H.265) yang dipercepat.

> **Important**
>
> Aplikasi yang menggunakan codec lain mungkin tidak meningkatkan kinerja pada instance VT1.

- Hanya satu tugas transcoding yang dapat dijalankan pada kartu U30. Setiap kartu memiliki dua perangkat yang terkait dengannya. Anda dapat menjalankan tugas transcoding sebanyak mungkin karena ada kartu untuk setiap instance VT1 Anda.
- Saat membuat layanan atau menjalankan tugas mandiri, Anda dapat menggunakan atribut tipe instance saat mengonfigurasi batasan penempatan tugas. Ini memastikan bahwa tugas diluncurkan pada instance container yang Anda tentukan. Melakukannya membantu memastikan bahwa Anda menggunakan sumber daya Anda secara efektif dan bahwa tugas Anda untuk beban kerja transcoding video ada di instans VT1 Anda. Untuk informasi selengkapnya, lihat Bagaimana Amazon ECS menempatkan tugas pada instans kontainer.

  Dalam contoh berikut, tugas dijalankan pada `vt1.3xlarge` instance di `default` cluster Anda.

  ```
  aws ecs run-task \
       --cluster default \
       --task-definition vt1-3xlarge-xffmpeg-processor \
       --placement-constraints type=memberOf,expression="attribute:ecs.instance-type ==
   vt1.3xlarge"
  ```

- Anda mengonfigurasi kontainer untuk menggunakan kartu U30 spesifik yang tersedia pada instance wadah host. Anda dapat melakukan ini dengan menggunakan `linuxParameters` parameter dan menentukan detail perangkat. Untuk informasi selengkapnya, lihat Persyaratan ketentuan tugas.

Menggunakan AMI VT1

Anda memiliki dua opsi untuk menjalankan AMI di Amazon EC2 untuk instans penampung Amazon ECS. Opsi pertama adalah menggunakan AMI resmi Xilinx di. AWS Marketplace Opsi kedua adalah membangun AMI Anda sendiri dari repositori sampel.

- Xilinx menawarkan AMI di. AWS Marketplace
- Amazon ECS menyediakan contoh repositori yang dapat Anda gunakan untuk membangun AMI untuk beban kerja transcoding video. AMI ini hadir dengan driver Xilinx U30. Anda dapat

menemukan repositori yang berisi skrip Packer di. GitHub Untuk informasi selengkapnya tentang Packer, lihat dokumentasi Packer.

Persyaratan ketentuan tugas

Untuk menjalankan kontainer transcoding video di Amazon ECS, definisi tugas Anda harus berisi aplikasi transcoding video yang menggunakan codec H.264/AVC dan H.265/HEVC yang dipercepat. Anda dapat membuat gambar kontainer dengan mengikuti langkah-langkah di GitHubXilinx.

Definisi tugas harus spesifik untuk jenis instance. Jenis instans adalah 3xlarge, 6xlarge, dan 24xlarge. Anda harus mengonfigurasi kontainer untuk menggunakan perangkat Xilinx U30 tertentu yang tersedia pada instance wadah host. Anda dapat melakukannya dengan menggunakan `linuxParameters` parameter. Tabel berikut merinci kartu dan perangkat SoCs yang spesifik untuk setiap jenis instance.

Tipe Instans	vCPU	RAM (GiB)	Kartu akselerator U30	Perangkat SoC XCU30 yang dapat dialamatkan	Jalur Perangkat
vt1.3xlarge	12	24	1	2	`/dev/dri/renderD128 ,/dev/dri/renderD129`
vt1.6xlarge	24	48	2	4	`/dev/dri/renderD128 ,/dev/dri/renderD129 ,/dev/dri/renderD130 ,/dev/dri/`

Tipe Instans	vCPU	RAM (GiB)	Kartu akselerator U30	Perangkat SoC XCU30 yang dapat dialamatkan	Jalur Perangkat
					`renderD131`

Tipe Instans	vCPU	RAM (GiB)	Kartu akselerator U30	Perangkat SoC XCU30 yang dapat dialamatkan	Jalur Perangkat
vt1.24xlarge	96	182	8	16	`/dev/dri/renderD128 ,/dev/dri/renderD129 ,/dev/dri/renderD130 ,/dev/dri/renderD131 ,/dev/dri/renderD132 ,/dev/dri/renderD133 ,/dev/dri/renderD134 ,/dev/dri/renderD135 ,/dev/dri/renderD136 ,/dev/dri/renderD137 ,/dev/dri/`

Tipe Instans	vCPU	RAM (GiB)	Kartu akselerator U30	Perangkat SoC XCU30 yang dapat dialamatkan	Jalur Perangkat
					`renderD138 ,/dev/dri/renderD139 ,/dev/dri/renderD140 ,/dev/dri/renderD141 ,/dev/dri/renderD142 ,/dev/dri/renderD143`

> **Important**
>
> Jika definisi tugas mencantumkan perangkat yang tidak dimiliki instans EC2, tugas gagal dijalankan. Ketika tugas gagal, pesan kesalahan berikut muncul `distoppedReason:CannotStartContainerError: Error response from daemon: error gathering device information while adding custom device "/dev/dri/renderD130": no such file or directory.`

Dalam contoh berikut, sintaks yang digunakan untuk definisi tugas dari wadah Linux di Amazon EC2 disediakan. Definisi tugas ini adalah untuk gambar kontainer yang dibangun mengikuti prosedur yang disediakan dalam dokumentasi Xilinx. Jika Anda menggunakan contoh ini, ganti `image` dengan gambar Anda sendiri, dan salin file video Anda ke instance di `/home/ec2-user` direktori.

vt1.3xlarge

1. Buat file teks yang diberi nama `vt1-3xlarge-ffmpeg-linux.json` dengan konten berikut.

```
{
    "family": "vt1-3xlarge-xffmpeg-processor",
    "requiresCompatibilities": ["EC2"],
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
        {
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == vt1.3xlarge"
        }
    ],
    "containerDefinitions": [
        {
            "entryPoint": [
                "/bin/bash",
                "-c"
            ],
            "command": ["/video/ecs_ffmpeg_wrapper.sh"],
            "linuxParameters": {
                "devices": [
                    {
                        "containerPath": "/dev/dri/renderD128",
                        "hostPath": "/dev/dri/renderD128",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD129",
                        "hostPath": "/dev/dri/renderD129",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    }
```

```
                ]
            },
            "mountPoints": [
                {
                    "containerPath": "/video",
                    "sourceVolume": "video_file"
                }
            ],
            "cpu": 0,
            "memory": 12000,
            "image": "0123456789012.dkr.ecr.us-west-2.amazonaws.com/aws/xilinx-
xffmpeg",
            "essential": true,
            "name": "xilinix-xffmpeg"
        }
    ],
    "volumes": [
        {
            "name": "video_file",
            "host": {"sourcePath": "/home/ec2-user"}
        }
    ]
}
```

2. Mendaftarkan ketentuan tugas.

```
aws ecs register-task-definition --family vt1-3xlarge-xffmpeg-processor --cli-
input-json file://vt1-3xlarge-xffmpeg-linux.json --region us-east-1
```

vt1.6xlarge

1. Buat file teks yang diberi nama `vt1-6xlarge-ffmpeg-linux.json` dengan konten berikut.

```
{
    "family": "vt1-6xlarge-xffmpeg-processor",
    "requiresCompatibilities": ["EC2"],
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
```

```
        {
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == vt1.6xlarge"
        }
    ],
    "containerDefinitions": [
        {
            "entryPoint": [
                "/bin/bash",
                "-c"
            ],
            "command": ["/video/ecs_ffmpeg_wrapper.sh"],
            "linuxParameters": {
                "devices": [
                    {
                        "containerPath": "/dev/dri/renderD128",
                        "hostPath": "/dev/dri/renderD128",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD129",
                        "hostPath": "/dev/dri/renderD129",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD130",
                        "hostPath": "/dev/dri/renderD130",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD131",
                        "hostPath": "/dev/dri/renderD131",
                        "permissions": [
                            "read",
                            "write"
```

```
                        ]
                    }
                ]
            },
            "mountPoints": [
                {
                    "containerPath": "/video",
                    "sourceVolume": "video_file"
                }
            ],
            "cpu": 0,
            "memory": 12000,
            "image": "0123456789012.dkr.ecr.us-west-2.amazonaws.com/aws/xilinx-
xffmpeg",
            "essential": true,
            "name": "xilinix-xffmpeg"
        }
    ],
    "volumes": [
        {
            "name": "video_file",
            "host": {"sourcePath": "/home/ec2-user"}
        }
    ]
}
```

2. Mendaftarkan ketentuan tugas.

```
aws ecs register-task-definition --family vt1-6xlarge-xffmpeg-processor --cli-
input-json file://vt1-6xlarge-xffmpeg-linux.json --region us-east-1
```

vt1.24xlarge

1. Buat file teks yang diberi nama `vt1-24xlarge-ffmpeg-linux.json` dengan konten berikut.

```
{
    "family": "vt1-24xlarge-xffmpeg-processor",
    "requiresCompatibilities": ["EC2"],
    "placementConstraints": [
        {
            "type": "memberOf",
```

```
            "expression": "attribute:ecs.os-type == linux"
        },
        {
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == vt1.24xlarge"
        }
    ],
    "containerDefinitions": [
        {
            "entryPoint": [
                "/bin/bash",
                "-c"
            ],
            "command": ["/video/ecs_ffmpeg_wrapper.sh"],
            "linuxParameters": {
                "devices": [
                    {
                        "containerPath": "/dev/dri/renderD128",
                        "hostPath": "/dev/dri/renderD128",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD129",
                        "hostPath": "/dev/dri/renderD129",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD130",
                        "hostPath": "/dev/dri/renderD130",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD131",
                        "hostPath": "/dev/dri/renderD131",
                        "permissions": [
```

```
                        "read",
                        "write"
                    ]
                },
                {
                    "containerPath": "/dev/dri/renderD132",
                    "hostPath": "/dev/dri/renderD132",
                    "permissions": [
                        "read",
                        "write"
                    ]
                },
                {
                    "containerPath": "/dev/dri/renderD133",
                    "hostPath": "/dev/dri/renderD133",
                    "permissions": [
                        "read",
                        "write"
                    ]
                },
                {
                    "containerPath": "/dev/dri/renderD134",
                    "hostPath": "/dev/dri/renderD134",
                    "permissions": [
                        "read",
                        "write"
                    ]
                },
                {
                    "containerPath": "/dev/dri/renderD135",
                    "hostPath": "/dev/dri/renderD135",
                    "permissions": [
                        "read",
                        "write"
                    ]
                },
                {
                    "containerPath": "/dev/dri/renderD136",
                    "hostPath": "/dev/dri/renderD136",
                    "permissions": [
                        "read",
                        "write"
                    ]
                },
```

```
                    {
                        "containerPath": "/dev/dri/renderD137",
                        "hostPath": "/dev/dri/renderD137",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD138",
                        "hostPath": "/dev/dri/renderD138",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD139",
                        "hostPath": "/dev/dri/renderD139",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD140",
                        "hostPath": "/dev/dri/renderD140",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD141",
                        "hostPath": "/dev/dri/renderD141",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD142",
                        "hostPath": "/dev/dri/renderD142",
                        "permissions": [
```

```
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD143",
                        "hostPath": "/dev/dri/renderD143",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    }
                ]
            },
            "mountPoints": [
                {
                    "containerPath": "/video",
                    "sourceVolume": "video_file"
                }
            ],
            "cpu": 0,
            "memory": 12000,
            "image": "0123456789012.dkr.ecr.us-west-2.amazonaws.com/aws/xilinx-
xffmpeg",
            "essential": true,
            "name": "xilinix-xffmpeg"
        }
    ],
    "volumes": [
        {
            "name": "video_file",
            "host": {"sourcePath": "/home/ec2-user"}
        }
    ]
}
```

2. Mendaftarkan ketentuan tugas.

```
aws ecs register-task-definition --family vt1-24xlarge-xffmpeg-processor --cli-
input-json file://vt1-24xlarge-xffmpeg-linux.json --region us-east-1
```

Menggunakan AWS Neuron di Amazon Linux 2 di Amazon ECS

Anda dapat mendaftarkan instans Amazon EC2 Trn1, Amazon EC2 Inf1, dan Amazon EC2 Inf2 ke cluster Anda untuk beban kerja pembelajaran mesin.

Instans Amazon EC2 Trn1 didukung oleh chip Trainium.AWS Instans ini memberikan pelatihan berkinerja tinggi dan biaya rendah untuk pembelajaran mesin di cloud. Anda dapat melatih model inferensi pembelajaran mesin menggunakan kerangka pembelajaran mesin dengan AWS Neuron pada instance Trn1. Kemudian, Anda dapat menjalankan model pada instance Inf1, atau instance Inf2 untuk menggunakan akselerasi chip AWS Inferentia.

Instans Amazon EC2 Inf1 dan instans Inf2 didukung oleh chip AWS Inferentia. Instans ini memberikan kinerja tinggi dan inferensi biaya terendah di cloud.

Model pembelajaran mesin digunakan ke wadah menggunakan AWS Neuron, yang merupakan Kit Pengembang Perangkat Lunak (SDK) khusus. SDK terdiri dari compiler, runtime, dan alat profiling yang mengoptimalkan kinerja machine learning chip machine learning. AWS AWS Neuron mendukung kerangka kerja pembelajaran mesin populer seperti TensorFlow, PyTorch, dan Apache MXNet.

Pertimbangan

Sebelum Anda mulai menerapkan Neuron di Amazon ECS, pertimbangkan hal berikut:

- Cluster Anda dapat berisi campuran Trn1, Inf1, Inf2, dan instance lainnya.
- Anda memerlukan aplikasi Linux dalam wadah yang menggunakan kerangka pembelajaran mesin yang mendukung AWS Neuron.

  > **Important**
  >
  > Aplikasi yang menggunakan kerangka kerja lain mungkin tidak meningkatkan kinerja pada instance Trn1, Inf1, dan Inf2.

- Hanya satu tugas inferensi atau pelatihan inferensi yang dapat dijalankan pada setiap chip AWS Trainium atau Inferentia.AWS Untuk Inf1, setiap chip memiliki 4 NeuronCores. Untuk Trn1 dan Inf2 setiap chip memiliki 2. NeuronCores Anda dapat menjalankan tugas sebanyak mungkin karena ada chip untuk setiap instans Trn1, Inf1, dan Inf2 Anda.
- Saat membuat layanan atau menjalankan tugas mandiri, Anda dapat menggunakan atribut tipe instance saat mengonfigurasi batasan penempatan tugas. Ini memastikan bahwa tugas

diluncurkan pada instance container yang Anda tentukan. Melakukannya dapat membantu Anda mengoptimalkan pemanfaatan sumber daya secara keseluruhan dan memastikan bahwa tugas untuk beban kerja inferensi ada di instans Trn1, Inf1, dan Inf2 Anda. Untuk informasi selengkapnya, lihat Bagaimana Amazon ECS menempatkan tugas pada instans kontainer.

Dalam contoh berikut, tugas dijalankan pada `Inf1.xlarge` instance di `default` cluster Anda.

```
aws ecs run-task \
     --cluster default \
     --task-definition ecs-inference-task-def \
     --placement-constraints type=memberOf,expression="attribute:ecs.instance-type ==
 Inf1.xlarge"
```

- Persyaratan sumber daya neuron tidak dapat didefinisikan dalam definisi tugas. Sebagai gantinya, Anda mengonfigurasi wadah untuk menggunakan chip AWS Trainium atau AWS Inferentia tertentu yang tersedia di instance wadah host. Lakukan ini dengan menggunakan `linuxParameters` parameter dan menentukan detail perangkat. Untuk informasi selengkapnya, lihat Persyaratan ketentuan tugas.

Menggunakan Amazon ECS yang dioptimalkan Amazon Linux 2 (Neuron) AMI

Amazon ECS menyediakan AMI Amazon ECS yang dioptimalkan yang didasarkan pada Amazon Linux 2 untuk beban kerja AWS Trainium dan AWS Inferentia. Muncul dengan driver AWS Neuron dan runtime untuk Docker. AMI ini membuat menjalankan beban kerja inferensi pembelajaran mesin lebih mudah di Amazon ECS.

Sebaiknya gunakan AMI Amazon Linux 2 (Neuron) Amazon ECS yang dioptimalkan oleh Amazon ECS saat meluncurkan instans Amazon EC2 Trn1, Inf1, dan Inf2 Anda.

Anda dapat mengambil Amazon ECS saat ini yang dioptimalkan Amazon Linux 2 (Neuron) AMI menggunakan AWS CLI perintah berikut.

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/inf/
recommended
```

Amazon ECS yang dioptimalkan Amazon Linux 2 (Neuron) AMI didukung di Wilayah berikut:

- AS Timur (N. Virginia)
- AS Timur (Ohio)

- AS Barat (California Utara)
- AS Barat (Oregon)
- Asia Pasifik (Mumbai)
- Asia Pasifik (Osaka)
- Asia Pasifik (Seoul)
- Asia Pasifik (Tokyo)
- Asia Pasifik (Singapura)
- Asia Pacific (Sydney)
- Kanada (Pusat)
- Eropa (Frankfurt)
- Eropa (Irlandia)
- Eropa (London)
- Eropa (Paris)
- Eropa (Stockholm)
- Amerika Selatan (Sao Paulo)

Persyaratan ketentuan tugas

Untuk menerapkan Neuron di Amazon ECS, definisi tugas Anda harus berisi definisi kontainer untuk wadah bawaan yang menyajikan model inferensi. TensorFlow Ini disediakan oleh AWS Deep Learning Containers. Wadah ini berisi runtime AWS Neuron dan aplikasi TensorFlow Serving. Saat startup, penampung ini mengambil model Anda dari Amazon S3, meluncurkan Penyajian TensorFlow Neuron dengan model yang disimpan, dan menunggu permintaan prediksi. Dalam contoh berikut, gambar kontainer memiliki TensorFlow 1.15 dan Ubuntu 18.04. Daftar lengkap Deep Learning Containers pra-bangun yang dioptimalkan untuk Neuron GitHub dipertahankan. Untuk informasi lebih lanjut, lihat Menggunakan TensorFlow Penyajian AWS Neuron.

```
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-neuron:1.15.4-neuron-
py37-ubuntu18.04
```

Atau, Anda dapat membangun sendiri citra kontainer Neuron sespan. Untuk informasi lebih lanjut, lihat Tutorial: TensorFlow Penyajian Neuron di Panduan AWS Deep Learning AMI Pengembang.

Definisi tugas harus spesifik untuk satu jenis instance. Anda harus mengonfigurasi wadah untuk menggunakan perangkat AWS Trainium atau AWS Inferentia tertentu yang tersedia di instance

wadah host. Anda dapat melakukannya dengan menggunakan `linuxParameters` parameter. Tabel berikut merinci chip yang spesifik untuk setiap jenis instance.

Tipe Instans	vCPU	RAM (GiB)	AWS Chip akselerator mL	Jalur Perangkat
trn1.2xlarge	8	32	1	`/dev/neuron0`
trn1.32xlarge	128	512	16	`/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3 ,/dev/neuron4 ,/dev/neuron5 ,/dev/neuron6 ,/dev/neuron7 ,/dev/neuron8 ,/dev/neuron9 ,/dev/neuron10 ,/dev/neuron11 ,/dev/neuron12 ,/dev/neuron13 ,/dev/neuron14 ,/dev/neuron15`
inf1.xlarge	4	8	1	`/dev/neuron0`

Tipe Instans	vCPU	RAM (GiB)	AWS Chip akselerator mL	Jalur Perangkat
inf1.2xlarge	8	16	1	`/dev/neuron0`
inf1.6xlarge	24	48	4	`/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3`
inf1.24xlarge	96	192	16	`/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3 ,/dev/neuron4 ,/dev/neuron5 ,/dev/neuron6 ,/dev/neuron7 ,/dev/neuron8 ,/dev/neuron9 ,/dev/neuron10 ,/dev/neuron11 ,/dev/neuron12 ,/dev/neuron13 ,/dev/neuron14 ,/dev/neuron15`

Tipe Instans	vCPU	RAM (GiB)	AWS Chip akselerator mL	Jalur Perangkat
inf2.xlarge	8	16	1	`/dev/neuron0`
inf2.8xlarge	32	64	1	`/dev/neuron0`
inf2.24xlarge	96	384	6	`/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3 ,/dev/neuron4 ,/dev/neuron5 ,`
inf2.48xlarge	192	768	12	`/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3 ,/dev/neuron4 ,/dev/neuron5 ,/dev/neuron6 ,/dev/neuron7 ,/dev/neuron8 ,/dev/neuron9 ,/dev/neuron10 ,/dev/neuron11`

Berikut ini adalah contoh definisi tugas Linux untuk`inf1.xlarge`, menampilkan sintaks yang akan digunakan.

```
{
    "family": "ecs-neuron",
    "requiresCompatibilities": ["EC2"],
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
        {
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == inf1.xlarge"
        }
    ],
    "executionRoleArn": "${YOUR_EXECUTION_ROLE}",
    "containerDefinitions": [
        {
            "entryPoint": [
                "/usr/local/bin/entrypoint.sh",
                "--port=8500",
                "--rest_api_port=9000",
                "--model_name=resnet50_neuron",
                "--model_base_path=s3://your-bucket-of-models/resnet50_neuron/"
            ],
            "portMappings": [
                {
                    "hostPort": 8500,
                    "protocol": "tcp",
                    "containerPort": 8500
                },
                {
                    "hostPort": 8501,
                    "protocol": "tcp",
                    "containerPort": 8501
                },
                {
                    "hostPort": 0,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ],
```

```
            "linuxParameters": {
                "devices": [
                    {
                        "containerPath": "/dev/neuron0",
                        "hostPath": "/dev/neuron0",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    }
                ],
                "capabilities": {
                    "add": [
                        "IPC_LOCK"
                    ]
                }
            },
            "cpu": 0,
            "memoryReservation": 1000,
            "image": "763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-
inference-neuron:1.15.4-neuron-py37-ubuntu18.04",
            "essential": true,
            "name": "resnet50"
        }
    ]
}
```

Menggunakan instans DL1 pembelajaran mendalam di Amazon ECS

Untuk menggunakan beban kerja pembelajaran mendalam di Amazon ECS, daftarkan instans Amazon EC2 DL1 ke cluster Anda. Instans Amazon EC2 DL1 didukung oleh akselerator Gaudi dari Habana Labs (perusahaan Intel). Gunakan Habana SynapSeai SDK untuk terhubung ke akselerator Habana Gaudi. SDK mendukung kerangka kerja pembelajaran mesin yang populer, TensorFlow dan. PyTorch

Pertimbangan

Sebelum Anda mulai menerapkan DL1 di Amazon ECS, pertimbangkan hal berikut:

- Cluster Anda dapat berisi campuran instance DL1 dan non-DL1.
- Saat membuat layanan atau menjalankan tugas mandiri, Anda dapat menggunakan atribut tipe instance secara khusus saat mengonfigurasi batasan penempatan tugas untuk memastikan bahwa

tugas Anda diluncurkan pada instance container yang Anda tentukan. Melakukannya memastikan bahwa sumber daya Anda digunakan secara efektif dan bahwa tugas Anda untuk beban kerja pembelajaran mendalam ada di instans DL1 Anda. Untuk informasi selengkapnya, lihat Bagaimana Amazon ECS menempatkan tugas pada instans kontainer.

Contoh berikut menjalankan tugas pada instans `dl1.24xlarge` di klaster `default` Anda.

```
aws ecs run-task \
     --cluster default \
     --task-definition ecs-dl1-task-def \
     --placement-constraints type=memberOf,expression="attribute:ecs.instance-type ==
 dl1.24xlarge"
```

Menggunakan AMI DL1

Anda memiliki tiga opsi untuk menjalankan AMI di instans Amazon EC2 DL1 untuk Amazon ECS:

- AWS Marketplace AMI yang disediakan oleh Habana di sini.
- Habana Deep Learning AMI yang disediakan oleh Amazon Web Services. Karena tidak termasuk, Anda perlu menginstal agen kontainer Amazon ECS secara terpisah.
- Gunakan Packer untuk membuat AMI khusus yang disediakan oleh GitHubrepo. Untuk informasi selengkapnya, lihat dokumentasi Packer.

Persyaratan ketentuan tugas

Untuk menjalankan wadah pembelajaran mendalam yang dipercepat Habana Gaudi di Amazon ECS, definisi tugas Anda harus berisi definisi wadah untuk wadah pra-bangun yang menyajikan model pembelajaran mendalam untuk TensorFlow atau PyTorch menggunakan Habana SynapSeai yang disediakan oleh Deep Learning Containers. AWS

Gambar kontainer berikut memiliki TensorFlow 2.7.0 dan Ubuntu 20.04. Daftar lengkap Deep Learning Containers pra-bangun yang dioptimalkan untuk akselerator Habana Gaudi dipertahankan. GitHub Untuk informasi lebih lanjut, lihat Wadah Pelatihan Habana.

```
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training-habana:2.7.0-hpu-py38-
synapseai1.2.0-ubuntu20.04
```

Berikut ini adalah contoh definisi tugas untuk wadah Linux di Amazon EC2, menampilkan sintaks yang akan digunakan. Contoh ini menggunakan gambar yang berisi Habana Labs System Management Interface Tool (HL-SMI) yang ditemukan di sini: `vault.habana.ai/gaudi-docker/1.1.0/ubuntu20.04/habanalabs/tensorflow-installer-tf-cpu-2.6.0:1.1.0-614`

```
{
    "family": "dl-test",
    "requiresCompatibilities": ["EC2"],
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
        {
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == dl1.24xlarge"
        }
    ],
    "networkMode": "host",
    "cpu": "10240",
    "memory": "1024",
    "containerDefinitions": [
        {
            "entryPoint": [
                "sh",
                "-c"
            ],
            "command": ["hl-smi"],
            "cpu": 8192,
            "environment": [
                {
                    "name": "HABANA_VISIBLE_DEVICES",
                    "value": "all"
                }
            ],
            "image": "vault.habana.ai/gaudi-docker/1.1.0/ubuntu20.04/habanalabs/
tensorflow-installer-tf-cpu-2.6.0:1.1.0-614",
            "essential": true,
            "name": "tensorflow-installer-tf-hpu"
        }
    ]
```

```
}
```

Bekerja dengan beban kerja ARM 64-bit di Amazon ECS

Amazon ECS mendukung penggunaan aplikasi ARM 64-bit. Anda dapat menjalankan aplikasi Anda di platform yang didukung oleh prosesor AWS Graviton2. Ini cocok untuk berbagai macam beban kerja. Ini termasuk beban kerja seperti server aplikasi, layanan mikro, komputasi kinerja tinggi, inferensi pembelajaran mesin berbasis CPU, pengkodean video, otomatisasi desain elektronik, game, database sumber terbuka, dan cache dalam memori.

Pertimbangan

Sebelum Anda mulai menerapkan definisi tugas yang menggunakan arsitektur ARM 64-bit, pertimbangkan hal berikut:

- Aplikasi dapat menggunakan jenis peluncuran Fargate atau EC2.
- Tugas Linux dengan arsitektur ARM64 tidak mendukung penyedia kapasitas Fargate Spot.
- Aplikasi hanya dapat menggunakan sistem operasi Linux.
- Untuk tipe Fargate, aplikasi harus menggunakan versi platform Fargate atau yang lebih baru. `1.4.0`
- Aplikasi dapat digunakan Fluent Bit atau CloudWatch untuk pemantauan.
- Untuk jenis peluncuran Fargate, berikut ini Wilayah AWS tidak mendukung beban kerja ARM 64-bit:
  - AS Timur (Virginia N.), Zona `use1-az3` Ketersediaan
- Untuk jenis peluncuran Amazon EC2, lihat berikut ini untuk memverifikasi bahwa Wilayah tempat Anda berada mendukung jenis instans yang ingin Anda gunakan:
  - Instans Amazon EC2 M6g
  - Instans Amazon EC2 T4G
  - Instans Amazon EC2 C6g
  - Instans Amazon EC2 R6gd
  - Instans Amazon EC2 x2GD

  Anda juga dapat menggunakan `describe-instance-type-offerings` perintah Amazon EC2 dengan filter untuk melihat penawaran instans untuk Wilayah Anda.

```
aws ec2 describe-instance-type-offerings --filters Name=instance-
type,Values=instance-type --region region
```

Contoh berikut memeriksa ketersediaan tipe instans M6 di Wilayah AS Timur (Virginia N.) (us-timur-1).

```
aws ec2 describe-instance-type-offerings --filters "Name=instance-type,Values=m6*" --
region us-east-1
```

Untuk informasi selengkapnya, lihat describe-instance-type-offerings di Referensi Baris Perintah Amazon EC2.

Menentukan arsitektur ARM dalam definisi tugas Anda

Untuk menggunakan arsitektur ARM, `ARM64` tentukan parameter definisi `cpuArchitecture` tugas.

Dalam contoh berikut, arsitektur ARM ditentukan dalam definisi tugas. Ini dalam format JSON.

```
{
    "runtimePlatform": {
        "operatingSystemFamily": "LINUX",
        "cpuArchitecture": "ARM64"
    },
...
}
```

Dalam contoh berikut, definisi tugas untuk arsitektur ARM menampilkan “hello world.”

```
{
 "family": "arm64-testapp",
 "networkMode": "awsvpc",
 "containerDefinitions": [
    {
        "name": "arm-container",
        "image": "arm64v8/busybox",
        "cpu": 100,
        "memory": 100,
        "essential": true,
        "command": [ "echo hello world" ],
```

```
        "entryPoint": [ "sh", "-c" ]
    }
 ],
 "requiresCompatibilities": [ "FARGATE" ],
 "cpu": "256",
 "memory": "512",
 "runtimePlatform": {
        "operatingSystemFamily": "LINUX",
        "cpuArchitecture": "ARM64"
  },
 "executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole"
}
```

Antarmuka untuk mengkonfigurasi ARM

Anda dapat mengonfigurasi arsitektur CPU ARM untuk definisi tugas Amazon ECS menggunakan salah satu antarmuka berikut:

- Konsol Amazon ECS
- AWS Command Line Interface (AWS CLI)
- AWS SDK
- AWS Kopilot

Menggunakan driver log awslogs

Anda dapat mengonfigurasi kontainer dalam tugas Anda untuk mengirim informasi CloudWatch log ke Log. Jika Anda menggunakan tipe peluncuran Fargate untuk tugas Anda, Anda dapat melihat log dari kontainer Anda. Jika Anda menggunakan tipe peluncuran EC2, Anda dapat melihat log yang berbeda dari kontainer Anda di satu lokasi yang nyaman, dan ini mencegah log kontainer Anda mengambil ruang disk pada instance kontainer Anda.Topik ini membahas bagaimana Anda dapat mulai menggunakan driver `awslogs` log dalam definisi tugas Anda.

> **Note**
>
> Jenis informasi yang dicatat oleh kontainer dalam tugas Anda sebagian besar bergantung pada perintah `ENTRYPOINT` mereka. Secara default, log yang diambil menunjukkan output perintah yang biasanya Anda lihat di terminal interaktif jika Anda menjalankan kontainer secara lokal, yang merupakan aliran `STDOUT` dan `STDERR` I/O. Driver `awslogs` log hanya meneruskan log ini dari Docker ke CloudWatch Logs. Untuk informasi selengkapnya tentang

cara log Docker diproses, termasuk cara alternatif untuk menangkap berbagai pengaliran atau data file, lihat Melihat log untuk kontainer atau layanan dalam dokumentasi Docker.

Untuk mengirim log sistem dari instans penampung Amazon ECS Anda ke CloudWatch Log, lihat Memantau Kuota File Log dan CloudWatch Log di Panduan Pengguna Amazon CloudWatch Logs.

Mengaktifkan driver log awslogs untuk kontainer Anda

Jika Anda menggunakan tipe peluncuran Fargate untuk tugas Anda, Anda perlu menambahkan `logConfiguration` parameter yang diperlukan ke definisi tugas Anda untuk mengaktifkan driver `awslogs` log. Untuk informasi selengkapnya, lihat Menentukan konfigurasi log dalam ketentuan tugas Anda.

Jika Anda menggunakan tipe peluncuran EC2 untuk tugas Anda dan ingin mengaktifkan driver `awslogs` log, instans penampung Amazon ECS Anda memerlukan setidaknya versi 1.9.0 dari agen penampung. Untuk informasi tentang cara memeriksa versi agen Anda dan memperbarui ke versi terbaru, lihatMemperbarui agen kontainer Amazon ECS.

Note

Jika Anda tidak menggunakan AMI yang dioptimalkan Amazon ECS (dengan setidaknya versi 1.9.0-1 `ecs-init` paket) untuk instance container, Anda juga perlu menentukan bahwa driver `awslogs` logging tersedia di instance container saat memulai agen dengan menggunakan variabel lingkungan berikut dalam file docker run pernyataan atau variabel lingkungan Anda. Untuk informasi selengkapnya, lihat Memasang agen kontainer Amazon ECS.

```
ECS_AVAILABLE_LOGGING_DRIVERS=["json-file","awslogs"]
```

Instans penampung Amazon ECS Anda juga memerlukan `logs:CreateLogStream` dan `logs:PutLogEvents` izin pada peran IAM yang dapat digunakan untuk meluncurkan instans penampung. Jika Anda membuat peran instans penampung Amazon ECS sebelum dukungan driver `awslogs` log diaktifkan di Amazon ECS, Anda mungkin perlu menambahkan izin ini. `ecsTaskExecutionRole`Ini digunakan saat ditugaskan ke tugas dan kemungkinan berisi izin yang benar. Untuk informasi tentang peran eksekusi tugas, lihatPeran IAM eksekusi tugas Amazon ECS. Jika instance container Anda menggunakan kebijakan IAM terkelola untuk instance container,

instance container Anda kemungkinan memiliki izin yang benar. Untuk informasi tentang kebijakan IAM terkelola untuk instance container, lihat. Peran IAM instans wadah Amazon ECS

Membuat grup log

Driver `awslogs` log dapat mengirim aliran log ke grup log yang ada di CloudWatch Log atau membuat grup log baru atas nama Anda. AWS Management Console Ini menyediakan opsi konfigurasi otomatis, yang membuat grup log atas nama Anda menggunakan nama keluarga definisi tugas dengan `ecs` sebagai awalan. Atau, Anda dapat secara manual menentukan opsi konfigurasi log Anda dan menentukan `awslogs-create-group` opsi dengan nilai`true`, yang membuat grup log atas nama Anda.

> **Note**
>
> Untuk menggunakan `awslogs-create-group` opsi untuk membuat grup log Anda, kebijakan peran IAM eksekusi tugas atau kebijakan peran instans EC2 harus menyertakan izin. `logs:CreateLogGroup`

Kode berikut menunjukkan cara mengatur `awslogs-create-group` opsi.

```
{
    "containerDefinitions": [
        {
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "firelens-container",
                    "awslogs-region": "us-west-2",
                    "awslogs-create-group": "true",
                    "awslogs-stream-prefix": "firelens"
                }
            }
    ]
}
```

Menggunakan fitur autokonfigurasi untuk membuat grup log

Saat Anda mendaftarkan definisi tugas, di konsol Amazon ECS, Anda dapat mengizinkan Amazon ECS untuk mengonfigurasi log Anda secara otomatis. CloudWatch Melakukan hal ini menyebabkan

grup log dibuat atas nama Anda menggunakan nama keluarga definisi tugas dengan `ecs` awalan. Untuk informasi selengkapnya, lihat the section called “Membuat definisi tugas menggunakan konsol”.

Opsi driver log awslogs yang tersedia

Driver `awslogs` log mendukung opsi berikut dalam definisi tugas Amazon ECS. Untuk informasi selengkapnya, lihat Driver logging CloudWatch log.

`awslogs-create-group`

Diperlukan: Tidak

Tentukan apakah Anda ingin grup log dibuat secara otomatis. Jika opsi ini tidak ditentukan, default-nya adalah `false`.

> **Note**
>
> Kebijakan IAM Anda harus menyertakan `logs:CreateLogGroup` izin sebelum Anda mencoba menggunakannya`awslogs-create-group`.

`awslogs-region`

Diperlukan: Ya

Tentukan Wilayah AWS driver `awslogs` log untuk mengirim log Docker Anda. Anda dapat memilih untuk mengirim semua log Anda dari kluster di Wilayah yang berbeda ke satu wilayah di CloudWatch Log. Ini agar mereka semua terlihat di satu lokasi. Jika tidak, Anda dapat memisahkannya berdasarkan Wilayah untuk perincian lebih lanjut. Pastikan grup log yang ditentukan ada di Wilayah yang Anda tentukan dengan opsi ini.

`awslogs-group`

Diperlukan: Ya

Pastikan untuk menentukan grup log tempat driver `awslogs` log mengirimkan aliran lognya. Untuk informasi selengkapnya, lihat Membuat grup log.

`awslogs-stream-prefix`

Diperlukan: Ya, saat menggunakan tipe peluncuran Fargate. Opsional untuk tipe peluncuran EC2, diperlukan untuk jenis peluncuran Fargate.

Gunakan `awslogs-stream-prefix` opsi untuk mengaitkan aliran log dengan awalan yang ditentukan, nama penampung, dan ID tugas Amazon ECS tempat penampung tersebut berada. Jika Anda menentukan awalan dengan opsi ini, maka aliran log mengambil format berikut.

```
prefix-name/container-name/ecs-task-id
```

Jika Anda tidak menentukan awalan dengan opsi ini, maka aliran log dinamai sesuai dengan ID penampung yang ditetapkan oleh daemon Docker pada instance container. Karena sulit untuk melacak log kembali ke wadah yang mengirimnya hanya dengan ID wadah Docker (yang hanya tersedia pada instance penampung), kami sarankan Anda menentukan awalan dengan opsi ini.

Untuk layanan Amazon ECS, Anda dapat menggunakan nama layanan sebagai awalan. Dengan melakukannya, Anda dapat melacak aliran log ke layanan tempat penampung milik, nama penampung yang mengirimnya, dan ID tugas yang menjadi milik penampung tersebut.

Anda harus menentukan awalan aliran agar log Anda muncul di panel Log saat menggunakan konsol Amazon ECS.

`awslogs-datetime-format`

Diperlukan: Tidak

Opsi ini mendefinisikan pola mulai multiline di Python format `strftime`. Pesan log terdiri dari baris yang cocok dengan pola dan baris berikut yang tidak cocok dengan pola. Baris yang cocok adalah pembatas antara pesan log.

Salah satu contoh kasus penggunaan format ini adalah untuk penguraian output seperti buangan tumpukan, yang mungkin bisa tercatat dalam beberapa entri. Pola yang benar akan memungkinkannya ditangkap dalam satu entri.

Untuk informasi lebih lanjut, lihat awslogs-datetime-format.

Anda tidak dapat mengonfigurasi opsi `awslogs-datetime-format` dan `awslogs-multiline-pattern` opsi.

> **Note**
>
> Pencatatan multibaris melakukan penguraian dan pencocokan ekspresi reguler dari semua pesan log. Ini mungkin berdampak negatif pada performa pencatatan.

`awslogs-multiline-pattern`

Diperlukan: Tidak

Opsi ini mendefinisikan pola mulai multiline yang menggunakan ekspresi reguler. Pesan log terdiri dari baris yang cocok dengan pola dan baris berikut yang tidak cocok dengan pola. Baris yang cocok adalah pembatas antara pesan log.

Untuk informasi lebih lanjut, lihat awslogs-multiline-pattern.

Opsi ini diabaikan jika `awslogs-datetime-format` juga dikonfigurasi.

Anda tidak dapat mengonfigurasi opsi `awslogs-datetime-format` dan `awslogs-multiline-pattern` opsi.

> **Note**
>
> Pencatatan multibaris melakukan penguraian dan pencocokan ekspresi reguler dari semua pesan log. Ini mungkin berdampak negatif pada performa pencatatan.

`mode`

Diperlukan: Tidak

Nilai yang valid: `non-blocking` | `blocking`

Nilai default: `blocking`

Opsi ini mendefinisikan mode pengiriman pesan log dari wadah ke CloudWatch Log. Mode pengiriman yang Anda pilih memengaruhi ketersediaan aplikasi saat aliran log dari kontainer ke CloudWatch terputus.

Jika Anda menggunakan `blocking` mode default dan aliran log ke CloudWatch terganggu, panggilan dari kode kontainer untuk menulis ke `stdout` dan `stderr` aliran akan diblokir. Thread logging aplikasi akan diblokir sebagai hasilnya. Hal ini dapat menyebabkan aplikasi menjadi tidak responsif dan menyebabkan kegagalan pemeriksaan kesehatan kontainer.

Jika Anda menggunakan `non-blocking` mode, log kontainer malah disimpan dalam buffer perantara dalam memori yang dikonfigurasi dengan opsi. `max-buffer-size` Ini mencegah aplikasi menjadi tidak responsif ketika log tidak dapat dikirim ke CloudWatch. Kami

merekomendasikan menggunakan mode ini jika Anda ingin memastikan ketersediaan layanan dan baik-baik saja dengan beberapa kehilangan log.

`max-buffer-size`

Diperlukan: Tidak

Nilai default: `1m`

Saat `non-blocking` mode digunakan, opsi `max-buffer-size` log mengontrol ukuran buffer yang digunakan untuk penyimpanan pesan perantara. Pastikan untuk menentukan ukuran buffer yang memadai berdasarkan aplikasi Anda. Ketika buffer terisi, log lebih lanjut tidak dapat disimpan. Log yang tidak dapat disimpan hilang.

Menentukan konfigurasi log dalam ketentuan tugas Anda

Sebelum kontainer Anda dapat mengirim log ke CloudWatch, Anda harus menentukan driver `awslogs` log untuk kontainer dalam definisi tugas Anda. Bagian ini menjelaskan konfigurasi log untuk kontainer untuk menggunakan driver log `awslogs`. Untuk informasi selengkapnya, lihat Membuat definisi tugas menggunakan konsol.

Definisi tugas JSON yang berikut memiliki `logConfiguration` objek yang ditentukan untuk setiap kontainer. Salah satunya adalah untuk WordPress wadah yang mengirim log ke grup log yang dipanggil`awslogs-wordpress`. Yang lainnya adalah untuk wadah MySQL yang mengirimkan log ke grup log yang dipanggil. `awslogs-mysql` Kedua kontainer menggunakan prefiks pengaliran log `awslogs-example`.

```
{
    "containerDefinitions": [
        {
            "name": "wordpress",
            "links": [
                "mysql"
            ],
            "image": "wordpress",
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80
                }
```

```
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-create-group": "true",
                    "awslogs-group": "awslogs-wordpress",
                    "awslogs-region": "us-west-2",
                    "awslogs-stream-prefix": "awslogs-example"
                }
            },
            "memory": 500,
            "cpu": 10
        },
        {
            "environment": [
                {
                    "name": "MYSQL_ROOT_PASSWORD",
                    "value": "password"
                }
            ],
            "name": "mysql",
            "image": "mysql",
            "cpu": 10,
            "memory": 500,
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-create-group": "true",
                    "awslogs-group": "awslogs-mysql",
                    "awslogs-region": "us-west-2",
                    "awslogs-stream-prefix": "awslogs-example",
                    "mode": "non-blocking",
                    "max-buffer-size": "25m"
                }
            }
        }
    ],
    "family": "awslogs-example"
}
```

Setelah Anda mendaftarkan definisi tugas dengan driver `awslogs` log dalam konfigurasi log definisi kontainer, Anda dapat menjalankan tugas atau membuat layanan dengan definisi tugas tersebut

untuk mulai mengirim CloudWatch log ke Log. Lihat informasi yang lebih lengkap di Jalankan aplikasi sebagai tugas Amazon ECS dan Membuat layanan menggunakan konsol.

Menggunakan perutean log khusus

Anda dapat menggunakan Amazon ECS FireLens untuk menggunakan parameter definisi tugas untuk merutekan log ke AWS layanan atau tujuan AWS Partner Network (APN) untuk penyimpanan log dan analitik. AWS Partner Network Ini adalah komunitas mitra global yang memanfaatkan program, keahlian, dan sumber daya untuk membangun, memasarkan, dan menjual penawaran pelanggan. Untuk informasi lebih lanjut lihat AWS Partner. FireLensbekerja dengan Fluentd danFluent Bit. Kami menyediakan AWS untuk Fluent Bit gambar atau Anda dapat menggunakan gambar Anda sendiri Fluentd atau Fluent Bit gambar.

Pertimbangan

Pertimbangkan hal berikut saat menggunakan FireLens untuk Amazon ECS:

- Kami menyarankan Anda `my_service_` menambahkan nama wadah log sehingga Anda dapat dengan mudah membedakan nama kontainer di konsol.
- Amazon ECS menambahkan dependensi pesanan kontainer awal antara container aplikasi dan FireLens container secara default. Saat Anda menentukan urutan kontainer antara wadah aplikasi dan FireLens kontainer, maka urutan kontainer awal default diganti.
- FireLensuntuk Amazon ECS didukung untuk tugas-tugas yang di-host AWS Fargate di Linux dan Amazon EC2 di Linux. Kontainer Windows tidak mendukungFireLens.

  Untuk informasi tentang cara mengonfigurasi pencatatan terpusat untuk kontainer Windows, lihat Pencatatan terpusat untuk kontainer Windows di Amazon ECS menggunakan Bit Lancar.
- FireLensuntuk Amazon ECS didukung dalam AWS CloudFormation template. Untuk informasi selengkapnya, lihat AWS::ECS::TaskDefinition FirelensConfigurationdi Panduan AWS CloudFormation Pengguna
- FireLensmendengarkan di port`24224`, jadi untuk memastikan bahwa router FireLens log tidak dapat dijangkau di luar tugas, Anda tidak boleh mengizinkan lalu lintas masuk pada port `24224` dalam grup keamanan yang digunakan tugas Anda. Untuk tugas yang menggunakan mode `awsvpc` jaringan, ini adalah grup keamanan yang terkait dengan tugas tersebut. Untuk tugas yang menggunakan mode `host` jaringan, ini adalah grup keamanan yang terkait dengan instans Amazon EC2 yang menghosting tugas tersebut. Untuk tugas yang menggunakan mode `bridge` jaringan, jangan membuat pemetaan port apa pun yang menggunakan port. `24224`

- Untuk tugas yang menggunakan mode `bridge` jaringan, wadah dengan FireLens konfigurasi harus dimulai sebelum wadah aplikasi apa pun yang mengandalkannya dimulai. Untuk mengendalikan urutan mulai kontainer Anda, gunakan syarat dependensi dalam ketentuan tugas Anda. Untuk informasi selengkapnya, lihat Dependensi kontainer.

  > **Note**
  >
  > Jika Anda menggunakan parameter kondisi ketergantungan dalam definisi kontainer dengan FireLens konfigurasi, pastikan bahwa setiap kontainer memiliki persyaratan `START` atau `HEALTHY` kondisi.

- Secara default, FireLens tambahkan nama definisi klaster dan tugas serta Nama Sumber Daya Amazon (ARN) klaster sebagai kunci metadata ke log kontainer stdout/stderr Anda. Berikut ini adalah contoh format metadata.

  ```
  "ecs_cluster": "cluster-name",
  "ecs_task_arn": "arn:aws:ecs:region:111122223333:task/cluster-
  name/f2ad7dba413f45ddb4EXAMPLE",
  "ecs_task_definition": "task-def-name:revision",
  ```

  Jika Anda tidak ingin metadata di log Anda, atur `enable-ecs-log-metadata` ke `false` `firelensConfiguration` bagian definisi tugas.

  ```
  "firelensConfiguration":{
     "type":"fluentbit",
     "options":{
        "enable-ecs-log-metadata":"false",
        "config-file-type":"file",
        "config-file-value":"/extra.conf"
  }
  ```

Izin IAM yang diperlukan

Untuk menggunakan fitur ini, Anda harus membuat peran IAM untuk tugas Anda yang memberikan izin yang diperlukan untuk menggunakan AWS layanan apa pun yang diperlukan tugas. Misalnya, jika kontainer merutekan log ke Firehose, tugas tersebut memerlukan izin untuk memanggil `firehose:PutRecordBatch` API. Untuk informasi selengkapnya, silakan lihat Menambahkan dan Menghapus Izin Identitas IAM dalam Panduan Pengguna IAM.

Contoh berikut kebijakan IAM menambahkan izin yang diperlukan untuk routing log ke Firehose.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "firehose:PutRecordBatch"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Contoh berikut kebijakan IAM menambahkan izin yang diperlukan untuk routing log ke Amazon Logs. CloudWatch

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": [
   "logs:CreateLogStream",
   "logs:CreateLogGroup",
   "logs:PutLogEvents"
  ],
  "Resource": "*"
 }]
}
```

Tugas Anda mungkin juga memerlukan peran eksekusi tugas Amazon ECS dalam kondisi berikut. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

- Jika tugas Anda di-host di Fargate dan Anda menarik gambar kontainer dari Amazon ECR atau mereferensikan data sensitif dari AWS Secrets Manager konfigurasi log Anda, maka Anda harus menyertakan peran IAM eksekusi tugas.

- Jika Anda menentukan file konfigurasi khusus yang di-host di Amazon S3, peran IAM eksekusi tugas Anda harus menyertakan `s3:GetObject` izin untuk file konfigurasi dan izin `s3:GetBucketLocation` pada bucket Amazon S3 tempat file tersebut berada. Untuk informasi selengkapnya, lihat Menentukan Izin dalam Kebijakan di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

  Contoh berikut kebijakan IAM menambahkan izin yang diperlukan untuk mengambil file dari Amazon S3. Tentukan nama bucket Amazon S3 dan nama file konfigurasi Anda.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::examplebucket/folder_name/config_file_name"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::examplebucket"
      ]
    }
  ]
}
```

Batas buffer fluentd

Saat Anda membuat definisi tugas, Anda dapat menentukan jumlah peristiwa yang di-buffer dalam memori dengan menentukan nilai (dalam byte) di file. `log-driver-buffer-limit` Untuk informasi selengkapnya, lihat Driver logging fluentd di dokumentasi Docker.

Gunakan opsi ini ketika ada throughput tinggi, karena Docker mungkin kehabisan memori buffer dan membuang pesan buffer sehingga dapat menambahkan pesan baru. Log yang hilang mungkin menyulitkan pemecahan masalah. Menyetel batas buffer dapat membantu mencegah masalah ini.

Berikut ini menunjukkan sintaks untuk menentukan. `log-driver-buffer-limit` Ganti `my_service_` dengan nama layanan Anda. :

```
{
    "containerDefinitions": [
        {
            "essential": true,
            "image": "906394416424.dkr.ecr.us-west-2.amazonaws.com/aws-for-fluent-
bit:stable",
            "name": "my_service_log_router",
            "firelensConfiguration": {
                "type": "fluentbit"
            },
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "firelens-container",
                    "awslogs-region": "us-west-2",
                    "awslogs-create-group": "true",
                    "awslogs-stream-prefix": "firelens"
                }
            },
            "memoryReservation": 50
        },
        {
            "essential": true,
            "image": "httpd",
            "name": "app",
            "logConfiguration": {
                "logDriver": "awsfirelens",
                "options": {
                    "Name": "firehose",
                    "region": "us-west-2",
                    "delivery_stream": "my-stream",
                    "log-driver-buffer-limit": "2097152"
                }
            },
            "dependsOn": [
                {
```

```
                    "containerName": "log_router",
                    "condition": "START"
                }
            ],
            "memoryReservation": 100
        }
    ]
}
```

Pertimbangkan hal berikut saat menggunakan FireLens Amazon ECS dengan opsi batas buffer:

- Opsi ini didukung pada jenis peluncuran Amazon EC2 dan jenis peluncuran Fargate dengan versi platform atau yang lebih baru. `1.4.0`
- Opsi ini hanya valid ketika `logDriver` diatur ke`awsfirelens`.
- Batas buffer default adalah 1 MiB.
- Nilai yang valid adalah `0` dan `536870912` (512 MiB).
- Jumlah total memori yang dialokasikan pada tingkat tugas harus lebih besar dari jumlah memori yang dialokasikan untuk semua wadah selain batas buffer memori. Jumlah total memori buffer yang ditentukan harus kurang dari `536870912` (512MiB) ketika Anda tidak menentukan wadah dan nilai. `memory` `memoryReservation` Lebih khusus lagi, Anda dapat memiliki wadah aplikasi dengan driver `awsfirelens` log dan `log-driver-buffer-limit` opsi disetel ke 300 MiB. Namun, Anda tidak akan diizinkan untuk menjalankan tugas jika Anda memiliki lebih dari dua kontainer dengan `log-driver-buffer-limit` set ke 300 MiB (300 MiB * 2 > 512 MiB).

Menggunakan pustaka Fluent logger atau Log4j melalui TCP

Saat driver `awsfirelens` log ditentukan dalam definisi tugas, agen penampung Amazon ECS menyuntikkan variabel lingkungan berikut ke dalam wadah:

`FLUENT_HOST`

Alamat IP yang ditetapkan ke FireLens wadah.

`FLUENT_PORT`

Port tempat protokol Fluent Forward mendengarkan.

Anda dapat menggunakan variabel `FLUENT_HOST` dan `FLUENT_PORT` lingkungan untuk log langsung ke router log dari kode alih-alih melalui`stdout`. Untuk informasi lebih lanjut, lihat fluent-logger-golangdi GitHub.

- the section called “AWS untuk Fluent Bit gambar”
- the section called “Menentukan FireLens konfigurasi dalam definisi tugas”
- the section called “Penyaringan Fluentd dan log Fluent Bit”
- the section called “Contoh definisi tugas opsi logging”

AWS untuk Fluent Bit gambar

AWS menyediakan Fluent Bit gambar dengan plugin untuk CloudWatch Log dan Firehose. Sebaiknya gunakan Fluent Bit sebagai router log Anda karena memiliki tingkat pemanfaatan sumber daya yang lebih rendah daripadaFluentd. Untuk informasi selengkapnya, lihat CloudWatch Log untuk Bit Lancar dan Amazon Kinesis Firehose for Fluent Bit.

Gambar AWS for Fluent Bit tersedia di Amazon ECR di Galeri Publik Amazon ECR dan di repositori Amazon ECR di sebagian besar untuk ketersediaan tinggi. Wilayah AWS

Galeri Publik Amazon ECR

Fluent BitGambar AWS untuk tersedia di Galeri Publik Amazon ECR. Ini adalah lokasi yang disarankan untuk mengunduh Fluent Bit gambar AWS for karena ini adalah repositori publik dan tersedia untuk digunakan dari semua. Wilayah AWS Untuk informasi lebih lanjut, lihat aws-for-fluent-bitdi Galeri Publik Amazon ECR.

Linux

AWS Untuk Fluent Bit gambar di Galeri Publik Amazon ECR mendukung sistem operasi Amazon Linux dengan`ARM 64`, atau `x86-64` arsitektur.

Anda dapat menarik Fluent Bit gambar AWS for dari Galeri Publik Amazon ECR dengan menentukan URL repositori dengan tag gambar yang diinginkan. Tag gambar yang tersedia dapat ditemukan di tab Tag gambar di Galeri Publik Amazon ECR.

Berikut ini menunjukkan sintaksis yang digunakan untuk Docker CLI.

```
docker pull public.ecr.aws/aws-observability/aws-for-fluent-bit:tag
```

Misalnya, Anda dapat menarik stable terbaru AWS untuk Fluent Bit gambar menggunakan perintah CLI Docker ini.

```
docker pull public.ecr.aws/aws-observability/aws-for-fluent-bit:stable
```

> **Note**
>
> Penarikan yang tidak diautentikasi diperbolehkan, tetapi memiliki batas kecepatan yang lebih rendah daripada tarikan yang diautentikasi. Untuk mengautentikasi menggunakan AWS akun Anda sebelum menarik, gunakan perintah berikut.
>
> ```
> aws ecr-public get-login-password --region us-east-1 | docker login --username
>  AWS --password-stdin public.ecr.aws
> ```

Windows

Fluent BitGambar AWS for di Galeri Publik Amazon ECR mendukung `AMD64` arsitektur dengan sistem operasi berikut:

- Windows Server 2022 Lengkap
- Windows Server 2022 Inti
- Windows Server 2019 Full
- Windows Server 2019 Core

Wadah Windows yang ada di AWS Fargate tidak mendukung. FireLens

Anda dapat menarik Fluent Bit gambar AWS for dari Galeri Publik Amazon ECR dengan menentukan URL repositori dengan tag gambar yang diinginkan. Tag gambar yang tersedia dapat ditemukan di tab Tag gambar di Galeri Publik Amazon ECR.

Berikut ini menunjukkan sintaksis yang digunakan untuk Docker CLI.

```
docker pull public.ecr.aws/aws-observability/aws-for-fluent-bit:tag
```

Misalnya, Anda dapat menarik stable terbaru AWS untuk Fluent Bit gambar menggunakan perintah CLI Docker ini.

```
docker pull public.ecr.aws/aws-observability/aws-for-fluent-bit:windowsservercore-
stable
```

> **Note**
>
> Penarikan yang tidak diautentikasi diperbolehkan, tetapi memiliki batas kecepatan yang lebih rendah daripada tarikan yang diautentikasi. Untuk mengautentikasi menggunakan AWS akun Anda sebelum menarik, gunakan perintah berikut.
>
> ```
> aws ecr-public get-login-password --region us-east-1 | docker login --username
>  AWS --password-stdin public.ecr.aws
> ```

Amazon ECR

Gambar AWS for Fluent Bit tersedia di Amazon ECR untuk ketersediaan tinggi. Gambar-gambar ini tersedia di sebagian besar Wilayah AWS, termasuk AWS GovCloud (US).

Linux

Stabil terbaru AWS untuk URI gambar Fluent Bit dapat diambil menggunakan perintah berikut.

```
aws ssm get-parameters \
      --names /aws/service/aws-for-fluent-bit/stable \
      --region us-east-1
```

Semua versi image AWS for Fluent Bit dapat dicantumkan menggunakan perintah berikut untuk menanyakan parameter Systems Manager Parameter Store.

```
aws ssm get-parameters-by-path \
      --path /aws/service/aws-for-fluent-bit \
      --region us-east-1
```

Stabil terbaru AWS untuk image Fluent Bit dapat direferensikan dalam AWS CloudFormation template dengan mereferensikan nama penyimpanan parameter Systems Manager. Berikut ini adalah contohnya:

```
Parameters:
  FireLensImage:
```

```
    Description: Fluent Bit image for the FireLens Container
    Type: AWS::SSM::Parameter::Value<String>
    Default: /aws/service/aws-for-fluent-bit/stable
```

Windows

Stabil terbaru AWS untuk URI gambar Fluent Bit dapat diambil menggunakan perintah berikut.

```
aws ssm get-parameters \
      --names /aws/service/aws-for-fluent-bit/windowsservercore:stable \
      --region us-east-1
```

Semua versi image AWS for Fluent Bit dapat dicantumkan menggunakan perintah berikut untuk menanyakan parameter Systems Manager Parameter Store.

```
aws ssm get-parameters-by-path \
      --path /aws/service/aws-for-fluent-bit/windowsservercore \
      --region us-east-1
```

Stabil terbaru AWS untuk image Fluent Bit dapat direferensikan dalam AWS CloudFormation template dengan mereferensikan nama penyimpanan parameter Systems Manager. Berikut ini adalah contohnya:

```
Parameters:
  FireLensImage:
    Description: Fluent Bit image for the FireLens Container
    Type: AWS::SSM::Parameter::Value<String>
    Default: /aws/service/aws-for-fluent-bit/windowsservercore:stable
```

Menentukan FireLens konfigurasi dalam definisi tugas

Untuk menggunakan perutean log kustom denganFireLens, Anda harus menentukan yang berikut dalam definisi tugas Anda:

- Sebuah wadah log router yang berisi FireLens konfigurasi. Kami merekomendasikan bahwa kontainer ditandai sebagai `essential`.
- Satu kontainer aplikasi atau lebih yang berisi konfigurasi log menentukan driver log `awsfirelens`.
- Peran IAM tugas Amazon Resource Name (ARN) yang berisi izin yang diperlukan untuk tugas untuk merutekan log. Untuk informasi selengkapnya tentang izin untuk peran tugas IAM, lihat. Izin IAM yang diperlukan

Saat membuat definisi tugas baru menggunakan AWS Management Console, ada bagian FireLens integrasi yang memudahkan untuk menambahkan wadah router log. Untuk informasi selengkapnya, lihat Membuat definisi tugas menggunakan konsol.

Amazon ECS mengonversi konfigurasi log dan menghasilkan konfigurasi output Fluentd atau Fluent Bit. Konfigurasi output dipasang dalam kontainer perutean log di `/fluent-bit/etc/fluent-bit.conf` untuk Fluent Bit dan `/fluentd/etc/fluent.conf` untuk Fluentd.

> **Important**
>
> FireLensmendengarkan di port`24224`. Oleh karena itu, untuk memastikan bahwa router FireLens log tidak dapat dijangkau di luar tugas, Anda tidak boleh mengizinkan lalu lintas masuk pada port `24224` dalam grup keamanan yang digunakan tugas Anda. Untuk tugas yang menggunakan mode `awsvpc` jaringan, ini adalah grup keamanan yang terkait dengan tugas. Untuk tugas yang menggunakan mode `host` jaringan, ini adalah grup keamanan yang terkait dengan instans Amazon EC2 yang menghosting tugas tersebut. Untuk tugas yang menggunakan mode `bridge` jaringan, jangan membuat pemetaan port apa pun yang menggunakan port. `24224`

Secara default, Amazon ECS menambahkan bidang tambahan di entri log Anda yang membantu mengidentifikasi sumber log.

- `ecs_cluster`— Nama cluster yang menjadi bagian dari tugas.
- `ecs_task_arn`— Nama Sumber Daya Amazon (ARN) lengkap dari tugas yang menjadi bagian dari wadah.
- `ecs_task_definition`— Nama definisi tugas dan revisi yang digunakan tugas.
- `ec2_instance_id`— ID instans Amazon EC2 tempat penampung di-host. Bidang ini hanya berlaku untuk tugas yang menggunakan tipe peluncuran EC2.

Anda dapat mengatur `enable-ecs-log-metadata` ke `false` jika Anda tidak ingin metadata.

Contoh definisi tugas berikut mendefinisikan wadah router log yang menggunakan Fluent Bit untuk merutekan lognya ke CloudWatch Log. Ini juga mendefinisikan wadah aplikasi yang menggunakan konfigurasi log untuk merutekan log ke Amazon Data Firehose dan menetapkan memori yang digunakan untuk buffer peristiwa ke 2 MiB.

> **Note**
>
> Untuk contoh definisi tugas lainnya, lihat FireLenscontoh Amazon ECS di GitHub.

```
{
    "family": "firelens-example-firehose",
    "taskRoleArn": "arn:aws:iam::123456789012:role/ecs_task_iam_role",
    "containerDefinitions": [
        {
            "essential": true,
            "image": "906394416424.dkr.ecr.us-west-2.amazonaws.com/aws-for-fluent-
bit:stable",
            "name": "log_router",
            "firelensConfiguration": {
                  "type":"fluentbit",
                   "options":{
                       "enable-ecs-log-metadata":"true"
            },
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "firelens-container",
                    "awslogs-region": "us-west-2",
                    "awslogs-create-group": "true",
                    "awslogs-stream-prefix": "firelens"
                }
            },
            "memoryReservation": 50
        },
        {
            "essential": true,
            "image": "httpd",
            "name": "app",
            "logConfiguration": {
                "logDriver": "awsfirelens",
                "options": {
                    "Name": "firehose",
                    "region": "us-west-2",
                    "delivery_stream": "my-stream",
                    "log-driver-buffer-limit": "2097152"
                }
```

```
            },
            "memoryReservation": 100
        }
    ]
}
```

Pasangan kunci-nilai yang ditentukan sebagai opsi dalam `logConfiguration` objek digunakan untuk menghasilkan konfigurasi Fluentd atau Fluent Bit output. Berikut ini adalah contoh kode dari definisi Fluent Bit output.

```
[OUTPUT]
    Name   firehose
    Match  app-firelens*
    region us-west-2
    delivery_stream my-stream
```

> **Note**
>
> FireLens mengelola `match` konfigurasi. Konfigurasi ini tidak ditentukan dalam definisi tugas Anda.

Menentukan file konfigurasi kustom

Selain file konfigurasi yang dibuat secara otomatis yang FireLens dibuat atas nama Anda, Anda juga dapat menentukan file konfigurasi khusus. Format file konfigurasi adalah format asli untuk router log yang Anda gunakan. Untuk informasi selengkapnya, lihat Sintaks File Konfigurasi Fluentd dan File Konfigurasi Bit Lancar.

Dalam file konfigurasi kustom Anda, untuk tugas yang menggunakan mode `awsvpc` jaringan `bridge` atau, jangan setel input forward Fluentd atau Fluent Bit melalui TCP karena FireLens menambahkannya ke konfigurasi input.

FireLensKonfigurasi Anda harus berisi opsi berikut untuk menentukan file konfigurasi kustom:

`config-file-type`

Lokasi sumber file konfigurasi kustom. Opsi yang tersedia adalah `s3` atau `file`.

> **Note**
>
> Tugas yang di-host AWS Fargate hanya mendukung jenis file `file` konfigurasi.

`config-file-value`

Sumber untuk file konfigurasi kustom. Jika jenis file `s3` konfigurasi digunakan, nilai file konfigurasi adalah ARN lengkap dari bucket dan file Amazon S3. Jika jenis file `file` konfigurasi digunakan, nilai file konfigurasi adalah jalur lengkap dari file konfigurasi yang ada baik dalam gambar kontainer atau pada volume yang dipasang di wadah.

> **Important**
>
> Saat menggunakan file konfigurasi khusus, Anda harus menentukan jalur yang berbeda dari yang FireLens digunakan. Amazon ECS mencadangkan jalur `/fluent-bit/etc/fluent-bit.conf` file untuk Fluent Bit dan Fluentd. `/fluentd/etc/fluent.conf`

Contoh berikut menunjukkan sintaksis yang diperlukan saat menentukan konfigurasi kustom.

> **Important**
>
> Untuk menentukan file konfigurasi khusus yang di-host di Amazon S3, pastikan Anda telah membuat peran IAM eksekusi tugas dengan izin yang tepat. Untuk informasi selengkapnya, lihat Izin IAM yang diperlukan.

Berikut ini menunjukkan sintaks yang diperlukan saat menentukan konfigurasi kustom.

```
{
   "containerDefinitions":[
      {
         "essential":true,
         "image":"906394416424.dkr.ecr.us-west-2.amazonaws.com/aws-for-fluent-
bit:stable",
         "name":"log_router",
         "firelensConfiguration":{
            "type":"fluentbit",
```

```
            "options":{
               "config-file-type":"s3 | file",
               "config-file-value":"arn:aws:s3:::mybucket/fluent.conf | filepath"
            }
         }
      }
   ]
}
```

> **Note**
>
> Tugas yang dihosting AWS Fargate hanya mendukung jenis file `file` konfigurasi.

Penyaringan Fluentd dan log Fluent Bit

Mungkin ada saat-saat ketika Anda ingin memfilter log apa yang dikirim, misalnya mengirim log yang berisi kesalahan tertentu.

Fluentddan Fluent Bit keduanya mendukung pemfilteran log berdasarkan kontennya. FireLens menyediakan metode sederhana untuk mengaktifkan penyaringan ini.

Anda dapat mengonfigurasinya dalam opsi LogConfiguration dalam definisi container dengan menentukan ekspresi reguler yang cocok.

Kunci `exclude-pattern` menyebabkan semua log yang cocok dengan ekspresi reguler yang akan dijatuhkan. `include-pattern`Satu-satunya mengirimkan log yang cocok dengan ekspresi reguler yang ditentukan. Anda dapat menggunakan tombol ini secara terpisah atau bersama-sama

Contoh berikut mendemonstrasikan cara menggunakan filter ini.

```
{
   "containerDefinitions":[
      {
         "logConfiguration":{
            "logDriver":"awsfirelens",
            "options":{
               "@type":"cloudwatch_logs",
               "log_group_name":"firelens-testing",
               "auto_create_stream":"true",
               "use_tag_as_stream":"true",
```

```
                "region":"us-west-2",
                "exclude-pattern":"^[a-z][aeiou].*$",
                "include-pattern":"^.*[aeiou]$"
            }
        }
    }
  ]
}
```

Contoh definisi tugas opsi logging

Berikut ini adalah beberapa ketentuan tugas contoh yang mendemonstrasikan opsi perutean log umum. Untuk contoh lainnya, lihat FireLenscontoh Amazon ECS di GitHub.

> **Note**
>
> Parameter definisi `logConfiguration` tugas berikut yang ditunjukkan dalam contoh ini digunakan untuk mengirim log Fluent Bit Anda AWS ke CloudWatch. AWS merekomendasikan konfigurasi ini sehingga Anda memiliki informasi tambahan CloudWatch untuk memecahkan masalah Bit AWS Lancar.
>
> ```
> "logConfiguration": {
>           "logDriver": "awslogs",
>           "options": {
>               "awslogs-group": "firelens-container",
>               "awslogs-region": "us-west-2",
>               "awslogs-create-group": "true",
>               "awslogs-stream-prefix": "firelens"
>           }
>       },
> ```

Topik

- Meneruskan log ke Log CloudWatch
- Meneruskan log ke aliran pengiriman Amazon Data Firehose
- Meneruskan log ke domain Layanan Amazon OpenSearch
- Mengurai log kontainer yang merupakan serial JSON
- Meneruskan ke Fluentd atau Fluent Bit eksternal

Meneruskan log ke Log CloudWatch

> **Note**
>
> Untuk contoh lainnya, lihat FireLens contoh Amazon ECS di GitHub.

Contoh definisi tugas berikut menunjukkan cara menentukan konfigurasi log yang meneruskan log ke grup log Log. CloudWatch Untuk informasi selengkapnya, lihat Apa itu Amazon CloudWatch Logs? di Panduan Pengguna CloudWatch Log Amazon.

Dalam opsi konfigurasi log, tentukan nama grup log dan nama Wilayah AWS itu ada. Agar Fluent Bit membuat grup log atas nama Anda, tentukan`"auto_create_group":"true"`, untuk mengatur fluentd-buffer-limit penggunaannya`log-driver-buffer-limit`. Anda juga dapat menentukan ID tugas sebagai awalan log stream, yang membantu dalam penyaringan. Untuk informasi selengkapnya, lihat Plugin Bit Lancar untuk CloudWatch Log.

```
{
    "family": "firelens-example-cloudwatch",
    "taskRoleArn": "arn:aws:iam::123456789012:role/ecs_task_iam_role",
    "containerDefinitions": [
        {
            "essential": true,
            "image": "906394416424.dkr.ecr.us-west-2.amazonaws.com/aws-for-fluent-
bit:latest",
            "name": "log_router",
            "firelensConfiguration": {
                "type": "fluentbit"
            },
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "firelens-container",
                    "awslogs-region": "us-west-2",
                    "awslogs-create-group": "true",
                    "awslogs-stream-prefix": "firelens"
                }
            },
            "memoryReservation": 50
        },
        {
            "essential": true,
```

```
            "image": "httpd",
            "name": "app",
            "logConfiguration": {
                "logDriver":"awsfirelens",
                "options": {
                   "Name": "cloudwatch",
                   "region": "us-west-2",
                   "log_group_name": "firelens-blog",
                   "auto_create_group": "true",
                   "log_stream_prefix": "from-fluent-bit",
                   "log-driver-buffer-limit": "2097152"
                }
            },
            "memoryReservation": 100
            }
    ]
}
```

Meneruskan log ke aliran pengiriman Amazon Data Firehose

> **Note**
>
> Untuk contoh lainnya, lihat FireLens contoh Amazon ECS di GitHub.

Contoh definisi tugas berikut menunjukkan cara menentukan konfigurasi log yang meneruskan log ke aliran pengiriman Amazon Data Firehose. Aliran pengiriman Firehose pasti sudah ada. Untuk informasi selengkapnya, lihat Membuat Aliran Pengiriman Firehose Data Amazon di Panduan Pengembang Amazon Data Firehose.

Dalam opsi konfigurasi log, tentukan nama aliran pengiriman dan Wilayah tempatnya berada. Untuk informasi selengkapnya, lihat Plugin Fluent Bit untuk Amazon Kinesis Firehose.

```
{
 "family": "firelens-example-firehose",
 "taskRoleArn": "arn:aws:iam::123456789012:role/ecs_task_iam_role",
 "containerDefinitions": [
  {
   "essential": true,
   "image": "906394416424.dkr.ecr.us-west-2.amazonaws.com/aws-for-fluent-bit:stable",
   "name": "log_router",
   "firelensConfiguration": {
```

```
    "type": "fluentbit"
   },
   "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
     "awslogs-group": "firelens-container",
     "awslogs-region": "us-west-2",
     "awslogs-create-group": "true",
     "awslogs-stream-prefix": "firelens"
    }
   },
   "memoryReservation": 50
   },
   {
    "essential": true,
    "image": "httpd",
    "name": "app",
    "logConfiguration": {
     "logDriver":"awsfirelens",
     "options": {
     "Name": "firehose",
     "region": "us-west-2",
     "delivery_stream": "my-stream"
    }
   },
   "memoryReservation": 100
            }
 ]
}
```

Meneruskan log ke domain Layanan Amazon OpenSearch

> **Note**
>
> Untuk contoh lainnya, lihat FireLens contoh Amazon ECS di GitHub.

Contoh definisi tugas berikut menunjukkan cara menentukan konfigurasi log yang meneruskan log ke domain OpenSearch Layanan Amazon;. Domain OpenSearch Layanan Amazon harus sudah ada. Untuk informasi selengkapnya, lihat Apa itu OpenSearch Layanan Amazon di Panduan Pengembang OpenSearch Layanan Amazon.

Dalam opsi konfigurasi log, tentukan opsi log yang diperlukan untuk Integrasi OpenSearch layanan. Untuk informasi selengkapnya, lihat Fluent Bit for Amazon OpenSearch Service.

```
{
    "family": "firelens-example-opensearch",
    "taskRoleArn": "arn:aws:iam::123456789012:role/ecs_task_iam_role",
    "containerDefinitions": [
        {
            "essential": true,
            "image": "906394416424.dkr.ecr.us-west-2.amazonaws.com/aws-for-fluent-
bit:stable",
            "name": "log_router",
            "firelensConfiguration": {
                "type": "fluentbit"
            },
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "firelens-container",
                    "awslogs-region": "us-west-2",
                    "awslogs-create-group": "true",
                    "awslogs-stream-prefix": "firelens"
                }
            },
            "memoryReservation": 50
        },
        {
            "essential": true,
            "image": "httpd",
            "name": "app",
            "logConfiguration": {
                "logDriver": "awsfirelens",
                "options": {
                    "Name": "es",
                    "Host": "vpc-fake-domain-ke7thhzo07jawrhmz6mb7ite7y.us-
west-2.es.amazonaws.com",
                    "Port": "443",
                    "Index": "my_index",
                    "Type": "my_type",
                    "AWS_Auth": "On",
                    "AWS_Region": "us-west-2",
                    "tls": "On"
                }
```

```
            },
            "memoryReservation": 100
        }
    ]
}
```

Mengurai log kontainer yang merupakan serial JSON

> **Note**
>
> Untuk contoh lainnya, lihat FireLens contoh Amazon ECS di GitHub.

Dimulai dengan AWS untuk Fluent Bit versi 1.3, ada parser JSON yang disertakan dalam gambar AWS for Fluent Bit. Contoh berikut menunjukkan cara mereferensikan parser JSON dalam FireLens konfigurasi definisi tugas Anda.

```
"firelensConfiguration": {
    "type": "fluentbit",
    "options": {
        "config-file-type": "file",
        "config-file-value": "/fluent-bit/configs/parse-json.conf"
    }
},
```

File konfigurasi Fluent Bit mem-parsing log apa pun yang ada di JSON (misalnya, jika log di tujuan Anda terlihat seperti berikut tanpa penguraian JSON).

```
{
    "source": "stdout",
    "log": "{\"requestID\": \"b5d716fca19a4252ad90e7b8ec7cc8d2\", \"requestInfo\":
 {\"ipAddress\": \"204.16.5.19\", \"path\": \"/activate\", \"user\": \"TheDoctor\"}}",
    "container_id": "e54cccfac2b87417f71877907f6787906842004282806 7ae0867e60a63529d35",
    "container_name": "/ecs-demo-6-container2-a4eafbb3d4c7f1e16e00"
    "ecs_cluster": "mycluster",
    "ecs_task_arn": "arn:aws:ecs:us-east-2:01234567891011:task/
mycluster/3de392df-6bfa-470b-97ed-aa6f482cd7a6",
    "ecs_task_definition": "demo:7"
    "ec2_instance_id": "i-06bc83dbc2ac2fdf8"
}
```

Dengan penguraian JSON, log terlihat seperti berikut ini.

```
{
    "source": "stdout",
    "container_id": "e54cccfac2b87417f71877907f6787906842004282806 7ae0867e60a63529d35",
    "container_name": "/ecs-demo-6-container2-a4eafbb3d4c7f1e16e00"
    "ecs_cluster": "mycluster",
    "ecs_task_arn": "arn:aws:ecs:us-east-2:01234567891011:task/
mycluster/3de392df-6bfa-470b-97ed-aa6f482cd7a6",
    "ecs_task_definition": "demo:7"
    "ec2_instance_id": "i-06bc83dbc2ac2fdf8"
    "requestID": "b5d716fca19a4252ad90e7b8ec7cc8d2",
    "requestInfo": {
        "ipAddress": "204.16.5.19",
        "path": "/activate",
        "user": "TheDoctor"
    }
}
```

JSON serial diperluas ke bidang tingkat atas dalam output JSON akhir. Untuk informasi selengkapnya tentang penguraian JSON, lihat Parser dalam dokumentasi Fluent Bit.

Meneruskan ke Fluentd atau Fluent Bit eksternal

> **Note**
>
> Untuk contoh lainnya, lihat FireLens contoh Amazon ECS di GitHub.

Contoh ketentuan tugas berikut mendemonstrasikan bagaimana cara untuk menentukan konfigurasi log yang meneruskan log ke host Fluentd eksternal atau Fluent Bit. Tentukan `host` dan `port` untuk lingkungan Anda.

```
{
 "family": "firelens-example-forward",
 "taskRoleArn": "arn:aws:iam::123456789012:role/ecs_task_iam_role",
 "containerDefinitions": [
  {
   "essential": true,
   "image": "906394416424.dkr.ecr.us-west-2.amazonaws.com/aws-for-fluent-bit:stable",
   "name": "log_router",
```

```
    "firelensConfiguration": {
     "type": "fluentbit"
    },
    "logConfiguration": {
     "logDriver": "awslogs",
     "options": {
      "awslogs-group": "firelens-container",
      "awslogs-region": "us-west-2",
      "awslogs-create-group": "true",
      "awslogs-stream-prefix": "firelens"
     }
    },
    "memoryReservation": 50
    },
    {
     "essential": true,
     "image": "httpd",
     "name": "app",
     "logConfiguration": {
      "logDriver":"awsfirelens",
      "options": {
      "Name": "forward",
      "Host": "fluentdhost",
      "Port": "24224"
     }
    },
    "memoryReservation": 100
   }
  ]
}
```

Autentikasi registri privat untuk tugas

Gunakan registri pribadi untuk menyimpan kredensyal Anda AWS Secrets Manager, dan kemudian mereferensikannya dalam definisi tugas Anda. Ini menyediakan cara untuk mereferensikan gambar kontainer yang ada di pendaftar pribadi di luar AWS yang memerlukan otentikasi dalam definisi tugas Anda. Fitur ini didukung oleh tugas yang dihosting di Fargate, instans Amazon EC2, dan instans eksternal yang menggunakan Amazon ECS Anywhere.

> **Important**
>
> Jika definisi tugas Anda mereferensikan gambar yang disimpan di Amazon ECR, topik ini tidak berlaku. Untuk informasi selengkapnya, lihat Menggunakan Gambar Amazon ECR dengan Amazon ECS di Panduan Pengguna Amazon Elastic Container Registry.

Untuk tugas yang dihosting di instans Amazon EC2, fitur ini memerlukan versi `1.19.0` atau yang lebih baru dari agen penampung. Akan tetapi, kami merekomendasikan untuk menggunakan versi agen kontainer terbaru. Untuk informasi tentang cara memeriksa versi agen Anda dan memperbarui ke versi terbaru, lihatMemperbarui agen kontainer Amazon ECS.

Untuk tugas yang dihosting di Fargate, fitur ini memerlukan versi platform `1.2.0` atau yang lebih baru. Untuk informasi, lihat Versi platform Fargate Linux.

Dalam definisi kontainer Anda, tentukan `repositoryCredentials` objek dengan detail rahasia yang Anda buat. Rahasia yang Anda referensikan bisa dari akun yang berbeda Wilayah AWS atau berbeda dari tugas yang menggunakannya.

> **Note**
>
> Saat menggunakan Amazon ECS API, AWS CLI, atau AWS SDK, jika rahasia ada Wilayah AWS sama dengan tugas yang Anda luncurkan maka Anda dapat menggunakan ARN lengkap atau nama rahasia. Jika rahasia ada di akun yang berbeda, ARN penuh rahasia harus ditentukan. Saat menggunakan AWS Management Console, ARN penuh rahasia harus ditentukan selalu.

Berikut ini adalah cuplikan definisi tugas yang menunjukkan parameter yang diperlukan:

Gantikan repo pribadi dengan nama host repositori pribadi dan gambar pribadi dengan nama gambar.

```
"containerDefinitions": [
    {
        "image": "private-repo/private-image",
        "repositoryCredentials": {
            "credentialsParameter":
 "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name"
```

```
        }
    }
]
```

> **Note**
>
> Metode lain untuk mengaktifkan otentikasi registri pribadi menggunakan variabel lingkungan agen penampung Amazon ECS untuk mengautentikasi ke pendaftar pribadi. Metode ini hanya didukung untuk tugas yang dihosting di instans Amazon EC2. Untuk informasi selengkapnya, lihat Autentikasi registri privat untuk instans kontainer.

Untuk menggunakan registri pribadi

1. Peran eksekusi tugas Amazon ECS diperlukan untuk menggunakan fitur ini. Hal ini mengizinkan agen kontainer untuk menarik citra kontainer. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

   Untuk memberikan akses ke rahasia yang Anda buat, tambahkan izin berikut sebagai kebijakan inline ke peran eksekusi tugas. Untuk informasi selengkapnya, lihat Menambahkan dan Menghapus Kebijakan IAM.

   - `secretsmanager:GetSecretValue`
   - `kms:Decrypt`—Diperlukan hanya jika kunci Anda menggunakan kunci KMS khusus dan bukan kunci default. Nama Sumber Daya Amazon (ARN) untuk kunci kustom Anda harus ditambahkan sebagai sumber daya.

   Berikut ini adalah contoh kebijakan inline yang menambahkan izin.

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "kms:Decrypt",
                   "secretsmanager:GetSecretValue"
               ],
               "Resource": [
   ```

```
 "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:secret_name",
                "arn:aws:kms:<region>:<aws_account_id>:key/key_id"
            ]
        }
    ]
}
```

2. Gunakan AWS Secrets Manager untuk membuat rahasia untuk kredensil registri pribadi Anda. Untuk informasi tentang cara membuat rahasia, lihat Membuat AWS Secrets Manager rahasia di Panduan AWS Secrets Manager Pengguna.

   Masukkan kredensi registri pribadi Anda menggunakan format berikut:

   ```
   {
     "username" : "privateRegistryUsername",
     "password" : "privateRegistryPassword"
   }
   ```

3. Mendaftarkan ketentuan tugas. Untuk informasi selengkapnya, lihat the section called “Membuat definisi tugas menggunakan konsol”.

Gunakan parameter definisi tugas untuk meneruskan variabel lingkungan ke wadah

> **Important**
>
> Sebaiknya simpan data sensitif Anda baik dalam parameter AWS Secrets Manager rahasia atau AWS Systems Manager Parameter Store. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah.
> Variabel lingkungan yang ditentukan dalam definisi tugas dapat dibaca oleh semua pengguna dan peran yang diizinkan `DescribeTaskDefinition` tindakan untuk definisi tugas.

Anda dapat meneruskan variabel lingkungan ke kontainer Anda dengan cara berikut:

- Secara individual menggunakan parameter ketentuan kontainer `environment`. Ini memetakan ke `--env` opsi untuk docker run.

- Secara massal, menggunakan parameter definisi `environmentFiles` kontainer untuk mencantumkan satu atau lebih file yang berisi variabel lingkungan. File harus di-host di Amazon S3. Ini memetakan ke `--env-file` opsi untuk docker run.

Berikut ini adalah potongan ketentuan tugas yang menunjukkan cara menentukan variabel lingkungan individual.

```
{
    "family": "",
    "containerDefinitions": [
        {
            "name": "",
            "image": "",
            ...
            "environment": [
                {
                    "name": "variable",
                    "value": "value"
                }
            ],
            ...
        }
    ],
    ...
}
```

Gunakan file untuk meneruskan variabel lingkungan ke wadah

> **Important**
>
> Sebaiknya simpan data sensitif Anda baik dalam parameter AWS Secrets Manager rahasia atau AWS Systems Manager Parameter Store. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah.
> File variabel lingkungan adalah objek di Amazon S3 dan semua pertimbangan keamanan Amazon S3 berlaku.

Anda dapat membuat file variabel lingkungan dan menyimpannya di Amazon S3 untuk meneruskan variabel lingkungan ke wadah Anda.

Dengan menentukan variabel lingkungan dalam file, Anda dapat menyuntikkan variabel lingkungan secara massal. Dalam definisi container Anda, tentukan `environmentFiles` objek dengan daftar bucket Amazon S3 yang berisi file variabel lingkungan Anda.

Amazon ECS tidak memberlakukan batas ukuran pada variabel lingkungan, tetapi file variabel lingkungan yang besar mungkin mengisi ruang disk. Setiap tugas yang menggunakan file variabel lingkungan menyebabkan salinan file diunduh ke disk. Amazon ECS menghapus file sebagai bagian dari pembersihan tugas.

Untuk informasi tentang variabel lingkungan yang didukung, lihat Parameter definisi kontainer lanjutan- Lingkungan.

Pertimbangan

Pertimbangkan hal berikut saat menentukan file variabel lingkungan dalam definisi wadah.

- Untuk tugas Amazon ECS di Amazon EC2, instans penampung Anda mengharuskan agen penampung adalah `1.39.0` versi atau yang lebih baru untuk menggunakan fitur ini. Untuk informasi tentang cara memeriksa versi agen Anda dan memperbarui ke versi terbaru, lihatMemperbarui agen kontainer Amazon ECS.
- Untuk tugas Amazon ECS di AWS Fargate, tugas Anda harus menggunakan `1.4.0` versi platform atau yang lebih baru (Linux) untuk menggunakan fitur ini. Untuk informasi selengkapnya, lihat Versi platform Fargate Linux.

  Verifikasi bahwa variabel didukung untuk platform sistem operasi. Lihat informasi yang lebih lengkap di the section called “Definisi kontainer” dan the section called “Parameter ketentuan tugas lainnya”.
- File harus menggunakan ekstensi file `.env` dan pengodean UTF-8.
- Ada batas 10 file per definisi tugas.
- Setiap baris dalam file lingkungan harus berisi variabel lingkungan dalam `VARIABLE=VALUE` format. Spasi atau tanda kutip disertakan sebagai bagian dari nilai untuk file Amazon ECS. Baris yang diawali dengan # diperlakukan sebagai komentar dan diabaikan. Untuk informasi selengkapnya tentang sintaks file variabel lingkungan, lihat Mendeklarasikan variabel lingkungan default dalam file.

  Berikut ini adalah sintaks yang sesuai.

```
#This is a comment and will be ignored
```

```
VARIABLE=VALUE
ENVIRONMENT=PRODUCTION
```

- Jika ada variabel lingkungan yang ditentukan menggunakan parameter `environment` dalam ketentuan kontainer, maka akan lebih diutamakan daripada variabel yang terkandung dalam file lingkungan.
- Jika beberapa file lingkungan ditentukan dan mereka berisi variabel yang sama, mereka diproses dalam urutan entri. Ini berarti bahwa nilai pertama dari variabel digunakan dan nilai-nilai berikutnya dari variabel duplikat diabaikan. Kami merekomendasikan agar Anda menggunakan nama variabel yang unik.
- Jika file lingkungan ditentukan sebagai penimpaan kontainer, itu digunakan. Selain itu, file lingkungan lain yang ditentukan dalam definisi kontainer diabaikan.
- Aturan berikut berlaku untuk jenis peluncuran Fargate:
  - File ini ditangani seperti file env Docker asli.
  - Tidak ada dukungan untuk penanganan pelarian shell.
  - Titik masuk kontainer menyelingi nilai-nilai. `VARIABLE`

Izin IAM yang diperlukan

Peran eksekusi tugas Amazon ECS diperlukan untuk menggunakan fitur ini. Ini memungkinkan agen kontainer untuk menarik file variabel lingkungan dari Amazon S3. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

Untuk menyediakan akses ke objek Amazon S3 yang Anda buat, tambahkan izin berikut secara manual sebagai kebijakan inline ke peran eksekusi tugas. Gunakan `Resource` parameter untuk cakupan izin ke bucket Amazon S3 yang berisi file variabel lingkungan. Untuk informasi selengkapnya, lihat Menambahkan dan Menghapus Kebijakan IAM.

- `s3:GetObject`
- `s3:GetBucketLocation`

Dalam contoh berikut, izin ditambahkan ke kebijakan sebaris.

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::examplebucket/folder_name/env_file_name"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::examplebucket"
      ]
    }
  ]
}
```

Contoh

Berikut ini adalah potongan ketentuan tugas yang menunjukkan cara menentukan file variabel lingkungan.

```
{
    "family": "",
    "containerDefinitions": [
        {
            "name": "",
            "image": "",
            ...
            "environmentFiles": [
                {
                    "value": "arn:aws:s3:::s3_bucket_name/envfile_object_name.env",
                    "type": "s3"
                }
            ],
            ...
        }
    ],
    ...
```

```
}
```

Meneruskan data sensitif ke wadah

Anda dapat meneruskan data sensitif dengan aman, seperti kredensyal ke database, ke dalam wadah Anda.

Anda dapat menggunakan AWS Secrets Manager atau sebagai parameter di AWS Systems Manager Parameter Store untuk menyimpan rahasia.

Anda dapat mengambil rahasia secara terprogram dari aplikasi, atau dengan menggunakan variabel lingkungan.

Untuk memulai, pertama-tama simpan data sensitif sebagai rahasia di AWS Secrets Manager atau sebagai parameter di AWS Systems Manager Parameter Store. Kemudian, gunakan salah satu cara berikut untuk mengekspos rahasia ke wadah.

Topik

Ambil rahasia Secrets Manager secara terprogram

Gunakan Secrets Manager untuk melindungi data sensitif dan memutar, mengelola, dan mengambil kredensyal database, kunci API, dan rahasia lainnya sepanjang siklus hidupnya.

Alih-alih hardcoding informasi sensitif dalam teks biasa di aplikasi Anda, Anda dapat menggunakan Secrets Manager untuk menyimpan data sensitif.

Kami merekomendasikan metode ini untuk mengambil data sensitif karena jika rahasia Secrets Manager kemudian diperbarui, aplikasi secara otomatis mengambil versi terbaru dari rahasia.

Buat rahasia di Secrets Manager. Setelah Anda membuat rahasia Secrets Manager, perbarui kode aplikasi Anda untuk mengambil rahasia.

Pertimbangan

Tinjau pertimbangan berikut sebelum mengamankan data sensitif di Secrets Manager.

- Hanya rahasia yang menyimpan data teks, yang merupakan rahasia yang dibuat dengan `SecretString` parameter CreateSecretAPI, yang didukung. Rahasia yang menyimpan data biner, yang merupakan rahasia yang dibuat dengan SecretBinary parameter CreateSecretAPI tidak didukung.
- Gunakan titik akhir VPC antarmuka untuk meningkatkan kontrol keamanan. Anda harus membuat titik akhir VPC antarmuka untuk Secrets Manager. Untuk informasi tentang titik akhir VPC, lihat Membuat titik akhir VPC di Panduan Pengguna.AWS Secrets Manager
- VPC yang digunakan tugas Anda harus menggunakan resolusi DNS.

Izin IAM yang diperlukan

Untuk menggunakan fitur ini, Anda harus memiliki peran tugas Amazon ECS dan mereferensikannya dalam definisi tugas Anda. Untuk informasi selengkapnya, lihat Tugas peran IAM.

Untuk memberikan akses ke rahasia Secrets Manager yang Anda buat, tambahkan izin berikut secara manual ke peran eksekusi tugas. Untuk informasi tentang cara mengelola izin, lihat Menambahkan dan Menghapus izin identitas IAM di Panduan Pengguna IAM.

- `secretsmanager:GetSecretValue`— Diperlukan jika Anda mereferensikan rahasia Secrets Manager. Menambahkan izin untuk mengambil rahasia dari Secrets Manager.

Contoh kebijakan berikut menambahkan izin yang diperlukan.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": [
        "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name"
      ]
    }
  ]
```

```
}
```

Buat rahasia Secrets Manager

Anda dapat menggunakan konsol Secrets Manager untuk membuat rahasia untuk data sensitif Anda. Untuk informasi tentang cara membuat rahasia, lihat Membuat AWS Secrets Manager rahasia di Panduan AWS Secrets Manager Pengguna.

Perbarui aplikasi Anda untuk mengambil rahasia Secrets Manager secara terprogram

Anda dapat mengambil rahasia dengan panggilan ke Secrets Manager API langsung dari aplikasi Anda. Untuk informasi tentang cara memperbarui kode Anda untuk meminta rahasia, lihat Mengambil rahasia dari AWS Secrets Manager dalam Panduan AWS Secrets Manager Pengguna.

Untuk mengambil data sensitif yang disimpan di dalam AWS Secrets Manager, lihat Contoh kode untuk AWS Secrets Manager menggunakan AWS SDK di Pustaka Kode Contoh Kode AWS SDK.

Ambil AWS Systems Manager parameter Parameter Store secara terprogram

Systems Manager Parameter Store menyediakan penyimpanan dan pengelolaan rahasia yang aman. Anda dapat menyimpan data seperti kata sandi, string database, ID instans Amazon Elastic Compute Cloud (Amazon EC2) dan ID AMI, dan kode lisensi sebagai nilai parameter. Anda dapat menyimpan nilai sebagai teks biasa atau data terenkripsi.

Alih-alih hardcoding informasi sensitif dalam teks biasa di aplikasi Anda, Anda dapat menggunakan Secrets Manager untuk menyimpan data sensitif.

Kami merekomendasikan metode ini untuk mengambil data sensitif karena jika parameter Systems Manager Parameter Store kemudian diperbarui, aplikasi secara otomatis mengambil versi terbaru.

Buat rahasia di Secrets Manager. Setelah Anda membuat rahasia Secrets Manager, perbarui kode aplikasi Anda untuk mengambil rahasia.

Pertimbangan

Tinjau pertimbangan berikut sebelum mengamankan data sensitif di Systems Manager Parameter Store.

- Hanya rahasia yang menyimpan data teks yang didukung. Rahasia yang menyimpan data biner tidak didukung.
- Gunakan titik akhir VPC antarmuka untuk meningkatkan kontrol keamanan.
- VPC yang digunakan tugas Anda harus menggunakan resolusi DNS.

Izin IAM yang diperlukan

Untuk menggunakan fitur ini, Anda harus memiliki peran tugas Amazon ECS dan mereferensikannya dalam definisi tugas Anda. Hal ini memungkinkan agen kontainer untuk menarik sumber daya Systems Manager yang diperlukan. Untuk informasi selengkapnya, lihat Tugas peran IAM.

> **Important**
>
> Untuk tugas yang menggunakan tipe peluncuran EC2, Anda harus menggunakan variabel konfigurasi agen ECS `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE=true` untuk menggunakan fitur ini. Anda dapat menambahkannya ke file `./etc/ecs/ecs.config` selama pembuatan instans kontainer atau Anda dapat menambahkannya ke instans yang ada, lalu memulai ulang agen ECS. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.

Untuk memberikan akses ke parameter Penyimpanan Parameter Systems Manager yang Anda buat, tambahkan izin berikut secara manual sebagai kebijakan ke peran eksekusi tugas. Untuk informasi tentang cara mengelola izin, lihat Menambahkan dan Menghapus izin identitas IAM di Panduan Pengguna IAM.

- `ssm:GetParameters`— Diperlukan jika Anda mereferensikan parameter Systems Manager Parameter Store dalam definisi tugas. Menambahkan izin untuk mengambil parameter Systems Manager.
- `secretsmanager:GetSecretValue`— Diperlukan jika Anda mereferensikan rahasia Secrets Manager baik secara langsung atau jika parameter Parameter Store Systems Manager Anda mereferensikan rahasia Secrets Manager dalam definisi tugas. Menambahkan izin untuk mengambil rahasia dari Secrets Manager.
- `kms:Decrypt`— Diperlukan hanya jika rahasia Anda menggunakan kunci yang dikelola pelanggan dan bukan kunci default. ARN untuk kunci khusus Anda harus ditambahkan sebagai sumber daya. Menambahkan izin untuk mendekripsi kunci yang dikelola pelanggan.

Contoh kebijakan berikut menambahkan izin yang diperlukan:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": [
        "ssm:GetParameters",
        "secretsmanager:GetSecretValue",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:ssm:region:aws_account_id:parameter/parameter_name",
        "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name",
        "arn:aws:kms:region:aws_account_id:key/key_id"
      ]
    }
  ]
}
```

Buat parameternya

Anda dapat menggunakan konsol Systems Manager untuk membuat parameter Systems Manager Parameter Store untuk data sensitif Anda. Untuk informasi selengkapnya, lihat Membuat parameter Systems Manager (konsol) atau Membuat parameter Systems Manager (AWS CLI) di Panduan AWS Systems Manager Pengguna.

Perbarui aplikasi Anda untuk mengambil rahasia Systems Manager Parameter Store secara terprogram

Untuk mengambil data sensitif yang disimpan dalam parameter Systems Manager Parameter Store, lihat Contoh kode untuk Systems Manager menggunakan AWS SDK di AWS SDK Code Examples Code Library.

Ambil rahasia Secrets Manager melalui variabel lingkungan

Saat Anda menyuntikkan rahasia sebagai variabel lingkungan, Anda dapat menentukan konten lengkap rahasia, kunci JSON tertentu di dalam rahasia, atau versi tertentu dari rahasia yang akan disuntikkan. Hal ini membantu Anda mengontrol data sensitif yang diekspos ke kontainer Anda. Untuk informasi selengkapnya tentang versioning rahasia, lihat Konsep dan Istilah Penting untuk AWS Secrets Manager dalam Panduan Pengguna AWS Secrets Manager .

Pertimbangan

Berikut ini harus dipertimbangkan saat menggunakan variabel lingkungan untuk menyuntikkan rahasia Secrets Manager ke dalam wadah.

- Data sensitif disuntikkan ke dalam kontainer Anda ketika kontainer awalnya dimulai. Jika rahasia kemudian diperbarui atau diputar, kontainer tidak akan menerima nilai yang diperbarui secara otomatis. Anda harus menjalankan tugas baru atau jika tugas Anda adalah bagian dari layanan Anda dapat memperbarui layanan dan menggunakan Memaksa deployment baru untuk memaksa layanan untuk meluncurkan tugas baru.
- Untuk tugas Amazon ECS aktif AWS Fargate, hal-hal berikut harus dipertimbangkan:
  - Untuk menyuntikkan konten penuh rahasia sebagai variabel lingkungan atau dalam konfigurasi log, Anda harus menggunakan versi `1.3.0` platform atau yang lebih baru. Untuk informasi, lihat Versi platform Fargate Linux.
  - Untuk menyuntikkan kunci JSON tertentu atau versi rahasia sebagai variabel lingkungan atau dalam konfigurasi log, Anda harus menggunakan versi platform `1.4.0` atau yang lebih baru (Linux) atau `1.0.0` (Windows). Untuk informasi, lihat Versi platform Fargate Linux.
- Untuk tugas Amazon ECS di EC2, hal-hal berikut harus dipertimbangkan:
  - Untuk menyuntikkan rahasia menggunakan kunci JSON atau versi rahasia tertentu, instans kontainer Anda harus memiliki versi `1.37.0` agen kontainer atau yang lebih baru. Namun, kami merekomendasikan untuk menggunakan versi agen kontainer terbaru. Untuk informasi tentang memeriksa versi agen Anda dan memperbarui ke versi terbaru, lihat Memperbarui agen kontainer Amazon ECS.

    Untuk menyuntikkan konten penuh rahasia sebagai variabel lingkungan atau untuk menyuntikkan rahasia dalam konfigurasi log, instans kontainer Anda harus memiliki versi `1.22.0` agen kontainer atau yang lebih baru.
- Gunakan titik akhir VPC antarmuka untuk meningkatkan kontrol keamanan. Anda harus membuat titik akhir VPC antarmuka untuk Secrets Manager. Untuk informasi tentang titik akhir VPC, lihat Membuat titik akhir VPC di Panduan Pengguna.AWS Secrets Manager
- Untuk tugas Windows yang dikonfigurasi untuk menggunakan driver pencatatan `awslogs`, Anda juga harus mengatur variabel lingkungan `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE` pada instans kontainer Anda. Ini dapat dilakukan dengan Data Pengguna dengan sintaks berikut:

```
<powershell>
[Environment]::SetEnvironmentVariable("ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE",
 $TRUE, "Machine")
Initialize-ECSAgent -Cluster <cluster name> -EnableTaskIAMRole -LoggingDrivers
 '["json-file","awslogs"]'
</powershell>
```

Izin IAM

Untuk menggunakan fitur ini, Anda harus memiliki peran eksekusi tugas Amazon ECS dan mereferensikannya dalam definisi tugas Anda. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

Untuk menyediakan akses ke rahasia Secrets Manager yang Anda buat, tambahkan izin berikut secara manual sebagai kebijakan inline ke peran eksekusi tugas. Untuk informasi selengkapnya, lihat Menambahkan dan Menghapus Kebijakan IAM.

- `secretsmanager:GetSecretValue`—Diperlukan jika Anda mereferensikan rahasia Secrets Manager. Menambahkan izin untuk mengambil rahasia dari Secrets Manager.
- `kms:Decrypt`—Diperlukan hanya jika rahasia Anda menggunakan kunci yang dikelola pelanggan dan bukan kunci default. ARN untuk kunci yang dikelola pelanggan Anda harus ditambahkan sebagai sumber daya. Menambahkan izin untuk mendekripsi kunci yang dikelola pelanggan.

Contoh kebijakan berikut menambahkan izin yang diperlukan:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name",
        "arn:aws:kms:region:aws_account_id:key/key_id"
      ]
    }
  ]
}
```

Buat AWS Secrets Manager rahasianya

Anda dapat menggunakan konsol Secrets Manager untuk membuat rahasia untuk data sensitif Anda. Untuk informasi selengkapnya, lihat Membuat AWS Secrets Manager rahasia di Panduan AWS Secrets Manager Pengguna.

Tambahkan variabel lingkungan ke definisi wadah

Dalam ketentuan kontainer Anda, Anda dapat menentukan hal berikut:

- Objek `secrets` yang berisi nama variabel lingkungan untuk diatur dalam kontainer
- Amazon Resource Name (ARN) rahasia Secrets Manager
- Parameter tambahan yang berisi data sensitif untuk hadir ke kontainer

Contoh berikut menunjukkan sintaks lengkap yang harus ditentukan untuk rahasia Secrets Manager.

```
arn:aws:secretsmanager:region:aws_account_id:secret:secret-name:json-key:version-stage:version-id
```

Bagian berikut menjelaskan parameter tambahan. Parameter ini opsional, tetapi jika Anda tidak menggunakannya, Anda harus menyertakan titik dua `:` untuk menggunakan nilai default. Contoh disediakan di bawah ini untuk konteks yang lebih.

`json-key`

Menentukan nama kunci dalam pasangan kunci-nilai dengan nilai yang ingin Anda tetapkan sebagai nilai variabel lingkungan. Hanya nilai dalam format JSON yang didukung. Jika Anda tidak menentukan kunci JSON, maka konten penuh dari rahasia tersebut akan digunakan.

`version-stage`

Menentukan label pementasan versi rahasia yang ingin Anda gunakan. Jika label pementasan versi ditentukan, Anda tidak dapat menentukan ID versi. Jika tidak ada tahap versi yang ditentukan, perilaku default adalah untuk mengambil rahasia dengan label pementasan `AWSCURRENT`.

Label tahapan digunakan untuk melacak berbagai versi rahasia saat diperbarui atau dirotasi. Setiap versi rahasia memiliki satu atau beberapa label tahapan dan satu ID. Untuk informasi selengkapnya, lihat Syarat dan Konsep Utama AWS Secrets Manager di Panduan AWS Secrets Manager Pengguna.

`version-id`

Menentukan pengenal unik dari versi rahasia yang ingin Anda gunakan. Jika ID versi ditentukan, Anda tidak dapat menentukan label pementasan versi. Jika tidak ada ID versi yang ditentukan, perilaku default adalah untuk mengambil rahasia dengan label pementasan `AWSCURRENT`.

ID versi digunakan untuk melacak berbagai versi rahasia saat diperbarui atau dirotasi. Setiap versi rahasia memiliki satu ID. Untuk informasi selengkapnya, lihat Syarat dan Konsep Utama AWS Secrets Manager di Panduan AWS Secrets Manager Pengguna.

Contoh ketentuan kontainer

Contoh berikut menunjukkan cara Anda dapat mereferensikan rahasia Secrets Manager dalam definisi container Anda.

Example mereferensikan rahasia penuh

Berikut ini adalah cuplikan ketentuan tugas yang menunjukkan format ketika mereferensikan teks lengkap dari rahasia Secrets Manager.

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-
AbCdEf"
    }]
  }]
}
```

Untuk mengakses nilai rahasia ini dari dalam wadah, Anda perlu memanggil file`$environment_variable_name`.

Example mereferensikan kunci tertentu dalam sebuah rahasia

Berikut ini menunjukkan contoh output dari get-secret-valueperintah yang menampilkan isi rahasia bersama dengan label pementasan versi dan ID versi yang terkait dengannya.

```
{
    "ARN": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-AbCdEf",
    "Name": "appauthexample",
    "VersionId": "871d9eca-18aa-46a9-8785-981ddEXAMPLE",
    "SecretString": "{\"username1\":\"password1\",\"username2\":\"password2\",
\"username3\":\"password3\"}",
    "VersionStages": [
        "AWSCURRENT"
    ],
    "CreatedDate": 1581968848.921
```

```
}
```

Referensikan kunci tertentu dari output sebelumnya dalam ketentuan kontainer dengan menentukan nama kunci di akhir ARN.

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf:username1::"
    }]
  }]
}
```

Example mereferensikan versi rahasia tertentu

Berikut ini menunjukkan contoh output dari perintah describe-secret yang menampilkan konten rahasia yang tidak terenkripsi bersama dengan metadata untuk semua versi rahasia.

```
{
    "ARN": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-AbCdEf",
    "Name": "appauthexample",
    "Description": "Example of a secret containing application authorization data.",
    "RotationEnabled": false,
    "LastChangedDate": 1581968848.926,
    "LastAccessedDate": 1581897600.0,
    "Tags": [],
    "VersionIdsToStages": {
        "871d9eca-18aa-46a9-8785-981ddEXAMPLE": [
            "AWSCURRENT"
        ],
        "9d4cb84b-ad69-40c0-a0ab-cead3EXAMPLE": [
            "AWSPREVIOUS"
        ]
    }
}
```

Referensikan label tahapan versi tertentu dari output sebelumnya dalam ketentuan kontainer dengan menentukan nama kunci di akhir ARN.

```
{
```

```
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf::AWSPREVIOUS:"
    }]
  }]
}
```

Referensikan ID versi tertentu dari output sebelumnya dalam ketentuan kontainer dengan menentukan nama kunci di akhir ARN.

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf:::9d4cb84b-ad69-40c0-a0ab-cead3EXAMPLE"
    }]
  }]
}
```

Example mereferensikan kunci dan label tahapan versi tertentu dari suatu rahasia

Bagian berikut menunjukkan cara mereferensikan kunci tertentu dalam rahasia dan label tahapan versi tertentu.

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf:username1:AWSPREVIOUS:"
    }]
  }]
}
```

Untuk menentukan kunci dan versi ID tertentu, gunakan sintaks berikut.

```
{
  "containerDefinitions": [{
    "secrets": [{
```

```
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf:username1::9d4cb84b-ad69-40c0-a0ab-cead3EXAMPLE"
    }]
  }]
}
```

Buat definisi tugas yang mereferensikan data sensitif

Anda dapat menggunakan konsol Amazon ECS untuk membuat definisi tugas yang mereferensikan rahasia Secrets Manager.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Penetapan tugas.
3. Pilih Buat penetapan tugas baru, Buat penetapan tugas baru.
4. Untuk Keluarga penetapan tugas, tentukan nama unik untuk penetapan tugas tersebut.
5. Untuk setiap kontainer untuk menentukan definisi tugas Anda, selesaikan langkah-langkah berikut.

   a. Untuk Nama, masukkan nama untuk wadah.

   b. Untuk URI Gambar, masukkan gambar yang akan digunakan untuk memulai wadah. Gambar dalam registri Galeri Publik Amazon ECR dapat ditentukan hanya menggunakan nama registri Publik Amazon ECR. Misalnya, jika `public.ecr.aws/ecs/amazon-ecs-agent:latest` ditentukan, wadah Amazon Linux yang dihosting di Galeri Publik Amazon ECR digunakan. Untuk semua repositori lainnya, tentukan repositori menggunakan format atau. `repository-url/image:tag` `repository-url/image@digest`

   c. Untuk wadah Essential, jika definisi tugas Anda memiliki dua atau lebih kontainer yang ditentukan, Anda dapat menentukan apakah penampung harus dianggap penting. Jika wadah ditandai sebagai penting, jika wadah itu berhenti maka tugas dihentikan. Setiap definisi tugas harus berisi setidaknya satu wadah penting.

   d. Pemetaan port memungkinkan kontainer untuk mengakses port pada host untuk mengirim atau menerima lalu lintas. Di bawah pemetaan Port, lakukan salah satu hal berikut:

      - Saat Anda menggunakan mode jaringan awsvpc, untuk port Container dan Protocol, pilih pemetaan port yang akan digunakan untuk wadah.
      - Saat Anda menggunakan mode jaringan jembatan, untuk port Container dan Protocol, pilih pemetaan port yang akan digunakan untuk wadah. Anda memilih mode jaringan

jembatan di halaman berikutnya. Setelah Anda memilihnya, pilih Sebelumnya, dan kemudian untuk port Host, tentukan nomor port pada instance kontainer yang akan dicadangkan untuk kontainer Anda.

Pilih Tambahkan lebih banyak pemetaan port untuk menentukan pemetaan port kontainer tambahan.

e. Untuk data sensitif untuk menyuntikkan sebagai variabel lingkungan, di bawah Lingkungan, untuk Variabel lingkungan, lengkapi kolom-kolom berikut:

   i. Untuk Kunci, masukkan nama variabel lingkungan untuk mengatur dalam kontainer. Hal ini sesuai dengan bidang `name` dalam bagian `secrets` dari ketentuan kontainer.

   ii. Untuk tipe Nilai, pilih ValueFrom. Untuk Tambah nilai, masukkan ARN rahasia Secrets Manager yang berisi data yang akan ditampilkan ke container Anda sebagai variabel lingkungan.

f. (Opsional) Pilih Tambahkan lebih banyak kontainer untuk menambahkan wadah tambahan ke definisi tugas. Pilih Berikutnya setelah semua kontainer telah ditentukan.

6. Untuk lingkungan Aplikasi, Ukuran tugas, dan ukuran Kontainer, isi kolom wajib yang tersisa dan kolom opsional apa pun.

7. (Opsional) Perluas peran tugas, mode jaringan bagian untuk menentukan yang berikut:

   • Untuk peran Tugas, pilih peran IAM yang akan ditetapkan ke tugas. Peran IAM tugas memberikan izin untuk kontainer dalam tugas untuk memanggil AWS API.

8. (Opsional) Bagian Penyimpanan digunakan untuk memperluas jumlah penyimpanan sementara untuk tugas yang dihosting di Fargate serta menambahkan konfigurasi volume data untuk tugas tersebut.

   • Untuk memperluas penyimpanan sementara yang tersedia di luar nilai default 20 GiB untuk tugas Fargate Anda, untuk Jumlah, masukkan nilai hingga. 200 GiB

9. Untuk data sensitif yang direferensikan dalam konfigurasi log untuk kontainer, di bawah Use log collection, untuk konfigurasi Log, selesaikan konfigurasi berikut:

   a. Pilih opsi log, dan kemudian di bawah Kunci, pilih Tambah.

   b. Untuk Kunci, masukkan nama opsi konfigurasi log untuk disetel.

   c. Untuk Nilai, masukkan ARN lengkap rahasia Secrets Manager yang berisi data untuk disajikan ke konfigurasi log Anda sebagai opsi log.

d. Untuk tipe Nilai, pilih ValueFrom.

10. Pilih Berikutnya untuk meninjau definisi tugas.
11. Pada halaman Tinjau dan buat, tinjau setiap bagian definisi tugas. Pilih Edit untuk membuat perubahan. Setelah definisi tugas selesai, pilih Buat untuk mendaftarkan definisi tugas.

Mengambil AWS Systems Manager parameter melalui variabel lingkungan

Amazon ECS memungkinkan Anda untuk menyuntikkan data sensitif ke dalam container Anda dengan menyimpan data sensitif Anda dalam AWS Systems Manager parameter Parameter Store dan kemudian mereferensikannya dalam definisi container Anda.

Pertimbangan

Berikut ini harus dipertimbangkan ketika menggunakan variabel lingkungan untuk menyuntikkan AWS Systems Manager rahasia ke dalam wadah.

- Data sensitif disuntikkan ke dalam kontainer Anda ketika kontainer awalnya dimulai. Jika rahasia kemudian diperbarui atau diputar, kontainer tidak akan menerima nilai yang diperbarui secara otomatis. Anda harus menjalankan tugas baru atau jika tugas Anda adalah bagian dari layanan Anda dapat memperbarui layanan dan menggunakan Memaksa deployment baru untuk memaksa layanan untuk meluncurkan tugas baru.
- Untuk tugas Amazon ECS aktif AWS Fargate, hal-hal berikut harus dipertimbangkan:
  - Untuk menyuntikkan konten penuh rahasia sebagai variabel lingkungan atau dalam konfigurasi log, Anda harus menggunakan versi `1.3.0` platform atau yang lebih baru. Untuk informasi, lihat Versi platform Fargate Linux.
  - Untuk menyuntikkan kunci JSON tertentu atau versi rahasia sebagai variabel lingkungan atau dalam konfigurasi log, Anda harus menggunakan versi platform `1.4.0` atau yang lebih baru (Linux) atau `1.0.0` (Windows). Untuk informasi, lihat Versi platform Fargate Linux.
- Untuk tugas Amazon ECS di EC2, hal-hal berikut harus dipertimbangkan:
  - Untuk menyuntikkan rahasia menggunakan kunci JSON atau versi rahasia tertentu, instans kontainer Anda harus memiliki versi `1.37.0` agen kontainer atau yang lebih baru. Namun, kami merekomendasikan untuk menggunakan versi agen kontainer terbaru. Untuk informasi tentang memeriksa versi agen Anda dan memperbarui ke versi terbaru, lihat Memperbarui agen kontainer Amazon ECS.

Untuk menyuntikkan konten penuh rahasia sebagai variabel lingkungan atau untuk menyuntikkan rahasia dalam konfigurasi log, instans kontainer Anda harus memiliki versi `1.22.0` agen kontainer atau yang lebih baru.

- Gunakan titik akhir VPC antarmuka untuk meningkatkan kontrol keamanan. Anda harus membuat titik akhir VPC antarmuka untuk. AWS Systems ManagerUntuk informasi tentang titik akhir VPC, lihat Membuat titik akhir VPC di Panduan Pengguna.AWS Systems Manager
- Untuk tugas Windows yang dikonfigurasi untuk menggunakan driver pencatatan `awslogs`, Anda juga harus mengatur variabel lingkungan `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE` pada instans kontainer Anda. Ini dapat dilakukan dengan Data Pengguna menggunakan sintaksis berikut:

```
<powershell>
[Environment]::SetEnvironmentVariable("ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE",
 $TRUE, "Machine")
Initialize-ECSAgent -Cluster <cluster name> -EnableTaskIAMRole -LoggingDrivers
 '["json-file","awslogs"]'
</powershell>
```

Izin IAM

Untuk menggunakan fitur ini, Anda harus memiliki peran eksekusi tugas Amazon ECS dan mereferensikannya dalam definisi tugas Anda. Hal ini memungkinkan agen kontainer untuk menarik AWS Systems Manager sumber daya yang diperlukan. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

Important

Untuk tugas yang menggunakan tipe peluncuran EC2, Anda harus menggunakan variabel konfigurasi agen ECS `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE=true` untuk menggunakan fitur ini. Anda dapat menambahkannya ke file `./etc/ecs/ecs.config` selama pembuatan instans kontainer atau Anda dapat menambahkannya ke instans yang ada, lalu memulai ulang agen ECS. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.

Untuk memberikan akses ke AWS Systems Manager parameter Parameter Store yang Anda buat, tambahkan izin berikut secara manual ke peran eksekusi tugas. Untuk informasi tentang cara mengelola izin, lihat Menambahkan dan Menghapus izin identitas IAM di Panduan Pengguna IAM.

- `ssm:GetParameters`— Diperlukan jika Anda mereferensikan parameter Systems Manager Parameter Store dalam definisi tugas. Menambahkan izin untuk mengambil parameter Systems Manager.
- `secretsmanager:GetSecretValue`— Diperlukan jika Anda mereferensikan rahasia Secrets Manager baik secara langsung atau jika parameter Parameter Store Systems Manager Anda mereferensikan rahasia Secrets Manager dalam definisi tugas. Menambahkan izin untuk mengambil rahasia dari Secrets Manager.
- `kms:Decrypt`— Diperlukan hanya jika rahasia Anda menggunakan kunci yang dikelola pelanggan dan bukan kunci default. ARN untuk kunci khusus Anda harus ditambahkan sebagai sumber daya. Menambahkan izin untuk mendekripsi kunci yang dikelola pelanggan.

Contoh kebijakan berikut menambahkan izin yang diperlukan:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetParameters",
        "secretsmanager:GetSecretValue",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:ssm:region:aws_account_id:parameter/parameter_name",
        "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name",
        "arn:aws:kms:region:aws_account_id:key/key_id"
      ]
    }
  ]
}
```

Buat AWS Systems Manager parameter

Anda dapat menggunakan konsol Systems Manager untuk membuat parameter Systems Manager Parameter Store untuk data sensitif Anda. Untuk informasi selengkapnya, lihat Membuat parameter

Systems Manager (konsol) atau Membuat parameter Systems Manager (AWS CLI) di Panduan AWS Systems Manager Pengguna.

Tambahkan variabel lingkungan ke definisi wadah

Dalam ketentuan kontainer Anda, tentukan `secrets` dengan nama variabel lingkungan yang akan ditetapkan dalam kontainer dan ARN lengkap parameter Systems Manager Parameter Store yang berisi data sensitif untuk diberikan ke kontainer. Untuk informasi selengkapnya, lihat `secrets`.

Berikut ini adalah cuplikan definisi tugas yang menunjukkan format saat mereferensikan parameter Systems Manager Parameter Store. Jika parameter Systems Manager Parameter Store ada di Region yang sama dengan tugas yang Anda luncurkan, maka Anda dapat menggunakan ARN lengkap atau nama parameter. Jika parameter ada di Wilayah yang berbeda, maka tentukan ARN lengkap.

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter/parameter_name"
    }]
  }]
}
```

Buat definisi tugas yang mereferensikan data sensitif

Anda dapat menggunakan konsol Amazon ECS untuk membuat definisi tugas yang mereferensikan parameter Systems Manager Parameter Store.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Penetapan tugas.
3. Pilih Buat penetapan tugas baru, Buat penetapan tugas baru.
4. Untuk Keluarga penetapan tugas, tentukan nama unik untuk penetapan tugas tersebut.
5. Untuk setiap kontainer untuk menentukan definisi tugas Anda, selesaikan langkah-langkah berikut.

    a. Untuk Nama, masukkan nama untuk wadah.

    b. Untuk URI Gambar, masukkan gambar yang akan digunakan untuk memulai wadah. Gambar dalam registri Galeri Publik Amazon ECR dapat ditentukan hanya menggunakan

nama registri Publik Amazon ECR. Misalnya, jika `public.ecr.aws/ecs/amazon-ecs-agent:latest` ditentukan, wadah Amazon Linux yang dihosting di Galeri Publik Amazon ECR digunakan. Untuk semua repositori lainnya, tentukan repositori menggunakan format atau. `repository-url/image:tag` `repository-url/image@digest`

c. Untuk wadah Essential, jika definisi tugas Anda memiliki dua atau lebih kontainer yang ditentukan, Anda dapat menentukan apakah penampung harus dianggap penting. Jika wadah ditandai sebagai penting, jika wadah itu berhenti maka tugas dihentikan. Setiap definisi tugas harus berisi setidaknya satu wadah penting.

d. Pemetaan port memungkinkan kontainer untuk mengakses port pada host untuk mengirim atau menerima lalu lintas. Di bawah pemetaan Port, lakukan salah satu hal berikut:

- Saat Anda menggunakan mode jaringan awsvpc, untuk port Container dan Protocol, pilih pemetaan port yang akan digunakan untuk wadah.
- Saat Anda menggunakan mode jaringan jembatan, untuk port Container dan Protocol, pilih pemetaan port yang akan digunakan untuk wadah. Anda memilih mode jaringan jembatan di halaman berikutnya. Setelah Anda memilihnya, pilih Sebelumnya, dan kemudian untuk port Host, tentukan nomor port pada instance kontainer yang akan dicadangkan untuk kontainer Anda.

Pilih Tambahkan lebih banyak pemetaan port untuk menentukan pemetaan port kontainer tambahan.

e. Untuk data sensitif untuk menyuntikkan sebagai variabel lingkungan, di bawah Lingkungan, untuk Variabel lingkungan, lengkapi kolom-kolom berikut:

i. Untuk Kunci, masukkan nama variabel lingkungan untuk mengatur dalam kontainer. Hal ini sesuai dengan bidang `name` dalam bagian `secrets` dari ketentuan kontainer.

ii. Untuk tipe Nilai, pilih ValueFrom. Untuk Nilai, masukkan nama atau ARN lengkap dari AWS Systems Manager parameter Parameter Store yang berisi data yang akan ditampilkan ke konfigurasi log Anda sebagai opsi log.

> **Note**
>
> Jika parameter Systems Manager Parameter Store ada di Region yang sama dengan tugas yang Anda luncurkan, maka Anda dapat menggunakan ARN

lengkap atau nama rahasianya. Jika parameter ada di Wilayah yang berbeda, maka tentukan ARN lengkap.

f. (Opsional) Pilih Tambahkan lebih banyak kontainer untuk menambahkan wadah tambahan ke definisi tugas. Pilih Berikutnya setelah semua kontainer telah ditentukan.

6. Untuk lingkungan Aplikasi, Ukuran tugas, dan ukuran Kontainer, isi kolom wajib yang tersisa dan kolom opsional apa pun.
7. (Opsional) Perluas peran tugas, mode jaringan bagian untuk menentukan yang berikut:

   - Untuk peran Tugas, pilih peran IAM yang akan ditetapkan ke tugas. Peran IAM tugas memberikan izin untuk kontainer dalam tugas untuk memanggil AWS API.

8. (Opsional) Bagian Penyimpanan digunakan untuk memperluas jumlah penyimpanan sementara untuk tugas yang dihosting di Fargate serta menambahkan konfigurasi volume data untuk tugas tersebut.

   - Untuk memperluas penyimpanan sementara yang tersedia di luar nilai default 20 GiB untuk tugas Fargate Anda, untuk Jumlah, masukkan nilai hingga. 200 GiB

9. Untuk data sensitif yang direferensikan dalam konfigurasi log untuk kontainer, di bawah Use log collection, untuk konfigurasi Log, selesaikan konfigurasi berikut:

   a. Pilih opsi log, dan kemudian di bawah Kunci, pilih Tambah.

   b. Untuk Kunci, masukkan nama opsi konfigurasi log untuk disetel.

   c. Untuk Nilai, masukkan nama atau ARN lengkap dari AWS Systems Manager parameter Parameter Store yang berisi data yang akan ditampilkan ke konfigurasi log Anda sebagai opsi log.

   > **Note**
   >
   > Jika parameter Systems Manager Parameter Store ada di Region yang sama dengan tugas yang Anda luncurkan, maka Anda dapat menggunakan ARN lengkap atau nama rahasianya. Jika parameter ada di Wilayah yang berbeda, maka tentukan ARN lengkap.

   d. Untuk tipe Nilai, pilih ValueFrom.

10. Pilih Berikutnya untuk meninjau definisi tugas.

11. Pada halaman Tinjau dan buat, tinjau setiap bagian definisi tugas. Pilih Edit untuk membuat perubahan. Setelah definisi tugas selesai, pilih Buat untuk mendaftarkan definisi tugas.

Ambil rahasia untuk konfigurasi logging

Menggunakan Secrets Manager

Dalam definisi kontainer Anda, saat menentukan a, `logConfiguration` Anda dapat menentukan `secretOptions` dengan nama opsi driver log untuk disetel dalam wadah dan ARN lengkap dari rahasia Secrets Manager yang berisi data sensitif untuk disajikan ke wadah.

Berikut ini adalah cuplikan definisi tugas yang menunjukkan format saat mereferensikan rahasia Secrets Manager.

```
{
  "containerDefinitions": [{
    "logConfiguration": [{
      "logDriver": "splunk",
      "options": {
        "splunk-url": "https://your_splunk_instance:8088"
      },
      "secretOptions": [{
        "name": "splunk-token",
        "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-
AbCdEf"
      }]
    }]
  }]
}
```

Menggunakan AWS Systems Manager

Anda dapat menyuntikkan data sensitif dalam konfigurasi log. Dalam ketentuan kontainer Anda, ketika menentukan `logConfiguration`, Anda dapat menentukan `secretOptions` dengan nama opsi driver log yang kan ditetapkan dalam kontainer dan ARN lengkap parameter Systems Manager Parameter Store yang berisi data sensitif untuk diberikan ke kontainer.

> **Important**
>
> Jika parameter Systems Manager Parameter Store ada di Region yang sama dengan tugas yang Anda luncurkan, maka Anda dapat menggunakan ARN lengkap atau nama parameter. Jika parameter ada di Wilayah yang berbeda, maka tentukan ARN lengkap.

Berikut ini adalah cuplikan definisi tugas yang menunjukkan format saat mereferensikan parameter Systems Manager Parameter Store.

```
{
  "containerDefinitions": [{
    "logConfiguration": [{
      "logDriver": "fluentd",
      "options": {
        "tag": "fluentd demo"
      },
      "secretOptions": [{
        "name": "fluentd-address",
        "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter:/parameter_name"
      }]
    }]
  }]
}
```

Contoh ketentuan tugas

Anda dapat menyalin contoh dan cuplikan untuk mulai membuat definisi tugas Anda sendiri.

Anda dapat menyalin contoh, lalu menempelkannya saat Anda menggunakan opsi Konfigurasi melalui JSON di konsol. Pastikan untuk menyesuaikan contoh, seperti menggunakan ID akun Anda. Anda dapat menyertakan cuplikan dalam definisi tugas JSON Anda. Lihat informasi yang lebih lengkap di Membuat definisi tugas menggunakan konsol dan Parameter ketentuan tugas.

Untuk contoh definisi tugas lainnya, lihat AWS Contoh Definisi Tugas pada GitHub.

Topik

- Server Web
- splunkdriver log

- fluentddriver log
- gelfdriver log
- Beban kerja pada instance eksternal
- Peran IAM definisi gambar dan tugas Amazon ECR
- Entrypoint dengan perintah
- Dependensi kontainer
- Ketentuan tugas sampel Windows

Server Web

Berikut ini adalah contoh definisi tugas menggunakan wadah Linux pada jenis peluncuran Fargate yang mengatur server web:

```
{
   "containerDefinitions": [
      {
         "command": [
            "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample App</
title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon
 ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html && httpd-
foreground\""
         ],
         "entryPoint": [
            "sh",
            "-c"
         ],
         "essential": true,
         "image": "httpd:2.4",
         "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
               "awslogs-group" : "/ecs/fargate-task-definition",
               "awslogs-region": "us-east-1",
               "awslogs-stream-prefix": "ecs"
            }
         },
         "name": "sample-fargate-app",
         "portMappings": [
```

```
            {
               "containerPort": 80,
               "hostPort": 80,
               "protocol": "tcp"
            }
         ]
      }
   ],
   "cpu": "256",
   "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
   "family": "fargate-task-definition",
   "memory": "512",
   "networkMode": "awsvpc",
   "runtimePlatform": {
        "operatingSystemFamily": "LINUX"
    },
   "requiresCompatibilities": [
       "FARGATE"
    ]
}
```

Berikut ini adalah contoh definisi tugas menggunakan wadah Windows pada jenis peluncuran Fargate yang mengatur server web:

```
{
    "containerDefinitions": [
        {
            "command": ["New-Item -Path C:\\inetpub\\wwwroot\\index.html -Type file
 -Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your
 application is now running on a container in Amazon ECS.</p>'; C:\\ServiceMonitor.exe
 w3svc"],
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "essential": true,
            "cpu": 2048,
            "memory": 4096,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "name": "sample_windows_app",
```

```
            "portMappings": [
                {
                    "hostPort": 80,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ]
        }
    ],
    "memory": "4096",
    "cpu": "2048",
    "networkMode": "awsvpc",
    "family": "windows-simple-iis-2019-core",
    "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
    "runtimePlatform": {"operatingSystemFamily": "WINDOWS_SERVER_2019_CORE"},
    "requiresCompatibilities": ["FARGATE"]
}
```

`splunk`driver log

Cuplikan berikut menunjukkan cara menggunakan driver `splunk` log dalam definisi tugas yang mengirimkan log ke layanan jarak jauh. Parameter token Splunk ditetapkan sebagai opsi rahasia karena dapat diperlakukan sebagai data sensitif. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah.

```
"containerDefinitions": [{
  "logConfiguration": {
   "logDriver": "splunk",
   "options": {
    "splunk-url": "https://cloud.splunk.com:8080",
    "tag": "tag_name",
   },
   "secretOptions": [{
    "name": "splunk-token",
    "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:splunk-token-
KnrBkD"
}],
```

`fluentd`driver log

Cuplikan berikut menunjukkan cara menggunakan driver `fluentd` log dalam definisi tugas yang mengirimkan log ke layanan jarak jauh. Nilai `fluentd-address` ditetapkan sebagai opsi rahasia

karena dapat diperlakukan sebagai data sensitif. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah.

```
"containerDefinitions": [{
 "logConfiguration": {
  "logDriver": "fluentd",
  "options": {
   "tag": "fluentd demo"
  },
  "secretOptions": [{
   "name": "fluentd-address",
   "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:fluentd-address-
KnrBkD"
  }]
 },
 "entryPoint": [],
 "portMappings": [{
             "hostPort": 80,
             "protocol": "tcp",
             "containerPort": 80
             },
             {
  "hostPort": 24224,
  "protocol": "tcp",
  "containerPort": 24224
 }]
}],
```

`gelf`driver log

Cuplikan berikut menunjukkan cara menggunakan driver `gelf` log dalam definisi tugas yang mengirimkan log ke host jarak jauh yang menjalankan Logstash yang mengambil log Gelf sebagai input. Untuk informasi selengkapnya, lihat `logConfiguration`.

```
"containerDefinitions": [{
 "logConfiguration": {
  "logDriver": "gelf",
  "options": {
   "gelf-address": "udp://logstash-service-address:5000",
   "tag": "gelf task demo"
  }
 },
```

```
 "entryPoint": [],
 "portMappings": [{
   "hostPort": 5000,
   "protocol": "udp",
   "containerPort": 5000
  },
  {
   "hostPort": 5000,
   "protocol": "tcp",
   "containerPort": 5000
  }
 ]
}],
```

Beban kerja pada instance eksternal

Saat mendaftarkan definisi tugas Amazon ECS, gunakan `requiresCompatibilities` parameter dan tentukan `EXTERNAL` yang memvalidasi bahwa definisi tugas kompatibel untuk digunakan saat menjalankan beban kerja Amazon ECS pada instans eksternal Anda. Jika Anda menggunakan konsol untuk mendaftarkan definisi tugas, Anda harus menggunakan editor JSON. Untuk informasi selengkapnya, lihat Membuat definisi tugas menggunakan konsol.

> **Important**
>
> Jika tugas Anda memerlukan peran IAM eksekusi tugas, pastikan itu ditentukan dalam definisi tugas.

Ketika Anda men-deploy beban kerja, gunakan tipe peluncuran `EXTERNAL` ketika membuat layanan atau menjalankan tugas mandiri Anda.

Berikut ini merupakan ketentuan tugas contoh.

Linux

```
{
 "requiresCompatibilities": [
  "EXTERNAL"
 ],
 "containerDefinitions": [{
  "name": "nginx",
```

```
  "image": "public.ecr.aws/nginx/nginx:latest",
  "memory": 256,
  "cpu": 256,
  "essential": true,
  "portMappings": [{
   "containerPort": 80,
   "hostPort": 8080,
   "protocol": "tcp"
  }]
 }],
 "networkMode": "bridge",
 "family": "nginx"
}
```

Windows

```
{
 "requiresCompatibilities": [
  "EXTERNAL"
 ],
 "containerDefinitions": [{
  "name": "windows-container",
  "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-ltsc2019",
  "memory": 256,
  "cpu": 512,
  "essential": true,
  "portMappings": [{
   "containerPort": 80,
   "hostPort": 8080,
   "protocol": "tcp"
  }]
 }],
 "networkMode": "bridge",
 "family": "windows-container"
}
```

Peran IAM definisi gambar dan tugas Amazon ECR

Cuplikan berikut menggunakan gambar Amazon ECR yang disebut `aws-nodejs-sample` dengan `v1` tag dari registri. `123456789012.dkr.ecr.us-west-2.amazonaws.com` Wadah

dalam tugas ini mewarisi izin IAM dari peran. `arn:aws:iam::123456789012:role/AmazonECSTaskS3BucketRole` Untuk informasi selengkapnya, lihat Tugas peran IAM.

```
{
    "containerDefinitions": [
        {
            "name": "sample-app",
            "image": "123456789012.dkr.ecr.us-west-2.amazonaws.com/aws-nodejs-
sample:v1",
            "memory": 200,
            "cpu": 10,
            "essential": true
        }
    ],
    "family": "example_task_3",
    "taskRoleArn": "arn:aws:iam::123456789012:role/AmazonECSTaskS3BucketRole"
}
```

Entrypoint dengan perintah

Cuplikan berikut menunjukkan sintaks untuk wadah Docker yang menggunakan titik masuk dan argumen perintah. Kontainer ini menge-ping `google.com` empat kali dan kemudian keluar.

```
{
    "containerDefinitions": [
        {
            "memory": 32,
            "essential": true,
            "entryPoint": ["ping"],
            "name": "alpine_ping",
            "readonlyRootFilesystem": true,
            "image": "alpine:3.4",
            "command": [
                "-c",
                "4",
                "example.com"
            ],
            "cpu": 16
        }
    ],
    "family": "example_task_2"
}
```

Dependensi kontainer

Cuplikan ini menunjukkan sintaks untuk definisi tugas dengan beberapa kontainer di mana dependensi kontainer ditentukan. Dalam definisi tugas berikut, `envoy` wadah harus mencapai status sehat, ditentukan oleh parameter pemeriksaan kesehatan wadah yang diperlukan, sebelum `app` wadah dimulai. Untuk informasi selengkapnya, lihat Dependensi kontainer.

```
{
  "family": "appmesh-gateway",
  "runtimePlatform": {
        "operatingSystemFamily": "LINUX"
  },
  "proxyConfiguration":{
      "type": "APPMESH",
      "containerName": "envoy",
      "properties": [
          {
              "name": "IgnoredUID",
              "value": "1337"
          },
          {
              "name": "ProxyIngressPort",
              "value": "15000"
          },
          {
              "name": "ProxyEgressPort",
              "value": "15001"
          },
          {
              "name": "AppPorts",
              "value": "9080"
          },
          {
              "name": "EgressIgnoredIPs",
              "value": "169.254.170.2,169.254.169.254"
          }
      ]
  },
  "containerDefinitions": [
    {
      "name": "app",
      "image": "application_image",
      "portMappings": [
```

```
        {
          "containerPort": 9080,
          "hostPort": 9080,
          "protocol": "tcp"
        }
      ],
      "essential": true,
      "dependsOn": [
        {
          "containerName": "envoy",
          "condition": "HEALTHY"
        }
      ]
    },
    {
      "name": "envoy",
      "image": "840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-
envoy:v1.15.1.0-prod",
      "essential": true,
      "environment": [
        {
          "name": "APPMESH_VIRTUAL_NODE_NAME",
          "value": "mesh/meshName/virtualNode/virtualNodeName"
        },
        {
          "name": "ENVOY_LOG_LEVEL",
          "value": "info"
        }
      ],
      "healthCheck": {
        "command": [
          "CMD-SHELL",
          "echo hello"
        ],
        "interval": 5,
        "timeout": 2,
        "retries": 3
      }
    }
  ],
  "executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
  "networkMode": "awsvpc"
}
```

Ketentuan tugas sampel Windows

Berikut ini adalah contoh definisi tugas untuk membantu Anda memulai dengan wadah Windows di Amazon ECS.

Example Aplikasi Contoh Konsol Amazon ECS untuk Windows

Definisi tugas berikut adalah aplikasi sampel konsol Amazon ECS yang diproduksi di wizard yang dijalankan pertama untuk Amazon ECS; itu telah di-porting untuk menggunakan gambar wadah Windows. `microsoft/iis`

```
{
  "family": "windows-simple-iis",
  "containerDefinitions": [
    {
      "name": "windows_sample_app",
      "image": "mcr.microsoft.com/windows/servercore/iis",
      "cpu": 1024,
      "entryPoint":["powershell", "-Command"],
      "command":["New-Item -Path C:\\inetpub\\wwwroot\\index.html -Type file -
Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your
 application is now running on a container in Amazon ECS.</p>'; C:\\ServiceMonitor.exe
 w3svc"],
      "portMappings": [
        {
          "protocol": "tcp",
          "containerPort": 80
        }
      ],
      "memory": 1024,
      "essential": true
    }
  ],
  "networkMode": "awsvpc",
  "memory": "1024",
  "cpu": "1024"
}
```

Cluster dan kapasitas Amazon ECS

Cluster Amazon ECS adalah pengelompokan tugas atau layanan yang logis. Selain tugas dan layanan, cluster terdiri dari sumber daya berikut:

- Kapasitas infrastruktur yang dapat menjadi kombinasi dari berikut ini:
  - Instans Amazon EC2 di cloud AWS
  - Tanpa server (AWS Fargate (Fargate)) di awan AWS
  - Mesin virtual lokal (VM) atau server
- Jaringan (VPC dan subnet) tempat tugas dan layanan Anda berjalan

  Saat Anda menggunakan instans Amazon EC2 untuk kapasitas, subnet dapat berada di Availability Zone, Local Zones, Wavelength Zones, atau. AWS Outposts
- Namespace opsional

  Namespace digunakan untuk service-to-service komunikasi dengan Service Connect.
- Opsi pemantauan

  CloudWatch Wawasan Kontainer datang dengan biaya tambahan dan merupakan layanan yang dikelola sepenuhnya. Secara otomatis mengumpulkan, mengumpulkan, dan merangkum metrik dan log Amazon ECS.

Konsep penyedia kapasitas

Penyedia kapasitas terdiri dari komponen-komponen berikut.

Penyedia kapasitas

Penyedia kapasitas menentukan kapasitas klaster yang meningkatkan dan menurunkan skala Amazon ECS dari infrastruktur yang Anda tentukan. Anda harus terlebih dahulu mengaitkan penyedia kapasitas dengan cluster sebelum Anda menggunakan penyedia kapasitas.

Anda menggunakan penyedia kapasitas dalam strategi penyedia kapasitas untuk menentukan infrastruktur tempat tugas dijalankan. Setiap tugas harus memiliki strategi penyedia kapasitas, jenis peluncuran, atau menggunakan strategi penyedia kapasitas default yang terkait dengan cluster yang dipilih. Anda harus mereferensikan strategi penyedia kapasitas dan bukan penyedia

kapasitas. Jika tugas menggunakan tipe peluncuran, kapasitas yang digunakannya tidak dihitung oleh penyedia kapasitas apa pun di cluster.

Karena AWS Fargate, penyedia kapasitas adalah `FARGATE` dan penyedia `FARGATE_SPOT` kapasitas yang AWS menciptakan. Anda mengaitkan penyedia kapasitas dengan klaster Anda, dan kemudian menambahkannya ke strategi penyedia kapasitas.

Untuk Amazon ECS pada pengguna Amazon EC2, penyedia kapasitas terdiri dari nama penyedia kapasitas, grup Auto Scaling. Penyedia kapasitas juga terdiri dari semua pengaturan untuk penskalaan terkelola dan perlindungan terminasi terkelola. Saat Anda mengaktifkan penskalaan terkelola, Amazon ECS menskalakan grup Auto Scaling masuk dan keluar atas nama Anda.

Strategi penyedia kapasitas default

Anda dapat mengaitkan strategi penyedia kapasitas default dengan kluster Amazon ECS. Setelah Anda melakukannya, Amazon ECS menggunakan strategi penyedia kapasitas default saat Anda membuat layanan atau menjalankan tugas mandiri di klaster dan tidak menentukan jenis peluncuran atau penyedia kapasitas khusus. Kami menyarankan Anda menentukan strategi penyedia kapasitas default untuk setiap cluster.

Strategi penyedia kapasitas

Sebuah strategi penyedia kapasitas terdiri dari satu penyedia kapasitas atau lebih. Anda dapat menentukan basis opsional dan nilai berat untuk kontrol yang lebih baik. Strategi penyedia kapasitas adalah bagian dari konfigurasi cluster, layanan, atau tugas. Namun, Anda tidak dapat membuat strategi penyedia kapasitas yang dapat digunakan kembali. Strategi penyedia kapasitas dari setiap cluster, layanan, atau strategi penyedia kapasitas tugas bersifat independen.

Jika strategi penyedia kapasitas default untuk klaster tidak memenuhi persyaratan kapasitas Anda, tentukan strategi penyedia kapasitas khusus saat membuat layanan atau menjalankan tugas mandiri.

> **Important**
>
> Saat Anda menetapkan jenis peluncuran alih-alih strategi penyedia kapasitas pada tugas dalam klaster tempat kapasitas dikelola oleh penyedia kapasitas, tugas tersebut tidak dihitung untuk tindakan penskalaan penyedia kapasitas.

Hanya penyedia kapasitas yang keduanya sudah terkait dengan cluster dan memiliki `UPDATING` status `ACTIVE` atau yang dapat digunakan dalam strategi penyedia kapasitas. Anda dapat mengaitkan penyedia kapasitas dengan klaster saat membuat klaster.

Dalam strategi penyedia kapasitas, nilai dasar opsional menunjukkan berapa banyak tugas, minimal, yang dijalankan pada penyedia kapasitas tertentu. Hanya satu penyedia kapasitas di strategi penyedia kapasitas yang dapat menentukan nilai dasar.

Nilai bobot menentukan persentase relatif dari jumlah total tugas yang diluncurkan yang menggunakan penyedia kapasitas yang ditentukan. Pertimbangkan contoh berikut. Anda memiliki strategi yang berisi dua penyedia kapasitas, dan keduanya memiliki bobot1. Ketika persentase dasar tercapai, tugas dibagi secara merata di dua penyedia kapasitas. Menggunakan logika yang sama, misalkan Anda menentukan bobot untuk CapacityProvidera dan bobot 1 untuk CapacityProviderB. *4* Kemudian, untuk setiap tugas yang dijalankan menggunakan CapacityProvidera, ada empat tugas yang menggunakan CapacityProviderB.

Penyedia kapasitas Amazon ECS

Penyedia kapasitas Amazon ECS mengelola penskalaan infrastruktur untuk tugas di cluster Anda. Setiap cluster dapat memiliki satu atau lebih penyedia kapasitas dan strategi penyedia kapasitas opsional. Strategi penyedia kapasitas menentukan bagaimana tugas-tugas tersebar di penyedia kapasitas klaster ini. Saat menjalankan tugas mandiri atau membuat layanan, Anda dapat menggunakan strategi penyedia kapasitas default kluster atau strategi penyedia kapasitas yang mengganti tugas default.

Penyedia kapasitas tersedia untuk tugas yang berjalan di Fargate atau di instans Amazon EC2. Anda tidak dapat menggunakan penyedia kapasitas untuk tugas yang berjalan pada instans penampung eksternal (Amazon ECS Anywhere).

Jenis penyedia kapasitas

Untuk beban kerja Amazon ECS yang di-host di Fargate, penyedia kapasitas yang telah ditentukan berikut tersedia:

- Fargate
- Spot Fargate

Untuk beban kerja Amazon ECS yang di-host di instans Amazon EC2, Anda harus membuat dan memelihara penyedia kapasitas yang terdiri dari komponen-komponen berikut:

- Sebuah nama
- Grup Auto Scaling
- Pengaturan untuk penskalaan terkelola dan perlindungan terminasi terkelola.

Anda dapat membuat grup Auto Scaling saat membuat klaster, atau Anda dapat membuat grup sebelum cluster, lalu menentukan nama grup saat Anda membuat klaster.

Pertimbangan penyedia kapasitas

Pertimbangkan hal berikut saat menggunakan penyedia kapasitas:

- Penyedia kapasitas harus dikaitkan dengan cluster sebelum ditentukan dalam strategi penyedia kapasitas.
- Saat Anda menentukan strategi penyedia kapasitas, jumlah penyedia kapasitas yang dapat Anda tentukan dibatasi hingga 20.
- Anda tidak dapat memperbarui layanan menggunakan penyedia kapasitas grup Auto Scaling untuk menggunakan penyedia kapasitas Fargate. Kebalikannya juga terjadi.
- Dalam strategi penyedia kapasitas, jika tidak ada `weight` nilai yang ditentukan untuk penyedia kapasitas di konsol, maka nilai default 1 digunakan. Jika menggunakan API atau AWS CLI, nilai default `0` digunakan.
- Ketika beberapa penyedia kapasitas ditentukan dalam strategi penyedia kapasitas, setidaknya salah satu penyedia kapasitas harus memiliki nilai bobot yang lebih besar dari nol. Selain itu, penyedia kapasitas apa pun dengan bobot nol tidak digunakan untuk menempatkan tugas. Jika Anda menentukan beberapa penyedia kapasitas dalam strategi dengan bobot nol yang sama, maka `CreateService` tindakan apa pun `RunTask` atau yang menggunakan strategi penyedia kapasitas gagal.
- Dalam strategi penyedia kapasitas, hanya satu penyedia kapasitas yang dapat memiliki nilai dasar yang ditentukan. Jika tidak ada nilai dasar yang ditentukan, nilai default nol digunakan.
- Sebuah cluster dapat berisi campuran penyedia kapasitas grup Auto Scaling dan penyedia kapasitas Fargate. Namun, strategi penyedia kapasitas hanya dapat berisi grup Auto Scaling atau penyedia kapasitas Fargate, tetapi tidak keduanya.

- Cluster dapat berisi campuran layanan dan tugas mandiri yang menggunakan penyedia kapasitas dan jenis peluncuran. Layanan dapat diperbarui untuk menggunakan strategi penyedia kapasitas daripada jenis peluncuran. Namun, Anda harus memaksa penerapan baru saat melakukannya.
- Saat Anda menggunakan proteksi terminasi terkelola, Anda juga harus menggunakan penskalaan terkelola. Jika tidak, perlindungan terminasi terkelola tidak berfungsi.

Topik

AWS Fargate penyedia kapasitas

Dengan Amazon ECS pada penyedia AWS Fargate kapasitas, Anda dapat menggunakan kapasitas Fargate dan Fargate Spot dengan tugas Amazon ECS Anda.

Dengan Fargate Spot, Anda dapat menjalankan tugas Amazon ECS toleran interupsi dengan tarif yang didiskon dibandingkan dengan harga Fargate. Spot Fargate menjalankan tugas pada kapasitas komputasi cadangan. Ketika AWS membutuhkan kapasitas kembali, tugas Anda terganggu dengan peringatan dua menit.

Pertimbangan penyedia kapasitas Fargate

Pertimbangkan hal berikut saat menggunakan penyedia kapasitas Fargate:

- Kontainer Windows di Fargate tidak mendukung penyedia kapasitas Fargate Spot.
- Tugas Linux dengan arsitektur ARM64 tidak mendukung penyedia kapasitas Fargate Spot. Fargate Spot hanya mendukung tugas-tugas Linux dengan arsitektur X86_64.
- Anda tidak perlu membuat penyedia kapasitas Fargate dan Fargate Spot. Mereka tersedia untuk semua akun. Untuk menggunakannya, yang perlu Anda lakukan adalah mengaitkannya dengan cluster.
- Untuk mengaitkan penyedia kapasitas Fargate dan Fargate Spot ke klaster, Anda harus menggunakan Amazon ECS API atau. AWS CLI Anda tidak dapat mengaitkannya menggunakan konsol.
- Penyedia kapasitas Fargate dan Fargate Spot dicadangkan dan tidak dapat dihapus. Namun, Anda dapat memisahkannya dari cluster menggunakan operasi `PutClusterCapacityProviders` API.

- Anda dapat mengaitkan penyedia kapasitas dengan klaster yang ada menggunakan operasi `PutClusterCapacityProviders` API.
- Jika Anda menggunakan Fargate Spot, tugas Anda harus menggunakan platform versi 1.3.0 atau yang lebih baru (untuk Linux). Untuk informasi selengkapnya, lihat Versi platform Fargate Linux.
- Saat tugas yang menggunakan penyedia kapasitas Fargate dan Fargate Spot dihentikan, peristiwa perubahan status tugas dikirim ke Amazon. EventBridge Alasan berhenti menggambarkan penyebabnya. Untuk informasi selengkapnya, lihat Acara perubahan status tugas Amazon ECS.
- Sebuah cluster dapat berisi campuran penyedia kapasitas grup Fargate dan Auto Scaling. Namun, strategi penyedia kapasitas hanya dapat berisi penyedia kapasitas grup Fargate atau Auto Scaling, tetapi tidak keduanya. Untuk informasi selengkapnya, lihat Penyedia Kapasitas Grup Auto Scaling di Panduan Pengembang Layanan Kontainer Elastis Amazon.

Menangani pemberitahuan penghentian Fargate Spot

Pahami bahwa konsekuensi berikut karena kapasitas Spot mungkin tidak tersedia sepanjang waktu.

- Selama periode permintaan yang sangat tinggi, kapasitas Fargate Spot mungkin tidak tersedia. Hal ini dapat menyebabkan tugas Fargate Spot tertunda. Dalam acara ini, layanan Amazon ECS mencoba lagi meluncurkan tugas hingga kapasitas yang diperlukan tersedia. Fargate tidak menggantikan kapasitas Spot dengan kapasitas on-demand.
- Ketika tugas yang menggunakan kapasitas Fargate Spot dihentikan karena gangguan Spot, peringatan dua menit dikirim sebelum tugas dihentikan. Peringatan dikirim sebagai peristiwa perubahan status tugas ke Amazon EventBridge dan sebagai sinyal SIGTERM ke tugas yang sedang berjalan. Jika Anda menggunakan Fargate Spot sebagai bagian dari layanan, maka dalam skenario ini penjadwal layanan menerima sinyal interupsi dan mencoba meluncurkan tugas tambahan di Fargate Spot jika ada kapasitas yang tersedia. Layanan dengan hanya satu tugas terganggu sampai kapasitas tersedia. Untuk informasi lebih lanjut tentang shutdown yang anggun, lihat Penutupan anggun dengan ECS.

Untuk memastikan bahwa kontainer Anda keluar dengan anggun sebelum tugas berhenti, Anda dapat mengonfigurasi hal berikut:

- Nilai `stopTimeout` selama `120` detik atau kurang dapat ditentukan dalam ketentuan kontainer yang digunakan oleh tugas. `stopTimeout`Nilai default adalah 30 detik. Anda dapat menentukan `stopTimeout` nilai yang lebih panjang untuk memberi diri Anda lebih banyak waktu antara saat

peristiwa perubahan status tugas diterima dan titik waktu ketika penampung dihentikan secara paksa. Untuk informasi selengkapnya, lihat Waktu habis kontainer.

- Sinyal SIGTERM harus diterima dari dalam kontainer untuk melakukan tindakan pembersihan. Kegagalan untuk memproses sinyal ini mengakibatkan tugas menerima sinyal SIGKILL setelah dikonfigurasi `stopTimeout` dan dapat mengakibatkan kehilangan data atau korupsi.

Berikut ini adalah cuplikan dari peristiwa perubahan status tugas. Cuplikan ini menampilkan alasan berhenti dan kode berhenti untuk gangguan Fargate Spot.

```
{
  "version": "0",
  "id": "9bcdac79-b31f-4d3d-9410-fbd727c29fab",
  "detail-type": "ECS Task State Change",
  "source": "aws.ecs",
  "account": "111122223333",
  "resources": [
    "arn:aws:ecs:us-east-1:111122223333:task/b99d40b3-5176-4f71-9a52-9dbd6f1cebef"
  ],
  "detail": {
    "clusterArn": "arn:aws:ecs:us-east-1:111122223333:cluster/default",
    "createdAt": "2016-12-06T16:41:05.702Z",
    "desiredStatus": "STOPPED",
    "lastStatus": "RUNNING",
    "stoppedReason": "Your Spot Task was interrupted.",
    "stopCode": "SpotInterruption",
    "taskArn": "arn:aws:ecs:us-east-1:111122223333:task/
b99d40b3-5176-4f71-9a52-9dbd6fEXAMPLE",
    ...
  }
}
```

Berikut ini adalah pola peristiwa yang digunakan untuk membuat EventBridge aturan untuk peristiwa perubahan status tugas Amazon ECS. Anda dapat secara opsional menentukan cluster di `detail` lapangan. Melakukannya berarti Anda akan menerima peristiwa perubahan status tugas untuk cluster itu. Untuk informasi selengkapnya, lihat Membuat EventBridge Aturan di Panduan EventBridge Pengguna Amazon.

```
{
    "source": [
        "aws.ecs"
```

```
    ],
    "detail-type": [
        "ECS Task State Change"
    ],
    "detail": {
        "clusterArn": [
            "arn:aws:ecs:us-west-2:111122223333:cluster/default"
        ]
    }
}
```

Penyedia kapasitas grup Auto Scaling Amazon EC2

Saat menggunakan instans Amazon EC2 untuk kapasitas Anda, Anda menggunakan grup Auto Scaling untuk mengelola instans Amazon EC2 yang terdaftar di klaster mereka. Auto Scaling membantu Anda memastikan bahwa Anda memiliki jumlah instans Amazon EC2 yang benar yang tersedia untuk menangani pemuatan aplikasi Anda.

Anda dapat menggunakan fitur penskalaan terkelola agar Amazon ECS mengelola tindakan penskalaan dan penskalaan grup Auto Scaling (penskalaan terkelola) atau Anda dapat mengelola tindakan penskalaan sendiri. Untuk informasi selengkapnya, lihat Penskalaan otomatis Klaster Amazon ECS.

Pertimbangan penyedia kapasitas grup Auto Scaling

Pertimbangkan hal berikut saat menggunakan penyedia kapasitas grup Auto Scaling di konsol:

- Kami menyarankan Anda membuat grup Auto Scaling kosong baru untuk digunakan dengan penyedia kapasitas daripada menggunakan yang sudah ada. Jika Anda menggunakan grup Auto Scaling yang sudah ada, instans Amazon EC2 apa pun yang terkait dengan grup yang sudah berjalan dan terdaftar ke klaster Amazon ECS sebelum grup Auto Scaling digunakan untuk membuat penyedia kapasitas mungkin tidak terdaftar dengan benar pada penyedia kapasitas. Ini dapat menyebabkan masalah saat menggunakan penyedia kapasitas dalam strategi penyedia kapasitas. Gunakan `DescribeContainerInstances` untuk mengonfirmasi apakah instance kontainer dikaitkan dengan penyedia kapasitas atau tidak.

> **Note**
>
> Untuk membuat grup Auto Scaling kosong, atur hitungan yang diinginkan ke nol. Setelah Anda membuat penyedia kapasitas dan menghubungkannya dengan cluster, Anda kemudian dapat menskalakannya.
> Saat Anda menggunakan konsol Amazon ECS Create Cluster dengan opsi instans Amazon EC2 di bawah Infrastruktur, Amazon ECS membuat konfigurasi peluncuran Auto Scaling Amazon EC2 dan grup Auto Scaling atas nama Anda sebagai bagian dari tumpukan. AWS CloudFormation Mereka diawali dengan`EC2ContainerService-<ClusterName>`, yang membuatnya mudah diidentifikasi. Anda dapat menggunakan grup Auto Scaling sebagai penyedia kapasitas untuk cluster tersebut.

- Grup Auto Scaling harus memiliki `MaxSize` lebih dari nol untuk diskalakan.
- Grup Auto Scaling tidak dapat memiliki pengaturan pembobotan instance. Pembobotan instans tidak didukung saat digunakan dengan penyedia kapasitas Amazon ECS.
- Jika grup Auto Scaling tidak dapat menskalakan untuk mengakomodasi jumlah tugas yang dijalankan, tugas gagal bertransisi di luar status. `PROVISIONING`
- Jangan mengubah sumber daya kebijakan penskalaan yang terkait dengan grup Auto Scaling yang dikelola oleh penyedia kapasitas.
- Saat Anda menggunakan proteksi terminasi terkelola, Anda juga harus menggunakan penskalaan terkelola. Jika tidak, perlindungan terminasi terkelola tidak akan berfungsi.
- Saat penskalaan terkelola diaktifkan, penyedia kapasitas grup Auto Scaling akan membuat sumber daya kebijakan penskalaan untuk mengelola penskalaan grup Auto Scaling Anda. Anda dapat mengidentifikasi sumber daya ini dengan `ECSManaged` awalan.

  Saat Anda menggunakan perlindungan penghentian terkelola, Amazon ECS hanya menghentikan instans EC2 yang tidak menjalankan tugas Amazon ECS.
- Jika perlindungan penghentian terkelola diaktifkan saat Anda membuat penyedia kapasitas, grup Auto Scaling dan setiap instans Amazon EC2 dalam grup Auto Scaling harus memiliki perlindungan instans dari skala yang diaktifkan. Untuk informasi selengkapnya, lihat Perlindungan Instance di Panduan AWS Auto Scaling Pengguna.
- Anda dapat menambahkan kolam hangat ke grup Auto Scaling Anda. Kolam hangat adalah sekelompok instans Amazon EC2 yang telah diinisialisasi sebelumnya yang siap disertakan dalam klaster kapan pun aplikasi Anda perlu diskalakan. Untuk informasi lebih lanjut tentang kolam hangat, lihatMenggunakan kolam hangat untuk grup Auto Scaling Anda.

- Jika penskalaan terkelola diaktifkan saat Anda membuat penyedia kapasitas, jumlah yang diinginkan grup Auto Scaling dapat diatur. `0` Saat penskalaan terkelola diaktifkan, Amazon ECS mengelola tindakan penskalaan dan penskalaan grup Auto Scaling.
- Pengurasan instans terkelola diaktifkan secara default saat Anda membuat penyedia kapasitas. Kami menyarankan Anda menggunakan fitur ini untuk mengaktifkan penghentian instans Amazon EC2 yang anggun tanpa mengganggu beban kerja Anda.

Untuk informasi selengkapnya tentang membuat template peluncuran Auto Scaling Amazon EC2, lihat Meluncurkan Template di Panduan Pengguna Auto Scaling Amazon EC2. Untuk informasi selengkapnya tentang membuat grup Auto Scaling Amazon EC2, lihat grup Auto Scaling di Panduan Pengguna Auto Scaling Amazon EC2.

Menggunakan kolam hangat untuk grup Auto Scaling Anda

Amazon ECS mendukung kolam hangat Amazon EC2 Auto Scaling. Kolam hangat adalah sekelompok instans Amazon EC2 pra-inisialisasi yang siap digunakan. Kapan pun aplikasi Anda perlu diskalakan, Auto Scaling Amazon EC2 menggunakan instance pra-inisialisasi dari kolam hangat daripada meluncurkan instance dingin, memungkinkan proses inisialisasi akhir apa pun berjalan, dan kemudian menempatkan instance ke dalam layanan.

Untuk mempelajari selengkapnya tentang kolam hangat dan cara menambahkan kolam hangat ke grup Auto Scaling Anda, lihat Kolam hangat untuk Penskalaan Otomatis Amazon EC2 di Panduan Pengguna Auto Scaling Amazon EC2.

Saat membuat atau memperbarui kolam hangat untuk grup Auto Scaling untuk Amazon ECS, Anda tidak dapat menyetel opsi yang mengembalikan instance ke kolam hangat pada skala di (). `ReuseOnScaleIn` Untuk informasi selengkapnya, lihat put-warm-pool di dalam Referensi AWS Command Line Interface .

Untuk menggunakan kolam hangat dengan kluster Amazon ECS Anda, setel variabel konfigurasi `ECS_WARM_POOLS_CHECK` agen ke `true` bidang data Pengguna templat peluncuran grup Auto Scaling Amazon EC2 Anda. Berikut ini menunjukkan contoh bagaimana variabel konfigurasi agen dapat ditentukan di bidang data pengguna dari template peluncuran Amazon EC2.

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_WARM_POOLS_CHECK=true
EOF
```

`ECS_WARM_POOLS_CHECK`Variabel hanya didukung pada versi agen `1.59.0` dan yang lebih baru. Untuk informasi lebih lanjut tentang variabel, lihatKonfigurasi agen kontainer Amazon ECS.

Penskalaan otomatis Klaster Amazon ECS

Amazon ECS dapat mengelola penskalaan instans Amazon EC2 yang terdaftar di klaster Anda. Ini disebut sebagai penskalaan otomatis cluster Amazon ECS. Ini dilakukan dengan menggunakan penyedia kapasitas grup Auto Scaling Amazon ECS dengan penskalaan terkelola diaktifkan. Saat Anda menggunakan penyedia kapasitas grup Auto Scaling dengan penskalaan terkelola diaktifkan, Anda menetapkan persentase target (the`targetCapacity`) untuk pemanfaatan instance dalam grup Auto Scaling ini. Amazon ECS membuat dua CloudWatch metrik khusus dan kebijakan penskalaan pelacakan target yang dilampirkan ke grup Auto Scaling Anda. Amazon ECS kemudian mengelola tindakan penskalaan dan penskalaan grup Auto Scaling berdasarkan pemanfaatan sumber daya yang digunakan tugas Anda dari penyedia kapasitas ini. Untuk informasi selengkapnya tentang penyedia kapasitas grup Auto Scaling, lihat. the section called “Penyedia kapasitas grup Auto Scaling Amazon EC2”

> **Note**
>
> Penskalaan otomatis cluster Amazon ECS hanya didukung oleh penyedia kapasitas grup Auto Scaling. Untuk beban kerja Amazon ECS yang di-host AWS Fargate, lihat. the section called “AWS Fargate penyedia kapasitas”

Berikut ini adalah alur kerja Anda untuk menggunakan penskalaan otomatis klaster Amazon ECS. Untuk informasi selengkapnya, lihat the section called “Aktifkan penskalaan otomatis cluster”.

1. Buat grup Auto Scaling.
2. Buat penyedia kapasitas yang menggunakan grup Auto Scaling tersebut.
3. Aktifkan penskalaan terkelola untuk penyedia kapasitas.
4. Kaitkan penyedia kapasitas dengan cluster.
5. Jalankan tugas atau buat layanan dengan strategi penyedia kapasitas yang menggunakan penyedia kapasitas.

   Strategi penyedia kapasitas menentukan bagaimana tugas-tugas tersebar di penyedia kapasitas klaster ini. Saat menjalankan tugas mandiri atau membuat layanan, Anda dapat menggunakan

strategi penyedia kapasitas default kluster atau strategi penyedia kapasitas yang mengganti tugas default.

6. (Opsional) Buat strategi penyedia kapasitas default untuk cluster.

Per 27 Mei 2022, Amazon ECS tidak lagi membuat rencana AWS Auto Scaling penskalaan untuk penyedia kapasitas yang baru dibuat. Sebagai gantinya, Amazon ECS menggunakan kebijakan penskalaan pelacakan target yang dilampirkan pada grup Auto Scaling untuk melakukan penskalaan dinamis berdasarkan spesifikasi kapasitas target Anda. Untuk informasi selengkapnya, lihat Penyedia kapasitas grup Auto Scaling Amazon EC2.

Dengan rilis baru ini, Anda dapat menggunakan grup Auto Scaling yang ada dengan kebijakan penskalaan untuk digunakan saat membuat penyedia kapasitas baru. Kami tidak menyarankan Anda memodifikasi kebijakan penskalaan terkelola ECS atau sumber daya rencana. Namun, saat membuat sumber daya penyedia kapasitas baru, jika Anda memiliki perkakas khusus yang membuat modifikasi pada rencana AWS Auto Scaling penskalaan, lakukan salah satu hal berikut:

- (Disarankan) Perbarui penyedia kapasitas Anda untuk mengubah pengaturan penskalaan terkelola Amazon ECS. Untuk informasi lebih lanjut, lihat UpdateCapacityProvider.
- Perbarui kebijakan penskalaan yang terkait dengan grup Auto Scaling Anda untuk mengubah konfigurasi pelacakan target yang digunakan. Untuk informasi lebih lanjut, lihat PutScalingPolicy.

Pertimbangan

Pertimbangkan hal berikut saat menggunakan penskalaan otomatis cluster:

- Jangan mengubah atau mengelola kapasitas yang diinginkan untuk grup Auto Scaling yang terkait dengan penyedia kapasitas dengan kebijakan penskalaan apa pun selain yang dikelola Amazon ECS.
- Amazon ECS menggunakan peran IAM `AWSServiceRoleForECS` terkait layanan untuk izin yang diperlukan untuk memanggil atas nama Anda. AWS Auto Scaling Untuk informasi selengkapnya tentang menggunakan dan membuat peran IAM terkait layanan Amazon ECS, lihat. Menggunakan peran terkait layanan untuk Amazon ECS
- Saat menggunakan penyedia kapasitas dengan grup Auto Scaling, pengguna, grup, atau peran yang membuat penyedia kapasitas memerlukan izin. `autoscaling:CreateOrUpdateTags` Ini karena Amazon ECS menambahkan tag ke grup Auto Scaling saat mengaitkannya dengan penyedia kapasitas.

> **Important**
>
> Pastikan alat apa pun yang Anda gunakan tidak menghapus `AmazonECSManaged` tag dari grup Auto Scaling. Jika tag ini dihapus, Amazon ECS tidak dapat mengelola penskalaan.

- Penskalaan otomatis cluster tidak mengubah MinimumCapacityatau MaximumCapacityuntuk grup. Agar kelompok dapat diskalakan, nilai untuk MaximumCapacityharus lebih besar dari nol.
- Saat Auto Scaling (penskalaan terkelola) diaktifkan, penyedia kapasitas hanya dapat dihubungkan ke satu cluster pada saat yang bersamaan. Jika penyedia kapasitas Anda telah mengelola penskalaan dimatikan, Anda dapat mengaitkannya dengan beberapa cluster.
- Saat penskalaan terkelola dimatikan, penyedia kapasitas tidak akan menskalakan atau memperkecil skala. Anda dapat menggunakan strategi penyedia kapasitas untuk menyeimbangkan tugas Anda antara penyedia kapasitas.
- `binpack`Strategi adalah strategi yang paling efisien dalam hal kapasitas.
- Ketika kapasitas target kurang dari 100%, strategi penempatan perlu menggunakan `binpack` strategi tanpa `spread` strategi memiliki urutan yang lebih tinggi dari `binpack` strategi. Ini mencegah penyedia kapasitas dari skala hingga setiap tugas memiliki instance khusus atau batasnya tercapai.

Pertimbangkan hal berikut saat Anda menggunakan konsol:

- Secara default, fitur penskalaan terkelola Amazon ECS aktif. Untuk informasi selengkapnya, lihat Perilaku penskalaan keluar yang terkelola.
- Secara default, perlindungan terminasi terkelola tidak aktif. Untuk informasi lebih lanjut, lihat bagian selanjutnyathe section called “Perlindungan terminasi terkelola”.
- Secara default, perlindungan penskalaan instans Auto Scaling tidak aktif. Untuk informasi selengkapnya, lihat Menggunakan perlindungan penskalaan ke dalam instans dalam Panduan Pengguna Amazon EC2 Auto Scaling.
- Grup Auto Scaling yang digunakan dengan penyedia kapasitas Anda tidak dapat menggunakan pengaturan pembobotan instans. Pembobotan instans tidak didukung saat digunakan bersama dengan penyedia kapasitas Amazon ECS.

Ikhtisar penskalaan otomatis cluster

Untuk setiap penyedia kapasitas grup Auto Scaling yang terkait dengan kluster, Amazon ECS membuat dan mengelola sumber daya berikut:

- CloudWatch Alarm nilai metrik rendah
- CloudWatch Alarm nilai metrik tinggi
- Kebijakan penskalaan pelacakan target

> **Note**
>
> Amazon ECS membuat kebijakan penskalaan pelacakan target dan menempelkannya ke grup Auto Scaling. Untuk memperbarui kebijakan penskalaan pelacakan target, perbarui pengaturan penskalaan terkelola penyedia kapasitas, daripada memperbarui kebijakan penskalaan secara langsung.

Saat Anda menonaktifkan penskalaan terkelola atau memisahkan penyedia kapasitas dari kluster, Amazon ECS akan menghapus CloudWatch metrik dan sumber daya kebijakan penskalaan pelacakan target.

Amazon ECS menggunakan metrik berikut untuk menentukan tindakan apa yang harus diambil:

`CapacityProviderReservation`

Persentase instance kontainer yang digunakan untuk penyedia kapasitas tertentu. Amazon ECS menghasilkan metrik ini.

Amazon ECS menetapkan `CapacityProviderReservation` nilai ke angka antara 0-100. Amazon ECS menggunakan rumus berikut untuk mewakili rasio berapa banyak kapasitas yang tersisa dalam grup Auto Scaling. Kemudian, Amazon ECS menerbitkan metrik ke. CloudWatch Untuk informasi selengkapnya tentang cara penghitungan metrik, lihat Deep Dive di Auto Scaling Amazon ECS Cluster

```
CapacityProviderReservation = (number of instances needed) / (number of running
 instances) x 100
```

`DesiredCapacity`

Jumlah kapasitas untuk grup Auto Scaling.

Amazon ECS melakukan proses penskalaan otomatis cluster untuk setiap penyedia kapasitas yang terkait dengan kluster Anda. Amazon ECS melakukan proses setiap menit.

Amazon ECS menerbitkan `CapacityProviderReservation` metrik ke CloudWatch dalam namespace. `AWS/ECS/ManagedScaling` `CapacityProviderReservation`Metrik menyebabkan salah satu tindakan berikut terjadi:

`CapacityProviderReservation`Nilainya sama dengan **`targetCapacity`**

Grup Auto Scaling tidak perlu menskalakan atau memperkecil skala. Target persentase pemanfaatan telah tercapai.

`CapacityProviderReservation`Nilainya lebih besar dari **`targetCapacity`**

Ada lebih banyak tugas menggunakan persentase kapasitas yang lebih tinggi daripada `targetCapacity` persentase Anda. Peningkatan nilai `CapacityProviderReservation` metrik menyebabkan CloudWatch alarm terkait bertindak. Alarm ini memperbarui `DesiredCapacity` nilai untuk grup Auto Scaling. Grup Auto Scaling menggunakan nilai ini untuk meluncurkan instans EC2, lalu mendaftarkannya ke cluster.

Ketika `targetCapacity` adalah nilai default 100%, tugas baru berada dalam `PENDING` status selama scale-out karena tidak ada kapasitas yang tersedia pada instance untuk menjalankan tugas. Setelah instans baru mendaftar dengan ECS, tugas-tugas ini akan dimulai pada instans baru.

`CapacityProviderReservation`Nilai kurang dari **`targetCapacity`**

Ada lebih sedikit tugas yang menggunakan persentase kapasitas yang lebih rendah daripada `targetCapacity` persentase Anda dan setidaknya ada satu contoh yang dapat dihentikan. Nilai `CapacityProviderReservation` metrik yang menurun menyebabkan CloudWatch alarm terkait bertindak. Alarm ini memperbarui `DesiredCapacity` nilai untuk grup Auto Scaling. Grup Auto Scaling menggunakan nilai ini untuk menghentikan instans kontainer EC2, dan kemudian membatalkan pendaftarannya dari cluster.

Grup Auto Scaling mengikuti kebijakan penghentian grup untuk menentukan instance mana yang dihentikan terlebih dahulu selama peristiwa penskalaan. Selain itu, ini menghindari instance dengan pengaturan perlindungan skala masuk instance diaktifkan. Penskalaan otomatis cluster dapat mengelola instance mana yang memiliki setelan perlindungan penskalaan instans jika Anda mengaktifkan perlindungan terminasi terkelola. Untuk informasi selengkapnya tentang perlindungan terminasi terkelola, lihatPerlindungan terminasi terkelola. Untuk informasi

selengkapnya tentang cara grup Auto Scaling menghentikan instans, lihat Mengontrol instans Auto Scaling mana yang dihentikan selama penskalaan di Panduan Pengguna Auto Scaling Amazon EC2.

Perlindungan terminasi terkelola

> **Important**
>
> Anda harus mengaktifkan perlindungan penskalaan instans Auto Scaling pada grup Auto Scaling untuk menggunakan fitur proteksi terminasi terkelola penskalaan otomatis cluster.

Timbangan penskalaan otomatis klaster Amazon ECS di grup Auto Scaling bila CapacityProviderReservation nilainya `targetCapacity` kurang dari persentase yang Anda tetapkan. Penskalaan otomatis cluster dapat mengontrol instance mana yang dihentikan jika Anda mengaktifkan perlindungan terminasi terkelola. Saat Anda menggunakan perlindungan penghentian terkelola, Amazon ECS hanya menghentikan instans EC2 yang tidak menjalankan tugas Amazon ECS. Tugas yang dijalankan oleh layanan yang menggunakan strategi `DAEMON` penjadwalan diabaikan dan sebuah instance dapat dihentikan oleh penskalaan otomatis cluster bahkan ketika instance menjalankan tugas-tugas ini. Ini karena semua instance di cluster menjalankan tugas-tugas ini.

Saat Anda menggunakan proteksi terminasi terkelola, Amazon ECS akan mengaktifkan opsi perlindungan penskalaan instans untuk instans EC2 di grup Auto Scaling terlebih dahulu. Kemudian, Amazon ECS menempatkan tugas pada instans. Saat semua tugas non-daemon dihentikan pada instance, Amazon ECS memulai proses scale-in dan menonaktifkan perlindungan scale-in untuk instans EC2. Grup Auto Scaling kemudian dapat menghentikan instance.

Perlindungan penskalaan instans Auto Scaling mengontrol instans EC2 mana yang dapat dihentikan oleh Auto Scaling. Instans dengan fitur scale-in diaktifkan tidak dapat dihentikan selama proses scale-in. Untuk informasi selengkapnya tentang perlindungan penskalaan instans Auto Scaling, lihat Menggunakan perlindungan penskalaan instans di Panduan Pengguna Auto Scaling Amazon EC2.

Anda dapat mengatur `targetCapacity` persentase sehingga Anda memiliki kapasitas cadangan. Ini membantu tugas future diluncurkan lebih cepat karena grup Auto Scaling tidak harus meluncurkan lebih banyak instance. Amazon ECS menggunakan nilai kapasitas target untuk mengelola CloudWatch metrik yang dibuat layanan untuk memfasilitasi penskalaan otomatis cluster. Amazon ECS mengelola CloudWatch metrik. Dengan cara ini, grup Auto Scaling diperlakukan sebagai kondisi

tunak sehingga tidak diperlukan tindakan penskalaan. Nilainya bisa dari 0-100%. Misalnya, untuk mengonfigurasi Amazon ECS agar kapasitas bebas 10% di atas yang digunakan oleh tugas Amazon ECS, tetapkan nilai kapasitas target menjadi 90%. Pertimbangkan hal berikut saat menetapkan `targetCapacity` nilai pada penyedia kapasitas.

- `targetCapacity`Nilai kurang dari 100% mewakili jumlah kapasitas bebas (instans Amazon EC2) yang perlu ada di cluster. Kapasitas bebas berarti tidak ada tugas yang berjalan.
- Batasan penempatan seperti Availability Zones, tanpa tambahan memaksa `binpack` Amazon ECS untuk akhirnya menjalankan satu tugas untuk setiap instance, yang mungkin bukan perilaku yang diinginkan. Untuk mencegah perilaku ini, jangan gunakan `spread` strategi bersama dengan `binpack` strategi.

Pertimbangkan hal berikut saat menggunakan perlindungan terminasi terkelola dengan konsol:

- Secara default, proteksi terminasi terkelola dimatikan untuk penyedia kapasitas baru yang Anda buat.
- Konsol tidak mengaktifkan perlindungan penskalaan instance dari grup Auto Scaling yang dipilih. Secara default, perlindungan penskalaan instans Auto Scaling tidak aktif. Anda harus mengaktifkan perlindungan penskalaan instans Auto Scaling pada grup Auto Scaling untuk menggunakan perlindungan terminasi terkelola. Jika Anda tidak mengaktifkan perlindungan skala dalam, maka mengaktifkan perlindungan terminasi terkelola dapat menyebabkan perilaku yang tidak diinginkan. Misalnya, Anda mungkin memiliki instance yang terjebak dalam keadaan menguras. Untuk informasi selengkapnya, lihat Menggunakan perlindungan penskalaan ke dalam instans dalam Panduan Pengguna Amazon EC2 Auto Scaling.
- Saat Anda menggunakan perlindungan penghentian dengan penyedia kapasitas, jangan lakukan tindakan manual apa pun, seperti melepaskan instance, pada grup Auto Scaling yang terkait dengan penyedia kapasitas. Tindakan manual dapat mematahkan operasi skala dalam penyedia kapasitas. Jika Anda melepaskan instance dari grup Auto Scaling, Anda juga perlu membatalkan pendaftaran instans terpisah dari cluster Amazon ECS.

Perilaku penskalaan keluar yang terkelola

Jika Anda memiliki penyedia kapasitas grup Auto Scaling yang menggunakan penskalaan terkelola, Amazon ECS memperkirakan jumlah instans optimal yang akan ditambahkan ke klaster Anda dan menggunakan nilainya untuk menentukan jumlah instans yang akan diminta. Berikut ini menjelaskan perilaku penskalaan keluar secara lebih rinci.

1. Amazon ECS memilih penyedia kapasitas untuk setiap tugas dengan mengikuti strategi penyedia kapasitas dari layanan, dari tugas mandiri, atau dari default cluster. Amazon ECS mengikuti langkah-langkah lainnya untuk penyedia kapasitas tunggal.

   Tugas tanpa strategi penyedia kapasitas diabaikan oleh penyedia kapasitas. Tugas yang tertunda yang tidak memiliki strategi penyedia kapasitas tidak akan menyebabkan penyedia kapasitas untuk meningkatkan skala. Tugas atau layanan tidak dapat menetapkan strategi penyedia kapasitas jika tugas atau layanan tersebut menetapkan jenis peluncuran.
2. Kelompokkan semua tugas penyediaan untuk penyedia kapasitas ini sehingga setiap kelompok memiliki persyaratan sumber daya yang sama persis.
3. Bila Anda menggunakan beberapa tipe instans dalam grup Auto Scaling, tipe instans dalam grup Auto Scaling akan diurutkan berdasarkan parameternya. Parameter ini termasuk vCPU, memori, antarmuka jaringan elastis (ENI), port, dan GPU. Jenis instance terkecil dan terbesar untuk setiap parameter dipilih. Untuk informasi selengkapnya tentang cara memilih jenis instans, lihat Mengkarakterisasi aplikasi Anda di Panduan Praktik Terbaik Amazon ECS.

   > **Important**
   >
   > Jika sekelompok tugas memiliki persyaratan sumber daya yang lebih besar dari jenis instans terkecil di grup Auto Scaling, maka grup tugas tersebut tidak dapat dijalankan dengan penyedia kapasitas ini. Penyedia kapasitas tidak menskalakan grup Auto Scaling. Tugas tetap di `PENDING` negara bagian.
   > Untuk mencegah tugas tetap berada dalam `PENDING` status, kami sarankan Anda membuat grup Auto Scaling terpisah dan penyedia kapasitas untuk persyaratan sumber daya minimum yang berbeda. Saat menjalankan tugas atau membuat layanan, tambahkan saja penyedia kapasitas ke strategi penyedia kapasitas yang dapat menjalankan tugas pada jenis instans terkecil di grup Auto Scaling. Untuk parameter lain, Anda dapat menggunakan batasan penempatan
4. Untuk setiap grup tugas, Amazon ECS menghitung jumlah instance yang diperlukan untuk menjalankan tugas yang tidak ditempatkan. Perhitungan ini menggunakan `binpack` strategi. Strategi ini memperhitungkan vCPU, memori, antarmuka jaringan elastis (ENI), port, dan persyaratan GPU dari tugas. Ini juga memperhitungkan ketersediaan sumber daya instans Amazon EC2. Nilai untuk jenis instans terbesar diperlakukan sebagai jumlah instans terhitung maksimum. Nilai untuk jenis instance terkecil digunakan sebagai perlindungan. Jika jenis instance terkecil tidak dapat menjalankan setidaknya satu instance tugas, perhitungan menganggap tugas sebagai tidak kompatibel. Akibatnya, tugas tersebut dikecualikan dari perhitungan scale-out. Ketika

semua tugas tidak kompatibel dengan jenis instance terkecil, penskalaan otomatis cluster berhenti dan `CapacityProviderReservation` nilainya tetap pada `targetCapacity` nilainya.

5. Amazon ECS menerbitkan `CapacityProviderReservation` metrik sehubungan CloudWatch dengan `minimumScalingStepSize` jika salah satu dari berikut ini adalah kasusnya. Entah, jumlah instans maksimum yang dihitung kurang dari ukuran langkah penskalaan minimum. Atau, nilai yang lebih rendah dari jumlah instans terhitung maksimum `maximumScalingStepSize` atau maksimum.
6. CloudWatch alarm menggunakan `CapacityProviderReservation` metrik untuk penyedia kapasitas. Ketika `CapacityProviderReservation` metrik lebih besar dari `targetCapacity` nilainya, alarm juga meningkatkan grup `DesiredCapacity` Auto Scaling. `targetCapacity`Nilainya adalah pengaturan penyedia kapasitas yang dikirim ke CloudWatch alarm selama fase aktivasi penskalaan otomatis cluster.

   Anda dapat mengatur `targetCapacity` kapan Anda membuat grup Auto Scaling, atau memodifikasi nilai setelah grup dibuat. Bawaannya adalah 100%.
7. Grup Auto Scaling meluncurkan instans EC2 tambahan. Untuk mencegah penyediaan berlebih, Auto Scaling memastikan bahwa kapasitas instans EC2 yang baru diluncurkan distabilkan sebelum meluncurkan instans baru. Auto Scaling memeriksa apakah semua instance yang ada telah melewati `instanceWarmupPeriod` (sekarang dikurangi waktu peluncuran instance). Penskalaan diblokir untuk instance yang ada di dalam. `instanceWarmupPeriod`

   Jumlah detik default untuk instance yang baru diluncurkan untuk pemanasan adalah 300.

Untuk informasi selengkapnya, lihat Menyelam mendalam tentang penskalaan otomatis klaster Amazon ECS.

Pertimbangan skala-out

Pertimbangkan hal berikut untuk proses scale-out:

1. Meskipun ada beberapa kendala penempatan, kami menyarankan Anda hanya menggunakan kendala penempatan `distinctInstance` tugas. Ini mencegah proses scale-out berhenti karena Anda menggunakan batasan penempatan yang tidak kompatibel dengan instance sampel.
2. Penskalaan terkelola berfungsi paling baik jika grup Auto Scaling Anda menggunakan jenis instans yang sama atau serupa. Untuk informasi selengkapnya, lihat Perilaku penskalaan keluar yang terkelola.

3. Ketika proses scale-out diperlukan dan tidak ada instance container yang sedang berjalan, Amazon ECS selalu melakukan scale-out ke dua instance pada awalnya, dan kemudian melakukan proses scale-out atau scale-in tambahan. Setiap scale-out tambahan menunggu periode pemanasan instance. Untuk proses scale-in, Amazon ECS menunggu 15 menit setelah proses scale-out sebelum memulai proses scale-in setiap saat.
4. Langkah scale-out kedua harus menunggu sampai `instanceWarmupPeriod` habis masa berlakunya, yang mungkin mempengaruhi batas skala keseluruhan. Jika Anda perlu mengurangi waktu ini, pastikan itu `instanceWarmupPeriod` cukup besar untuk instans EC2 untuk meluncurkan dan memulai agen Amazon ECS (yang mencegah penyediaan berlebih).
5. Penskalaan otomatis cluster mendukung Konfigurasi Peluncuran, Template Peluncuran, dan beberapa jenis instans di grup Auto Scaling penyedia kapasitas. Anda juga dapat menggunakan pemilihan tipe instans berbasis atribut tanpa beberapa jenis instance.
6. Saat menggunakan grup Auto Scaling dengan instans Sesuai Permintaan dan beberapa jenis instans atau Instans Spot, tempatkan jenis instans yang lebih besar lebih tinggi dalam daftar prioritas dan jangan tentukan bobot. Menentukan bobot tidak didukung saat ini. Untuk informasi selengkapnya, lihat grup Auto Scaling dengan beberapa jenis instans di AWS Auto Scaling Panduan Pengguna.
7. Amazon ECS kemudian meluncurkan`minimumScalingStepSize`, jika jumlah instans maksimum yang dihitung kurang dari ukuran langkah penskalaan minimum, atau yang lebih rendah dari nilai hitungan instans terhitung maksimum `maximumScalingStepSize` atau maksimum.
8. Jika layanan Amazon ECS atau `run-task` API meluncurkan tugas dan instans wadah penyedia kapasitas tidak memiliki sumber daya yang cukup untuk memulai tugas, Amazon ECS membatasi jumlah tugas dengan status ini untuk setiap klaster dan mencegah tugas apa pun melebihi batas ini. Untuk informasi selengkapnya, lihat Kuota layanan.

Perilaku scale-in yang dikelola

Amazon ECS memantau instans kontainer untuk setiap penyedia kapasitas dalam klaster. Saat instance container tidak menjalankan tugas apa pun, instance container dianggap kosong dan Amazon ECS memulai proses scale-in. Berikut ini menjelaskan perilaku scale-in secara lebih rinci:

1. Amazon ECS menghitung jumlah instance kontainer yang kosong. Sebuah instance container dianggap kosong bahkan ketika tugas daemon sedang berjalan.
2. Amazon ECS menetapkan `CapacityProviderReservation` nilai ke angka antara 0-100 yang menggunakan rumus berikut untuk mewakili rasio seberapa besar kelompok Auto Scaling

harus relatif terhadap seberapa besar sebenarnya, dinyatakan sebagai persentase. Kemudian, Amazon ECS menerbitkan metrik ke. CloudWatch Untuk informasi selengkapnya tentang cara penghitungan metrik, lihat Deep Dive di Auto Scaling Amazon ECS Cluster

```
CapacityProviderReservation = (number of instances needed) / (number of running
 instances) x 100
```

3. `CapacityProviderReservation`Metrik menghasilkan CloudWatch alarm. Alarm ini memperbarui `DesiredCapacity` nilai untuk grup Auto Scaling. Kemudian, salah satu tindakan berikut terjadi:
   - Jika Anda tidak menggunakan penghentian terkelola penyedia kapasitas, grup Auto Scaling akan memilih instans EC2 menggunakan kebijakan penghentian grup Auto Scaling dan mengakhiri instans hingga jumlah instans EC2 mencapai instans. `DesiredCapacity` Instance kontainer kemudian dideregistrasi dari cluster.
   - Jika semua instance container menggunakan proteksi terminasi terkelola, Amazon ECS akan menghapus proteksi scale-in pada instance container yang kosong. Grup Auto Scaling kemudian akan dapat menghentikan instans EC2. Instance kontainer kemudian dideregistrasi dari cluster.

Pertimbangan skala

Pertimbangkan hal berikut untuk proses penskalaan:

- Jika tidak ada tugas non-daemon yang berjalan, instans penampung Amazon ECS dianggap tersedia untuk diskalakan.
- CloudWatch alarm scale-in memerlukan 15 titik data (15 menit) sebelum proses scale-in untuk grup Auto Scaling dimulai. Setelah proses penskalaan dimulai hingga Amazon ECS perlu mengurangi jumlah instans kontainer yang terdaftar, grup Auto Scaling menetapkan `DesireCapacity` nilainya menjadi lebih besar dari satu instans dan kurang dari 50% setiap menit.
- Saat Amazon ECS meminta scale-out (bila `CapacityProviderReservation` lebih besar dari 100) saat proses scale-in sedang berlangsung, proses scale-in dihentikan dan dimulai dari awal jika diperlukan.

Perbarui cara Amazon ECS membuat sumber daya untuk penskalaan otomatis cluster

Pada 27 Mei 2022, Amazon ECS mengubah cara mengelola sumber daya yang memfasilitasi penskalaan otomatis cluster. Untuk menyederhanakan pengalaman, Amazon ECS tidak lagi memerlukan rencana penskalaan saat Anda mengaktifkan AWS Auto Scaling penskalaan terkelola untuk penyedia kapasitas grup Auto Scaling.

> **Important**
>
> Perubahan ini tidak berdampak fungsional pada alur kerja penskalaan otomatis klaster Anda dan tidak ada dampak harga atau kinerja.

Penyedia kapasitas dibuat sebelum 27 Mei 2022

Penyedia kapasitas yang dibuat sebelum 27 Mei 2022, dan yang menggunakan rencana AWS Auto Scaling penskalaan terus berfungsi seperti sebelumnya.

Tinjau pertimbangan berikut:

- Kami tidak menyarankan Anda memperbarui atau menghapus rencana `ECS-managed` AWS Auto Scaling penskalaan atau sumber daya kebijakan penskalaan yang terkait dengan penyedia kapasitas Anda.
- Anda dapat mengakses sumber daya rencana penskalaan untuk cluster di konsol Auto Scaling dan dengan `describe-clusters` lampiran with. Untuk informasi selengkapnya, lihat dokumentasi API DescribeClusters.
- Anda tidak dapat menambahkan kebijakan penskalaan ke grup Auto Scaling yang berfungsi sebagai penyedia kapasitas cluster.
- Jumlah paket Auto Scaling untuk setiap akun terbatas. Untuk informasi selengkapnya, lihat Kuota untuk paket penskalaan Anda di Panduan Pengguna Auto Scaling Amazon EC2.

Penyedia kapasitas dibuat pada atau setelah 27 Mei 2022

Per 27 Mei 2022, Amazon ECS tidak lagi membuat rencana AWS Auto Scaling penskalaan untuk penyedia kapasitas yang baru dibuat. Sebagai gantinya, Amazon ECS menggunakan kebijakan penskalaan pelacakan target yang dilampirkan pada grup Auto Scaling untuk melakukan penskalaan

dinamis berdasarkan spesifikasi kapasitas target Anda. Untuk informasi selengkapnya, lihat Penyedia kapasitas grup Auto Scaling Amazon EC2.

Dengan rilis baru ini, Anda dapat menggunakan grup Auto Scaling yang ada dengan kebijakan penskalaan untuk digunakan saat membuat penyedia kapasitas baru. Kami tidak menyarankan Anda memodifikasi kebijakan penskalaan terkelola ECS atau sumber daya rencana. Namun, saat membuat sumber daya penyedia kapasitas baru, jika Anda memiliki perkakas khusus yang membuat modifikasi pada rencana AWS Auto Scaling penskalaan, lakukan salah satu hal berikut:

- (Disarankan) Perbarui penyedia kapasitas Anda untuk mengubah pengaturan penskalaan terkelola Amazon ECS. Untuk informasi lebih lanjut, lihat UpdateCapacityProvider.
- Perbarui kebijakan penskalaan yang terkait dengan grup Auto Scaling Anda untuk mengubah konfigurasi pelacakan target yang digunakan. Untuk informasi lebih lanjut, lihat PutScalingPolicy.

Aktifkan penskalaan otomatis cluster

Anda dapat menggunakan AWS CLI untuk mengaktifkan penskalaan otomatis cluster.

Sebelum memulai, buat grup Auto Scaling dan penyedia kapasitas. Untuk informasi selengkapnya, lihat the section called “Penyedia kapasitas grup Auto Scaling Amazon EC2”.

Kaitkan penyedia kapasitas dengan cluster

Gunakan langkah-langkah berikut untuk mengaitkan penyedia kapasitas dengan cluster.

1. Gunakan `put-cluster-capacity-providers` perintah untuk mengaitkan satu atau lebih penyedia kapasitas dengan cluster.

   Untuk menambah penyedia AWS Fargate kapasitas, sertakan `FARGATE` dan penyedia `FARGATE_SPOT` kapasitas dalam permintaan. Untuk informasi selengkapnya, lihat `put-cluster-capacity-providers` dalam AWS CLI Referensi Perintah.

   ```
   aws ecs put-cluster-capacity-providers \
     --cluster ClusterName \
     --capacity-providers CapacityProviderName FARGATE FARGATE_SPOT \
     --default-capacity-provider-strategy capacityProvider=CapacityProvider,weight=1
   ```

Untuk menambahkan grup Auto Scaling untuk tipe peluncuran EC2, sertakan nama grup Auto Scaling dalam permintaan. Untuk informasi selengkapnya, lihat `put-cluster-capacity-providers` dalam AWS CLI Referensi Perintah.

```
aws ecs put-cluster-capacity-providers \
  --cluster ClusterName \
  --capacity-providers CapacityProviderName \
  --default-capacity-provider-strategy capacityProvider=CapacityProvider,weight=1
```

2. Gunakan `describe-clusters` perintah untuk memverifikasi bahwa asosiasi berhasil. Untuk informasi selengkapnya, lihat `describe-clusters` dalam AWS CLI Referensi Perintah.

```
aws ecs describe-clusters \
  --cluster ClusterName \
  --include ATTACHMENTS
```

Aktifkan penskalaan terkelola untuk penyedia kapasitas

Gunakan langkah-langkah berikut untuk mengaktifkan penskalaan terkelola untuk penyedia kapasitas.

- Gunakan `update-capacity-provider` perintah untuk mengaktifkan penskalaan otomatis terkelola untuk penyedia kapasitas. Untuk informasi selengkapnya, lihat `update-capacity-provider` dalam AWS CLI Referensi Perintah.

```
aws ecs update-capacity-provider \
  --capacity-providers CapacityProviderName \
  --auto-scaling-group-provider managedScaling=ENABLED
```

Matikan penskalaan otomatis cluster

Anda dapat menggunakan AWS CLI untuk mematikan penskalaan otomatis cluster.

Untuk menonaktifkan penskalaan otomatis klaster untuk kluster, Anda dapat memisahkan penyedia kapasitas dengan penskalaan terkelola yang diaktifkan dari klaster atau dengan memperbarui penyedia kapasitas untuk menonaktifkan penskalaan terkelola.

Putuskan hubungan penyedia kapasitas dengan cluster

Gunakan langkah-langkah berikut untuk memisahkan penyedia kapasitas dengan klaster.

1. Gunakan `put-cluster-capacity-providers` perintah untuk memisahkan penyedia kapasitas grup Auto Scaling dengan cluster. Cluster dapat menjaga hubungan dengan penyedia AWS Fargate kapasitas. Untuk informasi selengkapnya, lihat `put-cluster-capacity-providers` dalam AWS CLI Referensi Perintah.

   ```
   aws ecs put-cluster-capacity-providers \
     --cluster ClusterName \
     --capacity-providers FARGATE FARGATE_SPOT \
     --default-capacity-provider-strategy '[]'
   ```

   Gunakan `put-cluster-capacity-providers` perintah untuk memisahkan penyedia kapasitas grup Auto Scaling dengan cluster. Untuk informasi selengkapnya, lihat `put-cluster-capacity-providers` dalam AWS CLI Referensi Perintah.

   ```
   aws ecs put-cluster-capacity-providers \
     --cluster ClusterName \
     --capacity-providers [] \
     --default-capacity-provider-strategy '[]'
   ```

2. Gunakan `describe-clusters` perintah untuk memverifikasi bahwa disasosiasi berhasil. Untuk informasi selengkapnya, lihat `describe-clusters` dalam AWS CLI Referensi Perintah.

   ```
   aws ecs describe-clusters \
     --cluster ClusterName \
     --include ATTACHMENTS
   ```

Matikan penskalaan terkelola untuk penyedia kapasitas

Gunakan langkah-langkah berikut untuk menonaktifkan penskalaan terkelola untuk penyedia kapasitas.

- Gunakan `update-capacity-provider` perintah untuk mematikan penskalaan otomatis terkelola untuk penyedia kapasitas. Untuk informasi selengkapnya, lihat `update-capacity-provider` dalam AWS CLI Referensi Perintah.

```
aws ecs update-capacity-provider \
  --capacity-providers CapacityProviderName \
  --auto-scaling-group-provider managedScaling=DISABLED
```

Manajemen klaster

Berikut ini adalah konsep umum tentang cluster Amazon ECS.

- Amazon ECS membuat cluster default. Anda dapat membuat klaster tambahan pada sebuah akun untuk menjaga sumber daya Anda terpisah.
- Cluster Wilayah AWS spesifik.
- Berikut ini adalah kemungkinan keadaan di mana sebuah cluster dapat berada.

  AKTIF

  Cluster siap menerima tugas dan, jika berlaku, Anda dapat mendaftarkan instance kontainer dengan cluster.

  PENYEDIAAN

  Cluster memiliki penyedia kapasitas yang terkait dengannya dan sumber daya yang dibutuhkan untuk penyedia kapasitas sedang dibuat.

  PEMBATALAN PENYEDIAAN

  Cluster memiliki penyedia kapasitas yang terkait dengannya dan sumber daya yang dibutuhkan untuk penyedia kapasitas sedang dihapus.

  Failed

  Cluster memiliki penyedia kapasitas yang terkait dengannya dan sumber daya yang dibutuhkan untuk penyedia kapasitas gagal dibuat.

  TIDAK AKTIF

  Cluster telah dihapus. Klaster dengan status `INACTIVE` mungkin tetap dapat ditemukan di akun Anda selama jangka waktu tertentu. Namun, perilaku ini dapat berubah di masa depan, jadi pastikan Anda tidak bergantung pada `INACTIVE` cluster yang bertahan.
- Cluster dapat berisi campuran tugas yang di-host AWS Fargate, instans Amazon EC2, atau instans eksternal. Tugas dapat berjalan pada infrastruktur Fargate atau EC2 sebagai tipe peluncuran atau strategi penyedia kapasitas. Jika Anda menggunakan EC2 sebagai tipe peluncuran, ECS

tidak melacak dan menskalakan kapasitas grup Auto Scaling Amazon EC2. Untuk informasi selengkapnya tentang tipe peluncuran, lihat Jenis peluncuran Amazon ECS.

- Sebuah cluster dapat berisi campuran penyedia kapasitas grup Auto Scaling dan penyedia kapasitas Fargate. Namun, ketika Anda menentukan strategi penyedia kapasitas, mereka mungkin hanya berisi satu atau yang lain tetapi tidak keduanya. Untuk informasi selengkapnya, lihat Penyedia kapasitas Amazon ECS.
- Untuk tugas yang menggunakan tipe peluncuran EC2 atau penyedia kapasitas grup Auto Scaling, cluster dapat berisi beberapa jenis instans kontainer yang berbeda. Namun, setiap instance kontainer hanya dapat didaftarkan ke satu cluster pada satu waktu.
- Kebijakan IAM khusus dapat dibuat untuk mengizinkan atau membatasi akses pengguna ke cluster tertentu. Untuk informasi selengkapnya, lihat bagian Contoh klaster di Contoh kebijakan berbasis identitas untuk Amazon Elastic Container Service.
- Cluster dengan tugas Fargate dapat diskalakan menggunakan Service Auto Scaling. Untuk informasi selengkapnya, lihat Secara otomatis menskalakan layanan Amazon ECS Anda.
- *Anda dapat mengonfigurasi namespace Service Connect default untuk sebuah klaster. Setelah Anda menetapkan namespace Service Connect default, layanan baru apa pun yang dibuat di klaster dapat ditambahkan sebagai layanan klien di namespace dengan mengaktifkan Service Connect. Tidak diperlukan konfigurasi tambahan. Untuk informasi lebih lanjut, lihat Layanan Connect *.
- Jika Anda menggunakan instans EC2, kapasitas cluster dapat ditemukan di salah satu sumber daya berikut: AWS

  Untuk informasi tentang cara menggunakan sumber daya ini dengan Amazon ECS lihatAplikasi Amazon ECS di subnet bersama, Local Zones, dan Wavelength Zones.

  - Zona Ketersediaan
  - Zona Lokal
  - Wavelength Zones
  - AWS Outposts

Membuat cluster untuk jenis peluncuran Fargate dan Eksternal menggunakan konsol

Anda dapat membuat cluster Amazon ECS menggunakan konsol Amazon ECS. Sebelum memulai, pastikan Anda telah menyelesaikan langkah-langkah Siapkan untuk menggunakan Amazon ECS dan

menetapkan izin IAM yang sesuai. Untuk informasi selengkapnya, lihat the section called “Contoh klaster”. Konsol Amazon ECS menyediakan cara sederhana untuk membuat sumber daya yang dibutuhkan oleh cluster Amazon ECS dengan membuat AWS CloudFormation tumpukan.

Untuk membuat proses pembuatan cluster semudah mungkin, konsol memiliki pilihan default untuk banyak pilihan yang kami jelaskan di bawah ini. Ada juga panel bantuan yang tersedia untuk porsi besar bagian di konsol yang menyediakan konteks lebih lanjut.

- Membuat namespace default dengan nama AWS Cloud Map yang sama dengan cluster. Namespace memungkinkan layanan yang Anda buat di cluster untuk terhubung ke layanan lain di namespace tanpa konfigurasi tambahan.

  Untuk informasi selengkapnya, lihat Layanan interkoneksi.

Anda dapat mengubah opsi berikut:

- Ubah namespace default yang terkait dengan cluster.

  Namespace memungkinkan layanan yang Anda buat di cluster dapat terhubung ke layanan lain di namespace tanpa konfigurasi tambahan. Namespace default sama dengan nama cluster. Untuk informasi selengkapnya, lihat Layanan interkoneksi.
- Konfigurasikan cluster untuk instance eksternal
- Aktifkan Wawasan Kontainer.

  CloudWatch Container Insights mengumpulkan, mengumpulkan, dan merangkum metrik dan log dari aplikasi dan layanan mikro dalam kontainer Anda. Container Insights juga menyediakan informasi diagnostik, seperti kegagalan restart kontainer, yang Anda gunakan untuk mengisolasi masalah dan menyelesaikannya dengan cepat. Untuk informasi selengkapnya, lihat the section called “Pantau kontainer Amazon ECS menggunakan Wawasan Kontainer”.
- Tambahkan tag untuk membantu Anda mengidentifikasi klaster Anda.

Untuk membuat cluster baru (konsol Amazon ECS)

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.
3. Di panel navigasi, pilih Klaster.
4. Pada halaman Klaster, pilih Buat klaster.

5. Di bawah konfigurasi Cluster, konfigurasikan berikut ini:

   - Untuk nama Cluster, masukkan nama unik.

     Nama tersebut dapat berisi hingga 255 huruf (huruf besar dan huruf kecil), angka, dan tanda hubung.

   - (Opsional) Agar namespace yang digunakan untuk Service Connect berbeda dari nama cluster, untuk Namespace, masukkan nama yang unik.

6. (Opsional) Gunakan instans eksternal (Amazon ECS Anywhere) untuk kapasitas Anda. Perluas Infrastruktur, hapus AWS Fargate (tanpa server), lalu pilih Instans eksternal menggunakan ECS Anywhere.
7. (Opsional) Untuk mengaktifkan Wawasan Kontainer, perluas Pemantauan, lalu aktifkan Gunakan Wawasan Kontainer.
8. (Opsional) Untuk membantu mengidentifikasi klaster Anda, perluas Tag, lalu konfigurasikan tag Anda.

   [Tambahkan tag] Pilih Tambah tag dan lakukan hal berikut:

   - Untuk Kunci, masukkan nama kunci.
   - Untuk Nilai, masukkan nilai kunci.

   [Hapus tag] Pilih Hapus di sebelah kanan Kunci dan Nilai tag.
9. Pilih Create (Buat).

Langkah selanjutnya

Jika Anda menggunakan instance eksternal, Anda harus mendaftarkan instance dengan cluster. Untuk informasi selengkapnya, lihat Pendaftaran instans eksternal ke sebuah klaster.

Setelah Anda membuat cluster, Anda dapat membuat definisi tugas untuk aplikasi Anda dan kemudian menjalankannya sebagai tugas mandiri, atau sebagai bagian dari layanan. Untuk informasi selengkapnya, lihat hal berikut:

- Definisi tugas Amazon ECS
- Jalankan aplikasi sebagai tugas Amazon ECS
- Membuat layanan menggunakan konsol

Membuat cluster untuk jenis peluncuran Amazon EC2 menggunakan konsol

Anda dapat membuat cluster Amazon ECS menggunakan konsol. Sebelum memulai, pastikan Anda telah menyelesaikan langkah-langkah Siapkan untuk menggunakan Amazon ECS dan menetapkan izin IAM yang sesuai. Untuk informasi selengkapnya, lihat the section called “Contoh klaster”. Konsol Amazon ECS menyediakan cara sederhana untuk membuat sumber daya yang dibutuhkan oleh cluster Amazon ECS dengan membuat AWS CloudFormation tumpukan.

Untuk membuat proses pembuatan cluster semudah mungkin, konsol memiliki pilihan default untuk banyak pilihan yang kami jelaskan di bawah ini. Ada juga panel bantuan yang tersedia untuk porsi besar bagian di konsol yang menyediakan konteks lebih lanjut.

Anda dapat mendaftarkan instans Amazon EC2 saat membuat klaster atau mendaftarkan instance tambahan dengan klaster setelah dibuat.

Anda dapat memodifikasi opsi default berikut:

- Ubah subnet tempat instans Anda diluncurkan
- Mengubah grup keamanan yang digunakan untuk mengontrol lalu lintas ke instance container
- Ubah namespace default yang terkait dengan cluster.

  Namespace memungkinkan layanan yang Anda buat di cluster dapat terhubung ke layanan lain di namespace tanpa konfigurasi tambahan. Namespace default sama dengan nama cluster. Untuk informasi selengkapnya, lihat Layanan interkoneksi.
- Aktifkan Wawasan Kontainer.

  CloudWatch Container Insights mengumpulkan, mengumpulkan, dan merangkum metrik dan log dari aplikasi dan layanan mikro dalam kontainer Anda. Container Insights juga menyediakan informasi diagnostik, seperti kegagalan restart kontainer, yang Anda gunakan untuk mengisolasi masalah dan menyelesaikannya dengan cepat. Untuk informasi selengkapnya, lihat the section called “Pantau kontainer Amazon ECS menggunakan Wawasan Kontainer”.
- Tambahkan tag untuk membantu Anda mengidentifikasi klaster Anda.

Opsi grup Auto Scaling

Saat menggunakan instans Amazon EC2, Anda harus menentukan grup Auto Scaling untuk mengelola infrastruktur tempat tugas dan layanan Anda berjalan.

Bila Anda memilih untuk membuat grup Auto Scaling baru, grup Auto Scaling secara otomatis dikonfigurasi untuk perilaku berikut:

- Amazon ECS mengelola tindakan scale-in dan scale-out dari grup Auto Scaling.
- Amazon ECS tidak akan mencegah instans Amazon EC2 yang berisi tugas dan yang ada dalam grup Auto Scaling dihentikan selama tindakan penskalaan. Untuk informasi selengkapnya, lihat Perlindungan Instance di Panduan AWS Auto Scaling Pengguna.

Anda mengonfigurasi properti grup Auto Scaling berikut yang menentukan jenis dan jumlah instance yang akan diluncurkan untuk grup:

- AMI Amazon ECS yang dioptimalkan.
- Tipe instans.
- Key pair SSH yang membuktikan identitas Anda saat Anda terhubung ke instance. Untuk informasi tentang cara membuat kunci SSH, lihat pasangan kunci Amazon EC2 dan instans Linux di Panduan Pengguna Amazon EC2 untuk Instans Linux.
- Jumlah minimum instans yang akan diluncurkan untuk grup Auto Scaling.
- Jumlah maksimum instans yang dimulai untuk grup Auto Scaling.

  Agar grup dapat ditingkatkan, maksimum harus lebih besar dari 0.

Amazon ECS membuat template peluncuran Auto Scaling Amazon EC2 dan grup Auto Scaling atas nama Anda sebagai bagian dari tumpukan. AWS CloudFormation Nilai yang Anda tentukan untuk AMI, tipe instance, dan key pair SSH adalah bagian dari template peluncuran. Template diawali dengan`EC2ContainerService-<ClusterName>`, yang membuatnya mudah diidentifikasi. Grup Auto Scaling diawali dengan. `<ClusterName>-ECS-Infra-ECSAutoScalingGroup`

Instans yang diluncurkan untuk grup Auto Scaling menggunakan template peluncuran.

Opsi jaringan

Secara default instance diluncurkan ke subnet default untuk Wilayah. Grup keamanan, yang mengontrol lalu lintas ke instans kontainer Anda, saat ini terkait dengan subnet digunakan. Anda dapat mengubah subnet dan grup keamanan untuk instans.

Anda dapat memilih subnet yang ada. Anda dapat menggunakan grup keamanan yang ada, atau membuat yang baru, Saat membuat grup keamanan baru, Anda harus menentukan setidaknya satu aturan masuk.

Aturan masuk menentukan lalu lintas apa yang dapat mencapai instans kontainer Anda dan menyertakan properti berikut:

- Protokol untuk memungkinkan
- Kisaran port untuk memungkinkan
- Lalu lintas masuk (sumber)

Untuk mengizinkan lalu lintas masuk dari alamat tertentu atau blok CIDR, gunakan Custom for Source dengan CIDR yang diizinkan.

Untuk memungkinkan lalu lintas masuk dari semua tujuan, gunakan Anywhere for Source. Ini secara otomatis menambahkan blok CIDR IPv4 0.0.0.0/0 dan blok CIDR: :/0 IPv6.

Untuk mengizinkan lalu lintas masuk dari komputer lokal Anda, gunakan grup Sumber untuk Sumber. Ini secara otomatis menambahkan alamat IP saat ini dari komputer lokal Anda sebagai sumber yang diizinkan.

Untuk membuat cluster baru (konsol Amazon ECS)

Sebelum Anda mulai, tetapkan izin IAM yang sesuai. Untuk informasi selengkapnya, lihat the section called “Contoh klaster”.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.
3. Di panel navigasi, pilih Klaster.
4. Pada halaman Klaster, pilih Buat klaster.
5. Di bawah konfigurasi Cluster, konfigurasikan berikut ini:

    - Untuk nama Cluster, masukkan nama unik.

      Nama tersebut dapat berisi hingga 255 huruf (huruf besar dan huruf kecil), angka, dan tanda hubung.
    - (Opsional) Agar namespace yang digunakan untuk Service Connect berbeda dari nama cluster, untuk Namespace, masukkan nama yang unik.

6. Tambahkan instans Amazon EC2 ke klaster Anda, perluas Infrastruktur, hapus AWS Fargate (tanpa server), lalu pilih instans Amazon EC2. Selanjutnya, konfigurasikan grup Auto Scaling yang bertindak sebagai penyedia kapasitas:

   a. Untuk menggunakan grup Auto Scaling yang ada, dari grup Auto Scaling (ASG), pilih grup.

   b. Untuk membuat grup Auto Scaling, dari grup Auto Scaling (ASG), pilih Buat grup baru, lalu berikan detail berikut tentang grup:

      - Untuk model Provisioning, pilih apakah akan menggunakan instans On-Demand atau Instans Spot.
      - Jika Anda memilih untuk menggunakan Instans Spot, untuk Strategi Alokasi, pilih kumpulan kapasitas Spot (tipe instans dan Availability Zones) yang digunakan untuk instans.

        Untuk sebagian besar beban kerja, Anda dapat memilih Kapasitas harga dioptimalkan.

        Untuk informasi selengkapnya, lihat Strategi alokasi untuk Instans Spot di Panduan Pengguna Amazon EC2 untuk Instans Linux.
      - Untuk Sistem Operasi/Arsitektur, pilih AMI Amazon ECS yang dioptimalkan untuk instans grup Auto Scaling.
      - Untuk jenis instans EC2, pilih jenis instans untuk beban kerja Anda.

        Penskalaan terkelola berfungsi paling baik jika grup Auto Scaling Anda menggunakan jenis instans yang sama atau serupa.
      - Untuk peran instans EC2, pilih peran instans kontainer yang ada, atau Anda dapat membuat yang baru.

        Untuk informasi selengkapnya, lihat Peran IAM instans wadah Amazon ECS.
      - Untuk Kapasitas, masukkan jumlah minimum dan jumlah maksimum instans yang akan diluncurkan di grup Auto Scaling.
      - Untuk key pair SSH, pilih pair yang membuktikan identitas Anda saat Anda terhubung ke instance.
      - Untuk memungkinkan gambar dan penyimpanan yang lebih besar, untuk ukuran volume Root EBS, masukkan nilai dalam GiB.

7. (Opsional) Untuk mengubah VPC dan subnet, di bawah instans Networking for Amazon EC2, lakukan salah satu operasi berikut:

- Untuk menghapus subnet, di bawah Subnet, pilih X untuk setiap subnet yang ingin Anda hapus.
- Untuk mengubah ke VPC selain VPC default, di bawah VPC, pilih VPC yang ada, dan kemudian di bawah Subnet, pilih subnet.
- Pilih grup keamanan. Di bawah Grup keamanan, pilih salah satu opsi berikut:
  - Untuk menggunakan grup keamanan yang ada, pilih Gunakan grup keamanan yang ada, lalu pilih grup keamanan.
  - Untuk membuat grup keamanan, pilih Buat grup keamanan baru. Kemudian, pilih Tambahkan aturan untuk setiap aturan masuk.

    Untuk informasi tentang aturan masuk, lihatOpsi jaringan.
- Untuk secara otomatis menetapkan alamat IP publik ke instans penampung Amazon EC2 Anda, untuk menetapkan IP publik secara otomatis, pilih salah satu opsi berikut:
  - Gunakan pengaturan subnet — Tetapkan alamat IP publik ke instance ketika subnet yang diluncurkan instance adalah subnet publik.
  - Aktifkan — Tetapkan alamat IP publik ke instans.

8. (Opsional) Untuk mengaktifkan Wawasan Kontainer, perluas Pemantauan, lalu aktifkan Gunakan Wawasan Kontainer.
9. (Opsional)

   Jika Anda menggunakan Runtime Monitoring dengan opsi manual dan Anda ingin cluster ini dipantau GuardDuty, pilih Tambahkan tag dan lakukan hal berikut:

   - Untuk Key, masukkan **`guardDutyRuntimeMonitoringManaged`**
   - Untuk Nilai, masukkan **`true`**.
10. (Opsional) Untuk mengelola tag cluster, memperluas Tag, dan kemudian melakukan salah satu operasi berikut:

    [Tambahkan tag] Pilih Tambah tag dan lakukan hal berikut:

    - Untuk Kunci, masukkan nama kunci.
    - Untuk Nilai, masukkan nilai kunci.

    [Hapus tag] Pilih Hapus di sebelah kanan Kunci dan Nilai tag.
11. Pilih Create (Buat).

Langkah selanjutnya

Setelah Anda membuat cluster, Anda dapat membuat definisi tugas untuk aplikasi Anda dan kemudian menjalankannya sebagai tugas mandiri, atau sebagai bagian dari layanan. Untuk informasi selengkapnya, lihat hal berikut:

- Definisi tugas Amazon ECS
- Jalankan aplikasi sebagai tugas Amazon ECS
- Membuat layanan menggunakan konsol

Memperbarui cluster menggunakan konsol

Anda dapat memodifikasi properti cluster berikut:

- Tetapkan penyedia kapasitas default

  Setiap cluster dapat memiliki satu atau lebih penyedia kapasitas dan strategi penyedia kapasitas opsional. Strategi penyedia kapasitas menentukan bagaimana tugas-tugas tersebar di penyedia kapasitas klaster ini. Saat menjalankan tugas mandiri atau membuat layanan, Anda dapat menggunakan strategi penyedia kapasitas default kluster atau strategi penyedia kapasitas yang mengganti tugas default.

  Penyedia kapasitas adalah alternatif untuk jenis peluncuran. Untuk informasi selengkapnya, lihat Konsep penyedia kapasitas.
- Aktifkan Wawasan Kontainer.

  CloudWatch Container Insights mengumpulkan, mengumpulkan, dan merangkum metrik dan log dari aplikasi dan layanan mikro dalam kontainer Anda. Container Insights juga menyediakan informasi diagnostik, seperti kegagalan restart kontainer, yang Anda gunakan untuk mengisolasi masalah dan menyelesaikannya dengan cepat. Untuk informasi selengkapnya, lihat the section called “Pantau kontainer Amazon ECS menggunakan Wawasan Kontainer”.
- Tambahkan tag untuk membantu Anda mengidentifikasi cluster Anda.

Untuk memperbarui cluster (konsol Amazon ECS)

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.

3. Pada halaman Clusters, pilih cluster.
4. Pada halaman Cluster: ***name***, pilih Update cluster.
5. Untuk mengatur penyedia kapasitas default, di bawah Strategi penyedia kapasitas default, pilih Tambahkan lebih banyak.

   a. Untuk penyedia Kapasitas, pilih penyedia kapasitas.

   b. (Opsional) Untuk Base, masukkan jumlah minimum tugas yang dijalankan pada penyedia kapasitas.

      Anda hanya dapat menetapkan nilai Dasar untuk satu penyedia kapasitas.

   c. (Opsional) Untuk Berat, masukkan persentase relatif dari jumlah total tugas yang diluncurkan yang menggunakan penyedia kapasitas yang ditentukan.

   d. (Opsional) Ulangi langkah-langkah untuk penyedia kapasitas tambahan.
6. Untuk mengaktifkan atau menonaktifkan Wawasan Kontainer, perluas Pemantauan, lalu aktifkan Gunakan Wawasan Kontainer.
7. Untuk membantu mengidentifikasi klaster Anda, perluas Tag, lalu konfigurasikan tag Anda.

   [Tambahkan tag] Pilih Tambah tag dan lakukan hal berikut:

   - Untuk Kunci, masukkan nama kunci.
   - Untuk Nilai, masukkan nilai kunci.

   [Hapus tanda] Pilih Hapus di sebelah kanan Kunci dan Nilai tanda.
8. Pilih Perbarui.

Menghapus cluster menggunakan konsol

Jika Anda selesai menggunakan klaster, Anda dapat menghapusnya. Setelah Anda menghapus cluster, itu transisi ke `INACTIVE` status. Klaster dengan status `INACTIVE` mungkin tetap dapat ditemukan di akun Anda selama jangka waktu tertentu. Namun, perilaku ini dapat berubah di masa depan, jadi sebaiknya Anda tidak mengandalkan klaster `INACTIVE` yang tersimpan.

Sebelum Anda menghapus cluster, Anda harus melakukan operasi berikut:

- Hapus semua layanan di cluster. Untuk informasi selengkapnya, lihat the section called “Menghapus layanan”.

- Hentikan semua tugas yang sedang berjalan. Untuk informasi selengkapnya, lihat the section called “Hentikan tugas Amazon ECS”.
- Deregister semua instance kontainer terdaftar di cluster. Untuk informasi selengkapnya, lihat the section called “Membatalkan pendaftaran instans kontainer”.
- Hapus namespace. Untuk informasi selengkapnya, lihat Menghapus namespace di Panduan Developer AWS Cloud Map .

Untuk menghapus cluster (konsol Amazon ECS)

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.
3. Di panel navigasi, pilih Klaster.
4. Pada halaman klaster, pilih klaster untuk menghapus.
5. Di bagian kanan atas halaman, pilih Hapus Klaster.

   Pesan ditampilkan ketika Anda tidak menghapus semua sumber daya yang terkait dengan cluster.
6. Di kotak konfirmasi, masukkan hapus ***`nama cluster`***.

Membuat penyedia kapasitas untuk cluster menggunakan konsol

Setelah pembuatan cluster selesai, Anda dapat membuat penyedia kapasitas baru (grup Auto Scaling) untuk tipe peluncuran EC2.

Sebelum Anda membuat penyedia kapasitas, Anda perlu membuat grup Auto Scaling. Untuk informasi selengkapnya, lihat grup Auto Scaling di Panduan Pengguna Auto Scaling Amazon EC2.

Untuk membuat penyedia kapasitas untuk cluster (konsol Amazon ECS)

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih cluster.
4. Pada halaman Cluster: ***`name`***, pilih Infrastructure, lalu pilih Create.
5. Pada halaman Buat penyedia kapasitas, konfigurasikan opsi berikut.

a. Di bawah Detail dasar, untuk nama penyedia Kapasitas, masukkan nama penyedia kapasitas unik.

b. Di bawah grup Auto Scaling, untuk grup Menggunakan Auto Scaling yang ada, pilih grup Auto Scaling.

c. (Opsional) Untuk mengonfigurasi kebijakan penskalaan, di bawah Kebijakan penskalaan, konfigurasikan opsi berikut.

- Agar Amazon ECS mengelola tindakan scale-in dan scale-out, pilih Aktifkan penskalaan terkelola.
- Untuk mencegah instans EC2 yang menjalankan tugas Amazon ECS dihentikan, pilih Aktifkan perlindungan penskalaan.
- Untuk Menetapkan kapasitas target, masukkan nilai target untuk CloudWatch metrik yang digunakan dalam kebijakan penskalaan pelacakan target yang dikelola Amazon ECS.

6. Pilih Buat.

Memperbarui penyedia kapasitas untuk cluster menggunakan konsol

Bila Anda menggunakan grup Auto Scaling sebagai penyedia kapasitas, Anda dapat mengubah kebijakan penskalaan grup.

Untuk memperbarui penyedia kapasitas untuk cluster (konsol Amazon ECS)

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih cluster.
4. Pada halaman Cluster: ***name***, pilih Infrastructure, lalu pilih Update.
5. Pada halaman Buat penyedia kapasitas, konfigurasikan opsi berikut.

- Di bawah grup Auto Scaling, di bawah kebijakan Penskalaan, konfigurasikan opsi berikut.
  - Agar Amazon ECS mengelola tindakan scale-in dan scale-out, pilih Aktifkan penskalaan terkelola.
  - Untuk mencegah instans EC2 yang menjalankan tugas Amazon ECS dihentikan, pilih Aktifkan perlindungan penskalaan.

- Untuk Menetapkan kapasitas target, masukkan nilai target untuk CloudWatch metrik yang digunakan dalam kebijakan penskalaan pelacakan target yang dikelola Amazon ECS.

6. Pilih Perbarui.

Menghapus penyedia kapasitas untuk cluster menggunakan konsol

Jika Anda selesai menggunakan penyedia kapasitas grup Auto Scaling, Anda dapat menghapusnya. Setelah grup dihapus, penyedia kapasitas grup Auto Scaling akan beralih ke status. `INACTIVE` Penyedia kapasitas dengan status `INACTIVE` mungkin tetap dapat ditemukan di akun Anda selama jangka waktu tertentu. Namun, perilaku ini dapat berubah di masa depan, jadi sebaiknya Anda tidak mengandalkan penyedia kapasitas `INACTIVE` yang tersimpan. Sebelum penyedia kapasitas grup Auto Scaling dihapus, penyedia kapasitas harus dihapus dari strategi penyedia kapasitas dari semua layanan. Anda dapat menggunakan `UpdateService` API atau alur kerja layanan pembaruan di konsol Amazon ECS untuk menghapus penyedia kapasitas dari strategi penyedia kapasitas layanan. Gunakan opsi force new deployment untuk memastikan bahwa tugas apa pun yang menggunakan kapasitas instans Amazon EC2 yang disediakan oleh penyedia kapasitas dialihkan untuk menggunakan kapasitas dari penyedia kapasitas yang tersisa.

Untuk menghapus penyedia kapasitas untuk cluster (konsol Amazon ECS)

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih cluster.
4. Pada halaman Cluster: ***name***, pilih Infrastructure, grup Auto Scaling, lalu pilih Delete.
5. Di kotak konfirmasi, masukkan hapus nama grup ***Auto Scaling***
6. Pilih Hapus.

Pembuatan kapasitas

Instans penampung Amazon ECS adalah instans Amazon EC2 yang menjalankan agen penampung Amazon ECS dan telah terdaftar ke cluster Amazon ECS. Saat Anda menjalankan tugas dengan Amazon ECS menggunakan tipe peluncuran EC2, tipe peluncuran eksternal, atau penyedia kapasitas grup Auto Scaling, tugas Anda akan ditempatkan pada instance container aktif Anda. Anda bertanggung jawab atas manajemen dan pemeliharaan instans kontainer.

Amazon ECS menyediakan AMI yang dioptimalkan Amazon ECS yang telah dikonfigurasi sebelumnya dengan persyaratan dan rekomendasi untuk menjalankan beban kerja penampung Anda. Sebaiknya gunakan Amazon ECS yang dioptimalkan Amazon Linux 2023 AMI untuk instans Amazon EC2 Anda kecuali aplikasi Anda memerlukan instans berbasis GPU Amazon EC2, sistem operasi tertentu, atau versi Docker yang belum tersedia di AMI tersebut. Untuk informasi tentang instans Amazon Linux 2 dan Amazon Linux 2023, lihat Membandingkan Amazon Linux 2 dan Amazon Linux 2023 di Panduan Pengguna Amazon Linux 2023.

Meskipun Anda dapat membuat AMI instans Amazon EC2 Anda sendiri yang memenuhi spesifikasi dasar yang diperlukan untuk menjalankan beban kerja kontainer Anda di Amazon ECS, AMI Amazon ECS yang dioptimalkan telah dikonfigurasi sebelumnya dan diuji di Amazon ECS oleh para insinyur. AWS Ini adalah cara paling sederhana bagi Anda untuk memulai dan menjalankan kontainer Anda AWS dengan cepat.

Anda dapat menjalankan Linux atau Windows pada instance container Anda. Anda dapat memilih jenis instans berdasarkan kebutuhan aplikasi Anda seperti arsitektur CPU, throughput jaringan, dan arsitektur. Sebagai aturan umum, CPU dan memori harus cukup besar untuk menampung setidaknya satu replika tugas yang ingin Anda jalankan. Anda dapat meluncurkan lebih banyak tugas pada saat yang sama dengan instance yang lebih besar. Anda dapat meningkatkan skala dengan cara yang lebih halus untuk menghemat biaya dengan instance yang lebih kecil. Anda tidak perlu memilih satu jenis instans Amazon EC2 yang sesuai dengan semua aplikasi di cluster Anda. Anda dapat membuat beberapa Grup Auto Scaling dengan berbagai jenis instans agar sesuai dengan semua aplikasi. Untuk informasi selengkapnya, lihat Instans Amazon EC2.

Untuk menentukan jenis instans yang dapat Anda gunakan, mulailah dengan menghilangkan jenis instans atau keluarga instans yang tidak memenuhi persyaratan spesifik aplikasi Anda. Misalnya, jika aplikasi Anda memerlukan GPU, Anda dapat mengecualikan jenis instance apa pun yang tidak memiliki GPU. Namun, Anda juga harus mempertimbangkan persyaratan lain juga. Misalnya, pertimbangkan arsitektur CPU, throughput jaringan, dan jika penyimpanan instance adalah persyaratan. Selanjutnya, periksa jumlah CPU dan memori yang disediakan oleh setiap jenis instance. Sebagai aturan umum, CPU dan memori harus cukup besar untuk menampung setidaknya satu replika tugas yang ingin Anda jalankan.

Anda dapat memilih dari jenis instance yang kompatibel dengan aplikasi Anda. Dengan instance yang lebih besar, Anda dapat meluncurkan lebih banyak tugas secara bersamaan. Dan, dengan contoh yang lebih kecil, Anda dapat meningkatkan skala dengan cara yang lebih halus untuk menghemat biaya. Anda tidak perlu memilih satu jenis instans Amazon EC2 yang sesuai dengan semua aplikasi di cluster Anda. Sebagai gantinya, Anda dapat membuat beberapa Grup Auto

Scaling,. Setiap grup dapat memiliki jenis instance yang berbeda. Kemudian, Anda dapat membuat Penyedia Kapasitas Amazon EC2 untuk masing-masing grup ini. Terakhir, dalam strategi Penyedia Kapasitas layanan dan tugas Anda, Anda dapat memilih Penyedia Kapasitas yang paling sesuai dengan kebutuhannya. Untuk informasi selengkapnya, lihat Instans Amazon EC2.

Gunakan panduan berikut untuk jaringan instans Anda:

- Sebaiknya luncurkan instans kontainer Anda di dalam VPC, karena Amazon VPC memberikan kontrol lebih besar atas jaringan Anda dan menawarkan kemampuan konfigurasi yang lebih luas. Untuk informasi selengkapnya, lihat Amazon EC2 dan Amazon Virtual Private Cloud di Panduan Pengguna Amazon EC2 untuk Instans Linux.
- Jika salah satu kontainer yang terkait dengan tugas Anda memerlukan konektivitas eksternal, Anda dapat memetakan port jaringannya ke port pada instans penampung Amazon ECS host sehingga dapat dijangkau dari internet. Grup keamanan instans kontainer Anda harus mengizinkan akses masuk ke port yang ingin Anda ekspos. Untuk informasi selengkapnya, lihat Membuat Grup Keamanan di Panduan Memulai Amazon VPC.
- Instans kontainer memerlukan akses untuk berkomunikasi dengan titik akhir layanan Amazon ECS. Ini dapat dilakukan melalui VPC endpoint antarmuka atau melalui instans kontainer yang memiliki alamat IP publik.

  Untuk informasi lebih lanjut tentang VPC endpoint antarmuka, lihat Titik akhir VPC antarmuka Amazon ECS ()AWS PrivateLink.

  Jika Anda tidak memiliki VPC endpoint yang dikonfigurasi dan instans kontainer Anda tidak memiliki alamat IP publik, network address translation (NAT) harus digunakan untuk menyediakan akses ini. Untuk informasi lebih lanjut, lihat Gateway NAT dalam Panduan Pengguna Amazon VPC dan Konfigurasi proxy HTTP untuk instance kontainer Linux dalam panduan ini. Untuk informasi selengkapnya, lihat the section called “Buat virtual private cloud”.

Instans Spot

Kapasitas spot dapat memberikan penghematan biaya yang signifikan dibandingkan instans sesuai permintaan. Kapasitas spot adalah kelebihan kapasitas yang harganya jauh lebih rendah daripada kapasitas sesuai permintaan atau cadangan. Kapasitas spot cocok untuk pemrosesan batch dan beban kerja pembelajaran mesin, serta lingkungan pengembangan dan pementasan. Secara lebih umum, ini cocok untuk semua beban kerja yang mentolerir downtime sementara.

Pahami bahwa konsekuensi berikut karena kapasitas Spot mungkin tidak tersedia sepanjang waktu.

- Selama periode permintaan yang sangat tinggi, kapasitas Spot mungkin tidak tersedia. Hal ini dapat menyebabkan peluncuran instans Amazon EC2 Spot tertunda. Dalam acara ini, layanan Amazon ECS mencoba lagi meluncurkan tugas, dan grup Auto Scaling Amazon EC2 juga mencoba meluncurkan instans lagi, hingga kapasitas yang diperlukan tersedia. Amazon EC2 tidak menggantikan kapasitas Spot dengan kapasitas sesuai permintaan.
- Ketika permintaan keseluruhan untuk kapasitas meningkat, instans dan tugas Spot dapat dihentikan hanya dengan peringatan dua menit. Setelah peringatan dikirim, tugas harus memulai shutdown yang teratur jika perlu sebelum instance dihentikan sepenuhnya. Ini membantu meminimalkan kemungkinan kesalahan. Untuk informasi lebih lanjut tentang shutdown yang anggun, lihat Penutupan anggun dengan ECS.

Untuk membantu meminimalkan kekurangan kapasitas Spot, pertimbangkan rekomendasi berikut:

- Gunakan beberapa Wilayah dan Availability Zone - Kapasitas spot bervariasi menurut Wilayah dan Availability Zone. Anda dapat meningkatkan ketersediaan Spot dengan menjalankan beban kerja di beberapa Wilayah dan Availability Zone. Jika memungkinkan, tentukan subnet di semua Availability Zone di Wilayah tempat Anda menjalankan tugas dan instance.
- Gunakan beberapa jenis instans Amazon EC2 - Saat Anda menggunakan Kebijakan Instans Campuran dengan Auto Scaling Amazon EC2, beberapa jenis instans akan diluncurkan ke Grup Auto Scaling. Ini memastikan bahwa permintaan kapasitas Spot dapat dipenuhi saat dibutuhkan. Untuk memaksimalkan keandalan dan meminimalkan kompleksitas, gunakan tipe instans dengan jumlah CPU dan memori yang kira-kira sama dalam Kebijakan Instans Campuran Anda. Instance ini dapat berasal dari generasi yang berbeda, atau varian dari tipe instance dasar yang sama. Perhatikan bahwa mereka mungkin datang dengan fitur tambahan yang mungkin tidak Anda perlukan. Contoh daftar semacam itu dapat mencakup m4.large, m5.large, m5a.large, m5d.large, m5n.large, m5dn.large, dan m5ad.large. Untuk informasi selengkapnya, lihat Grup Auto Scaling dengan beberapa tipe instans dan opsi pembelian dalam Panduan Pengguna Amazon EC2 Auto Scaling.
- Gunakan strategi alokasi Spot yang dioptimalkan kapasitas - Dengan Amazon EC2 Spot, Anda dapat memilih antara strategi alokasi kapasitas dan biaya yang dioptimalkan. Jika Anda memilih strategi yang dioptimalkan kapasitas saat meluncurkan instans baru, Amazon EC2 Spot memilih jenis instans dengan ketersediaan terbesar di Availability Zone yang dipilih. Ini membantu mengurangi kemungkinan bahwa instance dihentikan segera setelah diluncurkan.

Pengurasan Instans Linux Spot

Amazon EC2 mengakhiri, menghentikan, atau hibernasi Instans Spot Anda ketika harga Spot melebihi harga maksimum untuk permintaan atau kapasitas Anda tidak lagi tersedia. Amazon EC2 menyediakan pemberitahuan interupsi dua menit Instans Spot untuk menghentikan dan menghentikan tindakan. Itu tidak memberikan pemberitahuan dua menit untuk tindakan hibernasi. Jika pengurasan Instans Spot Amazon ECS diaktifkan pada instans, ECS menerima pemberitahuan gangguan Instans Spot dan menempatkan instance dalam status. `DRAINING`

> **Important**
>
> Amazon ECS tidak menerima pemberitahuan dari Amazon EC2 saat instans dihapus oleh Auto Scaling Capacity Rebalancing. Untuk informasi selengkapnya, lihat Penyeimbangan Kembali Kapasitas Auto Scaling Amazon EC2.

Saat instance container disetel ke`DRAINING`, Amazon ECS mencegah tugas baru dijadwalkan untuk penempatan pada instance container. Tugas layanan pada instans kontainer pengurasan yang ada di status `PENDING` segera dihentikan. Jika ada instans kontainer di klaster yang tersedia, tugas layanan pengganti dimulai.

Pengurasan Instans Spot dimatikan secara default dan harus diaktifkan secara manual. Untuk mengaktifkan pengurasan Instance Spot untuk instance kontainer baru, saat meluncurkan instance penampung, tambahkan skrip berikut ke dalam bidang data Pengguna, ganti *`MyCluster`*dengan nama cluster untuk mendaftarkan instance kontainer.

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_ENABLE_SPOT_INSTANCE_DRAINING=true
EOF
```

Untuk informasi selengkapnya, lihat Meluncurkan instans penampung Amazon ECS Linux.

Untuk mengaktifkan pengurasan Instance Spot untuk instance kontainer yang ada

1. Connect ke Instans Spot melalui SSH.
2. Edit file `/etc/ecs/ecs.config` dan tambahkan berikut:

```
ECS_ENABLE_SPOT_INSTANCE_DRAINING=true
```

3. Mulai ulang layanan `ecs`.

   - Untuk Amazon ECS yang dioptimalkan Amazon Linux 2 AMI:

     ```
     sudo systemctl restart ecs
     ```

   - Untuk Amazon ECS yang dioptimalkan Amazon Linux AMI:

     ```
     sudo stop ecs && sudo start ecs
     ```

4. (Opsional) Anda dapat memastikan bahwa agen sedang berjalan dan melihat beberapa informasi tentang instans kontainer baru dengan melakukan kueri pada operasi API introspeksi agen. Untuk informasi selengkapnya, lihat the section called “Introspeksi wadah”.

   ```
   curl http://localhost:51678/v1/metadata
   ```

Pengurasan Instans Spot Windows

Amazon EC2 mengakhiri, menghentikan, atau hibernasi Instans Spot Anda ketika harga Spot melebihi harga maksimum untuk permintaan atau kapasitas Anda tidak lagi tersedia. Amazon EC2 memberikan pemberitahuan interupsi Instans Spot, yang memberikan peringatan dua menit pada instans sebelum diinterupsi. Jika pengurasan Instans Spot Amazon ECS diaktifkan pada instans, ECS menerima pemberitahuan gangguan Instans Spot dan menempatkan instance dalam status. `DRAINING`

> **Important**
>
> Amazon ECS memantau pemberitahuan gangguan Instans Spot yang memiliki dan tindakan instans. `terminate` `stop` Jika Anda menentukan perilaku interupsi `hibernate` instans saat meminta Instans Spot atau Armada Spot, maka pengurasan Instans Spot Amazon ECS tidak didukung untuk instans tersebut.

Saat instance container disetel ke`DRAINING`, Amazon ECS mencegah tugas baru dijadwalkan untuk penempatan pada instance container. Tugas layanan pada instans kontainer pengurasan yang ada di

status PENDING segera dihentikan. Jika ada instans kontainer di klaster yang tersedia, tugas layanan pengganti dimulai.

Anda harus mengatur ECS_ENABLE_SPOT_INSTANCE_DRAINING parameter sebelum memulai agen kontainer. Gunakan perintah berikut untuk mengaktifkan pengurasan Instance Spot secara manual. Gantikan *my-cluster* dengan nama cluster Anda.

```
[Environment]::SetEnvironmentVariable("ECS_ENABLE_SPOT_INSTANCE_DRAINING", "true",
 "Machine")

# Initialize the agent
Initialize-ECSAgent -Cluster my-cluster
```

Untuk informasi selengkapnya, lihat the section called “Meluncurkan instans kontainer”.

Instans Linux

Anda dapat menggunakan instans EC2 dengan sistem operasi Linux berikut untuk menjalankan aplikasi Anda:

- Amazon Linux: Ini adalah sistem operasi tujuan umum.
- Bottlerocket adalah sistem operasi open-source berbasis Linux yang dibuat khusus AWS untuk menjalankan kontainer pada mesin virtual atau host bare metal. AMI Bottlerocket yang dioptimalkan Amazon ECS aman dan hanya mencakup jumlah minimum paket yang diperlukan untuk menjalankan kontainer. Ini meningkatkan penggunaan sumber daya, mengurangi permukaan serangan keamanan, dan membantu menurunkan overhead manajemen. Untuk informasi tentang fitur dan panduan keamanan, lihat Fitur Keamanan dan Panduan Keamanan di GitHub situs web.

Amazon ECS menyediakan AMI yang dioptimalkan Amazon ECS yang telah dikonfigurasi sebelumnya dengan persyaratan dan rekomendasi untuk menjalankan beban kerja penampung Anda. Sebaiknya gunakan Amazon ECS yang dioptimalkan Amazon Linux 2023 AMI untuk instans Amazon EC2 Anda kecuali aplikasi Anda memerlukan instans berbasis GPU Amazon EC2, sistem operasi tertentu, atau versi Docker yang belum tersedia di AMI tersebut. Untuk informasi tentang instans Amazon Linux 2 dan Amazon Linux 2023, lihat Membandingkan Amazon Linux 2 dan Amazon Linux 2023 di Panduan Pengguna Amazon Linux 2023.

Meskipun Anda dapat membuat AMI instans Amazon EC2 Anda sendiri yang memenuhi spesifikasi dasar yang diperlukan untuk menjalankan beban kerja kontainer Anda di Amazon ECS, AMI Amazon ECS yang dioptimalkan telah dikonfigurasi sebelumnya dan diuji di Amazon ECS oleh para insinyur.

AWS Ini adalah cara paling sederhana bagi Anda untuk memulai dan menjalankan kontainer Anda AWS dengan cepat.

Spesifikasi instans kontainer Amazon ECS terdiri dari komponen-komponen berikut:

Diperlukan

- Distribusi Linux yang menjalankan setidaknya versi 3.10 dari kernel Linux.
- Agen kontainer Amazon ECS (lebih disukai versi terbaru). Untuk informasi selengkapnya, lihat Memperbarui agen kontainer Amazon ECS.
- Daemon Docker yang menjalankan setidaknya versi 1.9.0, dan dependensi runtime Docker apa pun. Untuk informasi lebih lanjut, lihat Memeriksa dependensi waktu aktif dalam dokumentasi Docker.

> **Note**
>
> Untuk pengalaman terbaik, kami merekomendasikan versi Docker yang dikirimkan dan diuji dengan versi agen kontainer Amazon ECS yang sesuai yang Anda gunakan.

Direkomendasikan

- Proses inisialisasi dan pengasuh untuk menjalankan dan memantau agen kontainer Amazon ECS. AMI Amazon ECS yang dioptimalkan menggunakan `ecs-init` RPM untuk mengelola agen. Untuk informasi lebih lanjut, lihat `ecs-init`proyek di GitHub.

Topik

- AMI Amazon ECS yang dioptimalkan
- AMI Amazon ECS Bottlerocket yang dioptimalkan
- Memasang agen kontainer Amazon ECS

AMI Amazon ECS yang dioptimalkan

Amazon ECS menyediakan AMI yang dioptimalkan Amazon ECS yang telah dikonfigurasi sebelumnya dengan persyaratan dan rekomendasi untuk menjalankan beban kerja penampung Anda. Sebaiknya gunakan Amazon ECS yang dioptimalkan Amazon Linux 2023 AMI untuk instans Amazon EC2 Anda kecuali aplikasi Anda memerlukan instans berbasis GPU Amazon EC2, sistem

operasi tertentu, atau versi Docker yang belum tersedia di AMI tersebut. Untuk informasi tentang instans Amazon Linux 2 dan Amazon Linux 2023, lihat Membandingkan Amazon Linux 2 dan Amazon Linux 2023 di Panduan Pengguna Amazon Linux 2023. Meluncurkan instans kontainer Anda dari AMI terbaru yang dioptimalkan Amazon ECS memastikan bahwa Anda menerima pembaruan keamanan terkini dan versi agen penampung. Untuk informasi tentang cara meluncurkan instance, lihatMeluncurkan instans penampung Amazon ECS Linux.

Varian Linux dari AMI Amazon ECS yang dioptimalkan menggunakan Amazon Linux 2 AMI sebagai basis mereka. Catatan rilis Amazon Linux 2 AMI juga tersedia. Untuk informasi selengkapnya, lihat catatan rilis Amazon Linux 2.

Kami menyarankan Anda menggunakan AMI dengan kernel Linux 5.10 karena kernel Linux 4.14 mencapai end-of-life pada 10 Januari. 2024.

Varian berikut dari AMI Amazon ECS yang dioptimalkan tersedia untuk instans Amazon EC2 Anda.

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
Amazon Linux 2023	Amazon ECS yang dioptimalkan Amazon Linux 2023 AMI	Amazon Linux 2023 adalah generasi berikutnya dari Amazon Linux dari AWS. Untuk sebagian besar kasus, disarankan untuk meluncurkan instans Amazon EC2 untuk beban kerja Amazon ECS Anda. Untuk informasi selengkapnya, lihat Apa itu Amazon Linux 2023 di Panduan Pengguna Amazon Linux 2023.	Secara default, Amazon ECS Amazon Linux 2023 AMI yang dioptimalkan Amazon dikirimkan dengan volume root 30-GiB tunggal. Anda dapat mengubah ukuran volume root 30 GiB pada waktu peluncuran untuk meningkatkan penyimpanan yang tersedia pada instans kontainer Anda. Penyimpanan ini digunakan untuk sistem operasi dan untuk citra Docker serta metadata.

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
Amazon Linux 2023 (arm64)	Amazon ECS yang dioptimalkan Amazon Linux 2023 (arm64) AMI	Berdasarkan Amazon Linux 2023, AMI ini direkomendasikan untuk digunakan saat meluncurkan instans Amazon EC2 Anda, yang didukung oleh Prosesor Graviton/ Graviton 2 AWS berbasis ARM, untuk beban kerja Amazon ECS Anda. Untuk informasi selengkap nya, lihat Instans Tujuan Umum di Panduan Pengguna Amazon EC2 untuk Instans Linux. Amazon ECS yang dioptimalkan Amazon Linux 2023 (arm64) AMI tidak disertakan dengan yang sudah diinstal sebelumnya. AWS CLI	Sistem file default untuk Amazon ECS yang dioptimalkan Amazon Linux 2023 AMI adalah`xfs`, dan Docker menggunak an driver penyimpan an. `overlay2` Untuk informasi selengkap nya, lihat Gunakan driver penyimpanan OverlayFS dalam dokumentasi Docker.

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
Amazon Linux 2023 (Neuron)	Amazon ECS mengoptimalkan Amazon Linux 2023 (Neuron) AMI	Berdasarkan Amazon Linux 2023, AMI ini direkomendasikan untuk digunakan saat meluncurkan instans Amazon EC2 Inf1 Anda. Muncul pra-konfi gurasi dengan driver AWS Inferentia dan runtime AWS Neuron untuk Docker yang membuat menjalank an beban kerja inferensi pembelaja ran mesin lebih mudah di Amazon ECS. Untuk informasi selengkapnya, lihat Menggunakan AWS Neuron di Amazon Linux 2 di Amazon ECS. Amazon ECS yang dioptimalkan Amazon Linux 2023 (Neuron) AMI tidak disertakan dengan yang sudah diinstal sebelumnya. AWS CLI	

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
Amazon Linux 2	Amazon ECS yang dioptimalkan Amazon Linux 2 kernel 5.10 AMI	Berdasarkan Amazon Linux 2, AMI ini digunakan saat meluncurkan instans Amazon EC2 Anda dan Anda ingin menggunakan kernel Linux 5.10 alih-alih kernel 4.14 untuk beban kerja Amazon ECS Anda. Kernel 5.10 AMI Amazon Linux 2 yang dioptimalkan Amazon ECS tidak disertakan dengan AWS CLI yang sudah diinstal sebelumnya.	Secara default, AMI yang dioptimalkan Amazon ECS berbasis Amazon Linux 2 (Amazon ECS yang dioptimalkan Amazon Linux 2 AMI, Amazon ECS Amazon Linux 2 (arm64) AMI yang dioptimalkan Amazon ECS, dan AMI yang dioptimalkan untuk GPU Amazon ECS) dikirimkan dengan satu volume root 30-GiB. Anda dapat mengubah ukuran volume root 30 GiB pada waktu peluncuran untuk meningkatkan penyimpanan yang tersedia pada instans kontainer Anda. Penyimpanan ini digunakan untuk sistem operasi dan untuk citra Docker serta metadata. Sistem file default untuk Amazon ECS yang dioptimalkan Amazon Linux 2
	Amazon ECS yang dioptimalkan Amazon Linux 2 AMI	Ini untuk beban kerja Amazon ECS Anda. Amazon ECS yang dioptimalkan Amazon Linux 2 AMI tidak disertakan dengan yang sudah diinstal sebelumnya. AWS CLI	
Amazon Linux 2 (arm64)	Amazon ECS yang dioptimalkan Amazon Linux 2 kernel 5.10 (arm64) AMI	Berdasarkan Amazon Linux 2, AMI ini untuk instans Amazon EC2 Anda, yang didukung oleh Prosesor AWS	

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
		Graviton/Graviton 2 berbasis ARM, dan Anda ingin menggunakan kernel Linux 5.10 alih-alih kernel Linux 4.14 untuk beban kerja Amazon ECS Anda. Untuk informasi selengkapnya, lihat Instans Tujuan Umum di Panduan Pengguna Amazon EC2 untuk Instans Linux. Amazon ECS yang dioptimalkan Amazon Linux 2 (arm64) AMI tidak disertakan dengan yang sudah diinstal sebelumnya. AWS CLI	AMI adalah `xfs`, dan Docker menggunakan driver penyimpanan. `overlay2` Untuk informasi selengkapnya, lihat Gunakan driver penyimpanan OverlayFS dalam dokumentasi Docker.

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
	Amazon ECS yang dioptimalkan Amazon Linux 2 (arm64) AMI	Berdasarkan Amazon Linux 2, AMI ini digunakan saat meluncurkan instans Amazon EC2 Anda, yang didukung oleh Prosesor AWS Graviton/Graviton 2 berbasis ARM, untuk beban kerja Amazon ECS Anda. Amazon ECS yang dioptimalkan Amazon Linux 2 (arm64) AMI tidak disertakan dengan yang sudah diinstal sebelumnya. AWS CLI	

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
Amazon Linux 2 (GPU)	Kernel Amazon ECS GPU yang dioptimal kan 5.10 AMI	Berdasarkan Amazon Linux 2, AMI ini direkomendasikan untuk digunakan saat meluncurkan instans berbasis GPU Amazon EC2 Anda dengan kernel Linux 5.10 untuk beban kerja Amazon ECS Anda. Muncul pra-konfigurasi dengan driver kernel NVIDIA dan runtime GPU Docker yang membuat beban kerja berjalan yang memanfaatkan GPU di Amazon ECS. Untuk informasi selengkapnya, lihat Bekerja dengan GPU di Amazon ECS.	

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
	AMI Amazon ECS GPU yang dioptimalkan	Berdasarkan Amazon Linux 2, AMI ini direkomendasikan untuk digunakan saat meluncurkan instans berbasis GPU Amazon EC2 Anda dengan kernel Linux 4.14 untuk beban kerja Amazon ECS Anda. Muncul pra-konfigurasi dengan driver kernel NVIDIA dan runtime GPU Docker yang membuat beban kerja berjalan yang memanfaatkan GPU di Amazon ECS. Untuk informasi selengkapnya, lihat Bekerja dengan GPU di Amazon ECS.	

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
Amazon Linux 2 (Neuron)	Amazon ECS mengoptimalkan kernel Amazon Linux 2 (Neuron) 5.10 AMI	Berdasarkan Amazon Linux 2, AMI ini untuk instans Amazon EC2 Inf1, Trn1 atau Inf2. Muncul pra-konfigurasi dengan AWS Inferentia dengan kernel Linux 5.10 dan driver AWS Trainium dan runtime AWS Neuron untuk Docker yang membuat menjalankan beban kerja inferensi pembelajaran mesin lebih mudah di Amazon ECS. Untuk informasi selengkapnya, lihat Menggunakan AWS Neuron di Amazon Linux 2 di Amazon ECS. Amazon ECS yang dioptimalkan Amazon Linux 2 (Neuron) AMI tidak disertakan dengan yang AWS CLI sudah diinstal sebelumnya.	

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
	Amazon ECS dioptimalkan Amazon Linux 2 (Neuron) AMI	Berdasarkan Amazon Linux 2, AMI ini untuk instans Amazon EC2 Inf1, Trn1 atau Inf2. Muncul pra-konfi gurasi dengan driver AWS Inferentia dan AWS Trainium dan runtime AWS Neuron untuk Docker yang membuat menjalank an beban kerja inferensi pembelaja ran mesin lebih mudah di Amazon ECS. Untuk informasi selengkapnya, lihat Menggunakan AWS Neuron di Amazon Linux 2 di Amazon ECS. Amazon ECS yang dioptimalkan Amazon Linux 2 (Neuron) AMI tidak disertakan dengan yang AWS CLI sudah diinstal sebelumnya.	

Amazon ECS menyediakan changelog untuk varian Linux dari AMI yang dioptimalkan Amazon ECS. GitHub Untuk informasi lebih lanjut, lihat Changelog.

Varian Linux dari AMI Amazon ECS yang dioptimalkan menggunakan Amazon Linux 2 AMI atau Amazon Linux 2023 AMI sebagai basis mereka. Anda dapat mengambil nama AMI sumber Amazon

Linux 2 atau nama AMI Amazon Linux 2023 untuk setiap varian dengan menanyakan Systems Manager Parameter Store API. Untuk informasi selengkapnya, lihat Mengambil metadata AMI Amazon ECS yang dioptimalkan. Catatan rilis Amazon Linux 2 AMI juga tersedia. Untuk informasi selengkapnya, lihat catatan rilis Amazon Linux 2. Catatan rilis Amazon Linux 2023 juga tersedia. Untuk informasi lebih lanjut lihat, catatan rilis Amazon Linux 2023.

Halaman-halaman berikut memberikan informasi tambahan tentang perubahan:

- Sumber catatan rilis AMI pada GitHub
- Catatan rilis Docker Engine dalam dokumentasi Docker
- Dokumentasi Driver NVIDIA dalam dokumentasi NVIDIA
- Amazon ECS Agent changelog aktif GitHub

  Kode sumber untuk `ecs-init` aplikasi dan skrip dan konfigurasi untuk pengemasan agen sekarang menjadi bagian dari repositori agen. Untuk versi `ecs-init` dan kemasan yang lebih lama, lihat Amazon ecs-init changelog di GitHub

Mengambil metadata AMI Amazon ECS yang dioptimalkan

Anda dapat mengambil metadata AMI Amazon ECS yang dioptimalkan secara terprogram. Metadata mencakup nama AMI, versi agen penampung Amazon ECS, dan versi runtime Amazon ECS yang mencakup versi Docker.

ID AMI, nama gambar, sistem operasi, versi agen kontainer, nama gambar sumber, dan versi runtime untuk setiap varian AMI yang dioptimalkan Amazon ECS dapat diambil secara terprogram dengan menanyakan Systems Manager Parameter Store API. Untuk informasi selengkapnya tentang Systems Manager Parameter Store API, lihat GetParametersdan GetParametersByPath.

> **Note**
>
> Pengguna administratif Anda harus memiliki izin IAM berikut untuk mengambil metadata AMI Amazon ECS yang dioptimalkan. Izin ini telah ditambahkan ke kebijakan `AmazonECS_FullAccess` IAM.
>
> - ssm: GetParameters
> - ssm: GetParameter
> - ssm: GetParametersByPath

Systems Manager Parameter Menyimpan format parameter

Berikut ini adalah format nama parameter untuk setiap varian AMI Amazon ECS yang dioptimalkan.

AMI yang dioptimalkan untuk Linux Amazon ECS

- Metadata AMI Amazon Linux 2023:

```
/aws/service/ecs/optimized-ami/amazon-linux-2023/<version>
```

- Amazon Linux 2023 (arm64) metadata AMI:

```
/aws/service/ecs/optimized-ami/amazon-linux-2023/arm64/<version>
```

- Amazon Linux 2023 (Neuron) metadata AMI:

```
/aws/service/ecs/optimized-ami/amazon-linux-2023/inf/<version>
```

- Amazon Linux 2 AMI metadata:

```
/aws/service/ecs/optimized-ami/amazon-linux-2/<version>
```

- Amazon Linux 2 kernel 5.10 metadata AMI:

```
/aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/<version>
```

- Amazon Linux 2 (arm64) AMI metadata:

```
/aws/service/ecs/optimized-ami/amazon-linux-2/arm64/<version>
```

- Amazon Linux 2 kernel 5.10 (arm64) metadata AMI:

```
/aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/arm64/<version>
```

- Metadata kernel 5.10 AMI yang dioptimalkan oleh Amazon ECS GPU:

```
/aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/gpu/<version>
```

- Amazon Linux 2 (GPU) AMI metadata:

```
/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/<version>
```

- Amazon ECS mengoptimalkan metadata kernel Amazon Linux 2 (Neuron) 5.10 AMI:

```
/aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/inf/<version>
```

- Amazon Linux 2 (Neuron) metadata AMI:

```
/aws/service/ecs/optimized-ami/amazon-linux-2/inf/<version>
```

Format nama parameter berikut mengambil ID gambar Amazon ECS stabil terbaru Amazon Linux 2 AMI yang dioptimalkan dengan menggunakan sub-parameter. `image_id`

```
/aws/service/ecs/optimized-ami/amazon-linux-2/recommended/image_id
```

Format nama parameter berikut mengambil metadata versi AMI tertentu yang dioptimalkan Amazon ECS dengan menentukan nama AMI.

- Metadata Amazon Linux 2 AMI Amazon ECS yang dioptimalkan oleh Amazon ECS:

```
/aws/service/ecs/optimized-ami/amazon-linux-2/amzn2-ami-ecs-hvm-2.0.20181112-x86_64-ebs
```

> **Note**
>
> Semua versi Amazon ECS yang dioptimalkan Amazon Linux 2 AMI tersedia untuk pengambilan. Hanya versi AMI Amazon ECS yang dioptimalkan `amzn-ami-2017.09.l-amazon-ecs-optimized` (Linux) dan yang lebih baru yang dapat diambil.

Contoh-contoh

Contoh berikut menunjukkan cara Anda dapat mengambil metadata untuk setiap varian AMI Amazon ECS yang dioptimalkan.

Mengambil metadata AMI terbaru yang dioptimalkan Amazon ECS stabil

Anda dapat mengambil AMI stabil Amazon ECS terbaru yang dioptimalkan menggunakan AWS CLI perintah berikut. AWS CLI

AMI yang dioptimalkan untuk Linux Amazon ECS

- Untuk Amazon ECS Amazon Linux 2023 AMI yang dioptimalkan oleh Amazon ECS:

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/
recommended --region us-east-1
```

- Untuk Amazon ECS Amazon Linux 2023 (arm64) AMI yang dioptimalkan oleh Amazon ECS:

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/
arm64/recommended --region us-east-1
```

- Untuk Amazon ECS yang dioptimalkan Amazon Linux 2 kernel 5.10 AMI:

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/recommended --region us-east-1
```

- Untuk Amazon ECS Amazon Linux 2 AMI yang dioptimalkan oleh Amazon:

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
recommended --region us-east-1
```

- Untuk Amazon ECS yang dioptimalkan Amazon Linux 2 kernel 5.10 (arm64) AMI:

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/arm64/recommended --region us-east-1
```

- Untuk Amazon ECS Amazon Linux 2 (arm64) AMI yang dioptimalkan oleh Amazon ECS:

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/arm64/
recommended --region us-east-1
```

- Untuk kernel 5.10 AMI Amazon ECS yang dioptimalkan oleh GPU:

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/gpu/recommended --region us-east-1
```

- Untuk AMI yang dioptimalkan oleh GPU Amazon ECS:

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/gpu/
recommended --region us-east-1
```

- Untuk Amazon ECS yang dioptimalkan Amazon Linux 2 (Neuron) kernel 5.10 AMI:

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/inf/recommended --region us-east-1
```

- Untuk Amazon ECS yang dioptimalkan Amazon Linux 2 (Neuron) AMI:

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/inf/
recommended --region us-east-1
```

Mengambil ID gambar dari Amazon ECS terbaru yang dioptimalkan Amazon ECS 2023 AMI yang dioptimalkan

Anda dapat mengambil ID gambar dari ID AMI Amazon Linux 2023 Amazon ECS terbaru yang dioptimalkan oleh Amazon ECS dengan menggunakan sub-parameter. `image_id`

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-
linux-2023/recommended/image_id --region us-east-1
```

Untuk mengambil nilai `image_id` saja, Anda dapat meng-kueri nilai parameter tertentu; misalnya:

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/
recommended/image_id --region us-east-1 --query "Parameters[0].Value"
```

Mengambil metadata versi Amazon Linux 2 AMI Amazon ECS tertentu yang dioptimalkan

Ambil metadata versi Amazon Linux Linux AMI yang dioptimalkan Amazon ECS tertentu menggunakan perintah berikut AWS CLI . AWS CLI Ganti nama AMI dengan nama Amazon ECS yang dioptimalkan Amazon Linux AMI untuk diambil.

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/amzn2-ami-
ecs-hvm-2.0.20200928-x86_64-ebs --region us-east-1
```

Mengambil metadata AMI Amazon Linux 2 kernel 5.10 AMI Amazon ECS yang dioptimalkan oleh Amazon ECS menggunakan Systems Manager API GetParametersByPath

Ambil metadata Amazon Linux 2 AMI Amazon ECS yang dioptimalkan oleh Amazon ECS dengan Systems GetParametersByPath Manager API menggunakan perintah AWS CLI berikut.

```
aws ssm get-parameters-by-path --path /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/ --region us-east-1
```

Mengambil ID gambar dari kernel Amazon Linux 2 5.10 AMI yang dioptimalkan Amazon ECS terbaru yang direkomendasikan

Anda dapat mengambil ID gambar dari ID AMI 5.10 kernel 5.10 Amazon ECS terbaru yang dioptimalkan Amazon ECS yang dioptimalkan dengan menggunakan sub-parameter. `image_id`

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/recommended/image_id --region us-east-1
```

Untuk mengambil nilai `image_id` saja, Anda dapat meng-kueri nilai parameter tertentu; misalnya:

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
recommended/image_id --region us-east-1 --query "Parameters[0].Value"
```

Menggunakan AMI terbaru yang dioptimalkan Amazon ECS yang dioptimalkan dalam template AWS CloudFormation

Anda dapat mereferensikan AMI terbaru yang dioptimalkan Amazon ECS yang dioptimalkan dalam AWS CloudFormation template dengan mereferensikan nama penyimpanan parameter Systems Manager.

Contoh Linux

```
Parameters:kernel-5.10
  LatestECSOptimizedAMI:
    Description: AMI ID
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ecs/optimized-ami/amazon-linux-2/recommended/image_id
```

Skrip pembuatan AMI Linux AMI yang dioptimalkan Amazon ECS

Amazon ECS telah membuka skrip build yang digunakan untuk membangun varian Linux dari AMI Amazon ECS yang dioptimalkan. Skrip build ini sekarang tersedia di GitHub. Untuk informasi lebih lanjut, lihat amazon-ecs-amidi GitHub.

Repositori skrip build menyertakan template HashiCorppengemas dan skrip build untuk menghasilkan setiap varian Linux dari AMI Amazon ECS yang dioptimalkan. Skrip ini adalah sumber

kebenaran untuk build AMI yang dioptimalkan Amazon ECS, sehingga Anda dapat mengikuti repositori untuk memantau GitHub perubahan pada AMI kami. Misalnya, mungkin Anda ingin AMI Anda sendiri menggunakan versi Docker yang sama dengan yang digunakan tim Amazon ECS untuk AMI resmi.

Untuk informasi selengkapnya, lihat repositori Amazon ECS AMI di aws/ on. amazon-ecs-ami GitHub

Untuk membangun AMI Linux Amazon ECS yang dioptimalkan

1. Kloning `aws/amazon-ecs-ami` GitHub repo.

   ```
   git clone https://github.com/aws/amazon-ecs-ami.git
   ```

2. Tambahkan variabel lingkungan untuk AWS Wilayah yang akan digunakan saat membuat AMI. Ganti `us-west-2` nilainya dengan Region yang akan digunakan.

   ```
   export REGION=us-west-2
   ```

3. Makefile disediakan untuk membangun AMI. Dari direktori root repositori kloning, gunakan salah satu perintah berikut, sesuai dengan varian Linux dari AMI Amazon ECS yang dioptimalkan yang ingin Anda buat.

   - Amazon ECS yang dioptimalkan Amazon Linux 2 AMI

     ```
     make al2
     ```

   - Amazon ECS yang dioptimalkan Amazon Linux 2 (arm64) AMI

     ```
     make al2arm
     ```

   - AMI Amazon ECS GPU yang dioptimalkan

     ```
     make al2gpu
     ```

   - Amazon ECS dioptimalkan Amazon Linux 2 (Neuron) AMI

     ```
     make al2inf
     ```

   - Amazon ECS yang dioptimalkan Amazon Linux 2023 AMI

     ```
     make al2023
     ```

- Amazon ECS yang dioptimalkan Amazon Linux 2023 (arm64) AMI

```
make al2023arm
```

- Amazon ECS mengoptimalkan Amazon Linux 2023 (Neuron) AMI

```
make al2023neu
```

AMI Amazon ECS Bottlerocket yang dioptimalkan

Bottlerocketadalah sistem operasi open-source Linux berbasis yang dibangun AWS untuk menjalankan kontainer pada mesin virtual atau host bare metal. BottlerocketAMI yang dioptimalkan Amazon ECS aman dan hanya menyertakan jumlah minimum paket yang diperlukan untuk menjalankan kontainer. Ini meningkatkan penggunaan sumber daya, mengurangi permukaan serangan keamanan, dan membantu menurunkan overhead manajemen. BottlerocketAMI juga terintegrasi dengan Amazon ECS untuk membantu mengurangi overhead operasional yang terlibat dalam memperbarui instance kontainer dalam sebuah cluster.

Bottlerocketberbeda dari Amazon Linux dengan cara berikut:

- Bottlerockettidak termasuk manajer paket, dan perangkat lunaknya hanya dapat dijalankan sebagai wadah. Pembaruan Bottlerocket keduanya diterapkan dan dapat diputar kembali dalam satu langkah, yang mengurangi kemungkinan kesalahan pembaruan.
- Mekanisme utama untuk mengelola Bottlerocket host adalah dengan penjadwal kontainer. Tidak seperti Amazon Linux, masuk ke Bottlerocket instance individual dimaksudkan untuk menjadi operasi yang jarang terjadi hanya untuk tujuan debugging dan pemecahan masalah tingkat lanjut.

Untuk informasi selengkapnyaBottlerocket, lihat dokumentasi dan rilis diGitHub.

Ada varian Bottlerocket AMI Amazon ECS yang dioptimalkan untuk kernel 6.1 dan kernel 5.10.

Varian berikut menggunakan kernel 6.1:

- `aws-ecs-2`
- `aws-ecs-2-nvidia`

Varian berikut menggunakan kernel 5.1.10:

- `aws-ecs-1`
- `aws-ecs-1-nvidia`

  Untuk informasi selengkapnya tentang `aws-ecs-1-nvidia` varian, lihat Mengumumkan dukungan GPU NVIDIA untuk Bottlerocket Amazon ECS.

Pertimbangan

Pertimbangkan hal berikut saat menggunakan Bottlerocket AMI dengan Amazon ECS.

- Bottlerocketmendukung instans Amazon EC2 dengan `x86_64` dan prosesor. `arm64` BottlerocketAMI tidak direkomendasikan untuk digunakan dengan instans Amazon EC2 dengan chip Inferentia.
- Bottlerocketgambar tidak menyertakan server SSH atau shell. Namun, Anda dapat menggunakan alat out-of-band manajemen untuk mendapatkan akses administrator SSH dan melakukan bootstrap. Untuk informasi lebih lanjut, lihat bagian ini di bottlerocket README.md di: GitHub
  - Eksplorasi
  - Kontainer admin
- Secara default, Bottlerocket memiliki wadah kontrol yang diaktifkan. Container ini menjalankan AWS Systems Manager agen yang dapat Anda gunakan untuk menjalankan perintah atau memulai sesi shell di instans Amazon EC2Bottlerocket. Untuk informasi selengkapnya, lihat Menyiapkan Pengelola Sesi di Panduan AWS Systems Manager Pengguna.

- Bottlerocket dioptimalkan untuk beban kerja kontainer dan memiliki fokus pada keamanan. Bottlerocket tidak menyertakan manajer paket dan tidak dapat diubah. Untuk informasi tentang fitur dan panduan keamanan, lihat Fitur Keamanan dan Panduan Keamanan di GitHub.
- Mode `awsvpc` jaringan didukung untuk versi AMI `1.1.0` Bottlerocket atau yang lebih baru.
- App Mesh dalam definisi tugas didukung untuk versi AMI `1.15.0` Bottlerocket atau yang lebih baru.
- Parameter definisi `initProcessEnabled` tugas didukung untuk versi AMI `1.19.0` Bottlerocket atau yang lebih baru.
- BottlerocketAMI juga tidak mendukung layanan dan fitur berikut:
  - ECS Di Mana Saja
  - Layanan Connect

- Amazon EFS dalam mode terenkripsi dan mode jaringan `awsvpc`
- Akselerator Elastic Inference

Mengambil AMI Amazon ECS Bottlerocket yang dioptimalkan

Anda dapat mengambil ID Amazon Machine Image (AMI) untuk AMI yang dioptimalkan Amazon ECS dengan menanyakan API Parameter Store. AWS Systems Manager Dengan menggunakan parameter ini, Anda tidak perlu mencari ID AMI yang dioptimalkan Amazon ECS secara manual. Untuk informasi selengkapnya tentang Systems Manager Parameter Store API, lihat GetParameter. Pengguna yang Anda gunakan harus memiliki izin `ssm:GetParameter` IAM untuk mengambil metadata AMI Amazon ECS yang dioptimalkan.

Mengambil varian AMI **`aws-ecs-2`** Bottlerocket

Anda dapat mengambil varian AMI `aws-ecs-2` Bottlerocket stabil terbaru Wilayah AWS dengan dan arsitektur dengan atau. AWS CLI AWS Management Console

- AWS CLI— Anda dapat mengambil ID gambar dari Bottlerocket AMI terbaru yang dioptimalkan Amazon ECS yang dioptimalkan dengan perintah AWS CLI berikut dengan menggunakan subparameter. `image_id` Ganti *`region`* dengan kode Region yang Anda inginkan ID AMI. Untuk informasi tentang dukungan Wilayah AWS, lihat Menemukan AMI di GitHub. Untuk mengambil versi selain yang terbaru, ganti `latest` dengan nomor versi.
  - Untuk arsitektur 64-bit (`x86_64`):

    ```
    aws ssm get-parameter --region us-east-2 --name "/aws/service/bottlerocket/aws-ecs-2/x86_64/latest/image_id" --query Parameter.Value --output text
    ```

  - Untuk arsitektur Arm (`arm64`) 64-bit:

    ```
    aws ssm get-parameter --region us-east-2 --name "/aws/service/bottlerocket/aws-ecs-2/arm64/latest/image_id" --query Parameter.Value --output text
    ```

- AWS Management Console— Anda dapat menanyakan ID AMI yang dioptimalkan Amazon ECS yang direkomendasikan menggunakan URL di. AWS Management Console URL membuka konsol Amazon EC2 Systems Manager dengan nilai ID untuk parameternya. Di URL berikut, ganti *`region`* dengan kode Region yang Anda inginkan ID AMI. Untuk informasi tentang dukungan Wilayah AWS, lihat Menemukan AMI di GitHub.
  - Untuk arsitektur 64-bit (`x86_64`):

```
https://console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/
aws-ecs-2/x86_64/latest/image_id/description?region=region#
```

- Untuk arsitektur Arm (`arm64`) 64-bit:

```
https://console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/
aws-ecs-2/arm64/latest/image_id/description?region=region#
```

Mengambil varian AMI **`aws-ecs-2-nvidia`** Bottlerocket

Anda dapat mengambil varian AMI `aws-ecs-2-nvdia` Bottlerocket stabil terbaru berdasarkan Wilayah dan arsitektur dengan atau. AWS CLI AWS Management Console

- AWS CLI— Anda dapat mengambil ID gambar dari Bottlerocket AMI terbaru yang dioptimalkan Amazon ECS yang dioptimalkan dengan perintah AWS CLI berikut dengan menggunakan subparameter. `image_id` Ganti *`region`* dengan kode Region yang Anda inginkan ID AMI. Untuk informasi tentang dukungan Wilayah AWS, lihat Menemukan AMI di GitHub. Untuk mengambil versi selain yang terbaru, ganti `latest` dengan nomor versi.
  - Untuk arsitektur 64-bit (`x86_64`):

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-
ecs-2-nvidia/x86_64/latest/image_id" --query Parameter.Value --output text
```

  - Untuk arsitektur Arm (`arm64`) 64 bit:

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-
ecs-2-nvidia/arm64/latest/image_id" --query Parameter.Value --output text
```

- AWS Management Console— Anda dapat meminta ID AMI yang dioptimalkan Amazon ECS yang direkomendasikan menggunakan URL di. AWS Management Console URL membuka konsol Amazon EC2 Systems Manager dengan nilai ID untuk parameternya. Di URL berikut, ganti *`region`* dengan kode Region yang Anda inginkan ID AMI. Untuk informasi tentang dukungan Wilayah AWS, lihat Menemukan AMI di GitHub.
  - Untuk arsitektur 64 bit (`x86_64`):

```
https://regionconsole.aws.amazon.com/systems-manager/parameters/aws/service/
bottlerocket/aws-ecs-2-nvidia/x86_64/latest/image_id/description?region=region#
```

- Untuk arsitektur Arm (`arm64`) 64 bit:

```
https://regionconsole.aws.amazon.com/systems-manager/parameters/aws/service/
bottlerocket/aws-ecs-2-nvidia/arm64/latest/image_id/description?region=region#
```

Mengambil varian AMI **`aws-ecs-1`** Bottlerocket

Anda dapat mengambil varian AMI `aws-ecs-1` Bottlerocket stabil terbaru Wilayah AWS dengan dan arsitektur dengan atau. AWS CLI AWS Management Console

- AWS CLI— Anda dapat mengambil ID gambar dari Bottlerocket AMI terbaru yang dioptimalkan Amazon ECS yang dioptimalkan dengan perintah AWS CLI berikut dengan menggunakan subparameter. `image_id` Ganti *`region`* dengan kode Region yang Anda inginkan ID AMI. Untuk informasi tentang dukungan Wilayah AWS, lihat Menemukan AMI di GitHub. Untuk mengambil versi selain yang terbaru, ganti `latest` dengan nomor versi.
  - Untuk arsitektur 64-bit (`x86_64`):

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-
ecs-1/x86_64/latest/image_id" --query Parameter.Value --output text
```

  - Untuk arsitektur Arm (`arm64`) 64-bit:

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-
ecs-1/arm64/latest/image_id" --query Parameter.Value --output text
```

- AWS Management Console— Anda dapat menanyakan ID AMI yang dioptimalkan Amazon ECS yang direkomendasikan menggunakan URL di. AWS Management Console URL membuka konsol Amazon EC2 Systems Manager dengan nilai ID untuk parameternya. Di URL berikut, ganti *`region`* dengan kode Region yang Anda inginkan ID AMI. Untuk informasi tentang dukungan Wilayah AWS, lihat Menemukan AMI di GitHub.
  - Untuk arsitektur 64-bit (`x86_64`):

```
https://region.console.aws.amazon.com/systems-manager/parameters/aws/service/
bottlerocket/aws-ecs-1/x86_64/latest/image_id/description
```

  - Untuk arsitektur Arm (`arm64`) 64-bit:

```
https://region.console.aws.amazon.com/systems-manager/parameters/aws/service/
bottlerocket/aws-ecs-1/arm64/latest/image_id/description
```

Mengambil varian AMI **`aws-ecs-1-nvidia`** Bottlerocket

Anda dapat mengambil varian AMI `aws-ecs-1-nvdia` Bottlerocket stabil terbaru berdasarkan Wilayah dan arsitektur dengan atau. AWS CLI AWS Management Console

- AWS CLI— Anda dapat mengambil ID gambar dari Bottlerocket AMI terbaru yang dioptimalkan Amazon ECS yang dioptimalkan dengan perintah AWS CLI berikut dengan menggunakan subparameter. `image_id` Ganti *`region`* dengan kode Region yang Anda inginkan ID AMI. Untuk informasi tentang dukungan Wilayah AWS, lihat Menemukan AMI di GitHub. Untuk mengambil versi selain yang terbaru, ganti `latest` dengan nomor versi.
  - Untuk arsitektur 64-bit (`x86_64`):

    ```
    aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-
    ecs-1-nvidia/x86_64/latest/image_id" --query Parameter.Value --output text
    ```

  - Untuk arsitektur Arm (`arm64`) 64 bit:

    ```
    aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-
    ecs-1-nvidia/arm64/latest/image_id" --query Parameter.Value --output text
    ```

- AWS Management Console— Anda dapat meminta ID AMI yang dioptimalkan Amazon ECS yang direkomendasikan menggunakan URL di. AWS Management Console URL membuka konsol Amazon EC2 Systems Manager dengan nilai ID untuk parameternya. Di URL berikut, ganti *`region`* dengan kode Region yang Anda inginkan ID AMI. Untuk informasi tentang dukungan Wilayah AWS, lihat Menemukan AMI di GitHub.
  - Untuk arsitektur 64 bit (`x86_64`):

    ```
    https://console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/
    aws-ecs-1-nvidia/x86_64/latest/image_id/description?region=region#
    ```

  - Untuk arsitektur Arm (`arm64`) 64 bit:

    ```
    https://console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/
    aws-ecs-1-nvidia/arm64/latest/image_id/description?region=region#
    ```

Langkah selanjutnya

Untuk panduan terperinci tentang cara memulai dengan sistem Bottlerocket operasi di Amazon ECS, lihat Menggunakan AMI Bottlerocket dengan Amazon ECS aktif GitHub dan Memulai dan Amazon ECS di situs blog Bottlerocket. AWS

Untuk informasi tentang cara meluncurkan instance Bottlerocket, lihat Meluncurkan sebuah Bottlerocket instance

Meluncurkan sebuah Bottlerocket instance

Anda dapat meluncurkan Bottlerocket instance sehingga Anda dapat menjalankan beban kerja kontainer Anda.

Anda dapat menggunakan AWS CLI untuk meluncurkan Bottlerocket instance.

1. Buat file yang dipanggil`userdata.toml`. File ini digunakan untuk data pengguna contoh. Ganti *`nama cluster dengan nama`* cluster Anda.

   ```
   [settings.ecs]
   cluster = "cluster-name"
   ```

2. Gunakan salah satu perintah yang disertakan the section called “Mengambil AMI Amazon ECS Bottlerocket yang dioptimalkan” untuk mendapatkan ID Bottlerocket AMI. Anda menggunakan ini pada langkah berikut.
3. Jalankan perintah berikut untuk meluncurkan Bottlerocket instance. Ingatlah untuk mengganti parameter berikut:

   - Ganti *`subnet`* dengan ID subnet pribadi atau publik tempat instans Anda akan diluncurkan.
   - Ganti *`bottlerocket_ami dengan ID AMI`* dari langkah sebelumnya.
   - Ganti *`t3.large`* dengan tipe instance yang ingin Anda gunakan.
   - Ganti *`wilayah`* dengan kode Wilayah.

   ```
   aws ec2 run-instances --key-name ecs-bottlerocket-example \
      --subnet-id subnet \
      --image-id bottlerocket_ami \
      --instance-type t3.large \
      --region region \
      --tag-specifications
    'ResourceType=instance,Tags=[{Key=bottlerocket,Value=example}]' \
   ```

```
    --user-data file://userdata.toml \
    --iam-instance-profile Name=ecsInstanceRole
```

4. Jalankan perintah berikut untuk memverifikasi bahwa instance container terdaftar ke cluster. Saat Anda menjalankan perintah ini, ingatlah untuk mengganti parameter berikut:

   - Ganti *cluster* dengan nama cluster Anda.
   - Ganti *wilayah* dengan kode Wilayah Anda.

```
aws ecs list-container-instances --cluster cluster-name --region region
```

Untuk panduan terperinci tentang cara memulai dengan sistem Bottlerocket operasi di Amazon ECS, lihat Menggunakan AMI Bottlerocket dengan Amazon ECS aktif GitHub dan Memulai dan Amazon ECS di situs blogBottlerocket. AWS

Memasang agen kontainer Amazon ECS

Jika instans penampung Anda tidak diluncurkan menggunakan AMI yang dioptimalkan Amazon ECS, Anda dapat menginstal agen penampung Amazon ECS secara manual menggunakan salah satu prosedur berikut. Agen kontainer Amazon ECS termasuk dalam AMI Amazon ECS yang dioptimalkan dan tidak memerlukan instalasi.

- Untuk instans Amazon Linux 2, Anda dapat menginstal agen menggunakan perintah `amazon-linux-extras`. Untuk informasi selengkapnya, lihat Menginstal agen kontainer Amazon ECS pada instans Amazon Linux 2 EC2.
- Untuk instans Amazon Linux AMI, Anda dapat menginstal agen menggunakan repo Amazon YUM. Untuk informasi selengkapnya, lihat Menginstal agen penampung Amazon ECS pada instans Amazon Linux EC2.
- Untuk Amazon Linux non-instance, Anda dapat mengunduh agen dari salah satu bucket S3 regional atau dari Amazon Elastic Container Registry Public. Jika Anda mengunduh dari salah satu bucket S3 wilayah, Anda dapat secara opsional memastikan validitas file agen kontainer menggunakan tanda tangan PGP. Untuk informasi selengkapnya, lihat Menginstal agen kontainer Amazon ECS pada instans Amazon Linux non-EC2.

> **Note**
>
> `systemd`Unit untuk layanan Amazon ECS dan Docker memiliki arahan untuk menunggu `cloud-init` hingga selesai sebelum memulai kedua layanan. `cloud-init`Proses ini tidak dianggap selesai sampai data pengguna Amazon EC2 Anda selesai berjalan. Oleh karena itu, memulai Amazon ECS atau Docker melalui data pengguna Amazon EC2 dapat menyebabkan kebuntuan. Untuk memulai agen kontainer menggunakan data pengguna Amazon EC2 yang dapat Anda gunakan. `systemctl enable --now --no-block ecs.service`

Menginstal agen kontainer Amazon ECS pada instans Amazon Linux 2 EC2

Untuk menginstal agen penampung Amazon ECS pada instans Amazon Linux 2 EC2 menggunakan `amazon-linux-extras` perintah, gunakan langkah-langkah berikut.

Untuk menginstal agen kontainer Amazon ECS pada instans Amazon Linux 2 EC2

1. Luncurkan instans Amazon Linux 2 EC2 dengan peran IAM yang memungkinkan akses ke Amazon ECS. Untuk informasi selengkapnya, lihat Peran IAM instans wadah Amazon ECS.
2. Hubungkan ke instans Anda.
3. Nonaktifkan repositori tambahan Amazon Linux `docker`. Repositori `ecs` Amazon Linux ekstra dikirimkan dengan versi Docker-nya sendiri, jadi `docker` tambahan harus dimatikan untuk menghindari potensi konflik di masa depan. Ini memastikan bahwa Anda selalu menggunakan versi Docker yang Amazon ECS maksudkan untuk Anda gunakan dengan versi tertentu dari agen penampung.

   ```
   [ec2-user ~]$ sudo amazon-linux-extras disable docker
   ```

4. Instal dan aktifkan repositori tambahan Amazon Linux `ecs`.

   ```
   [ec2-user ~]$ sudo amazon-linux-extras install -y ecs; sudo systemctl enable --now
    ecs
   ```

5. (Opsional) Anda dapat memastikan bahwa agen sedang berjalan dan melihat beberapa informasi tentang instans kontainer baru dengan API introspeksi agen. Untuk informasi selengkapnya, lihat the section called “Introspeksi wadah”.

```
[ec2-user ~]$ curl -s http://localhost:51678/v1/metadata | python -mjson.tool
```

> **Note**
>
> Jika Anda tidak mendapatkan respons, pastikan Anda mengaitkan peran IAM instans penampung Amazon ECS saat meluncurkan instance. Untuk informasi selengkapnya, lihat Peran IAM instans wadah Amazon ECS.

Menginstal agen penampung Amazon ECS pada instans Amazon Linux EC2

Untuk menginstal agen penampung Amazon ECS di instans Amazon Linux EC2 menggunakan repo Amazon YUM, gunakan langkah-langkah berikut.

Untuk menginstal agen penampung Amazon ECS pada instans Amazon Linux EC2

1. Luncurkan instans Amazon Linux EC2 dengan peran IAM yang memungkinkan akses ke Amazon ECS. Untuk informasi selengkapnya, lihat Peran IAM instans wadah Amazon ECS.
2. Hubungkan ke instans Anda.
3. Instal paket `ecs-init`. Untuk informasi selengkapnya`ecs-init`, lihat kode sumber di GitHub.

   ```
   [ec2-user ~]$ sudo yum install -y ecs-init
   ```
4. Mulai daemon Docker.

   ```
   [ec2-user ~]$ sudo service docker start
   ```

   Output:

   ```
   Starting cgconfig service:                                 [  OK  ]
   Starting docker:                                  [  OK  ]
   ```
5. Mulai tugas upstart `ecs-init`.

   ```
   [ec2-user ~]$ sudo service ecs start
   ```

   Output:

```
ecs start/running, process 2804
```

6. (Opsional) Anda dapat memastikan bahwa agen sedang berjalan dan melihat beberapa informasi tentang instans kontainer baru dengan API introspeksi agen. Untuk informasi selengkapnya, lihat the section called “Introspeksi wadah”.

```
[ec2-user ~]$ curl -s http://localhost:51678/v1/metadata | python -mjson.tool
```

Menginstal agen kontainer Amazon ECS pada instans Amazon Linux non-EC2

Untuk menginstal agen penampung Amazon ECS pada instans Amazon Linux non-EC2, Anda dapat mengunduh agen dari salah satu bucket S3 regional dan menginstalnya.

> **Note**
>
> Saat menggunakan AMI Linux non-Amazon, instans Amazon EC2 Anda `cgroupfs` memerlukan dukungan untuk `cgroup` driver agar agen Amazon ECS mendukung batas sumber daya tingkat tugas. Untuk informasi selengkapnya, lihat agen Amazon ECS di GitHub.

File agen kontainer Amazon ECS terbaru, menurut Wilayah, untuk setiap arsitektur sistem tercantum di bawah ini untuk referensi.

Wilayah	Nama wilayah	Amazon ECS init file deb	Amazon ECS memanas file rpm
us-east-2	AS Timur (Ohio)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
us-east-1	AS Timur (Virginia Utara)	Amazon ECS memanas amd64 (amd64)	Amazon ECS panas x86_64 (x86_64)

Wilayah	Nama wilayah	Amazon ECS init file deb	Amazon ECS memanas file rpm
		Amazon ECS init arm64 (arm64)	Amazon ECS panas aarch64 (aarch64)
us-west-1	AS Barat (California Utara)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
us-west-2	AS Barat (Oregon)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
ap-east-1	Asia Pasifik (Hong Kong)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
ap-northeast-1	Asia Pasifik (Tokyo)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
ap-northeast-2	Asia Pasifik (Seoul)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)

Wilayah	Nama wilayah	Amazon ECS init file deb	Amazon ECS memanas file rpm
ap-south-1	Asia Pasifik (Mumbai)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
ap-southeast-1	Asia Pasifik (Singapura)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
ap-southeast-2	Asia Pasifik (Sydney)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
ca-central-1	Kanada (Pusat)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
eu-central-1	Eropa (Frankfurt)	Amazon ECS memanas amd64 (amd64) Amazon ECS panas arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)

Wilayah	Nama wilayah	Amazon ECS init file deb	Amazon ECS memanas file rpm
eu-west-1	Eropa (Irlandia)	Amazon ECS memanaskan amd64 (amd64) Amazon ECS panas arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
eu-west-2	Eropa (London)	Amazon ECS memanaskan amd64 (amd64) Amazon ECS panas arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
eu-west-3	Eropa (Paris)	Amazon ECS memanaskan amd64 (amd64) Amazon ECS panas arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
sa-east-1	Amerika Selatan (Sao Paulo)	Amazon ECS memanaskan amd64 (amd64) Amazon ECS panas arm64 (arm64)	Amazon ECS panas x86_64 Amazon ECS panas aarch64 (aarch64)
us-gov-east-1	AWS GovCloud (AS-Timur)	Amazon ECS memanaskan amd64 (amd64) Amazon ECS panas arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)

Wilayah	Nama wilayah	Amazon ECS init file deb	Amazon ECS memanas file rpm
us-gov-west-1	AWS GovCloud (AS-Barat)	Amazon ECS memanaskan amd64 (amd64) Amazon ECS panas arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)

Untuk menginstal agen penampung Amazon ECS pada instans Amazon EC2 menggunakan non-AMI Amazon Linux

1. Luncurkan instans Amazon EC2 dengan peran IAM yang memungkinkan akses ke Amazon ECS. Untuk informasi selengkapnya, lihat Peran IAM instans wadah Amazon ECS.
2. Hubungkan ke instans Anda.
3. Instal versi terbaru Docker di instans Anda.
4. Periksa versi Docker Anda untuk memastikan bahwa sistem Anda memenuhi persyaratan versi minimum.

   > **Note**
   >
   > Versi Docker minimum untuk metrik yang andal adalah versi Docker `v20.10.13` dan yang lebih baru, yang disertakan dalam AMI yang dioptimalkan Amazon ECS dan yang lebih baru. `20220607`
   > Versi agen Amazon ECS `1.20.0` dan yang lebih baru memiliki dukungan yang tidak digunakan lagi untuk versi Docker yang lebih lama dari. `1.9.0`

   ```
   docker --version
   ```

5. Unduh file agen Amazon ECS yang sesuai untuk sistem operasi dan arsitektur sistem Anda dan instal.

   Untuk deb arsitektur:

```
ubuntu:~$ curl -O https://s3.us-west-2.amazonaws.com/amazon-ecs-agent-us-west-2/
amazon-ecs-init-latest.amd64.deb
ubuntu:~$ sudo dpkg -i amazon-ecs-init-latest.amd64.deb
```

Untuk `rpm` arsitektur:

```
fedora:~$ curl -O https://s3.us-west-2.amazonaws.com/amazon-ecs-agent-us-west-2/
amazon-ecs-init-latest.x86_64.rpm
fedora:~$ sudo yum localinstall -y amazon-ecs-init-latest.x86_64.rpm
```

6. Edit `/lib/systemd/system/ecs.service` file dan tambahkan baris berikut di akhir `[Unit]` bagian.

```
After=cloud-final.service
```

7. (Opsional) Untuk mendaftarkan instance dengan cluster selain `default` cluster, edit `/etc/ecs/ecs.config` file dan tambahkan konten berikut. Contoh berikut menentukan `MyCluster` cluster.

```
ECS_CLUSTER=MyCluster
```

Untuk informasi selengkapnya tentang opsi waktu aktif agen ini, lihat Konfigurasi agen kontainer Amazon ECS.

> **Note**
>
> Anda dapat menyimpan variabel lingkungan agen secara opsional di Amazon S3 (yang dapat diunduh ke instans penampung Anda pada waktu peluncuran menggunakan data pengguna Amazon EC2). Hal ini dianjurkan untuk informasi sensitif seperti kredensial autentikasi untuk repositori privat. Untuk informasi lebih lanjut, lihat Menyimpan konfigurasi instans kontainer di Amazon S3 dan Autentikasi registri privat untuk tugas.

8. Mulai layanan `ecs`.

```
ubuntu:~$ sudo systemctl start ecs
```

Menjalankan agen Amazon ECS dengan mode jaringan host

Saat menjalankan agen penampung Amazon ECS, `ecs-init` akan membuat wadah agen kontainer dengan mode `host` jaringan. Ini adalah satu-satunya mode jaringan untuk kontainer agen kontainer.

Ini memungkinkan Anda memblokir akses ke titik akhir layanan metadata instans Amazon EC2 (`http://169.254.169.254`) untuk kontainer yang dimulai oleh agen penampung. Hal ini memastikan bahwa kontainer tidak dapat mengakses kredensil peran IAM dari profil instance container dan memberlakukan bahwa tugas hanya menggunakan kredenal peran tugas IAM. Untuk informasi selengkapnya, lihat Tugas peran IAM.

Hal ini juga mengakibatkan agen kontainer tidak berebut lalu lintas koneksi dan jaringan di jembatan `docker0`.

Instans Windows

Instans penampung Amazon ECS adalah instans Amazon EC2 yang menjalankan agen penampung Amazon ECS dan telah terdaftar ke cluster Amazon ECS. Saat Anda menjalankan tugas dengan Amazon ECS menggunakan tipe peluncuran EC2 atau penyedia kapasitas grup Auto Scaling, tugas Anda ditempatkan pada instance container aktif Anda.

> **Note**
>
> Tugas yang menggunakan tipe peluncuran Fargate disebarkan ke infrastruktur yang dikelola oleh AWS, jadi topik ini tidak berlaku.

Topik

- AMI Amazon ECS yang dioptimalkan
- Membangun Windows AMI Amazon ECS Anda sendiri yang dioptimalkan

AMI Amazon ECS yang dioptimalkan

AMI yang dioptimalkan Amazon ECS telah dikonfigurasi sebelumnya dengan komponen yang diperlukan yang Anda perlukan untuk menjalankan beban kerja Amazon ECS. Meskipun Anda dapat membuat AMI instans kontainer sendiri yang memenuhi spesifikasi dasar yang diperlukan untuk menjalankan beban kerja kontainer Anda di Amazon ECS, AMI yang dioptimalkan Amazon ECS telah dikonfigurasi sebelumnya dan diuji di Amazon ECS oleh para insinyur. AWS Ini adalah cara termudah bagi Anda untuk memulai dan mendapatkan kontainer Anda yang berjalan pada AWS dengan cepat.

Metadata AMI Amazon ECS yang dioptimalkan, termasuk nama AMI, versi agen penampung Amazon ECS, dan versi runtime Amazon ECS yang menyertakan versi Docker, untuk setiap varian dapat diambil secara terprogram. Untuk informasi selengkapnya, lihat the section called “Mengambil metadata AMI Amazon ECS yang dioptimalkan”.

Anda dapat berlangganan topik Windows AMI Amazon SNS untuk diberi tahu saat AMI baru dirilis atau versi AMI ditandai pribadi. Untuk informasi selengkapnya, lihat Berlangganan pemberitahuan pembaruan AMI Amazon ECS yang dioptimalkan.

> **Important**
>
> Semua varian AMI yang dioptimalkan ECS yang diproduksi setelah Agustus akan bermigrasi dari Docker EE (Mirantis) ke Docker CE (proyek Moby).
> Untuk memastikan bahwa pelanggan memiliki pembaruan keamanan terbaru secara default, Amazon ECS mempertahankan setidaknya tiga AMI terakhir yang dioptimalkan untuk Windows Amazon ECS. Setelah merilis AMI baru yang dioptimalkan untuk Windows Amazon ECS, Amazon ECS membuat AMI yang dioptimalkan Windows Amazon ECS yang lebih lama menjadi pribadi. Jika ada AMI pribadi yang perlu Anda akses, beri tahu kami dengan mengajukan tiket dengan Cloud Support.

Varian AMI yang dioptimalkan Amazon ECS

Varian Windows Server berikut dari AMI Amazon ECS yang dioptimalkan tersedia untuk instans Amazon EC2 Anda.

> **Important**
>
> Semua varian AMI yang dioptimalkan ECS yang diproduksi setelah Agustus akan bermigrasi dari Docker EE (Mirantis) ke Docker CE (proyek Moby).

- Amazon ECS yang dioptimalkan Windows Server 2022 AMI Penuh
- Server Windows 2022 AMI Inti yang dioptimalkan Amazon ECS
- Amazon ECS yang dioptimalkan Windows Server 2019 AMI Penuh
- Amazon ECS yang dioptimalkan Windows Server 2019 Core AMI
- Amazon ECS yang dioptimalkan Windows Server 2016 AMI Penuh

- Amazon ECS yang dioptimalkan Windows Server 20H2 Core AMI

> **Important**
>
> Pada 9 Agustus 2022, Windows Server 20H2 Core AMI yang dioptimalkan Amazon ECS mencapai akhir tanggal dukungannya. Tidak ada versi baru dari AMI ini yang akan dirilis. Untuk informasi selengkapnya, lihat informasi rilis Windows Server.

Windows Server 2022, Windows Server 2019, dan Windows Server 2016 adalah rilis Long-Term Servicing Channel (LTSC). Windows Server 20H2 adalah rilis Semi-Annual Channel (SAC). Untuk informasi selengkapnya, lihat informasi rilis Windows Server.

peringatan kontainer Windows

Berikut adalah beberapa hal yang harus Anda ketahui tentang Amazon EC2 Windows container dan Amazon ECS.

- Kontainer Windows tidak dapat berjalan pada instans kontainer Linux dan sebaliknya. Untuk penempatan tugas Windows dan Linux yang lebih baik, instans kontainer Windows dan Linux harus tetap berada di klaster terpisah dan tempatkan tugas Windows hanya pada klaster Windows. Anda dapat memastikan bahwa ketentuan tugas Windows hanya ditempatkan pada instans Windows dengan menetapkan batasan penempatan berikut: `memberOf(ecs.os-type=='windows')`.
- Kontainer Windows didukung untuk tugas-tugas yang menggunakan jenis peluncuran EC2 dan Fargate.
- Kontainer Windows dan instans kontainer tidak dapat mendukung semua parameter ketentuan tugas yang tersedia untuk kontainer Linux dan instans kontainer. Beberapa parameter tidak didukung sama sekali, dan perilaku parameter lainnya berbeda saat di Windows dan saat di Linux. Untuk informasi selengkapnya, lihat Pertimbangan EC2 Windows.
- Untuk peran IAM untuk fitur tugas, Anda perlu mengonfigurasi instance wadah Windows Anda untuk mengizinkan fitur saat diluncurkan. Kontainer Anda harus menjalankan beberapa PowerShell kode yang disediakan saat mereka menggunakan fitur tersebut. Untuk informasi selengkapnya, lihat Konfigurasi tambahan untuk peran Windows IAM untuk tugas.
- Fitur peran IAM untuk tugas menggunakan proxy kredenal untuk memberikan kredensil ke kontainer. Proxy kredenal ini menempati port 80 pada instance kontainer, jadi jika Anda menggunakan peran IAM untuk tugas, port 80 tidak tersedia untuk tugas. Untuk kontainer layanan

web, Anda dapat menggunakan Application Load Balancer dan pemetaan port dinamis untuk menyediakan koneksi port HTTP 80 standar ke kontainer Anda. Untuk informasi selengkapnya, lihat Mendistribusikan lalu lintas layanan Amazon ECS menggunakan load balancing.

- Gambar Windows Server Docker berukuran besar (9 GiB). Jadi, instance container Windows Anda membutuhkan lebih banyak ruang penyimpanan daripada instance container Linux.
- Untuk menjalankan wadah Windows di Windows Server, versi OS gambar dasar wadah harus cocok dengan host. Untuk informasi selengkapnya, lihat Kompatibilitas versi penampung Windows di situs web dokumentasi Microsoft. Jika klaster Anda menjalankan beberapa versi Windows, Anda dapat memastikan bahwa tugas ditempatkan pada instans EC2 yang berjalan pada versi yang sama dengan menggunakan batasan penempatan:. `memberOf(attribute:ecs.os-family == WINDOWS_SERVER_<OS_Release>_<FULL or CORE>)` Untuk informasi selengkapnya, lihat the section called “Mengambil metadata AMI Amazon ECS yang dioptimalkan”.

Mengambil metadata AMI Amazon ECS yang dioptimalkan

ID AMI, nama gambar, sistem operasi, versi agen kontainer, dan versi runtime untuk setiap varian AMI yang dioptimalkan Amazon ECS dapat diambil secara terprogram dengan menanyakan Systems Manager Parameter Store API. Untuk informasi selengkapnya tentang Systems Manager Parameter Store API, lihat GetParametersdan GetParametersByPath.

> **Note**
>
> Pengguna administratif Anda harus memiliki izin IAM berikut untuk mengambil metadata AMI Amazon ECS yang dioptimalkan. Izin ini telah ditambahkan ke kebijakan `AmazonECS_FullAccess` IAM.
>
> - ssm: GetParameters
> - ssm: GetParameter
> - ssm: GetParametersByPath

Systems Manager Parameter Menyimpan format parameter

> **Note**
>
> Parameter Systems Manager Parameter Store API berikut tidak digunakan lagi dan tidak boleh digunakan untuk mengambil AMI Windows terbaru:

- `/aws/service/ecs/optimized-ami/windows_server/2016/english/full/recommended/image_id`
- `/aws/service/ecs/optimized-ami/windows_server/2019/english/full/recommended/image_id`

Berikut ini adalah format nama parameter untuk setiap varian AMI Amazon ECS yang dioptimalkan.

- Windows Server 2022 Metadata AMI lengkap:

```
/aws/service/ami-windows-latest/Windows_Server-2022-English-Full-ECS_Optimized
```

- Metadata AMI Inti Windows Server 2022:

```
/aws/service/ami-windows-latest/Windows_Server-2022-English-Core-ECS_Optimized
```

- Metadata AMI Windows Server 2019 Full:

```
/aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized
```

- Metadata AMI Windows Server 2019 Core:

```
/aws/service/ami-windows-latest/Windows_Server-2019-English-Core-ECS_Optimized
```

- Metadata AMI Windows Server 2016 Full:

```
/aws/service/ami-windows-latest/Windows_Server-2016-English-Full-ECS_Optimized
```

Format nama parameter berikut mengambil metadata dari Windows Server 2019 Full AMI stabil terbaru

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized
```

Berikut ini adalah contoh objek JSON yang dikembalikan untuk nilai parameter.

```
{
    "Parameters": [
```

```
        {
            "Name": "/aws/service/ami-windows-latest/Windows_Server-2019-English-Full-
ECS_Optimized",
            "Type": "String",
            "Value": "{\"image_name\":\"Windows_Server-2019-English-Full-
ECS_Optimized-2023.06.13\",\"image_id\":\"ami-0debc1fb48e4aee16\",\"ecs_runtime_version
\":\"Docker (CE) version 20.10.21\",\"ecs_agent_version\":\"1.72.0\"}",
            "Version": 58,
            "LastModifiedDate": "2023-06-22T19:37:37.841000-04:00",
            "ARN": "arn:aws:ssm:us-east-1::parameter/aws/service/ami-windows-latest/
Windows_Server-2019-English-Full-ECS_Optimized",
            "DataType": "text"
        }
    ],
    "InvalidParameters": []
}
```

Masing-masing kolom dalam output di atas tersedia untuk dilakukan kueri sebagai sub-parameter. Buat jalur parameter untuk sub-parameter dengan menambahkan nama sub-parameter ke jalur untuk AMI yang dipilih. Sub-parameter berikut tersedia:

- `schema_version`
- `image_id`
- `image_name`
- `os`
- `ecs_agent_version`
- `ecs_runtime_version`

Contoh-contoh

Contoh berikut menunjukkan cara Anda dapat mengambil metadata untuk setiap varian AMI Amazon ECS yang dioptimalkan.

Mengambil metadata AMI terbaru yang dioptimalkan Amazon ECS stabil

Anda dapat mengambil AMI stabil Amazon ECS terbaru yang dioptimalkan menggunakan AWS CLI perintah berikut. AWS CLI

- Untuk AMI Lengkap Windows Server 2022 yang dioptimalkan Amazon ECS:

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2022-English-Full-ECS_Optimized --region us-east-1
```

- Untuk AMI Inti Windows Server 2022 yang dioptimalkan oleh Amazon ECS:

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2022-English-Core-ECS_Optimized --region us-east-1
```

- Untuk AMI Lengkap Windows Server 2019 yang dioptimalkan Amazon ECS:

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized --region us-east-1
```

- Untuk Windows Server 2019 Core AMI yang dioptimalkan Amazon ECS:

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Core-ECS_Optimized --region us-east-1
```

- Untuk Amazon ECS yang dioptimalkan Windows Server 2016 AMI Penuh:

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2016-English-Full-ECS_Optimized --region us-east-1
```

Menggunakan AMI terbaru yang dioptimalkan Amazon ECS yang dioptimalkan dalam template AWS CloudFormation

Anda dapat mereferensikan AMI terbaru yang dioptimalkan Amazon ECS yang dioptimalkan dalam AWS CloudFormation template dengan mereferensikan nama penyimpanan parameter Systems Manager.

```
Parameters:
  LatestECSOptimizedAMI:
    Description: AMI ID
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized/image_id
```

Berlangganan pemberitahuan pembaruan AMI Amazon ECS yang dioptimalkan

AWS menyediakan dua ARN topik Amazon SNS untuk pemberitahuan yang terkait dengan AMI Windows Server. Satu topik mengirimkan pemberitahuan pembaruan ketika AMI Windows Server baru dirilis. Topik yang lainnya mengirimkan notifikasi ketika AMIs Server Windows yang dirilis sebelumnya dibuat privat. Meskipun topik ini tidak spesifik untuk AMI Windows Amazon ECS yang dioptimalkan, karena AMI Windows yang dioptimalkan Amazon ECS mengikuti jadwal rilis yang sama, Anda dapat menggunakan pemberitahuan ini untuk indikasi kapan AMI Windows yang dioptimalkan Amazon ECS baru diperbarui. Untuk informasi selengkapnya tentang berlangganan notifikasi AMI Windows, lihat Berlangganan notifikasi AMI Windows di Panduan Pengguna Amazon EC2 untuk Instans Windows.

> **Note**
>
> Pengguna Anda, atau peran yang dilampirkan ke pengguna Anda harus memiliki izin `sns::subscribe` IAM untuk berlangganan topik Amazon SNS.

Versi AMI Amazon ECS yang dioptimalkan

Lihat versi saat ini dan sebelumnya dari AMI yang dioptimalkan Amazon ECS dan versi terkait dari agen kontainer Amazon ECS, Docker, dan paket. `ecs-init`

Metadata AMI Amazon ECS yang dioptimalkan, termasuk ID AMI, untuk setiap varian dapat diambil secara terprogram. Untuk informasi selengkapnya, lihat the section called “Mengambil metadata AMI Amazon ECS yang dioptimalkan”.

Versi AMI yang dioptimalkan untuk Windows Amazon ECS

Tab berikut menampilkan daftar versi AMI yang dioptimalkan Windows Amazon ECS.

> **Note**
>
> Untuk detail tentang referensi parameter Systems Manager Parameter Store dalam AWS CloudFormation template, lihatMenggunakan AMI terbaru yang dioptimalkan Amazon ECS yang dioptimalkan dalam template AWS CloudFormation.

> **Important**
>
> Untuk memastikan bahwa pelanggan memiliki pembaruan keamanan terbaru secara default, Amazon ECS mempertahankan setidaknya tiga AMI terakhir yang dioptimalkan untuk Windows Amazon ECS. Setelah merilis AMI baru yang dioptimalkan untuk Windows Amazon ECS, Amazon ECS membuat AMI yang dioptimalkan Windows Amazon ECS yang lebih lama menjadi pribadi. Jika ada AMI pribadi yang perlu Anda akses, beri tahu kami dengan mengajukan tiket dengan Cloud Support.

Windows Server 2022 Full AMI versions

Tabel di bawah ini mencantumkan versi saat ini dan sebelumnya dari AMI Penuh Windows Server 2022 yang dioptimalkan Amazon ECS dan versi terkait dari agen penampung Amazon ECS dan Docker.

Amazon ECS Server Windows 2022 AMI Penuh yang dioptimalkan ECS	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2024.02.13	`1.81.0`	`20.10.23 (Docker CE)`	Publik
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2024.01.09	`1.79.2`	`20.10.23 (Docker CE)`	Publik
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2023.12.12	`1.79.1`	`20.10.23 (Docker CE)`	Publik
Windows_server-2022-Inggris-penuh-EC	`1.79.0`	`20.10.23 (Docker CE)`	Publik

Amazon ECS Server Windows 2022 AMI Penuh yang dioptimalkan ECS	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
S_dioptimal-2023.11.14			
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2023.10.11	`1.77.0`	`20.10.21 (Docker CE)`	Publik
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2023.09.15	`1.75.3`	`20.10.21 (Docker CE)`	Privat
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2023.08.09	`1.74.1`	`20.10.21 (Docker CE)`	Privat
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2023.07.11	`1.73.1`	`20.10.21 (Docker CE)`	Privat
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2023.06.13	`1.72.0`	`20.10.21 (Docker CE)`	Privat
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2023.05.18	`1.71.1`	`20.10.21 (Docker CE)`	Privat

Amazon ECS Server Windows 2022 AMI Penuh yang dioptimalkan ECS	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2023.04.18	1.70.2	20.10.21 (Docker CE)	Privat
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2023.03.21	1.69.0	20.10.21 (Docker CE)	Privat
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2023.02.21	1.68.2	20.10.21 (Docker CE)	Privat
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2023.01.11	1.68.0	20.10.21 (Docker CE)	Privat
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2022.12.14	1.67.2	20.10.21 (Docker CE)	Privat
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2022.11.09	1.65.1	20.10.21 (Docker CE)	Privat

Amazon ECS Server Windows 2022 AMI Penuh yang dioptimalkan ECS	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2022.10.12	`1.64.0`	`20.10.17 (Docker CE)`	Privat
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2022.09.22	`1.63.1`	`20.10.17 (Docker CE)`	Privat
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2022.09.14	`1.62.2`	`20.10.17 (Docker CE)`	Privat
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2022.08.15	`1.62.1`	`20.10.9`	Privat
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2022.07.13	`1.61.3`	`20.10.9`	Privat
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2022.06.15	`1.61.2`	`20.10.9`	Privat

Amazon ECS Server Windows 2022 AMI Penuh yang dioptimalkan ECS	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2022.01.18	1.57.1	20.10.9	Privat
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2021.12.16	1.57.1	20.10.7	Privat
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2021.11.11	1.57.0	20.10.7	Privat
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2021.09.23	1.55.3	20.10.7	Privat

Gunakan AWS CLI perintah berikut untuk mengambil AMI Penuh Windows Server 2022 yang dioptimalkan Amazon ECS saat ini.

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2022-English-Full-ECS_Optimized
```

Windows Server 2022 Core AMI versions

Tabel di bawah ini mencantumkan versi saat ini dan sebelumnya dari Windows Server 2022 Core AMI yang dioptimalkan Amazon ECS dan versi terkait dari agen penampung Amazon ECS dan Docker.

Server Windows 2022 AMI Inti yang dioptimalkan Amazon ECS	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2022-Inggris-core-ECS_Optimized-2024.02.13	`1.81.0`	`20.10.23 (Docker CE)`	Publik
Windows_Server-2022-Inggris-Core-ECS_Optimized-2024.01.09	`1.79.2`	`20.10.23 (Docker CE)`	Publik
Windows_Server-2022-Inggris-Core-ECS_Optimized-2023.12.12	`1.79.1`	`20.10.23 (Docker CE)`	Publik
Windows_Server-2022-Inggris-Core-ECS_Optimized-2023.11.14	`1.79.0`	`20.10.23 (Docker CE)`	Publik
Windows_Server-2022-Inggris-Core-ECS_Optimized-2023.10.11	`1.77.0`	`20.10.21 (Docker CE)`	Publik
Windows_Server-2022-Inggris-Core-ECS_Optimized-2023.09.15	`1.75.3`	`20.10.21 (Docker CE)`	Privat
Windows_Server-2022-Inggris-Core-ECS	`1.74.1`	`20.10.21 (Docker CE)`	Privat

Server Windows 2022 AMI Inti yang dioptimalkan Amazon ECS	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
_Optimized-2023.08.09			
Windows_server-2022-Inggris-core-ECS_Optimized-2023.07.11	1.73.1	20.10.21 (Docker CE)	Privat
Windows_Server-2022-Inggris-Core-ECS_Optimized-2023.06.13	1.72.0	20.10.21 (Docker CE)	Privat
Windows_server-2022-Inggris-core-ECS_Optimized-2023.05.18	1.71.1	20.10.21 (Docker CE)	Privat
Windows_server-2022-Inggris-core-ECS_Optimized-2023.04.18	1.70.2	20.10.21 (Docker CE)	Privat
Windows_Server-2022-Inggris-Core-ECS_Optimized-2023.03.21	1.69.0	20.10.21 (Docker CE)	Privat
Windows_Server-2022-Inggris-Core-ECS_Optimized-2023.02.21	1.68.2	20.10.21 (Docker CE)	Privat

Server Windows 2022 AMI Inti yang dioptimalkan Amazon ECS	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_Server-2022-Inggris-Core-ECS_Optimized-2023.01.11	1.68.0	20.10.21 (Docker CE)	Privat
Windows_Server-2022-Inggris-Core-ECS_Optimized-2022.12.14	1.67.2	20.10.21 (Docker CE)	Privat
Windows_Server-2022-Inggris-Core-ECS_Optimized-2022.11.09	1.65.1	20.10.21 (Docker CE)	Privat
Windows_Server-2022-Inggris-Core-ECS_Optimized-2022.10.12	1.64.0	20.10.17 (Docker CE)	Privat
Windows_Server-2022-Inggris-Core-ECS_Optimized-2022.09.22	1.63.1	20.10.17 (Docker CE)	Privat
Windows_Server-2022-Inggris-Core-ECS_Optimized-2022.09.14	1.62.2	20.10.17 (Docker CE)	Privat

Server Windows 2022 AMI Inti yang dioptimalkan Amazon ECS	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_Server-2022-Inggris-Core-ECS_Optimized-2022.08.15	1.62.1	20.10.9	Privat
Windows_server-2022-Inggris-core-ECS_Optimized-2022.07.13	1.61.3	20.10.9	Privat
Windows_Server-2022-Inggris-Core-ECS_Optimized-2022.06.15	1.61.2	20.10.9	Privat
Windows_Server-2022-Inggris-Core-ECS_Optimized-2021.12.16	1.57.1	20.10.7	Privat
Windows_Server-2022-Inggris-Core-ECS_Optimized-2021.11.11	1.57.0	20.10.7	Privat
Windows_Server-2022-Inggris-Core-ECS_Optimized-2021.009.23	1.55.3	20.10.7	Privat

Gunakan AWS CLI perintah berikut untuk mengambil AMI Penuh Windows Server 2022 yang dioptimalkan Amazon ECS saat ini.

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2022-English-Core-ECS_Optimized
```

Windows Server 2019 Full AMI versions

Tabel di bawah ini mencantumkan versi saat ini dan sebelumnya dari AMI Penuh Windows Server 2019 yang dioptimalkan Amazon ECS dan versi terkait dari agen penampung Amazon ECS dan Docker.

Amazon ECS yang dioptimalkan Windows Server 2019 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2024.02.13	`1.81.0`	`20.10.23 (Docker CE)`	Publik
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2024.01.09	`1.79.2`	`20.10.23 (Docker CE)`	Publik
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2023.12.12	`1.79.1`	`20.10.23 (Docker CE)`	Publik
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2023.11.14	`1.79.0`	`20.10.23 (Docker CE)`	Publik
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2023.10.11	`1.77.0`	`20.10.21 (Docker CE)`	Publik

Amazon ECS yang dioptimalkan Windows Server 2019 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2023.09.15	1.75.3	20.10.21 (Docker CE)	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2023.08.09	1.74.1	20.10.21 (Docker CE)	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2023.07.11	1.73.1	20.10.21 (Docker CE)	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2023.06.13	1.72.0	20.10.21 (Docker CE)	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2023.05.18	1.71.1	20.10.21 (Docker CE)	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2023.04.18	1.70.2	20.10.21 (Docker CE)	Privat

Amazon ECS yang dioptimalkan Windows Server 2019 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2023.03.21	1.69.0	20.10.21 (Docker CE)	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2023.02.21	1.68.2	20.10.21 (Docker CE)	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2023.01.11	1.68.0	20.10.21 (Docker CE)	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2022.12.14	1.67.2	20.10.21 (Docker CE)	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2022.11.09	1.65.1	20.10.21 (Docker CE)	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2022.10.12	1.64.0	20.10.17 (Docker CE)	Privat

Amazon ECS yang dioptimalkan Windows Server 2019 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2022.09.22	1.63.1	20.10.17 (Docker CE)	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2022.09.14	1.62.2	20.10.17 (Docker CE)	Privat
Windows_server-2019-Inggris-penuh-ECS_Optimized-2022.08.15	1.62.1	20.10.9	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2022.07.13	1.61.3	20.10.9	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2022.06.15	1.61.2	20.10.9	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2022.01.18	1.57.1	20.10.9	Privat

Amazon ECS yang dioptimalkan Windows Server 2019 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2021.12.16	1.57.1	20.10.7	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2021.11.11	1.57.0	20.10.7	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2021.09.23	1.55.3	20.10.7	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2021.08.12	1.55.0	20.10.6	Publik
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2021.07.13	1.54.02	20.10.6	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2021.07.08	1.54.0	20.10.5	Privat

Amazon ECS yang dioptimalkan Windows Server 2019 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2021.06.11	1.53.0	20.10.5	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2021.05.21	1.52.2	20.10.4	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2021.04.14	1.51.0	20.10.0	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2021.03.11	1.50.2	19.03.14	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2021.02.10	1.50.0	19.03.14	Privat
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2021.01.13	1.49.0	19.03.14	Privat

Amazon ECS yang dioptimalkan Windows Server 2019 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2019-Inggris-penuh-ECS_Optimized-2020.11.18	1.48.0	19.03.13	Privat
Windows_server-2019-Inggris-penuh-ECS_Optimized-2020.11.06	1.47.0	19.03.11	Privat
Windows_server-2019-Inggris-penuh-ECS_Optimized-2020.10.14	1.45.0	19.03.11	Privat
Windows_server-2019-Inggris-penuh-ECS_Optimized-2020.08.12	1.43.0	19.03.11	Privat
Windows_server-2019-Inggris-penuh-ECS_Optimized-2020.07.15	1.41.1	19.03.5	Privat
Windows_server-2019-Inggris-penuh-ECS_Optimized-2020.06.11	1.40.0	19.03.5	Privat

Amazon ECS yang dioptimalkan Windows Server 2019 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2019-Inggris-penuh-ECS_Optimized-2020.05.14	1.39.0	19.03.5	Privat
Windows_server-2019-Inggris-penuh-ECS_Optimized-2020.01.15	1.35.0	19.03.5	Privat
Windows_server-2019-Inggris-penuh-ECS_Optimized-2019.12.16	1.34.0	19.03.5	Privat
Windows_server-2019-Inggris-penuh-ECS_Optimized-2019.11.25	1.34.0	19.03.4	Privat
Windows_server-2019-Inggris-penuh-ECS_Optimized-2019.11.13	1.32.1	19.03.4	Privat
Windows_server-2019-Inggris-penuh-ECS_Optimized-2019.10.09	1.32.0	19.03.2	Privat

Amazon ECS yang dioptimalkan Windows Server 2019 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2019-Inggris-penuh-ECS_Optimized-2019.09.11	1.30.0	19.03.1	Privat
Windows_server-2019-Inggris-penuh-ECS_Optimized-2019.08.16	1.29.1	19.03.1	Privat
Windows_server-2019-Inggris-penuh-ECS_Optimized-2019.07.19	1.29.0	18.09.8	Privat
Windows_server-2019-Inggris-penuh-ECS_Optimized-2019.05.10	1.27.0	18.09.4	Privat

Gunakan AWS CLI perintah berikut untuk mengambil AMI Penuh Windows Server 2019 yang dioptimalkan Amazon ECS saat ini.

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized
```

Windows Server 2019 Core AMI versions

Tabel di bawah ini mencantumkan versi saat ini dan sebelumnya dari Windows Server 2019 Core AMI yang dioptimalkan Amazon ECS dan versi terkait dari agen penampung Amazon ECS dan Docker.

AMI Inti Windows Server 2019 yang dioptimalkan Amazon ECS	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2019-Inggris-core-ECS_Optimized-2024.02.13	1.81.0	20.10.23 (Docker CE)	Publik
Windows_server-2019-Inggris-Core-ECS_Optimized-2024.01.09	1.79.2	20.10.23 (Docker CE)	Publik
Windows_server-2019-Inggris-Core-ECS_Optimized-2023.12.12	1.79.1	20.10.23 (Docker CE)	Publik
Windows_server-2019-Inggris-Core-ECS_Optimized-2023.11.14	1.79.0	20.10.23 (Docker CE)	Publik
Windows_server-2019-Inggris-Core-ECS_Optimized-2023.10.11	1.77.0	20.10.21 (Docker CE)	Publik
Windows_server-2019-Inggris-Core-ECS_Optimized-2023.09.15	1.75.3	20.10.21 (Docker CE)	Privat
Windows_server-2019-Inggris-Core-ECS	1.74.1	20.10.21 (Docker CE)	Privat

AMI Inti Windows Server 2019 yang dioptimalkan Amazon ECS	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
_Optimized-2023.08.09			
Windows_server-2019-Inggris-Core-ECS_Optimized-2023.07.11	1.73.1	20.10.21 (Docker CE)	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2023.06.13	1.72.0	20.10.21 (Docker CE)	Privat
Windows_server-2019-Inggris-core-ECS_Optimized-2023.05.18	1.71.1	20.10.21 (Docker CE)	Privat
Windows_server-2019-Inggris-core-ECS_Optimized-2023.04.18	1.70.2	20.10.21 (Docker CE)	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2023.03.21	1.69.0	20.10.21 (Docker CE)	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2023.02.21	1.68.2	20.10.21 (Docker CE)	Privat

AMI Inti Windows Server 2019 yang dioptimalkan Amazon ECS	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2019-Inggris-Core-ECS_Optimized-2023.01.11	`1.68.0`	`20.10.21 (Docker CE)`	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2022.12.14	`1.67.2`	`20.10.21 (Docker CE)`	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2022.11.09	`1.65.1`	`20.10.21 (Docker CE)`	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2022.10.12	`1.64.0`	`20.10.17 (Docker CE)`	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2022.09.22	`1.63.1`	`20.10.17 (Docker CE)`	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2022.09.14	`1.62.2`	`20.10.17 (Docker CE)`	Privat

AMI Inti Windows Server 2019 yang dioptimalkan Amazon ECS	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2019-Inggris-Core-ECS_Optimized-2022.08.15	1.62.1	20.10.9	Privat
Windows_server-2019-Inggris-core-ECS_Optimized-2022.07.13	1.61.3	20.10.9	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2022.06.15	1.61.2	20.10.9	Privat
Windows_server-2019-Inggris-core-ECS_Optimized-2022.01.18	1.57.1	20.10.9	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2021.12.16	1.57.1	20.10.7	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2021.11.11	1.57.0	20.10.7	Privat

AMI Inti Windows Server 2019 yang dioptimalkan Amazon ECS	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2019-Inggris-Core-ECS_Optimized-2021.09.23	1.55.3	20.10.7	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2021.08.12	1.55.0	20.10.6	Privat
Windows_server-2019-Inggris-core-ECS_Optimized-2021.07.13	1.54.02	20.10.6	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2021.07.08	1.54.0	20.10.6	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2021.06.11	1.53.0	20.10.5	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2021.05.21	1.52.2	20.10.4	Privat

AMI Inti Windows Server 2019 yang dioptimalkan Amazon ECS	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2019-Inggris-Core-ECS_Optimized-2021.04.14	1.51.0	20.10.0	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2021.03.11	1.50.2	19.03.14	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2021.02.10	1.50.0	19.03.14	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2021.01.13	1.49.0	19.03.14	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2020.11.18	1.48.0	19.03.13	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2020.11.06	1.47.0	19.03.11	Privat

AMI Inti Windows Server 2019 yang dioptimalkan Amazon ECS	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2019-Inggris-Core-ECS_Optimized-2020.10.14	1.45.0	19.03.11	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2020.09.09	1.44.3	19.03.11	Privat
Windows_server-2019-Inggris-core-ECS_Optimized-2020.08.12	1.43.0	19.03.11	Privat
Windows_server-2019-Inggris-core-ECS_Optimized-2020.07.15	1.41.1	19.03.5	Privat
Windows_server-2019-Inggris-core-ECS_Optimized-2020.06.11	1.40.0	19.03.5	Privat
Windows_server-2019-Inggris-core-ECS_Optimized-2020.05.14	1.39.0	19.03.5	Privat

AMI Inti Windows Server 2019 yang dioptimalkan Amazon ECS	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2019-Inggris-Core-ECS_Optimized-2020.01.15	1.35.0	19.03.5	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2019.12.16	1.34.0	19.03.5	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2019.11.25	1.34.0	19.03.4	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2019.11.13	1.32.1	19.03.4	Privat
Windows_server-2019-Inggris-Core-ECS_Optimized-2019.10.09	1.32.0	19.03.2	Privat

Gunakan AWS CLI perintah berikut untuk mengambil AMI Penuh Windows Server 2019 yang dioptimalkan Amazon ECS saat ini.

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Core-ECS_Optimized
```

Windows Server 2016 Full AMI versions

Tabel di bawah ini mencantumkan versi saat ini dan sebelumnya dari Amazon ECS yang dioptimalkan Windows Server 2016 Full AMI dan versi terkait dari agen penampung Amazon ECS dan Docker.

Amazon ECS yang dioptimalkan Windows Server 2016 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2016-Inggris-penuh-ECS_Optimized-2024.02.13	`1.81.0`	`20.10.23 (Docker CE)`	Publik
Windows_server-2016-Inggris-penuh-ECS_Optimized-2024.01.09	`1.79.2`	`20.10.23 (Docker CE)`	Publik
Windows_server-2016-Inggris-penuh-ECS_Optimized-2023.12.12	`1.79.1`	`20.10.23 (Docker CE)`	Publik
Windows_server-2016-Inggris-penuh-ECS_Optimized-2023.11.14	`1.79.0`	`20.10.23 (Docker CE)`	Publik
Windows_server-2016-Inggris-penuh-ECS_Optimized-2023.10.11	`1.77.0`	`20.10.21 (Docker CE)`	Publik
Windows_server-2016-Inggris-penuh-EC	`1.75.3`	`20.10.21 (Docker CE)`	Privat

Amazon ECS yang dioptimalkan Windows Server 2016 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
S_Optimized-2023.09.15			
Windows_server-2016-Inggris-penuh-ECS_Optimized-2023.08.09	1.74.1	20.10.21 (Docker CE)	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2023.07.11	1.73.1	20.10.21 (Docker CE)	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2023.06.13	1.72.0	20.10.21 (Docker CE)	Privat
Windows_server-2016-Inggris-penuh-ECS_dioptimal-2023.05.18	1.71.1	20.10.21 (Docker CE)	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2023.04.18	1.70.2	20.10.21 (Docker CE)	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2023.03.21	1.69.0	20.10.21 (Docker CE)	Privat

Amazon ECS yang dioptimalkan Windows Server 2016 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2016-Inggris-penuh-ECS_Optimized-2023.02.21	1.68.2	20.10.21 (Docker CE)	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2023.01.11	1.68.0	20.10.21 (Docker CE)	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2022.12.14	1.67.2	20.10.21 (Docker CE)	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2022.11.09	1.65.1	20.10.21 (Docker CE)	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2022.10.12	1.64.0	20.10.17 (Docker CE)	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2022.09.22	1.63.1	20.10.17 (Docker CE)	Privat

Amazon ECS yang dioptimalkan Windows Server 2016 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2016-Inggris-penuh-ECS_Optimized-2022.09.14	`1.62.2`	`20.10.17 (Docker CE)`	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2022.08.15	`1.62.1`	`20.10.9`	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2022.07.13	`1.61.3`	`20.10.9`	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2022.06.15	`1.61.2`	`20.10.9`	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2022.01.18	`1.57.1`	`20.10.9`	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2021.12.16	`1.57.1`	`20.10.7`	Privat

Amazon ECS yang dioptimalkan Windows Server 2016 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2016-Inggris-penuh-ECS_Optimized-2021.11.11	`1.57.0`	`20.10.7`	Privat
Windows_server-2016-Inggris-penuh-ECS_dioptimal-2021.09.23	`1.55.3`	`20.10.7`	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2021.08.12	`1.55.0`	`20.10.6`	Privat
Windows_server-2016-Inggris-penuh-ECS_dioptimal-2021.07.13	`1.54.02`	`20.10.6`	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2021.07.08	`1.54.0`	`20.10.5`	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2021.06.11	`1.53.0`	`20.10.5`	Privat

Amazon ECS yang dioptimalkan Windows Server 2016 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2016-Inggris-penuh-ECS_Optimized-2021.05.21	1.52.2	20.10.4	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2021.04.14	1.51.0	20.10.0	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2021.03.11	1.50.2	19.03.14	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2021.02.10	1.50.0	19.03.14	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2021.01.13	1.49.0	19.03.14	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2020.11.18	1.48.0	19.03.13	Privat

Amazon ECS yang dioptimalkan Windows Server 2016 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2016-Inggris-penuh-ECS_Optimized-2020.11.06	`1.47.0`	`19.03.11`	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2020.10.14	`1.45.0`	`19.03.12`	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2020.09.09	`1.44.3`	`19.03.11`	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2020.08.12	`1.43.0`	`19.03.11`	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2020.07.15	`1.41.1`	`19.03.5`	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2020.06.11	`1.40.0`	`19.03.5`	Privat

Amazon ECS yang dioptimalkan Windows Server 2016 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2016-Inggris-penuh-ECS_Optimized-2020.05.14	1.39.0	19.03.5	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2020.01.15	1.35.0	19.03.5	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2019.12.16	1.34.0	19.03.5	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2019.11.25	1.34.0	19.03.4	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2019.11.13	1.32.1	19.03.4	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2019.10.09	1.32.0	19.03.2	Privat

Amazon ECS yang dioptimalkan Windows Server 2016 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2016-Inggris-penuh-ECS_Optimized-2019.09.11	1.30.0	19.03.1	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2019.08.16	1.29.1	19.03.1	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2019.07.19	1.29.0	18.09.8	Privat
Windows_server-2016-Inggris-penuh-ECS_Optimized-2019.03.07	1.26.0	18.03.1	Privat

Gunakan AMI Penuh Windows Server 2016 yang dioptimalkan AWS CLI Amazon ECS berikut ini.

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2016-English-Full-ECS_Optimized
```

Membangun Windows AMI Amazon ECS Anda sendiri yang dioptimalkan

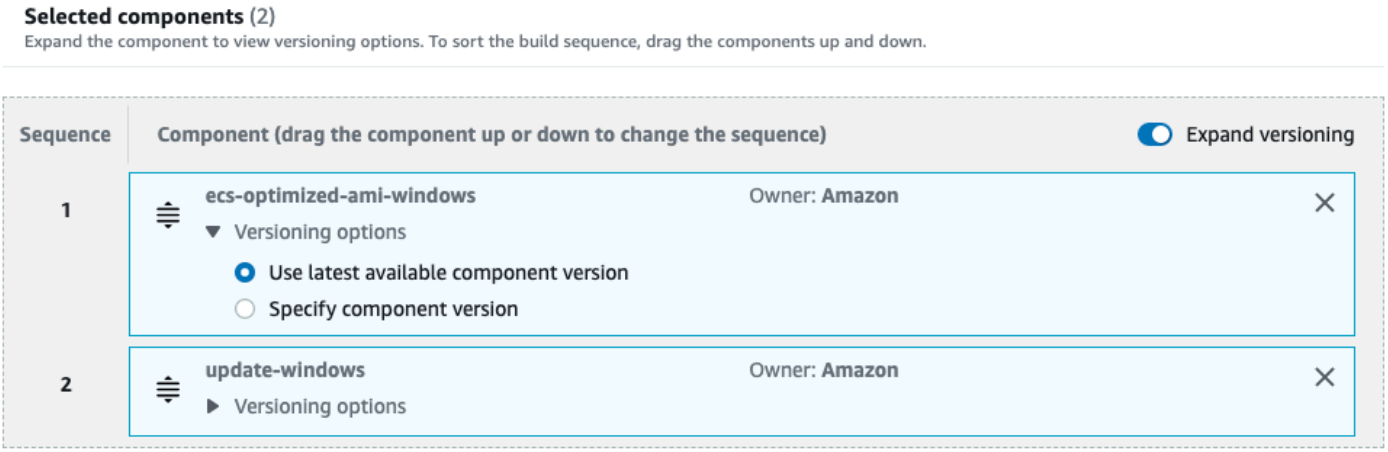
Gunakan EC2 Image Builder untuk membuat Windows AMI Amazon ECS kustom Anda sendiri yang dioptimalkan. Ini membuatnya mudah untuk menggunakan AMI Windows dengan lisensi Anda sendiri di Amazon ECS. Amazon ECS menyediakan komponen Image Builder terkelola yang menyediakan konfigurasi sistem yang diperlukan untuk menjalankan instance Windows untuk meng-host container Anda. Setiap komponen yang dikelola Amazon ECS menyertakan agen kontainer tertentu dan

versi Docker. Anda dapat menyesuaikan gambar untuk menggunakan komponen terkelola Amazon ECS terbaru, atau jika agen penampung atau versi Docker yang lebih lama diperlukan, Anda dapat menentukan komponen yang berbeda.

Untuk panduan lengkap menggunakan EC2 Image Builder, lihat Memulai EC2 Image Builder di Panduan Pengguna EC2 Image Builder.

Saat membuat Windows AMI Amazon ECS Anda sendiri yang dioptimalkan menggunakan EC2 Image Builder, Anda membuat resep gambar. Formula citra Anda harus memenuhi persyaratan berikut:

- Gambar Sumber harus didasarkan pada Windows Server 2016 Full, Windows Server 2019 Core, Windows Server 2019 Full, Windows Server 2022 Core, atau Windows Server 2022 Full. Sistem operasi Windows lain tidak didukung dan mungkin tidak kompatibel dengan komponen.
- Ketika menentukan komponen Pembangunan, maka diperlukan komponen `ecs-optimized-ami-windows`. Komponen `update-windows` disarankan, untuk menjamin bahwa citra memiliki pembaruan keamanan terbaru.



  Untuk menentukan versi komponen yang berbeda, perluas menu Opsi versioning dan tentukan versi komponen yang ingin Anda gunakan. Untuk informasi selengkapnya, lihat Membuat daftar versi komponen `ecs-optimized-ami-windows`.

Membuat daftar versi komponen **`ecs-optimized-ami-windows`**

Saat membuat resep EC2 Image Builder dan menentukan `ecs-optimized-ami-windows` komponen, Anda dapat menggunakan opsi default atau Anda dapat menentukan versi komponen tertentu. Untuk menentukan versi komponen apa yang tersedia, bersama dengan agen kontainer

Amazon ECS dan versi Docker yang terdapat dalam komponen, Anda dapat menggunakan. AWS Management Console

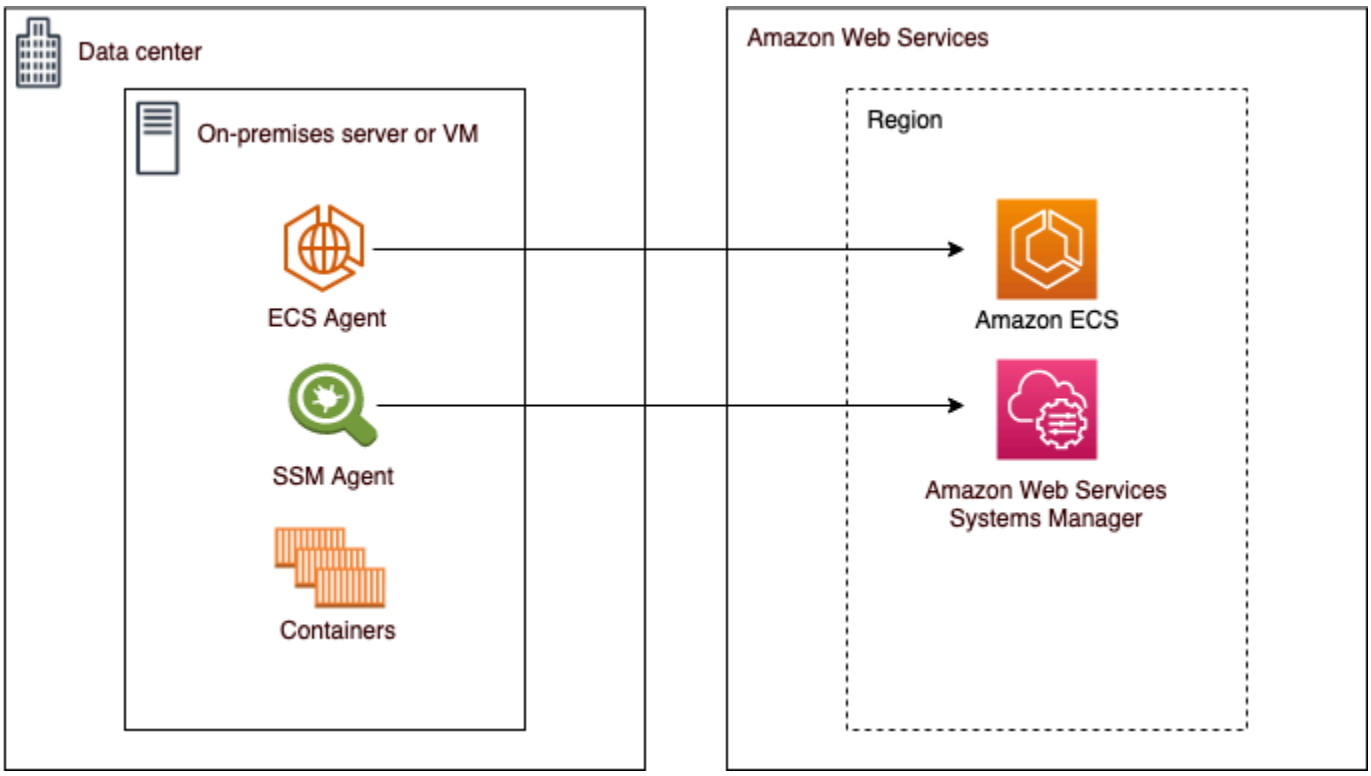
Untuk mendaftarkan versi komponen **`ecs-optimized-ami-windows`** yang tersedia

1. Buka konsol EC2 Image Builder di https://console.aws.amazon.com/imagebuilder/.
2. Pada bilah navigasi, pilih Wilayah yang sedang membangun citra Anda.
3. Di panel navigasi, di bawah menu konfigurasi Tersimpan, pilih Komponen.
4. Pada halaman Komponen, di bilah pencarian, ketik `ecs-optimized-ami-windows` dan tarik menu kualifikasi ke bawah lalu pilih Quick start (Amazon-terkelola).
5. Gunakan kolom Deskripsi untuk menentukan versi komponen dengan agen penampung Amazon ECS dan versi Docker yang dibutuhkan gambar Anda.

Instans eksternal (Amazon ECS Anywhere)

Amazon ECS Anywhere menyediakan dukungan untuk mendaftarkan instans eksternal seperti server lokal atau mesin virtual (VM), ke kluster Amazon ECS Anda. Instans eksternal dioptimalkan guna menjalankan aplikasi yang menghasilkan lalu lintas keluar atau data proses. Jika aplikasi Anda memerlukan lalu lintas masuk, kurangnya dukungan Elastic Load Balancing membuat menjalankan beban kerja ini menjadi kurang efisien. Amazon ECS menambahkan jenis `EXTERNAL` peluncuran baru yang dapat Anda gunakan untuk membuat layanan atau menjalankan tugas pada instans eksternal Anda.

Berikut ini memberikan ikhtisar arsitektur sistem tingkat tinggi Amazon ECS Anywhere.



Topik

- Sistem pengoperasian dan arsitektur sistem yang didukung
- Pertimbangan
- Pendaftaran instans eksternal ke sebuah klaster
- Membatalkan pendaftaran instans eksternal
- Memperbarui AWS Systems Manager agen kontainer Agen dan Amazon ECS pada instans eksternal

Sistem pengoperasian dan arsitektur sistem yang didukung

Berikut ini merupakan daftar sistem pengoperasian dan arsitektur sistem yang didukung.

- Amazon Linux 2
- CentOS 7
- CentOS Aliran 8
- RHEL 7, RHEL 8 - Baik repositori paket terbuka Docker atau RHEL tidak mendukung penginstalan Docker secara native di RHEL. Anda harus memastikan bahwa Docker telah diinstal sebelum menjalankan skrip penginstalan yang dijelaskan dalam dokumen ini.

- Fedora 32, Fedora 33
- openSUSE Tumbleweed
- Ubuntu 18, Ubuntu 20, Ubuntu 22
- Debian 10

> **Important**
>
> Debian 9 Long Term Support (dukungan LTS) berakhir pada 30 Juni 2022 dan tidak lagi didukung oleh Amazon ECS Anywhere.

- SUSE Enterprise Server 15
- Arsitektur CPU `x86_64` dan `ARM64` didukung.
- Versi sistem operasi Windows berikut didukung:
  - Windows Server 2022
  - Windows Server 2019
  - Windows Server 2016
  - Windows Server 20H2

Pertimbangan

Sebelum Anda mulai menggunakan instance eksternal, perhatikan pertimbangan berikut.

- Anda dapat mendaftarkan instans eksternal ke dalam satu klaster dalam satu waktu. Untuk instruksi tentang cara mendaftarkan instans eksternal dengan klaster yang berbeda, lihat Membatalkan pendaftaran instans eksternal.
- Instance eksternal Anda memerlukan peran IAM yang memungkinkan mereka berkomunikasi dengan AWS API. Untuk informasi selengkapnya, lihat Peran IAM ECS Anywhere.
- Instance eksternal Anda seharusnya tidak memiliki rantai kredensi instans yang telah dikonfigurasi yang ditentukan secara lokal karena ini akan mengganggu skrip pendaftaran.
- Untuk mengirim log kontainer ke CloudWatch Log, pastikan Anda membuat dan menentukan peran IAM eksekusi tugas dalam definisi tugas Anda.
- Ketika instans eksternal didaftarkan ke sebuah klaster, atribut `ecs.capability.external` dikaitkan dengan instans tersebut. Atribut ini mengidentifikasikan instans tersebut sebagai instans eksternal. Anda dapat menambahkan atribut khusus ke instans eksternal untuk digunakan sebagai batasan penempatan tugas. Untuk informasi selengkapnya, lihat Atribut kustom.

- Anda dapat menambahkan tanda sumber daya ke instans eksternal Anda. Untuk informasi selengkapnya, lihat Menandai instance kontainer eksternal.
- ECS Exec didukung pada instans eksternal. Untuk informasi selengkapnya, lihat Pantau wadah Amazon ECS dengan ECS Exec.
- Berikut ini merupakan pertimbangan tambahan yang khusus untuk jaringan dengan instans eksternal Anda. Untuk informasi selengkapnya, lihat Jaringan dengan ECS Anywhere.
  - Penyeimbangan beban layanan tidak didukung.
  - Penemuan layanan tidak didukung.
  - Tugas yang berjalan pada instans eksternal harus menggunakan mode jaringan `bridge`, `host`, atau `none`. Mode jaringan `awsvpc` tidak didukung.
  - Ada domain layanan Amazon ECS di setiap AWS Wilayah. Domain layanan ini harus diizinkan untuk mengirim lalu lintas ke instans eksternal Anda.
  - Agen SSM yang diinstal pada instans eksternal Anda mempertahankan kredensional IAM yang diputar setiap 30 menit menggunakan sidik jari perangkat keras. Jika instans eksternal Anda kehilangan koneksi AWS, Agen SSM secara otomatis menyegarkan kredensialnya setelah koneksi dibuat kembali. Untuk informasi selengkapnya, lihat Memvalidasi server lokal dan mesin virtual menggunakan sidik jari perangkat keras di AWS Systems Manager Panduan Pengguna.
- API `UpdateContainerAgent` tidak didukung. Untuk petunjuk tentang cara memperbarui Agen SSM atau agen Amazon ECS pada instans eksternal Anda, lihat. Memperbarui AWS Systems Manager agen kontainer Agen dan Amazon ECS pada instans eksternal
- Penyedia kapasitas Amazon ECS tidak didukung. Untuk membuat layanan atau menjalankan tugas mandiri pada instans eksternal Anda, gunakan tipe peluncuran `EXTERNAL`.
- SELinux tidak didukung.
- Menggunakan volume Amazon EFS atau menentukan `EFSVolumeConfiguration` tidak didukung.
- Integrasi dengan App Mesh tidak didukung.
- Jika Anda menggunakan konsol untuk membuat definisi tugas instance eksternal, Anda harus membuat definisi tugas dengan editor JSON konsol.
- Ketika Anda menjalankan ECS Anywhere di Windows, Anda harus menggunakan lisensi Windows Anda sendiri pada infrastruktur lokal.
- Bila Anda menggunakan AMI yang tidak dioptimalkan ECS Amazon, jalankan perintah berikut pada instance container eksternal untuk mengonfigurasi aturan guna menggunakan peran IAM untuk

tugas. Untuk informasi selengkapnya, lihat Menggunakan peran IAM tugas di Amazon EC2 atau instans eksternal.

```
$ sysctl -w net.ipv4.conf.all.route_localnet=1
$ iptables -t nat -A PREROUTING -p tcp -d 169.254.170.2 --dport 80 -j DNAT --to-
destination 127.0.0.1:51679
$ iptables -t nat -A OUTPUT -d 169.254.170.2 -p tcp -m tcp --dport 80 -j REDIRECT --
to-ports 51679
```

Jaringan dengan ECS Anywhere

Instans eksternal Amazon ECS dioptimalkan untuk menjalankan aplikasi yang menghasilkan lalu lintas keluar atau data proses. Jika aplikasi Anda memerlukan lalu lintas masuk, seperti layanan web, kurangnya dukungan Elastic Load Balancing membuat menjalankan beban kerja ini kurang efisien karena tidak ada dukungan untuk menempatkan beban kerja ini di belakang penyeimbang beban.

Berikut ini merupakan pertimbangan tambahan yang khusus untuk jaringan dengan instans eksternal Anda.

- Penyeimbangan beban layanan tidak didukung.
- Penemuan layanan tidak didukung.
- Tugas Linux yang berjalan pada instance eksternal harus menggunakan`bridge`,`host`, atau mode `none` jaringan. Mode jaringan `awsvpc` tidak didukung.

  Untuk informasi selengkapnya tentang setiap mode jaringan, lihat Memilih mode jaringan di Panduan Praktik Terbaik Amazon ECS.
- Tugas Windows yang berjalan pada instance eksternal harus menggunakan mode `default` jaringan.
- Ada domain layanan Amazon ECS di setiap Wilayah dan harus diizinkan untuk mengirim lalu lintas ke instans eksternal Anda.
- Agen SSM yang diinstal pada instans eksternal Anda mempertahankan kredensional IAM yang diputar setiap 30 menit menggunakan sidik jari perangkat keras. Jika instans eksternal Anda kehilangan koneksi AWS, Agen SSM secara otomatis menyegarkan kredensialnya setelah koneksi dibuat kembali. Untuk informasi selengkapnya, lihat Memvalidasi server lokal dan mesin virtual menggunakan sidik jari perangkat keras di AWS Systems Manager Panduan Pengguna.

Domain berikut digunakan untuk komunikasi antara layanan Amazon ECS dan agen Amazon ECS yang diinstal pada instans eksternal Anda. Pastikan lalu lintas diizinkan dan resolusi DNS berfungsi. Untuk setiap titik akhir, *wilayah* mewakili pengenal Wilayah untuk AWS Wilayah yang didukung oleh Amazon ECS, seperti `us-east-2` untuk Wilayah AS Timur (Ohio). Titik akhir untuk semua Wilayah yang Anda gunakan harus diizinkan. Untuk titik akhir `ecs-a` dan `ecs-t`, Anda harus menyertakan tanda bintang (misalnya, `ecs-a-*`).

- `ecs-a-*.region.amazonaws.com`Endpoint ini digunakan saat mengelola tugas.
- `ecs-t-*.region.amazonaws.com`— Titik akhir ini digunakan untuk mengelola metrik tugas dan kontainer.
- `ecs.region.amazonaws.com`- Ini adalah titik akhir layanan untuk Amazon ECS.
- `ssm.region.amazonaws.com` — Ini adalah titik akhir layanan untuk AWS Systems Manager.
- `ec2messages.region.amazonaws.com`— Ini adalah titik akhir layanan yang AWS Systems Manager digunakan untuk berkomunikasi antara agen Systems Manager dan layanan Systems Manager di cloud.
- `ssmmessages.region.amazonaws.com`— Ini adalah titik akhir layanan yang diperlukan untuk membuat dan menghapus saluran sesi dengan layanan Session Manager di cloud.
- Jika tugas Anda memerlukan komunikasi dengan AWS layanan lain, pastikan titik akhir layanan tersebut diizinkan. Contoh aplikasi termasuk menggunakan Amazon ECR untuk menarik gambar kontainer atau menggunakan CloudWatch untuk CloudWatch Log. Untuk informasi selengkapnya, lihat Titik akhir layanan di Referensi Umum AWS .

Amazon FSx for Windows File Server dengan ECS Di Mana Saja

Untuk menggunakan instans eksternal Amazon ECS, Anda harus membuat sambungan antara pusat data lokal dan. Amazon FSx for Windows File Server AWS Cloud Untuk informasi tentang opsi untuk menghubungkan jaringan ke VPC, lihat Opsi Konektivitas Amazon Virtual Private Cloud.

GMsa dengan ECS Di Mana Saja

Kasus penggunaan berikut didukung untuk ECS Anywhere.

- Direktori Aktif ada di AWS Cloud - Untuk konfigurasi ini, Anda membuat sambungan antara jaringan lokal dan AWS Cloud menggunakan AWS Direct Connect koneksi. Untuk informasi tentang cara membuat sambungan, lihat Opsi Konektivitas Amazon Virtual Private Cloud. Anda membuat Direktori Aktif di AWS Cloud. Untuk informasi tentang cara memulai AWS Directory

Service, lihat Menyiapkan AWS Directory Service di Panduan AWS Directory Service Administrasi. Anda kemudian dapat menggabungkan instance eksternal Anda ke domain menggunakan AWS Direct Connect koneksi. Untuk informasi tentang bekerja dengan GMSA dengan Amazon ECS, lihat. the section called “Menggunakan GMSAs untuk Windows Container di Amazon EC2”

- Active Directory berada di pusat data lokal. - Untuk konfigurasi ini, Anda menggabungkan instance eksternal Anda ke Active Directory lokal. Anda kemudian menggunakan kredensil yang tersedia secara lokal saat menjalankan tugas Amazon ECS.

Pendaftaran instans eksternal ke sebuah klaster

Untuk setiap instans eksternal yang Anda daftarkan dengan kluster Amazon ECS, instans tersebut harus memiliki Agen SSM, agen penampung Amazon ECS, dan Docker yang diinstal. Untuk mendaftarkan instans eksternal ke kluster Amazon ECS, instans harus didaftarkan terlebih dahulu sebagai instans AWS Systems Manager terkelola. Anda dapat membuat skrip instalasi dalam beberapa klik pada konsol Amazon ECS. Skrip instalasi mencakup kunci aktivasi Systems Manager dan perintah untuk menginstal masing-masing agen dan Docker yang diperlukan. Skrip penginstalan harus dijalankan di server on-premise atau VM Anda untuk menyelesaikan langkah-langkah instalasi dan pendaftaran.

> **Note**
>
> Sebelum mendaftarkan instance eksternal Linux Anda dengan cluster, buat `/etc/ecs/ecs.config` file pada instance eksternal Anda dan tambahkan parameter konfigurasi agen kontainer apa pun yang Anda inginkan. Anda tidak dapat melakukan hal tersebut setelah mendaftarkan instans eksternal ke sebuah klaster. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.

AWS Management Console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.
3. Di panel navigasi, pilih Klaster.
4. Pada halaman Klaster, pilih sebuah klaster tempat Anda mendaftarkan instans eksternal.
5. Pada halaman Cluster: ***name***, pilih tab Infrastructure.
6. Pada halaman Daftar instans eksternal, selesaikan langkah-langkah berikut.

a. Untuk Durasi kunci aktivasi (dalam hari), masukkan jumlah hari untuk kunci aktivasi tetap dalam status aktif. Setelah jumlah hari yang Anda masukkan terlewati, kunci tidak lagi berfungsi saat mendaftarkan instans eksternal.

b. Untuk Jumlah instans, masukkan jumlah instans eksternal yang ingin Anda daftarkan ke klaster Anda dengan kunci aktivasi.

c. Untuk peran Instance, pilih peran IAM untuk dikaitkan dengan instans eksternal Anda. Jika peran belum dibuat, pilih Buat peran baru agar Amazon ECS membuat peran atas nama Anda. Untuk informasi selengkapnya tentang izin IAM yang diperlukan untuk instans eksternal Anda, lihat. Peran IAM ECS Anywhere

d. Salin perintah pendaftaran. Perintah ini harus dijalankan pada setiap instans eksternal yang ingin Anda daftarkan ke klaster.

> **Important**
>
> Bagian bash skrip harus dijalankan sebagai root. Jika perintah tidak dijalankan sebagai root, maka akan terjadi kesalahan.

e. Pilih Tutup.

AWS CLI for Linux operating systems

1. Buat pasangan aktivasi Systems Manager. Ini digunakan untuk aktivasi instans terkelola Systems Manager. Output meliputi `ActivationId` dan `ActivationCode`. Kedua output tersebut akan digunakan di langkah berikutnya. Pastikan Anda menentukan peran IAM ECS Anywhere yang Anda buat. Untuk informasi selengkapnya, lihat Peran IAM ECS Anywhere.

```
aws ssm create-activation --iam-role ecsAnywhereRole | tee ssm-activation.json
```

2. Di server on-premise atau mesin virtual (VM) Anda, unduh skrip instalasi.

```
curl --proto "https" -o "/tmp/ecs-anywhere-install.sh" "https://amazon-ecs-agent.s3.amazonaws.com/ecs-anywhere-install-latest.sh"
```

3. (Opsional) Di server on-premise atau mesin virtual (VM) Anda, lakukan langkah-langkah berikut untuk memverifikasi penulisan instalasi menggunakan file standar penulisan.

a. Unduh dan instal GnuPG. Untuk informasi selengkapnya tentang GnuPG, lihat situs web GnuPG. Untuk sistem Linux, instal gpg menggunakan pengelola paket pada tipe instans VM Linux Anda.

b. Ambil kunci publik Amazon ECS PGP.

```
gpg --keyserver hkp://keys.gnupg.net:80 --recv BCE9D9A42D51784F
```

c. Unduh tanda tangan skrip instalasi. Tanda tangan tersebut merupakan tandatangan PGP ascii terpisah yang disimpan dalam sebuah file dengan ekstensi `.asc`.

```
curl --proto "https" -o "/tmp/ecs-anywhere-install.sh.asc" "https://amazon-
ecs-agent.s3.amazonaws.com/ecs-anywhere-install-latest.sh.asc"
```

d. Verifikasi file skrip instalasi menggunakan kunci.

```
gpg --verify /tmp/ecs-anywhere-install.sh.asc /tmp/ecs-anywhere-install.sh
```

Berikut adalah output yang diharapkan.

```
gpg: Signature made Tue 25 May 2021 07:16:29 PM UTC
gpg:                using RSA key 50DECCC4710E61AF
gpg: Good signature from "Amazon ECS <ecs-security@amazon.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the
 owner.
Primary key fingerprint: F34C 3DDA E729 26B0 79BE  AEC6 BCE9 D9A4 2D51 784F
     Subkey fingerprint: D64B B6F9 0CF3 77E9 B5FB  346F 50DE CCC4 710E 61AF
```

4. Di server on-premise atau mesin virtual (VM) Anda, jalankan skrip instalasi. Tentukan nama cluster, Wilayah, dan ID aktivasi Systems Manager dan kode aktivasi dari langkah pertama.

```
sudo bash /tmp/ecs-anywhere-install.sh \
    --region $REGION \
    --cluster $CLUSTER_NAME \
    --activation-id $ACTIVATION_ID \
    --activation-code $ACTIVATION_CODE
```

Untuk server lokal atau mesin virtual (VM) yang memiliki driver NVIDIA diinstal untuk beban kerja GPU, Anda harus menambahkan `--enable-gpu` tanda ke skrip instalasi. Ketika tanda

ini ditentukan, skrip penginstalan memverifikasi bahwa driver NVIDIA sedang berjalan dan kemudian menambahkan variabel konfigurasi yang diperlukan untuk menjalankan tugas Amazon ECS Anda. Untuk informasi selengkapnya tentang menjalankan beban kerja GPU dan menentukan persyaratan GPU dalam definisi tugas, lihat. Menentukan GPU dalam ketentuan tugas Anda

```
sudo bash /tmp/ecs-anywhere-install.sh \
    --region $REGION \
    --cluster $CLUSTER_NAME \
    --activation-id $ACTIVATION_ID \
    --activation-code $ACTIVATION_CODE \
    --enable-gpu
```

Gunakan langkah-langkah berikut untuk mendaftarkan instans eksternal yang sudah ada dengan klaster yang berbeda.

Untuk mendaftarkan instans eksternal yang sudah ada dengan klaster yang berbeda

1. Hentikan agen kontainer Amazon ECS.

```
sudo systemctl stop ecs.service
```

2. Edit file `/etc/ecs/ecs.config` dan pada baris `ECS_CLUSTER`, pastikan bahwa nama klaster cocok dengan nama klaster untuk mendaftarkan instans eksternal.
3. Hapus data agen Amazon ECS yang ada.

```
sudo rm /var/lib/ecs/data/agent.db
```

4. Mulai agen kontainer Amazon ECS.

```
sudo systemctl start ecs.service
```

AWS CLI for Windows operating systems

1. Buat pasangan aktivasi Systems Manager. Ini digunakan untuk aktivasi instans terkelola Systems Manager. Output meliputi `ActivationId` dan `ActivationCode`. Kedua output tersebut akan digunakan di langkah berikutnya. Pastikan Anda menentukan peran IAM ECS Anywhere yang Anda buat. Untuk informasi selengkapnya, lihat Peran IAM ECS Anywhere.

```
aws ssm create-activation --iam-role ecsAnywhereRole | tee ssm-activation.json
```

2. Di server on-premise atau mesin virtual (VM) Anda, unduh skrip instalasi.

```
Invoke-RestMethod -URI "https://amazon-ecs-agent.s3.amazonaws.com/ecs-anywhere-
install.ps1" -OutFile “ecs-anywhere-install.ps1”
```

3. (Opsional) Skrip Powershell ditandatangani oleh Amazon dan oleh karena itu, Windows secara otomatis melakukan validasi sertifikat pada saat yang sama. Anda tidak perlu melakukan validasi manual apa pun.

   Untuk memverifikasi sertifikat secara manual, klik kanan pada file, navigasikan ke properti dan gunakan tab Tanda Tangan Digital untuk mendapatkan detail lebih lanjut.

   Opsi ini hanya tersedia ketika host memiliki sertifikat di toko sertifikat.

   Verifikasi harus mengembalikan informasi yang serupa dengan yang berikut:

```
# Verification (PowerShell)
Get-AuthenticodeSignature -FilePath .\ecs-anywhere-install.ps1

SignerCertificate                         Status      Path
-----------------                         ------      ----
EXAMPLECERTIFICATE                        Valid       ecs-anywhere-install.ps1

...

Subject              : CN="Amazon Web Services, Inc.",...

----
```

4. Di server on-premise atau mesin virtual (VM) Anda, jalankan skrip instalasi. Tentukan nama cluster, Wilayah, dan ID aktivasi Systems Manager dan kode aktivasi dari langkah pertama.

```
.\ecs-anywhere-install.ps1 -Region $Region -Cluster $Cluster -
ActivationID $ActivationID -ActivationCode $ActivationCode
```

5. Verifikasi bahwa agen penampung Amazon ECS sedang berjalan.

```
Get-Service AmazonECS
```

```
Status   Name               DisplayName
------   ----               -----------
Running  AmazonECS          Amazon ECS
```

Gunakan langkah-langkah berikut untuk mendaftarkan instans eksternal yang sudah ada dengan klaster yang berbeda.

Untuk mendaftarkan instans eksternal yang sudah ada dengan klaster yang berbeda

1. Hentikan agen kontainer Amazon ECS.

   ```
   Stop-Service AmazonECS
   ```

2. Ubah ECS_CLUSTER parameter sehingga nama cluster cocok dengan nama cluster untuk mendaftarkan instance eksternal.

   ```
   [Environment]::SetEnvironmentVariable("ECS_CLUSTER", $ECSCluster,
    [System.EnvironmentVariableTarget]::Machine)
   ```

3. Hapus data agen Amazon ECS yang ada.

   ```
   Remove-Item -Recurse -Force $env:ProgramData\Amazon\ECS\data\*
   ```

4. Mulai agen kontainer Amazon ECS.

   ```
   Start-Service AmazonECS
   ```

Ini AWS CLI dapat digunakan untuk membuat aktivasi Systems Manager sebelum menjalankan skrip instalasi untuk menyelesaikan proses pendaftaran instance eksternal.

Membatalkan pendaftaran instans eksternal

Sebaiknya, setelah Anda selesai menggunakan instans eksternal, Anda membatalkan pendaftaran instans dari Amazon ECS dan. AWS Systems Manager Setelah pembatalan pendaftaran, instans eksternal tidak lagi dapat menerima tugas baru.

Jika Anda memiliki tugas yang berjalan di instans kontainer saat membatalkan pendaftarannya, maka tugas tersebut akan tetap berjalan hingga berhenti melalui beberapa cara lain. Namun, tugas-tugas ini tidak lagi dipantau atau diperhitungkan oleh Amazon ECS. Jika tugas-tugas ini pada instans

eksternal Anda merupakan bagian dari layanan Amazon ECS, maka penjadwal layanan memulai salinan lain dari tugas itu, pada instance yang berbeda, jika memungkinkan.

Untuk mendaftarkan instans eksternal ke klaster baru, setelah instans eksternal dideregistrasi dari Amazon ECS dan Systems Manager, Anda dapat membersihkan AWS sumber daya yang tersisa pada instans dan mendaftarkannya ke cluster baru.

AWS Management Console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah tempat instans eksternal Anda didaftarkan.
3. Di panel navigasi, pilih Klaster, kemudian pilih klaster yang meng-host instans eksternal.
4. Pada halaman Cluster: ***name***, pilih tab Infrastructure.
5. Di bawah Instance Container, pilih ID instans eksternal untuk membatalkan pendaftaran. Anda dialihkan ke halaman detail instans kontainer.
6. Pada halaman Instans Kontainer: ***id***, pilih Batalkan pendaftaran.
7. Tinjau pesan pembatalan pendaftaran. Pilih Deregister from AWS Systems Manager untuk juga membatalkan pendaftaran instance eksternal sebagai instance terkelola Systems Manager. Pilih Batalkan pendaftaran.

   > **Note**
   >
   > Anda dapat membatalkan pendaftaran instans eksternal sebagai instans terkelola Systems Manager di konsol Systems Manager. Untuk petunjuk, lihat Membatalkan pendaftaran instans terkelola di Panduan Pengguna.AWS Systems Manager

8. Setelah membatalkan pendaftaran instans, bersihkan AWS sumber daya di server lokal atau VM Anda.

Sistem operasi	Langkah-langkah	
Linux	a. Hentikan agen kontainer Amazon ECS dan layanan Agen SSM pada instans.	

Sistem operasi	Langkah-langkah	
	`sudo systemctl stop ecs amazon-ssm-agent`	

	b. Hapus paket Amazon ECS dan Systems Manager.
	
	Untuk CentOS 7, CentOS 8, dan RHEL 7
	
	`sudo yum remove -y amazon-ecs-init amazon-ssm-agent`
	
	Untuk SUSE Enterprise Server 15
	
	`sudo zypper remove -y amazon-ecs-init amazon-ssm-agent`
	
	Untuk Debian dan Ubuntu
	
	`sudo apt remove -y amazon-ecs-init amazon-ssm-agent`	

	c. Hapus direktori yang tersisa. `sudo rm -rf /var/lib/ecs /etc/ecs / var/lib/amazon/ss`	

Sistem operasi	Langkah-langkah
	`m /var/log/ecs / var/log/amazon/ssm`

Windows	a. Hentikan agen kontainer Amazon ECS dan layanan Agen SSM pada instans.
	`Stop-Service AmazonECS`
	`Stop-Service AmazonSSMAgent`
	b. Hapus paket Amazon ECS.
	`.\ecs-anywhere-install.ps1 -Uninstall`

AWS CLI

1. Anda memerlukan ID instance dan ARN instance kontainer untuk membatalkan pendaftaran instance container. Jika Anda tidak memiliki nilai tesis, jalankan comand berikut

   Jalankan perintah berikutuntuk mendapatkan ID instance.

   Anda menggunakan instance ID (`instanceID`) untuk mendapatkan instance kontainer ARN ()`containerInstanceARN`.

   ```
   instanceId=$(aws ssm describe-instance-information --region "{{ region }}" |
    jq ".InstanceInformationList[] |select(.IPAddress==\"{{ IPv4 Address }}\")
    | .InstanceId" | tr -d'"'
   ```

   Jalankan perintah berikut.

Anda menggunakan parameter `containerInstanceArn` as a dalam perintah untuk membatalkan pendaftaran instance (). `deregister-container-instance`

```
instances=$(aws ecs list-container-instances --cluster "{{ cluster }}" --region
 "{{ region }}" | jq -c '.containerInstanceArns')
containerInstanceArn=$(aws ecs describe-container-instances --cluster
 "{{ cluster }}" --region "{{ region }}" --container-instances $instances
 | jq ".containerInstances[] | select(.ec2InstanceId==\"{{ instanceId }}\")
 | .containerInstanceArn" | tr -d '"')
```

2. Jalankan perintah berikut untuk menguras instance.

```
aws ecs update-container-instances-state --cluster "{{ cluster }}" --region
 "{{ region }}" --container-instances "{{ containerInstanceArn }}" --status
 DRAINING
```

3. Setelah instance container selesai menguras, jalankan perintah berikut untuk membatalkan pendaftaran instance.

```
aws ecs deregister-container-instance --cluster "{{ cluster }}" --region
 "{{ region }}" --container-instance "{{ containerInstanceArn }}"
```

4. Jalankan perintah berikut untuk menghapus instance container dari SSM.

```
aws ssm deregister-managed-instance --region "{{ region }}" --instance-id
 "{{ instanceId }}"
```

5. Setelah membatalkan pendaftaran instans, bersihkan AWS sumber daya di server lokal atau VM Anda.

Sistem operasi	Langkah-langkah	
Linux	a. Hentikan agen kontainer Amazon ECS dan layanan Agen SSM pada instans.	

Sistem operasi	Langkah-langkah	

	`sudo systemctl stop ecs amazon-ssm-agent`
	b. Hapus paket Amazon ECS dan Systems Manager.
	`sudo (yum/apt/zypper) remove amazon-ecs-init amazon-ssm-agent`
	c. Hapus direktori yang tersisa.
	`sudo rm -rf /var/lib/ecs /etc/ecs /var/lib/amazon/ssm /var/log/ecs /var/log/amazon/ssm`	

Sistem operasi	Langkah-langkah	

Windows	a. Hentikan agen kontainer Amazon ECS dan layanan Agen SSM pada instans.
	`Stop-Service AmazonECS`
	`Stop-Service AmazonSSMAgent`
	b. Hapus paket Amazon ECS.
	`.\ecs-anywhere-install.ps1 -Uninstall`	

Memperbarui AWS Systems Manager agen kontainer Agen dan Amazon ECS pada instans eksternal

Server lokal atau VM Anda harus menjalankan AWS Systems Manager Agen (Agen SSM) dan agen penampung Amazon ECS saat menjalankan beban kerja Amazon ECS. AWS merilis versi baru dari agen ini ketika kemampuan apa pun ditambahkan atau diperbarui. Jika instans eksternal Anda menggunakan versi yang lebih lama dari salah satu agen, maka Anda dapat memperbaruinya menggunakan prosedur berikut.

Memperbarui SSM Agent pada instans eksternal

AWS Systems Manager merekomendasikan agar Anda mengotomatiskan proses memperbarui Agen SSM pada instans Anda. Beberapa metode disediakan untuk mengotomatisasi pembaruan. Untuk informasi selengkapnya, lihat Mengotomatiskan pembaruan ke Agen SSM di AWS Systems Manager Panduan Pengguna.

Memperbarui agen Amazon ECS pada instans eksternal

Pada instans eksternal Anda, agen penampung Amazon ECS diperbarui dengan memutakhirkan paket. `ecs-init` Memperbarui agen Amazon ECS tidak mengganggu tugas atau layanan yang sedang berjalan. Amazon ECS menyediakan `ecs-init` paket dan file tanda tangan dalam bucket Amazon S3 di setiap Wilayah. Dimulai dengan `ecs-init` versi`1.52.1-1`, Amazon ECS menyediakan `ecs-init` paket terpisah untuk digunakan tergantung pada sistem operasi dan arsitektur sistem yang digunakan instans eksternal Anda.

Gunakan tabel berikut untuk menentukan paket `ecs-init` yang harus Anda unduh berdasarkan sistem pengoperasian dan arsitektur sistem yang digunakan instans eksternal Anda.

> **Note**
>
> Anda dapat menentukan sistem pengoperasian dan sistem arsitektur yang menggunakan instans eksternal Anda dengan menggunakan perintah berikut.
>
> ```
> cat /etc/os-release
> uname -m
> ```

Sistem pengoperasian (arsitektur)	Paket ecs-init
CentOS 7 (x86_64) CentOS 8 (x86_64) SUSE Enterprise Server 15 (x86_64) RHEL 7 (x86_64) RHEL 8 (x86_64)	`amazon-ecs-init-latest.x86_64.rpm`
CentOS 7 (aarch64) CentOS 8 (aarch64) RHEL 7 (aarch64)	`amazon-ecs-init-latest.aarch64.rpm`
Debian 9 (x86_64)	`amazon-ecs-init-latest.amd64.deb`

Sistem pengoperasian (arsitektur)	Paket ecs-init
Debian 10 (x86_64) Ubuntu 18 (x86_64) Ubuntu 20 (x86_64)	
Debian 9 (aarch64) Debian 10 (aarch64) Ubuntu 18 (aarch64) Ubuntu 20 (aarch64)	`amazon-ecs-init-latest.arm64.deb`

Ikuti langkah-langkah ini untuk memperbarui agen Amazon ECS.

Untuk memperbarui agen Amazon ECS

1. Konfirmasikan versi agen Amazon ECS yang sedang Anda jalankan.

   ```
   curl -s 127.0.0.1:51678/v1/metadata | python3 -mjson.tool
   ```

2. Unduh `ecs-init` paket untuk sistem operasi dan arsitektur sistem Anda. Amazon ECS menyediakan file `ecs-init` paket dalam bucket Amazon S3 di setiap Wilayah. Pastikan bahwa Anda mengganti pengidentifikasi *`<region>`* di dalam perintah dengan nama Wilayah (misalnya, `us-west-2`) yang paling dekat dengan Anda secara geografis.

   amazon-ecs-init-latest.x86_64.rpm

   ```
   curl -o amazon-ecs-init.rpm https://s3.<region>.amazonaws.com/amazon-ecs-
   agent-<region>/amazon-ecs-init-latest.x86_64.rpm
   ```

   amazon-ecs-init-latest.aarch64.rpm

   ```
   curl -o amazon-ecs-init.rpm https://s3.<region>.amazonaws.com/amazon-ecs-
   agent-<region>/amazon-ecs-init-latest.aarch64.rpm
   ```

   amazon-ecs-init-latest.amd64.deb

```
curl -o amazon-ecs-init.deb https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.amd64.deb
```

amazon-ecs-init-latest.arm64.deb

```
curl -o amazon-ecs-init.deb https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.arm64.deb
```

3. (Opsional) Verifikasi validitas file paket `ecs-init` menggunakan tanda tangan PGP.

   a. Unduh dan instal GnuPG. Untuk informasi selengkapnya tentang GnuPG, lihat situs web GnuPG. Untuk sistem Linux, instal `gpg` menggunakan pengelola paket pada tipe instans VM Linux Anda.

   b. Ambil kunci publik Amazon ECS PGP.

   ```
   gpg --keyserver hkp://keys.gnupg.net:80 --recv BCE9D9A42D51784F
   ```

   c. Unduh tanda tangan paket `ecs-init`. Tanda tangan adalah tanda tangan PGP terpisah ASCII yang disimpan dalam file dengan ekstensi. `.asc` Amazon ECS menyediakan file tanda tangan dalam bucket Amazon S3 di setiap Wilayah. Pastikan bahwa Anda mengganti pengidentifikasi *`<region>`* di dalam perintah dengan nama Wilayah (misalnya, `us-west-2`) yang paling dekat dengan Anda secara geografis.

   amazon-ecs-init-latest.x86_64.rpm

   ```
   curl -o amazon-ecs-init.rpm.asc https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.x86_64.rpm.asc
   ```

   amazon-ecs-init-latest.aarch64.rpm

   ```
   curl -o amazon-ecs-init.rpm.asc https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.aarch64.rpm.asc
   ```

   amazon-ecs-init-latest.amd64.deb

   ```
   curl -o amazon-ecs-init.deb.asc https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.amd64.deb.asc
   ```

amazon-ecs-init-latest.arm64.deb

```
curl -o amazon-ecs-init.deb.asc https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.arm64.deb.asc
```

d. Verifikasi file paket `ecs-init` menggunakan kunci.

Untuk **`rpm`** paket-paketnya

```
gpg --verify amazon-ecs-init.rpm.asc ./amazon-ecs-init.rpm
```

Untuk **`deb`** paket-paketnya

```
gpg --verify amazon-ecs-init.deb.asc ./amazon-ecs-init.deb
```

Berikut adalah output yang diharapkan.

```
gpg: Signature made Fri 14 May 2021 09:31:36 PM UTC
gpg:                using RSA key 50DECCC4710E61AF
gpg: Good signature from "Amazon ECS <ecs-security@amazon.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: F34C 3DDA E729 26B0 79BE  AEC6 BCE9 D9A4 2D51 784F
     Subkey fingerprint: D64B B6F9 0CF3 77E9 B5FB  346F 50DE CCC4 710E 61AF
```

4. Instal paket `ecs-init`.

Untuk **`rpm`** paket di CentOS 7, CentOS 8, dan RHEL 7

```
sudo yum install -y ./amazon-ecs-init.rpm
```

Untuk **`rpm`** paket di SUSE Enterprise Server 15

```
sudo zypper install -y --allow-unsigned-rpm ./amazon-ecs-init.rpm
```

Untuk **`deb`** paketnya

```
sudo dpkg -i ./amazon-ecs-init.deb
```

5. Mulai ulang layanan `ecs`.

```
sudo systemctl restart ecs
```

6. Verifikasi bahwa versi agen Amazon ECS telah diperbarui.

```
curl -s 127.0.0.1:51678/v1/metadata | python3 -mjson.tool
```

Manajemen kapasitas

AWS mengelola kapasitas Fargate Anda, tetapi Anda bertanggung jawab untuk mengelola instans EC2 dan instans eksternal Anda.

Manajemen mencakup penambalan keamanan serta memperbarui instans penampung sehingga Anda memiliki akses ke fitur Amazon ECS terbaru.

Selain manajemen agen kontainer, ada manajemen instance tambahan yang mungkin diperlukan tergantung pada konfigurasi Anda. Hal ini mencakup:

- Meluncurkan instans kontainer
- Bootstrapping instance kontainer
- Memulai tugas saat peluncuran
- Menggunakan trunking ENI
- Mengelola memori
- Mengelola instance container Anda dari jarak jauh
- Menggunakan proxy HTTP untuk agen kontainer dan daemon Docker
- Memperbarui agen kontainer
- Menderegistrasi instance kontainer
- Pengeringan instance terkelola

Topik

- Agen kontainer Amazon ECS Linux
- Manajemen instance kontainer Linux
- Manajemen instance wadah Windows

- Pengurasan instans terkelola Amazon ECS

Untuk informasi selengkapnya tentang penyedia kapasitas Fargate dan penyedia kapasitas grup Auto Scaling Amazon EC2, lihat Penyedia kapasitas.

Agen kontainer Amazon ECS Linux

Setiap versi agen penampung Amazon ECS mendukung set fitur yang berbeda dan menyediakan perbaikan bug dari versi sebelumnya. Jika memungkinkan, kami selalu merekomendasikan menggunakan versi terbaru dari agen kontainer Amazon ECS. Untuk memperbarui agen kontainer ke versi terbaru, lihat Memperbarui agen kontainer Amazon ECS.

Untuk melihat fitur dan penyempurnaan mana yang disertakan dengan setiap rilis agen, lihat https://github.com/aws/ amazon-ecs-agent /releases.

> **Important**
>
> Versi Docker minimum untuk metrik yang andal adalah versi Docker `v20.10.13` dan yang lebih baru, yang disertakan dalam AMI yang dioptimalkan Amazon ECS dan yang lebih baru. `20220607`
> Versi agen Amazon ECS `1.20.0` dan yang lebih baru memiliki dukungan yang tidak digunakan lagi untuk versi Docker yang lebih lama dari. `1.9.0`

Siklus hidup

Saat agen penampung Amazon ECS mendaftarkan instans Amazon EC2 ke klaster Anda, instans Amazon EC2 melaporkan statusnya `ACTIVE` sebagai dan status koneksi agennya sebagai. `TRUE` Instance container ini dapat menerima permintaan tugas run.

Jika Anda menghentikan (tidak menghentikan) instance kontainer, statusnya tetap ada`ACTIVE`, tetapi status koneksi agen bertransisi ke `FALSE` dalam beberapa menit. Tugas yang berjalan pada instans kontainer berhenti. Jika Anda memulai instance penampung lagi, agen penampung akan terhubung kembali dengan layanan Amazon ECS, dan Anda dapat menjalankan tugas pada instance lagi.

> **Important**
>
> Jika Anda menghentikan dan memulai instance container, atau me-reboot instance tersebut, beberapa versi lama dari agen penampung Amazon ECS akan mendaftarkan instance

tersebut lagi tanpa membatalkan pendaftaran ID instance container asli. Dalam hal ini, Amazon ECS mencantumkan lebih banyak instance container di cluster Anda daripada yang sebenarnya Anda miliki. (Jika Anda memiliki ID instans penampung duplikat untuk ID instans Amazon EC2 yang sama, Anda dapat membatalkan pendaftaran duplikat yang terdaftar `ACTIVE` dengan status koneksi agen dengan aman.) `FALSE` Masalah ini telah diperbaiki dalam versi agen kontainer Amazon ECS saat ini. Untuk informasi selengkapnya tentang memperbarui versi CNI Anda, lihat Memperbarui agen kontainer Amazon ECS.

Jika Anda mengubah status instans kontainer ke `DRAINING`, maka tugas baru tidak ditempatkan pada instans kontainer. Tugas layanan yang berjalan pada instans kontainer akan dihapus jika memungkinkan, agar Anda dapat melakukan pembaruan sistem. Untuk informasi selengkapnya, lihat Pengurasan instans kontainer.

Jika Anda membatalkan pendaftaran atau mengakhiri instans kontainer, maka status instans kontainer seketika berubah menjadi `INACTIVE`, dan instans kontainer tidak lagi dilaporkan ketika Anda mencantumkan instans kontainer Anda. Namun, Anda tetap dapat menjelaskan instans kontainer selama satu jam setelah penghentian. Setelah satu jam, deskripsi instans tidak lagi tersedia.

Important

Anda dapat menguras instance secara manual, atau membuat hook siklus hidup grup Auto Scaling untuk menyetel status instance. `DRAINING` Lihat kait siklus hidup Auto Scaling Amazon EC2 untuk informasi selengkapnya tentang kait siklus hidup Auto Scaling.

AMI Amazon ECS yang dioptimalkan

Varian Linux dari AMI Amazon ECS yang dioptimalkan menggunakan Amazon Linux 2 AMI sebagai basis mereka. Nama AMI sumber Amazon Linux 2 untuk setiap varian dapat diambil dengan menanyakan Systems Manager Parameter Store API. Untuk informasi selengkapnya, lihat Mengambil metadata AMI Amazon ECS yang dioptimalkan. Saat Anda meluncurkan instans kontainer kami dari Amazon Linux 2 AMI Amazon ECS terbaru yang dioptimalkan Amazon ECS, Anda menerima versi agen kontainer saat ini. Untuk meluncurkan instans kontainer dengan Amazon Linux 2 AMI Amazon ECS terbaru yang dioptimalkan oleh Amazon ECS, lihat. Meluncurkan instans penampung Amazon ECS Linux

Menggunakan Sistem Operasi Linux lainnya

Untuk menginstal versi terbaru agen penampung Amazon ECS di sistem operasi lain, lihatMemasang agen kontainer Amazon ECS. Tabel di AMI Amazon ECS yang dioptimalkan menunjukkan versi Docker yang diuji pada Amazon Linux 2 untuk setiap versi agen.

Informasi tambahan

Halaman-halaman berikut memberikan informasi tambahan tentang perubahan:

- Amazon ECS Agent changelog aktif GitHub
- Kode sumber untuk `ecs-init` aplikasi dan skrip dan konfigurasi untuk pengemasan agen sekarang menjadi bagian dari repositori agen. Untuk versi `ecs-init` dan kemasan yang lebih lama, lihat Amazon ecs-init changelog di GitHub
- Catatan rilis Amazon Linux 2.
- Catatan rilis Docker Engine dalam dokumentasi Docker
- Dokumentasi Driver NVIDIA dalam dokumentasi NVIDIA

Konfigurasi agen kontainer Amazon ECS

Agen penampung Amazon ECS mendukung sejumlah opsi konfigurasi, yang sebagian besar harus diatur melalui variabel lingkungan. Variabel lingkungan berikut tersedia, dan semuanya bersifat opsional.

Jika instance container Anda diluncurkan dengan varian Linux dari AMI Amazon ECS yang dioptimalkan, Anda dapat mengatur variabel lingkungan ini dalam `/etc/ecs/ecs.config` file dan kemudian memulai ulang agen. Anda juga dapat menulis variabel konfigurasi ini ke instans penampung Anda dengan data pengguna Amazon EC2 pada waktu peluncuran. Untuk informasi selengkapnya, lihat Bootstrapping instance container dengan data pengguna Amazon EC2.

Jika instance container Anda diluncurkan dengan varian Windows dari AMI Amazon ECS yang dioptimalkan, Anda dapat mengatur variabel lingkungan ini dengan PowerShell SetEnvironmentVariable perintah dan kemudian memulai ulang agen. Untuk informasi selengkapnya, lihat Menjalankan perintah pada instans Windows Anda saat diluncurkan di Panduan Pengguna Amazon EC2 untuk Instans Windows dan. the section called “Instans Kontainer Bootstrap”

Jika Anda secara manual memulai agen penampung Amazon ECS (untuk AMI yang tidak dioptimalkan oleh Amazon ECS), Anda dapat menggunakan variabel lingkungan ini dalam

docker run perintah yang Anda gunakan untuk memulai agen. Gunakan variabel ini dengan `--env=`*`VARIABLE_NAME`*`=`*`VARIABLE_VALUE`* sintaksis. Untuk informasi sensitif, seperti kredensial autentikasi untuk repositori privat, Anda harus menyimpan variabel lingkungan agen Anda dalam file dan melakukan semuanya pada satu waktu dengan opsi `--env-file` *`path_to_env_file`*. Anda dapat menggunakan perintah berikut untuk menambahkan variabel.

```
sudo systemctl stop ecs
sudo vi /etc/ecs/ecs.config
# And add the environment variables with VARIABLE_NAME=VARIABLE_VALUE format.
sudo systemctl start ecs
```

Parameter yang tersedia

Untuk informasi tentang parameter konfigurasi agen kontainer Amazon ECS yang tersedia, lihat Agen Kontainer Amazon ECS di. GitHub

Menyimpan konfigurasi instans kontainer di Amazon S3

Konfigurasi agen kontainer Amazon ECS dikontrol dengan variabel lingkungan yang dijelaskan di bagian sebelumnya. Varian Linux dari AMI yang dioptimalkan Amazon ECS mencari variabel-variabel ini `/etc/ecs/ecs.config` saat agen penampung memulai dan mengonfigurasi agen yang sesuai. Variabel lingkungan tertentu yang tidak berbahaya, sepertiECS_CLUSTER, dapat diteruskan ke instance container saat diluncurkan melalui data pengguna Amazon EC2 dan ditulis ke file ini tanpa konsekuensi. Namun, informasi sensitif lainnya, seperti AWS kredensi Anda atau ECS_ENGINE_AUTH_DATA variabel, tidak boleh diteruskan ke instance dalam data pengguna atau ditulis dengan cara yang memungkinkan mereka muncul dalam file. `/etc/ecs/ecs.config` `.bash_history`

Menyimpan informasi konfigurasi dalam bucket pribadi di Amazon S3 dan memberikan akses hanya-baca ke peran IAM instance container Anda adalah cara yang aman dan nyaman untuk mengizinkan konfigurasi instance container saat diluncurkan. Anda dapat menyimpan salinan file `ecs.config` dalam bucket privat. Anda kemudian dapat menggunakan data pengguna Amazon EC2 untuk menginstal AWS CLI dan menyalin informasi konfigurasi Anda `/etc/ecs/ecs.config` saat instans diluncurkan.

Untuk mengizinkan akses hanya-baca Amazon S3 untuk peran instance container Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.

2. Di panel navigasi, pilih Peran dan pilih peran IAM yang akan digunakan untuk instance container Anda. Peran ini kemungkinan berjudul `ecsInstanceRole`. Untuk informasi selengkapnya, lihat Peran IAM instans wadah Amazon ECS.
3. Di bawah Kebijakan Terkelola, pilih Lampirkan Kebijakan.
4. Untuk mempersempit hasil kebijakan, pada halaman Lampirkan Kebijakan, untuk Filter, ketik `S3`.
5. Pilih kotak di sebelah kiri kebijakan AmazonS3 dan pilih ReadOnlyAccess Lampirkan Kebijakan.

Untuk menyimpan **`ecs.config`** file di Amazon S3

1. Buat `ecs.config` file dengan variabel konfigurasi agen Amazon ECS yang valid menggunakan format berikut. Contoh ini mengonfigurasi autentikasi registri privat. Untuk informasi selengkapnya, lihat Autentikasi registri privat untuk tugas.

   ```
   ECS_ENGINE_AUTH_TYPE=dockercfg
   ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
   {"auth":"zq212MzEXAMPLE7o6T25Dk0i","email":"email@example.com"}}
   ```

   > **Note**
   >
   > Untuk daftar lengkap variabel konfigurasi agen Amazon ECS yang tersedia, lihat Agen Kontainer Amazon ECS di. GitHub

2. Untuk menyimpan file konfigurasi Anda, buat bucket pribadi di Amazon S3. Untuk informasi selengkapnya, lihat Membuat Bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.
3. Unggah file `ecs.config` ke bucket S3 Anda. Untuk informasi selengkapnya, lihat Menambahkan Objek ke Bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Untuk memuat **`ecs.config`** file dari Amazon S3 saat peluncuran

1. Selesaikan prosedur sebelumnya di bagian ini untuk mengizinkan akses Amazon S3 hanya-baca ke instans penampung Anda dan simpan `ecs.config` file di bucket S3 pribadi.
2. Luncurkan instans kontainer baru dengan mengikuti langkah-langkah di Meluncurkan instans penampung Amazon ECS Linux. DalamStep 6.g, gunakan contoh skrip berikut yang menginstal AWS CLI dan menyalin file konfigurasi Anda ke`/etc/ecs/ecs.config`.

```
#!/bin/bash
yum install -y aws-cli
aws s3 cp s3://your_bucket_name/ecs.config /etc/ecs/ecs.config
```

Autentikasi registri privat untuk instans kontainer

Agen kontainer Amazon ECS dapat mengautentikasi dengan pendaftar pribadi, menggunakan otentikasi dasar. Bila Anda mengaktifkan autentikasi registri privat, Anda dapat menggunakan citra Docker privat dalam ketentuan tugas Anda. Fitur ini hanya didukung oleh tugas menggunakan tipe peluncuran EC2.

Metode lain untuk mengaktifkan otentikasi registri pribadi digunakan AWS Secrets Manager untuk menyimpan kredenal registri pribadi Anda dengan aman dan kemudian mereferensikannya dalam definisi kontainer Anda. Hal ini mengizinkan tugas Anda untuk menggunakan citra dari repositori privat. Metode ini mendukung tugas menggunakan jenis peluncuran EC2 atau Fargate. Untuk informasi selengkapnya, lihat Autentikasi registri privat untuk tugas.

Agen kontainer Amazon ECS mencari dua variabel lingkungan saat diluncurkan:

- `ECS_ENGINE_AUTH_TYPE`, yang menentukan tipe data autentikasi yang sedang dikirim.
- `ECS_ENGINE_AUTH_DATA`, yang berisi kredensial autentikasi aktual.

Varian Linux dari AMI Amazon ECS yang dioptimalkan memindai `/etc/ecs/ecs.config` file untuk variabel-variabel ini ketika instance container diluncurkan, dan setiap kali layanan dimulai (dengan perintah). sudo start ecs AMI yang tidak dioptimalkan Amazon ECS harus menyimpan variabel lingkungan ini dalam file dan meneruskannya dengan `--env-file` *`path_to_env_file`* opsi ke docker run perintah yang memulai agen penampung.

> **Important**
>
> Kami tidak menyarankan Anda menyuntikkan variabel lingkungan otentikasi ini saat peluncuran instans dengan data pengguna Amazon EC2 atau meneruskannya dengan opsi `--env` ke perintah. docker run Metode ini tidak sesuai untuk data sensitif, seperti kredensial autentikasi. Untuk informasi tentang menambahkan kredensial autentikasi secara aman ke instans kontainer Anda, lihat Menyimpan konfigurasi instans kontainer di Amazon S3.

Format autentikasi

Ada dua format yang tersedia untuk autentikasi registri privat, `dockercfg` dan `docker`.

Format autentikasi dockercfg

Format `dockercfg` menggunakan informasi autentikasi yang disimpan dalam file konfigurasi yang dibuat ketika Anda menjalankan perintah docker login. Anda dapat membuat file ini dengan menjalankan docker login di sistem lokal dan memasukkan nama pengguna registri, kata sandi, dan alamat email. Anda juga dapat masuk ke instans kontainer dan menjalankan perintah di sana. Tergantung pada versi Docker Anda, file ini disimpan sebagai `~/.dockercfg` atau `~/.docker/config.json`.

```
cat ~/.docker/config.json
```

Output:

```
{
  "auths": {
    "https://index.docker.io/v1/": {
      "auth": "zq212MzEXAMPLE7o6T25Dk0i"
    }
  }
}
```

> **Important**
>
> Versi Docker yang lebih baru membuat file konfigurasi seperti yang ditunjukkan di atas dengan objek `auths`. Agen Amazon ECS hanya mendukung data `dockercfg` otentikasi yang dalam format di bawah ini, tanpa objek. `auths` Jika Anda memiliki utilitas jq yang terinstal, Anda dapat mengekstrak data ini dengan perintah berikut: cat ~/.docker/config.json | jq .auths

```
cat ~/.docker/config.json | jq .auths
```

Output:

```
{
```

```
  "https://index.docker.io/v1/": {
    "auth": "zq212MzEXAMPLE7o6T25Dk0i",
    "email": "email@example.com"
  }
}
```

Dalam contoh di atas, variabel lingkungan berikut harus ditambahkan ke file variabel lingkungan (`/etc/ecs/ecs.config`untuk AMI yang dioptimalkan Amazon ECS) yang dimuat agen kontainer Amazon ECS saat runtime. Jika Anda tidak menggunakan AMI Amazon ECS yang dioptimalkan dan Anda memulai agen secara manualdocker run, tentukan file variabel lingkungan dengan `--env-file` *`path_to_env_file`* opsi saat Anda memulai agen.

```
ECS_ENGINE_AUTH_TYPE=dockercfg
ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
{"auth":"zq212MzEXAMPLE7o6T25Dk0i","email":"email@example.com"}}
```

Anda dapat mengonfigurasi beberapa registri privat dengan sintaksis berikut:

```
ECS_ENGINE_AUTH_TYPE=dockercfg
ECS_ENGINE_AUTH_DATA={"repo.example-01.com":
{"auth":"zq212MzEXAMPLE7o6T25Dk0i","email":"email@example-01.com"},"repo.example-02.com":
{"auth":"fQ172MzEXAMPLEoF7225DU0j","email":"email@example-02.com"}}
```

Format autentikasi dockercfg

Format `docker` menggunakan representasi server registri JSON yang harus diautentikasi dengan agen. Hal ini juga mencakup parameter autentikasi yang diperlukan oleh registri tersebut (seperti nama pengguna, kata sandi, dan alamat email untuk akun tersebut). Untuk akun Docker Hub, representasi JSON terlihat seperti berikut ini:

```
{
  "https://index.docker.io/v1/": {
    "username": "my_name",
    "password": "my_password",
    "email": "email@example.com"
  }
}
```

Dalam contoh ini, variabel lingkungan berikut harus ditambahkan ke file variabel lingkungan (`/etc/ecs/ecs.config`untuk AMI yang dioptimalkan Amazon ECS) yang dimuat agen kontainer Amazon

ECS saat runtime. Jika Anda tidak menggunakan AMI Amazon ECS yang dioptimalkan, dan Anda memulai agen secara manualdocker run, tentukan file variabel lingkungan dengan `--env-file` *`path_to_env_file`* opsi saat Anda memulai agen.

```
ECS_ENGINE_AUTH_TYPE=docker
ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
{"username":"my_name","password":"my_password","email":"email@example.com"}}
```

Anda dapat mengonfigurasi beberapa registri privat dengan sintaksis berikut:

```
ECS_ENGINE_AUTH_TYPE=docker
ECS_ENGINE_AUTH_DATA={"repo.example-01.com":
{"username":"my_name","password":"my_password","email":"email@example-01.com"},"repo.example-02
{"username":"another_name","password":"another_password","email":"email@example-02.com"}}
```

Mengaktifkan pendaftar pribadi

Gunakan prosedur berikut untuk mengaktifkan pendaftar pribadi untuk instance kontainer Anda.

Untuk mengaktifkan pendaftar pribadi di AMI Amazon ECS yang dioptimalkan

1. Masuk ke instans kontainer Anda menggunakan SSH.
2. Buka file `/etc/ecs/ecs.config` dan tambahkan nilai `ECS_ENGINE_AUTH_TYPE` dan `ECS_ENGINE_AUTH_DATA` untuk registri dan akun Anda:

   ```
   sudo vi /etc/ecs/ecs.config
   ```

   Contoh ini mengautentikasi akun pengguna Docker Hub:

   ```
   ECS_ENGINE_AUTH_TYPE=docker
   ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
   {"username":"my_name","password":"my_password","email":"email@example.com"}}
   ```
3. Lakukan pemeriksaan untuk melihat apakah agen Anda menggunakan variabel lingkungan `ECS_DATADIR` untuk menyimpan statusnya:

   ```
   docker inspect ecs-agent | grep ECS_DATADIR
   ```

   Output:

```
"ECS_DATADIR=/data",
```

> **Important**
>
> Jika perintah sebelumnya tidak mengembalikan variabel lingkungan `ECS_DATADIR`, Anda harus menghentikan setiap tugas yang berjalan pada instans kontainer ini sebelum menghentikan agen. Agen yang lebih baru dengan variabel lingkungan `ECS_DATADIR` menyimpan statusnya dan Anda dapat menghentikannya saat tugas sedang berjalan tanpa masalah. Untuk informasi selengkapnya, lihat Memperbarui agen kontainer Amazon ECS.

4. Hentikan layanan `ecs`:

```
sudo stop ecs
```

Output:

```
ecs stop/waiting
```

5. Mulai ulang layanan `ecs`.

   - Untuk Amazon ECS yang dioptimalkan Amazon Linux 2 AMI:

     ```
     sudo systemctl restart ecs
     ```

   - Untuk Amazon ECS yang dioptimalkan Amazon Linux AMI:

     ```
     sudo stop ecs && sudo start ecs
     ```

6. (Opsional) Anda dapat memastikan bahwa agen sedang berjalan dan melihat beberapa informasi tentang instans kontainer baru dengan melakukan kueri pada operasi API introspeksi agen. Untuk informasi selengkapnya, lihat the section called “Introspeksi wadah”.

```
curl http://localhost:51678/v1/metadata
```

Pembersihan tugas dan citra otomatis

Setiap kali tugas ditempatkan pada instance kontainer, agen penampung Amazon ECS memeriksa untuk melihat apakah gambar yang direferensikan dalam tugas adalah yang terbaru dari tag yang ditentukan dalam repositori. Jika tidak, perilaku default mengizinkan agen untuk menarik citra dari repositori masing-masing. Jika Anda sering memperbarui citra dalam tugas dan layanan Anda, penyimpanan instans kontainer dapat dengan cepat terisi citra Docker yang tidak lagi Anda gunakan dan mungkin tidak akan pernah digunakan kembali. Misalnya, Anda dapat menggunakan integrasi dan alur (CI/CD) deployment berkelanjutan.

> **Note**
>
> Perilaku tarik gambar agen Amazon ECS dapat disesuaikan menggunakan `ECS_IMAGE_PULL_BEHAVIOR` parameter. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.

Demikian juga, kontainer dengan tugas berhenti juga dapat memakan penyimpanan instans kontainer dengan informasi log, volume data, dan artefak lainnya. Artefak ini berguna untuk debugging kontainer yang telah berhenti tiba-tiba, tetapi sebagian besar penyimpanan ini dapat dengan aman dibersihkan setelah jangka waktu tertentu.

Secara default, agen penampung Amazon ECS secara otomatis membersihkan tugas yang dihentikan dan gambar Docker yang tidak digunakan oleh tugas apa pun pada instance penampung Anda.

> **Note**
>
> Fitur pembersihan gambar otomatis memerlukan setidaknya versi 1.13.0 dari agen penampung Amazon ECS. Untuk memperbarui agen ke versi terbaru, lihat Memperbarui agen kontainer Amazon ECS.

Parameter yang bisa disetel

Variabel konfigurasi agen berikut tersedia untuk menyetel tugas otomatis dan pengalaman pembersihan citra Anda. Untuk informasi lebih lanjut tentang cara mengatur variabel ini pada instans kontainer Anda, lihat Konfigurasi agen kontainer Amazon ECS.

`ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION`

Variabel ini menentukan waktu tunggu sebelum menghapus kontainer dengan tugas berhenti. Proses pembersihan citra tidak dapat dilakukan selama ada kontainer yang mereferensikannya. Setelah gambar tidak direferensikan oleh kontainer apa pun (baik dihentikan atau berjalan), gambar menjadi kandidat untuk pembersihan. Secara default, parameter ini diatur ke 3 jam, tetapi Anda dapat mengurangi periode ini menjadi serendah 1 detik jika Anda perlu untuk aplikasi Anda. Parameter diabaikan jika Anda menetapkan nilai kurang dari 1 detik.

`ECS_DISABLE_IMAGE_CLEANUP`

Jika Anda menyetel variabel ini`true`, maka pembersihan gambar otomatis dimatikan pada instance penampung Anda dan tidak ada gambar yang dihapus secara otomatis.

`ECS_IMAGE_CLEANUP_INTERVAL`

Variabel ini menentukan seberapa sering proses pembersihan citra otomatis harus diperiksa untuk menghapus citra. Pengaturan defaultnya adalah setiap 30 menit tetapi Anda dapat mengurangi periode ini hingga 10 menit untuk lebih sering menghapus citra.

`ECS_IMAGE_MINIMUM_CLEANUP_AGE`

Variabel ini menentukan jumlah minimum waktu antara ketika citra ditarik dan kapan citra tersebut akan dihapus. Hal ini digunakan agar citra yang baru saja ditarik tidak dihapus. Pengaturan default-nya adalah 1 jam.

`ECS_NUM_IMAGES_DELETE_PER_CYCLE`

Variabel ini menentukan berapa banyak gambar yang dapat dihapus selama siklus pembersihan tunggal. Pengaturan default-nya adalah 5 dan nilai minimumnya adalah 1.

Bersihkan alur kerja

Saat agen penampung Amazon ECS berjalan dan pembersihan gambar otomatis tidak dimatikan, agen memeriksa gambar Docker yang tidak direferensikan dengan menjalankan atau menghentikan kontainer pada frekuensi yang ditentukan oleh variabel. `ECS_IMAGE_CLEANUP_INTERVAL` Jika citra yang tidak digunakan ditemukan dan citra tersebut lebih tua dari waktu pembersihan minimum yang ditentukan oleh variabel `ECS_IMAGE_MINIMUM_CLEANUP_AGE`, agen tersebut menghapus citra hingga jumlah maksimum yang ditentukan oleh variabel `ECS_NUM_IMAGES_DELETE_PER_CYCLE`. Citra yang paling sedikit direferensikan baru-baru ini akan dihapus terlebih dahulu. Setelah citra dihapus, agen menunggu sampai interval berikutnya dan mengulangi proses lagi.

Menambahkan atribut ke instans penampung Amazon ECS

Anda dapat menambahkan atribut kustom pada saat pendaftaran instans menggunakan agen kontainer atau secara manual, menggunakan AWS Management Console. Untuk informasi tentang parameter konfigurasi agen penampung Amazon ECS yang tersedia, lihat Agen Kontainer Amazon ECS di. GitHub

Untuk menambahkan atribut kustom menggunakan konsol

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Clusters, lalu pilih cluster.
3. Pilih Infrastruktur.
4. Pilih instance container, lalu coose Attributes.
5. Pilih Tambahkan atribut kustom.
6. Untuk nama dan Nilai Atribut, masukkan nama dan nilai untuk atribut, lalu pilih Kirim.

   Ulangi untuk setiap atribut yang ingin Anda tambahkan.

Menambahkan atribut khusus menggunakan AWS CLI

Contoh-contoh berikut ini mendemonstrasikan cara menambahkan atribut kustom menggunakan perintah letakkan-atribut.

Contoh: Atribut tunggal

Contoh berikut menambahkan atribut kustom “tumpukan=prod” untuk instans kontainer tertentu dalam klaster default.

```
aws ecs put-attributes --attributes name=stack,value=prod,targetId=arn
```

Contoh: Beberapa Atribut

Contoh berikut menambahkan atribut kustom “tumpukan=prod” dan “proyek=a” untuk instans kontainer tertentu di klaster default.

```
aws ecs put-attributes --attributes name=stack,value=prod,targetId=arn
 name=project,value=a,targetId=arn
```

Memfilter berdasarkan atribut menggunakan konsol

Anda dapat menerapkan filter untuk instans kontainer Anda, sehingga Anda dapat melihat atribut kustom.

Saring instans kontainer dengan atribut menggunakan konsol

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Clusters, lalu pilih cluster.
3. Pilih Infrastruktur
4. Pilih contoh kontainer.
5. Menggunakan bidang Filter atribut kustom menurut kunci atau nilai teks, masukkan atribut yang ingin Anda filter. Formatnya harus AttributeName: AttributeValue.

Filter contoh kontainer dengan atribut menggunakan AWS CLI

Contoh berikut menunjukkan cara memfilter instance kontainer dengan atribut menggunakan list-constainer-instancesperintah. Untuk informasi selengkapnya tentang sintaksis pemfilteran, lihat Buat ekspresi untuk menentukan instance kontainer untuk tugas Amazon ECS.

Contoh: Atribut bawaan

Contoh berikut menggunakan atribut bawaan untuk mencantumkan instans g2.2xlarge.

```
aws ecs list-container-instances --filter "attribute:ecs.instance-type == g2.2xlarge"
```

Contoh: Atribut kustom

Contoh berikut mencantumkan instans dengan atribut kustom “tumpukan=prod”.

```
aws ecs list-container-instances --filter "attribute:stack == prod"
```

Contoh: Mengecualikan nilai atribut

Contoh berikut mencantumkan contoh dengan atribut kustom “tumpukan” kecuali nilai atributnya adalah “prod”.

```
aws ecs list-container-instances --filter "attribute:stack != prod"
```

Contoh: Beberapa nilai atribut

Contoh berikut menggunakan atribut bawaan untuk mencantumkan instans tipe `t2.small` atau `t2.medium`.

```
aws ecs list-container-instances --filter "attribute:ecs.instance-type in [t2.small,
 t2.medium]"
```

Contoh: Beberapa atribut

Contoh berikut menggunakan atribut bawaan untuk mencantumkan instans T2 di Availability Zone us–east-1a.

```
aws ecs list-container-instances --filter "attribute:ecs.instance-type =~ t2.* and
 attribute:ecs.availability-zone == us-east-1a"
```

Manajemen instance kontainer Linux

Manajemen instance wadah Linux meliputi:

- Meluncurkan instans kontainer
- Bootstrapping instance kontainer
- Memulai tugas saat peluncuran
- Menggunakan trunking ENI
- Mengelola memori
- Mengelola instance kontainer Anda dari jarak jauh
- Menggunakan proxy HTTP untuk agen kontainer dan daemon Docker
- Memperbarui agen kontainer

  Setiap versi agen penampung Amazon ECS mendukung set fitur yang berbeda dan menyediakan perbaikan bug dari versi sebelumnya. Jika memungkinkan, kami selalu merekomendasikan menggunakan versi terbaru dari agen kontainer Amazon ECS. Untuk memperbarui agen kontainer ke versi terbaru, lihat Memperbarui agen kontainer Amazon ECS.

  Untuk melihat fitur dan penyempurnaan mana yang disertakan dengan setiap rilis agen, lihat https://github.com/aws/ amazon-ecs-agent /releases.

> **Important**
>
> Versi Docker minimum untuk metrik yang andal adalah versi Docker `v20.10.13` dan yang lebih baru, yang disertakan dalam AMI yang dioptimalkan Amazon ECS dan yang lebih baru. `20220607`
> Versi agen Amazon ECS `1.20.0` dan yang lebih baru memiliki dukungan yang tidak digunakan lagi untuk versi Docker yang lebih lama dari. `1.9.0`

Meluncurkan instans penampung Amazon ECS Linux

Instans penampung Amazon ECS Anda dibuat menggunakan konsol Amazon EC2. Sebelum memulai, pastikan Anda telah menyelesaikan langkah-langkah tersebut di Siapkan untuk menggunakan Amazon ECS.

Anda dapat meluncurkan instans dengan berbagai metode termasuk konsol Amazon EC2 AWS CLI, dan SDK. Prosedur pada halaman ini mencakup panduan peluncuran di konsol Amazon EC2. Untuk informasi tentang metode lain untuk meluncurkan instans, lihat Meluncurkan instans Anda di Panduan Pengguna Amazon EC2 untuk Instans Linux.

Untuk informasi selengkapnya tentang panduan peluncuran, lihat Meluncurkan instance menggunakan wizard instans peluncuran baru di Panduan Pengguna Amazon EC2 untuk Instans Linux.

Wisaya instans peluncuran Amazon EC2 baru

Anda dapat menggunakan wizard Amazon EC2 baru untuk meluncurkan instans. Wizard peluncuran instans menentukan parameter peluncuran yang diperlukan untuk meluncurkan sebuah instans. Anda dapat menggunakan daftar berikut untuk parameter dan membiarkan parameter tidak terdaftar sebagai default. Instruksi berikut membawa Anda melalui setiap kelompok parameter.

Parameter untuk konfigurasi instans

- Memulai peluncuran instans
- Nama dan tanda
- Aplikasi dan Gambar OS (Gambar Mesin Amazon)
- Jenis instans
- Pasangan kunci (login)

- Pengaturan jaringan
- Mengonfigurasi penyimpanan
- Detail lanjutan

Memulai peluncuran instans

1. Buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.
2. Di bilah navigasi di bagian atas layar, AWS Wilayah saat ini ditampilkan (misalnya, AS Timur (Ohio)). Pilih Wilayah tempat instans akan diluncurkan.
3. Dari dasbor konsol Amazon EC2, pilih Luncurkan instans.

Nama dan tanda

Nama instans adalah tanda, di mana kuncinya adalah Name, dan nilainya adalah nama yang Anda tentukan. Anda dapat menandai instance, volume, dan grafik elastis. Untuk Instans Spot, Anda hanya dapat menandai permintaan Instans Spot.

Menentukan nama instans dan tanda tambahan bersifat opsional.

- Untuk Nama, masukkan nama deskriptif untuk instans tersebut. Jika Anda tidak menentukan nama, instans dapat diidentifikasi berdasarkan ID-nya, yang secara otomatis dihasilkan saat Anda meluncurkan instans tersebut.
- Untuk menambahkan tanda tambahan, pilih Tambahkan tanda tambahan. Pilih Tambahkan tanda, lalu masukkan kunci dan nilai, lalu pilih jenis sumber daya yang akan diberi tanda. Pilih Tambah tanda lagi untuk setiap tanda tambahan yang akan ditambahkan.

Aplikasi dan Gambar OS (Gambar Mesin Amazon)

Amazon Machine Image (AMI) berisi informasi yang diperlukan untuk membuat instans. Misalnya, AMI mungkin berisi perangkat lunak yang diperlukan untuk bertindak sebagai server web, seperti Apache, dan situs web Anda.

Gunakan bilah Pencarian untuk menemukan AMI yang dioptimalkan Amazon ECS yang dioptimalkan yang diterbitkan oleh. AWS

1. Masukkan salah satu istilah berikut di bilah Pencarian.
   - **`ami-ecs`**

- Nilai AMI Amazon ECS yang dioptimalkan.

  Untuk AMI terbaru yang dioptimalkan Amazon ECS dan nilainya, lihat AMI yang dioptimalkan untuk Linux Amazon ECS.

2. Tekan Enter.
3. Pada halaman Pilih Gambar Mesin Amazon (AMI), pilih tab AWS Marketplace AMI.
4. Dari panel hasil Refine kiri, pilih Amazon Web Services sebagai Publisher.
5. Pilih Pilih pada baris AMI yang ingin Anda gunakan.

   Atau, pilih Batal (di kanan atas) untuk kembali ke wizard instance peluncuran tanpa memilih AMI. AMI default akan dipilih. Pastikan AMI memenuhi persyaratan yang diuraikan dalam instance Linux.

Jenis instans

Tipe instans mendefinisikan konfigurasi perangkat keras dan ukuran instans. Tipe instans yang lebih besar memiliki lebih banyak CPU dan memori. Untuk informasi selengkapnya, lihat Tipe instans.

- Untuk Tipe instans, pilih tipe instans untuk instans tersebut.

  Tipe instans yang Anda pilih menentukan sumber daya yang tersedia untuk menjalankan tugas Anda.

Pasangan kunci (login)

Untuk Nama pasangan kunci, pilih pasangan kunci yang ada, atau pilih Buat pasangan kunci baru untuk membuat yang baru.

> **Important**
>
> Jika Anda memilih opsi Lanjutkan tanpa pasangan kunci (Tidak direkomendasikan), Anda tidak akan dapat terhubung ke instans tersebut, kecuali Anda memilih sebuah AMI yang dikonfigurasi agar pengguna dapat masuk dengan cara lain.

Pengaturan jaringan

Konfigurasikan pengaturan jaringan, sesuai keperluan.

- Platform jaringan: Pilih Virtual Private Cloud (VPC), lalu tentukan subnet di bagian Network interface.
- VPC: Pilih VPC yang ada untuk membuat grup keamanan.
- Subnet: Anda dapat meluncurkan sebuah instans di subnet yang terkait dengan Zona Ketersediaan, Local Zone, Wavelength Zone, atau Outpost.

  Untuk meluncurkan instans di Zona Ketersediaan, pilih subnet tempat Anda akan meluncurkan instans. Untuk membuat subnet baru, pilih Buat subnet baru untuk membuka konsol Amazon VPC. Setelah selesai, kembali ke wizard peluncuran instans dan pilih ikon Segarkan untuk memuat subnet Anda dalam daftar.

  Untuk meluncurkan instans di Local Zone, pilih subnet yang Anda buat di Local Zone.

  Untuk meluncurkan sebuah instans di Outpost, pilih subnet di VPC yang Anda kaitkan dengan Outpost.
- Auto-assign IP Publik: Jika instans Anda harus dapat diakses dari internet, verifikasi bahwa bidang Auto-assign Public IP diatur ke Aktifkan. Jika tidak, atur bidang ini ke Nonaktifkan.

  > **Note**
  >
  > Instans kontainer memerlukan akses untuk berkomunikasi dengan titik akhir layanan Amazon ECS. Ini bisa melalui titik akhir VPC antarmuka atau melalui instance container Anda yang memiliki alamat IP publik.
  > Untuk informasi selengkapnya tentang titik akhir VPC antarmuka, lihat Titik akhir VPC antarmuka Amazon ECS ()AWS PrivateLink
  > Jika Anda tidak memiliki antarmuka VPC endpoint yang dikonfigurasi dan instance container Anda tidak memiliki alamat IP publik, maka mereka harus menggunakan terjemahan alamat jaringan (NAT) untuk menyediakan akses ini. Untuk informasi selengkapnya, lihat gateway NAT di Panduan Pengguna Amazon VPC dan dalam panduan ini. Konfigurasi proxy HTTP untuk instance kontainer Linux
- Firewall (grup keamanan): Gunakan grup keamanan untuk menentukan aturan firewall untuk instance container Anda. Aturan ini menentukan lalu lintas jaringan masuk yang dikirim ke instans kontainer Anda. Semua lalu lintas lainnya diabaikan.
  - Untuk memilih grup keamanan yang ada, pilih Pilih grup keamanan yang ada, dan pilih grup keamanan yang Anda buatSiapkan untuk menggunakan Amazon ECS.

Mengonfigurasi penyimpanan

AMI yang Anda pilih mencakup satu atau lebih volume penyimpanan, termasuk volume root. Anda dapat menentukan volume tambahan untuk dilampirkan ke instans.

Anda dapat menggunakan tampilan Sederhana.

- Jenis penyimpanan: Konfigurasikan penyimpanan untuk instance kontainer Anda.

  Jika Anda menggunakan Amazon Linux 2 AMI Amazon ECS yang dioptimalkan Amazon, instans Anda memiliki satu volume 30 GiB yang dikonfigurasi, yang dibagi antara sistem operasi dan Docker.

  Jika Anda menggunakan AMI Amazon ECS yang dioptimalkan, instans Anda memiliki dua volume yang dikonfigurasi. Volume Root adalah untuk penggunaan sistem operasi, dan volume Amazon EBS kedua (dilampirkan ke`/dev/xvdcz`) adalah untuk penggunaan Docker.

  Anda dapat menambah atau mengurangi ukuran volume bagi instans Anda untuk memenuhi kebutuhan aplikasi Anda.

Detail lanjutan

Untuk Detail lanjutan, perluas bagian untuk melihat kolom dan menentukan parameter tambahan apa pun untuk instans.

- Opsi pembelian: Pilih Minta Instans Spot untuk meminta Instans Spot. Anda juga perlu mengatur bidang lain yang terkait dengan Instans Spot. Untuk informasi selengkapnya, lihat Permintaan Instans Spot.

  > **Note**
  >
  > Jika Anda menggunakan Instans Spot dan melihat `Not available` pesan, Anda mungkin perlu memilih jenis instans yang berbeda.

  .

- Profil instans IAM: Pilih peran IAM instance container Anda. Hal ini biasanya bernama `ecsInstanceRole`.

> **Important**
>
> Jika Anda tidak meluncurkan instans penampung dengan izin IAM yang tepat, agen Amazon ECS Anda tidak dapat terhubung ke klaster Anda. Untuk informasi selengkapnya, lihat Peran IAM instans wadah Amazon ECS.

- (Opsional) Data pengguna: Konfigurasikan instans penampung Amazon ECS Anda dengan data pengguna, seperti variabel lingkungan agen dariKonfigurasi agen kontainer Amazon ECS. Skrip data pengguna Amazon EC2 dijalankan hanya satu kali, saat instans pertama kali diluncurkan. Berikut ini adalah contoh umum dari kegunaan data pengguna:
  - Secara default, instans kontainer Anda meluncurkan ke klaster default Anda. Untuk meluncurkan ke klaster non-default, pilih daftar Detail Lanjutan. Kemudian, paste script berikut ke bidang Data pengguna, mengganti *nama_klaster_anda* dengan nama klaster Anda.

    ```
    #!/bin/bash
    echo ECS_CLUSTER=your_cluster_name >> /etc/ecs/ecs.config
    ```

  - Jika Anda memiliki **ecs.config** file di Amazon S3 dan telah mengaktifkan akses hanya-baca Amazon S3 ke peran instance container, pilih daftar Detail Lanjutan. Kemudian, tempel skrip berikut ke bidang Data pengguna, ganti *your_bucket_name dengan nama* bucket Anda untuk menginstal AWS CLI dan menulis file konfigurasi Anda pada waktu peluncuran.

    > **Note**
    >
    > Untuk informasi lebih lanjut tentang konfigurasi ini, lihat Menyimpan konfigurasi instans kontainer di Amazon S3.

    ```
    #!/bin/bash
    yum install -y aws-cli
    aws s3 cp s3://your_bucket_name/ecs.config /etc/ecs/ecs.config
    ```

  - Tentukan tanda untuk instans kontainer Anda menggunakan parameter konfigurasi `ECS_CONTAINER_INSTANCE_TAGS`. Ini membuat tag yang terkait dengan Amazon ECS saja, mereka tidak dapat dicantumkan menggunakan Amazon EC2 API.

> **Important**
>
> Jika Anda meluncurkan instance container menggunakan grup Auto Scaling Amazon EC2, Anda harus menggunakan parameter konfigurasi agen ECS_CONTAINER_INSTANCE_TAGS untuk menambahkan tag. Hal ini disebabkan oleh cara tag ditambahkan ke instans Amazon EC2 yang diluncurkan menggunakan grup Auto Scaling.

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=your_cluster_name
ECS_CONTAINER_INSTANCE_TAGS={"tag_key": "tag_value"}
EOF
```

- Tentukan tag untuk instance container Anda, lalu gunakan parameter `ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM` konfigurasi untuk menyebarkannya dari Amazon EC2 ke Amazon ECS

  Berikut adalah instans dari skrip data pengguna yang akan menyebarkan tanda yang terkait dengan instans kontainer, serta mendaftarkan instans kontainer dengan klaster bernama `your_cluster_name`:

  ```
  #!/bin/bash
  cat <<'EOF' >> /etc/ecs/ecs.config
  ECS_CLUSTER=your_cluster_name
  ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM=ec2_instance
  EOF
  ```

Untuk informasi selengkapnya, lihat Bootstrapping instance container dengan data pengguna Amazon EC2.

Wisaya instans peluncuran Amazon EC2 lama

Untuk meluncurkan sebuah instans kontainer

1. Buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.

3. Dari Dasbor EC2, pilih Meluncurkan instans.
4. Pada halaman Pilih Amazon Machine Image (AMI) , selesaikan langkah-langkah berikut ini:

   a. Pilih Marketplace AWS .

   b. Pilih AMI untuk instance kontainer Anda. Anda dapat mencari salah satu AMI yang dioptimalkan Amazon ECS, misalnya AMI Amazon Linux 2 yang dioptimalkan Amazon ECS. Jika Anda tidak memilih AMI Amazon ECS yang dioptimalkan, Anda harus mengikuti prosedur di. Memasang agen kontainer Amazon ECS

      Untuk informasi selengkapnya tentang AMI terbaru yang dioptimalkan Amazon ECS, lihat. AMI Amazon ECS yang dioptimalkan
5. Pada halaman Pilih Tipe Instans, pilih konfigurasi perangkat keras untuk instans Anda. Pilih tipe instans `t2.micro` yang dipilih secara default. Tipe instans yang Anda pilih menentukan sumber daya yang tersedia untuk menjalankan tugas Anda.

   Pilih Berikutnya: Konfigurasi Detail Instans saat Anda telah selesai.
6. Pada halaman Konfigurasi Detail Instans, lengkapi langkah-langkah berikut:

   a. Atur bidang Jumlah instans tergantung pada berapa banyak instans kontainer yang ingin ditambahkan ke klaster Anda.

   b. (Opsional) Untuk menggunakan Instans Spot, untuk opsi Pembelian, pilih kotak centang di sebelah Minta Instans Spot. Anda juga perlu mengatur bidang lain yang terkait dengan Instans Spot. Untuk informasi selengkapnya, lihat Permintaan Instans Spot.

      > **Note**
      >
      > Jika Anda menggunakan Instans Spot dan melihat `Not available` pesan, Anda mungkin perlu memilih jenis instans yang berbeda.

   c. Untuk Jaringan, pilih VPC yang akan digunakan untuk meluncurkan instans kontainer Anda.

   d. Untuk Subnet, pilih subnet yang akan digunakan, atau pertahankan opsi default untuk memilih subnet default di Availability Zone mana pun.

   e. Pengaturan bidang Penetapan Otomatis IP Publik bergantung pada apakah Anda ingin instans Anda dapat diakses dari internet publik. Jika instans Anda seharusnya dapat diakses dari internet, pastikan bahwa bidang Penetapan Otomatis IP Publik diatur ke Aktifkan. Jika tidak, atur bidang ini ke Nonaktifkan.

> **Note**
>
> Instans kontainer memerlukan akses untuk berkomunikasi dengan titik akhir layanan Amazon ECS. Ini dapat dilakukan melalui VPC endpoint antarmuka atau melalui instans kontainer yang memiliki alamat IP publik.
> Untuk informasi lebih lanjut tentang VPC endpoint antarmuka, lihat Titik akhir VPC antarmuka Amazon ECS ()AWS PrivateLink.
> Jika Anda tidak memiliki VPC endpoint yang dikonfigurasi dan instans kontainer Anda tidak memiliki alamat IP publik, network address translation (NAT) harus digunakan untuk menyediakan akses ini. Untuk informasi lebih lanjut, lihat Gateway NAT dalam Panduan Pengguna Amazon VPC dan Konfigurasi proxy HTTP untuk instance kontainer Linux dalam panduan ini. Untuk informasi selengkapnya, lihat the section called “Buat virtual private cloud”.

f. Pilih peran IAM instance container Anda. Hal ini biasanya bernama `ecsInstanceRole`.

> **Important**
>
> Jika Anda tidak meluncurkan instans penampung dengan izin IAM yang tepat, agen Amazon ECS Anda tidak dapat terhubung ke klaster Anda. Untuk informasi selengkapnya, lihat Peran IAM instans wadah Amazon ECS.

g. (Opsional) Konfigurasikan instans penampung Amazon ECS Anda dengan data pengguna, seperti variabel lingkungan agen dariKonfigurasi agen kontainer Amazon ECS. Skrip data pengguna Amazon EC2 dijalankan hanya satu kali, saat instans pertama kali diluncurkan. Berikut ini adalah contoh umum dari kegunaan data pengguna:

- Secara default, instans kontainer Anda meluncurkan ke klaster default Anda. Untuk meluncurkan ke klaster non-default, pilih daftar Detail Lanjutan. Kemudian, paste script berikut ke bidang Data pengguna, mengganti *`nama_klaster_anda`* dengan nama klaster Anda.

```
#!/bin/bash
echo ECS_CLUSTER=your_cluster_name >> /etc/ecs/ecs.config
```

- Jika Anda memiliki **`ecs.config`** file di Amazon S3 dan telah mengaktifkan akses hanya-baca Amazon S3 ke peran instance container, pilih daftar Detail Lanjutan. Kemudian,

tempel skrip berikut ke bidang Data pengguna, ganti *your_bucket_name dengan nama* bucket Anda untuk menginstal AWS CLI dan menulis file konfigurasi Anda pada waktu peluncuran.

> **Note**
>
> Untuk informasi lebih lanjut tentang konfigurasi ini, lihat Menyimpan konfigurasi instans kontainer di Amazon S3.

```
#!/bin/bash
yum install -y aws-cli
aws s3 cp s3://your_bucket_name/ecs.config /etc/ecs/ecs.config
```

- Tentukan tanda untuk instans kontainer Anda menggunakan parameter konfigurasi `ECS_CONTAINER_INSTANCE_TAGS`. Ini membuat tag yang terkait dengan Amazon ECS saja, mereka tidak dapat dicantumkan menggunakan Amazon EC2 API.

> **Important**
>
> Jika Anda meluncurkan instance container menggunakan grup Auto Scaling Amazon EC2, Anda harus menggunakan parameter konfigurasi agen ECS_CONTAINER_INSTANCE_TAGS untuk menambahkan tag. Hal ini disebabkan oleh cara tag ditambahkan ke instans Amazon EC2 yang diluncurkan menggunakan grup Auto Scaling.

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=your_cluster_name
ECS_CONTAINER_INSTANCE_TAGS={"tag_key": "tag_value"}
EOF
```

- Tentukan tag untuk instance container Anda, lalu gunakan parameter `ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM` konfigurasi untuk menyebarkannya dari Amazon EC2 ke Amazon ECS

Berikut adalah instans dari skrip data pengguna yang akan menyebarkan tanda yang terkait dengan instans kontainer, serta mendaftarkan instans kontainer dengan klaster bernama `your_cluster_name`:

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=your_cluster_name
ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM=ec2_instance
EOF
```

Untuk informasi selengkapnya, lihat Bootstrapping instance container dengan data pengguna Amazon EC2.

h. Pilih Berikutnya: Tambahkan Penyimpanan.

7. Pada halaman Tambahkan Penyimpanan , mengonfigurasi penyimpanan untuk instans kontainer Anda.

   Jika Anda menggunakan Amazon Linux 2 AMI Amazon ECS yang dioptimalkan Amazon, instans Anda memiliki satu volume 30 GiB yang dikonfigurasi, yang dibagi antara sistem operasi dan Docker.

   Jika Anda menggunakan AMI Amazon ECS yang dioptimalkan, instans Anda memiliki dua volume yang dikonfigurasi. Volume Root adalah untuk penggunaan sistem operasi, dan volume Amazon EBS kedua (dilampirkan ke`/dev/xvdcz`) adalah untuk penggunaan Docker.

   Anda dapat menambah atau mengurangi ukuran volume bagi instans Anda untuk memenuhi kebutuhan aplikasi Anda.

   Setelah selesai mengonfigurasi volume, pilih Berikutnya: Tambahkan Tanda.

8. Pada halaman Tambahkan Tanda , tentukan tanda dengan memberikan kombinasi kunci dan nilai untuk instans kontainer. Pilih Tambahkan tanda lain untuk menambahkan lebih dari satu tanda ke instans kontainer Anda. Untuk informasi selengkapnya tentang sumber daya, lihat Penandaan sumber daya Amazon ECS.

   Pilih Berikutnya: Konfigurasikan Grup Keamanan setelah Anda selesai.

9. Di halaman Konfigurasikan Grup Keamanan, gunakan grup keamanan untuk menentukan aturan firewall untuk instans kontainer Anda. Aturan ini menentukan lalu lintas jaringan masuk yang

dikirim ke instans kontainer Anda. Semua lalu lintas lainnya diabaikan. Pilih atau buat grup keamanan sebagai berikut, lalu pilih Tinjau dan Luncurkan.

10. Pada halaman Tinjau Peluncuran Instans, di bawah Grup Keamanan, Anda akan melihat bahwa wizard akan membuat dan memilih grup keamanan untuk Anda. Sebaliknya, pilih grup keamanan yang Anda buat di Siapkan untuk menggunakan Amazon ECS dengan menggunakan langkah-langkah berikut:

    a. Pilih Edit grup keamanan.

    b. Pada halaman Konfigurasi Grup Keamanan, pilih opsi Pilih grup keamanan yang ada.

    c. Pilih grup keamanan yang Anda buat untuk instans kontainer Anda dari daftar grup keamanan yang ada, dan pilih Tinjau dan Luncurkan.

11. Pada halaman Tinjau Peluncuran Instans, pilih Luncurkan.

12. Di kotak dialog Pilih pasangan kunci yang sudah ada atau buat pasangan kunci baru, pilih Pilih pasangan kunci yang sudah ada lalu pilih pasangan kunci yang Anda buat saat menyiapkan.

    Saat Anda siap, pilih bidang pengakuan, lalu pilih Luncurkan Instans.

13. Halaman konfirmasi memberi tahu Anda bahwa instans Anda akan diluncurkan. Pilih Lihat Instans untuk menutup halaman konfirmasi dan kembali ke konsol tersebut.

14. Pada layar Instans, Anda dapat melihat status instans Anda. Hanya butuh waktu singkat untuk meluncurkan suatu instans. Saat Anda meluncurkan instans, status awalnya adalah `pending`. Setelah instans dimulai, statusnya berubah menjadi `running`, dan ia menerima nama DNS publik. Jika kolom DNS Publik tersembunyi, pilih Tunjukan/Sembunyikan, DNS Publik.

Menggunakan Instans Spot

Instans Spot adalah instans Amazon EC2 yang tidak digunakan yang tersedia dengan harga lebih rendah dari harga Instans Sesuai Permintaan. Karena Instans Spot memungkinkan Anda meminta instans EC2 yang tidak digunakan dengan diskon besar, Anda dapat menurunkan biaya Amazon EC2 secara signifikan. Harga per jam untuk Instans Spot disebut harga Spot. Harga Spot untuk setiap jenis instans di setiap Availability Zone ditetapkan oleh Amazon EC2, dan disesuaikan secara bertahap berdasarkan penawaran dan permintaan jangka panjang untuk Instans Spot. Untuk informasi selengkapnya, lihat Instans Spot dalam Panduan Pengguna Amazon EC2 untuk Instans Linux.

Anda dapat mendaftarkan Instans Spot ke kluster Amazon ECS Anda. Untuk informasi selengkapnya, lihat Meluncurkan instans penampung Amazon ECS Linux.

Pengeringan Instans Spot

Amazon EC2 mengakhiri, menghentikan, atau hibernasi Instans Spot Anda ketika harga Spot melebihi harga maksimum untuk permintaan atau kapasitas Anda tidak lagi tersedia. Amazon EC2 menyediakan pemberitahuan interupsi dua menit Instans Spot untuk menghentikan dan menghentikan tindakan. Itu tidak memberikan pemberitahuan dua menit untuk tindakan hibernasi. Jika pengurasan Instans Spot Amazon ECS diaktifkan pada instans, ECS menerima pemberitahuan gangguan Instans Spot dan menempatkan instance dalam status. `DRAINING`

> **Important**
>
> Amazon ECS tidak menerima pemberitahuan dari Amazon EC2 saat instans dihapus oleh Auto Scaling Capacity Rebalancing. Untuk informasi selengkapnya, lihat Penyeimbangan Kembali Kapasitas Auto Scaling Amazon EC2.

Saat instance container disetel ke`DRAINING`, Amazon ECS mencegah tugas baru dijadwalkan untuk penempatan pada instance container. Tugas layanan pada instans kontainer pengurasan yang ada di status `PENDING` segera dihentikan. Jika ada instans kontainer di klaster yang tersedia, tugas layanan pengganti dimulai.

Pengurasan Instans Spot dimatikan secara default dan harus diaktifkan secara manual. Untuk mengaktifkan pengurasan Instance Spot untuk instance kontainer baru, saat meluncurkan instance penampung, tambahkan skrip berikut ke dalam bidang data Pengguna, ganti *`MyCluster`*dengan nama cluster untuk mendaftarkan instance kontainer.

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_ENABLE_SPOT_INSTANCE_DRAINING=true
EOF
```

Untuk informasi selengkapnya, lihat Meluncurkan instans penampung Amazon ECS Linux.

Untuk mengaktifkan pengurasan Instance Spot untuk instance kontainer yang ada

1. Connect ke Instans Spot melalui SSH.
2. Edit file `/etc/ecs/ecs.config` dan tambahkan berikut:

```
ECS_ENABLE_SPOT_INSTANCE_DRAINING=true
```

3. Mulai ulang layanan `ecs`.

   - Untuk Amazon ECS yang dioptimalkan Amazon Linux 2 AMI:

     ```
     sudo systemctl restart ecs
     ```

   - Untuk Amazon ECS yang dioptimalkan Amazon Linux AMI:

     ```
     sudo stop ecs && sudo start ecs
     ```

4. (Opsional) Anda dapat memastikan bahwa agen sedang berjalan dan melihat beberapa informasi tentang instans kontainer baru dengan melakukan kueri pada operasi API introspeksi agen. Untuk informasi selengkapnya, lihat the section called “Introspeksi wadah”.

   ```
   curl http://localhost:51678/v1/metadata
   ```

Pengurasan instans kontainer

Mungkin ada saat-saat ketika Anda perlu menghapus instance container dari cluster Anda; misalnya, untuk melakukan pembaruan sistem, memperbarui daemon Docker, atau untuk menurunkan kapasitas cluster. Amazon ECS menyediakan kemampuan untuk mentransisikan instans kontainer ke `DRAINING` status. Ini disebut sebagai pengurasan instans kontainer. Saat instance container disetel ke`DRAINING`, Amazon ECS mencegah tugas baru dijadwalkan untuk penempatan pada instance container.

Menguras perilaku untuk layanan

Tugas yang merupakan bagian dari layanan yang ada di status `PENDING` dihentikan dengan segera. Jika terdapat kapasitas instans kontainer dalam klaster, penjadwal layanan akan memulai tugas penggantian. Jika kapasitas instans kontainer tidak cukup, pesan peristiwa layanan yang menunjukkan masalah tersebut akan dikirim.

Tugas yang merupakan bagian dari layanan pada instans kontainer yang berada di status `RUNNING` dialihkan ke status `STOPPED`. Penjadwal layanan mencoba mengganti tugas sesuai dengan jenis penyebaran layanan dan parameter konfigurasi penerapan, dan. `minimumHealthyPercent`

`maximumPercent` Lihat informasi yang lebih lengkap di Jenis Penerapan Amazon ECS dan Parameter ketentuan layanan.

- Jika `minimumHealthyPercent` di bawah 100%, penjadwal dapat mengabaikan `desiredCount` untuk sementara selama penggantian tugas. Misalnya, `desiredCount` adalah empat tugas, setidaknya 50% mengizinkan penjadwal menghentikan dua tugas yang ada sebelum memulai dua tugas baru. Jika minimumnya adalah 100%, penjadwal layanan tidak dapat menghapus tugas yang ada hingga tugas penggantian dianggap sehat. Jika tugas untuk layanan yang tidak menggunakan penyeimbang beban berada di status `RUNNING`, maka mereka dianggap sehat. Tugas untuk layanan yang menggunakan penyeimbang beban dianggap sehat jika berada di status `RUNNING` dan instans kontainer tempat mereka di-host dilaporkan dalam kondisi baik oleh penyeimbang beban.

  > **Important**
  >
  > Jika Anda menggunakan Instans Spot dan `minimumHealthyPercent` lebih besar dari atau sama dengan 100%, maka layanan tidak akan memiliki cukup waktu untuk mengganti tugas sebelum Instans Spot berakhir.

- `maximumPercent`Parameter mewakili batas atas jumlah tugas yang berjalan selama penggantian tugas, yang memungkinkan Anda menentukan ukuran batch pengganti. Contohnya, jika `desiredCount` dari empat tugas, maksimal 200% memulai empat tugas baru sebelum menghentikan empat tugas yang akan dikuras (asalkan sumber daya klaster yang diperlukan untuk melakukan ini tersedia). Jika maksimumnya adalah 100%, maka tugas penggantian tidak dapat dimulai hingga tugas pengeringan berhenti.

  > **Important**
  >
  > Jika `maximumPercent` keduanya `minimumHealthyPercent` dan 100%, maka layanan tidak dapat menghapus tugas yang ada, dan juga tidak dapat memulai tugas penggantian. Ini mencegah pengeringan instance kontainer yang berhasil dan mencegah pembuatan penerapan baru.

Menguras perilaku untuk tugas mandiri

Setiap tugas mandiri di status `PENDING` atau `RUNNING` tidak terpengaruh; Anda harus menunggu tugas tersebut berhenti sendiri atau menghentikannya secara manual. Instance kontainer akan tetap dalam `DRAINING` status.

Sebuah instans kontainer telah selesai menguras ketika semua tugas yang berjalan di ditransisi ke status `STOPPED`. Instans kontainer tetap dalam status `DRAINING` hingga diaktifkan kembali atau dihapus. Anda dapat memverifikasi status tugas pada instance container dengan menggunakan ListTasksoperasi dengan `containerInstance` parameter untuk mendapatkan daftar tugas pada instance diikuti dengan DescribeTasksoperasi dengan Amazon Resource Name (ARN) atau ID dari setiap tugas untuk memverifikasi status tugas.

Ketika Anda merasa instans kontainer siap untuk memulai tugas menghosting kembali, maka Anda mengubah status instans kontainer dari `DRAINING` ke `ACTIVE`. Penjadwal layanan Amazon ECS kemudian akan mempertimbangkan instance kontainer untuk penempatan tugas lagi.

Menguras instans kontainer

Langkah-langkah berikut dapat digunakan untuk mengatur instance kontainer ke pengeringan menggunakan yang baru AWS Management Console.

Anda juga dapat menggunakan tindakan UpdateContainerInstancesStateAPI atau update-container-instances-stateperintah untuk mengubah status instance container menjadi`DRAINING`.

AWS Management Console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih klaster yang meng-host instance Anda.
4. Pada halaman Cluster: ***name***, pilih tab Infrastructure. Kemudian, di bawah Contoh kontainer pilih kotak centang untuk setiap instance kontainer yang ingin Anda tiriskan.
5. Pilih Tindakan, Tiriskan.

Bootstrapping instance container dengan data pengguna Amazon EC2

Saat meluncurkan instans Amazon EC2, Anda memiliki opsi untuk meneruskan data pengguna ke instans. Data dapat digunakan untuk melakukan tugas konfigurasi otomatis umum dan bahkan

menjalankan skrip ketika boot instans. Untuk Amazon ECS, kasus penggunaan yang paling umum untuk data pengguna adalah meneruskan informasi konfigurasi ke daemon Docker dan agen penampung Amazon ECS.

Anda dapat meneruskan beberapa jenis data pengguna ke Amazon EC2, termasuk boothook cloud, skrip shell, dan arahan. `cloud-init` Untuk informasi selengkapnya tentang hal ini dan tipe format lainnya, lihat dokumentasi Cloud-init.

Anda dapat meneruskan data pengguna ini saat menggunakan wizard peluncuran Amazon EC2. Untuk informasi selengkapnya, lihat Meluncurkan instans penampung Amazon ECS Linux.

Topik

- Agen kontainer Amazon ECS
- Daemon docker

Agen kontainer Amazon ECS

Varian Linux dari AMI Amazon ECS yang dioptimalkan mencari data konfigurasi agen dalam `/etc/ecs/ecs.config` file saat agen penampung dimulai. Anda dapat menentukan data konfigurasi ini saat diluncurkan dengan data pengguna Amazon EC2. Untuk informasi selengkapnya tentang variabel konfigurasi agen penampung Amazon ECS yang tersedia, lihatKonfigurasi agen kontainer Amazon ECS.

Untuk mengatur hanya pada variabel konfigurasi agen tunggal, seperti nama klaster, gunakan echo untuk menyalin variabel ke file konfigurasi:

```
#!/bin/bash
echo "ECS_CLUSTER=MyCluster" >> /etc/ecs/ecs.config
```

Jika Anda memiliki beberapa variabel untuk ditulis pada `/etc/ecs/ecs.config`, gunakan pilihan berikut ini pada format `heredoc`. Format ini menulis segalanya di antara baris dimulai dengan cat dan EOF pada file konfigurasi.

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_ENGINE_AUTH_TYPE=docker
ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
{"username":"my_name","password":"my_password","email":"email@example.com"}}
```

```
ECS_LOGLEVEL=debug
ECS_WARM_POOLS_CHECK=true
EOF
```

Untuk mengatur atribut instance kustom, atur variabel `ECS_INSTANCE_ATTRIBUTES` lingkungan.

```
#!/bin/bash
cat <<'EOF' >> ecs.config
ECS_INSTANCE_ATTRIBUTES={"envtype":"prod"}
EOF
```

Daemon docker

Anda dapat menentukan informasi konfigurasi daemon Docker dengan data pengguna Amazon EC2. Untuk informasi selengkapnya tentang pilihan konfigurasi, lihat dokumentasi daemon Docker.

Pada contoh di bawah ini, pilihan kustom ditambahkan ke file konfigurasi daemon Docker, `/etc/docker/daemon.json` yang kemudian ditentukan dalam data pengguna saat instans diluncurkan.

```
#!/bin/bash
cat <<EOF >/etc/docker/daemon.json
{"debug": true}
EOF
systemctl restart docker --no-block
```

Pada contoh di bawah ini, pilihan kustom ditambahkan ke file konfigurasi daemon Docker, `/etc/docker/daemon.json` yang kemudian ditentukan dalam data pengguna saat instans diluncurkan. Contoh ini menunjukkan cara menonaktifkan docker-proxy di file konfigurasi daemon Docker.

```
#!/bin/bash
cat <<EOF >/etc/docker/daemon.json
{"userland-proxy": false}
EOF
systemctl restart docker --no-block
```

Memulai tugas pada waktu peluncuran instans kontainer

Bergantung pada desain arsitektur aplikasi Anda, Anda mungkin perlu menjalankan kontainer tertentu pada setiap instans kontainer untuk menangani operasi atau masalah keamanan seperti pemantauan, keamanan, metrik, penemuan layanan, atau pencatatan.

Untuk melakukannya, Anda dapat mengonfigurasi instans kontainer Anda untuk memanggil perintah docker run dengan skrip data pengguna saat peluncuran, atau dalam beberapa sistem unit seperti Upstart atau systemd. Meskipun metode ini berfungsi, ia memiliki beberapa kelemahan karena Amazon ECS tidak memiliki pengetahuan tentang wadah dan tidak dapat memantau CPU, memori, port, atau sumber daya lain yang digunakan. Untuk memastikan Amazon ECS dapat memperhitungkan semua sumber daya tugas dengan benar, buat definisi tugas agar container dapat dijalankan pada instance container Anda. Kemudian, gunakan Amazon ECS untuk menempatkan tugas pada waktu peluncuran dengan data pengguna Amazon EC2.

Skrip data pengguna Amazon EC2 dalam prosedur berikut menggunakan API introspeksi Amazon ECS untuk mengidentifikasi instance container. Kemudian, ia menggunakan AWS CLI dan start-task perintah untuk menjalankan tugas tertentu pada dirinya sendiri selama startup.

Untuk memulai tugas pada waktu peluncuran instans kontainer

1. Jika Anda belum melakukannya, buat ketentuan tugas dengan kontainer yang ingin Anda jalankan pada instans kontainer saat peluncuran dengan mengikuti prosedur di Membuat definisi tugas menggunakan konsol.
2. Modifikasi `ecsInstanceRole` IAM role Anda untuk menambahkan izin untuk Operasi API `StartTask` . Untuk informasi selengkapnya, lihat Peran IAM instans wadah Amazon ECS.
   a. Buka konsol IAM di https://console.aws.amazon.com/iam/.
   b. Di panel navigasi, pilih Peran.
   c. Pilih `ecsInstanceRole`. Jika peran tidak ada, gunakan prosedur di Peran IAM instans wadah Amazon ECS untuk membuat peran dan kembali ke prosedur ini. Jika peran ada, pilih peran tersebut untuk melihat kebijakan terlampir.
   d. Pada tab Izin, pilih Tambahkan kebijakan inline.
   e. Untuk Layanan, pilih Pilih layanan, Layanan Kontainer Elastis.
   f. Untuk Tindakan, ketik StartTaskdi bidang pencarian, lalu pilih StartTask.
   g. Untuk Sumber Daya, pilih Semua sumber daya, lalu pilih Tinjau kebijakan.
   h. Pada halaman Tinjau kebijakan, masukkan nama untuk kebijakan Anda, seperti `ecs-start-task` dan pilih Buat kebijakan.
3. Luncurkan satu atau beberapa instance kontainer menggunakan Amazon Linux 2 AMI yang dioptimalkan Amazon ECS dengan mengikuti prosedur Meluncurkan instans penampung Amazon ECS Linux di, tetapi Step 6.g dalam salin dan tempel skrip data pengguna multi-bagian MIME di bawah ini ke bidang Data pengguna. Ganti *`nama_klaster_anda`* dengan klaster untuk

instans kontainer untuk mendaftar ke dan *my_task_def* dengan ketentuan tugas yang berjalan pada instans saat peluncuran.

> **Note**
>
> Konten multibagian MIME di bawah ini menggunakan skrip shell untuk menetapkan nilai konfigurasi dan menginstal paket. Hal ini juga menggunakan tugas sistem untuk memulai tugas setelah layanan ecs berjalan dan API introspeksi tersedia.

```
Content-Type: multipart/mixed; boundary="==BOUNDARY=="
MIME-Version: 1.0

--==BOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
# Specify the cluster that the container instance should register into
cluster=your_cluster_name

# Write the cluster configuration variable to the ecs.config file
# (add any other configuration variables here also)
echo ECS_CLUSTER=$cluster >> /etc/ecs/ecs.config

START_TASK_SCRIPT_FILE="/etc/ecs/ecs-start-task.sh"
cat <<- 'EOF' > ${START_TASK_SCRIPT_FILE}
 exec 2>>/var/log/ecs/ecs-start-task.log
 set -x

 # Install prerequisite tools
 yum install -y jq aws-cli

 # Wait for the ECS service to be responsive
 until curl -s http://localhost:51678/v1/metadata
 do
  sleep 1
 done

 # Grab the container instance ARN and AWS Region from instance metadata
 instance_arn=$(curl -s http://localhost:51678/v1/metadata | jq -r '.
 | .ContainerInstanceArn' | awk -F/ '{print $NF}' )
```

```
 cluster=$(curl -s http://localhost:51678/v1/metadata | jq -r '. | .Cluster' | awk
 -F/ '{print $NF}' )
 region=$(curl -s http://localhost:51678/v1/metadata | jq -r '.
 | .ContainerInstanceArn' | awk -F: '{print $4}')

 # Specify the task definition to run at launch
 task_definition=my_task_def

 # Run the AWS CLI start-task command to start your task on this container instance
 aws ecs start-task --cluster $cluster --task-definition $task_definition --
container-instances $instance_arn --started-by $instance_arn --region $region
EOF

# Write systemd unit file
UNIT="ecs-start-task.service"
cat <<- EOF > /etc/systemd/system/${UNIT}
      [Unit]
      Description=ECS Start Task
      Requires=ecs.service
      After=ecs.service

      [Service]
      Restart=on-failure
      RestartSec=30
      ExecStart=/usr/bin/bash ${START_TASK_SCRIPT_FILE}

      [Install]
      WantedBy=default.target
EOF

# Enable our ecs.service dependent service with `--no-block` to prevent systemd
 deadlock
# See https://github.com/aws/amazon-ecs-agent/issues/1707
systemctl enable --now --no-block "${UNIT}"
--===BOUNDARY==--
```

4. Verifikasi bahwa instans kontainer Anda meluncurkan ke klaster yang benar dan bahwa tugas Anda telah dimulai.

   a. Buka konsol di https://console.aws.amazon.com/ecs/v2.

   b. Dari bilah navigasi, pilih Wilayah tempat klaster Anda berada.

   c. Di panel navigasi, pilih Klaster dan pilih klaster yang meng-host instans kontainer Anda.

d. Pada halaman Cluster, pilih Tasks, lalu pilih yor task.

   Setiap instance kontainer yang Anda luncurkan harus menjalankan tugas Anda di atasnya.

   Jika Anda tidak melihat tugas Anda, Anda dapat masuk ke instans kontainer Anda dengan SSH dan memeriksa file `/var/log/ecs/ecs-start-task.log` untuk informasi debugging.

Pembuatan torso antarmuka jaringan elastis

> **Note**
>
> Fitur ini tidak tersedia di Fargate.

Setiap tugas Amazon ECS yang menggunakan mode `awsvpc` jaringan menerima elastic network interface (ENI) miliknya sendiri, yang dilampirkan ke instance container yang menghostingnya. Ada batasan default untuk jumlah antarmuka jaringan yang dapat dilampirkan ke instans Amazon EC2, dan antarmuka jaringan utama dihitung sebagai satu. Sebagai contoh, secara default instans `c5.large` mungkin memiliki hingga tiga ENI yang dilampirkan. Antarmuka jaringan utama untuk instans dihitung sebagai satu, sehingga Anda dapat melampirkan dua ENI tambahan ke instans. Karena setiap tugas yang menggunakan mode `awsvpc` jaringan memerlukanENI, Anda biasanya hanya dapat menjalankan dua tugas tersebut pada jenis instance ini.

Amazon ECS mendukung peluncuran instans kontainer dengan peningkatan ENI kepadatan menggunakan jenis instans Amazon EC2 yang didukung. Saat Anda menggunakan jenis instans ini dan mengaktifkan setelan `awsvpcTrunking` akun, ENI tambahan tersedia pada instance container yang baru diluncurkan. Konfigurasi ini mengizinkan Anda untuk menempatkan lebih banyak tugas dengan menggunakan mode jaringan `awsvpc` pada setiap instans kontainer. Untuk informasi tentang pengaturan `awsvpcTrunking` akun, lihatMengakses fitur Amazon ECS melalui pengaturan akun.

Menggunakan fitur ini, `c5.large` instance dengan `awsvpcTrunking` diaktifkan memiliki ENI batas peningkatan dua belas. Instance container akan memiliki antarmuka jaringan utama dan Amazon ECS membuat dan melampirkan antarmuka jaringan “trunk” ke instance container. Jadi, konfigurasi ini mengizinkan Anda meluncurkan sepuluh tugas pada instans kontainer, tidak hanya dua tugas seperti saat ini.

Antarmuka jaringan trunk dikelola sepenuhnya oleh Amazon ECS dan dihapus saat Anda menghentikan atau membatalkan pendaftaran instance container dari cluster. Untuk informasi selengkapnya, lihat Jaringan tugas untuk tugas di instans Amazon EC2.

Pertimbangan

Ada beberapa hal yang perlu dipertimbangkan saat menggunakan fitur ENI trunking.

- Hanya varian Linux dari AMI Amazon ECS yang dioptimalkan, atau varian Amazon Linux lainnya dengan versi `1.28.1` atau yang lebih baru dari agen kontainer dan versi `1.28.1-2` atau yang lebih baru dari paket ecs-init, yang mendukung peningkatan batas. ENI Jika Anda menggunakan varian Linux terbaru dari AMI Amazon ECS yang dioptimalkan, persyaratan ini akan dipenuhi. Kontainer Windows tidak didukung saat ini.
- Hanya instans Amazon EC2 baru yang diluncurkan setelah mengaktifkan `awsvpcTrunking` menerima peningkatan ENI batas dan antarmuka jaringan trunk. Instans yang diluncurkan sebelumnya tidak menerima fitur ini tanpa memperhatikan tindakan yang dilakukan.
- Instans Amazon EC2 harus menonaktifkan permintaan DNS IPv4 berbasis sumber daya. Untuk menonaktifkan opsi ini, pastikan opsi Aktifkan permintaan DNS IPV4 (Catatan) berbasis sumber daya tidak dipilih saat membuat instance baru menggunakan konsol Amazon EC2. Untuk menonaktifkan opsi ini menggunakan AWS CLI, gunakan perintah berikut.

  ```
  aws ec2 modify-private-dns-name-options --instance-id i-xxxxxxx --no-enable-resource-name-dns-a-record --no-dry-run
  ```

- Instans Amazon EC2 di subnet bersama tidak didukung. Instans tersebut akan gagal untuk didaftarkan pada klaster jika instans digunakan.
- Tugas Amazon ECS Anda harus menggunakan mode `awsvpc` jaringan dan jenis peluncuran EC2. Tugas yang menggunakan tipe peluncuran Fargate selalu menerima dedicated ENI terlepas dari berapa banyak yang diluncurkan, jadi fitur ini tidak diperlukan.
- Tugas Amazon ECS Anda harus diluncurkan di VPC Amazon yang sama dengan instans penampung Anda. Tugas Anda akan gagal untuk dimulai bersama dengan sebuah kesalahan pada atribut jika tugas tersebut tidak berada dalam VPC yang sama.
- Ketika meluncurkan sebuah instans kontainer yang baru, transisi instans menuju status `REGISTERING` sementara antarmuka jaringan elastis torso telah disediakan untuk instans. Jika pendaftaran gagal, transisi instans menuju status `REGISTRATION_FAILED`. Anda dapat memecahkan masalah pada gagalnya pendaftaran dengan menjelaskan instans kontainer untuk melihat bidang `statusReason` yang menjelaskan alasan pada kegagalan. instans kontainer

kemudian secara manual dapat membatalkan pendaftaran atau mengakhirinya. Setelah instance kontainer berhasil dideregistrasi atau dihentikan, Amazon ECS akan menghapus bagasi. ENI

> **Note**
>
> Amazon ECS memancarkan peristiwa perubahan status instans kontainer yang dapat Anda pantau untuk instance yang bertransisi ke status. `REGISTRATION_FAILED` Untuk informasi selengkapnya, lihat Acara perubahan status instans penampung Amazon ECS.

- Setelah instans kontainer dihentikan, transisi instans menuju status `DEREGISTERING` sementara torso pada antarmuka jaringan elastis telah dicabut. Instans tersebut kemudian melakukan transisi menuju status `INACTIVE`.
- Jika instance kontainer di subnet publik dengan ENI batas yang meningkat dihentikan dan kemudian dimulai ulang, instance kehilangan alamat IP publiknya, dan agen kontainer kehilangan koneksinya.
- Saat Anda mengaktifkan`awsvpcTrunking`, instance container menerima tambahan ENI yang menggunakan grup keamanan default VPC, dan dikelola oleh Amazon ECS.

Prasyarat

Sebelum Anda meluncurkan instance kontainer dengan ENI batas yang meningkat, prasyarat berikut harus diselesaikan.

- Peran terkait layanan untuk Amazon ECS harus dibuat. Peran terkait layanan Amazon ECS memberi Amazon ECS izin untuk melakukan panggilan ke layanan lain AWS atas nama Anda. Peran ini dibuat untuk Anda secara otomatis ketika Anda membuat sebuah klaster, atau jika Anda membuat atau memperbarui layanan di AWS Management Console. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS. Anda juga dapat membuat peran terkait layanan dengan perintah berikut AWS CLI .

```
aws iam create-service-linked-role --aws-service-name ecs.amazonaws.com
```

- Peran IAM instans akun atau kontainer Anda harus mengaktifkan pengaturan `awsvpcTrunking` akun. Kami menyarankan Anda membuat 2 container instance role (`ecsInstanceRole`). Anda kemudian dapat mengaktifkan pengaturan `awsvpcTrunking` akun untuk satu peran dan menggunakan peran tersebut untuk tugas yang memerlukan trunking ENI. Untuk informasi tentang peran instance container, lihatPeran IAM instans wadah Amazon ECS.

Setelah prasyarat terpenuhi, Anda dapat meluncurkan instance container baru menggunakan salah satu jenis instans Amazon EC2 yang didukung, dan instance akan memiliki batas yang ditingkatkan. ENI Untuk daftar tipe data yang didukung, lihat Tipe instans Amazon EC2 yang didukung. instans kontainer harus memiliki versi `1.28.1` atau yang lebih baru dari agen kontainer dan versi `1.28.1-2` atau yang lebih baru dari paket ecs-init. Jika Anda menggunakan varian Linux terbaru dari AMI Amazon ECS yang dioptimalkan, persyaratan ini akan dipenuhi. Untuk informasi selengkapnya, lihat Meluncurkan instans penampung Amazon ECS Linux.

> **Important**
>
> Instans Amazon EC2 harus menonaktifkan permintaan DNS IPv4 berbasis sumber daya. Untuk menonaktifkan opsi ini, pastikan opsi Aktifkan permintaan DNS IPV4 (Catatan) berbasis sumber daya tidak dipilih saat membuat instance baru menggunakan konsol Amazon EC2. Untuk menonaktifkan opsi ini menggunakan AWS CLI, gunakan perintah berikut.
>
> ```
> aws ec2 modify-private-dns-name-options --instance-id i-xxxxxxx --no-enable-resource-name-dns-a-record --no-dry-run
> ```

Untuk melihat instance penampung Anda dengan ENI batas yang ditingkatkan dengan AWS CLI

Setiap instans kontainer memiliki antarmuka jaringan default, yang disebut sebagai antarmuka jaringan torso. Gunakan perintah berikut untuk membuat daftar instance kontainer Anda dengan ENI batas yang ditingkatkan dengan menanyakan `ecs.awsvpc-trunk-id` atribut, yang menunjukkan bahwa ia memiliki antarmuka jaringan trunk.

- daftar-atribut (AWS CLI)

```
aws ecs list-attributes \
      --target-type container-instance \
      --attribute-name ecs.awsvpc-trunk-id \
      --cluster cluster_name \
      --region us-east-1
```

- Dapatkan-ECS (AttributeList)AWS Tools for Windows PowerShell

```
Get-ECSAttributeList -TargetType container-instance -AttributeName ecs.awsvpc-trunk-id -Region us-east-1
```

Tipe instans Amazon EC2 yang didukung

Berikut ini menunjukkan jenis instans Amazon EC2 yang didukung dan berapa banyak tugas yang menggunakan mode `awsvpc` jaringan yang dapat diluncurkan pada setiap jenis instans sebelum dan sesudah mengaktifkan pengaturan akun. `awsvpcTrunking` Untuk batas elastic network interface (ENI) pada setiap jenis instance, tambahkan satu ke batas tugas saat ini, karena antarmuka jaringan utama diperhitungkan terhadap batas, dan tambahkan dua ke batas tugas baru, karena antarmuka jaringan utama dan antarmuka jaringan trunk menghitung lagi batas.

Important

Meskipun jenis instance lain didukung dalam keluarga instance yang sama, tipe `a1.metalc5.metal`,`c5a.8xlarge`,`c5ad.8xlarge`, `c5d.metalm5.metal`,`p3dn.24xlarge`,`r5.metal`,`r5.8xlarge`,, dan `r5d.metal` instance tidak didukung.
Keluarga`c5n`,`d3`, `d3eng3`,`g3s`,`g4dn`,`i3`,`i3en`,`inf1`,`m5dn`,`m5n`, `m5znmac1`,`r5b`,`r5n`,`r5dn`,`u-12tb1`,`u-6tb1`,`u-9tb1`,, dan `z1d` contoh tidak didukung.

Topik

Tujuan umum

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
a1.medium	1	10
a1.large	2	10
a1.xlarge	3	20

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
a1.2xlarge	3	40
a1.4xlarge	7	60
m5.large	2	10
m5.xlarge	3	20
m5.2xlarge	3	40
m5.4xlarge	7	60
m5.8xlarge	7	60
m5.12xlarge	7	60
m5.16xlarge	14	120
m5.24xlarge	14	120
m5a.large	2	10
m5a.xlarge	3	20
m5a.2xlarge	3	40
m5a.4xlarge	7	60
m5a.8xlarge	7	60
m5a.12xlarge	7	60
m5a.16xlarge	14	120
m5a.24xlarge	14	120
m5ad.large	2	10
m5ad.xlarge	3	20

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
m5ad.2xlarge	3	40
m5ad.4xlarge	7	60
m5ad.8xlarge	7	60
m5ad.12xlarge	7	60
m5ad.16xlarge	14	120
m5ad.24xlarge	14	120
m5d.large	2	10
m5d.xlarge	3	20
m5d.2xlarge	3	40
m5d.4xlarge	7	60
m5d.8xlarge	7	60
m5d.12xlarge	7	60
m5d.16xlarge	14	120
m5d.24xlarge	14	120
m5d.metal	14	120
m5n.large	2	10
m5n.xlarge	3	20
m5n.2xlarge	3	40
m5n.4xlarge	7	60
m5n.8xlarge	7	60

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
m5n.12xlarge	7	60
m5n.16xlarge	14	120
m5zn.large	2	14
m5zn.xlarge	3	31
m5zn.2xlarge	3	64
m5zn.3xlarge	7	98
m5zn.6xlarge	7	120
m6a.large	2	10
m6a.xlarge	3	20
m6a.2xlarge	3	40
m6a.4xlarge	7	60
m6a.8xlarge	7	90
m6a.12xlarge	7	120
m6a.16xlarge	14	120
m6a.24xlarge	14	120
m6a.32xlarge	14	120
m6a.48xlarge	14	120
m6a.metal	14	120
m6g.medium	1	4
m6g.large	2	10

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
m6g.xlarge	3	20
m6g.2xlarge	3	40
m6g.4xlarge	7	60
m6g.8xlarge	7	60
m6g.12xlarge	7	60
m6g.16xlarge	14	120
m6g.metal	14	120
m6gd.medium	1	4
m6gd.large	2	10
m6gd.xlarge	3	20
m6gd.2xlarge	3	40
m6gd.4xlarge	7	60
m6gd.8xlarge	7	60
m6gd.12xlarge	7	60
m6gd.16xlarge	14	120
m6gd.metal	14	120
m6i.large	2	10
m6i.xlarge	3	20
m6i.2xlarge	3	40
m6i.4xlarge	7	60

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
m6i.8xlarge	7	90
m6i.12xlarge	7	120
m6i.16xlarge	14	120
m6i.24xlarge	14	120
m6i.32xlarge	14	120
m6i.metal	14	120
m6id.large	2	10
m6id.xlarge	3	20
m6id.2xlarge	3	40
m6id.4xlarge	7	60
m6id.8xlarge	7	90
m6id.12xlarge	7	120
m6id.16xlarge	14	120
m6id.24xlarge	14	120
m6id.32xlarge	14	120
m6id.metal	14	120
m6idn.large	2	10
m6idn.xlarge	3	20
m6idn.2xlarge	3	40
m6idn.4xlarge	7	60

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
m6idn.8xlarge	7	90
m6idn.12xlarge	7	120
m6idn.16xlarge	14	120
m6idn.24xlarge	14	120
m6idn.32xlarge	13	120
m6idn.metal	13	120
m6in.large	2	10
m6in.xlarge	3	20
m6in.2xlarge	3	40
m6in.4xlarge	7	60
m6in.8xlarge	7	90
m6in.12xlarge	7	120
m6in.16xlarge	14	120
m6in.24xlarge	14	120
m6in.32xlarge	13	120
m6in.metal	13	120
m7a.medium	1	4
m7a.large	2	10
m7a.xlarge	3	20
m7a.2xlarge	3	40

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
m7a.4xlarge	7	60
m7a.8xlarge	7	90
m7a.12xlarge	7	120
m7a.16xlarge	14	120
m7a.24xlarge	14	120
m7a.32xlarge	14	120
m7a.48xlarge	14	120
m7a.metal-48xl	14	120
m7g.medium	1	4
m7g.large	2	10
m7g.xlarge	3	20
m7g.2xlarge	3	40
m7g.4xlarge	7	60
m7g.8xlarge	7	60
m7g.12xlarge	7	60
m7g.16xlarge	14	120
m7g.metal	14	120
m7gd.medium	1	4
m7gd.large	2	10
m7gd.xlarge	3	20

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
m7gd.2xlarge	3	40
m7gd.4xlarge	7	60
m7gd.8xlarge	7	60
m7gd.12xlarge	7	60
m7gd.16xlarge	14	120
m7gd.metal	14	120
m7i.large	2	10
m7i.xlarge	3	20
m7i.2xlarge	3	40
m7i.4xlarge	7	60
m7i.8xlarge	7	90
m7i.12xlarge	7	120
m7i.16xlarge	14	120
m7i.24xlarge	14	120
m7i.48xlarge	14	120
m7i.metal-24xl	14	120
m7i.metal-48xl	14	120
m7i-flex.large	2	4
m7i-flex.xlarge	3	10
m7i-flex.2xlarge	3	20

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
m7i-flex.4xlarge	7	40
m7i-flex.8xlarge	7	60
mac2.metal	7	12
mac2-m2.metal	7	12
mac2-m2pro.metal	7	12

Komputasi yang dioptimalkan

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
c5.large	2	10
c5.xlarge	3	20
c5.2xlarge	3	40
c5.4xlarge	7	60
c5.9xlarge	7	60
c5.12xlarge	7	60
c5.18xlarge	14	120
c5.24xlarge	14	120
c5a.large	2	10
c5a.xlarge	3	20
c5a.2xlarge	3	40

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
c5a.4xlarge	7	60
c5a.12xlarge	7	60
c5a.16xlarge	14	120
c5a.24xlarge	14	120
c5ad.large	2	10
c5ad.xlarge	3	20
c5ad.2xlarge	3	40
c5ad.4xlarge	7	60
c5ad.12xlarge	7	60
c5ad.16xlarge	14	120
c5ad.24xlarge	14	120
c5d.large	2	10
c5d.xlarge	3	20
c5d.2xlarge	3	40
c5d.4xlarge	7	60
c5d.9xlarge	7	60
c5d.12xlarge	7	60
c5d.18xlarge	14	120
c5d.24xlarge	14	120
c6a.large	2	10

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
c6a.xlarge	3	20
c6a.2xlarge	3	40
c6a.4xlarge	7	60
c6a.8xlarge	7	90
c6a.12xlarge	7	120
c6a.16xlarge	14	120
c6a.24xlarge	14	120
c6a.32xlarge	14	120
c6a.48xlarge	14	120
c6a.metal	14	120
c6g.medium	1	4
c6g.large	2	10
c6g.xlarge	3	20
c6g.2xlarge	3	40
c6g.4xlarge	7	60
c6g.8xlarge	7	60
c6g.12xlarge	7	60
c6g.16xlarge	14	120
c6g.metal	14	120
c6gd.medium	1	4

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
c6gd.large	2	10
c6gd.xlarge	3	20
c6gd.2xlarge	3	40
c6gd.4xlarge	7	60
c6gd.8xlarge	7	60
c6gd.12xlarge	7	60
c6gd.16xlarge	14	120
c6gd.metal	14	120
c6gn.medium	1	4
c6gn.large	2	10
c6gn.xlarge	3	20
c6gn.2xlarge	3	40
c6gn.4xlarge	7	60
c6gn.8xlarge	7	60
c6gn.12xlarge	7	60
c6gn.16xlarge	14	120
c6i.large	2	10
c6i.xlarge	3	20
c6i.2xlarge	3	40
c6i.4xlarge	7	60

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
c6i.8xlarge	7	90
c6i.12xlarge	7	120
c6i.16xlarge	14	120
c6i.24xlarge	14	120
c6i.32xlarge	14	120
c6i.metal	14	120
c6id.large	2	10
c6id.xlarge	3	20
c6id.2xlarge	3	40
c6id.4xlarge	7	60
c6id.8xlarge	7	90
c6id.12xlarge	7	120
c6id.16xlarge	14	120
c6id.24xlarge	14	120
c6id.32xlarge	14	120
c6id.metal	14	120
c6in.large	2	10
c6in.xlarge	3	20
c6in.2xlarge	3	40
c6in.4xlarge	7	60

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
c6in.8xlarge	7	90
c6in.12xlarge	7	120
c6in.16xlarge	14	120
c6in.24xlarge	14	120
c6in.32xlarge	13	120
c6in.metal	13	120
c7a.medium	1	4
c7a.large	2	10
c7a.xlarge	3	20
c7a.2xlarge	3	40
c7a.4xlarge	7	60
c7a.8xlarge	7	90
c7a.12xlarge	7	120
c7a.16xlarge	14	120
c7a.24xlarge	14	120
c7a.32xlarge	14	120
c7a.48xlarge	14	120
c7a.metal-48xl	14	120
c7g.medium	1	4
c7g.large	2	10

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
c7g.xlarge	3	20
c7g.2xlarge	3	40
c7g.4xlarge	7	60
c7g.8xlarge	7	60
c7g.12xlarge	7	60
c7g.16xlarge	14	120
c7g.metal	14	120
c7gd.medium	1	4
c7gd.large	2	10
c7gd.xlarge	3	20
c7gd.2xlarge	3	40
c7gd.4xlarge	7	60
c7gd.8xlarge	7	60
c7gd.12xlarge	7	60
c7gd.16xlarge	14	120
c7gd.metal	14	120
c7gn.medium	1	4
c7gn.large	2	10
c7gn.xlarge	3	20
c7gn.2xlarge	3	40

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
c7gn.4xlarge	7	60
c7gn.8xlarge	7	60
c7gn.12xlarge	7	60
c7gn.16xlarge	14	120
c7i.large	2	10
c7i.xlarge	3	20
c7i.2xlarge	3	40
c7i.4xlarge	7	60
c7i.8xlarge	7	90
c7i.12xlarge	7	120
c7i.16xlarge	14	120
c7i.24xlarge	14	120
c7i.48xlarge	14	120
c7i.metal-24xl	14	120
c7i.metal-48xl	14	120

Memori yang dioptimalkan

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
r5.large	2	10

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
r5.xlarge	3	20
r5.2xlarge	3	40
r5.4xlarge	7	60
r5.12xlarge	7	60
r5.16xlarge	14	120
r5.24xlarge	14	120
r5a.large	2	10
r5a.xlarge	3	20
r5a.2xlarge	3	40
r5a.4xlarge	7	60
r5a.8xlarge	7	60
r5a.12xlarge	7	60
r5a.16xlarge	14	120
r5a.24xlarge	14	120
r5ad.large	2	10
r5ad.xlarge	3	20
r5ad.2xlarge	3	40
r5ad.4xlarge	7	60
r5ad.8xlarge	7	60
r5ad.12xlarge	7	60

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
r5ad.16xlarge	14	120
r5ad.24xlarge	14	120
r5b.16xlarge	14	120
r5d.large	2	10
r5d.xlarge	3	20
r5d.2xlarge	3	40
r5d.4xlarge	7	60
r5d.8xlarge	7	60
r5d.12xlarge	7	60
r5d.16xlarge	14	120
r5d.24xlarge	14	120
r5dn.16xlarge	14	120
r6a.large	2	10
r6a.xlarge	3	20
r6a.2xlarge	3	40
r6a.4xlarge	7	60
r6a.8xlarge	7	90
r6a.12xlarge	7	120
r6a.16xlarge	14	120
r6a.24xlarge	14	120

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
r6a.32xlarge	14	120
r6a.48xlarge	14	120
r6a.metal	14	120
r6g.medium	1	4
r6g.large	2	10
r6g.xlarge	3	20
r6g.2xlarge	3	40
r6g.4xlarge	7	60
r6g.8xlarge	7	60
r6g.12xlarge	7	60
r6g.16xlarge	14	120
r6g.metal	14	120
r6gd.medium	1	4
r6gd.large	2	10
r6gd.xlarge	3	20
r6gd.2xlarge	3	40
r6gd.4xlarge	7	60
r6gd.8xlarge	7	60
r6gd.12xlarge	7	60
r6gd.16xlarge	14	120

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
r6gd.metal	14	120
r6i.large	2	10
r6i.xlarge	3	20
r6i.2xlarge	3	40
r6i.4xlarge	7	60
r6i.8xlarge	7	90
r6i.12xlarge	7	120
r6i.16xlarge	14	120
r6i.24xlarge	14	120
r6i.32xlarge	14	120
r6i.metal	14	120
r6idn.large	2	10
r6idn.xlarge	3	20
r6idn.2xlarge	3	40
r6idn.4xlarge	7	60
r6idn.8xlarge	7	90
r6idn.12xlarge	7	120
r6idn.16xlarge	14	120
r6idn.24xlarge	14	120
r6idn.32xlarge	13	120

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
r6idn.metal	13	120
r6in.large	2	10
r6in.xlarge	3	20
r6in.2xlarge	3	40
r6in.4xlarge	7	60
r6in.8xlarge	7	90
r6in.12xlarge	7	120
r6in.16xlarge	14	120
r6in.24xlarge	14	120
r6in.32xlarge	13	120
r6in.metal	13	120
r6id.large	2	10
r6id.xlarge	3	20
r6id.2xlarge	3	40
r6id.4xlarge	7	60
r6id.8xlarge	7	90
r6id.12xlarge	7	120
r6id.16xlarge	14	120
r6id.24xlarge	14	120
r6id.32xlarge	14	120

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
r6id.metal	14	120
r7a.medium	1	4
r7a.large	2	10
r7a.xlarge	3	20
r7a.2xlarge	3	40
r7a.4xlarge	7	60
r7a.8xlarge	7	90
r7a.12xlarge	7	120
r7a.16xlarge	14	120
r7a.24xlarge	14	120
r7a.32xlarge	14	120
r7a.48xlarge	14	120
r7a.metal-48xl	14	120
r7g.medium	1	4
r7g.large	2	10
r7g.xlarge	3	20
r7g.2xlarge	3	40
r7g.4xlarge	7	60
r7g.8xlarge	7	60
r7g.12xlarge	7	60

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
r7g.16xlarge	14	120
r7g.metal	14	120
r7gd.medium	1	4
r7gd.large	2	10
r7gd.xlarge	3	20
r7gd.2xlarge	3	40
r7gd.4xlarge	7	60
r7gd.8xlarge	7	60
r7gd.12xlarge	7	60
r7gd.16xlarge	14	120
r7gd.logam	14	120
r7i.large	2	10
r7i.xlarge	3	20
r7i.2xlarge	3	40
r7i.4xlarge	7	60
r7i.8xlarge	7	90
r7i.12xlarge	7	120
r7i.16xlarge	14	120
r7i.24xlarge	14	120
r7i.48xlarge	14	120

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
r7i.metal-24xl	14	120
r7i.metal-48xl	14	120
r7iz.large	2	10
r7iz.xlarge	3	20
r7iz.2xlarge	3	40
r7iz.4xlarge	7	60
r7iz.8xlarge	7	90
r7iz.12xlarge	7	120
r7iz.16xlarge	14	120
r7iz.32xlarge	14	120
r7iz.metal-16xl	14	120
r7iz.metal-32xl	14	120
u-3tb1.56xlarge	7	12
u-6tb1.56xlarge	14	12
u-18tb1.112xlarge	14	12
u-18tb1.metal	14	12
u-24tb1.112xlarge	14	12
u-24tb1.metal	14	12
x2gd.medium	1	10
x2gd.large	2	10

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
x2gd.xlarge	3	20
x2gd.2xlarge	3	40
x2gd.4xlarge	7	60
x2gd.8xlarge	7	60
x2gd.12xlarge	7	60
x2gd.16xlarge	14	120
x2gd.metal	14	120
x2idn.16xlarge	14	120
x2idn.24xlarge	14	120
x2idn.32xlarge	14	120
x2idn.metal	14	120
x2iedn.xlarge	3	13
x2iedn.2xlarge	3	29
x2iedn.4xlarge	7	60
x2iedn.8xlarge	7	120
x2iedn.16xlarge	14	120
x2iedn.24xlarge	14	120
x2iedn.32xlarge	14	120
x2iedn.metal	14	120
x2iezn.2xlarge	3	64

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
x2iezn.4xlarge	7	120
x2iezn.6xlarge	7	120
x2iezn.8xlarge	7	120
x2iezn.12xlarge	14	120
x2iezn.metal	14	120

Penyimpanan yang dioptimalkan

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
i4g.large	2	10
i4g.xlarge	3	20
i4g.2xlarge	3	40
i4g.4xlarge	7	60
i4g.8xlarge	7	60
i4g.16xlarge	14	120
i4i.xlarge	3	8
i4i.2xlarge	3	28
i4i.4xlarge	7	58
i4i.8xlarge	7	118
i4i.12xlarge	7	118

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
i4i.16xlarge	14	248
i4i.24xlarge	14	118
i4i.32xlarge	14	498
i4i.metal	14	498
im4gn.large	2	10
im4gn.xlarge	3	20
im4gn.2xlarge	3	40
im4gn.4xlarge	7	60
im4gn.8xlarge	7	60
im4gn.16xlarge	14	120
is4gen.medium	1	4
is4gen.large	2	10
is4gen.xlarge	3	20
is4gen.2xlarge	3	40
is4gen.4xlarge	7	60
is4gen.8xlarge	7	60

Komputasi yang dipercepat

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
dl1.24xlarge	59	120
dl2q.24xlarge	14	120
g4ad.xlarge	1	12
g4ad.2xlarge	1	12
g4ad.4xlarge	2	12
g4ad.8xlarge	3	12
g4ad.16xlarge	7	12
g5.xlarge	3	6
g5.2xlarge	3	19
g5.4xlarge	7	40
g5.8xlarge	7	90
g5.12xlarge	14	120
g5.16xlarge	7	120
g5.24xlarge	14	120
g5.48xlarge	6	120
g5g.xlarge	3	20
g5g.2xlarge	3	40
g5g.4xlarge	7	60
g5g.8xlarge	7	60

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
g5g.16xlarge	14	120
g5g.metal	14	120
inf2.xlarge	3	20
inf2.8xlarge	7	90
inf2.24xlarge	14	120
inf2.48xlarge	14	120
p4d.24xlarge	59	120
p4de.24xlarge	59	120
p5.48xlarge	63	242
trn1.2xlarge	3	19
trn1.32xlarge	39	120
trn1n.32xlarge	79	242
vt1.3xlarge	3	40
vt1.6xlarge	7	60
vt1.24xlarge	14	120

Manajemen memori instance kontainer

Saat agen penampung Amazon ECS mendaftarkan instance kontainer ke dalam klaster, agen harus menentukan berapa banyak memori yang tersedia untuk dicadangkan oleh instans penampung untuk tugas Anda. Karena overhead memori platform dan memori yang ditempati oleh kernel sistem, jumlah ini berbeda dengan jumlah memori terinstal yang diiklankan untuk instans Amazon EC2. Contohnya,

instans `m4.large` memiliki memori terinstal sebesar 8 GiB. Namun, ini tidak selalu diterjemahkan ke tepat 8192 MiB memori yang tersedia untuk tugas saat instance container mendaftar.

Agen penampung Amazon ECS menyediakan variabel konfigurasi yang disebut`ECS_RESERVED_MEMORY`, yang dapat Anda gunakan untuk menghapus sejumlah memori MiB tertentu dari kumpulan yang dialokasikan untuk tugas Anda. Ini secara efektif menyimpan memori untuk proses sistem yang penting.

Jika Anda menempati semua memori pada instance kontainer dengan tugas Anda, maka ada kemungkinan bahwa tugas Anda akan bersaing dengan proses sistem penting untuk memori dan mungkin memulai kegagalan sistem.

Misalnya, jika Anda menentukan `ECS_RESERVED_MEMORY=256` dalam file konfigurasi agen kontainer Anda, agen akan mendaftarkan memori total dikurangi 256 MiB untuk instans tersebut, dan 256 MiB memori tidak dapat dialokasikan untuk tugas ECS. Untuk informasi lebih lanjut tentang variabel konfigurasi agen dan cara mengaturnya, lihat Konfigurasi agen kontainer Amazon ECS dan Bootstrapping instance container dengan data pengguna Amazon EC2.

Jika Anda menentukan 8192 MiB untuk tugas tersebut, dan tidak ada instance conatiner Anda yang memiliki 8192 MiB atau lebih besar memori yang tersedia untuk memenuhi persyaratan ini, maka tugas tersebut tidak dapat ditempatkan di cluster Anda. Jika Anda menggunakan lingkungan komputasi terkelola, maka AWS Batch harus meluncurkan jenis instans yang lebih besar untuk mengakomodasi permintaan.

Anda juga harus menyimpan beberapa memori untuk agen penampung Amazon ECS dan proses sistem penting lainnya pada instance penampung Anda, sehingga kontainer tugas Anda tidak bersaing untuk memori yang sama dan mungkin memulai kegagalan sistem.

Agen kontainer Amazon ECS menggunakan fungsi `ReadMemInfo()` Docker untuk kueri total memori yang tersedia untuk sistem operasi. Baik Linux dan Windows menyediakan utilitas baris perintah untuk menentukan total memori.

Example - Menentukan memori total Linux

Perintah free menampilkan memori total yang diakui oleh sistem operasi.

```
$ free -b
```

Contoh output untuk instans `m4.large` yang menjalankan AMI Amazon Linux yang dioptimalkan untuk Amazon ECS.

```
             total       used       free     shared    buffers     cached
Mem:    8373026816  348180480 8024846336      90112   25534464  205418496
-/+ buffers/cache:  117227520 8255799296
```

Instans ini memiliki 8373026816 byte memori total, yang diterjemahkan menjadi 7985 MiB yang tersedia untuk tugas.

Example - Menentukan memori total Windows

Perintah wmic menampilkan memori total yang diakui oleh sistem operasi.

```
C:\> wmic ComputerSystem get TotalPhysicalMemory
```

Contoh keluaran untuk `m4.large` instance yang menjalankan AMI Windows Server Amazon ECS yang dioptimalkan.

```
TotalPhysicalMemory
8589524992
```

Instans ini memiliki 8589524992 byte memori total, yang diterjemahkan menjadi 8191 MiB yang tersedia untuk tugas.

Melihat memori instance kontainer

Anda dapat melihat berapa banyak memori yang didaftarkan oleh instans kontainer di konsol Amazon ECS (atau dengan operasi DescribeContainerInstancesAPI). Jika Anda mencoba memaksimalkan pemanfaatan sumber daya Anda dengan menyediakan tugas Anda sebanyak mungkin memori untuk jenis instance tertentu, Anda dapat mengamati memori yang tersedia untuk instance kontainer itu dan kemudian menetapkan tugas Anda sebanyak itu memori.

Untuk melihat memori instance kontainer

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Clusters, lalu pilih cluster yang menghosting instance container Anda.
3. Pilih Infrastruktur, dan kemudian di bawah Instance Container, pilih instance container.
4. Bagian Resources menunjukkan memori terdaftar dan tersedia untuk instance container.

   Nilai memori Terdaftar adalah contoh kontainer; terdaftar dengan Amazon ECS saat pertama kali diluncurkan, dan nilai memori yang tersedia adalah apa yang belum dialokasikan untuk tugas.

Mengelola instans kontainer dari jarak jauh menggunakan AWS Systems Manager

Anda dapat menggunakan kemampuan Run Command di AWS Systems Manager (Systems Manager) untuk mengelola konfigurasi instans container Amazon ECS secara aman dan jarak jauh. Run Command menyediakan cara sederhana untuk melakukan tugas-tugas administratif umum tanpa harus masuk secara lokal pada instans. Anda dapat mengelola perubahan konfigurasi di klaster Anda dengan cara menjalankan perintah pada beberapa instans kontainer secara bersamaan. Run Command melaporkan status dan hasil dari setiap perintah.

Berikut adalah beberapa contoh tipe tugas yang dapat Anda lakukan dengan Run Command:

- Menginstal atau menghapus paket.
- Melakukan pembaruan keamanan.
- Membersihkan citra Docker.
- Menghentikan atau memulai layanan.
- Melihat sumber daya sistem.
- Melihat berkas log.
- Melakukan operasi file.

Untuk informasi selengkapnya tentang Jalankan Perintah, lihat AWS Systems Manager Menjalankan Perintah di Panduan AWS Systems Manager Pengguna.

Topik

- Jalankan kebijakan IAM Command
- Menggunakan Run Command

Jalankan kebijakan IAM Command

Sebelum Anda dapat mengirim perintah ke instance container Anda dengan Run Command, Anda harus melampirkan kebijakan IAM yang memungkinkan `ecsInstanceRole` untuk memiliki akses ke Systems Manager API. Prosedur berikut menjelaskan cara melampirkan kebijakan terkelola Systems Manager ke peran instance container Anda sehingga instance yang diluncurkan dengan peran ini dapat menggunakan Run Command.

Untuk melampirkan kebijakan Systems Manager ke **`ecsInstanceRole`**

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.

2. Di panel navigasi, pilih Peran.
3. Pilih `ecsInstanceRole`. Jika peran belum ada, ikuti prosedur di Peran IAM instans wadah Amazon ECS untuk membuat peran.
4. Pilih tab Izin.
5. Pilih Lampirkan kebijakan.
6. Untuk mempersempit kebijakan yang tersedia untuk dilampirkan, untuk Filter, ketik `SSM`.
7. Dalam daftar kebijakan, pilih kotak berikutnya AmazonSSM ManagedInstanceCore. Gunakan kebijakan ini untuk memberikan izin minimum yang diperlukan untuk menggunakan Systems Manager.

   Untuk informasi tentang kebijakan lain yang dapat Anda berikan untuk operasi Systems Manager, lihat Membuat Profil Instans IAM untuk Systems Manager di Panduan AWS Systems Manager Pengguna.
8. Pilih Lampirkan Kebijakan.

Menggunakan Run Command

Setelah melampirkan kebijakan terkelola Systems Manager ke `ecsInstanceRole` dan memverifikasi bahwa AWS Systems Manager Agen (Agen SSM) diinstal pada instance container, Anda dapat mulai menggunakan Run Command untuk mengirim perintah ke instance container Anda. Untuk informasi tentang menjalankan perintah dan skrip shell pada instance Anda dan melihat output yang dihasilkan, lihat Menjalankan Perintah Menggunakan Systems Manager Run Command and Run Command Walkthroughs di Panduan Pengguna.AWS Systems Manager

Contoh: Memperbarui perangkat lunak instans kontainer dengan Run Command

Kasus penggunaan umum untuk Run Command adalah memperbarui perangkat lunak instans pada seluruh armada instans kontainer Anda pada satu waktu.

1. Lampirkan kebijakan terkelola Systems Manager ke Anda`ecsInstanceRole`.
2. Pastikan bahwa Agen SSM diinstal pada instans kontainer Anda. Untuk informasi selengkapnya, lihat Menginstal Agen SSM secara manual pada instans EC2 untuk Linux.
3. Buka konsol Systems Manager di https://console.aws.amazon.com/systems-manager.
4. Pada panel navigasi kiri, pilih Run Command, lalu pilih Run Command.
5. Untuk Dokumen perintah, pilih `AWS-RunShellScript`.

6. Di bagian Perintah, masukkan perintah atau beberapa perintah yang akan dikirimkan ke instans kontainer Anda. Dalam contoh ini, perintah berikut memperbarui perangkat lunak instans:

   ```
   $ yum update -y
   ```

7. Di bagian Instans Target, pilih kotak di sebelah instans kontainer untuk menjalankan perintah perbarui.
8. Pilih Jalankan untuk mengirim perintah ke instans tertentu.
9. (Opsional) Pilih ikon segarkan untuk memantau status perintah.
10. (Opsional) Dalam Target dan output, pilih tombol di sebelah ID instans, kemudian pilih Tampilkan output.

Konfigurasi proxy HTTP untuk instance kontainer Linux

Anda dapat mengonfigurasi instans penampung Amazon ECS untuk menggunakan proxy HTTP untuk agen penampung Amazon ECS dan daemon Docker. Ini berguna jika instance container Anda tidak memiliki akses jaringan eksternal melalui gateway internet VPC Amazon, gateway NAT, atau instance.

Untuk mengonfigurasi instans penampung Amazon ECS Linux Anda agar menggunakan proxy HTTP, setel variabel berikut dalam file yang relevan pada waktu peluncuran (dengan data pengguna Amazon EC2). Anda juga dapat mengedit file konfigurasi secara manual dan kemudian memulai ulang agen.

`/etc/ecs/ecs.config`(Amazon Linux 2dan AmazonLinux AMI)

`HTTP_PROXY=`*`10.0.0.131`*`:`*`3128`*

Tetapkan nilai ini ke nama host (atau alamat IP) dan nomor port proxy HTTP yang akan digunakan agen Amazon ECS untuk terhubung ke internet. Misalnya, instance container Anda mungkin tidak memiliki akses jaringan eksternal melalui gateway internet VPC Amazon, gateway NAT, atau instans.

`NO_PROXY=169.254.169.254,169.254.170.2,/var/run/docker.sock`

Tetapkan nilai ini `169.254.169.254,169.254.170.2,/var/run/docker.sock` untuk memfilter metadata instans EC2, peran IAM untuk tugas, dan lalu lintas daemon Docker dari proxy.

`/etc/systemd/system/ecs.service.d/http-proxy.conf`(Amazon Linux 2 saja)

`Environment="HTTP_PROXY=10.0.0.131:3128/"`

Atur nilai ini pada nama host (atau alamat IP) dan nomor port proxy HTTP yang akan digunakan `ecs-init` untuk terhubung ke internet. Misalnya, instance container Anda mungkin tidak memiliki akses jaringan eksternal melalui gateway internet VPC Amazon, gateway NAT, atau instans.

`Environment="NO_PROXY=169.254.169.254,169.254.170.2,/var/run/docker.sock"`

Tetapkan nilai ini `169.254.169.254,169.254.170.2,/var/run/docker.sock` untuk memfilter metadata instans EC2, peran IAM untuk tugas, dan lalu lintas daemon Docker dari proxy.

`/etc/init/ecs.override`(Hanya AMI Amazon Linux)

`env HTTP_PROXY=10.0.0.131:3128`

Atur nilai ini pada nama host (atau alamat IP) dan nomor port proxy HTTP yang akan digunakan `ecs-init` untuk terhubung ke internet. Misalnya, instance container Anda mungkin tidak memiliki akses jaringan eksternal melalui gateway internet VPC Amazon, gateway NAT, atau instans.

`env NO_PROXY=169.254.169.254,169.254.170.2,/var/run/docker.sock`

Tetapkan nilai ini `169.254.169.254,169.254.170.2,/var/run/docker.sock` untuk memfilter metadata instans EC2, peran IAM untuk tugas, dan lalu lintas daemon Docker dari proxy.

`/etc/systemd/system/docker.service.d/http-proxy.conf` (Amazon Linux 2 saja)

`Environment="HTTP_PROXY=http://10.0.0.131:3128"`

Tetapkan nilai ini ke nama host (atau alamat IP) dan nomor port proxy HTTP yang akan digunakan untuk daemon Docker untuk terhubung ke internet. Misalnya, instance container Anda mungkin tidak memiliki akses jaringan eksternal melalui gateway internet VPC Amazon, gateway NAT, atau instans.

`Environment="NO_PROXY=169.254.169.254"`

Tetapkan nilai ini ke `169.254.169.254` untuk memfilter metadata instans EC2 dari proxy.

`/etc/sysconfig/docker`(Amazon Linux AMI dan Amazon Linux 2 hanya)

`export HTTP_PROXY=http://10.0.0.131:3128`

Tetapkan nilai ini ke nama host (atau alamat IP) dan nomor port proxy HTTP yang akan digunakan untuk daemon Docker untuk terhubung ke internet. Misalnya, instance container Anda mungkin tidak memiliki akses jaringan eksternal melalui gateway internet VPC Amazon, gateway NAT, atau instans.

`export NO_PROXY=169.254.169.254,169.254.170.2`

Tetapkan nilai ini ke `169.254.169.254` untuk memfilter metadata instans EC2 dari proxy.

Menyetel variabel lingkungan ini dalam file di atas hanya memengaruhi agen penampung Amazon ECS`ecs-init`, dan daemon Docker. Mereka tidak mengonfigurasi layanan lain (seperti yum) untuk menggunakan proxy.

Example Skrip data pengguna proxy HTTP Amazon Linux

Contoh `cloud-boothook` skrip data pengguna di bawah ini mengonfigurasi agen penampung Amazon ECS`ecs-init`, daemon Docker, dan menggunakan proxy HTTP yang yum Anda tentukan. Anda juga dapat menentukan klaster tempat instans kontainer akan terdaftar.

Untuk menggunakan skrip ini ketika Anda meluncurkan instans kontainer, ikuti langkah-langkah di Meluncurkan instans penampung Amazon ECS Linux, dan di Step 6.g. Kemudian, salin dan tempel skrip `cloud-boothook` di bawah ini ke bidang Data pengguna (pastikan untuk mengganti nilai contoh merah dengan informasi proxy dan klaster Anda sendiri).

> **Note**
>
> Skrip data pengguna di bawah ini hanya mendukung Amazon Linux 2 dan varian Amazon Linux AMI dari AMI Amazon ECS yang dioptimalkan.

```
#cloud-boothook
# Configure Yum, the Docker daemon, and the ECS agent to use an HTTP proxy

# Specify proxy host, port number, and ECS cluster name to use
PROXY_HOST=10.0.0.131
PROXY_PORT=3128
CLUSTER_NAME=proxy-test
```

```
if grep -q 'Amazon Linux release 2' /etc/system-release ; then
    OS=AL2
    echo "Setting OS to Amazon Linux 2"
elif grep -q 'Amazon Linux AMI' /etc/system-release ; then
    OS=ALAMI
    echo "Setting OS to Amazon Linux AMI"
else
    echo "This user data script only supports Amazon Linux 2 and Amazon Linux AMI."
fi

# Set Yum HTTP proxy
if [ ! -f /var/lib/cloud/instance/sem/config_yum_http_proxy ]; then
    echo "proxy=http://$PROXY_HOST:$PROXY_PORT" >> /etc/yum.conf
    echo "$$: $(date +%s.%N | cut -b1-13)" > /var/lib/cloud/instance/sem/
config_yum_http_proxy
fi

# Set Docker HTTP proxy (different methods for Amazon Linux 2 and Amazon Linux AMI)
# Amazon Linux 2
if [ $OS == "AL2" ] && [ ! -f /var/lib/cloud/instance/sem/config_docker_http_proxy ];
 then
    mkdir /etc/systemd/system/docker.service.d
    cat <<EOF > /etc/systemd/system/docker.service.d/http-proxy.conf
[Service]
Environment="HTTP_PROXY=http://$PROXY_HOST:$PROXY_PORT/"
Environment="HTTPS_PROXY=https://$PROXY_HOST:$PROXY_PORT/"
Environment="NO_PROXY=169.254.169.254,169.254.170.2"
EOF
    systemctl daemon-reload
    if [ "$(systemctl is-active docker)" == "active" ]
    then
        systemctl restart docker
    fi
    echo "$$: $(date +%s.%N | cut -b1-13)" > /var/lib/cloud/instance/sem/
config_docker_http_proxy
fi
# Amazon Linux AMI
if [ $OS == "ALAMI" ] && [ ! -f /var/lib/cloud/instance/sem/config_docker_http_proxy ];
 then
    echo "export HTTP_PROXY=http://$PROXY_HOST:$PROXY_PORT/" >> /etc/sysconfig/docker
    echo "export HTTPS_PROXY=https://$PROXY_HOST:$PROXY_PORT/" >> /etc/sysconfig/docker
    echo "export NO_PROXY=169.254.169.254,169.254.170.2" >> /etc/sysconfig/docker
```

```
    echo "$$: $(date +%s.%N | cut -b1-13)" > /var/lib/cloud/instance/sem/
config_docker_http_proxy
fi

# Set ECS agent HTTP proxy
if [ ! -f /var/lib/cloud/instance/sem/config_ecs-agent_http_proxy ]; then
    cat <<EOF > /etc/ecs/ecs.config
ECS_CLUSTER=$CLUSTER_NAME
HTTP_PROXY=$PROXY_HOST:$PROXY_PORT
NO_PROXY=169.254.169.254,169.254.170.2,/var/run/docker.sock
EOF
    echo "$$: $(date +%s.%N | cut -b1-13)" > /var/lib/cloud/instance/sem/config_ecs-
agent_http_proxy
fi

# Set ecs-init HTTP proxy (different methods for Amazon Linux 2 and Amazon Linux AMI)
# Amazon Linux 2
if [ $OS == "AL2" ] && [ ! -f /var/lib/cloud/instance/sem/config_ecs-init_http_proxy ];
 then
    mkdir /etc/systemd/system/ecs.service.d
    cat <<EOF > /etc/systemd/system/ecs.service.d/http-proxy.conf
[Service]
Environment="HTTP_PROXY=$PROXY_HOST:$PROXY_PORT/"
Environment="NO_PROXY=169.254.169.254,169.254.170.2,/var/run/docker.sock"
EOF
    systemctl daemon-reload
    if [ "$(systemctl is-active ecs)" == "active" ]; then
        systemctl restart ecs
    fi
    echo "$$: $(date +%s.%N | cut -b1-13)" > /var/lib/cloud/instance/sem/config_ecs-
init_http_proxy
fi
# Amazon Linux AMI
if [ $OS == "ALAMI" ] && [ ! -f /var/lib/cloud/instance/sem/config_ecs-
init_http_proxy ]; then
    cat <<EOF > /etc/init/ecs.override
env HTTP_PROXY=$PROXY_HOST:$PROXY_PORT
env NO_PROXY=169.254.169.254,169.254.170.2,/var/run/docker.sock
EOF
    echo "$$: $(date +%s.%N | cut -b1-13)" > /var/lib/cloud/instance/sem/config_ecs-
init_http_proxy
fi
```

Memperbarui agen kontainer Amazon ECS

Terkadang, Anda mungkin perlu memperbarui agen penampung Amazon ECS untuk mengambil perbaikan bug dan fitur baru. Memperbarui agen penampung Amazon ECS tidak mengganggu tugas atau layanan yang sedang berjalan pada instance container. Proses untuk memperbarui agen berbeda tergantung pada apakah instans penampung Anda diluncurkan dengan AMI yang dioptimalkan Amazon ECS atau sistem operasi lain.

> **Note**
>
> Pembaruan agen tidak berlaku pada instans kontainer Windows. Kami menyarankan Anda agar meluncurkan instans kontainer baru untuk memperbarui versi agen di klaster Windows Anda.

Topik

Memeriksa versi agen penampung Amazon ECS

Anda dapat memeriksa versi agen kontainer yang berjalan pada instans kontainer Anda untuk melihat apakah pembaruan diperlukan. Tampilan instance container di konsol Amazon ECS menyediakan versi agen. Gunakan prosedur berikut untuk memeriksa versi agen Anda.

Amazon ECS console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah tempat instans eksternal Anda didaftarkan.
3. Di panel navigasi, pilih Klaster, kemudian pilih klaster yang meng-host instans eksternal.
4. Pada halaman Cluster: ***name***, pilih tab Infrastructure.
5. Di bawah Instance Container, perhatikan kolom versi Agen untuk instance kontainer Anda. Jika instans kontainer tidak memiliki versi instans kontainer terbaru, konsol akan memberi peringatan melalui pesan dan bendera bahwa versi agen kedaluwarsa.

Jika versi agen Anda kedaluwarsa, Anda dapat memperbarui agen kontainer Anda dengan prosedur berikut:

- Jika instance container Anda menjalankan AMI Amazon ECS yang dioptimalkan, lihat. Memperbarui agen kontainer Amazon ECS pada AMI Amazon ECS yang dioptimalkan
- Jika instance container Anda tidak menjalankan AMI yang dioptimalkan Amazon ECS, lihat. Memperbarui agen kontainer Amazon ECS secara manual (untuk AMI yang dioptimalkan ECS non-Amazon)

> **Important**
>
> Untuk memperbarui versi agen Amazon ECS dari versi sebelum v1.0.0 pada AMI yang dioptimalkan Amazon ECS, sebaiknya hentikan instans penampung saat ini dan meluncurkan instans baru dengan versi AMI terbaru. Setiap instans kontainer yang menggunakan versi pratinjau harus pensiun dan diganti dengan AMI terbaru. Untuk informasi selengkapnya, lihat Meluncurkan instans penampung Amazon ECS Linux.

Amazon ECS container agent introspection API

Anda juga dapat menggunakan untuk memeriksa versi API introspeksi agen penampung Amazon ECS dari instance container itu sendiri. Untuk informasi selengkapnya, lihat Introspeksi wadah Amazon ECS.

Untuk memeriksa apakah agen penampung Amazon ECS Anda menjalankan versi terbaru dengan API introspeksi

1. Masuk ke instans kontainer Anda melalui SSH.
2. Kueri API introspeksi.

```
[ec2-user ~]$ curl -s 127.0.0.1:51678/v1/metadata | python -mjson.tool
```

> **Note**
>
> API introspeksi menambahkan `Version` informasi dalam versi v1.0.0 dari agen penampung Amazon ECS. Jika `Version` tidak ada ketika sedang melakukan kueri API introspeksi, atau API introspeksi sama sekali tidak ada dalam agen Anda, maka

versi yang Anda jalankan adalah v0.0.3 atau sebelumnya. Anda harus memperbarui versi Anda.

Memperbarui agen kontainer Amazon ECS pada AMI Amazon ECS yang dioptimalkan

Jika Anda menggunakan AMI Amazon ECS yang dioptimalkan, Anda memiliki beberapa opsi untuk mendapatkan versi terbaru dari agen penampung Amazon ECS (ditampilkan dalam urutan rekomendasi):

- Hentikan instans penampung dan luncurkan versi terbaru Amazon ECS Amazon Linux 2 AMI yang dioptimalkan Amazon (baik secara manual atau dengan memperbarui konfigurasi peluncuran Auto Scaling Anda dengan AMI terbaru). Ini menyediakan instance container baru dengan versi terbaru yang diuji dan divalidasi dariAmazon Linux, Docker`ecs-init`, dan agen penampung Amazon ECS. Untuk informasi selengkapnya, lihat AMI Amazon ECS yang dioptimalkan.
- Hubungkan ke instans dengan SSH dan perbarui paket `ecs-init` (dan dependensinya) ke versi terbaru. Operasi ini menyediakan versi Docker yang diuji dan divalidasi terbaru dan `ecs-init` yang tersedia di Amazon Linux repositori dan versi terbaru dari agen penampung Amazon ECS. Untuk informasi selengkapnya, lihat Untuk memperbarui `ecs-init` paket pada AMI Amazon ECS yang dioptimalkan.
- Perbarui agen penampung dengan operasi `UpdateContainerAgent` API, baik melalui konsol atau dengan AWS SDK AWS CLI atau. Untuk informasi selengkapnya, lihat Memperbarui agen penampung Amazon ECS dengan operasi `UpdateContainerAgent` API.

> **Note**
>
> Pembaruan agen tidak berlaku pada instans kontainer Windows. Kami menyarankan Anda agar meluncurkan instans kontainer baru untuk memperbarui versi agen di klaster Windows Anda.

Untuk memperbarui **`ecs-init`** paket pada AMI Amazon ECS yang dioptimalkan

1. Masuk ke instans kontainer Anda melalui SSH.
2. Perbarui paket `ecs-init` dengan perintah berikut.

```
sudo yum update -y ecs-init
```

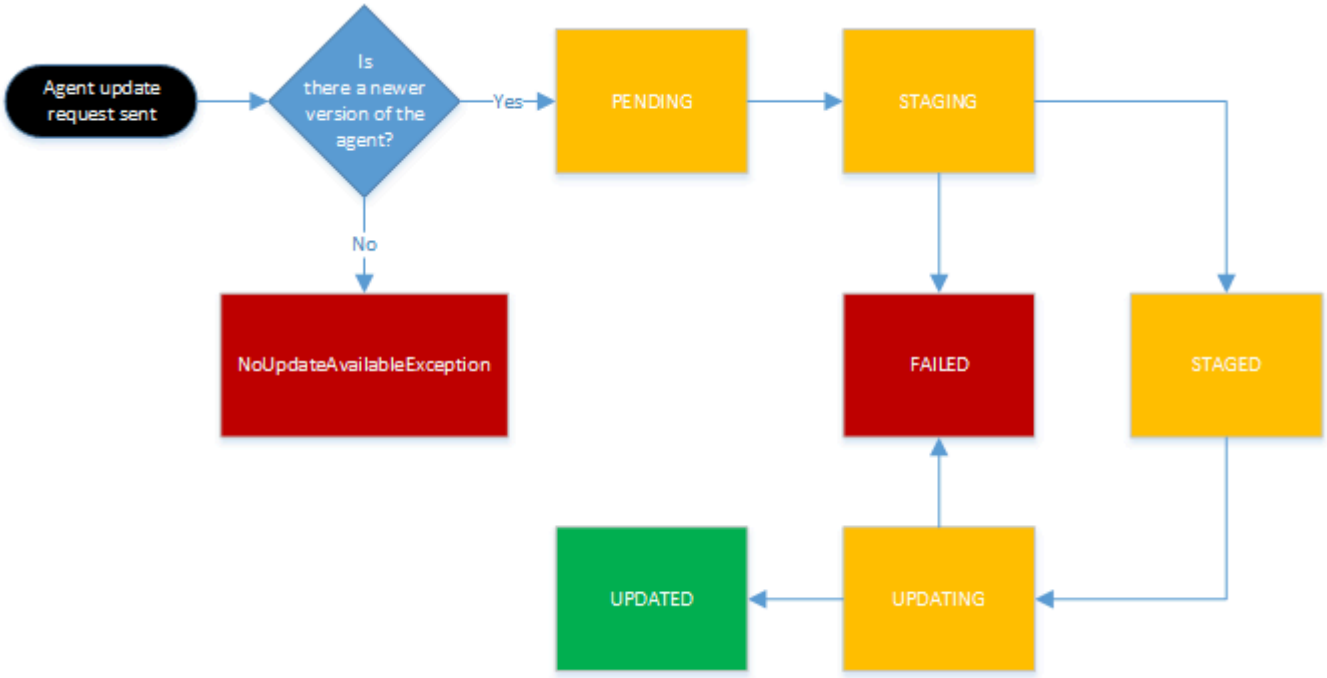
> **Note**
>
> `ecs-init`Paket dan agen kontainer Amazon ECS segera diperbarui. Namun, versi Docker yang lebih baru tidak dimuat sampai daemon Docker di mulai ulang. Pemulaian ulang dilakukan baik dengan booting ulang instans, atau dengan menjalankan perintah berikut pada instans Anda:
>
> - AMI Amazon Linux 2 yang dioptimalkan untuk Amazon ECS:
>
>   ```
>   sudo systemctl restart docker
>   ```
>
> - AMI Amazon Linux yang dioptimalkan untuk Amazon ECS:
>
>   ```
>   sudo service docker restart && sudo start ecs
>   ```

Memperbarui agen penampung Amazon ECS dengan operasi **`UpdateContainerAgent`** API

> **Important**
>
> `UpdateContainerAgent`API ini hanya didukung pada varian Linux dari AMI Amazon ECS yang dioptimalkan, dengan pengecualian AMI Amazon Linux 2 (arm64) Amazon ECS yang dioptimalkan Amazon. Untuk instance kontainer yang menggunakan AMI Amazon Linux 2 (arm64) Amazon ECS yang dioptimalkan Amazon, `ecs-init` perbarui paket untuk memperbarui agen. Untuk instans kontainer yang sedang menjalankan sistem operasi lain, lihat Memperbarui agen kontainer Amazon ECS secara manual (untuk AMI yang dioptimalkan ECS non-Amazon). Jika Anda menggunakan instans kontainer Windows, Kami menyarankan Anda agar meluncurkan instans kontainer baru untuk memperbarui versi agen di klaster Windows Anda.

Proses `UpdateContainerAgent` API dimulai saat Anda meminta pembaruan agen, baik melalui konsol atau dengan AWS SDK AWS CLI atau. Amazon ECS memeriksa versi agen Anda saat ini terhadap versi agen terbaru yang tersedia, dan jika pembaruan memungkinkan, proses pembaruan berlangsung seperti yang ditunjukkan pada diagram alir di bawah ini. Jika pembaruan tidak tersedia,

misalnya, jika agen sudah menjalankan versi terbaru, maka `NoUpdateAvailableException` dikembalikan.



Tahapan dalam proses pembaruan yang ditunjukkan di atas adalah sebagai berikut:

`PENDING`

Pembaruan agen tersedia, dan proses pembaruan telah dimulai.

`STAGING`

Agen telah mulai mengunduh pembaruan agen. Jika agen tidak dapat mengunduh pembaruan, atau jika isi pembaruan salah atau rusak, maka agen akan mengirimkan notifikasi kegagalan dan transisi pembaruan ke status `FAILED`.

`STAGED`

Pengunduhan agen selesai dan isi agen telah diverifikasi.

`UPDATING`

Layanan `ecs-init` dimulai ulang dan versi agen baru diambil. Jika agen karena alasan tertentu tidak dapat memulai ulang, pembaruan beralih ke `FAILED` status; jika tidak, agen memberi sinyal kepada Amazon ECS bahwa pembaruan telah selesai.

> **Note**
>
> Pembaruan agen tidak berlaku pada instans kontainer Windows. Kami menyarankan Anda agar meluncurkan instans kontainer baru untuk memperbarui versi agen di klaster Windows Anda.

Untuk memperbarui agen penampung Amazon ECS pada AMI Amazon ECS yang dioptimalkan di konsol

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah tempat instans eksternal Anda didaftarkan.
3. Di panel navigasi, pilih Klaster dan pilih klaster.
4. Pada halaman Cluster: ***name***, pilih tab Infrastructure.
5. Di bawah Instance Container, pilih instance yang akan diperbarui, lalu pilih Actions, Update agent.

Memperbarui agen kontainer Amazon ECS secara manual (untuk AMI yang dioptimalkan ECS non-Amazon)

Untuk memperbarui agen penampung Amazon ECS secara manual (untuk AMI yang dioptimalkan untuk ECS non-Amazon)

> **Note**
>
> Pembaruan agen tidak berlaku pada instans kontainer Windows. Kami menyarankan Anda agar meluncurkan instans kontainer baru untuk memperbarui versi agen di klaster Windows Anda.

1. Masuk ke instans kontainer Anda melalui SSH.
2. Lakukan pemeriksaan untuk melihat apakah agen Anda menggunakan variabel lingkungan `ECS_DATADIR` untuk menyimpan statusnya.

   ```
   ubuntu:~$ docker inspect ecs-agent | grep ECS_DATADIR
   ```

   Output:

```
"ECS_DATADIR=/data",
```

> **Important**
>
> Jika perintah sebelumnya tidak mengembalikan variabel lingkungan `ECS_DATADIR`, Anda harus menghentikan tugas apa pun yang berjalan pada instans kontainer ini sebelum memperbarui agen Anda. Agen yang lebih baru dengan variabel lingkungan `ECS_DATADIR` menyimpan statusnya dan Anda dapat memperbaruinya saat tugas sedang berjalan tanpa masalah.

3. Hentikan agen kontainer Amazon ECS.

```
ubuntu:~$ docker stop ecs-agent
```

4. Hapus kontainer agen.

```
ubuntu:~$ docker rm ecs-agent
```

5. Pastikan `/etc/ecs` direktori dan file konfigurasi agen penampung Amazon ECS ada di`/etc/ecs/ecs.config`.

```
ubuntu:~$ sudo mkdir -p /etc/ecs && sudo touch /etc/ecs/ecs.config
```

6. Edit file `/etc/ecs/ecs.config` dan pastikan bahwa file berisi setidaknya deklarasi variabel berikut. Jika Anda tidak ingin instans kontainer Anda untuk terdaftar dengan klaster default, tentukan nama klaster Anda sebagai nilai untuk `ECS_CLUSTER`.

```
ECS_DATADIR=/data
ECS_ENABLE_TASK_IAM_ROLE=true
ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST=true
ECS_LOGFILE=/log/ecs-agent.log
ECS_AVAILABLE_LOGGING_DRIVERS=["json-file","awslogs"]
ECS_LOGLEVEL=info
ECS_CLUSTER=default
```

Untuk informasi selengkapnya tentang opsi waktu aktif agen ini, lihat Konfigurasi agen kontainer Amazon ECS.

> **Note**
>
> Anda dapat menyimpan variabel lingkungan agen secara opsional di Amazon S3 (yang dapat diunduh ke instans penampung Anda pada waktu peluncuran menggunakan data pengguna Amazon EC2). Hal ini dianjurkan untuk informasi sensitif seperti kredensial autentikasi untuk repositori privat. Untuk informasi lebih lanjut, lihat Menyimpan konfigurasi instans kontainer di Amazon S3 dan Autentikasi registri privat untuk tugas.

7. Tarik gambar agen kontainer Amazon ECS terbaru dari Amazon Elastic Container Registry Public.

```
ubuntu:~$ docker pull public.ecr.aws/ecs/amazon-ecs-agent:latest
```

Output:

```
Pulling repository amazon/amazon-ecs-agent
a5a56a5e13dc: Download complete
511136ea3c5a: Download complete
9950b5d678a1: Download complete
c48ddcf21b63: Download complete
Status: Image is up to date for amazon/amazon-ecs-agent:latest
```

8. Jalankan agen penampung Amazon ECS terbaru pada instance container Anda.

> **Note**
>
> Gunakan kebijakan mulai ulang Docker atau manajer proses (seperti upstart atau systemd) untuk memperlakukan agen kontainer sebagai suatu layanan atau daemon dan pastikan mulai ulang dilakukan setelah keluar. Untuk informasi selengkapnya, lihat Secara otomatis memulai kontainer dan Kebijakan mulai ulang dalam dokumentasi Docker. AMI Amazon ECS yang dioptimalkan menggunakan `ecs-init` RPM untuk tujuan ini, dan Anda dapat melihat kode sumber untuk RPM ini aktif. GitHub

Contoh run command agen berikut dipecah menjadi baris terpisah agar setiap opsi dapat ditunjukkan. Untuk informasi selengkapnya tentang opsi waktu aktif agen ini, lihat Konfigurasi agen kontainer Amazon ECS.

Important

Pengaktifan sistem operasi dengan SELinux memerlukan opsi `--privileged` di perintah docker run Anda. Selain itu, untuk instans kontainer yang mengaktifkan Selinux, kami sarankan agar Anda menambahkan opsi `:Z` pada pemasangan volume `/log` dan `/data`. Namun, pemasangan host untuk volume ini harus ada sebelum Anda menjalankan perintah atau Anda menerima kesalahan `no such file or directory`. Lakukan tindakan berikut jika Anda mengalami kesulitan menjalankan agen Amazon ECS pada instance container berkemampuan SELinux:

- Buat titik-titik pemasangan volume host pada instans kontainer Anda.

```
ubuntu:~$ sudo mkdir -p /var/log/ecs /var/lib/ecs/data
```

- Tambahkan opsi `--privileged` pada perintah docker run di bawah ini.
- Tambahkan opsi `:Z` pada pemasangan volume kontainer `/log` dan `/data` (misalnya, `--volume=/var/log/ecs/:/log:Z`) pada perintah docker run di bawah ini.

```
ubuntu:~$ sudo docker run --name ecs-agent \
--detach=true \
--restart=on-failure:10 \
--volume=/var/run:/var/run \
--volume=/var/log/ecs/:/log \
--volume=/var/lib/ecs/data:/data \
--volume=/etc/ecs:/etc/ecs \
--volume=/etc/ecs:/etc/ecs/pki \
--net=host \
--env-file=/etc/ecs/ecs.config \
amazon/amazon-ecs-agent:latest
```

Note

Jika Anda menerima pesan `Error response from daemon: Cannot start container`, Anda dapat menghapus kontainer yang gagal dengan perintah sudo docker rm ecs-agent dan coba jalankan agen kembali.

Manajemen instance wadah Windows

Manajemen instance kontainer Windows meliputi:

- Meluncurkan instans kontainer
- Bootstrapping instance kontainer
- Menghubungkan ke instance kontainer Anda
- Menggunakan proxy HTTP untuk agen kontainer dan daemon Docker
- Menderegistrasi instance kontainer

Pembaruan agen tidak berlaku pada instans kontainer Windows. Kami menyarankan Anda agar meluncurkan instans kontainer baru untuk memperbarui versi agen di klaster Windows Anda.

Meluncurkan instans penampung Amazon ECS Windows

Instans penampung Amazon ECS Anda dibuat menggunakan konsol Amazon EC2. Sebelum memulai, pastikan Anda telah menyelesaikan langkah-langkah tersebut di Siapkan untuk menggunakan Amazon ECS.

Untuk informasi selengkapnya tentang panduan peluncuran, lihat Meluncurkan instance menggunakan wizard instans peluncuran baru di Panduan Pengguna Amazon EC2 untuk Instans Windows.

Wisaya instans peluncuran Amazon EC2 baru

Anda dapat menggunakan wizard Amazon EC2 baru untuk meluncurkan instans. Anda dapat menggunakan daftar berikut untuk parameter dan membiarkan parameter tidak terdaftar sebagai default. Instruksi berikut membawa Anda melalui setiap kelompok parameter.

Parameter untuk konfigurasi instans

- Memulai peluncuran instans
- Nama dan tanda
- Aplikasi dan Gambar OS (Gambar Mesin Amazon)
- Jenis instans
- Pasangan kunci (login)
- Pengaturan jaringan
- Mengonfigurasi penyimpanan

- Detail lanjutan

Memulai peluncuran instans

1. Buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.
2. Di bilah navigasi di bagian atas layar, AWS Wilayah saat ini ditampilkan (misalnya, AS Timur (Ohio)). Pilih Wilayah tempat instans akan diluncurkan. Pilihan ini penting karena beberapa sumber daya Amazon EC2 dapat dibagikan di antara Wilayah, sedangkan sumber daya yang lainnya tidak.
3. Dari dasbor konsol Amazon EC2, pilih Luncurkan instans.

Nama dan tanda

Nama instans adalah tanda, di mana kuncinya adalah Name, dan nilainya adalah nama yang Anda tentukan. Anda dapat menandai instance, volume, dan grafik elastis. Untuk Instans Spot, Anda hanya dapat menandai permintaan Instans Spot.

Menentukan nama instans dan tanda tambahan bersifat opsional.

- Untuk Nama, masukkan nama deskriptif untuk instans tersebut. Jika Anda tidak menentukan nama, instans dapat diidentifikasi berdasarkan ID-nya, yang secara otomatis dihasilkan saat Anda meluncurkan instans tersebut.
- Untuk menambahkan tanda tambahan, pilih Tambahkan tanda tambahan. Pilih Tambahkan tanda, lalu masukkan kunci dan nilai, lalu pilih jenis sumber daya yang akan diberi tanda. Pilih Tambah tanda lagi untuk setiap tanda tambahan yang akan ditambahkan.

Aplikasi dan Gambar OS (Gambar Mesin Amazon)

Amazon Machine Image (AMI) berisi informasi yang diperlukan untuk membuat instans. Misalnya, AMI mungkin berisi perangkat lunak yang diperlukan untuk bertindak sebagai server web, seperti Apache, dan situs web Anda.

Untuk AMI terbaru yang dioptimalkan Amazon ECS dan nilainya, lihat AMI yang dioptimalkan untuk Windows Amazon ECS.

Gunakan bilah Pencarian untuk menemukan AMI yang dioptimalkan Amazon ECS yang dioptimalkan yang diterbitkan oleh. AWS

1. Berdasarkan kebutuhan Anda, masukkan salah satu AMI berikut di bilah Pencarian dan tekan Enter.
   - Windows_server-2022-Inggris-penuh-ECS_dioptimalkan
   - Windows_Server-2022-Inggris-Core-ECS_Optimized
   - Windows_server-2019-Inggris-penuh-ECS_dioptimalkan
   - Windows_Server-2019-Inggris-Core-ECS_Optimized
   - Windows_server-2016-Inggris-penuh-ECS_dioptimalkan
2. Pada halaman Pilih Gambar Mesin Amazon (AMI), pilih tab AMI Komunitas.
3. Dari daftar yang muncul, pilih AMI yang diverifikasi Microsoft dengan tanggal publikasi terbaru dan klik Pilih.

Jenis instans

Tipe instans mendefinisikan konfigurasi perangkat keras dan ukuran instans. Tipe instans yang lebih besar memiliki lebih banyak CPU dan memori. Untuk informasi selengkapnya, lihat Tipe instans.

- Untuk Tipe instans, pilih tipe instans untuk instans tersebut.

  Tipe instans yang Anda pilih menentukan sumber daya yang tersedia untuk menjalankan tugas Anda.

Pasangan kunci (login)

Untuk Nama pasangan kunci, pilih pasangan kunci yang ada, atau pilih Buat pasangan kunci baru untuk membuat yang baru.

> **Important**
>
> Jika Anda memilih opsi Lanjutkan tanpa pasangan kunci (Tidak direkomendasikan), Anda tidak akan dapat terhubung ke instans tersebut, kecuali Anda memilih sebuah AMI yang dikonfigurasi agar pengguna dapat masuk dengan cara lain.

Pengaturan jaringan

Konfigurasikan pengaturan jaringan, sesuai keperluan.

- Platform jaringan: Pilih Virtual Private Cloud (VPC), lalu tentukan subnet di bagian Network interface.
- VPC: Pilih VPC yang ada untuk membuat grup keamanan.
- Subnet: Anda dapat meluncurkan sebuah instans di subnet yang terkait dengan Zona Ketersediaan, Local Zone, Wavelength Zone, atau Outpost.

  Untuk meluncurkan instans di Zona Ketersediaan, pilih subnet tempat Anda akan meluncurkan instans. Untuk membuat subnet baru, pilih Buat subnet baru untuk membuka konsol Amazon VPC. Setelah selesai, kembali ke wizard peluncuran instans dan pilih ikon Segarkan untuk memuat subnet Anda dalam daftar.

  Untuk meluncurkan instans di Local Zone, pilih subnet yang Anda buat di Local Zone.

  Untuk meluncurkan sebuah instans di Outpost, pilih subnet di VPC yang Anda kaitkan dengan Outpost.
- Auto-assign IP Publik: Jika instans Anda harus dapat diakses dari internet, verifikasi bahwa bidang Auto-assign Public IP diatur ke Aktifkan. Jika tidak, atur bidang ini ke Nonaktifkan.

  > **Note**
  >
  > Instans kontainer memerlukan akses untuk berkomunikasi dengan titik akhir layanan Amazon ECS. Ini bisa melalui titik akhir VPC antarmuka atau melalui instance container Anda yang memiliki alamat IP publik.
  > Untuk informasi selengkapnya tentang titik akhir VPC antarmuka, lihat Titik akhir VPC antarmuka Amazon ECS ()AWS PrivateLink
  > Jika Anda tidak memiliki antarmuka VPC endpoint yang dikonfigurasi dan instance container Anda tidak memiliki alamat IP publik, maka mereka harus menggunakan terjemahan alamat jaringan (NAT) untuk menyediakan akses ini. Untuk informasi selengkapnya, lihat gateway NAT di Panduan Pengguna Amazon VPC dan dalam panduan ini. Konfigurasi proxy HTTP untuk instance kontainer Linux
- Firewall (grup keamanan): Gunakan grup keamanan untuk menentukan aturan firewall untuk instance container Anda. Aturan ini menentukan lalu lintas jaringan masuk yang dikirim ke instans kontainer Anda. Semua lalu lintas lainnya diabaikan.
  - Untuk memilih grup keamanan yang ada, pilih Pilih grup keamanan yang ada, dan pilih grup keamanan yang Anda buat Siapkan untuk menggunakan Amazon ECS

Mengonfigurasi penyimpanan

AMI yang Anda pilih mencakup satu atau lebih volume penyimpanan, termasuk volume root. Anda dapat menentukan volume tambahan untuk dilampirkan ke instans.

Anda dapat menggunakan tampilan Sederhana.

- Jenis penyimpanan: Konfigurasikan penyimpanan untuk instance kontainer Anda.

  Jika Anda menggunakan Amazon Linux 2 AMI Amazon ECS yang dioptimalkan Amazon, instans Anda memiliki satu volume 30 GiB yang dikonfigurasi, yang dibagi antara sistem operasi dan Docker.

  Jika Anda menggunakan AMI Amazon ECS yang dioptimalkan, instans Anda memiliki dua volume yang dikonfigurasi. Volume Root adalah untuk penggunaan sistem operasi, dan volume Amazon EBS kedua (dilampirkan ke`/dev/xvdcz`) adalah untuk penggunaan Docker.

  Anda dapat menambah atau mengurangi ukuran volume bagi instans Anda untuk memenuhi kebutuhan aplikasi Anda.

Detail lanjutan

Untuk Detail lanjutan, perluas bagian untuk melihat kolom dan menentukan parameter tambahan apa pun untuk instans.

- Opsi pembelian: Pilih Minta Instans Spot untuk meminta Instans Spot. Anda juga perlu mengatur bidang lain yang terkait dengan Instans Spot. Untuk informasi selengkapnya, lihat Permintaan Instans Spot.

  > **Note**
  >
  > Jika Anda menggunakan Instans Spot dan melihat `Not available` pesan, Anda mungkin perlu memilih jenis instans yang berbeda.

  .

- Profil instans IAM: Pilih peran IAM instance container Anda. Hal ini biasanya bernama `ecsInstanceRole`.

> **Important**
>
> Jika Anda tidak meluncurkan instans penampung dengan izin IAM yang tepat, agen Amazon ECS Anda tidak dapat terhubung ke klaster Anda. Untuk informasi selengkapnya, lihat Peran IAM instans wadah Amazon ECS.

- (Opsional) Data pengguna: Konfigurasikan instans penampung Amazon ECS Anda dengan data pengguna, seperti variabel lingkungan agen dariKonfigurasi agen kontainer Amazon ECS. Skrip data pengguna Amazon EC2 dijalankan hanya satu kali, saat instans pertama kali diluncurkan. Berikut ini adalah contoh umum dari kegunaan data pengguna:
  - Secara default, instans kontainer Anda meluncurkan ke klaster default Anda. Untuk meluncurkan ke klaster non-default, pilih daftar Detail Lanjutan. Kemudian, paste script berikut ke bidang Data pengguna, mengganti *nama_klaster_anda* dengan nama klaster Anda.

    `EnableTaskIAMRole`Menghidupkan fitur peran IAM Tugas untuk tugas.

    Selain itu, opsi berikut tersedia saat Anda menggunakan mode `awsvpc` jaringan.

    - `EnableTaskENI`: Bendera ini mengaktifkan jaringan tugas dan diperlukan saat Anda menggunakan mode `awsvpc` jaringan.
    - `AwsvpcBlockIMDS`: Bendera opsional ini memblokir akses IMDS untuk wadah tugas yang berjalan dalam mode `awsvpc` jaringan.
    - `AwsvpcAdditionalLocalRoutes`: Bendera opsional ini memungkinkan Anda memiliki rute tambahan di namespace tugas.

      Ganti `ip-address` dengan Alamat IP untuk rute tambahan, misalnya 172.31.42.23/32.

```
<powershell>
Import-Module ECSTools
Initialize-ECSAgent -Cluster your_cluster_name -EnableTaskIAMRole -EnableTaskENI -
AwsvpcBlockIMDS -AwsvpcAdditionalLocalRoutes
'["ip-address"]'
</powershell>
```

Wisaya instans peluncuran Amazon EC2 lama

Untuk meluncurkan sebuah instans kontainer

1. Buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.
3. Dari Dasbor EC2, pilih Meluncurkan instans.
4. Pada halaman Pilih Amazon Machine Image (AMI) , selesaikan langkah-langkah berikut ini:

   a. Pilih Marketplace AWS .

   b. Pilih AMI untuk instance kontainer Anda. Anda dapat mencari salah satu AMI Amazon ECS yang dioptimalkan, misalnya Windows_2019_Full_ECS_Optimized. Jika Anda tidak memilih AMI Amazon ECS yang dioptimalkan, Anda harus mengikuti prosedur di. Memasang agen kontainer Amazon ECS
5. Pada halaman Pilih Tipe Instans, pilih konfigurasi perangkat keras untuk instans Anda. Pilih tipe instans `t2.micro` yang dipilih secara default. Tipe instans yang Anda pilih menentukan sumber daya yang tersedia untuk menjalankan tugas Anda.

   Pilih Berikutnya: Konfigurasi Detail Instans saat Anda telah selesai.
6. Pada halaman Konfigurasi Detail Instans, lengkapi langkah-langkah berikut:

   a. Atur bidang Jumlah instans tergantung pada berapa banyak instans kontainer yang ingin ditambahkan ke klaster Anda.

   b. (Opsional) Untuk menggunakan Instans Spot, untuk opsi Pembelian, pilih kotak centang di sebelah Minta Instans Spot. Anda juga perlu mengatur bidang lain yang terkait dengan Instans Spot. Untuk informasi selengkapnya, lihat Permintaan Instans Spot.

   > **Note**
   >
   > Jika Anda menggunakan Instans Spot dan melihat `Not available` pesan, Anda mungkin perlu memilih jenis instans yang berbeda.

   c. Untuk Jaringan, pilih VPC yang akan digunakan untuk meluncurkan instans kontainer Anda.

   d. Untuk Subnet, pilih subnet yang akan digunakan, atau pertahankan opsi default untuk memilih subnet default di Availability Zone mana pun.

   e. Pengaturan bidang Penetapan Otomatis IP Publik bergantung pada apakah Anda ingin instans Anda dapat diakses dari internet publik. Jika instans Anda seharusnya dapat diakses

dari internet, pastikan bahwa bidang Penetapan Otomatis IP Publik diatur ke Aktifkan. Jika tidak, atur bidang ini ke Nonaktifkan.

> **Note**
>
> Instans kontainer memerlukan akses untuk berkomunikasi dengan titik akhir layanan Amazon ECS. Ini dapat dilakukan melalui VPC endpoint antarmuka atau melalui instans kontainer yang memiliki alamat IP publik.
> Untuk informasi lebih lanjut tentang VPC endpoint antarmuka, lihat Titik akhir VPC antarmuka Amazon ECS ()AWS PrivateLink.
> Jika Anda tidak memiliki VPC endpoint yang dikonfigurasi dan instans kontainer Anda tidak memiliki alamat IP publik, network address translation (NAT) harus digunakan untuk menyediakan akses ini. Untuk informasi lebih lanjut, lihat Gateway NAT dalam Panduan Pengguna Amazon VPC dan Konfigurasi proxy HTTP untuk instance kontainer Linux dalam panduan ini. Untuk informasi selengkapnya, lihat the section called “Buat virtual private cloud”.

f. Pilih peran IAM instance container Anda. Hal ini biasanya bernama `ecsInstanceRole`.

> **Important**
>
> Jika Anda tidak meluncurkan instans penampung dengan izin IAM yang tepat, agen Amazon ECS Anda tidak dapat terhubung ke klaster Anda. Untuk informasi selengkapnya, lihat Peran IAM instans wadah Amazon ECS.

g. Konfigurasikan instans penampung Amazon ECS Anda dengan data pengguna, seperti variabel lingkungan agen dariKonfigurasi agen kontainer Amazon ECS. Skrip data pengguna Amazon EC2 dijalankan hanya satu kali, saat instans pertama kali diluncurkan. Berikut ini adalah contoh umum dari kegunaan data pengguna:

- Secara default, instans kontainer Anda meluncurkan ke klaster default Anda. Untuk meluncurkan ke klaster non-default, pilih daftar Detail Lanjutan. Kemudian, paste script berikut ke bidang Data pengguna, mengganti *`nama_klaster_anda`* dengan nama klaster Anda.

  `EnableTaskIAMRole`Menghidupkan fitur peran IAM Tugas untuk tugas.

  Selain itu, opsi berikut tersedia saat Anda menggunakan mode `awsvpc` jaringan.

- `EnableTaskENI`: Bendera ini mengaktifkan jaringan tugas dan diperlukan saat Anda menggunakan mode `awsvpc` jaringan.
- `AwsvpcBlockIMDS`: Bendera opsional ini memblokir akses IMDS untuk wadah tugas yang berjalan dalam mode `awsvpc` jaringan.
- `AwsvpcAdditionalLocalRoutes`: Bendera opsional ini memungkinkan Anda memiliki rute tambahan di namespace tugas.

  Ganti `ip-address` dengan Alamat IP untuk rute tambahan, misalnya 172.31.42.23/32.

```
<powershell>
Import-Module ECSTools
Initialize-ECSAgent -Cluster your_cluster_name -EnableTaskIAMRole -
EnableTaskENI -AwsvpcBlockIMDS -AwsvpcAdditionalLocalRoutes
'["ip-address"]'
</powershell>
```

h. Pilih Berikutnya: Tambahkan Penyimpanan.

7. Pada halaman Tambahkan Penyimpanan , mengonfigurasi penyimpanan untuk instans kontainer Anda.

   Anda dapat menambah atau mengurangi ukuran volume bagi instans Anda untuk memenuhi kebutuhan aplikasi Anda.

   Setelah selesai mengonfigurasi volume, pilih Berikutnya: Tambahkan Tanda.

8. Pada halaman Tambahkan Tanda , tentukan tanda dengan memberikan kombinasi kunci dan nilai untuk instans kontainer. Pilih Tambahkan tanda lain untuk menambahkan lebih dari satu tanda ke instans kontainer Anda. Untuk informasi selengkapnya tentang sumber daya, lihat Penandaan sumber daya Amazon ECS.

   Pilih Berikutnya: Konfigurasikan Grup Keamanan setelah Anda selesai.

9. Di halaman Konfigurasikan Grup Keamanan, gunakan grup keamanan untuk menentukan aturan firewall untuk instans kontainer Anda. Aturan ini menentukan lalu lintas jaringan masuk yang dikirim ke instans kontainer Anda. Semua lalu lintas lainnya diabaikan. Pilih atau buat grup keamanan sebagai berikut, lalu pilih Tinjau dan Luncurkan.

10. Pada halaman Tinjau Peluncuran Instans, di bawah Grup Keamanan, Anda akan melihat bahwa wizard akan membuat dan memilih grup keamanan untuk Anda. Sebaliknya, pilih grup keamanan yang Anda buat di Siapkan untuk menggunakan Amazon ECS dengan menggunakan langkah-langkah berikut:

a. Pilih Edit grup keamanan.
b. Pada halaman Konfigurasi Grup Keamanan, pilih opsi Pilih grup keamanan yang ada.
c. Pilih grup keamanan yang Anda buat untuk instans kontainer Anda dari daftar grup keamanan yang ada, dan pilih Tinjau dan Luncurkan.

11. Pada halaman Tinjau Peluncuran Instans, pilih Luncurkan.
12. Di kotak dialog Pilih pasangan kunci yang sudah ada atau buat pasangan kunci baru, pilih Pilih pasangan kunci yang sudah ada lalu pilih pasangan kunci yang Anda buat saat menyiapkan.

    Saat Anda siap, pilih bidang pengakuan, lalu pilih Luncurkan Instans.
13. Halaman konfirmasi memberi tahu Anda bahwa instans Anda akan diluncurkan. Pilih Lihat Instans untuk menutup halaman konfirmasi dan kembali ke konsol tersebut.
14. Pada layar Instans, Anda dapat melihat status instans Anda. Hanya butuh waktu singkat untuk meluncurkan suatu instans. Saat Anda meluncurkan instans, status awalnya adalah `pending`. Setelah instans dimulai, statusnya berubah menjadi `running`, dan ia menerima nama DNS publik. Jika kolom DNS Publik tersembunyi, pilih Tunjukan/Sembunyikan, DNS Publik.

Menggunakan Instans Spot

Instans Spot adalah instans Amazon EC2 yang tidak digunakan yang tersedia dengan harga lebih rendah dari harga Instans Sesuai Permintaan. Karena Instans Spot memungkinkan Anda untuk meminta instans EC2 yang tidak digunakan dengan diskon besar, Anda dapat menurunkan biaya Amazon EC2 secara signifikan. Harga per jam untuk Instans Spot disebut harga Spot. Harga Spot untuk setiap jenis instans di setiap Availability Zone ditetapkan oleh Amazon EC2, dan disesuaikan secara bertahap berdasarkan penawaran dan permintaan jangka panjang untuk Instans Spot. Untuk informasi selengkapnya, lihat Instans Spot di Panduan Pengguna Amazon EC2 untuk Instans Windows.

Anda dapat mendaftarkan Instans Spot ke kluster Amazon ECS Anda. Untuk informasi selengkapnya, lihat Meluncurkan instans penampung Amazon ECS Linux.

Pengeringan Instans Spot

Amazon EC2 mengakhiri, menghentikan, atau hibernasi Instans Spot Anda ketika harga Spot melebihi harga maksimum untuk permintaan atau kapasitas Anda tidak lagi tersedia. Amazon EC2 memberikan pemberitahuan interupsi Instans Spot, yang memberikan peringatan dua menit pada instans sebelum diinterupsi. Jika pengurasan Instans Spot Amazon ECS diaktifkan pada instans,

ECS menerima pemberitahuan gangguan Instans Spot dan menempatkan instance dalam status. `DRAINING`

> **Important**
>
> Amazon ECS memantau pemberitahuan gangguan Instans Spot yang memiliki dan tindakan instans. `terminate` `stop` Jika Anda menentukan perilaku interupsi `hibernate` instans saat meminta Instans Spot atau Armada Spot, maka pengurasan Instans Spot Amazon ECS tidak didukung untuk instans tersebut.

Saat instance container disetel ke`DRAINING`, Amazon ECS mencegah tugas baru dijadwalkan untuk penempatan pada instance container. Tugas layanan pada instans kontainer pengurasan yang ada di status `PENDING` segera dihentikan. Jika ada instans kontainer di klaster yang tersedia, tugas layanan pengganti dimulai.

Anda harus mengatur `ECS_ENABLE_SPOT_INSTANCE_DRAINING` parameter sebelum memulai agen kontainer. Gunakan perintah berikut untuk mengaktifkan pengurasan Instance Spot secara manual. Gantikan *`my-cluster`* dengan nama cluster Anda.

```
[Environment]::SetEnvironmentVariable("ECS_ENABLE_SPOT_INSTANCE_DRAINING", "true",
 "Machine")

# Initialize the agent
Initialize-ECSAgent -Cluster my-cluster
```

Untuk informasi selengkapnya, lihat the section called “Meluncurkan instans kontainer”.

Bootstrapping instance wadah Windows dengan data pengguna Amazon EC2

Saat meluncurkan instans penampung Amazon ECS, Anda memiliki opsi untuk meneruskan data pengguna ke instans. Data dapat digunakan untuk melakukan tugas konfigurasi otomatis umum dan menjalankan skrip saat instance boot. Untuk Amazon ECS, kasus penggunaan yang paling umum untuk data pengguna adalah meneruskan informasi konfigurasi ke daemon Docker dan agen penampung Amazon ECS.

Anda dapat meneruskan beberapa jenis data pengguna ke Amazon EC2, termasuk boothook cloud, skrip shell, dan arahan. `cloud-init` Untuk informasi selengkapnya tentang hal ini dan tipe format lainnya, lihat dokumentasi Cloud-init.

Anda dapat meneruskan data pengguna ini saat menggunakan wizard peluncuran Amazon EC2. Untuk informasi selengkapnya, lihat Meluncurkan instans penampung Amazon ECS Linux.

Topik

- Data pengguna Windows default
- Data pengguna instalasi agen Windows
- IAM role Windows untuk tugas

Data pengguna Windows default

Contoh skrip data pengguna ini menunjukkan data pengguna default yang diterima instance container Windows Anda jika Anda menggunakan konsol. Script di bawah ini melakukan hal berikut:

- Menetapkan nama cluster ke nama yang Anda masukkan.
- Menetapkan peran IAM untuk tugas.
- Atur `json-file` dan `awslogs` sebagai driver pencatatan yang tersedia.

Selain itu, opsi berikut tersedia saat Anda menggunakan mode `awsvpc` jaringan.

- `EnableTaskENI`: Bendera ini mengaktifkan jaringan tugas dan diperlukan saat Anda menggunakan mode `awsvpc` jaringan.
- `AwsvpcBlockIMDS`: Bendera opsional ini memblokir akses IMDS untuk wadah tugas yang berjalan dalam mode `awsvpc` jaringan.
- `AwsvpcAdditionalLocalRoutes`: Bendera opsional ini memungkinkan Anda memiliki rute tambahan.

  Ganti `ip-address` dengan Alamat IP untuk rute tambahan, misalnya 172.31.42.23/32.

Anda dapat menggunakan skrip ini untuk instance kontainer Anda sendiri (asalkan diluncurkan dari AMI Server Windows Amazon ECS yang dioptimalkan).

Ganti `-Cluster` *`cluster-name`* baris untuk menentukan nama cluster Anda sendiri.

```
<powershell>
Initialize-ECSAgent -Cluster cluster-name -EnableTaskIAMRole -LoggingDrivers '["json-
file","awslogs"]' -EnableTaskENI -AwsvpcBlockIMDS -AwsvpcAdditionalLocalRoutes
```

```
'["ip-address"]'
</powershell>
```

Untuk tugas Windows yang dikonfigurasi untuk menggunakan driver pencatatan `awslogs`, Anda juga harus mengatur variabel lingkungan `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE` pada instans kontainer Anda. Gunakan sintaksis berikut:

Ganti `-Cluster cluster-name` baris untuk menentukan nama cluster Anda sendiri.

```
<powershell>
[Environment]::SetEnvironmentVariable("ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE",
 $TRUE, "Machine")
Initialize-ECSAgent -Cluster cluster-name -EnableTaskIAMRole -LoggingDrivers '["json-
file","awslogs"]'
</powershell>
```

Data pengguna instalasi agen Windows

Contoh skrip data pengguna ini menginstal agen penampung Amazon ECS pada instance yang diluncurkan dengan AMI Windows_Server-2016-English-Full-Containers. Ini telah diadaptasi dari instruksi instalasi agen di halaman README GitHubrepositori Agen Kontainer Amazon ECS.

> **Note**
>
> Skrip ini dibagikan untuk keperluan contoh. Jauh lebih mudah untuk memulai dengan wadah Windows dengan menggunakan AMI Windows Server Amazon ECS yang dioptimalkan. Untuk informasi selengkapnya, lihat Membuat cluster untuk jenis peluncuran Fargate dan Eksternal menggunakan konsol.

Anda dapat menggunakan skrip ini untuk instans kontainer Anda sendiri (asalkan mereka diluncurkan dengan versi AMI Windows_Server-2016-English-Full-Containers). Pastikan untuk mengganti baris *windows* untuk menentukan nama klaster Anda sendiri (jika Anda tidak menggunakan sebuah klaster bernama `windows`).

```
<powershell>
# Set up directories the agent uses
New-Item -Type directory -Path ${env:ProgramFiles}\Amazon\ECS -Force
New-Item -Type directory -Path ${env:ProgramData}\Amazon\ECS -Force
```

```
New-Item -Type directory -Path ${env:ProgramData}\Amazon\ECS\data -Force
# Set up configuration
$ecsExeDir = "${env:ProgramFiles}\Amazon\ECS"
[Environment]::SetEnvironmentVariable("ECS_CLUSTER", "windows", "Machine")
[Environment]::SetEnvironmentVariable("ECS_LOGFILE", "${env:ProgramData}\Amazon\ECS\log
\ecs-agent.log", "Machine")
[Environment]::SetEnvironmentVariable("ECS_DATADIR", "${env:ProgramData}\Amazon\ECS
\data", "Machine")
# Download the agent
$agentVersion = "latest"
$agentZipUri = "https://s3.amazonaws.com/amazon-ecs-agent/ecs-agent-windows-
$agentVersion.zip"
$zipFile = "${env:TEMP}\ecs-agent.zip"
Invoke-RestMethod -OutFile $zipFile -Uri $agentZipUri
# Put the executables in the executable directory.
Expand-Archive -Path $zipFile -DestinationPath $ecsExeDir -Force
Set-Location ${ecsExeDir}
# Set $EnableTaskIAMRoles to $true to enable task IAM roles
# Note that enabling IAM roles will make port 80 unavailable for tasks.
[bool]$EnableTaskIAMRoles = $false
if (${EnableTaskIAMRoles}) {
  $HostSetupScript = Invoke-WebRequest https://raw.githubusercontent.com/aws/amazon-
ecs-agent/master/misc/windows-deploy/hostsetup.ps1
  Invoke-Expression $($HostSetupScript.Content)
}
# Install the agent service
New-Service -Name "AmazonECS" `
        -BinaryPathName "$ecsExeDir\amazon-ecs-agent.exe -windows-service" `
        -DisplayName "Amazon ECS" `
        -Description "Amazon ECS service runs the Amazon ECS agent" `
        -DependsOn Docker `
        -StartupType Manual
sc.exe failure AmazonECS reset=300 actions=restart/5000/restart/30000/restart/60000
sc.exe failureflag AmazonECS 1
Start-Service AmazonECS
</powershell>
```

IAM role Windows untuk tugas

Lihat contoh Windows berikut mengenai bootstrapping peran tugas IAM.

- Konfigurasi tambahan untuk peran Windows IAM untuk tugas
- Peran IAM untuk skrip bootstrap wadah tugas

Sambungkan ke instans Windows kontainer Anda

Anda dapat terhubung ke instance Windows Anda untuk melakukan tugas administratif dasar, seperti menginstal atau memperbarui perangkat lunak atau mengakses log diagnostik. Untuk terhubung ke instans Anda menggunakan Remote Desktop Protocol (RDP), instans Windows Anda harus memenuhi prasyarat berikut.

- Instans Amazon EC2 yang dibuat dari sebagian besar AMI Windows memungkinkan Anda terhubung menggunakan Remote Desktop Protocol (RDP). RDP memungkinkan Anda untuk terhubung dan menggunakan instance Anda dengan cara yang sama seperti Anda menggunakan komputer yang duduk di depan Anda. Ini tersedia di sebagian besar edisi Windows dan tersedia untuk Mac OS.
- Instans Windows Anda harus telah diluncurkan dengan key pair Amazon EC2 yang valid. Instans Amazon EC2 tidak memiliki kata sandi, Anda menggunakan key pair untuk akses melalui RDP. Jika Anda tidak menentukan pasangan kunci saat meluncurkan instans Anda, tidak ada cara untuk terhubung ke instans. Untuk informasi selengkapnya, lihat the section called “Meluncurkan instans kontainer”.
- Pastikan grup keamanan yang terkait dengan instans Anda mengizinkan lalu lintas RDP masuk (port 3389) dari alamat IP Anda. Grup keamanan default tidak mengizinkan lalu lintas RDP masuk secara default. Untuk informasi selengkapnya, lihat Mengotorisasi lalu lintas masuk untuk instans Windows Anda di Panduan Pengguna Amazon EC2 untuk Instans Windows.

1. Cari IP publik atau alamat DNS untuk instans kontainer Anda.
   a. Buka konsol di https://console.aws.amazon.com/ecs/v2.
   b. Di panel navigasi, pilih Clusters dan pilih cluster yang menghosting instance.
   c. Pada halaman Cluster: ***name***, pilih tab Infrastructure.
   d. Di bawah Instance Container, pilih ID instance.
   e. Pada halaman Instans, rekam IP Publik atau DNS Publik untuk instans Anda.
2. Temukan nama pengguna default untuk instance kontainer AMI Anda.
3. Anda dapat terhubung menuju instans Anda dengan menggunakan RDP. Untuk informasi selengkapnya, lihat Connect ke instans Windows menggunakan RDP di Panduan Pengguna Amazon EC2 untuk Instans Windows.

Konfigurasi proxy HTTP untuk instance kontainer Windows

Anda dapat mengonfigurasi instans penampung Amazon ECS untuk menggunakan proxy HTTP untuk agen penampung Amazon ECS dan daemon Docker. Ini berguna jika instance container Anda tidak memiliki akses jaringan eksternal melalui gateway internet VPC Amazon, gateway NAT, atau instance.

Untuk mengonfigurasi instans penampung Amazon ECS Windows Anda agar menggunakan proxy HTTP, setel variabel berikut pada waktu peluncuran (dengan data pengguna Amazon EC2).

```
[Environment]::SetEnvironmentVariable("HTTP_PROXY",
"http://proxy.mydomain:port", "Machine")
```

Setel `HTTP_PROXY` ke nama host (atau alamat IP) dan nomor port proxy HTTP yang akan digunakan agen Amazon ECS untuk terhubung ke internet. Misalnya, instance container Anda mungkin tidak memiliki akses jaringan eksternal melalui gateway internet VPC Amazon, gateway NAT, atau instans.

```
[Environment]::SetEnvironmentVariable("NO_PROXY",
"169.254.169.254,169.254.170.2,\\.\pipe\docker_engine", "Machine")
```

Setel `NO_PROXY` `169.254.169.254,169.254.170.2,\\.\pipe\docker_engine` untuk memfilter metadata instans EC2, peran IAM untuk tugas, dan lalu lintas daemon Docker dari proxy.

Example Skrip data pengguna proxy HTTP Windows

Contoh PowerShell skrip data pengguna di bawah ini mengonfigurasi agen penampung Amazon ECS dan daemon Docker untuk menggunakan proxy HTTP yang Anda tentukan. Anda juga dapat menentukan klaster tempat instans kontainer akan terdaftar.

Untuk menggunakan skrip ini ketika Anda meluncurkan instans kontainer, ikuti langkah-langkah di the section called “Meluncurkan instans kontainer”. Cukup salin dan tempel PowerShell skrip di bawah ini ke bidang data Pengguna (pastikan untuk mengganti nilai contoh merah dengan informasi proxy dan cluster Anda sendiri).

> **Note**
>
> `-EnableTaskIAMRole`Opsi ini diperlukan untuk mengaktifkan peran IAM untuk tugas. Untuk informasi selengkapnya, lihat Konfigurasi tambahan untuk peran Windows IAM untuk tugas.

```
<powershell>
Import-Module ECSTools

$proxy = "http://proxy.mydomain:port"
[Environment]::SetEnvironmentVariable("HTTP_PROXY", $proxy, "Machine")
[Environment]::SetEnvironmentVariable("NO_PROXY", "169.254.169.254,169.254.170.2,\\.
\pipe\docker_engine", "Machine")

Restart-Service Docker
Initialize-ECSAgent -Cluster MyCluster -EnableTaskIAMRole
</powershell>
```

Membatalkan pendaftaran instans kontainer yang didukung Amazon EC2

> **Important**
>
> Topik ini hanya untuk instance container yang dibuat di Amazon EC2. Untuk informasi lebih lanjut tentang membatalkan pendaftaran instans eksternal, lihat Membatalkan pendaftaran instans eksternal.

Setelah selesai dengan instans penampung yang didukung Amazon EC2, Anda harus membatalkan pendaftarannya dari klaster. Mengikuti pembatalan pendaftaran, instans kontainer tidak lagi mampu menerima tugas baru.

Jika Anda memiliki tugas yang berjalan pada instans kontainer saat membatalkan pendaftaran tersebut, maka tugas ini tetap berjalan sampai Anda mengakhiri instans atau tugas berhenti melalui beberapa cara lain. Namun, tugas-tugas ini menjadi yatim piatu yang berarti mereka tidak lagi dipantau atau diperhitungkan oleh Amazon ECS. Jika tugas yatim piatu pada instance container Anda adalah bagian dari layanan Amazon ECS, maka penjadwal layanan akan memulai salinan lain dari tugas tersebut, pada instance container yang berbeda, jika memungkinkan. Setiap kontainer dalam tugas layanan yatim piatu yang terdaftar pada kelompok sasaran Application Load Balancer dideregistrasi. Kontainer tersebut memulai menguras koneksi sesuai dengan pengaturan pada penyeimbang beban atau grup target. Jika tugas tanpa sumber menggunakan mode jaringan `awsvpc`, maka antarmuka jaringan elastis tugas tersebut dihapus.

Jika Anda ingin menggunakan instans kontainer untuk beberapa tujuan lain setelah pembatalan pendaftaran, Anda harus menghentikan semua tugas yang berjalan pada instans kontainer sebelum

pembatalan pendaftaran. Penghentian ini menghentikan tugas tanpa sumber mengonsumsi sumber daya.

Saat membatalkan pendaftaran sebuah instans kontainer, pertimbangan beberapa hal berikut.

- Karena setiap instans kontainer memiliki status informasi yang unik, maka instans tersebut tidak boleh dibatalkan pendaftarannya dari suatu klaster kemudian di didaftarkan kembali di klaster lain. Untuk memindahkan sumber daya instans kontainer, kami menyarankan Anda untuk mengakhiri instans kontainer dari suatu klaster kemudian meluncurkan instans kontainer baru di klaster baru. Untuk informasi selengkapnya, lihat Menghentikan instans Anda di Panduan Pengguna Amazon EC2 untuk Instans Linux dan. Meluncurkan instans penampung Amazon ECS Linux
- Jika instance container dikelola oleh grup Auto Scaling atau AWS CloudFormation tumpukan, hentikan instance dengan memperbarui grup atau tumpukan Auto Scaling. AWS CloudFormation Jika tidak, grup Auto Scaling atau AWS CloudFormation akan membuat instance baru setelah Anda menghentikannya.
- Jika Anda menghentikan instance container yang sedang berjalan dengan agen container Amazon ECS yang terhubung, agen akan secara otomatis membatalkan pendaftaran instance dari klaster Anda. Instans kontainer yang dihentikan atau instans dengan agen yang tidak terkoneksi tidak secara otomatis dibatalkan pendaftarannya saat diakhiri.
- Membatalkan pendaftaran instance container akan menghapus instance dari klaster, tetapi instans Amazon EC2 tidak dihentikan. Jika Anda selesai menggunakan instans, pastikan Anda mengakhirinya untuk menghentikan penagihan. Untuk informasi lebih lanjut, lihat Akhiri Instans Anda dalam Panduan Pengguna Amazon EC2 untuk Instans Linux.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah tempat instans eksternal Anda didaftarkan.
3. Di panel navigasi, pilih Clusters dan pilih cluster yang menghosting instance.
4. Pada halaman Cluster: ***name***, pilih tab Infrastructure.
5. Di bawah Instance Container, pilih ID instance untuk deregister. Anda dialihkan ke halaman detail instans kontainer.
6. Pada halaman Instans Kontainer: ***id***, pilih Batalkan pendaftaran.
7. Pada layar konfirmasi, pilih Deregister.
8. Jika Anda sudah selesai dengan instance container, hentikan instans Amazon EC2 yang mendasarinya. Untuk informasi lebih lanjut, lihat Akhiri Instans Anda di Panduan Pengguna Amazon EC2 untuk Instans Linux.

Pengurasan instans terkelola Amazon ECS

Pengurasan instans terkelola memfasilitasi penghentian instans Amazon EC2 yang anggun. Hal ini memungkinkan beban kerja Anda berhenti dengan aman dan dijadwal ulang ke instans non-terminating. Pemeliharaan dan pembaruan infrastruktur dilakukan tanpa khawatir akan gangguan pada beban kerja. Dengan menggunakan pengeringan instans terkelola, Anda menyederhanakan alur kerja manajemen infrastruktur yang memerlukan penggantian instans Amazon EC2 sambil memastikan ketahanan dan ketersediaan aplikasi.

Pengurasan instans terkelola Amazon ECS berfungsi dengan penggantian instans grup Auto Scaling. Berdasarkan penyegaran instans dan masa pakai instans maksimum, pelanggan dapat memastikan bahwa mereka tetap mematuhi OS terbaru dan mandat keamanan untuk kapasitas mereka.

Pengurasan instans terkelola hanya dapat digunakan dengan penyedia kapasitas Amazon ECS dan diaktifkan saat membuat atau memperbarui penyedia kapasitas grup Auto Scaling menggunakan konsol AWS CLI Amazon ECS, atau SDK.

Peristiwa berikut dicakup oleh pengeringan instans terkelola Amazon ECS.

- Penyegaran instans grup Auto Scaling - Gunakan penyegaran instans untuk melakukan penggantian bergulir instans Amazon EC2 di grup Auto Scaling alih-alih melakukannya secara manual dalam batch. Ini sangat berguna ketika Anda perlu mengganti sejumlah besar instance. Penyegaran instans dimulai melalui konsol Amazon EC2 atau `StartInstanceRefresh` API. Pastikan Anda memilih `Replace` perlindungan Scale-in saat menelepon `StartInstanceRefresh` jika Anda menggunakan proteksi terminasi terkelola.
- Masa pakai instans maksimum - Anda dapat menentukan masa pakai maksimum saat mengganti instance grup Auto Scaling. Hal ini berguna untuk menjadwalkan instance penggantian berdasarkan kebijakan keamanan internal atau kepatuhan.
- Penskalaan grup Auto Scaling - Berdasarkan kebijakan penskalaan dan tindakan penskalaan terjadwal, grup Auto Scaling mendukung penskalaan otomatis instans. Dengan menggunakan grup Auto Scaling dengan penyedia kapasitas Amazon ECS, yang memfasilitasi penskalaan otomatis cluster, Anda dapat menskalakan instans grup Auto Scaling saat tidak ada tugas yang berjalan di dalamnya.
- Pemeriksaan kesehatan grup Auto Scaling - Grup Auto Scaling mendukung banyak pemeriksaan kesehatan untuk mengelola penghentian kejadian yang tidak sehat.
- AWS CloudFormation pembaruan tumpukan - Anda dapat menambahkan `UpdatePolicy` atribut ke Anda AWS CloudFormation untuk melakukan pembaruan bergulir saat grup berubah.

- Penyeimbangan kembali kapasitas spot - Grup Auto Scaling mencoba mengganti instans Spot secara proaktif yang memiliki risiko interupsi lebih tinggi berdasarkan pemberitahuan penyeimbangan kembali kapasitas Amazon EC2. Grup Auto Scaling menghentikan instance lama saat penggantian diluncurkan dan sehat. Pengurasan instans yang dikelola Amazon ECS menguras instans Spot yang berakhir seperti menguras instance Non-Spot.
- Gangguan spot - Instans spot diakhiri dengan pemberitahuan dua menit. Pengurasan instans yang dikelola Amazon ECS menempatkan instance dalam status menguras sebagai respons.

Kait siklus hidup Auto Scaling Amazon EC2 dengan pengurasan instans terkelola

Kait siklus hidup grup Auto Scaling memungkinkan pelanggan membuat solusi yang dipicu oleh peristiwa tertentu dalam siklus hidup instance dan melakukan tindakan kustom saat peristiwa tertentu terjadi. Grup Auto Scaling memungkinkan hingga 50 kait. Beberapa kait terminasi dapat ada dan dieksekusi secara paralel, dan grup Auto Scaling menunggu semua kait selesai sebelum mengakhiri sebuah instance.

Selain penghentian kait yang dikelola Amazon ECS, Anda juga dapat mengonfigurasi kait penghentian siklus hidup Anda sendiri. Pengait siklus hidup memilikidefault action, dan sebaiknya continue setelan sebagai default untuk memastikan kait lain, seperti hook yang dikelola Amazon ECS, tidak terpengaruh oleh kesalahan apa pun dari kait khusus.

Jika Anda telah mengonfigurasi hook siklus hidup penghentian grup Auto Scaling dan juga mengaktifkan pengurasan instans yang dikelola Amazon ECS, kedua kait siklus hidup akan dijalankan. Pengaturan waktu relatif, bagaimanapun, tidak dijamin. Kait siklus hidup memiliki default action pengaturan untuk menentukan tindakan yang akan diambil saat batas waktu berlalu. Jika terjadi kegagalan, kami sarankan untuk menggunakan continue sebagai hasil default di kait khusus Anda. Ini memastikan kait lain, terutama kait yang dikelola Amazon ECS, tidak terpengaruh oleh kesalahan apa pun di kait lifecyle kustom Anda. Hasil alternatif abandon menyebabkan semua kait lainnya dilewati dan harus dihindari.

Tugas dan pengeringan instans terkelola

Pengurasan instans terkelola Amazon ECS menggunakan fitur pengurasan yang ada yang ditemukan dalam instance kontainer. Fitur pengurasan instans kontainer melakukan penggantian dan penghentian untuk tugas replika yang termasuk dalam layanan Amazon ECS. Tugas mandiri, seperti yang dipanggil olehRunTask, yang berada dalam running status pending atau akan tetap tidak terpengaruh. Anda harus menunggu sampai ini selesai atau menghentikannya secara manual.

Instance kontainer tetap dalam `draining` status sampai semua tugas dihentikan atau 48 jam telah berlalu. Tugas daemon adalah yang terakhir berhenti setelah semua tugas replika berhenti.

Pengurasan instans terkelola dan perlindungan terminasi terkelola

Karena pengeringan instans terkelola Amazon ECS memfasilitasi pengeringan instans Amazon EC2 yang anggun, ini membuat aplikasi Anda tidak terganggu oleh peristiwa penghentian apa pun. Pengurasan instans terkelola memfasilitasi penghentian tugas layanan Amazon ECS secara anggun untuk instans yang sedang diskalakan, bahkan jika penghentian terkelola dinonaktifkan.

Tabel berikut merangkum perilaku untuk kombinasi yang berbeda dari terminasi terkelola dan pengeringan terkelola.

Pengakhiran terkelola	Pengeringan terkelola	Hasil
Diaktifkan	Diaktifkan	Amazon ECS melindungi instans Amazon EC2 yang menjalankan tugas agar tidak dihentikan oleh peristiwa penskalaan. Setiap instance yang mengalami penghentian, seperti yang tidak memiliki perlindungan

Pengakhiran terkelola	Pengeringan terkelola	Hasil
		terminasi yang ditetapkan, telah menerima interupsi Spot, atau dipaksa oleh penyegaran instans akan terkuras dengan baik.

Pengakhiran terkelola	Pengeringan terkelola	Hasil
Nonaktif	Diaktifkan	Amazon ECS tidak melindungi instans Amazon EC2 yang menjalankan tugas agar tidak diskalakan. Namun, setiap contoh yang sedang dihentikan akan dikeringkan dengan baik.

Pengakhiran terkelola	Pengeringan terkelola	Hasil
Diaktifkan	Dinonaktifkan	Amazon ECS melindungi instans Amazon EC2 yang menjalankan tugas agar tidak dihentikan oleh peristiwa penskalaan. Namun, instance masih dapat dihentikan oleh gangguan Spot atau penyegaran instans paksa, atau jika mereka tidak menjalankan tugas apa pun. Amazon ECS tidak melakukan pengering

Pengakhiran terkelola	Pengeringan terkelola	Hasil
		an yang baik untuk instans ini, dan meluncurkan tugas layanan pengganti setelah berhenti.
Nonaktif	Nonaktif	Instans Amazon EC2 dapat diskalakan atau dihentikan kapan saja, meskipun menjalankan tugas Amazon ECS. Amazon ECS akan meluncurkan tugas layanan pengganti setelah berhenti.

Pengurasan instans terkelola dan pengeringan instans Spot

Dengan pengurasan instans Amazon ECS Spot, Anda dapat menyetel variabel lingkungan `ECS_ENABLE_SPOT_INSTANCE_DRAINING` di Agen Amazon ECS yang memungkinkan Amazon ECS menempatkan instans dalam status pengurasan sebagai respons terhadap gangguan Spot dua menit. Pengurasan instans terkelola Amazon ECS memfasilitasi penutupan instans Amazon EC2 yang mengalami penghentian karena berbagai alasan, bukan hanya gangguan Spot. Misalnya, Anda dapat menggunakan penyeimbangan ulang kapasitas Auto Scaling Amazon EC2 untuk secara proaktif mengganti instans Spot dengan risiko interupsi yang tinggi, dan pengurasan instans terkelola melakukan shutdown yang anggun pada instans Spot yang diganti. Saat menggunakan pengurasan instans terkelola, Anda tidak perlu mengaktifkan pengurasan instans Spot secara terpisah, sehingga data pengguna `ECS_ENABLE_SPOT_INSTANCE_DRAINING` di ASG berlebihan.

Amazon ECS mengelola pemecahan masalah pengurasan instans

Acara pengeringan instans terkelola Amazon ECS dipublikasikan ke Amazon EventBridge, dan Amazon ECS membuat aturan EventBridge terkelola di bus default akun Anda untuk mendukung pengeringan instans terkelola. Anda dapat memfilter peristiwa ini ke AWS layanan lain seperti Lambda, Amazon SNS, dan Amazon SQS untuk memantau dan memecahkan masalah.

- Auto Scaling Amazon EC2 mengirimkan peristiwa EventBridge ke mana kait siklus hidup dipanggil.
- Pemberitahuan gangguan spot dipublikasikan ke. EventBridge
- Amazon ECS menghasilkan pesan kesalahan yang dapat diambil di konsol Amazon ECS dan API jika ada kegagalan saat menyediakan instans terkelola yang menguras sumber daya.
- EventBridge memiliki mekanisme coba lagi yang dibangun sebagai mitigasi untuk kegagalan sementara.

Menggunakan pengeringan instans terkelola Amazon ECS

Anda mengaktifkan pengurasan instans terkelola saat membuat atau memperbarui penyedia kapasitas grup Auto Scaling menggunakan konsol Amazon ECS dan. AWS CLI

> **Note**
>
> Pengurasan instans terkelola diaktifkan secara default saat Anda membuat penyedia kapasitas.

Berikut ini adalah contoh penggunaan AWS CLI untuk membuat penyedia kapasitas dengan pengeringan instans terkelola diaktifkan dan memungkinkan pengeringan instans terkelola untuk penyedia kapasitas kluster yang ada.

Buat penyedia kapasitas dengan pengurasan instans terkelola diaktifkan

Untuk membuat penyedia kapasitas dengan pengurasan instans terkelola diaktifkan, gunakan `create-capacity-provider` perintah.

```
aws ecs create-capacity-provider \
--name capacity-provider \
--auto-scaling-group-provider '{
  "autoScalingGroupArn": "asg-arn",
  "managedScaling": {
    "status": "ENABLED",
    "targetCapacity": 100,
    "minimumScalingStepSize": 1,
    "maximumScalingStepSize": 1
  },
  "managedDraining": "ENABLED",
  "managedTerminationProtection": "ENABLED",
}'
```

Respons:

```
{
    "capacityProvider": {
        "capacityProviderArn": "capacity-provider-arn",
        "name": "capacity-provider",
        "status": "ACTIVE",
        "autoScalingGroupProvider": {
            "autoScalingGroupArn": "asg-arn",
            "managedScaling": {
                "status": "ENABLED",
                "targetCapacity": 100,
                "minimumScalingStepSize": 1,
                "maximumScalingStepSize": 1
            },
            "managedTerminationProtection": "ENABLED"
            "managedDraining": "ENABLED"
        }
    }
```

```
}
```

Aktifkan pengeringan instans terkelola untuk penyedia kapasitas kluster yang ada

Aktifkan pengeringan instans terkelola untuk penyedia kapasitas kluster yang ada menggunakan `update-capacity-provider` perintah. Anda melihat bahwa `managedDraining` saat ini mengatakan `DISABLED` dan `updateStatus` berkata`UPDATE_IN_PROGRESS`.

```
aws ecs update-capacity-provider \
--name cp-draining \
--auto-scaling-group-provider '{
  "managedDraining": "ENABLED"
}
```

Respons:

```
{
    "capacityProvider": {
        "capacityProviderArn": "cp-draining-arn",
        "name": "cp-draining",
        "status": "ACTIVE",
        "autoScalingGroupProvider": {
            "autoScalingGroupArn": "asg-draining-arn",
            "managedScaling": {
                "status": "ENABLED",
                "targetCapacity": 100,
                "minimumScalingStepSize": 1,
                "maximumScalingStepSize": 1,
                "instanceWarmupPeriod": 300
            },
            "managedTerminationProtection": "DISABLED",
            "managedDraining": "DISABLED" // before update
        },
        "updateStatus": "UPDATE_IN_PROGRESS", // in progress and need describe again to
 find out the result
        "tags": [
        ]
    }
}
```

Gunakan `describe-clusters` perintah dan sertakan`ATTACHMENTS`. `status`Lampiran pengeringan instance terkelola adalah`PRECREATED`, dan keseluruhannya `attachmentsStatus` adalah`UPDATING`.

```
aws ecs describe-clusters --clusters cluster-name --include ATTACHMENTS
```

Respons:

```
{
    "clusters": [
        {
            ...

            "capacityProviders": [
                "cp-draining"
            ],
            "defaultCapacityProviderStrategy": [],
            "attachments": [
                # new precreated managed draining attachment
                {
                    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
                    "type": "managed_draining",
                    "status": "PRECREATED",
                    "details": [
                        {
                            "name": "capacityProviderName",
                            "value": "cp-draining"
                        },
                        {
                            "name": "autoScalingLifecycleHookName",
                            "value": "ecs-managed-draining-termination-hook"
                        }
                    ]
                },

                ...

            ],
            "attachmentsStatus": "UPDATING"
        }
    ],
    "failures": []
```

```
}
```

Ketika pembaruan selesai, gunakan`describe-capacity-providers`, dan Anda lihat `managedDraining` sekarang`ENABLED`.

```
aws ecs describe-capacity-providers --capacity-providers cp-draining
```

Respons:

```
{
    "capacityProviders": [
        {
            "capacityProviderArn": "cp-draining-arn",
            "name": "cp-draining",
            "status": "ACTIVE",
            "autoScalingGroupProvider": {
                "autoScalingGroupArn": "asg-draning-arn",
                "managedScaling": {
                    "status": "ENABLED",
                    "targetCapacity": 100,
                    "minimumScalingStepSize": 1,
                    "maximumScalingStepSize": 1,
                    "instanceWarmupPeriod": 300
                },
                "managedTerminationProtection": "DISABLED",
                "managedDraining": "ENABLED" // successfully update
            },
            "updateStatus": "UPDATE_COMPLETE",
            "tags": []
        }
    ]
}
```

Menjadwalkan kontainer Anda di Amazon ECS

Amazon Elastic Container Service (Amazon ECS) adalah sistem konkurensi optimis status bersama yang menyediakan kemampuan penjadwalan fleksibel untuk beban kerja kontainer Anda. Penjadwal Amazon ECS menggunakan informasi status cluster yang sama dengan Amazon ECS API untuk membuat keputusan penempatan yang tepat.

Amazon ECS menyediakan penjadwal layanan untuk tugas dan aplikasi yang berjalan lama. Ini juga menyediakan kemampuan untuk menjalankan tugas mandiri atau tugas terjadwal untuk pekerjaan batch atau tugas lari tunggal. Anda dapat menentukan strategi dan kendala penempatan tugas untuk menjalankan tugas yang paling sesuai dengan kebutuhan Anda. Misalnya, Anda dapat menentukan apakah tugas berjalan di beberapa Availability Zone atau dalam Availability Zone tunggal. Dan, secara opsional, Anda dapat mengintegrasikan tugas dengan penjadwal kustom atau pihak ketiga Anda sendiri.

Penjadwal layanan

Penjadwal layanan cocok untuk layanan dan aplikasi stateless yang berjalan lama. Penjadwal layanan memastikan bahwa strategi penjadwalan yang Anda tentukan diikuti dan menjadwal ulang tugas ketika tugas gagal. Sebagai contoh, jika infrastruktur pokok gagal, penjadwal layanan dapat menjadwal ulang tugas.

Penjadwal layanan juga menggantikan tugas yang ditentukan tidak sehat setelah pemeriksaan kesehatan kontainer atau pemeriksaan kesehatan kelompok sasaran penyeimbang beban gagal. Penggantian ini tergantung pada parameter definisi `maximumPercent` dan `desiredCount` layanan. Jika tugas ditandai tidak sehat, penjadwal layanan akan memulai tugas pengganti terlebih dahulu. Jika tugas penggantian memiliki status kesehatan`HEALTHY`, penjadwal layanan menghentikan tugas yang tidak sehat. Jika tugas penggantian memiliki status kesehatan`UNHEALTHY`, penjadwal akan menghentikan tugas penggantian yang tidak sehat atau tugas tidak sehat yang ada untuk mendapatkan jumlah tugas total yang sama`desiredCount`. Jika `maximumPercent` parameter membatasi penjadwal untuk memulai tugas penggantian terlebih dahulu, penjadwal akan menghentikan tugas yang tidak sehat satu per satu secara acak untuk membebaskan kapasitas, dan kemudian memulai tugas pengganti. Proses start dan stop berlanjut sampai semua tugas yang tidak sehat diganti dengan tugas yang sehat. Setelah semua tugas yang tidak sehat telah diganti dan hanya tugas sehat yang berjalan, jika jumlah tugas total melebihi`desiredCount`, tugas sehat dihentikan secara acak hingga jumlah tugas total sama. `desiredCount` Untuk informasi selengkapnya tentang `maximumPercent` dan`desiredCount`, lihat Parameter definisi layanan.

> **Note**
>
> Perilaku ini tidak berlaku untuk tugas yang dijalankan dan dikelola oleh layanan yang menggunakan jenis penerapan pembaruan bergulir. Selama pembaruan bergulir, penjadwal layanan pertama-tama menghentikan tugas yang tidak sehat dan kemudian memulai tugas penggantian.

Ada dua strategi penjadwal layanan yang tersedia:

- `REPLICA`Strategi penjadwalan replika menempatkan dan mempertahankan jumlah tugas yang diinginkan di seluruh cluster Anda. Secara default tugas tersebar di seluruh Availability Zone. Anda dapat menggunakan strategi penempatan tugas dan kendala untuk menyesuaikan keputusan penempatan tugas. Untuk informasi selengkapnya, lihat Replika.
- `DAEMON`—Strategi penjadwalan daemon menerapkan tepat satu tugas pada setiap instance kontainer aktif yang memenuhi semua batasan penempatan tugas yang Anda tentukan di cluster Anda. Saat menggunakan strategi ini, tidak perlu menentukan jumlah tugas yang diinginkan, strategi penempatan tugas, atau menggunakan kebijakan Auto Scaling Layanan. Untuk informasi selengkapnya, lihat Daemon.

  > **Note**
  >
  > Tugas Fargate tidak mendukung strategi `DAEMON` penjadwalan.

Penjadwal layanan secara opsional juga memastikan bahwa tugas terdaftar terhadap penyeimbang beban Elastic Load Balancing. Anda dapat memperbarui layanan Anda yang dikelola oleh penjadwal layanan. Ini mungkin termasuk men-deploy ketentuan tugas baru atau mengubah jumlah tugas yang diinginkan yang sedang berjalan. Secara default, penjadwal layanan menyebar tugas di beberapa Availability Zone. Namun, Anda dapat menggunakan strategi penempatan tugas dan kendala untuk menyesuaikan keputusan penempatan tugas. Untuk informasi selengkapnya, lihat Layanan-layanan Amazon ECS.

Tugas mandiri

Anda dapat menjalankan tugas mandiri ketika Anda memiliki proses seperti pekerjaan batch yang melakukan pekerjaan dan kemudian berhenti. Misalnya, Anda dapat memiliki panggilan proses

`RunTask` ketika pekerjaan masuk ke antrian. Tugas menarik pekerjaan dari antrean, melaksanakan pekerjaan, dan kemudian keluar. Menggunakan `RunTask`, Anda dapat mengizinkan strategi penempatan tugas default untuk mendistribusikan tugas secara acak di klaster Anda. Hal ini meminimalkan kemungkinan instans tunggal mendapatkan jumlah tugas yang tidak proporsional. Kalau tidak, Anda dapat menggunakan `RunTask` untuk menyesuaikan bagaimana penjadwal menempatkan tugas menggunakan strategi dan kendala penempatan tugas.

Penjadwal kustom

Dengan Amazon ECS, Anda dapat membuat penjadwal sendiri atau menggunakan penjadwal pihak ketiga. Untuk informasi selengkapnya, lihat Cara membuat penjadwal khusus untuk Amazon ECS. Penjadwal khusus menggunakan operasi StartTaskAPI untuk menempatkan tugas pada instance kontainer tertentu dalam klaster Anda.

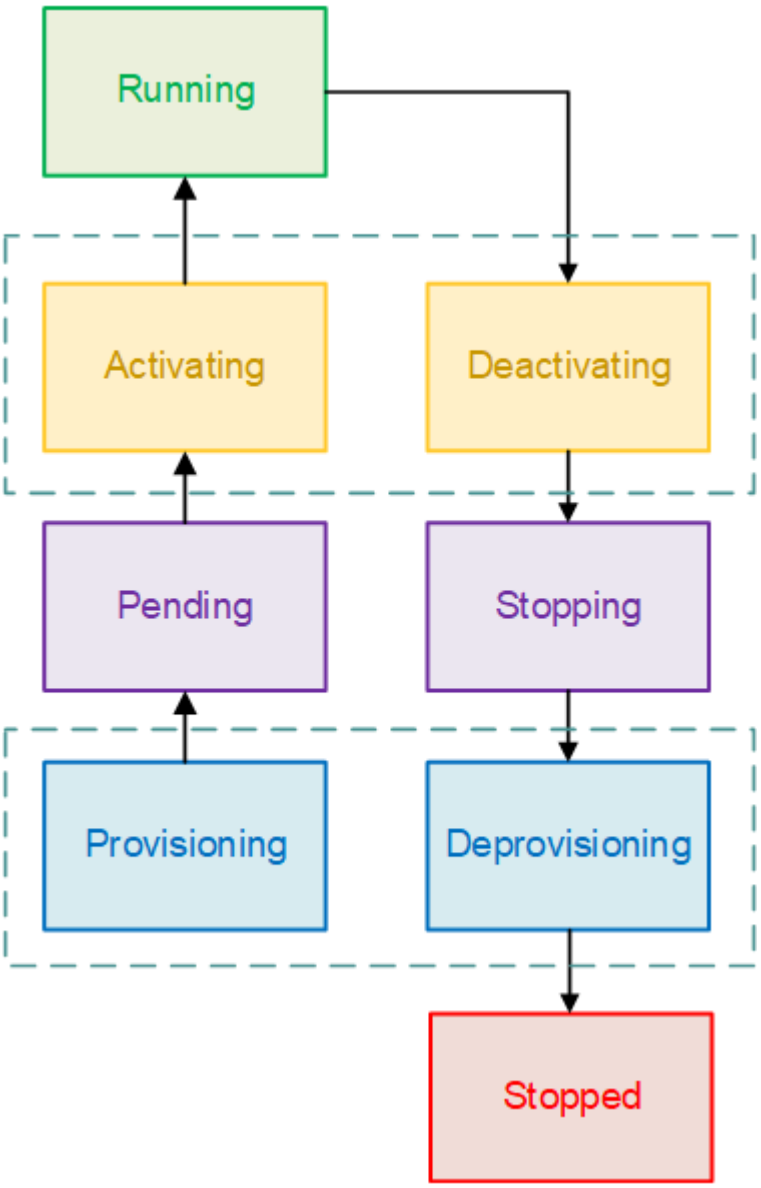
> **Note**
>
> Penjadwal khusus hanya kompatibel dengan tugas yang dihosting pada instans EC2. Jika Anda menggunakan Amazon ECS di Fargate, StartTask API tidak berfungsi.

Siklus hidup tugas Amazon ECS

Ketika tugas dimulai, baik secara manual atau sebagai bagian dari layanan, itu dapat melalui beberapa status sebelum selesai sendiri atau dihentikan secara manual. Beberapa tugas dimaksudkan untuk dijalankan sebagai tugas batch yang secara alami terus berjalan dari `PENDING` ke `RUNNING` ke `STOPPED`. Tugas-tugas lain, yang dapat menjadi bagian dari layanan, dimaksudkan untuk terus berjalan tanpa batas waktu, atau untuk dinaik-turunkan skalanya sesuai kebutuhan.

Ketika perubahan status tugas diminta, seperti menghentikan tugas atau memperbarui jumlah layanan yang diinginkan untuk menskalakannya naik atau turun, agen penampung Amazon ECS melacak perubahan ini sebagai status (`lastStatus`) tugas terakhir yang diketahui dan status (`desiredStatus`) tugas yang diinginkan. Kedua status yang diketahui terakhir dan status yang diinginkan dari tugas dapat dilihat baik di konsol atau dengan menggambarkan tugas dengan API atau AWS CLI.

Diagram alur di bawah ini menunjukkan alur siklus hidup tugas.



Status siklus hidup

Berikut ini adalah deskripsi dari masing-masing status siklus hidup tugas.

PENYEDIAAN

Amazon ECS harus melakukan langkah-langkah tambahan sebelum tugas diluncurkan. Misalnya, untuk tugas yang menggunakan mode jaringan `awsvpc`, antarmuka jaringan elastis perlu ditetapkan.

MENUNGGU

Ini adalah keadaan transisi di mana Amazon ECS sedang menunggu agen kontainer untuk mengambil tindakan lebih lanjut. Tugas tetap dalam status tertunda sampai ada sumber daya yang tersedia untuk tugas tersebut.

MENGAKTIFKAN

Ini adalah keadaan transisi di mana Amazon ECS harus melakukan langkah-langkah tambahan setelah tugas diluncurkan tetapi sebelum tugas dapat bertransisi ke `RUNNING` status. Misalnya, untuk tugas-tugas yang layanan pencariannya dikonfigurasi, sumber daya layanan pencarian harus dibuat. Untuk tugas yang merupakan bagian dari layanan yang dikonfigurasi untuk menggunakan beberapa grup target Elastic Load Balancing, pendaftaran grup target terjadi selama status ini.

BERJALAN

Tugas ini berhasil berjalan.

MENONAKTIFKAN

Ini adalah keadaan transisi di mana Amazon ECS harus melakukan langkah-langkah tambahan sebelum tugas dihentikan. Misalnya, untuk tugas yang merupakan bagian dari layanan yang dikonfigurasi untuk menggunakan beberapa grup target Elastic Load Balancing, deregistrasi grup target terjadi selama status ini.

BERHENTI

Ini adalah keadaan transisi di mana Amazon ECS sedang menunggu agen kontainer untuk mengambil tindakan lebih lanjut.

Untuk wadah Linux, agen kontainer akan mengirim `SIGTERM` sinyal untuk memberi tahu aplikasi harus selesai dan dimatikan, dan kemudian mengirim `SIGKILL` setelah menunggu `StopTimeout` durasi yang ditetapkan dalam definisi tugas.

PEMBATALAN PENYEDIAAN

Amazon ECS harus melakukan langkah-langkah tambahan setelah tugas berhenti tetapi sebelum tugas beralih ke negara bagian`STOPPED`. Misalnya, untuk tugas yang menggunakan mode jaringan`awsvpc`, antarmuka jaringan elastis perlu dilepas dan dihapus.

DIHENTIKAN

Tugas telah berhasil dihentikan.

DIHAPUS

Ini adalah keadaan transisi ketika tugas berhenti. Status ini tidak ditampilkan di konsol, tetapi ditampilkan di`describe-tasks`.

Bagaimana Amazon ECS menempatkan tugas pada instans kontainer

Anda dapat menggunakan penempatan tugas untuk mengonfigurasi Amazon ECS untuk menempatkan tugas Anda pada instance container yang memenuhi kriteria tertentu, misalnya Availability Zone atau tipe instans.

Berikut ini adalah komponen penempatan tugas:

- Strategi penempatan tugas - Algoritma untuk memilih instance kontainer untuk penempatan tugas atau tugas untuk penghentian. Misalnya, Amazon ECS dapat memilih instans kontainer secara acak, atau dapat memilih instance kontainer sedemikian rupa sehingga tugas didistribusikan secara merata di seluruh grup instance.
- Kelompok tugas - Sekelompok tugas terkait, misalnya tugas database.
- Kendala penempatan tugas - Ini adalah aturan yang harus dipenuhi untuk menempatkan tugas pada instance kontainer. Jika keadaan tidak terpenuhi, tugas tidak ditempatkan dan tetap di `PENDING` negara bagian. Misalnya, Anda dapat menggunakan kendala untuk menempatkan tugas hanya pada jenis instance tertentu.

Amazon ECS memiliki algoritma yang berbeda untuk jenis peluncuran.

Jenis peluncuran EC2

Untuk tugas yang menggunakan tipe peluncuran EC2, Amazon ECS harus menentukan tempat untuk menempatkan tugas berdasarkan persyaratan yang ditentukan dalam definisi tugas, seperti CPU dan memori. Demikian pula, saat Anda menurunkan jumlah tugas, Amazon ECS harus menentukan tugas mana yang akan dihentikan. Anda dapat menerapkan strategi dan batasan penempatan tugas untuk menyesuaikan cara Amazon ECS menempatkan dan mengakhiri tugas.

Strategi penempatan tugas default bergantung pada apakah Anda menjalankan tugas secara manual (tugas mandiri) atau dalam layanan. Untuk tugas yang berjalan sebagai bagian dari layanan Amazon

ECS, strategi penempatan tugas `spread` menggunakan. `attribute:ecs.availability-zone` Tidak ada batasan penempatan tugas default untuk tugas dalam layanan. Untuk informasi selengkapnya, lihat Menjadwalkan kontainer Anda di Amazon ECS.

> **Note**
>
> Strategi penempatan tugas adalah upaya terbaik. Amazon ECS masih mencoba untuk menempatkan tugas bahkan ketika opsi penempatan paling optimal tidak tersedia. Namun, kendala penempatan tugas bersifat mengikat, dan mereka dapat mencegah penempatan tugas.

Anda dapat menggunakan strategi dan kendala penempatan tugas bersama-sama. Misalnya, Anda dapat menggunakan strategi penempatan tugas dan kendala penempatan tugas untuk mendistribusikan tugas di tugas Availability Zone dan paket bin berdasarkan memori dalam setiap Availability Zone, tetapi hanya untuk instans G2.

Saat Amazon ECS menempatkan tugas, Amazon menggunakan proses berikut untuk memilih instance container:

1. Identifikasi instance container yang memenuhi persyaratan CPU, GPU, memori, dan port dalam definisi tugas.
2. Identifikasi instance kontainer yang memenuhi kendala penempatan tugas.
3. Identifikasi instance kontainer yang memenuhi strategi penempatan tugas.
4. Pilih instance kontainer untuk penempatan tugas.

Jenis peluncuran Fargate

Strategi penempatan tugas dan kendala tidak didukung untuk tugas yang menggunakan tipe peluncuran Fargate. Fargate akan mencoba yang terbaik untuk menyebarkan tugas di seluruh Availability Zone yang dapat diakses. Jika penyedia kapasitas mencakup Fargate dan Fargate Spot, perilaku spread bersifat independen untuk setiap penyedia kapasitas.

Gunakan strategi untuk menentukan penempatan tugas Amazon ECS

Untuk tugas yang menggunakan tipe peluncuran EC2, Amazon ECS harus menentukan tempat untuk menempatkan tugas berdasarkan persyaratan yang ditentukan dalam definisi tugas, seperti CPU dan

memori. Demikian pula, saat Anda menurunkan jumlah tugas, Amazon ECS harus menentukan tugas mana yang akan dihentikan. Anda dapat menerapkan strategi dan batasan penempatan tugas untuk menyesuaikan cara Amazon ECS menempatkan dan mengakhiri tugas.

Strategi penempatan tugas default bergantung pada apakah Anda menjalankan tugas secara manual (tugas mandiri) atau dalam layanan. Untuk tugas yang berjalan sebagai bagian dari layanan Amazon ECS, strategi penempatan tugas `spread` menggunakan. `attribute:ecs.availability-zone` Tidak ada batasan penempatan tugas default untuk tugas dalam layanan. Untuk informasi selengkapnya, lihat Menjadwalkan kontainer Anda di Amazon ECS.

> **Note**
>
> Strategi penempatan tugas adalah upaya terbaik. Amazon ECS masih mencoba untuk menempatkan tugas bahkan ketika opsi penempatan paling optimal tidak tersedia. Namun, kendala penempatan tugas bersifat mengikat, dan mereka dapat mencegah penempatan tugas.

Anda dapat menggunakan strategi dan kendala penempatan tugas bersama-sama. Misalnya, Anda dapat menggunakan strategi penempatan tugas dan kendala penempatan tugas untuk mendistribusikan tugas di tugas Availability Zone dan paket bin berdasarkan memori dalam setiap Availability Zone, tetapi hanya untuk instans G2.

Saat Amazon ECS menempatkan tugas, Amazon menggunakan proses berikut untuk memilih instance container:

1. Identifikasi instance container yang memenuhi persyaratan CPU, GPU, memori, dan port dalam definisi tugas.
2. Identifikasi instance kontainer yang memenuhi kendala penempatan tugas.
3. Identifikasi instance kontainer yang memenuhi strategi penempatan tugas.
4. Pilih instance kontainer untuk penempatan tugas.

Anda menentukan strategi penempatan tugas dalam definisi layanan, atau definisi tugas menggunakan `placementStrategy` parameter.

```
"placementStrategy": [
    {
```

```
        "field": "The field to apply the placement strategy against",
        "type": "The placement strategy to use"
    }
]
```

Anda dapat menentukan strategi saat menjalankan task (RunTask), membuat layanan baru (CreateService), atau memperbarui layanan yang ada (UpdateService).

Tabel berikut menjelaskan jenis dan bidang yang tersedia.

tipe	Nilai bidang yang valid	
`binpack` Tugas ditempatkan pada instans kontainer sehingga meninggalkan jumlah paling sedikit untuk CPU atau memori yang tidak terpakai. Strategi ini meminimalkan jumlah instans kontainer yang digunakan. Ketika strategi ini digunakan dan tindakan scale-in diambil, Amazon ECS menghenti kan tugas. Hal ini dilakukan berdasarkan jumlah sumber daya yang tersisa pada instance container setelah tugas dihentikan. Instance kontainer yang memiliki sumber daya paling banyak yang tersisa setelah penghenti an tugas telah menghentikan tugas tersebut.	• cpu • memori	
`random`	Tidak digunakan	

tipe	Nilai bidang yang valid	
Tugas ditempatkan secara acak.		
`spread` Tugas ditempatkan secara merata berdasarkan nilai yang ditentukan. Tugas layanan tersebar berdasarkan tugas dari layanan tersebut. Tugas mandiri tersebar berdasarkan tugas dari grup tugas yang sama. Untuk informasi selengkapnya tentang grup tugas, lihat Tugas Amazon ECS terkait grup . Saat `spread` strategi digunakan dan tindakan penskalaan diambil, Amazon ECS memilih tugas untuk dihentikan yang menjaga keseimbangan di seluruh Availability Zone. Dalam Availability Zone, tugas dipilih secara acak.	• `instanceId` (atau`host`, yang memiliki efek yang sama) • platform atau atribut khusus apa pun yang diterapkan ke instance container, seperti `attribute:ecs.availability-zone`	

Strategi penempatan tugas dapat diperbarui untuk layanan yang ada juga. Untuk informasi selengkapnya, lihat Bagaimana Amazon ECS menempatkan tugas pada instans kontainer.

Anda dapat membuat strategi penempatan tugas yang menggunakan beberapa strategi dengan membuat array strategi sesuai urutan yang Anda inginkan. Misalnya, jika Anda ingin menyebarkan tugas di seluruh Availability Zones dan kemudian bin pack task berdasarkan memori dalam setiap Availability Zone, tentukan strategi Availability Zone yang diikuti oleh strategi memori. Misalnya strategi, lihatContoh strategi penempatan tugas Amazon ECS.

Contoh strategi penempatan tugas Amazon ECS

Anda dapat menentukan strategi penempatan tugas dengan tindakan berikut: CreateService, UpdateService, dan RunTask.

Contoh-contoh

- Mendistribusikan tugas secara merata di seluruh Availability Zone
- Mendistribusikan tugas secara merata di semua contoh
- Tugas bin pack berdasarkan memori
- Tempatkan tugas secara acak
- Mendistribusikan tugas secara merata di seluruh Availability Zone dan kemudian mendistribusikan tugas secara merata di seluruh instance dalam setiap Availability Zone
- Mendistribusikan tugas secara merata di seluruh Availability Zones dan kemudian bin pack task berdasarkan memori dalam setiap Availability Zone
- Mendistribusikan tugas secara merata di seluruh instance dan kemudian tugas bin pack berdasarkan memori

Mendistribusikan tugas secara merata di seluruh Availability Zone

Strategi berikut mendistribusikan tugas secara merata di seluruh Availability Zone.

```
"placementStrategy": [
    {
        "field": "attribute:ecs.availability-zone",
        "type": "spread"
    }
]
```

Mendistribusikan tugas secara merata di semua contoh

Strategi berikut mendistribusikan tugas secara merata di semua instans.

```
"placementStrategy": [
    {
        "field": "instanceId",
        "type": "spread"
    }
```

```
]
```

Tugas bin pack berdasarkan memori

Berikut tugas bin pack strategi berdasarkan memori.

```
"placementStrategy": [
    {
        "field": "memory",
        "type": "binpack"
    }
]
```

Tempatkan tugas secara acak

Strategi berikut menempatkan tugas secara acak.

```
"placementStrategy": [
    {
        "type": "random"
    }
]
```

Mendistribusikan tugas secara merata di seluruh Availability Zone dan kemudian mendistribusikan tugas secara merata di seluruh instance dalam setiap Availability Zone

Strategi berikut mendistribusikan tugas secara merata di seluruh Availability Zone dan kemudian mendistribusikan tugas secara merata di seluruh instans dalam setiap Availability Zone.

```
"placementStrategy": [
    {
        "field": "attribute:ecs.availability-zone",
        "type": "spread"
    },
    {
        "field": "instanceId",
        "type": "spread"
    }
]
```

Mendistribusikan tugas secara merata di seluruh Availability Zones dan kemudian bin pack task berdasarkan memori dalam setiap Availability Zone

Strategi berikut mendistribusikan tugas secara merata di seluruh Availability Zone dan kemudian tugas bin pack berdasarkan memori dalam setiap Availability Zone.

```
"placementStrategy": [
    {
        "field": "attribute:ecs.availability-zone",
        "type": "spread"
    },
    {
        "field": "memory",
        "type": "binpack"
    }
]
```

Mendistribusikan tugas secara merata di seluruh instance dan kemudian tugas bin pack berdasarkan memori

Strategi berikut mendistribusikan tugas secara merata di semua instance dan kemudian mengemas tugas berdasarkan memori dalam setiap instance.

```
"placementStrategy": [
    {
        "field": "instanceId",
        "type": "spread"
    },
    {
        "field": "memory",
        "type": "binpack"
    }
]
```

Tugas Amazon ECS terkait grup

Anda dapat mengidentifikasi serangkaian tugas terkait dan menempatkannya dalam kelompok tugas. Semua tugas dengan nama grup tugas yang sama dianggap sebagai satu rangkaian ketika menggunakan strategi penempatan tugas `spread`. Misalnya, Anda menjalankan aplikasi yang berbeda dalam satu cluster, seperti database dan server web. Untuk memastikan bahwa

basis data Anda seimbang di seluruh Availability Zone, tambahkan basis data ke grup tugas bernama `databases` dan kemudian gunakan strategi penempatan tugas `spread`. Untuk informasi selengkapnya, lihat Gunakan strategi untuk menentukan penempatan tugas Amazon ECS.

Kelompok tugas juga dapat digunakan sebagai kendala penempatan tugas. Saat Anda menentukan grup tugas dalam `memberOf` batasan, tugas hanya dikirim ke instance kontainer yang menjalankan tugas dalam grup tugas yang ditentukan. Sebagai contoh, lihat Contoh kendala penempatan tugas Amazon ECS.

Secara default, tugas mandiri menggunakan nama keluarga definisi tugas (misalnya,`family:my-task-definition`) sebagai nama grup tugas jika nama grup tugas kustom tidak ditentukan. Tugas yang diluncurkan sebagai bagian dari layanan menggunakan nama layanan sebagai nama grup tugas dan tidak dapat diubah.

Persyaratan berikut untuk kelompok tugas berlaku.

- Nama grup tugas harus 255 karakter atau kurang.
- Setiap tugas bisa berada dalam satu grup.
- Setelah meluncurkan tugas, Anda tidak dapat memodifikasi grup tugasnya.

Tentukan instance kontainer mana yang digunakan Amazon ECS untuk tugas

Batasan penempatan tugas adalah aturan tentang instance kontainer yang digunakan Amazon ECS untuk menentukan apakah tugas diizinkan untuk dijalankan pada instance. Setidaknya satu instance kontainer harus cocok dengan kendala. Jika tidak ada instance yang cocok dengan kendala, tugas tetap dalam keadaan. `PENDING` Saat membuat layanan baru atau memperbarui layanan yang sudah ada, Anda dapat menentukan batasan penempatan tugas untuk tugas layanan.

Anda dapat menentukan batasan penempatan tugas dalam definisi layanan, definisi tugas, atau tugas menggunakan parameter. `placementConstraint`

```
"placementConstraint": [
    {
        "expression": "The expression that defines the task placement constraints",
        "type": "The placement constraint type to use"
    }
]
```

Tabel berikut menjelaskan cara menggunakan parameter.

Jenis kendala	Dapat ditentukan kapan	
`distinctInstance` Tempatkan setiap tugas pada instans kontainer yang berbeda. **Important** Kami menyarankan agar pelanggan yang mencari isolasi yang kuat untuk tugas mereka menggunakan Fargate. Fargate menjalankan setiap tugas dalam lingkungan virtualisasi perangkat keras. Ini memastikan bahwa beban kerja kontainer ini tidak berbagi antarmuka jaringan, penyimpanan sementara Fargate, CPU, atau memori dengan tugas lain. Untuk informasi selengkapnya, lihat Gambaran Umum Keamanan AWS Fargate.	• Menjalankan tugas RunTask • Membuat layanan baru CreateService,	
`memberOf`	• Menjalankan tugas RunTask	

Jenis kendala	Dapat ditentukan kapan	
Tempatkan tugas pada instans kontainer yang memenuhi ekspresi.	• Membuat layanan baru CreateService, • Membuat definisi tugas baru RegisterTaskDefinition • Membuat revisi baru dari definisi tugas RegisterTaskDefinition • Memperbarui layanan UpdateService	

Bila menggunakan tipe `memberOf` constraint, Anda dapat membuat ekspresi menggunakan bahasa kueri klaster yang menentukan instance container tempat Amazon ECS dapat menempatkan tugas. Ekspresi adalah cara bagi Anda untuk mengelompokkan instance container Anda berdasarkan atribut. Ekspresi masuk dalam `expression` parameter`placementConstraint`.

Atribut instans wadah Amazon ECS

Anda dapat menambahkan metadata kustom ke instans kontainer Anda, yang dikenal sebagai atribut. Tiap atribut memiliki nama dan nilai string opsional. Anda dapat menggunakan atribut bawaan yang disediakan oleh Amazon ECS atau menentukan atribut khusus.

Bagian berikut berisi contoh built-in, opsional, dan atribut kustom.

Atribut bawaan

Amazon ECS secara otomatis menerapkan atribut berikut ke instance container Anda.

`ecs.ami-id`

ID AMI yang digunakan untuk meluncurkan instans. Nilai contoh untuk atribut ini adalah `ami-1234abcd`.

`ecs.availability-zone`

Availability Zone untuk instans. Nilai contoh untuk atribut ini adalah `us-east-1a`.

`ecs.instance-type`

Jenis instans untuk instans tersebut. Nilai contoh untuk atribut ini adalah `g2.2xlarge`.

`ecs.os-type`

Sistem operasi untuk instans. Nilai yang mungkin untuk atribut ini adalah `linux` dan `windows`.

`ecs.os-family`

Versi sistem operasi untuk contoh.

Untuk contoh Linux, nilai yang valid adalah`LINUX`. Untuk instance Windows, ECS menetapkan nilai dalam format. `WINDOWS_SERVER_<OS_Release>_<FULL or CORE>` Nilai yang valid adalah`WINDOWS_SERVER_2022_FULL`,`WINDOWS_SERVER_2022_CORE`,`WINDOWS_SERVER_20H2_CORE` dan`WINDOWS_SERVER_2016_FULL`.

Ini penting untuk wadah Windows dan Windows containers on AWS Fargate karena versi OS dari setiap wadah Windows harus cocok dengan host. Jika versi Windows dari gambar kontainer berbeda dari host, penampung tidak dimulai. Untuk informasi selengkapnya, lihat Kompatibilitas versi penampung Windows di situs web dokumentasi Microsoft.

Jika klaster Anda menjalankan beberapa versi Windows, Anda dapat memastikan bahwa tugas ditempatkan pada instans EC2 yang berjalan pada versi yang sama dengan menggunakan batasan penempatan:. `memberOf(attribute:ecs.os-family == WINDOWS_SERVER_<OS_Release>_<FULL or CORE>)` Untuk informasi selengkapnya, lihat the section called “Mengambil metadata AMI Amazon ECS yang dioptimalkan”.

`ecs.cpu-architecture`

Arsitektur CPU untuk instans. Nilai contoh untuk atribut ini adalah `x86_64` dan `arm64`.

`ecs.vpc-id`

VPC tempat instans diluncurkan. Nilai contoh untuk atribut ini adalah `vpc-1234abcd`.

`ecs.subnet-id`

Subnet yang digunakan instans. Nilai contoh untuk atribut ini adalah `subnet-1234abcd`.

Atribut opsional

Amazon ECS dapat menambahkan atribut berikut ke instance container Anda.

`ecs.awsvpc-trunk-id`

Jika atribut ini ada, instans memiliki antarmuka jaringan trunk. Untuk informasi selengkapnya, lihat Pembuatan torso antarmuka jaringan elastis.

`ecs.outpost-arn`

Jika atribut ini ada, itu berisi Amazon Resource Name (ARN) dari Outpost. Untuk informasi selengkapnya, lihat the section called “Layanan Kontainer Elastis Amazon aktif AWS Outposts”.

`ecs.capability.external`

Jika atribut ini ada, instans diidentifikasi sebagai instans eksternal. Untuk informasi selengkapnya, lihat Instans eksternal (Amazon ECS Anywhere).

Atribut kustom

Anda dapat menerapkan atribut kustom ke instans kontainer Anda. Misalnya, Anda dapat menentukan atribut dengan nama “stack” dan nilai “prod”.

Saat menentukan atribut khusus, Anda harus mempertimbangkan hal berikut.

- `name` harus berisi antara 1 hingga 128 karakter dan nama boleh berisi huruf (huruf besar dan huruf kecil), angka, tanda hubung, garis bawah, garis miring ke depan, garis miring belakang, atau titik.
- `value` harus berisi antara 1 hingga 128 karakter dan boleh berisi huruf (huruf besar dan huruf kecil), angka, tanda hubung, garis bawah, titik, tanda (@), garis miring ke depan, garis miring belakang, titik dua, atau spasi. Nilai tidak dapat berisi spasi putih utama atau di belakang.

Buat ekspresi untuk menentukan instance kontainer untuk tugas Amazon ECS

Kueri klaster adalah ekspresi yang memungkinkan Anda mengelompokkan objek. Misalnya, Anda dapat mengelompokkan instans kontainer dengan atribut seperti Availability Zone, tipe instans, atau metadata kustom. Untuk informasi selengkapnya, lihat Atribut instans wadah Amazon ECS.

Setelah menetapkan grup instans penampung, Anda dapat menyesuaikan Amazon ECS untuk menempatkan tugas pada instans penampung berdasarkan grup. Untuk informasi selengkapnya, lihat Jalankan aplikasi sebagai tugas Amazon ECS, dan Membuat layanan menggunakan konsol. Anda juga dapat menerapkan filter grup ketika mencantumkan instans kontainer. Untuk informasi selengkapnya, lihat Memfilter berdasarkan atribut menggunakan konsol.

Sintaksis ekspresi

Ekspresi memiliki sintaksis berikut:

```
subject operator [argument]
```

Subjek

Atribut atau bidang yang perlu dievaluasi.

`agentConnected`

Pilih instans kontainer berdasarkan status koneksi agen penampung Amazon ECS mereka. Anda dapat menggunakan filter ini untuk mencari instans dengan agen kontainer yang terputus.

Operasi yang valid: equals (==), not_equals (!=), in, not_in (!in), matches (=~), not_matches (!~)

`agentVersion`

Pilih instans kontainer berdasarkan versi agen penampung Amazon ECS mereka. Anda dapat menggunakan filter ini untuk menemukan instance yang menjalankan versi lama dari agen penampung Amazon ECS.

Operator yang valid: equals (==), not_equals (!=), greater_than (>), greater_than_equal (>=), less_than (<), less_than_equal (<=)

`attribute:`*`attribute-name`*

Pilih instans kontainer berdasarkan atribut. Untuk informasi selengkapnya, lihat Atribut instans wadah Amazon ECS.

`ec2InstanceId`

Pilih instans kontainer berdasarkan ID instans Amazon EC2 mereka.

Operasi yang valid: equals (==), not_equals (!=), in, not_in (!in), matches (=~), not_matches (!~)

`registeredAt`

Pilih instans kontainer berdasarkan tanggal pendaftaran instans kontainer mereka. Anda dapat menggunakan filter ini untuk menemukan instans yang baru terdaftar atau instans yang sangat usang.

Operasi yang valid: equals (==), not_equals (!=), greater_than (>), greater_than_equal (>=), less_than (<), less_than_equal (<=)

Format tanggal yang valid: 2018-06-18T22:28:28+00:00, 2018-06-18T22:28:28Z, 2018-06-18T22:28:28, 2018-06-18

`runningTasksCount`

Pilih instans kontainer berdasarkan jumlah tugas yang sedang berjalan. Anda dapat menggunakan filter ini untuk menemukan instans yang kosong atau hampir kosong (hanya beberapa tugas yang berjalan pada instans tersebut).

Operasi yang valid: equals (==), not_equals (!=), greater_than (>), greater_than_equal (>=), less_than (<), less_than_equal (<=)

`task:group`

Pilih instans kontainer oleh grup tugas. Untuk informasi selengkapnya, lihat Tugas Amazon ECS terkait grup .

Operator

Operasi perbandingan. Mendukung operasi berikut ini.

Operasi	Deskripsi
==, equals	Kesamaan string
!=, not_equals	Ketidaksamaan string
>, greater_than	Lebih besar dari
>=, greater_than_equal	Lebih besar dari atau sama dengan
<, less_than	Kurang dari
<=, less_than_equal	Kurang dari atau sama dengan
exists	Subjek ada
!exists, not_exists	Subjek tidak ada
in	Nilai dalam daftar argumen
!in, not_in	Nilai tidak dalam daftar argumen
=~, matches	Pola cocok

Operasi	Deskripsi
!~, not_matches	Pola tidak cocok

> **Note**
>
> Sebuah ekspresi tunggal tidak dapat berisi tanda kurung. Namun, tanda kurung dapat digunakan untuk menunjukkan prioritas dalam ekspresi majemuk.

Pendapat

Di sebagian besar operasi, argumen adalah nilai literal.

Operasi `in` dan `not_in` menganggap daftar argumen sebagai argumen. Anda menentukan daftar argumen sebagai berikut:

```
[argument1, argument2, ..., argumentN]
```

Operasi matches dan not_matches menerima argumen yang sesuai dengan sintaksis ekspresi reguler Java. Untuk informasi selengkapnya, lihat java.util.regex.Pattern.

Ekspresi majemuk

Anda dapat menggabungkan ekspresi menggunakan operasi Boolean berikut:

- &&, dan
- ||, atau
- !, tidak

Anda dapat menentukan prioritas menggunakan tanda kurung:

```
(expression1 or expression2) and expression3
```

Contoh ekspresi

Berikut ini adalah contoh-contoh ekspresi.

Contoh: String Sama Dengan

Ekspresi berikut memilih instans dengan tipe instans yang ditentukan.

```
attribute:ecs.instance-type == t2.small
```

Contoh: Daftar Argumen

Ekspresi berikut memilih instans di Availability Zone us-east-1a atau us-east-1b.

```
attribute:ecs.availability-zone in [us-east-1a, us-east-1b]
```

Contoh: Ekspresi Majemuk

Ekspresi berikut memilih instance G2 yang tidak berada di Availability Zone us-east-1d.

```
attribute:ecs.instance-type =~ g2.* and attribute:ecs.availability-zone != us-east-1d
```

Contoh: Afinitas Tugas

Ekspresi berikut memilih instans yang meng-host tugas di grup`service:production`.

```
task:group == service:production
```

Contoh: Afinitas Terbalik Tugas

Ekspresi berikut memilih instance yang tidak menghosting tugas dalam grup database.

```
not(task:group == database)
```

Contoh: Menjalankan penghitungan tugas

Ekspresi berikut memilih instans yang hanya menjalankan satu tugas.

```
runningTasksCount == 1
```

Contoh: Amazon ECS versi agen kontainer

Ekspresi berikut memilih instans yang menjalankan versi agen kontainer di bawah 1.14.5.

```
agentVersion < 1.14.5
```

Contoh: Waktu pendaftaran instans

Ekspresi berikut memilih instans yang terdaftar sebelum 13 Februari 2018.

```
registeredAt < 2018-02-13
```

Contoh: ID instans Amazon EC2

Ekspresi berikut memilih instance dengan ID instans Amazon EC2 berikut.

```
ec2InstanceId in ['i-abcd1234', 'i-wxyx7890']
```

Contoh kendala penempatan tugas Amazon ECS

Berikut ini adalah contoh kendala penempatan tugas.

Contoh ini menggunakan `memberOf` kendala untuk menempatkan tugas pada instance t2. Ini dapat ditentukan dengan tindakan berikut: CreateService, UpdateService, RegisterTaskDefinition, dan RunTask.

```
"placementConstraints": [
    {
        "expression": "attribute:ecs.instance-type =~ t2.*",
        "type": "memberOf"
    }
]
```

Contoh menggunakan `memberOf` kendala untuk menempatkan tugas replika pada instance dengan tugas di grup tugas layanan daemon, dengan menghormati strategi penempatan `daemon-service` tugas apa pun yang juga ditentukan. Batasan ini memastikan bahwa tugas layanan daemon ditempatkan pada instans EC2 sebelum tugas layanan replika.

Ganti `daemon-service` dengan nama layanan daemon.

```
"placementConstraints": [
    {
        "expression": "task:group == service:daemon-service",
        "type": "memberOf"
    }
]
```

Contoh tersebut menggunakan kendala `memberOf` untuk menempatkan tugas pada instans dengan tugas-tugas lain di kelompok tugas `databases`, perihal setiap strategi penempatan tugas yang juga

ditentukan. Untuk informasi selengkapnya tentang grup tugas, lihat Tugas Amazon ECS terkait grup . Ini dapat ditentukan dengan tindakan berikut: CreateService, UpdateService, RegisterTaskDefinition, dan RunTask.

```
"placementConstraints": [
    {
        "expression": "task:group == databases",
        "type": "memberOf"
    }
]
```

Kendala `distinctInstance` menempatkan setiap tugas dalam grup pada instans yang berbeda. Hal ini dapat ditentukan dengan tindakan berikut: CreateService, UpdateService, dan RunTask

```
"placementConstraints": [
    {
        "type": "distinctInstance"
    }
]
```

Tugas mandiri Amazon ECS

Anda dapat menjalankan aplikasi Anda sebagai tugas ketika Anda memiliki aplikasi yang melakukan beberapa pekerjaan, dan kemudian berhenti, misalnya proses batch. Jika Anda ingin menjalankan tugas satu kali, Anda dapat menggunakan konsol, API AWS CLI, atau SDK.

Jika Anda perlu menjalankan aplikasi berdasarkan tarif, berbasis cron, atau jadwal satu kali, Anda dapat membuat jadwal menggunakan Scheduler. EventBridge

Alur kerja tugas

Saat Anda meluncurkan tugas Amazon ECS (tugas mandiri atau oleh layanan Amazon ECS), tugas dibuat dan awalnya dipindahkan ke status. `PROVISIONING` Saat tugas dalam `PROVISIONING` status, baik tugas maupun kontainer tidak ada karena Amazon ECS perlu menemukan kapasitas komputasi untuk menempatkan tugas.

Amazon ECS memilih kapasitas komputasi yang sesuai untuk tugas Anda berdasarkan jenis peluncuran atau konfigurasi penyedia kapasitas. Anda dapat menggunakan penyedia kapasitas dan strategi penyedia kapasitas dengan jenis peluncuran Fargate dan Amazon EC2. Dengan Fargate, Anda tidak perlu memikirkan penyediaan, konfigurasi, dan penskalaan kapasitas cluster Anda.

Fargate menangani semua manajemen infrastruktur untuk tugas Anda. Untuk jenis peluncuran EC2, Anda dapat mengelola kapasitas klaster dengan mendaftarkan instans Amazon EC2 ke klaster, atau Anda dapat menggunakan penskalaan otomatis cluster untuk menyederhanakan manajemen kapasitas komputasi. Penskalaan otomatis cluster menangani penskalaan kapasitas klaster secara dinamis, sehingga Anda dapat fokus menjalankan tugas. Amazon ECS menentukan tempat untuk menempatkan tugas berdasarkan persyaratan yang Anda tentukan dalam definisi tugas, seperti CPU dan memori, serta batasan dan strategi penempatan Anda. Untuk informasi selengkapnya, lihat, Bagaimana Amazon ECS menempatkan tugas pada instans kontainer.

Setelah menemukan kapasitas untuk menempatkan tugas Anda, Amazon ECS menyediakan lampiran yang diperlukan (misalnya, Antarmuka Jaringan Elastis (ENI) untuk tugas dalam mode). `awsvpc` Ini menggunakan agen kontainer Amazon ECS untuk menarik gambar kontainer Anda, dan kemudian memulai wadah Anda. Setelah penyediaan selesai dan kontainer yang relevan diluncurkan, Amazon ECS memindahkan tugas ke status. `RUNNING`

Topik

Praktik terbaik untuk meningkatkan waktu peluncuran tugas Amazon ECS

Untuk mempercepat peluncuran tugas Anda, pertimbangkan rekomendasi berikut.

- Cache gambar wadah dan instance binpack

  Jika Anda menggunakan tipe peluncuran EC2, Anda dapat mengonfigurasi perilaku tarik agen penampung Amazon ECS ke`ECS_IMAGE_PULL_BEHAVIOR`:. `prefer-cached` Gambar ditarik dari jarak jauh jika tidak ada gambar yang di-cache. Jika tidak, citra cache pada instans digunakan. Pembersihan gambar otomatis dimatikan untuk wadah untuk memastikan bahwa gambar yang di-cache tidak dihapus. Ini mengurangi waktu tarik gambar untuk peluncuran berikutnya. Efek caching bahkan lebih besar ketika Anda memiliki kepadatan tugas yang tinggi dalam instance container Anda, yang dapat Anda konfigurasikan menggunakan strategi `binpack` penempatan. Caching gambar kontainer sangat bermanfaat untuk beban kerja berbasis windows yang biasanya memiliki ukuran gambar kontainer besar (puluhan GB). Saat menggunakan strategi `binpack` penempatan, Anda juga dapat mempertimbangkan untuk menggunakan trunking Elastic Network

Interface (ENI) untuk menempatkan lebih banyak tugas dengan mode `awsvpc` jaringan pada setiap instance container. Trunking ENI meningkatkan jumlah tugas yang dapat Anda jalankan dalam mode. `awsvpc` Misalnya, instance c5.large yang mungkin mendukung menjalankan hanya 2 tugas secara bersamaan, dapat menjalankan hingga 10 tugas dengan trunking ENI.

- Pilih mode jaringan yang optimal

  Meskipun ada banyak contoh di mana mode `awsvpc` jaringan ideal, mode jaringan ini secara inheren dapat meningkatkan latensi peluncuran tugas, karena untuk setiap tugas dalam mode`awsvpc`, alur kerja Amazon ECS perlu menyediakan dan melampirkan ENI dengan menjalankan API Amazon EC2 yang menambahkan overhead beberapa detik ke peluncuran tugas Anda. Sebaliknya, keuntungan utama menggunakan mode `awsvpc` jaringan adalah bahwa setiap tugas memiliki grup keamanan untuk mengizinkan atau menolak lalu lintas. Ini berarti Anda memiliki fleksibilitas yang lebih besar untuk mengontrol komunikasi antara tugas dan layanan pada tingkat yang lebih terperinci. Jika kecepatan penerapan adalah prioritas Anda, Anda dapat mempertimbangkan untuk menggunakan `bridge` mode untuk mempercepat peluncuran tugas. Untuk informasi selengkapnya, lihat the section called “AWSVPC modus”.

- Lacak siklus hidup peluncuran tugas Anda untuk menemukan peluang pengoptimalan

  Seringkali sulit untuk mengetahui jumlah waktu yang dibutuhkan aplikasi Anda untuk memulai. Meluncurkan image kontainer Anda, menjalankan skrip start-up, dan konfigurasi lainnya selama start-up aplikasi dapat memakan waktu yang mengejutkan. Anda dapat menggunakan titik akhir metadata Tugas untuk memposting metrik untuk melacak waktu mulai aplikasi dari saat aplikasi Anda siap `ContainerStartTime` untuk melayani lalu lintas. Dengan data ini, Anda dapat memahami bagaimana aplikasi Anda berkontribusi terhadap total waktu peluncuran, dan menemukan area di mana Anda dapat mengurangi overhead khusus aplikasi yang tidak perlu dan mengoptimalkan gambar kontainer Anda. Untuk informasi selengkapnya, lihat Ambil metadata Amazon ECS.

- Pilih jenis instans yang optimal (saat menggunakan tipe peluncuran EC2)

  Memilih jenis instans yang benar didasarkan pada reservasi sumber daya (misalnya, CPU, memori) yang Anda konfigurasikan pada tugas Anda. Oleh karena itu, saat mengukur instance, Anda dapat menghitung berapa banyak tugas yang dapat ditempatkan pada satu instance. Contoh sederhana dari tugas yang ditempatkan dengan baik, adalah hosting 4 tugas yang membutuhkan 0,5 vCPU dan 2GB reservasi memori dalam instance m5.large (mendukung 2 vCPU dan memori 8 GB). Reservasi definisi tugas ini memanfaatkan sepenuhnya sumber daya instans.

Jalankan aplikasi sebagai tugas Amazon ECS

Anda dapat membuat tugas untuk proses satu kali menggunakan. AWS Management Console

Untuk membuat tugas mandiri ()AWS Management Console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Konsol Amazon ECS memungkinkan Anda membuat tugas mandiri dari halaman detail klaster atau dari daftar revisi definisi tugas. Gunakan langkah-langkah berikut untuk membuat tugas mandiri Anda tergantung pada halaman sumber daya yang Anda pilih.

Untuk memulai layanan dari	Langkah-langkah
halaman detail cluster...	a. Pada halaman Klaster, pilih klaster untuk membuat layanan. b. Dari tab Tugas, pilih Jalankan tugas baru.
halaman revisi definisi tugas...	a. Pada halaman Definisi tugas, pilih keluarga definisi tugas untuk menampilkan revisi untuk keluarga tersebut. b. Pilih revisi yang ingin Anda gunakan. c. Dari menu Deploy, pilih Jalankan tugas.

3. (Opsional) Bagian Konfigurasi komputasi (lanjutan) adalah tempat Anda memilih bagaimana tugas Anda akan didistribusikan. Anda dapat menggunakan strategi penyedia Kapasitas atau tipe Peluncuran. Untuk menggunakan strategi penyedia kapasitas, Anda harus mengonfigurasi penyedia kapasitas Anda di tingkat klaster. Untuk informasi selengkapnya, lihat Penyedia kapasitas Amazon ECS. Jika Anda belum mengonfigurasi klaster untuk menggunakan penyedia kapasitas, gunakan jenis peluncuran sebagai gantinya.

Metode distribusi	Langkah-langkah
Strategi penyedia kapasitas	a. Di bagian Opsi komputasi , pilih Strategi penyedia kapasitas. b. Pilih strategi: • Untuk menggunak an strategi penyedia kapasitas default cluster, pilih Use cluster default. • Jika klaster Anda tidak memiliki strategi penyedia kapasitas default, atau menggunakan strategi khusus, pilih Gunakan strategi penyedia kustom, Tambahkan kapasitas, dan tentukan strategi penyedia kapasitas khusus Anda dengan menentuka n Basis, penyedia Kapasitas, dan Berat. **Note** Untuk menggunakan penyedia kapasitas dalam suatu strategi, penyedia kapasitas harus dikaitkan dengan cluster.

Metode distribusi	Langkah-langkah
	Untuk informasi selengkapnya tentang strategi penyedia kapasitas , lihat Penyedia kapasitas Amazon ECS.
Jenis peluncuran	a. Di bagian Compute options, pilih Launch type. b. Untuk jenis Peluncuran, pilih jenis peluncuran. c. (Opsional) Ketika tipe peluncuran Fargate diwajibkan, untuk Versi platform, tentukan versi platform yang akan digunakan. Jika versi platform tidak ditentuka n, versi LATEST platform akan digunakan.

4. Untuk Jenis aplikasi, pilih Tugas.
5. Untuk definisi Tugas, pilih keluarga definisi tugas dan revisi.

> **Important**
>
> Konsol memvalidasi pilihan untuk memastikan bahwa keluarga dan revisi definisi tugas yang dipilih kompatibel dengan konfigurasi komputasi yang ditentukan.

6. Untuk tugas yang diinginkan, masukkan jumlah tugas yang akan diluncurkan.
7. Jika definisi tugas Anda menggunakan mode `awsvpc` jaringan, perluas Jaringan. Gunakan langkah-langkah berikut untuk menentukan konfigurasi kustom.

    a. Untuk VPC, pilih VPC yang akan digunakan.

b. Untuk Subnet, pilih satu atau beberapa subnet di VPC yang dipertimbangkan oleh penjadwal tugas saat menempatkan tugas Anda.

> **Important**
>
> Hanya subnet pribadi yang didukung untuk mode jaringan `awsvpc`. Tugas tidak menerima alamat IP publik. Oleh karena itu, gateway NAT diperlukan untuk akses internet keluar, dan lalu lintas internet masuk diarahkan melalui penyeimbang beban.

c. Untuk grup Keamanan, Anda dapat memilih grup keamanan yang ada atau membuat yang baru. Untuk menggunakan grup keamanan yang ada, pilih grup keamanan dan lanjutkan ke langkah berikutnya. Untuk membuat grup keamanan baru, pilih Buat grup keamanan baru. Anda harus menentukan nama grup keamanan, deskripsi, dan kemudian tambahkan satu atau beberapa aturan masuk untuk grup keamanan.

d. Untuk IP Publik, pilih apakah akan menetapkan alamat IP publik secara otomatis ke antarmuka jaringan elastis (ENI) tugas.

   AWS Fargate tugas dapat diberikan alamat IP publik ketika dijalankan di subnet publik sehingga mereka memiliki rute ke internet. Untuk informasi selengkapnya, lihat Jaringan tugas Fargate di Panduan Pengguna Layanan Kontainer Elastis Amazon untuk. AWS Fargate

8. Jika tugas Anda menggunakan volume data yang kompatibel dengan konfigurasi saat penerapan, Anda dapat mengonfigurasi volume dengan memperluas Volume.

   Nama volume dan jenis volume dikonfigurasi saat membuat revisi definisi tugas dan tidak dapat diubah saat Anda menjalankan tugas mandiri. Untuk memperbarui nama dan jenis volume, Anda harus membuat revisi definisi tugas baru dan menjalankan tugas dengan menggunakan revisi baru.

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
Amazon EBS	a. Untuk jenis volume EBS, pilih jenis volume EBS yang ingin Anda lampirkan ke tugas Anda.	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	b. Untuk Ukuran (GiB), masukkan nilai yang valid untuk ukuran volume dalam gibibytes (GiB). Anda dapat menentuka n minimal 1 GiB dan maksimum ukuran volume 16,384 GiB. Nilai ini diperlukan kecuali Anda memberikan ID snapshot. c. Untuk IOPS, masukkan jumlah maksimum operasi input/output (IOPS) yang harus disediakan volume. Nilai ini hanya dapat dikonfigurasi untuk`io1`,`io2`, dan jenis `gp3` volume. d. Untuk Throughput (MIB/ s), masukkan throughpu t yang harus disediakan volume, dalam mebibytes per detik (, atau MIB/ s). MiBps Nilai ini hanya dapat dikonfigurasi untuk jenis `gp3` volume. e. Untuk ID Snapshot, pilih snapshot volume Amazon EBS yang ada atau masukkan ARN snapshot jika Anda ingin membuat volume dari snapshot. Anda juga dapat membuat	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	volume baru yang kosong dengan tidak memilih atau memasukkan ID snapshot. f. Untuk kebijakan Penghentian, batalkan centang kotak jika Anda ingin volume yang dikonfigurasi untuk lampiran ke tugas dipertahankan setelah tugas dihentikan. Secara default, volume EBS yang dilampirkan ke tugas dihapus saat tugas dihentikan. g. Untuk tipe sistem File, pilih jenis sistem file yang akan digunakan untuk penyimpanan data dan pengambilan pada volume. Anda dapat memilih default sistem operasi atau jenis sistem file tertentu. Default untuk Linux adalah`XFS`. Untuk volume yang dibuat dari snapshot, Anda harus menentukan jenis sistem file yang sama dengan volume yang digunakan saat snapshot dibuat. Jika ada ketidakcocokan tipe	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	sistem file, tugas akan gagal dimulai. h. Untuk peran Infrastruktur, pilih peran IAM dengan izin yang diperlukan yang memungkinkan Amazon ECS mengelola volume Amazon EBS untuk tugas. Anda dapat melampirkan kebijakan `AmazonECSInfrastructureRolePolicyForVolumes` terkelola ke peran, atau Anda dapat menggunakan kebijakan sebagai panduan untuk membuat dan melampirkan kebijakan Anda sendiri dengan izin yang memenuhi kebutuhan spesifik Anda. Untuk informasi selengkapnya tentang izin yang diperlukan, lihatPeran IAM infrastruktur Amazon ECS. i. Untuk Enkripsi, pilih Default jika Anda ingin menggunakan enkripsi Amazon EBS dengan pengaturan default. Jika akun Anda memiliki Enkripsi yang dikonfigu	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	rasi secara default, volume akan dienkripsi dengan kunci AWS Key Management Service (AWS KMS) yang ditentukan dalam pengaturan. Jika Anda memilih Default dan enkripsi default Amazon EBS tidak diaktifkan, volume tidak akan dienkripsi. Jika Anda memilih Kustom, Anda dapat menentukan AWS KMS key pilihan Anda untuk enkripsi volume. Jika Anda memilih None, volume tidak akan dienkripsi kecuali Anda memiliki enkripsi secara default dikonfigurasi, atau jika Anda membuat volume dari snapshot terenkripsi. j. Jika Anda telah memilih Kustom untuk Enkripsi, Anda harus menentukan AWS KMS key yang ingin Anda gunakan. Untuk kunci KMS, pilih AWS KMS key atau	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	masukkan tombol ARN. Jika Anda memilih untuk mengenkripsi volume dengan menggunakan kunci terkelola pelanggan simetris, pastikan Anda memiliki izin yang tepat yang ditentukan dalam kebijakan Anda. AWS KMS key Untuk informasi selengkapnya, lihat Enkripsi data untuk volume Amazon EBS. k. (Opsional) Di bawah Tag, Anda dapat menambahk an tag ke volume Amazon EBS Anda dengan menyebarkan tag dari definisi tugas atau dengan memberikan tag Anda sendiri. Jika Anda ingin menyebarkan tag dari definisi tugas, pilih Definisi tugas untuk menyebarkan tag dari. Jika Anda memilih Jangan menyebarkan, atau jika Anda tidak memilih nilai, tag tidak disebarkan. Jika Anda ingin memberikan tag Anda	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	sendiri, pilih Tambah tag dan kemudian berikan kunci dan nilai untuk setiap tag yang Anda tambahkan. Untuk informasi selengkapnya tentang menandai volume Amazon EBS, lihat Menandai volume Amazon EBS.	

9. (Opsional) Untuk menggunakan strategi penempatan tugas selain default, perluas Penempatan Tugas, lalu pilih dari opsi berikut.

   Untuk informasi selengkapnya, lihat Bagaimana Amazon ECS menempatkan tugas pada instans kontainer.

   - AZ Balanced Spread — Mendistribusikan tugas di seluruh Availability Zone dan di seluruh instance container di Availability Zone.
   - AZ Balanced BinPack — Mendistribusikan tugas di seluruh Availability Zone dan di seluruh instans kontainer dengan memori yang paling sedikit tersedia.
   - BinPack— Mendistribusikan tugas berdasarkan jumlah CPU atau memori yang paling sedikit tersedia.
   - Satu Tugas Per Host - Tempatkan, paling banyak, satu tugas dari layanan pada setiap instance kontainer.
   - Kustom - Tentukan strategi penempatan tugas Anda sendiri.

   Jika Anda memilih Kustom, tentukan algoritme untuk menempatkan tugas dan aturan yang dipertimbangkan selama penempatan tugas.

   - Di bawah Strategi, untuk Jenis dan Bidang, pilih algoritma dan entitas yang akan digunakan untuk algoritme.

     Anda dapat memasukkan maksimal 5 strategi.

- Di bawah Constraint, untuk Type dan Expression, pilih aturan dan atribut untuk kendala.

  Misalnya, untuk menyetel batasan untuk menempatkan tugas pada instance T2, untuk Ekspresi, masukkan atribut:ecs.instance-type =~ t2. *.

  Anda dapat memasukkan maksimal 10 kendala.

10. (Opsional) Untuk mengganti peran IAM tugas, atau peran eksekusi tugas yang ditentukan dalam definisi tugas Anda, perluas penggantian Tugas, lalu selesaikan langkah-langkah berikut:

    a. Untuk peran Tugas, pilih peran IAM untuk tugas ini. Untuk informasi selengkapnya, lihat Tugas peran IAM.

       Hanya peran dengan hubungan kepercayaan `ecs-tasks.amazonaws.com` yang ditampilkan. Untuk petunjuk tentang cara membuat peran IAM untuk tugas Anda, lihatMembuat peran dan kebijakan IAM untuk tugas Anda.

    b. Untuk peran eksekusi tugas, pilih peran eksekusi tugas. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

11. (Opsional) Untuk mengganti perintah kontainer dan variabel lingkungan, perluas Container Overrides, lalu perluas container.

    - Untuk mengirim perintah ke wadah selain perintah definisi tugas, untuk penggantian Command, masukkan perintah Docker.

      Untuk informasi selengkapnya tentang perintah Docker run, lihat referensi Docker Run di Manual Referensi Docker.
    - Untuk menambahkan variabel lingkungan, pilih Tambahkan Variabel Lingkungan. Untuk Kunci, masukkan nama variabel lingkungan Anda. Untuk Nilai, masukkan nilai string untuk nilai lingkungan Anda (tanpa tanda kutip ganda (" ")).

      AWS mengelilingi string dengan tanda kutip ganda (” “) dan meneruskan string ke wadah dalam format berikut:

      ```
      MY_ENV_VAR="This variable contains a string."
      ```

12. (Opsional) Untuk membantu mengidentifikasi tugas Anda, perluas bagian Tag, lalu konfigurasikan tag Anda.

    Agar Amazon ECS secara otomatis menandai semua tugas yang baru diluncurkan dengan nama cluster dan tag definisi tugas, pilih Aktifkan tag terkelola Amazon ECS, lalu pilih Definisi tugas.

Menambah atau menghapus tanda.

- [Tambahkan tag] Pilih Tambah tag, lalu lakukan hal berikut:
  - Untuk Kunci, masukkan nama kunci.
  - Untuk Nilai, masukkan nilai kunci.
- [Menghapus tanda] Di samping tanda, pilih Hapus tanda.

13. Pilih Buat.

Jalankan tugas Amazon ECS pada jadwal menggunakan Amazon Scheduler EventBridge

EventBridge Scheduler adalah penjadwal tanpa server yang memungkinkan Anda membuat, menjalankan, dan mengelola tugas dari satu layanan terpusat yang dikelola. Ini menyediakan fungsionalitas penjadwalan satu kali dan berulang yang independen dari bus dan aturan acara. EventBridge Scheduler sangat dapat disesuaikan, dan menawarkan skalabilitas yang ditingkatkan dibandingkan aturan EventBridge terjadwal, dengan serangkaian operasi dan layanan API target yang lebih luas. AWS EventBridge Scheduler menyediakan jadwal berikut yang dapat Anda konfigurasi untuk tugas Anda di konsol EventBridge Scheduler:

- Berbasis tarif
- Berbasis cron

  Anda dapat mengonfigurasi jadwal berbasis cron di zona waktu mana pun.
- Jadwal satu kali

  Anda dapat mengonfigurasi jadwal satu kali di zona waktu mana pun.

Anda dapat menjadwalkan Amazon ECS Anda menggunakan Amazon EventBridge Scheduler.

Meskipun Anda dapat membuat tugas terjadwal di konsol Amazon ECS, saat ini konsol EventBridge Scheduler menyediakan lebih banyak fungsionalitas.

Selesaikan langkah-langkah berikut sebelum Anda menjadwalkan tugas:

1. Gunakan konsol VPC untuk mendapatkan ID subnet tempat tugas dijalankan dan ID grup keamanan untuk subnet. Untuk informasi selengkapnya, lihat Melihat subnet Anda, dan Lihat grup keamanan Anda di Panduan Pengguna Amazon VPC.
2. Konfigurasikan peran eksekusi EventBridge Scheduler. Untuk informasi selengkapnya, lihat Mengatur peran eksekusi di Panduan Pengguna EventBridge Penjadwal Amazon.

Untuk membuat jadwal baru menggunakan konsol

1. Buka konsol Amazon EventBridge Scheduler di https://console.aws.amazon.com/scheduler/home.
2. Pada halaman Jadwal, pilih Buat jadwal.
3. Pada halaman Tentukan detail jadwal, di bagian Nama jadwal dan deskripsi, lakukan hal berikut:
   a. Untuk nama Jadwal, masukkan nama untuk jadwal Anda. Misalnya, **`MyTestSchedule`**.
   b. (Opsional) Untuk Deskripsi, masukkan deskripsi untuk jadwal Anda. Misalnya, **`TestSchedule`**.
   c. Untuk grup Jadwal, pilih grup jadwal dari daftar dropdown. Jika Anda tidak memiliki grup, pilih default. Untuk membuat grup jadwal, pilih buat jadwal Anda sendiri.

      Anda menggunakan grup jadwal untuk menambahkan tag ke grup jadwal.
4. Pilih opsi jadwal Anda.

Kejadian	Lakukan ini...
Jadwal satu kali Jadwal satu kali memanggil target hanya sekali pada tanggal dan waktu yang Anda tentukan.	Untuk tanggal dan waktu, lakukan hal berikut: • Masukkan tanggal yang valid dalam `YYYY/MM/DD` format. • Masukkan stempel waktu dalam format 24 jam`hh:mm`. • Untuk Timezone, pilih zona waktu.

Kejadian	Lakukan ini...
Jadwal berulang Jadwal berulang memanggil target pada tingkat yang Anda tentukan menggunak an cron ekspresi atau ekspresi tingkat.	a. Untuk jenis Jadwal, lakukan salah satu hal berikut: • Untuk menggunakan ekspresi cron untuk menentukan jadwal, pilih Jadwal berbasis Cron dan masukkan ekspresi cron. • Untuk menggunakan ekspresi tingkat untuk menentukan jadwal, pilih Jadwal berbasis tingkat dan masukkan ekspresi laju. Untuk informasi selengkapnya tentang ekspresi cron dan rate, lihat Menjadwalkan jenis pada EventBridge Scheduler di Panduan Pengguna EventBridge Penjadwal Amazon. b. Untuk jendela waktu Fleksibel, pilih Nonaktif untuk mematikan opsi, atau pilih salah satu jendela waktu yang telah ditentukan sebelumny a. Misalnya, jika Anda memilih 15 menit dan Anda menetapkan jadwal berulang untuk memanggil

Kejadian	Lakukan ini...
	targetnya setiap jam sekali, jadwal berjalan dalam 15 menit setelah dimulainya setiap jam.

5. (Opsional) Jika Anda memilih Jadwal berulang pada langkah sebelumnya, di bagian Jangka Waktu, lakukan hal berikut:

   a. Untuk Timezone, pilih zona waktu.

   b. Untuk Tanggal dan waktu mulai, masukkan tanggal yang valid dalam `YYYY/MM/DD` format, lalu tentukan stempel waktu dalam format 24 jam`hh:mm`.

   c. Untuk Tanggal dan waktu berakhir, masukkan tanggal yang valid dalam `YYYY/MM/DD` format, lalu tentukan stempel waktu dalam format 24 jam`hh:mm`.

6. Pilih Berikutnya.

7. Pada halaman Pilih target, lakukan hal berikut:

   a. Pilih Semua API, lalu di kotak pencarian masukkan ECS.

   b. Pilih Amazon ECS.

   c. Di kotak pencarian, masukkan RunTask, lalu pilih RunTask.

   d. Untuk cluster ECS, pilih cluster.

   e. Untuk tugas ECS, pilih definisi tugas yang akan digunakan untuk tugas tersebut.

   f. Untuk menggunakan tipe peluncuran, perluas opsi Komputasi, lalu pilih Jenis peluncuran. Kemudian, pilih jenis peluncuran.

      Ketika jenis peluncuran Fargate ditentukan, untuk versi Platform, masukkan versi platform yang akan digunakan. Jika tidak ada platform yang ditentukan, versi `LATEST` platform digunakan.

   g. Untuk Subnet, masukkan ID subnet untuk menjalankan tugas.

   h. Untuk grup Keamanan, masukkan ID grup keamanan untuk subnet.

   i. (Opsional) Untuk menggunakan strategi penempatan tugas selain default, perluas batasan Penempatan, lalu masukkan batasan.

      Untuk informasi selengkapnya, lihat Bagaimana Amazon ECS menempatkan tugas pada instans kontainer.

j. (Opsional) Untuk membantu mengidentifikasi tugas Anda, di bawah Tag, konfigurasikan tag Anda.

   Agar Amazon ECS secara otomatis menandai semua tugas yang baru diluncurkan dengan tag definisi tugas, pilih Aktifkan tag terkelola Amazon ECS.

8. Pilih Berikutnya.
9. Pada halaman Pengaturan, lakukan hal berikut:

   a. Untuk mengaktifkan jadwal, di bawah Status jadwal, alihkan Aktifkan jadwal.

   b. Untuk mengonfigurasi kebijakan coba lagi untuk jadwal Anda, di bawah Kebijakan Coba lagi dan antrean surat mati (DLQ), lakukan hal berikut:

      - Beralih Coba lagi.
      - Untuk Waktu retensi maksimum acara, masukkan jam maksimum dan min yang harus disimpan oleh EventBridge Scheduler untuk menyimpan acara yang belum diproses.
      - Waktu maksimum adalah 24 jam.
      - Untuk percobaan ulang Maksimum, masukkan jumlah maksimum kali EventBridge Scheduler mencoba ulang jadwal jika target mengembalikan kesalahan.

        Nilai maksimumnya adalah 185 percobaan ulang.

      Dengan kebijakan coba lagi, jika jadwal gagal memanggil targetnya, EventBridge Scheduler menjalankan kembali jadwal. Jika dikonfigurasi, Anda harus mengatur waktu retensi maksimum dan mencoba ulang untuk jadwal.

   c. Pilih tempat EventBridge Scheduler menyimpan acara yang tidak terkirim.

Opsi antrian surat mati (DLQ)	Lakukan ini...
Jangan simpan	Pilih Tidak Ada.
Simpan acara di tempat yang sama Akun AWS di mana Anda membuat jadwal	a. Pilih Pilih antrian Amazon SQS di saya Akun AWS sebagai DLQ.

Opsi antrian surat mati (DLQ)	Lakukan ini...	
	b. Pilih Nama Sumber Daya Amazon (ARN) dari antrian Amazon SQS.	
Simpan acara di tempat yang berbeda Akun AWS dari tempat Anda membuat jadwal	a. Pilih Tentukan antrean Amazon SQS di lain Akun AWS sebagai DLQ. b. Masukkan Nama Sumber Daya Amazon (ARN) dari antrian Amazon SQS.	

d. Untuk menggunakan kunci yang dikelola pelanggan untuk mengenkripsi input target Anda, di bawah Enkripsi, pilih Sesuaikan pengaturan enkripsi (lanjutan).

   Jika Anda memilih opsi ini, masukkan ARN kunci KMS yang ada atau pilih AWS KMS key Buat untuk menavigasi ke AWS KMS konsol. Untuk informasi selengkapnya tentang cara EventBridge Scheduler mengenkripsi data Anda saat istirahat, lihat Enkripsi saat istirahat di Panduan Pengguna EventBridge Penjadwal Amazon.

e. Untuk Izin, pilih Gunakan peran yang ada, lalu pilih peran.

   Agar EventBridge Scheduler membuat peran eksekusi baru untuk Anda, pilih Buat peran baru untuk jadwal ini. Kemudian, masukkan nama untuk nama Peran. Jika Anda memilih opsi ini, EventBridge Scheduler melampirkan izin yang diperlukan untuk target template Anda ke peran.

10. Pilih Berikutnya.
11. Di halaman Tinjau dan buat jadwal, tinjau detail jadwal Anda. Di setiap bagian, pilih Edit untuk kembali ke langkah itu dan mengedit detailnya.
12. Pilih Buat jadwal.

    Anda dapat melihat daftar jadwal baru dan yang sudah ada di halaman Jadwal. Di bawah kolom Status, verifikasi bahwa jadwal baru Anda Diaktifkan.

Langkah selanjutnya

Anda dapat menggunakan konsol EventBridge Scheduler atau AWS CLI untuk mengelola jadwal. Untuk informasi selengkapnya, lihat Mengelola jadwal di Panduan Pengguna EventBridge Penjadwal Amazon.

Hentikan tugas Amazon ECS

Jika Anda tidak perlu lagi menjalankan tugas mandiri, Anda dapat menghentikan tugas tersebut. Konsol Amazon ECS memudahkan untuk menghentikan satu atau lebih tugas.

Jika Anda ingin menghentikan layanan, lihatMenghapus layanan menggunakan konsol.

Untuk menghentikan tugas mandiri ()AWS Management Console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih cluster untuk menavigasi ke halaman detail cluster.
4. Pada halaman detail cluster, pilih tab Tugas.
5. Anda dapat memfilter tugas berdasarkan jenis peluncuran menggunakan daftar jenis peluncuran Filter.

Tugas untuk berhenti	Langkah-langkah
Satu atau lebih	a. Pilih tugas, lalu pilih Berhenti, Berhenti dipilih. b. Pada halaman Stop task confirmation, pilih Stop
Deployment blue/green	**Important** Jika Anda memilih untuk menghenti kan semua tugas menggunakan konsol, Amazon ECS menghentikan semua tugas dan

Tugas untuk berhenti	Langkah-langkah
	tugas mandiri yang merupakan bagian dari layanan. Karena itu, kami sarankan berhati-hati saat menggunakan opsi ini. a. Pilih Stop, Stop all. b. Pada halaman Hentikan konfirmasi tugas, masukkan Hentikan semua tugas, lalu pilih Berhenti.

Layanan-layanan Amazon ECS

Anda dapat menggunakan layanan Amazon ECS untuk menjalankan dan mempertahankan jumlah instans definisi tugas yang ditentukan secara bersamaan di kluster Amazon ECS. Jika salah satu tugas Anda gagal atau berhenti, penjadwal layanan Amazon ECS meluncurkan contoh lain dari definisi tugas Anda untuk menggantikannya. Ini membantu mempertahankan jumlah tugas yang Anda inginkan dalam layanan.

Anda juga dapat menjalankan layanan Anda secara opsional di belakang penyeimbang beban. Penyeimbang beban mendistribusikan lalu lintas ke seluruh tugas yang terkait dengan layanan.

Konsep penjadwal layanan

Kami menyarankan Anda menggunakan penjadwal layanan untuk layanan dan aplikasi stateless yang berjalan lama. Penjadwal layanan memastikan bahwa strategi penjadwalan yang Anda tentukan diikuti dan menjadwal ulang tugas ketika tugas gagal. Misalnya, jika infrastruktur yang mendasarinya gagal, penjadwal layanan menjadwal ulang tugas. Anda dapat menggunakan strategi dan batasan penempatan tugas untuk menyesuaikan cara penjadwal menempatkan dan mengakhiri tugas. Jika tugas dalam layanan berhenti, penjadwal meluncurkan tugas baru untuk menggantikannya. Proses

ini berlanjut hingga layanan Anda mencapai jumlah tugas yang Anda inginkan berdasarkan strategi penjadwalan yang digunakan layanan. Strategi penjadwalan layanan juga disebut sebagai jenis layanan.

Penjadwal layanan juga menggantikan tugas yang ditentukan tidak sehat setelah pemeriksaan kesehatan kontainer atau pemeriksaan kesehatan kelompok sasaran penyeimbang beban gagal. Penggantian ini tergantung pada parameter definisi `maximumPercent` dan `desiredCount` layanan. Jika tugas ditandai tidak sehat, penjadwal layanan akan memulai tugas pengganti terlebih dahulu. Jika tugas penggantian memiliki status kesehatan`HEALTHY`, penjadwal layanan menghentikan tugas yang tidak sehat. Jika tugas penggantian memiliki status kesehatan`UNHEALTHY`, penjadwal akan menghentikan tugas penggantian yang tidak sehat atau tugas tidak sehat yang ada untuk mendapatkan jumlah tugas total yang sama`desiredCount`. Jika `maximumPercent` parameter membatasi penjadwal untuk memulai tugas penggantian terlebih dahulu, penjadwal akan menghentikan tugas yang tidak sehat satu per satu secara acak untuk membebaskan kapasitas, dan kemudian memulai tugas pengganti. Proses start dan stop berlanjut sampai semua tugas yang tidak sehat diganti dengan tugas yang sehat. Setelah semua tugas yang tidak sehat telah diganti dan hanya tugas sehat yang berjalan, jika jumlah tugas total melebihi`desiredCount`, tugas sehat dihentikan secara acak hingga jumlah tugas total sama. `desiredCount` Untuk informasi selengkapnya tentang `maximumPercent` dan`desiredCount`, lihat Parameter definisi layanan.

> **Note**
>
> Perilaku ini tidak berlaku untuk tugas yang dijalankan dan dikelola oleh layanan yang menggunakan jenis penerapan pembaruan bergulir. Selama pembaruan bergulir, penjadwal layanan pertama-tama menghentikan tugas yang tidak sehat dan kemudian memulai tugas penggantian.

Penjadwal layanan menyertakan logika yang membatasi seberapa sering tugas dimulai ulang jika tugas berulang kali gagal dimulai. Jika tugas dihentikan tanpa memasukkan `RUNNING` status, penjadwal layanan mulai memperlambat upaya peluncuran dan mengirimkan pesan acara layanan. Perilaku ini mencegah sumber daya yang tidak perlu digunakan untuk tugas yang gagal sebelum Anda dapat menyelesaikan masalah. Setelah layanan diperbarui, penjadwal layanan melanjutkan perilaku penjadwalan normal. Lihat informasi yang lebih lengkap di Logika throttle layanan Amazon ECS dan Pesan peristiwa layanan.

Ada dua strategi penjadwal layanan yang tersedia:

- `REPLICA`Strategi penjadwalan replika menempatkan dan mempertahankan jumlah tugas yang diinginkan di seluruh cluster Anda. Secara default tugas tersebar di seluruh Availability Zone. Anda dapat menggunakan strategi penempatan tugas dan kendala untuk menyesuaikan keputusan penempatan tugas. Untuk informasi selengkapnya, lihat Replika.
- `DAEMON`—Strategi penjadwalan daemon menerapkan tepat satu tugas pada setiap instance kontainer aktif yang memenuhi semua batasan penempatan tugas yang Anda tentukan di cluster Anda. Saat menggunakan strategi ini, tidak perlu menentukan jumlah tugas yang diinginkan, strategi penempatan tugas, atau menggunakan kebijakan Auto Scaling Layanan. Untuk informasi selengkapnya, lihat Daemon.

> **Note**
>
> Tugas Fargate tidak mendukung strategi `DAEMON` penjadwalan.

Daemon

Strategi penjadwalan daemon men-deploy tepat satu tugas untuk setiap instans kontainer aktif yang memenuhi semua batasan penempatan tugas yang ditentukan di klaster Anda. Penjadwal layanan juga mengevaluasi batasan penempatan tugas untuk menjalankan tugas, dan menghentikan tugas yang tidak memenuhi batasan penempatan. Saat menggunakan strategi ini, Anda tidak perlu menentukan jumlah tugas yang diinginkan, strategi penempatan tugas, atau menggunakan kebijakan Auto Scaling Service.

Amazon ECS menyimpan sumber daya komputasi instans kontainer termasuk CPU, memori, dan antarmuka jaringan untuk tugas daemon. Saat Anda meluncurkan layanan daemon di klaster dengan layanan replika lainnya, Amazon ECS memprioritaskan tugas daemon. Ini berarti bahwa tugas daemon adalah tugas pertama yang diluncurkan pada instance dan tugas terakhir yang harus dihentikan. Strategi ini memastikan bahwa sumber daya tidak digunakan oleh tugas replika yang tertunda dan tersedia untuk tugas daemon.

Penjadwal layanan daemon tidak menempatkan tugas apa pun pada instance yang memiliki status. `DRAINING` Jika instance container bertransisi ke `DRAINING` status, tugas daemon di atasnya dihentikan. Penjadwal layanan memastikan bahwa tugas daemon adalah yang terakhir berhenti setelah semua tugas replika dihentikan. Penjadwal layanan juga memantau saat instans kontainer baru ditambahkan ke klaster Anda dan menambahkan tugas daemon ke dalamnya.

Jika konfigurasi deployment ditentukan, parameter persentase maksimum harus `100`. Nilai default untuk layanan daemon `maximumPercent` adalah 200%. Nilai default untuk layanan daemon untuk `minimumHealthyPercent` adalah 0%.

Anda harus memulai ulang layanan ketika Anda mengubah batasan penempatan untuk layanan daemon. Amazon ECS secara dinamis memperbarui sumber daya yang dicadangkan pada instans yang memenuhi syarat untuk tugas daemon. Untuk instans yang sudah ada, penjadwal mencoba menempatkan tugas pada instans.

Penerapan baru dimulai ketika ada perubahan pada ukuran tugas atau reservasi sumber daya kontainer dalam definisi tugas. Amazon ECS mengambil reservasi CPU dan memori yang diperbarui untuk daemon, dan kemudian memblokir kapasitas itu untuk tugas daemon.

Jika sumber daya tidak mencukupi untuk salah satu kasus di atas, hal-hal berikut akan terjadi:

- Penempatan tugas gagal.
- Sebuah CloudWatch peristiwa dihasilkan.
- Amazon ECS terus mencoba dan menjadwalkan tugas pada instance dengan menunggu sumber daya tersedia.
- Amazon ECS membebaskan instans cadangan yang tidak lagi memenuhi kriteria batasan penempatan dan menghentikan tugas daemon terkait.

Strategi penjadwalan daemon dapat digunakan dalam kasus-kasus berikut:

- Menjalankan kontainer aplikasi
- Menjalankan kontainer dukungan untuk tugas pencatatan, pemantauan dan pelacakan

Tugas yang menggunakan tipe peluncuran Fargate atau tipe pengontrol `EXTERNAL` penerapan `CODE_DEPLOY` atau tidak mendukung strategi penjadwalan daemon.

Saat penjadwal layanan berhenti menjalankan tugas, penjadwal layanan mencoba untuk menjaga keseimbangan di seluruh Availability Zone di klaster Anda. Penjadwal menggunakan logika berikut:

- Jika strategi penempatan ditentukan, gunakan strategi tersebut untuk memilih tugas mana yang akan diakhiri. Misalnya, jika layanan memiliki strategi penyebaran Availability Zone yang ditentukan, tugas dipilih yang meninggalkan tugas yang tersisa dengan spread terbaik.
- Jika tidak ada strategi penempatan yang ditentukan, gunakan logika berikut untuk menjaga keseimbangan di seluruh Availability Zone di klaster Anda:

- Urutkan instance kontainer yang valid. Berikan prioritas pada instance yang memiliki jumlah tugas berjalan terbesar untuk layanan ini di Availability Zone masing-masing. Misalnya, jika zona A memiliki satu tugas layanan yang berjalan dan zona B dan C masing-masing memiliki dua tugas layanan yang berjalan, instance kontainer di zona B atau C dianggap optimal untuk penghentian.
- Hentikan tugas pada instance container di Availability Zone optimal berdasarkan langkah sebelumnya. Mendukung instance kontainer dengan jumlah tugas yang berjalan terbesar untuk layanan ini.

Replika

Strategi penjadwalan replika menempatkan dan mempertahankan jumlah tugas yang diinginkan di klaster Anda.

Untuk layanan yang menjalankan tugas di Fargate, saat penjadwal layanan meluncurkan tugas baru atau berhenti menjalankan tugas, penjadwal layanan menggunakan upaya terbaik untuk menjaga keseimbangan di seluruh Availability Zone. Anda tidak perlu menentukan strategi penempatan tugas atau kendala.

Saat membuat layanan yang menjalankan tugas pada instans EC2, Anda dapat menentukan strategi dan batasan penempatan tugas secara opsional untuk menyesuaikan keputusan penempatan tugas. Jika tidak ada strategi penempatan tugas atau kendala yang ditentukan, maka secara default penjadwal layanan menyebarkan tugas di seluruh Availability Zones. Penjadwal layanan menggunakan logika berikut:

- Menentukan instance kontainer mana di klaster Anda yang dapat mendukung definisi tugas layanan Anda (misalnya, atribut CPU, memori, port, dan instance container yang diperlukan).
- Menentukan instance kontainer mana yang memenuhi batasan penempatan apa pun yang ditentukan untuk layanan.
- Bila Anda memiliki layanan replika yang bergantung pada layanan daemon (misalnya, tugas router log daemon yang perlu dijalankan sebelum tugas dapat menggunakan logging), buat batasan penempatan tugas yang memastikan bahwa tugas layanan daemon ditempatkan pada instans EC2 sebelum tugas layanan replika. Untuk informasi selengkapnya, lihat Contoh kendala penempatan tugas Amazon ECS.
- Ketika ada strategi penempatan yang ditentukan, gunakan strategi itu untuk memilih instance dari kandidat yang tersisa.

- Jika tidak ada strategi penempatan yang ditentukan, gunakan logika berikut untuk menyeimbangkan tugas di seluruh Availability Zone di klaster Anda:
  - Mengurutkan instance kontainer yang valid. Memberikan prioritas pada instance yang memiliki jumlah tugas yang berjalan paling sedikit untuk layanan ini di Availability Zone masing-masing. Misalnya, jika zona A memiliki satu tugas layanan berjalan, sedangkan zona B dan C masing-masing tidak memliki tugas layanan berjalan, maka instans kontainer yang valid, baik di zona B ataupun C dianggap optimal untuk penempatan.
  - Menempatkan tugas layanan baru pada instance kontainer yang valid di Availability Zone yang optimal berdasarkan langkah sebelumnya. Menyukai instance kontainer dengan jumlah tugas yang berjalan paling sedikit untuk layanan ini.

Konsep layanan tambahan

- Anda dapat secara opsional menjalankan layanan di belakang penyeimbang beban. Untuk informasi selengkapnya, lihat Mendistribusikan lalu lintas layanan Amazon ECS menggunakan load balancing.
- Anda dapat menentukan konfigurasi deployment untuk layanan Anda. Penerapan dimulai dengan memperbarui definisi tugas dari suatu layanan. Selama deloyment, penjadwal layanan menggunakan parameter persentase minimum yang sehat dan persentase maksimum untuk menentukan strategi deployment. Untuk informasi selengkapnya, lihat Parameter ketentuan layanan.
- Anda dapat mengonfigurasi layanan Anda secara opsional untuk menggunakan penemuan layanan Amazon ECS. Penemuan layanan menggunakan API AWS Cloud Map autonaming untuk mengelola entri DNS untuk tugas layanan Anda. Ini membuat mereka dapat ditemukan dari dalam VPC Anda. Untuk informasi selengkapnya, lihat Penemuan Layanan .
- Saat Anda menghapus layanan, jika masih ada tugas yang berjalan yang memerlukan pembersihan, layanan akan berpindah dari status `ACTIVE` ke `DRAINING` status, dan layanan tidak lagi terlihat di konsol atau dalam operasi `ListServices` API. Setelah semua tugas bertransisi ke `STOPPED` status `STOPPING` atau, layanan berpindah dari `INACTIVE` status `DRAINING` ke status. Anda dapat melihat layanan dalam `INACTIVE` status `DRAINING` atau menggunakan operasi `DescribeServices` API. Namun, di masa depan, `INACTIVE` layanan mungkin dibersihkan dan dibersihkan dari pencatatan Amazon ECS, dan `DescribeServices` panggilan pada layanan tersebut mengembalikan kesalahan. `ServiceNotFoundException`
- Waktu pemanggangan adalah periode waktu setelah versi layanan baru telah ditingkatkan dan versi layanan lama telah ditingkatkan, di mana Amazon ECS terus memantau alarm yang terkait

dengan penyebaran. Amazon ECS menghitung periode waktu ini berdasarkan konfigurasi alarm yang terkait dengan penerapan.

Waktu pemanggangan hanya berlaku saat Anda menggunakan CloudWatch alarm untuk mendeteksi kegagalan penerapan. Untuk informasi selengkapnya, lihat the section called “Metode deteksi kegagalan”.

Praktik Terbaik untuk meningkatkan waktu penerapan Amazon ECS

Saat Anda menggunakan opsi penerapan pembaruan bergulir, Anda dapat memodifikasi parameter untuk mempercepat penerapan Anda.

Saat Anda memilih jenis penerapan pembaruan bergulir, penjadwal layanan Amazon ECS menggantikan tugas yang sedang berjalan dengan tugas baru setiap kali penerapan layanan baru dimulai. Konfigurasi penerapan menentukan jumlah tugas tertentu yang ditambahkan atau dihapus Amazon ECS dari layanan selama pembaruan bergulir. Berikut ini adalah ikhtisar proses penyebaran:

1. Penjadwal memulai aplikasi Anda.
2. Penjadwal kemudian memutuskan apakah aplikasi Anda siap untuk lalu lintas.
3. Saat Anda menurunkan atau membuat versi baru aplikasi, penjadwal memutuskan apakah aplikasi Anda aman untuk dihentikan. Pada saat yang sama, itu harus menjaga ketersediaan aplikasi selama penyebaran pembaruan bergulir.

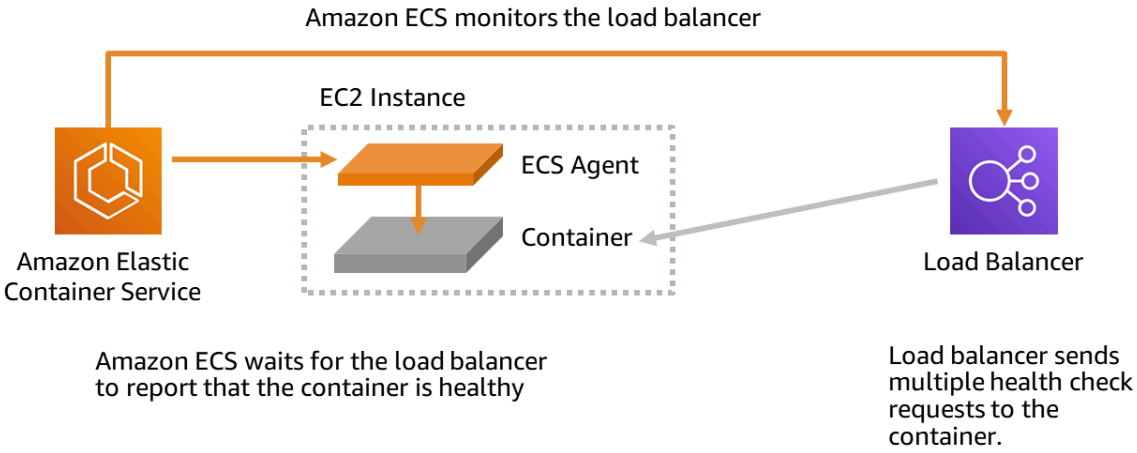
Strategi menjaga ketersediaan tugas dapat menyebabkan penerapan memakan waktu lebih lama dari yang Anda harapkan.

Untuk mempercepat waktu penerapan, ubah opsi penyeimbang beban default, agen Amazon ECS, layanan, dan definisi tugas.

Parameter pemeriksaan kesehatan penyeimbang beban untuk waktu penerapan Amazon ECS yang lebih cepat

Load balancer merutekan permintaan hanya ke target sehat di Availability Zones untuk penyeimbang beban. Setiap penyeimbang beban memeriksa kesehatan setiap target, menggunakan pengaturan pemeriksaan kesehatan untuk kelompok sasaran yang dengannya target terdaftar. Setelah target Anda terdaftar, target itu harus lulus satu pemeriksaan kondisi agar dapat dianggap sehat. Diagram berikut menjelaskan proses pemeriksaan kesehatan penyeimbang beban. Penyeimbang beban

secara berkala mengirimkan pemeriksaan kesehatan ke wadah Amazon ECS. Agen Amazon ECS memantau dan menunggu penyeimbang beban melaporkan kesehatan kontainer. Ia melakukan ini sebelum menganggap wadah berada dalam status sehat.



Dua parameter pemeriksaan kesehatan Elastic Load Balancing memengaruhi kecepatan penerapan:

- Interval pemeriksaan kesehatan: Menentukan perkiraan jumlah waktu, dalam hitungan detik, antara pemeriksaan kesehatan wadah individu. Secara default, penyeimbang beban memeriksa setiap 30 detik.

  Parameter ini dinamai:

  - `HealthCheckIntervalSeconds`di Elastic Load Balancing API
  - Interval pada konsol Amazon EC2
- Jumlah ambang batas yang sehat: Menentukan jumlah keberhasilan pemeriksaan kesehatan berturut-turut yang diperlukan sebelum mempertimbangkan wadah yang tidak sehat sehat. Secara default, penyeimbang beban memerlukan lima pemeriksaan kesehatan yang lewat sebelum melaporkan bahwa wadah target sehat.

  Parameter ini dinamai:

  - `HealthyThresholdCount`di Elastic Load Balancing API

- Ambang batas yang sehat di konsol Amazon EC2

Dengan pengaturan default, total waktu untuk menentukan kesehatan wadah adalah dua menit dan 30 detik (`30 seconds * 5 = 150 seconds`).

Anda dapat mempercepat proses pemeriksaan kesehatan jika layanan Anda dimulai dan stabil dalam waktu kurang dari 10 detik. Untuk mempercepat proses, kurangi jumlah pemeriksaan kesehatan dan interval antara pemeriksaan.

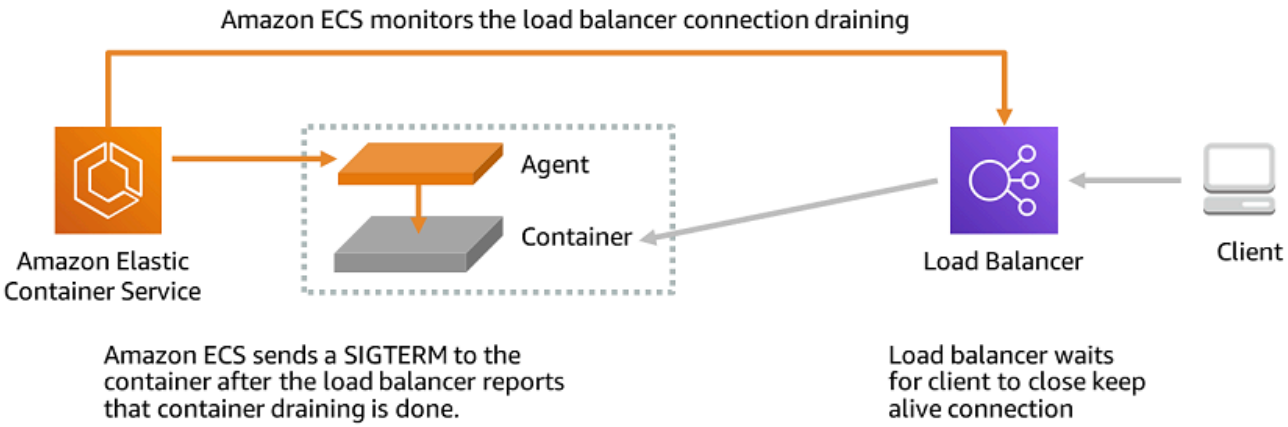
- `HealthCheckIntervalSeconds`(Nama Elastic Load Balancing API) atau Interval (nama konsol Amazon EC2): 5
- `HealthyThresholdCount`(Nama Elastic Load Balancing API) atau ambang Healthy (nama konsol Amazon EC2): 2

Dengan pengaturan ini, proses pemeriksaan kesehatan membutuhkan waktu 10 detik dibandingkan dengan default dua menit dan 30 detik.

Untuk informasi selengkapnya tentang parameter pemeriksaan kesehatan Elastic Load Balancing, lihat Pemeriksaan Kesehatan untuk grup target Anda di Panduan Pengguna Elastic Load Balancing.

Parameter pengurasan koneksi penyeimbang beban untuk waktu penerapan Amazon ECS yang lebih cepat

Untuk memungkinkan pengoptimalan, klien mempertahankan koneksi tetap hidup ke layanan kontainer. Ini agar permintaan selanjutnya dari klien tersebut dapat menggunakan kembali koneksi yang ada. Saat Anda ingin menghentikan lalu lintas ke kontainer, Anda memberi tahu penyeimbang beban. Ketika Anda memberi tahu penyeimbang beban untuk menghentikan lalu lintas ke kontainer, secara berkala memeriksa untuk melihat apakah klien menutup koneksi tetap hidup. Agen Amazon ECS memantau penyeimbang beban, dan menunggu penyeimbang beban melaporkan bahwa koneksi tetap hidup ditutup (target dalam keadaan). `UNUSED`



Jumlah waktu yang menunggu penyeimbang beban untuk memindahkan target ke `UNUSED` status adalah penundaan deregistrasi. Anda dapat mengonfigurasi parameter penyeimbang beban berikut untuk mempercepat penerapan Anda.

- `deregistration_delay.timeout_seconds`: 300 (default)

Ketika Anda memiliki layanan dengan waktu respons di bawah satu detik, atur parameter ke nilai berikut agar penyeimbang beban hanya menunggu 5 detik sebelum memutuskan koneksi antara klien dan layanan back-end:

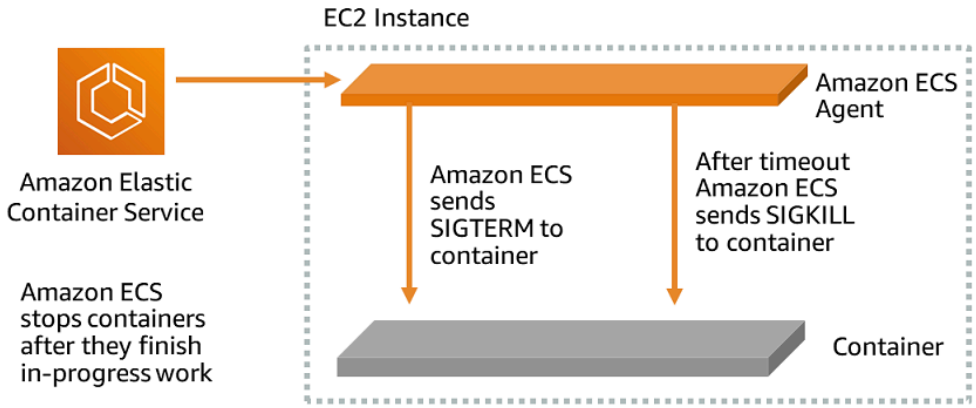
- `deregistration_delay.timeout_seconds`: 5

> **Note**
>
> Jangan atur nilainya menjadi 5 detik saat Anda memiliki layanan dengan permintaan yang berumur panjang, seperti unggahan file yang lambat atau koneksi streaming.

Responsif SIGTERM

Amazon ECS pertama kali mengirimkan sinyal SIGTERM ke tugas untuk memberi tahu aplikasi harus diselesaikan dan dimatikan. Kemudian, Amazon ECS mengirimkan pesan SIGKILL. Ketika aplikasi mengabaikan SIGTERM, layanan Amazon ECS harus menunggu untuk mengirim sinyal SIGKILL untuk menghentikan proses.

Diagram berikut menunjukkan bagaimana Amazon ECS mengakhiri tugas.



Jumlah waktu yang Amazon ECS menunggu untuk mengirim pesan SIGKILL ditentukan oleh opsi agen Amazon ECS berikut:

- `ECS_CONTAINER_STOP_TIMEOUT`: 30 (default)

  Untuk informasi selengkapnya tentang parameter agen kontainer, lihat Agen Kontainer Amazon ECS di GitHub.

Untuk mempercepat masa tunggu, atur parameter agen Amazon ECS ke nilai berikut:

> **Note**
>
> Jika aplikasi Anda membutuhkan waktu lebih dari 1 detik, kalikan nilainya dengan 2 dan gunakan angka itu sebagai nilainya.

- `ECS_CONTAINER_STOP_TIMEOUT`: 2

Dalam hal ini, Amazon ECS menunggu 2 detik hingga wadah dimatikan, dan kemudian Amazon ECS mengirimkan pesan SIGKILL ketika aplikasi tidak berhenti.

Anda juga dapat memodifikasi kode aplikasi untuk menjebak sinyal SIGTERM dan bereaksi terhadapnya. Berikut ini adalah contoh di JavaScript:

```
process.on('SIGTERM', function() {
  server.close();
})
```

Kode ini menyebabkan server HTTP berhenti mendengarkan permintaan baru, menyelesaikan menjawab permintaan dalam penerbangan, dan kemudian proses Node.js berakhir. Ini karena loop acaranya tidak ada lagi yang bisa dilakukan. Mengingat hal ini, jika dibutuhkan proses hanya 500 ms untuk menyelesaikan permintaan dalam penerbangannya, itu berakhir lebih awal tanpa harus menunggu batas waktu berhenti dan dikirimi SIGKILL.

Perilaku menarik gambar kontainer untuk penerapan Amazon ECS yang lebih cepat

Waktu yang dibutuhkan wadah untuk memulai bervariasi, berdasarkan gambar kontainer yang mendasarinya. Misalnya, gambar yang lebih gemuk (versi lengkap Debian, Ubuntu, dan Amazon1/2) mungkin membutuhkan waktu lebih lama untuk memulai karena ada lebih banyak layanan yang berjalan di wadah dibandingkan dengan versi ramping masing-masing (Debian-slim, Ubuntu-slim, dan Amazon-slim) atau gambar dasar yang lebih kecil (Alpine).

Jenis peluncuran Fargate

Fargate tidak menyimpan gambar, dan oleh karena itu seluruh gambar ditarik dari registri saat tugas berjalan. Kami merekomendasikan yang berikut untuk gambar yang digunakan untuk tugas Fargate:

- Gunakan ukuran tugas yang lebih besar dengan vCPU tambahan.

Ukuran tugas yang lebih besar dapat membantu mengurangi waktu yang diperlukan untuk mengekstrak gambar saat tugas diluncurkan.

- Gunakan gambar dasar yang lebih kecil.
- Miliki repositori yang menyimpan gambar di Wilayah yang sama dengan tugas.

AMI Windows pada jenis peluncuran Fargate

Fargate Windows AMI cache bulan terakhir, dan bulan sebelumnya, gambar dasar servercore yang disediakan oleh Microsoft. Gambar-gambar ini cocok dengan patch nomor KB/build yang diperbarui setiap Patch Selasa. Misalnya, pada 8/8/2023 Microsoft merilis KB5029247 (17763.4737) untuk Windows Server 2019. Bulan sebelumnya KB pada 7/11/2023 adalah KB5028168 (17763.4645). Jadi untuk platform `WINDOWS_SERVER_2019_CORE` dan `WINDOWS_SERVER_2019_FULL` gambar kontainer berikut di-cache:

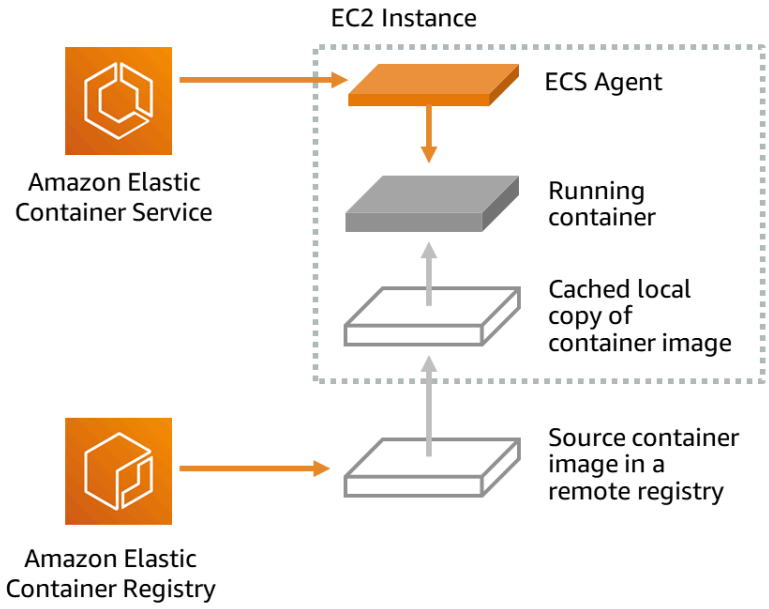
- `mcr.microsoft.com/windows/servercore:ltsc2019`
- `mcr.microsoft.com/windows/servercore:10.0.17763.4737`
- `mcr.microsoft.com/windows/servercore:10.0.17763.4645`

Selain itu, pada 8/8/2023 Microsoft merilis KB5029250 (20348.1906) untuk Windows Server 2022. Bulan sebelumnya KB pada 7/11/2023 adalah KB5028171 (20348.1850). Jadi untuk platform `WINDOWS_SERVER_2022_CORE` dan `WINDOWS_SERVER_2022_FULL` gambar kontainer berikut di-cache:

- `mcr.microsoft.com/windows/servercore:ltsc2022`
- `mcr.microsoft.com/windows/servercore:10.0.20348.1906`
- `mcr.microsoft.com/windows/servercore:10.0.20348.1850`

Jenis peluncuran Amazon EC2

Ketika agen Amazon ECS memulai tugas, ia menarik gambar Docker dari registri jarak jauhnya, dan kemudian menyimpan salinan lokal. Bila Anda menggunakan tag gambar baru untuk setiap rilis aplikasi Anda, perilaku ini tidak diperlukan.



Parameter `ECS_IMAGE_PULL_BEHAVIOR` agen menentukan perilaku tarik gambar dan memiliki nilai berikut:

- `ECS_IMAGE_PULL_BEHAVIOR`: `default`

  Gambar akan ditarik dari jarak jauh. Jika tarikan gagal, gambar cache dalam instance digunakan.
- `ECS_IMAGE_PULL_BEHAVIOR`: `always`

  Gambar akan ditarik dari jarak jauh. Jika tarikan gagal, tugas gagal.

Untuk mempercepat penerapan, setel parameter agen Amazon ECS ke salah satu nilai berikut:

- `ECS_IMAGE_PULL_BEHAVIOR`: `once`

  Gambar ditarik dari jarak jauh hanya jika tidak ditarik oleh tugas sebelumnya pada instance penampung yang sama atau jika gambar yang di-cache telah dihapus oleh proses pembersihan gambar otomatis. Jika tidak, citra cache pada instans digunakan. Hal ini memastikan bahwa tidak ada percobaan penarikan citra yang tidak perlu.
- `ECS_IMAGE_PULL_BEHAVIOR`: `prefer-cached`

Gambar ditarik dari jarak jauh jika tidak ada gambar yang di-cache. Jika tidak, citra cache pada instans digunakan. Pembersihan gambar otomatis dimatikan untuk wadah untuk memastikan bahwa gambar yang di-cache tidak dihapus.

Menyetel `ECS_IMAGE_PULL_BEHAVIOR` parameter ke salah satu nilai sebelumnya dapat menghemat waktu karena agen Amazon ECS menggunakan gambar yang diunduh yang ada. Untuk gambar Docker yang lebih besar, waktu pengunduhan mungkin memakan waktu 10-20 detik untuk melewati jaringan.

Parameter layanan Amazon ECS untuk penerapan Amazon ECS yang lebih cepat

Untuk memastikan tidak ada downtime aplikasi, proses penerapan adalah sebagai berikut:

1. Mulai wadah aplikasi baru sambil menjaga kontainer yang ada tetap berjalan.
2. Periksa apakah wadah baru itu sehat.
3. Hentikan wadah lama.

Bergantung pada konfigurasi penerapan Anda dan jumlah ruang kosong dan tanpa cadangan di klaster Anda, mungkin diperlukan beberapa putaran untuk menyelesaikannya, ganti semua tugas lama dengan tugas baru.

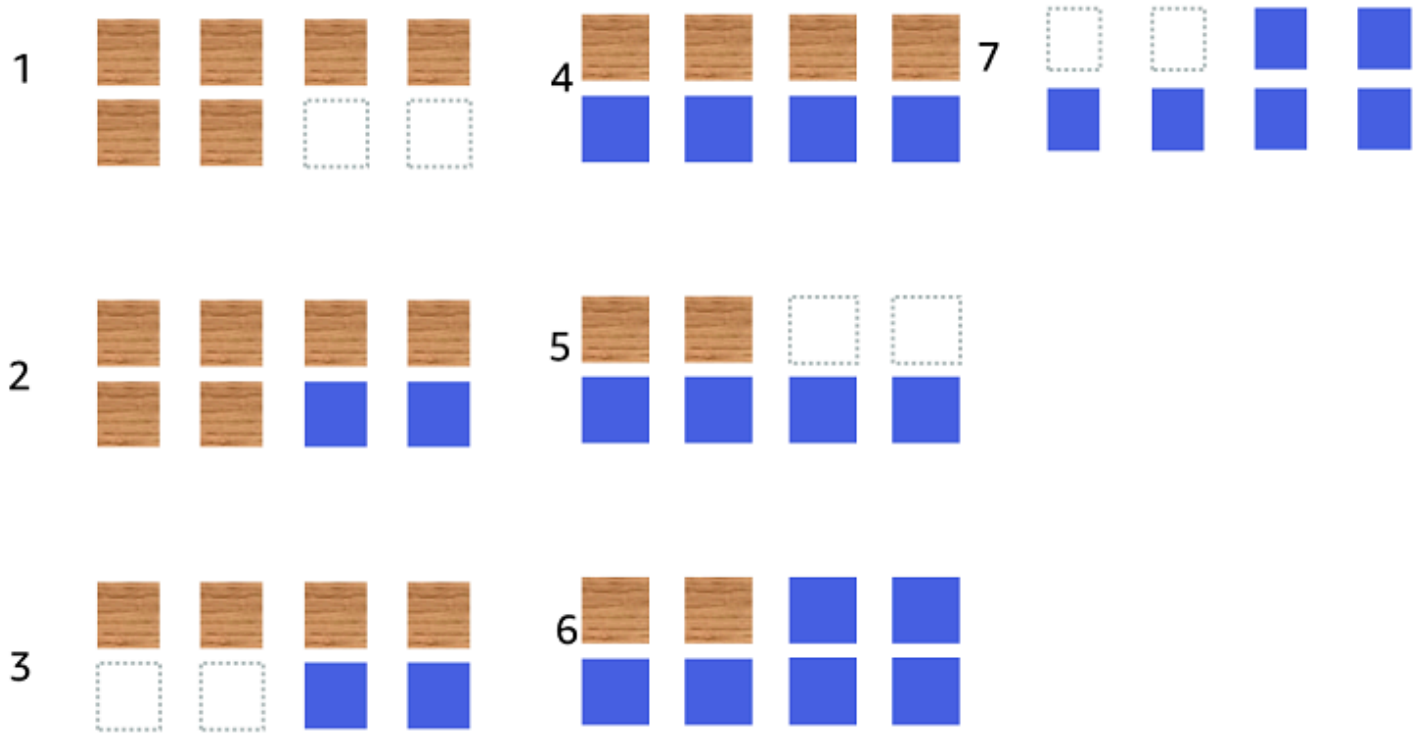
Ada dua opsi konfigurasi layanan ECS yang dapat Anda gunakan untuk mengubah nomor:

- `minimumHealthyPercent`: 100% (default)

  Batas bawah pada jumlah tugas untuk layanan Anda yang harus tetap dalam `RUNNING` status selama penerapan. Ini adalah persentase dari yang `desiredCount` dibulatkan ke bilangan bulat terdekat. Parameter ini memungkinkan Anda untuk menyebarkan tanpa menggunakan kapasitas cluster tambahan.
- `maximumPercent`: 200% (default)

  Batas atas jumlah tugas untuk layanan Anda yang diizinkan di `RUNNING` atau `PENDING` status selama penerapan. Ini adalah persentase dari yang `desiredCount` dibulatkan ke bawah ke bilangan bulat terdekat.

Pertimbangkan layanan berikut yang memiliki enam tugas tan, yang digunakan dalam cluster yang memiliki ruang untuk total delapan tugas. Opsi konfigurasi layanan Amazon ECS default tidak memungkinkan penerapan berada di bawah 100% dari enam tugas yang diinginkan.



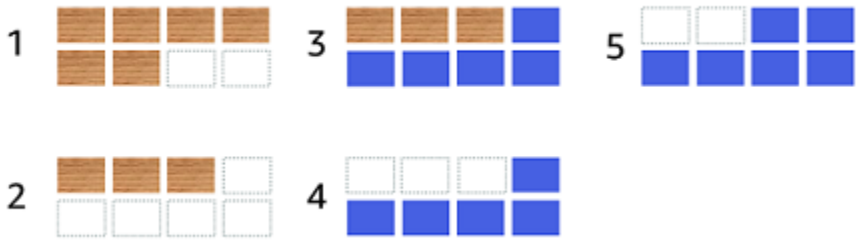
Proses penyebaran adalah sebagai berikut:

1. Tujuannya adalah untuk mengganti tugas cokelat dengan tugas biru.
2. Penjadwal memulai dua tugas biru baru karena pengaturan default mengharuskan ada enam tugas yang berjalan.
3. Penjadwal menghentikan dua tugas tan karena akan ada total enam tugas (empat tan dan dua biru).
4. Penjadwal memulai dua tugas biru tambahan.
5. Penjadwal menutup dua tugas tan.
6. Penjadwal memulai dua tugas biru tambahan.
7. Penjadwal menutup dua tugas tan terakhir.

Dalam contoh di atas, jika Anda menggunakan nilai default untuk opsi, ada 2,5 menit menunggu untuk setiap tugas baru yang dimulai. Selain itu, penyeimbang beban mungkin harus menunggu 5 menit agar tugas lama berhenti.

Anda dapat mempercepat penerapan dengan menyetel `minimumHealthyPercent` nilainya menjadi 50%.

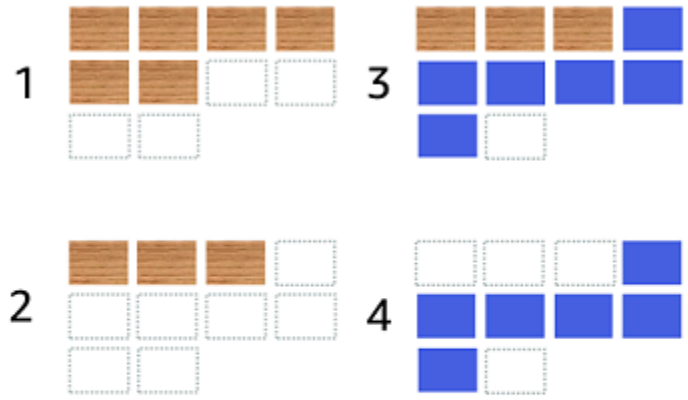
Pertimbangkan layanan berikut yang memiliki enam tugas tan, yang digunakan dalam cluster yang memiliki ruang untuk total delapan tugas.



Proses penyebaran adalah sebagai berikut:

1. Tujuannya adalah untuk mengganti tugas cokelat dengan tugas biru.
2. Penjadwal menghentikan tiga tugas tan. Masih ada tiga tugas tan yang berjalan yang memenuhi `minimumHealthyPercent` nilainya.
3. Penjadwal memulai lima tugas biru.
4. Penjadwal menghentikan tiga tugas tan yang tersisa.
5. Penjadwal memulai tugas biru terakhir.

Anda juga dapat menambahkan ruang kosong tambahan sehingga Anda dapat menjalankan tugas tambahan.



Proses penyebaran adalah sebagai berikut:

1. Tujuannya adalah untuk mengganti tugas cokelat dengan tugas biru.
2. Penjadwal menghentikan tiga tugas tan
3. Penjadwal memulai enam tugas biru
4. Penjadwal menghentikan tiga tugas tan.

Gunakan nilai berikut untuk opsi konfigurasi layanan Amazon ECS saat tugas Anda menganggur selama beberapa waktu dan tidak memiliki tingkat pemanfaatan yang tinggi.

- `minimumHealthyPercent`: 50%
- `maximumPercent`: 200%

Membuat layanan menggunakan konsol

Anda dapat membuat layanan menggunakan konsol.

Pertimbangkan hal berikut saat Anda menggunakan konsol:

- Ada dua opsi komputasi yang mendistribusikan tugas Anda.

  - Strategi penyedia kapasitas menyebabkan Amazon ECS mendistribusikan tugas Anda di satu atau di beberapa penyedia kapasitas.
  - Jenis peluncuran menyebabkan Amazon ECS meluncurkan tugas kami secara langsung di Fargate atau di instans Amazon EC2 yang terdaftar ke cluster Anda.
- Penentuan tugas yang menggunakan mode atau layanan jaringan `awsvpc` yang dikonfigurasi untuk menggunakan penyeimbang beban harus memiliki konfigurasi jaringan. Secara default, konsol memilih VPC Amazon default bersama dengan semua subnet dan grup keamanan default dalam VPC Amazon default.
- Strategi penempatan tugas default mendistribusikan tugas secara merata di seluruh Availability Zone.
- Saat Anda menggunakan Jenis Peluncuran untuk penyebaran layanan Anda, secara default layanan dimulai di subnet di VPC klaster Anda.
- Untuk strategi penyedia kapasitas, konsol memilih opsi komputasi secara default. Berikut dijelaskan tentang urutan yang digunakan konsol untuk memilih default:
  - Jika klaster Anda memiliki strategi penyedia kapasitas default yang ditentukan, klaster akan dipilih.
  - Jika klaster Anda tidak memiliki strategi penyedia kapasitas default yang ditentukan tetapi Anda memiliki penyedia kapasitas Fargate yang ditambahkan ke klaster, strategi penyedia kapasitas khusus yang menggunakan penyedia `FARGATE` kapasitas dipilih.
  - Jika klaster Anda tidak memiliki strategi penyedia kapasitas default yang ditentukan tetapi Anda memiliki satu atau beberapa penyedia kapasitas grup Auto Scaling yang ditambahkan ke cluster, opsi Use custom (Advanced) akan dipilih dan Anda perlu menentukan strategi secara manual.
  - Jika klaster Anda tidak memiliki strategi penyedia kapasitas default yang ditentukan dan tidak ada penyedia kapasitas yang ditambahkan ke cluster, jenis peluncuran Fargate akan dipilih.
- Opsi default deteksi kegagalan penerapan default adalah menggunakan opsi pemutus sirkuit penyebaran Amazon ECS dengan opsi Rollback on failure.

  Untuk informasi selengkapnya, lihat Pemutus sirkuit penyebaran.
- Jika Anda ingin menggunakan opsi penyebaran biru/hijau, tentukan cara CodeDeploy memindahkan aplikasi. Pilihan berikut tersedia:
  - CodeDeployDefault.ECS AllAtOnce: Menggeser semua lalu lintas ke wadah Amazon ECS yang diperbarui sekaligus
  - CodeDeployDefault.ecslinear10 PercentEvery 1Menit: Menggeser 10 persen lalu lintas setiap menit sampai semua lalu lintas bergeser.

  - CodeDeployDefault.ecslinear10 PercentEvery 3Minutes: Menggeser 10 persen lalu lintas setiap 3 menit sampai semua lalu lintas bergeser.
  - CodeDeployDefault.ecscanary10percent5minutes: Menggeser 10 persen lalu lintas pada kenaikan pertama. Sisanya 90 persen dikerahkan lima menit kemudian.
  - CodeDeployDefault.ecscanary10Percent15minutes: Menggeser 10 persen lalu lintas pada kenaikan pertama. Sisa 90 persen di-deploy 15 menit kemudian.
- Jika Anda memerlukan aplikasi untuk terhubung ke aplikasi lain yang berjalan di Amazon ECS, tentukan opsi yang sesuai dengan arsitektur Anda. Untuk informasi selengkapnya, lihat Layanan interkoneksi.
- Anda harus menggunakan AWS CloudFormation atau AWS Command Line Interface untuk menyebarkan layanan yang menggunakan salah satu parameter berikut:
  - Melacak kebijakan dengan metrik khusus
  - Perbarui Layanan - Anda tidak dapat memperbarui konfigurasi `awsvpc` jaringan dan masa tenggang pemeriksaan kesehatan.

  Untuk informasi tentang cara membuat layanan menggunakan AWS CLI, lihat create-servicedi AWS Command Line Interface Referensi.

  Untuk informasi tentang cara membuat layanan menggunakan AWS CloudFormation, lihat AWS::ECS::Servicedi Panduan AWS CloudFormation Pengguna.

Buat layanan dengan cepat

Anda dapat menggunakan konsol untuk membuat dan menyebarkan layanan dengan cepat. Layanan ini memiliki konfigurasi berikut:

- Menyebarkan di VPC dan subnet yang terkait dengan cluster Anda
- Menyebarkan satu tugas
- Menggunakan penerapan bergulir
- Menggunakan strategi penyedia kapasitas dengan penyedia kapasitas default Anda
- Menggunakan pemutus sirkuit penyebaran untuk mendeteksi kegagalan dan menetapkan opsi untuk secara otomatis memutar kembali penerapan pada kegagalan

Untuk menerapkan layanan menggunakan parameter default ikuti langkah-langkah ini.

Untuk membuat layanan (konsol Amazon ECS)

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di halaman navigasi, pilih Cluster.
3. Pada halaman Clusters, pilih cluster untuk membuat layanan di.
4. Dari tab Layanan, pilih Buat.
5. Di bawah konfigurasi Deployment, tentukan cara aplikasi Anda di-deploy.

    a. Untuk jenis Aplikasi, pilih Layanan.

    b. Untuk ketentuan tugas, pilih keluarga ketentuan tugas dan revisi yang akan digunakan.

    c. Untuk nama Layanan, masukkan nama untuk layanan Anda.

    d. Untuk tugas yang diinginkan, masukkan jumlah tugas untuk diluncurkan dan dipelihara dalam layanan.
6. (Opsional) Untuk membantu mengidentifikasi layanan dan tugas Anda, perluas bagian Tag, lalu konfigurasikan tag Anda.

    Agar Amazon ECS secara otomatis menandai semua tugas yang baru diluncurkan dengan nama cluster dan tag definisi tugas, pilih Aktifkan tag terkelola Amazon ECS, lalu pilih Definisi tugas.

    Agar Amazon ECS secara otomatis menandai semua tugas yang baru diluncurkan dengan nama cluster dan tag layanan, pilih Aktifkan tag terkelola Amazon ECS, lalu pilih Layanan.

    Menambah atau menghapus tanda.

    - [Tambahkan tag] Pilih Tambah tag, lalu lakukan hal berikut:
        - Untuk Kunci, masukkan nama kunci.
        - Untuk Nilai, masukkan nilai kunci.
    - [Menghapus tanda] Di samping tanda, pilih Hapus tanda.

Buat layanan menggunakan parameter yang ditentukan

Untuk membuat layanan dengan menggunakan parameter yang ditentukan, ikuti langkah-langkah ini.

Untuk membuat layanan (konsol Amazon ECS)

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Tentukan sumber daya dari tempat Anda meluncurkan layanan.

Untuk memulai layanan dari	Langkah-langkah
Klaster	a. Pada halaman Klaster, pilih klaster untuk membuat layanan. b. Dari tab Layanan, pilih Buat.
Jenis peluncuran	a. Pada halaman Definisi tugas, pilih tombol opsi di sebelah definisi tugas. b. Pada menu Deploy, pilih Buat layanan.

3. (Opsional) Pilih bagaimana tugas Anda didistribusikan di seluruh infrastruktur klaster Anda. Perluas konfigurasi Compute, lalu pilih opsi Anda.

Metode distribusi	Langkah-langkah
Strategi penyedia kapasitas	a. Di bawah Opsi komputasi , pilih Strategi penyedia kapasitas. b. Pilih strategi: • Untuk menggunak an strategi penyedia kapasitas default cluster, pilih Use cluster default. • Jika klaster Anda tidak memiliki strategi penyedia kapasitas default, atau menggunakan strategi kustom, pilih Gunakan strategi penyedia

Metode distribusi	Langkah-langkah
	kustom, Tambahkan kapasitas, lalu tentukan strategi penyedia kapasitas khusus Anda dengan menentuka n Basis, penyedia Kapasitas, dan Berat. **Note** Untuk menggunakan penyedia kapasitas dalam suatu strategi, penyedia kapasitas harus dikaitkan dengan cluster. Untuk informasi selengkapnya tentang strategi penyedia kapasitas , lihat Penyedia kapasitas Amazon ECS.

Metode distribusi	Langkah-langkah
Jenis peluncuran	a. Di bagian Compute options, pilih Launch type. b. Untuk jenis Peluncuran, pilih jenis peluncuran. c. (Opsional) Ketika tipe peluncuran Fargate diwajibkan, untuk Versi platform, tentukan versi platform yang akan digunakan. Jika versi platform tidak ditentuka n, versi LATEST platform akan digunakan.

4. Untuk menentukan bagaimana layanan Anda digunakan, buka bagian konfigurasi Deployment, lalu pilih opsi Anda.

   a. Untuk jenis Aplikasi, tinggalkan pilihan sebagai Layanan.

   b. Untuk definisi Tugas dan Revisi, pilih keluarga definisi tugas dan revisi yang akan digunakan.

   c. Untuk nama Layanan, masukkan nama untuk layanan Anda.

   d. Untuk jenis Layanan, pilih strategi penjadwalan layanan.

      - Agar penjadwal menerapkan tepat satu tugas pada setiap instance kontainer aktif yang memenuhi semua batasan penempatan tugas, pilih Daemon.
      - Untuk menempatkan penjadwal dan mempertahankan jumlah tugas yang diinginkan di cluster Anda, pilih Replika.

      Untuk informasi selengkapnya, lihat the section called “Konsep penjadwal layanan”.

   e. Jika Anda memilih Replika, untuk tugas yang diinginkan, masukkan jumlah tugas yang akan diluncurkan dan dipelihara dalam layanan.

   f. Tentukan jenis penerapan untuk layanan Anda. Perluas opsi Deployment, lalu tentukan parameter berikut.

Jenis deployment	Langkah-langkah	
Pembaruan bergulir	a. Untuk tugas yang menjalankan Min, masukkan batas bawah pada jumlah tugas dalam layanan yang harus tetap dalam `RUNNING` status selama penerapan, sebagai persentase dari jumlah tugas yang diinginkan (dibulatkan ke bilangan bulat terdekat). Untuk informasi selengkap nya, lihat Konfigurasi penerapan. b. Untuk tugas yang berjalan Max, masukkan batas atas jumlah tugas dalam layanan yang diizinkan dalam `PENDING` status `RUNNING` atau selama penerapan, sebagai persentase dari jumlah tugas yang diinginkan (dibulatkan ke bilangan bulat terdekat).	

Jenis deployment	Langkah-langkah	
Deployment blue/green	a. Untuk konfigurasi Deployment, pilih cara CodeDeploy merutekan lalu lintas produksi ke tugas pengganti yang ditetapkan selama penerapan. b. Untuk peran Service CodeDeploy, pilih peran IAM yang digunakan layanan untuk membuat permintaan API ke otorisasi Layanan AWS.	

g. Untuk mengonfigurasi cara Amazon ECS mendeteksi dan menangani kegagalan penerapan, perluas deteksi kegagalan Deployment, lalu pilih opsi Anda.

   i. Untuk menghentikan penerapan saat tugas tidak dapat dimulai, pilih Gunakan pemutus sirkuit penyebaran Amazon ECS.

      Agar perangkat lunak secara otomatis memutar kembali penerapan ke status penerapan yang terakhir selesai saat pemutus sirkuit penyebaran mengatur penerapan ke status gagal, pilih Rollback on failure.

   ii. Untuk menghentikan penerapan berdasarkan metrik aplikasi, pilih Gunakan CloudWatch alarm. Kemudian, dari nama CloudWatch alarm, pilih alarm. Untuk membuat alarm baru, buka CloudWatch konsol.

      Agar perangkat lunak secara otomatis memutar kembali penerapan ke status penerapan yang terakhir selesai saat CloudWatch alarm menyetel penerapan ke status gagal, pilih Rollback on failure.

5. (Opsional) Untuk menggunakan Service Connect, pilih Aktifkan Service Connect, lalu tentukan yang berikut ini:

   a. Di bawah konfigurasi Service Connect, tentukan mode klien.

- Jika layanan Anda menjalankan aplikasi klien jaringan yang hanya perlu terhubung ke layanan lain di namespace, pilih sisi Klien saja.
- Jika layanan Anda menjalankan aplikasi jaringan atau layanan web dan perlu menyediakan titik akhir untuk layanan ini, dan terhubung ke layanan lain di namespace, pilih Klien dan server.

b. Untuk menggunakan namespace yang bukan namespace cluster default, untuk Namespace, pilih namespace layanan.

c. (Opsional) Pilih opsi Gunakan koleksi log untuk menentukan konfigurasi log. Untuk setiap driver log yang tersedia, ada opsi driver log untuk ditentukan. Opsi default mengirimkan log kontainer ke CloudWatch Log. Opsi driver log lainnya dikonfigurasi menggunakan AWS FireLens. Untuk informasi selengkapnya, lihat Menggunakan perutean log khusus.

Berikut ini menjelaskan setiap tujuan log kontainer secara lebih rinci.

- Amazon CloudWatch — Konfigurasikan tugas untuk mengirim log kontainer ke CloudWatch Log. Opsi driver log default disediakan, yang membuat grup CloudWatch log atas nama Anda. Untuk menentukan nama grup log yang berbeda, ubah nilai opsi driver.
- Amazon Data Firehose — Konfigurasikan tugas untuk mengirim log kontainer ke Firehose. Opsi driver log default disediakan, yang mengirim log ke aliran pengiriman Firehose. Untuk menentukan nama aliran pengiriman yang berbeda, ubah nilai opsi driver.
- Amazon Kinesis Data Streams — Konfigurasikan tugas untuk mengirim log kontainer ke Kinesis Data Streams. Opsi driver log default disediakan, yang mengirim log ke aliran Kinesis Data Streams. Untuk menentukan nama aliran yang berbeda, ubah nilai opsi driver.
- Amazon OpenSearch Service - Konfigurasikan tugas untuk mengirim log kontainer ke domain OpenSearch Layanan. Opsi driver log harus disediakan. Untuk informasi selengkapnya, lihat Meneruskan log ke domain Layanan Amazon OpenSearch .
- Amazon S3 — Konfigurasikan tugas untuk mengirim log kontainer ke bucket Amazon S3. Opsi driver log default disediakan, tetapi Anda harus menentukan nama bucket Amazon S3 yang valid.

6. (Opsional) Untuk menggunakan Penemuan Layanan, pilih Gunakan penemuan layanan, lalu tentukan yang berikut ini.

a. Untuk menggunakan namespace baru, pilih Buat namespace baru di bawah Konfigurasi namespace, lalu berikan nama dan deskripsi namespace. Untuk menggunakan namespace yang ada, pilih Pilih namespace yang ada lalu pilih namespace yang ingin Anda gunakan.

b. Memberikan informasi layanan Service Discovery seperti nama dan deskripsi layanan.

c. Agar Amazon ECS melakukan pemeriksaan kesehatan tingkat kontainer secara berkala, pilih Aktifkan propagasi kesehatan tugas Amazon ECS.

d. Untuk jenis rekaman DNS, pilih jenis rekaman DNS yang akan dibuat untuk layanan Anda. Penemuan layanan Amazon ECS hanya mendukung catatan A dan SRV, tergantung pada mode jaringan yang ditentukan oleh definisi tugas Anda. Untuk informasi selengkapnya tentang jenis rekaman ini, lihat Jenis Rekaman DNS yang Didukung di Panduan Pengembang Amazon Route 53.

   - Jika definisi tugas yang ditentukan oleh tugas layanan Anda menggunakan mode `host` jaringan `bridge` atau, hanya jenis catatan SRV yang didukung. Pilih nama kontainer dan kombinasi port untuk dikaitkan dengan catatan.
   - Jika definisi tugas yang ditentukan oleh tugas layanan Anda menggunakan mode `awsvpc` jaringan, pilih jenis catatan A atau SRV. Jika Anda memilih A, lewati ke langkah berikutnya. Jika Anda memilih SRV, tentukan port tempat layanan dapat ditemukan atau nama kontainer dan kombinasi port untuk dikaitkan dengan catatan.

   Untuk TTL, masukkan waktu dalam hitungan detik berapa lama set rekaman di-cache oleh resolver DNS dan oleh browser web.

7. (Opsional) Untuk mengonfigurasi penyeimbang beban untuk layanan Anda, perluas Load balancing.

   Pilih penyeimbang beban.

Untuk menggunakan penyeimbang beban ini	Lakukan hal berikut
Penyeimbang Beban Aplikasi	a. Untuk tipe Load balancer, pilih Application Load Balancer. b. Pilih Buat penyeimba ng beban baru untuk

Untuk menggunakan penyeimbang beban ini	Lakukan hal berikut	
	membuat Application Load Balancer baru atau Gunakan penyeimbang beban yang ada untuk memilih Application Load Balancer yang ada. c. Untuk nama Load balancer, masukkan nama unik. d. Untuk Pilih kontainer untuk memuat keseimban gan, pilih kontainer yang menampung layanan. e. Untuk Listener, masukkan port dan protokol untuk Application Load Balancer untuk mendengarkan permintaa n koneksi. Secara default, penyeimbang beban akan dikonfigurasi untuk menggunakan port 80 dan HTTP. f. Untuk nama grup Target, masukkan nama dan protokol untuk grup target yang diminta oleh Application Load Balancer. Secara default, grup target merutekan permintaan ke kontainer pertama yang ditentukan dalam definisi tugas Anda.	

Untuk menggunakan penyeimbang beban ini	Lakukan hal berikut	
	g. Untuk penundaan degregistrasi, masukkan jumlah detik untuk penyeimbang beban untuk mengubah status target menjadi. `UNUSED` Waktu default adalah 300 detik. h. Untuk jalur pemeriksaan Kesehatan, masukkan jalur yang ada di dalam wadah Anda di mana Application Load Balancer secara berkala mengirimkan permintaan untuk memverifikasi kesehatan koneksi antara Application Load Balancer dan container. Defaultnya adalah root directory (`/`). i. Untuk Health check masa tenggang, masukkan jumlah waktu (dalam detik) bahwa penjadwal layanan harus mengabaikan pemeriksaan kesehatan target Elastic Load Balancing yang tidak sehat.	

Untuk menggunakan penyeimbang beban ini	Lakukan hal berikut
Network Load Balancer	a. Untuk jenis Load balancer, pilih Network Load Balancer. b. Untuk Load Balancer, pilih Network Load Balancer yang sudah ada. c. Untuk Pilih kontainer untuk memuat keseimban gan, pilih kontainer yang menampung layanan. d. Untuk nama grup Target, masukkan nama dan protokol untuk grup target yang diminta oleh Network Load Balancer. Secara default, grup target merutekan permintaan ke kontainer pertama yang ditentukan dalam definisi tugas Anda. e. Untuk penundaan degregistrasi, masukkan jumlah detik untuk penyeimbang beban untuk mengubah status target menjadi. `UNUSED` Waktu default adalah 300 detik. f. Untuk jalur pemeriksaan Kesehatan, masukkan jalur yang ada di dalam wadah Anda di mana Network Load Balancer

Untuk menggunakan penyeimbang beban ini	Lakukan hal berikut	
	secara berkala mengirimkan permintaan untuk memverifikasi kesehatan koneksi antara Application Load Balancer dan container. Defaultnya adalah root directory (/). g. Untuk Health check masa tenggang, masukkan jumlah waktu (dalam detik) bahwa penjadwal layanan harus mengabaikan pemeriksaan kesehatan target Elastic Load Balancing yang tidak sehat.	

8. (Opsional) Untuk mengonfigurasi layanan Auto Scaling, perluas Service auto scaling, lalu tentukan parameter berikut.

   a. Untuk menggunakan penskalaan otomatis servis, pilih Penskalaan otomatis layanan.

   b. Untuk Jumlah tugas minimum, masukkan batas bawah jumlah tugas untuk penskalaan otomatis servis yang akan digunakan. Hitungan yang diinginkan tidak akan berada di bawah hitungan ini.

   c. Untuk Jumlah tugas maksimum, masukkan batas atas jumlah tugas untuk penskalaan otomatis servis yang akan digunakan. Hitungan yang diinginkan tidak akan melebihi hitungan ini.

   d. Pilih jenis kebijakan. Di bawah Jenis kebijakan penskalaan, pilih salah satu opsi berikut.

Untuk menggunakan jenis kebijakan ini...	Lakukan ini...	
Pelacakan target	a. Untuk Jenis kebijakan penskalaan, pilih Pelacakan target. b. Untuk nama Kebijakan , masukkan nama kebijakan. c. Untuk metrik layanan ECS, pilih salah satu metrik berikut. • ECS ServiceAverage CPUUutilization — Pemanfaatan CPU rata-rata layanan. • ECS ServiceAverageMemoryUtilization — Pemanfaatan memori rata-rata layanan. • ALB RequestCountPerTarget — Jumlah permintaan yang diselesaikan per target dalam kelompok sasaran Application Load Balancer. d. Untuk nilai Target, masukkan nilai yang dipertahankan layanan untuk metrik yang dipilih. e. Untuk periode cooldown scale-out, masukkan	

Untuk menggunakan jenis kebijakan ini...	Lakukan ini...	
	jumlah waktu, dalam detik, setelah aktivitas scale-out (tambahka n tugas) yang harus dilewati sebelum aktivitas scale-out lainnya dapat dimulai. f. Untuk periode cooldown Scale-in, masukkan jumlah waktu, dalam detik, setelah aktivitas scale-in (hapus tugas) yang harus dilewati sebelum aktivitas scale-in lainnya dapat dimulai. g. Untuk mencegah kebijakan melakukan aktivitas penskalaan, pilih Matikan penskalaa n. h. • (Opsional) Pilih Matikan penskalaa n jika Anda ingin kebijakan penskalaa n Anda ditingkatkan untuk meningkatkan lalu lintas tetapi tidak membutuhkannya untuk menskalakan saat lalu lintas menurun.	

Untuk menggunakan jenis kebijakan ini...	Lakukan ini...	
Penskalaan langkah	a. Untuk Jenis kebijakan penskalaan, pilih Penskalaan langkah. b. Untuk nama Kebijakan , masukkan nama kebijakan. c. Untuk Nama alarm, masukkan nama yang unik untuk alarm. d. Untuk metrik layanan Amazon ECS, pilih metrik yang akan digunakan untuk alarm. e. Untuk Statistik, pilih statistik alarm. f. Untuk Periode, pilih periode untuk alarm. g. Untuk kondisi Alarm, pilih cara membandin gkan metrik yang dipilih dengan ambang batas yang ditentukan. h. Untuk Ambang untuk membandingkan metrik dan periode Evaluasi untuk memulai alarm, masukkan ambang batas yang digunakan untuk alarm dan berapa lama untuk mengevaluasi ambang batas.	

Untuk menggunakan jenis kebijakan ini...	Lakukan ini...	
	i. Di bawah Tindakan penskalaan, lakukan hal berikut: • Untuk Tindakan, pilih apakah akan menambah, menghapus, atau menetapkan jumlah tertentu yang diinginkan untuk layanan Anda. • Jika Anda memilih untuk menambah atau menghapus tugas, untuk Nilai, masukkan jumlah tugas (atau persen tugas yang ada) untuk ditambahkan atau dihapus saat tindakan penskalaan dimulai. Jika Anda memilih untuk mengatur hitungan yang diinginkan, masukkan jumlah tugas. Untuk Jenis, pilih apakah Nilai adalah bilangan bulat atau nilai persen dari jumlah yang diinginkan yang ada. • Untuk batas Bawah dan Batas atas, masukkan batas	

Untuk menggunakan jenis kebijakan ini...	Lakukan ini...	
	bawah dan batas atas penyesuai an penskalaan langkah Anda. Secara default, batas bawah untuk kebijakan penambahan adalah ambang batas alarm, sedangkan batas atas adalah positif (+) tak terbatas. Secara default, batas atas untuk kebijakan penghapusan adalah ambang batas alarm, sedangkan batas bawah adalah negatif (-) tak terbatas. • (Opsional) Tambahkan opsi penskalaan tambahan. Pilih Tambahkan tindakan penskalaan baru, lalu ulangi langkah tindakan Penskalaan. • Untuk periode Cooldown, masukkan jumlah waktu, dalam detik, untuk menunggu aktivitas penskalaan sebelumnya diterapka n. Untuk kebijakan add, ini adalah waktu	

Untuk menggunakan jenis kebijakan ini...	Lakukan ini...	
	setelah aktivitas scale-out yang kebijakan penskalaan memblokir aktivitas scale-in dan membatasi berapa banyak tugas yang dapat diskalakan pada suatu waktu. Untuk kebijakan penghapusan, ini adalah waktu setelah aktivitas penskalaan yang harus diteruskan sebelum aktivitas penskalaan lainnya dapat dimulai.	

9. (Opsional) Untuk menggunakan strategi penempatan tugas selain default, perluas Penempatan Tugas, lalu pilih dari opsi berikut.

   Untuk informasi selengkapnya, lihat Bagaimana Amazon ECS menempatkan tugas pada instans kontainer.

   - AZ Balanced Spread — Mendistribusikan tugas di seluruh Availability Zone dan di seluruh instance container di Availability Zone.
   - AZ Balanced BinPack — Mendistribusikan tugas di seluruh Availability Zone dan di seluruh instans kontainer dengan memori yang paling sedikit tersedia.
   - BinPack— Mendistribusikan tugas berdasarkan jumlah CPU atau memori yang paling sedikit tersedia.
   - Satu Tugas Per Host - Tempatkan, paling banyak, satu tugas dari layanan pada setiap instance kontainer.
   - Kustom - Tentukan strategi penempatan tugas Anda sendiri.

Jika Anda memilih Kustom, tentukan algoritme untuk menempatkan tugas dan aturan yang dipertimbangkan selama penempatan tugas.

- Di bawah Strategi, untuk Jenis dan Bidang, pilih algoritma dan entitas yang akan digunakan untuk algoritme.

  Anda dapat memasukkan maksimal 5 strategi.

- Di bawah Constraint, untuk Type dan Expression, pilih aturan dan atribut untuk kendala.

  Misalnya, untuk menyetel batasan untuk menempatkan tugas pada instance T2, untuk Ekspresi, masukkan atribut:ecs.instance-type =~ t2. *.

  Anda dapat memasukkan maksimal 10 kendala.

10. Jika definisi tugas Anda menggunakan mode `awsvpc` jaringan, perluas Jaringan. Gunakan langkah-langkah berikut untuk menentukan konfigurasi kustom.

    a. Untuk VPC, pilih VPC yang akan digunakan.

    b. Untuk Subnet, pilih satu atau beberapa subnet di VPC yang dipertimbangkan oleh penjadwal tugas saat menempatkan tugas Anda.

    > **Important**
    >
    > Hanya subnet pribadi yang didukung untuk mode jaringan `awsvpc`. Tugas tidak menerima alamat IP publik. Oleh karena itu, gateway NAT diperlukan untuk akses internet keluar, dan lalu lintas internet masuk diarahkan melalui penyeimbang beban.

    c. Untuk Grup keamanan, Anda dapat memilih grup keamanan yang sudah ada atau menciptakan grup keamanan baru. Untuk menggunakan grup keamanan yang sudah ada, pilih grup keamanan dan lanjutkan ke langkah selanjutnya. Untuk membuat grup keamanan baru, pilih Buat grup keamanan baru. Anda harus menentukan nama grup keamanan, deskripsi, dan kemudian tambahkan satu atau beberapa aturan masuk untuk grup keamanan.

11. Jika tugas Anda menggunakan volume data yang kompatibel dengan konfigurasi saat penerapan, Anda dapat mengonfigurasi volume dengan memperluas Volume.

Nama volume dan jenis volume dikonfigurasi saat Anda membuat revisi definisi tugas dan tidak dapat diubah saat membuat layanan. Untuk memperbarui nama dan jenis volume, Anda harus membuat revisi definisi tugas baru dan membuat layanan dengan menggunakan revisi baru.

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
Amazon EBS	a. Untuk jenis volume EBS, pilih jenis volume EBS yang ingin Anda lampirkan ke tugas Anda. b. Untuk Ukuran (GiB), masukkan nilai yang valid untuk ukuran volume dalam gibibytes (GiB). Anda dapat menentuka n minimal 1 GiB dan maksimum ukuran volume 16,384 GiB. Nilai ini diperlukan kecuali Anda memberikan ID snapshot. c. Untuk IOPS, masukkan jumlah maksimum operasi input/output (IOPS) yang harus disediakan volume. Nilai ini hanya dapat dikonfigurasi untuk`io1`,`io2`, dan jenis `gp3` volume. d. Untuk Throughput (MIB/ s), masukkan throughpu t yang harus disediakan volume, dalam mebibytes per detik (, atau MIB/ s). MiBps Nilai ini hanya dapat dikonfigurasi untuk jenis `gp3` volume. e. Untuk ID Snapshot, pilih snapshot volume Amazon	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	EBS yang ada atau masukkan ARN snapshot jika Anda ingin membuat volume dari snapshot. Anda juga dapat membuat volume baru yang kosong dengan tidak memilih atau memasukkan ID snapshot. f. Untuk tipe sistem File, pilih jenis sistem file yang akan digunakan untuk penyimpanan data dan pengambilan pada volume. Anda dapat memilih default sistem operasi atau jenis sistem file tertentu. Default untuk Linux adalah`XFS`. Untuk volume yang dibuat dari snapshot, Anda harus menentukan jenis sistem file yang sama dengan volume yang digunakan saat snapshot dibuat. Jika ada ketidakcocokan tipe sistem file, tugas akan gagal dimulai. g. Untuk peran Infrastruktur, pilih peran IAM dengan izin yang diperlukan yang memungkinkan Amazon ECS mengelola volume Amazon EBS	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	untuk tugas. Anda dapat melampirkan kebijakan `AmazonECSInfrastructureRolePolicyForVolumes` terkelola ke peran, atau Anda dapat menggunakan kebijakan sebagai panduan untuk membuat dan melampirkan kebijakan Anda sendiri dengan izin yang memenuhi kebutuhan spesifik Anda. Untuk informasi selengkapnya tentang izin yang diperlukan, lihat Peran IAM infrastruktur Amazon ECS. h. Untuk Enkripsi, pilih Default jika Anda ingin menggunakan enkripsi Amazon EBS dengan pengaturan default. Jika akun Anda memiliki Enkripsi yang dikonfigurasi secara default, volume akan dienkripsi dengan kunci AWS Key Management Service (AWS KMS) yang ditentukan dalam pengaturan. Jika Anda memilih Default dan	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	enkripsi default Amazon EBS tidak diaktifkan, volume tidak akan dienkripsi. Jika Anda memilih Kustom, Anda dapat menentukan AWS KMS key pilihan Anda untuk enkripsi volume. Jika Anda memilih None, volume tidak akan dienkripsi kecuali Anda memiliki enkripsi secara default dikonfigurasi, atau jika Anda membuat volume dari snapshot terenkripsi. i. Jika Anda telah memilih Kustom untuk Enkripsi, Anda harus menentukan AWS KMS key yang ingin Anda gunakan. Untuk kunci KMS, pilih AWS KMS key atau masukkan tombol ARN. Jika Anda memilih untuk mengenkripsi volume dengan menggunakan kunci terkelola pelanggan simetris, pastikan Anda memiliki izin yang tepat yang ditentukan dalam	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	kebijakan Anda. AWS KMS key Untuk informasi selengkapnya, lihat Enkripsi data untuk volume Amazon EBS. j. (Opsional) Di bawah Tag, Anda dapat menambahk an tag ke volume Amazon EBS Anda dengan menyebarkan tag dari definisi tugas atau layanan, atau dengan menyediakan tag Anda sendiri. Jika Anda ingin menyebarkan tag dari definisi tugas, pilih Definisi tugas untuk menyebarkan tag dari. Jika Anda ingin menyebarkan tag dari layanan, pilih Layanan untuk menyebarkan tag dari. Jika Anda memilih Jangan menyebarkan, atau jika Anda tidak memilih nilai, tag tidak disebarkan. Jika Anda ingin memberikan tag Anda sendiri, pilih Tambah tag dan kemudian berikan kunci dan nilai untuk	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	setiap tag yang Anda tambahkan. Untuk informasi selengkapnya tentang menandai volume Amazon EBS, lihat Menandai volume Amazon EBS.	

12. (Opsional) Untuk membantu mengidentifikasi layanan dan tugas Anda, perluas bagian Tag, lalu konfigurasikan tag Anda.

    Agar Amazon ECS secara otomatis menandai semua tugas yang baru diluncurkan dengan nama cluster dan tag definisi tugas, pilih Aktifkan tag terkelola Amazon ECS, lalu untuk menyebarkan tag dari, pilih Definisi tugas.

    Agar Amazon ECS secara otomatis menandai semua tugas yang baru diluncurkan dengan nama cluster dan tag layanan, pilih Aktifkan tag terkelola Amazon ECS, lalu untuk menyebarkan tag dari, pilih Layanan.

    Menambah atau menghapus tanda.

    - [Tambahkan tag] Pilih Tambah tag, lalu lakukan hal berikut:
      - Untuk Kunci, masukkan nama kunci.
      - Untuk Nilai, masukkan nilai kunci.
    - [Menghapus tanda] Di samping tanda, pilih Hapus tanda.

Memperbarui layanan menggunakan konsol

Anda dapat memperbarui layanan Amazon ECS menggunakan konsol Amazon ECS. Konfigurasi layanan saat ini sudah diisi sebelumnya. Anda dapat memperbarui definisi tugas, jumlah tugas yang diinginkan, strategi penyedia kapasitas, versi platform, dan konfigurasi penerapan; atau kombinasi apa pun dari ini.

Untuk informasi tentang cara memperbarui konfigurasi penerapan biru/hijau, lihat. Memperbarui konfigurasi penerapan biru/hijau menggunakan konsol

Pertimbangkan hal berikut saat Anda menggunakan konsol:

Jika Anda ingin menghentikan sementara layanan Anda, atur tugas yang diinginkan ke 0. Kemudian, ketika Anda siap untuk memulai layanan, perbarui layanan dengan jumlah tugas yang diinginkan asli.

Pertimbangkan hal berikut saat Anda menggunakan konsol:

- Anda harus menggunakan AWS Command Line Interface untuk memperbarui layanan yang menggunakan salah satu parameter berikut:
  - Penerapan biru/hijau
  - Service Discovery — Anda hanya dapat melihat konfigurasi Service Discovery Anda.
  - Melacak kebijakan dengan metrik khusus
  - Perbarui Layanan - Anda tidak dapat memperbarui konfigurasi `awsvpc` jaringan dan masa tenggang pemeriksaan kesehatan.

  Untuk informasi tentang cara memperbarui layanan menggunakan AWS CLI, lihat update-servicedi AWS Command Line Interface Referensi.
- Jika Anda mengubah port yang digunakan oleh kontainer dalam definisi tugas, Anda mungkin perlu memperbarui grup keamanan agar instance kontainer berfungsi dengan port yang diperbarui.
- Amazon ECS tidak secara otomatis memperbarui grup keamanan yang terkait dengan penyeimbang beban Elastic Load Balancing atau instans kontainer Amazon ECS.
- Jika layanan Anda menggunakan penyeimbang beban, konfigurasi penyeimbang beban yang ditentukan untuk layanan Anda saat dibuat tidak dapat diubah menggunakan konsol. Sebagai gantinya, Anda dapat menggunakan AWS CLI atau SDK untuk memodifikasi konfigurasi penyeimbang beban. Untuk informasi tentang cara mengubah konfigurasi, lihat UpdateServicedi Referensi API Amazon Elastic Container Service.
- Jika Anda memperbarui definisi tugas untuk layanan, nama kontainer dan port kontainer yang ditentukan dalam konfigurasi penyeimbang beban harus tetap dalam definisi tugas.

Anda dapat memperbarui layanan yang ada untuk mengubah beberapa parameter konfigurasi layanan, seperti jumlah tugas yang dikelola oleh layanan, definisi tugas mana yang digunakan oleh tugas, atau jika tugas Anda menggunakan jenis peluncuran Fargate, Anda dapat mengubah versi platform yang digunakan layanan Anda. Layanan yang menggunakan versi platform Linux tidak dapat diperbarui untuk menggunakan versi platform Windows dan sebaliknya. Jika Anda memiliki aplikasi yang membutuhkan kapasitas lebih besar, maka Anda dapat menaikkan skala layanan. Jika Anda

memiliki kapasitas yang tidak digunakan untuk menurunkan skala, maka Anda dapat mengurangi jumlah tugas yang diinginkan dalam layanan dan mengosongkan sumber daya.

Jika Anda ingin menggunakan citra kontainer yang diperbarui untuk tugas Anda, Anda dapat membuat revisi penentuan tugas baru dengan citra tersebut dan men-deploy-nya ke layanan Anda dengan menggunakan opsi memaksa deployment baru di dalam konsol.

Penjadwal layanan menggunakan parameter persentase minimum dan maksimum yang sehat (dalam konfigurasi deployment untuk layanan) untuk menentukan strategi deployment.

Jika layanan menggunakan tipe deployment (`ECS`) pembaruan bergulir, persentase minimum yang sehat mewakili batas bawah jumlah tugas dalam layanan yang harus tetap berada dalam status `RUNNING` selama deployment, sebagai persentase jumlah tugas yang diinginkan (dibulatkan ke atas ke bilangan bulat terdekat). Parameter ini juga berlaku saat instance kontainer apa pun berada dalam `DRAINING` status jika layanan berisi tugas menggunakan tipe peluncuran EC2. Gunakan parameter ini untuk menyebarkan tanpa menggunakan kapasitas cluster tambahan. Misalnya, jika layanan Anda memiliki jumlah empat tugas yang diinginkan dan persentase sehat minimum 50 persen, penjadwal dapat menghentikan dua tugas yang ada untuk membebaskan kapasitas cluster sebelum memulai dua tugas baru. Tugas untuk layanan yang tidak menggunakan penyeimbang beban dianggap sehat jika berada dalam status `RUNNING`. Tugas untuk layanan yang menggunakan penyeimbang beban dianggap sehat jika berada dalam status `RUNNING` dan dilaporkan sehat oleh penyeimbang beban. Nilai default untuk persen sehat minimum adalah 100 persen.

Jika layanan menggunakan tipe penerapan rolling update (`ECS`), parameter persen maksimum mewakili batas atas jumlah tugas dalam layanan yang diizinkan dalam,, atau `STOPPING` status selama penerapan `PENDINGRUNNING`, sebagai persentase dari jumlah tugas yang diinginkan (dibulatkan ke bilangan bulat terdekat). Parameter ini juga berlaku saat instance kontainer apa pun berada dalam `DRAINING` status jika layanan berisi tugas menggunakan tipe peluncuran EC2. Gunakan parameter ini untuk menentukan ukuran batch deployment. Misalnya, jika layanan Anda memiliki jumlah empat tugas yang diinginkan dan nilai persen maksimum 200 persen, penjadwal dapat memulai empat tugas baru sebelum menghentikan empat tugas lama. Itu asalkan sumber daya cluster yang diperlukan untuk melakukan ini tersedia. Nilai default untuk persen maksimum adalah 200 persen.

Saat penjadwal layanan mengganti tugas selama pembaruan, maka layanan terlebih dahulu menghapus tugas dari penyeimbang beban (jika digunakan) dan menunggu koneksi dialihkan. Kemudian, docker stop yang setara dikeluarkan untuk kontainer yang berjalan dalam tugas. Hal ini menyebabkan sinyal `SIGTERM` dan waktu habis dalam 30 detik, setelah `SIGKILL` dikirim dan kontainer dihentikan secara paksa. Jika kontainer menangani sinyal `SIGTERM` dengan baik dan

keluar dalam waktu 30 detik dari saat menerimanya, maka sinyal `SIGKILL` tidak dikirim. Penjadwal layanan memulai dan menghentikan tugas seperti yang ditentukan melalui pengaturan persentase minimum dan maksimum yang sehat.

Penjadwal layanan juga menggantikan tugas yang ditentukan tidak sehat setelah pemeriksaan kesehatan kontainer atau pemeriksaan kesehatan kelompok sasaran penyeimbang beban gagal. Penggantian ini tergantung pada parameter definisi `maximumPercent` dan `desiredCount` layanan. Jika tugas ditandai tidak sehat, penjadwal layanan akan memulai tugas pengganti terlebih dahulu. Jika tugas penggantian memiliki status kesehatan`HEALTHY`, penjadwal layanan menghentikan tugas yang tidak sehat. Jika tugas penggantian memiliki status kesehatan`UNHEALTHY`, penjadwal akan menghentikan tugas penggantian yang tidak sehat atau tugas tidak sehat yang ada untuk mendapatkan jumlah tugas total yang sama`desiredCount`. Jika `maximumPercent` parameter membatasi penjadwal untuk memulai tugas penggantian terlebih dahulu, penjadwal akan menghentikan tugas yang tidak sehat satu per satu secara acak untuk membebaskan kapasitas, dan kemudian memulai tugas pengganti. Proses start dan stop berlanjut sampai semua tugas yang tidak sehat diganti dengan tugas yang sehat. Setelah semua tugas yang tidak sehat telah diganti dan hanya tugas sehat yang berjalan, jika jumlah tugas total melebihi`desiredCount`, tugas sehat dihentikan secara acak hingga jumlah tugas total sama. `desiredCount` Untuk informasi selengkapnya tentang `maximumPercent` dan`desiredCount`, lihat Parameter definisi layanan.

> **Note**
>
> Perilaku ini tidak berlaku untuk tugas yang dijalankan dan dikelola oleh layanan yang menggunakan jenis penerapan pembaruan bergulir. Selama pembaruan bergulir, penjadwal layanan pertama-tama menghentikan tugas yang tidak sehat dan kemudian memulai tugas penggantian.

> **Important**
>
> Jika Anda mengubah port yang digunakan oleh kontainer dalam penentuan tugas, maka Anda mungkin perlu memperbarui grup keamanan untuk instans kontainer agar berfungsi dengan port yang diperbarui.
> Jika Anda memperbarui ketentuan tugas untuk layanan, nama kontainer dan port kontainer yang ditentukan ketika layanan dibuat harus tetap dalam ketentuan tugas.
> Amazon ECS tidak secara otomatis memperbarui grup keamanan yang terkait dengan penyeimbang beban Elastic Load Balancing atau instans kontainer Amazon ECS.

Untuk memperbarui layanan (konsol Amazon ECS)

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih cluster.
3. Pada halaman detail klaster, di bagian Layanan, pilih kotak centang di sebelah layanan, lalu pilih Perbarui.
4. Agar layanan Anda memulai penerapan baru, pilih Paksa penerapan baru.
5. Untuk definisi Tugas, pilih keluarga definisi tugas dan revisi.

   > **Important**
   >
   > Konsol memvalidasi bahwa keluarga dan revisi definisi tugas yang dipilih kompatibel dengan konfigurasi komputasi yang ditentukan. Jika Anda menerima peringatan, verifikasi kompatibilitas definisi tugas dan konfigurasi komputasi yang Anda pilih.

6. Untuk tugas yang diinginkan, masukkan jumlah tugas yang ingin Anda jalankan untuk layanan.
7. Untuk tugas yang menjalankan Min, masukkan batas bawah pada jumlah tugas dalam layanan yang harus tetap dalam `RUNNING` status selama penerapan, sebagai persentase dari jumlah tugas yang diinginkan (dibulatkan ke bilangan bulat terdekat). Untuk informasi selengkapnya, lihat Konfigurasi penerapan.
8. Untuk tugas yang berjalan Max, masukkan batas atas jumlah tugas dalam layanan yang diizinkan dalam `PENDING` status `RUNNING` atau selama penerapan, sebagai persentase dari jumlah tugas yang diinginkan (dibulatkan ke bilangan bulat terdekat).
9. Untuk mengonfigurasi cara Amazon ECS mendeteksi dan menangani kegagalan penerapan, perluas deteksi kegagalan Deployment, lalu pilih opsi Anda.

   a. Untuk menghentikan penerapan saat tugas tidak dapat dimulai, pilih Gunakan pemutus sirkuit penyebaran Amazon ECS.

      Agar perangkat lunak secara otomatis memutar kembali penerapan ke status penerapan yang terakhir selesai saat pemutus sirkuit penyebaran mengatur penerapan ke status gagal, pilih Rollback on failure.

   b. Untuk menghentikan penerapan berdasarkan metrik aplikasi, pilih Gunakan CloudWatch alarm. Kemudian, dari nama CloudWatch alarm, pilih alarm. Untuk membuat alarm baru, buka CloudWatch konsol.

Agar perangkat lunak secara otomatis memutar kembali penerapan ke status penerapan yang terakhir selesai saat CloudWatch alarm menyetel penerapan ke status gagal, pilih Rollback on failure.

10. Untuk mengubah opsi komputasi, perluas konfigurasi Compute, lalu lakukan hal berikut:

    a. Untuk layanan di AWS Fargate, untuk versi Platform, pilih versi baru.

    b. Untuk layanan yang menggunakan strategi penyedia kapasitas, untuk strategi penyedia Kapasitas, lakukan hal berikut:

    - Untuk menambahkan penyedia kapasitas tambahan, pilih Tambahkan lebih banyak. Kemudian, untuk penyedia Kapasitas, pilih penyedia kapasitas.
    - Untuk menghapus penyedia kapasitas, di sebelah kanan penyedia kapasitas, pilih Hapus.

    Layanan yang menggunakan penyedia kapasitas grup Auto Scaling tidak dapat diperbarui untuk menggunakan penyedia kapasitas Fargate. Layanan yang menggunakan penyedia kapasitas Fargate tidak dapat diperbarui untuk menggunakan penyedia kapasitas grup Auto Scaling.

11. (Opsional) Untuk mengonfigurasi layanan Auto Scaling, perluas Service auto scaling, lalu tentukan parameter berikut.

    a. Untuk menggunakan penskalaan otomatis servis, pilih Penskalaan otomatis layanan.

    b. Untuk Jumlah tugas minimum, masukkan batas bawah jumlah tugas untuk penskalaan otomatis servis yang akan digunakan. Hitungan yang diinginkan tidak akan berada di bawah hitungan ini.

    c. Untuk Jumlah tugas maksimum, masukkan batas atas jumlah tugas untuk penskalaan otomatis servis yang akan digunakan. Hitungan yang diinginkan tidak akan melebihi hitungan ini.

    d. Pilih jenis kebijakan. Di bawah Jenis kebijakan penskalaan, pilih salah satu opsi berikut.

Untuk menggunakan jenis kebijakan ini...	Lakukan ini...	
Pelacakan target	a. Untuk Jenis kebijakan penskalaan, pilih Pelacakan target.	

Untuk menggunakan jenis kebijakan ini...	Lakukan ini...	
	b. Untuk nama Kebijakan , masukkan nama kebijakan. c. Untuk metrik layanan ECS, pilih salah satu metrik berikut. • ECS ServiceAverage CPUUutilization — Pemanfaatan CPU rata-rata layanan. • ECS ServiceAv erageMemoryUtiliza tion — Pemanfaat an memori rata-rata layanan. • ALB RequestCo untPerTarget — Jumlah permintaan yang diselesaikan per target dalam kelompok sasaran Application Load Balancer. d. Untuk nilai Target, masukkan nilai yang dipertahankan layanan untuk metrik yang dipilih. e. Untuk periode cooldown scale-out, masukkan jumlah waktu, dalam detik, setelah aktivitas scale-out (tambahka n tugas) yang harus	

Untuk menggunakan jenis kebijakan ini...	Lakukan ini...	
	dilewati sebelum aktivitas scale-out lainnya dapat dimulai. f. Untuk periode cooldown Scale-in, masukkan jumlah waktu, dalam detik, setelah aktivitas scale-in (hapus tugas) yang harus dilewati sebelum aktivitas scale-in lainnya dapat dimulai. g. Untuk mencegah kebijakan melakukan aktivitas penskalaan, pilih Matikan penskalaan. h. • (Opsional) Pilih Matikan penskalaan jika Anda ingin kebijakan penskalaan Anda ditingkatkan untuk meningkatkan lalu lintas tetapi tidak membutuhkannya untuk menskalakan saat lalu lintas menurun.	

Untuk menggunakan jenis kebijakan ini...	Lakukan ini...	
Penskalaan langkah	a. Untuk Jenis kebijakan penskalaan, pilih Penskalaan langkah. b. Untuk nama Kebijakan , masukkan nama kebijakan. c. Untuk Nama alarm, masukkan nama yang unik untuk alarm. d. Untuk metrik layanan Amazon ECS, pilih metrik yang akan digunakan untuk alarm. e. Untuk Statistik, pilih statistik alarm. f. Untuk Periode, pilih periode untuk alarm. g. Untuk kondisi Alarm, pilih cara membandin gkan metrik yang dipilih dengan ambang batas yang ditentukan. h. Untuk Ambang untuk membandingkan metrik dan periode Evaluasi untuk memulai alarm, masukkan ambang batas yang digunakan untuk alarm dan berapa lama untuk mengevaluasi ambang batas.	

Untuk menggunakan jenis kebijakan ini...	Lakukan ini...	
	i. Di bawah Tindakan penskalaan, lakukan hal berikut: • Untuk Tindakan, pilih apakah akan menambah, menghapus, atau menetapkan jumlah tertentu yang diinginkan untuk layanan Anda. • Jika Anda memilih untuk menambah atau menghapus tugas, untuk Nilai, masukkan jumlah tugas (atau persen tugas yang ada) untuk ditambahkan atau dihapus saat tindakan penskalaan dimulai. Jika Anda memilih untuk mengatur hitungan yang diinginkan, masukkan jumlah tugas. Untuk Jenis, pilih apakah Nilai adalah bilangan bulat atau nilai persen dari jumlah yang diinginkan yang ada. • Untuk batas Bawah dan Batas atas, masukkan batas	

Untuk menggunakan jenis kebijakan ini...	Lakukan ini...	
	bawah dan batas atas penyesuai an penskalaan langkah Anda. Secara default, batas bawah untuk kebijakan penambahan adalah ambang batas alarm, sedangkan batas atas adalah positif (+) tak terbatas. Secara default, batas atas untuk kebijakan penghapusan adalah ambang batas alarm, sedangkan batas bawah adalah negatif (-) tak terbatas. • (Opsional) Tambahkan opsi penskalaan tambahan. Pilih Tambahkan tindakan penskalaan baru, lalu ulangi langkah tindakan Penskalaan. • Untuk periode Cooldown, masukkan jumlah waktu, dalam detik, untuk menunggu aktivitas penskalaan sebelumnya diterapka n. Untuk kebijakan add, ini adalah waktu	

Untuk menggunakan jenis kebijakan ini...	Lakukan ini...	
	setelah aktivitas scale-out yang kebijakan penskalaan memblokir aktivitas scale-in dan membatasi berapa banyak tugas yang dapat diskalakan pada suatu waktu. Untuk kebijakan penghapusan, ini adalah waktu setelah aktivitas penskalaan yang harus diteruskan sebelum aktivitas penskalaan lainnya dapat dimulai.	

12. (Opsional) Untuk menggunakan Service Connect, pilih Aktifkan Service Connect, lalu tentukan yang berikut ini:

    a. Di bawah konfigurasi Service Connect, tentukan mode klien.

    - Jika layanan Anda menjalankan aplikasi klien jaringan yang hanya perlu terhubung ke layanan lain di namespace, pilih sisi Klien saja.
    - Jika layanan Anda menjalankan aplikasi jaringan atau layanan web dan perlu menyediakan titik akhir untuk layanan ini, dan terhubung ke layanan lain di namespace, pilih Klien dan server.

    b. Untuk menggunakan namespace yang bukan namespace cluster default, untuk Namespace, pilih namespace layanan.

13. Jika tugas Anda menggunakan volume data yang kompatibel dengan konfigurasi saat penerapan, Anda dapat mengonfigurasi volume dengan memperluas Volume.

    Nama volume dan jenis volume dikonfigurasi saat Anda membuat revisi definisi tugas dan tidak dapat diubah saat memperbarui layanan. Untuk memperbarui nama dan jenis volume, Anda

harus membuat revisi definisi tugas baru dan memperbarui layanan dengan menggunakan revisi baru.

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
Amazon EBS	a. Untuk jenis volume EBS, pilih jenis volume EBS yang ingin Anda lampirkan ke tugas Anda. b. Untuk Ukuran (GiB), masukkan nilai yang valid untuk ukuran volume dalam gibibytes (GiB). Anda dapat menentuka n minimal 1 GiB dan maksimum ukuran volume 16,384 GiB. Nilai ini diperlukan kecuali Anda memberikan ID snapshot. c. Untuk IOPS, masukkan jumlah maksimum operasi input/output (IOPS) yang harus disediakan volume. Nilai ini hanya dapat dikonfigurasi untuk`io1`,`io2`, dan jenis `gp3` volume. d. Untuk Throughput (MIB/ s), masukkan throughpu t yang harus disediakan volume, dalam mebibytes per detik (, atau MIB/ s). MiBps Nilai ini hanya dapat dikonfigurasi untuk jenis `gp3` volume. e. Untuk ID Snapshot, pilih snapshot volume Amazon	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	EBS yang ada atau masukkan ARN snapshot jika Anda ingin membuat volume dari snapshot. Anda juga dapat membuat volume baru yang kosong dengan tidak memilih atau memasukkan ID snapshot. f. Untuk tipe sistem File, pilih jenis sistem file yang akan digunakan untuk penyimpanan data dan pengambilan pada volume. Anda dapat memilih default sistem operasi atau jenis sistem file tertentu. Default untuk Linux adalah`XFS`. Untuk volume yang dibuat dari snapshot, Anda harus menentukan jenis sistem file yang sama dengan volume yang digunakan saat snapshot dibuat. Jika ada ketidakcocokan tipe sistem file, tugas akan gagal dimulai. g. Untuk peran Infrastruktur, pilih peran IAM dengan izin yang diperlukan yang memungkinkan Amazon ECS mengelola volume Amazon EBS	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	untuk tugas. Anda dapat melampirkan kebijakan `AmazonECSInfrastructureRolePolicyForVolumes` terkelola ke peran, atau Anda dapat menggunakan kebijakan sebagai panduan untuk membuat dan melampirkan kebijakan Anda sendiri dengan izin yang memenuhi kebutuhan spesifik Anda. Untuk informasi selengkapnya tentang izin yang diperlukan, lihat Peran IAM infrastruktur Amazon ECS. h. Untuk Enkripsi, pilih Default jika Anda ingin menggunakan enkripsi Amazon EBS dengan pengaturan default. Jika akun Anda memiliki Enkripsi yang dikonfigurasi secara default, volume akan dienkripsi dengan kunci AWS Key Management Service (AWS KMS) yang ditentukan dalam pengaturan. Jika Anda memilih Default dan	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	enkripsi default Amazon EBS tidak diaktifka n, volume tidak akan dienkripsi. Jika Anda memilih Kustom, Anda dapat menentukan AWS KMS key pilihan Anda untuk enkripsi volume. Jika Anda memilih None, volume tidak akan dienkripsi kecuali Anda memiliki enkripsi secara default dikonfigurasi, atau jika Anda membuat volume dari snapshot terenkripsi. i. Jika Anda telah memilih Kustom untuk Enkripsi, Anda harus menentuka n AWS KMS key yang ingin Anda gunakan. Untuk kunci KMS, pilih AWS KMS key atau masukkan tombol ARN. Jika Anda memilih untuk mengenkripsi volume dengan menggunakan kunci terkelola pelanggan simetris, pastikan Anda memiliki izin yang tepat yang ditentukan dalam	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	kebijakan Anda. AWS KMS key Untuk informasi selengkapnya, lihat Enkripsi data untuk volume Amazon EBS. j. (Opsional) Di bawah Tag, Anda dapat menambahk an tag ke volume Amazon EBS Anda dengan menyebarkan tag dari definisi tugas atau layanan, atau dengan menyediakan tag Anda sendiri. Jika Anda ingin menyebarkan tag dari definisi tugas, pilih Definisi tugas untuk menyebarkan tag dari. Jika Anda ingin menyebarkan tag dari layanan, pilih Layanan untuk menyebarkan tag dari. Jika Anda memilih Jangan menyebarkan, atau jika Anda tidak memilih nilai, tag tidak disebarkan. Jika Anda ingin memberikan tag Anda sendiri, pilih Tambah tag dan kemudian berikan kunci dan nilai untuk	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	setiap tag yang Anda tambahkan. Untuk informasi selengkapnya tentang menandai volume Amazon EBS, lihat Menandai volume Amazon EBS.	

14. (Opsional) Untuk membantu mengidentifikasi layanan Anda, perluas bagian Tag, lalu konfigurasikan tag Anda.
    - [Tambahkan tag] Pilih Tambah tag, dan lakukan hal berikut:
      - Untuk Kunci, masukkan nama kunci.
      - Untuk Nilai, masukkan nilai kunci.
    - [Menghapus tanda] Di samping tanda, pilih Hapus tanda.
15. Pilih Perbarui.

Memperbarui konfigurasi penerapan biru/hijau menggunakan konsol

Anda dapat memperbarui konfigurasi penerapan biru/hijau menggunakan konsol Amazon ECS. Konfigurasi penerapan biru/hijau saat ini sudah diisi sebelumnya. Anda dapat memperbarui opsi penerapan biru/hijau berikut:

- Nama grup penyebaran - Pengaturan CodeDeploy penyebaran
- Nama aplikasi - CodeDeploy Grup penyebaran
- Konfigurasi penerapan - Cara CodeDeploy merutekan lalu lintas produksi ke tugas pengganti yang ditetapkan selama penerapan
- Uji pendengar pada penyeimbang beban - CodeDeploy menggunakan pendengar pengujian untuk merutekan lalu lintas pengujian Anda ke set tugas pengganti selama penerapan

Anda harus mengonfigurasi opsi baru sebelum memperbarui konfigurasi.

Untuk memperbarui konfigurasi penerapan biru/hijau (konsol Amazon ECS)

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih cluster.
3. Pada halaman Ikhtisar cluster, pilih layanan, lalu pilih Perbarui.
4. Perluas opsi Deployment - Didukung oleh CodeDeploy, lalu pilih opsi mana yang akan diperbarui:
   - Untuk memodifikasi grup CodeDeploy penyebaran, untuk nama Aplikasi, pilih grup penyebaran.
   - Untuk mengubah setelan CodeDeploy penerapan, untuk nama grup Deployment, pilih grup.
   - Untuk mengubah cara CodeDeploy merutekan lalu lintas produksi ke tugas pengganti yang ditetapkan selama penerapan, untuk konfigurasi Deployment, pilih opsi.
5. Pilih kait peristiwa siklus hidup penerapan dan fungsi Lambda terkait untuk dijalankan sebagai bagian dari revisi baru penerapan layanan. Kait siklus hidup yang tersedia adalah:
   - BeforeInstall— Gunakan hook peristiwa siklus hidup penerapan ini untuk menjalankan fungsi Lambda sebelum set tugas pengganti dibuat. Hasil dari fungsi Lambda pada peristiwa siklus hidup ini tidak memulai rollback.
   - AfterInstall— Gunakan hook peristiwa siklus hidup penerapan ini untuk menjalankan fungsi Lambda setelah set tugas pengganti dibuat. Hasil dari fungsi Lambda pada peristiwa siklus hidup ini dapat memulai rollback.
   - BeforeAllowTraffic— Gunakan hook peristiwa siklus hidup penerapan ini untuk menjalankan fungsi Lambda sebelum lalu lintas produksi dialihkan ke set tugas pengganti. Hasil dari fungsi Lambda pada peristiwa siklus hidup ini dapat memulai rollback.
   - AfterAllowTraffic— Gunakan hook peristiwa siklus hidup penerapan ini untuk menjalankan fungsi Lambda setelah lalu lintas produksi dialihkan ke set tugas pengganti. Hasil dari fungsi Lambda pada peristiwa siklus hidup ini dapat memulai rollback.
6. Untuk memodifikasi listener pengujian, memperluas Load balancing, lalu untuk Test listener untuk CodeDeploy penerapan, pilih listener pengujian.
7. Pilih Perbarui.

Menghapus layanan menggunakan konsol

Anda dapat menghapus layanan Amazon ECS menggunakan konsol. Layanan secara otomatis diperkecil menjadi nol sebelum dihapus. Sumber daya penyeimbang beban atau sumber daya penemuan layanan yang terkait dengan layanan tidak terpengaruh oleh penghapusan layanan. Untuk menghapus sumber daya Elastic Load Balancing Anda, lihat salah satu topik berikut, tergantung pada jenis penyeimbang beban Anda: Hapus Application Load Balancer atau Hapus Network Load Balancer.

Saat Anda menghapus layanan, jika masih ada tugas yang berjalan yang memerlukan pembersihan, status layanan berpindah dari `ACTIVE` ke`DRAINING`, dan layanan tidak lagi terlihat di konsol atau dalam operasi `ListServices` API. Setelah semua tugas dialihkan, baik ke status `STOPPING` ataupun `STOPPED`, maka status layanan bergerak dari `DRAINING` ke `INACTIVE`. Layanan dalam `INACTIVE` status `DRAINING` atau masih dapat dilihat dengan operasi `DescribeServices` API.

> **Important**
>
> Jika Anda mencoba membuat layanan baru dengan nama yang sama dengan layanan yang ada di salah satu `ACTIVE` atau `DRAINING` status, Anda akan menerima kesalahan.

Untuk menghapus layanan (konsol Amazon ECS)

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih cluster untuk layanan.
3. Pada halaman Clusters, pilih cluster.
4. Pada halaman Cluster: ***name***, pilih tab Services.
5. Pilih layanan, lalu pilih Hapus.
6. Untuk menghapus layanan meskipun tidak diperkecil menjadi nol tugas, pilih Paksa hapus layanan.
7. Pada prompt konfirmasi, masukkan hapus, lalu pilih Hapus.

Jenis Penerapan Amazon ECS

Jenis penyebaran Amazon ECS menentukan strategi penerapan yang digunakan layanan Anda. Ada tiga tipe deployment: pembaruan bergulir, biru/hijau, dan eksternal.

Anda dapat melihat informasi tentang jenis penyebaran layanan di halaman detail layanan, atau dengan menggunakan `describe-services` API. Untuk informasi selengkapnya, lihat DescribeServicesdi Referensi API Amazon Elastic Container Service.

Topik

- Pembaruan bergulir
- Penerapan Biru/Hijau dengan CodeDeploy
- Menggunakan pengontrol pihak ketiga untuk penerapan eksternal Amazon ECS

Pembaruan bergulir

Saat Anda membuat layanan yang menggunakan tipe penerapan rolling update (`ECS`), penjadwal layanan Amazon ECS menggantikan tugas yang sedang berjalan dengan tugas baru. Jumlah tugas yang ditambahkan atau dihapus Amazon ECS dari layanan selama pembaruan bergulir dikendalikan oleh konfigurasi penyebaran layanan. Konfigurasi penyebaran terdiri dari yang berikut:

- `minimumHealthyPercent` mewakili batas bawah pada jumlah tugas yang seharusnya berjalan untuk layanan selama deployment atau saat instans kontainer dikosongkan, sebagai persentase dari jumlah tugas yang diinginkan untuk layanan. Nilai ini dibulatkan ke atas. Sebagai contoh, jika persentase minimum yang sehat adalah `50` dan jumlah tugas yang diinginkan adalah empat, maka penjadwal dapat menghentikan dua tugas yang sudah ada sebelum memulai dua tugas baru. Demikian juga, jika persentase minimum yang sehat adalah 75% dan jumlah tugas yang diinginkan adalah dua, maka penjadwal tidak dapat menghentikan tugas apa pun karena nilai yang dihasilkan juga dua.
- `maximumPercent` mewakili batas atas jumlah tugas yang harus berjalan untuk layanan selama deployment atau saat instans kontainer dikosongkan, sebagai persentase dari jumlah tugas yang diinginkan untuk layanan. Nilai ini dibulatkan ke bawah. Misalnya jika persentase maksimum adalah `200` dan jumlah tugas yang diinginkan adalah empat maka penjadwal dapat memulai empat tugas baru sebelum menghentikan empat tugas yang ada. Demikian juga, jika persentase maksimum adalah `125` dan jumlah tugas yang diinginkan adalah tiga, maka penjadwal tidak dapat memulai tugas apa pun karena nilai yang dihasilkan juga tiga.

> **Important**
>
> Saat menetapkan persen sehat minimum atau persen maksimum, Anda harus memastikan bahwa penjadwal dapat menghentikan atau memulai setidaknya satu tugas saat

penerapan dimulai. Jika layanan Anda memiliki deployment yang macet akibat konfigurasi deployment yang tidak valid, maka pesan kejadian layanan akan dikirimkan. Untuk informasi selengkapnya, lihat Layanan (*nama layanan*) tidak dapat menghentikan atau memulai tugas selama deployment dikarenakan konfigurasi layanan deployment. Perbarui nilai minimumHealthyPercent atau MaximumPercent dan coba lagi..

Penyebaran bergulir menggunakan pemutus sirkuit penyebaran untuk menentukan apakah tugas mencapai kondisi mapan. Pemutus sirkuit penyebaran secara opsional dapat memutar kembali penerapan pada kegagalan.

Metode deteksi kegagalan

Ini adalah dua metode yang menyediakan cara untuk mengidentifikasi dengan cepat kapan penerapan gagal, dan kemudian secara opsional mengembalikan kegagalan ke penerapan kerja terakhir.

- the section called “Pemutus sirkuit penyebaran”
- the section called “CloudWatch alarm”

Metode ini dapat digunakan secara terpisah atau bersama-sama. Ketika kedua metode digunakan, penerapan disetel ke gagal segera setelah kriteria kegagalan untuk salah satu metode kegagalan terpenuhi.

Gunakan panduan berikut untuk membantu menentukan metode mana yang akan digunakan:

- Pemutus sirkuit - Gunakan metode ini saat Anda ingin menghentikan penerapan saat tugas tidak dapat dimulai.
- CloudWatch alarm - Gunakan metode ini ketika Anda ingin menghentikan penyebaran berdasarkan metrik aplikasi.

Untuk informasi tentang praktik terbaik proses penerapan Amazon ECS, lihat Penerapan tugas di Panduan Praktik Terbaik Amazon ECS.

Pemutus sirkuit penyebaran

Pemutus sirkuit penyebaran adalah mekanisme pembaruan bergulir yang menentukan apakah tugas mencapai kondisi mapan. Pemutus sirkuit penyebaran memiliki opsi yang secara otomatis akan memutar kembali penerapan yang gagal ke penerapan yang ada dalam status. `COMPLETED`

Saat penerapan layanan mengubah status, Amazon ECS mengirimkan peristiwa perubahan status penerapan layanan ke. EventBridge Hal ini menyediakan cara terprogram untuk memantau status deployment layanan Anda. Untuk informasi selengkapnya, lihat Acara perubahan status penerapan layanan Amazon ECS. Kami menyarankan Anda membuat dan memantau EventBridge aturan dengan angka `eventName` `SERVICE_DEPLOYMENT_FAILED` sehingga Anda dapat mengambil tindakan manual untuk memulai penerapan Anda. Untuk informasi selengkapnya, lihat Membuat EventBridge Aturan di Panduan EventBridge Pengguna Amazon.

Ketika pemutus sirkuit penyebaran menentukan bahwa penerapan gagal, ia mencari penerapan terbaru yang berada dalam keadaan. `COMPLETED` Ini adalah penerapan yang digunakannya sebagai penerapan roll-back. Saat rollback dimulai, penerapan berubah dari a ke. `COMPLETED` `IN_PROGRESS` Ini berarti bahwa penerapan tidak memenuhi syarat untuk rollback lain hingga mencapai status. `COMPLETED` Ketika pemutus sirkuit penyebaran tidak menemukan penyebaran yang dalam `COMPLETED` keadaan, pemutus sirkuit tidak meluncurkan tugas baru dan penyebaran terhenti.

Saat Anda membuat layanan, penjadwal melacak tugas yang gagal diluncurkan dalam dua tahap.

- Tahap 1 - Penjadwal memantau tugas untuk melihat apakah mereka bertransisi ke status RUNNING.
  - Sukses - Penerapan memiliki peluang untuk beralih ke status SELESAI karena ada lebih dari satu tugas yang dialihkan ke status RUNNING. Kriteria kegagalan dilewati dan pemutus sirkuit bergerak ke tahap 2.
  - Kegagalan - Ada tugas berturut-turut yang tidak bertransisi ke status RUNNING dan penerapan mungkin bertransisi ke status GAGAL.
- Tahap 2 - Penerapan memasuki tahap ini ketika setidaknya ada satu tugas dalam status RUNNING. Pemutus sirkuit memeriksa pemeriksaan kesehatan untuk tugas-tugas dalam penyebaran saat ini yang sedang dievaluasi. Pemeriksaan kesehatan yang divalidasi adalah Elastic Load Balancing AWS Cloud Map , pemeriksaan kesehatan pelayanan, dan pemeriksaan kesehatan kontainer.
  - Sukses - Setidaknya ada satu tugas dalam keadaan berjalan dengan pemeriksaan kesehatan yang telah berlalu.

- Kegagalan - Tugas yang diganti karena kegagalan pemeriksaan kesehatan telah mencapai ambang kegagalan.

Pertimbangkan hal berikut ketika Anda menggunakan metode deployment circuit breaker pada suatu layanan. EventBridgemenghasilkan aturan.

- `DescribeServices`Respons memberikan wawasan tentang keadaan penyebaran, `rolloutState` dan`rolloutStateReason`. Saat deployment baru dimulai, status peluncuran dimulai dalam status `IN_PROGRESS`. Saat layanan mencapai kondisi stabil, status peluncuran bertransisi ke `COMPLETED`. Jika layanan gagal mencapai kondisi tunak dan pemutus sirkuit dihidupkan, penyebaran akan beralih ke keadaan. `FAILED` Penerapan dalam `FAILED` status tidak meluncurkan tugas baru apa pun.
- Selain peristiwa perubahan status penerapan layanan yang dikirim Amazon ECS untuk penerapan yang telah dimulai dan telah selesai, Amazon ECS juga mengirimkan peristiwa ketika penerapan dengan pemutus sirkuit diaktifkan gagal. Kejadian ini menyediakan detail tentang mengapa deployment gagal atau jika deployment dimulai karena rollback. Untuk informasi selengkapnya, lihat Acara perubahan status penerapan layanan Amazon ECS.
- Jika penerapan baru dimulai karena penerapan sebelumnya gagal dan terjadi rollback, `reason` bidang peristiwa perubahan status penerapan layanan menunjukkan penerapan dimulai karena rollback.
- Pemutus sirkuit penyebaran hanya didukung untuk layanan Amazon ECS yang menggunakan pengontrol penyebaran pembaruan bergulir (`ECS`).
- Anda harus menggunakan konsol Amazon ECS, atau AWS CLI ketika Anda menggunakan pemutus sirkuit penyebaran dengan opsi. CloudWatch Untuk informasi selengkapnya, lihat the section called “Buat layanan menggunakan parameter yang ditentukan” dan buat layanan di Referensi.AWS Command Line Interface

`create-service` AWS CLI Contoh berikut menunjukkan cara membuat layanan Linux ketika pemutus sirkuit deployment digunakan dengan opsi rollback.

```
aws ecs create-service \
     --service-name MyService \
     --deployment-controller type=ECS \
     --desired-count 3 \
     --deployment-configuration "deploymentCircuitBreaker={enable=true,rollback=true}"
 \
```

```
    --task-definition sample-fargate:1 \
    --launch-type FARGATE \
    --platform-family LINUX \
    --platform-version 1.4.0 \
    --network-configuration
"awsvpcConfiguration={subnets=[subnet-12344321],securityGroups=[sg-12344321],assignPublicIp=EN
```

Contoh:

Deployment 1 dalam `COMPLETED` keadaan.

Deployment 2 tidak dapat dimulai, sehingga pemutus sirkuit berputar kembali ke Deployment 1. Penerapan 1 transisi ke negara bagian. `IN_PROGRESS`

Deployment 3 dimulai dan tidak ada penerapan di `COMPLETED` status, jadi Deployment 3 tidak dapat memutar kembali, atau meluncurkan tugas.

Ambang batas kegagalan

Pemutus sirkuit penyebaran menghitung nilai ambang batas, dan kemudian menggunakan nilai untuk menentukan kapan harus memindahkan penyebaran ke status. `FAILED`

Pemutus sirkuit penyebaran memiliki ambang minimum 3 dan ambang maksimum 200. dan menggunakan nilai-nilai dalam rumus berikut untuk menentukan kegagalan penerapan.

```
Minimum threshold <= 0.5 * desired task count => maximum threshold
```

Ketika hasil perhitungan lebih besar dari minimum 3, tetapi lebih kecil dari maksimum 200, ambang kegagalan diatur ke ambang batas yang dihitung (dibulatkan ke atas).

> **Note**
>
> Anda tidak dapat mengubah salah satu nilai ambang batas.

Ada dua tahap untuk pemeriksaan status penerapan.

1. Pemutus sirkuit penyebaran memantau tugas-tugas yang merupakan bagian dari penyebaran dan memeriksa tugas-tugas yang ada di negara bagian. `RUNNING` Penjadwal mengabaikan kriteria kegagalan ketika tugas dalam penerapan saat ini dalam `RUNNING` status dan melanjutkan ke tahap berikutnya. Ketika tugas gagal dicapai di `RUNNING` negara bagian, pemutus sirkuit

penyebaran meningkatkan jumlah kegagalan satu per satu. Ketika jumlah kegagalan sama dengan ambang batas, penerapan ditandai sebagai. `FAILED`

2. Tahap ini dimasukkan ketika ada satu atau lebih tugas di `RUNNING` negara bagian. Pemutus sirkuit penyebaran melakukan pemeriksaan kesehatan pada sumber daya berikut untuk tugas-tugas dalam penerapan saat ini:
   - Penyeimbang beban Elastic Load Balancing
   - AWS Cloud Map layanan
   - Pemeriksaan kesehatan wadah Amazon ECS

   Ketika pemeriksaan kesehatan gagal untuk tugas tersebut, pemutus sirkuit penyebaran meningkatkan jumlah kegagalan satu per satu. Ketika jumlah kegagalan sama dengan ambang batas, penerapan ditandai sebagai. `FAILED`

Tabel berikut menunjukkan beberapa contoh.

Hitungan tugas yang diinginkan	Penghitungan	Ambang
1	`3 <= 0.5 * 1 => 200`	3 (nilai yang dihitung kurang dari minimum)
25	`3 <= 0.5 * 25 => 200`	13 (nilainya dibulatkan ke atas)
400	`3 <= 0.5 * 400 => 200`	200
800	`3 <= 0.5 * 800 => 200`	200 (nilai yang dihitung lebih besar dari maksimum)

Misalnya, ketika ambang batas adalah 3, pemutus sirkuit dimulai dengan jumlah kegagalan yang ditetapkan pada 0. Ketika tugas gagal mencapai `RUNNING` status, pemutus sirkuit penyebaran meningkatkan jumlah kegagalan satu per satu. Ketika jumlah kegagalan sama dengan 3, penerapan ditandai sebagai. `FAILED`

Untuk contoh tambahan tentang cara menggunakan opsi rollback, lihat Mengumumkan pemutus sirkuit penyebaran Amazon ECS.

CloudWatch alarm

Anda dapat mengonfigurasi Amazon ECS untuk menyetel penerapan menjadi gagal saat mendeteksi bahwa CloudWatch alarm tertentu telah masuk ke status. `ALARM`

Anda dapat mengatur konfigurasi secara opsional untuk mengembalikan penerapan yang gagal ke penerapan yang terakhir selesai.

`create-service` AWS CLI Contoh berikut menunjukkan cara membuat layanan Linux ketika alarm penyebaran digunakan dengan opsi rollback.

```
aws ecs create-service \
     --service-name MyService \
     --deployment-controller type=ECS \
     --desired-count 3 \
     --deployment-configuration
 "alarms={alarmNames=[alarm1Name,alarm2Name],enable=true,rollback=true}" \
     --task-definition sample-fargate:1 \
     --launch-type FARGATE \
     --platform-family LINUX \
     --platform-version 1.4.0 \
     --network-configuration
 "awsvpcConfiguration={subnets=[subnet-12344321],securityGroups=[sg-12344321],assignPublicIp=EN
```

Pertimbangkan hal berikut saat Anda menggunakan metode CloudWatch alarm Amazon pada suatu layanan.

- Parameter `deploymentConfiguration` permintaan sekarang berisi tipe `alarms` data. Anda dapat menentukan nama alarm, apakah akan menggunakan metode ini, dan apakah akan memulai rollback ketika alarm menunjukkan kegagalan penerapan. Untuk informasi selengkapnya, lihat CreateServicedi Referensi API Amazon Elastic Container Service.
- `DescribeServices`Respons memberikan wawasan tentang keadaan penyebaran, `rolloutState` dan`rolloutStateReason`. Ketika penerapan baru dimulai, status peluncuran dimulai dalam suatu keadaan. `IN_PROGRESS` Ketika layanan mencapai kondisi mapan dan waktu pemanggangan selesai, status peluncuran beralih ke. `COMPLETED` Jika layanan gagal mencapai kondisi mapan dan alarm telah masuk ke `ALARM` status, penyebaran akan beralih ke `FAILED` keadaan. Sebuah deployment dalam status `FAILED` tidak akan meluncurkan tugas baru apa pun.
- Selain peristiwa perubahan status penerapan layanan yang dikirim Amazon ECS untuk penerapan yang telah dimulai dan telah selesai, Amazon ECS juga mengirimkan peristiwa ketika penerapan yang menggunakan alarm gagal. Kejadian ini menyediakan detail tentang mengapa deployment

gagal atau jika deployment dimulai karena rollback. Untuk informasi selengkapnya, lihat Acara perubahan status penerapan layanan Amazon ECS.

- Jika penerapan baru dimulai karena penerapan sebelumnya gagal dan rollback diaktifkan, `reason` bidang peristiwa perubahan status penerapan layanan akan menunjukkan penerapan dimulai karena rollback.
- Jika Anda menggunakan pemutus sirkuit penerapan dan CloudWatch alarm Amazon untuk mendeteksi kegagalan, salah satu dapat memulai kegagalan penerapan segera setelah kriteria untuk salah satu metode terpenuhi. Rollback terjadi ketika Anda menggunakan opsi rollback untuk metode yang memulai kegagalan penerapan.
- CloudWatch Alarm Amazon hanya didukung untuk layanan Amazon ECS yang menggunakan pengontrol penyebaran pembaruan bergulir (`ECS`).
- Anda dapat mengonfigurasi opsi ini dengan menggunakan konsol Amazon ECS, atau. AWS CLIUntuk informasi selengkapnya, lihat the section called “Buat layanan menggunakan parameter yang ditentukan” dan buat layanan di Referensi.AWS Command Line Interface
- Anda mungkin memperhatikan bahwa status penerapan tetap `IN_PROGRESS` untuk waktu yang lama. Alasan untuk ini adalah bahwa Amazon ECS tidak mengubah status sampai telah menghapus penerapan aktif, dan ini tidak terjadi sampai setelah waktu pemanggangan. Bergantung pada konfigurasi alarm Anda, penerapan mungkin tampak memakan waktu beberapa menit lebih lama daripada saat Anda tidak menggunakan alarm (meskipun set tugas utama baru ditingkatkan dan penerapan lama diperkecil). Jika Anda menggunakan CloudFormation batas waktu, pertimbangkan untuk meningkatkan batas waktu. Untuk informasi selengkapnya, lihat Membuat kondisi tunggu di templat di Panduan AWS CloudFormation Pengguna.
- Amazon ECS memanggil `DescribeAlarms` untuk melakukan polling alarm. Panggilan untuk `DescribeAlarms` menghitung kuota CloudWatch layanan yang terkait dengan akun Anda. Jika Anda memiliki AWS layanan lain yang meneleponDescribeAlarms, mungkin ada dampak pada Amazon ECS untuk melakukan polling alarm. Misalnya, jika layanan lain membuat `DescribeAlarms` panggilan yang cukup untuk mencapai kuota, layanan tersebut dibatasi dan Amazon ECS juga dibatasi dan tidak dapat melakukan polling alarm. Jika alarm dihasilkan selama periode pelambatan, Amazon ECS mungkin melewatkan alarm dan putaran kembali mungkin tidak terjadi. Tidak ada dampak lain pada penyebaran. Untuk informasi selengkapnya tentang kuota CloudWatch layanan, lihat kuota CloudWatch layanan di CloudWatch Panduan Pengguna.
- Jika alarm dalam `ALARM` keadaan di awal penerapan, Amazon ECS tidak akan memantau alarm selama penyebaran itu (Amazon ECS mengabaikan konfigurasi alarm). Perilaku ini membahas kasus di mana Anda ingin memulai penerapan baru untuk memperbaiki kegagalan penerapan awal.

Alarm-alarm yang direkomendasikan

Kami menyarankan Anda menggunakan metrik alarm berikut:

- Jika Anda menggunakan Application Load Balancer, gunakan metrik `HTTPCode_ELB_5XX_Count` dan `HTTPCode_ELB_4XX_Count` Application Load Balancer. Metrik ini memeriksa lonjakan HTTP. Untuk informasi selengkapnya tentang metrik Application Load Balancer, lihat CloudWatch metrik untuk Application Load Balancer di Panduan Pengguna untuk Application Load Balancer.
- Jika Anda memiliki aplikasi yang sudah ada, gunakan `MemoryUtilization` metrik `CPUUtilization` dan. Metrik ini memeriksa persentase CPU dan memori yang digunakan cluster atau layanan. Untuk informasi selengkapnya, lihat the section called “Pertimbangan”.
- Jika Anda menggunakan Amazon Simple Queue Service antrian dalam tugas, gunakan metrik Amazon `ApproximateNumberOfMessagesNotVisible` SQS. Metrik ini memeriksa jumlah pesan dalam antrian yang tertunda dan tidak tersedia untuk dibaca segera. Untuk informasi selengkapnya tentang metrik Amazon SQS, lihat Metrik yang tersedia untuk CloudWatch Amazon SQS di Panduan Pengembang Layanan Antrian Sederhana Amazon.

Penerapan Biru/Hijau dengan CodeDeploy

Tipe penyebaran biru/hijau menggunakan model penyebaran biru/hijau yang dikendalikan oleh. CodeDeploy Gunakan jenis penerapan ini untuk memverifikasi penyebaran layanan baru sebelum mengirim lalu lintas produksi ke layanan tersebut. Untuk informasi selengkapnya, lihat Apa yang Ada CodeDeploy di Panduan AWS CodeDeploy Pengguna.

Ada tiga cara lalu lintas dapat bergeser selama penyebaran biru/hijau:

- Canary — Lalu lintas digeser dalam dua peningkatan. Anda dapat memilih dari opsi Canary yang telah ditentukan sebelumnya yang menentukan persentase lalu lintas yang dialihkan ke set tugas yang diperbarui pada kenaikan pertama dan interval, dalam menit, sebelum lalu lintas yang tersisa dialihkan pada kenaikan kedua.
- Linear — Lalu lintas digeser dalam peningkatan yang sama dengan jumlah menit yang sama antara setiap kenaikan. Anda dapat memilih dari opsi linier yang telah ditentukan sebelumnya yang menentukan persentase lalu lintas yang dialihkan pada setiap kenaikan dan jumlah menit di antara setiap kenaikan.
- A ll-at-once - Semua lalu lintas digeser dari set tugas asli ke set tugas yang diperbarui sekaligus.

Berikut ini adalah komponen CodeDeploy yang digunakan Amazon ECS saat layanan menggunakan jenis penyebaran biru/hijau:

CodeDeploy aplikasi

Kumpulan sumber CodeDeploy daya. Ini terdiri dari satu atau beberapa grup deployment.

CodeDeploy kelompok penyebaran

Pengaturan deployment. Pengaturan tersebut terdiri dari hal-hal berikut:

- Cluster dan layanan Amazon ECS
- Grup target penyeimbang beban dan informasi listener
- Strategi roll back deployment
- Pengaturan perutean ulang lalu lintas
- Pengaturan pengakhiran revisi asli
- Konfigurasi deployment
- CloudWatch konfigurasi alarm yang dapat diatur untuk menghentikan penerapan
- Pengaturan SNS atau CloudWatch Acara untuk notifikasi

Untuk informasi selengkapnya, lihat Bekerja dengan Grup Penerapan di Panduan AWS CodeDeploy Pengguna.

CodeDeploy konfigurasi penerapan

Menentukan cara CodeDeploy merutekan lalu lintas produksi ke tugas pengganti Anda yang ditetapkan selama penerapan. Konfigurasi deployment canary dan linier yang telah ditentukan berikut ini tersedia. Anda juga dapat membuat deployment canary dan linier yang ditentukan secara khusus. Untuk informasi selengkapnya, lihat Bekerja dengan Konfigurasi Penerapan di AWS CodeDeploy Panduan Pengguna.

- CodeDeployDefault.ECS AllAtOnce: Menggeser semua lalu lintas ke wadah Amazon ECS yang diperbarui sekaligus
- CodeDeployDefault.ecslinear10 PercentEvery 1Menit: Menggeser 10 persen lalu lintas setiap menit sampai semua lalu lintas bergeser.
- CodeDeployDefault.ecslinear10 PercentEvery 3Minutes: Menggeser 10 persen lalu lintas setiap 3 menit sampai semua lalu lintas bergeser.
- CodeDeployDefault.ecscanary10percent5minutes: Menggeser 10 persen lalu lintas pada kenaikan pertama. Sisanya 90 persen dikerahkan lima menit kemudian.

- CodeDeployDefault.ecscanary10Percent15minutes: Menggeser 10 persen lalu lintas pada kenaikan pertama. Sisa 90 persen di-deploy 15 menit kemudian.

Revisi

Revisi adalah file spesifikasi CodeDeploy aplikasi (AppSpec file). Dalam AppSpec file tersebut, Anda menentukan ARN lengkap dari definisi tugas dan wadah serta port set tugas pengganti Anda di mana lalu lintas akan dirutekan saat penerapan baru dibuat. Nama kontainer harus merupakan salah satu dari nama kontainer yang direferensikan dalam penentuan tugas Anda. Jika konfigurasi jaringan atau versi platform telah diperbarui dalam definisi layanan, Anda juga harus menentukan detail tersebut dalam AppSpec file. Anda juga dapat menentukan fungsi Lambda yang akan dijalankan selama peristiwa siklus hidup penerapan. Fungsi Lambda memungkinkan Anda menjalankan pengujian dan mengembalikan metrik selama penerapan. Untuk informasi selengkapnya, lihat Referensi AppSpec File di Panduan AWS CodeDeploy Pengguna.

Pertimbangan Deployment Biru/Hijau

Pertimbangkan hal berikut saat menggunakan tipe deployment biru/hijau:

- Saat layanan Amazon ECS yang menggunakan jenis penyebaran biru/hijau awalnya dibuat, kumpulan tugas Amazon ECS dibuat.
- Anda harus mengonfigurasi layanan untuk menggunakan Application Load Balancer atau Network Load Balancer. Berikut ini adalah persyaratan penyeimbang beban:
  - Anda harus menambahkan listener produksi ke penyeimbang beban, yang digunakan untuk merutekan lalu lintas produksi.
  - Listener uji opsional dapat ditambahkan ke penyeimbang beban, yang digunakan untuk merutekan lalu lintas uji. Jika Anda menentukan listener pengujian, CodeDeploy rute lalu lintas pengujian Anda ke set tugas pengganti selama penerapan.
  - Baik listener produksi maupun listener uji, harus memiliki penyeimbang beban yang sama.
  - Anda harus menentukan grup target untuk penyeimbang beban. Grup target merutekan lalu lintas ke set tugas asli yang di layanan melalui listener produksi.
  - Ketika Network Load Balancer digunakan, hanya konfigurasi `CodeDeployDefault.ECSAllAtOnce` penerapan yang didukung.
- Untuk layanan yang dikonfigurasi untuk menggunakan penskalaan otomatis layanan dan tipe deployment biru/hijau, penskalaan otomatis tidak akan diblokir selama deployment, akan

tetapi deployment mungkin akan mengalami kegagalan di beberapa keadaan. Hal-hal berikut menjelaskan perilaku tersebut secara lebih detail.

- Jika layanan sedang diskalakan dan penerapan dimulai, set tugas hijau dibuat dan CodeDeploy akan menunggu hingga satu jam untuk tugas hijau yang disetel mencapai kondisi tunak dan tidak akan mengalihkan lalu lintas apa pun hingga selesai.
- Jika layanan sedang dalam proses deployment biru/hijau dan kejadian penskalaan terjadi, maka lalu lintas akan terus beralih selama 5 menit. Jika layanan tidak mencapai kondisi tunak dalam 5 menit, CodeDeploy akan menghentikan penerapan dan menandainya sebagai gagal.
- Jika layanan sedang dalam proses penerapan biru/hijau dan peristiwa penskalaan terjadi, jumlah tugas yang diinginkan mungkin disetel ke nilai yang tidak terduga. Hal ini disebabkan oleh penskalaan otomatis yang mempertimbangkan jumlah tugas yang berjalan sebagai kapasitas saat ini, yang merupakan dua kali jumlah tugas yang sesuai yang digunakan dalam perhitungan hitungan tugas yang diinginkan.

- Tugas yang menggunakan tipe peluncuran Fargate atau tipe pengontrol `CODE_DEPLOY` penerapan tidak mendukung strategi penjadwalan. `DAEMON`
- Saat Anda awalnya membuat grup CodeDeploy aplikasi dan penyebaran, Anda harus menentukan yang berikut:
  - Anda harus menentukan dua grup target untuk penyeimbang beban. Satu grup target harus menjadi grup target awal yang ditentukan untuk penyeimbang beban saat layanan Amazon ECS dibuat. Satu-satunya persyaratan grup target kedua adalah tidak dapat dikaitkan dengan penyeimbang beban yang berbeda dari yang digunakan oleh layanan.
- Saat Anda membuat CodeDeploy penerapan untuk layanan Amazon ECS, CodeDeploy buat set tugas pengganti (atau set tugas hijau) dalam penerapan. Jika Anda menambahkan pendengar pengujian ke penyeimbang beban, CodeDeploy rutekan lalu lintas pengujian Anda ke set tugas pengganti. Ini adalah saat Anda dapat menjalankan uji validasi apa pun. Kemudian CodeDeploy merutekan ulang lalu lintas produksi dari set tugas asli ke set tugas pengganti sesuai dengan pengaturan perutean ulang lalu lintas untuk grup deployment.

Penerapan biru/hijau memerlukan izin IAM

Penerapan biru/hijau Amazon ECS dimungkinkan oleh kombinasi Amazon ECS dan API. CodeDeploy Pengguna harus memiliki izin yang sesuai untuk layanan ini sebelum mereka dapat menggunakan penerapan biru/hijau Amazon ECS di atau dengan SDK atau. AWS Management Console AWS CLI

Selain izin IAM standar untuk membuat dan memperbarui layanan, Amazon ECS memerlukan izin berikut. Izin ini telah ditambahkan ke kebijakan `AmazonECS_FullAccess` IAM. Untuk informasi selengkapnya, lihat Amazonecs_ FullAccess.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codedeploy:CreateApplication",
                "codedeploy:CreateDeployment",
                "codedeploy:CreateDeploymentGroup",
                "codedeploy:GetApplication",
                "codedeploy:GetDeployment",
                "codedeploy:GetDeploymentGroup",
                "codedeploy:ListApplications",
                "codedeploy:ListDeploymentGroups",
                "codedeploy:ListDeployments",
                "codedeploy:StopDeployment",
                "codedeploy:GetDeploymentTarget",
                "codedeploy:ListDeploymentTargets",
                "codedeploy:GetDeploymentConfig",
                "codedeploy:GetApplicationRevision",
                "codedeploy:RegisterApplicationRevision",
                "codedeploy:BatchGetApplicationRevisions",
                "codedeploy:BatchGetDeploymentGroups",
                "codedeploy:BatchGetDeployments",
                "codedeploy:BatchGetApplications",
                "codedeploy:ListApplicationRevisions",
                "codedeploy:ListDeploymentConfigs",
                "codedeploy:ContinueDeployment",
                "sns:ListTopics",
                "cloudwatch:DescribeAlarms",
                "lambda:ListFunctions"
            ],
            "Resource": ["*"]
        }
    ]
}
```

> **Note**
>
> Selain izin Amazon ECS standar yang diperlukan untuk menjalankan tugas dan layanan, pengguna juga memerlukan `iam:PassRole` izin untuk menggunakan peran IAM untuk tugas.

CodeDeploy memerlukan izin untuk memanggil Amazon ECS API, memodifikasi Elastic Load Balancing, menjalankan fungsi Lambda, dan CloudWatch menjelaskan alarm, serta izin untuk mengubah jumlah layanan yang diinginkan atas nama Anda. Sebelum membuat layanan Amazon ECS yang menggunakan jenis penyebaran biru/hijau, Anda harus membuat peran IAM (). `ecsCodeDeployRole` Untuk informasi selengkapnya, lihat Peran Amazon ECS CodeDeploy IAM.

Contoh kebijakan Perbarui contoh layanan IAM Buat contoh layanan dan IAM menunjukkan izin yang diperlukan bagi pengguna untuk menggunakan penerapan biru/hijau Amazon ECS di file. AWS Management Console

Menggunakan pengontrol pihak ketiga untuk penerapan eksternal Amazon ECS

Jenis penyebaran eksternal memungkinkan Anda menggunakan pengontrol penyebaran pihak ketiga untuk kontrol penuh atas proses penyebaran untuk layanan Amazon ECS. Detail untuk layanan Anda dikelola, baik oleh tindakan API manajemen layanan (`CreateService`, `UpdateService`, dan `DeleteService`) ataupun tindakan API manajemen set tugas (`CreateTaskSet`, `UpdateTaskSet`, `UpdateServicePrimaryTaskSet`, dan `DeleteTaskSet`). Setiap tindakan API mengelola subset parameter definisi layanan.

Tindakan API `UpdateService` memperbarui jumlah yang diinginkan dan parameter masa tenggang pemeriksaan kondisi untuk suatu layanan. Jika tipe peluncuran, versi platform, detail penyeimbang beban, konfigurasi jaringan, atau penentuan tugas perlu diperbarui, maka Anda harus membuat set tugas baru.

Tindakan API `UpdateTaskSet` hanya memperbarui parameter skala untuk set tugas.

Tindakan API `UpdateServicePrimaryTaskSet` memodifikasi set tugas mana dalam layanan yang merupakan set tugas utama. Saat Anda memanggil tindakan API `DescribeServices`, tindakan tersebut mengembalikan semua bidang yang ditentukan untuk set tugas utama. Jika set tugas utama untuk layanan diperbarui, maka nilai parameter set tugas apa pun yang ada di set tugas utama baru yang berbeda dari set tugas utama lama di layanan akan diperbarui ke nilai baru saat set tugas

utama baru ditentukan. Jika tidak ada set tugas utama yang ditentukan untuk layanan, maka saat menjelaskan layanan, bidang set tugas adalah nol.

Pertimbangan penyebaran eksternal

Pertimbangkan hal berikut saat menggunakan tipe deployment eksternal:

- Jenis penyeimbang beban yang didukung adalah Application Load Balancer atau Penyeimbang Beban Jaringan.
- Jenis peluncuran Fargate atau tipe pengontrol `EXTERNAL` penerapan tidak mendukung strategi penjadwalan. `DAEMON`

Alur kerja penerapan eksternal

Berikut ini adalah alur kerja dasar untuk mengelola penyebaran eksternal di Amazon ECS.

Untuk mengelola layanan Amazon ECS menggunakan pengontrol penyebaran eksternal

1. Buat layanan Amazon ECS. Satu-satunya parameter yang diperlukan adalah nama layanan. Anda dapat menentukan parameter berikut saat membuat layanan menggunakan pengendali deployment eksternal. Semua parameter layanan lainnya ditentukan saat membuat set tugas dalam layanan.

   `serviceName`

   Tipe: String

   Diperlukan: Ya

   Nama layanan Anda. Mengizinkan hingga 255 huruf (huruf besar dan huruf kecil), angka, tanda hubung, dan garis bawah. Nama layanan harus unik dalam sebuah klaster, tetapi Anda dapat memiliki layanan yang bernama sama di beberapa klaster dalam satu Wilayah atau lebih.

   `desiredCount`

   Jumlah instantiasi dari tugas yang ditentukan menetapkan penentuan tugas untuk ditempatkan dan tetap berjalan dalam layanan.

`deploymentConfiguration`

Parameter deployment opsional yang mengendalikan seberapa banyak tugas yang dijalankan selama deployment dan urutan penghentian serta mulainya tugas. Untuk informasi selengkapnya, lihat deploymentConfiguration.

`tags`

Tipe: Array objek

Diperlukan: Tidak

Metadata yang Anda terapkan ke layanan untuk membantu Anda mengkategorikan dan mengaturnya. Setiap tanda terdiri dari kunci dan nilai opsional, yang keduanya Anda tentukan. Ketika layanan dihapus, tag akan dihapus juga. Maksimal 50 tag dapat diterapkan ke layanan. Untuk informasi selengkapnya, lihat Penandaan sumber daya Amazon ECS.

`key`

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Diperlukan: Tidak

Satu bagian dari pasangan nilai kunci yang membentuk tanda. Kunci adalah label umum yang bertindak seperti kategori untuk nilai tanda yang lebih spesifik.

`value`

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 256.

Diperlukan: Tidak

Bagian opsional pasangan nilai kunci yang membentuk tanda. Nilai bertindak sebagai deskriptor dalam kategori tanda (kunci).

`enableECSManagedTags`

Menentukan apakah akan menggunakan tag terkelola Amazon ECS untuk tugas dalam layanan. Untuk informasi selengkapnya, lihat Menandai sumber daya Anda untuk penagihan.

`propagateTags`

Tipe: String

Nilai yang valid: `TASK_DEFINITION` | `SERVICE`

Diperlukan: Tidak

Menentukan apakah akan menyalin tag dari definisi tugas atau layanan untuk tugas-tugas dalam layanan. Jika tidak ada nilai yang ditentukan, tag tidak disalin. Tag hanya dapat disalin ke tugas dalam layanan selama pembuatan layanan. Untuk menambahkan tag ke tugas setelah pembuatan layanan atau pembuatan tugas, gunakan tindakan `TagResource` API.

`healthCheckGracePeriodSeconds`

Tipe: Integer

Wajib: Tidak

Periode waktu, dalam hitungan detik, bahwa penjadwal layanan Amazon ECS harus mengabaikan pemeriksaan kesehatan target Elastic Load Balancing yang tidak sehat, pemeriksaan kesehatan kontainer, dan pemeriksaan kesehatan Route 53 setelah tugas memasuki suatu keadaan. `RUNNING` Ini hanya berlaku jika layanan Anda dikonfigurasi untuk menggunakan penyeimbang beban. Jika layanan Anda memiliki penyeimbang beban yang ditentukan dan Anda tidak menentukan nilai tenggang pemeriksaan kesehatan, nilai default akan `0` digunakan.

Jika tugas layanan Anda membutuhkan waktu cukup lama untuk memulai dan menanggapi pemeriksaan kesehatan, Anda dapat menentukan masa tenggang pemeriksaan kesehatan hingga 2.147.483.647 detik selama penjadwal layanan ECS mengabaikan status pemeriksaan kesehatan. Masa tenggang ini dapat mencegah penjadwal layanan ECS menandai tugas sebagai tidak sehat dan menghentikannya sebelum mereka punya waktu untuk muncul.

Jika Anda tidak menggunakan Elastic Load Balancing, kami sarankan Anda menggunakan `startPeriod` parameter pemeriksaan kesehatan definisi tugas. Untuk informasi lebih lanjut, lihat Pemeriksaan Kesehatan.

`schedulingStrategy`

Strategi penjadwalan yang akan digunakan. Layanan yang menggunakan pengendali deployment eksternal hanya mendukung strategi penjadwalan `REPLICA`. Untuk informasi selengkapnya, lihat Konsep penjadwal layanan.

`placementConstraints`

Susunan objek batasan penempatan yang akan digunakan untuk tugas di layanan Anda. Anda dapat menentukan maksimal 10 batasan per tugas (batas ini mencakup batasan dalam penentuan tugas dan batasan yang ditentukan pada waktu aktif). Jika Anda menggunakan tipe peluncuran Fargate, batasan penempatan tugas tidak didukung.

`placementStrategy`

Strategi batasan objek yang akan digunakan untuk tugas-tugas dalam layanan Anda. Anda dapat menentukan maksimal empat aturan strategi per layanan.

Berikut ini adalah contoh penentuan layanan untuk membuat layanan menggunakan pengendali deployment eksternal.

```
{
    "cluster": "",
    "serviceName": "",
    "desiredCount": 0,
    "role": "",
    "deploymentConfiguration": {
        "maximumPercent": 0,
        "minimumHealthyPercent": 0
    },
    "placementConstraints": [
        {
            "type": "distinctInstance",
            "expression": ""
        }
    ],
    "placementStrategy": [
        {
            "type": "binpack",
            "field": ""
        }
    ],
```

```
    "healthCheckGracePeriodSeconds": 0,
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "EXTERNAL"
    },
    "tags": [
        {
            "key": "",
            "value": ""
        }
    ],
    "enableECSManagedTags": true,
    "propagateTags": "TASK_DEFINITION"
}
```

2. Buat set tugas awal. Set tugas berisi detail berikut tentang layanan Anda:

`taskDefinition`

Penentuan tugas untuk tugas-tugas dalam set tugas yang akan digunakan.

`launchType`

Tipe: String

Nilai yang valid: `EC2` | `FARGATE` | `EXTERNAL`

Diperlukan: Tidak

Jenis peluncuran untuk menjalankan layanan Anda. Jika jenis peluncuran tidak ditentukan, default `capacityProviderStrategy` digunakan secara default. Untuk informasi selengkapnya, lihat Jenis peluncuran Amazon ECS.

Jika `launchType` ditentukan, parameter `capacityProviderStrategy` harus dihilangkan.

`platformVersion`

Tipe: String

Wajib: Tidak

Versi platform tempat tugas Anda dalam layanan berjalan. Versi platform hanya ditentukan untuk tugas yang menggunakan tipe peluncuran Fargate. Jika tidak ditentukan, versi terbaru (`LATEST`) digunakan secara default.

AWS Versi platform Fargate digunakan untuk merujuk ke lingkungan runtime tertentu untuk infrastruktur tugas Fargate. Saat menentukan versi `LATEST` platform saat menjalankan tugas atau membuat layanan, Anda mendapatkan versi platform terbaru yang tersedia untuk tugas Anda. Saat Anda meningkatkan layanan Anda, tugas-tugas tersebut menerima versi platform yang ditentukan pada penerapan layanan saat ini. Untuk informasi selengkapnya, lihat Versi platform Fargate Linux.

> **Note**
>
> Versi platform tidak ditentukan untuk tugas yang menggunakan tipe peluncuran EC2.

`loadBalancers`

Objek penyeimbang beban yang mewakili penyeimbang beban yang akan digunakan bersama layanan Anda. Saat menggunakan pengontrol penyebaran eksternal, hanya Application Load Balancer dan Network Load Balancer yang didukung. Jika Anda menggunakan Application Load Balancer, hanya satu grup target Application Load Balancer yang diizinkan per set tugas.

Cuplikan berikut menunjukkan contoh `loadBalancer` objek untuk digunakan.

```
"loadBalancers": [
        {
            "targetGroupArn": "",
            "containerName": "",
            "containerPort": 0
        }
]
```

> **Note**
>
> Saat menentukan objek `loadBalancer`, Anda harus menentukan `targetGroupArn` dan menghilangkan parameter `loadBalancerName`.

`networkConfiguration`

Konfigurasi jaringan untuk layanan. Parameter ini diperlukan untuk penentuan tugas yang menggunakan mode jaringan `awsvpc` untuk menerima antarmuka jaringan elastisnya sendiri, dan tidak didukung untuk mode jaringan lainnya. Untuk informasi selengkapnya tentang jaringan untuk jenis peluncuran Amazon EC2, lihat Jaringan Tugas Fargate. Untuk informasi selengkapnya tentang jaringan untuk jenis peluncuran Fargate, lihat. Jaringan tugas untuk tugas-tugas di Fargate

`serviceRegistries`

Detail registri penemuan layanan yang akan ditetapkan ke layanan ini. Untuk informasi selengkapnya, lihat Penemuan Layanan .

`scale`

Persentase titik mengambang dari jumlah tugas yang diinginkan untuk menempatkan dan agar tetap berjalan di serangkaian tugas. Nilai ditetapkan sebagai total persentase dari `desiredCount` layanan. Nilai yang diterima adalah angka antara 0 hingga 100.

Berikut ini adalah contoh JSON untuk membuat set tugas untuk pengendali deployment eksternal.

```
{
    "service": "",
    "cluster": "",
    "externalId": "",
    "taskDefinition": "",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": [
                ""
            ],
            "securityGroups": [
                ""
            ],
            "assignPublicIp": "DISABLED"
        }
    },
    "loadBalancers": [
```

```
        {
            "targetGroupArn": "",
            "containerName": "",
            "containerPort": 0
        }
    ],
    "serviceRegistries": [
        {
            "registryArn": "",
            "port": 0,
            "containerName": "",
            "containerPort": 0
        }
    ],
    "launchType": "EC2",
    "capacityProviderStrategy": [
        {
            "capacityProvider": "",
            "weight": 0,
            "base": 0
        }
    ],
    "platformVersion": "",
    "scale": {
        "value": null,
        "unit": "PERCENT"
    },
    "clientToken": ""
}
```

3. Saat perubahan layanan dibutuhkan, gunakan tindakan API `UpdateService`, `UpdateTaskSet`, atau `CreateTaskSet` tergantung pada parameter mana yang Anda perbarui. Jika Anda membuat set tugas, maka gunakan parameter `scale` untuk setiap set tugas di layanan untuk menentukan berapa banyak tugas yang harus tetap berjalan di layanan. Misalnya, jika Anda memiliki layanan yang berisi `tasksetA` dan Anda membuat `tasksetB`, maka Anda dapat menguji validitas `tasksetB` sebelum akan menransisikan lalu lintas produksi ke sana. Anda dapat mengatur `scale` untuk kedua set tugas ke `100`, dan saat Anda siap untuk menransisikan semua lalu lintas produksi ke `tasksetB`, Anda dapat memperbarui `scale` untuk `tasksetA` ke `0` untuk menurunkan skalanya.

Mendistribusikan lalu lintas layanan Amazon ECS menggunakan load balancing

Layanan Amazon ECS Anda secara opsional dapat dikonfigurasi untuk menggunakan Elastic Load Balancing untuk mendistribusikan lalu lintas secara merata di seluruh tugas dalam layanan Anda.

> **Note**
>
> Bila Anda menggunakan set tugas, semua tugas dalam set harus dikonfigurasi untuk menggunakan Elastic Load Balancing atau untuk tidak menggunakan Elastic Load Balancing.

Layanan Amazon ECS yang di-host AWS Fargate mendukung jenis penyeimbang beban Application Load Balancer dan Network Load Balancer. Application Load Balancers digunakan untuk merutekan lalu lintas HTTP/HTTPS (atau layer 7). Network Load Balancer digunakan untuk merutekan lalu lintas TCP atau UDP (atau layer 4).

Application Load Balancers menawarkan beberapa fitur yang membuatnya menarik untuk digunakan dengan layanan Amazon ECS:

- Setiap layanan dapat melayani lalu lintas dari beberapa penyeimbang beban dan mengekspos beberapa port yang diseimbangkan beban dengan menentukan beberapa grup target.
- Mereka didukung oleh tugas yang dihosting di instance Fargate dan EC2.
- Application Load Balancers memungkinkan kontainer untuk menggunakan pemetaan port host dinamis (sehingga beberapa tugas dari layanan yang sama diperbolehkan per instance kontainer).
- Application Load Balancers mendukung routing berbasis jalur dan aturan prioritas (sehingga beberapa layanan dapat menggunakan port listener yang sama pada satu Application Load Balancer).

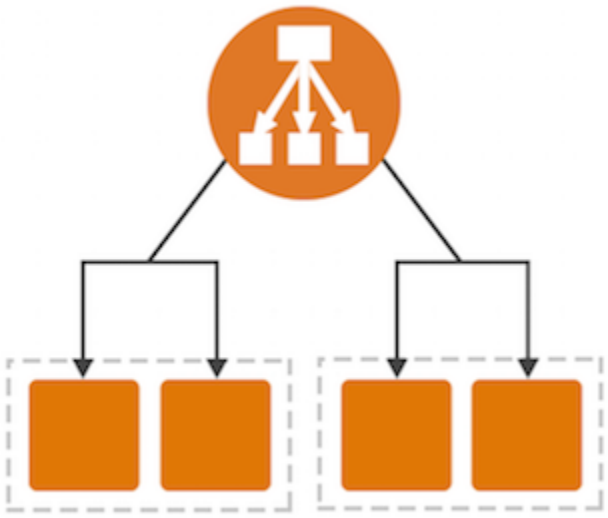
Kami menyarankan Anda menggunakan Application Load Balancer untuk layanan Amazon ECS Anda sehingga Anda dapat memanfaatkan fitur-fitur terbaru ini, kecuali layanan Anda memerlukan fitur yang hanya tersedia dengan Network Load Balancers. Untuk informasi selengkapnya tentang Elastic Load Balancing dan perbedaan antara jenis load balancer, lihat Panduan Pengguna Elastic Load Balancing.

Dengan penyeimbang beban, Anda hanya membayar apa yang Anda gunakan. Untuk informasi selengkapnya, lihat Harga Elastic Load Balancing.

Topik

Menggunakan Application Load Balancer untuk Amazon ECS

Application Load Balancer membuat keputusan routing di layer aplikasi (HTTP/HTTPS), mendukung routing berbasis jalur, dan dapat merutekan permintaan ke satu atau beberapa port pada setiap instance container di cluster Anda. Aplikasi Load Balancer mendukung pemetaan port host dinamis. Misalnya, jika definisi kontainer tugas Anda menentukan port 80 untuk port kontainer NGINX, dan port 0 untuk port host, maka port host dipilih secara dinamis dari rentang port sementara instance kontainer (seperti 32768 hingga 61000 pada AMI Amazon ECS terbaru yang dioptimalkan). Ketika tugas diluncurkan, kontainer NGINX terdaftar dengan Application Load Balancer sebagai ID instance dan kombinasi port, dan lalu lintas didistribusikan ke ID instance dan port yang sesuai dengan container tersebut. Pemetaan dinamis ini mengizinkan Anda memiliki banyak tugas dari satu layanan pada instans kontainer yang sama. Untuk informasi selengkapnya, lihat Panduan Pengguna untuk Penyeimbang Beban Aplikasi.



Pertimbangan Application Load Balancer

Pertimbangan berikut khusus untuk layanan Amazon ECS yang menggunakan Application Load Balancers atau Network Load Balancer:

- Amazon ECS memerlukan peran IAM terkait layanan yang menyediakan izin yang diperlukan untuk mendaftarkan dan membatalkan pendaftaran target dengan penyeimbang beban Anda saat tugas dibuat dan dihentikan. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS.
- Untuk layanan yang menggunakan Application Load Balancer atau Penyeimbang Beban Jaringan, Anda tidak dapat melampirkan lebih dari lima grup target ke layanan.
- Untuk layanan dengan tugas menggunakan mode `awsvpc` jaringan, saat Anda membuat grup target untuk layanan Anda, Anda harus memilih `ip` sebagai jenis target, bukan`instance`. Ini karena tugas yang menggunakan mode `awsvpc` jaringan dikaitkan dengan elastic network interface, bukan instans Amazon EC2.
- Jika layanan Anda menggunakan Application Load Balancer dan memerlukan akses ke beberapa port load balanced, seperti port 80 dan port 443 untuk layanan HTTP/HTTPS, Anda dapat mengonfigurasi dua listener. Satu listener bertanggung jawab untuk HTTPS yang meneruskan permintaan ke layanan, dan listener lainnya bertanggung jawab untuk mengarahkan permintaan HTTP ke port HTTPS yang sesuai. Untuk informasi selengkapnya, lihat Membuat pendengar ke Application Load Balancer di Panduan Pengguna untuk Application Load Balancers.
- Konfigurasi subnet penyeimbang beban Anda harus menyertakan semua Availability Zone tempat instans kontainer Anda berada.
- Setelah Anda membuat layanan, konfigurasi penyeimbang beban tidak dapat diubah dari. AWS Management Console Anda dapat menggunakan AWS Copilot, AWS CloudFormation, AWS CLI atau SDK untuk memodifikasi konfigurasi penyeimbang beban hanya untuk pengontrol penerapan `ECS` bergulir, bukan biru/hijau atau eksternal. AWS CodeDeploy Saat Anda menambahkan, memperbarui, atau menghapus konfigurasi penyeimbang beban, Amazon ECS memulai penerapan baru dengan konfigurasi Elastic Load Balancing yang diperbarui. Hal ini menyebabkan tugas mendaftar dan membatalkan pendaftaran dari penyeimbang beban. Kami menyarankan Anda memverifikasi ini di lingkungan pengujian sebelum memperbarui konfigurasi Elastic Load Balancing. Untuk informasi tentang cara mengubah konfigurasi, lihat UpdateServicedi Referensi API Amazon Elastic Container Service.
- Jika tugas layanan gagal dalam kriteria pemeriksaan kesehatan penyeimbang beban, tugas dihentikan dan dimulai ulang. Proses ini berlanjut hingga layanan Anda mencapai jumlah tugas berjalan yang diinginkan.
- Saat Anda menggunakan Network Load Balancer yang dikonfigurasi dengan alamat IP sebagai target dan Pelestarian IP Klien dinonaktifkan, permintaan akan terlihat berasal dari alamat IP pribadi Network Load Balancers. Ini berarti bahwa layanan di balik Network Load Balancer secara

efektif terbuka untuk dunia segera setelah Anda mengizinkan permintaan masuk dan pemeriksaan kesehatan di grup keamanan target.

- Menggunakan Network Load Balancer untuk merutekan lalu lintas UDP ke tugas Amazon ECS Anda di Fargate memerlukan tugas untuk menggunakan versi platform `1.4.0` (Linux) atau (Windows). `1.0.0`
- Minimalkan kesalahan dalam aplikasi klien Anda dengan menyetel definisi tugas lebih lama dari penundaan deregistrasi grup target, yang seharusnya lebih lama dari batas waktu koneksi klien Anda. `StopTimeout` Lihat Builders Library untuk informasi selengkapnya tentang konfigurasi klien yang direkomendasikan di sini.

  Selain itu, atribut grup target Network Load Balancer untuk penghentian koneksi menutup semua koneksi yang tersisa setelah waktu deregistrasi. Ini dapat menyebabkan klien menampilkan pesan kesalahan yang tidak diinginkan, jika klien tidak menanganinya.
- Jika Anda mengalami masalah dengan layanan yang diaktifkan penyeimbang beban, lihat Pemecahan permasalahan terhadap layanan penyeimbang beban.
- Tugas dan penyeimbang beban Anda (Application Load Balancer atau Network Load Balancer) harus dalam VPC yang sama.
- Pelestarian alamat IP klien Network Load Balancer juga kompatibel dengan target Fargate.
- Gunakan grup target unik untuk setiap layanan.

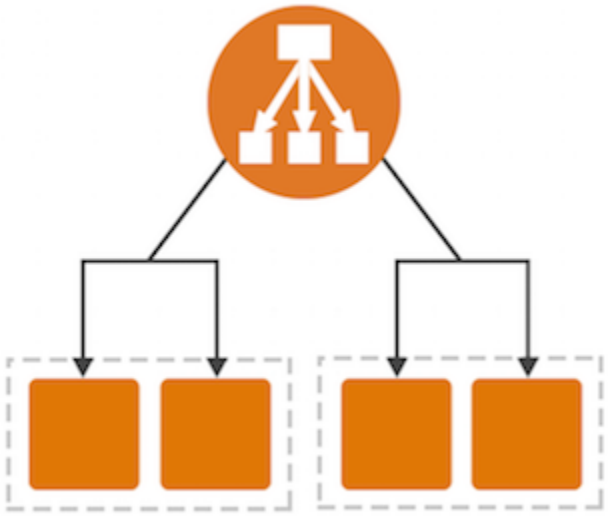
  Menggunakan grup target yang sama untuk beberapa layanan dapat menyebabkan masalah selama penerapan layanan.

Untuk selengkapnya tentang cara membuat Application Load Balancer, lihat Membuat Application Load Balancer di Application Load Balancers

Menggunakan Network Load Balancer untuk Amazon ECS

Network Load Balancer membuat keputusan routing pada layer transport (TCP/SSL). Hal itu dapat menangani jutaan permintaan per detik. Setelah penyeimbang beban menerima koneksi, penyeimbang akan memilih target dari grup target untuk aturan default menggunakan algoritme perutean hash alur. Penyeimbang mencoba untuk membuka koneksi TCP ke target yang dipilih pada port yang ditentukan dalam konfigurasi listener. Ini meneruskan permintaan tanpa memodifikasi header. Network Load Balancers mendukung pemetaan port host dinamis. Misalnya, jika definisi kontainer tugas Anda menentukan port 80 untuk port kontainer NGINX, dan port 0 untuk port host, maka port host dipilih secara dinamis dari rentang port sementara instance kontainer (seperti

32768 hingga 61000 pada AMI Amazon ECS terbaru yang dioptimalkan). Ketika tugas diluncurkan, kontainer NGINX terdaftar dengan Network Load Balancer sebagai ID instance dan kombinasi port, dan lalu lintas didistribusikan ke ID instance dan port yang sesuai dengan container tersebut. Pemetaan dinamis ini mengizinkan Anda memiliki banyak tugas dari satu layanan pada instans kontainer yang sama. Untuk informasi selengkapnya, lihat Panduan Pengguna untuk Network Load Balancer.



Pertimbangan Network Load Balancer

Pertimbangan berikut khusus untuk layanan Amazon ECS yang menggunakan Application Load Balancers atau Network Load Balancer:

- Amazon ECS memerlukan peran IAM terkait layanan yang menyediakan izin yang diperlukan untuk mendaftarkan dan membatalkan pendaftaran target dengan penyeimbang beban Anda saat tugas dibuat dan dihentikan. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS.
- Untuk layanan yang menggunakan Application Load Balancer atau Penyeimbang Beban Jaringan, Anda tidak dapat melampirkan lebih dari lima grup target ke layanan.
- Untuk layanan dengan tugas menggunakan mode `awsvpc` jaringan, saat Anda membuat grup target untuk layanan Anda, Anda harus memilih `ip` sebagai jenis target, bukan`instance`. Ini karena tugas yang menggunakan mode `awsvpc` jaringan dikaitkan dengan elastic network interface, bukan instans Amazon EC2.
- Jika layanan Anda menggunakan Application Load Balancer dan memerlukan akses ke beberapa port load balanced, seperti port 80 dan port 443 untuk layanan HTTP/HTTPS, Anda dapat

mengonfigurasi dua listener. Satu listener bertanggung jawab untuk HTTPS yang meneruskan permintaan ke layanan, dan listener lainnya bertanggung jawab untuk mengarahkan permintaan HTTP ke port HTTPS yang sesuai. Untuk informasi selengkapnya, lihat Membuat pendengar ke Application Load Balancer di Panduan Pengguna untuk Application Load Balancers.

- Konfigurasi subnet penyeimbang beban Anda harus menyertakan semua Availability Zone tempat instans kontainer Anda berada.
- Setelah Anda membuat layanan, konfigurasi penyeimbang beban tidak dapat diubah dari. AWS Management Console Anda dapat menggunakan AWS Copilot, AWS CloudFormation, AWS CLI atau SDK untuk memodifikasi konfigurasi penyeimbang beban hanya untuk pengontrol penerapan `ECS` bergulir, bukan biru/hijau atau eksternal. AWS CodeDeploy Saat Anda menambahkan, memperbarui, atau menghapus konfigurasi penyeimbang beban, Amazon ECS memulai penerapan baru dengan konfigurasi Elastic Load Balancing yang diperbarui. Hal ini menyebabkan tugas mendaftar dan membatalkan pendaftaran dari penyeimbang beban. Kami menyarankan Anda memverifikasi ini di lingkungan pengujian sebelum memperbarui konfigurasi Elastic Load Balancing. Untuk informasi tentang cara mengubah konfigurasi, lihat UpdateServicedi Referensi API Amazon Elastic Container Service.
- Jika tugas layanan gagal dalam kriteria pemeriksaan kesehatan penyeimbang beban, tugas dihentikan dan dimulai ulang. Proses ini berlanjut hingga layanan Anda mencapai jumlah tugas berjalan yang diinginkan.
- Saat Anda menggunakan Network Load Balancer yang dikonfigurasi dengan alamat IP sebagai target dan Pelestarian IP Klien dinonaktifkan, permintaan akan terlihat berasal dari alamat IP pribadi Network Load Balancers. Ini berarti bahwa layanan di balik Network Load Balancer secara efektif terbuka untuk dunia segera setelah Anda mengizinkan permintaan masuk dan pemeriksaan kesehatan di grup keamanan target.
- Menggunakan Network Load Balancer untuk merutekan lalu lintas UDP ke tugas Amazon ECS Anda di Fargate memerlukan tugas untuk menggunakan versi platform `1.4.0` (Linux) atau (Windows). `1.0.0`
- Minimalkan kesalahan dalam aplikasi klien Anda dengan menyetel definisi tugas lebih lama dari penundaan deregistrasi grup target, yang seharusnya lebih lama dari batas waktu koneksi klien Anda. `StopTimeout` Lihat Builders Library untuk informasi selengkapnya tentang konfigurasi klien yang direkomendasikan di sini.

  Selain itu, atribut grup target Network Load Balancer untuk penghentian koneksi menutup semua koneksi yang tersisa setelah waktu deregistrasi. Ini dapat menyebabkan klien menampilkan pesan kesalahan yang tidak diinginkan, jika klien tidak menanganinya.

- Jika Anda mengalami masalah dengan layanan yang diaktifkan penyeimbang beban, lihat Pemecahan permasalahan terhadap layanan penyeimbang beban.
- Tugas dan penyeimbang beban Anda (Application Load Balancer atau Network Load Balancer) harus dalam VPC yang sama.
- Pelestarian alamat IP klien Network Load Balancer juga kompatibel dengan target Fargate.
- Gunakan grup target unik untuk setiap layanan.

  Menggunakan grup target yang sama untuk beberapa layanan dapat menyebabkan masalah selama penerapan layanan.

Untuk selengkapnya tentang cara membuat Network Load Balancer, lihat Membuat Network Load Balancer di Network Load Balancer

> **Important**
>
> Jika definisi tugas layanan Anda menggunakan mode `awsvpc` jaringan (yang diperlukan untuk jenis peluncuran Fargate), Anda harus memilih `ip` sebagai jenis target, bukan. `instance` Ini karena tugas yang menggunakan mode `awsvpc` jaringan dikaitkan dengan elastic network interface, bukan instans Amazon EC2.
> Anda tidak dapat mendaftarkan instans berdasarkan ID instans jika mereka memiliki tipe instans berikut: C1, CC1, CC2, CG1, CG2, CR1, G1, G2, HI1, HS1, M1, M2, M3, dan T1. Anda dapat mendaftarkan instans tipe ini berdasarkan alamat IP.

Mendaftarkan beberapa grup target dengan layanan

Layanan Amazon ECS Anda dapat melayani lalu lintas dari beberapa penyeimbang beban dan mengekspos beberapa port seimbang beban saat Anda menentukan beberapa grup target dalam definisi layanan.

Untuk membuat layanan yang menentukan beberapa grup target, Anda harus membuat layanan menggunakan Amazon ECS API, SDK AWS CLI, atau templat. AWS CloudFormation Setelah layanan dibuat, Anda dapat melihat layanan dan grup target yang terdaftar dengan AWS Management Console. Anda harus menggunakan `UpdateService` untuk memodifikasi konfigurasi penyeimbang beban dari layanan yang ada.

Beberapa grup target dapat ditentukan dalam penentuan layanan menggunakan format berikut. Untuk sintaksis lengkap penentuan layanan, lihat Templat definisi layanan.

```
"loadBalancers":[
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_1/1234567890123456",
      "containerName":"container_name",
      "containerPort":container_port
   },
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_2/6543210987654321",
      "containerName":"container_name",
      "containerPort":container_port
   }
]
```

Pertimbangan beberapa grup target

Hal-hal berikut harus dipertimbangkan saat Anda menentukan beberapa grup target dalam penentuan layanan.

- Untuk layanan yang menggunakan Application Load Balancer atau Penyeimbang Beban Jaringan, Anda tidak dapat melampirkan lebih dari lima grup target ke layanan.
- Penentuan beberapa grup target dalam penentuan layanan hanya didukung dalam kondisi berikut:
  - Layanan harus menggunakan Application Load Balancer atau Network Load Balancer.
  - Layanan harus menggunakan tipe pengendali deployment (`ECS`) pembaruan bergulir.
- Menentukan beberapa grup target didukung untuk layanan yang berisi tugas menggunakan jenis peluncuran Fargate dan EC2.
- Saat membuat layanan yang menentukan beberapa grup target, peran terkait layanan Amazon ECS harus dibuat. Peran dibuat dengan menghilangkan parameter `role` dalam permintaan API, atau properti `Role` di AWS CloudFormation. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS.

Contoh penentuan layanan

Berikut adalah beberapa contoh kasus penggunaan untuk menentukan beberapa grup target dalam penentuan layanan. Untuk sintaksis penentuan layanan lengkap, lihat Templat definisi layanan.

Contoh: Memiliki penyeimbang beban terpisah untuk lalu lintas internal dan eksternal

Dalam kasus penggunaan berikut, layanan menggunakan dua penyeimbang beban terpisah, satu untuk lalu lintas internal dan yang kedua untuk lalu lintas yang berhadapan dengan internet, untuk kontainer dan port yang sama.

```
"loadBalancers":[
   //Internal ELB
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_1/1234567890123456",
      "containerName":"nginx",
      "containerPort":8080
   },
   //Internet-facing ELB
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_2/6543210987654321",
      "containerName":"nginx",
      "containerPort":8080
   }
]
```

Contoh: Mengekspos beberapa port dari kontainer yang sama

Dalam kasus penggunaan berikut, layanan menggunakan satu penyeimbang beban, akan tetapi mengekspos beberapa port dari kontainer yang sama. Misalnya, kontainer Jenkins mungkin mengekspos port 8080 untuk antarmuka web Jenkins dan port 50000 untuk API.

```
"loadBalancers":[
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_1/1234567890123456",
      "containerName":"jenkins",
      "containerPort":8080
   },
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_2/6543210987654321",
```

```
      "containerName":"jenkins",
      "containerPort":50000
   }
]
```

Contoh: Mengekspos port dari beberapa kontainer

Dalam kasus penggunaan berikut, layanan menggunakan satu penyeimbang beban dan dua grup target untuk mengekspos port dari kontainer terpisah.

```
"loadBalancers":[
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_1/1234567890123456",
      "containerName":"webserver",
      "containerPort":80
   },
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_2/6543210987654321",
      "containerName":"database",
      "containerPort":3306
   }
]
```

Secara otomatis menskalakan layanan Amazon ECS Anda

Penskalaan otomatis adalah kemampuan untuk menambah atau mengurangi jumlah tugas yang diinginkan di layanan Amazon ECS Anda secara otomatis. Amazon ECS memanfaatkan layanan Application Auto Scaling untuk menyediakan fungsionalitas ini. Untuk informasi selengkapnya, lihat Panduan Pengguna Application Auto Scaling.

Amazon ECS menerbitkan CloudWatch metrik dengan penggunaan CPU dan memori rata-rata layanan Anda. Untuk informasi selengkapnya, lihat Pemanfaatan layanan. Anda dapat menggunakan metrik ini dan CloudWatch metrik lainnya untuk meningkatkan skala layanan Anda (menambahkan lebih banyak tugas) untuk menangani permintaan tinggi pada waktu puncak, dan untuk meningkatkan skala dalam layanan Anda (menjalankan lebih sedikit tugas) untuk mengurangi biaya selama periode pemanfaatan rendah.

Amazon ECS Service Auto Scaling mendukung jenis penskalaan otomatis berikut:

- Skala layanan Amazon ECS Anda menggunakan nilai metrik target— Menambah atau mengurangi jumlah tugas yang dijalankan layanan Anda berdasarkan nilai target untuk metrik tertentu. Hal ini mirip dengan cara termostat mempertahankan suhu di rumah Anda. Anda mengatur suhu dan termostat melakukan sisanya.
- Skalakan layanan Amazon ECS Anda menggunakan kenaikan yang telah ditentukan berdasarkan alarm CloudWatch Meningkatkan atau mengurangi jumlah tugas yang dijalankan layanan Anda berdasarkan serangkaian penyesuaian penskalaan, yang dikenal sebagai penyesuaian langkah, yang bervariasi berdasarkan ukuran pelanggaran alarm.
- Skala layanan Amazon ECS Anda menggunakan jadwalMeningkatkan atau mengurangi jumlah tugas yang dijalankan layanan Anda berdasarkan tanggal dan waktu.

Pertimbangan

Saat menggunakan kebijakan penskalaan, pertimbangkan hal berikut:

- Amazon ECS mengirimkan metrik dalam interval 1 menit ke. CloudWatch Metrik tidak tersedia sampai kluster dan layanan mengirimkan metrik CloudWatch, dan Anda tidak dapat membuat CloudWatch alarm untuk metrik yang tidak ada.
- Kebijakan penskalaan mendukung periode pendinginan. Ini adalah jumlah detik untuk menunggu hingga aktivitas penskalaan sebelumnya berlaku.
  - Dengan kebijakan penskalaan, tujuannya adalah untuk terus menerus (tetapi tidak berlebihan) menskalakan naik. Setelah Service Auto Scaling berhasil diskalakan menggunakan kebijakan penskalaan, maka mulai menghitung waktu cooldown. Kebijakan penskalaan tidak akan meningkatkan kapasitas yang diinginkan lagi kecuali jika skala yang lebih besar dimulai atau periode cooldown berakhir. Selama periode pendinginan penskalaan keluar berlaku, kapasitas yang ditambahkan dengan cara memulai aktivitas penskalaan keluar dihitung sebagai bagian dari kapasitas yang diinginkan untuk aktivitas penskalaan keluar berikutnya.
  - Untuk kejadian penskalaan kedalam, tujuannya adalah untuk melakukan penskalaan kedalam secara konservatif guna melindungi ketersediaan aplikasi Anda, sehingga aktivitas penskalaan kedalam diblokir hingga periode pendinginan berakhir. Namun, jika alarm lain memulai aktivitas scale-out selama periode cooldown scale-in, Service Auto Scaling segera menskalakan target. Dalam hal ini, periode pendinginan penskalaan kedalam berhenti dan tidak selesai.

- Penjadwal layanan menghormati jumlah yang diinginkan setiap saat, tetapi selama Anda memiliki kebijakan penskalaan aktif dan alarm pada layanan, Service Auto Scaling dapat mengubah hitungan yang diinginkan yang ditetapkan secara manual oleh Anda.
- Jika jumlah yang diinginkan layanan ditetapkan di bawah nilai kapasitas minimumnya, dan alarm memulai aktivitas scale-out, Service Auto Scaling menskalakan jumlah yang diinginkan hingga nilai kapasitas minimum dan kemudian melanjutkan skala sesuai kebutuhan, berdasarkan kebijakan penskalaan yang terkait dengan alarm. Namun, kegiatan penskalaan kedalam tidak menyesuaikan jumlah yang diinginkan, karena sudah di bawah nilai kapasitas minimal.
- Jika jumlah yang diinginkan layanan ditetapkan di atas nilai kapasitas maksimumnya, dan alarm memulai skala dalam aktivitas, Auto Scaling Service menskalakan hitungan yang diinginkan ke nilai kapasitas maksimum dan kemudian melanjutkan penskalaan sesuai kebutuhan, berdasarkan kebijakan penskalaan yang terkait dengan alarm. Namun, aktivitas penskalaan keluar tidak menyesuaikan jumlah yang diinginkan, karena sudah di atas nilai kapasitas maksimal.
- Selama aktivitas penskalaan, jumlah tugas yang sebenarnya berjalan dalam layanan adalah nilai yang digunakan Service Auto Scaling sebagai titik awalnya, sebagai lawan dari hitungan yang diinginkan. Inilah yang seharusnya menjadi kapasitas pemrosesan. Ini mencegah penskalaan berlebihan (runaway) yang mungkin tidak terpenuhi, misalnya, jika tidak ada sumber daya instance kontainer yang cukup untuk menempatkan tugas tambahan. Jika nantinya kapasitas instans kontainer tersedia, maka aktivitas penskalaan yang tertunda kemungkinan akan berhasil, dan kemudian aktivitas penskalaan lebih lanjut dapat diteruskan setelah periode pendinginan.
- Jika Anda ingin jumlah tugas Anda menjadi nol ketika tidak ada pekerjaan yang harus dilakukan, tetapkan kapasitas minimum 0. Dengan kebijakan penskalaan pelacakan target, ketika kapasitas aktual adalah 0 dan metrik menunjukkan bahwa ada permintaan beban kerja, Service Auto Scaling menunggu satu titik data dikirim sebelum penskalaan keluar. Dalam hal ini, penskalaan otomatis layanan akan menskalakan keluar berdasarkan jumlah minimal yang memungkinkan sebagai titik awal, dan kemudian melanjutkan penskalaan berdasarkan jumlah tugas berjalan yang aktual.
- Application Auto Scaling menonaktifkan proses scale-in saat penerapan Amazon ECS sedang berlangsung. Namun, proses penskalaan keluar terus terjadi, kecuali ditangguhkan, selama deployment. Untuk informasi selengkapnya, lihat penskalaan otomatis dan deployment layanan.
- Anda memiliki beberapa opsi Application Auto Scaling untuk tugas Amazon ECS. Pelacakan target adalah mode termudah untuk digunakan. Dengan itu, yang perlu Anda lakukan adalah menetapkan nilai target untuk metrik, seperti pemanfaatan rata-rata CPU. Kemudian, auto scaler secara otomatis mengelola jumlah tugas yang diperlukan untuk mencapai nilai itu. Dengan penskalaan langkah, Anda dapat bereaksi lebih cepat terhadap perubahan permintaan, karena Anda menentukan ambang batas tertentu untuk metrik penskalaan Anda, dan berapa banyak tugas

yang harus ditambahkan atau dihapus saat ambang batas dilewati. Dan, yang lebih penting, Anda dapat bereaksi sangat cepat terhadap perubahan permintaan dengan meminimalkan jumlah waktu alarm ambang batas dilanggar.

Praktik terbaik untuk penskalaan otomatis layanan Amazon ECS

Layanan Amazon ECS adalah kumpulan tugas yang dikelola. Setiap layanan memiliki definisi tugas terkait, jumlah tugas yang diinginkan, dan strategi penempatan opsional. Penskalaan otomatis layanan Amazon ECS diimplementasikan melalui layanan Application Auto Scaling. Application Auto Scaling menggunakan CloudWatch metrik sebagai sumber untuk menskalakan metrik. Ini juga menggunakan CloudWatch alarm untuk menetapkan ambang batas kapan harus menskalakan layanan Anda masuk atau keluar. Anda memberikan ambang batas untuk penskalaan, baik dengan menetapkan target metrik, disebut sebagai penskalaan pelacakan target, atau dengan menentukan ambang batas, yang disebut sebagai penskalaan langkah. Setelah Application Auto Scaling dikonfigurasi, ia terus menghitung jumlah tugas yang diinginkan yang sesuai untuk layanan. Ini juga memberi tahu Amazon ECS ketika jumlah tugas yang diinginkan harus berubah, baik dengan menskalakannya atau menskalakannya.

Untuk menggunakan penskalaan otomatis servis secara efektif, Anda harus memilih metrik penskalaan yang sesuai.

Aplikasi harus ditingkatkan jika permintaan diperkirakan lebih besar dari kapasitas saat ini. Sebaliknya, aplikasi dapat ditingkatkan untuk menghemat biaya ketika sumber daya melebihi permintaan.

Identifikasi metrik

Untuk menskalakan secara efektif, penting untuk mengidentifikasi metrik yang menunjukkan pemanfaatan atau saturasi. Metrik ini harus menunjukkan properti berikut agar berguna untuk penskalaan.

- Metrik harus berkorelasi dengan permintaan. Ketika sumber daya tetap stabil, tetapi permintaan berubah, nilai metrik juga harus berubah. Metrik harus meningkat atau menurun ketika permintaan meningkat atau menurun.
- Nilai metrik harus diskalakan sebanding dengan kapasitas. Ketika permintaan tetap konstan, menambahkan lebih banyak sumber daya harus menghasilkan perubahan proporsional dalam nilai metrik. Jadi, menggandakan jumlah tugas akan menyebabkan metrik berkurang 50%.

Cara terbaik untuk mengidentifikasi metrik pemanfaatan adalah melalui pengujian beban di lingkungan pra-produksi seperti lingkungan pementasan. Solusi pengujian beban komersial dan sumber terbuka tersedia secara luas. Solusi ini biasanya dapat menghasilkan beban sintetis atau mensimulasikan lalu lintas pengguna nyata.

Untuk memulai proses pengujian beban, buat dasbor untuk metrik pemanfaatan aplikasi Anda. Metrik ini termasuk pemanfaatan CPU, pemanfaatan memori, operasi I/O, kedalaman antrian I/O, dan throughput jaringan. Anda dapat mengumpulkan metrik ini dengan layanan seperti Wawasan Kontainer. Untuk informasi selengkapnya, lihat Pantau kontainer Amazon ECS menggunakan Wawasan Kontainer. Selama proses ini, pastikan Anda mengumpulkan dan memplot metrik untuk waktu respons aplikasi atau tingkat penyelesaian pekerjaan Anda.

Mulailah dengan permintaan kecil atau tingkat penyisipan pekerjaan. Pertahankan kecepatan ini stabil selama beberapa menit untuk memungkinkan aplikasi Anda memanas. Kemudian, perlahan-lahan tingkatkan laju dan tahan selama beberapa menit. Ulangi siklus ini, tingkatkan laju setiap kali hingga waktu respons atau penyelesaian aplikasi Anda terlalu lambat untuk memenuhi tujuan tingkat layanan (SLO) Anda.

Saat pengujian beban, periksa masing-masing metrik pemanfaatan. Metrik yang meningkat seiring dengan beban adalah kandidat teratas untuk dijadikan metrik pemanfaatan terbaik Anda.

Selanjutnya, identifikasi sumber daya yang mencapai saturasi. Pada saat yang sama, periksa juga metrik pemanfaatan untuk melihat mana yang rata pada tingkat tinggi terlebih dahulu, atau mencapai puncak dan kemudian crash aplikasi Anda terlebih dahulu. Misalnya, jika pemanfaatan CPU meningkat dari 0% menjadi 70-80% saat Anda menambahkan beban, maka tetap pada tingkat itu setelah lebih banyak beban ditambahkan, maka aman untuk mengatakan bahwa CPU jenuh. Tergantung pada arsitektur CPU, mungkin tidak akan pernah mencapai 100%. Misalnya, asumsikan bahwa pemanfaatan memori meningkat saat Anda menambahkan beban, dan kemudian aplikasi Anda tiba-tiba macet saat mencapai tugas atau batas memori instans Amazon EC2. Dalam situasi ini, kemungkinan memori telah dikonsumsi sepenuhnya. Beberapa sumber daya dapat dikonsumsi oleh aplikasi Anda. Oleh karena itu, pilih metrik yang mewakili sumber daya yang habis terlebih dahulu.

Terakhir, coba uji muat lagi setelah menggandakan jumlah tugas atau instans Amazon EC2. Asumsikan bahwa metrik kunci meningkat, atau menurun, pada setengah tingkat seperti sebelumnya. Jika ini masalahnya, maka metrik sebanding dengan kapasitas. Ini adalah metrik pemanfaatan yang baik untuk penskalaan otomatis.

Sekarang pertimbangkan skenario hipotetis ini. Misalkan Anda memuat pengujian aplikasi dan menemukan bahwa pemanfaatan CPU akhirnya mencapai 80% pada 100 permintaan per detik.

Ketika lebih banyak beban ditambahkan, itu tidak membuat pemanfaatan CPU meningkat lagi. Namun, itu membuat aplikasi Anda merespons lebih lambat. Kemudian, Anda menjalankan uji beban lagi, menggandakan jumlah tugas tetapi menahan laju pada nilai puncak sebelumnya. Jika Anda menemukan penggunaan CPU rata-rata turun menjadi sekitar 40%, maka pemanfaatan CPU rata-rata adalah kandidat yang baik untuk metrik penskalaan. Di sisi lain, jika pemanfaatan CPU tetap pada 80% setelah meningkatkan jumlah tugas, maka pemanfaatan CPU rata-rata bukanlah metrik penskalaan yang baik. Dalam hal ini, diperlukan lebih banyak penelitian untuk menemukan metrik yang sesuai.

Model aplikasi umum dan properti penskalaan

Perangkat lunak dari semua jenis dijalankan AWS. Banyak beban kerja yang homegrown, sedangkan yang lain didasarkan pada perangkat lunak open-source populer. Terlepas dari mana asalnya, kami telah mengamati beberapa pola desain umum untuk layanan. Cara menskalakan secara efektif sebagian besar tergantung pada pola.

Server CPU-bound yang efisien

Server CPU-bound yang efisien menggunakan hampir tidak ada sumber daya selain CPU dan throughput jaringan. Setiap permintaan dapat ditangani oleh aplikasi saja. Permintaan tidak bergantung pada layanan lain seperti database. Aplikasi ini dapat menangani ratusan ribu permintaan bersamaan, dan secara efisien dapat memanfaatkan beberapa CPU untuk melakukannya. Setiap permintaan dilayani oleh thread khusus dengan overhead memori rendah, atau ada loop peristiwa asinkron yang berjalan pada setiap CPU yang diminta layanan. Setiap replika aplikasi sama-sama mampu menangani permintaan. Satu-satunya sumber daya yang mungkin habis sebelum CPU adalah bandwidth jaringan. Dalam layanan batas CPU, pemanfaatan memori, bahkan pada throughput puncak, adalah sebagian kecil dari sumber daya yang tersedia.

Jenis aplikasi ini cocok untuk penskalaan otomatis berbasis CPU. Aplikasi ini menikmati fleksibilitas maksimum dalam hal penskalaan. Ini dapat diskalakan secara vertikal dengan menyediakan instans Amazon EC2 yang lebih besar atau vCPU Fargate ke dalamnya. Dan, itu juga dapat diskalakan secara horizontal dengan menambahkan lebih banyak replika. Menambahkan lebih banyak replika, atau menggandakan ukuran instans, memotong rata-rata pemanfaatan CPU relatif terhadap kapasitas hingga setengahnya.

Jika Anda menggunakan kapasitas Amazon EC2 untuk aplikasi ini, pertimbangkan untuk menempatkannya pada instans yang dioptimalkan komputasi seperti atau keluarga. `c5` `c6g`

Server terikat memori yang efisien

Server yang terikat memori yang efisien mengalokasikan sejumlah besar memori per permintaan. Pada konkurensi maksimum, tetapi belum tentu throughput, memori habis sebelum sumber daya CPU habis. Memori yang terkait dengan permintaan dibebaskan saat permintaan berakhir. Permintaan tambahan dapat diterima selama ada memori yang tersedia.

Jenis aplikasi ini cocok untuk penskalaan otomatis berbasis memori. Aplikasi ini menikmati fleksibilitas maksimum dalam hal penskalaan. Ini dapat diskalakan baik secara vertikal dengan menyediakan sumber daya memori Amazon EC2 atau Fargate yang lebih besar untuknya. Dan, itu juga dapat diskalakan secara horizontal dengan menambahkan lebih banyak replika. Menambahkan lebih banyak replika, atau menggandakan ukuran instans, dapat memotong pemanfaatan memori rata-rata relatif terhadap kapasitas hingga setengahnya.

Jika Anda menggunakan kapasitas Amazon EC2 untuk aplikasi ini, pertimbangkan untuk menempatkannya pada instans yang dioptimalkan memori seperti atau keluarga. `r5` `r6g`

Beberapa aplikasi yang terikat memori tidak membebaskan memori yang terkait dengan permintaan ketika itu berakhir, sehingga pengurangan konkurensi tidak menghasilkan pengurangan memori yang digunakan. Untuk ini, kami tidak menyarankan Anda menggunakan penskalaan berbasis memori.

Server berbasis pekerja

Server berbasis pekerja memproses satu permintaan untuk setiap thread pekerja individu satu demi satu. Benang pekerja dapat berupa benang ringan, seperti utas POSIX. Mereka juga bisa menjadi benang yang lebih berat, seperti proses UNIX. Apa pun utas mereka, selalu ada konkurensi maksimum yang dapat didukung aplikasi. Biasanya batas konkurensi diatur secara proporsional dengan sumber daya memori yang tersedia. Jika batas konkurensi tercapai, permintaan tambahan ditempatkan ke antrean backlog. Jika antrian backlog meluap, permintaan masuk tambahan segera ditolak. Aplikasi umum yang sesuai dengan pola ini termasuk server web Apache dan Gunicorn.

Request concurrency biasanya merupakan metrik terbaik untuk menskalakan aplikasi ini. Karena ada batas konkurensi untuk setiap replika, penting untuk menskalakan sebelum batas rata-rata tercapai.

Cara terbaik untuk mendapatkan metrik konkurensi permintaan adalah meminta aplikasi Anda melaporkannya. CloudWatch Setiap replika aplikasi Anda dapat mempublikasikan jumlah permintaan bersamaan sebagai metrik khusus pada frekuensi tinggi. Sebaiknya frekuensinya diatur setidaknya sekali setiap menit. Setelah beberapa laporan dikumpulkan, Anda dapat menggunakan konkurensi rata-rata sebagai metrik penskalaan. Metrik ini dihitung dengan mengambil konkurensi total dan

membaginya dengan jumlah replika. Misalnya, jika total konkurensi adalah 1000 dan jumlah replika adalah 10, maka konkurensi rata-rata adalah 100.

Jika aplikasi Anda berada di belakang Application Load Balancer, Anda juga dapat menggunakan `ActiveConnectionCount` metrik untuk penyeimbang beban sebagai faktor dalam metrik penskalaan. `ActiveConnectionCount`Metrik harus dibagi dengan jumlah replika untuk mendapatkan nilai rata-rata. Nilai rata-rata harus digunakan untuk penskalaan, sebagai lawan dari nilai hitungan mentah.

Agar desain ini bekerja paling baik, standar deviasi latensi respons harus kecil pada tingkat permintaan rendah. Kami merekomendasikan bahwa, selama periode permintaan rendah, sebagian besar permintaan dijawab dalam waktu singkat, dan tidak ada banyak permintaan yang membutuhkan waktu lebih lama daripada waktu rata-rata untuk merespons. Waktu respons rata-rata harus mendekati waktu respons persentil ke-95. Jika tidak, luapan antrian mungkin terjadi sebagai hasilnya. Ini mengarah pada kesalahan. Kami menyarankan Anda memberikan replika tambahan jika diperlukan untuk mengurangi risiko meluap.

Server yang menunggu

Server menunggu melakukan beberapa pemrosesan untuk setiap permintaan, tetapi sangat tergantung pada satu atau lebih layanan hilir untuk berfungsi. Aplikasi kontainer sering menggunakan layanan hilir seperti database dan layanan API lainnya. Butuh beberapa waktu bagi layanan ini untuk merespons, terutama dalam skenario berkapasitas tinggi atau konkurensi tinggi. Ini karena aplikasi ini cenderung menggunakan beberapa sumber daya CPU dan memanfaatkan konkurensi maksimumnya dalam hal memori yang tersedia.

Layanan tunggu cocok baik dalam pola server terikat memori atau pola server berbasis pekerja, tergantung pada bagaimana aplikasi dirancang. Jika konkurensi aplikasi hanya dibatasi oleh memori, maka pemanfaatan memori rata-rata harus digunakan sebagai metrik penskalaan. Jika konkurensi aplikasi didasarkan pada batas pekerja, maka konkurensi rata-rata harus digunakan sebagai metrik penskalaan.

Server berbasis Java

Jika server berbasis Java Anda terikat CPU dan skala secara proporsional dengan sumber daya CPU, maka mungkin cocok untuk pola server terikat CPU yang efisien. Jika demikian, penggunaan CPU rata-rata mungkin sesuai sebagai metrik penskalaan. Namun, banyak aplikasi Java tidak terikat CPU, membuatnya sulit untuk diskalakan.

Untuk kinerja terbaik, kami menyarankan Anda mengalokasikan memori sebanyak mungkin ke tumpukan Java Virtual Machine (JVM). Versi terbaru dari JVM, termasuk pembaruan Java 8 191 atau yang lebih baru, secara otomatis mengatur ukuran tumpukan sebesar mungkin agar muat di dalam wadah. Ini berarti bahwa, di Jawa, pemanfaatan memori jarang sebanding dengan pemanfaatan aplikasi. Ketika tingkat permintaan dan konkurensi meningkat, pemanfaatan memori tetap konstan. Karena itu, kami tidak menyarankan penskalaan server berbasis Java berdasarkan pemanfaatan memori. Sebagai gantinya, kami biasanya merekomendasikan penskalaan pada pemanfaatan CPU.

Dalam beberapa kasus, server berbasis Java mengalami banyak kelelahan sebelum menghabiskan CPU. Jika aplikasi Anda rentan terhadap kelelahan heap pada konkurensi tinggi, maka koneksi rata-rata adalah metrik penskalaan terbaik. Jika aplikasi Anda rentan terhadap heap exhaustion pada throughput tinggi, maka tingkat permintaan rata-rata adalah metrik penskalaan terbaik.

Server yang menggunakan runtime yang dikumpulkan sampah lainnya

Banyak aplikasi server didasarkan pada runtime yang melakukan pengumpulan sampah seperti .NET dan Ruby. Aplikasi server ini mungkin cocok dengan salah satu pola yang dijelaskan sebelumnya. Namun, seperti halnya Java, kami tidak merekomendasikan penskalaan aplikasi ini berdasarkan memori, karena pemanfaatan memori rata-rata yang diamati seringkali tidak berkorelasi dengan throughput atau konkurensi.

Untuk aplikasi ini, kami menyarankan Anda menskalakan pemanfaatan CPU jika aplikasi terikat CPU. Jika tidak, kami menyarankan Anda menskalakan throughput rata-rata atau konkurensi rata-rata, berdasarkan hasil pengujian beban Anda.

Prosesor Job

Banyak beban kerja melibatkan pemrosesan pekerjaan asinkron. Mereka termasuk aplikasi yang tidak menerima permintaan secara real time, tetapi berlangganan antrian kerja untuk menerima pekerjaan. Untuk jenis aplikasi ini, metrik penskalaan yang tepat hampir selalu kedalaman antrian. Pertumbuhan antrian merupakan indikasi bahwa pekerjaan yang tertunda melebihi kapasitas pemrosesan, sedangkan antrian kosong menunjukkan bahwa ada lebih banyak kapasitas daripada pekerjaan yang harus dilakukan.

AWS layanan pesan, seperti Amazon SQS dan Amazon Kinesis Data Streams, CloudWatch menyediakan metrik yang dapat digunakan untuk penskalaan. Untuk Amazon SQS, `ApproximateNumberOfMessagesVisible` adalah metrik terbaik. Untuk Kinesis Data Streams, pertimbangkan untuk `MillisBehindLatest` menggunakan metrik, yang diterbitkan oleh Kinesis Client Library (KCL). Metrik ini harus dirata-ratakan di semua konsumen sebelum menggunakannya untuk penskalaan.

penskalaan otomatis dan deployment layanan

Application Auto Scaling menonaktifkan proses scale-in saat penerapan Amazon ECS sedang berlangsung. Namun, proses penskalaan keluar terus terjadi, kecuali ditangguhkan, selama deployment. Jika Anda ingin menangguhkan proses penskalaan keluar saat deployment dalam progres, lakukan langkah-langkah berikut.

1. Panggil describe-scalable-targetsperintah, tentukan ID sumber daya layanan yang terkait dengan target yang dapat diskalakan di Application Auto Scaling (Contoh:). `service/default/sample-webapp` Catat outputnya. Anda akan membutuhkannya saat memanggil perintah berikutnya.
2. Panggil register-scalable-targetperintah, tentukan ID sumber daya, namespace, dan dimensi yang dapat diskalakan. Tentukan `true`, baik untuk `DynamicScalingInSuspended` maupun `DynamicScalingOutSuspended`.
3. Setelah penerapan selesai, Anda dapat memanggil register-scalable-targetperintah untuk melanjutkan penskalaan.

Untuk informasi lebih lanjut, lihat Menangguhkan dan melanjutkan penskalaan untuk Application Auto Scaling.

Skala layanan Amazon ECS Anda menggunakan nilai metrik target

Dengan kebijakan penskalaan pelacakan target, Anda memilih metrik dan menetapkan nilai target. Auto Scaling Amazon ECS Service membuat dan mengelola CloudWatch alarm yang mengontrol kebijakan penskalaan dan menghitung penyesuaian penskalaan berdasarkan metrik dan nilai target. Kebijakan penskalaan menambahkan atau menghapus tugas layanan sebagaimana diperlukan untuk menjaga metrik pada, atau mendekati nilai target yang ditentukan. Selain menjaga agar metrik tetap mendekati nilai target, kebijakan penskalaan pelacakan target juga menyesuaikan dengan fluktuasi metrik akibat pola beban yang berfluktuasi, serta meminimalkan fluktuasi yang cepat dalam jumlah tugas yang berjalan di layanan Anda.

Pertimbangan

Pertimbangkan hal berikut saat menggunakan kebijakan pelacakan target:

- Kebijakan penskalaan pelacakan target mengasumsikan bahwa penskalaan ke luar harus dilakukan saat metrik yang ditentukan berada di atas nilai target. Anda tidak dapat menggunakan

kebijakan penskalaan pelacakan target untuk menskalakan keluar jika metrik yang ditentukan berada di bawah nilai target.

- Kebijakan penskalaan pelacakan target tidak melakukan penskalaan saat metrik yang ditentukan tidak memiliki data yang mencukupi. Kebijakan penskalaan pelacakan target tidak melakukan penskalaan ke dalam karena data yang tidak mencukupi tidak ditafsirkan sebagai pemanfaatan yang rendah.
- Anda mungkin melihat kesenjangan antara nilai target dan titik data metrik yang aktual. Ini karena Service Auto Scaling selalu bertindak konservatif dengan membulatkan ke atas atau ke bawah ketika menentukan berapa banyak kapasitas untuk menambah atau menghapus. Hal ini mencegahnya menambahkan kapasitas yang tidak mencukupi atau membuang terlalu banyak kapasitas.
- Untuk memastikan ketersediaan aplikasi, layanan menskalakan keluar secara proporsional ke dalam metrik secepat mungkin, namun penskalaan kedalam meningkat secara bertahap.
- Application Auto Scaling menonaktifkan proses scale-in saat penerapan Amazon ECS sedang berlangsung. Namun, proses penskalaan keluar terus terjadi, kecuali ditangguhkan, selama deployment. Untuk informasi selengkapnya, lihat penskalaan otomatis dan deployment layanan.
- Anda dapat memiliki beberapa kebijakan penskalaan pelacakan target untuk layanan Amazon ECS, asalkan masing-masing menggunakan metrik yang berbeda. Tujuan Service Auto Scaling adalah untuk selalu memprioritaskan ketersediaan, sehingga perilakunya berbeda tergantung pada apakah kebijakan pelacakan target siap untuk skala atau skala. Ini akan meningkatkan skala layanan jika ada kebijakan pelacakan target yang siap untuk diskalakan, tetapi akan menskalakan hanya jika semua kebijakan pelacakan target (dengan bagian penskalaan dihidupkan) siap untuk diskalakan.
- Jangan mengedit atau menghapus CloudWatch alarm yang dikelola Service Auto Scaling untuk kebijakan penskalaan pelacakan target. Service Auto Scaling akan menghapus alarm secara otomatis saat Anda menghapus kebijakan penskalaan.
- `ALBRequestCountPerTarget`Metrik untuk kebijakan penskalaan pelacakan target tidak didukung untuk jenis penerapan biru/hijau.

Untuk informasi lebih lanjut tentang kebijakan penskalaan pelacakan target, lihat Kebijakan penskalaan pelacakan target dalam Panduan Pengguna Application Auto Scaling.

Untuk mengonfigurasi kebijakan penskalaan target untuk layanan Amazon ECS Anda menggunakan konsol Amazon ECS

1. Selain izin IAM standar untuk membuat dan memperbarui layanan, Anda memerlukan izin tambahan. Untuk informasi selengkapnya, lihat Izin IAM diperlukan untuk penskalaan otomatis servis.
2. Anda dapat mengonfigurasi kebijakan penskalaan saat membuat atau memperbarui layanan. Untuk informasi selengkapnya, lihat salah satu dari berikut ini:
   - Buat layanan menggunakan parameter yang ditentukan— Buat layanan baru
   - Memperbarui layanan menggunakan konsol— Perbarui layanan yang ada

Untuk mengonfigurasi kebijakan penskalaan target untuk layanan Amazon ECS Anda menggunakan AWS CLI

1. Selain izin IAM standar untuk membuat dan memperbarui layanan, Anda memerlukan izin tambahan. Untuk informasi selengkapnya, lihat Izin IAM diperlukan untuk penskalaan otomatis servis.
2. Daftarkan layanan Amazon ECS Anda sebagai target yang dapat diskalakan menggunakan perintah. register-scalable-target
3. Buat kebijakan penskalaan menggunakan put-scaling-policyperintah.

Skalakan layanan Amazon ECS Anda menggunakan kenaikan yang telah ditentukan berdasarkan alarm CloudWatch

Dengan kebijakan penskalaan langkah, Anda menentukan CloudWatch alarm yang memulai proses penskalaan. Misalnya, jika Anda ingin meningkatkan skala ketika pemanfaatan CPU mencapai tingkat tertentu, buat alarm menggunakan `CPUUtilization` metrik yang disediakan. Saat membuat kebijakan penskalaan bertahap, Anda harus menentukan salah satu dari tipe penyesuaian penskalaan berikut:

- Tambah - Tingkatkan jumlah tugas dengan jumlah unit kapasitas tertentu atau persentase tertentu dari kapasitas saat ini.
- Hapus - Kurangi jumlah tugas dengan jumlah unit kapasitas tertentu atau persentase tertentu dari kapasitas saat ini.
- Atur ke - Atur jumlah tugas ke jumlah unit kapasitas yang ditentukan.

Sebagai contoh, misalkan kapasitas target dan kapasitas yang terpenuhi adalah 10 serta kebijakan penskalaan menambahkan 1. Ketika alarm dilanggar, proses penskalaan otomatis menambahkan 1 hingga 10 untuk mendapatkan 11, sehingga Amazon ECS meluncurkan 1 tugas untuk layanan tersebut.

Kami sangat menyarankan agar Anda menggunakan kebijakan penskalaan pelacakan target untuk menskalakan metrik seperti pemanfaatan CPU rata-rata atau jumlah permintaan rata-rata per target. Metrik yang menurun ketika kapasitas meningkat dan meningkat ketika kapasitas menurun dapat digunakan untuk skala proporsional atau dalam jumlah tugas menggunakan pelacakan target. Ini membantu memastikan bahwa Service Auto Scaling mengikuti kurva permintaan untuk aplikasi Anda dengan cermat.

Untuk gambaran umum tentang kebijakan penskalaan langkah dan cara kerjanya, lihat Kebijakan penskalaan langkah di Panduan Pengguna Application Auto Scaling. Setelah Anda membaca pendahuluan ini, lihat bagian berikut untuk mempelajari cara mengonfigurasi penskalaan langkah untuk Amazon ECS menggunakan konsol dan. AWS Command Line Interface

Untuk mengonfigurasi kebijakan penskalaan langkah untuk layanan Amazon ECS Anda menggunakan konsol Amazon ECS

1. Selain izin IAM standar untuk membuat dan memperbarui layanan, Anda memerlukan izin tambahan. Untuk informasi selengkapnya, lihat Izin IAM diperlukan untuk penskalaan otomatis servis.
2. Anda dapat mengonfigurasi kebijakan penskalaan saat membuat atau memperbarui layanan. Untuk informasi selengkapnya, lihat salah satu dari berikut ini:

   - Buat layanan menggunakan parameter yang ditentukan— Buat layanan baru
   - Memperbarui layanan menggunakan konsol— Perbarui layanan yang ada

Untuk mengonfigurasi kebijakan penskalaan langkah untuk layanan Amazon ECS Anda menggunakan AWS CLI

1. Selain izin IAM standar untuk membuat dan memperbarui layanan, Anda memerlukan izin tambahan. Untuk informasi selengkapnya, lihat Izin IAM diperlukan untuk penskalaan otomatis servis.
2. Daftarkan layanan Amazon ECS Anda sebagai target yang dapat diskalakan menggunakan perintah. register-scalable-target
3. Buat kebijakan penskalaan menggunakan put-scaling-policyperintah.

4. Buat alarm yang memulai kebijakan penskalaan menggunakan perintah. put-metric-alarm

Skala layanan Amazon ECS Anda menggunakan jadwal

Dengan penskalaan terjadwal, Anda dapat mengatur penskalaan otomatis untuk aplikasi berdasarkan perubahan beban yang dapat diprediksi dengan membuat tindakan terjadwal yang menambah atau mengurangi kapasitas pada waktu tertentu. Ini memungkinkan Anda menskalakan aplikasi secara proaktif agar sesuai dengan perubahan beban yang dapat diprediksi.

Tindakan penskalaan terjadwal ini memungkinkan Anda mengoptimalkan biaya dan kinerja. Aplikasi Anda memiliki kapasitas yang cukup untuk menangani puncak lalu lintas pertengahan minggu, tetapi tidak menyediakan kapasitas yang tidak dibutuhkan secara berlebihan di lain waktu.

Anda dapat menggunakan kebijakan penskalaan dan penskalaan terjadwal bersama-sama untuk mendapatkan manfaat dari pendekatan proaktif dan reaktif untuk penskalaan. Setelah tindakan penskalaan terjadwal berjalan, kebijakan penskalaan dapat terus membuat keputusan tentang apakah akan meningkatkan kapasitas skala lebih lanjut. Ini membantu memastikan bahwa Anda memiliki kapasitas yang cukup untuk menangani beban untuk aplikasi Anda. Meskipun aplikasi Anda menskalakan sesuai dengan permintaan, kapasitas saat ini harus berada di antara kapasitas minimum dan maksimum yang ditetapkan oleh tindakan terjadwal Anda.

Anda dapat mengonfigurasi penskalaan jadwal menggunakan. AWS CLI Untuk informasi selengkapnya tentang penskalaan terjadwal, lihat Penskalaan Terjadwal di Panduan Pengguna Application Auto Scaling.

Layanan interkoneksi

Aplikasi yang berjalan dalam tugas Amazon ECS seringkali perlu menerima koneksi dari internet atau untuk terhubung ke aplikasi lain yang berjalan di layanan Amazon ECS. Jika Anda membutuhkan koneksi eksternal dari internet, sebaiknya gunakan Elastic Load Balancing. Untuk informasi selengkapnya tentang penyeimbangan beban terintegrasi, lihatthe section called “Mendistribusikan lalu lintas layanan Amazon ECS menggunakan load balancing”.

Memilih metode interkoneksi

Jika Anda memerlukan aplikasi untuk terhubung ke aplikasi lain yang berjalan di layanan Amazon ECS, Amazon ECS menyediakan cara-cara berikut untuk melakukan ini tanpa penyeimbang beban:

- Amazon ECS Service Connect

Kami merekomendasikan Service Connect, yang menyediakan konfigurasi Amazon ECS untuk penemuan layanan, konektivitas, dan pemantauan lalu lintas. Dengan Service Connect, aplikasi Anda dapat menggunakan nama pendek dan port standar untuk terhubung ke layanan Amazon ECS di cluster yang sama, cluster lain, termasuk di seluruh VPC yang sama. Wilayah AWS

Saat Anda menggunakan Service Connect, Amazon ECS mengelola semua bagian penemuan layanan: membuat nama yang dapat ditemukan, mengelola entri secara dinamis untuk setiap tugas saat tugas dimulai dan dihentikan, menjalankan agen di setiap tugas yang dikonfigurasi untuk menemukan nama. Aplikasi Anda dapat mencari nama dengan menggunakan fungsionalitas standar untuk nama DNS dan membuat koneksi. Jika aplikasi Anda sudah melakukan ini, Anda tidak perlu memodifikasi aplikasi Anda untuk menggunakan Service Connect.

Anda menyediakan konfigurasi lengkap di dalam setiap layanan dan definisi tugas. Amazon ECS mengelola perubahan pada konfigurasi ini di setiap penyebaran layanan, untuk memastikan bahwa semua tugas dalam penerapan berperilaku dengan cara yang sama. Misalnya, masalah umum dengan DNS sebagai penemuan layanan adalah mengendalikan migrasi. Jika Anda mengubah nama DNS untuk menunjuk ke alamat IP pengganti baru, mungkin diperlukan waktu TTL maksimum sebelum semua klien mulai menggunakan layanan baru. Dengan Service Connect, penyebaran klien memperbarui konfigurasi dengan mengganti tugas klien. Anda dapat mengonfigurasi pemutus sirkuit penyebaran dan konfigurasi penerapan lainnya untuk memengaruhi perubahan Service Connect dengan cara yang sama seperti penerapan lainnya.

Untuk informasi selengkapnya, lihat Layanan Connect.

- Penemuan layanan Amazon ECS

  Pendekatan lain untuk service-to-service komunikasi adalah komunikasi langsung menggunakan penemuan layanan. Dalam pendekatan ini, Anda dapat menggunakan integrasi penemuan AWS Cloud Map layanan dengan Amazon ECS. Menggunakan penemuan layanan, Amazon ECS menyinkronkan daftar tugas yang diluncurkan ke AWS Cloud Map, yang mempertahankan nama host DNS yang menyelesaikan ke alamat IP internal dari satu atau lebih tugas dari layanan tertentu. Layanan lain di Amazon VPC dapat menggunakan nama host DNS ini untuk mengirim lalu lintas langsung ke wadah lain menggunakan alamat IP internalnya.

  Pendekatan service-to-service komunikasi ini memberikan latensi rendah. Tidak ada komponen tambahan di antara wadah. Lalu lintas bergerak langsung dari satu kontainer ke kontainer lainnya.

  Pendekatan ini cocok saat menggunakan mode `awsvpc` jaringan, di mana setiap tugas memiliki alamat IP uniknya sendiri. Sebagian besar perangkat lunak hanya mendukung penggunaan A

catatan DNS, yang menyelesaikan langsung ke alamat IP. Saat menggunakan mode `awsvpc` jaringan, alamat IP untuk setiap tugas adalah A catatan. Namun, jika Anda menggunakan mode `bridge` jaringan, beberapa kontainer dapat berbagi alamat IP yang sama. Selain itu, pemetaan port dinamis menyebabkan kontainer diberi nomor port secara acak pada alamat IP tunggal itu. Pada titik ini, A catatan tidak lagi cukup untuk penemuan layanan. Anda juga harus menggunakan SRV catatan. Jenis catatan ini dapat melacak alamat IP dan nomor port tetapi mengharuskan Anda mengonfigurasi aplikasi dengan tepat. Beberapa aplikasi bawaan yang Anda gunakan mungkin tidak mendukung SRV catatan.

Keuntungan lain dari mode `awsvpc` jaringan adalah Anda memiliki grup keamanan unik untuk setiap layanan. Anda dapat mengonfigurasi grup keamanan ini untuk mengizinkan koneksi masuk hanya dari layanan hulu tertentu yang perlu berbicara dengan layanan tersebut.

Kerugian utama dari service-to-service komunikasi langsung menggunakan penemuan layanan adalah Anda harus menerapkan logika tambahan untuk mencoba ulang dan menangani kegagalan koneksi. Catatan DNS memiliki periode time-to-live (TTL) yang mengontrol berapa lama mereka di-cache. Dibutuhkan beberapa waktu agar catatan DNS diperbarui dan cache kedaluwarsa sehingga aplikasi Anda dapat mengambil versi terbaru dari catatan DNS. Jadi, aplikasi Anda mungkin akan menyelesaikan catatan DNS untuk menunjuk ke wadah lain yang sudah tidak ada lagi. Aplikasi Anda perlu menangani percobaan ulang dan memiliki logika untuk mengabaikan backend yang buruk.

Lihat informasi yang lebih lengkap di Penemuan Layanan

Tabel kompatibilitas mode jaringan

Tabel berikut mencakup kompatibilitas antara opsi ini dan mode jaringan tugas. Dalam tabel, “klien” mengacu pada aplikasi yang membuat koneksi dari dalam tugas Amazon ECS.

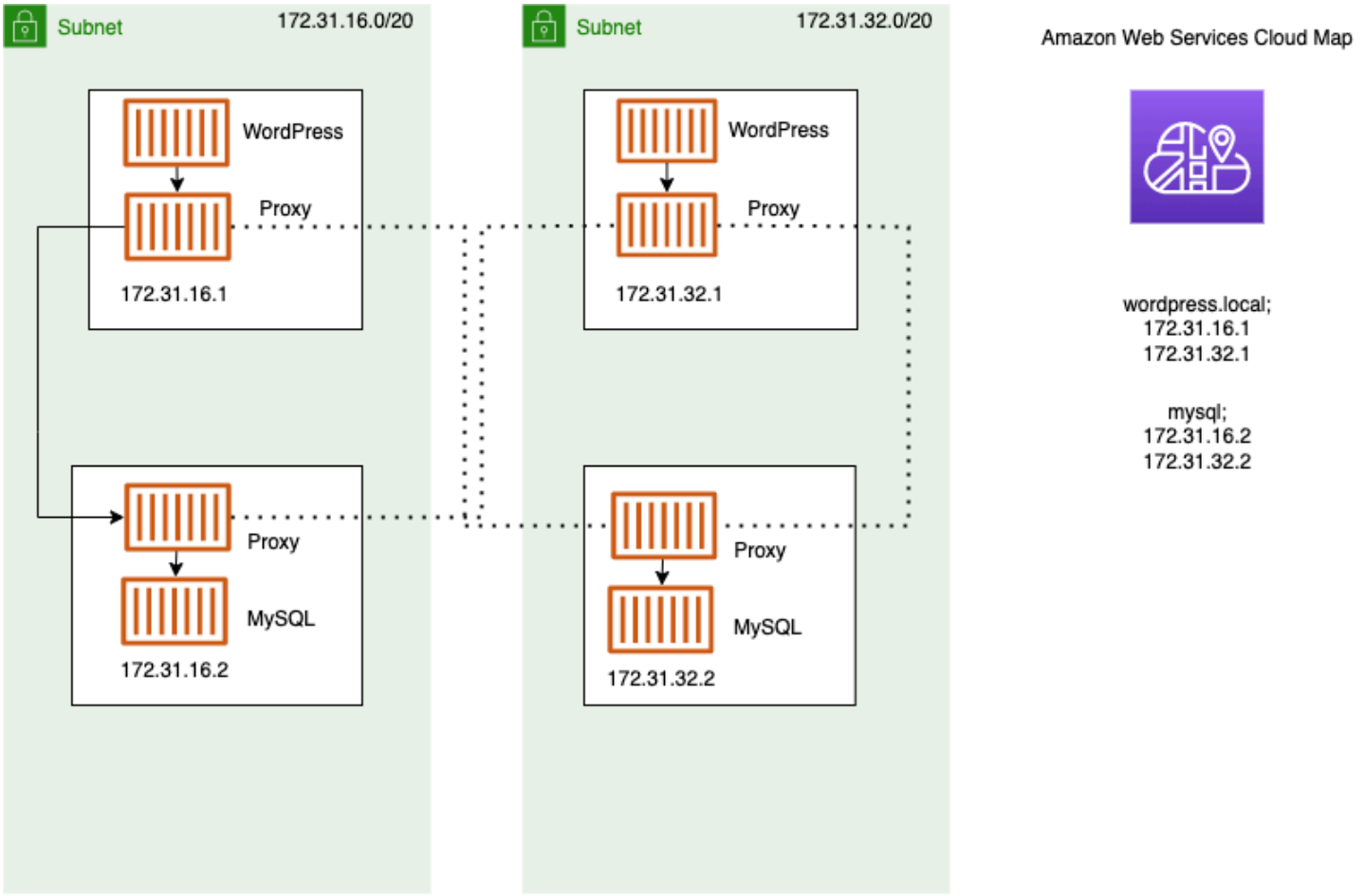
Opsi Interkoneksi	Dijembatani	**`awsvpc`**	Host
Penemuan layanan	ya, tetapi mengharuskan klien mengetahui catatan SRV di DNS tanpa. `hostPort`	Ya	ya, tetapi mengharuskan klien mengetahui catatan SRV di DNS tanpa. `hostPort`

Opsi Interkoneksi	Dijembatani	**awsvpc**	Host
Layanan Connect	Ya	ya	tidak

Layanan Connect

Amazon ECS Service Connect menyediakan manajemen service-to-service komunikasi sebagai konfigurasi Amazon ECS. Ini dilakukan dengan membangun penemuan layanan dan mesh layanan di Amazon ECS. Ini menyediakan konfigurasi lengkap di dalam setiap layanan Amazon ECS yang Anda kelola berdasarkan penerapan layanan, cara terpadu untuk merujuk ke layanan Anda dalam ruang nama yang tidak bergantung pada konfigurasi DNS VPC Amazon, serta metrik dan log standar untuk memantau semua aplikasi Anda di Amazon ECS. Amazon ECS Service Connect hanya menghubungkan layanan Amazon ECS.

Diagram berikut menunjukkan contoh jaringan Service Connect dengan 2 subnet di VPC dan 2 layanan. Layanan klien yang berjalan WordPress dengan 1 tugas di setiap subnet. Layanan server yang menjalankan MySQL dengan 1 tugas di setiap subnet. Kedua layanan sangat tersedia dan tahan terhadap masalah tugas dan Availability Zone karena setiap layanan menjalankan beberapa tugas yang tersebar di 2 subnet. Panah padat menunjukkan koneksi dari WordPress ke MySQL. Misalnya, perintah `mysql --host=mysql` CLI yang dijalankan dari dalam WordPress wadah dalam tugas dengan alamat IP. `172.31.16.1` Perintah menggunakan nama pendek `mysql` pada port default untuk MySQL. Nama dan port ini terhubung ke proxy Service Connect dalam tugas yang sama. Proxy dalam WordPress tugas menggunakan penyeimbangan beban round-robin dan informasi kegagalan sebelumnya dalam deteksi outlier untuk memilih tugas MySQL mana yang akan dihubungkan. Seperti yang ditunjukkan oleh panah padat dalam diagram, proxy terhubung ke proxy kedua dalam tugas MySQL dengan Alamat IP. `172.31.16.2` Proxy kedua terhubung ke server MySQL lokal dalam tugas yang sama. Kedua proxy melaporkan kinerja koneksi yang terlihat dalam grafik di Amazon ECS dan konsol CloudWatch Amazon sehingga Anda bisa mendapatkan metrik kinerja dari semua jenis aplikasi dengan cara yang sama.



Ikhtisar langkah-langkah untuk mengkonfigurasi Service Connect

Ikuti langkah-langkah ini untuk mengonfigurasi Service Connect untuk sekelompok layanan terkait.

> **Important**
>
> - Amazon ECS Service Connect membuat AWS Cloud Map layanan di akun Anda. Memodifikasi AWS Cloud Map sumber daya ini dengan mendaftarkan/membatalkan pendaftaran instance secara manual, mengubah atribut instance, atau menghapus layanan dapat menyebabkan perilaku tak terduga untuk lalu lintas aplikasi Anda atau penerapan berikutnya.
> - Amazon ECS Service Connect tidak mendukung tautan dalam definisi tugas.

1. Tambahkan nama port ke pemetaan port dalam definisi tugas Anda. Selain itu, Anda dapat mengidentifikasi protokol lapisan 7 aplikasi, untuk mendapatkan metrik tambahan.

2. Buat cluster ECS dengan AWS Cloud Map namespace atau buat namespace secara terpisah. Untuk organisasi sederhana, buat klaster Amazon ECS dengan nama yang Anda inginkan untuk namespace dan tentukan nama yang identik untuk namespace. Dalam hal ini, Amazon ECS membuat namespace HTTP baru dengan konfigurasi yang diperlukan. Amazon ECS Service Connect tidak menggunakan atau membuat zona yang dihosting DNS di Amazon Route 53.
3. Konfigurasikan layanan untuk membuat titik akhir Service Connect di dalam namespace.
4. Menyebarkan layanan untuk membuat titik akhir. Amazon ECS menambahkan container proxy Service Connect ke setiap tugas, dan membuat titik akhir Service Connect. AWS Cloud Map Wadah ini tidak dikonfigurasi dalam definisi tugas, dan definisi tugas dapat digunakan kembali tanpa modifikasi untuk membuat beberapa layanan di namespace yang sama atau di beberapa ruang nama.
5. Menerapkan aplikasi klien sebagai layanan untuk terhubung ke titik akhir. Amazon ECS menghubungkannya ke titik akhir Service Connect melalui proxy Service Connect di setiap tugas.

   Aplikasi hanya menggunakan proxy untuk terhubung ke titik akhir Service Connect. Tidak ada konfigurasi tambahan untuk menggunakan proxy. Proxy melakukan penyeimbangan beban round-robin, deteksi outlier, dan percobaan ulang. Untuk informasi selengkapnya tentang proxy, lihatProksi Service Connect.
6. Pantau lalu lintas melalui proxy Service Connect di Amazon CloudWatch.

Wilayah dengan Service Connect

Amazon ECS Service Connect tersedia di AWS Wilayah berikut:

Nama Wilayah	Wilayah
US East (Ohio)	us-east-2
US East (N. Virginia)	us-east-1
US West (N. California)	us-west-1
US West (Oregon)	as-barat-2
Afrika (Cape Town)	af-selatan-1
Asia Pasifik (Hong Kong)	ap-east-1

Nama Wilayah	Wilayah
Asia Pasifik (Jakarta)	ap-southeast-3
Asia Pasifik (Mumbai)	ap-south-1
Asia Pasifik (Hyderabad)	ap-south-2
Asia Pacific (Osaka)	ap-northeast-3
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pasifik (Melbourne)	ap-southeast-4
Asia Pasifik (Tokyo)	ap-northeast-1
Canada (Central)	ca-central-1
Kanada Barat (Calgary)	ca-west-1
Tiongkok (Beijing)	cn-north-1 (Catatan: TLS untuk Service Connect tidak tersedia di wilayah ini.)
Tiongkok (Ningxia)	cn-northwest-1 (Catatan: TLS untuk Service Connect tidak tersedia di wilayah ini.)
Eropa (Frankfurt)	eu-central-1
Europe (Ireland)	eu-west-1
Europe (London)	eu-west-2
Europe (Paris)	eu-west-3
Eropa (Milan)	eu-south-1
Eropa (Spanyol)	eu-south-2

Nama Wilayah	Wilayah
Eropa (Stockholm)	eu-north-1
Europe (Zurich)	eu-central-2
Israel (Tel Aviv)	il-central-1
Timur Tengah (Bahrain)	me-south-1
Timur Tengah (UEA)	me-central-1
Amerika Selatan (Sao Paulo)	sa-east-1

Konsep Service Connect

Fitur Service Connect menciptakan jaringan virtual layanan terkait. Konfigurasi layanan yang sama dapat digunakan di beberapa ruang nama yang berbeda untuk menjalankan set aplikasi yang independen namun identik. Service Connect mendefinisikan wadah proxy di layanan Amazon ECS. Dengan cara ini, definisi tugas yang sama dapat digunakan untuk menjalankan aplikasi identik di ruang nama yang berbeda dengan konfigurasi Service Connect yang berbeda. Setiap tugas yang dibuat oleh layanan Amazon ECS menjalankan penampung proxy dalam tugas.

Service Connect cocok untuk koneksi antara layanan Amazon ECS dalam namespace yang sama. Untuk aplikasi berikut, Anda perlu menggunakan metode interkoneksi tambahan untuk menyambung ke layanan Amazon ECS yang dikonfigurasi dengan Service Connect:

- Tugas Amazon ECS yang dikonfigurasi di ruang nama lain
- Tugas Amazon ECS yang tidak dikonfigurasi untuk Service Connect
- Aplikasi lain di luar Amazon ECS

Aplikasi ini dapat terhubung melalui proxy Service Connect tetapi tidak dapat menyelesaikan nama titik akhir Service Connect.

Agar aplikasi ini dapat menyelesaikan alamat IP tugas Amazon ECS, Anda perlu menggunakan metode interkoneksi lain. Untuk daftar metode interkoneksi, lihatMemilih metode interkoneksi.

Terminologi Service Connect

Istilah berikut digunakan dengan Service Connect.

nama port

Konfigurasi definisi tugas Amazon ECS yang memberikan nama ke pemetaan port tertentu. Konfigurasi ini hanya digunakan oleh Amazon ECS Service Connect.

alias klien

Konfigurasi layanan Amazon ECS yang menetapkan nomor port yang digunakan di titik akhir. Selain itu, alias klien dapat menetapkan nama DNS dari titik akhir, mengesampingkan nama penemuan. Jika nama penemuan tidak disediakan di layanan Amazon ECS, nama alias klien akan mengganti nama port sebagai nama titik akhir. Untuk contoh endpoint, lihat definisi endpoint. Beberapa alias klien dapat ditetapkan ke layanan Amazon ECS. Konfigurasi ini hanya digunakan oleh Amazon ECS Service Connect.

nama penemuan

Nama perantara opsional yang dapat Anda buat untuk port tertentu dari definisi tugas. Nama ini digunakan untuk membuat AWS Cloud Map layanan. Jika nama ini tidak disediakan, nama port dari definisi tugas akan digunakan. Beberapa nama penemuan dapat ditetapkan ke port tertentu layanan Amazon ECS. Konfigurasi ini hanya digunakan oleh Amazon ECS Service Connect.

AWS Cloud Map nama layanan harus unik dalam namespace. Karena batasan ini, Anda hanya dapat memiliki satu konfigurasi Service Connect tanpa nama penemuan untuk definisi tugas tertentu di setiap namespace.

titik akhir

URL untuk terhubung ke API atau situs web. URL berisi protokol, nama DNS, dan port. Untuk informasi lebih lanjut tentang titik akhir secara umum, lihat titik akhir dalam AWS glosarium di. Referensi Umum Amazon Web

Service Connect membuat titik akhir yang terhubung ke layanan Amazon ECS dan mengonfigurasi tugas di layanan Amazon ECS untuk terhubung ke titik akhir. URL berisi protokol, nama DNS, dan port. Anda memilih protokol dan nama port dalam definisi tugas, karena port harus cocok dengan aplikasi yang ada di dalam gambar kontainer. Dalam layanan, Anda memilih setiap port dengan nama dan dapat menetapkan nama DNS. Jika Anda tidak menentukan nama DNS dalam konfigurasi layanan Amazon ECS, nama port dari definisi tugas akan digunakan

secara default. Misalnya, titik akhir Service Connect dapat berupa`http://blog:80`,`grpc://checkout:8080`, atau`http://_db.production.internal:99`.

Layanan Connect Service

Konfigurasi titik akhir tunggal dalam layanan Amazon ECS. Ini adalah bagian dari konfigurasi Service Connect, yang terdiri dari satu baris dalam konfigurasi Service Connect dan discovery name di konsol, atau satu objek dalam `services` daftar dalam konfigurasi JSON dari layanan Amazon ECS. Konfigurasi ini hanya digunakan oleh Amazon ECS Service Connect.

Untuk informasi selengkapnya, lihat ServiceConnectServicedi Referensi API Amazon Elastic Container Service.

namespace

Nama pendek atau nama sumber daya Amazon lengkap (ARN) dari AWS Cloud Map namespace untuk digunakan dengan Service Connect. Namespace harus Wilayah AWS sama dengan layanan Amazon ECS dan cluster. Jenis namespace di AWS Cloud Map tidak memengaruhi Service Connect.

Service Connect menggunakan AWS Cloud Map namespace sebagai pengelompokan logis tugas Amazon ECS yang berbicara satu sama lain. Setiap layanan Amazon ECS hanya dapat dimiliki oleh satu namespace. Layanan dalam namespace dapat tersebar di berbagai kluster Amazon ECS yang berbeda dalam hal yang sama Wilayah AWS . Akun AWS Karena setiap cluster dapat menjalankan tugas dari setiap sistem operasi, arsitektur CPU, VPC, dan EC2, Fargate, dan tipe Eksternal, Anda dapat dengan bebas mengatur layanan Anda dengan kriteria apa pun yang Anda pilih.

layanan klien

Layanan Amazon ECS yang menjalankan aplikasi klien jaringan. Layanan ini harus memiliki namespace yang dikonfigurasi. Setiap tugas dalam layanan dapat menemukan dan terhubung ke semua titik akhir di namespace melalui container proxy Service Connect.

Jika salah satu kontainer Anda dalam tugas perlu terhubung ke titik akhir dari layanan di namespace, pilih layanan klien. Jika aplikasi frontend, reverse proxy, atau load balancer menerima lalu lintas eksternal melalui metode lain seperti dari Elastic Load Balancing, itu bisa menggunakan konfigurasi Service Connect jenis ini.

layanan client-server

Layanan Amazon ECS yang menjalankan aplikasi jaringan atau layanan web. Layanan ini harus memiliki namespace dan setidaknya satu titik akhir yang dikonfigurasi. Setiap tugas dalam

layanan dapat dijangkau dengan menggunakan titik akhir. Container proxy Service Connect mendengarkan nama endpoint dan port untuk mengarahkan lalu lintas ke container aplikasi dalam tugas.

Jika salah satu kontainer mengekspos dan mendengarkan pada port untuk lalu lintas jaringan, pilih layanan client-server. Aplikasi ini tidak perlu terhubung ke layanan client-server lain di namespace yang sama, tetapi konfigurasi klien dikonfigurasi. Backend, middleware, tingkat bisnis, atau sebagian besar layanan mikro akan menggunakan jenis konfigurasi Service Connect ini. Jika Anda ingin aplikasi frontend, reverse proxy, atau load balancer menerima lalu lintas dari layanan lain yang dikonfigurasi dengan Service Connect di namespace yang sama, layanan ini harus menggunakan jenis konfigurasi Service Connect ini.

Konfigurasi cluster

Anda dapat mengatur namespace default untuk Service Connect saat membuat klaster atau dengan memperbarui klaster. Jika Anda menentukan nama namespace yang tidak ada di akun Wilayah AWS dan yang sama, namespace HTTP baru akan dibuat.

Jika Anda membuat klaster dan menentukan namespace Service Connect default, klaster menunggu dalam status `PROVISIONING` sementara Amazon ECS membuat namespace. Anda dapat melihat status klaster yang menunjukkan status namespace. `attachment` Lampiran tidak ditampilkan secara default di AWS CLI, Anda harus menambahkan `--include ATTACHMENTS` untuk melihatnya.

Konfigurasi layanan Service Connect

Service Connect dirancang untuk memerlukan konfigurasi minimum. Anda perlu menetapkan nama untuk setiap pemetaan port yang ingin Anda gunakan dengan Service Connect dalam definisi tugas. Dalam layanan, Anda perlu mengaktifkan Service Connect dan memilih namespace untuk membuat layanan klien. Untuk membuat layanan client-server, Anda perlu menambahkan satu konfigurasi layanan Service Connect yang cocok dengan nama salah satu pemetaan port. Amazon ECS menggunakan kembali nomor port dan nama port dari definisi tugas untuk menentukan layanan dan titik akhir Service Connect. Untuk mengganti nilai tersebut, Anda dapat menggunakan parameter lain Discovery, DNS, dan Port di konsol, atau `discoveryName` dan`clientAliases`, masing-masing di Amazon ECS API.

Contoh berikut menunjukkan setiap jenis konfigurasi Service Connect yang digunakan bersama-sama dalam layanan Amazon ECS yang sama. Komentar Shell disediakan, namun perhatikan bahwa konfigurasi JSON yang digunakan untuk layanan Amazon ECS tidak mendukung komentar.

```
{
    ...
    serviceConnectConfiguration: {
        enabled: true,
        namespace: "internal",
        #config for client services can end here, only these two parameters are
 required.
        services: [{
                portName: "http"
            }, #minimal client - server service config can end here.portName must match
 the "name"
            parameter of a port mapping in the task definition. {
                discoveryName: "http-second"
                #name the discoveryName to avoid a Task def port name collision with
 the minimal config in the same Cloud Map namespace
                portName: "http"
            },
            {
                clientAliases: [{
                        dnsName: "db",
                        port: 81
                    }] #use when the port in Task def is not the port that client apps
 use.Client apps can use http: //db:81 to connect
                    discoveryName: "http-three"
                portName: "http"
            },
            {
                clientAliases: [{
                    dnsName: "db.app",
                    port: 81
                }] #use when the port in Task def is not the port that client apps
 use.duplicates are fine as long as the discoveryName is different.
                discoveryName: "http-four"
                portName: "http",
                ingressPortOverride: 99 #If App should also accept traffic directly on
 Task def port.
            }
        ]
    }
}
```

Urutan penyebaran

Saat Anda menggunakan Amazon ECS Service Connect, Anda mengonfigurasi setiap layanan Amazon ECS baik untuk menjalankan aplikasi server yang menerima permintaan jaringan (layanan client-server) atau untuk menjalankan aplikasi klien yang membuat permintaan (layanan klien).

Saat Anda bersiap untuk mulai menggunakan Service Connect, mulailah dengan layanan client-server. Anda dapat menambahkan konfigurasi Service Connect ke layanan baru atau layanan yang sudah ada. Setelah Anda mengedit dan memperbarui layanan Amazon ECS untuk menambahkan konfigurasi Service Connect, Amazon ECS membuat titik akhir Service Connect di namespace. Selain itu, Amazon ECS membuat penerapan baru di layanan untuk menggantikan tugas yang sedang berjalan.

Tugas yang ada dan aplikasi lain dapat terus terhubung ke titik akhir yang ada, dan aplikasi eksternal. Jika layanan client-server menambahkan tugas dengan skala keluar, koneksi baru dari klien akan segera seimbang di antara semua tugas. Jika layanan client-server diperbarui, koneksi baru dari klien akan segera seimbang antara tugas-tugas versi baru.

Tugas yang ada tidak dapat menyelesaikan dan terhubung ke titik akhir yang baru. Hanya tugas Amazon ECS baru yang memiliki konfigurasi Service Connect di namespace yang sama dan yang mulai berjalan setelah penerapan ini dapat menyelesaikan dan terhubung ke titik akhir ini. Misalnya, layanan Amazon ECS yang menjalankan aplikasi klien harus di-deploy ulang untuk terhubung ke endpoint server database baru. Mulai penyebaran klien setelah penerapan selesai untuk server.

Ini berarti bahwa operator aplikasi klien menentukan kapan konfigurasi aplikasi mereka berubah, meskipun operator aplikasi server dapat mengubah konfigurasi mereka kapan saja. Daftar titik akhir di namespace dapat berubah setiap kali layanan Amazon ECS di namespace diterapkan, tetapi tugas dan tugas pengganti yang ada terus berperilaku sama seperti yang mereka lakukan setelah penerapan terbaru.

Pertimbangkan contoh berikut.

Pertama, asumsikan bahwa Anda membuat aplikasi yang tersedia untuk internet publik dalam satu AWS CloudFormation template dan AWS CloudFormation tumpukan tunggal. Penemuan dan jangkauan publik harus dibuat terakhir oleh AWS CloudFormation, termasuk layanan klien frontend. Layanan perlu dibuat dalam urutan ini untuk mencegah periode waktu ketika layanan klien frontend berjalan dan tersedia untuk umum, tetapi backend tidak. Ini menghilangkan pesan kesalahan agar tidak dikirim ke publik selama periode waktu tersebut. Di AWS CloudFormation, Anda harus menggunakan `dependsOn` untuk menunjukkan AWS CloudFormation bahwa beberapa

layanan Amazon ECS tidak dapat dibuat secara paralel atau bersamaan. Anda harus menambahkan `dependsOn` ke layanan klien frontend untuk setiap layanan client-server backend yang terhubung dengan tugas klien.

Kedua, asumsikan bahwa layanan frontend ada tanpa konfigurasi Service Connect. Tugas menghubungkan ke layanan backend yang ada. Tambahkan konfigurasi Service Connect client-server ke layanan backend terlebih dahulu, menggunakan nama yang sama di DNS atau `clientAlias` yang digunakan frontend. Ini menciptakan penerapan baru, sehingga semua deteksi rollback penerapan atau AWS Management Console, AWS CLI, AWS SDK dan metode lain untuk memutar kembali dan mengembalikan layanan backend ke penerapan dan konfigurasi sebelumnya. Jika Anda puas dengan kinerja dan perilaku layanan backend, tambahkan konfigurasi Service Connect klien atau client-server ke layanan frontend. Hanya tugas dalam penerapan baru yang menggunakan proxy Service Connect yang ditambahkan ke tugas baru tersebut. Jika Anda memiliki masalah dengan konfigurasi ini, Anda dapat memutar kembali dan kembali ke konfigurasi sebelumnya dengan menggunakan deteksi rollback penerapan atau AWS Management Console AWS CLI, AWS SDK dan metode lain untuk memutar kembali dan mengembalikan layanan backend ke penerapan dan konfigurasi sebelumnya. Jika Anda menggunakan sistem penemuan layanan lain yang didasarkan pada DNS dan bukan Service Connect, aplikasi frontend atau klien apa pun mulai menggunakan titik akhir baru dan mengubah konfigurasi titik akhir setelah cache DNS lokal kedaluwarsa, biasanya memakan waktu beberapa jam.

Jaringan

Dalam konfigurasi default, proxy Service Connect mendengarkan pemetaan `containerPort` dari port dalam definisi tugas. Anda memerlukan aturan dalam grup keamanan Anda untuk mengizinkan lalu lintas masuk (masuknya) ke port ini dari CIDR VPC, atau secara khusus dari subnet tempat klien akan dijalankan.

Bahkan jika Anda menetapkan nomor port dalam konfigurasi layanan Service Connect, ini tidak mengubah port untuk layanan client-server yang didengarkan proxy Service Connect. Saat Anda menyetel nomor port ini, Amazon ECS mengubah port titik akhir yang terhubung dengan layanan klien, pada proxy Service Connect di dalam tugas tersebut. Proxy dalam layanan klien terhubung ke proxy di layanan client-server menggunakan file. `containerPort`

Jika Anda ingin mengubah port yang didengarkan proxy Service Connect, ubah konfigurasi Service Connect pada layanan client-server. `ingressPortOverride` Jika Anda mengubah nomor port ini, Anda harus mengizinkan lalu lintas masuk pada port ini di grup keamanan Amazon VPC yang digunakan oleh lalu lintas ke layanan ini.

Lalu lintas yang dikirim aplikasi Anda ke layanan Amazon ECS yang dikonfigurasi untuk Service Connect mengharuskan Amazon VPC dan subnet memiliki aturan tabel rute dan aturan ACL jaringan yang memungkinkan `containerPort` `ingressPort0verride` dan nomor port yang Anda gunakan.

Anda dapat mengirim lalu lintas antar VPC dengan Service Connect. Anda harus mempertimbangkan persyaratan yang sama untuk aturan tabel rute, ACL jaringan, dan grup keamanan yang berlaku untuk kedua VPC.

Misalnya, dua cluster membuat tugas di VPC yang berbeda. Layanan di setiap cluster dikonfigurasi untuk menggunakan namespace yang sama. Aplikasi dalam dua layanan ini dapat menyelesaikan setiap titik akhir di namespace tanpa konfigurasi DNS VPC apa pun. Namun, proxy tidak dapat terhubung kecuali VPC peering, VPC atau tabel rute subnet, dan ACL jaringan VPC memungkinkan lalu lintas pada dan nomor port yang Anda gunakan. `containerPort` `ingressPort0verride`

Untuk tugas yang menggunakan mode `bridge` jaringan, Anda harus membuat grup keamanan dengan aturan masuk yang memungkinkan lalu lintas pada rentang port dinamis atas. Kemudian, tetapkan grup keamanan ke semua instans EC2 di kluster Service Connect.

Proksi Service Connect

Jika Anda membuat atau memperbarui layanan Amazon ECS dengan konfigurasi Service Connect, Amazon ECS menambahkan wadah baru ke setiap tugas baru saat dimulai. Pola penggunaan wadah terpisah ini disebut `asidecar`. Wadah ini tidak ada dalam definisi tugas dan Anda tidak dapat mengonfigurasinya. Amazon ECS mengelola konfigurasi wadah ini di layanan Amazon ECS. Karena itu, Anda dapat menggunakan kembali definisi tugas yang sama antara beberapa layanan Amazon ECS, ruang nama, dan Anda dapat menjalankan tugas tanpa Service Connect juga.

Sumber daya proxy

- CPU tugas dan batas memori adalah satu-satunya parameter yang perlu Anda konfigurasikan untuk wadah ini dalam definisi tugas. Satu-satunya parameter yang perlu Anda konfigurasikan untuk wadah ini di layanan Amazon ECS adalah konfigurasi log, yang akan Anda temukan di konfigurasi Service Connect. Untuk informasi selengkapnya tentang konfigurasi Service Connect, lihatKonfigurasi layanan Service Connect.
- Definisi tugas harus menetapkan batas memori tugas untuk menggunakan Service Connect. CPU dan memori tambahan dalam batas tugas yang tidak Anda alokasikan dalam batas kontainer di container Anda yang lain digunakan oleh container proxy Service Connect dan container lain yang tidak menetapkan batas kontainer.

- Sebaiknya tambahkan 256 unit CPU dan setidaknya 64 MiB memori ke CPU tugas dan memori Anda untuk wadah proxy Service Connect. Di AWS Fargate, jumlah memori terendah yang dapat Anda atur adalah memori 512 MiB. Di Amazon EC2, memori tugas bersifat opsional, tetapi diperlukan untuk Service Connect.
- Jika Anda mengharapkan tugas dalam layanan ini menerima lebih dari 500 permintaan per detik pada beban puncaknya, sebaiknya tambahkan 512 unit CPU ke CPU tugas Anda dalam definisi tugas ini untuk container proxy Service Connect.
- Jika Anda berharap untuk membuat lebih dari 100 layanan Service Connect di namespace atau 2000 tugas secara total di semua layanan Amazon ECS dalam namespace, sebaiknya tambahkan 128 MiB memori ke memori tugas Anda untuk container proxy Service Connect. Anda harus melakukan ini di setiap definisi tugas yang digunakan oleh semua layanan Amazon ECS di namespace.

Konfigurasi proxy

Aplikasi Anda terhubung ke proxy di wadah sespan dalam tugas yang sama dengan aplikasi. Amazon ECS mengonfigurasi tugas dan kontainer sehingga aplikasi hanya terhubung ke proxy jika aplikasi terhubung ke nama titik akhir di namespace yang sama. Semua lalu lintas lainnya tidak menggunakan proxy. Lalu lintas lainnya termasuk alamat IP di VPC yang sama, titik akhir AWS layanan, dan lalu lintas eksternal.

Penyeimbangan beban

Service Connect mengonfigurasi proxy untuk menggunakan strategi round-robin untuk load balancing antara tugas di titik akhir Service Connect. Proxy lokal yang ada dalam tugas dari mana koneksi berasal, memilih salah satu tugas dalam layanan client-server yang menyediakan titik akhir.

Misalnya, pertimbangkan tugas yang berjalan WordPress di layanan Amazon ECS yang dikonfigurasi sebagai layanan klien di namespace yang disebut lokal. Ada layanan lain dengan 2 tugas yang menjalankan database MySQL. Layanan ini dikonfigurasi untuk menyediakan titik akhir yang dipanggil `mysql` melalui Service Connect di namespace yang sama. Dalam WordPress tugas tersebut, WordPress aplikasi terhubung ke database menggunakan nama endpoint. Karena konfigurasi Service Connect, koneksi ke nama ini masuk ke proxy yang berjalan dalam wadah sespan dalam tugas yang sama. Kemudian, proxy dapat terhubung ke salah satu tugas MySQL menggunakan strategi round-robin.

Strategi penyeimbangan beban: round-robin

Deteksi outlier

Fitur ini menggunakan data yang dimiliki proxy tentang koneksi gagal sebelumnya untuk menghindari pengiriman koneksi baru ke host yang memiliki koneksi gagal. Service Connect mengonfigurasi fitur deteksi outlier dari proxy untuk memberikan pemeriksaan kesehatan pasif.

Misalnya, pertimbangkan tugas yang berjalan WordPress di layanan Amazon ECS yang dikonfigurasi sebagai layanan klien di namespace yang disebut. `local` Ada layanan lain dengan 2 tugas yang menjalankan database MySQL. Layanan ini dikonfigurasi untuk menyediakan titik akhir yang dipanggil `mysql` melalui Service Connect di namespace yang sama. Dalam WordPress tugas, WordPress aplikasi terhubung ke proxy yang berjalan dalam wadah sespan dalam tugas yang sama. Proxy dapat terhubung ke salah satu tugas MySQL. Jika proxy membuat beberapa koneksi ke tugas MySQL tertentu, dan 5 atau lebih koneksi gagal dalam 30 detik terakhir, maka proxy menghindari tugas MySQL itu selama 30 hingga 300 detik.

Percobaan ulang

Service Connect mengonfigurasi proxy untuk mencoba lagi koneksi yang melewati proxy dan gagal, dan upaya kedua menghindari penggunaan host dari koneksi sebelumnya. Ini memastikan bahwa setiap koneksi melalui Service Connect tidak gagal karena alasan satu kali.

Jumlah percobaan ulang: 2

Waktu habis

Service Connect mengonfigurasi proxy untuk menunggu waktu maksimum agar aplikasi client-server Anda merespons. Nilai batas waktu default adalah 15 detik, tetapi dapat diperbarui.

Parameter opsional:

IdleTimeout - Jumlah waktu dalam detik koneksi akan tetap aktif saat idle. Nilai yang `0` dinonaktifkan`idleTimeout`.

`idleTimeout`Default untuk`HTTP/HTTP2/GRPC`adalah 5 menit.

`idleTimeout`Default untuk `TCP` adalah 1 jam.

perRequestTimeout- Jumlah waktu menunggu hulu untuk merespons dengan respons lengkap per permintaan. Nilai yang `0` dinonaktifkan`perRequestTimeout`. Hanya dapat diatur ketika wadah `appProtocol` untuk aplikasi adalah`HTTP/HTTP2/GRPC`. Defaultnya adalah 15 detik.

> **Note**
>
> Jika `idleTimeout` diatur ke waktu yang kurang dari`perRequestTimeout`, koneksi akan ditutup ketika `idleTimeout` tercapai dan bukan`perRequestTimeout`.

Pertimbangan Service Connect

- Kontainer Windows tidak didukung dengan Service Connect.
- Tugas yang berjalan di Fargate harus menggunakan platform Fargate Linux versi 1.4.0 atau lebih tinggi untuk menggunakan Service Connect.
- Versi agen ECS pada instance container harus 1.67.2 atau lebih tinggi.
- Instans penampung harus menjalankan Amazon ECS yang dioptimalkan Amazon Linux 2023 versi AMI atau yang lebih baru, atau `20230428` versi Amazon Linux 2 AMI Amazon ECS yang dioptimalkan untuk menggunakan Service Connect. `2.0.20221115` Versi ini memiliki agen Service Connect selain agen kontainer Amazon ECS. Untuk informasi selengkapnya tentang agen Service Connect, lihat Agen Amazon ECS Service Connect di GitHub.
- Instance kontainer harus memiliki `ecs:Poll` izin untuk sumber `arn:aws:ecs:`*`region`*`:`*`0123456789012`*`:task-set/`*`cluster`*`/*` daya. Jika Anda menggunakan`ecsInstanceRole`, Anda tidak perlu menambahkan izin tambahan. Kebijakan `AmazonEC2ContainerServiceforEC2Role` terkelola memiliki izin yang diperlukan. Untuk informasi selengkapnya, lihat Peran IAM instans wadah Amazon ECS.
- `External`instance container untuk Amazon ECS Anywhere tidak didukung dengan Service Connect.
- Hanya layanan yang menggunakan penerapan bergulir yang didukung dengan Service Connect. Layanan yang menggunakan jenis penerapan biru/hijau dan eksternal tidak didukung.
- Definisi tugas harus menetapkan batas memori tugas untuk menggunakan Service Connect. Untuk informasi selengkapnya, lihat Proksi Service Connect.
- Definisi tugas yang menetapkan batas memori kontainer untuk semua kontainer alih-alih menyetel batas memori tugas tidak didukung.

  Anda dapat mengatur batas memori kontainer pada kontainer Anda, tetapi Anda harus mengatur batas memori tugas ke angka yang lebih besar dari jumlah batas memori kontainer. CPU dan memori tambahan dalam batas tugas yang tidak dialokasikan dalam batas kontainer digunakan

oleh kontainer proxy Service Connect dan wadah lain yang tidak menetapkan batas kontainer. Untuk informasi selengkapnya, lihat Proksi Service Connect.

- Anda dapat mengonfigurasi Service Connect di layanan untuk menggunakan AWS Cloud Map namespace apa pun Wilayah AWS dalam hal yang sama. Akun AWS
- Setiap layanan Amazon ECS hanya dapat dimiliki oleh satu namespace.
- Hanya tugas yang dibuat oleh layanan Amazon ECS yang didukung. Tugas mandiri tidak dapat dikonfigurasi untuk Service Connect.
- Semua titik akhir harus unik dalam namespace.
- Semua nama penemuan harus unik dalam namespace.
- Layanan yang ada harus digunakan kembali sebelum aplikasi di dalamnya dapat menyelesaikan titik akhir baru. Titik akhir baru yang ditambahkan ke namespace setelah penerapan terbaru tidak akan ditambahkan ke konfigurasi tugas. Untuk informasi selengkapnya, lihat the section called “Urutan penyebaran”.
- Anda dapat membuat namespace saat membuat cluster baru. Amazon ECS Service Connect tidak menghapus ruang nama saat cluster dihapus. Anda harus menghapus ruang nama secara langsung AWS Cloud Map jika Anda selesai menggunakannya.
- Service Connect tidak mendukung HTTP 1.0.
- Lalu lintas Application Load Balancer default ke routing melalui agen Service Connect dalam mode jaringan. `awsvpc` Jika Anda ingin lalu lintas non-layanan melewati agen Service Connect, gunakan `ingressPortOverride` parameter dalam konfigurasi layanan Service Connect Anda.

Pengalaman konsol Service Connect

Untuk membuat namespace baru, buat cluster Amazon ECS baru menggunakan konsol Amazon ECS dan tentukan nama namespace yang akan dibuat, atau gunakan konsol. AWS Cloud Map Amazon ECS Service Connect dapat menggunakan jenis AWS Cloud Map namespace penemuan instans apa pun. Kami merekomendasikan jenis panggilan API untuk membuat jumlah minimum sumber daya tambahan. Untuk membuat cluster dan namespace Amazon ECS baru di konsol Amazon ECS, lihat. Membuat cluster untuk jenis peluncuran Fargate dan Eksternal menggunakan konsol

Setiap AWS Cloud Map namespace dalam hal ini Akun AWS dalam yang dipilih Wilayah AWS ditampilkan di Namespaces di konsol Amazon ECS.

Untuk menghapus namespace, gunakan konsol. AWS Cloud Map Namespace harus kosong sebelum dapat dihapus.

Untuk membuat definisi tugas Amazon ECS baru, atau mendaftarkan revisi baru ke definisi tugas yang ada dan menggunakan Service Connect, lihat. Membuat definisi tugas menggunakan konsol

Untuk membuat layanan Amazon ECS baru yang menggunakan Service Connect, lihatMembuat layanan menggunakan konsol.

Harga Service Connect

Harga Amazon ECS Service Connect bergantung pada apakah Anda menggunakan AWS Fargate atau infrastruktur Amazon EC2 untuk meng-host beban kerja kontainer Anda. Saat menggunakan Amazon ECS aktif AWS Outposts, harga mengikuti model yang sama yang digunakan saat Anda menggunakan Amazon EC2 secara langsung. Untuk informasi selengkapnya, lihat Harga Amazon ECS.

AWS Cloud Map penggunaan sepenuhnya gratis, ketika dikonsumsi melalui Amazon ECS Service Connect.

TLS dengan Service Connect

Amazon ECS Service Connect mendukung enkripsi lalu lintas otomatis dengan sertifikat Transport Layer Security (TLS) untuk layanan Amazon ECS. Saat Anda mengarahkan layanan Amazon ECS ke AWS Private Certificate Authority (AWS Private CA), Amazon ECS secara otomatis menyediakan sertifikat TLS untuk mengenkripsi lalu lintas antara layanan Amazon ECS Service Connect Anda. Amazon ECS menghasilkan, memutar, dan mendistribusikan sertifikat TLS yang digunakan untuk enkripsi lalu lintas.

Enkripsi lalu lintas otomatis dengan Amazon ECS Service Connect menggunakan kemampuan enkripsi terdepan di industri untuk mengamankan komunikasi antar-layanan yang membantu Anda memenuhi persyaratan keamanan. Ini mendukung sertifikat AWS Private Certificate Authority TLS dengan `256-bit ECDSA` dan `2048-bit RSA` enkripsi. Secara default, TLS 1.3 didukung, tetapi TLS 1.0 - 1.2 tidak didukung. Anda juga memiliki kontrol penuh atas sertifikat pribadi dan kunci penandatanganan untuk membantu Anda memenuhi persyaratan kepatuhan.

> **Note**
>
> Hanya lalu lintas masuk dan keluar yang lewat melalui Amazon ECS Service Connect Agent dienkripsi.

AWS Private Certificate Authority Sertifikat dan Service Connect

AWS Private Certificate Authority mode untuk Service Connect

AWS Private Certificate Authority dapat berjalan dalam dua mode: tujuan umum dan berumur pendek.

- Tujuan umum - Sertifikat yang dapat dikonfigurasi dengan tanggal kedaluwarsa apa pun.
- Berumur pendek - Sertifikat dengan validitas maksimum tujuh hari.

Meskipun Amazon ECS mendukung kedua mode, kami sarankan untuk menggunakan sertifikat berumur pendek. Secara default, sertifikat berputar setiap lima hari, dan berjalan dalam mode berumur pendek menawarkan penghematan biaya yang signifikan dibandingkan tujuan umum.

Service Connect tidak mendukung pencabutan sertifikat dan sebaliknya memanfaatkan sertifikat berumur pendek dengan rotasi sertifikat yang sering. Anda memiliki wewenang untuk memodifikasi frekuensi rotasi, menonaktifkan, atau menghapus rahasia menggunakan rotasi terkelola di Secrets Manager, tetapi hal itu dapat disertai dengan konsekuensi yang mungkin berikut.

- Frekuensi Rotasi Lebih Pendek - Frekuensi rotasi yang lebih pendek menimbulkan biaya yang lebih tinggi karena AWS Private CA, AWS KMS dan Secrets Manager, dan Auto Scaling mengalami peningkatan beban kerja untuk rotasi.
- Frekuensi Rotasi Lebih Panjang - Komunikasi aplikasi Anda gagal jika frekuensi rotasi melebihi tujuh hari.
- Penghapusan Rahasia - Menghapus rahasia menghasilkan kegagalan rotasi dan berdampak pada komunikasi aplikasi pelanggan.

Jika rotasi rahasia Anda gagal, sebuah `RotationFailed` acara dipublikasikan di AWS CloudTrail. Anda juga dapat mengatur CloudWatch Alarm untuk`RotationFailed`.

Otoritas Sertifikat Bawahan

Anda dapat membawa apa saja AWS Private CA, root atau bawahan, ke Amazon ECS Service Connect TLS untuk menerbitkan sertifikat entitas akhir untuk layanan. Penerbit yang disediakan diperlakukan sebagai penandatangan dan akar kepercayaan di mana-mana. Anda dapat menerbitkan sertifikat entitas akhir untuk berbagai bagian aplikasi Anda dari CA bawahan yang berbeda. Saat menggunakan AWS CLI, berikan Nama Sumber Daya Amazon (ARN) CA untuk membangun rantai kepercayaan.

Otoritas Sertifikat Lokal

Untuk menggunakan CA lokal, Anda membuat dan mengonfigurasi CA bawahan. AWS Private Certificate Authority Ini memastikan semua sertifikat TLS yang dikeluarkan untuk beban kerja Amazon ECS Anda berbagi rantai kepercayaan dengan beban kerja yang Anda jalankan di tempat dan dapat terhubung dengan aman.

> **Important**
>
> Harap tambahkan tag yang diperlukan `AmazonECSManaged : true` di Anda AWS Private CA.

Kebijakan IAM AmazonECS InfrastructureRolePolicyForServiceConnectTransport LayerSecurity

Amazon ECS menyediakan kebijakan kepercayaan sumber daya terkelola baru yang menguraikan kumpulan izin yang diperlukan untuk menerbitkan sertifikat. Untuk informasi selengkapnya tentang kebijakan baru ini, lihat AmazonECS di kebijakan InfrastructureRolePolicyForServiceConnectTransport LayerSecurity AWS terkelola untuk Amazon ECS.

Service Connect dan AWS Key Management Service

Anda dapat menggunakan AWS Key Management Serviceuntuk mengenkripsi dan mendekripsi sumber daya Service Connect Anda. AWS KMS adalah layanan yang dikelola oleh AWS tempat Anda dapat membuat dan mengelola kunci kriptografi yang melindungi data Anda.

Saat menggunakan AWS KMS Service Connect, Anda dapat memilih untuk menggunakan kunci AWS milik yang AWS mengelola untuk Anda, atau Anda dapat memilih AWS KMS kunci yang ada. Anda juga dapat membuat AWS KMS kunci baru untuk digunakan.

Menyediakan kunci enkripsi Anda sendiri

Anda dapat menyediakan materi utama Anda sendiri, atau Anda dapat menggunakan penyimpanan kunci eksternal melalui AWS Key Management Service Impor kunci Anda sendiri AWS KMS, lalu tentukan Nama Sumber Daya Amazon (ARN) kunci tersebut di Amazon ECS Service Connect.

Berikut ini adalah contoh AWS KMS kebijakan. Ganti nilai *merah* dengan nilai Anda sendiri.

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Sid": "id",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/role-name"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:GenerateDataKey",
        "kms:GenerateDataKeyPair"
      ],
      "Resource": "*"
    }
  ]
}
```

Untuk informasi selengkapnya tentang kebijakan utama, lihat Membuat kebijakan kunci di Panduan AWS Key Management Service Pengembang.

> **Note**
>
> Service Connect hanya mendukung AWS KMS kunci enkripsi simetris. Anda tidak dapat menggunakan jenis AWS KMS kunci lain untuk mengenkripsi sumber daya Service Connect. Untuk bantuan menentukan apakah AWS KMS kunci adalah kunci enkripsi simetris, lihat Mengidentifikasi kunci simetris dan AWS KMS asimetris.

Untuk informasi selengkapnya tentang kunci enkripsi AWS Key Management Service simetris, lihat AWS KMS Kunci enkripsi simetris di Panduan AWS Key Management Service Pengembang.

Aktifkan TLS dengan Service Connect

Anda mengaktifkan enkripsi lalu lintas saat membuat atau memperbarui layanan Service Connect.

Untuk mengaktifkan enkripsi lalu lintas untuk layanan di namespace yang ada menggunakan AWS Management Console

1. Pilih Namespace dengan Layanan yang ingin Anda aktifkan enkripsi lalu lintas.
2. Pilih Layanan yang ingin Anda aktifkan enkripsi lalu lintas.

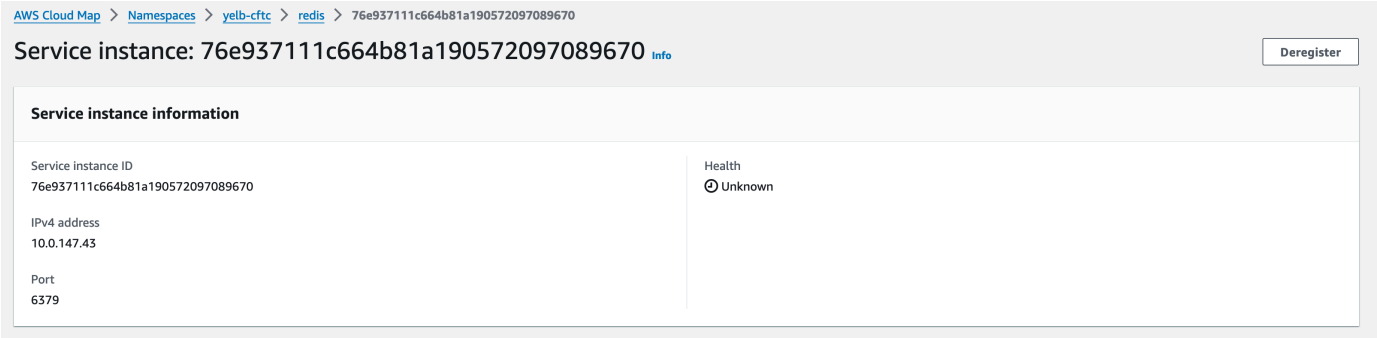
3. Pilih Perbarui Layanan di sudut kanan atas dan gulir ke bawah ke bagian Service Connect.
4. Pilih Aktifkan enkripsi lalu lintas di bawah informasi layanan Anda untuk mengaktifkan TLS.
5. Untuk peran Service Connect TLS, pilih peran yang ada atau buat yang baru.
6. Untuk otoritas sertifikat Penandatangan, pilih otoritas sertifikat yang ada atau untuk membuat yang baru.
7. Untuk Pilih AWS KMS key, pilih kunci yang AWS dimiliki dan dikelola atau Anda dapat memilih kunci yang berbeda. Anda juga dapat memilih untuk membuat yang baru.

Untuk contoh penggunaan untuk mengkonfigurasi TLS AWS CLI untuk layanan Anda, lihat Tutorial: Menggunakan Service Connect in Fargate with the. AWS CLI

Verifikasi TLS diaktifkan

Service Connect memulai TLS di agen Service Connect dan menghentikannya di agen tujuan. Akibatnya, kode aplikasi tidak pernah melihat interaksi TLS. Untuk memverifikasi bahwa TLS diaktifkan, Anda dapat menggunakan langkah-langkah berikut.

1. Pastikan gambar aplikasi Anda memiliki `openssl` CLI.
2. Aktifkan ECS Exec pada layanan Anda untuk terhubung ke tugas Anda melalui SSM. Sebagai alternatif, Anda dapat memutar instans Amazon EC2 di VPC Amazon yang sama dengan layanan.
3. Ambil IP dan port tugas dari layanan yang ingin Anda verifikasi. Misalnya, jika `redis` layanan Anda mengaktifkan TLS, Anda dapat mengambil IP tugasnya dengan menavigasi ke AWS Cloud Map, menemukan layanan, dan melihat IP dan port dari satu instance.



4. Masuk ke salah satu tugas Amazon ECS Anda menggunakan `execute-command` seperti pada contoh berikut. Sebagai alternatif, masuk ke instans Amazon EC2 yang dibuat di Langkah 2.

```
$ aws ecs execute-command --cluster cluster-name \
    --task < TASK_ID>  \
```

```
    --container app \
    --interactive \
    --command "/bin/sh"
```

> **Note**
>
> Memanggil nama DNS secara langsung tidak mengungkapkan sertifikat.

5. Di shell yang terhubung, gunakan `openssl` CLI untuk memverifikasi dan melihat sertifikat yang dilampirkan pada tugas.

   Contoh:

```
openssl s_client -connect 10.0.147.43:6379 < /dev/null 2> /dev/null \
| openssl x509 -noout -text
```

   Contoh respons:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            <serial-number>
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: <issuer>
        Validity
            Not Before: Jan 23 21:38:12 2024 GMT
            Not After : Jan 30 22:38:12 2024 GMT
        Subject: <subject>
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    <pub>
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Subject Alternative Name:
                DNS:redis.yelb-cftc
            X509v3 Basic Constraints:
                CA:FALSE
```

```
            X509v3 Authority Key Identifier:
                keyid:<key-id>

            X509v3 Subject Key Identifier:
                1D:<id>
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
    Signature Algorithm: ecdsa-with-SHA256
        <hash>
```

Parameter Service Connect

Parameter berikut memiliki bidang tambahan saat menggunakan Service Connect.

Lokasi parameter	Jenis aplikasi	Deskripsi	Diperlukan?
Ketentuan tugas	Klien	Tidak ada perubahan yang tersedia untuk Service Connect dalam definisi tugas klien.	N/A
Ketentuan tugas	Peladen-klien	Server harus menambahkan `name` bidang ke port di `portMappings` wadah. Lihat informasi yang lebih lengkap di `portMappings`	Ya
Ketentuan tugas	Peladen-klien	Server secara opsional dapat menyediakan protokol aplikasi (misalnya, HTTP) untuk menerima metrik spesifik protokol untuk aplikasi server mereka (misalnya,). `HTTP 5xx`	Tidak
Definisi layanan	Klien	Layanan klien harus menambahkan a `serviceConnectConfiguration` untuk mengkonfigurasi namespace untuk bergabung. Namespace ini harus berisi semua layanan server	Ya

Lokasi parameter	Jenis aplikasi	Deskripsi	Diperlukan?
		yang perlu ditemukan oleh layanan ini. Untuk informasi selengkapnya, lihat `serviceConnectConfiguration`.	
Definisi layanan	Peladen-klien	Layanan server harus menambahkan a `serviceConnectConfiguration` untuk mengonfigurasi nama DNS, nomor port, dan namespace tempat layanan tersedia. Untuk informasi selengkapnya, lihat `serviceConnectConfiguration`.	Ya
Klaster	Klien	Cluster dapat menambahkan namespace Service Connect default. Layanan baru di cluster mewarisi namespace saat Service Connect dikonfigurasi dalam layanan. Untuk informasi selengkapnya, lihat klaster Amazon ECS.	Tidak
Klaster	Peladen-klien	Tidak ada perubahan yang tersedia untuk Service Connect di cluster yang berlaku untuk layanan server. Definisi tugas server dan layanan harus mengatur konfigurasi masing-masing.	N/A

Menggunakan Service Connect di Fargate dengan AWS CLI

Tutorial berikut menunjukkan cara membuat layanan Amazon ECS dengan tugas Fargate yang menggunakan Service Connect dengan. AWS CLI

Amazon ECS mendukung fitur Service Connect dalam Wilayah AWS Regions with Service Connect daftar.

Prasyarat

Tutorial ini mengasumsikan bahwa prasyarat berikut telah diselesaikan:

- Versi terbaru diinstal dan dikonfigurasi. AWS CLI Untuk informasi lebih lanjut, lihat Menginstal AWS Command Line Interface.
- Langkah-langkah di Siapkan untuk menggunakan Amazon ECS telah selesai.
- AWS Pengguna Anda memiliki izin yang diperlukan yang ditentukan dalam contoh kebijakan Amazonecs_ FullAccess IAM.
- Anda memiliki VPC, subnet, tabel rute, dan grup keamanan yang dibuat untuk digunakan. Untuk informasi selengkapnya, lihat the section called “Buat virtual private cloud”.
- Anda memiliki peran eksekusi tugas dengan nama `ecsTaskExecutionRole` dan kebijakan `AmazonECSTaskExecutionRolePolicy` terkelola dilampirkan ke peran. Peran ini memungkinkan Fargate untuk menulis log aplikasi NGINX dan log proxy Service Connect ke Amazon Logs. CloudWatch Untuk informasi selengkapnya, lihat Membuat peran eksekusi tugas (`ecsTaskExecutionRole`).

Langkah 1: Buat cluster Amazon ECS

Gunakan langkah-langkah berikut untuk membuat cluster dan namespace Amazon ECS Anda.

Untuk membuat cluster dan namespace Amazon ECS AWS Cloud Map

1. Buat klaster Amazon ECS bernama `tutorial` untuk digunakan. Parameter `--service-connect-defaults` menetapkan namespace default cluster. Dalam contoh keluaran, AWS Cloud Map namespace nama `service-connect` tidak ada di akun ini dan Wilayah AWS, jadi namespace dibuat oleh Amazon ECS. Namespace dibuat AWS Cloud Map di akun, dan terlihat dengan semua ruang nama lainnya, jadi gunakan nama yang menunjukkan tujuannya.

   ```
   aws ecs create-cluster --cluster-name tutorial --service-connect-defaults
    namespace=service-connect
   ```

   Output:

   ```
   {
       "cluster": {
           "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/tutorial",
           "clusterName": "tutorial",
   ```

```
        "serviceConnectDefaults": {
            "namespace": "arn:aws:servicediscovery:us-
west-2:123456789012:namespace/ns-EXAMPLE"
        },
        "status": "PROVISIONING",
        "registeredContainerInstancesCount": 0,
        "runningTasksCount": 0,
        "pendingTasksCount": 0,
        "activeServicesCount": 0,
        "statistics": [],
        "tags": [],
        "settings": [
            {
                "name": "containerInsights",
                "value": "disabled"
            }
        ],
        "capacityProviders": [],
        "defaultCapacityProviderStrategy": [],
        "attachments": [
            {
                "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
                "type": "sc",
                "status": "ATTACHING",
                "details": []
            }
        ],
        "attachmentsStatus": "UPDATE_IN_PROGRESS"
    }
}
}
```

2. Verifikasi bahwa cluster dibuat:

```
aws ecs describe-clusters --clusters tutorial
```

Output:

```
{
    "clusters": [
        {
            "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/tutorial",
            "clusterName": "tutorial",
```

```
            "serviceConnectDefaults": {
                "namespace": "arn:aws:servicediscovery:us-
west-2:123456789012:namespace/ns-EXAMPLE"
            },
            "status": "ACTIVE",
            "registeredContainerInstancesCount": 0,
            "runningTasksCount": 0,
            "pendingTasksCount": 0,
            "activeServicesCount": 0,
            "statistics": [],
            "tags": [],
            "settings": [],
            "capacityProviders": [],
            "defaultCapacityProviderStrategy": []
        }
    ],
    "failures": []
}
```

3. (Opsional) Verifikasi bahwa namespace dibuat di. AWS Cloud Map Anda dapat menggunakan AWS Management Console atau AWS CLI konfigurasi normal saat ini dibuat AWS Cloud Map.

   Misalnya, gunakan AWS CLI:

   ```
   aws servicediscovery --region us-west-2 get-namespace --id ns-EXAMPLE
   ```

   Output:

   ```
   {
       "Namespace": {
           "Id": "ns-EXAMPLE",
           "Arn": "arn:aws:servicediscovery:us-west-2:123456789012:namespace/ns-
   EXAMPLE",
           "Name": "service-connect",
           "Type": "HTTP",
           "Properties": {
               "DnsProperties": {
                   "SOA": {}
               },
               "HttpProperties": {
                   "HttpName": "service-connect"
               }
   ```

```
        },
        "CreateDate": 1661749852.422,
        "CreatorRequestId": "service-connect"
    }
}
```

Langkah 2: Buat layanan Amazon ECS untuk server

Fitur Service Connect ditujukan untuk menghubungkan beberapa aplikasi di Amazon ECS. Setidaknya salah satu dari aplikasi tersebut perlu menyediakan layanan web untuk terhubung. Pada langkah ini, Anda membuat:

- Definisi tugas yang menggunakan image kontainer NGINX resmi yang tidak dimodifikasi dan menyertakan konfigurasi Service Connect.
- Definisi layanan Amazon ECS yang mengonfigurasi Service Connect untuk menyediakan penemuan layanan dan proxy mesh layanan untuk lalu lintas ke layanan ini. Konfigurasi menggunakan kembali namespace default dari konfigurasi cluster untuk mengurangi jumlah konfigurasi layanan yang Anda buat untuk setiap layanan.
- Layanan Amazon ECS. Ini menjalankan satu tugas menggunakan definisi tugas, dan menyisipkan wadah tambahan untuk proxy Service Connect. Proxy mendengarkan port dari pemetaan port kontainer dari definisi tugas. Dalam aplikasi klien yang berjalan di Amazon ECS, proxy dalam tugas klien mendengarkan koneksi keluar ke nama port definisi tugas, nama penemuan layanan atau nama alias klien layanan, dan nomor port dari alias klien.

Untuk membuat layanan web dengan Amazon ECS Service Connect

1. Daftarkan definisi tugas yang kompatibel dengan Fargate dan gunakan mode `awsvpc` jaringan. Ikuti langkah-langkah ini:

    a. Buat file yang diberi nama `service-connect-nginx.json` dengan isi definisi tugas berikut.

    Definisi tugas ini mengonfigurasi Service Connect dengan menambahkan `name` dan `appProtocol` parameter ke pemetaan port. Nama port membuat port ini lebih dapat diidentifikasi dalam konfigurasi layanan ketika beberapa port digunakan. Nama port juga digunakan secara default sebagai nama yang dapat ditemukan untuk digunakan oleh aplikasi lain di namespace.

Definisi tugas berisi peran IAM tugas karena layanan mengaktifkan ECS Exec.

> **Important**
>
> Definisi tugas ini menggunakan a `logConfiguration` untuk mengirim output nginx dari `stdout` dan `stderr` ke Amazon Logs. CloudWatch Peran eksekusi tugas ini tidak memiliki izin tambahan yang diperlukan untuk membuat grup CloudWatch log Log. Buat grup log di CloudWatch Log menggunakan AWS Management Console or AWS CLI. Jika Anda tidak ingin mengirim log nginx ke Log, Anda CloudWatch dapat menghapus file. `logConfiguration`
> Ganti Akun AWS id dalam peran eksekusi dengan Akun AWS id Anda.

```
{
    "family": "service-connect-nginx",
    "executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::123456789012:role/ecsTaskRole",
    "networkMode": "awsvpc",
    "containerDefinitions": [
        {
        "name": "webserver",
        "image": "public.ecr.aws/docker/library/nginx:latest",
        "cpu": 100,
        "portMappings": [
            {
                "name": "nginx",
                "containerPort": 80,
                "protocol": "tcp",
                "appProtocol": "http"
            }
        ],
        "essential": true,
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-nginx",
                "awslogs-region": "region",
                "awslogs-stream-prefix": "nginx"
            }
```

```
        }
        }
    ],
    "cpu": "256",
    "memory": "512"
}
```

b. Daftarkan definisi tugas menggunakan `service-connect-nginx.json` file:

```
aws ecs register-task-definition --cli-input-json file://service-connect-nginx.json
```

2. Buat layanan ECS dengan mengikuti langkah-langkah berikut:

a. Buat file yang diberi nama `service-connect-nginx-service.json` dengan konten layanan Amazon ECS yang Anda buat. Contoh ini menggunakan definisi tugas yang dibuat pada langkah sebelumnya. `awsvpcConfiguration` diperlukan karena ketentuan tugas contoh menggunakan mode jaringan `awsvpc`.

Saat Anda membuat layanan ECS, tentukan jenis peluncuran Fargate, dan versi platform `LATEST` yang mendukung Service Connect. Itu `securityGroups` dan `subnets` harus milik VPC yang memiliki persyaratan untuk menggunakan Amazon ECS. Anda dapat memperoleh grup keamanan dan ID subnet dari Konsol VPC Amazon.

Layanan ini mengkonfigurasi Service Connect dengan menambahkan `serviceConnectConfiguration` parameter. Namespace tidak diperlukan karena cluster memiliki namespace default yang dikonfigurasi. Aplikasi klien yang berjalan di ECS di namespace terhubung ke layanan ini dengan menggunakan `portName` dan port di. `clientAliases` Misalnya, layanan ini dapat dijangkau menggunakan`http://nginx:80/`, karena nginx menyediakan halaman selamat datang di lokasi root. `/` Aplikasi eksternal yang tidak berjalan di Amazon ECS atau tidak berada di namespace yang sama dapat menjangkau aplikasi ini melalui proxy Service Connect dengan menggunakan alamat IP tugas dan nomor port dari definisi tugas. Untuk `tls` konfigurasi Anda, tambahkan sertifikat `arn` untuk peran IAM Anda `awsPcaAuthorityArnkmsKey`, Anda, dan `roleArn` Anda.

Layanan ini menggunakan a `logConfiguration` untuk mengirim layanan menghubungkan output proxy dari `stdout` dan `stderr` ke Amazon CloudWatch Logs. Peran eksekusi tugas ini tidak memiliki izin tambahan yang diperlukan untuk membuat grup

CloudWatch log Log. Buat grup log di CloudWatch Log menggunakan AWS Management Console or AWS CLI. Kami menyarankan Anda membuat grup log ini dan menyimpan log proxy di CloudWatch Log. Jika Anda tidak ingin mengirim log proxy ke CloudWatch Log, Anda dapat menghapus`logConfiguration`.

```
{
    "cluster": "tutorial",
    "deploymentConfiguration": {
        "maximumPercent": 200,
        "minimumHealthyPercent": 0
    },
    "deploymentController": {
        "type": "ECS"
    },
    "desiredCount": 1,
    "enableECSManagedTags": true,
    "enableExecuteCommand": true,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "assignPublicIp": "ENABLED",
            "securityGroups": [
                "sg-EXAMPLE"
            ],
            "subnets": [
                "subnet-EXAMPLE",
                "subnet-EXAMPLE",
                "subnet-EXAMPLE"
            ]
           }
    },
    "platformVersion": "LATEST",
    "propagateTags": "SERVICE",
    "serviceName": "service-connect-nginx-service",
    "serviceConnectConfiguration": {
        "enabled": true,
        "services": [
            {
                "portName": "nginx",
                "clientAliases": [
                    {
                        "port": 80
```

```
                    }
                ],
                "tls": {
                   "issuerCertificateAuthority": {
                      "awsPcaAuthorityArn": "certificateArn"
                   },
                   "kmsKey": "kmsKey",
                   "roleArn": "iamRoleArn"
                }
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "region",
                "awslogs-stream-prefix": "service-connect-proxy"
            }
        }
    },
    "taskDefinition": "service-connect-nginx"
}
```

b. Buat layanan ECS menggunakan `service-connect-nginx-service.json` file:

```
aws ecs create-service --cluster tutorial --cli-input-json file://service-
connect-nginx-service.json
```

Output:

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/tutorial/
service-connect-nginx-service",
        "serviceName": "service-connect-nginx-service",
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/tutorial",
        "loadBalancers": [],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 1,
        "runningCount": 0,
```

```
        "pendingCount": 0,
        "launchType": "FARGATE",
        "platformVersion": "LATEST",
        "platformFamily": "Linux",
        "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/
service-connect-nginx:1",
        "deploymentConfiguration": {
            "deploymentCircuitBreaker": {
                "enable": false,
                "rollback": false
            },
            "maximumPercent": 200,
            "minimumHealthyPercent": 0
        },
        "deployments": [
            {
                "id": "ecs-svc/3763308422771520962",
                "status": "PRIMARY",
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/service-connect-nginx:1",
                "desiredCount": 1,
                "pendingCount": 0,
                "runningCount": 0,
                "failedTasks": 0,
                "createdAt": 1661210032.602,
                "updatedAt": 1661210032.602,
                "launchType": "FARGATE",
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "assignPublicIp": "ENABLED",
                        "securityGroups": [
                            "sg-EXAMPLE"
                        ],
                        "subnets": [
                            "subnet-EXAMPLEf",
                            "subnet-EXAMPLE",
                            "subnet-EXAMPLE"
                        ]
                    }
                },
                "rolloutState": "IN_PROGRESS",
```

```
                "rolloutStateReason": "ECS deployment ecs-
svc/3763308422771520962 in progress.",
                "failedLaunchTaskCount": 0,
                "replacedTaskCount": 0,
                "serviceConnectConfiguration": {
                    "enabled": true,
                    "namespace": "service-connect",
                    "services": [
                        {
                            "portName": "nginx",
                            "clientAliases": [
                                {
                                    "port": 80
                                }
                            ]
                        }
                    ],
                    "logConfiguration": {
                        "logDriver": "awslogs",
                        "options": {
                            "awslogs-group": "/ecs/service-connect-proxy",
                            "awslogs-region": "us-west-2",
                            "awslogs-stream-prefix": "service-connect-proxy"
                        },
                        "secretOptions": []
                    }
                },
                "serviceConnectResources": [
                    {
                        "discoveryName": "nginx",
                        "discoveryArn": "arn:aws:servicediscovery:us-
west-2:123456789012:service/srv-EXAMPLE"
                    }
                ]
            }
        ],
        "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
        "version": 0,
        "events": [],
        "createdAt": 1661210032.602,
        "placementConstraints": [],
        "placementStrategy": [],
        "networkConfiguration": {
```

```
            "awsvpcConfiguration": {
                "assignPublicIp": "ENABLED",
                "securityGroups": [
                    "sg-EXAMPLE"
                ],
                "subnets": [
                    "subnet-EXAMPLE",
                    "subnet-EXAMPLE",
                    "subnet-EXAMPLE"
                ]
            }
        },
        "schedulingStrategy": "REPLICA",
        "enableECSManagedTags": true,
        "propagateTags": "SERVICE",
        "enableExecuteCommand": true
    }
}
```

`serviceConnectConfiguration`Yang Anda berikan muncul di dalam penyebaran pertama output. Saat Anda membuat perubahan pada layanan ECS dengan cara yang perlu melakukan perubahan pada tugas, penerapan baru dibuat oleh Amazon ECS.

Langkah 3: Verifikasi bahwa Anda dapat terhubung

Untuk memverifikasi bahwa Service Connect dikonfigurasi dan berfungsi, ikuti langkah-langkah berikut untuk menyambung ke layanan web dari aplikasi eksternal. Kemudian, lihat metrik tambahan CloudWatch yang dibuat oleh proxy Service Connect.

Untuk terhubung ke layanan web dari aplikasi eksternal

- Connect ke alamat IP tugas dan port kontainer menggunakan alamat IP tugas

  Gunakan AWS CLI untuk mendapatkan ID tugas, menggunakan file`aws ecs list-tasks --cluster tutorial`.

  Jika subnet dan grup keamanan Anda mengizinkan lalu lintas dari internet publik di port dari definisi tugas, Anda dapat terhubung ke IP publik dari komputer Anda. IP publik tidak tersedia dari `describe-tasks`, jadi langkah-langkahnya melibatkan pergi ke Amazon EC2 AWS Management Console atau AWS CLI untuk mendapatkan detail dari elastic network interface.

Dalam contoh ini, instans Amazon EC2 di VPC yang sama menggunakan IP pribadi tugas tersebut. Aplikasi ini nginx, tetapi `server: envoy` header menunjukkan bahwa proxy Service Connect digunakan. Proxy Service Connect mendengarkan pada port kontainer dari definisi tugas.

```
$ curl -v 10.0.19.50:80/
*   Trying 10.0.19.50:80...
* Connected to 10.0.19.50 (10.0.19.50) port 80 (#0)
> GET / HTTP/1.1
> Host: 10.0.19.50
> User-Agent: curl/7.79.1
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< server: envoy
< date: Tue, 23 Aug 2022 03:53:06 GMT
< content-type: text/html
< content-length: 612
< last-modified: Tue, 16 Apr 2019 13:08:19 GMT
< etag: "5cb5d3c3-264"
< accept-ranges: bytes
< x-envoy-upstream-service-time: 0
<
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
```

```
<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Untuk melihat metrik Service Connect

Proxy Service Connect membuat metrik aplikasi (HTTP, HTTP2, gRPC, atau koneksi TCP) dalam metrik. CloudWatch Saat Anda menggunakan CloudWatch konsol, lihat dimensi metrik tambahan DiscoveryName, (,, ClusterName) DiscoveryName ServiceName, dan (TargetDiscoveryName, TargetDiscoveryName ServiceName, ClusterName) di bawah namespace ECS. Untuk informasi selengkapnya tentang metrik dan dimensi ini, lihatMetrik dan dimensi yang tersedia untuk Amazon ECS.

Penemuan Layanan

Layanan Amazon ECS Anda secara opsional dapat dikonfigurasi untuk menggunakan penemuan layanan Amazon ECS. Penemuan layanan menggunakan tindakan AWS Cloud Map API untuk mengelola ruang nama HTTP dan DNS untuk layanan Amazon ECS Anda. Untuk informasi lebih lanjut, lihat Apa itu AWS Cloud Map? di Panduan AWS Cloud Map Pengembang.

Penemuan layanan tersedia di AWS Wilayah berikut:

Nama Wilayah	Wilayah
AS Timur (Virginia Utara)	us-east-1
US East (Ohio)	us-east-2
AS Barat (California Utara)	us-west-1
US West (Oregon)	as-barat-2
Afrika (Cape Town)	af-selatan-1

Nama Wilayah	Wilayah
Asia Pasifik (Hong Kong)	ap-timur-1
Asia Pasifik (Mumbai)	ap-south-1
Asia Pasifik (Hyderabad)	ap-south-2
Asia Pasifik (Tokyo)	ap-northeast-1
Asia Pasifik (Seoul)	ap-northeast-2
Asia Pacific (Osaka)	ap-northeast-3
Asia Pasifik (Singapura)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pasifik (Jakarta)	ap-southeast-3
Asia Pacific (Melbourne)	ap-southeast-4
Kanada (Pusat)	ca-central-1
Kanada Barat (Calgary)	ca-west-1
Tiongkok (Beijing)	cn-north-1
China (Ningxia)	cn-northwest-1
Europe (Frankfurt)	eu-central-1
Eropa (Zürich)	eu-central-2
Eropa (Irlandia)	eu-west-1
Europe (London)	eu-west-2
Europe (Paris)	eu-west-3
Eropa (Milan)	eu-south-1

Nama Wilayah	Wilayah
Eropa (Stockholm)	eu-north-1
Israel (Tel Aviv)	il-central-1
Eropa (Spanyol)	eu-south-2
Timur Tengah (UEA)	me-central-1
Timur Tengah (Bahrain)	me-selatan-1
Amerika Selatan (São Paulo)	sa-east-1
AWS GovCloud (AS-Timur)	us-gov-east-1
AWS GovCloud (AS-Barat)	us-gov-west-1

Konsep Penemuan Layanan

Penemuan layanan terdiri dari komponen-komponen berikut:

- Ruang nama penemuan layanan: Grup logis layanan penemuan layanan yang berbagi nama domain yang sama, seperti. `example.com` Ini adalah nama domain tempat Anda ingin mengarahkan lalu lintas ke. Anda dapat membuat namespace dengan panggilan ke `aws servicediscovery create-private-dns-namespace` perintah atau di konsol Amazon ECS. Anda dapat menggunakan `aws servicediscovery list-namespaces` perintah untuk melihat informasi ringkasan tentang ruang nama yang dibuat oleh akun saat ini. Untuk informasi selengkapnya tentang perintah penemuan layanan, lihat `create-private-dns-namespace` dan `list-namespaces` di Panduan AWS CLI Referensi AWS Cloud Map (penemuan layanan).
- Layanan penemuan layanan: Ada dalam namespace penemuan layanan dan terdiri dari nama layanan dan konfigurasi DNS untuk namespace. Layanan tersebut menyediakan komponen inti berikut:
  - Registri layanan: Memungkinkan Anda mencari layanan melalui tindakan DNS atau AWS Cloud Map API dan mendapatkan kembali satu atau lebih titik akhir yang tersedia yang dapat digunakan untuk terhubung ke layanan.
- Contoh penemuan layanan: Ada dalam layanan penemuan layanan dan terdiri dari atribut yang terkait dengan setiap layanan Amazon ECS di direktori layanan.

- Atribut instans: Metadata berikut ditambahkan sebagai atribut khusus untuk setiap layanan Amazon ECS yang dikonfigurasi untuk menggunakan penemuan layanan:
  - **`AWS_INSTANCE_IPV4`**— Sebagai `A` catatan, alamat IPv4 yang dikembalikan Route 53 sebagai respons terhadap kueri DNS dan AWS Cloud Map dikembalikan saat menemukan detail instance, misalnya,. `192.0.2.44`
  - **`AWS_INSTANCE_PORT`**— Nilai port yang terkait dengan layanan penemuan layanan.
  - **`AVAILABILITY_ZONE`**— Zona Ketersediaan tempat tugas diluncurkan. Untuk tugas yang menggunakan tipe peluncuran EC2, ini adalah Availability Zone tempat instance container ada. Untuk tugas yang menggunakan tipe peluncuran Fargate, ini adalah Availability Zone di mana elastic network interface ada.
  - **`REGION`**— Wilayah di mana tugas itu ada.
  - **`ECS_SERVICE_NAME`**— Nama layanan Amazon ECS tempat tugas tersebut berada.
  - **`ECS_CLUSTER_NAME`**— Nama cluster Amazon ECS tempat tugas tersebut berada.
  - **`EC2_INSTANCE_ID`**— ID dari instance kontainer tempat tugas ditempatkan. Atribut kustom ini tidak ditambahkan jika tugas menggunakan jenis peluncuran Fargate.
  - **`ECS_TASK_DEFINITION_FAMILY`**— Keluarga definisi tugas yang digunakan tugas.
  - **`ECS_TASK_SET_EXTERNAL_ID`**— Jika set tugas dibuat untuk penyebaran eksternal dan dikaitkan dengan registri penemuan layanan, maka `ECS_TASK_SET_EXTERNAL_ID` atribut akan berisi ID eksternal dari set tugas.
- Pemeriksaan kesehatan Amazon ECS: Amazon ECS melakukan pemeriksaan kesehatan tingkat kontainer berkala. Jika titik akhir tidak lulus pemeriksaan kondisi, maka titik akhir akan dihapus dari perutean DNS dan ditandai dengan kondisi tidak baik.

Pertimbangan penemuan layanan

Berikut ini harus dipertimbangkan saat menggunakan penemuan layanan:

- Penemuan layanan didukung untuk tugas di Fargate yang menggunakan platform versi 1.1.0 atau yang lebih baru. Untuk informasi selengkapnya, lihat Versi platform Fargate Linux.
- Layanan yang dikonfigurasi untuk menggunakan penemuan layanan memiliki batas 1.000 tugas per layanan. Hal ini dikarenakan kuota layanan Route 53.
- Alur kerja Create Service di konsol Amazon ECS hanya mendukung pendaftaran layanan ke ruang nama DNS pribadi. Ketika namespace DNS AWS Cloud Map pribadi dibuat, zona host pribadi Route 53 akan dibuat secara otomatis.

- Atribut DNS VPC harus dikonfigurasi agar resolusi DNS berhasil. Untuk informasi tentang cara mengonfigurasi atribut, lihat Dukungan DNS di VPC Anda di Panduan Pengguna Amazon VPC.
- Catatan DNS yang dibuat untuk layanan penemuan layanan selalu mendaftar dengan alamat IP pribadi untuk tugas tersebut, bukan alamat IP publik, bahkan ketika ruang nama publik digunakan.
- Penemuan layanan mengharuskan tugas menentukan mode `awsvpc`bridge,, atau `host` jaringan (`none`tidak didukung).
- Jika definisi tugas layanan menggunakan mode `awsvpc` jaringan, Anda dapat membuat kombinasi `A` atau catatan SRV untuk setiap tugas layanan. Jika Anda menggunakan catatan SRV, maka port diperlukan.
- Jika definisi tugas layanan menggunakan mode `host` jaringan `bridge` atau, catatan SRV adalah satu-satunya jenis catatan DNS yang didukung. Buat catatan SRV untuk setiap tugas layanan. Catatan SRV harus menentukan kombinasi nama kontainer dan port kontainer dari penentuan tugas.
- Catatan DNS untuk layanan penemuan layanan dapat ditanyakan dalam VPC Anda. Mereka menggunakan format berikut: `<service discovery service name>.<service discovery namespace>`.
- Saat melakukan kueri DNS pada nama layanan, `A` catatan mengembalikan satu set alamat IP yang sesuai dengan tugas Anda. Catatan SRV mengembalikan satu set alamat IP dan port untuk setiap tugas.
- Jika Anda memiliki delapan atau kurang catatan sehat, Route 53 merespons semua kueri DNS dengan semua catatan sehat.
- Saat semua catatan tidak sehat, Route 53 akan merespons kueri DNS dengan hingga delapan catatan yang tidak sehat.
- Anda dapat mengonfigurasi penemuan layanan untuk layanan yang berada di belakang penyeimbang beban, tetapi lalu lintas penemuan layanan selalu diarahkan ke tugas dan bukan penyeimbang beban.
- Penemuan layanan tidak mendukung penggunaan Classic Load Balancer.
- Kami menyarankan Anda menggunakan pemeriksaan kesehatan tingkat kontainer yang dikelola oleh Amazon ECS untuk layanan penemuan layanan Anda.
  - HealthCheckCustomConfigAmazon ECS mengelola pemeriksaan kesehatan atas nama Anda. Amazon ECS menggunakan informasi dari wadah dan pemeriksaan kesehatan, dan status tugas Anda, untuk memperbarui kesehatan. AWS Cloud Map Ini ditentukan menggunakan `--health-check-custom-config` parameter saat membuat layanan penemuan layanan Anda. Untuk

informasi selengkapnya, lihat HealthCheckCustomConfig di dalam Referensi API AWS Cloud Map .

- Sumber AWS Cloud Map daya yang dibuat saat penemuan layanan digunakan harus dibersihkan secara manual.
- Tugas dan instance didaftarkan `UNHEALTHY` sampai pemeriksaan kesehatan kontainer mengembalikan nilai. Jika pemeriksaan kesehatan lulus, status diperbarui ke`HEALTHY`. Jika pemeriksaan kesehatan kontainer gagal, instance penemuan layanan dideregistrasi.

Harga penemuan layanan

Pelanggan yang menggunakan penemuan layanan Amazon ECS dikenakan biaya untuk sumber daya Route 53 dan operasi API AWS Cloud Map penemuan. Ini melibatkan biaya untuk membuat zona yang dihosting Route 53 dan kueri ke registri layanan. Untuk informasi selengkapnya, lihat AWS Cloud Map Harga di Panduan AWS Cloud Map Pengembang.

Amazon ECS melakukan pemeriksaan kesehatan tingkat kontainer dan memaparkannya ke operasi API pemeriksaan kesehatan AWS Cloud Map khusus. Layanan tersebut saat ini tersedia untuk pelanggan tanpa biaya tambahan. Jika Anda mengonfigurasi pemeriksaan kesehatan jaringan tambahan untuk tugas yang terpapar publik, Anda dikenakan biaya untuk pemeriksaan kesehatan tersebut.

Lindungi tugas Amazon ECS Anda agar tidak dihentikan oleh peristiwa penskalaan

Anda dapat menggunakan perlindungan penskalaan tugas Amazon ECS untuk melindungi tugas agar tidak dihentikan oleh peristiwa penskalaan baik dari Auto Scaling atau penerapan Layanan Otomatis.

Aplikasi tertentu memerlukan mekanisme untuk melindungi tugas-tugas penting misi dari penghentian oleh peristiwa penskalaan selama masa pemanfaatan rendah atau selama penerapan layanan. Sebagai contoh:

- Anda memiliki aplikasi asinkron pemrosesan antrian seperti pekerjaan transcoding video di mana beberapa tugas perlu dijalankan selama berjam-jam bahkan ketika pemanfaatan layanan kumulatif rendah.
- Anda memiliki aplikasi game yang menjalankan server game sebagai tugas Amazon ECS yang perlu terus berjalan bahkan jika semua pengguna telah logged-out untuk mengurangi latensi start-up dari reboot server.

- Saat Anda menerapkan versi kode baru, Anda memerlukan tugas untuk terus berjalan karena akan mahal untuk diproses ulang.

Untuk melindungi tugas yang termasuk dalam layanan Anda agar tidak dihentikan dalam peristiwa scale-in, setel atribut ke`protectionEnabled`. `true` Secara default, tugas dilindungi selama 2 jam. Anda dapat menyesuaikan periode perlindungan dengan menggunakan `expiresInMinutes` atribut. Anda dapat melindungi tugas Anda selama minimal 1 menit dan hingga maksimum 2880 menit (48 jam).

Setelah tugas menyelesaikan pekerjaan yang diperlukan, Anda dapat mengatur `protectionEnabled` atribut ke`false`, memungkinkan tugas dihentikan oleh peristiwa skala berikutnya.

Mekanisme perlindungan skala tugas

Anda dapat mengatur dan mendapatkan perlindungan skala tugas menggunakan titik akhir agen penampung Amazon ECS atau Amazon ECS API.

- Titik akhir agen kontainer Amazon ECS

  Sebaiknya gunakan titik akhir agen penampung Amazon ECS untuk tugas yang dapat menentukan sendiri kebutuhan untuk dilindungi. Gunakan pendekatan ini untuk beban kerja berbasis antrian atau pemrosesan pekerjaan.

  Saat wadah mulai memproses pekerjaan, misalnya dengan menggunakan pesan SQS, Anda dapat mengatur `ProtectionEnabled` atribut melalui jalur titik akhir perlindungan skala tugas `$ECS_AGENT_URI/task-protection/v1/state` dari dalam wadah. Amazon ECS tidak akan menghentikan tugas ini selama acara scale-in. Setelah tugas Anda selesai, Anda dapat menghapus `ProtectionEnabled` atribut menggunakan titik akhir yang sama, membuat tugas memenuhi syarat untuk penghentian selama peristiwa penskalaan berikutnya.

  Untuk informasi selengkapnya tentang titik akhir agen penampung Amazon ECS, lihat. Titik akhir perlindungan skala tugas Amazon ECS
- Amazon ECS API

  Anda dapat menggunakan Amazon ECS API untuk menyetel dan mengambil perlindungan skala tugas jika aplikasi Anda memiliki komponen yang melacak status tugas aktif. Gunakan `UpdateTaskProtection` untuk menandai satu atau lebih tugas sebagai dilindungi. Gunakan `GetTaskProtection` untuk mengambil status perlindungan.

Contoh dari pendekatan ini adalah jika aplikasi Anda menghosting sesi server game sebagai tugas Amazon ECS. Saat pengguna masuk ke sesi di server (tugas), Anda dapat menandai tugas sebagai dilindungi. Setelah pengguna log out, Anda dapat menghapus perlindungan khusus untuk tugas ini atau secara berkala menghapus perlindungan untuk tugas serupa yang tidak lagi memiliki sesi aktif, tergantung pada kebutuhan Anda untuk menjaga server menganggur.

Untuk informasi selengkapnya, lihat UpdateTaskProtectiondan GetTaskProtectiondi Referensi API Amazon Elastic Container Service.

Anda dapat menggabungkan kedua pendekatan. Misalnya, gunakan titik akhir agen Amazon ECS untuk mengatur perlindungan tugas dari dalam wadah dan gunakan Amazon ECS API untuk menghapus perlindungan tugas dari layanan pengontrol eksternal Anda.

Pertimbangan

Pertimbangkan poin-poin berikut sebelum menggunakan perlindungan skala tugas:

- Sebaiknya gunakan titik akhir agen penampung Amazon ECS karena agen Amazon ECS memiliki mekanisme coba ulang bawaan dan antarmuka yang lebih sederhana.
- Anda dapat mengatur ulang periode kedaluwarsa perlindungan skala tugas dengan memanggil tugas `UpdateTaskProtection` yang telah mengaktifkan perlindungan.
- Tentukan berapa lama tugas yang dibutuhkan untuk menyelesaikan pekerjaan yang diperlukan dan mengatur `expiresInMinutes` properti yang sesuai. Jika Anda mengatur kedaluwarsa perlindungan lebih lama dari yang diperlukan, maka Anda akan dikenakan biaya dan menghadapi keterlambatan dalam penyebaran tugas baru.
- Perlindungan skala tugas didukung pada agen `1.65.0` kontainer Amazon ECS atau yang lebih baru.

  Anda dapat menambahkan dukungan untuk fitur ini di instans Amazon EC2 menggunakan versi lama agen penampung Amazon ECS dengan memperbarui agen ke versi terbaru. Untuk informasi selengkapnya, lihat Memperbarui agen kontainer Amazon ECS.
- Pertimbangan penyebaran:
  - Jika layanan menggunakan pembaruan bergulir, tugas baru akan dibuat tetapi tugas yang menjalankan versi lama tidak akan dihentikan sampai dihapus atau `protectionEnabled` kedaluwarsa. Anda dapat menyesuaikan `maximumPercentage` parameter dalam konfigurasi penerapan ke nilai yang memungkinkan tugas baru dibuat saat tugas lama dilindungi.

- Jika pembaruan biru/hijau diterapkan, penerapan biru dengan tugas yang dilindungi tidak akan dihapus jika tugas memiliki. `protectionEnabled` Lalu lintas akan dialihkan ke tugas baru yang muncul dan tugas lama hanya akan dihapus ketika dihapus atau `protectionEnabled` kedaluwarsa. Bergantung pada batas waktu CodeDeploy atau CloudFormation pembaruan, penerapan mungkin batas waktu dan tugas Biru yang lebih lama mungkin masih ada.
- Jika Anda menggunakan CloudFormation, update-stack memiliki batas waktu 3 jam. Oleh karena itu, jika Anda mengatur perlindungan tugas Anda lebih dari 3 jam, maka CloudFormation penerapan Anda dapat mengakibatkan kegagalan dan kemunduran.

  Selama tugas lama Anda dilindungi, CloudFormation tumpukan ditampilkan`UPDATE_IN_PROGRESS`. Jika perlindungan skala tugas dihapus atau kedaluwarsa dalam jendela 3 jam, penerapan Anda akan berhasil dan pindah ke status. `UPDATE_COMPLETE` Jika penerapan macet `UPDATE_IN_PROGRESS` selama lebih dari 3 jam, itu akan gagal dan menunjukkan `UPDATE_FAILED` status, dan kemudian akan digulung kembali ke set tugas lama.
- Amazon ECS mengirimkan peristiwa layanan saat tugas yang dilindungi menjaga penerapan (bergulir atau biru/hijau) agar tidak mencapai kondisi tunak, sehingga Anda dapat mengambil tindakan perbaikan. Saat mencoba memperbarui status perlindungan suatu tugas, jika Anda menerima pesan `DEPLOYMENT_BLOCKED` kesalahan, itu berarti layanan memiliki tugas yang lebih terlindungi daripada jumlah tugas yang diinginkan untuk layanan. Untuk mengatasi kesalahan ini, lakukan salah satu hal berikut:
  - Tunggu hingga perlindungan tugas saat ini kedaluwarsa. Kemudian atur perlindungan tugas.
  - Tentukan tugas mana yang bisa dihentikan. Kemudian gunakan `UpdateTaskProtection` dengan `protectionEnabled` opsi yang disetel `false` untuk tugas-tugas ini.
  - Tingkatkan jumlah tugas yang diinginkan dari layanan menjadi lebih dari jumlah tugas yang dilindungi.

Izin IAM diperlukan untuk perlindungan skala tugas

Permintaan harus memiliki peran tugas Amazon ECS dengan izin berikut:

- `ecs:GetTaskProtection`: Memungkinkan agen kontainer Amazon ECS untuk meneleponGetTaskProtection.
- `ecs:UpdateTaskProtection`: Memungkinkan agen kontainer Amazon ECS untuk meneleponUpdateTaskProtection.

Titik akhir perlindungan skala tugas Amazon ECS

Agen penampung Amazon ECS secara otomatis menyuntikkan variabel `ECS_AGENT_URI` lingkungan ke dalam wadah tugas Amazon ECS untuk menyediakan metode untuk berinteraksi dengan titik akhir API agen penampung.

Sebaiknya gunakan titik akhir agen penampung Amazon ECS untuk tugas yang dapat menentukan sendiri kebutuhan untuk dilindungi.

Saat kontainer mulai memproses pekerjaan, Anda dapat menyetel `protectionEnabled` atribut menggunakan jalur titik akhir perlindungan skala tugas `$ECS_AGENT_URI/task-protection/v1/state` dari dalam wadah.

Gunakan permintaan PUT ke URI ini dari dalam wadah untuk mengatur perlindungan skala tugas. Permintaan GET ke URI ini mengembalikan status perlindungan tugas saat ini.

Parameter permintaan perlindungan skala tugas

Anda dapat mengatur perlindungan skala tugas menggunakan `${ECS_AGENT_URI}/task-protection/v1/state` titik akhir dengan parameter permintaan berikut.

`ProtectionEnabled`

Tentukan `true` untuk menandai tugas untuk perlindungan. Tentukan `false` untuk menghapus perlindungan dan membuat tugas memenuhi syarat untuk penghentian.

Jenis: Boolean

Diperlukan: Ya

`ExpiresInMinutes`

Jumlah menit tugas dilindungi. Anda dapat menentukan minimal 1 menit hingga 2.880 menit (48 jam). Selama periode waktu ini, tugas Anda tidak akan dihentikan oleh peristiwa penskalaan dari Auto Scaling atau penerapan layanan. Setelah periode waktu ini berlalu, `protectionEnabled` parameter diatur ke`false`.

Jika Anda tidak menentukan waktu, maka tugas secara otomatis dilindungi selama 120 menit (2 jam).

Tipe: Integer

Wajib: Tidak

Contoh berikut menunjukkan cara mengatur perlindungan tugas dengan durasi yang berbeda.

Contoh cara melindungi tugas dengan periode waktu default

Contoh ini menunjukkan cara melindungi tugas dengan periode waktu default 2 jam.

```
curl --request PUT --header 'Content-Type: application/json' ${ECS_AGENT_URI}/task-
protection/v1/state --data '{"ProtectionEnabled":true}'
```

Contoh cara melindungi tugas selama 60 menit

Contoh ini menunjukkan cara melindungi tugas selama 60 menit menggunakan `expiresInMinutes` parameter.

```
curl --request PUT --header 'Content-Type: application/json' ${ECS_AGENT_URI}/task-
protection/v1/state --data '{"ProtectionEnabled":true,"ExpiresInMinutes":60}'
```

Contoh cara melindungi tugas selama 24 jam

Contoh ini menunjukkan cara melindungi tugas selama 24 jam menggunakan `expiresInMinutes` parameter.

```
curl --request PUT --header 'Content-Type: application/json' ${ECS_AGENT_URI}/task-
protection/v1/state --data '{"ProtectionEnabled":true,"ExpiresInMinutes":1440}'
```

Permintaan PUT mengembalikan respons berikut.

```
{
  "protection": {
    "ExpirationDate": "2023-12-20T21:57:44.837Z",
    "ProtectionEnabled": true,
    "TaskArn": "arn:aws:ecs:us-west-2:111122223333:task/1234567890abcdef0"
  }
}
```

Parameter respons perlindungan skala tugas

Informasi berikut dikembalikan dari titik akhir perlindungan skala tugas dalam respons `${ECS_AGENT_URI}/task-protection/v1/state` JSON.

`ExpirationDate`

Waktu zaman ketika perlindungan untuk tugas akan kedaluwarsa. Jika tugas tidak dilindungi, nilai ini adalah nol.

`ProtectionEnabled`

Status perlindungan tugas. Jika perlindungan scale-in diaktifkan untuk tugas, nilainya adalah. `true` Kalau tidak, itu`false`.

`TaskArn`

Nama lengkap Amazon Resource Name (ARN) dari tugas milik kontainer.

Contoh berikut menunjukkan detail yang dikembalikan untuk tugas yang dilindungi.

```
curl --request GET ${ECS_AGENT_URI}/task-protection/v1/state
```

```
{
    "protection":{
        "ExpirationDate":"2023-12-20T21:57:44Z",
        "ProtectionEnabled":true,
        "TaskArn":"arn:aws:ecs:us-west-2:111122223333:task/1234567890abcdef0"
    }
}
```

Informasi berikut dikembalikan ketika terjadi kegagalan.

`Arn`

Nama Sumber Daya Amazon (ARN) lengkap dari tugas tersebut.

`Detail`

Detail terkait dengan kegagalan.

`Reason`

Sebab kegagalan.

Contoh berikut menunjukkan rincian yang dikembalikan untuk tugas yang tidak dilindungi.

```
{
```

```
    "failure":{
        "Arn":"arn:aws:ecs:us-west-2:111122223333:task/1234567890abcdef0",
        "Detail":null,
        "Reason":"TASK_NOT_VALID"
    }
}
```

Informasi berikut dikembalikan ketika pengecualian terjadi.

`requestID`

ID AWS permintaan untuk panggilan Amazon ECS API yang menghasilkan pengecualian.

`Arn`

Nama Sumber Daya Amazon (ARN) lengkap dari tugas atau layanan.

`Code`

Kode kesalahan.

`Message`

Pesan kesalahan.

> **Note**
>
> Jika `RequestTimeout` kesalahan `RequestError` atau muncul, kemungkinan itu adalah masalah jaringan. Coba gunakan titik akhir VPC untuk Amazon ECS.

Contoh berikut menunjukkan rincian yang dikembalikan ketika terjadi kesalahan.

```
{
    "requestID":"12345-abc-6789-0123-abc",
    "error":{
        "Arn":"arn:aws:ecs:us-west-2:555555555555:task/my-cluster-
name/1234567890abcdef0",
        "Code":"AccessDeniedException",
        "Message":"User: arn:aws:sts::444455556666:assumed-role/my-ecs-task-
role/1234567890abcdef0 is not authorized to perform: ecs:GetTaskProtection on resource:
 arn:aws:ecs:us-west-2:555555555555:task/test/1234567890abcdef0 because no identity-
based policy allows the ecs:GetTaskProtection action"
    }
```

```
}
```

Kesalahan berikut muncul jika agen Amazon ECS tidak dapat mendapatkan respons dari titik akhir Amazon ECS karena alasan seperti masalah jaringan atau bidang kontrol Amazon ECS sedang down.

```
{
  "error": {
    "Arn": "arn:aws:ecs:us-west-2:555555555555:task/my-cluster-name/1234567890abcdef0",
    "Code": "RequestCanceled",
    "Message": "Timed out calling Amazon ECS Task Protection API"
  }
}
```

Kesalahan berikut muncul ketika agen Amazon ECS mendapatkan pengecualian pelambatan dari Amazon ECS.

```
{
  "requestID": "12345-abc-6789-0123-abc",
  "error": {
    "Arn": "arn:aws:ecs:us-west-2:555555555555:task/my-cluster-name/1234567890abcdef0",
    "Code": "ThrottlingException",
    "Message": "Rate exceeded"
  }
}
```

Logika throttle layanan Amazon ECS

Penjadwal layanan Amazon ECS menyertakan logika yang membatasi seberapa sering tugas layanan diluncurkan jika berulang kali gagal memulai.

Jika tugas untuk layanan berulang kali gagal memasuki `RUNNING` status (maju langsung dari a `PENDING` ke `STOPPED` status), maka waktu antara upaya restart berikutnya secara bertahap ditingkatkan hingga maksimum 27 menit. Periode maksimum ini dapat berubah di masa depan. Perilaku ini mengurangi efek tugas yang gagal terhadap sumber daya kluster Amazon ECS atau biaya infrastruktur Fargate Anda. Jika layanan Anda memulai logika throttle, Anda menerima pesan peristiwa layanan berikut:

```
(service service-name) is unable to consistently start tasks successfully.
```

Amazon ECS tidak pernah menghentikan layanan yang gagal untuk mencoba lagi. Itu juga tidak mencoba untuk memodifikasinya dengan cara apa pun selain meningkatkan waktu antara restart. Logika throttle layanan tidak menyediakan parameter yang dapat disetel pengguna.

Jika Anda memperbarui layanan untuk menggunakan penentuan tugas baru, maka layanan Anda akan segera kembali ke status normal dan bukan di-throttling. Untuk informasi selengkapnya, lihat Memperbarui layanan menggunakan konsol.

Berikut ini adalah beberapa penyebab umum yang memulai logika ini. Kami menyarankan Anda mengambil tindakan manual untuk mengatasi masalah ini:

- Kurangnya sumber daya untuk meng-host tugas Anda, seperti port, memori, atau unit CPU di cluster Anda. Dalam hal ini, Anda juga melihat pesan kejadian layanan sumber daya tidak mencukupi.
- Agen penampung Amazon ECS tidak dapat menarik gambar Docker tugas Anda. Ini mungkin karena nama gambar kontainer yang buruk, gambar, atau tag, atau kurangnya otentikasi registri pribadi atau izin. Dalam hal ini, Anda juga melihat `CannotPullContainerError` di menghentikan kesalahan tugas.
- Ruang disk pada instans kontainer Anda tidak cukup untuk membuat kontainer. Dalam hal ini, Anda juga melihat `CannotCreateContainerError` di menghentikan kesalahan tugas. Untuk informasi selengkapnya, lihat `CannotCreateContainerError: API error (500): devmapper`.

> **Important**
>
> Tugas yang dihentikan setelah mencapai `RUNNING` status tidak memulai logika throttle atau pesan peristiwa layanan terkait. Misalnya, asumsikan bahwa pemeriksaan kesehatan Elastic Load Balancing yang gagal untuk suatu layanan menyebabkan tugas ditandai sebagai tidak sehat, dan Amazon ECS membatalkan pendaftarannya dan menghentikan tugas. Pada titik ini, tugas tidak dibatasi. Bahkan jika perintah kontainer tugas segera keluar dengan kode keluar bukan nol, tugas sudah dipindahkan ke status. `RUNNING` Tugas yang segera gagal karena kesalahan perintah tidak menyebabkan throttle atau pesan peristiwa layanan.

Penandaan sumber daya Amazon ECS

Untuk membantu mengelola sumber daya Amazon ECS, Anda dapat menetapkan metadata Anda sendiri secara opsional ke setiap sumber daya menggunakan tag. Setiap tag terdiri dari kunci dan nilai opsional.

Anda dapat menggunakan tag untuk mengkategorikan sumber daya Amazon ECS Anda dengan berbagai cara, misalnya, berdasarkan tujuan, pemilik, atau lingkungan. Hal ini berguna jika Anda memiliki banyak sumber daya dengan jenis yang sama. Anda dapat dengan cepat mengidentifikasi sumber daya tertentu berdasarkan tag yang Anda tetapkan padanya. Misalnya, Anda dapat menentukan satu set tag untuk instans penampung Amazon ECS akun Anda. Ini membantu Anda melacak setiap pemilik instans dan tingkat tumpukan.

Anda dapat menggunakan tag untuk laporan Biaya dan Penggunaan Anda. Anda dapat menggunakan laporan ini untuk menganalisis biaya dan penggunaan sumber daya Amazon ECS Anda. Untuk informasi selengkapnya, lihat the section called “Laporan Penggunaan”.

> **Warning**
>
> Ada banyak API yang mengembalikan kunci tag dan nilainya. Menolak akses ke `DescribeTags` tidak secara otomatis menolak akses ke tanda yang ditampilkan oleh API lain. Sebagai praktik terbaik, sebaiknya Anda tidak menyertakan data sensitif ke dalam tanda.

Sebaiknya rancang serangkaian kunci tanda yang memenuhi kebutuhan setiap tipe sumber daya. Anda dapat menggunakan satu set kunci tag yang konsisten untuk memudahkan pengelolaan sumber daya Anda. Anda dapat mencari dan memfilter sumber daya berdasarkan tanda yang Anda tambahkan.

Tag tidak memiliki arti semantik untuk Amazon ECS dan ditafsirkan secara ketat sebagai serangkaian karakter. Anda dapat mengedit kunci dan nilai tanda, dan dapat menghapus tanda dari sumber daya kapan saja. Anda dapat mengatur nilai tanda ke string kosong, akan tetapi Anda tidak dapat mengatur nilai tanda ke nol. Jika Anda menambahkan tanda yang memiliki kunci yang sama dengan tanda yang ada pada sumber daya tersebut, nilai baru akan menimpa nilai sebelumnya. Saat Anda menghapus sumber daya, tag apa pun untuk sumber daya juga dihapus.

Jika Anda menggunakan AWS Identity and Access Management (IAM), Anda dapat mengontrol pengguna mana di AWS akun Anda yang memiliki izin untuk mengelola tag.

Bagaimana sumber daya ditandai

Ada beberapa cara untuk menandai tugas, layanan, definisi tugas, dan cluster Amazon ECS:

- Pengguna secara manual menandai sumber daya dengan menggunakan AWS Management Console, Amazon ECS API, the AWS CLI, atau AWS SDK.
- Pengguna membuat layanan atau menjalankan tugas mandiri dan memilih opsi tag yang dikelola Amazon ECS.

  Amazon ECS secara otomatis menandai semua tugas yang baru diluncurkan. Untuk informasi selengkapnya, lihat the section called “Tag yang dikelola Amazon ECS”.
- Pengguna membuat sumber daya menggunakan konsol. Konsol secara otomatis menandai sumber daya.

  Tag ini dikembalikan dalam AWS CLI, dan respons AWS SDK dan ditampilkan di konsol. Anda tidak dapat mengubah atau menghapus tag ini.

  Untuk informasi tentang tag yang ditambahkan, lihat kolom Tag yang ditambahkan secara otomatis oleh konsol di tabel sumber daya Dukungan penandaan untuk Amazon ECS.

Jika Anda menentukan tag saat membuat sumber daya dan tag tidak dapat diterapkan, Amazon ECS akan mengembalikan proses pembuatan. Hal ini untuk memastikan bahwa sumber daya dibuat dengan tanda atau tidak dibuat sama sekali, dan tidak ada sumber daya yang dibiarkan tanpa tanda kapan saja. Dengan menandai sumber daya saat sedang dibuat, Anda dapat menghilangkan kebutuhan untuk menjalankan skrip penandaan khusus setelah pembuatan sumber daya.

Tabel berikut menjelaskan sumber daya Amazon ECS yang mendukung penandaan.

Menandai dukungan untuk sumber daya Amazon ECS

Sumber daya	Mendukung tanda	Penyebaran tanda Support	Tag secara otomatis ditambahkan oleh konsol
Tugas-tugas Amazon ECS	Ya	Ya, dari ketentuan tugas.	Kunci: `aws:ecs:clusterName` Nilai: `cluster-name`

Sumber daya	Mendukung tanda	Penyebaran tanda Support	Tag secara otomatis ditambahkan oleh konsol
Layanan-layanan Amazon ECS	Ya	Ya, baik dari ketentuan tugas ataupun layanan ke dalam tugas dalam layanan.	Kunci: `ecs:service:stackId` Nilai `arn:aws:cloudformation: arn`
Set tugas Amazon ECS	Ya	Tidak	N/A
Definisi tugas Amazon ECS	Ya	Tidak	Kunci: `ecs:taskDefinition:createdFrom` Nilai: `ecs-console-v2`

Sumber daya	Mendukung tanda	Penyebaran tanda Support	Tag secara otomatis ditambahkan oleh konsol
Klaster-klaster Amazon ECS	Ya	Tidak	Kunci: `aws:cloudformation:logical-id` Nilai: `ECSCluster` Kunci: `aws:cloudformation:stack-id` Nilai: `arn:aws:cloudformation: arn` Kunci: `aws:cloudformation:stack-name` Nilai: `ECS-Console-V2-Cluster- EXAMPLE`
Instans kontainer Amazon ECS	Ya	Ya, dari instans Amazon EC2. Untuk informasi selengkapnya, lihat Menambahkan tag ke instans penampung Amazon EC2.	N/A
Instans Eksternal Amazon ECS	Ya	Tidak	N/A

Sumber daya	Mendukung tanda	Penyebaran tanda Support	Tag secara otomatis ditambahkan oleh konsol
Penyedia kapasitas Amazon ECS	Ya. Anda tidak dapat menandai penyedia standar `FARGATE` dan `FARGATE_SPOT` kapasitas.	Tidak	N/A

Menandai sumber daya pada pembuatan

Sumber daya berikut mendukung penandaan pada pembuatan menggunakan Amazon ECS API, AWS CLI, atau AWS SDK:

- Tugas-tugas Amazon ECS
- Layanan-layanan Amazon ECS
- Definisi tugas Amazon ECS
- Set tugas Amazon ECS
- Klaster-klaster Amazon ECS
- Instans kontainer Amazon ECS
- Penyedia kapasitas Amazon ECS

Amazon ECS memiliki opsi untuk menggunakan otorisasi penandaan untuk pembuatan sumber daya. Ketika Akun AWS dikonfigurasi untuk menandai otorisasi, pengguna harus memiliki izin untuk tindakan yang membuat sumber daya, seperti. `ecsCreateCluster` Jika Anda menentukan tag dalam tindakan pembuatan sumber daya, AWS lakukan otorisasi tambahan untuk memverifikasi apakah pengguna atau peran memiliki izin untuk membuat tag. Oleh karena itu, Anda harus memberikan izin eksplisit untuk menggunakan tindakan. `ecs:TagResource` Untuk informasi selengkapnya, lihat the section called “Memberi tanda pada sumber daya saat sumber daya dibuat”. Untuk informasi tentang cara mengkonfigurasi opsi, lihatthe section called “Otorisasi penandaan”.

Pembatasan

Pembatasan berikut berlaku untuk tag:

- Maksimal 50 tag dapat dikaitkan dengan sumber daya.
- Kunci tag tidak dapat diulang untuk satu sumber daya. Setiap kunci tag harus unik, dan hanya dapat memiliki satu nilai.
- Panjang tombol dapat mencapai 128 karakter di UTF-8.
- Nilai dapat mencapai 256 karakter di UTF-8.
- Jika beberapa Layanan AWS dan sumber daya menggunakan skema penandaan Anda, batasi jenis karakter yang Anda gunakan. Beberapa layanan mungkin memiliki batasan pada karakter yang diizinkan. Umumnya, karakter yang diizinkan adalah huruf, angka, spasi, dan karakter berikut: **+ - = . _ : /@**.
- Kunci dan nilai tag peka huruf besar-kecil.
- Anda tidak dapat menggunakan `aws:AWS:`,, atau kombinasi huruf besar atau kecil seperti awalan untuk kunci atau nilai. Ini hanya disediakan untuk AWS digunakan. Anda tidak dapat menyunting atau menghapus kunci atau nilai tag dengan awalan ini. Tag dengan awalan ini tidak dihitung terhadap tags-per-resource batas Anda.

Tag yang dikelola Amazon ECS

Saat Anda menggunakan tag yang dikelola Amazon ECS, Amazon ECS secara otomatis menandai semua tugas yang baru diluncurkan dan volume Amazon EBS apa pun yang dilampirkan ke tugas dengan informasi klaster dan tag definisi tugas yang ditambahkan pengguna atau tag layanan. Berikut ini menjelaskan tag yang ditambahkan:

- Tugas mandiri — tag dengan Key as `aws:ecs:clusterName` dan Value yang disetel ke nama cluster. Semua tag definisi tugas yang ditambahkan oleh pengguna. Volume Amazon EBS yang dilampirkan ke tugas mandiri akan menerima tag dengan Kunci sebagai `aws:ecs:clusterName` dan Nilai yang disetel ke nama cluster. Untuk informasi selengkapnya tentang penandaan volume Amazon EBS, lihat Menandai volume Amazon EBS.
- Tugas yang merupakan bagian dari layanan — tag dengan Kunci sebagai `aws:ecs:clusterName` dan Nilai yang disetel ke nama cluster. Tag dengan Key as `aws:ecs:serviceName` dan Value diatur ke nama layanan. Tag dari salah satu sumber berikut:
  - Definisi tugas - Semua tag definisi tugas yang ditambahkan oleh pengguna.

- Layanan — Semua tag layanan yang ditambahkan oleh pengguna.

  Volume Amazon EBS yang dilampirkan ke tugas yang merupakan bagian dari layanan akan menerima tag dengan Kunci sebagai `aws:ecs:clusterName` dan Nilai yang disetel ke nama cluster, dan tag dengan Kunci sebagai `aws:ecs:serviceName` dan Nilai yang disetel ke nama layanan. Untuk informasi selengkapnya tentang penandaan volume Amazon EBS, lihat Menandai volume Amazon EBS.

Opsi berikut diperlukan untuk fitur ini:

- Anda harus ikut serta dalam format Amazon Resource Name (ARN) dan resource identifier (ID) yang baru. Untuk informasi selengkapnya, lihat Amazon Resource Name (ARN) dan ID.
- Saat Anda menggunakan API untuk membuat layanan atau menjalankan tugas, Anda harus menyetel `enableECSManagedTags` ke `true` for `run-task` and`create-service`. Untuk informasi selengkapnya, lihat create-service dan run-task di Referensi API.AWS Command Line Interface
- Amazon ECS menggunakan tag terkelola untuk menentukan kapan beberapa fitur diaktifkan, misalnya Auto Scaling cluster. Kami menyarankan Anda untuk tidak memodifikasi tag secara manual sehingga Amazon ECS dapat mengelola fitur secara efektif.

Menandai sumber daya Anda untuk penagihan

AWS menyediakan alat pelaporan yang disebut Cost Explorer yang dapat Anda gunakan untuk menganalisis biaya dan penggunaan sumber daya Amazon ECS Anda.

Anda dapat menggunakan Cost Explorer untuk melihat bagan penggunaan dan biaya Anda. Anda dapat melihat data dari 13 bulan terakhir, dan memperkirakan berapa banyak kemungkinan Anda akan menghabiskan untuk tiga bulan ke depan. Anda dapat menggunakan Cost Explorer untuk melihat pola terkait berapa banyak sumber daya AWS yang Anda belanjakan dari waktu ke waktu. Misalnya, Anda dapat menggunakannya untuk mengidentifikasi area yang memerlukan penyelidikan lebih lanjut dan melihat tren yang dapat Anda gunakan untuk memahami biaya Anda. Anda juga dapat menentukan rentang waktu untuk data, dan melihat data waktu menurut hari atau bulan.

Anda dapat menggunakan tag yang dikelola Amazon ECS atau tag yang ditambahkan pengguna untuk Laporan Biaya dan Penggunaan Anda. Untuk informasi selengkapnya, lihat Laporan penggunaan Amazon ECS.

Untuk melihat biaya sumber daya gabungan, Anda dapat mengatur informasi penagihan berdasarkan sumber daya yang memiliki nilai kunci tanda yang sama. Misalnya, Anda dapat memberi tanda pada beberapa sumber daya dengan nama aplikasi tertentu, lalu organisir informasi penagihan Anda untuk melihat total biaya aplikasi tersebut pada beberapa layanan. Untuk informasi selengkapnya tentang pengaturan laporan alokasi biaya dengan tanda, lihat Laporan Alokasi Biaya Bulanan dalam Panduan Pengguna AWS Billing .

Selain itu, Anda dapat mengaktifkan Data Alokasi Biaya Split untuk mendapatkan CPU tingkat tugas dan data penggunaan memori di Laporan Biaya dan Penggunaan Anda. Untuk informasi selengkapnya, lihat Laporan Biaya dan Penggunaan Tingkat Tugas.

> **Note**
>
> Jika Anda mengaktifkan pelaporan, diperlukan waktu hingga 24 jam sebelum data untuk bulan berjalan tersedia untuk dilihat.

Bekerja dengan tanda menggunakan konsol

Anda dapat menggunakan konsol Amazon ECS, Anda dapat mengelola tag yang terkait dengan tugas, layanan, definisi tugas, cluster, atau instans penampung baru atau yang sudah ada.

Saat Anda memilih halaman khusus sumber daya di konsol Amazon ECS, halaman tersebut akan menampilkan daftar sumber daya tersebut. Misalnya, jika Anda memilih Cluster dari panel navigasi, konsol akan menampilkan daftar cluster Amazon ECS. Saat Anda memilih sumber daya dari salah satu daftar (misalnya, klaster tertentu) yang mendukung tanda ini, Anda dapat melihat dan mengelola tandanya di tab Tanda.

> **Warning**
>
> Sebagai praktik terbaik, sebaiknya Anda tidak menyertakan data sensitif ke dalam tanda.

Daftar Isi

Penambahan tanda pada sumber daya individu selama peluncuran

Anda dapat menggunakan sumber daya berikut untuk menentukan tag saat Anda membuat sumber daya.

Tugas	Konsol
Jalankan satu tugas atau lebih.	Jalankan aplikasi sebagai tugas Amazon ECS
Buat sebuah layanan.	Membuat layanan menggunakan konsol
Buat satu set tugas.	Menggunakan pengontrol pihak ketiga untuk penerapan eksternal Amazon ECS
Daftarkan ketentuan tugas.	the section called “Membuat definisi tugas menggunakan konsol”
Buat sebuah klaster.	Membuat cluster untuk jenis peluncuran Fargate dan Eksternal menggunakan konsol
Jalankan satu atau beberapa instans kontainer.	Meluncurkan instans penampung Amazon ECS Linux

Mengelola tag sumber daya individu

Anda dapat menambahkan atau menghapus tag yang terkait dengan cluster, layanan, tugas, dan definisi tugas langsung dari halaman sumber daya. Untuk informasi tentang menandai instance container Anda, lihat. Menambahkan tag ke instans penampung Amazon EC2

> **Warning**
>
> Jangan menambahkan informasi pengenal pribadi (PII) atau informasi rahasia atau sensitif lainnya dalam tag. Tag dapat diakses oleh banyak layanan AWS , termasuk penagihan. Tag tidak dimaksudkan untuk digunakan dalam data sensitif atau privat.

Untuk memodifikasi tag untuk sumber daya individu

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih yang Wilayah AWS akan digunakan.
3. Di panel navigasi, pilih tipe sumber daya (misalnya, Klaster).
4. Pilih sumber daya dari daftar sumber daya, pilih tab Tag, lalu pilih Kelola tag.
5. Konfigurasikan tag Anda.

   [Tambahkan tag] Pilih Tambah tag, lalu lakukan hal berikut:

   - Untuk Kunci, masukkan nama kunci.
   - Untuk Nilai, masukkan nilai kunci.

   [Hapus tanda] Pilih Hapus di sebelah kanan Kunci dan Nilai tanda.
6. Pilih Simpan.

Menambahkan tag ke instans penampung Amazon EC2

Anda dapat mengaitkan tanda dengan instans kontainer menggunakan salah satu metode berikut:

- Metode 1 — Saat membuat instance container menggunakan Amazon EC2 API, CLI, atau konsol, tentukan tag dengan meneruskan data pengguna ke instance menggunakan parameter konfigurasi agen penampung. `ECS_CONTAINER_INSTANCE_TAGS` Ini membuat tag yang terkait dengan

instance container di Amazon ECS saja, tag tersebut tidak dapat dicantumkan menggunakan Amazon EC2 API. Untuk informasi selengkapnya, lihat Bootstrapping instance container dengan data pengguna Amazon EC2.

> **Important**
>
> Jika Anda meluncurkan instance container menggunakan grup Auto Scaling Amazon EC2, Anda harus menggunakan parameter konfigurasi agen ECS_CONTAINER_INSTANCE_TAGS untuk menambahkan tag. Hal ini disebabkan oleh cara tag ditambahkan ke instans Amazon EC2 yang diluncurkan menggunakan grup Auto Scaling.

Berikut ini adalah contoh skrip data pengguna yang mengaitkan tag dengan instance container Anda:

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_CONTAINER_INSTANCE_TAGS={"tag_key": "tag_value"}
EOF
```

- Metode 2 — Saat Anda membuat instance container menggunakan Amazon EC2 API, CLI, atau konsol, tentukan tag terlebih dahulu menggunakan parameter tersebut. `TagSpecification.N` Kemudian, berikan data pengguna ke instance dengan menggunakan parameter konfigurasi agen kontainer`ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM`. Melakukannya menyebarkannya dari Amazon EC2 ke Amazon ECS.

  Berikut ini adalah contoh skrip data pengguna yang menyebarkan tag yang terkait dengan instans Amazon EC2, dan mendaftarkan instance dengan cluster yang diberi nama. `MyCluster`

  ```
  #!/bin/bash
  cat <<'EOF' >> /etc/ecs/ecs.config
  ECS_CLUSTER=MyCluster
  ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM=ec2_instance
  EOF
  ```

  Untuk menyediakan akses agar tag instans penampung dapat disebarkan dari Amazon EC2 ke Amazon ECS, tambahkan izin berikut secara manual sebagai kebijakan sebaris ke peran

IAM instans penampung Amazon ECS. Untuk informasi selengkapnya, lihat Menambahkan dan Menghapus Kebijakan IAM.

- `ec2:DescribeTags`

Berikut ini adalah contoh kebijakan yang digunakan untuk menambahkan izin ini.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
             "ec2:DescribeTags"
            ],
            "Resource": "*"
        }
    ]
}
```

Menandai instance kontainer eksternal

Anda dapat mengaitkan tag dengan instance kontainer eksternal Anda dengan menggunakan salah satu metode berikut.

- Metode 1 — Sebelum menjalankan skrip instalasi untuk mendaftarkan instans eksternal Anda dengan cluster Anda, buat atau edit file konfigurasi agen penampung Amazon ECS di `/etc/ecs/ecs.config` dan tambahkan parameter konfigurasi agen `ECS_CONTAINER_INSTANCE_TAGS` penampung. Metode ini membuat tanda yang terkait dengan instans eksternal.

  Berikut ini adalah contoh sintaks.

  ```
  ECS_CONTAINER_INSTANCE_TAGS={"tag_key": "tag_value"}
  ```

- Metode 2 - Setelah instance eksternal Anda terdaftar ke cluster Anda, Anda dapat menggunakan AWS Management Console untuk menambahkan tag. Untuk informasi selengkapnya, lihat Mengelola tag sumber daya individu.

Bekerja dengan tanda menggunakan CLI atau API

Anda dapat menggunakan AWS Command Line Interface atau API untuk menandai sumber daya Amazon ECS Anda.

Gunakan hal-hal berikut untuk menambahkan, memperbarui, mencantumkan, dan menghapus tanda untuk sumber daya Anda. Dokumentasi yang sesuai menyediakan contoh.

> **Warning**
>
> Jangan menambahkan informasi identitas pribadi (PII) atau informasi rahasia atau sensitif lainnya dalam tag. Tag dapat diakses oleh banyak orang Layanan AWS, termasuk penagihan. Tag tidak dimaksudkan untuk digunakan untuk data pribadi atau sensitif.

Menandai dukungan untuk sumber daya Amazon ECS

Tugas	AWS CLI	Tindakan API
Tambahkan atau timpa satu atau beberapa tanda.	tag-sumber daya	TagResource
Hapus satu atau beberapa tanda.	untag-sumber daya	UntagResource

Anda dapat menggunakan beberapa tindakan pembuatan sumber daya untuk menentukan tag saat Anda membuat sumber daya. Tindakan berikut mendukung penandaan saat pembuatan.

Anda harus memiliki `ecsTagResource` izin. Untuk informasi selengkapnya, lihat Berikan izin untuk menandai sumber daya pada pembuatan.

Tugas	AWS CLI	AWS Tools for Windows PowerShell	Tindakan API
	jalankan-tugas	Mulai-ecstask	RunTask

Tugas	AWS CLI	AWS Tools for Windows PowerShell	Tindakan API
Jalankan satu atau beberapa tugas.			
Buat sebuah layanan.	membuat-layanan	Baru-ECSservice	CreateService
Buat satu set tugas.	create-task-set	Baru-ECS TaskSet	CreateTaskSet
Mendaftarkan ketentuan tugas.	register-task-definition	Daftarkan-ECS TaskDefinition	RegisterTaskDefinition
Buat sebuah klaster.	buat-cluster	Escluster baru	CreateCluster
Jalankan satu atau beberapa instans kontainer.	run-instance	New-EC2Instance	RunInstances

Laporan penggunaan Amazon ECS

AWS menyediakan alat pelaporan yang disebut Cost Explorer yang dapat Anda gunakan untuk menganalisis biaya dan penggunaan sumber daya Amazon ECS Anda.

Anda dapat menggunakan Cost Explorer untuk melihat bagan penggunaan dan biaya Anda. Anda dapat melihat data dari 13 bulan terakhir, dan memperkirakan berapa banyak kemungkinan Anda akan menghabiskan untuk tiga bulan ke depan. Anda dapat menggunakan Cost Explorer untuk melihat pola terkait berapa banyak sumber daya AWS yang Anda belanjakan dari waktu ke waktu. Misalnya, Anda dapat menggunakannya untuk mengidentifikasi area yang memerlukan penyelidikan lebih lanjut dan melihat tren yang dapat Anda gunakan untuk memahami biaya Anda. Anda juga dapat menentukan rentang waktu untuk data, dan melihat data waktu menurut hari atau bulan.

Data pengukuran dalam Laporan Biaya dan Penggunaan menunjukkan penggunaan di semua tugas Amazon ECS Anda. Data pengukuran meliputi penggunaan CPU sebagai `vCPU-Hours`

dan penggunaan memori sebagai `GB-Hours` untuk setiap tugas yang dijalankan. Bagaimana data tersebut disajikan tergantung pada tipe peluncuran tugas.

Untuk tugas yang menggunakan tipe peluncuran Fargate, `lineItem/Operation` kolom menunjukkan `FargateTask` dan Anda akan melihat biaya yang terkait dengan setiap tugas.

Untuk tugas yang menggunakan tipe peluncuran EC2, `lineItem/Operation` kolom ditampilkan `ECSTask-EC2` dan tugas tidak memiliki biaya langsung yang terkait dengannya. Data pengukuran yang ditampilkan dalam laporan, seperti penggunaan memori, mewakili total sumber daya yang dicadangkan tugas selama periode penagihan yang Anda tentukan. Anda dapat menggunakan data ini untuk menentukan biaya cluster instans Amazon EC2 yang mendasari Anda. Data biaya dan penggunaan untuk instans Amazon EC2 Anda tercantum secara terpisah di bawah layanan Amazon EC2.

Anda juga dapat menggunakan tag terkelola Amazon ECS untuk mengidentifikasi layanan atau klaster yang dimiliki setiap tugas. Untuk informasi selengkapnya, lihat Menandai sumber daya Anda untuk penagihan.

> **Important**
>
> Data pengukuran hanya dapat dilihat untuk tugas yang diluncurkan pada atau setelah 16 November 2018. Tugas yang diluncurkan sebelum tanggal ini tidak menampilkan data pengukuran.

Berikut ini adalah contoh dari beberapa bidang yang dapat Anda gunakan untuk mengurutkan data alokasi biaya di Cost Explorer.

- Nama klaster
- Nama layanan
- Tanda sumber daya
- Jenis peluncuran
- Wilayah AWS
- Jenis penggunaan

Untuk informasi selengkapnya tentang membuat Laporan AWS Biaya dan Penggunaan, lihat Laporan AWS Biaya dan Penggunaan di Panduan AWS Billing Pengguna.

Laporan Biaya dan Penggunaan Tingkat Tugas

AWS Cost Management dapat menyediakan data penggunaan CPU dan memori di AWS Cost and Usage Report untuk setiap tugas di Amazon ECS, termasuk tugas di Fargate dan tugas di EC2. Data ini disebut Split Cost Allocation Data. Anda dapat menggunakan data ini untuk menganalisis biaya dan penggunaan untuk aplikasi. Selain itu, Anda dapat membagi dan mengalokasikan biaya ke unit bisnis individu dan tim dengan tag alokasi biaya dan kategori biaya. Untuk informasi selengkapnya tentang Split Data Alokasi Biaya, lihat Memahami data alokasi biaya terpisah di AWS Cost and Usage Report Panduan Pengguna.

Anda dapat ikut serta dalam Data Alokasi Biaya Split tingkat tugas untuk akun di. AWS Cost Management Console Jika Anda memiliki akun manajemen (pembayar), Anda dapat ikut serta dari akun pembayar untuk menerapkan konfigurasi ini ke setiap akun yang ditautkan.

Setelah Anda mengatur Data Alokasi Biaya Split, akan ada kolom tambahan di bawah splitLineItemheader dalam laporan. Untuk informasi selengkapnya lihat Membagi detail item baris di Panduan AWS Cost and Usage Report Pengguna

Untuk tugas di EC2, data ini membagi biaya instans EC2 berdasarkan penggunaan sumber daya atau reservasi dan sumber daya yang tersisa pada instans.

Prasyarat

- Untuk menggunakan Split Cost Allocation Data, Anda harus membuat laporan, dan pilih Split data alokasi biaya. Untuk informasi selengkapnya, lihat Membuat Laporan Biaya dan Penggunaan di Panduan AWS Cost and Usage Report Pengguna.
- Versi Docker minimum untuk metrik yang andal adalah Docker versi v20.10.13 dan yang lebih baru, yang disertakan dalam AMI 20220607 yang dioptimalkan Amazon ECS dan yang lebih baru.
- Pastikan bahwa agen Amazon ECS memiliki `ECS_DISABLE_METRICS` konfigurasi yang disetel ke`false`. Saat pengaturan ini`false`, agen Amazon ECS mengirimkan metrik ke Amazon. CloudWatch Di Linux, pengaturan ini secara `false` default dan metrik dikirim ke CloudWatch. Di Windows, pengaturan ini secara `true` default, jadi Anda harus mengubah pengaturan `false` untuk mengirim metrik CloudWatch AWS Cost Management untuk digunakan. Untuk informasi selengkapnya tentang konfigurasi agen ECS, lihatKonfigurasi agen kontainer Amazon ECS.

> **Note**
>
> AWS Cost Management menghitung Data Alokasi Biaya Split dengan CPU tugas dan penggunaan memori. AWS Cost Management dapat menggunakan CPU tugas dan reservasi memori alih-alih penggunaan, jika penggunaan tidak tersedia. Jika Anda melihat CUR menggunakan reservasi, periksa apakah instance container Anda memenuhi prasyarat dan metrik penggunaan sumber daya tugas muncul. CloudWatch

Menyiapkan Laporan Biaya dan Penggunaan Tingkat Tugas

Anda dapat mengaktifkan Data Alokasi Biaya Pisahkan untuk ECS di Konsol Manajemen Biaya AWS Command Line Interface, atau SDK AWS .

Gunakan yang berikut ini untuk Split Data Alokasi Biaya.

1. Pilih untuk Membagi Data Alokasi Biaya. Untuk informasi selengkapnya, lihat Mengaktifkan data alokasi biaya terpisah di AWS Cost and Usage Report Panduan Pengguna.
2. Sertakan data dalam laporan baru atau yang sudah ada.
3. Lihat laporannya. Anda dapat menggunakan konsol Billing and Cost Management atau melihat file laporan di Amazon Simple Storage Service.

Memantau Amazon ECS

Pemantauan adalah bagian penting dalam menjaga keandalan, ketersediaan, dan kinerja Amazon ECS dan AWS solusi Anda. Anda harus mengumpulkan data pemantauan dari semua bagian AWS solusi Anda sehingga Anda dapat lebih mudah men-debug kegagalan multi-titik jika terjadi. Sebelum Anda mulai memantau Amazon ECS, buat rencana pemantauan yang mencakup jawaban atas pertanyaan-pertanyaan berikut:

- Apa tujuan pemantauan Anda?
- Sumber daya apa yang akan Anda pantau?
- Seberapa sering Anda akan memantau sumber daya ini?
- Alat pemantauan apa yang akan Anda gunakan?
- Siapa yang akan melakukan tugas pemantauan?
- Siapa yang harus diberi tahu saat terjadi kesalahan?

Metrik yang tersedia bergantung pada tipe peluncuran tugas dan layanan dalam klaster Anda. Jika Anda menggunakan jenis peluncuran Fargate untuk layanan Anda, maka metrik pemanfaatan CPU dan memori disediakan untuk membantu pemantauan layanan Anda. Untuk jenis peluncuran Amazon EC2, Anda memiliki dan perlu memantau instans EC2 yang membuat infrastruktur dasar Anda. Metrik reservasi dan pemanfaatan CPU dan memori tambahan tersedia di cluster, layanan, dan tugas.

Langkah selanjutnya adalah menetapkan dasar untuk kinerja Amazon ECS normal di lingkungan Anda, dengan mengukur kinerja pada berbagai waktu dan dalam kondisi beban yang berbeda. Saat Anda memantau Amazon ECS, simpan data pemantauan historis sehingga Anda dapat membandingkannya dengan data kinerja saat ini, mengidentifikasi pola kinerja normal dan anomali kinerja, dan merancang metode untuk mengatasi masalah.

Untuk menetapkan batas dasar, setidaknya Anda harus memantau item berikut:

- Metrik reservasi dan pemanfaatan CPU dan memori untuk kluster Amazon ECS Anda
- Metrik pemanfaatan CPU dan memori untuk layanan Amazon ECS Anda

  Untuk informasi selengkapnya, lihat Melihat metrik Amazon ECS.

Topik

Praktik terbaik untuk memantau Amazon ECS

Gunakan praktik terbaik berikut untuk memantau Amazon ECS.

- Jadikan pemantauan sebagai prioritas untuk mengatasi masalah kecil sebelum menjadi masalah besar
- Buat rencana pemantauan yang mencakup jawaban atas pertanyaan berikut
  - Apa tujuan pemantauan Anda?
  - Sumber daya apa yang akan Anda pantau?
  - Seberapa sering Anda akan memantau sumber daya ini?
  - Alat pemantauan apa yang akan Anda gunakan?
  - Siapa yang akan melakukan tugas pemantauan?
  - Siapa yang harus diberi tahu saat terjadi kesalahan?
- Otomatiskan pemantauan sebanyak mungkin.
- Periksa file log Amazon ECS. Untuk informasi selengkapnya, lihat Lokasi file log Amazon ECS.

Alat pemantauan untuk Amazon ECS

AWS menyediakan berbagai alat yang dapat Anda gunakan untuk memantau Amazon ECS. Anda dapat mengonfigurasi beberapa alat ini untuk melakukan pemantauan untuk Anda, sementara beberapa alat memerlukan intervensi manual. Kami menyarankan agar Anda mengautomasi tugas pemantauan sebanyak mungkin.

Alat pemantauan otomatis

Anda dapat menggunakan alat pemantauan otomatis berikut untuk menonton Amazon ECS dan melaporkan ketika ada sesuatu yang salah:

- CloudWatch Alarm Amazon — Tonton satu metrik selama periode waktu yang Anda tentukan, dan lakukan satu atau beberapa tindakan berdasarkan nilai metrik relatif terhadap ambang batas tertentu selama beberapa periode waktu. Tindakannya adalah pemberitahuan yang dikirim ke topik Amazon Simple Notification Service (Amazon SNS) atau kebijakan Amazon EC2 Auto Scaling. CloudWatch alarm tidak memanggil tindakan hanya karena mereka berada dalam keadaan tertentu; negara harus telah berubah dan dipertahankan untuk sejumlah periode tertentu. Untuk informasi selengkapnya, lihat Pantau Amazon ECS menggunakan CloudWatch .

  Untuk layanan dengan tugas yang menggunakan tipe peluncuran Fargate, Anda dapat menggunakan CloudWatch alarm untuk menskalakan dan memperkecil tugas dalam layanan Anda berdasarkan CloudWatch metrik, seperti CPU dan pemanfaatan memori. Untuk informasi selengkapnya, lihat Secara otomatis menskalakan layanan Amazon ECS Anda.

  Untuk cluster dengan tugas atau layanan yang menggunakan tipe peluncuran EC2, Anda dapat menggunakan CloudWatch alarm untuk menskalakan dan memperkecil instance container berdasarkan CloudWatch metrik, seperti reservasi memori cluster.

  Untuk instans penampung yang diluncurkan dengan Amazon Linux AMI Amazon ECS yang dioptimalkan Amazon, Anda dapat CloudWatch menggunakan Log untuk melihat log yang berbeda dari instans penampung Anda di satu lokasi yang nyaman. Anda harus menginstal CloudWatch agen pada instance kontainer Anda. Untuk informasi selengkapnya, lihat Mengunduh dan mengonfigurasi CloudWatch agen menggunakan baris perintah di Panduan CloudWatch Pengguna Amazon. Anda juga harus menambahkan `ECS-CloudWatchLogs` kebijakan ke `ecsInstanceRole` peran tersebut. Untuk informasi selengkapnya, lihat Izin yang diperlukan untuk memantau instance kontainer.

- Amazon CloudWatch Logs — Pantau, simpan, dan akses file log dari container di tugas Amazon ECS Anda dengan menentukan driver `awslogs` log dalam definisi tugas Anda. Untuk informasi selengkapnya, lihat Menggunakan driver log awslogs.

  Anda juga dapat memantau, menyimpan, dan mengakses sistem operasi dan file log agen penampung Amazon ECS dari instans penampung Amazon ECS Anda. Metode ini untuk mengakses log dapat digunakan untuk kontainer menggunakan tipe peluncuran EC2.
- CloudWatch Acara Amazon — Cocokkan acara dan arahkan ke satu atau beberapa fungsi atau aliran target untuk membuat perubahan, menangkap informasi status, dan mengambil tindakan korektif. Untuk informasi lebih lanjut, lihat Otomatiskan tanggapan terhadap kesalahan Amazon ECS menggunakan EventBridge di panduan ini dan Apa Itu CloudWatch Acara Amazon? di Panduan Pengguna CloudWatch Acara Amazon.
- Wawasan Kontainer — Kumpulkan, agregat, dan rangkum metrik dan log dari aplikasi dan layanan mikro dalam kontainer Anda. Wawasan Kontainer mengumpulkan data sebagai peristiwa log performa dengan menggunakan format metrik tersemat. Peristiwa log kinerja ini adalah entri yang menggunakan skema JSON terstruktur yang memungkinkan data kardinalitas tinggi untuk dicerna dan disimpan dalam skala besar. Dari data ini, CloudWatch buat metrik agregat di tingkat klaster, tugas, dan layanan sebagai CloudWatch metrik. Metrik yang dikumpulkan Container Insights tersedia di dasbor CloudWatch otomatis, dan juga dapat dilihat di bagian Metrik konsol. CloudWatch
- AWS CloudTrail pemantauan log - Bagikan file log antar akun, pantau file CloudTrail log secara real time dengan mengirimkannya ke CloudWatch Log, menulis aplikasi pemrosesan log di Java, dan validasi bahwa file log Anda tidak berubah setelah pengiriman oleh CloudTrail. Untuk informasi selengkapnya, lihat Log panggilan Amazon ECS API menggunakan AWS CloudTrail di panduan ini, dan Bekerja dengan File CloudTrail Log di Panduan AWS CloudTrail Pengguna.
- Runtime Monitoring — Mendeteksi ancaman untuk cluster dan kontainer dalam lingkungan Anda AWS . Runtime Monitoring menggunakan agen GuardDuty keamanan yang menambahkan visibilitas runtime ke beban kerja Amazon ECS individual, misalnya, akses file, eksekusi proses, dan koneksi jaringan.

Alat pemantauan manual

Bagian penting lainnya dari pemantauan Amazon ECS melibatkan pemantauan secara manual item-item yang tidak CloudWatch tercakup oleh alarm. Dasbor AWS konsol CloudWatch Trusted Advisor,, dan lainnya memberikan at-a-glance tampilan status AWS lingkungan Anda. Kami menyarankan Anda untuk memeriksa file log pada instans kontainer Anda dan kontainer dalam tugas Anda.

- Konsol Amazon ECS:
  - Metrik cluster untuk tipe peluncuran EC2
  - Metrik Layanan
  - Status kondisi layanan
  - Acara penyebaran layanan
- CloudWatch halaman rumah:
  - Alarm dan status saat ini
  - Grafik alarm dan sumber daya
  - Status kondisi layanan

  Selain itu, Anda dapat menggunakan CloudWatch untuk melakukan hal berikut:
  - Buat dasbor yang disesuaikan untuk memantau layanan yang Anda pedulikan.
  - Data metrik grafik untuk memecahkan masalah dan menemukan tren.
  - Cari dan telusuri semua metrik AWS sumber daya Anda.
  - Membuat dan mengedit alarm agar diberi tahu tentang masalah.
- Pemeriksaan kesehatan kontainer - Ini adalah perintah yang berjalan secara lokal pada wadah dan memvalidasi kesehatan dan ketersediaan aplikasi. Anda mengonfigurasi ini per kontainer dalam definisi tugas Anda.
- AWS Trusted Advisor dapat membantu Anda memantau AWS sumber daya Anda untuk meningkatkan kinerja, keandalan, keamanan, dan efektivitas biaya. Empat Trusted Advisor cek tersedia untuk semua pengguna; lebih dari 50 cek tersedia untuk pengguna dengan paket dukungan Bisnis atau Perusahaan. Untuk informasi selengkapnya, lihat AWS Trusted Advisor.

  Trusted Advisor memiliki pemeriksaan ini yang berhubungan dengan Amazon ECS:
  - Toleransi kesalahan yang menunjukkan bahwa Anda memiliki layanan yang berjalan di satu Availability Zone.
  - Toleransi kesalahan yang menunjukkan bahwa Anda belum menggunakan strategi penempatan spread untuk beberapa Availability Zone.
- AWS Compute Optimizer adalah layanan yang menganalisis konfigurasi dan metrik pemanfaatan sumber daya Anda. AWS Ini melaporkan apakah sumber daya Anda optimal, dan menghasilkan rekomendasi pengoptimalan untuk mengurangi biaya dan meningkatkan kinerja beban kerja Anda.

  Untuk informasi selengkapnya, lihat AWS Compute Optimizer rekomendasi untuk Amazon ECS.

Pantau Amazon ECS menggunakan CloudWatch

Anda dapat memantau sumber daya Amazon ECS menggunakan Amazon CloudWatch, yang mengumpulkan dan memproses data mentah dari Amazon ECS menjadi metrik yang dapat dibaca, mendekati waktu nyata. Statistik ini dicatat untuk jangka waktu dua minggu sehingga Anda dapat mengakses informasi historis dan mendapatkan perspektif yang lebih baik tentang kinerja cluster atau layanan Anda. Data metrik Amazon ECS secara otomatis dikirim ke CloudWatch dalam periode 1 menit. Untuk informasi selengkapnya CloudWatch, lihat Panduan CloudWatch Pengguna Amazon.

Amazon ECS menyediakan metrik gratis untuk cluster dan layanan. Dengan biaya tambahan, Anda dapat mengaktifkan Amazon ECS CloudWatch Container Insights untuk klaster Anda untuk metrik per tugas, termasuk pemanfaatan sistem file CPU, memori, dan EBS. Untuk informasi selengkapnya tentang Wawasan Kontainer ini, silakan lihat Pantau kontainer Amazon ECS menggunakan Wawasan Kontainer.

Pertimbangan

Berikut ini harus dipertimbangkan saat menggunakan CloudWatch metrik Amazon ECS.

- Setiap layanan Amazon ECS yang dihosting di Fargate CloudWatch memiliki metrik pemanfaatan CPU dan memori secara otomatis, jadi Anda tidak perlu mengambil langkah manual apa pun.

- Untuk tugas atau layanan Amazon ECS apa pun yang dihosting di instans Amazon EC2, instans Amazon EC2 memerlukan `1.4.0` versi atau yang lebih baru (Linux) `1.0.0` atau yang lebih baru (Windows) dari agen CloudWatch penampung untuk metrik yang akan dihasilkan. Namun, kami menyarankan untuk menggunakan versi agen kontainer terbaru. Untuk informasi tentang memeriksa versi agen Anda dan memperbarui ke versi terbaru, lihat Memperbarui agen kontainer Amazon ECS.
- Versi Docker minimum untuk CloudWatch metrik yang andal adalah versi Docker dan yang lebih baru`20.10.13`.
- Instans Amazon EC2 Anda juga memerlukan `ecs:StartTelemetrySession` izin pada peran IAM yang digunakan untuk meluncurkan instans Amazon EC2. Jika Anda membuat peran IAM instans penampung Amazon ECS sebelum CloudWatch metrik tersedia untuk Amazon ECS, Anda mungkin perlu menambahkan izin ini. Untuk informasi tentang peran IAM instance container dan melampirkan kebijakan IAM terkelola untuk instance container, lihat. Peran IAM instans wadah Amazon ECS

- Anda dapat menonaktifkan pengumpulan CloudWatch metrik di instans Amazon EC2 dengan menyetel konfigurasi agen `ECS_DISABLE_METRICS=true` penampung Amazon ECS Anda. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.

Metrik dan dimensi yang tersedia untuk Amazon ECS

Amazon ECS menyediakan CloudWatch metrik gratis yang dapat Anda gunakan untuk memantau sumber daya Anda.

Metrik mewakili kumpulan titik data yang diurutkan waktu yang dipublikasikan ke CloudWatch. Infrastruktur tugas Amazon ECS Anda di-host di cluster Anda menentukan metrik mana yang tersedia.

CloudWatch metrik memiliki ruang nama, dimensi, dan statistik. Ruang nama digunakan untuk mengisolasi metrik. Namespace Amazon ECS adalah. `AWS/ECS`

Dimensi yang merupakan pasangan nama/nilai yang merupakan bagian dari identitas metrik. Dimensi seperti kategori untuk metrik. Dimensi untuk Amazon ECS meliputi`ClusterName`, dan`ServiceName`.

Statistik adalah agregasi data metrik selama periode waktu tertentu. CloudWatch menyediakan statistik berdasarkan titik data metrik yang disediakan oleh data kustom Anda atau disediakan oleh layanan lain di dalamnya AWS CloudWatch. Agregasi dilakukan menggunakan namespace, nama metrik, dimensi, dan unit titik data dari ukuran, dalam periode waktu yang Anda tentukan.

Amazon ECS mengirimkan metrik berikut ke CloudWatch setiap menit. Saat Amazon ECS mengumpulkan metrik, Amazon mengumpulkan beberapa titik data setiap menit. Kemudian mengagregat mereka ke satu titik data sebelum mengirim data ke CloudWatch. Jadi dalam CloudWatch, satu jumlah sampel sebenarnya adalah agregat dari beberapa titik data selama satu menit.

Statistik	Deskripsi
`Average`	Nilai dari `Sum` / `SampleCount` selama periode yang ditentukan. Dengan membandingkan statistik ini dengan `Minimum` dan `Maximum`, Anda dapat menentukan cakupan suatu metrik dan seberapa dekat rata-rata penggunaa n dengan `Minimum` dan `Maximum`. Perbandingan ini membantu Anda untuk

Statistik	Deskripsi
	mengetahui kapan harus menambah atau mengurangi sumber daya Anda sesuai kebutuhan.
`Maximum`	Nilai tertinggi yang diamati selama periode yang ditentukan. Anda dapat menggunakan nilai ini untuk menentukan volume aktivitas yang tinggi pada aplikasi Anda.
`Minimum`	Nilai terendah yang diamati selama periode yang ditentukan. Anda dapat menggunakan nilai ini untuk menentukan volume aktivitas yang rendah pada aplikasi Anda.
`SampleCount`	Hitungan (jumlah) titik data yang digunakan untuk penghitungan statistik.
`Sum`	Semua nilai yang dikirimkan untuk metrik yang cocok disatukan. Statistik ini dapat berguna untuk menentukan total volume metrik.

Topik

Metrik Amazon ECS

Amazon ECS menyediakan CloudWatch metrik gratis yang dapat Anda gunakan untuk memantau sumber daya Anda. Reservasi CPU dan memori serta pemanfaatan sistem file CPU, memori, dan EBS di seluruh cluster Anda secara keseluruhan, dan pemanfaatan sistem file CPU, memori, dan EBS pada layanan di cluster Anda dapat diukur menggunakan metrik ini. Untuk beban kerja GPU, Anda dapat mengukur reservasi GPU di klaster Anda.

Infrastruktur tugas Amazon ECS Anda di-host di cluster Anda menentukan metrik mana yang tersedia. Untuk tugas yang dihosting di infrastruktur Fargate, Amazon ECS menyediakan CPU,

memori, dan metrik pemanfaatan sistem file EBS yang disediakan untuk membantu pemantauan layanan Anda. Untuk tugas yang dihosting di instans EC2, Amazon ECS menyediakan metrik reservasi CPU, memori, dan GPU serta metrik pemanfaatan CPU dan memori di tingkat klaster dan layanan. Anda perlu memantau instans Amazon EC2 yang membuat infrastruktur dasar Anda secara terpisah. Untuk informasi selengkapnya tentang memantau instans Amazon EC2, lihat Memantau Amazon EC2 di Panduan Pengguna Amazon EC2 untuk Instans Linux.

`AWS/ECS`Namespace CloudWatch termasuk metrik berikut.

`CPUReservation`

Persentase unit CPU yang dicadangkan dalam cluster atau layanan.

Reservasi CPU (disaring oleh`ClusterName`) diukur sebagai total unit CPU yang dicadangkan oleh tugas Amazon ECS di cluster, dibagi dengan total unit CPU untuk semua instans Amazon EC2 yang terdaftar di cluster. Hanya instans Amazon EC2 dalam `ACTIVE` atau `DRAINING` status yang akan memengaruhi metrik reservasi CPU. Metrik ini hanya didukung untuk tugas yang dihosting di instans Amazon EC2.

Alasan untuk menggunakan: Anda dapat menggunakan alarm ini untuk menentukan kapan harus meningkatkan skala atau menambahkan kapasitas tambahan.

Nilai yang disarankan: Kami menyarankan Anda mengatur metrik ini ke 90%. Kami tidak menyarankan Anda menggunakan alarm ini ketika Anda memiliki penyedia kapasitas EC2 dengan penskalaan terkelola diaktifkan, atau Anda memiliki penyedia kapasitas Fargate. Dalam kasus ini, Amazon ECS mengelola penskalaan atas nama Anda.

Dimensi yang valid: `ClusterName`.

Statistik yang berguna: Rata-rata, Minimum, Maksimum

Unit: Persen.

`CPUUtilization`

Persentase unit CPU yang digunakan oleh cluster atau layanan.

Pemanfaatan CPU tingkat cluster (disaring oleh`ClusterName`) diukur sebagai total unit CPU yang digunakan oleh tugas Amazon ECS di cluster, dibagi dengan total unit CPU untuk semua instans Amazon EC2 yang terdaftar di cluster. Hanya instans Amazon EC2 dalam `ACTIVE` atau `DRAINING` status yang akan memengaruhi metrik reservasi CPU. Metrik tingkat klaster hanya didukung untuk tugas yang dihosting di instans Amazon EC2.

Pemanfaatan CPU tingkat layanan (disaring oleh`ClusterName`,`ServiceName`) diukur sebagai total unit CPU yang digunakan oleh tugas-tugas yang termasuk dalam layanan, dibagi dengan jumlah total unit CPU yang dicadangkan untuk tugas-tugas yang termasuk dalam layanan. Metrik tingkat layanan didukung untuk tugas yang dihosting di instans Amazon EC2 dan Fargate.

Alasan penggunaan: Anda dapat menggunakan metrik ini untuk menentukan kapan harus menambahkan lebih banyak CPU untuk instance atau tugas Anda sehingga tidak ada hambatan sumber daya atau masalah kinerja aplikasi.

Nilai yang disarankan: Meskipun metrik tingkat layanan mungkin melebihi pemanfaatan 100%, kami sarankan Anda menetapkan ambang batas ke 90-95%. Ini dapat membantu menghindari dampak pada layanan lain. Kami menyarankan Anda memperbarui definisi tugas Anda untuk mencerminkan penggunaan aktual untuk mencegah masalah di masa mendatang dengan layanan.

Dimensi yang valid: `ClusterName`, `ServiceName`.

Statistik yang berguna: Rata-rata, Minimum, Maksimum

Unit: Persen.

`MemoryReservation`

Persentase memori yang disimpan oleh tugas yang berjalan di klaster.

Reservasi memori cluster diukur sebagai total memori yang dicadangkan oleh tugas Amazon ECS di cluster, dibagi dengan jumlah total memori untuk semua instans Amazon EC2 yang terdaftar di cluster. Metrik ini hanya dapat disaring oleh`ClusterName`. Hanya instans Amazon EC2 dalam `ACTIVE` atau `DRAINING` status yang akan memengaruhi metrik reservasi memori. Metrik reservasi memori tingkat cluster hanya didukung untuk tugas yang dihosting di instans Amazon EC2.

> **Note**
>
> Saat menghitung pemanfaatan memori, jika `MemoryReservation` ditentukan, itu digunakan dalam perhitungan alih-alih memori total.

Alasan untuk menggunakan: Anda dapat menggunakan metrik ini untuk menentukan kapan harus skala cluster. Ketika unit memori total untuk klaster tersebut telah tercapai, hal itu dapat menyebabkan klaster tidak dapat meluncurkan tugas baru.

Nilai yang disarankan: Kami menyarankan Anda mengatur metrik ini ke 90%. Kami tidak menyarankan Anda menggunakan alarm ini ketika Anda memiliki penyedia kapasitas EC2 dengan penskalaan terkelola diaktifkan, atau Anda memiliki penyedia kapasitas Fargate. Dalam kasus ini, Amazon ECS mengelola penskalaan atas nama Anda.

Dimensi yang valid: `ClusterName`.

Statistik yang berguna: Rata-rata, Minimum, Maksimum

Unit: Persen.

`MemoryUtilization`

Persentase memori yang digunakan oleh cluster atau layanan.

Pemanfaatan memori tingkat cluster (disaring oleh`ClusterName`) diukur sebagai total memori yang digunakan oleh tugas Amazon ECS di cluster, dibagi dengan total memori untuk semua instans Amazon EC2 yang terdaftar di cluster. Hanya instans Amazon EC2 dalam `ACTIVE` atau `DRAINING` status yang akan memengaruhi metrik pemanfaatan memori. Metrik tingkat klaster hanya didukung untuk tugas yang dihosting di instans Amazon EC2.

Pemanfaatan memori tingkat layanan (disaring oleh`ClusterName`,`ServiceName`) diukur sebagai memori total yang digunakan oleh tugas-tugas milik layanan, dibagi dengan total memori yang disediakan untuk tugas-tugas yang termasuk dalam layanan. Metrik tingkat layanan didukung untuk tugas yang dihosting di instans Amazon EC2 dan Fargate.

Alasan penggunaan: Anda dapat menggunakan metrik ini untuk menentukan kapan harus menambahkan lebih banyak memori untuk instance atau tugas Anda sehingga tidak ada hambatan sumber daya atau masalah kinerja aplikasi.

Nilai yang disarankan: Kami menyarankan Anda menetapkan ambang batas ke 90-95%. Ini dapat membantu menghindari dampak pada layanan lain. Kami menyarankan Anda memperbarui definisi tugas Anda untuk mencerminkan penggunaan aktual untuk mencegah masalah di masa mendatang dengan layanan.

Dimensi yang valid: `ClusterName`, `ServiceName`.

Statistik yang berguna: Rata-rata, Minimum, Maksimum

Unit: Persen.

`EBSFilesystemUtilization`

Persentase sistem file Amazon EBS yang digunakan oleh tugas dalam suatu layanan.

Metrik pemanfaatan sistem file EBS tingkat layanan (disaring oleh`ClusterName`,`ServiceName`) diukur sebagai jumlah total sistem file EBS yang digunakan oleh tugas-tugas yang termasuk dalam layanan, dibagi dengan jumlah total penyimpanan sistem file EBS yang dialokasikan untuk semua tugas yang termasuk dalam layanan. Metrik pemanfaatan sistem file EBS tingkat layanan hanya tersedia untuk tugas yang dihosting di instans Amazon EC2 (menggunakan versi agen kontainer) `1.79.0` dan Fargate (menggunakan versi platform) yang memiliki volume EBS yang terpasang. `1.4.0`

> **Note**
>
> Untuk tugas yang di-host di Fargate, ada ruang pada disk yang hanya digunakan oleh Fargate. Tidak ada biaya yang terkait dengan ruang yang digunakan Fargate, tetapi Anda akan melihat penyimpanan tambahan ini menggunakan alat seperti. `df`

Dimensi yang valid: `ClusterName, ServiceName`.

Statistik yang berguna: Rata-rata, Minimum, Maksimum

Unit: Persen.

`GPUReservation`

Persentase total GPU tersedia yang disimpan oleh tugas yang berjalan di klaster.

Metrik reservasi GPU tingkat cluster diukur sebagai jumlah GPU yang dicadangkan oleh tugas Amazon ECS di cluster, dibagi dengan jumlah total GPU yang tersedia di semua instans Amazon EC2 dengan GPU yang terdaftar di cluster. Hanya instans Amazon EC2 dalam `ACTIVE` atau `DRAINING` status yang akan memengaruhi metrik reservasi GPU.

Dimensi yang valid: `ClusterName`.

Statistik yang berguna: Rata-rata, Minimum, Maksimum

Semua statistik: Rata-rata, Minimum, Maksimum, Jumlah, Jumlah Sampel.

Unit: Persen.

`ActiveConnectionCount`

Jumlah total koneksi bersamaan yang aktif dari klien ke proxy Amazon ECS Service Connect yang berjalan dalam tugas yang berbagi yang dipilih. `DiscoveryName`

Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect.

Dimensi yang valid: `DiscoveryName` dan`DiscoveryName, ServiceName, ClusterName`.

Statistik yang berguna: Rata-rata, Minimum, Maksimum, Jumlah.

Satuan: Hitung.

`NewConnectionCount`

Jumlah total koneksi baru yang dibuat dari klien ke proxy Amazon ECS Service Connect yang berjalan dalam tugas yang berbagi yang dipilih. `DiscoveryName`

Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect.

Dimensi yang valid: `DiscoveryName` dan`DiscoveryName, ServiceName, ClusterName`.

Statistik yang berguna: Rata-rata, Minimum, Maksimum, Jumlah.

Satuan: Hitung.

`ProcessedBytes`

Jumlah total byte lalu lintas masuk yang diproses oleh proxy Service Connect.

Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect.

Dimensi yang valid: `DiscoveryName` dan`DiscoveryName, ServiceName, ClusterName`.

Statistik yang berguna: Rata-rata, Minimum, Maksimum, Jumlah.

Unit: Byte.

`RequestCount`

Jumlah permintaan lalu lintas masuk yang diproses oleh proxy Service Connect.

Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect.

Dimensi yang valid: `DiscoveryName` dan`DiscoveryName, ServiceName, ClusterName`.

Statistik yang berguna: Rata-rata, Minimum, Maksimum, Jumlah.

Satuan: Hitung.

`GrpcRequestCount`

Jumlah permintaan lalu lintas gRPC masuk yang diproses oleh proxy Service Connect.

Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect dan `appProtocol` berada `GRPC` di pemetaan port dalam definisi tugas.

Dimensi yang valid: `DiscoveryName` dan`DiscoveryName, ServiceName, ClusterName`.

Statistik yang berguna: Rata-rata, Minimum, Maksimum, Jumlah.

Satuan: Hitung.

`HTTPCode_Target_2XX_Count`

Jumlah kode respons HTTP dengan angka 200 hingga 299 yang dihasilkan oleh aplikasi dalam tugas-tugas ini. Tugas-tugas ini adalah targetnya. Metrik ini hanya menghitung respons yang dikirim ke proxy Service Connect oleh aplikasi dalam tugas ini, bukan tanggapan yang dikirim secara langsung.

Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect dan `appProtocol` sedang `HTTP` atau `HTTP2` dalam pemetaan port dalam definisi tugas.

Dimensi yang valid: `TargetDiscoveryName` dan`TargetDiscoveryName, ServiceName, ClusterName`.

Statistik yang berguna: Rata-rata, Minimum, Maksimum, Jumlah.

Satuan: Hitung.

`HTTPCode_Target_3XX_Count`

Jumlah kode respons HTTP dengan angka 300 hingga 399 yang dihasilkan oleh aplikasi dalam tugas-tugas ini. Tugas-tugas ini adalah targetnya. Metrik ini hanya menghitung respons yang dikirim ke proxy Service Connect oleh aplikasi dalam tugas ini, bukan tanggapan yang dikirim secara langsung.

Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect dan `appProtocol` sedang `HTTP` atau `HTTP2` dalam pemetaan port dalam definisi tugas.

Dimensi yang valid: `TargetDiscoveryName` dan`TargetDiscoveryName, ServiceName, ClusterName`.

Statistik yang berguna: Rata-rata, Minimum, Maksimum, Jumlah.

Satuan: Hitung.

`HTTPCode_Target_4XX_Count`

Jumlah kode respons HTTP dengan angka 400 hingga 499 yang dihasilkan oleh aplikasi dalam tugas-tugas ini. Tugas-tugas ini adalah targetnya. Metrik ini hanya menghitung respons yang dikirim ke proxy Service Connect oleh aplikasi dalam tugas ini, bukan tanggapan yang dikirim secara langsung.

Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect dan `appProtocol` sedang `HTTP` atau `HTTP2` dalam pemetaan port dalam definisi tugas.

Dimensi yang valid: `TargetDiscoveryName` dan`TargetDiscoveryName, ServiceName, ClusterName`.

Statistik yang berguna: Rata-rata, Minimum, Maksimum, Jumlah

Satuan: Hitung.

`HTTPCode_Target_5XX_Count`

Jumlah kode respons HTTP dengan angka 500 hingga 599 yang dihasilkan oleh aplikasi dalam tugas-tugas ini. Tugas-tugas ini adalah targetnya. Metrik ini hanya menghitung respons yang dikirim ke proxy Service Connect oleh aplikasi dalam tugas ini, bukan tanggapan yang dikirim secara langsung.

Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect dan `appProtocol` sedang `HTTP` atau `HTTP2` dalam pemetaan port dalam definisi tugas.

Alasan penggunaan: Anda dapat menggunakan metrik ini untuk mendeteksi jumlah kesalahan sisi server yang tinggi untuk suatu layanan.

Nilai yang disarankan: Kami menyarankan agar Anda awalnya menetapkan nilai menjadi 5% dari lalu lintas rata-rata Anda. Anda dapat menggunakan `RequestCount` metrik untuk menemukan lalu lintas rata-rata. Anda mungkin perlu menyesuaikan nilainya agar alarm tidak terlalu sensitif.

Statistik yang berguna: Rata-rata, Minimum, Maksimum, Jumlah.

Satuan: Hitung.

`RequestCountPerTarget`

Rata-rata jumlah permintaan yang diterima oleh setiap target yang berbagi yang dipilih`DiscoveryName`.

Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect.

Dimensi yang valid: `TargetDiscoveryName` dan`TargetDiscoveryName, ServiceName, ClusterName`.

Statistik yang berguna: Rata-rata.

Satuan: Hitung.

`TargetProcessedBytes`

Jumlah total byte yang diproses oleh proxy Service Connect.

Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect.

Dimensi yang valid: `TargetDiscoveryName` dan`TargetDiscoveryName, ServiceName, ClusterName`.

Statistik yang berguna: Rata-rata, Minimum, Maksimum, Jumlah.

Unit: Byte.

`TargetResponseTime`

Latensi pemrosesan permintaan aplikasi. Waktu berlalu, dalam milidetik, setelah permintaan mencapai proxy Service Connect di tugas target hingga respons dari aplikasi target diterima kembali ke proxy.

Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect.

Alasan untuk menggunakan: Anda dapat menggunakan metrik ini untuk mendeteksi waktu respons layanan yang tinggi. Anda kemudian dapat melihat metrik lain seperti `CPUUtilization` untuk menentukan apakah layanan memiliki sumber daya yang cukup.

Nilai yang disarankan: Tinjau waktu respons kritis dan perilaku historis metrik untuk menentukan nilai.

Dimensi yang valid: `TargetDiscoveryName` dan`TargetDiscoveryName, ServiceName, ClusterName`.

Statistik yang berguna: Rata-rata, Minimum, Maksimum.

Semua statistik: Rata-rata, Minimum, Maksimum, Jumlah, Jumlah Sampel.

Satuan: Milidetik.

`ClientTLSNegotiationErrorCount`

Jumlah total kali koneksi TLS gagal. Metrik ini hanya digunakan ketika TLS diaktifkan.

Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect.

Dimensi yang valid: `DiscoveryName` dan`DiscoveryName`,`ServiceName`,`ClusterName`.

Statistik yang berguna: Rata-rata, Minimum, Maksimum, Jumlah.

Satuan: Hitung.

`TargetTLSNegotiationErrorCount`

Jumlah total kali koneksi TLS gagal karena sertifikat klien hilang, AWS Private CA verifikasi gagal, atau verifikasi SAN gagal. Metrik ini hanya digunakan ketika TLS diaktifkan.

Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect.

Dimensi yang valid: `ServiceNameClusterName`,, `TargetDiscoveryName` dan`TargetDiscoveryName`.

Statistik yang berguna: Rata-rata, Minimum, Maksimum, Jumlah.

Satuan: Hitung.

Dimensi untuk metrik Amazon ECS

Metrik Amazon ECS menggunakan `AWS/ECS` namespace dan menyediakan metrik untuk dimensi berikut. Amazon ECS hanya mengirimkan metrik untuk sumber daya yang memiliki tugas di negara bagian. `RUNNING` Misalnya, jika Anda memiliki sebuah klaster dengan satu layanan di dalamnya namun layanan tersebut tidak memiliki tugas tidak di status `RUNNING`, maka tidak akan ada metrik

yang dikirimkan ke CloudWatch. Jika Anda memiliki dua layanan dan salah satunya memiliki tugas yang sedang berjalan sementara yang satunya tidak, hanya metrik untuk layanan dengan tugas yang sedang berjalanlah yang akan dikirim.

`ClusterName`

Dimensi ini memfilter data yang Anda minta untuk semua sumber daya di klaster tertentu. Semua metrik Amazon ECS difilter oleh. `ClusterName`

`ServiceName`

Dimensi ini memfilter data yang Anda minta untuk semua sumber daya di layanan tertentu dalam klaster tertentu.

`DiscoveryName`

Dimensi ini menyaring data yang Anda minta untuk metrik lalu lintas ke nama penemuan Service Connect yang ditentukan di semua kluster Amazon ECS.

Perhatikan bahwa port tertentu dalam wadah yang sedang berjalan dapat memiliki beberapa nama penemuan.

`DiscoveryName, ServiceName, ClusterName`

Dimensi ini menyaring data yang Anda minta untuk metrik lalu lintas ke nama penemuan Service Connect tertentu di seluruh tugas yang memiliki nama penemuan ini dan yang dibuat oleh layanan ini di klaster ini.

Gunakan dimensi ini untuk melihat metrik lalu lintas masuk untuk layanan tertentu, jika Anda telah menggunakan kembali nama penemuan yang sama di beberapa layanan di ruang nama yang berbeda.

Perhatikan bahwa port tertentu dalam wadah yang sedang berjalan dapat memiliki beberapa nama penemuan.

`TargetDiscoveryName`

Dimensi ini menyaring data yang Anda minta untuk metrik lalu lintas ke nama penemuan Service Connect yang ditentukan di semua kluster Amazon ECS.

Berbeda dari`DiscoveryName`, metrik lalu lintas ini hanya mengukur lalu lintas masuk ke ini `DiscoveryName` yang berasal dari tugas Amazon ECS lain yang memiliki konfigurasi Service

Connect di namespace ini. Ini termasuk tugas yang dibuat oleh layanan dengan konfigurasi Service Connect khusus klien atau server klien.

Perhatikan bahwa port tertentu dalam wadah yang sedang berjalan dapat memiliki beberapa nama penemuan.

`TargetDiscoveryName, ServiceName, ClusterName`

Dimensi ini memfilter data yang Anda minta untuk metrik lalu lintas ke nama penemuan Service Connect yang ditentukan tetapi hanya menghitung lalu lintas dari tugas yang dibuat oleh layanan ini di klaster ini.

Gunakan dimensi ini untuk melihat metrik lalu lintas masuk yang berasal dari klien tertentu di layanan lain.

Berbeda dari`DiscoveryName, ServiceName, ClusterName`, metrik lalu lintas ini hanya mengukur lalu lintas masuk ke ini `DiscoveryName` yang berasal dari tugas Amazon ECS lain yang memiliki konfigurasi Service Connect di namespace ini. Ini termasuk tugas yang dibuat oleh layanan dengan konfigurasi Service Connect khusus klien atau server klien.

Perhatikan bahwa port tertentu dalam wadah yang sedang berjalan dapat memiliki beberapa nama penemuan.

Metrik reservasi klaster

Metrik reservasi klaster diukur sebagai persentase CPU, memori, dan GPU yang dicadangkan oleh semua tugas Amazon ECS di klaster jika dibandingkan dengan CPU agregat, memori, dan GPU yang terdaftar untuk setiap instans kontainer aktif di cluster. Hanya instans kontainer di status `ACTIVE` atau `DRAINING` yang akan mempengaruhi metrik reservasi klaster. Metrik ini hanya digunakan pada cluster dengan tugas atau layanan yang dihosting pada instans EC2. Itu tidak didukung pada cluster dengan tugas yang dihosting. AWS Fargate

```
                              (Total CPU units reserved by tasks in cluster) x 100
Cluster CPU reservation =
 ---------------------------------------------------------------
                           (Total CPU units registered by container instances in
 cluster)
```

```
                                  (Total MiB of memory reserved by tasks in cluster x
 100)
```

```
Cluster memory reservation =
 ------------------------------------------------------------------
                              (Total MiB of memory registered by container instances in
 cluster)
```

```
                                  (Total GPUs reserved by tasks in cluster x 100)
Cluster GPU reservation =
 ------------------------------------------------------------------
                              (Total GPUs registered by container instances in cluster)
```

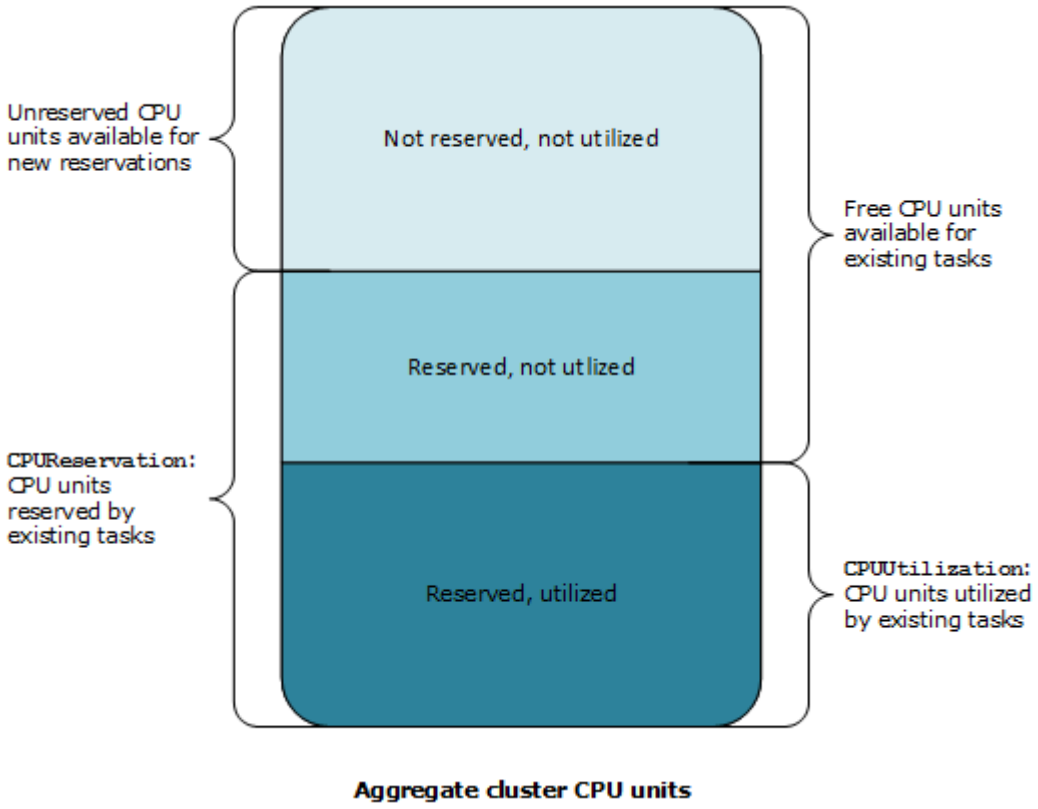
Saat Anda menjalankan tugas di klaster, Amazon ECS mem-parsing definisi tugasnya dan menyimpan unit CPU agregat, MiB memori, dan GPU yang ditentukan dalam definisi kontainernya. Setiap menit, Amazon ECS menghitung jumlah unit CPU, MiB memori, dan GPU yang saat ini dicadangkan untuk setiap tugas yang berjalan di cluster. Jumlah total CPU, memori, dan GPU yang disediakan untuk semua tugas yang berjalan di cluster dihitung, dan angka-angka tersebut dilaporkan CloudWatch sebagai persentase dari total sumber daya terdaftar untuk cluster. Jika Anda menentukan batas lunak (`memoryReservation`) dalam definisi tugas, itu digunakan untuk menghitung jumlah memori cadangan. Jika tidak, hard limit (`memory`) akan digunakan. Total MiB memori yang dicadangkan oleh tugas dalam cluster juga mencakup ukuran volume sistem file sementara (`tmpfs`) dan `sharedMemorySize` jika didefinisikan dalam definisi tugas. Untuk informasi selengkapnya tentang batas keras dan lunak, ukuran memori bersama, dan ukuran volume tmpfs, lihat Parameter Definisi Tugas.

Misalnya, sebuah klaster memiliki dua instans kontainer aktif yang terdaftar: instans `c4.4xlarge` dan instans `c4.large`. Instans `c4.4xlarge` mendaftar ke klaster dengan 16.384 unit CPU dan 30.158 MiB memori. Instans `c4.large` mendaftar dengan 2.048 unit CPU dan 3.768 MiB memori. Sumber daya agregat klaster ini adalah 18.432 unit CPU dan 33.926 MiB memori.

Jika suatu ketentuan tugas menyimpan 1.024 unit CPU dan 2.048 MiB memori, dan sepuluh tugas dimulai dengan ketentuan tugas di klaster ini (dan tidak ada tugas lain yang sedang berjalan), maka sejumlah 10.240 unit CPU dan 20,480 MiB memori direservasi. Ini dilaporkan CloudWatch sebagai reservasi CPU 55% dan reservasi memori 60% untuk cluster.

Ilustrasi berikut menunjukkan jumlah unit CPU yang terdaftar dalam sebuah klaster serta makna reservasi dan pemanfaatan tersebut bagi tugas yang ada dan penempatan tugas baru. Blok bawah (Reserved, used) dan center (Reserved, not used) mewakili total unit CPU yang dicadangkan untuk tugas yang ada yang berjalan di cluster, atau `CPUReservation` CloudWatch metrik. Blok bawah mewakili unit CPU cadangan yang sebenarnya digunakan oleh tugas yang sedang berjalan di cluster, atau `CPUUtilization` CloudWatch metrik. Blok atas mewakili unit CPU yang tidak disimpan oleh

tugas yang ada; unit CPU ini tersedia untuk penempatan tugas baru. Tugas yang ada juga dapat menggunakan unit CPU yang tidak tersimpan, jika kebutuhan sumber daya CPU meningkat. Untuk informasi selengkapnya, lihat dokumentasi parameter ketentuan tugas cpu.



Pemanfaatan klaster

Metrik pemanfaatan cluster tersedia untuk CPU, memori, dan, ketika ada volume EBS yang melekat pada tugas Anda, pemanfaatan sistem file EBS. Metrik ini hanya tersedia untuk klaster dengan tugas atau layanan yang dihosting di instans Amazon EC2. Mereka tidak didukung pada cluster dengan tugas yang dihosting. AWS Fargate

Pemanfaatan CPU dan memori tingkat cluster

Pemanfaatan CPU dan memori diukur sebagai persentase CPU dan memori yang digunakan oleh semua tugas pada cluster jika dibandingkan dengan CPU agregat dan memori yang terdaftar untuk setiap instans Amazon EC2 aktif yang terdaftar ke cluster. Hanya instans Amazon EC2 dalam `ACTIVE` atau `DRAINING` status yang akan memengaruhi metrik pemanfaatan klaster.

```
                                (Total CPU units used by tasks in cluster) x 100
Cluster CPU utilization =
 ---------------------------------------------------------------
```

```
                          (Total CPU units registered by container instances in
 cluster)
```

```
                                    (Total MiB of memory used by tasks in cluster x
 100)
Cluster memory utilization =
 ------------------------------------------------------------------
                             (Total MiB of memory registered by container instances in
 cluster)
```

Setiap menit, agen penampung Amazon ECS di setiap instans Amazon EC2 menghitung jumlah unit CPU dan MiB memori yang saat ini digunakan untuk setiap tugas yang berjalan pada instans Amazon EC2 itu, dan informasi ini dilaporkan kembali ke Amazon ECS. Jumlah total CPU dan memori yang digunakan untuk semua tugas yang berjalan di cluster dihitung, dan angka-angka tersebut dilaporkan CloudWatch sebagai persentase dari total sumber daya terdaftar untuk cluster.

Misalnya, klaster memiliki dua instans Amazon EC2 aktif yang terdaftar, `c4.4xlarge` instance dan instance. `c4.large` `c4.4xlarge`Instance mendaftar ke dalam cluster dengan unit `16,384` CPU dan `30,158` MiB memori. `c4.large`Instans mendaftar dengan unit `2,048` CPU dan `3,768` MiB memori. Sumber daya agregat dari cluster ini adalah unit `18,432` CPU dan `33,926` MiB memori.

Jika sepuluh tugas berjalan pada cluster ini dan setiap tugas menghabiskan unit `1,024` CPU dan `2,048` MiB memori, `10,240` total unit CPU dan `20,480` MiB memori digunakan pada cluster. Ini dilaporkan CloudWatch sebagai 55% pemanfaatan CPU dan 60% pemanfaatan memori untuk cluster.

Pemanfaatan sistem file EBS tingkat cluster

Metrik pemanfaatan sistem file EBS tingkat cluster diukur sebagai jumlah total sistem file EBS yang digunakan oleh tugas yang berjalan di cluster, dibagi dengan jumlah total penyimpanan sistem file EBS yang dialokasikan untuk semua tugas di cluster.

```
                                       (Total GB of EBS filesystem used by tasks in
 cluster x 100)
Cluster EBS filesystem utilization =
 ------------------------------------------------------------------
                                       (Total GB of EBS filesystem allocated to tasks
 in cluster)
```

Pemanfaatan layanan

Metrik pemanfaatan layanan tersedia untuk CPU, memori, dan, ketika ada volume EBS yang melekat pada tugas Anda, pemanfaatan sistem file EBS. Metrik tingkat layanan didukung untuk layanan dengan tugas yang dihosting di instans Amazon EC2 dan Fargate.

Tingkat layanan CPU dan pemanfaatan memori

Pemanfaatan CPU dan memori diukur sebagai persentase CPU dan memori yang digunakan oleh tugas Amazon ECS yang termasuk dalam layanan pada cluster bila dibandingkan dengan CPU dan memori yang ditentukan dalam definisi tugas layanan.

```
                                        (Total CPU units used by tasks in service) x 100
Service CPU utilization =
 ------------------------------------------------------------------------------
                           (Total CPU units specified in task definition) x (number of
 tasks in service)
```

```
                                         (Total MiB of memory used by tasks in service) x
 100
Service memory utilization =
 ------------------------------------------------------------------------------
                              (Total MiB of memory specified in task definition) x
 (number of tasks in service)
```

Setiap menit, agen penampung Amazon ECS menghitung jumlah unit CPU dan MiB memori yang saat ini digunakan untuk setiap tugas yang dimiliki oleh layanan, dan informasi ini dilaporkan kembali ke Amazon ECS. Jumlah total CPU dan memori yang digunakan untuk semua tugas yang dimiliki oleh layanan yang berjalan di cluster dihitung, dan angka-angka tersebut dilaporkan CloudWatch sebagai persentase dari total sumber daya yang ditentukan untuk layanan dalam definisi tugas layanan. Jika Anda menentukan soft limit (`memoryReservation`), maka ketentuan tersebut digunakan untuk menghitung jumlah memori yang disimpan. Jika tidak, hard limit (`memory`) akan digunakan. Untuk informasi lebih lanjut tentang batas keras dan lunak, lihatUkuran tugas.

Misalnya, ketentuan tugas untuk layanan menentukan total 512 unit CPU dan 1.024 MiB memori (dengan parameter hard limit `memory`) untuk semua kontainernya. Layanan ini memiliki hitungan yang diinginkan dari 1 tugas yang berjalan, layanan tersebut berjalan pada sebuah klaster dengan 1 instans kontainer `c4.large` (dengan 2.048 unit CPU dan 3.768 MiB total memori), dan tidak ada tugas lain yang berjalan di klaster. Meskipun tugas tersebut menentukan 512 unit CPU, karena itu

adalah satu-satunya tugas yang berjalan pada instans kontainer dengan 2,048 unit CPU, ia dapat menggunakan hingga empat kali jumlah yang ditentukan (2,048 / 512). Namun, memori sebesar 1.024 MiB yang telah ditentukan adalah hard limit dan tidak dapat dilampaui, sehingga dalam hal ini pemanfaatan memori layanan tidak dapat melebihi 100%.

Jika contoh sebelumnya menggunakan soft limit `memoryReservation` dan bukan parameter hard limit `memory`, tugas layanan dapat menggunakan lebih dari memori yang telah ditentukan sebesar 1.024 MiB sesuai keperluan. Dalam hal ini, pemanfaatan memori layanan dapat melebihi 100%.

Jika aplikasi Anda memiliki lonjakan mendadak dalam pemanfaatan memori untuk waktu yang singkat, Anda tidak akan melihat pemanfaatan memori layanan meningkat karena Amazon ECS mengumpulkan beberapa titik data setiap menit, dan kemudian menggabungkannya ke satu titik data yang dikirim ke. CloudWatch

Jika tugas ini sedang melakukan pekerjaan dengan penggunaan CPU yang intensif selama periode tertentu dan menggunakan semua 2.048 unit CPU dan 512 MiB memori yang tersedia, maka layanan melaporkan pemanfaatan CPU 400% dan pemanfaatan memori 50%. Jika tugas idle dan menggunakan 128 CPU unit dan 128 MiB memori, maka layanan melaporkan pemanfaatan CPU 25% dan pemanfaatan memori 12.5%.

> **Note**
>
> Dalam contoh ini, pemanfaatan CPU hanya akan naik di atas 100% ketika unit CPU didefinisikan pada tingkat kontainer. Jika Anda menentukan unit CPU pada tingkat tugas, pemanfaatan tidak akan melebihi batas tingkat tugas yang ditentukan.

Pemanfaatan sistem file EBS tingkat layanan

Pemanfaatan sistem file EBS tingkat layanan diukur sebagai jumlah total sistem file EBS yang digunakan oleh tugas-tugas milik layanan, dibagi dengan jumlah total penyimpanan sistem file EBS yang dialokasikan untuk semua tugas yang termasuk dalam layanan.

```
                                    (Total GB of EBS filesystem used by tasks in the
 service x 100)
Service EBS filesystem utilization =
 ------------------------------------------------------------------
                                    (Total GB of EBS filesystem allocated to tasks
 in the service)
```

Jumlah tugas layanan **RUNNING**

Anda dapat menggunakan CloudWatch metrik untuk melihat jumlah tugas di layanan Anda yang berada di `RUNNING` negara bagian. Misalnya, Anda dapat menyetel CloudWatch alarm untuk metrik ini untuk memberi tahu Anda jika jumlah tugas yang berjalan di layanan Anda berada di bawah nilai yang ditentukan.

Jumlah **RUNNING** tugas layanan di Amazon ECS CloudWatch Container Insights

Metrik "Jumlah Tugas yang Menjalankan" (`RunningTaskCount`) tersedia per klaster dan per layanan saat Anda menggunakan Amazon ECS CloudWatch Container Insights. Anda dapat menggunakan Container Insights untuk semua cluster baru yang dibuat dengan memilih masuk ke setelan `containerInsights` akun, pada kluster individual dengan mengaktifkan setelan klaster selama pembuatan klaster, atau pada cluster yang ada dengan menggunakan API. UpdateClusterSettings Metrik yang dikumpulkan oleh CloudWatch Container Insights dikenakan biaya sebagai metrik kustom. Untuk informasi lebih lanjut tentang harga CloudWatch , lihat Harga CloudWatch.

Untuk melihat metrik ini, lihat Metrik Wawasan Kontainer Amazon ECS di Panduan Pengguna Amazon CloudWatch .

AWS Fargate metrik penggunaan

Anda dapat menggunakan metrik CloudWatch penggunaan untuk memberikan visibilitas ke akun Anda penggunaan sumber daya. Gunakan metrik ini untuk memvisualisasikan penggunaan layanan Anda saat ini pada CloudWatch grafik dan dasbor.

AWS Fargate metrik penggunaan sesuai dengan kuota AWS layanan. Anda dapat mengonfigurasi alarm yang memberi tahu Anda saat penggunaan mendekati kuota layanan. Untuk informasi lebih lanjut tentang kuota layanan Fargate, lihat AWS Fargate kuota layanan.

AWS Fargate menerbitkan metrik berikut di namespace. `AWS/Usage`

Metrik	Deskripsi
`ResourceCount`	Jumlah sumber daya tertentu yang berjalan di akun Anda. Sumber daya ditentukan oleh dimensi yang terkait dengan metrik.

Dimensi berikut ini digunakan untuk memilah penggunaan metrik yang dipublikasikan oleh AWS Fargate.

Dimensi	Deskripsi
`Service`	Nama AWS layanan yang berisi sumber daya. Untuk metrik penggunaan AWS Fargate , nilai untuk dimensi ini adalah `Fargate`.
`Type`	Jenis entitas yang dilaporkan. Saat ini, satu-satunya nilai yang valid untuk metrik AWS Fargate penggunaan adalah`Resource`.
`Resource`	Tipe sumber daya yang sedang berjalan. Tipe sumber daya yang sedang berjalan. Saat ini, satu-satunya nilai yang valid untuk metrik AWS Fargate penggunaan adalah `vCPU` yang mengembalikan informasi tentang instance yang sedang berjalan.
`Class`	Kelas sumber daya yang dilacak. Kelas sumber daya yang dilacak. Untuk metrik AWS Fargate penggunaan dengan vCPU sebagai nilai dimensi Resource, nilai `Standard/OnDemand` yang valid adalah dan. `Standard/Spot`

Anda dapat menggunakan konsol Service Quotas untuk memvisualisasikan penggunaan Anda pada grafik dan mengonfigurasi alarm yang mengingatkan Anda saat AWS Fargate penggunaan mendekati kuota layanan. Untuk informasi tentang cara membuat CloudWatch alarm untuk memberi tahu Anda saat mendekati ambang nilai kuota, lihat Service Quotas dan alarm CloudWatch Service Quotas Amazon di Panduan Pengguna

.

Melihat metrik Amazon ECS

Setelah sumber daya berjalan di klaster, Anda dapat melihat metrik di Amazon ECS dan CloudWatch konsol. Konsol Amazon ECS menyediakan tampilan maksimum, minimum, dan rata-rata 24 jam untuk metrik klaster dan layanan Anda. Konsol CloudWatch menyediakan tampilan sumber daya yang detail dan disesuaikan, serta jumlah tugas yang berjalan dalam layanan.

Konsol Amazon ECS

Amazon ECS service CPU dan metrik pemanfaatan memori tersedia di konsol Amazon ECS. Tampilan yang disediakan untuk metrik klaster menunjukkan nilai rata-rata, minimum, dan maksimum dalam periode 24 jam sebelumnya, dengan titik data yang tersedia dalam interval 5 menit. Untuk informasi selengkapnya, lihat Pemanfaatan layanan.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pilih klaster yang metriknya ingin Anda lihat.
3. Tentukan metrik yang akan dilihat.

Toview metrik dari	Langkah-langkah
Klaster	Pada halaman detail cluster, pilih tab Metrik. Ada juga tautan yang disediakan ke CloudWatch konsol untuk melihat metrik CloudWatc h Wawasan Kontainer jika Anda mengaktifkannya.
Layanan	Pada halaman detail cluster, pada tab Layanan, pilih layanan. Metrik kemudian tersedia di tab Kesehatan dan metrik.

CloudWatch konsol

Untuk jenis peluncuran Fargate, metrik layanan Amazon ECS juga dapat dilihat di konsol. CloudWatch Konsol menyediakan tampilan metrik Amazon ECS yang paling detail, dan Anda dapat menyesuaikan tampilan sesuai dengan kebutuhan Anda. Anda dapat melihat pemanfaatan layanan dan jumlah tugas RUNNING layanan.

Untuk jenis peluncuran EC2, kluster Amazon ECS dan metrik layanan juga dapat dilihat di konsol. CloudWatch Konsol menyediakan tampilan metrik Amazon ECS yang paling detail, dan Anda dapat menyesuaikan tampilan sesuai dengan kebutuhan Anda.

Untuk informasi tentang cara melihat metrik, lihat Melihat metrik yang tersedia di CloudWatch Panduan Pengguna Amazon.

Otomatiskan tanggapan terhadap kesalahan Amazon ECS menggunakan EventBridge

Menggunakan Amazon EventBridge, Anda dapat mengotomatiskan AWS layanan Anda dan merespons secara otomatis peristiwa sistem seperti masalah ketersediaan aplikasi atau perubahan sumber daya. Acara dari AWS layanan dikirimkan ke EventBridge dalam waktu dekat. Anda dapat menuliskan aturan sederhana untuk menunjukkan peristiwa mana yang sesuai kepentingan Anda, dan tindakan otomatis yang diambil ketika suatu peristiwa sesuai dengan suatu aturan. Tindakan yang dapat dikonfigurasi secara otomatis untuk menyertakan yang berikut:

- Menambahkan peristiwa ke grup log di CloudWatch Log
- Memanggil fungsi AWS Lambda
- Meminta Perintah Amazon EC2 Run
- Mengirim peristiwa ke Amazon Kinesis Data Streams
- Mengaktifkan mesin AWS Step Functions negara
- Memberitahu topik Amazon SNS atau antrian Amazon Simple Queue Service (Amazon SQS)

Untuk informasi selengkapnya, lihat Memulai Amazon EventBridge di Panduan EventBridge Pengguna Amazon.

Anda dapat menggunakan acara Amazon ECS EventBridge untuk menerima notifikasi mendekati waktu nyata mengenai status klaster Amazon ECS Anda saat ini. Jika tugas Anda menggunakan tipe peluncuran EC2, Anda dapat melihat status instance container dan status saat ini dari semua tugas yang berjalan pada instance container tersebut. Jika tugas Anda menggunakan tipe peluncuran Fargate, Anda dapat melihat status instance container.

Dengan menggunakan EventBridge, Anda dapat membuat penjadwal khusus di atas Amazon ECS yang bertanggung jawab untuk mengatur tugas di seluruh cluster dan memantau status cluster dalam waktu dekat. Anda dapat menghilangkan kode penjadwalan dan pemantauan yang terus-menerus melakukan polling layanan Amazon ECS untuk perubahan status dan sebagai gantinya menangani perubahan status Amazon ECS secara asinkron menggunakan target apa pun. EventBridge Target mungkin termasuk AWS Lambda, Layanan Antrian Sederhana Amazon, Layanan Pemberitahuan Sederhana Amazon, atau Amazon Kinesis Data Streams.

Aliran acara Amazon ECS memastikan bahwa setiap acara dikirimkan setidaknya satu kali. Jika peristiwa duplikat dikirim, peristiwa menyediakan informasi yang cukup untuk mengidentifikasi duplikat. Untuk informasi selengkapnya, lihat Menangani acara Amazon ECS.

Peristiwa relatif berurutan, sehingga Anda dapat dengan mudah memberitahu kapan suatu peristiwa terjadi dalam kaitannya dengan peristiwa lain.

Topik

Acara Amazon ECS

Amazon ECS melacak status setiap tugas dan layanan Anda. Jika status tugas atau layanan berubah, peristiwa dibuat dan dikirim ke Amazon EventBridge. Peristiwa ini diklasifikasikan sebagai peristiwa perubahan status tugas dan peristiwa tindakan layanan. Peristiwa dan kemungkinan penyebabnya dijelaskan secara lebih detail di bagian berikut.

Amazon ECS menghasilkan dan mengirimkan jenis peristiwa berikut ke EventBridge: peristiwa perubahan status instance container, peristiwa perubahan status tugas, tindakan layanan, dan peristiwa perubahan status penerapan layanan.

- Perubahan status instance kontainer
- Perubahan status tugas
- Perubahan status penerapan
- Tindakan layanan

> **Note**
>
> Amazon ECS dapat menambahkan jenis, sumber, dan detail acara lainnya di masa mendatang. Jika Anda melakukan de-serialisasi data JSON peristiwa dalam kode, pastikan aplikasi Anda siap menangani properti yang tidak dikenal untuk menghindari masalah jika dan kapan properti tambahan ini ditambahkan.

Dalam beberapa kasus, beberapa peristiwa dihasilkan untuk aktivitas yang sama. Misalnya, ketika tugas dimulai pada instance kontainer, peristiwa perubahan status tugas dihasilkan untuk tugas baru.

Peristiwa perubahan status instans kontainer dihasilkan untuk memperhitungkan perubahan sumber daya yang tersedia, seperti CPU, memori, dan port yang tersedia, pada instance kontainer. Demikian juga, jika instance kontainer dihentikan, peristiwa dihasilkan untuk instance container, status koneksi agen kontainer, dan setiap tugas yang berjalan pada instance container.

Peristiwa perubahan status kontainer dan perubahan status tugas berisi dua bidang `version`: satu di bagian utama peristiwa, dan satu di objek `detail` peristiwa. Berikut ini menjelaskan perbedaan antara dua bidang tersebut:

- Bidang `version` dalam bagian utama peristiwa diatur ke `0` pada semua peristiwa. Untuk informasi selengkapnya tentang EventBridge parameter, lihat Peristiwa dan Pola Peristiwa di Panduan EventBridge Pengguna Amazon.
- Bidang `version` dalam objek `detail` peristiwa menggambarkan versi sumber daya terkait. Setiap kali sumber daya berubah status, versi ini bertambah. Karena peristiwa dapat dikirim beberapa kali, bidang ini mengizinkan Anda untuk mengidentifikasi peristiwa duplikat. Peristiwa duplikat memiliki versi yang sama di objek `detail`. Jika Anda mereplikasi instans penampung Amazon ECS dan status tugas dengan EventBridge, Anda dapat membandingkan versi sumber daya yang dilaporkan oleh Amazon ECS API dengan versi yang dilaporkan EventBridge untuk sumber daya (di dalam `detail` objek) untuk memverifikasi bahwa versi dalam aliran peristiwa Anda saat ini.

Peristiwa tindakan layanan hanya berisi bidang `version` dalam bagian utama.

Untuk informasi tambahan tentang cara mengintegrasikan Amazon ECS dan EventBridge, lihat Mengintegrasikan Amazon dan EventBridge Amazon ECS.

Acara perubahan status instans penampung Amazon ECS

Skenario berikut menyebabkan peristiwa perubahan status instance container:

Anda memanggil operasi API `StartTask`, `RunTask`, atau `StopTask`, secara langsung maupun dengan AWS Management Console atau SDK.

> Menempatkan atau menghentikan tugas pada instans kontainer memodifikasi sumber daya yang tersedia pada instans kontainer, seperti CPU, memori, dan port yang tersedia.

Penjadwal layanan Amazon ECS memulai atau menghentikan tugas.

> Menempatkan atau menghentikan tugas pada instans kontainer memodifikasi sumber daya yang tersedia pada instans kontainer, seperti CPU, memori, dan port yang tersedia.

Agen penampung Amazon ECS memanggil operasi `SubmitTaskStateChange` API dengan `STOPPED` status untuk tugas dengan status yang diinginkan. `RUNNING`

Agen penampung Amazon ECS memantau status tugas pada instans penampung Anda, dan ia melaporkan perubahan status apa pun. Jika tugas yang seharusnya `RUNNING` dialihkan ke `STOPPED`, maka agen merilis sumber daya yang dialokasikan ke tugas yang berhenti, seperti CPU, memori, dan port yang tersedia.

Anda membatalkan pendaftaran instance container dengan operasi `DeregisterContainerInstance` API, baik secara langsung maupun dengan atau SDK AWS Management Console .

Membatalkan pendaftaran instance container mengubah status instance container dan status koneksi agen container Amazon ECS.

Tugas dihentikan ketika instans EC2 berhenti.

Ketika Anda menghentikan instans kontainer, tugas yang berjalan di dalamnya dialihkan ke status `STOPPED`.

Agen kontainer Amazon ECS mendaftarkan instance kontainer untuk pertama kalinya.

Pertama kali agen penampung Amazon ECS mendaftarkan instance kontainer (saat peluncuran atau saat pertama kali dijalankan secara manual), ini membuat peristiwa perubahan status untuk instance tersebut.

Agen kontainer Amazon ECS menghubungkan atau memutuskan sambungan dari Amazon ECS.

Saat agen penampung Amazon ECS menghubungkan atau memutuskan sambungan dari backend Amazon ECS, agen penampung akan mengubah `agentConnected` status instans penampung.

> **Note**
>
> Agen kontainer Amazon ECS terputus dan menyambung kembali beberapa kali per jam sebagai bagian dari operasi normalnya, sehingga peristiwa koneksi agen harus diharapkan. Peristiwa ini bukan merupakan indikasi bahwa terdapat masalah dengan agen kontainer atau instans kontainer Anda.

Anda memutakhirkan agen penampung Amazon ECS pada sebuah instans.

> Detail instans kontainer berisi objek untuk versi agen kontainer. Jika Anda memutakhirkan agen, informasi versi ini berubah dan menghasilkan acara.

Example Peristiwa perubahan status instans kontainer

Peristiwa perubahan status instans kontainer disampaikan dalam format berikut. `detail`Bagian di bawah ini menyerupai ContainerInstanceobjek yang dikembalikan dari operasi DescribeContainerInstancesAPI di Referensi API Amazon Elastic Container Service. Untuk informasi selengkapnya tentang EventBridge parameter, lihat Peristiwa dan Pola Peristiwa di Panduan EventBridge Pengguna Amazon.

```
{
  "version": "0",
  "id": "8952ba83-7be2-4ab5-9c32-6687532d15a2",
  "detail-type": "ECS Container Instance State Change",
  "source": "aws.ecs",
  "account": "111122223333",
  "time": "2016-12-06T16:41:06Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ecs:us-east-1:111122223333:container-instance/
b54a2a04-046f-4331-9d74-3f6d7f6ca315"
  ],
  "detail": {
    "agentConnected": true,
    "attributes": [
      {
        "name": "com.amazonaws.ecs.capability.logging-driver.syslog"
      },
      {
        "name": "com.amazonaws.ecs.capability.task-iam-role-network-host"
      },
      {
        "name": "com.amazonaws.ecs.capability.logging-driver.awslogs"
      },
      {
        "name": "com.amazonaws.ecs.capability.logging-driver.json-file"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.17"
```

```
      },
      {
        "name": "com.amazonaws.ecs.capability.privileged-container"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.18"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.19"
      },
      {
        "name": "com.amazonaws.ecs.capability.ecr-auth"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.20"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.21"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.22"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.23"
      },
      {
        "name": "com.amazonaws.ecs.capability.task-iam-role"
      }
    ],
    "clusterArn": "arn:aws:ecs:us-east-1:111122223333:cluster/default",
    "containerInstanceArn": "arn:aws:ecs:us-east-1:111122223333:container-instance/
b54a2a04-046f-4331-9d74-3f6d7f6ca315",
    "ec2InstanceId": "i-f3a8506b",
    "registeredResources": [
      {
        "name": "CPU",
        "type": "INTEGER",
        "integerValue": 2048
      },
      {
        "name": "MEMORY",
        "type": "INTEGER",
        "integerValue": 3767
      },
```

```
      {
        "name": "PORTS",
        "type": "STRINGSET",
        "stringSetValue": [
          "22",
          "2376",
          "2375",
          "51678",
          "51679"
        ]
      },
      {
        "name": "PORTS_UDP",
        "type": "STRINGSET",
        "stringSetValue": []
      }
    ],
    "remainingResources": [
      {
        "name": "CPU",
        "type": "INTEGER",
        "integerValue": 1988
      },
      {
        "name": "MEMORY",
        "type": "INTEGER",
        "integerValue": 767
      },
      {
        "name": "PORTS",
        "type": "STRINGSET",
        "stringSetValue": [
          "22",
          "2376",
          "2375",
          "51678",
          "51679"
        ]
      },
      {
        "name": "PORTS_UDP",
        "type": "STRINGSET",
        "stringSetValue": []
      }
```

```
    ],
    "status": "ACTIVE",
    "version": 14801,
    "versionInfo": {
      "agentHash": "aebcbca",
      "agentVersion": "1.13.0",
      "dockerVersion": "DockerVersion: 1.11.2"
    },
    "updatedAt": "2016-12-06T16:41:06.991Z"
  }
}
```

Acara perubahan status tugas Amazon ECS

Skenario berikut menyebabkan peristiwa perubahan status tugas:

Anda memanggil operasi API `StartTask`, `RunTask`, atau `StopTask`, baik secara langsung atau dengan AWS Management Console, AWS CLI, atau SDK.

> Memulai atau menghentikan tugas membuat sumber daya tugas baru atau mengubah status sumber daya tugas yang ada.

Penjadwal layanan Amazon ECS memulai atau menghentikan tugas.

> Memulai atau menghentikan tugas membuat sumber daya tugas baru atau mengubah status sumber daya tugas yang ada.

Agen kontainer Amazon ECS memanggil operasi `SubmitTaskStateChange` API.

> Untuk jenis peluncuran Fargate, agen kontainer Amazon ECS memantau status instans kontainer Anda. Untuk jenis peluncuran Amazon EC2, agen kontainer Amazon ECS memantau status tugas Anda pada instans penampung Anda. Agen kontainer Amazon ECS melaporkan perubahan status apa pun. Perubahan status dapat mencakup perubahan dari `PENDING` ke `RUNNING` atau dari `RUNNING` ke `STOPPED`.

Anda memaksa deregistrasi instance container yang mendasarinya dengan operasi `DeregisterContainerInstance` API dan `force` flag, baik secara langsung maupun dengan atau SDK. AWS Management Console

> Membatalkan pendaftaran instance container mengubah status instance container dan status koneksi agen container Amazon ECS. Jika tugas berjalan di instans kontainer, bendera `force`

harus diatur untuk mengizinkan pembatalan pendaftaran. Hal ini menghentikan semua tugas pada instans.

Instans kontainer yang mendasarinya dihentikan atau diakhiri.

Saat Anda menghentikan instans kontainer, maka tugas yang berjalan di dalamnya dialihkan ke status `STOPPED`.

Kontainer dalam tugas mengubah status.

Agen kontainer Amazon ECS memantau keadaan kontainer dalam tugas. Misalnya, jika kontainer yang berjalan dalam tugas berhenti, perubahan status kontainer ini menghasilkan peristiwa.

Tugas yang menggunakan penyedia kapasitas Fargate Spot menerima pemberitahuan penghentian.

Saat tugas menggunakan penyedia `FARGATE_SPOT` kapasitas dan dihentikan karena gangguan Spot, peristiwa perubahan status tugas akan dihasilkan.

Example Peristiwa perubahan status tugas

Peristiwa perubahan status tugas disampaikan dalam format berikut. `detail`Bagian di bawah ini menyerupai objek Task yang dikembalikan dari operasi DescribeTasksAPI di Referensi API Amazon Elastic Container Service. Jika kontainer Anda menggunakan gambar yang dihosting dengan Amazon ECR, `imageDigest` bidang akan dikembalikan.

> **Note**
>
> Nilai untuk`createdAt`,,,`connectivityAt`, `pullStartedAt` `startedAt``pullStoppedAt`, dan `updatedAt` bidang adalah stempel waktu UNIX dalam respons `DescribeTasks` tindakan sedangkan dalam peristiwa perubahan status tugas mereka adalah stempel waktu string ISO.

Untuk informasi selengkapnya tentang parameter CloudWatch Peristiwa, lihat Peristiwa dan Pola Peristiwa di Panduan EventBridge Pengguna Amazon.

Untuk informasi tentang cara mengonfigurasi aturan EventBridge peristiwa Amazon yang hanya menangkap peristiwa tugas di mana tugas telah berhenti berjalan karena salah satu container pentingnya telah dihentikan, lihat Mengirim peringatan Amazon Simple Notification Service untuk acara yang dihentikan tugas

```
{
```

```
    "version": "0",
    "id": "3317b2af-7005-947d-b652-f55e762e571a",
    "detail-type": "ECS Task State Change",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2020-01-23T17:57:58Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:task/FargateCluster/
c13b4cb40f1f4fe4a2971f76ae5a47ad"
    ],
    "detail": {
        "attachments": [
            {
                "id": "1789bcae-ddfb-4d10-8ebe-8ac87ddba5b8",
                "type": "eni",
                "status": "ATTACHED",
                "details": [
                    {
                        "name": "subnetId",
                        "value": "subnet-abcd1234"
                    },
                    {
                        "name": "networkInterfaceId",
                        "value": "eni-abcd1234"
                    },
                    {
                        "name": "macAddress",
                        "value": "0a:98:eb:a7:29:ba"
                    },
                    {
                        "name": "privateIPv4Address",
                        "value": "10.0.0.139"
                    }
                ]
            }
        ],
        "availabilityZone": "us-west-2c",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/FargateCluster",
        "containers": [
            {
                "containerArn": "arn:aws:ecs:us-west-2:111122223333:container/
cf159fd6-3e3f-4a9e-84f9-66cbe726af01",
                "lastStatus": "RUNNING",
```

```
                "name": "FargateApp",
                "image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/hello-
repository:latest",
                "imageDigest":
 "sha256:74b2c688c700ec95a93e478cdb959737c148df3fbf5ea706abe0318726e885e6",
                "runtimeId":
 "ad64cbc71c7fb31c55507ec24c9f77947132b03d48d9961115cf24f3b7307e1e",
                "taskArn": "arn:aws:ecs:us-west-2:111122223333:task/FargateCluster/
c13b4cb40f1f4fe4a2971f76ae5a47ad",
                "networkInterfaces": [
                    {
                        "attachmentId": "1789bcae-ddfb-4d10-8ebe-8ac87ddba5b8",
                        "privateIpv4Address": "10.0.0.139"
                    }
                ],
                "cpu": "0"
            }
        ],
        "createdAt": "2020-01-23T17:57:34.402Z",
        "launchType": "FARGATE",
        "cpu": "256",
        "memory": "512",
        "desiredStatus": "RUNNING",
        "group": "family:sample-fargate",
        "lastStatus": "RUNNING",
        "overrides": {
            "containerOverrides": [
                {
                    "name": "FargateApp"
                }
            ]
        },
        "connectivity": "CONNECTED",
        "connectivityAt": "2020-01-23T17:57:38.453Z",
        "pullStartedAt": "2020-01-23T17:57:52.103Z",
        "startedAt": "2020-01-23T17:57:58.103Z",
        "pullStoppedAt": "2020-01-23T17:57:55.103Z",
        "updatedAt": "2020-01-23T17:57:58.103Z",
        "taskArn": "arn:aws:ecs:us-west-2:111122223333:task/FargateCluster/
c13b4cb40f1f4fe4a2971f76ae5a47ad",
        "taskDefinitionArn": "arn:aws:ecs:us-west-2:111122223333:task-definition/
sample-fargate:1",
        "version": 4,
        "platformVersion": "1.3.0"
```

```
    }
}
```

Acara tindakan layanan Amazon ECS

Amazon ECS mengirimkan peristiwa tindakan layanan dengan jenis detail ECS Service Action. Tidak seperti peristiwa perubahan status instans kontainer dan tugas, peristiwa tindakan layanan tidak mencakup nomor versi di bidang tanggapan `details`. Berikut ini adalah pola peristiwa yang digunakan untuk membuat EventBridge aturan untuk peristiwa tindakan layanan Amazon ECS. Untuk informasi selengkapnya, lihat Membuat EventBridge Aturan di Panduan EventBridge Pengguna Amazon.

```
{
    "source": [
        "aws.ecs"
    ],
    "detail-type": [
        "ECS Service Action"
    ]
}
```

Amazon ECS mengirimkan acara dengan`INFO`,`WARN`, dan jenis `ERROR` acara. Berikut ini adalah peristiwa tindakan layanan.

Peristiwa tindakan layanan dengan tipe peristiwa **`INFO`**

`SERVICE_STEADY_STATE`

Kondisi layanan ini baik dan pada jumlah tugas yang diinginkan, sehingga mencapai status stabil. Penjadwal layanan melaporkan status secara berkala, sehingga Anda mungkin menerima pesan ini beberapa kali.

`TASKSET_STEADY_STATE`

Tugas yang ditetapkan dalam kondisi baik dan pada jumlah tugas yang diinginkan, sehingga mencapai status stabil.

`CAPACITY_PROVIDER_STEADY_STATE`

Penyedia kapasitas yang terkait dengan layanan mencapai status stabil.

`SERVICE_DESIRED_COUNT_UPDATED`

Ketika penjadwal layanan memperbarui jumlah hitungan yang diinginkan untuk set layanan atau tugas. Peristiwa ini tidak dikirim ketika jumlah yang diinginkan diperbarui secara manual oleh pengguna.

Peristiwa tindakan layanan dengan tipe peristiwa **`WARN`**

`SERVICE_TASK_START_IMPAIRED`

Layanan ini tidak dapat berhasil memulai tugas secara konsisten.

`SERVICE_DISCOVERY_INSTANCE_UNHEALTHY`

Layanan yang menggunakan penemuan layanan berisi tugas yang kondisinya tidak baik. Penjadwal layanan mendeteksi bahwa kondisi tugas dalam registri layanan tidak baik.

Peristiwa tindakan layanan dengan tipe peristiwa **`ERROR`**

`SERVICE_DAEMON_PLACEMENT_CONSTRAINT_VIOLATED`

Tugas dalam layanan yang menggunakan strategi penjadwal layanan `DAEMON` tidak lagi memenuhi strategi kendala penempatan untuk layanan.

`ECS_OPERATION_THROTTLED`

Penjadwal layanan telah dibatasi karena batas throttle Amazon ECS API.

`SERVICE_DISCOVERY_OPERATION_THROTTLED`

Penjadwal layanan telah dibatasi karena batas throttle AWS Cloud Map API. Hal ini dapat terjadi pada layanan yang dikonfigurasi untuk menggunakan penemuan layanan.

`SERVICE_TASK_PLACEMENT_FAILURE`

Penjadwal layanan tidak dapat menempatkan tugas. Penyebabnya akan dijelaskan dalam bidang `reason`.

Penyebab umum untuk acara layanan ini dihasilkan adalah karena kurangnya sumber daya di cluster untuk menempatkan tugas. Misalnya, kurangnya kapasitas CPU atau memori pada instans kontainer yang tersedia atau tidak tersedianya instans kontainer. Penyebab umum lainnya adalah ketika agen penampung Amazon ECS terputus pada instance penampung, menyebabkan penjadwal tidak dapat menempatkan tugas.

`SERVICE_TASK_CONFIGURATION_FAILURE`

Penjadwal layanan tidak dapat menempatkan tugas karena kesalahan konfigurasi. Penyebabnya akan dijelaskan dalam bidang `reason`.

Penyebab umum peristiwa layanan ini dihasilkan adalah karena tag diterapkan ke layanan tetapi pengguna atau peran belum memilih format Amazon Resource Name (ARN) baru di Wilayah. Untuk informasi selengkapnya, lihat Amazon Resource Name (ARN) dan ID. Penyebab umum lainnya adalah Amazon ECS tidak dapat mengambil peran tugas IAM yang disediakan.

Example Peristiwa status stabil layanan

Peristiwa status stabil layanan disampaikan dalam format berikut. Untuk informasi selengkapnya tentang EventBridge parameter, lihat Peristiwa dan Pola Peristiwa di Panduan EventBridge Pengguna Amazon.

```
{
    "version": "0",
    "id": "af3c496d-f4a8-65d1-70f4-a69d52e9b584",
    "detail-type": "ECS Service Action",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2019-11-19T19:27:22Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
        "eventType": "INFO",
        "eventName": "SERVICE_STEADY_STATE",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
        "createdAt": "2019-11-19T19:27:22.695Z"
    }
}
```

Example Peristiwa status stabil penyedia kapasitas

Peristiwa status stabil penyedia kapasitas disampaikan dalam format berikut.

```
{
    "version": "0",
```

```
    "id": "b9baa007-2f33-0eb1-5760-0d02a572d81f",
    "detail-type": "ECS Service Action",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2019-11-19T19:37:00Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
        "eventType": "INFO",
        "eventName": "CAPACITY_PROVIDER_STEADY_STATE",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
        "capacityProviderArns": [
            "arn:aws:ecs:us-west-2:111122223333:capacity-provider/ASG-tutorial-
capacity-provider"
        ],
        "createdAt": "2019-11-19T19:37:00.807Z"
    }
}
```

Example Peristiwa gangguan memulai tugas layanan

Peristiwa gangguan memulai tugas layanan disampaikan dalam format berikut.

```
{
    "version": "0",
    "id": "57c9506e-9d21-294c-d2fe-e8738da7e67d",
    "detail-type": "ECS Service Action",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2019-11-19T19:55:38Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
        "eventType": "WARN",
        "eventName": "SERVICE_TASK_START_IMPAIRED",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
        "createdAt": "2019-11-19T19:55:38.725Z"
    }
}
```

Example Peristiwa kegagalan penempatan tugas layanan

Peristiwa kegagalan penempatan tugas disampaikan dalam format berikut. Untuk informasi selengkapnya tentang EventBridge parameter, lihat Peristiwa dan Pola Peristiwa di Panduan EventBridge Pengguna Amazon.

Dalam contoh berikut, tugas mencoba untuk menggunakan penyedia kapasitas `FARGATE_SPOT` tetapi penjadwal layanan tidak dapat memperoleh kapasitas Fargate Spot.

```
{
    "version": "0",
    "id": "ddca6449-b258-46c0-8653-e0e3a6d0468b",
    "detail-type": "ECS Service Action",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2019-11-19T19:55:38Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
        "eventType": "ERROR",
        "eventName": "SERVICE_TASK_PLACEMENT_FAILURE",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
        "capacityProviderArns": [
            "arn:aws:ecs:us-west-2:111122223333:capacity-provider/FARGATE_SPOT"
        ],
        "reason": "RESOURCE:FARGATE",
        "createdAt": "2019-11-06T19:09:33.087Z"
    }
}
```

Dalam contoh berikut untuk tipe peluncuran EC2, tugas tersebut dicoba untuk diluncurkan pada Instance Container `2dd1b186f39845a584488d2ef155c131` tetapi penjadwal layanan tidak dapat menempatkan tugas karena CPU tidak mencukupi.

```
{
  "version": "0",
  "id": "ddca6449-b258-46c0-8653-e0e3a6d0468b",
  "detail-type": "ECS Service Action",
  "source": "aws.ecs",
  "account": "111122223333",
```

```
  "time": "2019-11-19T19:55:38Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
  ],
  "detail": {
    "eventType": "ERROR",
    "eventName": "SERVICE_TASK_PLACEMENT_FAILURE",
    "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
    "containerInstanceArns": [
    "arn:aws:ecs:us-west-2:111122223333:container-instance/
default/2dd1b186f39845a584488d2ef155c131"
    ],
    "reason": "RESOURCE:CPU",
    "createdAt": "2019-11-06T19:09:33.087Z"
  }
}
```

Acara perubahan status penerapan layanan Amazon ECS

Amazon ECS mengirimkan peristiwa perubahan status penerapan layanan dengan jenis detail Perubahan Status Penerapan ECS. Berikut ini adalah pola peristiwa yang digunakan untuk membuat EventBridge aturan untuk peristiwa perubahan status penerapan layanan Amazon ECS. Untuk informasi selengkapnya, lihat Membuat EventBridge Aturan di Panduan EventBridge Pengguna Amazon.

```
{
    "source": [
        "aws.ecs"
    ],
    "detail-type": [
        "ECS Deployment State Change"
    ]
}
```

Amazon ECS mengirimkan acara dengan `INFO` dan jenis `ERROR` acara. Berikut adalah peristiwa perubahan status deployment layanan.

`SERVICE_DEPLOYMENT_IN_PROGRESS`

Deployment layanan sedang berlangsung. Peristiwa ini dikirim untuk deployment awal dan deployment rollback.

`SERVICE_DEPLOYMENT_COMPLETED`

Deployment layanan telah selesai. Peristiwa ini dikirim setelah layanan mencapai status stabil setelah deployment.

`SERVICE_DEPLOYMENT_FAILED`

Deployment layanan gagal. Acara ini dikirim untuk layanan dengan logika pemutus sirkuit penyebaran diaktifkan.

Example Peristiwa deployment layanan sedang berlangsung

Peristiwa deployment layanan yang sedang berlangsung disampaikan ketika deployment awal dan rollback dimulai. Perbedaan antara keduanya ada di bidang `reason`. Untuk informasi selengkapnya tentang EventBridge parameter, lihat Peristiwa dan Pola Peristiwa di Panduan EventBridge Pengguna Amazon.

Berikut ini menampilkan contoh output untuk permulaan deployment awal.

```
{
   "version": "0",
   "id": "ddca6449-b258-46c0-8653-e0e3a6EXAMPLE",
   "detail-type": "ECS Deployment State Change",
   "source": "aws.ecs",
   "account": "111122223333",
   "time": "2020-05-23T12:31:14Z",
   "region": "us-west-2",
   "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
   ],
   "detail": {
        "eventType": "INFO",
        "eventName": "SERVICE_DEPLOYMENT_IN_PROGRESS",
        "deploymentId": "ecs-svc/123",
        "updatedAt": "2020-05-23T11:11:11Z",
        "reason": "ECS deployment deploymentId in progress."
   }
}
```

Berikut ini menampilkan contoh output untuk permulaan deployment rollback. Bidang `reason` menyediakan ID deployment yang digunakan untuk mematalkan layanan.

```
{
   "version": "0",
   "id": "ddca6449-b258-46c0-8653-e0e3aEXAMPLE",
   "detail-type": "ECS Deployment State Change",
   "source": "aws.ecs",
   "account": "111122223333",
   "time": "2020-05-23T12:31:14Z",
   "region": "us-west-2",
   "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
   ],
   "detail": {
        "eventType": "INFO",
        "eventName": "SERVICE_DEPLOYMENT_IN_PROGRESS",
        "deploymentId": "ecs-svc/123",
        "updatedAt": "2020-05-23T11:11:11Z",
        "reason": "ECS deployment circuit breaker: rolling back to
 deploymentId deploymentID."
   }
}
```

Example Peristiwa deployment layanan yang telah selesai

Peristiwa status deployment layanan yang telah selesai disampaikan dalam format berikut. Untuk informasi selengkapnya, lihat Pembaruan bergulir.

```
{
   "version": "0",
   "id": "ddca6449-b258-46c0-8653-e0e3aEXAMPLE",
   "detail-type": "ECS Deployment State Change",
   "source": "aws.ecs",
   "account": "111122223333",
   "time": "2020-05-23T12:31:14Z",
   "region": "us-west-2",
   "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
   ],
   "detail": {
        "eventType": "INFO",
        "eventName": "SERVICE_DEPLOYMENT_COMPLETED",
        "deploymentId": "ecs-svc/123",
        "updatedAt": "2020-05-23T11:11:11Z",
        "reason": "ECS deployment deploymentID completed."
```

```
    }
}
```

Example Peristiwa deployment layanan yang gagal

Peristiwa deployment layanan yang gagal disampaikan dalam format berikut. Peristiwa status gagal penerapan layanan hanya akan dikirim untuk layanan yang mengaktifkan logika pemutus sirkuit penerapan. Untuk informasi selengkapnya, lihat Pembaruan bergulir.

```
{
    "version": "0",
    "id": "ddca6449-b258-46c0-8653-e0e3aEXAMPLE",
    "detail-type": "ECS Deployment State Change",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2020-05-23T12:31:14Z",
    "region": "us-west-2",
    "resources": [
         "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
         "eventType": "ERROR",
         "eventName": "SERVICE_DEPLOYMENT_FAILED",
         "deploymentId": "ecs-svc/123",
         "updatedAt": "2020-05-23T11:11:11Z",
         "reason": "ECS deployment circuit breaker: task failed to start."
    }
}
```

Menangani acara Amazon ECS

Amazon ECS mengirimkan acara setidaknya sekali. Ini berarti Anda mungkin menerima banyak salinan dari acara tertentu. Selain itu, peristiwa mungkin tidak dikirimkan ke listener peristiwa Anda sesuai urutan peristiwa yang terjadi.

Untuk memesan acara dengan benar, `detail` bagian dari setiap acara berisi `version` properti. Setiap kali sumber daya mengubah status, `version` ini bertambah. Peristiwa duplikat memiliki `version` yang sama di objek `detail`. Jika Anda mereplikasi instans penampung Amazon ECS dan status tugas dengan EventBridge, Anda dapat membandingkan versi sumber daya yang dilaporkan oleh Amazon ECS API dengan sumber daya yang `version` dilaporkan dalam EventBridge untuk memverifikasi bahwa versi dalam aliran peristiwa Anda saat ini. Peristiwa dengan jumlah properti

versi yang lebih tinggi harus diperlakukan sebagai peristiwa yang terjadi setelah peristiwa dengan jumlah versi yang lebih rendah.

Contoh: Menangani peristiwa di fungsi AWS Lambda

Contoh berikut menunjukkan fungsi Lambda yang ditulis dengan Python 3.9 yang menangkap peristiwa perubahan status instance tugas dan kontainer dan menyimpannya ke salah satu dari dua tabel Amazon DynamoDB:

- ECS CtrInstanceState - Menyimpan status terbaru untuk instance kontainer. ID tabel adalah nilai `containerInstanceArn` instans kontainer.
- ECS TaskState — Menyimpan status terbaru untuk suatu tugas. ID tabel adalah nilai `taskArn` tugas.

```
import json
import boto3

def lambda_handler(event, context):
    id_name = ""
    new_record = {}

    # For debugging so you can see raw event format.
    print('Here is the event:')
    print((json.dumps(event)))

    if event["source"] != "aws.ecs":
       raise ValueError("Function only supports input from events with a source type
 of: aws.ecs")

    # Switch on task/container events.
    table_name = ""
    if event["detail-type"] == "ECS Task State Change":
        table_name = "ECSTaskState"
        id_name = "taskArn"
        event_id = event["detail"]["taskArn"]
    elif event["detail-type"] == "ECS Container Instance State Change":
        table_name = "ECSCtrInstanceState"
        id_name =  "containerInstanceArn"
        event_id = event["detail"]["containerInstanceArn"]
    else:
```

```
        raise ValueError("detail-type for event is not a supported type. Exiting
without saving event.")

    new_record["cw_version"] = event["version"]
    new_record.update(event["detail"])

    # "status" is a reserved word in DDB, but it appears in containerPort
    # state change messages.
    if "status" in event:
        new_record["current_status"] = event["status"]
        new_record.pop("status")


    # Look first to see if you have received a newer version of an event ID.
    # If the version is OLDER than what you have on file, do not process it.
    # Otherwise, update the associated record with this latest information.
    print("Looking for recent event with same ID...")
    dynamodb = boto3.resource("dynamodb", region_name="us-east-1")
    table = dynamodb.Table(table_name)
    saved_event = table.get_item(
        Key={
            id_name : event_id
        }
    )
    if "Item" in saved_event:
        # Compare events and reconcile.
        print(("EXISTING EVENT DETECTED: Id " + event_id + " - reconciling"))
        if saved_event["Item"]["version"] < event["detail"]["version"]:
            print("Received event is a more recent version than the stored event -
updating")
            table.put_item(
                Item=new_record
            )
        else:
            print("Received event is an older version than the stored event -
ignoring")
    else:
        print(("Saving new event - ID " + event_id))

        table.put_item(
            Item=new_record
        )
```

Contoh Fargate berikut menunjukkan fungsi Lambda yang ditulis dengan Python 3.9 yang menangkap peristiwa perubahan status tugas dan menyimpannya ke tabel Amazon DynamoDB berikut:

```
import json
import boto3

def lambda_handler(event, context):
    id_name = ""
    new_record = {}

    # For debugging so you can see raw event format.
    print('Here is the event:')
    print((json.dumps(event)))

    if event["source"] != "aws.ecs":
       raise ValueError("Function only supports input from events with a source type
 of: aws.ecs")

    # Switch on task/container events.
    table_name = ""
    if event["detail-type"] == "ECS Task State Change":
        table_name = "ECSTaskState"
        id_name = "taskArn"
        event_id = event["detail"]["taskArn"]
    else:
        raise ValueError("detail-type for event is not a supported type. Exiting
 without saving event.")

    new_record["cw_version"] = event["version"]
    new_record.update(event["detail"])

    # "status" is a reserved word in DDB, but it appears in containerPort
    # state change messages.
    if "status" in event:
        new_record["current_status"] = event["status"]
        new_record.pop("status")


    # Look first to see if you have received a newer version of an event ID.
    # If the version is OLDER than what you have on file, do not process it.
    # Otherwise, update the associated record with this latest information.
    print("Looking for recent event with same ID...")
```

```
    dynamodb = boto3.resource("dynamodb", region_name="us-east-1")
    table = dynamodb.Table(table_name)
    saved_event = table.get_item(
        Key={
            id_name : event_id
        }
    )
    if "Item" in saved_event:
        # Compare events and reconcile.
        print(("EXISTING EVENT DETECTED: Id " + event_id + " - reconciling"))
        if saved_event["Item"]["version"] < event["detail"]["version"]:
            print("Received event is a more recent version than the stored event -
updating")
            table.put_item(
                Item=new_record
            )
        else:
            print("Received event is an older version than the stored event -
ignoring")
    else:
        print(("Saving new event - ID " + event_id))

        table.put_item(
            Item=new_record
        )
```

Pantau kontainer Amazon ECS menggunakan Wawasan Kontainer

CloudWatch Container Insights mengumpulkan, mengumpulkan, dan merangkum metrik dan log dari aplikasi dan layanan mikro dalam kontainer Anda.

Container Insights menggunakan CloudWatch agen versi kontainer untuk menemukan semua kontainer yang sedang berjalan dalam klaster dan mengumpulkan data kinerja di setiap lapisan tumpukan kinerja. Data operasional dikumpulkan sebagai log acara performa. Ini adalah entri yang menggunakan skema JSON terstruktur untuk data kardinalitas tinggi untuk dicerna dan disimpan dalam skala besar. Dari data ini, CloudWatch buat metrik agregat tingkat yang lebih tinggi di tingkat klaster, layanan, dan tugas sebagai metrik. CloudWatch Metrik tersebut mencakup pemanfaatan sumber daya seperti CPU, memori, disk, dan jaringan. Metrik tersebut tersedia di dasbor otomatis CloudWatch. Untuk informasi tentang metrik yang tersedia, lihat metrik Amazon ECS Container Insights di Panduan Pengguna Amazon. CloudWatch

> **Important**
>
> Metrik yang dikumpulkan oleh CloudWatch Container Insights dikenakan biaya sebagai metrik kustom. Untuk informasi selengkapnya tentang CloudWatch harga, lihat CloudWatchHarga. Amazon ECS juga menyediakan metrik pemantauan yang disediakan tanpa biaya tambahan. Untuk informasi selengkapnya, lihat Pantau Amazon ECS menggunakan CloudWatch .

Pertimbangan

Berikut ini harus dipertimbangkan saat menggunakan CloudWatch Wawasan Kontainer.

- CloudWatch Metrik Container Insights hanya mencerminkan sumber daya dengan menjalankan tugas selama rentang waktu yang ditentukan. Misalnya, jika Anda memiliki klaster dengan satu layanan di dalamnya namun layanan tersebut tidak memiliki tugas dengan status `RUNNING`, maka tidak akan ada metrik yang dikirimkan ke CloudWatch. Jika Anda memiliki dua layanan dan salah satunya memiliki tugas yang sedang berjalan sementara yang satunya tidak, maka hanya metrik untuk layanan dengan tugas yang sedang berjalan lah yang akan dikirim.
- Metrik jaringan tersedia untuk semua tugas yang dijalankan di Fargate dan tugas yang dijalankan di instans Amazon EC2 yang menggunakan mode atau jaringan. `bridge` `awsvpc`

Anda dapat melihat tugas Amazon ECS dan peristiwa siklus hidup layanan dalam konsol CloudWatch Container Insights. Hal ini akan membantu Anda dalam mengkorelasikan metrik-metrik kontainer, log, dan peristiwa dalam satu tampilan untuk memberikan Anda visibilitas operasional yang lebih lengkap.

Acara yang dapat Anda lihat adalah peristiwa yang dikirimkan Amazon ECS ke Amazon EventBridge. Untuk informasi selengkapnya, lihat acara Amazon ECS.

Anda dapat memilih untuk mengonfigurasi metrik kinerja untuk cluster, tugas, atau layanan. Bergantung pada sumber daya yang Anda pilih, peristiwa berikut dilaporkan:

- Peristiwa perubahan status instans kontainer
- Peristiwa tindakan layanan
- Peristiwa perubahan status tugas

Mengkonfigurasi CloudWatch Wawasan Kontainer untuk Amazon ECS

Anda dapat mengonfigurasi Wawasan Kontainer menggunakan konsol Amazon ECS, API AWS CLI, dan SDK.

Gunakan tabel berikut untuk menentukan tindakan yang akan diambil untuk menambahkan Wawasan Kontainer.

Menandai dukungan untuk sumber daya Amazon ECS

Tugas	Konsol	AWS CLI	Tindakan API
Ubah default untuk semua pengguna	Mengubah pengaturan akun	put-account-setting-default	PutAccountSettingDefault
Ubah default untuk pengguna tertentu	Mengubah pengaturan akun	put-account-setting	PutAccountSetting
Konfigurasikan Wawasan Kontainer untuk klaster tertentu	Membuat cluster untuk jenis peluncuran Fargate dan Eksternal menggunakan konsol Membuat cluster untuk jenis peluncuran Amazon EC2 menggunakan konsol Memperbarui cluster menggunakan konsol	buat-cluster UpdateCluster	CreateCluster UpdateCluster

Important

Untuk cluster yang berisi tugas atau layanan yang menggunakan tipe peluncuran EC2, instance container Anda harus menjalankan agen Amazon ECS versi 1.29.0 atau yang lebih baru. Untuk informasi selengkapnya, lihat Manajemen instance kontainer Linux.

Izin yang diperlukan untuk CloudWatch Wawasan Kontainer untuk melihat peristiwa siklus hidup Amazon ECS

Anda harus mengonfigurasi izin yang benar, lalu Anda dapat mengonfigurasi dan melihat peristiwa di konsol CloudWatch Container Insights. Untuk informasi selengkapnya, lihat peristiwa siklus hidup Amazon ECS dalam Wawasan Penampung di Panduan Pengguna Amazon. CloudWatch Untuk informasi selengkapnya tentang kebijakan IAM CloudWatch, lihat AWS Identity and Access Management untuk CloudWatch.

Izin diperlukan untuk mengonfigurasi Wawasan Kontainer untuk melihat peristiwa siklus hidup Amazon ECS

Izin berikut diperlukan dalam peran tugas untuk mengonfigurasi peristiwa siklus hidup:

- peristiwa: PutRule
- peristiwa: PutTargets
- log: CreateLogGroup

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutRule",
        "events:PutTargets",
        "logs:CreateLogGroup"
      ],
      "Resource": "*"
    }
  ]
}
```

Izin diperlukan untuk melihat peristiwa siklus hidup Amazon ECS di Wawasan Kontainer

Izin berikut diperlukan untuk melihat peristiwa siklus hidup. Tambahkan izin berikut sebagai kebijakan inline ke peran eksekusi tugas. Untuk informasi selengkapnya, lihat Menambahkan dan Menghapus Kebijakan IAM.

- peristiwa: DescribeRule
- peristiwa: ListTargetsByRule
- log: DescribeLogGroups

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:DescribeRule",
        "events:ListTargetsByRule",
        "logs:DescribeLogGroups"
      ],
      "Resource": "*"
    }
  ]
}
```

Tentukan kesehatan tugas Amazon ECS menggunakan pemeriksaan kesehatan kontainer

Saat Anda membuat definisi tugas, Anda dapat mengonfigurasi pemeriksaan kesehatan untuk contaners Anda. Pemeriksaan Kesehatan adalah perintah yang berjalan secara lokal pada wadah dan memvalidasi kesehatan dan ketersediaan aplikasi.

Agen kontainer Amazon ECS hanya memantau dan melaporkan pemeriksaan kesehatan yang ditentukan dalam definisi tugas. Amazon ECS tidak memantau pemeriksaan kesehatan Docker yang disematkan dalam gambar kontainer tetapi tidak ditentukan dalam definisi penampung. Parameter pemeriksaan kondisi yang ditentukan dalam ketentuan kontainer menimpa setiap pemeriksaan kondisi Docker yang ada di citra kontainer.

Ketika pemeriksaan kesehatan didefinisikan dalam definisi tugas, wadah menjalankan proses pemeriksaan kesehatan di dalam wadah, dan kemudian mengevaluasi kode keluar untuk menentukan kesehatan aplikasi.

Pemeriksaan kesehatan terdiri dari parameter berikut:

- Command — Perintah yang dijalankan kontainer untuk menentukan apakah itu sehat. Array string dapat dimulai dengan `CMD` untuk menjalankan argumen perintah secara langsung, atau `CMD-SHELL` untuk menjalankan perintah dengan shell default container.
- Interval — Periode waktu (dalam detik) antara setiap pemeriksaan kesehatan.
- Timeout — Periode waktu (dalam hitungan detik) untuk menunggu pemeriksaan kesehatan berhasil sebelum dianggap gagal.
- Mencoba lagi — Berapa kali untuk mencoba kembali pemeriksaan kesehatan yang gagal sebelum wadah dianggap tidak sehat.
- Periode mulai - Masa tenggang opsional untuk menyediakan waktu kontainer untuk bootstrap sebelum pemeriksaan kesehatan gagal dihitung terhadap jumlah maksimum percobaan ulang.

Untuk informasi tentang cara menentukan pemeriksaan kesehatan dalam definisi tugas, lihatPemeriksaan kondisi.

Berikut ini menjelaskan kemungkinan nilai status kesehatan untuk sebuah wadah:

- `HEALTHY`—Pemeriksaan kesehatan kontainer telah berhasil lolos.
- `UNHEALTHY`—Pemeriksaan kesehatan kontainer gagal.
- `UNKNOWN`—Pemeriksaan kesehatan kontainer sedang dievaluasi, tidak ada pemeriksaan kesehatan kontainer yang ditentukan, atau Amazon ECS tidak memiliki status kesehatan wadah.

Perintah pemeriksaan kesehatan berjalan di wadah. Oleh karena itu Anda harus menyertakan perintah dalam gambar kontainer.

Pemeriksaan kesehatan terhubung ke aplikasi melalui antarmuka loopback kontainer di `localhost` atau. `127.0.0.1` Kode keluar `0` menunjukkan keberhasilan, dan kode keluar bukan nol menunjukkan kegagalan.

Pertimbangkan hal-hal berikut saat menggunakan pemeriksaan kesehatan kontainer:

- Ketika agen Amazon ECS tidak dapat terhubung ke layanan Amazon ECS, layanan melaporkan penampung sebagai. UNHEALTHY
- Status pemeriksaan kesehatan adalah tanggapan “terakhir terdengar dari” dari agen Amazon ECS. Tidak ada asumsi yang dibuat tentang status pemeriksaan kesehatan kontainer.
- Pemeriksaan kondisi kontainer memerlukan versi 1.17.0 atau yang lebih baru dari agen kontainer Amazon ECS.
- Pemeriksaan kesehatan kontainer didukung untuk tugas Fargate jika Anda menggunakan versi platform Linux atau yang lebih besar `1.1.0` atau versi `1.1.0` platform Windows atau yang lebih besar

Bagaimana Amazon ECS menentukan kesehatan tugas

Wadah yang penting dan memiliki perintah pemeriksaan kesehatan dalam definisi tugas adalah satu-satunya yang dipertimbangkan untuk menentukan kesehatan tugas.

Aturan berikut dievaluasi secara berurutan:

1. Jika status satu wadah penting adalahUNHEALTHY, maka status tugasnya adalahUNHEALTHY.
2. Jika status satu wadah penting adalahUNKNOWN, maka status tugasnya adalahUNKNOWN.
3. Jika status semua wadah penting adalahHEALTHY, maka status tugasnya adalahHEALTHY.

Pertimbangkan contoh kesehatan tugas berikut dengan 2 wadah penting.

Kontainer 1 kesehatan	Kontainer 2 kesehatan	Kesehatan tugas
UNHEALTHY	UNKNOWN	UNHEALTHY
UNHEALTHY	HEALTHY	UNHEALTHY
HEALTHY	UNKNOWN	UNKNOWN
HEALTHY	HEALTHY	HEALTHY

Perhatikan contoh kesehatan tugas berikut dengan 3 kontainer.

Kontainer 1 kesehatan	Kontainer 2 kesehatan	Kontainer 3 kesehatan	Kesehatan tugas
UNHEALTHY	UNKNOWN	UNKNOWN	UNHEALTHY
UNHEALTHY	UNKNOWN	HEALTHY	UNHEALTHY
UNHEALTHY	HEALTHY,	HEALTHY	UNHEALTHY
HEALTHY	UNKNOWN	HEALTHY	UNKNOWN
HEALTHY	UNKNOWN	UNKNOWN	UNKNOWN
HEALTHY	HEALTHY	HEALTHY	HEALTHY

Lihat kesehatan wadah Amazon ECS

Anda dapat melihat kesehatan kontainer di konsol, dan `DescribeTasks` responsnya. Untuk informasi selengkapnya, lihat DescribeTasksdi Referensi API Amazon Elastic Container Service.

Jika Anda menggunakan logging untuk penampung Anda, misalnya Amazon CloudWatch, Anda dapat mengonfigurasi perintah pemeriksaan kesehatan untuk meneruskan output kesehatan kontainer ke log Anda. Pastikan untuk menggunakan `2&1` untuk menangkap `stderr` informasi `stdout` dan informasi.

```
"command": [
     "CMD-SHELL",
     "curl -f http://localhost/ >> /proc/1/fd/1 2>&1  || exit 1"
   ],
```

Pantau kesehatan instans kontainer Amazon ECS

Amazon ECS menyediakan pemantauan kesehatan instans kontainer. Anda dapat dengan cepat menentukan apakah Amazon ECS telah mendeteksi masalah apa pun yang mungkin mencegah instance penampung Anda menjalankan container. Amazon ECS melakukan pemeriksaan otomatis pada setiap instance container yang berjalan dengan versi agen `1.57.0` atau yang lebih baru untuk mengidentifikasi masalah. Untuk informasi selengkapnya tentang memverifikasi versi agen sebuah instance container, lihatMemperbarui agen kontainer Amazon ECS.

Anda harus menggunakan AWS CLI versi `1.22.3` atau yang lebih baru atau AWS CLI versi `2.3.6` atau yang lebih baru. Untuk informasi tentang cara memperbarui AWS CLI, lihat Menginstal atau memperbarui versi terbaru dari AWS CLI Panduan AWS Command Line Interface Pengguna Versi 2.

Pemeriksaan status dilakukan sekitar dua kali per menit, mengembalikan pass atau status gagal. Jika semua cek lulus, status keseluruhan instance adalah`OK`. Jika satu atau lebih pemeriksaan gagal, status keseluruhannya adalah`IMPAIRED`. Pemeriksaan status dibangun ke dalam agen penampung Amazon ECS, sehingga tidak dapat dimatikan atau dihapus. Anda dapat melihat hasil dari pemeriksaan status ini untuk mengidentifikasi masalah spesifik yang dapat dideteksi. Untuk informasi selengkapnya, lihat the section called “Pemeriksaan kondisi”.

Jalankan `DescribeContainerInstances` API dengan `CONTAINER_INSTANCE_HEALTH` opsi untuk mengambil kesehatan instance container.

```
aws ecs describe-container-instances \
     --cluster cluster_name \
     --container-instances 47279cd2cadb41cbaef2dcEXAMPLE \
     --include CONTAINER_INSTANCE_HEALTH
```

Berikut ini adalah contoh objek status kesehatan dalam output.

```
"healthStatus": {
 "overallStatus": "OK",
 "details": [{
  "type": "CONTAINER_RUNTIME",
  "status": "OK",
  "lastUpdated": "2021-11-10T03:30:26+00:00",
  "lastStatusChange": "2021-11-10T03:26:41+00:00"
 }]
}
```

Topik terkait

- Pantau Amazon ECS menggunakan CloudWatch

Identifikasi peluang pengoptimalan Amazon ECS menggunakan data pelacakan aplikasi

Amazon ECS terintegrasi dengan AWS Distro OpenTelemetry untuk mengumpulkan data jejak dari aplikasi Anda. Amazon ECS menggunakan AWS Distro untuk wadah OpenTelemetry sespan untuk mengumpulkan dan merutekan data penelusuran ke. AWS X-Ray Untuk informasi selengkapnya, lihat Menyiapkan AWS Distro untuk OpenTelemetry Kolektor di Amazon ECS. Anda kemudian dapat menggunakan AWS X-Ray untuk mengidentifikasi kesalahan dan pengecualian, menganalisis kemacetan kinerja dan waktu respons.

Agar AWS Distro for OpenTelemetry Collector dapat mengirim data jejak AWS X-Ray, aplikasi Anda harus dikonfigurasi untuk membuat data jejak. Untuk informasi selengkapnya, lihat Menginstrumentasi aplikasi Anda AWS X-Ray di Panduan AWS X-Ray Pengembang.

Izin IAM yang diperlukan untuk AWS Distro untuk integrasi dengan OpenTelemetry AWS X-Ray

Integrasi Amazon ECS dengan AWS Distro for OpenTelemetry mengharuskan Anda membuat peran IAM tugas dan menentukan peran dalam definisi tugas Anda. Kami merekomendasikan bahwa AWS Distro untuk OpenTelemetry sespan juga dikonfigurasi untuk merutekan log kontainer ke CloudWatch Log yang memerlukan peran IAM eksekusi tugas dibuat dan ditentukan dalam definisi tugas Anda juga. Konsol Amazon ECS menangani peran IAM eksekusi tugas atas nama Anda, tetapi peran tugas IAM harus dibuat secara manual. Untuk informasi selengkapnya tentang membuat peran IAM eksekusi tugas, lihatPeran IAM eksekusi tugas Amazon ECS.

> **Important**
>
> Jika Anda juga mengumpulkan metrik aplikasi menggunakan AWS Distro untuk OpenTelemetry integrasi, pastikan peran IAM tugas Anda juga berisi izin yang diperlukan untuk integrasi tersebut. Untuk informasi selengkapnya, lihat Korelasikan kinerja aplikasi Amazon ECS menggunakan metrik aplikasi.

Untuk membuat peran layanan untuk Elastic Container Service (konsol IAM)

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.

2. Di panel navigasi konsol IAM, pilih Peran, dan lalu pilih Buat peran.
3. Untuk jenis entitas Tepercaya, pilih Layanan AWS.
4. Untuk Service atau use case, pilih Elastic Container Service, lalu pilih kasus penggunaan Elastic Container Service Task.
5. Pilih Berikutnya.
6. Di bagian Tambahkan izin, cari AWSDistroOpenTelemetryPolicyForXray, lalu pilih kebijakan.
7. (Opsional) Tetapkan batas izin. Ini adalah fitur lanjutan yang tersedia untuk peran layanan, tetapi bukan peran tertaut layanan.

   a. Buka bagian Setel batas izin, lalu pilih Gunakan batas izin untuk mengontrol izin peran maksimum.

      IAM menyertakan daftar kebijakan yang AWS dikelola dan dikelola pelanggan di akun Anda.

   b. Pilih kebijakan yang akan digunakan untuk batas izin.
8. Pilih Berikutnya.
9. Masukkan nama peran atau akhiran nama peran untuk membantu Anda mengidentifikasi tujuan peran.

   Important

   Saat Anda memberi nama peran, perhatikan hal berikut:

   - Nama peran harus unik di dalam diri Anda Akun AWS, dan tidak dapat dibuat unik berdasarkan kasus.

     Misalnya, jangan membuat peran bernama keduanya **`PRODROLE`** dan**`prodrole`**. Ketika nama peran digunakan dalam kebijakan atau sebagai bagian dari ARN, nama peran tersebut peka huruf besar/kecil, namun ketika nama peran muncul kepada pelanggan di konsol, seperti selama proses masuk, nama peran tersebut tidak peka huruf besar/kecil.
   - Anda tidak dapat mengedit nama peran setelah dibuat karena entitas lain mungkin mereferensikan peran tersebut.
10. (Opsional) Untuk Deskripsi, masukkan deskripsi untuk peran tersebut.
11. (Opsional) Untuk mengedit kasus penggunaan dan izin untuk peran, di Langkah 1: Pilih entitas tepercaya atau Langkah 2: Tambahkan izin, pilih Edit.

12. (Opsional) Untuk membantu mengidentifikasi, mengatur, atau mencari peran, tambahkan tag sebagai pasangan nilai kunci. Untuk informasi selengkapnya tentang penggunaan tanda di IAM, lihat Menandai sumber daya IAM di Panduan Pengguna IAM.
13. Tinjau peran lalu pilih Buat peran.

Menentukan AWS Distro untuk OpenTelemetry sespan untuk AWS X-Ray integrasi dalam definisi tugas Anda

Konsol Amazon ECS menyederhanakan pembuatan AWS Distro untuk wadah OpenTelemetry sespan dengan menggunakan opsi Use trace collection. Untuk informasi selengkapnya, lihat Membuat definisi tugas menggunakan konsol.

Jika Anda tidak menggunakan konsol Amazon ECS, Anda dapat menambahkan AWS Distro untuk wadah OpenTelemetry sespan ke definisi tugas Anda. Cuplikan definisi tugas berikut menunjukkan definisi container untuk menambahkan AWS Distro untuk OpenTelemetry sespan untuk integrasi. AWS X-Ray

```
{
 "family": "otel-using-xray",
 "taskRoleArn": "arn:aws:iam::111122223333:role/AmazonECS_OpenTelemetryXrayRole",
 "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
 "containerDefinitions": [{
   "name": "aws-otel-emitter",
   "image": "application-image",
   "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
     "awslogs-create-group": "true",
     "awslogs-group": "/ecs/aws-otel-emitter",
     "awslogs-region": "us-east-1",
     "awslogs-stream-prefix": "ecs"
    }
   },
   "dependsOn": [{
    "containerName": "aws-otel-collector",
    "condition": "START"
   }]
  },
  {
   "name": "aws-otel-collector",
```

```
    "image": "public.ecr.aws/aws-observability/aws-otel-collector:v0.30.0",
    "essential": true,
    "command": [
     "--config=/etc/ecs/otel-instance-metrics-config.yaml"
    ],
    "logConfiguration": {
     "logDriver": "awslogs",
     "options": {
      "awslogs-create-group": "True",
      "awslogs-group": "/ecs/ecs-aws-otel-sidecar-collector",
      "awslogs-region": "us-east-1",
      "awslogs-stream-prefix": "ecs"
     }
    }
   }
  ],
  "networkMode": "awsvpc",
  "requiresCompatibilities": [
   "FARGATE"
  ],
  "cpu": "1024",
  "memory": "3072"
}
```

Korelasikan kinerja aplikasi Amazon ECS menggunakan metrik aplikasi

Amazon ECS di Fargate mendukung pengumpulan metrik dari aplikasi Anda yang berjalan di Fargate dan mengekspornya ke Amazon atau CloudWatch Amazon Managed Service untuk Prometheus.

Anda dapat menggunakan metadata yang dikumpulkan untuk mengkorelasikan data kinerja aplikasi dengan data infrastruktur yang mendasarinya, sehingga mengurangi waktu rata-rata untuk menyelesaikan masalah.

Amazon ECS menggunakan AWS Distro untuk wadah OpenTelemetry sespan untuk mengumpulkan dan merutekan metrik aplikasi Anda ke tujuan. Pengalaman konsol Amazon ECS menyederhanakan proses menambahkan integrasi ini saat membuat definisi tugas Anda.

Topik

- Mengekspor metrik aplikasi ke Amazon CloudWatch

- Mengekspor metrik aplikasi ke Amazon Managed Service untuk Prometheus

Mengekspor metrik aplikasi ke Amazon CloudWatch

Amazon ECS di Fargate mendukung ekspor metrik aplikasi kustom Anda ke Amazon CloudWatch sebagai metrik khusus. Ini dilakukan dengan menambahkan AWS Distro untuk wadah OpenTelemetry sespan ke definisi tugas Anda. Konsol Amazon ECS menyederhanakan proses ini dengan menambahkan opsi Gunakan koleksi metrik saat membuat definisi tugas baru. Untuk informasi selengkapnya, lihat Membuat definisi tugas menggunakan konsol.

Metrik aplikasi diekspor ke CloudWatch Log dengan nama grup log `/aws/ecs/application/metrics` dan metrik dapat dilihat di namespace. `ECS/AWSOTel/Application` Aplikasi Anda harus diinstrumentasi dengan OpenTelemetry SDK. Untuk informasi selengkapnya, lihat Pengantar AWS Distro untuk OpenTelemetry di AWS Distro untuk OpenTelemetry dokumentasi.

Pertimbangan

Berikut ini harus dipertimbangkan saat menggunakan Amazon ECS pada integrasi Fargate AWS dengan Distro OpenTelemetry untuk mengirim metrik aplikasi ke Amazon. CloudWatch

- Integrasi ini hanya mengirimkan metrik aplikasi kustom Anda ke CloudWatch. Jika menginginkan metrik tingkat tugas, Anda dapat mengaktifkan Wawasan Kontainer di konfigurasi cluster Amazon ECS. Untuk informasi selengkapnya, lihat Pantau kontainer Amazon ECS menggunakan Wawasan Kontainer.
- AWS Distro untuk OpenTelemetry integrasi didukung untuk beban kerja Amazon ECS yang dihosting di beban kerja Fargate dan Amazon ECS yang dihosting di instans Amazon EC2. Instance eksternal saat ini tidak didukung.
- CloudWatch mendukung maksimum 30 dimensi per metrik. Secara default, Amazon ECS secara default menyertakan`TaskARN`,,,, `ClusterARN` `LaunchType` `TaskDefinitionFamily`, dan `TaskDefinitionRevision` dimensi ke metrik. 25 dimensi yang tersisa dapat ditentukan oleh aplikasi Anda. Jika lebih dari 30 dimensi dikonfigurasi, tidak CloudWatch dapat menampilkannya. Ketika ini terjadi, metrik aplikasi akan muncul di namespace `ECS/AWSOTel/Application` CloudWatch metrik tetapi tanpa dimensi apa pun. Anda dapat instrumen aplikasi Anda untuk menambahkan dimensi tambahan. Untuk informasi selengkapnya, lihat Menggunakan CloudWatch metrik dengan AWS Distro untuk OpenTelemetry dokumentasi di AWS Distro. OpenTelemetry

Izin IAM yang diperlukan untuk AWS Distro untuk OpenTelemetry integrasi dengan Amazon CloudWatch

Integrasi Amazon ECS dengan AWS Distro for OpenTelemetry mengharuskan Anda membuat peran IAM tugas dan menentukan peran dalam definisi tugas Anda. Kami merekomendasikan bahwa AWS Distro untuk OpenTelemetry sespan juga dikonfigurasi untuk merutekan log kontainer ke CloudWatch Log yang memerlukan peran IAM eksekusi tugas dibuat dan ditentukan dalam definisi tugas Anda juga. Konsol Amazon ECS menangani peran IAM eksekusi tugas atas nama Anda, tetapi peran IAM tugas harus dibuat secara manual dan ditambahkan ke definisi tugas Anda. Untuk informasi selengkapnya tentang peran IAM eksekusi tugas, lihatPeran IAM eksekusi tugas Amazon ECS.

> **Important**
>
> Jika Anda juga mengumpulkan data jejak aplikasi menggunakan AWS Distro untuk OpenTelemetry integrasi, pastikan peran IAM tugas Anda juga berisi izin yang diperlukan untuk integrasi tersebut. Untuk informasi selengkapnya, lihat Identifikasi peluang pengoptimalan Amazon ECS menggunakan data pelacakan aplikasi.
> Jika aplikasi Anda memerlukan izin tambahan, Anda harus menambahkannya ke kebijakan ini. Setiap definisi tugas hanya dapat menentukan satu peran IAM tugas. Misalnya, jika Anda menggunakan file konfigurasi kustom yang disimpan di Systems Manager, Anda harus menambahkan `ssm:GetParameters` izin ke kebijakan IAM ini.

Untuk membuat peran layanan untuk Elastic Container Service (konsol IAM)

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi konsol IAM, pilih Peran, dan lalu pilih Buat peran.
3. Untuk jenis entitas Tepercaya, pilih Layanan AWS.
4. Untuk Service atau use case, pilih Elastic Container Service, lalu pilih kasus penggunaan Elastic Container Service Task.
5. Pilih Berikutnya.
6. Di bagian Tambahkan izin, cari AWSDistroOpenTelemetryPolicyForXray, lalu pilih kebijakan.
7. (Opsional) Tetapkan batas izin. Ini adalah fitur lanjutan yang tersedia untuk peran layanan, tetapi bukan peran tertaut layanan.

a. Buka bagian Setel batas izin, lalu pilih Gunakan batas izin untuk mengontrol izin peran maksimum.

   IAM menyertakan daftar kebijakan yang AWS dikelola dan dikelola pelanggan di akun Anda.

b. Pilih kebijakan yang akan digunakan untuk batas izin.

8. Pilih Berikutnya.
9. Masukkan nama peran atau akhiran nama peran untuk membantu Anda mengidentifikasi tujuan peran.

> **Important**
>
> Saat Anda memberi nama peran, perhatikan hal berikut:
>
> - Nama peran harus unik di dalam diri Anda Akun AWS, dan tidak dapat dibuat unik berdasarkan kasus.
>
>   Misalnya, jangan membuat peran bernama keduanya **`PRODROLE`** dan**`prodrole`**. Ketika nama peran digunakan dalam kebijakan atau sebagai bagian dari ARN, nama peran tersebut peka huruf besar/kecil, namun ketika nama peran muncul kepada pelanggan di konsol, seperti selama proses masuk, nama peran tersebut tidak peka huruf besar/kecil.
> - Anda tidak dapat mengedit nama peran setelah dibuat karena entitas lain mungkin mereferensikan peran tersebut.

10. (Opsional) Untuk Deskripsi, masukkan deskripsi untuk peran tersebut.
11. (Opsional) Untuk mengedit kasus penggunaan dan izin untuk peran, di Langkah 1: Pilih entitas tepercaya atau Langkah 2: Tambahkan izin, pilih Edit.
12. (Opsional) Untuk membantu mengidentifikasi, mengatur, atau mencari peran, tambahkan tag sebagai pasangan nilai kunci. Untuk informasi selengkapnya tentang penggunaan tanda di IAM, lihat Menandai sumber daya IAM di Panduan Pengguna IAM.
13. Tinjau peran lalu pilih Buat peran.

Menentukan AWS Distro untuk OpenTelemetry sespan dalam definisi tugas Anda

Konsol Amazon ECS menyederhanakan pengalaman membuat AWS Distro untuk wadah OpenTelemetry sespan dengan menggunakan opsi Gunakan koleksi metrik. Untuk informasi selengkapnya, lihat Membuat definisi tugas menggunakan konsol.

Jika Anda tidak menggunakan konsol Amazon ECS, Anda dapat menambahkan AWS Distro untuk wadah OpenTelemetry sespan ke definisi tugas Anda secara manual. Contoh definisi tugas berikut menunjukkan definisi container untuk menambahkan AWS Distro untuk OpenTelemetry sespan untuk integrasi Amazon. CloudWatch

```
{
 "family": "otel-using-cloudwatch",
 "taskRoleArn": "arn:aws:iam::111122223333:role/AmazonECS_OpenTelemetryCloudWatchRole",
 "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
 "containerDefinitions": [
    {
   "name": "aws-otel-emitter",
   "image": "application-image",
   "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
     "awslogs-create-group": "true",
     "awslogs-group": "/ecs/aws-otel-emitter",
     "awslogs-region": "us-east-1",
     "awslogs-stream-prefix": "ecs"
    }
   },
   "dependsOn": [{
    "containerName": "aws-otel-collector",
    "condition": "START"
   }]
  },
  {
   "name": "aws-otel-collector",
   "image": "public.ecr.aws/aws-observability/aws-otel-collector:v0.30.0",
   "essential": true,
   "command": [
    "--config=/etc/ecs/ecs-cloudwatch.yaml"
   ],
   "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
```

```
      "awslogs-create-group": "True",
      "awslogs-group": "/ecs/ecs-aws-otel-sidecar-collector",
      "awslogs-region": "us-east-1",
      "awslogs-stream-prefix": "ecs"
     }
    }
   }
  ],
  "networkMode": "awsvpc",
  "requiresCompatibilities": [
   "FARGATE"
  ],
  "cpu": "1024",
  "memory": "3072"
}
```

Mengekspor metrik aplikasi ke Amazon Managed Service untuk Prometheus

Amazon ECS mendukung ekspor CPU tingkat tugas, memori, jaringan, dan metrik penyimpanan serta metrik aplikasi kustom Anda ke Amazon Managed Service for Prometheus. Ini dilakukan dengan menambahkan AWS Distro untuk wadah OpenTelemetry sespan ke definisi tugas Anda. Konsol Amazon ECS menyederhanakan proses ini dengan menambahkan opsi Gunakan koleksi metrik saat membuat definisi tugas baru. Untuk informasi selengkapnya, lihat Membuat definisi tugas menggunakan konsol.

Metrik diekspor ke Amazon Managed Service untuk Prometheus dan dapat dilihat menggunakan dasbor Grafana Terkelola Amazon. Aplikasi Anda harus diinstrumentasi dengan pustaka Prometheus atau dengan SDK. OpenTelemetry Untuk informasi selengkapnya tentang menginstrumentasi aplikasi Anda dengan OpenTelemetry SDK, lihat Pengantar AWS Distro untuk OpenTelemetry di Distro untuk dokumentasi AWS . OpenTelemetry

Saat menggunakan pustaka Prometheus, aplikasi Anda harus mengekspos titik akhir yang digunakan untuk mengikis `/metrics` data metrik. Untuk informasi selengkapnya tentang menginstrumentasi aplikasi Anda dengan pustaka Prometheus, lihat pustaka klien Prometheus dalam dokumentasi Prometheus.

Pertimbangan

Berikut ini harus dipertimbangkan saat menggunakan Amazon ECS pada integrasi Fargate AWS dengan Distro OpenTelemetry untuk mengirim metrik aplikasi ke Amazon Managed Service untuk Prometheus.

- AWS Distro untuk OpenTelemetry integrasi didukung untuk beban kerja Amazon ECS yang dihosting di beban kerja Fargate dan Amazon ECS yang dihosting di instans Amazon EC2. Instance eksternal saat ini tidak didukung.
- Secara default, AWS Distro for OpenTelemetry menyertakan semua dimensi tingkat tugas yang tersedia untuk metrik aplikasi Anda saat mengekspor ke Amazon Managed Service untuk Prometheus. Anda juga dapat instrumen aplikasi Anda untuk menambahkan dimensi tambahan. Untuk informasi selengkapnya, lihat Memulai Prometheus Remote Write Exporter for Amazon Managed Service for Prometheus di Distro untuk dokumentasi. AWS OpenTelemetry

Izin IAM yang diperlukan untuk AWS Distro untuk integrasi OpenTelemetry dengan Amazon Managed Service untuk Prometheus

Integrasi Amazon ECS dengan Amazon Managed Service untuk Prometheus menggunakan AWS Distro OpenTelemetry untuk sespan mengharuskan Anda membuat peran IAM tugas dan menentukan peran dalam definisi tugas Anda. Peran IAM tugas ini harus dibuat secara manual menggunakan langkah-langkah di bawah ini sebelum mendaftarkan definisi tugas Anda.

Kami merekomendasikan bahwa AWS Distro untuk OpenTelemetry sespan juga dikonfigurasi untuk merutekan log kontainer ke CloudWatch Log yang memerlukan peran IAM eksekusi tugas dibuat dan ditentukan dalam definisi tugas Anda juga. Konsol Amazon ECS menangani peran IAM eksekusi tugas atas nama Anda, tetapi peran tugas IAM harus dibuat secara manual. Untuk informasi selengkapnya tentang membuat peran IAM eksekusi tugas, lihatPeran IAM eksekusi tugas Amazon ECS.

> **Important**
>
> Jika Anda juga mengumpulkan data jejak aplikasi menggunakan AWS Distro untuk OpenTelemetry integrasi, pastikan peran IAM tugas Anda juga berisi izin yang diperlukan untuk integrasi tersebut. Untuk informasi selengkapnya, lihat Identifikasi peluang pengoptimalan Amazon ECS menggunakan data pelacakan aplikasi.

Untuk membuat peran layanan untuk Elastic Container Service (konsol IAM)

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi konsol IAM, pilih Peran, dan lalu pilih Buat peran.
3. Untuk jenis entitas Tepercaya, pilih Layanan AWS.
4. Untuk Service atau use case, pilih Elastic Container Service, lalu pilih kasus penggunaan Elastic Container Service Task.
5. Pilih Berikutnya.
6. Di bagian Tambahkan izin, cari AmazonPrometheusRemoteWriteAccess, lalu pilih kebijakan.
7. (Opsional) Tetapkan batas izin. Ini adalah fitur lanjutan yang tersedia untuk peran layanan, tetapi bukan peran tertaut layanan.

   a. Buka bagian Setel batas izin, lalu pilih Gunakan batas izin untuk mengontrol izin peran maksimum.

      IAM menyertakan daftar kebijakan yang AWS dikelola dan dikelola pelanggan di akun Anda.

   b. Pilih kebijakan yang akan digunakan untuk batas izin.
8. Pilih Berikutnya.
9. Masukkan nama peran atau akhiran nama peran untuk membantu Anda mengidentifikasi tujuan peran.

> **Important**
>
> Saat Anda memberi nama peran, perhatikan hal berikut:
>
> - Nama peran harus unik di dalam diri Anda Akun AWS, dan tidak dapat dibuat unik berdasarkan kasus.
>
>   Misalnya, jangan membuat peran bernama keduanya **`PRODROLE`** dan**`prodrole`**. Ketika nama peran digunakan dalam kebijakan atau sebagai bagian dari ARN, nama peran tersebut peka huruf besar/kecil, namun ketika nama peran muncul kepada pelanggan di konsol, seperti selama proses masuk, nama peran tersebut tidak peka huruf besar/kecil.
> - Anda tidak dapat mengedit nama peran setelah dibuat karena entitas lain mungkin mereferensikan peran tersebut.

10. (Opsional) Untuk Deskripsi, masukkan deskripsi untuk peran tersebut.
11. (Opsional) Untuk mengedit kasus penggunaan dan izin untuk peran, di Langkah 1: Pilih entitas tepercaya atau Langkah 2: Tambahkan izin, pilih Edit.
12. (Opsional) Untuk membantu mengidentifikasi, mengatur, atau mencari peran, tambahkan tag sebagai pasangan nilai kunci. Untuk informasi selengkapnya tentang penggunaan tanda di IAM, lihat Menandai sumber daya IAM di Panduan Pengguna IAM.
13. Tinjau peran lalu pilih Buat peran.

Menentukan AWS Distro untuk OpenTelemetry sespan dalam definisi tugas Anda

Konsol Amazon ECS menyederhanakan pengalaman membuat AWS Distro untuk wadah OpenTelemetry sespan dengan menggunakan opsi Gunakan koleksi metrik. Untuk informasi selengkapnya, lihat Membuat definisi tugas menggunakan konsol.

Jika Anda tidak menggunakan konsol Amazon ECS, Anda dapat menambahkan AWS Distro untuk wadah OpenTelemetry sespan ke definisi tugas Anda secara manual. Contoh definisi tugas berikut menunjukkan definisi container untuk menambahkan AWS Distro untuk OpenTelemetry sespan untuk Amazon Managed Service untuk integrasi Prometheus.

```
{
 "family": "otel-using-cloudwatch",
 "taskRoleArn": "arn:aws:iam::111122223333:role/AmazonECS_OpenTelemetryCloudWatchRole",
 "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
 "containerDefinitions": [{
   "name": "aws-otel-emitter",
   "image": "application-image",
   "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
     "awslogs-create-group": "true",
     "awslogs-group": "/ecs/aws-otel-emitter",
     "awslogs-region": "aws-region",
     "awslogs-stream-prefix": "ecs"
    }
   },
   "dependsOn": [{
    "containerName": "aws-otel-collector",
    "condition": "START"
   }]
  },
```

```
  {
   "name": "aws-otel-collector",
   "image": "public.ecr.aws/aws-observability/aws-otel-collector:v0.30.0",
   "essential": true,
   "command": [
    "--config=/etc/ecs/ecs-amp.yaml"
   ],
   "environment": [{
    "name": "AWS_PROMETHEUS_ENDPOINT",
    "value": "https://aps-workspaces.aws-region.amazonaws.com/workspaces/
ws-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/api/v1/remote_write"
   }],
   "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
     "awslogs-create-group": "True",
     "awslogs-group": "/ecs/ecs-aws-otel-sidecar-collector",
     "awslogs-region": "aws-region",
     "awslogs-stream-prefix": "ecs"
    }
   }
  }
 ],
 "networkMode": "awsvpc",
 "requiresCompatibilities": [
  "FARGATE"
 ],
 "cpu": "1024",
 "memory": "3072"
}
```

Log panggilan Amazon ECS API menggunakan AWS CloudTrail

Amazon ECS terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di Amazon ECS. CloudTrail menangkap semua panggilan API untuk Amazon ECS sebagai peristiwa, termasuk panggilan dari konsol Amazon ECS dan dari panggilan kode ke operasi Amazon ECS API. Untuk melindungi VPC Anda, permintaan yang ditolak oleh kebijakan titik akhir VPC, tetapi sebaliknya diizinkan, tidak dicatat. CloudTrail

Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara berkelanjutan ke bucket Amazon S3, termasuk acara untuk Amazon ECS. Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat

acara. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat ke Amazon ECS, alamat IP dari mana permintaan itu dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk informasi selengkapnya, silakan lihat Panduan Pengguna AWS CloudTrail.

Informasi Amazon ECS di CloudTrail

CloudTrail diaktifkan di AWS akun Anda saat Anda membuat akun. Ketika aktivitas terjadi di Amazon ECS, aktivitas tersebut direkam dalam suatu CloudTrail peristiwa bersama dengan peristiwa AWS layanan lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh acara terbaru di AWS akun Anda. Untuk informasi selengkapnya, lihat Melihat Acara dengan Riwayat CloudTrail Acara.

Untuk catatan peristiwa yang sedang berlangsung di AWS akun Anda, termasuk peristiwa untuk Amazon ECS, buat jejak yang CloudTrail digunakan untuk mengirimkan file log ke bucket Amazon S3. Secara default, ketika Anda membuat jejak di konsol tersebut, jejak diterapkan ke semua Wilayah . Jejak mencatat peristiwa dari semua Wilayah di AWS partisi dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Lihat informasi yang lebih lengkap di:

- Ikhtisar Pembuatan Jejak
- CloudTrail Layanan dan Integrasi yang Didukung
- Mengkonfigurasi Notifikasi Amazon SNS untuk CloudTrail
- Menerima File CloudTrail Log dari Beberapa Wilayah dan Menerima File CloudTrail Log dari Beberapa Akun

Semua tindakan Amazon ECS dicatat oleh CloudTrail dan didokumentasikan dalam Referensi API Amazon Elastic Container Service. Misalnya, panggilan ke`CreateService`, `RunTask` dan `DeleteCluster` bagian menghasilkan entri dalam file CloudTrail log.

Setiap peristiwa atau entri log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan berikut hal ini:

- Baik permintaan tersebut dibuat dengan kredensial pengguna root atau pengguna.
- Apakah permintaan dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna gabungan.

- Apakah permintaan itu dibuat oleh AWS layanan lain.

Untuk informasi lain, lihat Elemen userIdentity CloudTrail .

Memahami entri file log Amazon ECS

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa mewakili permintaan tunggal dari sumber mana pun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, sehingga file tersebut tidak muncul dalam urutan tertentu.

> **Note**
>
> Contoh-contoh ini telah diformat untuk meningkatkan keterbacaan. Dalam file CloudTrail log, semua entri dan peristiwa digabungkan menjadi satu baris. Selain itu, contoh ini terbatas pada satu entri Amazon ECS. Dalam file CloudTrail log nyata, Anda melihat entri dan peristiwa dari beberapa AWS layanan.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan `CreateCluster` tindakan:

```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
        "arn": "arn:aws:sts::123456789012:user/Mary_Major",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-06-20T18:32:25Z"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/Admin",
```

```
                "accountId": "123456789012",
                "userName": "Mary_Major"
            }
        }
    },
    "eventTime": "2018-06-20T19:04:36Z",
    "eventSource": "ecs.amazonaws.com",
    "eventName": "CreateCluster",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "203.0.113.12",
    "userAgent": "console.amazonaws.com",
    "requestParameters": {
        "clusterName": "default"
    },
    "responseElements": {
        "cluster": {
            "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/default",
            "pendingTasksCount": 0,
            "registeredContainerInstancesCount": 0,
            "status": "ACTIVE",
            "runningTasksCount": 0,
            "statistics": [],
            "clusterName": "default",
            "activeServicesCount": 0
        }
    },
    "requestID": "cb8c167e-EXAMPLE",
    "eventID": "e3c6f4ce-EXAMPLE",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


Identifikasi perilaku yang tidak sah menggunakan Runtime Monitoring

Amazon GuardDuty adalah layanan deteksi ancaman yang membantu melindungi akun, wadah, beban kerja, dan data di AWS lingkungan Anda. Menggunakan model machine learning (ML), serta kemampuan deteksi anomali dan ancaman, GuardDuty terus memantau berbagai sumber log dan aktivitas runtime untuk mengidentifikasi dan memprioritaskan potensi risiko keamanan dan aktivitas berbahaya di lingkungan Anda.

Runtime Monitoring in GuardDuty melindungi beban kerja yang berjalan pada instance container Fargate dan EC2 dengan terus memantau aktivitas AWS log dan jaringan untuk mengidentifikasi perilaku berbahaya atau tidak sah. Runtime Monitoring menggunakan agen GuardDuty keamanan ringan yang dikelola sepenuhnya yang menganalisis perilaku on-host, seperti akses file, eksekusi proses, dan koneksi jaringan. Ini mencakup masalah termasuk peningkatan hak istimewa, penggunaan kredensional yang terbuka, atau komunikasi dengan alamat IP berbahaya, domain, dan keberadaan malware di instans Amazon EC2 dan beban kerja kontainer Anda. Untuk informasi selengkapnya, lihat GuardDutyRuntime Monitoring di Panduan GuardDuty Pengguna.

Administrator keamanan Anda mengaktifkan Runtime Monitoring untuk satu atau beberapa akun di AWS Organizations for GuardDuty. Mereka juga memilih apakah GuardDuty secara otomatis menyebarkan agen GuardDuty keamanan saat Anda menggunakan Fargate. Semua cluster Anda secara otomatis dilindungi, dan GuardDuty mengelola agen keamanan atas nama Anda.

Anda juga dapat mengonfigurasi agen GuardDuty keamanan secara manual dalam kasus berikut:

- Anda menggunakan instans kontainer EC2
- Anda memerlukan kontrol granular untuk mengaktifkan Runtime Monitoring di tingkat cluster

Untuk menggunakan Runtime Monitoring, Anda harus mengonfigurasi cluster yang dilindungi, dan menginstal serta mengelola agen GuardDuty keamanan pada instans kontainer EC2 Anda.

Bagaimana Runtime Monitoring bekerja dengan Amazon ECS

Runtime Monitoring menggunakan agen GuardDuty keamanan ringan yang memantau aktivitas beban kerja Amazon ECS untuk bagaimana aplikasi meminta, mendapatkan akses, dan mengkonsumsi sumber daya sistem yang mendasarinya.

Untuk tugas Fargate, agen GuardDuty keamanan berjalan sebagai wadah sespan untuk setiap tugas.

Untuk instans kontainer EC2, agen GuardDuty keamanan berjalan sebagai proses pada instance.

Agen GuardDuty keamanan mengumpulkan data dari sumber daya berikut, dan kemudian mengirimkan data GuardDuty untuk diproses. Anda dapat melihat temuan di GuardDuty konsol. Anda juga dapat mengirimnya ke pihak lain Layanan AWS seperti AWS Security Hub, atau vendor keamanan pihak ketiga untuk agregasi dan remediasi. Untuk informasi tentang cara melihat dan mengelola temuan, lihat Mengelola GuardDuty temuan Amazon di Panduan GuardDuty Pengguna Amazon.

- Tanggapan dari panggilan Amazon ECS API berikut:

- DescribeClusters

  Parameter respons menyertakan tag Runtime Monitoring (saat tag disetel) saat Anda menggunakan `--include TAGS` opsi.

- DescribeTasks

  Untuk tipe peluncuran Fargate, parameter respons termasuk wadah GuardDuty sespan.

- ListAccountSettings

  Parameter respons mencakup pengaturan akun Runtime Monitoring, yang ditetapkan oleh administrator keamanan Anda.

- Data introspeksi agen kontainer. Untuk informasi selengkapnya, lihat Introspeksi wadah Amazon ECS.
- Titik akhir metadata tugas untuk jenis peluncuran:
  - Titik akhir metadata tugas Amazon ECS versi 4
  - Titik akhir metadata tugas Amazon ECS versi 4 untuk tugas di Fargate

Pertimbangan

Pertimbangkan hal berikut saat menggunakan Runtime Monitoring:

- Runtime Monitoring memiliki biaya yang terkait dengannya. Untuk informasi selengkapnya, lihat GuardDuty Harga Amazon.
- Runtime Monitoring tidak didukung di Amazon ECS Anywhere.
- Runtime Monitoring tidak didukung untuk sistem operasi Windows.
- Saat Anda menggunakan Amazon ECS Exec di Fargate, Anda harus menentukan nama penampung karena agen GuardDuty keamanan berjalan sebagai wadah sespan.
- Anda tidak dapat menggunakan Amazon ECS Exec pada wadah GuardDuty sespan agen keamanan.
- Pengguna IAM yang mengontrol Runtime Monitoring di tingkat cluster, harus memiliki izin IAM yang sesuai untuk penandaan. Untuk informasi selengkapnya, lihat tutorial IAM: Menentukan izin untuk mengakses AWS sumber daya berdasarkan tag di Panduan Pengguna IAM.
- Tugas Fargate harus menggunakan peran eksekusi tugas. Peran ini memberikan izin tugas untuk mengambil, memperbarui, dan mengelola agen GuardDuty keamanan, yang disimpan dalam repositori pribadi Amazon ECR, atas nama Anda.

Pemanfaatan Sumber Daya

Tag yang Anda tambahkan ke klaster dihitung terhadap kuota tag cluster.

Kontainer sespan GuardDuty agen tidak dihitung terhadap kontainer per kuota definisi tugas.

Seperti kebanyakan perangkat lunak keamanan, ada sedikit biaya overhead untuk GuardDuty. Untuk informasi tentang batas memori Fargate, lihat batas CPU dan memori di GuardDuty Panduan Pengguna. Untuk informasi tentang batas memori Amazon EC2, lihat CPU dan batas memori untuk GuardDuty agen.

GuardDuty manajemen agen

Jika Anda menggunakan instans kontainer EC2, Anda harus mengonfigurasi Runtime Monitoring secara manual. Untuk informasi selengkapnya, lihat Manajemen Pemantauan Runtime Manual.

Saat Anda menggunakan manajemen GuardDuty agen GuardDuty , lakukan operasi berikut:

- Membuat titik akhir VPC untuk GuardDuty setiap VPC yang menghosting cluster.
- Mengambil, dan menginstal agen GuardDuty keamanan terbaru sebagai wadah sespan pada semua tugas Fargate mandiri baru, dan penerapan layanan baru.

  Penyebaran layanan baru terjadi saat pertama kali Anda meluncurkan layanan, atau saat Anda memperbarui layanan yang ada dengan opsi force new deployment.

Prasyarat

Berikut ini adalah prasyarat untuk menggunakan Runtime Monitoring:

- Versi platform Fargate harus `1.4.0` atau lebih baru untuk Linux.
- Untuk informasi tentang sistem operasi dan arsitektur Linux yang didukung, lihat Model operasi dan beban kerja mana yang didukung GuardDuty Runtime Monitoring.
- Peran dan izin IAM untuk Amazon ECS:
  - Tugas Fargate harus menggunakan peran eksekusi tugas. Peran ini memberikan izin tugas untuk mengambil, memperbarui, dan mengelola agen GuardDuty keamanan atas nama Anda. Untuk mengetahui informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.
  - Anda mengontrol Runtime Monitoring untuk klaster dengan tag yang telah ditentukan sebelumnya. Jika kebijakan akses Anda membatasi akses berdasarkan tag, Anda harus

memberikan izin eksplisit kepada pengguna IAM Anda untuk menandai klaster. Untuk informasi selengkapnya, lihat tutorial IAM: Menentukan izin untuk mengakses AWS sumber daya berdasarkan tag di Panduan Pengguna IAM.

- Menghubungkan ke repositori Amazon ECR:

  Agen GuardDuty keamanan disimpan dalam repositori Amazon ECR. Setiap tugas mandiri dan layanan harus memiliki akses ke repositori. Anda dapat menggunakan salah satu opsi berikut:

  - Untuk tugas di subnet publik, Anda dapat menggunakan alamat IP publik untuk tugas tersebut, atau membuat titik akhir VPC untuk Amazon ECR di subnet tempat tugas berjalan. Untuk informasi selengkapnya, lihat Antarmuka Amazon ECR VPC endpoint AWS PrivateLink() di Panduan Pengguna Amazon Elastic Container Registry.
  - Untuk tugas di subnet pribadi, Anda dapat menggunakan gateway Network Address Translation (NAT), atau membuat titik akhir VPC untuk Amazon ECR di subnet tempat tugas berjalan.

    Untuk informasi selengkapnya, lihat Menggunakan subnet pribadi dan gateway NAT.

- Anda harus memiliki `AWSServiceRoleForAmazonGuardDuty` peran untuk GuardDuty. Untuk informasi selengkapnya, lihat Izin peran terkait layanan GuardDuty di Panduan Pengguna Amazon GuardDuty.
- File apa pun yang ingin Anda lindungi dengan Runtime Monitoring harus dapat diakses oleh pengguna root. Jika Anda mengubah izin file secara manual, Anda harus mengaturnya ke`755`.

Mengaktifkan Runtime Monitoring

Anda mengaktifkan Runtime Monitoring di GuardDuty. Untuk informasi tentang cara mengaktifkan fitur, lihat Mengaktifkan Pemantauan Runtime di GuardDuty Panduan Pengguna Amazon.

Menambahkan Runtime Monitoring ke tugas Fargate yang ada

Saat Anda mengaktifkan Runtime Monitoring, semua tugas mandiri baru, dan penerapan layanan baru di klaster akan terlindungi secara otomatis. Untuk mempertahankan kendala kekekalan, tugas yang ada tidak terpengaruh. Untuk segera melindungi tugas, Anda perlu melakukan salah satu tindakan berikut:

- Untuk tugas mandiri, hentikan tugas, lalu mulailah.
- Untuk tugas yang merupakan bagian dari layanan, perbarui layanan dengan opsi “paksa penerapan baru”.

Menghapus Runtime Monitoring dari cluster

Anda mungkin ingin mengecualikan klaster tertentu dari perlindungan, misalnya cluster yang Anda gunakan untuk pengujian. Hal ini menyebabkan GuardDuty untuk melakukan operasi berikut pada sumber daya di cluster:

- Tidak lagi menyebarkan agen GuardDuty keamanan ke tugas Fargate mandiri baru, atau penerapan layanan baru.

  Untuk mempertahankan kendala kekekalan, tugas dan penerapan yang ada dengan Runtime Monitoring diaktifkan tidak terpengaruh.
- Hentikan penagihan dan tidak lagi menerima acara run time untuk tugas.

Lakukan operasi berikut untuk menghapus Runtime Monitoring dari cluster.

1. Gunakan konsol Amazon ECS atau AWS CLI untuk menyetel kunci `GuardDutyManaged` tag pada cluster. `false` Untuk informasi selengkapnya, lihat Memperbarui klaster atau Bekerja dengan tag menggunakan CLI atau API. Gunakan nilai berikut untuk tag.

   > **Note**
   >
   > Kunci dan Nilai peka huruf besar/kecil dan harus sama persis dengan string.

   Kunci =`GuardDutyManaged`, Nilai = `false`
2. Hapus titik akhir GuardDuty VPC untuk cluster. Untuk informasi selengkapnya tentang cara menghapus titik akhir VPC, lihat Menghapus titik akhir antarmuka di Panduan Pengguna.AWS PrivateLink

Menghapus Runtime Monitoring dari akun

Saat Anda tidak lagi ingin menggunakan Runtime Monitoring, nonaktifkan fitur di GuardDuty. Untuk informasi tentang cara menonaktifkan fitur, lihat Mengaktifkan Pemantauan Runtime di GuardDuty Panduan Pengguna Amazon.

GuardDuty melakukan operasi berikut:

- Menghapus titik akhir VPC GuardDuty untuk setiap VPC yang menghosting cluster.

- Tidak lagi menyebarkan agen GuardDuty keamanan ke tugas Fargate mandiri baru, atau penerapan layanan baru.

  Untuk mempertahankan kendala kekekalan, tugas dan penerapan yang ada tidak terpengaruh sampai dihentikan, direplikasi, atau diskalakan.
- Menghentikan penagihan dan tidak lagi menerima acara run time untuk tugas.

Manajemen Pemantauan Runtime Manual

Gunakan ini saat Anda menggunakan instans EC2 untuk kapasitas Anda, atau saat Anda memerlukan kontrol granular untuk mengaktifkan Runtime Monitoring di tingkat cluster di Fargate.

Anda menyediakan kluster untuk Runtime Monitoring dengan menambahkan tag yang telah ditentukan sebelumnya.

Untuk instans kontainer EC2, Anda mengunduh, menginstal, dan mengelola agen GuardDuty keamanan.

Untuk Fargate, GuardDuty mengelola agen keamanan atas nama Anda.

Prasyarat

Berikut ini adalah prasyarat untuk menggunakan Runtime Monitoring:

- Versi platform Fargate harus `1.4.0` atau lebih baru untuk Linux.
- Peran dan izin IAM untuk Amazon ECS:
  - Tugas Fargate harus menggunakan peran eksekusi tugas. Peran ini memberikan izin tugas untuk mengambil, memperbarui, dan mengelola agen GuardDuty keamanan atas nama Anda. Untuk mengetahui informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.
  - Anda mengontrol Runtime Monitoring untuk klaster dengan tag yang telah ditentukan sebelumnya. Jika kebijakan akses Anda membatasi akses berdasarkan tag, Anda harus memberikan izin eksplisit kepada pengguna IAM Anda untuk menandai klaster. Untuk informasi selengkapnya, lihat tutorial IAM: Menentukan izin untuk mengakses AWS sumber daya berdasarkan tag di Panduan Pengguna IAM.
- Menghubungkan ke repositori Amazon ECR:

  Agen GuardDuty keamanan disimpan dalam repositori Amazon ECR. Setiap tugas mandiri dan layanan harus memiliki akses ke repositori. Anda dapat menggunakan salah satu opsi berikut:

  - Untuk tugas di subnet publik, Anda dapat menggunakan alamat IP publik untuk tugas tersebut, atau membuat titik akhir VPC untuk Amazon ECR di subnet tempat tugas berjalan. Untuk informasi selengkapnya, lihat Antarmuka Amazon ECR VPC endpoint AWS PrivateLink() di Panduan Pengguna Amazon Elastic Container Registry.
  - Untuk tugas di subnet pribadi, Anda dapat menggunakan gateway Network Address Translation (NAT), atau membuat titik akhir VPC untuk Amazon ECR di subnet tempat tugas berjalan.

    Untuk informasi selengkapnya, lihat Menggunakan subnet pribadi dan gateway NAT.
- Anda harus memiliki `AWSServiceRoleForAmazonGuardDuty` peran untuk GuardDuty. Untuk informasi selengkapnya, lihat Izin peran terkait layanan GuardDuty di Panduan Pengguna Amazon GuardDuty.
- File apa pun yang ingin Anda lindungi dengan Runtime Monitoring harus dapat diakses oleh pengguna root. Jika Anda mengubah izin file secara manual, Anda harus mengaturnya ke`755`.

Berikut ini adalah prasyarat untuk menggunakan Runtime Monitoring pada instance container EC2:

- Anda harus menggunakan versi `20230929` atau yang lebih baru dari Amazon ECS-AMI.
- Anda harus menjalankan agen Amazon ECS ke versi `1.77` atau yang lebih baru pada instance container.
- Anda harus menggunakan versi kernel `5.10` atau yang lebih baru.
- Untuk informasi tentang sistem operasi dan arsitektur Linux yang didukung, lihat Model operasi dan beban kerja mana yang didukung GuardDuty Runtime Monitoring.
- Anda dapat menggunakan Systems Manager untuk mengelola instance container Anda. Untuk informasi selengkapnya, lihat Menyiapkan Systems Manager untuk instans EC2 di AWS Systems Manager Session Manager Panduan Pengguna.

Mengaktifkan Runtime Monitoring

Anda mengaktifkan fitur ini di GuardDuty. Untuk informasi tentang cara mengaktifkan fitur, lihat Mengaktifkan Pemantauan Runtime di GuardDuty Panduan Pengguna Amazon.

Mengkonfigurasi Runtime Monitoring untuk cluster

Konfigurasikan Runtime Monitoring untuk klaster, lalu instal agen GuardDuty keamanan pada instans kontainer EC2 Anda.

Menambahkan Runtime Monitoring ke cluster

Lakukan operasi berikut untuk menambahkan Runtime Monitoring ke cluster.

1. Buat titik akhir VPC GuardDuty untuk setiap VPC cluster. Untuk informasi selengkapnya, lihat Membuat titik akhir Amazon VPC secara manual di Panduan Pengguna. GuardDuty
2. Konfigurasikan instance kontainer EC2.

   a. Perbarui agen Amazon ECS ke versi `1.77` atau yang lebih baru pada instans kontainer EC2 di cluster. Untuk mengetahui informasi selengkapnya, lihat Memperbarui agen kontainer Amazon ECS.

   b. Instal agen GuardDuty keamanan pada instans kontainer EC2 di cluster. Untuk informasi selengkapnya, lihat Mengelola agen keamanan di instans Amazon EC2 secara manual di GuardDuty Panduan Pengguna.

   Semua tugas baru dan yang sudah ada, dan penerapan segera dilindungi karena agen GuardDuty keamanan berjalan sebagai proses pada instance kontainer EC2.
3. Gunakan konsol Amazon ECS atau AWS CLI untuk menyetel kunci `GuardDutyManaged` tag pada cluster. `true` Untuk informasi selengkapnya, lihat Memperbarui klaster atau Bekerja dengan tag menggunakan CLI atau API. Gunakan nilai berikut untuk tag.

   > **Note**
   >
   > Kunci dan Nilai peka huruf besar/kecil dan harus sama persis dengan string.

   Kunci =`GuardDutyManaged`, Nilai = `true`

Menambahkan Runtime Monitoring ke tugas Fargate yang ada

Saat Anda mengaktifkan Runtime Monitoring, semua tugas mandiri baru, dan penerapan layanan baru di klaster akan terlindungi secara otomatis. Untuk mempertahankan kendala kekekalan, tugas yang ada tidak terpengaruh. Untuk segera melindungi tugas, Anda perlu melakukan salah satu tindakan berikut:

- Untuk tugas mandiri, hentikan tugas, lalu mulailah.
- Untuk tugas yang merupakan bagian dari layanan, perbarui layanan dengan opsi “paksa penerapan baru”.

Menghapus Runtime Monitoring dari cluster

Anda dapat menghapus Runtime Monitoring dari sebuah cluster. Hal ini GuardDuty menyebabkan berhenti memantau semua sumber daya di cluster.

Untuk menghapus Runtime Monitoring dari cluster.

1. Gunakan konsol Amazon ECS atau AWS CLI untuk menyetel kunci `GuardDutyManaged` tag pada cluster. `false` Untuk informasi selengkapnya, lihat Memperbarui klaster atau Bekerja dengan tag menggunakan CLI atau API.

   > **Note**
   >
   > Kunci dan Nilai peka huruf besar/kecil dan harus sama persis dengan string.

   Kunci =`GuardDutyManaged`, Nilai = `false`
2. Copot pemasangan agen GuardDuty keamanan pada instans kontainer EC2 Anda di cluster.

   Untuk informasi selengkapnya, lihat Menghapus instalan agen keamanan secara manual di Panduan GuardDuty Pengguna.
3. Hapus titik akhir GuardDuty VPC untuk setiap VPC cluster. Untuk informasi selengkapnya tentang cara menghapus titik akhir VPC, lihat Menghapus titik akhir antarmuka di Panduan Pengguna.AWS PrivateLink

Memperbarui agen GuardDuty keamanan pada instans kontainer EC2 Anda

Untuk informasi tentang cara memperbarui agen GuardDuty keamanan pada instans kontainer EC2 Anda, lihat Memperbarui agen GuardDuty keamanan di GuardDuty Panduan Pengguna Amazon.

Menghapus Runtime Monitoring dari akun

Saat Anda tidak lagi ingin menggunakan Runtime Monitoring, nonaktifkan fitur di GuardDuty. Untuk informasi tentang cara menonaktifkan fitur, lihat Mengaktifkan Pemantauan Runtime di GuardDuty Panduan Pengguna Amazon.

Hapus Runtime Monitoring dari semua cluster. Untuk informasi selengkapnya, lihat Menghapus Runtime Monitoring dari cluster.

FAQ Pemecahan Masalah Runtime Monitoring

Anda mungkin perlu memecahkan masalah atau memverifikasi bahwa Runtime Monitoring diaktifkan dan berjalan pada tugas dan container Anda.

Topik

Bagaimana cara mengetahui apakah Runtime Monitoring aktif di akun saya?

Di konsol Amazon ECS, informasinya ada di halaman Pengaturan Akun.

Anda juga dapat menjalankan `list-account-settings` dengan `effective-settings` opsi.

```
aws ecs list-account-settings --effective-settings
```

Output

Pengaturan dengan nama diatur ke `guardDutyActivate` dan nilai ditetapkan untuk on menunjukkan bahwa akun dikonfigurasi. Anda harus memeriksa dengan GuardDuty administrator Anda untuk melihat apakah pengelolaannya otomatis atau manual.

```
{
    "setting": {
        "name": "guardDutyActivate",
        "value": "enabled",
        "principalArn": "arn:aws:iam::123456789012:root",
        "type": "aws-managed"
```

```
    }
}
```

Bagaimana saya bisa tahu apakah Runtime Monitoring aktif di cluster?

Di konsol Amazon ECS, informasinya ada di tab Tag pada halaman detail Cluster.

Anda juga dapat menjalankan `describe-clusters` dengan `TAGS` opsi.

Contoh berikut menunjukkan output untuk cluster default

```
aws ecs describe-clusters --cluster default --include TAGS
```

Output

Tag dengan Key set ke `GuardDutyManaged` dan Value set untuk `true` menunjukkan bahwa cluster dikonfigurasi untuk Runtime Monitoring.

```
{
    "clusters": [
        {
            "clusterArn": "arn:aws:ecs:us-east-1:1234567890:cluster/default",
            "clusterName": "default",
            "status": "ACTIVE",
            "registeredContainerInstancesCount": 0,
            "runningTasksCount": 1,
            "pendingTasksCount": 0,
            "activeServicesCount": 0,
            "statistics": [],
            "tags": [
                {
                    "key": "GuardDutyManaged",
                    "value": "true"
                }
            ],
            "settings": [],
            "capacityProviders": [],
            "defaultCapacityProviderStrategy": []
        }
    ],
    "failures": []
}
```

Bagaimana saya bisa tahu apakah agen GuardDuty keamanan menjalankan tugas Fargate?

Agen GuardDuty keamanan berjalan sebagai wadah sespan untuk tugas Fargate.

Di konsol Amazon ECS, sespan ditampilkan di bawah Kontainer di halaman Detail tugas.

Anda dapat menjalankan `describe-tasks` dan mencari wadah dengan nama yang disetel ke `aws-gd-agent` dan LastStatus disetel ke. `RUNNING`

Contoh berikut menunjukkan output untuk cluster default untuk tugas`aws:ecs:us-east-1:123456789012:task/0b69d5c0-d655-4695-98cd-5d2d5EXAMPLE`.

```
aws ecs describe-tasks --cluster default --tasks aws:ecs:us-
east-1:123456789012:task/0b69d5c0-d655-4695-98cd-5d2d5EXAMPLE
```

Output

Wadah bernama `gd-agent` ada di `RUNNING` negara bagian.

```
"containers": [
      {
        "containerArn": "arn:aws:ecs:us-east-1:123456789012:container/4df26bb4-
f057-467b-a079-96167EXAMPLE",
        "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/0b69d5c0-
d655-4695-98cd-5d2d5EXAMPLE",
        "lastStatus": "RUNNING",
        "healthStatus": "UNKNOWN",
        "memory": "string",
        "name": "aws-gd-agent"
      }
    ]
```

Bagaimana saya bisa tahu apakah agen GuardDuty keamanan berjalan pada instance kontainer EC2?

Jalankan perintah berikut untuk melihat status:

```
sudo systemctl status amazon-guardduty-agent
```

File log berada di lokasi berikut:

```
/var/log/amzn-guardduty-agent
```

Apa yang terjadi ketika tidak ada peran eksekusi tugas untuk tugas yang berjalan di cluster?

Untuk tugas Fargate, tugas dimulai tanpa wadah sespan agen GuardDuty keamanan. GuardDuty Dasbor akan menunjukkan bahwa tugas tersebut tidak memiliki perlindungan di dasbor statistik cakupan.

Bagaimana saya bisa tahu apakah saya memiliki izin yang benar untuk menandai cluster untuk Runtime Monitoring?

Untuk menandai cluster, Anda harus memiliki `ecs:TagResource` tindakan untuk keduanya `CreateCluster` dan`UpdateCluster`.

Berikut ini adalah cuplikan dari contoh kebijakan.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
         "ecs:TagResource"
      ],
      "Resource": "*",
      "Condition": {
         "StringEquals": {
             "ecs:CreateAction" : "CreateCluster",
             "ecs:CreateAction" : "UpdateCluster",
          }
       }
    }
  ]
}
```

Apa yang terjadi ketika tidak ada koneksi Amazon ECR?

Untuk tugas Fargate, tugas dimulai tanpa wadah sespan agen GuardDuty keamanan. GuardDuty Dasbor akan menunjukkan bahwa tugas tersebut tidak memiliki perlindungan di dasbor statistik cakupan.

Bagaimana cara mengatasi kesalahan memori pada tugas Fargate saya setelah mengaktifkan Runtime Monitoring?

Agen GuardDuty keamanan adalah proses yang ringan. Namun, prosesnya masih mengkonsumsi sumber daya sesuai dengan ukuran beban kerja. Sebaiknya gunakan alat pelacak sumber daya kontainer, seperti Amazon CloudWatch Container Insights untuk mementaskan GuardDuty penerapan di klaster Anda. Alat-alat ini membantu Anda menemukan profil konsumsi agen GuardDuty keamanan untuk aplikasi Anda. Anda kemudian dapat menyesuaikan ukuran tugas Fargate Anda, jika diperlukan, untuk menghindari potensi keluar dari kondisi memori.

Pantau wadah Amazon ECS dengan ECS Exec

Dengan Amazon ECS Exec, Anda dapat langsung berinteraksi dengan kontainer tanpa perlu terlebih dahulu berinteraksi dengan sistem operasi wadah host, membuka port masuk, atau mengelola kunci SSH. Anda dapat menggunakan ECS Exec untuk menjalankan perintah di atau mendapatkan shell ke wadah yang berjalan pada instans Amazon EC2 atau di. AWS Fargate Hal tersebut mempermudah pengumpulan informasi diagnostik dan memecahkan masalah kesalahan dengan cepat. Misalnya, dalam konteks pengembangan, Anda dapat menggunakan ECS Exec untuk berinteraksi dengan mudah dengan berbagai proses dalam wadah Anda dan memecahkan masalah aplikasi Anda. Dan dalam skenario produksi, Anda dapat menggunakannya untuk mendapatkan akses break-glass ke kontainer Anda untuk men-debug masalah.

Anda dapat menjalankan perintah dalam wadah Linux atau Windows yang sedang berjalan menggunakan ECS Exec dari Amazon ECS API, AWS Command Line Interface (AWS CLI), AWS SDK, atau CLI Copilot. AWS Untuk detail tentang penggunaan ECS Exec, serta panduan video, menggunakan Copilot AWS CLI, lihat dokumentasi Copilot. GitHub

Anda juga dapat menggunakan ECS Exec untuk mempertahankan kebijakan kontrol akses yang lebih ketat dan mengaudit akses kontainer. Dengan mengaktifkan fitur ini secara selektif, Anda dapat mengendalikan siapa yang dapat menjalankan perintah dan ditugas mana mereka dapat menjalankan perintah tersebut. Dengan log dari setiap perintah dan outputnya, Anda dapat menggunakan ECS Exec untuk mengaudit tugas mana yang dijalankan dan Anda dapat menggunakan CloudTrail untuk mengaudit siapa yang mengakses wadah.

Pertimbangan untuk menggunakan ECS Exec

Untuk topik ini, Anda harus terbiasa dengan aspek-aspek berikut yang terlibat dengan penggunaan ECS Exec:

- ECS Exec saat ini tidak didukung menggunakan file. AWS Management Console
- ECS Exec didukung untuk tugas-tugas yang berjalan pada infrastruktur berikut:
  - Wadah Linux di Amazon EC2 pada AMI yang dioptimalkan Amazon ECS, termasuk Bottlerocket
  - Wadah Linux dan Windows pada instance eksternal (Amazon ECS Anywhere)
  - Wadah Linux dan Windows di AWS Fargate
  - Kontainer Windows di Amazon EC2 pada AMI yang dioptimalkan Windows Amazon ECS berikut (dengan versi `1.56` agen kontainer atau yang lebih baru):
    - Amazon ECS yang dioptimalkan Windows Server 2022 AMI Penuh
    - Server Windows 2022 AMI Inti yang dioptimalkan Amazon ECS
    - Amazon ECS yang dioptimalkan Windows Server 2019 AMI Penuh
    - Amazon ECS yang dioptimalkan Windows Server 2019 Core AMI
    - Amazon ECS yang dioptimalkan Windows Server 20H2 Core AMI
- ECS Exec dan Amazon VPC
  - Jika Anda menggunakan antarmuka titik akhir Amazon VPC dengan Amazon ECS, Anda harus membuat antarmuka titik akhir Amazon VPC untuk Systems Manager Session Manager (). `ssmmessages` Untuk informasi selengkapnya tentang titik akhir VPC Systems Manager, lihat Menggunakan AWS PrivateLink untuk menyiapkan titik akhir VPC untuk Session Manager di Panduan Pengguna.AWS Systems Manager
  - Jika Anda menggunakan antarmuka titik akhir Amazon VPC dengan Amazon ECS, dan Anda menggunakan AWS KMS key untuk enkripsi, maka Anda harus membuat antarmuka titik akhir Amazon VPC untuk. AWS KMS keyUntuk informasi selengkapnya, lihat Menghubungkan ke AWS KMS key melalui titik akhir VPC di Panduan Pengembang.AWS Key Management Service
  - Jika Anda memiliki tugas yang berjalan di instans Amazon EC2, gunakan `awsvpc` mode jaringan. Jika Anda tidak memiliki akses internet, seperti tidak dikonfigurasi untuk menggunakan gateway NAT), Anda harus membuat antarmuka titik akhir Amazon VPC untuk Systems Manager Session Manager (). `ssmmessages` Untuk informasi selengkapnya tentang pertimbangan mode `awsvpc` jaringan, lihat Pertimbangan. Untuk informasi selengkapnya tentang titik akhir VPC Systems Manager, lihat Menggunakan AWS PrivateLink untuk menyiapkan titik akhir VPC untuk Session Manager di Panduan Pengguna.AWS Systems Manager
- ECS Exec dan SSM
  - Ketika pengguna menjalankan perintah pada wadah menggunakan ECS Exec, perintah ini dijalankan sebagai pengguna. `root` Agen SSM dan proses turunannya berjalan sebagai root bahkan ketika Anda menentukan ID pengguna untuk wadah.

  - Agen SSM mengharuskan sistem file kontainer dapat ditulis untuk membuat direktori dan file yang diperlukan. Oleh karena itu, membuat sistem file root hanya-baca menggunakan parameter ketentuan tugas `readonlyRootFilesystem`, atau metode lainnya, tidak didukung.
  - Meskipun memulai sesi SSM di luar `execute-command` tindakan dimungkinkan, ini mengakibatkan sesi tidak dicatat dan dihitung terhadap batas sesi. Kami menyarankan untuk membatasi akses ini dengan menolak `ssm:start-session` tindakan menggunakan kebijakan IAM. Untuk informasi selengkapnya, lihat Membatasi akses ke tindakan Memulai Sesi.
- Fitur berikut berjalan sebagai wadah sespan. Oleh karena itu, Anda harus menentukan nama kontainer untuk menjalankan perintah.
  - Pemantauan Runtime
  - Layanan Connect
- Pengguna dapat menjalankan semua perintah yang tersedia dalam konteks kontainer. Tindakan berikut mungkin mengakibatkan proses tanpa induk dan zombie: mengakhiri proses utama kontainer, mengakhiri agen perintah, dan menghapus dependensi. Untuk membersihkan proses zombie, sebaiknya tambahkan flag `initProcessEnabled` untuk ketentuan tugas Anda.
- ECS Exec menggunakan beberapa CPU dan memori. Anda akan ingin mengakomodasinya saat menentukan alokasi sumber daya CPU dan memori dalam ketentuan tugas Anda.
- Anda harus menggunakan AWS CLI versi `1.22.3` atau yang lebih baru atau AWS CLI versi `2.3.6` atau yang lebih baru. Untuk informasi tentang cara memperbarui AWS CLI, lihat Menginstal atau memperbarui versi terbaru dari AWS CLI Panduan AWS Command Line Interface Pengguna Versi 2.
- Anda hanya dapat memiliki satu sesi ECS Exec per namespace ID proses (PID). Jika Anda berbagi namespace PID dalam tugas, Anda hanya dapat memulai sesi ECS Exec ke dalam satu wadah.
- Sesi ECS Exec memiliki waktu tunggu idle 20 menit. Nilai ini tidak dapat diubah.
- Anda tidak dapat mengaktifkan ECS Exec untuk tugas yang ada. Itu hanya dapat dihidupkan untuk tugas-tugas baru.
- Anda tidak dapat menggunakan ECS Exec saat digunakan `run-task` untuk meluncurkan tugas di klaster yang menggunakan penskalaan terkelola dengan penempatan asinkron (meluncurkan tugas tanpa instance).
- Anda tidak dapat menjalankan ECS Exec terhadap kontainer Microsoft Nano Server. Untuk informasi selengkapnya tentang kontainer Server Nano, lihat Server Nano di situs web Docker.

Prasyarat untuk menggunakan ECS Exec

Sebelum Anda mulai menggunakan ECS Exec, pastikan Anda telah menyelesaikan tindakan ini:

- Instal dan konfigurasikan AWS CLI. Untuk informasi selengkapnya, lihat AWS CLI.
- Instal plugin Session Manager untuk file AWS CLI. Untuk informasi selengkapnya, lihat Instal plugin Pengelola Sesi untuk AWS CLI.
- Anda harus menggunakan peran tugas dengan izin yang sesuai untuk ECS Exec. Untuk informasi selengkapnya, lihat Peran IAM Tugas.
- ECS Exec memiliki persyaratan versi tergantung pada apakah tugas Anda di-host di Amazon EC2 atau: AWS Fargate
  - Jika Anda menggunakan Amazon EC2, Anda harus menggunakan AMI Amazon ECS yang dioptimalkan yang dirilis setelah 20 Januari 2021, dengan versi agen 1.50.2 atau lebih tinggi. Untuk informasi selengkapnya, lihat AMI yang dioptimalkan Amazon ECS.
  - Jika Anda menggunakan AWS Fargate, Anda harus menggunakan versi platform `1.4.0` atau lebih tinggi (Linux) atau `1.0.0` (Windows). Untuk informasi selengkapnya, lihat Versi platform AWS Fargate.

Arsitektur

ECS Exec menggunakan Manajer Sesi AWS Systems Manager (SSM) untuk membuat koneksi dengan wadah yang sedang berjalan dan menggunakan kebijakan AWS Identity and Access Management (IAM) untuk mengontrol akses ke perintah yang sedang berjalan dalam wadah yang sedang berjalan. Hal ini dimungkinkan dengan mengikat pemasangan biner SSM agent yang diperlukan ke dalam kontainer. Amazon ECS atau AWS Fargate agen bertanggung jawab untuk memulai agen inti SSM di dalam wadah bersama kode aplikasi Anda. Untuk informasi selengkapnya, lihat Systems Manager Session Manager.

Anda dapat mengaudit pengguna mana yang mengakses container menggunakan `ExecuteCommand` event in AWS CloudTrail dan mencatat setiap perintah (dan outputnya) ke Amazon S3 atau Amazon CloudWatch Logs. Untuk mengenkripsi data antara klien lokal dan wadah dengan kunci enkripsi Anda sendiri, Anda harus memberikan kunci AWS Key Management Service (AWS KMS).

Menggunakan ECS Exec

Perubahan ketentuan tugas opsional

Jika Anda menyetel parameter definisi tugas `initProcessEnabled` ke`true`, ini memulai proses init di dalam wadah. Ini menghapus proses anak agen SSM zombie yang ditemukan. Contoh disediakan seperti berikut.

```
{
    "taskRoleArn": "ecsTaskRole",
    "networkMode": "awsvpc",
    "requiresCompatibilities": [
        "EC2",
        "FARGATE"
    ],
    "executionRoleArn": "ecsTaskExecutionRole",
    "memory": ".5 gb",
    "cpu": ".25 vcpu",
    "containerDefinitions": [
        {
            "name": "amazon-linux",
            "image": "amazonlinux:latest",
            "essential": true,
            "command": ["sleep","3600"],
            "linuxParameters": {
                "initProcessEnabled": true
            }
        }
    ],
    "family": "ecs-exec-task"
}
```

Mengaktifkan ECS Exec untuk tugas dan layanan Anda

Anda dapat mengaktifkan fitur ECS Exec untuk layanan dan tugas mandiri Anda dengan menentukan `--enable-execute-command` tanda saat menggunakan salah satu AWS CLI perintah berikut: `create-service`,,, `update-service`atau. `start-task``run-task`

Misalnya, jika Anda menjalankan perintah berikut, fitur ECS Exec diaktifkan untuk layanan yang baru dibuat yang berjalan di Fargate. Untuk informasi selengkapnya tentang membuat layanan, lihat buat-layanan.

```
aws ecs create-service \
    --cluster cluster-name \
    --task-definition task-definition-name \
    --enable-execute-command \
    --service-name service-name \
    --launch-type FARGATE
     --network-configuration
 "awsvpcConfiguration={subnets=[subnet-12344321],securityGroups=[sg-12344321],assignPublicIp=EN
 \
    --desired-count 1
```

Setelah Anda mengaktifkan ECS Exec untuk tugas, Anda dapat menjalankan perintah berikut untuk mengonfirmasi tugas siap digunakan. Jika properti `lastStatus` dari `ExecuteCommandAgent` terdaftar sebagai `RUNNING` dan properti `enableExecuteCommand` diatur ke `true`, maka tugas Anda sudah siap.

```
aws ecs describe-tasks \
    --cluster cluster-name \
    --tasks task-id
```

Potongan output berikut adalah contoh apa yang mungkin Anda lihat.

```
{
    "tasks": [
        {
            ...
            "containers": [
                {
                    ...
                    "managedAgents": [
                        {
                            "lastStartedAt": "2021-03-01T14:49:44.574000-06:00",
                            "name": "ExecuteCommandAgent",
                            "lastStatus": "RUNNING"
                        }
                    ]
                }
            ],
            ...
            "enableExecuteCommand": true,
            ...
        }
```

```
    ]
}
```

Menjalankan perintah menggunakan ECS Exec

Setelah Anda telah mengonfirmasi bahwa `ExecuteCommandAgent` sedang berjalan, Anda dapat membuka shell interaktif pada kontainer Anda dengan menggunakan perintah berikut. Jika tugas Anda berisi beberapa kontainer, Anda harus menentukan nama kontainer menggunakan `--container` bendera. Amazon ECS hanya mendukung memulai sesi interaktif, jadi Anda harus menggunakan bendera. `--interactive`

Perintah berikut akan menjalankan /bin/sh perintah interaktif terhadap kontainer bernama container-name untuk tugas dengan ID task-id.

Task-id adalah Nama Sumber Daya Amazon (ARN) dari tugas tersebut.

```
aws ecs execute-command --cluster cluster-name \
    --task task-id \
    --container container-name \
    --interactive \
    --command "/bin/sh"
```

Logging dan Audit menggunakan ECS Exec

Mengaktifkan pencatatan dan audit dalam tugas dan layanan Anda

> **Important**
>
> Untuk informasi selengkapnya tentang CloudWatch harga, lihat CloudWatch Harga. Amazon ECS juga menyediakan metrik pemantauan yang disediakan tanpa biaya tambahan. Untuk informasi selengkapnya, lihat Pantau Amazon ECS menggunakan CloudWatch .

Amazon ECS menyediakan konfigurasi default untuk perintah logging yang dijalankan menggunakan ECS Exec dengan mengirimkan log ke CloudWatch Log menggunakan driver `awslogs` log yang dikonfigurasi dalam definisi tugas Anda. Jika Anda ingin menyediakan konfigurasi kustom, AWS CLI mendukung flag `--configuration` untuk kedua perintah `create-cluster` dan `update-cluster`. Penting juga untuk mengetahui bahwa image kontainer memerlukan `script` dan `cat`

diinstal agar log perintah diunggah dengan benar ke Amazon S3 CloudWatch atau Log. Untuk informasi selengkapnya tentang pembuatan klaster, lihat buat-klaster.

> **Note**
>
> Konfigurasi ini hanya menangani pencatatan sesi `execute-command`. Konfigurasi tersebut tidak memengaruhi pencatatan aplikasi Anda.

Contoh berikut membuat klaster dan kemudian mencatat output ke CloudWatch Log LogGroup bernama `cloudwatch-log-group-name` dan bucket Amazon S3 Anda bernama. `s3-bucket-name`

Anda harus menggunakan kunci yang dikelola AWS KMS pelanggan untuk mengenkripsi grup log saat Anda mengatur `CloudWatchEncryptionEnabled` opsi ke`true`. Untuk informasi tentang cara mengenkripsi grup log, lihat Mengenkripsi data CloudWatch log di Log menggunakan AWS Key Management Service, di Amazon CloudWatch Logs Panduan Pengguna.

```
aws ecs create-cluster \
    --cluster-name cluster-name \
    --configuration executeCommandConfiguration="{ \
        kmsKeyId=string, \
        logging=OVERRIDE, \
        logConfiguration={ \
            cloudWatchLogGroupName=cloudwatch-log-group-name, \
            cloudWatchEncryptionEnabled=true, \
            s3BucketName=s3-bucket-name, \
            s3EncryptionEnabled=true, \
            s3KeyPrefix=demo \
        } \
    }"
```

`logging`Properti menentukan perilaku kemampuan logging ECS Exec:

- `NONE`: logging dimatikan.
- `DEFAULT`: log dikirim ke `awslogs` driver yang dikonfigurasi. Jika driver tidak dikonfigurasi, maka tidak ada log yang disimpan.
- `OVERRIDE`: log dikirim ke CloudWatch Log Amazon yang disediakan LogGroup, bucket Amazon S3, atau keduanya.

Izin IAM diperlukan untuk Amazon CloudWatch Log atau Amazon S3 Logging

Untuk mengaktifkan logging, peran tugas Amazon ECS yang direferensikan dalam definisi tugas Anda harus memiliki izin tambahan. Izin tambahan ini dapat ditambahkan sebagai kebijakan untuk peran tugas. Mereka berbeda tergantung pada apakah Anda mengarahkan log Anda ke Amazon CloudWatch Log atau Amazon S3.

Amazon CloudWatch Logs

Kebijakan contoh berikut menambahkan izin Amazon CloudWatch Logs yang diperlukan.

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
               "logs:CreateLogStream",
               "logs:DescribeLogStreams",
               "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:region:account-id:log-group:/aws/
ecs/cloudwatch-log-group-name:*"
        }
   ]
}
```

Amazon S3

Kebijakan contoh berikut menambahkan izin Amazon S3 yang diperlukan.

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
               "s3:GetBucketLocation"
            ],
            "Resource": "*"
        },
        {
           "Effect": "Allow",
           "Action": [
               "s3:GetEncryptionConfiguration"
           ],
           "Resource": "arn:aws:s3:::s3-bucket-name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::s3-bucket-name/*"
        }
    ]
   }
```

Izin IAM diperlukan untuk enkripsi menggunakan milik Anda sendiri AWS KMS key (kunci KMS)

Secara default, data yang ditransfer antara klien lokal Anda dan penampung menggunakan enkripsi TLS 1.2 yang AWS menyediakan. Untuk mengenkripsi data lebih lanjut menggunakan kunci KMS Anda sendiri, Anda harus membuat kunci KMS dan menambahkan `kms:Decrypt` izin ke peran IAM tugas Anda. Izin ini digunakan oleh kontainer Anda untuk mendekripsi data. Untuk informasi selengkapnya tentang membuat kunci KMS, lihat Membuat kunci.

Anda menambahkan kebijakan sebaris berikut ke peran IAM tugas yang memerlukan izin. AWS KMS Untuk informasi selengkapnya, lihat Izin IAM diperlukan untuk ECS Exec.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
```

```
            ],
            "Resource": "kms-key-arn"
        }
    ]
}
```

Agar data dienkripsi menggunakan kunci KMS Anda sendiri, pengguna atau grup yang menggunakan `execute-command` tindakan harus diberikan izin. `kms:GenerateDataKey`

Contoh kebijakan berikut untuk pengguna atau grup Anda berisi izin yang diperlukan untuk menggunakan kunci KMS Anda sendiri. Anda harus menentukan Nama Sumber Daya Amazon (ARN) dari kunci KMS Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey"
            ],
            "Resource": "kms-key-arn"
        }
    ]
}
```

Menggunakan kebijakan IAM untuk membatasi akses ke ECS Exec

Anda membatasi akses pengguna ke tindakan API perintah eksekusi dengan menggunakan satu atau beberapa kunci kondisi kebijakan IAM berikut:

- `aws:ResourceTag/clusterTagKey`
- `ecs:ResourceTag/clusterTagKey`
- `aws:ResourceTag/taskTagKey`
- `ecs:ResourceTag/taskTagKey`
- `ecs:container-name`
- `ecs:cluster`
- `ecs:task`
- `ecs:enable-execute-command`

Dengan contoh berikut kebijakan IAM, pengguna dapat menjalankan perintah dalam wadah yang berjalan dalam tugas dengan tag yang memiliki `environment` kunci dan `development` nilai dan dalam klaster yang diberi `cluster-name` nama.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:ExecuteCommand",
                "ecs:DescribeTasks"
            ],
         "Resource": [
                "arn:aws:ecs:region:aws-account-id:task/cluster-name/*",
                "arn:aws:ecs:region:aws-account-id:cluster/*"
            ],
            "Condition": {
                "StringEquals": {
                    "ecs:ResourceTag/environment": "development"
                }
            }
        }
    ]
}
```

Dengan contoh kebijakan IAM berikut, pengguna tidak dapat menggunakan `execute-command` API saat nama kontainer. `production-app`

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "ecs:ExecuteCommand"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:container-name": "production-app"
                }
```

```
            }
        }
    ]
}
```

Dengan kebijakan IAM berikut, pengguna hanya dapat meluncurkan tugas ketika ECS Exec dimatikan.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask",
                "ecs:StartTask",
                "ecs:CreateService",
                "ecs:UpdateService"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:enable-execute-command": "false"
                }
            }
        }
    ]
}
```

> **Note**
>
> Karena tindakan API `execute-command` hanya berisi sumber daya tugas dan klaster dalam sebuah permintaan, maka yang dievaluasi adalah tanda klaster dan tugas.

Untuk informasi selengkapnya tentang kunci kondisi kebijakan IAM, lihat Kunci tindakan, sumber daya, dan kondisi untuk Amazon Elastic Container Service di Referensi Otorisasi Layanan.

Membatasi akses ke tindakan Memulai Sesi

Meskipun memulai sesi SSM di wadah Anda di luar ECS Exec dimungkinkan, ini berpotensi mengakibatkan sesi tidak dicatat. Sesi yang dimulai di luar ECS Exec juga dihitung terhadap kuota sesi. Sebaiknya batasi akses ini dengan menolak `ssm:start-session` tindakan secara langsung untuk tugas Amazon ECS Anda menggunakan kebijakan IAM. Anda dapat menolak akses ke semua tugas Amazon ECS atau ke tugas tertentu berdasarkan tag yang digunakan.

Berikut ini adalah contoh kebijakan IAM yang menolak akses ke `ssm:start-session` tindakan untuk tugas di semua Wilayah dengan nama cluster tertentu. Anda dapat menyertakan wildcard dengan *`cluster-name`* secara opsional.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "ssm:StartSession",
            "Resource": [
                "arn:aws:ecs:region:aws-account-id:task/cluster-name/*",
                "arn:aws:ecs:region:aws-account-id:cluster/*"
            ]
        }
    ]
}
```

Berikut ini adalah contoh kebijakan IAM yang menolak akses ke `ssm:start-session` tindakan pada sumber daya di semua Wilayah yang ditandai dengan kunci tag `Task-Tag-Key` dan nilai tag. `Exec-Task`

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "ssm:StartSession",
            "Resource": "arn:aws:ecs:*:*:task/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Task-Tag-Key": "Exec-Task"
                }
```

```
            }
        }
    ]
}
```

Untuk bantuan terkait masalah apa pun yang mungkin Anda hadapi saat menggunakan Amazon ECS Exec, lihat Memecahkan masalah dengan Exec.

AWS Compute Optimizer rekomendasi untuk Amazon ECS

AWS Compute Optimizer menghasilkan rekomendasi untuk tugas Amazon ECS dan ukuran wadah. Untuk informasi selengkapnya, lihat Apa itu AWS Compute Optimizer? dalam Panduan Pengguna AWS Compute Optimizer .

Rekomendasi ukuran tugas dan wadah untuk layanan Amazon ECS di AWS Fargate

AWS Compute Optimizer menghasilkan rekomendasi untuk layanan Amazon ECS di AWS Fargate. AWS Compute Optimizer merekomendasikan CPU tugas dan ukuran memori tugas dan CPU kontainer, memori kontainer dan ukuran reservasi memori kontainer. Rekomendasi ini ditampilkan di halaman berikut dari konsol Compute Optimizer.

- Rekomendasi untuk layanan Amazon ECS di halaman Fargate
- Layanan Amazon ECS di halaman detail Fargate

Untuk informasi selengkapnya, lihat Melihat rekomendasi untuk layanan Amazon ECS di Fargate di Panduan Pengguna AWS Compute Optimizer .

Keamanan di Amazon Elastic Container Service

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model tanggung jawab bersama menjelaskan hal ini sebagai keamanan dari cloud dan keamanan di cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Auditor pihak ketiga secara berkala menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari AWS program kepatuhan. Untuk mempelajari tentang program kepatuhan yang berlaku untuk Amazon Elastic Container Service, lihat AWS Layanan dalam Lingkup menurut Program Kepatuhan.
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, yang mencakup sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan Amazon ECS. Topik berikut menunjukkan cara mengonfigurasi Amazon ECS untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga mempelajari cara menggunakan AWS layanan lain yang membantu Anda memantau dan mengamankan sumber daya Amazon ECS Anda.

Topik

Identity and Access Management untuk Amazon Elastic Container Service

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diotorisasi (memiliki izin) untuk menggunakan sumber daya Amazon ECS. IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di Amazon ECS.

Pengguna layanan - Jika Anda menggunakan layanan Amazon ECS untuk melakukan pekerjaan Anda, administrator Anda memberi Anda kredensyal dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak fitur Amazon ECS untuk melakukan pekerjaan Anda, Anda mungkin memerlukan izin tambahan. Memahami cara akses dikelola dapat membantu Anda meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur di Amazon ECS, lihatMemecahkan masalah identitas dan akses Amazon Elastic Container Service.

Administrator layanan - Jika Anda bertanggung jawab atas sumber daya Amazon ECS di perusahaan Anda, Anda mungkin memiliki akses penuh ke Amazon ECS. Tugas Anda adalah menentukan fitur dan sumber daya Amazon ECS mana yang harus diakses pengguna layanan Anda. Kemudian, Anda harus mengirimkan permintaan kepada administrator IAM Anda untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep Basic IAM. Untuk mempelajari lebih lanjut tentang bagaimana perusahaan Anda dapat menggunakan IAM dengan Amazon ECS, lihat. Bagaimana Amazon Elastic Container Service bekerja dengan IAM

Administrator IAM - Jika Anda administrator IAM, Anda mungkin ingin mempelajari detail tentang cara menulis kebijakan untuk mengelola akses ke Amazon ECS. Untuk melihat contoh kebijakan berbasis identitas Amazon ECS yang dapat Anda gunakan di IAM, lihat. Contoh kebijakan berbasis identitas untuk Amazon Elastic Container Service

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensyal identitas Anda. Anda harus diautentikasi (masuk ke AWS) sebagai Pengguna root akun AWS, sebagai pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk AWS sebagai identitas federasi dengan menggunakan kredensyal yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (IAM Identity Center), autentikasi masuk tunggal perusahaan Anda, dan kredensyal Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas gabungan, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal AWS akses. Untuk informasi selengkapnya tentang masuk AWS, lihat Cara masuk ke Panduan AWS Sign-In Pengguna Anda Akun AWS.

Jika Anda mengakses AWS secara terprogram, AWS sediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensyal Anda. Jika Anda tidak menggunakan AWS alat, Anda harus menandatangani permintaan sendiri. Untuk informasi selengkapnya tentang penggunaan metode yang disarankan untuk menandatangani permintaan sendiri, lihat Menandatangani permintaan AWS API di Panduan Pengguna IAM.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Misalnya, AWS merekomendasikan agar Anda menggunakan otentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari selengkapnya, lihat Autentikasi multi-faktor dalam Panduan Pengguna AWS IAM Identity Center dan Menggunakan autentikasi multi-faktor (MFA) di AWS dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya di akun. Identitas ini disebut pengguna Akun AWS root dan diakses dengan masuk dengan alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari Anda. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar lengkap tugas yang mengharuskan Anda masuk sebagai pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Identitas terfederasi

Sebagai praktik terbaik, mewajibkan pengguna manusia, termasuk pengguna yang memerlukan akses administrator, untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS dengan menggunakan kredensyal sementara.

Identitas federasi adalah pengguna dari direktori pengguna perusahaan Anda, penyedia identitas web, direktori Pusat Identitas AWS Directory Service, atau pengguna mana pun yang mengakses Layanan AWS dengan menggunakan kredensyal yang disediakan melalui sumber identitas. Ketika identitas federasi mengakses Akun AWS, mereka mengambil peran, dan peran memberikan kredensyal sementara.

Untuk manajemen akses terpusat, kami sarankan Anda menggunakan AWS IAM Identity Center. Anda dapat membuat pengguna dan grup di Pusat Identitas IAM, atau Anda dapat menghubungkan dan menyinkronkan ke sekumpulan pengguna dan grup di sumber identitas Anda sendiri untuk digunakan di semua aplikasi Akun AWS dan aplikasi Anda. Untuk informasi tentang Pusat Identitas IAM, lihat Apa itu Pusat Identitas IAM? dalam Panduan Pengguna AWS IAM Identity Center .

Pengguna dan grup IAM

Pengguna IAM adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, sebaiknya andalkan kredensial sementara daripada membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan khusus yang memerlukan kredensial jangka panjang dengan pengguna IAM, kami sarankan Anda merotasi kunci akses. Untuk informasi selengkapnya, lihat Rotasikan kunci akses secara rutin untuk kasus penggunaan yang memerlukan kredensial jangka panjang dalam Panduan Pengguna IAM.

Grup IAM adalah identitas yang menentukan kumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin untuk beberapa pengguna sekaligus. Grup membuat izin lebih mudah dikelola untuk sekelompok besar pengguna. Misalnya, Anda dapat memiliki grup yang bernama IAMAdmins dan memberikan izin kepada grup tersebut untuk mengelola sumber daya IAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, silakan lihat Kapan harus membuat pengguna IAM (bukan peran) dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Anda dapat mengambil peran IAM untuk sementara AWS Management Console dengan beralih peran. Anda dapat mengambil peran dengan memanggil operasi AWS CLI atau AWS API atau dengan menggunakan URL kustom. Untuk informasi selengkapnya tentang cara menggunakan peran, lihat Menggunakan peran IAM dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna gabungan – Untuk menetapkan izin ke sebuah identitas gabungan, Anda dapat membuat peran dan menentukan izin untuk peran tersebut. Saat identitas terfederasi mengautentikasi, identitas tersebut akan dikaitkan dengan peran dan diberi izin yang ditentukan oleh peran tersebut. Untuk informasi tentang peran-peran untuk federasi, lihat Membuat peran untuk Penyedia Identitas pihak ketiga dalam Panduan Pengguna IAM. Jika Anda menggunakan Pusat Identitas IAM, Anda perlu mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM mengorelasikan izin yang diatur ke peran dalam IAM. Untuk informasi tentang rangkaian izin, lihat Rangkaian izin dalam Panduan Pengguna AWS IAM Identity Center .
- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM guna mendapatkan berbagai izin secara sementara untuk tugas tertentu.
- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (pengguna utama tepercaya) dengan akun berbeda untuk mengakses sumber daya yang ada di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa Layanan AWS, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara kebijakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Bagaimana peran IAM berbeda dari kebijakan berbasis sumber daya dalam Panduan Pengguna IAM.
- Akses lintas layanan — Beberapa Layanan AWS menggunakan fitur lain Layanan AWS. Sebagai contoh, ketika Anda melakukan panggilan dalam suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Sebuah layanan mungkin melakukannya menggunakan izin prinsipal yang memanggil, menggunakan peran layanan, atau peran terkait layanan.
  - Sesi akses teruskan (FAS) — Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Saat Anda menggunakan beberapa layanan, Anda mungkin melakukan tindakan yang kemudian memulai tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan saat membuat permintaan FAS, lihat Teruskan sesi akses.
  - Peran layanan – Peran layanan adalah peran IAM yang diambil oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, memodifikasi, dan menghapus

peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

- Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke peran layanan. Layanan AWS Layanan dapat menggunakan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

- Aplikasi yang berjalan di Amazon EC2 — Anda dapat menggunakan peran IAM untuk mengelola kredensyal sementara untuk aplikasi yang berjalan pada instans EC2 dan membuat atau permintaan API. AWS CLI AWS Cara ini lebih dianjurkan daripada menyimpan kunci akses dalam instans EC2. Untuk menetapkan AWS peran ke instans EC2 dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instans berisi peran dan memungkinkan program yang berjalan di instans EC2 mendapatkan kredensial sementara. Untuk informasi selengkapnya, lihat Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan di instans Amazon EC2 dalam Panduan Pengguna IAM.

Untuk mempelajari apakah kita harus menggunakan peran IAM atau pengguna IAM, lihat Kapan harus membuat peran IAM (bukan pengguna) dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan adalah objek AWS yang, ketika dikaitkan dengan identitas atau sumber daya, menentukan izinnya. AWS mengevaluasi kebijakan ini ketika prinsipal (pengguna, pengguna root, atau sesi peran) membuat permintaan. Izin dalam kebijakan dapat menentukan permintaan yang diizinkan atau ditolak. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang struktur dan konten dokumen kebijakan JSON, lihat Gambaran umum kebijakan JSON dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, pengguna utama manakah yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat menjalankan peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk operasi. Sebagai contoh, anggap saja Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut bisa mendapatkan informasi peran dari AWS Management Console, API AWS CLI, atau AWS API.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan pengguna dan peran, di sumber daya mana, dan dengan ketentuan apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan terkelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran dalam. Akun AWS Kebijakan AWS terkelola mencakup kebijakan terkelola dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan terkelola atau kebijakan inline, lihat Memilih antara kebijakan terkelola dan kebijakan inline dalam Panduan Pengguna IAM.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya yang dilampiri kebijakan tersebut, kebijakan ini menentukan jenis tindakan yang dapat dilakukan oleh pengguna utama tertentu di sumber daya tersebut dan apa ketentuannya. Anda harus menentukan pengguna utama dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

Daftar kontrol akses (ACL)

Daftar kontrol akses (ACL) mengendalikan pengguna utama mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL sama dengan kebijakan berbasis sumber daya, meskipun tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung ACL. Untuk mempelajari ACL selengkapnya, silakan lihat Gambaran umum daftar kontrol akses (ACL) di Panduan Developer Layanan Penyimpanan Ringkas Amazon.

Tipe kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Tipe-tipe kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda berdasarkan tipe kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan di mana Anda menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM (pengguna atau peran IAM). Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan secara eksplisit terhadap salah satu kebijakan ini akan mengesampingkan izin tersebut. Untuk informasi selengkapnya tentang batasan izin, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCP) — SCP adalah kebijakan JSON yang menentukan izin maksimum untuk organisasi atau unit organisasi (OU) di. AWS Organizations AWS Organizations adalah layanan untuk mengelompokkan dan mengelola secara terpusat beberapa Akun AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur di sebuah organisasi, maka Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke salah satu atau semua akun Anda. SCP membatasi izin untuk entitas di akun anggota, termasuk masing-masing. Pengguna root akun AWS Untuk informasi selengkapnya tentang Organisasi dan SCP, lihat Cara kerja SCP dalam Panduan Pengguna AWS Organizations .
- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda teruskan sebagai parameter saat Anda membuat sesi sementara secara terprogram untuk peran atau pengguna gabungan. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan eksplisit dalam salah satu kebijakan ini membatalkan izin. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Beberapa jenis kebijakan

Ketika beberapa jenis kebijakan berlaku untuk sebuah permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat Logika evaluasi kebijakan di Panduan Pengguna IAM.

Bagaimana Amazon Elastic Container Service bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke Amazon ECS, pelajari fitur IAM apa yang tersedia untuk digunakan dengan Amazon ECS.

Fitur IAM yang dapat Anda gunakan dengan Amazon Elastic Container Service

Fitur IAM	Dukungan Amazon ECS
Kebijakan berbasis identitas	Ya
Kebijakan berbasis sumber daya	Tidak
Tindakan kebijakan	Ya
Sumber daya kebijakan	Parsial
Kunci persyaratan kebijakan	Ya
ACL	Tidak
ABAC (tanda dalam kebijakan)	Ya
Kredensial sementara	Ya
Sesi akses teruskan (FAS)	Ya
Peran layanan	Ya
Peran terkait layanan	Ya

Untuk mendapatkan tampilan tingkat tinggi tentang cara kerja Amazon ECS dan AWS layanan lainnya dengan sebagian besar fitur IAM, lihat AWS layanan yang bekerja dengan IAM di Panduan Pengguna IAM.

Kebijakan berbasis identitas untuk Amazon ECS

Mendukung kebijakan berbasis identitas	Ya

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan pengguna dan peran, di sumber daya mana, dan dengan ketentuan apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan tindakan dan sumber daya yang diizinkan atau ditolak, serta ketentuan terkait jenis tindakan yang diizinkan atau ditolak. Anda tidak dapat menentukan pengguna utama dalam kebijakan berbasis identitas karena kebijakan ini berlaku untuk pengguna atau peran yang dilampiri kebijakan. Untuk mempelajari semua elemen yang dapat digunakan dalam kebijakan JSON, lihat Referensi elemen kebijakan JSON IAM dalam Panduan Pengguna IAM.

Contoh kebijakan berbasis identitas untuk Amazon ECS

Untuk melihat contoh kebijakan berbasis identitas Amazon ECS, lihat. Contoh kebijakan berbasis identitas untuk Amazon Elastic Container Service

Kebijakan berbasis sumber daya dalam Amazon ECS

Mendukung kebijakan berbasis sumber daya	Tidak

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya yang dilampiri kebijakan tersebut, kebijakan ini menentukan jenis tindakan yang dapat dilakukan oleh pengguna utama tertentu di sumber daya tersebut dan apa ketentuannya. Anda harus menentukan pengguna utama dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Untuk mengaktifkan akses lintas akun, Anda dapat menentukan seluruh akun atau entitas IAM di akun lain sebagai pengguna utama dalam kebijakan berbasis sumber daya. Menambahkan pengguna utama lintas akun ke kebijakan berbasis sumber daya bagian dari membangun hubungan kepercayaan. Ketika prinsipal dan sumber daya berbeda Akun AWS, administrator IAM di akun tepercaya juga harus memberikan izin entitas utama (pengguna atau peran) untuk mengakses sumber daya. Izin diberikan dengan melampirkan kebijakan berbasis identitas ke entitas tersebut.

Namun, jika kebijakan berbasis sumber daya memberikan akses kepada pengguna utama dalam akun yang sama, kebijakan berbasis identitas lainnya tidak diperlukan. Untuk informasi selengkapnya, lihat Perbedaan peran IAM dengan kebijakan berbasis sumber daya di Panduan Pengguna IAM.

Tindakan kebijakan untuk Amazon ECS

Mendukung tindakan kebijakan	Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, pengguna utama mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Tindakan kebijakan biasanya memiliki nama yang sama dengan operasi AWS API terkait. Ada beberapa pengecualian, misalnya tindakan hanya izin yang tidak memiliki operasi API yang cocok. Ada juga beberapa operasi yang memerlukan beberapa tindakan dalam suatu kebijakan. Tindakan tambahan ini disebut tindakan dependen.

Menyertakan tindakan dalam suatu kebijakan untuk memberikan izin melakukan operasi terkait.

Untuk melihat daftar tindakan Amazon ECS, lihat Tindakan yang ditentukan oleh Amazon Elastic Container Service di Referensi Otorisasi Layanan.

Tindakan kebijakan di Amazon ECS menggunakan awalan berikut sebelum tindakan:

```
ecs
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan-tindakan tersebut dengan koma.

```
"Action": [
      "ecs:action1",
      "ecs:action2"
         ]
```

Anda juga dapat menentukan beberapa tindakan menggunakan wildcard (*). Sebagai contoh, untuk menentukan semua tindakan yang dimulai dengan kata `Describe`, sertakan tindakan berikut:

```
"Action": "ecs:Describe*"
```

Untuk melihat contoh kebijakan berbasis identitas Amazon ECS, lihat. Contoh kebijakan berbasis identitas untuk Amazon Elastic Container Service

Sumber daya kebijakan untuk Amazon ECS

Mendukung sumber daya kebijakan	Parsial

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, pengguna utama mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Elemen kebijakan JSON `Resource` menentukan objek atau beberapa objek yang menjadi target penerapan tindakan. Pernyataan harus menyertakan elemen `Resource` atau `NotResource`. Praktik terbaiknya, tentukan sumber daya menggunakan Amazon Resource Name (ARN). Anda dapat melakukan ini untuk tindakan yang mendukung jenis sumber daya tertentu, yang dikenal sebagai izin tingkat sumber daya.

Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, misalnya operasi pencantuman, gunakan wildcard (*) untuk mengindikasikan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*"
```

Untuk melihat daftar jenis sumber daya Amazon ECS dan ARNnya, lihat Sumber daya yang ditentukan oleh Amazon Elastic Container Service di Referensi Otorisasi Layanan. Untuk mempelajari tindakan mana yang dapat Anda tentukan ARN dari setiap sumber daya, lihat Tindakan yang ditentukan oleh Amazon Elastic Container Service.

Beberapa tindakan Amazon ECS API mendukung banyak sumber daya. Misalnya, beberapa klaster dapat direferensikan ketika memanggil tindakan API `DescribeClusters`. Untuk menentukan beberapa sumber daya dalam pernyataan tunggal, pisahkan ARN dengan koma.

```
"Resource": [
      "EXAMPLE-RESOURCE-1",
      "EXAMPLE-RESOURCE-2"
```

Misalnya, sumber daya cluster Amazon ECS memiliki ARN berikut:

```
arn:${Partition}:ecs:${Region}:${Account}:cluster/${clusterName}
```

Untuk menentukan `my-cluster-1` dan `my-cluster-2` mengelompokkan dalam pernyataan Anda, gunakan ARN berikut:

```
"Resource": [
        "arn:aws:ecs:us-east-1:123456789012:cluster/my-cluster-1",
        "arn:aws:ecs:us-east-1:123456789012:cluster/my-cluster-2"
```

Untuk menentukan semua klaster milik akun tertentu, gunakan wildcard (*):

```
"Resource": "arn:aws:ecs:us-east-1:123456789012:cluster/*"
```

Untuk definisi tugas, Anda dapat menentukan revisi terbaru, atau revisi tertentu.

Untuk menentukan definisi tugas terbaru, gunakan:

```
"Resource:arn:${Partition}:ecs:${Region}:${Account}:task-definition/
${TaskDefinitionFamilyName}"
```

Untuk menentukan revisi definisi tugas tertentu, gunakan $ {TaskDefinitionRevisionNumber}:

```
"Resource:arn:${Partition}:ecs:${Region}:${Account}:task-definition/
${TaskDefinitionFamilyName}:${TaskDefinitionRevisionNumber}"
```

Untuk melihat contoh kebijakan berbasis identitas Amazon ECS, lihat. Contoh kebijakan berbasis identitas untuk Amazon Elastic Container Service

Kunci kondisi kebijakan untuk Amazon ECS

Mendukung kunci kondisi kebijakan spesifik layanan	Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, pengguna utama mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Elemen `Condition` (atau blok `Condition`) memungkinkan Anda menentukan kondisi di mana suatu pernyataan akan diterapkan. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi kondisional yang menggunakan operator kondisi, misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta.

Jika Anda menentukan beberapa elemen `Condition` dalam satu pernyataan, atau beberapa kunci dalam satu elemen `Condition`, AWS akan mengevaluasinya dengan menggunakan operasi `AND` logis. Jika Anda menentukan beberapa nilai untuk satu kunci kondisi, AWS mengevaluasi kondisi menggunakan `OR` operasi logis. Semua kondisi harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan kondisi. Misalnya, Anda dapat memberikan izin kepada pengguna IAM untuk mengakses sumber daya hanya jika izin tersebut mempunyai tag yang sesuai dengan nama pengguna IAM mereka. Untuk informasi selengkapnya, lihat Elemen kebijakan IAM: variabel dan tag di Panduan Pengguna IAM.

AWS mendukung kunci kondisi global dan kunci kondisi khusus layanan. Untuk melihat semua kunci kondisi AWS global, lihat kunci konteks kondisi AWS global di Panduan Pengguna IAM.

Amazon ECS mendukung kunci kondisi khusus layanan berikut yang dapat Anda gunakan untuk menyediakan pemfilteran halus untuk kebijakan IAM Anda:

Kunci Syarat	Deskripsi	Jenis Evaluasi
aws: RequestTa g /$ {} TagKey	Kunci konteks diformat `"aws:RequestTag/ tag-key":"tag-value "` tempat *`kunci-tanda`* dan *`nilai-tanda`* adalah kunci tanda dan pasangan nilai. Memeriksa apakah pasangan kunci tag — nilai ada dalam permintaan. AWS Misalnya, Anda dapat memeriksa apakah permintaan tersebut beserta kunci tanda `"Dept"` dan memiliki nilai `"Accounting"` .	Rangkaian
aws: ResourceT ag /$ {} TagKey	Kunci konteks diformat `"aws:ResourceTag/ tag-key":"tag-value "` tempat *`kunci tanda`* dan *`nilai tanda`* adalah kunci tanda dan pasangan nilai.	String

Kunci Syarat	Deskripsi	Jenis Evaluasi
	Periksa apakah tanda yang dilampirkan ke sumber daya identitas (pengguna atau peran) cocok dengan nama dan nilai kunci yang ditentukan.	
aws: TagKeys	Kunci konteks ini diformat "`aws:TagKeys":"` *`tag-key`*" dengan *`key-tag`* adalah daftar kunci tanda tanpa nilai (misalnya, `["Dept","Cost-Center"]`). Memeriksa kunci tag yang ada dalam AWS permintaan.	String
ecs: ResourceTag /$ {} TagKey	Kunci konteks diformat "`ecs:ResourceTag/` *`tag-key`*":"*`tag-value`* " tempat *`kunci tanda`* dan *`nilai tanda`* adalah kunci tanda dan pasangan nilai. Periksa apakah tanda yang dilampirkan ke sumber daya identitas (pengguna atau peran) cocok dengan nama dan nilai kunci yang ditentukan.	Rangkaian
ecs:klaster	Kunci konteks diformat "`ecs:cluster":"` *`cluster-arn`* " di mana *`cluster-arn adalah ARN`* untuk cluster Amazon ECS.	ARN, Null
ecs:instans kontainer	Kunci konteks diformat "`ecs:container-instances":"` *`container-instance-arns`* " di mana *`container-instance-arns`* satu atau lebih ARN instance kontainer.	ARN, Null
ecs:nama-kontainer	Kunci konteks diformat "`ecs:container-name":"` *`container-name`* " di mana *`container-instance-`* adalah nama wadah Amazon ECS yang didefinisikan dalam definisi tugas.	String
ecs: enable-execute-command	Kunci konteks diformat "`ecs:enable-execute-command":"` *`value`*" di mana *`nilai-`* adalah "" atau `true` "`false`”.	String

Kunci Syarat	Deskripsi	Jenis Evaluasi
ecs: enable-service-connect	Kunci konteks diformat "ecs:enable-service-connect":" *value*" di mana *nilai* adalah "true" atau"false".	String
ecs: enable-ebs-volumes	Kunci konteks diformat "ecs:enable-ebs-volumes":" *value*" di mana *nilai* adalah "true" atau"false".	String
ecs:namespace	Kunci konteks diformat "ecs:namespace":" *namespace-arn* " di mana *namespace-arn adalah ARN* untuk namespace. AWS Cloud Map	ARN, Null
ecs:layanan	Kunci konteks diformat "ecs:service":" *service-arn* " di mana *service-arn adalah ARN* untuk layanan Amazon ECS.	ARN, Null
ecs:ketentuan tugas-definis	Kunci konteks diformat "ecs:task-definition":" *task-definition-arn* " di mana *task-definition-arn* ARN untuk definisi tugas Amazon ECS.	ARN, Null
ecs:pengaturan akun	Kunci konteks diformat "ecs:account-setting":" *account-setting* " di mana pengaturan *akun adalah nama pengaturan* akun Amazon ECS.	String

Untuk melihat daftar kunci kondisi Amazon ECS, lihat Kunci kondisi untuk Amazon Elastic Container Service di Referensi Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat Tindakan yang ditentukan oleh Amazon Elastic Container Service.

Untuk melihat contoh kebijakan berbasis identitas Amazon ECS, lihat. Contoh kebijakan berbasis identitas untuk Amazon Elastic Container Service

Daftar kontrol akses (ACL) di Amazon ECS

Mendukung ACL	Tidak

Daftar kontrol akses (ACL) mengontrol pengguna utama (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL sama dengan kebijakan berbasis sumber daya, meskipun tidak menggunakan format dokumen kebijakan JSON.

Kontrol akses berbasis atribut (ABAC) dengan Amazon ECS

> **Important**
>
> Amazon ECS mendukung kontrol akses berbasis atribut untuk semua sumber daya Amazon ECS. Untuk menentukan apakah Anda dapat menggunakan atribut untuk cakupan tindakan, gunakan tabel Tindakan yang ditentukan oleh Amazon ECS di Referensi Otorisasi Layanan. Pertama verifikasi bahwa ada sumber daya di kolom Sumber Daya. Kemudian, gunakan kolom tombol Kondisi untuk melihat kunci untuk kombinasi tindakan/sumber daya.

Mendukung ABAC (tanda dalam kebijakan)	Ya

Kontrol akses berbasis atribut (ABAC) adalah strategi otorisasi yang menentukan izin berdasarkan atribut. Dalam AWS, atribut ini disebut tag. Anda dapat melampirkan tag ke entitas IAM (pengguna atau peran) dan ke banyak AWS sumber daya. Pemberian tanda ke entitas dan sumber daya adalah langkah pertama dari ABAC. Kemudian, rancanglah kebijakan ABAC untuk mengizinkan operasi saat tag milik pengguna utama cocok dengan tag yang ada di sumber daya yang ingin diakses.

ABAC sangat berguna di lingkungan yang berkembang dengan cepat dan berguna di situasi di mana pengelolaan kebijakan menjadi rumit.

Untuk mengendalikan akses berdasarkan tag, berikan informasi tentang tag di elemen kondisi dari kebijakan dengan menggunakan kunci kondisi `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`.

Jika sebuah layanan mendukung ketiga kunci kondisi untuk setiap jenis sumber daya, nilainya adalah Ya untuk layanan tersebut. Jika suatu layanan mendukung ketiga kunci kondisi hanya untuk beberapa jenis sumber daya, nilainya adalah Parsial.

Untuk informasi selengkapnya tentang ABAC, lihat Apa itu ABAC? di Panduan Pengguna IAM. Untuk melihat tutorial terkait langkah-langkah penyiapan ABAC, lihat Menggunakan kontrol akses berbasis atribut (ABAC) di Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang menandai sumber daya Amazon ECS, lihat. Penandaan sumber daya Amazon ECS

Untuk melihat contoh kebijakan berbasis identitas untuk membatasi akses ke sumber daya berdasarkan tag pada sumber daya tersebut, lihat Menjelaskan layanan Amazon ECS berdasarkan tag.

Menggunakan kredensyal Sementara dengan Amazon ECS

Mendukung kredensial sementara	Ya

Beberapa Layanan AWS tidak berfungsi saat Anda masuk menggunakan kredensyal sementara. Untuk informasi tambahan, termasuk yang Layanan AWS bekerja dengan kredensyal sementara, lihat Layanan AWS yang bekerja dengan IAM di Panduan Pengguna IAM.

Anda menggunakan kredensi sementara jika Anda masuk AWS Management Console menggunakan metode apa pun kecuali nama pengguna dan kata sandi. Misalnya, ketika Anda mengakses AWS menggunakan tautan masuk tunggal (SSO) perusahaan Anda, proses tersebut secara otomatis membuat kredensil sementara. Anda juga akan membuat kredensial sementara secara otomatis saat masuk ke konsol sebagai pengguna dan kemudian beralih peran. Untuk informasi selengkapnya tentang cara beralih peran, lihat Beralih peran (konsol) di Panduan Pengguna IAM.

Anda dapat membuat kredensil sementara secara manual menggunakan API AWS CLI atau AWS . Anda kemudian dapat menggunakan kredensyal sementara tersebut untuk mengakses. AWS AWS merekomendasikan agar Anda menghasilkan kredensyal sementara secara dinamis alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, lihat Kredensial keamanan sementara di IAM.

Teruskan sesi akses untuk Amazon ECS

Mendukung sesi akses maju (FAS)	Ya

Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Saat Anda menggunakan beberapa layanan, Anda mungkin melakukan tindakan yang kemudian memulai tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima

permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan saat membuat permintaan FAS, lihat Teruskan sesi akses.

Peran layanan untuk Amazon ECS

Mendukung peran layanan	Ya

Peran layanan adalah peran IAM yang diambil oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

> **Warning**
>
> Mengubah izin untuk peran layanan dapat merusak fungsionalitas Amazon ECS. Edit peran layanan hanya jika Amazon ECS memberikan panduan untuk melakukannya.

Peran terkait layanan untuk Amazon ECS

Mendukung peran terkait layanan	Ya

Peran terkait layanan adalah jenis peran layanan yang ditautkan ke. Layanan AWS Layanan dapat menggunakan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

Untuk detail tentang membuat atau mengelola peran terkait layanan Amazon ECS, lihat. Menggunakan peran terkait layanan untuk Amazon ECS

Contoh kebijakan berbasis identitas untuk Amazon Elastic Container Service

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi sumber daya Amazon ECS. Mereka juga tidak dapat melakukan tugas dengan menggunakan AWS

Management Console, AWS Command Line Interface (AWS CLI), atau AWS API. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat menjalankan peran.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Untuk detail tentang tindakan dan jenis sumber daya yang ditentukan oleh Amazon ECS, termasuk format ARN untuk setiap jenis sumber daya, lihat Kunci tindakan, sumber daya, dan kondisi untuk Amazon Elastic Container Service di Referensi Otorisasi Layanan.

Topik

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya Amazon ECS di akun Anda. Tindakan ini dikenai biaya untuk Akun AWS Anda. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan AWS terkelola dan beralih ke izin hak istimewa paling sedikit — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan AWS terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di Anda Akun

AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola AWS pelanggan yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat kebijakan yang dikelola AWS atau kebijakan yang dikelola AWS untuk fungsi pekerjaan di Panduan Pengguna IAM.

- Menerapkan izin dengan hak akses paling rendah – Ketika Anda menetapkan izin dengan kebijakan IAM, hanya berikan izin yang diperlukan untuk melakukan tugas. Anda melakukan ini dengan menentukan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, juga dikenal sebagai izin hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk menerapkan izin, lihat Kebijakan dan izin di IAM di Panduan Pengguna IAM.
- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Sebagai contoh, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakan SSL. Anda juga dapat menggunakan ketentuan untuk memberikan akses ke tindakan layanan jika digunakan melalui yang spesifik Layanan AWS, seperti AWS CloudFormation. Untuk informasi selengkapnya, lihat Elemen kebijakan JSON IAM: Syarat di Panduan Pengguna IAM.
- Menggunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda guna memastikan izin yang aman dan berfungsi – IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat validasi kebijakan Analizer Akses IAM di Panduan Pengguna IAM.
- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Anda, Akun AWS aktifkan MFA untuk keamanan tambahan. Untuk mewajibkan MFA saat operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat Mengonfigurasi akses API yang dilindungi MFA di Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat Praktik terbaik keamanan di IAM di Panduan Pengguna IAM.

Izinkan pengguna melihat izin mereka sendiri

Contoh ini menunjukkan cara membuat kebijakan yang mengizinkan para pengguna IAM melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini

mencakup izin untuk menyelesaikan tindakan ini di konsol atau menggunakan API atau secara terprogram. AWS CLI AWS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Contoh klaster

Kebijakan IAM berikut memungkinkan izin untuk membuat dan membuat daftar cluster. Tindakan `CreateCluster` dan `ListClusters` tidak menerima sumber daya apa pun, sehingga definisi sumber daya diatur ke * untuk semua sumber daya.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:CreateCluster",
                "ecs:ListClusters"
            ],
            "Resource": ["*"]
        }
    ]
}
```

Kebijakan IAM berikut memungkinkan izin untuk mendeskripsikan dan menghapus klaster tertentu. Tindakan `DescribeClusters` dan `DeleteCluster` menerima ARN klaster sebagai sumber daya.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:DescribeClusters",
                "ecs:DeleteCluster"
            ],
            "Resource": ["arn:aws:ecs:us-east-1:<aws_account_id>:cluster/
<cluster_name>"]
        }
    ]
}
```

Kebijakan IAM berikut dapat dilampirkan ke pengguna atau grup yang hanya mengizinkan pengguna atau grup tersebut untuk melakukan operasi pada klaster tertentu.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ecs:Describe*",
                "ecs:List*"
```

```
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "ecs:DeleteCluster",
                "ecs:DeregisterContainerInstance",
                "ecs:ListContainerInstances",
                "ecs:RegisterContainerInstance",
                "ecs:SubmitContainerStateChange",
                "ecs:SubmitTaskStateChange"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:ecs:us-east-1:<aws_account_id>:cluster/default"
        },
        {
            "Action": [
                "ecs:DescribeContainerInstances",
                "ecs:DescribeTasks",
                "ecs:ListTasks",
                "ecs:UpdateContainerAgent",
                "ecs:StartTask",
                "ecs:StopTask",
                "ecs:RunTask"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "ArnEquals": {"ecs:cluster": "arn:aws:ecs:us-
east-1:<aws_account_id>:cluster/default"}
            }
        }
    ]
}
```

Contoh contoh kontainer

Pendaftaran instans kontainer ditangani oleh agen Amazon ECS, tetapi mungkin ada saat-saat di mana Anda ingin mengizinkan pengguna membatalkan pendaftaran instance secara manual dari klaster. Mungkin instans kontainer secara tidak sengaja didaftarkan ke klaster yang salah, atau instans diakhiri dengan tugas yang masih berjalan di dalamnya.

Kebijakan IAM berikut memungkinkan pengguna untuk membuat daftar dan membatalkan pendaftaran instance kontainer dalam klaster tertentu:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:DeregisterContainerInstance",
                "ecs:ListContainerInstances"
            ],
            "Resource": ["arn:aws:ecs:<region>:<aws_account_id>:cluster/
<cluster_name>"]
        }
    ]
}
```

Kebijakan IAM berikut memungkinkan pengguna untuk mendeskripsikan instance kontainer tertentu dalam cluster tertentu. Untuk membuka izin ini hingga semua instans kontainer dalam sebuah klaster, Anda dapat mengganti UUID instans kontainer dengan *.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:DescribeContainerInstances"],
            "Condition": {
                "ArnEquals": {"ecs:cluster":
 "arn:aws:ecs:<region>:<aws_account_id>:cluster/<cluster_name>"}
            },
            "Resource": ["arn:aws:ecs:<region>:<aws_account_id>:container-instance/
<cluster_name>/<container_instance_UUID>"]
        }
    ]
}
```

Contoh definisi tugas

Kebijakan IAM definisi tugas tidak mendukung izin tingkat sumber daya, tetapi kebijakan IAM berikut memungkinkan pengguna untuk mendaftar, mendaftar, dan menjelaskan definisi tugas:

Jika Anda menggunakan konsol, Anda harus menambahkan `CloudFormation: CreateStack` sebagai`Action`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RegisterTaskDefinition",
                "ecs:ListTaskDefinitions",
                "ecs:DescribeTaskDefinition"
            ],
            "Resource": ["*"]
        }
    ]
}
```

Jalankan Contoh Tugas

Sumber daya untuk `RunTask` adalah ketentuan tugas. Untuk membatasi klaster mana yang pengguna dapat menjalankan ketentuan tugas, Anda dapat menentukannya dalam blok `Condition`. Keuntungannya adalah Anda tidak perlu membuat daftar ketentuan tugas dan klaster di sumber daya Anda untuk mengizinkan akses yang sesuai. Anda bisa menerapkan satu, yang lain, atau keduanya.

Kebijakan IAM berikut memungkinkan izin untuk menjalankan revisi definisi tugas tertentu pada klaster tertentu:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:RunTask"],
            "Condition": {
                "ArnEquals": {"ecs:cluster":
 "arn:aws:ecs:<region>:<aws_account_id>:cluster/<cluster_name>"}
            },
            "Resource": ["arn:aws:ecs:<region>:<aws_account_id>:task-definition/
<task_family>:*"]
        }
    ]
```

```
}
```

Mulai contoh tugas

Sumber daya untuk `StartTask` adalah ketentuan tugas. Untuk membatasi klaster dan instans kontainer mana pengguna dapat memulai ketentuan tugas, Anda dapat menentukannya dalam blok `Condition`. Keuntungannya adalah Anda tidak perlu membuat daftar ketentuan tugas dan klaster di sumber daya Anda untuk mengizinkan akses yang sesuai. Anda bisa menerapkan satu, yang lain, atau keduanya.

Kebijakan IAM berikut memungkinkan izin untuk memulai revisi definisi tugas tertentu pada cluster tertentu dan instance container tertentu.

> **Note**
>
> Untuk contoh ini, saat Anda memanggil `StartTask` API dengan AWS CLI atau AWS SDK lain, Anda harus menentukan revisi definisi tugas agar `Resource` pemetaan cocok.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:StartTask"],
            "Condition": {
                "ArnEquals": {
                    "ecs:cluster": "arn:aws:ecs:<region>:<aws_account_id>:cluster/
<cluster_name>",
                    "ecs:container-instances":
 ["arn:aws:ecs:<region>:<aws_account_id>:container-instance/<cluster_name>/
<container_instance_UUID>"]
                }
            },
            "Resource": ["arn:aws:ecs:<region>:<aws_account_id>:task-definition/
<task_family>:*"]
        }
    ]
}
```

Buat daftar dan jelaskan contoh tugas

Kebijakan IAM berikut memungkinkan pengguna untuk membuat daftar tugas untuk klaster tertentu:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:ListTasks"],
            "Condition": {
                "ArnEquals": {"ecs:cluster":
 "arn:aws:ecs:<region>:<aws_account_id>:cluster/<cluster_name>"}
            },
            "Resource": ["*"]
        }
    ]
}
```

Kebijakan IAM berikut memungkinkan pengguna untuk mendeskripsikan tugas tertentu dalam klaster tertentu:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:DescribeTasks"],
            "Condition": {
                "ArnEquals": {"ecs:cluster":
 "arn:aws:ecs:<region>:<aws_account_id>:cluster/<cluster_name>"}
            },
            "Resource": ["arn:aws:ecs:<region>:<aws_account_id>:task/<cluster_name>/
<task_UUID>"]
        }
    ]
}
```

Buat contoh layanan

Kebijakan IAM berikut memungkinkan pengguna untuk membuat layanan Amazon ECS di: AWS Management Console

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:Describe*",
                "application-autoscaling:PutScalingPolicy",
                "application-autoscaling:RegisterScalableTarget",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:PutMetricAlarm",
                "ecs:List*",
                "ecs:Describe*",
                "ecs:CreateService",
                "elasticloadbalancing:Describe*",
                "iam:GetPolicy",
                "iam:GetPolicyVersion",
                "iam:GetRole",
                "iam:ListAttachedRolePolicies",
                "iam:ListRoles",
                "iam:ListGroups",
                "iam:ListUsers"
            ],
            "Resource": ["*"]
        }
    ]
}
```

Perbarui contoh layanan

Kebijakan IAM berikut memungkinkan pengguna untuk memperbarui layanan Amazon ECS di: AWS Management Console

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:Describe*",
                "application-autoscaling:PutScalingPolicy",
                "application-autoscaling:DeleteScalingPolicy",
                "application-autoscaling:RegisterScalableTarget",
```

```
                "cloudwatch:DescribeAlarms",
                "cloudwatch:PutMetricAlarm",
                "ecs:List*",
                "ecs:Describe*",
                "ecs:UpdateService",
                "iam:GetPolicy",
                "iam:GetPolicyVersion",
                "iam:GetRole",
                "iam:ListAttachedRolePolicies",
                "iam:ListRoles",
                "iam:ListGroups",
                "iam:ListUsers"
            ],
            "Resource": ["*"]
        }
    ]
}
```

Menjelaskan layanan Amazon ECS berdasarkan tag

Anda dapat menggunakan kondisi dalam kebijakan berbasis identitas untuk mengontrol akses ke sumber daya Amazon ECS berdasarkan tag. Contoh berikut menunjukkan cara agar Anda dapat membuat kebijakan yang mengizinkan penjelasan layanan. Namun, izin diberikan hanya jika tanda layanan `Owner` memiliki nilai nama pengguna dari pengguna tersebut. Kebijakan ini juga memberi izin yang diperlukan untuk menyelesaikan tindakan ini pada konsol tersebut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DescribeServices",
            "Effect": "Allow",
            "Action": "ecs:DescribeServices",
            "Resource": "*"
        },
        {
            "Sid": "ViewServiceIfOwner",
            "Effect": "Allow",
            "Action": "ecs:DescribeServices",
            "Resource": "arn:aws:ecs:*:*:service/*",
            "Condition": {
                "StringEquals": {"ecs:ResourceTag/Owner": "${aws:username}"}
            }
```

```
        }
    ]
}
```

Anda dapat melampirkan kebijakan ini ke pengguna IAM di akun Anda. Jika pengguna bernama `richard-roe` mencoba mendeskripsikan layanan Amazon ECS, layanan tersebut harus diberi tag `0wner=richard-roe` atau. `owner=richard-roe` Jika tidak, aksesnya akan ditolak. Kunci tanda syarat `0wner` cocok dengan `0wner` dan `owner` karena nama kunci syarat tidak terpengaruh huruf besar/kecil. Untuk informasi lebih lanjut, lihat Elemen Kebijakan IAM JSON: Persyaratan dalam Panduan Pengguna IAM.

Contoh Override Namespace Deny Service Connect

Kebijakan IAM berikut menyangkal pengguna untuk mengganti namespace Service Connect default dalam konfigurasi layanan. Namespace default diatur dalam cluster. Namun, Anda dapat menggantinya dalam konfigurasi layanan. Untuk konsistensi, pertimbangkan untuk mengatur semua layanan baru Anda untuk menggunakan namespace yang sama. Gunakan kunci konteks berikut untuk meminta layanan menggunakan namespace tertentu. Ganti`<region>`,`<aws_account_id>`, `<cluster_name>` dan `<namespace_id>` dengan milik Anda sendiri dalam contoh berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:CreateService",
                "ecs:UpdateService"
            ],
            "Condition": {
                "ArnEquals": {
                    "ecs:cluster": "arn:aws:ecs:<region>:<aws_account_id>:cluster/
<cluster_name>",
                    "ecs:namespace":
 "arn:aws:servicediscovery:<region>:<aws_account_id>:namespace/<namespace_id>"
                }
            },
            "Resource": "*"
        }
    ]
}
```

AWS kebijakan terkelola untuk Amazon Elastic Container Service

Untuk menambahkan izin ke pengguna, grup, dan peran, lebih mudah menggunakan kebijakan AWS terkelola daripada menulis kebijakan sendiri. Dibutuhkan waktu dan keahlian untuk membuat kebijakan yang dikelola pelanggan IAM yang hanya memberi tim Anda izin yang mereka butuhkan. Untuk memulai dengan cepat, Anda dapat menggunakan kebijakan AWS terkelola kami. Kebijakan ini mencakup kasus penggunaan umum dan tersedia di AWS akun Anda. Untuk informasi selengkapnya tentang kebijakan AWS AWS terkelola, lihat kebijakan terkelola di Panduan Pengguna IAM.

AWS layanan memelihara dan memperbarui kebijakan AWS terkelola. Anda tidak dapat mengubah izin dalam kebijakan AWS terkelola. Layanan terkadang menambahkan izin tambahan ke kebijakan AWS terkelola untuk mendukung fitur baru. Jenis pembaruan ini akan memengaruhi semua identitas (pengguna, grup, dan peran) di mana kebijakan tersebut dilampirkan. Layanan kemungkinan besar akan memperbarui kebijakan AWS terkelola saat fitur baru diluncurkan atau saat operasi baru tersedia. Layanan tidak menghapus izin dari kebijakan AWS terkelola, sehingga pembaruan kebijakan tidak akan merusak izin yang ada.

Selain itu, AWS mendukung kebijakan terkelola untuk fungsi pekerjaan yang mencakup beberapa layanan. Misalnya, kebijakan ReadOnlyAccess AWS terkelola menyediakan akses hanya-baca ke semua AWS layanan dan sumber daya. Saat layanan meluncurkan fitur baru, AWS tambahkan izin hanya-baca untuk operasi dan sumber daya baru. Untuk melihat daftar dan deskripsi dari kebijakan fungsi tugas, lihat kebijakan yang dikelola AWS untuk fungsi tugas di Panduan Pengguna IAM.

Amazon ECS dan Amazon ECR menyediakan beberapa kebijakan terkelola dan hubungan kepercayaan yang dapat Anda lampirkan ke pengguna, grup, peran, instans Amazon EC2, dan tugas Amazon ECS yang memungkinkan berbagai tingkat kontrol atas sumber daya dan operasi API. Anda dapat menerapkan kebijakan ini secara langsung atau menggunakannya sebagai titik awal untuk membuat kebijakan Anda sendiri. Untuk informasi selengkapnya tentang kebijakan terkelola Amazon ECR, lihat kebijakan yang dikelola Amazon ECR.

Amazonecs_ FullAccess

Anda dapat melampirkan kebijakan `AmazonECS_FullAccess` ke identitas IAM Anda.

Kebijakan ini memberikan akses administratif ke sumber daya Amazon ECS dan memberikan identitas IAM (seperti pengguna, grup, atau peran) akses ke layanan AWS yang terintegrasi dengan Amazon ECS untuk menggunakan semua fitur Amazon ECS. Menggunakan kebijakan ini memungkinkan akses ke semua fitur Amazon ECS yang tersedia di. AWS Management Console

Detail izin

Kebijakan IAM `AmazonECS_FullAccess` terkelola mencakup izin berikut. Mengikuti praktik terbaik pemberian hak istimewa paling rendah, Anda dapat menggunakan kebijakan terkelola `AmazonECS_FullAccess` sebagai templat untuk membuat kebijakan kustom Anda sendiri. Dengan begitu, Anda dapat mengambil atau menambahkan izin ke dan dari kebijakan terkelola berdasarkan kebutuhan khusus Anda.

- `ecs`— Memungkinkan kepala sekolah akses penuh ke semua operasi Amazon ECS API.
- `application-autoscaling`— Memungkinkan prinsipal untuk membuat, mendeskripsikan, dan mengelola sumber daya Application Auto Scaling. Ini diperlukan saat mengaktifkan penskalaan otomatis layanan untuk layanan Amazon ECS Anda.
- `appmesh`— Memungkinkan kepala sekolah untuk mencantumkan jerat layanan App Mesh dan node virtual dan menjelaskan node virtual App Mesh. Ini diperlukan saat mengintegrasikan layanan Amazon ECS Anda dengan App Mesh.
- `autoscaling`— Memungkinkan prinsipal untuk membuat, mengelola, dan menjelaskan sumber daya Auto Scaling Amazon EC2. Ini diperlukan saat mengelola grup Auto Scaling Amazon EC2 saat menggunakan fitur penskalaan otomatis cluster.
- `cloudformation`— Memungkinkan kepala sekolah untuk membuat dan mengelola tumpukan. AWS CloudFormation Ini diperlukan saat membuat cluster Amazon ECS menggunakan AWS Management Console dan pengelolaan cluster tersebut selanjutnya.
- `cloudwatch`— Memungkinkan kepala sekolah untuk membuat, mengelola, dan mendeskripsikan alarm Amazon. CloudWatch
- `codedeploy`— Memungkinkan prinsipal untuk membuat dan mengelola penerapan aplikasi dan melihat konfigurasi, revisi, dan target penyebaran mereka.
- `sns`— Memungkinkan kepala sekolah untuk melihat daftar topik Amazon SNS.
- `lambda`— Memungkinkan prinsipal untuk melihat daftar AWS Lambda fungsi dan konfigurasi spesifik versinya.
- `ec2`— Memungkinkan prinsipal menjalankan instans Amazon EC2 dan membuat serta mengelola rute, tabel rute, gateway internet, grup peluncuran, grup keamanan, cloud pribadi virtual, Armada Spot, dan subnet.
- `elasticloadbalancing`— Memungkinkan prinsipal untuk membuat, mendeskripsikan, dan menghapus penyeimbang beban Elastic Load Balancing. Prinsipal juga akan dapat menambahkan tag ke grup target, pendengar, dan aturan pendengar yang baru dibuat untuk penyeimbang beban.

- `events`— Memungkinkan prinsipal untuk membuat, mengelola, dan menghapus EventBridge aturan Amazon dan targetnya.
- `iam`— Memungkinkan kepala sekolah untuk membuat daftar peran IAM dan kebijakan terlampir mereka. Prinsipal juga dapat mencantumkan profil instans yang tersedia untuk instans Amazon EC2 Anda.
- `logs`— Memungkinkan prinsipal untuk membuat dan mendeskripsikan grup CloudWatch log Amazon Log. Prinsipiel juga dapat mencantumkan log acara pada grup log ini.
- `route53`— Memungkinkan prinsipal untuk membuat, mengelola, dan menghapus zona yang dihosting Amazon Route 53. Kepala sekolah juga dapat melihat konfigurasi dan informasi pemeriksaan kesehatan Amazon Route 53. Untuk informasi selengkapnya tentang zona yang di-hosting, lihat Bekerja dengan zona yang di-hosting.
- `servicediscovery`— Memungkinkan prinsipal untuk membuat, mengelola, dan menghapus AWS Cloud Map layanan dan membuat ruang nama DNS pribadi.

Berikut ini adalah contoh kebijakan `AmazonECS_FullAccess`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:DeleteScalingPolicy",
                "application-autoscaling:DeregisterScalableTarget",
                "application-autoscaling:DescribeScalableTargets",
                "application-autoscaling:DescribeScalingActivities",
                "application-autoscaling:DescribeScalingPolicies",
                "application-autoscaling:PutScalingPolicy",
                "application-autoscaling:RegisterScalableTarget",
                "appmesh:DescribeVirtualGateway",
                "appmesh:DescribeVirtualNode",
                "appmesh:ListMeshes",
                "appmesh:ListVirtualGateways",
                "appmesh:ListVirtualNodes",
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:CreateLaunchConfiguration",
                "autoscaling:DeleteAutoScalingGroup",
                "autoscaling:DeleteLaunchConfiguration",
                "autoscaling:Describe*",
```

```
                "autoscaling:UpdateAutoScalingGroup",
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStack*",
                "cloudformation:UpdateStack",
                "cloudwatch:DeleteAlarms",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:PutMetricAlarm",
                "codedeploy:BatchGetApplicationRevisions",
                "codedeploy:BatchGetApplications",
                "codedeploy:BatchGetDeploymentGroups",
                "codedeploy:BatchGetDeployments",
                "codedeploy:ContinueDeployment",
                "codedeploy:CreateApplication",
                "codedeploy:CreateDeployment",
                "codedeploy:CreateDeploymentGroup",
                "codedeploy:GetApplication",
                "codedeploy:GetApplicationRevision",
                "codedeploy:GetDeployment",
                "codedeploy:GetDeploymentConfig",
                "codedeploy:GetDeploymentGroup",
                "codedeploy:GetDeploymentTarget",
                "codedeploy:ListApplicationRevisions",
                "codedeploy:ListApplications",
                "codedeploy:ListDeploymentConfigs",
                "codedeploy:ListDeploymentGroups",
                "codedeploy:ListDeployments",
                "codedeploy:ListDeploymentTargets",
                "codedeploy:RegisterApplicationRevision",
                "codedeploy:StopDeployment",
                "ec2:AssociateRouteTable",
                "ec2:AttachInternetGateway",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CancelSpotFleetRequests",
                "ec2:CreateInternetGateway",
                "ec2:CreateLaunchTemplate",
                "ec2:CreateRoute",
                "ec2:CreateRouteTable",
                "ec2:CreateSecurityGroup",
                "ec2:CreateSubnet",
                "ec2:CreateVpc",
                "ec2:DeleteLaunchTemplate",
                "ec2:DeleteSubnet",
```

```
                "ec2:DeleteVpc",
                "ec2:Describe*",
                "ec2:DetachInternetGateway",
                "ec2:DisassociateRouteTable",
                "ec2:ModifySubnetAttribute",
                "ec2:ModifyVpcAttribute",
                "ec2:RequestSpotFleet",
                "ec2:RunInstances",
                "ecs:*",
                "elasticfilesystem:DescribeAccessPoints",
                "elasticfilesystem:DescribeFileSystems",
                "elasticloadbalancing:CreateListener",
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:CreateRule",
                "elasticloadbalancing:CreateTargetGroup",
                "elasticloadbalancing:DeleteListener",
                "elasticloadbalancing:DeleteLoadBalancer",
                "elasticloadbalancing:DeleteRule",
                "elasticloadbalancing:DeleteTargetGroup",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeRules",
                "elasticloadbalancing:DescribeTargetGroups",
                "events:DeleteRule",
                "events:DescribeRule",
                "events:ListRuleNamesByTarget",
                "events:ListTargetsByRule",
                "events:PutRule",
                "events:PutTargets",
                "events:RemoveTargets",
                "fsx:DescribeFileSystems",
                "iam:ListAttachedRolePolicies",
                "iam:ListInstanceProfiles",
                "iam:ListRoles",
                "lambda:ListFunctions",
                "logs:CreateLogGroup",
                "logs:DescribeLogGroups",
                "logs:FilterLogEvents",
                "route53:CreateHostedZone",
                "route53:DeleteHostedZone",
                "route53:GetHealthCheck",
                "route53:GetHostedZone",
                "route53:ListHostedZonesByName",
                "servicediscovery:CreatePrivateDnsNamespace",
```

```
                "servicediscovery:CreateService",
                "servicediscovery:DeleteService",
                "servicediscovery:GetNamespace",
                "servicediscovery:GetOperation",
                "servicediscovery:GetService",
                "servicediscovery:ListNamespaces",
                "servicediscovery:ListServices",
                "servicediscovery:UpdateService",
                "sns:ListTopics"
            ],
            "Resource": ["*"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameter",
                "ssm:GetParameters",
                "ssm:GetParametersByPath"
            ],
            "Resource": "arn:aws:ssm:*:*:parameter/aws/service/ecs*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DeleteInternetGateway",
                "ec2:DeleteRoute",
                "ec2:DeleteRouteTable",
                "ec2:DeleteSecurityGroup"
            ],
            "Resource": ["*"],
            "Condition": {
                "StringLike": {"ec2:ResourceTag/aws:cloudformation:stack-name":
"EC2ContainerService-*"}
            }
        },
        {
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource": ["*"],
            "Condition": {
                "StringLike": {"iam:PassedToService": "ecs-tasks.amazonaws.com"}
            }
        },
        {
```

```
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource": ["arn:aws:iam::*:role/ecsInstanceRole*"],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "ec2.amazonaws.com",
                        "ec2.amazonaws.com.cn"
                    ]
                }
            }
        },
        {
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource": ["arn:aws:iam::*:role/ecsAutoscaleRole*"],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "application-autoscaling.amazonaws.com",
                        "application-autoscaling.amazonaws.com.cn"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": [
                        "autoscaling.amazonaws.com",
                        "ecs.amazonaws.com",
                        "ecs.application-autoscaling.amazonaws.com",
                        "spot.amazonaws.com",
                        "spotfleet.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": ["elasticloadbalancing:AddTags"],
```

```
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "elasticloadbalancing:CreateAction": [
                        "CreateTargetGroup",
                        "CreateRule",
                        "CreateListener",
                        "CreateLoadBalancer"
                    ]
                }
            }
        }
    ]
}
```

AmazonECS InfrastructureRolePolicyForVolumes

Kebijakan IAM `AmazonECSInfrastructureRolePolicyForVolumes` terkelola memberikan izin yang diperlukan oleh Amazon ECS untuk melakukan panggilan AWS API atas nama Anda. Anda dapat melampirkan kebijakan ini ke peran IAM yang Anda berikan dengan konfigurasi volume saat meluncurkan tugas dan layanan Amazon ECS. Peran ini memungkinkan Amazon ECS untuk mengelola volume yang melekat pada tugas Anda. Untuk informasi selengkapnya, lihat peran IAM infrastruktur Amazon ECS.

Detail izin

Kebijakan IAM `AmazonECSInfrastructureRolePolicyForVolumes` terkelola mencakup izin berikut. Mengikuti saran keamanan standar untuk memberikan hak istimewa paling sedikit, Anda dapat menggunakan kebijakan `AmazonECSInfrastructureRolePolicyForVolumes` terkelola sebagai templat untuk membuat kebijakan kustom Anda sendiri yang hanya mencakup izin yang Anda perlukan.

- `ec2:CreateVolume`— Memungkinkan kepala sekolah untuk membuat volume Amazon EBS jika dan hanya jika mereka ditandai dengan `AmazonECSCreated` dan `AmazonECSManaged` tag. Izin ini diperlukan untuk membuat volume Amazon EBS yang dilampirkan ke tugas Amazon ECS dan meminimalkan izin yang diberikan ke Amazon ECS oleh kebijakan ini.
- `ec2:CreateTags`— Memungkinkan prinsipal untuk menambahkan tag ke volume Amazon EBS sebagai bagian dari. `ec2:CreateVolume` Izin ini diperlukan oleh Amazon ECS untuk menambahkan tag yang ditentukan pelanggan ke volume Amazon EBS yang dibuat atas nama Anda.

- `ec2:AttachVolume`— Memungkinkan prinsipal untuk melampirkan volume Amazon EBS ke instans Amazon EC2. Izin ini diperlukan oleh Amazon ECS untuk melampirkan volume Amazon EBS ke instans Amazon EC2 yang menghosting tugas Amazon ECS terkait.
- `ec2:DescribeVolume`— Memungkinkan kepala sekolah untuk mengambil informasi tentang volume Amazon EBS. Izin ini diperlukan untuk mengelola siklus hidup volume Amazon EBS.
- `ec2:DescribeAvailabilityZones`— Memungkinkan kepala sekolah untuk mengambil informasi tentang Availability Zones di akun Anda. Ini diperlukan untuk mengelola siklus hidup Volume EBS.
- `ec2:DetachVolume`— Memungkinkan prinsipal untuk melepaskan volume Amazon EBS dari instans Amazon EC2. Izin ini diperlukan oleh Amazon ECS untuk melepaskan volume Amazon EBS dari instans Amazon EC2 yang menghosting tugas Amazon ECS terkait saat tugas berakhir.
- `ec2:DeleteVolume`— Memungkinkan kepala sekolah untuk menghapus volume Amazon EBS. Izin ini diperlukan oleh Amazon ECS untuk menghapus volume Amazon EBS yang tidak lagi digunakan oleh tugas Amazon ECS.
- `ec2:DeleteTags`— Memungkinkan kepala sekolah untuk menghapus `AmazonECSManaged` tag dari volume Amazon EBS. Izin ini diperlukan oleh Amazon ECS untuk menghapus akses ke volume Amazon EBS setelah tidak lagi dikaitkan dengan beban kerja Amazon ECS. Ini hanya berlaku jika volume Amazon EBS tidak dihapus setelah tugas dimatikan.

Berikut ini adalah contoh kebijakan `AmazonECSInfrastructureRolePolicyForVolumes`.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "CreateEBSManagedVolume",
   "Effect": "Allow",
   "Action": "ec2:CreateVolume",
   "Resource": "arn:aws:ec2:*:*:volume/*",
   "Condition": {
    "ArnLike": {
     "aws:RequestTag/AmazonECSCreated": "arn:aws:ecs:*:*:task/*"
    },
    "StringEquals": {
     "aws:RequestTag/AmazonECSManaged": "true"
    }
   }
  },
```

```
{
 "Sid": "TagOnCreateVolume",
 "Effect": "Allow",
 "Action": "ec2:CreateTags",
 "Resource": "arn:aws:ec2:*:*:volume/*",
 "Condition": {
  "ArnLike": {
   "aws:RequestTag/AmazonECSCreated": "arn:aws:ecs:*:*:task/*"
  },
  "StringEquals": {
   "ec2:CreateAction": "CreateVolume",
   "aws:RequestTag/AmazonECSManaged": "true"
  }
 }
},
{
 "Sid": "DescribeVolumesForLifecycle",
 "Effect": "Allow",
 "Action": [
  "ec2:DescribeVolumes",
  "ec2:DescribeAvailabilityZones"
 ],
 "Resource": "*"
},
{
 "Sid": "ManageEBSVolumeLifecycle",
 "Effect": "Allow",
 "Action": [
  "ec2:AttachVolume",
  "ec2:DetachVolume"
 ],
 "Resource": "arn:aws:ec2:*:*:volume/*",
 "Condition": {
  "StringEquals": {
   "aws:ResourceTag/AmazonECSManaged": "true"
  }
 }
},
{
 "Sid": "ManageVolumeAttachmentsForEC2",
 "Effect": "Allow",
 "Action": [
  "ec2:AttachVolume",
  "ec2:DetachVolume"
```

```
    ],
    "Resource": "arn:aws:ec2:*:*:instance/*"
   },
   {
    "Sid": "DeleteEBSManagedVolume",
    "Effect": "Allow",
    "Action": "ec2:DeleteVolume",
    "Resource": "arn:aws:ec2:*:*:volume/*",
    "Condition": {
     "ArnLike": {
      "aws:ResourceTag/AmazonECSCreated": "arn:aws:ecs:*:*:task/*"
     },
     "StringEquals": {
      "aws:ResourceTag/AmazonECSManaged": "true"
     }
    }
   }
  ]
}
```

AmazonEC2 EC2peran ContainerServicefor

Amazon ECS melampirkan kebijakan ini ke peran layanan yang memungkinkan Amazon ECS melakukan tindakan atas nama Anda terhadap instans Amazon EC2 atau instans eksternal.

Kebijakan ini memberikan izin administratif yang memungkinkan instans penampung Amazon ECS melakukan panggilan atas nama Anda. AWS Untuk informasi selengkapnya, lihat Peran IAM instans wadah Amazon ECS.

Pertimbangan

Anda harus mempertimbangkan rekomendasi dan pertimbangan berikut saat menggunakan kebijakan IAM `AmazonEC2ContainerServiceforEC2Role` terkelola.

- Dengan mengikuti saran keamanan standar pemberian hak istimewa paling rendah, Anda dapat memodifikasi kebijakan terkelola `AmazonEC2ContainerServiceforEC2Role` agar sesuai dengan kebutuhan spesifik Anda. Jika salah satu izin yang diberikan dalam kebijakan terkelola tidak diperlukan untuk kasus penggunaan Anda, buat kebijakan kustom dan hanya tambahkan izin yang Anda perlukan. Misalnya, `UpdateContainerInstancesState` izin diberikan untuk pengeringan Instans Spot. Jika izin tersebut tidak diperlukan untuk kasus penggunaan Anda, izin tersebut dapat dikecualikan dengan menggunakan kebijakan kustom. Untuk informasi selengkapnya, lihat Detail izin.

- Kontainer yang berjalan pada instans kontainer Anda memiliki akses ke semua izin yang disediakan untuk peran instans kontainer melalui Metadata instans. Kami merekomendasikan Anda untuk membatasi izin dalam peran instans kontainer Anda agar hanya menyediakan izin minimal di kebijakan `AmazonEC2ContainerServiceforEC2Role` terkelola. Jika kontainer dalam tugas Anda memerlukan izin tambahan yang tidak terdaftar, sebaiknya berikan tugas tersebut dengan peran IAM mereka sendiri. Untuk informasi selengkapnya, lihat Tugas peran IAM.

  Kontainer pada jembatan `docker0` dapat dicegah untuk tidak mengakses izin yang disediakan untuk peran instans kontainer.. Tindakan ini masih bisa dilakukan bersamaan dengan berlakunya izin yang diberikan Tugas peran IAM dengan menjalankan perintah iptables pada instans kontainer Anda. Kontainer tidak dapat melakukan kueri terhadap metadata instans saat aturan ini berlaku. Perintah ini mengasumsikan konfigurasi jembatan Docker default dan tidak bekerja dengan kontainer yang menerapkan mode jaringan `host`. Untuk informasi selengkapnya, lihat Mode jaringan.

  ```
  sudo yum install -y iptables-services; sudo iptables --insert DOCKER USER 1 --in-interface docker+ --destination 169.254.169.254/32 --jump DROP
  ```

  Anda harus menyimpan aturan iptables ini pada instans kontainer Anda untuk itu untuk bertahan saat booting ulang. Untuk AMI Amazon ECS yang dioptimalkan, gunakan perintah berikut. Untuk sistem operasi lain, konsultasikan dokumentasi untuk OS tersebut.

  - Untuk Amazon ECS yang dioptimalkan Amazon Linux 2 AMI:

    ```
    sudo iptables-save | sudo tee /etc/sysconfig/iptables && sudo systemctl enable --now iptables
    ```

  - Untuk Amazon ECS yang dioptimalkan Amazon Linux AMI:

    ```
    sudo service iptables save
    ```

Detail izin

Kebijakan IAM `AmazonEC2ContainerServiceforEC2Role` terkelola mencakup izin berikut. Dengan mengikuti saran keamanan standar untuk memberikan hak istimewa paling rendah, kebijakan terkelola `AmazonEC2ContainerServiceforEC2Role` dapat digunakan sebagai panduan. Jika salah satu izin yang diberikan dalam kebijakan terkelola tidak diperlukan untuk kasus penggunaan Anda, buat kebijakan kustom dan hanya tambahkan izin yang Anda perlukan.

- `ec2:DescribeTags`— Memungkinkan prinsipal untuk mendeskripsikan tag yang terkait dengan instans Amazon EC2. Izin ini digunakan oleh agen penampung Amazon ECS untuk mendukung propagasi tag sumber daya. Untuk informasi selengkapnya, lihat Bagaimana sumber daya ditandai.
- `ecs:CreateCluster`— Memungkinkan kepala sekolah untuk membuat cluster Amazon ECS. Izin ini digunakan oleh agen penampung Amazon ECS untuk membuat `default` klaster, jika belum ada.
- `ecs:DeregisterContainerInstance`— Memungkinkan prinsipal untuk membatalkan pendaftaran instans kontainer Amazon ECS dari sebuah cluster. Agen penampung Amazon ECS tidak memanggil operasi API ini, tetapi izin ini tetap untuk membantu memastikan kompatibilitas mundur.
- `ecs:DiscoverPollEndpoint`— Tindakan ini mengembalikan titik akhir yang digunakan agen penampung Amazon ECS untuk melakukan polling untuk pembaruan.
- `ecs:Poll`— Memungkinkan agen kontainer Amazon ECS untuk berkomunikasi dengan bidang kontrol Amazon ECS untuk melaporkan perubahan status tugas.
- `ecs:RegisterContainerInstance`— Memungkinkan prinsipal untuk mendaftarkan instance kontainer dengan cluster. Izin ini digunakan oleh agen penampung Amazon ECS untuk mendaftarkan instans Amazon EC2 dengan klaster dan untuk mendukung propagasi tag sumber daya.
- `ecs:StartTelemetrySession`— Memungkinkan agen kontainer Amazon ECS untuk berkomunikasi dengan pesawat kontrol Amazon ECS untuk melaporkan informasi kesehatan dan metrik untuk setiap wadah dan tugas.
- `ecs:TagResource`— Memungkinkan agen penampung Amazon ECS untuk menandai klaster saat pembuatan dan menandai instance kontainer saat terdaftar ke cluster.
- `ecs:UpdateContainerInstancesState`— Memungkinkan kepala sekolah untuk memodifikasi status instans penampung Amazon ECS. Izin ini digunakan oleh agen kontainer Amazon ECS untuk pengeringan Instans Spot. Untuk informasi selengkapnya, lihat Pengeringan Instans Spot.
- `ecs:Submit*`— Ini termasuk`SubmitAttachmentStateChanges`,`SubmitContainerStateChange`, dan tindakan `SubmitTaskStateChange` API. Mereka digunakan oleh agen penampung Amazon ECS untuk melaporkan perubahan status untuk setiap sumber daya ke bidang kontrol Amazon ECS. `SubmitContainerStateChange`Izin tersebut tidak lagi digunakan oleh agen penampung Amazon ECS tetapi tetap untuk membantu memastikan kompatibilitas mundur.
- `ecr:GetAuthorizationToken`— Memungkinkan kepala sekolah untuk mengambil token otorisasi. Token otorisasi mewakili kredensyal autentikasi IAM Anda dan dapat digunakan untuk

mengakses registri ECR Amazon apa pun yang dapat diakses oleh prinsipal IAM. Token otorisasi yang diterima berlaku selama 12 jam.

- `ecr:BatchCheckLayerAvailability`— Saat gambar kontainer didorong ke repositori pribadi Amazon ECR, setiap lapisan gambar diperiksa untuk memverifikasi apakah sudah didorong. Jika sudah didorong, maka lapisan citra dilewati.
- `ecr:GetDownloadUrlForLayer`— Ketika gambar kontainer ditarik dari repositori pribadi Amazon ECR, API ini dipanggil sekali untuk setiap lapisan gambar yang belum di-cache.
- `ecr:BatchGetImage`— Saat gambar kontainer ditarik dari repositori pribadi Amazon ECR, API ini dipanggil sekali untuk mengambil manifes gambar.
- `logs:CreateLogStream`— Memungkinkan prinsipal untuk membuat aliran CloudWatch log Log untuk grup log tertentu.
- `logs:PutLogEvents`— Memungkinkan prinsipal untuk mengunggah sekumpulan peristiwa log ke aliran log tertentu.

Berikut ini adalah contoh kebijakan `AmazonEC2ContainerServiceforEC2Role`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeTags",
                "ecs:CreateCluster",
                "ecs:DeregisterContainerInstance",
                "ecs:DiscoverPollEndpoint",
                "ecs:Poll",
                "ecs:RegisterContainerInstance",
                "ecs:StartTelemetrySession",
                "ecs:UpdateContainerInstancesState",
                "ecs:Submit*",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
```

```
        },
        {
            "Effect": "Allow",
            "Action": "ecs:TagResource",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:CreateAction": [
                        "CreateCluster",
                        "RegisterContainerInstance"
                    ]
                }
            }
        }
    ]
}
```

AmazonEC2 ContainerServiceEventsRole

Kebijakan ini memberikan izin yang memungkinkan Amazon EventBridge (sebelumnya CloudWatch Acara) menjalankan tugas atas nama Anda. Kebijakan ini dapat dilampirkan ke peran IAM yang ditentukan saat Anda membuat tugas terjadwal. Untuk informasi selengkapnya, lihat Peran Amazon ECS EventBridge IAM.

Detail izin

Kebijakan ini mencakup izin berikut.

- `ecs`— Memungkinkan kepala sekolah dalam layanan untuk memanggil Amazon ECS RunTask API. Memungkinkan prinsipal dalam layanan untuk menambahkan tag (`TagResource`) saat mereka memanggil Amazon ECS RunTask API.
- `iam`- Memungkinkan meneruskan peran layanan IAM apa pun ke tugas Amazon ECS apa pun.

Berikut ini adalah contoh kebijakan `AmazonEC2ContainerServiceEventsRole`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:RunTask"],
```

```
            "Resource": ["*"]
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["*"],
            "Condition": {
                "StringLike": {"iam:PassedToService": "ecs-tasks.amazonaws.com"}
            }
        },
        {
            "Effect": "Allow",
            "Action": "ecs:TagResource",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:CreateAction": ["RunTask"]
                }
            }
        }
    ]
}
```

AmazonECS TaskExecutionRolePolicy

Kebijakan IAM `AmazonECSTaskExecutionRolePolicy` terkelola memberikan izin yang diperlukan oleh agen kontainer Amazon ECS dan AWS Fargate agen kontainer untuk melakukan panggilan AWS API atas nama Anda. Kebijakan ini dapat ditambahkan ke peran IAM eksekusi tugas Anda. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

Detail izin

Kebijakan IAM `AmazonECSTaskExecutionRolePolicy` terkelola mencakup izin berikut. Dengan mengikuti saran keamanan standar untuk memberikan hak istimewa paling rendah, kebijakan terkelola `AmazonECSTaskExecutionRolePolicy` dapat digunakan sebagai panduan. Jika salah satu izin yang diberikan dalam kebijakan terkelola tidak diperlukan untuk kasus penggunaan Anda, buat kebijakan kustom dan hanya tambahkan izin yang Anda perlukan.

- `ecr:GetAuthorizationToken`— Memungkinkan kepala sekolah untuk mengambil token otorisasi. Token otorisasi mewakili kredensyal autentikasi IAM Anda dan dapat digunakan untuk mengakses registri ECR Amazon apa pun yang dapat diakses oleh prinsipal IAM. Token otorisasi yang diterima berlaku selama 12 jam.

- `ecr:BatchCheckLayerAvailability`— Saat gambar kontainer didorong ke repositori pribadi Amazon ECR, setiap lapisan gambar diperiksa untuk memverifikasi apakah sudah didorong. Jika sudah didorong, maka lapisan citra dilewati.
- `ecr:GetDownloadUrlForLayer`— Ketika gambar kontainer ditarik dari repositori pribadi Amazon ECR, API ini dipanggil sekali untuk setiap lapisan gambar yang belum di-cache.
- `ecr:BatchGetImage`— Saat gambar kontainer ditarik dari repositori pribadi Amazon ECR, API ini dipanggil sekali untuk mengambil manifes gambar.
- `logs:CreateLogStream`— Memungkinkan prinsipal untuk membuat aliran CloudWatch log Log untuk grup log tertentu.
- `logs:PutLogEvents`— Memungkinkan prinsipal untuk mengunggah sekumpulan peristiwa log ke aliran log tertentu.

Berikut ini adalah contoh kebijakan `AmazonECSTaskExecutionRolePolicy`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

AmazonECS ServiceRolePolicy

Kebijakan IAM `AmazonECSServiceRolePolicy` terkelola memungkinkan Amazon Elastic Container Service mengelola klaster Anda. Kebijakan ini dapat ditambahkan ke peran IAM eksekusi tugas Anda. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

Detail izin

Kebijakan IAM `AmazonECSServiceRolePolicy` terkelola mencakup izin berikut. Dengan mengikuti saran keamanan standar untuk memberikan hak istimewa paling rendah, kebijakan terkelola `AmazonECSServiceRolePolicy` dapat digunakan sebagai panduan. Jika salah satu izin yang diberikan dalam kebijakan terkelola tidak diperlukan untuk kasus penggunaan Anda, buat kebijakan kustom dan hanya tambahkan izin yang Anda perlukan.

- `autoscaling`— Memungkinkan prinsipal untuk membuat, mengelola, dan menjelaskan sumber daya Auto Scaling Amazon EC2. Ini diperlukan saat mengelola grup Auto Scaling Amazon EC2 saat menggunakan fitur penskalaan otomatis cluster.
- `autoscaling-plans`— Memungkinkan prinsipal untuk membuat, menghapus, dan menjelaskan rencana penskalaan otomatis.
- `cloudwatch`— Memungkinkan kepala sekolah untuk membuat, mengelola, dan mendeskripsikan alarm Amazon. CloudWatch
- `ec2`— Memungkinkan prinsipal berjalan ke instans Amazon EC2 dan membuat serta mengelola antarmuka dan tag jaringan.
- `elasticloadbalancing`— Memungkinkan prinsipal untuk membuat, mendeskripsikan, dan menghapus penyeimbang beban Elastic Load Balancing. Prinsipal juga akan dapat menambah dan mendeskripsikan kelompok sasaran.
- `logs`— Memungkinkan prinsipal untuk membuat dan mendeskripsikan grup CloudWatch log Amazon Log. Prinsipiel juga dapat mencantumkan log acara pada grup log ini.
- `route53`— Memungkinkan prinsipal untuk membuat, mengelola, dan menghapus zona yang dihosting Amazon Route 53. Kepala sekolah juga dapat melihat konfigurasi dan informasi pemeriksaan kesehatan Amazon Route 53. Untuk informasi selengkapnya tentang zona yang di-hosting, lihat Bekerja dengan zona yang di-hosting.
- `servicediscovery`— Memungkinkan prinsipal untuk membuat, mengelola, dan menghapus AWS Cloud Map layanan dan membuat ruang nama DNS pribadi.
- `events`— Memungkinkan prinsipal untuk membuat, mengelola, dan menghapus EventBridge aturan Amazon dan targetnya.

Berikut ini adalah contoh kebijakan `AmazonECSServiceRolePolicy`.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "ECSTaskManagement",
            "Effect": "Allow",
            "Action": [
                "ec2:AttachNetworkInterface",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:Describe*",
                "ec2:DetachNetworkInterface",
                "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
                "elasticloadbalancing:DeregisterTargets",
                "elasticloadbalancing:Describe*",
                "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
                "elasticloadbalancing:RegisterTargets",
                "route53:ChangeResourceRecordSets",
                "route53:CreateHealthCheck",
                "route53:DeleteHealthCheck",
                "route53:Get*",
                "route53:List*",
                "route53:UpdateHealthCheck",
                "servicediscovery:DeregisterInstance",
                "servicediscovery:Get*",
                "servicediscovery:List*",
                "servicediscovery:RegisterInstance",
                "servicediscovery:UpdateInstanceCustomHealthStatus"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AutoScaling",
            "Effect": "Allow",
            "Action": [
                "autoscaling:Describe*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AutoScalingManagement",
            "Effect": "Allow",
            "Action": [
                "autoscaling:DeletePolicy",
                "autoscaling:PutScalingPolicy",
```

```
                "autoscaling:SetInstanceProtection",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:PutLifecycleHook",
                "autoscaling:DeleteLifecycleHook",
                "autoscaling:CompleteLifecycleAction",
                "autoscaling:RecordLifecycleActionHeartbeat"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "autoscaling:ResourceTag/AmazonECSManaged": "false"
                }
            }
        },
        {
            "Sid": "AutoScalingPlanManagement",
            "Effect": "Allow",
            "Action": [
                "autoscaling-plans:CreateScalingPlan",
                "autoscaling-plans:DeleteScalingPlan",
                "autoscaling-plans:DescribeScalingPlans",
                "autoscaling-plans:DescribeScalingPlanResources"
            ],
            "Resource": "*"
        },
        {
            "Sid": "EventBridge",
            "Effect": "Allow",
            "Action": [
                "events:DescribeRule",
                "events:ListTargetsByRule"
            ],
            "Resource": "arn:aws:events:*:*:rule/ecs-managed-*"
        },
        {
            "Sid": "EventBridgeRuleManagement",
            "Effect": "Allow",
            "Action": [
                "events:PutRule",
                "events:PutTargets"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
```

```
                    "events:ManagedBy": "ecs.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CWAlarmManagement",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:DeleteAlarms",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:PutMetricAlarm"
            ],
            "Resource": "arn:aws:cloudwatch:*:*:alarm:*"
        },
        {
            "Sid": "ECSTagging",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "arn:aws:ec2:*:*:network-interface/*"
        },
        {
            "Sid": "CWLogGroupManagement",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:DescribeLogGroups",
                "logs:PutRetentionPolicy"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/ecs/*"
        },
        {
            "Sid": "CWLogStreamManagement",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/ecs/*:log-stream:*"
        },
        {
            "Sid": "ExecuteCommandSessionManagement",
```

```
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeSessions"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ExecuteCommand",
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession"
            ],
            "Resource": [
                "arn:aws:ecs:*:*:task/*",
                "arn:aws:ssm:*:*:document/AmazonECS-ExecuteInteractiveCommand"
            ]
        },
        {
            "Sid": "CloudMapResourceCreation",
            "Effect": "Allow",
            "Action": [
                "servicediscovery:CreateHttpNamespace",
                "servicediscovery:CreateService"
            ],
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "AmazonECSManaged"
                    ]
                }
            }
        },
        {
            "Sid": "CloudMapResourceTagging",
            "Effect": "Allow",
            "Action": "servicediscovery:TagResource",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aws:RequestTag/AmazonECSManaged": "*"
                }
            }
        },
```

```
        {
            "Sid": "CloudMapResourceDeletion",
            "Effect": "Allow",
            "Action": [
                "servicediscovery:DeleteService"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/AmazonECSManaged": "false"
                }
            }
        },
        {
            "Sid": "CloudMapResourceDiscovery",
            "Effect": "Allow",
            "Action": [
                "servicediscovery:DiscoverInstances",
                "servicediscovery:DiscoverInstancesRevision"
            ],
            "Resource": "*"
        }
    ]
}
```

AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecur

Menyediakan akses administratif ke AWS Private Certificate Authority, Secrets Manager, dan AWS Layanan lain yang diperlukan untuk mengelola fitur Amazon ECS Service Connect TLS atas nama Anda.

Detail izin

Kebijakan IAM `AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity` terkelola mencakup izin berikut. Dengan mengikuti saran keamanan standar untuk memberikan hak istimewa paling rendah, kebijakan terkelola `AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity` dapat digunakan sebagai panduan. Jika salah satu izin yang diberikan dalam kebijakan terkelola tidak diperlukan untuk kasus penggunaan Anda, buat kebijakan kustom dan hanya tambahkan izin yang Anda perlukan.

- `secretsmanager:CreateSecret`— Memungkinkan kepala sekolah untuk membuat rahasia. Ini diperlukan untuk Service Connect TLS, Amazon ECS menyimpan kunci pribadi pelanggan dalam rahasia Secrets Manager pelanggan.
- `secretsmanager:TagResource`— Memungkinkan kepala sekolah untuk melampirkan tag pada rahasia yang dibuat. Ini diperlukan untuk Service Connect TLS, karena Amazon ECS membuat rahasia atas nama pelanggan dan melampirkan tag dengan sumber daya. Tag ini memberikan cara yang lebih mudah bagi pelanggan untuk mengidentifikasi rahasia yang dikelola dan membatasi tindakan pada rahasia ini.
- `secretsmanager:DescribeSecret`— Izinkan kepala sekolah untuk menggambarkan rahasia dan mengambil tahap versi saat ini. Amazon ECS diperlukan untuk melakukan rotasi bahan Amazon ECS Service Connect TLS.
- `secretsmanager:UpdateSecret`— Izinkan kepala sekolah untuk memperbarui rahasia. Amazon ECS diperlukan untuk melakukan rotasi bahan Amazon ECS Service Connect TLS dan memperbarui rahasia dengan materi baru.
- `secretsmanager:GetSecretValue`— Izinkan kepala sekolah untuk mendapatkan nilai rahasia. Amazon ECS diperlukan untuk melakukan rotasi bahan Amazon ECS Service Connect TLS.
- `secretsmanager:PutSecretValue`— Izinkan kepala sekolah untuk menempatkan nilai rahasia. Amazon ECS diperlukan untuk melakukan rotasi bahan Amazon ECS Service Connect TLS.
- `secretsmanager:UpdateSecretVersionStage`— Izinkan kepala sekolah untuk memperbarui tahap versi rahasia. Amazon ECS diperlukan untuk melakukan rotasi bahan Amazon ECS Service Connect TLS.
- `acm-pca:IssueCertificate`— Izinkan kepala sekolah IssueCertificate `End entity certificate` untuk memanggil Amazon ECS Service Connect TLS. Diperlukan ECS untuk menghasilkan sertifikat untuk layanan hulu pelanggan.
- `acm-pca:GetCertificate`— Izinkan kepala sekolah GetCertificate `End entity certificate` untuk memanggil Amazon ECS Service Connect TLS.
- `acm-pca:GetCertificateAuthorityCertificate`— Izinkan kepala sekolah untuk mendapatkan sertifikat otoritas sertifikat. Ini diperlukan untuk Amazon ECS Service Connect TLS sehingga layanan hilir pelanggan dapat mempercayai sertifikat entitas akhir hulu.
- `acm-pca:DescribeCertificateAuthority`— Izinkan kepala sekolah untuk mendapatkan rincian tentang otoritas sertifikat. Amazon ECS Service Connect TLS diperlukan untuk menggunakan kembali informasi seperti algoritma penandatanganan untuk membuat CSR (Permintaan Penandatanganan Sertifikat).

Berikut ini adalah contoh kebijakan `AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateSecret",
            "Effect": "Allow",
            "Action": "secretsmanager:CreateSecret",
            "Resource": "arn:aws:secretsmanager:*:*:secret:ecs-sc!*",
            "Condition": {
                "ArnLike": {
                    "aws:RequestTag/AmazonECSCreated": [
                        "arn:aws:ecs:*:*:service/*/*",
                        "arn:aws:ecs:*:*:task-set/*/*"
                    ]
                },
                "StringEquals": {
                    "aws:RequestTag/AmazonECSManaged": "true",
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        },
        {
            "Sid": "TagOnCreateSecret",
            "Effect": "Allow",
            "Action": "secretsmanager:TagResource",
            "Resource": "arn:aws:secretsmanager:*:*:secret:ecs-sc!*",
            "Condition": {
                "ArnLike": {
                     "aws:RequestTag/AmazonECSCreated": [
                        "arn:aws:ecs:*:*:service/*/*",
                        "arn:aws:ecs:*:*:task-set/*/*"
                    ]
                },
                "StringEquals": {
                    "aws:RequestTag/AmazonECSManaged": "true",
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        },
        {
```

```
            "Sid": "RotateTLSCertificateSecret",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:UpdateSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "secretsmanager:DeleteSecret",
                "secretsmanager:RotateSecret",
                "secretsmanager:UpdateSecretVersionStage"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:ecs-sc!*",
            "Condition": {
                "StringEquals": {
                    "secretsmanager:ResourceTag/aws:secretsmanager:owningService":
"ecs-sc",
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        },
        {
            "Sid": "ManagePrivateCertificateAuthority",
            "Effect": "Allow",
            "Action": [
                "acm-pca:GetCertificate",
                "acm-pca:GetCertificateAuthorityCertificate",
                "acm-pca:DescribeCertificateAuthority"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/AmazonECSManaged": "true"
                }
            }
        },
        {
            "Sid": "ManagePrivateCertificateAuthorityForIssuingEndEntityCertificate",
            "Effect": "Allow",
            "Action": [
                "acm-pca:IssueCertificate"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
```

```
                    "aws:ResourceTag/AmazonECSManaged": "true",
                    "acm-pca:TemplateArn":"arn:aws:acm-pca:::template/
EndEntityCertificate/V1"
                }
            }
        }
    ]
}
```

AWSApplicationAutoscalingECSServicePolicy

Anda tidak dapat melampirkan AWSApplicationAutoscalingECSServicePolicy ke entitas IAM Anda. Kebijakan ini dilampirkan pada peran terkait layanan yang memungkinkan Application Auto Scaling untuk melakukan tindakan atas nama Anda. Untuk informasi selengkapnya. lihat Peran tertaut layanan untuk Application Auto Scaling.

AWSCodeDeployRoleForECS

Anda tidak dapat melampirkan AWSCodeDeployRoleForECS ke entitas IAM Anda. Kebijakan ini dilampirkan pada peran terkait layanan yang memungkinkan CodeDeploy untuk melakukan tindakan atas nama Anda. Untuk informasi selengkapnya, lihat Membuat peran layanan CodeDeploy di Panduan AWS CodeDeploy Pengguna.

AWSCodeDeployRoleForECSLimited

Anda tidak dapat melampirkan AWSCodeDeployRoleForECSLimited ke entitas IAM Anda. Kebijakan ini dilampirkan pada peran terkait layanan yang memungkinkan CodeDeploy untuk melakukan tindakan atas nama Anda. Untuk informasi selengkapnya, lihat Membuat peran layanan CodeDeploy di Panduan AWS CodeDeploy Pengguna.

Amazon ECS memperbarui kebijakan AWS terkelola

Lihat detail tentang pembaruan kebijakan AWS terkelola untuk Amazon ECS sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman riwayat Dokumen Amazon ECS.

Perubahan	Deskripsi	Tanggal
Tambahkan kebijakan AmazonECS Infrastru	Menambahkan kebijakan AmazonECS baru yang	Januari 22, 2024

Perubahan	Deskripsi	Tanggal
ctureRolePolicyForServiceConnectTransport LayerSecurity baru	menyediakan Infrastructure RolePolicyForServiceConnectTransport LayerSecurity akses administratif ke, AWS Private Certificate Authority Secrets Manager AWS KMS, dan memungkinkan fitur Amazon ECS Service Connect TLS berfungsi dengan baik.	
Tambahkan kebijakan baru AmazonECS InfrastructureRolePolicyForVolumes	`AmazonECSInfrastructureRolePolicyForVolumes` Kebijakan itu ditambahkan. Kebijakan ini memberikan izin yang diperlukan oleh Amazon ECS untuk melakukan panggilan AWS API guna mengelola volume Amazon EBS yang terkait dengan beban kerja Amazon ECS.	Januari 11, 2024
Tambahkan izin ke AmazonECS ServiceRolePolicy	Kebijakan IAM `AmazonECSServiceRolePolicy` terkelola telah diperbarui dengan `events` izin baru `autoscaling` dan `autoscaling-plans` tambahan serta izin.	Desember 4, 2023

Perubahan	Deskripsi	Tanggal
Tambahkan izin ke AmazonEC2 Container ServiceEventsRole	Kebijakan IAM `AmazonECS ServiceRolePolicy` terkelola telah diperbarui untuk mengizinkan akses ke operasi AWS Cloud Map `DiscoverI nstancesRevision` API.	4 Oktober 2023
Tambahkan izin ke Container ServiceforAmazonec2 EC2role	`AmazonEC2Container Servicefo rEC2Role` Kebijakan diubah untuk menambahkan `ecs:TagResource` izin, yang mencakup kondisi yang membatasi izin hanya untuk cluster yang baru dibuat dan instance container terdaftar.	6 Maret 2023
Tambahkan izin ke the section called “Amazonecs_ FullAcces s”	`AmazonECS_FullAcce ss` Kebijakan diubah untuk menambahkan `elasticlo adbalancing:AddTags` izin, yang mencakup kondisi yang membatasi izin hanya untuk penyeimbang beban, grup target, aturan, dan pendengar yang baru dibuat. Izin ini tidak mengizinkan tag ditambahkan ke sumber daya Elastic Load Balancing yang sudah dibuat.	4 Januari 2023
Amazon ECS mulai melacak perubahan	Amazon ECS mulai melacak perubahan untuk kebijakan yang AWS dikelola.	8 Juni 2021

Menghapus kebijakan IAM AWS terkelola untuk Amazon Elastic Container Service

Kebijakan IAM AWS terkelola berikut dihapus. Kebijakan ini sekarang digantikan oleh kebijakan yang sudah diperbarui. Kami menyarankan Anda untuk memperbarui pengguna atau peran untuk menggunakan kebijakan yang diperbarui tersebut.

AmazonEC2 ContainerServiceFullAccess

> **Important**
>
> Kebijakan IAM yang `AmazonEC2ContainerServiceFullAccess` dikelola dihapus pada 29 Januari 2021, sebagai tanggapan atas temuan keamanan dengan izin tersebut`iam:passRole`. Izin ini memberikan akses ke semua sumber daya termasuk kredensial untuk berperan di akun. Setelah kebijakan dihapus, Anda tidak dapat melampirkan kebijakan ke pengguna atau peran baru. Setiap pengguna atau peran yang sudah memiliki kebijakan yang dilampirkan dapat terus menggunakannya. Namun, kami menyarankan agar Anda memperbarui pengguna atau peran Anda untuk menggunakan kebijakan `AmazonECS_FullAccess` terkelola sebagai gantinya. Untuk informasi selengkapnya, lihat Migrasi ke kebijakan terkelola `AmazonECS_FullAccess`.

AmazonEC2 ContainerServiceRole

> **Important**
>
> Kebijakan IAM yang `AmazonEC2ContainerServiceRole` dikelola dihapus. Sekarang digantikan oleh peran terkait layanan Amazon ECS. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS.

AmazonEC2 ContainerServiceAutoscaleRole

> **Important**
>
> Kebijakan IAM yang `AmazonEC2ContainerServiceAutoscaleRole` dikelola dihapus. Sekarang digantikan oleh peran terkait layanan Application Auto Scaling untuk Amazon ECS. Untuk informasi selengkapnya, lihat Peran yang ditautkan dengan layanan untuk Application Auto Scaling dalam Panduan Pengguna Application Auto Scaling.

Migrasi ke kebijakan terkelola **`AmazonECS_FullAccess`**

Kebijakan IAM yang `AmazonEC2ContainerServiceFullAccess` dikelola dihapus pada 29 Januari 2021, sebagai tanggapan atas temuan keamanan dengan izin. `iam:passRole` Izin ini memberikan akses ke semua sumber daya termasuk kredensial untuk berperan di akun. Setelah kebijakan dihapus, Anda tidak dapat melampirkan kebijakan tersebut ke grup, pengguna atau peran baru. Setiap grup, pengguna, atau peran yang sudah memiliki kebijakan yang dilampirkan dapat terus menggunakannya. Namun, kami menyarankan Anda memperbarui grup, pengguna, atau peran Anda untuk menggunakan kebijakan `AmazonECS_FullAccess` terkelola sebagai gantinya.

Izin yang diberikan oleh kebijakan `AmazonECS_FullAccess` termasuk daftar lengkap izin yang diperlukan untuk menggunakan ECS sebagai administrator. Jika saat ini Anda menggunakan izin yang diberikan oleh `AmazonEC2ContainerServiceFullAccess` kebijakan yang tidak ada dalam `AmazonECS_FullAccess` kebijakan, Anda dapat menambahkannya ke pernyataan kebijakan sebaris. Untuk informasi selengkapnya, lihat AWS kebijakan terkelola untuk Amazon Elastic Container Service.

Gunakan langkah-langkah berikut untuk menentukan apakah Anda memiliki grup, pengguna, atau peran yang saat ini menggunakan kebijakan IAM `AmazonEC2ContainerServiceFullAccess` terkelola. Kemudian perbarui grup, pengguna, atau peran tersebut untuk melepaskan kebijakan sebelumnya dan melampirkan kebijakan `AmazonECS_FullAccess`.

Untuk memperbarui grup, pengguna, atau peran untuk menggunakan kebijakan FullAccess Amazonecs_ ()AWS Management Console

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Kebijakan lalu cari dan pilih kebijakan `AmazonEC2ContainerServiceFullAccess`.
3. Pilih tab Penggunaan kebijakan yang menampilkan peran IAM apa pun yang saat ini menggunakan kebijakan ini.
4. Untuk setiap peran IAM yang saat ini menggunakan `AmazonEC2ContainerServiceFullAccess` kebijakan, pilih peran dan gunakan langkah-langkah berikut untuk melepaskan kebijakan yang dihapus secara bertahap dan melampirkan kebijakan. `AmazonECS_FullAccess`
    a. Pada tab Izin, pilih X di sebelah kebijakan AmazonEC2 ContainerServiceFullAccess.
    b. Pilih Tambahkan izin.

c. Pilih Lampirkan kebijakan yang ada secara langsung, cari dan pilih FullAccess kebijakan AmazonECS_, lalu pilih Berikutnya: Tinjau.

d. Tinjau perubahan lalu pilih Tambahkan izin.

e. Ulangi langkah-langkah ini untuk setiap grup, pengguna, atau peran yang menggunakan `AmazonEC2ContainerServiceFullAccess` kebijakan.

Untuk memperbarui grup, pengguna, atau peran untuk menggunakan **`AmazonECS_FullAccess`** kebijakan (AWS CLI)

1. Gunakan generate-service-last-accessed-detailsperintah untuk membuat laporan yang menyertakan detail tentang kapan kebijakan yang dihapus terakhir kali digunakan.

```
aws iam generate-service-last-accessed-details \
     --arn arn:aws:iam::aws:policy/AmazonEC2ContainerServiceFullAccess
```

Contoh output:

```
{
    "JobId": "32bb1fb0-1ee0-b08e-3626-ae83EXAMPLE"
}
```

2. Gunakan ID pekerjaan dari output sebelumnya dengan get-service-last-accessed-detailsperintah untuk mengambil laporan layanan yang terakhir diakses. Laporan ini menampilkan Nama Sumber Daya Amazon (ARN) entitas IAM yang terakhir menggunakan kebijakan penghapusan bertahap.

```
aws iam get-service-last-accessed-details \
      --job-id 32bb1fb0-1ee0-b08e-3626-ae83EXAMPLE
```

3. Gunakan salah satu perintah berikut untuk melepaskan `AmazonEC2ContainerServiceFullAccess` kebijakan dari grup, pengguna, atau peran.

   - detach-group-policy
   - detach-role-policy
   - detach-user-policy

4. Gunakan salah satu perintah berikut untuk melampirkan `AmazonECS_FullAccess` kebijakan ke grup, pengguna, atau peran.

- attach-group-policy
- attach-role-policy
- attach-user-policy

Menggunakan peran terkait layanan untuk Amazon ECS

Amazon Elastic Container Service menggunakan AWS Identity and Access Management peran terkait layanan (IAM). Peran terkait layanan adalah jenis peran IAM unik yang ditautkan langsung ke Amazon ECS. Peran terkait layanan telah ditentukan sebelumnya oleh Amazon ECS dan mencakup semua izin yang diperlukan layanan untuk memanggil layanan lain AWS atas nama Anda.

Peran terkait layanan membuat pengaturan Amazon ECS lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. Amazon ECS mendefinisikan izin peran terkait layanannya, dan kecuali ditentukan lain, hanya Amazon ECS yang dapat mengambil perannya. Izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, serta bahwa kebijakan izin tidak dapat dilampirkan ke entitas IAM lainnya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, silakan lihat layanan AWS yang bisa digunakan dengan IAM dan carilah layanan yang memiliki opsi Ya di kolom Peran terkait layanan. Pilih Ya dengan sebuah tautan untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Izin peran terkait layanan untuk Amazon ECS

Amazon ECS menggunakan peran terkait layanan bernama. AWSServiceRoleForECS

Peran AWSServiceRoleForECS terkait layanan mempercayai layanan berikut untuk mengambil peran:

- `ecs.amazonaws.com`

Kebijakan izin peran bernama AmazonECS ServiceRolePolicy memungkinkan Amazon ECS menyelesaikan tindakan berikut pada sumber daya yang ditentukan:

- Tindakan: Saat menggunakan mode `awsvpc` jaringan untuk tugas Amazon ECS Anda, Amazon ECS mengelola siklus hidup antarmuka jaringan elastis yang terkait dengan tugas tersebut. Ini juga termasuk tag yang ditambahkan Amazon ECS ke antarmuka jaringan elastis Anda.

- Tindakan: Saat menggunakan penyeimbang beban dengan layanan Amazon ECS Anda, Amazon ECS mengelola pendaftaran dan deregistrasi sumber daya dengan penyeimbang beban.
- Tindakan: Saat menggunakan penemuan layanan Amazon ECS, Amazon ECS mengelola Route 53 dan AWS Cloud Map sumber daya yang diperlukan agar penemuan layanan berfungsi.
- Tindakan: Saat menggunakan penskalaan otomatis layanan Amazon ECS, Amazon ECS mengelola sumber daya Auto Scaling yang diperlukan.
- Tindakan: Amazon ECS membuat dan mengelola CloudWatch alarm dan aliran log yang membantu pemantauan sumber daya Amazon ECS Anda.
- Tindakan: Saat menggunakan Amazon ECS Exec, Amazon ECS mengelola izin yang diperlukan untuk memulai sesi Amazon ECS Exec ke tugas Anda.
- Tindakan: Saat menggunakan Amazon ECS Service Connect, Amazon ECS mengelola AWS Cloud Map sumber daya yang diperlukan untuk menggunakan fitur tersebut.
- Tindakan: Saat menggunakan penyedia kapasitas Amazon ECS, Amazon ECS mengelola izin yang diperlukan untuk memodifikasi grup Auto Scaling dan instans Amazon EC2-nya.

Anda harus mengonfigurasi izin untuk mengizinkan entitas IAM (seperti pengguna, grup, atau peran) untuk membuat, mengedit, atau menghapus peran terkait layanan. Untuk informasi selengkapnya, lihat Izin peran tertaut layanan dalam Panduan Pengguna IAM.

Membuat peran terkait layanan untuk Amazon ECS

Dalam kebanyakan kasus, Anda tidak perlu membuat peran terkait layanan secara manual. Saat Anda membuat klaster atau membuat atau memperbarui layanan di AWS Management Console, API AWS CLI, atau AWS API, Amazon ECS membuat peran terkait layanan untuk Anda. Jika Anda tidak melihat AWSServiceRoleForECSperan setelah membuat klaster, lakukan hal berikut untuk memperbaiki masalah:

- Verifikasi dan konfigurasikan izin untuk mengizinkan Amazon ECS membuat, mengedit, atau menghapus peran terkait layanan atas nama Anda. Untuk informasi selengkapnya, lihat Izin peran terkait layanan dalam Panduan Pengguna IAM.
- Coba lagi operasi pembuatan klaster, atau buat peran terkait layanan secara manual.

  Anda dapat menggunakan konsol IAM untuk membuat peran AWSServiceRoleForECSterkait layanan. Di AWS CLI atau AWS API, buat peran terkait layanan dengan nama `ecs.amazonaws.com` layanan. Untuk informasi selengkapnya, silakan lihat Membuat peran terkait layanan dalam Panduan Pengguna IAM.

> **Important**
>
> Peran tertaut layanan ini dapat muncul di akun Anda jika Anda menyelesaikan tindakan di layanan lain yang menggunakan fitur yang disupport oleh peran ini.

Jika Anda menghapus peran terkait layanan ini, dan ingin membuatnya lagi, Anda dapat mengulangi proses yang sama untuk membuat kembali peran tersebut di akun Anda. Saat Anda membuat klaster atau membuat atau memperbarui layanan, Amazon ECS akan membuat peran terkait layanan untuk Anda lagi.

Jika Anda menghapus peran tertaut layanan ini, Anda dapat menggunakan proses IAM yang sama untuk membuat ulang peran tersebut.

Mengedit peran terkait layanan untuk Amazon ECS

Amazon ECS tidak mengizinkan Anda mengedit peran AWSServiceRoleForECS terkait layanan. Setelah membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin merujuk peran tersebut. Namun, Anda dapat mengedit deskripsi peran ini menggunakan IAM. Untuk informasi selengkapnya, lihat Mengedit peran terkait layanan dalam Panduan Pengguna IAM.

Menghapus peran terkait layanan untuk Amazon ECS

Jika tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran tertaut layanan, sebaiknya Anda menghapus peran tersebut. Dengan begitu, Anda tidak memiliki entitas yang tidak digunakan yang tidak dipantau atau dipelihara secara aktif. Tetapi, Anda harus membersihkan sumber daya peran yang terhubung dengan layanan sebelum menghapusnya secara manual.

> **Note**
>
> Jika layanan Amazon ECS menggunakan peran saat Anda mencoba menghapus sumber daya, maka penghapusan mungkin gagal. Jika hal itu terjadi, tunggu beberapa menit dan coba mengoperasikannya lagi.

Untuk memeriksa apakah peran yang terhubung dengan layanan memiliki sesi aktif

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.

2. Di panel navigasi, pilih Peran dan pilih nama AWSServiceRoleForECS (bukan kotak centang).
3. Di halaman Summary (Ringkasan), pilih Access Advisor (Penasihat Akses) dan tinjau aktivitas terbaru untuk peran yang terhubung dengan layanan.

> **Note**
>
> Jika Anda tidak yakin apakah Amazon ECS menggunakan AWSServiceRoleForECS peran tersebut, Anda dapat mencoba menghapus peran tersebut. Jika layanan menggunakan peran tersebut, maka penghapusan gagal dan Anda dapat melihat wilayah tempat peran tersebut digunakan. Jika peran tersebut sedang digunakan, Anda harus menunggu hingga sesi ini berakhir sebelum dapat menghapus peran tersebut. Anda tidak dapat mencabut sesi untuk peran terkait layanan.

Untuk menghapus sumber daya Amazon ECS yang digunakan oleh peran terkait AWSServiceRoleForECS layanan

Anda harus menghapus semua klaster Amazon ECS di semua AWS Wilayah sebelum Anda dapat menghapus peran. AWSServiceRoleForECS

1. Skala semua layanan Amazon ECS ke hitungan 0 yang diinginkan di semua wilayah, lalu hapus layanan. Untuk informasi selengkapnya, lihat Memperbarui layanan menggunakan konsol dan Menghapus layanan menggunakan konsol.
2. Paksa hapus semua registrasi instans kontainer dari semua klaster di semua wilayah. Untuk informasi selengkapnya, lihat Membatalkan pendaftaran instans kontainer yang didukung Amazon EC2.
3. Hapus semua cluster Amazon ECS di semua wilayah. Untuk informasi selengkapnya, lihat Menghapus cluster menggunakan konsol.

Untuk menghapus peran terkait layanan secara manual menggunakan IAM

Gunakan konsol IAM, the AWS CLI, atau AWS API untuk menghapus peran AWSServiceRoleForECS terkait layanan. Untuk informasi selengkapnya, lihat Menghapus peran terkait layanan dalam Panduan Pengguna IAM.

Wilayah yang didukung untuk peran terkait layanan Amazon ECS

Amazon ECS mendukung penggunaan peran terkait layanan di semua wilayah tempat layanan tersedia. Untuk informasi selengkapnya, silakan lihat Wilayah AWS dan titik akhir.

Izin diperlukan untuk konsol Amazon ECS

Mengikuti praktik terbaik pemberian hak istimewa paling rendah, Anda dapat menggunakan kebijakan terkelola `AmazonECS_FullAccess` sebagai templat untuk membuat kebijakan kustom Anda sendiri. Dengan begitu, Anda dapat mengambil atau menambahkan izin ke dan dari kebijakan terkelola berdasarkan kebutuhan khusus Anda. Untuk informasi selengkapnya, lihat Detail izin.

Konsol Amazon ECS didukung oleh AWS CloudFormation dan memerlukan izin IAM tambahan dalam kasus berikut:

- Membuat klaster
- Membuat sebuah layanan
- Menciptakan penyedia kapasitas

Anda dapat membuat kebijakan untuk izin tambahan, lalu melampirkannya ke peran IAM yang Anda gunakan untuk mengakses konsol. Untuk informasi selengkapnya, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Izin IAM diperlukan untuk membuat cluster

Saat membuat klaster di konsol, Anda memerlukan izin tambahan yang memberi Anda izin untuk mengelola AWS CloudFormation tumpukan.

Izin tambahan berikut diperlukan:

- `cloudformation`— Memungkinkan kepala sekolah untuk membuat dan mengelola tumpukan. AWS CloudFormation Ini diperlukan saat membuat cluster Amazon ECS menggunakan AWS Management Console dan pengelolaan cluster tersebut selanjutnya.

Kebijakan berikut berisi AWS CloudFormation izin yang diperlukan, dan membatasi tindakan ke sumber daya yang dibuat di konsol Amazon ECS.

```
{
```

```
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStack*",
                "cloudformation:UpdateStack"
             ],
            "Resource": [
                "arn:*:cloudformation:*:*:stack/Infra-ECS-Cluster-*"
            ]
      }
   ]
}
```

Jika Anda belum membuat peran instans penampung Amazon ECS (`ecsInstanceRole`), dan Anda membuat klaster yang menggunakan instans Amazon EC2, konsol akan membuat peran tersebut atas nama Anda.

Selain itu, jika Anda menggunakan grup Auto Scaling, maka Anda memerlukan izin tambahan agar konsol dapat menambahkan tag ke grup penskalaan otomatis saat menggunakan fitur penskalaan otomatis cluster.

Izin tambahan berikut diperlukan:

- `autoscaling`— Memungkinkan konsol untuk menandai grup Auto Scaling Amazon EC2. Ini diperlukan saat mengelola grup penskalaan otomatis Amazon EC2 saat menggunakan fitur penskalaan otomatis cluster. Tag adalah tag yang dikelola ECS yang secara otomatis ditambahkan konsol ke grup untuk menunjukkan dibuat di konsol.
- `iam`— Memungkinkan kepala sekolah untuk membuat daftar peran IAM dan kebijakan terlampir mereka. Prinsipal juga dapat mencantumkan profil instans yang tersedia untuk instans Amazon EC2 Anda.

Kebijakan berikut berisi izin IAM yang diperlukan, dan membatasi tindakan untuk peran. `ecsInstanceRole`

Izin Auto Scaling tidak terbatas.

```
{
```

```
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "iam:AttachRolePolicy",
              "iam:CreateRole",
              "iam:CreateInstanceProfile",
              "iam:AddRoleToInstanceProfile",
              "iam:ListInstanceProfilesForRole",
              "iam:GetRole"
            ],
            "Resource": "arn:aws:iam::*:role/ecsInstanceRole"
        },
        {
            "Effect": "Allow",
            "Action": "autoscaling:CreateOrUpdateTags",
            "Resource": "*"
        }
    ]
}
```

Izin IAM diperlukan untuk membuat penyedia kapasitas

Saat membuat layanan di konsol, Anda memerlukan izin tambahan yang memberi Anda izin untuk mengelola AWS CloudFormation tumpukan. Izin tambahan berikut diperlukan:

- `cloudformation`— Memungkinkan kepala sekolah untuk membuat dan mengelola tumpukan. AWS CloudFormation Ini diperlukan saat membuat penyedia kapasitas Amazon ECS menggunakan AWS Management Console dan pengelolaan berikutnya dari penyedia kapasitas tersebut.

Kebijakan berikut berisi izin yang diperlukan, dan membatasi tindakan ke sumber daya yang dibuat di konsol Amazon ECS.

```
{
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStack*",
```

```
                "cloudformation:UpdateStack"
            ],
            "Resource": [
                "arn:*:cloudformation:*:*:stack/Infra-ECS-CapacityProvider-*"
            ]
        }
    ]
}
```

Izin IAM diperlukan untuk membuat layanan

Saat membuat layanan di konsol, Anda memerlukan izin tambahan yang memberi Anda izin untuk mengelola AWS CloudFormation tumpukan. Izin tambahan berikut diperlukan:

- `cloudformation`— Memungkinkan kepala sekolah untuk membuat dan mengelola tumpukan. AWS CloudFormation Ini diperlukan saat membuat layanan Amazon ECS menggunakan AWS Management Console dan pengelolaan selanjutnya dari layanan tersebut.

Kebijakan berikut berisi izin yang diperlukan, dan membatasi tindakan ke sumber daya yang dibuat di konsol Amazon ECS.

```
{
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStack*",
                "cloudformation:UpdateStack"
            ],
            "Resource": [
                "arn:*:cloudformation:*:*:stack/ECS-Console-V2-Service-*"
            ]
      }
   ]
}
```

Izin untuk membuat peran IAM

Tindakan berikut memerlukan izin tambahan untuk menyelesaikan operasi:

- Mendaftarkan instance eksternal - untuk informasi selengkapnya, lihat Peran IAM ECS Anywhere
- Mendaftarkan definisi tugas - untuk informasi lebih lanjut, lihat Peran IAM eksekusi tugas Amazon ECS
- Membuat EventBridge aturan yang akan digunakan untuk menjadwalkan tugas - untuk informasi selengkapnya, lihat Peran Amazon ECS EventBridge IAM

Anda dapat menambahkan izin ini dengan membuat peran di IAM sebelum menggunakannya di konsol Amazon ECS. Jika Anda tidak membuat peran, konsol Amazon ECS akan membuatnya atas nama Anda.

Izin IAM diperlukan untuk mendaftarkan instans eksternal ke cluster

Anda memerlukan izin tambahan saat mendaftarkan instance eksternal ke klaster dan Anda ingin membuat peran instance eksternal (`escExternalInstanceRole`) baru.

Izin tambahan berikut diperlukan:

- `iam`— Memungkinkan kepala sekolah untuk membuat dan membuat daftar peran IAM dan kebijakan terlampir mereka.
- `ssm`— Memungkinkan prinsipal untuk mendaftarkan instance eksternal dengan Systems Manager.

> **Note**
>
> Untuk memilih yang sudah ada`escExternalInstanceRole`, Anda harus memiliki `iam:GetRole` dan `iam:PassRole` izin.

Kebijakan berikut berisi izin yang diperlukan, dan membatasi tindakan untuk `escExternalInstanceRole` peran.

```
{
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "iam:AttachRolePolicy",
              "iam:CreateRole",
              "iam:CreateInstanceProfile",
```

```
                "iam:AddRoleToInstanceProfile",
                "iam:ListInstanceProfilesForRole",
                "iam:GetRole"
            ],
            "Resource": "arn:aws:iam::*:role/escExternalInstanceRole"
        },
        {
            "Effect": "Allow",
            "Action": ["iam:PassRole", "ssm:CreateActivation"],
            "Resource": "arn:aws:iam::*:role/escExternalInstanceRole"
        }
    ]
}
```

Izin IAM diperlukan untuk mendaftarkan definisi tugas

Anda memerlukan izin tambahan saat mendaftarkan definisi tugas dan Anda ingin membuat peran eksekusi tugas (`ecsTaskExecutionRole`) baru.

Izin tambahan berikut diperlukan:

- `iam`— Memungkinkan kepala sekolah untuk membuat dan membuat daftar peran IAM dan kebijakan terlampir mereka.

> **Note**
>
> Untuk memilih yang sudah ada`ecsTaskExecutionRole`, Anda harus memiliki `iam:GetRole` izin.

Kebijakan berikut berisi izin yang diperlukan, dan membatasi tindakan untuk `ecsTaskExecutionRole` peran.

```
{
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "iam:AttachRolePolicy",
              "iam:CreateRole",
              "iam:GetRole"
```

```
            ],
            "Resource": "arn:aws:iam::*:role/ecsTaskExecutionRole"
        }
    ]
}
```

Izin IAM diperlukan untuk membuat EventBridge aturan untuk tugas terjadwal

Anda memerlukan izin tambahan saat menjadwalkan tugas dan Anda ingin membuat peran peran (`ecsEventsRole`) CloudWatch Acara baru.

Izin tambahan berikut diperlukan:

- `iam`— Memungkinkan kepala sekolah untuk membuat dan mencantumkan peran IAM dan kebijakan terlampirnya, dan mengizinkan Amazon ECS meneruskan peran tersebut ke layanan lain untuk mengambil peran tersebut.

> **Note**
>
> Untuk memilih yang sudah ada`ecsEventsRole`, Anda harus memiliki `iam:GetRole` dan `iam:PassRole` izin.

Kebijakan berikut berisi izin yang diperlukan, dan membatasi tindakan untuk `ecsEventsRole` peran.

```
{
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "iam:AttachRolePolicy",
              "iam:CreateRole",
              "iam:GetRole",
              "iam: PassRole"
            ],
            "Resource": "arn:aws:iam::*:role/ecsEventsRole"
        }
    ]
}
```

Peran IAM eksekusi tugas Amazon ECS

Peran eksekusi tugas memberikan izin kepada kontainer Amazon ECS dan agen Fargate untuk melakukan panggilan AWS API atas nama Anda. IAM role eksekusi tugas diperlukan sesuai dengan persyaratan tugas Anda. Anda dapat memiliki beberapa peran eksekusi tugas untuk berbagai tujuan dan layanan yang terkait dengan akun Anda. Untuk izin IAM yang perlu dijalankan aplikasi Anda, lihat. Tugas peran IAM

Berikut ini adalah kasus penggunaan umum untuk peran IAM eksekusi tugas:

- Tugas Anda di-host pada AWS Fargateatau pada instans eksternal dan:
  - sedang menarik gambar kontainer dari repositori pribadi Amazon ECR.
  - sedang menarik gambar kontainer dari repositori pribadi Amazon ECR di akun berbeda dari akun yang menjalankan tugas.
  - mengirim log kontainer ke CloudWatch Log menggunakan driver `awslogs` log. Untuk informasi selengkapnya, lihat Menggunakan driver log awslogs.
- Tugas Anda di-host di salah satu AWS Fargate atau instans Amazon EC2 dan...
  - menggunakan autentikasi registri privat. Untuk informasi selengkapnya, lihat Izin IAM yang diperlukan untuk otentikasi registri pribadi.
  - menggunakan Runtime Monitoring.
  - definisi tugas merujuk data sensitif menggunakan rahasia Secrets Manager atau parameter AWS Systems Manager Parameter Store. Untuk informasi selengkapnya, lihat Izin IAM yang diperlukan untuk rahasia Amazon ECS.

> **Note**
>
> Peran eksekusi tugas didukung oleh agen penampung Amazon ECS versi 1.16.0 dan yang lebih baru.

Amazon ECS menyediakan kebijakan terkelola bernama `AmazonECSTaskExecutionRolePolicy` yang berisi izin yang diperlukan oleh kasus penggunaan umum yang dijelaskan di atas. Mungkin perlu menambahkan kebijakan sebaris ke peran eksekusi tugas Anda untuk kasus penggunaan khusus yang diuraikan di bawah ini.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

Peran eksekusi tugas Amazon ECS dapat dibuat untuk Anda di konsol Amazon ECS; namun, Anda harus secara manual melampirkan kebijakan IAM terkelola untuk tugas agar Amazon ECS menambahkan izin untuk fitur dan penyempurnaan future saat diperkenalkan. Anda dapat menggunakan pencarian konsol IAM untuk mencari `ecsTaskExecutionRole` dan melihat apakah akun Anda sudah memiliki peran eksekusi tugas. Untuk informasi selengkapnya, lihat pencarian konsol IAM di panduan pengguna IAM.

Membuat peran eksekusi tugas (**`ecsTaskExecutionRole`**)

Jika akun Anda belum memiliki peran eksekusi tugas, gunakan langkah-langkah berikut untuk membuat peran tersebut.

AWS Management Console

Untuk membuat peran IAM eksekusi tugas ()AWS Management Console

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Dalam panel navigasi, pilih Roles (Peran), lalu Create role (Buat peran).
3. Di bagian Jenis entitas tepercaya, pilih AWS layanan, Layanan Kontainer Elastis.
4. Untuk kasus penggunaan, pilih Elastic Container Service Task, dan kemudian pilih Berikutnya.
5. Di bagian Tambahkan izin, lakukan hal berikut:

a. Cari AmazonECS, TaskExecutionRolePolicy lalu pilih kebijakan.

b. Di bawah Setel batas izin - opsional, pilih Buat peran tanpa batas izin.

c. Pilih Berikutnya.

6. Di bawah Nama, tinjau, dan buat, lakukan hal berikut:

a. Untuk Nama Peran, ketik `ecsTaskExecutionRole`.

b. Untuk Tambahkan tag (opsional), tentukan tag kustom apa pun yang akan dikaitkan dengan kebijakan.

7. Pilih Buat peran.

AWS CLI

Ganti semua *`input pengguna`* dengan informasi Anda sendiri.

1. Buat file bernama `ecs-tasks-trust-policy.json` yang berisi kebijakan kepercayaan yang akan digunakan untuk peran IAM. File tersebut harus berisi hal berikut:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs-tasks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. Buat peran IAM bernama `ecsTaskExecutionRole` menggunakan kebijakan kepercayaan yang dibuat pada langkah sebelumnya.

```
aws iam create-role \
      --role-name ecsTaskExecutionRole \
      --assume-role-policy-document file://ecs-tasks-trust-policy.json
```

3. Lampirkan `AmazonECSTaskExecutionRolePolicy` kebijakan yang AWS dikelola ke `ecsTaskExecutionRole` peran.

```
aws iam attach-role-policy \
      --role-name ecsTaskExecutionRole \
      --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy
```

Izin IAM yang diperlukan untuk otentikasi registri pribadi

Peran eksekusi tugas Amazon ECS diperlukan untuk menggunakan fitur ini. Hal ini mengizinkan agen kontainer untuk menarik citra kontainer.

Untuk memberikan akses ke rahasia yang Anda buat, tambahkan izin berikut sebagai kebijakan inline ke peran eksekusi tugas. Untuk informasi selengkapnya, lihat Menambahkan dan Menghapus Kebijakan IAM.

- `secretsmanager:GetSecretValue`
- `kms:Decrypt`—Diperlukan hanya jika kunci Anda menggunakan kunci KMS khusus dan bukan kunci default. Nama Sumber Daya Amazon (ARN) untuk kunci kustom Anda harus ditambahkan sebagai sumber daya.

Berikut ini adalah contoh kebijakan inline yang menambahkan izin.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:secret_name",
                "arn:aws:kms:<region>:<aws_account_id>:key/key_id"
            ]
        }
    ]
```

```
}
```

Izin IAM yang diperlukan untuk rahasia Amazon ECS

Untuk menggunakan fitur rahasia Amazon ECS, Anda harus memiliki peran eksekusi tugas Amazon ECS dan mereferensikannya dalam definisi tugas Anda. Hal ini memungkinkan agen kontainer untuk menarik sumber daya Secrets Manager yang diperlukan AWS Systems Manager atau Secrets. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah.

Menggunakan Secrets Manager

Untuk memberikan akses ke rahasia Secrets Manager yang Anda buat, tambahkan izin berikut secara manual ke peran eksekusi tugas. Untuk informasi tentang cara mengelola izin, lihat Menambahkan dan Menghapus izin identitas IAM di Panduan Pengguna IAM.

- `secretsmanager:GetSecretValue`— Diperlukan jika Anda mereferensikan rahasia Secrets Manager. Menambahkan izin untuk mengambil rahasia dari Secrets Manager.

Contoh kebijakan berikut menambahkan izin yang diperlukan.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": [
        "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name"
      ]
    }
  ]
}
```

Menggunakan Systems Manager

> **Important**
>
> Untuk tugas yang menggunakan tipe peluncuran EC2, Anda harus menggunakan variabel konfigurasi agen ECS `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE=true` untuk

menggunakan fitur ini. Anda dapat menambahkannya ke file `./etc/ecs/ecs.config` selama pembuatan instans kontainer atau Anda dapat menambahkannya ke instans yang ada, lalu memulai ulang agen ECS. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.

Untuk memberikan akses ke parameter Penyimpanan Parameter Systems Manager yang Anda buat, tambahkan izin berikut secara manual sebagai kebijakan ke peran eksekusi tugas. Untuk informasi tentang cara mengelola izin, lihat Menambahkan dan Menghapus izin identitas IAM di Panduan Pengguna IAM.

- `ssm:GetParameters`— Diperlukan jika Anda mereferensikan parameter Systems Manager Parameter Store dalam definisi tugas. Menambahkan izin untuk mengambil parameter Systems Manager.
- `secretsmanager:GetSecretValue`— Diperlukan jika Anda mereferensikan rahasia Secrets Manager baik secara langsung atau jika parameter Parameter Store Systems Manager Anda mereferensikan rahasia Secrets Manager dalam definisi tugas. Menambahkan izin untuk mengambil rahasia dari Secrets Manager.
- `kms:Decrypt`— Diperlukan hanya jika rahasia Anda menggunakan kunci yang dikelola pelanggan dan bukan kunci default. ARN untuk kunci khusus Anda harus ditambahkan sebagai sumber daya. Menambahkan izin untuk mendekripsi kunci yang dikelola pelanggan.

Contoh kebijakan berikut menambahkan izin yang diperlukan:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetParameters",
        "secretsmanager:GetSecretValue",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:ssm:region:aws_account_id:parameter/parameter_name",
        "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name",
        "arn:aws:kms:region:aws_account_id:key/key_id"
      ]
```

```
    }
  ]
}
```

Izin IAM opsional untuk tugas Fargate yang menarik gambar Amazon ECR melalui titik akhir antarmuka

Saat meluncurkan tugas yang menggunakan tipe peluncuran Fargate yang menarik gambar dari Amazon ECR saat Amazon ECR dikonfigurasi untuk menggunakan titik akhir VPC antarmuka, Anda dapat membatasi akses tugas ke titik akhir VPC atau VPC tertentu. Lakukan ini dengan membuat peran eksekusi tugas untuk tugas yang akan digunakan yang menggunakan kunci kondisi IAM.

Gunakan kunci kondisi global IAM berikut untuk membatasi akses ke titik akhir VPC atau VPC tertentu. Untuk informasi selengkapnya, lihat Kunci Konteks Syarat Global AWS.

- `aws:SourceVpc`—Membatasi akses ke VPC tertentu.
- `aws:SourceVpce`—Membatasi akses ke titik akhir VPC tertentu.

Kebijakan peran eksekusi tugas berikut memberikan contoh untuk menambahkan kunci syarat:

> **Important**
>
> Tindakan `ecr:GetAuthorizationToken` API tidak dapat memiliki `aws:sourceVpc` atau `aws:sourceVpce` kunci-kunci kondisi yang diterapkan padanya karena panggilan GetAuthorizationToken API melewati elastic network interface yang dimiliki oleh AWS Fargate daripada elastic network interface dari tugas tersebut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:sourceVpce": "vpce-xxxxxx",
                    "aws:sourceVpc": "vpc-xxxxx"
                }
            }
        }
    ]
}
```

Tugas peran IAM

Tugas Amazon ECS Anda dapat memiliki peran IAM yang terkait dengannya. Izin yang diberikan dalam peran IAM diasumsikan oleh kontainer yang berjalan dalam tugas. Untuk izin IAM yang dibutuhkan Amazon ECS untuk menarik gambar kontainer dan menjalankan tugas, lihat. Peran IAM eksekusi tugas Amazon ECS

Jika aplikasi kontainer Anda perlu memanggil AWS API, mereka harus menandatangani permintaan AWS API mereka dengan AWS kredensialnya, dan peran IAM tugas menyediakan strategi untuk mengelola kredensyal untuk digunakan aplikasi Anda, mirip dengan cara profil instans Amazon EC2 memberikan kredensyal ke instans Amazon EC2. Alih-alih membuat dan mendistribusikan AWS kredensyal Anda ke container atau menggunakan peran instans Amazon EC2, Anda dapat mengaitkan peran IAM dengan definisi tugas Amazon ECS atau operasi API. `RunTask` Container Anda kemudian dapat menggunakan AWS SDK atau AWS CLI membuat permintaan API ke AWS layanan resmi.

Berikut ini menjelaskan manfaat menggunakan peran IAM dengan tugas Anda.

- Isolasi Kredensial: Kontainer hanya dapat mengambil kredensial untuk IAM role yang ditetapkan di dalam ketentuan tugas miliknya; kontainer yang tidak pernah memiliki akses ke kredensial yang dimaksudkan untuk kontainer lain milik tugas lain.

- Otorisasi: Kontainer tidak sah tidak dapat mengakses kredensial IAM role yang ditetapkan untuk tugas lainnya.
- Auditabilitas: Akses dan pencatatan peristiwa tersedia CloudTrail untuk memastikan audit retrospektif. Kredensyal tugas memiliki konteks `taskArn` yang dilampirkan ke sesi, sehingga CloudTrail log menunjukkan tugas mana yang menggunakan peran mana.

> **Note**
>
> Saat Anda menentukan peran IAM untuk tugas, SDK AWS CLI atau lainnya dalam wadah untuk tugas tersebut menggunakan AWS kredensyal yang disediakan oleh peran tugas secara eksklusif dan mereka tidak lagi mewarisi izin IAM apa pun dari Amazon EC2 atau instans eksternal yang dijalankan.

Anda dapat menentukan peran IAM tugas dalam definisi tugas, atau Anda dapat menggunakan `taskRoleArn` penggantian saat menjalankan tugas secara manual dengan operasi `RunTask` API. Agen Amazon ECS menerima pesan payload untuk memulai tugas dengan bidang tambahan yang berisi kredensyal peran. Agen Amazon ECS menetapkan ID kredensyal tugas unik sebagai token identifikasi dan memperbarui cache kredensyal internalnya sehingga token identifikasi untuk tugas menunjuk ke kredensyal peran yang diterima dalam muatan. Agen Amazon ECS mengisi variabel `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` lingkungan dalam `Env` objek (tersedia dengan docker inspect ***container_id*** perintah) untuk semua kontainer yang termasuk dalam tugas ini dengan URI relatif berikut:. `/`*credential_provider_version*`/credentials?id=`*task_credential_id*

Dari dalam kontainer, Anda dapat melakukan kueri terhadap titik akhir kredensial dengan perintah berikut:

```
curl 169.254.170.2$AWS_CONTAINER_CREDENTIALS_RELATIVE_URI
```

Output:

```
{
    "AccessKeyId": "ACCESS_KEY_ID",
    "Expiration": "EXPIRATION_DATE",
    "RoleArn": "TASK_ROLE_ARN",
    "SecretAccessKey": "SECRET_ACCESS_KEY",
    "Token": "SECURITY_TOKEN_STRING"
```

```
}
```

Jika instans Amazon EC2 Anda menggunakan setidaknya versi agen `1.11.0` penampung dan versi AWS CLI atau SDK yang didukung, klien SDK akan melihat bahwa `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` variabel tersebut tersedia, dan akan menggunakan kredensyal yang disediakan untuk melakukan panggilan ke API. AWS Untuk informasi selengkapnya, lihat Menggunakan peran IAM tugas di Amazon EC2 atau instans eksternal.

Setiap kali penyedia kredensial digunakan, permintaan dicatat secara lokal pada instans kontainer host di `/var/log/ecs/audit.log.`*`YYYY`*`-`*`MM`*`-`*`DD`*`-`*`HH`*. Untuk informasi selengkapnya, lihat Peran IAM untuk Log Audit Kredensyal Tugas.

Pertimbangan untuk tugas yang dihosting di instans Amazon EC2

Saat menggunakan peran IAM dengan tugas Anda yang berjalan di instans Amazon EC2, container tidak dicegah untuk mengakses kredensyal yang diberikan ke profil instans Amazon EC2 (melalui server metadata instans Amazon EC2). Sebaiknya Anda membatasi izin dalam peran instance container Anda ke daftar izin minimal yang digunakan dalam kebijakan IAM `AmazonEC2ContainerServiceforEC2Role` terkelola. Untuk informasi selengkapnya, lihat Peran IAM instans wadah Amazon ECS.

Berikut ini juga harus dipertimbangkan saat menggunakan peran IAM tugas untuk tugas yang dihosting di instans Amazon EC2.

- Untuk mencegah kontainer yang dijalankan oleh tugas yang menggunakan mode `awsvpc` jaringan mengakses informasi kredensyal yang diberikan ke profil instans Amazon EC2, sambil tetap mengizinkan izin yang disediakan oleh peran tugas, setel `ECS_AWSVPC_BLOCK_IMDS` variabel konfigurasi agen `true` ke dalam file konfigurasi agen dan mulai ulang agen. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.
- Untuk mencegah container yang dijalankan oleh tugas yang menggunakan mode `bridge` jaringan mengakses informasi kredensyal yang diberikan ke profil instans Amazon EC2, sambil tetap mengizinkan izin yang disediakan oleh peran tugas, dengan menjalankan perintah iptables berikut di instans Amazon EC2 Anda. Perintah ini tidak memengaruhi kontainer dalam tugas yang menggunakan mode `host` atau `awsvpc` jaringan. Untuk informasi selengkapnya, lihat Mode jaringan.

  ```
  sudo yum install -y iptables-services; sudo iptables --insert DOCKER-USER 1 --in-interface docker+ --destination 169.254.169.254/32 --jump DROP
  ```

Anda harus menyimpan iptables aturan ini di instans Amazon EC2 Anda agar dapat bertahan dari reboot. Saat menggunakan AMI Amazon ECS yang dioptimalkan, Anda dapat menggunakan perintah berikut. Untuk sistem operasi lain, lihat dokumentasi untuk sistem operasi tersebut.

```
sudo iptables-save | sudo tee /etc/sysconfig/iptables && sudo systemctl enable --now
 iptables
```

Menggunakan peran IAM tugas di Amazon EC2 atau instans eksternal

Amazon EC2 atau instans eksternal Anda memerlukan setidaknya versi agen `1.11.0` penampung untuk menggunakan peran IAM tugas; namun, sebaiknya gunakan versi agen penampung terbaru. Untuk informasi tentang memeriksa versi agen Anda dan memperbarui ke versi terbaru, lihat Memperbarui agen kontainer Amazon ECS. Jika Anda menggunakan AMI Amazon ECS yang dioptimalkan, instans Anda memerlukan setidaknya `1.11.0-1` paket. `ecs-init` Jika instans Anda menggunakan AMI terbaru yang dioptimalkan Amazon ECS, maka instans tersebut berisi versi yang diperlukan dari agen penampung dan. `ecs-init` Untuk informasi selengkapnya, lihat AMI Amazon ECS yang dioptimalkan.

Jika Anda tidak menggunakan AMI Amazon ECS yang dioptimalkan untuk instance container Anda, pastikan untuk menambahkan `--net=host` opsi ke docker run perintah Anda yang memulai agen dan variabel konfigurasi agen berikut untuk konfigurasi yang Anda inginkan (untuk informasi selengkapnya, lihat): Konfigurasi agen kontainer Amazon ECS

`ECS_ENABLE_TASK_IAM_ROLE=true`

Menggunakan peran IAM untuk tugas untuk kontainer dengan mode `bridge` dan `default` jaringan.

`ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST=true`

Menggunakan peran IAM untuk tugas untuk kontainer dengan mode `host` jaringan. Variabel ini hanya didukung pada agen versi 1.12.0 dan yang lebih baru.

Untuk contoh run command, lihat Memperbarui agen kontainer Amazon ECS secara manual (untuk AMI yang dioptimalkan ECS non-Amazon). Anda juga perlu mengatur perintah jaringan berikut pada instance container Anda sehingga container dalam tugas Anda dapat mengambil AWS kredensialnya:

```
sudo sysctl -w net.ipv4.conf.all.route_localnet=1
```

```
sudo iptables -t nat -A PREROUTING -p tcp -d 169.254.170.2 --dport 80 -j DNAT --to-
destination 127.0.0.1:51679
sudo iptables -t nat -A OUTPUT -d 169.254.170.2 -p tcp -m tcp --dport 80 -j REDIRECT --
to-ports 51679
```

Anda harus menyimpan aturan iptables ini pada instans kontainer Anda agar dapat bertahan saat booting ulang. Anda dapat menggunakan perintah iptables-save dan iptables-restore untuk menyimpan aturan iptables dan memulihkannya saat booting. Untuk informasi selengkapnya, konsultasikan dokumentasi sistem operasi tertentu Anda.

Membuat peran dan kebijakan IAM untuk tugas Anda

Saat membuat kebijakan IAM untuk tugas yang akan digunakan, kebijakan tersebut harus menyertakan izin yang Anda inginkan untuk diasumsikan oleh container dalam tugas Anda. Anda dapat menggunakan kebijakan AWS terkelola yang ada, atau Anda dapat membuat kebijakan khusus dari awal yang memenuhi kebutuhan spesifik Anda. Untuk informasi selengkapnya, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

> **Important**
>
> Untuk tugas Amazon ECS (untuk semua jenis peluncuran), sebaiknya gunakan kebijakan dan peran IAM untuk tugas Anda. Kredensyal ini memungkinkan tugas Anda untuk membuat permintaan AWS API tanpa memanggil `sts:AssumeRole` untuk mengambil peran yang sama yang sudah dikaitkan dengan tugas. Jika tugas Anda mengharuskan peran mengasumsikan dirinya sendiri, Anda harus membuat kebijakan kepercayaan yang secara eksplisit memungkinkan peran itu untuk mengambil alih dirinya sendiri. Untuk informasi selengkapnya, lihat Memodifikasi kebijakan kepercayaan peran dalam Panduan Pengguna IAM.

Setelah kebijakan IAM dibuat, Anda dapat membuat peran IAM yang menyertakan kebijakan yang Anda referensikan dalam definisi tugas Amazon ECS Anda. Anda dapat membuat peran menggunakan kasus penggunaan Elastic Container Service Task di konsol IAM. Kemudian, Anda dapat melampirkan kebijakan IAM spesifik Anda ke peran yang memberikan kontainer dalam tugas Anda izin yang Anda inginkan. Prosedur di bawah menjelaskan cara melakukannya.

Jika Anda memiliki beberapa definisi tugas atau layanan yang memerlukan izin IAM, Anda harus mempertimbangkan untuk membuat peran untuk setiap definisi tugas atau layanan tertentu

dengan izin minimum yang diperlukan untuk tugas yang akan dioperasikan sehingga Anda dapat meminimalkan akses yang Anda berikan untuk setiap tugas.

Untuk informasi tentang titik akhir layanan untuk Wilayah Anda, lihat Titik akhir layanan di Panduan Referensi Umum Amazon Web Referensi.

Peran tugas IAM harus memiliki kebijakan kepercayaan yang menentukan layanan. `ecs-tasks.amazonaws.com` `sts:AssumeRole`Izin tersebut memungkinkan tugas Anda untuk mengambil peran IAM yang berbeda dari yang digunakan instans Amazon EC2. Dengan cara ini, tugas Anda tidak mewarisi peran yang terkait dengan instans Amazon EC2. Berikut ini adalah contoh kebijakan kepercayaan. Ganti pengenal Wilayah dan tentukan nomor AWS akun yang Anda gunakan saat meluncurkan tugas.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":[
               "ecs-tasks.amazonaws.com"
            ]
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "ArnLike":{
            "aws:SourceArn":"arn:aws:ecs:us-west-2:111122223333:*"
            },
            "StringEquals":{
               "aws:SourceAccount":"111122223333"
            }
         }
      }
   ]
}
```

Important

Saat membuat peran IAM tugas Anda, Anda disarankan untuk menggunakan `aws:SourceAccount` atau kunci `aws:SourceArn` kondisi baik dalam hubungan kepercayaan atau kebijakan IAM yang terkait dengan peran untuk cakupan izin lebih lanjut

untuk mencegah masalah keamanan wakil yang membingungkan. Menggunakan tombol `aws:SourceArn` kondisi untuk menentukan klaster tertentu saat ini tidak didukung, Anda harus menggunakan wildcard untuk menentukan semua cluster. Untuk mempelajari lebih lanjut tentang masalah deputi yang membingungkan dan cara melindungi AWS akun Anda, lihat Masalah wakil yang bingung di Panduan Pengguna IAM.

Prosedur berikut menjelaskan cara membuat kebijakan untuk mengambil objek dari Amazon S3 dengan kebijakan contoh. Ganti semua *`input pengguna`* dengan nilai Anda sendiri.

AWS Management Console

Cara menggunakan editor kebijakan JSON untuk membuat kebijakan

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi di sebelah kiri, pilih Kebijakan.

   Jika ini pertama kalinya Anda memilih Kebijakan, akan muncul halaman Selamat Datang di Kebijakan Terkelola. Pilih Memulai.
3. Di bagian atas halaman, pilih Buat kebijakan.
4. Di bagian Editor kebijakan, pilih opsi JSON.
5. Masukkan dokumen kebijakan JSON berikut:

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject"
         ],
         "Resource":[
            "arn:aws:s3:::my-task-secrets-bucket/*"
         ],
         "Condition":{
            "ArnLike":{
            "aws:SourceArn":"arn:aws:ecs:region:123456789012:*"
            },
            "StringEquals":{
```

```
                "aws:SourceAccount":"123456789012"
            }
         }
      }
   ]
}
```

6. Pilih Berikutnya.

> **Note**
>
> Anda dapat beralih antara opsi editor Visual dan JSON kapan saja. Namun, jika Anda melakukan perubahan atau memilih Berikutnya di editor Visual, IAM dapat merestrukturisasi kebijakan Anda untuk mengoptimalkannya bagi editor visual. Untuk informasi selengkapnya, lihat Restrukturisasi kebijakan dalam Panduan Pengguna IAM.

7. Pada halaman Tinjau dan buat, masukkan Nama kebijakan dan Deskripsi (opsional) untuk kebijakan yang Anda buat. Tinjau Izin yang ditentukan dalam kebijakan ini untuk melihat izin yang diberikan oleh kebijakan Anda.
8. Pilih Buat kebijakan untuk menyimpan kebijakan baru Anda.

AWS CLI

Ganti semua *input pengguna* dengan nilai Anda sendiri.

1. Buat file bernama `s3-policy.json` dengan konten berikut.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject"
         ],
         "Resource":[
            "arn:aws:s3:::my-task-secrets-bucket/*"
         ],
         "Condition":{
            "ArnLike":{
```

```
            "aws:SourceArn":"arn:aws:ecs:region:123456789012:*"
            },
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
            }
         }
      }
   ]
}
```

2. Gunakan perintah berikut untuk membuat kebijakan IAM menggunakan file dokumen kebijakan JSON.

```
aws iam create-policy \
      --policy-name taskRolePolicy \
      --policy-document file://s3-policy.json
```

Prosedur berikut menjelaskan cara membuat peran IAM tugas dengan melampirkan kebijakan IAM yang Anda buat.

AWS Management Console

Untuk membuat peran IAM untuk tugas Anda ()AWS Management Console

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Dalam panel navigasi, pilih Roles (Peran), lalu Create role (Buat peran).
3. Untuk bagian Pilih entitas tepercaya, pilih AWS layanan.
4. Untuk kasus Penggunaan, menggunakan menu tarik-turun, pilih Layanan Kontainer Elastis dan kemudian kasus penggunaan Tugas Layanan Kontainer Elastis dan kemudian pilih Berikutnya.
5. Untuk Menambahkan izin, cari dan pilih kebijakan yang Anda buat, lalu pilih Berikutnya.
6. Pada Langkah 3: Beri nama, tinjau, dan buat, lakukan hal berikut:
   a. Untuk Nama peran, masukkan nama peran Anda. Untuk contoh ini, ketik `AmazonECSTaskS3BucketRole` untuk memberi nama peran.
   b. (Opsional) Untuk Deskripsi. tentukan deskripsi untuk peran IAM ini.
   c. Tinjau kebijakan entitas dan izin tepercaya untuk peran tersebut.

d. Untuk Tambahkan tag (Opsional), masukkan tag metadata apa pun yang ingin Anda kaitkan dengan peran IAM, lalu pilih Buat peran.

AWS CLI

Ganti semua *input pengguna* dengan nilai Anda sendiri.

1. Buat file bernama `ecs-tasks-trust-policy.json` yang berisi kebijakan kepercayaan yang akan digunakan untuk peran IAM tugas. File harus berisi yang berikut ini. Ganti pengenal Wilayah dan tentukan nomor AWS akun yang Anda gunakan saat meluncurkan tugas.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":[
               "ecs-tasks.amazonaws.com"
            ]
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "ArnLike":{
            "aws:SourceArn":"arn:aws:ecs:us-west-2:111122223333:*"
            },
            "StringEquals":{
               "aws:SourceAccount":"111122223333"
            }
         }
      }
   ]
}
```

2. Buat peran IAM bernama `ecsTaskRole` menggunakan kebijakan kepercayaan yang dibuat pada langkah sebelumnya.

```
aws iam create-role \
      --role-name ecsTaskRole \
      --assume-role-policy-document file://ecs-tasks-trust-policy.json
```

3. Ambil ARN dari kebijakan IAM yang Anda buat menggunakan perintah berikut. Ganti *taskRolePolicy*dengan nama kebijakan yang Anda buat.

```
aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`taskRolePolicy`].Arn'
```

4. Lampirkan kebijakan IAM yang Anda buat ke `ecsTaskRole` peran. Ganti `policy-arn` dengan ARN dari kebijakan yang Anda buat.

```
aws iam attach-role-policy \
      --role-name ecsTaskRole \
      --policy-arn arn:aws:iam:111122223333:aws:policy/taskRolePolicy
```

Izin IAM diperlukan untuk ECS Exec

Fitur ECS Exec memerlukan peran IAM tugas untuk memberikan kontainer izin yang diperlukan untuk komunikasi antara agen SSM terkelola (`execute-command`agen) dan layanan SSM. Anda harus menambahkan izin berikut ke peran IAM tugas dan menyertakan peran IAM tugas dalam definisi tugas Anda. Untuk informasi lebih lanjut, lihat Menambahkan dan Menghapus Kebijakan IAM dalam Panduan Pengguna IAM.

Gunakan kebijakan berikut untuk peran IAM tugas Anda untuk menambahkan izin SSM yang diperlukan.

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
       "Effect": "Allow",
       "Action": [
            "ssmmessages:CreateControlChannel",
            "ssmmessages:CreateDataChannel",
            "ssmmessages:OpenControlChannel",
            "ssmmessages:OpenDataChannel"
       ],
      "Resource": "*"
      }
   ]
}
```

Menentukan peran IAM untuk tugas Anda

Setelah membuat peran dan melampirkan kebijakan ke peran tersebut, Anda dapat menjalankan tugas yang berperan. Anda mempunyai beberapa opsi untuk melakukan ini:

- Tentukan peran IAM untuk tugas Anda dalam definisi tugas. Anda dapat membuat ketentuan tugas baru atau revisi baru ketentuan tugas yang ada dan tentukan peran yang Anda buat sebelumnya. Jika Anda menggunakan konsol untuk membuat definisi tugas, pilih peran IAM Anda di bidang Peran Tugas. Untuk informasi selengkapnya, lihat Membuat definisi tugas menggunakan konsol.

  Jika Anda menggunakan AWS CLI atau SDK, tentukan Nama Sumber Daya Amazon (ARN) peran tugas Anda menggunakan `taskRoleArn` parameter. Untuk informasi selengkapnya, lihat TaskDefinitiondi Referensi API Amazon ECS danParameter ketentuan tugas.

  > **Note**
  >
  > Opsi ini diperlukan jika Anda ingin menggunakan peran tugas IAM dalam layanan Amazon ECS.

- Tentukan penggantian peran tugas IAM saat menjalankan tugas. Anda dapat menentukan penggantian peran tugas IAM saat menjalankan tugas. Jika Anda menggunakan SDK AWS CLI atau, tentukan ARN peran tugas Anda menggunakan parameter `taskRoleArn` di `overrides` objek JSON. Untuk informasi selengkapnya, tentang `overrides` parameter, lihat RunTaskdan TaskOverridedi Referensi API Amazon ECS. Untuk informasi selengkapnya tentang penggantian menggunakan konsol, lihatJalankan aplikasi sebagai tugas Amazon ECS.

> **Note**
>
> Selain izin Amazon ECS standar yang diperlukan untuk menjalankan tugas dan layanan, pengguna juga memerlukan `iam:PassRole` izin untuk menggunakan peran IAM untuk tugas.

Peran IAM infrastruktur Amazon ECS

Peran IAM infrastruktur Amazon ECS memungkinkan Amazon ECS mengelola sumber daya infrastruktur di klaster Anda atas nama Anda, dan digunakan saat:

- Anda ingin melampirkan volume Amazon EBS ke tugas Amazon ECS jenis peluncuran Fargate atau EC2 Anda. Peran infrastruktur memungkinkan Amazon ECS mengelola volume Amazon EBS untuk tugas Anda.
- Anda ingin menggunakan Transport Layer Security (TLS) untuk mengenkripsi lalu lintas antara layanan Amazon ECS Service Connect Anda.

Ketika Amazon ECS mengasumsikan peran ini untuk mengambil tindakan atas nama Anda, acara akan terlihat di. AWS CloudTrail Jika Amazon ECS menggunakan peran untuk mengelola volume Amazon EBS yang dilampirkan ke tugas Anda, CloudTrail lognya `roleSessionName` akan menjadi. `ECSTaskVolumesForEBS` Jika peran digunakan untuk mengenkripsi lalu lintas antara layanan Amazon ECS Service Connect Anda, CloudTrail log `roleSessionName` akan menjadi. `ECSServiceConnectForTLS` Anda dapat menggunakan nama ini untuk mencari peristiwa di CloudTrail konsol dengan memfilter nama Pengguna.

Amazon ECS menyediakan kebijakan terkelola berikut bernama `AmazonECSInfrastructureRolePolicyForVolumes` dan `AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity` yang masing-masing berisi izin yang diperlukan untuk lampiran volume dan TLS. Untuk informasi selengkapnya, lihat AWS kebijakan terkelola untuk Amazon Elastic Container Service.

AmazonECSInfrastructureRolePolicyForVolumes

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "CreateEBSManagedVolume",
   "Effect": "Allow",
   "Action": "ec2:CreateVolume",
   "Resource": "arn:aws:ec2:*:*:volume/*",
   "Condition": {
    "ArnLike": {
     "aws:RequestTag/AmazonECSCreated": "arn:aws:ecs:*:*:task/*"
    },
    "StringEquals": {
     "aws:RequestTag/AmazonECSManaged": "true"
    }
   }
  },
  {
```

```
  "Sid": "TagOnCreateVolume",
  "Effect": "Allow",
  "Action": "ec2:CreateTags",
  "Resource": "arn:aws:ec2:*:*:volume/*",
  "Condition": {
   "ArnLike": {
    "aws:RequestTag/AmazonECSCreated": "arn:aws:ecs:*:*:task/*"
   },
   "StringEquals": {
    "ec2:CreateAction": "CreateVolume",
    "aws:RequestTag/AmazonECSManaged": "true"
   }
  }
 },
 {
  "Sid": "DescribeVolumesForLifecycle",
  "Effect": "Allow",
  "Action": [
   "ec2:DescribeVolumes",
   "ec2:DescribeAvailabilityZones"
  ],
  "Resource": "*"
 },
 {
  "Sid": "ManageEBSVolumeLifecycle",
  "Effect": "Allow",
  "Action": [
   "ec2:AttachVolume",
   "ec2:DetachVolume"
  ],
  "Resource": "arn:aws:ec2:*:*:volume/*",
  "Condition": {
   "StringEquals": {
    "aws:ResourceTag/AmazonECSManaged": "true"
   }
  }
 },
 {
  "Sid": "ManageVolumeAttachmentsForEC2",
  "Effect": "Allow",
  "Action": [
   "ec2:AttachVolume",
   "ec2:DetachVolume"
  ],
```

```
   "Resource": "arn:aws:ec2:*:*:instance/*"
  },
  {
   "Sid": "DeleteEBSManagedVolume",
   "Effect": "Allow",
   "Action": "ec2:DeleteVolume",
   "Resource": "arn:aws:ec2:*:*:volume/*",
   "Condition": {
    "ArnLike": {
     "aws:ResourceTag/AmazonECSCreated": "arn:aws:ecs:*:*:task/*"
    },
    "StringEquals": {
     "aws:ResourceTag/AmazonECSManaged": "true"
    }
   }
  }
 ]
}
```

AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateSecret",
            "Effect": "Allow",
            "Action": "secretsmanager:CreateSecret",
            "Resource": "arn:aws:secretsmanager:*:*:secret:ecs-sc!*",
            "Condition": {
                "ArnLike": {
                    "aws:RequestTag/AmazonECSCreated": [
                        "arn:aws:ecs:*:*:service/*/*",
                        "arn:aws:ecs:*:*:task-set/*/*"
                    ]
                },
                "StringEquals": {
                    "aws:RequestTag/AmazonECSManaged": "true",
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        },
        {
```

```
            "Sid": "TagOnCreateSecret",
            "Effect": "Allow",
            "Action": "secretsmanager:TagResource",
            "Resource": "arn:aws:secretsmanager:*:*:secret:ecs-sc!*",
            "Condition": {
                "ArnLike": {
                     "aws:RequestTag/AmazonECSCreated": [
                        "arn:aws:ecs:*:*:service/*/*",
                        "arn:aws:ecs:*:*:task-set/*/*"
                    ]
                },
                "StringEquals": {
                    "aws:RequestTag/AmazonECSManaged": "true",
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        },
        {
            "Sid": "RotateTLSCertificateSecret",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:UpdateSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "secretsmanager:DeleteSecret",
                "secretsmanager:RotateSecret",
                "secretsmanager:UpdateSecretVersionStage"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:ecs-sc!*",
            "Condition": {
                "StringEquals": {
                    "secretsmanager:ResourceTag/aws:secretsmanager:owningService":
"ecs-sc",
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        },
        {
            "Sid": "ManagePrivateCertificateAuthority",
            "Effect": "Allow",
            "Action": [
                "acm-pca:GetCertificate",
                "acm-pca:GetCertificateAuthorityCertificate",
```

```
                "acm-pca:DescribeCertificateAuthority"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/AmazonECSManaged": "true"
                }
            }
        },
        {
            "Sid":
 "ManagePrivateCertificateAuthorityForIssuingEndEntityCertificate",
            "Effect": "Allow",
            "Action": [
                "acm-pca:IssueCertificate"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/AmazonECSManaged": "true",
                    "acm-pca:TemplateArn":"arn:aws:acm-pca:::template/
EndEntityCertificate/V1"
                }
            }
        }
    ]
}
```

Menciptakan peran infrastruktur ECS () **ecsInfrastructureRole**

Ganti semua *input pengguna* dengan informasi Anda sendiri.

1. Buat file bernama `ecs-infrastructure-trust-policy.json` yang berisi kebijakan kepercayaan yang akan digunakan untuk peran IAM. File tersebut harus berisi hal berikut:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessToECSForInfrastructureManagement",
      "Effect": "Allow",
      "Principal": {
```

```
        "Service": "ecs.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. Gunakan AWS CLI perintah berikut untuk membuat peran IAM bernama `ecsInfrastructureRole` menggunakan kebijakan kepercayaan yang Anda buat pada langkah sebelumnya.

```
aws iam create-role \
      --role-name ecsInfrastructureRole \
      --assume-role-policy-document file://ecs-infrastructure-trust-policy.json
```

3. Bergantung pada kasus penggunaan Anda, lampirkan `AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity` kebijakan `AmazonECSInfrastructureRolePolicyForVolumes` atau AWS terkelola ke `ecsInfrastructureRole` peran tersebut.

```
aws iam attach-role-policy \
      --role-name ecsInfrastructureRole \
      --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSInfrastructureRolePolicyForVolumes
```

```
aws iam attach-role-policy \
      --role-name ecsInfrastructureRole \
      --policy-arn arn:aws::iam::aws:policy/service-role/
AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity
```

Anda juga dapat menggunakan alur kerja kebijakan kepercayaan kustom konsol IAM (https://console.aws.amazon.com/iam/) untuk membuat peran. Untuk informasi selengkapnya, lihat Membuat peran menggunakan kebijakan kepercayaan khusus (konsol) di Panduan Pengguna IAM.

Memberikan izin pengguna untuk meneruskan peran infrastruktur ECS ke Amazon ECS

Untuk menggunakan peran IAM infrastruktur ECS, Anda harus memberikan izin kepada pengguna untuk meneruskan peran tersebut ke Amazon ECS. Lampirkan `iam:PassRole` izin berikut ke pengguna Anda. Ganti *`ecsInfrastructureRole`*dengan nama peran infrastruktur yang Anda buat.

```
{
    "Version": "2012-10-17",
    "Statement": [

        {
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource": ["arn:aws:iam::*:role/ecsInfrastructureRole"],
            "Condition": {
                "StringEquals": {"iam:PassedToService": "ecs.amazonaws.com"}
            }
        }
    ]
}
```

Untuk informasi selengkapnya tentang *`iam:Passrole`* dan memperbarui izin untuk pengguna Anda, lihat Memberikan izin pengguna untuk meneruskan peran ke AWS layanan dan Mengubah izin untuk pengguna IAM di Panduan Pengguna.AWS Identity and Access Management

Konfigurasi tambahan untuk peran Windows IAM untuk tugas

> **Important**
>
> Untuk wadah Windows di Fargate yang menggunakan peran tugas, tidak diperlukan tindakan lebih lanjut. Untuk wadah Windows di EC2 yang menggunakan peran tugas, ikuti langkah-langkah ini.

Peran IAM untuk tugas-tugas dengan fitur Windows memerlukan konfigurasi tambahan pada EC2, tetapi banyak dari konfigurasi ini mirip dengan mengkonfigurasi peran IAM untuk tugas-tugas pada instance container Linux. Persyaratan berikut harus dipenuhi untuk mengonfigurasi peran IAM untuk tugas untuk wadah Windows.

- Ketika Anda meluncurkan instans kontainer, Anda harus mengatur opsi `-EnableTaskIAMRole` skrip data pengguna instans kontainer. `EnableTaskIAMRole`Menghidupkan fitur peran Task IAM untuk tugas. Sebagai contoh:

```
<powershell>
Import-Module ECSTools
Initialize-ECSAgent -Cluster 'windows' -EnableTaskIAMRole
</powershell>
```

- Anda harus bootstrap kontainer Anda dengan perintah jaringan yang disediakan di Peran IAM untuk skrip bootstrap wadah tugas.
- Anda harus membuat peran dan kebijakan IAM untuk tugas Anda. Untuk informasi selengkapnya, lihat Membuat peran dan kebijakan IAM untuk tugas Anda.
- Anda harus menentukan peran IAM yang Anda buat untuk tugas saat mendaftarkan definisi tugas, atau sebagai penggantian saat menjalankan tugas. Untuk informasi selengkapnya, lihat Menentukan peran IAM untuk tugas Anda.
- Peran IAM untuk penyedia kredensyal tugas menggunakan port 80 pada instance container. Oleh karena itu, jika Anda mengonfigurasi peran IAM untuk tugas pada instance container Anda, container Anda tidak dapat menggunakan port 80 untuk port host di pemetaan port apa pun. Untuk mengekspos kontainer Anda pada port 80, kami sarankan mengonfigurasi layanan untuk kontainer yang menggunakan penyeimbangan beban. Anda dapat menggunakan port 80 pada penyeimbang beban. Dengan demikian, lalu lintas dapat diarahkan ke port host lain pada instans kontainer Anda. Untuk informasi selengkapnya, lihat Mendistribusikan lalu lintas layanan Amazon ECS menggunakan load balancing.
- Jika instans Windows Anda dimulai ulang, Anda harus menghapus antarmuka proxy dan menginisialisasi agen penampung Amazon ECS lagi untuk mengembalikan proxy kredensi.

Peran IAM untuk skrip bootstrap wadah tugas

Sebelum kontainer dapat mengakses proxy kredensial pada instans kontainer untuk mendapatkan kredensial, kontainer harus di-bootstrap dengan perintah jaringan yang diperlukan. Skrip contoh kode berikut harus dijalankan pada kontainer Anda ketika dimulai.

> **Note**
>
> Anda tidak perlu menjalankan skrip ini saat Anda menggunakan mode jaringan `awsvpc` pada Windows.

Jika Anda menjalankan wadah Windows yang menyertakan Powershell, maka gunakan skrip berikut:

```
# Copyright Amazon.com Inc. or its affiliates. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License"). You may
# not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# or in the "license" file accompanying this file. This file is distributed
# on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
# express or implied. See the License for the specific language governing
# permissions and limitations under the License.

$gateway = (Get-NetRoute | Where { $_.DestinationPrefix -eq '0.0.0.0/0' } | Sort-Object
 RouteMetric | Select NextHop).NextHop
$ifIndex = (Get-NetAdapter -InterfaceDescription "Hyper-V Virtual Ethernet*" | Sort-
Object | Select ifIndex).ifIndex
New-NetRoute -DestinationPrefix 169.254.170.2/32 -InterfaceIndex $ifIndex -NextHop
 $gateway -PolicyStore ActiveStore # credentials API
New-NetRoute -DestinationPrefix 169.254.169.254/32 -InterfaceIndex $ifIndex -NextHop
 $gateway -PolicyStore ActiveStore # metadata API
```

Jika Anda menjalankan wadah Windows yang hanya memiliki shell Command, maka gunakan skrip berikut:

```
# Copyright Amazon.com Inc. or its affiliates. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License"). You may
# not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# or in the "license" file accompanying this file. This file is distributed
```

```
# on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
# express or implied. See the License for the specific language governing
# permissions and limitations under the License.

for /f "tokens=1" %i  in ('netsh interface ipv4 show interfaces ^| findstr /x /r
 ".*vEthernet.*"') do set interface=%i
for /f "tokens=3" %i  in ('netsh interface ipv4 show addresses %interface% ^| findstr /
x /r ".*Default.Gateway.*"') do set gateway=%i
netsh interface ipv4 add route prefix=169.254.170.2/32 interface="%interface%"
 nexthop="%gateway%" store=active # credentials API
netsh interface ipv4 add route prefix=169.254.169.254/32 interface="%interface%"
 nexthop="%gateway%" store=active # metadata API
```

Peran IAM instans wadah Amazon ECS

Instans penampung Amazon ECS, termasuk Amazon EC2 dan instans eksternal, menjalankan agen penampung Amazon ECS dan memerlukan peran IAM agar layanan mengetahui bahwa agen tersebut milik Anda. Sebelum meluncurkan instance kontainer dan mendaftarkannya ke klaster, Anda harus membuat peran IAM agar instance container Anda dapat digunakan. Peran dibuat di akun yang Anda gunakan untuk masuk ke konsol atau menjalankan AWS CLI perintah.

> **Important**
>
> Jika Anda mendaftarkan instance eksternal ke klaster, peran IAM yang Anda gunakan juga memerlukan izin Systems Manager. Untuk informasi selengkapnya, lihat Peran IAM ECS Anywhere.

Amazon ECS menyediakan kebijakan IAM `AmazonEC2ContainerServiceforEC2Role` terkelola yang berisi izin yang diperlukan untuk menggunakan set fitur Amazon ECS lengkap. Kebijakan terkelola ini dapat dilampirkan ke peran IAM dan dikaitkan dengan instance container Anda. Atau, Anda dapat menggunakan kebijakan terkelola sebagai panduan saat membuat kebijakan kustom untuk digunakan. Peran instance container memberikan izin yang diperlukan untuk agen penampung Amazon ECS dan daemon Docker untuk memanggil AWS API atas nama Anda. Untuk informasi selengkapnya tentang kebijakan terkelola, lihat AmazonEC2 EC2peran ContainerServicefor.

Amazon ECS mendukung peluncuran instans kontainer dengan peningkatan kepadatan ENI menggunakan jenis instans Amazon EC2 yang didukung. Saat Anda menggunakan fitur ini, kami sarankan Anda membuat 2 peran instance container. Aktifkan pengaturan `awsvpcTrunking` akun untuk satu peran dan gunakan peran itu untuk tugas yang memerlukan trunking ENI. Untuk informasi

tentang pengaturan `awsvpcTrunking` akun, lihatMengakses fitur Amazon ECS melalui pengaturan akun.

Membuat peran container instance (**`ecsInstanceRole`**)

> **Important**
>
> Jika Anda mendaftarkan instans eksternal untuk klaster Anda, lihat Peran IAM ECS Anywhere.

Anda dapat membuat peran secara manual dan melampirkan kebijakan IAM terkelola untuk instance container agar Amazon ECS dapat menambahkan izin untuk fitur dan penyempurnaan future saat diperkenalkan. Gunakan prosedur berikut untuk melampirkan kebijakan IAM terkelola jika diperlukan.

AWS Management Console

Untuk membuat peran **`ecsInstanceRole`** IAM untuk instance container Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Peran, lalu pilih Buat peran.
3. Pilih jenis peran AWS layanan, lalu di bawah Kasus penggunaan untuk AWS layanan lain, pilih Layanan Kontainer Elastis.
4. Pilih Peran EC2 untuk kasus penggunaan Layanan Kontainer Elastis, lalu pilih Berikutnya: Izin.
5. Di bagian Kebijakan izin, verifikasi kebijakan AmazonEC2 ContainerServicefor EC2role dipilih, lalu pilih Berikutnya.

   > **Important**
   >
   > Kebijakan terkelola ContainerServiceforAmazonEC2 EC2role harus dilampirkan ke peran IAM instance container, jika tidak, Anda akan menerima kesalahan saat menggunakan untuk membuat cluster. AWS Management Console

6. Untuk nama Peran, masukkan ecsInstanceRoledan opsional Anda dapat memasukkan deskripsi.
7. Untuk Tambahkan tag (opsional), masukkan tag kustom apa pun yang akan dikaitkan dengan kebijakan, lalu pilih Berikutnya: Tinjau.

8. Tinjau informasi peran Anda, lalu pilih Buat peran untuk menyelesaikan.

AWS CLI

Ganti semua *input pengguna* dengan nilai Anda sendiri.

1. Buat file bernama `instance-role-trust-policy.json` dengan konten berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": { "Service": "ec2.amazonaws.com"},
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. Gunakan perintah berikut untuk membuat peran IAM instance menggunakan dokumen kebijakan kepercayaan.

```
aws iam create-role \
    --role-name ecsInstanceRole \
    --assume-role-policy-document file://instance-role-trust-policy.json
```

3. Buat profil instance bernama `ecsInstanceRole-profile` menggunakan create-instance-profileperintah.

```
aws iam create-instance-profile --instance-profile-name ecsInstanceRole-profile
```

Contoh tanggapan

```
{
    "InstanceProfile": {
        "InstanceProfileId": "AIPAJTLBPJLEGREXAMPLE",
        "Roles": [],
        "CreateDate": "2022-04-12T23:53:34.093Z",
        "InstanceProfileName": "ecsInstanceRole-profile",
        "Path": "/",
```

```
        "Arn": "arn:aws:iam::123456789012:instance-profile/ecsInstanceRole-
profile"
    }
}
```

4. Tambahkan peran *ecsInstanceRole* pada profil instans *ecsInstanceRole-profile*.

```
aws iam add-role-to-instance-profile \
    --instance-profile-name ecsInstanceRole-profile \
    --role-name ecsInstanceRole
```

5. Lampirkan kebijakan `AmazonEC2ContainerServiceRoleForEC2Role` terkelola ke peran menggunakan perintah berikut.

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role \
    --role-name ecsInstanceRole
```

Menambahkan akses hanya-baca Amazon S3 ke peran instance penampung () `ecsInstanceRole`

Menyimpan informasi konfigurasi dalam bucket pribadi di Amazon S3 dan memberikan akses hanya-baca ke peran IAM instance container Anda adalah cara yang aman dan nyaman untuk mengizinkan konfigurasi instance container pada waktu peluncuran. Anda dapat menyimpan salinan `ecs.config` file Anda dalam bucket pribadi, menggunakan data pengguna Amazon EC2 untuk menginstal AWS CLI dan kemudian menyalin informasi konfigurasi Anda ke `/etc/ecs/ecs.config` saat instance diluncurkan.

Untuk informasi selengkapnya tentang membuat `ecs.config` file, menyimpannya di Amazon S3, dan meluncurkan instance dengan konfigurasi ini, lihat. Menyimpan konfigurasi instans kontainer di Amazon S3

Anda dapat menggunakan AWS CLI perintah berikut untuk mengizinkan akses hanya-baca Amazon S3 untuk peran instance container Anda. Ganti *ecsInstanceRole*dengan nama peran yang Anda buat.

```
aws iam attach-role-policy \
      --role-name ecsInstanceRole \
```

```
    --policy-arn arn:aws::iam::aws:policy/AmazonS3ReadOnlyAccess
```

Anda juga dapat menggunakan konsol IAM untuk menambahkan akses `AmazonS3ReadOnlyAccess` () hanya-baca Amazon S3 ke peran Anda. Untuk informasi selengkapnya, lihat Memodifikasi kebijakan izin peran (konsol) di AWS Identity and Access Management Panduan Pengguna.

Izin yang diperlukan untuk memantau instance kontainer

Sebelum instance container Anda dapat mengirim data CloudWatch log ke Log, Anda harus membuat kebijakan IAM agar instance container Anda dapat menggunakan API CloudWatch Log, lalu Anda harus melampirkan kebijakan tersebut. `ecsInstanceRole`

AWS Management Console

Cara menggunakan editor kebijakan JSON untuk membuat kebijakan

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi di sebelah kiri, pilih Kebijakan.

   Jika ini pertama kalinya Anda memilih Kebijakan, akan muncul halaman Selamat Datang di Kebijakan Terkelola. Pilih Memulai.
3. Di bagian atas halaman, pilih Buat kebijakan.
4. Di bagian Editor kebijakan, pilih opsi JSON.
5. Masukkan dokumen kebijakan JSON berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": ["arn:aws:logs:*:*:*"]
        }
```

```
    ]
}
```

6. Pilih Berikutnya.

> **Note**
>
> Anda dapat beralih antara opsi editor Visual dan JSON kapan saja. Namun, jika Anda melakukan perubahan atau memilih Berikutnya di editor Visual, IAM dapat merestrukturisasi kebijakan Anda untuk mengoptimalkannya bagi editor visual. Untuk informasi selengkapnya, lihat Restrukturisasi kebijakan dalam Panduan Pengguna IAM.

7. Pada halaman Tinjau dan buat, masukkan Nama kebijakan dan Deskripsi (opsional) untuk kebijakan yang Anda buat. Tinjau Izin yang ditentukan dalam kebijakan ini untuk melihat izin yang diberikan oleh kebijakan Anda.
8. Pilih Buat kebijakan untuk menyimpan kebijakan baru Anda.

Setelah Anda membuat kebijakan, lampirkan kebijakan ke peran instance container. Untuk informasi tentang cara melampirkan kebijakan ke peran, lihat Memodifikasi kebijakan izin peran (konsol) di AWS Identity and Access Management Panduan Pengguna.

AWS CLI

1. Buat file bernama `instance-cw-logs.json` dengan konten berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": ["arn:aws:logs:*:*:*"]
        }
    ]
```

```
}
```

2. Gunakan perintah berikut untuk membuat kebijakan IAM menggunakan file dokumen kebijakan JSON.

```
aws iam create-policy \
      --policy-name cwlogspolicy \
      --policy-document file://instance-cw-logs.json
```

3. Ambil ARN dari kebijakan IAM yang Anda buat menggunakan perintah berikut. Ganti *cwlogspolicy* dengan nama kebijakan yang Anda buat.

```
aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`cwlogsppolicy`].Arn'
```

4. Gunakan perintah berikut untuk melampirkan kebijakan ke peran IAM instance container menggunakan ARN kebijakan.

```
aws iam attach-role-policy \
      --role-name ecsInstanceRole \
      --policy-arn arn:aws:iam:111122223333:aws:policy/cwlogspolicy
```

Peran IAM ECS Anywhere

Saat Anda mendaftarkan server lokal atau mesin virtual (VM) ke klaster, server atau VM memerlukan peran IAM untuk berkomunikasi dengan API. AWS Anda hanya perlu membuat peran IAM ini sekali untuk setiap AWS akun. Namun, peran IAM ini harus dikaitkan dengan setiap server atau VM yang Anda daftarkan ke cluster. Peran ini adalah `ECSAnywhereRole`. Anda dapat membuat peran ini secara manual. Atau, Amazon ECS dapat membuat peran atas nama Anda saat Anda mendaftarkan instans eksternal di AWS Management Console. Anda dapat menggunakan pencarian konsol IAM untuk mencari `ecsAnywhereRole` dan melihat apakah akun Anda sudah memiliki peran tersebut. Untuk informasi selengkapnya, lihat pencarian konsol IAM di panduan pengguna IAM.

AWS menyediakan dua kebijakan IAM terkelola yang dapat digunakan saat membuat peran IAM ECS Anywhere, dan kebijakan. `AmazonSSMManagedInstanceCore` `AmazonEC2ContainerServiceforEC2Role` Kebijakan `AmazonEC2ContainerServiceforEC2Role` meliputi izin yang menyediakan lebih banyak akses daripada yang Anda butuhkan. Oleh karena itu, dengan bergantung pada kasus penggunaan khusus Anda, kami sarankan supaya Anda membuat kebijakan kustom yang hanya menambahkan izin dari

kebijakan yang Anda perlukan di dalamnya. Untuk informasi selengkapnya, lihat Peran IAM instans wadah Amazon ECS.

Membuat peran ECS Anywhere (**ecsAnywhereRole**)

Ganti semua *input pengguna* dengan informasi Anda sendiri.

1. Buat file lokal bernama `ssm-trust-policy.json` dengan kebijakan kepercayaan berikut.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"Service": [
      "ssm.amazonaws.com"
    ]},
    "Action": "sts:AssumeRole"
  }
}
```

2. Buat peran dan lampirkan kebijakan kepercayaan dengan menggunakan AWS CLI perintah berikut.

```
aws iam create-role --role-name ecsAnywhereRole --assume-role-policy-document
 file://ssm-trust-policy.json
```

3. Lampirkan kebijakan AWS terkelola dengan menggunakan perintah berikut.

```
aws iam attach-role-policy --role-name ecsAnywhereRole --policy-arn
 arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
aws iam attach-role-policy --role-name ecsAnywhereRole --policy-arn
 arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceforEC2Role
```

Anda juga dapat menggunakan alur kerja kebijakan kepercayaan kustom konsol IAM (https://console.aws.amazon.com/iam/) untuk membuat peran. Untuk informasi selengkapnya, lihat Membuat peran menggunakan kebijakan kepercayaan khusus (konsol) di Panduan Pengguna IAM.

Izin IAM bersyarat

Peran IAM eksekusi tugas memberikan izin agen penampung Amazon ECS untuk melakukan panggilan AWS API atas nama Anda. Ketika peran IAM eksekusi tugas digunakan, itu harus

ditentukan dalam definisi tugas Anda. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

Peran pelaksanaan tugas diperlukan jika salah satu syarat berikut berlaku:

- Anda mengirim log kontainer ke CloudWatch Log menggunakan driver `awslogs` log.
- Definisi tugas Anda menentukan gambar kontainer yang di-host di repositori pribadi Amazon ECR. Namun, jika peran `ECSAnywhereRole` IAM yang terkait dengan instans eksternal Anda juga menyertakan izin yang diperlukan untuk menarik gambar dari Amazon ECR maka peran eksekusi tugas Anda tidak perlu menyertakannya.

Peran Amazon ECS CodeDeploy IAM

Sebelum Anda dapat menggunakan jenis penyebaran CodeDeploy biru/hijau dengan Amazon ECS, CodeDeploy layanan memerlukan izin untuk memperbarui layanan Amazon ECS Anda atas nama Anda. Izin ini disediakan oleh peran CodeDeploy IAM ()`ecsCodeDeployRole`.

> **Note**
>
> Pengguna juga memerlukan izin untuk digunakan CodeDeploy; izin ini dijelaskan dalam. Penerapan biru/hijau memerlukan izin IAM

Ada dua kebijakan terkelola yang disediakan. `AWSCodeDeployRoleForECS`Kebijakan, yang ditunjukkan di bawah ini, memberikan CodeDeploy izin untuk memperbarui sumber daya apa pun menggunakan tindakan terkait. `AWSCodeDeployRoleForECSLimited`Kebijakan, yang ditunjukkan di bawah ini, memberikan izin yang CodeDeploy lebih terbatas.

AWSCodeDeployRoleForECS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ecs:DescribeServices",
                "ecs:CreateTaskSet",
                "ecs:UpdateServicePrimaryTaskSet",
                "ecs:DeleteTaskSet",
```

```
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:ModifyListener",
                "elasticloadbalancing:DescribeRules",
                "elasticloadbalancing:ModifyRule",
                "lambda:InvokeFunction",
                "cloudwatch:DescribeAlarms",
                "sns:Publish",
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": ["iam:PassRole"],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": ["ecs-tasks.amazonaws.com"]
                }
            }
        }
    ]
}
```

AWSCodeDeployRoleForECSLimited

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ecs:DescribeServices",
                "ecs:CreateTaskSet",
                "ecs:UpdateServicePrimaryTaskSet",
                "ecs:DeleteTaskSet",
                "cloudwatch:DescribeAlarms"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
```

```
        {
            "Action": ["sns:Publish"],
            "Resource": "arn:aws:sns:*:*:CodeDeployTopic_*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:ModifyListener",
                "elasticloadbalancing:DescribeRules",
                "elasticloadbalancing:ModifyRule"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": ["lambda:InvokeFunction"],
            "Resource": "arn:aws:lambda:*:*:function:CodeDeployHook_*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {"s3:ExistingObjectTag/UseWithCodeDeploy": "true"}
            },
            "Effect": "Allow"
        },
        {
            "Action": ["iam:PassRole"],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam::*:role/ecsTaskExecutionRole",
                "arn:aws:iam::*:role/ECSTaskExecution*"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": ["ecs-tasks.amazonaws.com"]
                }
            }
```

```
        }
    ]
}
```

Menciptakan CodeDeploy **AWSCodeDeployRoleForECS** peran

Anda dapat menggunakan prosedur berikut untuk membuat CodeDeploy peran untuk Amazon ECS

AWS Management Console

Untuk membuat peran IAM untuk CodeDeploy

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Dalam panel navigasi, pilih Roles (Peran), lalu Create role (Buat peran).
3. Untuk bagian Pilih tipe entitas terpercaya, pilih Layanan AWS .
4. Untuk Pilih layanan yang akan menggunakan peran ini, pilih CodeDeploy.
5. Untuk Pilih kasus penggunaan Anda, pilih CodeDeploy - ECS, Berikutnya.
6. Di bagian Tambahkan izin, lakukan hal berikut
   a. Pastikan AWSCodeDeployRoleForECSkebijakan tersebut dipilih.
   b. Di bawah Setel batas izin - opsional, pilih Buat peran tanpa batas izin.
   c. Pilih Berikutnya.
7. Di bawah Nama, tinjau, dan buat, lakukan hal berikut:
   a. Untuk nama Peran, masukkan`ecsCodeDeployRole`, dan masukkan deskripsi opsional.
   b. Untuk Tambahkan tag (opsional), masukkan tag kustom apa pun untuk dikaitkan dengan kebijakan.
8. Pilih Buat peran.

AWS CLI

Ganti semua *`input pengguna`* dengan informasi Anda sendiri.

1. Buat file bernama `codedeploy-trust-policy.json` yang berisi kebijakan kepercayaan yang akan digunakan untuk peran CodeDeploy IAM.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": ["codedeploy.amazonaws.com"]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

2. Buat peran IAM bernama `ecsCodedeployRole` menggunakan kebijakan kepercayaan yang dibuat pada langkah sebelumnya.

```
aws iam create-role \
      --role-name ecsCodedeployRole \
      --assume-role-policy-document file://codedeploy-trust-policy.json
```

3. Lampirkan kebijakan `AWSCodeDeployRoleForECS` atau `AWSCodeDeployRoleForECSLimited` terkelola ke `ecsTaskRole` peran tersebut.

```
aws iam attach-role-policy \
      --role-name ecsCodedeployRole \
      --policy-arn arn:aws::iam::aws:policy/AWSCodeDeployRoleForECS
```

```
aws iam attach-role-policy \
      --role-name ecsCodedeployRole \
      --policy-arn arn:aws::iam::aws:policy/AWSCodeDeployRoleForECSLimited
```

Menambahkan izin untuk penerapan biru/hijau

Jika tugas di layanan Amazon ECS Anda yang menggunakan jenis penerapan biru/hijau memerlukan penggunaan peran eksekusi tugas atau penggantian peran tugas, Anda harus menambahkan `iam:PassRole` izin untuk setiap peran eksekusi tugas atau penggantian peran tugas ke peran IAM sebagai kebijakan. CodeDeploy Lihat informasi yang lebih lengkap di Peran IAM eksekusi tugas Amazon ECS dan Tugas peran IAM.

Gunakan prosedur berikut untuk membuat kebijakan

AWS Management Console

Cara menggunakan editor kebijakan JSON untuk membuat kebijakan

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi di sebelah kiri, pilih Kebijakan.

   Jika ini pertama kalinya Anda memilih Kebijakan, akan muncul halaman Selamat Datang di Kebijakan Terkelola. Pilih Memulai.
3. Di bagian atas halaman, pilih Buat kebijakan.
4. Di bagian Editor kebijakan, pilih opsi JSON.
5. Masukkan dokumen kebijakan JSON berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["arn:aws:iam::<aws_account_id>:role/
<ecsTaskExecutionRole_or_TaskRole_name>"]
        }
    ]
}
```

6. Pilih Berikutnya.

   > **Note**
   >
   > Anda dapat beralih antara opsi editor Visual dan JSON kapan saja. Namun, jika Anda melakukan perubahan atau memilih Berikutnya di editor Visual, IAM dapat merestrukturisasi kebijakan Anda untuk mengoptimalkannya bagi editor visual. Untuk informasi selengkapnya, lihat Restrukturisasi kebijakan dalam Panduan Pengguna IAM.
7. Pada halaman Tinjau dan buat, masukkan Nama kebijakan dan Deskripsi (opsional) untuk kebijakan yang Anda buat. Tinjau Izin yang ditentukan dalam kebijakan ini untuk melihat izin yang diberikan oleh kebijakan Anda.

8. Pilih Buat kebijakan untuk menyimpan kebijakan baru Anda.

Setelah Anda membuat kebijakan, lampirkan kebijakan ke CodeDeploy peran tersebut. Untuk informasi tentang cara melampirkan kebijakan ke peran, lihat Memodifikasi kebijakan izin peran (konsol) di AWS Identity and Access Management Panduan Pengguna.

AWS CLI

Ganti semua *input pengguna* dengan informasi Anda sendiri.

1. Buat file bernama `blue-green-iam-passrole.json` dengan konten berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["arn:aws:iam::<aws_account_id>:role/
<ecsTaskExecutionRole_or_TaskRole_name>"]
        }
    ]
}
```

2. Gunakan perintah berikut untuk membuat kebijakan IAM menggunakan file dokumen kebijakan JSON.

```
aws iam create-policy \
      --policy-name cdTaskExecutionPolicy \
      --policy-document file://blue-green-iam-passrole.json
```

3. Ambil ARN dari kebijakan IAM yang Anda buat menggunakan perintah berikut.

```
aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`cdTaskExecutionPolicy`].Arn'
```

4. Gunakan perintah berikut untuk melampirkan kebijakan ke peran CodeDeploy IAM.

```
aws iam attach-role-policy \
      --role-name ecsCodedeployRole \
      --policy-arn arn:aws:iam:111122223333:aws:policy/cdTaskExecutionPolicy
```

Peran Amazon ECS EventBridge IAM

Sebelum Anda dapat menggunakan tugas terjadwal Amazon ECS dengan EventBridge aturan dan target, EventBridge layanan memerlukan izin untuk menjalankan tugas Amazon ECS atas nama Anda. Izin ini disediakan oleh peran EventBridge IAM ()`ecsEventsRole`.

Kebijakan `AmazonEC2ContainerServiceEventsRole` ditunjukkan di bawah ini.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:RunTask"],
            "Resource": ["*"]
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["*"],
            "Condition": {
                "StringLike": {"iam:PassedToService": "ecs-tasks.amazonaws.com"}
            }
        },
        {
            "Effect": "Allow",
            "Action": "ecs:TagResource",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:CreateAction": ["RunTask"]
                }
            }
        }
    ]
}
```

Jika tugas terjadwal Anda memerlukan penggunaan peran eksekusi tugas, peran tugas, atau penggantian peran tugas, Anda harus menambahkan `iam:PassRole` izin untuk setiap peran eksekusi tugas, peran tugas, atau penggantian peran tugas ke peran IAM. EventBridge Untuk informasi selengkapnya tentang peran eksekusi tugas, lihat Peran IAM eksekusi tugas Amazon ECS.

> **Note**
>
> Tentukan ARN penuh dari peran eksekusi tugas Anda atau penimpaan peran tugas.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["arn:aws:iam::<aws_account_id>:role/
<ecsTaskExecutionRole_or_TaskRole_name>"]
        }
    ]
}
```

Anda dapat memilih untuk membiarkan AWS Management Console membuat EventBridge peran untuk Anda ketika Anda mengkonfigurasi tugas terjadwal. Untuk informasi selengkapnya, lihat Jalankan tugas Amazon ECS pada jadwal menggunakan Amazon Scheduler EventBridge .

Membuat peran Amazon ECS EventBridge (**`ecsEventsRole`**)

Ganti semua *`input pengguna`* dengan informasi Anda sendiri.

1. Buat file bernama `eventbridge-trust-policy.json` yang berisi kebijakan kepercayaan yang akan digunakan untuk peran IAM. File tersebut harus berisi hal berikut:

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "",
         "Effect": "Allow",
         "Principal": {
           "Service": "events.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }
     ]
   }
   ```

2. Gunakan perintah berikut untuk membuat peran IAM bernama `ecsEventsRole` menggunakan kebijakan kepercayaan yang Anda buat pada langkah sebelumnya.

```
aws iam create-role \
      --role-name ecsEventsRole \
      --assume-role-policy-document file://eventbridge-policy.json
```

3. Lampirkan yang AWS dikelola `AmazonEC2ContainerServiceEventsRole` ke `ecsEventsRole` peran menggunakan perintah berikut.

```
aws iam attach-role-policy \
      -\-role-name ecsEventsRole \
      -\-policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceEventsRole
```

Anda juga dapat menggunakan alur kerja kebijakan kepercayaan kustom konsol IAM (https://console.aws.amazon.com/iam/) untuk membuat peran. Untuk informasi selengkapnya, lihat Membuat peran menggunakan kebijakan kepercayaan khusus (konsol) di Panduan Pengguna IAM.

Melampirkan kebijakan pada peran **ecsEventsRole**

Anda dapat menggunakan prosedur berikut untuk menambahkan izin untuk peran eksekusi tugas ke peran EventBridge IAM.

AWS Management Console

Cara menggunakan editor kebijakan JSON untuk membuat kebijakan

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi di sebelah kiri, pilih Kebijakan.

   Jika ini pertama kalinya Anda memilih Kebijakan, akan muncul halaman Selamat Datang di Kebijakan Terkelola. Pilih Memulai.
3. Di bagian atas halaman, pilih Buat kebijakan.
4. Di bagian Editor kebijakan, pilih opsi JSON.
5. Masukkan dokumen kebijakan JSON berikut:

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["arn:aws:iam::<aws_account_id>:role/
<ecsTaskExecutionRole_or_TaskRole_name>"]
        }
    ]
}
```

6. Pilih Berikutnya.

> **Note**
>
> Anda dapat beralih antara opsi editor Visual dan JSON kapan saja. Namun, jika Anda melakukan perubahan atau memilih Berikutnya di editor Visual, IAM dapat merestrukturisasi kebijakan Anda untuk mengoptimalkannya bagi editor visual. Untuk informasi selengkapnya, lihat Restrukturisasi kebijakan dalam Panduan Pengguna IAM.

7. Pada halaman Tinjau dan buat, masukkan Nama kebijakan dan Deskripsi (opsional) untuk kebijakan yang Anda buat. Tinjau Izin yang ditentukan dalam kebijakan ini untuk melihat izin yang diberikan oleh kebijakan Anda.
8. Pilih Buat kebijakan untuk menyimpan kebijakan baru Anda.

Setelah Anda membuat kebijakan, lampirkan kebijakan ke EventBridge peran tersebut. Untuk informasi tentang cara melampirkan kebijakan ke peran, lihat Memodifikasi kebijakan izin peran (konsol) di AWS Identity and Access Management Panduan Pengguna.

AWS CLI

Ganti semua *input pengguna* dengan informasi Anda sendiri.

1. Buat file bernama `ev-iam-passrole.json` dengan konten berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": "iam:PassRole",
            "Resource": ["arn:aws:iam::<aws_account_id>:role/
<ecsTaskExecutionRole_or_TaskRole_name>"]
        }
    ]
}
```

2. Gunakan AWS CLI perintah berikut untuk membuat kebijakan IAM menggunakan file dokumen kebijakan JSON.

```
aws iam create-policy \
      --policy-name eventsTaskExecutionPolicy \
      --policy-document file://ev-iam-passrole.json
```

3. Ambil ARN dari kebijakan IAM yang Anda buat menggunakan perintah berikut.

```
aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`eventsTaskExecutionPolicy`].Arn'
```

4. Gunakan perintah berikut untuk melampirkan kebijakan ke peran EventBridge IAM dengan menggunakan kebijakan ARN.

```
aws iam attach-role-policy \
      --role-name ecsEventsRole \
      --policy-arn arn:aws:iam:111122223333:aws:policy/eventsTaskExecutionPolicy
```

Izin IAM diperlukan untuk penskalaan otomatis servis

Service Auto Scaling dimungkinkan oleh kombinasi Amazon ECS, CloudWatch, dan Application Auto Scaling API. Layanan dibuat dan diperbarui dengan Amazon ECS, alarm dibuat dengan CloudWatch, dan kebijakan penskalaan dibuat dengan Application Auto Scaling.

Selain izin IAM standar untuk membuat dan memperbarui layanan, izin berikut diperlukan untuk berinteraksi dengan pengaturan Auto Scaling Layanan seperti yang ditunjukkan dalam contoh kebijakan berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:*",
                "ecs:DescribeServices",
                "ecs:UpdateService",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DeleteAlarms",
                "cloudwatch:DescribeAlarmHistory",
                "cloudwatch:DescribeAlarmsForMetric",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "cloudwatch:DisableAlarmActions",
                "cloudwatch:EnableAlarmActions",
                "iam:CreateServiceLinkedRole",
                "sns:CreateTopic",
                "sns:Subscribe",
                "sns:Get*",
                "sns:List*"
            ],
            "Resource": ["*"]
        }
    ]
}
```

Contoh kebijakan Perbarui contoh layanan IAM Buat contoh layanan dan IAM menunjukkan izin yang diperlukan untuk menggunakan Auto Scaling Service di. AWS Management Console

Layanan Application Auto Scaling juga memerlukan izin untuk menjelaskan layanan dan CloudWatch alarm Amazon ECS Anda, dan izin untuk mengubah jumlah layanan yang diinginkan atas nama Anda. `sns:`Izin adalah untuk notifikasi yang CloudWatch dikirim ke topik Amazon SNS ketika ambang batas telah terlampaui. Jika Anda menggunakan penskalaan otomatis untuk layanan Amazon ECS Anda, itu akan membuat peran terkait layanan bernama. `AWSServiceRoleForApplicationAutoScaling_ECSService` Peran terkait layanan ini memberikan izin Application Auto Scaling untuk menjelaskan alarm kebijakan Anda, memantau jumlah tugas layanan yang sedang berjalan, dan mengubah jumlah layanan yang diinginkan. Peran Amazon ECS yang dikelola asli untuk Application Auto Scaling `ecsAutoscaleRole` adalah, tetapi tidak lagi diperlukan. Peran tertaut layanan adalah peran default untuk Penskalaan Otomatis Aplikasi. Untuk informasi selengkapnya, lihat Peran yang ditautkan dengan layanan untuk Application Auto Scaling dalam Panduan Pengguna Application Auto Scaling.

Jika Anda membuat peran instans penampung Amazon ECS sebelum CloudWatch metrik tersedia untuk Amazon ECS, Anda mungkin perlu menambahkan izin. `ecs:StartTelemetrySession` Untuk informasi selengkapnya, lihat Pertimbangan.

Berikan izin untuk menandai sumber daya pada pembuatan

Tindakan penandaan Amazon ECS API berikut memungkinkan Anda menentukan tag saat membuat sumber daya. Jika tag ditentukan dalam tindakan pembuatan sumber daya, AWS lakukan otorisasi tambahan untuk memverifikasi bahwa izin yang benar ditetapkan untuk membuat tag.

- `CreateCapacityProvider`
- `CreateCluster`
- `CreateService`
- `CreateTaskSet`
- `RegisterContainerInstance`
- `RegisterTaskDefinition`
- `RunTask`
- `StartTask`

Anda dapat menggunakan tanda sumber daya untuk menerapkan pengendalian berbasis atribut (ABAC). Untuk informasi selengkapnya, lihat the section called “Kontrol akses ke sumber daya Amazon ECS menggunakan tag sumber daya” dan Penandaan sumber daya Amazon ECS.

Untuk mengizinkan penandaan pada pembuatan, buat atau ubah kebijakan untuk menyertakan izin untuk menggunakan tindakan yang membuat sumber daya, seperti `ecs:CreateCluster` atau `ecs:RunTask` dan tindakan. `ecs:TagResource`

Contoh berikut menunjukkan kebijakan yang memungkinkan pengguna membuat cluster dan menambahkan tag selama pembuatan klaster. Para pengguna tidak diizinkan untuk memberi tanda pada sumber daya yang sudah ada (mereka tidak dapat memerintahkan tindakan `ecs:TagResource` secara langsung).

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
         "ecs:CreateCluster"
```

```
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
         "ecs:TagResource"
      ],
      "Resource": "*",
      "Condition": {
         "StringEquals": {
              "StringEquals": {
                  "ecs:CreateAction": [
                      "CreateCluster",
                      "CreateCapacityProvider",
                      "CreateService",
                      "CreateTaskSet",
                      "RegisterContainerInstance",
                      "RegisterTaskDefinition",
                      "RunTask",
                      "StartTask"
                  ]
          }
       }
    }
  ]
}
```

Tindakan `ecs:TagResource` akan dievaluasi hanya jika tanda diterapkan selama tindakan pembuatan sumber daya. Oleh karena itu, seorang pengguna yang memiliki izin untuk membuat sumber daya (dengan asumsi tidak ada syarat untuk pemberian tanda) tidak memerlukan izin untuk menggunakan tindakan `ecs:TagResource` jika tidak ada tanda yang ditentukan dalam permintaan. Akan tetapi, jika pengguna tersebut mencoba untuk membuat sumber daya dengan tanda, maka permintaan akan gagal jika pengguna tidak memiliki izin untuk menggunakan tindakan `ecs:TagResource`.

Mengendalikan akses ke tanda-tanda tertentu

Anda dapat menggunakan syarat tambahan dalam elemen `Condition` dari kebijakan IAM Anda untuk mengontrol kunci tanda dan nilai tanda yang dapat diterapkan ke sumber daya.

Kunci syarat berikut dapat digunakan dengan contoh-contoh pada bagian sebelumnya:

- `aws:RequestTag`: Untuk mengindikasikan bahwa kunci tanda tertentu atau kunci dan nilai tanda tertentu harus ada di permintaan. Tanda-tanda yang lain juga dapat ditentukan dalam permintaan.
  - Gunakan bersama dengan operator syarat `StringEquals` untuk memberlakukan kombinasi kunci dan nilai tanda tertentu, misalnya, untuk memberlakukan tanda `cost-center=cc123`:

    ```
    "StringEquals": { "aws:RequestTag/cost-center": "cc123" }
    ```

  - Gunakan bersama dengan operator syarat `StringLike` untuk memberlakukan kunci tanda tertentu dalam permintaan, misalnya, untuk memberlakukan kunci tanda `purpose`:

    ```
    "StringLike": { "aws:RequestTag/purpose": "*" }
    ```

- `aws:TagKeys`: Untuk memberlakukan kunci tanda yang digunakan dalam permintaan.
  - Gunakan bersama dengan pemodifikasi `ForAllValues` untuk menerapkan kunci tanda tertentu jika disediakan dalam permintaan (jika tanda ditentukan dalam permintaan, hanya kunci tanda tertentu saja yang diperbolehkan; tidak ada tanda lain yang diperbolehkan). Sebagai contoh, kunci tanda `environment` atau `cost-center` diperbolehkan:

    ```
    "ForAllValues:StringEquals": { "aws:TagKeys": ["environment","cost-center"] }
    ```

  - Gunakan pemodifikasi `ForAnyValue` untuk memaksakan keberadaan setidaknya salah satu kunci tanda tertentu dalam permintaan. Sebagai contoh, setidaknya salah satu kunci tanda `environment` atau `webserver` harus ada dalam permintaan:

    ```
    "ForAnyValue:StringEquals": { "aws:TagKeys": ["environment","webserver"] }
    ```

Kunci kondisi ini dapat diterapkan ke tindakan pembuatan sumber daya yang mendukung penandaan, serta tindakan. `ecs:TagResource` Untuk mengetahui apakah tindakan Amazon ECS API mendukung penandaan, lihat Tindakan, sumber daya, dan kunci kondisi untuk Amazon ECS.

Untuk memaksa para pengguna menentukan tanda pada saat mereka membuat sumber daya, Anda harus menggunakan kunci syarat `aws:RequestTag` atau kunci syarat `aws:TagKeys` dengan pemodifikasi `ForAnyValue` pada tindakan yang digunakan untuk membuat sumber daya. Tindakan `ecs:TagResource` tidak akan dievaluasi jika pengguna tidak menentukan tanda untuk tindakan yang digunakan untuk pembuatan sumber daya.

Untuk syarat, kunci syarat tidak bersifat peka terhadap huruf besar dan kecil dan nilai syarat bersifat peka huruf besar dan kecil. Oleh karena itu, untuk memaksakan sifat peka terhadap huruf besar atau

kecil dari kunci tanda, gunakan kunci syarat `aws:TagKeys`, di mana kunci tanda ditetapkan sebagai nilai dalam syarat tersebut.

Untuk informasi selengkapnya, lihat Membuat Syarat yang Menguji Beberapa Nilai Kunci dalam Panduan Pengguna IAM.

Kontrol akses ke sumber daya Amazon ECS menggunakan tag sumber daya

Saat membuat kebijakan IAM yang memberikan izin kepada pengguna untuk menggunakan sumber daya Amazon ECS, Anda dapat menyertakan informasi tag dalam `Condition` elemen kebijakan untuk mengontrol akses berdasarkan tag. Hal ini dikenal sebagai kendali akses berbasis atribut (ABAC). ABAC memberikan Anda kendali yang lebih baik atas sumber daya mengenai sumber daya mana yang dapat diubah, digunakan, atau dihapus oleh seorang pengguna. Untuk informasi lebih lanjut, lihat Apa fungsi ABAC untuk AWS?

Misalnya, Anda dapat membuat kebijakan yang memungkinkan pengguna menghapus klaster, tetapi menolak tindakan jika klaster memiliki tag`environment=production`. Untuk melakukan hal ini, Anda bisa menggunakan kunci syarat `aws:ResourceTag` untuk mengizinkan atau menolak akses ke sumber daya berdasarkan tanda yang dilampirkan pada sumber daya.

```
"StringEquals": { "aws:ResourceTag/environment": "production" }
```

Untuk mempelajari apakah tindakan Amazon ECS API mendukung pengendalian akses menggunakan kunci `aws:ResourceTag` kondisi, lihat Tindakan, sumber daya, dan kunci kondisi untuk Amazon ECS. Perhatikan bahwa tindakan `Describe` tidak mendukung izin tingkat sumber daya, sehingga Anda harus menentukannya dalam pernyataan terpisah yang tidak disertai syarat.

Untuk contoh kebijakan IAM, lihat Contoh kebijakan .

Jika Anda mengizinkan atau menolak akses para pengguna ke sumber daya berdasarkan tanda, maka Anda harus mempertimbangkan untuk menolak secara eksplisit memberikan kemampuan kepada pengguna untuk menambahkan atau menghapus tanda tersebut dari sumber daya yang sama. Jika tidak, pengguna dapat mengakali pembatasan Anda dan mendapatkan akses atas sumber daya dengan melakukan modifikasi pada tanda dari sumber daya tersebut.

Contoh kebijakan

Anda dapat menggunakan kebijakan IAM untuk memberikan izin kepada pengguna untuk melihat dan bekerja dengan sumber daya tertentu di konsol Amazon ECS. Anda dapat menggunakan contoh

kebijakan di bagian sebelumnya; namun, kebijakan tersebut dirancang untuk permintaan yang dibuat dengan AWS CLI atau AWS SDK.

Contoh: Izinkan pengguna untuk menghapus cluster berdasarkan tag

Kebijakan berikut memungkinkan pengguna untuk menghapus cluster ketika tag memiliki pasangan kunci/nilai “Tujuan/Pengujian”.

```
{
  "Version": "2012-10-17",
  "Statement": [
       {
      "Action": [
        "ecs:DeleteCluster"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:ecs:region:account-id:cluster/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Purpose": "Testing"
        }
      }
    }
  ]
}
```

Memecahkan masalah identitas dan akses Amazon Elastic Container Service

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan Amazon ECS dan IAM.

Topik

- Saya tidak berwenang untuk melakukan tindakan di Amazon ECS
- Saya tidak berwenang untuk melakukan iam: PassRole
- Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya Amazon ECS saya

Saya tidak berwenang untuk melakukan tindakan di Amazon ECS

Jika Anda menerima pesan kesalahan bahwa Anda tidak memiliki otorisasi untuk melakukan tindakan, kebijakan Anda harus diperbarui agar Anda dapat melakukan tindakan tersebut.

Contoh kesalahan berikut terjadi ketika pengguna IAM `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya *`my-example-widget`* rekaan, tetapi tidak memiliki izin `ecs:`*`GetWidget`* rekaan.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 ecs:GetWidget on resource: my-example-widget
```

Dalam hal ini, kebijakan untuk pengguna `mateojackson` harus diperbarui untuk mengizinkan akses ke sumber daya *`my-example-widget`* dengan menggunakan tindakan `ecs:`*`GetWidget`*.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan bahwa Anda tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke Amazon ECS.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol untuk melakukan tindakan di Amazon ECS. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya Amazon ECS saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau pengguna di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi pengguna akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa hal berikut:

- Untuk mengetahui apakah Amazon ECS mendukung fitur-fitur ini, lihatBagaimana Amazon Elastic Container Service bekerja dengan IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang Anda miliki di Panduan Pengguna IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat Memberikan akses kepada pengguna eksternal yang sah (federasi identitas) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara penggunaan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Perbedaan antara peran IAM dan kebijakan berbasis sumber daya di Panduan Pengguna IAM.

Logging dan Monitoring di Amazon Elastic Container Service

Pemantauan merupakan bagian penting dalam menjaga keandalan, ketersediaan, dan kinerja Amazon Elastic Container Service dan AWS solusi Anda. Anda harus mengumpulkan data pemantauan dari semua bagian AWS solusi Anda sehingga Anda dapat lebih mudah men-debug kegagalan multi-titik jika terjadi. AWS menyediakan beberapa alat untuk memantau sumber daya Amazon ECS Anda dan menanggapi potensi insiden:

CloudWatch Alarm Amazon

> Melihat metrik tunggal selama periode waktu yang Anda tentukan, dan lakukan satu atau beberapa tindakan berdasarkan nilai metrik relatif terhadap ambang batas tertentu selama

sejumlah periode waktu. Tindakannya adalah pemberitahuan yang dikirim ke topik Amazon Simple Notification Service (Amazon SNS) atau kebijakan Amazon EC2 Auto Scaling. CloudWatch alarm tidak memanggil tindakan hanya karena mereka berada dalam keadaan tertentu; negara harus telah berubah dan dipertahankan untuk sejumlah periode tertentu. Untuk informasi selengkapnya, lihat Pantau Amazon ECS menggunakan CloudWatch .

Untuk layanan dengan tugas yang menggunakan tipe peluncuran Fargate, Anda dapat menggunakan CloudWatch alarm untuk menskalakan dan memperkecil tugas dalam layanan Anda berdasarkan CloudWatch metrik, seperti CPU dan pemanfaatan memori. Untuk informasi selengkapnya, lihat Secara otomatis menskalakan layanan Amazon ECS Anda.

Untuk cluster dengan tugas atau layanan yang menggunakan tipe peluncuran EC2, Anda dapat menggunakan CloudWatch alarm untuk menskalakan dan memperkecil instance container berdasarkan CloudWatch metrik, seperti reservasi memori cluster.

CloudWatch Log Amazon

Pantau, simpan, dan akses file log dari kontainer di tugas Amazon ECS Anda dengan menentukan driver `awslogs` log dalam definisi tugas Anda. Untuk informasi selengkapnya, lihat Menggunakan driver awslogs.

Anda juga dapat memantau, menyimpan, dan mengakses sistem operasi dan file log agen penampung Amazon ECS dari instans penampung Amazon ECS Anda. Metode ini untuk mengakses log dapat digunakan untuk kontainer menggunakan tipe peluncuran EC2..

CloudWatch Acara Amazon

Mencocokkan peristiwa dan merutekan mereka ke satu atau beberapa fungsi atau aliran target untuk membuat perubahan, menangkap informasi status, dan mengambil tindakan korektif. Untuk informasi lebih lanjut, lihat Otomatiskan tanggapan terhadap kesalahan Amazon ECS menggunakan EventBridge di panduan ini dan Apa Itu CloudWatch Acara Amazon? di Panduan Pengguna CloudWatch Acara Amazon.

AWS CloudTrail Log

CloudTrail menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di Amazon ECS. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat ke Amazon ECS, alamat IP dari mana permintaan itu dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan. Untuk informasi selengkapnya, lihat Log panggilan Amazon ECS API menggunakan AWS CloudTrail.

AWS Trusted Advisor

Trusted Advisor mengacu pada praktik terbaik yang dipelajari dari melayani ratusan ribu AWS pelanggan. Trusted Advisor memeriksa AWS lingkungan Anda dan kemudian membuat rekomendasi ketika ada peluang untuk menghemat uang, meningkatkan ketersediaan dan kinerja sistem, atau membantu menutup kesenjangan keamanan. Semua AWS pelanggan memiliki akses ke lima Trusted Advisor cek. Pelanggan dengan paket dukungan Bisnis atau Perusahaan dapat melihat semua Trusted Advisor pemeriksaan.

Untuk informasi selengkapnya, lihat AWS Trusted Advisor di AWS Support Panduan Pengguna.

AWS Compute Optimizer

AWS Compute Optimizer adalah layanan yang menganalisis konfigurasi dan metrik pemanfaatan sumber daya Anda. AWS Ini melaporkan apakah sumber daya Anda optimal, dan menghasilkan rekomendasi pengoptimalan untuk mengurangi biaya dan meningkatkan kinerja beban kerja Anda.

Untuk informasi selengkapnya, lihat AWS Compute Optimizer rekomendasi untuk Amazon ECS.

Bagian penting lainnya dari pemantauan Amazon ECS melibatkan pemantauan secara manual item-item yang tidak CloudWatch tercakup oleh alarm. Dasbor AWS konsol CloudWatch Trusted Advisor,, dan lainnya memberikan at-a-glance tampilan status AWS lingkungan Anda. Kami menyarankan Anda untuk memeriksa file log pada instans kontainer Anda dan kontainer dalam tugas Anda.

Validasi kepatuhan untuk Amazon Elastic Container Service

Untuk mempelajari apakah an Layanan AWS berada dalam lingkup program kepatuhan tertentu, lihat Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat Program AWS Kepatuhan Program AWS .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact .

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- Panduan Memulai Cepat Keamanan dan Kepatuhan — Panduan penerapan ini membahas pertimbangan arsitektur dan memberikan langkah-langkah untuk menerapkan lingkungan dasar AWS yang berfokus pada keamanan dan kepatuhan.
- Arsitektur untuk Keamanan dan Kepatuhan HIPAA di Amazon Web Services — Whitepaper ini menjelaskan bagaimana perusahaan dapat menggunakan AWS untuk membuat aplikasi yang memenuhi syarat HIPAA.

> **Note**
>
> Tidak semua memenuhi Layanan AWS syarat HIPAA. Untuk informasi selengkapnya, lihat Referensi Layanan yang Memenuhi Syarat HIPAA.

- AWS Sumber Daya AWS — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- AWS Panduan Kepatuhan Pelanggan - Memahami model tanggung jawab bersama melalui lensa kepatuhan. Panduan ini merangkum praktik terbaik untuk mengamankan Layanan AWS dan memetakan panduan untuk kontrol keamanan di berbagai kerangka kerja (termasuk Institut Standar dan Teknologi Nasional (NIST), Dewan Standar Keamanan Industri Kartu Pembayaran (PCI), dan Organisasi Internasional untuk Standardisasi (ISO)).
- Mengevaluasi Sumber Daya dengan Aturan dalam Panduan AWS Config Pengembang — AWS Config Layanan menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- AWS Security Hub— Ini Layanan AWS memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS. Security Hub menggunakan kontrol keamanan untuk mengevaluasi sumber daya AWS Anda dan memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik. Untuk daftar layanan dan kontrol yang didukung, lihat Referensi kontrol Security Hub.
- AWS Audit ManagerIni Layanan AWS membantu Anda terus mengaudit AWS penggunaan Anda untuk menyederhanakan cara Anda mengelola risiko dan kepatuhan terhadap peraturan dan standar industri.

AWS Fargate Standar Pemrosesan Informasi Federal (FIPS-140)

Standar Pemrosesan Informasi Federal (FIPS). FIPS-140 adalah standar pemerintah AS dan Kanada yang menetapkan persyaratan keamanan untuk modul kriptografi yang melindungi informasi sensitif.

FIPS-140 mendefinisikan satu set fungsi kriptografi yang divalidasi yang dapat digunakan untuk mengenkripsi data dalam transit dan data saat istirahat.

Saat Anda mengaktifkan kepatuhan FIPS-140, Anda dapat menjalankan beban kerja di Fargate dengan cara yang sesuai dengan FIPS-140. Untuk informasi selengkapnya tentang kepatuhan FIPS-140, lihat Federal Information Processing Standard (FIPS) 140-2.

Pertimbangan

Pertimbangkan hal berikut saat menggunakan kepatuhan FIPS-140 pada Fargate:

- Kepatuhan FIPS-140 hanya tersedia di Wilayah. AWS GovCloud (US)
- Kepatuhan FIPS-140 dimatikan secara default. Anda harus menyalakannya.
- Tugas Anda harus menggunakan konfigurasi berikut untuk kepatuhan FIPS-140:
  - `operatingSystemFamily`Pasti`LINUX`.
  - `cpuArchitecture`Pasti`X86_64`.
  - Versi platform Fargate harus `1.4.0` atau lebih baru.

Gunakan FIPS di Fargate

Gunakan prosedur berikut untuk menggunakan kepatuhan FIPS-140 pada Fargate.

1. Nyalakan kepatuhan FIPS-140. Untuk informasi selengkapnya, lihat the section called “AWS Fargate Kepatuhan Standar Pemrosesan Informasi Federal (FIPS-140)”.
2. Anda dapat menggunakan ECS Exec secara opsional untuk menjalankan perintah berikut untuk memverifikasi status kepatuhan FIPS-140 untuk sebuah cluster.

   Ganti *my-cluster* dengan nama klaster Anda.

   Nilai pengembalian “1" menunjukkan bahwa Anda menggunakan FIPS.

   ```
   aws ecs execute-command --cluster cluster-name \
       --interactive \
       --command "cat /proc/sys/crypto/fips_enabled"
   ```

Gunakan CloudTrail untuk audit

CloudTrail diaktifkan di AWS akun Anda saat Anda membuat akun. Saat aktivitas API dan konsol terjadi di Amazon ECS, aktivitas tersebut direkam dalam suatu CloudTrail peristiwa bersama dengan peristiwa AWS layanan lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh acara terbaru di AWS akun Anda. Untuk informasi selengkapnya, lihat Melihat Acara dengan Riwayat CloudTrail Acara.

Untuk catatan peristiwa yang sedang berlangsung di AWS akun Anda, termasuk peristiwa untuk Amazon ECS, buat jejak yang CloudTrail digunakan untuk mengirimkan file log ke bucket Amazon S3. Secara default, ketika Anda membuat jejak di konsol tersebut, jejak diterapkan ke semua Wilayah . Jejak mencatat peristiwa dari semua Wilayah di AWS partisi dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi selengkapnya, lihat the section called “Log panggilan Amazon ECS API menggunakan AWS CloudTrail”.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan tindakan `PutAccountSettingDefault` API:

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAIV5AJI5LXF5EXAMPLE",
         "arn": "arn:aws:iam::123456789012:user/jdoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIPWIOFC3EXAMPLE",
    },
    "eventTime": "2023-03-01T21:45:18Z",
    "eventSource": "ecs.amazonaws.com",
    "eventName": "PutAccountSettingDefault",
    "awsRegion": "us-gov-east-1",
    "sourceIPAddress": "52.94.133.131",
    "userAgent": "aws-cli/2.9.8 Python/3.9.11 Windows/10 exe/AMD64 prompt/off command/
ecs.put-account-setting",
    "requestParameters": {
        "name": "fargateFIPSMode",
        "value": "enabled"
    },
    "responseElements": {
```

```
        "setting": {
            "name": "fargateFIPSMode",
            "value": "enabled",
            "principalArn": "arn:aws:iam::123456789012:user/jdoe"
        }
    },
    "requestID": "acdc731e-e506-447c-965d-f5f75EXAMPLE",
    "eventID": "6afced68-75cd-4d44-8076-0beEXAMPLE",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
        "clientProvidedHostHeader": "ecs-fips.us-gov-east-1.amazonaws.com"
    }
}
```

Keamanan Infrastruktur di Amazon Elastic Container Service

Sebagai layanan terkelola, Amazon Elastic Container Service dilindungi oleh keamanan jaringan AWS global. Untuk informasi tentang layanan AWS keamanan dan cara AWS melindungi infrastruktur, lihat Keamanan AWS Cloud. Untuk mendesain AWS lingkungan Anda menggunakan praktik terbaik untuk keamanan infrastruktur, lihat Perlindungan Infrastruktur dalam Kerangka Kerja yang AWS Diarsiteksikan dengan Baik Pilar Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses Amazon ECS melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Pengangkutan (TLS). Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Sandi cocok dengan sistem kerahasiaan maju sempurna (perfect forward secrecy, PFS) seperti DHE (Ephemeral Diffie-Hellman) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Selain itu, permintaan harus ditandatangani dengan menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan pengguna utama IAM. Atau Anda dapat menggunakan AWS Security Token Service (AWS STS) untuk menghasilkan kredensial keamanan sementara untuk menandatangani permintaan.

Anda dapat memanggil operasi API ini dari lokasi jaringan mana pun. Amazon ECS mendukung kebijakan akses berbasis sumber daya, yang dapat mencakup pembatasan berdasarkan alamat IP sumber, jadi pastikan kebijakan tersebut memperhitungkan alamat IP untuk lokasi jaringan. Anda juga dapat menggunakan kebijakan Amazon ECS untuk mengontrol akses dari titik akhir Amazon Virtual Private Cloud tertentu atau VPC tertentu. Secara efektif, ini mengisolasi akses jaringan ke sumber daya Amazon ECS tertentu hanya dari VPC tertentu dalam jaringan. AWS Untuk informasi selengkapnya, lihat Titik akhir VPC antarmuka Amazon ECS ()AWS PrivateLink.

Titik akhir VPC antarmuka Amazon ECS ()AWS PrivateLink

Anda dapat meningkatkan postur keamanan VPC Anda dengan mengonfigurasi Amazon ECS untuk menggunakan titik akhir VPC antarmuka. Endpoint antarmuka didukung oleh AWS PrivateLink, teknologi yang memungkinkan Anda mengakses Amazon ECS API secara pribadi dengan menggunakan alamat IP pribadi. AWS PrivateLink membatasi semua lalu lintas jaringan antara VPC Anda dan Amazon ECS ke jaringan Amazon. Anda tidak memerlukan sebuah gateway internet, perangkat NAT, atau gateway privat virtual.

Untuk informasi selengkapnya tentang AWS PrivateLink dan titik akhir VPC, lihat Titik Akhir VPC di Panduan Pengguna Amazon VPC.

Pertimbangan untuk titik akhir Amazon ECS VPC

Pertimbangan untuk titik akhir Amazon ECS VPC untuk jenis peluncuran Fargate

Sebelum Anda mengatur titik akhir VPC antarmuka untuk Amazon ECS, perhatikan pertimbangan berikut:

- Tugas yang menggunakan tipe peluncuran Fargate tidak memerlukan titik akhir VPC antarmuka untuk Amazon ECS, tetapi Anda mungkin memerlukan titik akhir VPC antarmuka untuk Amazon ECR, Secrets Manager, atau Amazon Logs yang dijelaskan dalam poin berikut. CloudWatch
  - Untuk memungkinkan tugas Anda menarik gambar pribadi dari Amazon ECR, Anda harus membuat titik akhir VPC antarmuka untuk Amazon ECR. Untuk informasi selengkapnya, lihat Antarmuka VPC Endpoints (AWS PrivateLink) di Panduan Pengguna Amazon Elastic Container Registry.

    Jika VPC Anda tidak memiliki gateway internet, Anda harus membuat titik akhir gateway untuk Amazon S3. Untuk informasi selengkapnya, lihat Membuat titik akhir gateway Amazon S3 di Panduan Pengguna Amazon Elastic Container Registry. Titik akhir antarmuka untuk Amazon S3 tidak dapat digunakan dengan Amazon ECR.

> **Important**
>
> Jika Anda mengonfigurasi Amazon ECR untuk menggunakan titik akhir VPC antarmuka, Anda dapat membuat peran eksekusi tugas yang menyertakan kunci kondisi untuk membatasi akses ke titik akhir VPC atau VPC tertentu. Untuk informasi selengkapnya, lihat Izin IAM opsional untuk tugas Fargate yang menarik gambar Amazon ECR melalui titik akhir antarmuka.

- Untuk memungkinkan tugas Anda menarik data sensitif dari Secrets Manager, Anda harus membuat titik akhir VPC antarmuka untuk Secrets Manager. Untuk informasi selengkapnya, lihat Menggunakan Secrets Manager dengan VPC Endpoint dalam Panduan Pengguna AWS Secrets Manager .
- Jika VPC Anda tidak memiliki gateway internet dan tugas Anda menggunakan driver `awslogs` log untuk mengirim informasi log ke Log, Anda harus membuat antarmuka VPC endpoint untuk CloudWatch Log. CloudWatch Untuk informasi selengkapnya, lihat Menggunakan CloudWatch Log dengan Titik Akhir VPC Antarmuka di Panduan Pengguna Amazon CloudWatch Logs.

- VPC endpoint saat ini tidak mendukung permintaan lintas Wilayah. Pastikan Anda membuat titik akhir di Wilayah yang sama tempat Anda berencana mengeluarkan panggilan API ke Amazon ECS. Misalnya, asumsikan bahwa Anda ingin menjalankan tugas di AS Timur (Virginia N.). Kemudian, Anda harus membuat titik akhir Amazon ECS VPC di AS Timur (Virginia N.). Titik akhir VPC Amazon ECS yang dibuat di wilayah lain mana pun tidak dapat menjalankan tugas di AS Timur (Virginia Utara).
- Titik akhir VPC hanya mendukung DNS yang disediakan Amazon melalui Amazon Route 53. Jika Anda ingin menggunakan DNS Anda sendiri, Anda dapat menggunakan penerusan DNS bersyarat. Untuk informasi selengkapnya, lihat Pengaturan DHCP dalam Panduan Pengguna Amazon VPC.
- Grup keamanan yang terpasang pada titik akhir VPC harus mengizinkan koneksi masuk pada port TCP 443 dari subnet pribadi VPC.
- Manajemen Service Connect dari proxy Envoy menggunakan titik akhir VPC`com.amazonaws.`*`region`*`.ecs-agent`. Bila Anda tidak menggunakan endpoint VPC, manajemen Service Connect dari proxy Envoy menggunakan endpoint di Region tersebut`ecs-sc`. Untuk daftar titik akhir Amazon ECS di setiap Wilayah, lihat titik akhir dan kuota Amazon ECS.

Pertimbangan untuk titik akhir Amazon ECS VPC untuk jenis peluncuran EC2

Sebelum Anda mengatur titik akhir VPC antarmuka untuk Amazon ECS, perhatikan pertimbangan berikut:

- Tugas yang menggunakan tipe peluncuran EC2 mengharuskan instans penampung yang diluncurkan untuk menjalankan versi `1.25.1` atau yang lebih baru dari agen penampung Amazon ECS. Untuk informasi selengkapnya, lihat Manajemen instance kontainer Linux.
- Untuk memungkinkan tugas Anda menarik data sensitif dari Secrets Manager, Anda harus membuat titik akhir VPC antarmuka untuk Secrets Manager. Untuk informasi selengkapnya, lihat Menggunakan Secrets Manager dengan VPC Endpoint dalam Panduan Pengguna AWS Secrets Manager .
- Jika VPC Anda tidak memiliki gateway internet dan tugas Anda menggunakan driver `awslogs` log untuk mengirim informasi log ke Log, Anda harus membuat antarmuka VPC endpoint untuk CloudWatch Log. CloudWatch Untuk informasi selengkapnya, lihat Menggunakan CloudWatch Log dengan Titik Akhir VPC Antarmuka di Panduan Pengguna Amazon CloudWatch Logs.
- VPC endpoint saat ini tidak mendukung permintaan lintas Wilayah. Pastikan Anda membuat titik akhir di Wilayah yang sama tempat Anda berencana mengeluarkan panggilan API ke Amazon ECS. Misalnya, asumsikan bahwa Anda ingin menjalankan tugas di AS Timur (Virginia N.). Kemudian, Anda harus membuat titik akhir Amazon ECS VPC di AS Timur (Virginia N.). Titik akhir VPC Amazon ECS yang dibuat di wilayah lain mana pun tidak dapat menjalankan tugas di AS Timur (Virginia Utara).
- Titik akhir VPC hanya mendukung DNS yang disediakan Amazon melalui Amazon Route 53. Jika Anda ingin menggunakan DNS Anda sendiri, Anda dapat menggunakan penerusan DNS bersyarat. Untuk informasi selengkapnya, lihat Pengaturan DHCP dalam Panduan Pengguna Amazon VPC.
- Grup keamanan yang terpasang pada titik akhir VPC harus mengizinkan koneksi masuk pada port TCP 443 dari subnet pribadi VPC.

Membuat Endpoint VPC untuk Amazon ECS

Untuk membuat titik akhir VPC untuk layanan Amazon ECS, gunakan prosedur Membuat Titik Akhir Antarmuka di Panduan Pengguna Amazon VPC untuk membuat titik akhir berikut. Jika Anda memiliki instans kontainer yang berada dalam VPC Anda, Anda harus membuat titik akhir dalam urutan terdaftar mereka. Urutan langkah tidak membuat masalah jika Anda berencana untuk membuat VPC endpoint terlebih dahulu kemudian instans kontainer Anda.

- `com.amazonaws.region.ecs-agent`

- `com.amazonaws.region.ecs-telemetry`
- `com.amazonaws.region.ecs`

> **Note**
>
> *wilayah* mewakili pengenal Wilayah untuk AWS Wilayah yang didukung oleh Amazon ECS, seperti `us-east-2` untuk Wilayah AS Timur (Ohio).

`ecs-agent`Titik akhir menggunakan `ecs:poll` API, dan `ecs-telemetry` titik akhir menggunakan API `ecs:poll` dan`ecs:StartTelemetrySession`.

Jika Anda memiliki tugas yang ada yang menggunakan tipe peluncuran EC2, setelah Anda membuat titik akhir VPC, setiap instance container perlu mengambil konfigurasi baru. Agar hal ini terjadi, Anda harus me-reboot setiap instance kontainer atau memulai ulang agen penampung Amazon ECS pada setiap instance container. Untuk memulai ulang agen kontainer, lakukan hal berikut.

Untuk memulai ulang agen kontainer Amazon ECS

1. Masuk ke instans kontainer Anda melalui SSH.
2. Hentikan kontainer agen.

   ```
   sudo docker stop ecs-agent
   ```
3. Mulai kontainer agen.

   ```
   sudo docker start ecs-agent
   ```

Setelah Anda membuat titik akhir VPC dan memulai ulang agen penampung Amazon ECS di setiap instance container, semua tugas yang baru diluncurkan akan mengambil konfigurasi baru.

Membuat kebijakan titik akhir VPC untuk Amazon ECS

Anda dapat melampirkan kebijakan titik akhir ke titik akhir VPC yang mengontrol akses ke Amazon ECS. Kebijakan titik akhir menentukan informasi berikut:

- Prinsipal yang dapat melakukan tindakan.
- Tindakan yang dapat dilakukan.

- Sumber daya yang menjadi target tindakan.

Untuk informasi selengkapnya, lihat Mengontrol akses ke layanan dengan titik akhir VPC dalam Panduan Pengguna Amazon VPC.

Contoh: Kebijakan titik akhir VPC untuk tindakan Amazon ECS

Berikut ini adalah contoh kebijakan endpoint untuk Amazon ECS. Saat dilampirkan ke titik akhir, kebijakan ini memberikan akses ke izin untuk membuat dan mencantumkan klaster. `ListClusters`Tindakan `CreateCluster` dan tidak menerima sumber daya apa pun, sehingga definisi sumber daya diatur ke* untuk semua sumber daya.

```
{
   "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "ecs:CreateCluster",
        "ecs:ListClusters"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Praktik Terbaik Keamanan

Panduan ini memberikan rekomendasi keamanan dan kepatuhan untuk melindungi informasi, sistem, dan aset Anda lainnya yang bergantung pada Amazon ECS. Ini juga memperkenalkan beberapa penilaian risiko dan strategi mitigasi yang dapat Anda gunakan untuk memiliki pegangan yang lebih baik pada kontrol keamanan yang dibangun untuk cluster Amazon ECS dan beban kerja yang mereka dukung. Setiap topik dalam panduan ini dimulai dengan ikhtisar singkat, diikuti dengan daftar rekomendasi dan praktik terbaik yang dapat Anda gunakan untuk mengamankan kluster Amazon ECS Anda.

Topik

- AWS Identity and Access Management

- Menggunakan peran IAM dengan tugas Amazon ECS
- Keamanan jaringan
- Manajemen rahasia
- Menggunakan kredensial keamanan sementara dengan operasi API
- Kepatuhan dan keamanan
- Pencatatan log dan pemantauan
- AWS Fargate keamanan
- Pertimbangan keamanan contoh kontainer EC2
- Keamanan tugas dan kontainer
- Keamanan runtime
- Praktik terbaik AMI
- AWS Mitra

AWS Identity and Access Management

Anda dapat menggunakan AWS Identity and Access Management (IAM) untuk mengelola dan mengontrol akses ke AWS layanan dan sumber daya Anda melalui kebijakan berbasis aturan untuk tujuan otentikasi dan otorisasi. Lebih khusus lagi, melalui layanan ini, Anda mengontrol akses ke AWS sumber daya Anda dengan menggunakan kebijakan yang diterapkan pada pengguna, grup, atau peran. Di antara ketiganya, pengguna adalah akun yang dapat memiliki akses ke sumber daya Anda. Dan, peran IAM adalah sekumpulan izin yang dapat diasumsikan oleh identitas yang diautentikasi, yang tidak terkait dengan identitas tertentu di luar IAM. Untuk informasi selengkapnya, lihat Gambaran umum manajemen akses: Izin dan kebijakan.

Mengelola akses ke Amazon ECS

Anda dapat mengontrol akses ke Amazon ECS dengan membuat dan menerapkan kebijakan IAM. Kebijakan ini terdiri dari serangkaian tindakan yang berlaku untuk serangkaian sumber daya tertentu. Tindakan kebijakan menentukan daftar operasi (seperti Amazon ECS API) yang diizinkan atau ditolak, sedangkan sumber daya mengontrol objek Amazon ECS yang diterapkan oleh tindakan tersebut. Kondisi dapat ditambahkan ke kebijakan untuk mempersempit ruang lingkupnya. Misalnya, kebijakan dapat ditulis untuk hanya mengizinkan tindakan dilakukan terhadap tugas dengan kumpulan tag tertentu. Untuk informasi selengkapnya, lihat Cara Amazon ECS bekerja dengan IAM di Panduan Pengembang Layanan Kontainer Elastis Amazon.

Rekomendasi

Kami menyarankan Anda melakukan hal berikut saat menyiapkan peran dan kebijakan IAM Anda.

Ikuti kebijakan akses yang paling tidak memiliki hak istimewa

Buat kebijakan yang dicakup untuk memungkinkan pengguna melakukan pekerjaan yang ditentukan. Misalnya, jika pengembang perlu menghentikan tugas secara berkala, buat kebijakan yang hanya mengizinkan tindakan tertentu tersebut. Contoh berikut hanya memungkinkan pengguna untuk menghentikan tugas yang dimiliki oleh tertentu `task_family` di klaster dengan Nama Sumber Daya Amazon (ARN) tertentu. Mengacu pada ARN dalam suatu kondisi juga merupakan contoh penggunaan izin tingkat sumber daya. Anda dapat menggunakan izin tingkat sumber daya untuk menentukan sumber daya yang Anda inginkan untuk diterapkan tindakan.

> **Note**
>
> Saat mereferensikan ARN dalam kebijakan, gunakan format ARN baru yang lebih panjang. Untuk informasi selengkapnya, lihat Nama Sumber Daya Amazon (ARN) dan ID di Panduan Pengembang Layanan Amazon Elastic Container.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:StopTask"
      ],
      "Condition": {
        "ArnEquals": {
          "ecs:cluster": "arn:aws:ecs:region:account_id:cluster/cluster_name"
        }
      },
      "Resource": [
        "arn:aws:ecs:region:account_id:task-definition/task_family:*"
      ]
    }
  ]
}
```

Biarkan sumber daya cluster berfungsi sebagai batas administratif

Kebijakan yang terlalu sempit cakupan dapat menyebabkan proliferasi peran dan meningkatkan overhead administratif. Daripada membuat peran yang tercakup pada tugas atau layanan tertentu saja, buat peran yang tercakup ke cluster dan gunakan klaster sebagai batas administratif utama Anda.

Mengisolasi pengguna akhir dari Amazon ECS API dengan membuat pipeline otomatis

Anda dapat membatasi tindakan yang dapat digunakan pengguna dengan membuat pipeline yang secara otomatis mengemas dan menyebarkan aplikasi ke kluster Amazon ECS. Ini secara efektif mendelegasikan tugas membuat, memperbarui, dan menghapus tugas ke pipeline. Untuk informasi selengkapnya, lihat Tutorial: Penerapan standar Amazon ECS dengan CodePipeline di AWS CodePipeline Panduan Pengguna.

Gunakan kondisi kebijakan untuk lapisan keamanan tambahan

Jika Anda membutuhkan lapisan keamanan tambahan, tambahkan kondisi ke kebijakan Anda. Ini dapat berguna jika Anda melakukan operasi istimewa atau ketika Anda perlu membatasi serangkaian tindakan yang dapat dilakukan terhadap sumber daya tertentu. Contoh kebijakan berikut memerlukan otorisasi multi-faktor saat menghapus klaster.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:DeleteCluster"
      ],
      "Condition": {
        "Bool": {
          "aws:MultiFactorAuthPresent": "true"
        }
      },
    "Resource": ["*"]
    }
  ]
}
```

Tag yang diterapkan pada layanan disebarkan ke semua tugas yang merupakan bagian dari layanan itu. Karena itu, Anda dapat membuat peran yang tercakup ke sumber daya Amazon ECS dengan tag

tertentu. Dalam kebijakan berikut, prinsipal IAM memulai dan menghentikan semua tugas dengan kunci tag dan nilai tag. `Department` `Accounting`

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:StartTask",
                "ecs:StopTask",
                "ecs:RunTask"
            ],
            "Resource": "arn:aws:ecs:*",
            "Condition": {
                "StringEquals": {"ecs:ResourceTag/Department": "Accounting"}
            }
        }
    ]
}
```

Mengaudit akses secara berkala ke Amazon ECS API

Seorang pengguna dapat mengubah peran. Setelah mereka mengubah peran, izin yang sebelumnya diberikan kepada mereka mungkin tidak lagi berlaku. Pastikan Anda mengaudit siapa yang memiliki akses ke Amazon ECS API dan apakah akses tersebut masih diperlukan. Pertimbangkan untuk mengintegrasikan IAM dengan solusi manajemen siklus hidup pengguna yang secara otomatis mencabut akses saat pengguna meninggalkan organisasi. Untuk informasi selengkapnya, lihat pedoman audit keamanan Amazon ECS di bagian. Referensi Umum Amazon Web Services

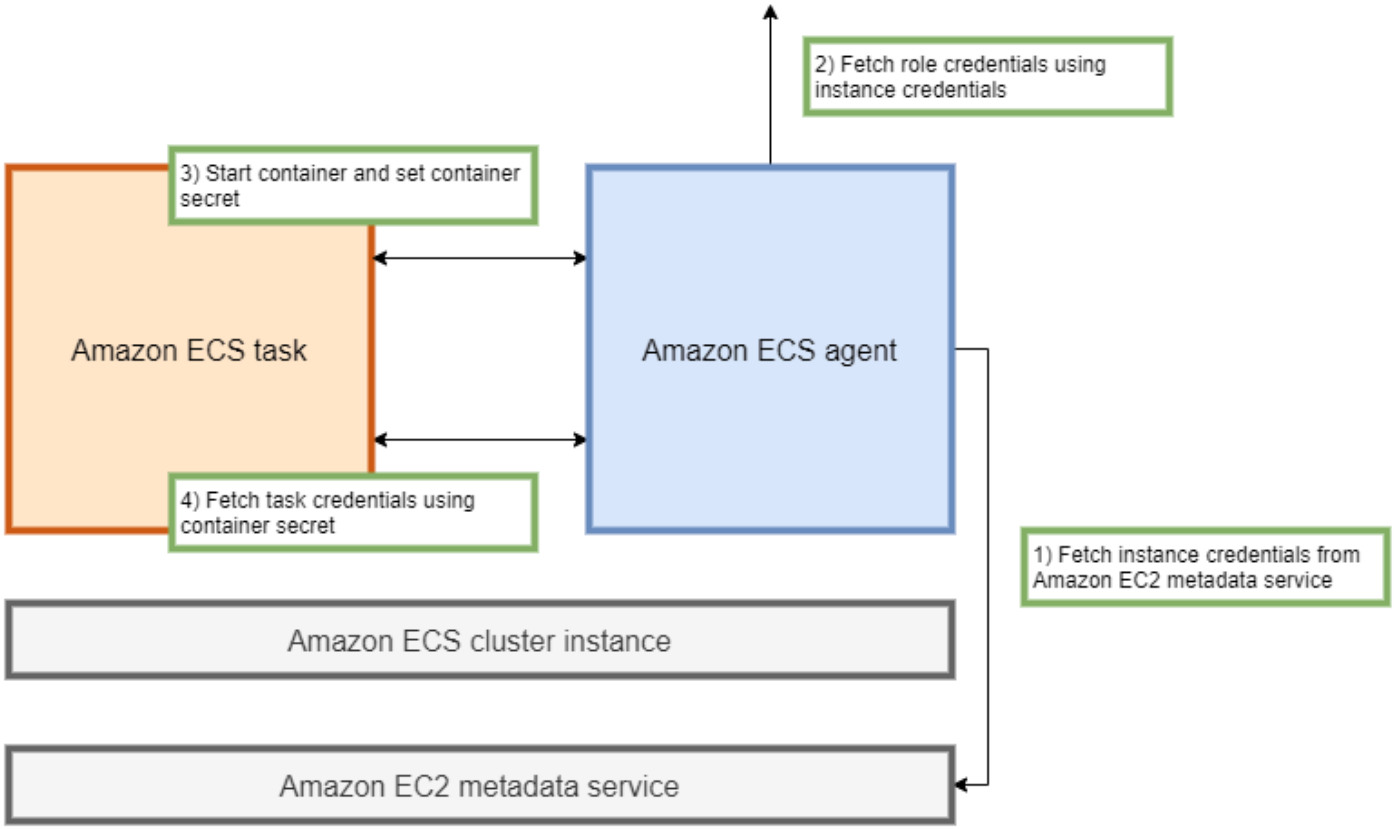
Menggunakan peran IAM dengan tugas Amazon ECS

Kami menyarankan Anda menetapkan tugas peran IAM. Perannya dapat dibedakan dari peran instans Amazon EC2 yang sedang berjalan. Menetapkan setiap tugas peran selaras dengan prinsip akses yang paling tidak memiliki hak istimewa dan memungkinkan kontrol terperinci yang lebih besar atas tindakan dan sumber daya.

Saat menetapkan peran IAM untuk tugas, Anda harus menggunakan kebijakan kepercayaan berikut sehingga setiap tugas Anda dapat mengambil peran IAM yang berbeda dari yang digunakan instans EC2 Anda. Dengan cara ini, tugas Anda tidak mewarisi peran instans EC2 Anda.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs-tasks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Saat Anda menambahkan peran tugas ke definisi tugas, agen penampung Amazon ECS secara otomatis membuat token dengan ID kredensyal unik (misalnya,`12345678-90ab-cdef-1234-567890abcdef`) untuk tugas tersebut. Token ini dan kredensyal peran kemudian ditambahkan ke cache internal agen. Agen mengisi variabel lingkungan `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` dalam wadah dengan URI ID kredensyal (misalnya,`/v2/credentials/12345678-90ab-cdef-1234-567890abcdef`).



Anda dapat secara manual mengambil kredensyal peran sementara dari dalam wadah dengan menambahkan variabel lingkungan ke alamat IP agen penampung Amazon ECS dan menjalankan `curl` perintah pada string yang dihasilkan.

```
curl 169.254.170.2$AWS_CONTAINER_CREDENTIALS_RELATIVE_URI
```

Output yang diharapkan adalah sebagai berikut:

```
{
 "RoleArn": "arn:aws:iam::123456789012:role/SSMTaskRole-SSMFargateTaskIAMRole-
DASWWSF2WGD6",
 "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
 "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
 "Token": "IQoJb3JpZ2luX2VjEEM/Example==",
 "Expiration": "2021-01-16T00:51:53Z"
}
```

Versi AWS SDK yang lebih baru secara otomatis mengambil kredensi ini dari variabel `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` lingkungan saat melakukan panggilan API. AWS

Outputnya mencakup pasangan kunci akses yang terdiri dari ID kunci akses rahasia dan kunci rahasia yang digunakan aplikasi Anda untuk mengakses AWS sumber daya. Ini juga termasuk token yang AWS digunakan untuk memverifikasi bahwa kredensialnya valid. Secara default, kredenal yang ditetapkan ke tugas menggunakan peran tugas valid selama enam jam. Setelah itu, mereka secara otomatis diputar oleh agen kontainer Amazon ECS.

Peran pelaksanaan tugas

Peran eksekusi tugas digunakan untuk memberikan izin agen penampung Amazon ECS untuk memanggil tindakan AWS API tertentu atas nama Anda. Misalnya, saat Anda menggunakan AWS Fargate, Fargate memerlukan peran IAM yang memungkinkannya menarik gambar dari Amazon ECR dan menulis log ke Log. CloudWatch Peran IAM juga diperlukan saat tugas mereferensikan rahasia yang disimpan AWS Secrets Manager, seperti rahasia tarik gambar.

> **Note**
>
> Jika Anda menarik gambar sebagai pengguna yang diautentikasi, Anda cenderung tidak terpengaruh oleh perubahan yang terjadi pada batas kecepatan tarik Docker Hub. Untuk informasi selengkapnya lihat, Autentikasi registri pribadi untuk instance kontainer. Dengan menggunakan Amazon ECR dan Amazon ECR Public, Anda dapat menghindari batasan yang diberlakukan oleh Docker. Jika Anda menarik gambar dari Amazon ECR, ini juga membantu mempersingkat waktu tarik jaringan dan mengurangi perubahan transfer data saat lalu lintas meninggalkan VPC Anda.

> **Important**
>
> Saat Anda menggunakan Fargate, Anda harus mengautentikasi ke registri gambar pribadi menggunakan. `repositoryCredentials` Tidak mungkin menyetel variabel lingkungan agen penampung Amazon ECS `ECS_ENGINE_AUTH_TYPE` `ECS_ENGINE_AUTH_DATA` atau memodifikasi `ecs.config` file untuk tugas yang dihosting di Fargate. Untuk informasi selengkapnya, lihat Autentikasi registri pribadi untuk tugas.

Peran instans wadah Amazon EC2

Agen kontainer Amazon ECS adalah wadah yang berjalan di setiap instans Amazon EC2 di cluster Amazon ECS. Ini diinisialisasi di luar Amazon ECS menggunakan `init` perintah

yang tersedia di sistem operasi. Akibatnya, itu tidak dapat diberikan izin melalui peran tugas. Sebagai gantinya, izin harus ditetapkan ke instans Amazon EC2 yang dijalankan agen. Daftar tindakan dalam `AmazonEC2ContainerServiceforEC2Role` kebijakan contoh harus diberikan kepada`ecsInstanceRole`. Jika Anda tidak melakukan ini, instance Anda tidak dapat bergabung dengan cluster.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeTags",
                "ecs:CreateCluster",
                "ecs:DeregisterContainerInstance",
                "ecs:DiscoverPollEndpoint",
                "ecs:Poll",
                "ecs:RegisterContainerInstance",
                "ecs:StartTelemetrySession",
                "ecs:UpdateContainerInstancesState",
                "ecs:Submit*",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

Dalam kebijakan ini, tindakan `ecr` dan `logs` API memungkinkan container yang berjalan pada instance Anda untuk menarik gambar dari Amazon ECR dan menulis log ke Amazon. CloudWatch `ecs`Tindakan tersebut memungkinkan agen untuk mendaftarkan dan membatalkan pendaftaran instans dan berkomunikasi dengan pesawat kontrol Amazon ECS. Dari jumlah tersebut, `ecs:CreateCluster` tindakannya opsional.

Peran terkait layanan

Anda dapat menggunakan peran terkait layanan untuk Amazon ECS untuk memberikan izin layanan Amazon ECS untuk memanggil API layanan lain atas nama Anda. Amazon ECS memerlukan izin untuk membuat dan menghapus antarmuka jaringan, mendaftar, dan membatalkan pendaftaran target dengan grup target. Ini juga membutuhkan izin yang diperlukan untuk membuat dan menghapus kebijakan penskalaan. Izin ini diberikan melalui peran terkait layanan. Peran ini dibuat atas nama Anda saat pertama kali Anda menggunakan layanan.

> **Note**
>
> Jika Anda secara tidak sengaja menghapus peran terkait layanan, Anda dapat membuatnya kembali. Untuk petunjuknya, lihat Membuat peran terkait layanan.

Rekomendasi

Kami menyarankan Anda melakukan hal berikut saat menyiapkan peran dan kebijakan IAM tugas Anda.

Blokir akses ke metadata Amazon EC2

Saat menjalankan tugas di instans Amazon EC2, kami sangat menyarankan Anda memblokir akses ke metadata Amazon EC2 untuk mencegah container mewarisi peran yang ditetapkan ke instans tersebut. Jika aplikasi Anda harus memanggil tindakan AWS API, gunakan peran IAM untuk tugas sebagai gantinya.

Untuk mencegah tugas yang berjalan dalam mode jembatan mengakses metadata Amazon EC2, jalankan perintah berikut atau perbarui data pengguna instans. Untuk instruksi selengkapnya tentang memperbarui data pengguna sebuah instans, lihat Artikel AWS Support ini. Untuk informasi selengkapnya tentang mode jembatan definisi tugas, lihat mode jaringan definisi tugas.

```
sudo yum install -y iptables-services; sudo iptables --insert FORWARD 1 --in-interface
 docker+ --destination 192.0.2.0/32 --jump DROP
```

Agar perubahan ini tetap ada setelah reboot, jalankan perintah berikut yang khusus untuk Amazon Machine Image (AMI) Anda:

- Amazon Linux 2

```
sudo iptables-save | sudo tee /etc/sysconfig/iptables && sudo systemctl enable --now
 iptables
```

- Amazon Linux

```
sudo service iptables save
```

Untuk tugas yang menggunakan mode `awsvpc` jaringan, atur variabel lingkungan `ECS_AWSVPC_BLOCK_IMDS` ke `true` dalam `/etc/ecs/ecs.config` file.

Anda harus menyetel `ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST` variabel ke `false` dalam `ecs-agent config` file untuk mencegah kontainer yang berjalan di dalam `host` jaringan mengakses metadata Amazon EC2.

Gunakan mode **`awsvpc`** jaringan

Gunakan mode jaringan `awsvpc` jaringan untuk membatasi arus lalu lintas antara tugas yang berbeda atau antara tugas Anda dan layanan lain yang berjalan dalam VPC Amazon Anda. Ini menambahkan lapisan keamanan tambahan. Mode `awsvpc` jaringan menyediakan isolasi jaringan tingkat tugas untuk tugas yang berjalan di Amazon EC2. Ini adalah mode default AWS Fargate aktif. itu satu-satunya mode jaringan yang dapat Anda gunakan untuk menetapkan grup keamanan untuk tugas.

Gunakan IAM Access Advisor untuk menyempurnakan peran

Kami menyarankan Anda menghapus tindakan apa pun yang tidak pernah digunakan atau belum digunakan selama beberapa waktu. Ini mencegah akses yang tidak diinginkan terjadi. Untuk melakukan ini, tinjau hasil yang dihasilkan oleh IAM Access Advisor, lalu hapus tindakan yang tidak pernah digunakan atau belum digunakan baru-baru ini. Anda dapat melakukan ini dengan mengikuti langkah-langkah berikut.

Jalankan perintah berikut untuk menghasilkan laporan yang menampilkan informasi akses terakhir untuk kebijakan yang direferensikan:

```
aws iam generate-service-last-accessed-details --arn arn:aws:iam::123456789012:policy/
ExamplePolicy1
```

gunakan `JobId` yang ada di output untuk menjalankan perintah berikut. Setelah Anda melakukan ini, Anda dapat melihat hasil laporan.

```
aws iam get-service-last-accessed-details --job-id 98a765b4-3cde-2101-2345-example678f9
```

Untuk informasi selengkapnya, lihat IAM Access Advisor.

Pantau AWS CloudTrail aktivitas mencurigakan

Anda dapat memantau AWS CloudTrail aktivitas mencurigakan apa pun. Sebagian besar panggilan AWS API dicatat AWS CloudTrail sebagai peristiwa. Mereka dianalisis oleh AWS CloudTrail Insights, dan Anda diberi tahu tentang perilaku mencurigakan apa pun yang terkait dengan `write` panggilan API. Ini mungkin termasuk lonjakan volume panggilan. Peringatan ini mencakup informasi seperti waktu aktivitas yang tidak biasa terjadi dan ARN identitas teratas yang berkontribusi pada API.

Anda dapat mengidentifikasi tindakan yang dilakukan oleh tugas dengan peran IAM AWS CloudTrail dengan melihat `userIdentity` properti acara. Dalam contoh berikut, `arn` termasuk nama peran yang diasumsikan,`s3-write-go-bucket-role`, diikuti dengan nama tugas,`7e9894e088ad416eb5cab92afExample`.

```
"userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROA36C6WWEJ2YEXAMPLE:7e9894e088ad416eb5cab92afExample",
    "arn": "arn:aws:sts::123456789012:assumed-role/s3-write-go-bucket-
role/7e9894e088ad416eb5cab92afExample",
    ...
}
```

> **Note**
>
> Saat tugas yang mengambil peran dijalankan pada instans penampung Amazon EC2, permintaan dicatat oleh agen penampung Amazon ECS ke log audit agen yang terletak di alamat dalam format. `/var/log/ecs/audit.log.YYYY-MM-DD-HH` Untuk informasi selengkapnya, lihat Log Peran IAM Tugas dan Logging Insights Events for Trails.

Keamanan jaringan

Keamanan jaringan adalah topik luas yang mencakup beberapa subtopik. Ini termasuk encryption-in-transit, segmentasi dan isolasi jaringan, firewall, perutean lalu lintas, dan observabilitas.

Enkripsi dalam bergerak

Mengenkripsi lalu lintas jaringan mencegah pengguna yang tidak sah mencegat dan membaca data ketika data tersebut ditransmisikan melalui jaringan. Dengan Amazon ECS, enkripsi jaringan dapat diimplementasikan dengan salah satu cara berikut.

- Dengan service mesh (TLS):

  Dengan AWS App Mesh, Anda dapat mengonfigurasi koneksi TLS antara proxy Envoy yang digunakan dengan titik akhir mesh. Dua contoh adalah virtual node dan virtual gateway. Sertifikat TLS dapat berasal dari AWS Certificate Manager (ACM). Atau, itu bisa berasal dari otoritas sertifikat pribadi Anda sendiri.

  - Mengaktifkan Transport Layer Security (TLS)
  - Aktifkan enkripsi lalu lintas antar layanan dalam AWS App Mesh menggunakan sertifikat ACM atau sertifikat yang disediakan pelanggan
  - Panduan TLS ACM
  - Panduan file TLS
  - Utusan
- Menggunakan instance Nitro:

  Secara default, lalu lintas secara otomatis dienkripsi antara jenis instans Nitro berikut: C5n, G4, i3en, M5dn, M5n, P3dn, R5dn, dan R5n. Lalu lintas tidak dienkripsi saat dialihkan melalui gateway transit, penyeimbang beban, atau perantara serupa.

  - Enkripsi dalam perjalanan
  - Apa pengumuman baru dari 2019
  - Pembicaraan ini dari re:Inforce 2019
- Menggunakan Indikasi Nama Server (SNI) dengan Application Load Balancer:

  Application Load Balancer (ALB) dan Network Load Balancer (NLB) mendukung Server Name Indication (SNI). Dengan menggunakan SNI, Anda dapat menempatkan beberapa aplikasi aman di belakang satu pendengar. Untuk ini, masing-masing memiliki sertifikat TLS sendiri. Kami menyarankan Anda memberikan sertifikat untuk load balancer using AWS Certificate Manager (ACM) dan kemudian menambahkannya ke daftar sertifikat pendengar. AWS Load balancer menggunakan algoritma pemilihan sertifikat cerdas dengan SNI. Jika nama host yang disediakan oleh klien cocok dengan satu sertifikat dalam daftar sertifikat, penyeimbang beban memilih sertifikat tersebut. Jika nama host yang disediakan oleh klien cocok dengan beberapa sertifikat

dalam daftar, penyeimbang beban memilih sertifikat yang dapat didukung klien. Contohnya termasuk sertifikat yang ditandatangani sendiri atau sertifikat yang dihasilkan melalui ACM.

- SNI dengan Application Load Balancer
- SNI dengan Network Load Balancer

- nd-to-end Enkripsi E dengan sertifikat TLS:

  Ini melibatkan penerapan sertifikat TLS dengan tugas. Ini bisa berupa sertifikat yang ditandatangani sendiri atau sertifikat dari otoritas sertifikat tepercaya. Anda dapat memperoleh sertifikat dengan mereferensikan rahasia untuk sertifikat. Jika tidak, Anda dapat memilih untuk menjalankan kontainer yang mengeluarkan Permintaan Penandatanganan Sertifikat (CSR) ke ACM dan kemudian memasang rahasia yang dihasilkan ke volume bersama.

  - Menjaga keamanan lapisan transport sampai ke kontainer Anda menggunakan Network Load Balancer dengan Amazon ECS bagian 1
  - Mempertahankan Transport Layer Security (TLS) sampai ke container Anda bagian 2: Menggunakan AWS Private Certificate Authority

Jaringan tugas

Rekomendasi berikut ini mempertimbangkan cara kerja Amazon ECS. Amazon ECS tidak menggunakan jaringan overlay. Sebagai gantinya, tugas dikonfigurasi untuk beroperasi dalam mode jaringan yang berbeda. Misalnya, tugas yang dikonfigurasi untuk menggunakan `bridge` mode memperoleh alamat IP yang tidak dapat dirutekan dari jaringan Docker yang berjalan di setiap host. Tugas yang dikonfigurasi untuk menggunakan mode `awsvpc` jaringan memperoleh alamat IP dari subnet host. Tugas yang dikonfigurasi dengan `host` jaringan menggunakan antarmuka jaringan host. `awsvpc`adalah mode jaringan yang disukai. Ini karena ini adalah satu-satunya mode yang dapat Anda gunakan untuk menetapkan grup keamanan ke tugas. Ini juga satu-satunya mode yang tersedia untuk AWS Fargate tugas-tugas di Amazon ECS.

Grup keamanan untuk tugas

Kami menyarankan Anda mengonfigurasi tugas Anda untuk menggunakan mode `awsvpc` jaringan. Setelah Anda mengonfigurasi tugas Anda untuk menggunakan mode ini, agen Amazon ECS secara otomatis menyediakan dan melampirkan Antarmuka Jaringan Elastis (ENI) ke tugas tersebut. Ketika ENI disediakan, tugas terdaftar dalam kelompok keamanan. AWS Grup keamanan bertindak sebagai firewall virtual yang dapat Anda gunakan untuk mengontrol lalu lintas masuk dan keluar.

AWS PrivateLink

AWS PrivateLink adalah teknologi jaringan yang memungkinkan Anda membuat titik akhir pribadi untuk berbagai AWS layanan, termasuk Amazon ECS. Titik akhir diperlukan di lingkungan kotak pasir di mana tidak ada Internet Gateway (IGW) yang terpasang ke VPC Amazon dan tidak ada rute alternatif ke Internet. Menggunakan AWS PrivateLink memastikan bahwa panggilan ke layanan Amazon ECS tetap berada dalam VPC Amazon dan tidak melintasi internet. Untuk petunjuk tentang cara membuat AWS PrivateLink titik akhir untuk Amazon ECS dan layanan terkait lainnya, lihat Antarmuka Amazon ECS Titik akhir Amazon VPC.

> **Important**
>
> AWS Fargate tugas tidak memerlukan AWS PrivateLink titik akhir untuk Amazon ECS.

Amazon ECR dan Amazon ECS keduanya mendukung kebijakan titik akhir. Kebijakan ini memungkinkan Anda untuk memperbaiki akses ke API layanan. Misalnya, Anda dapat membuat kebijakan endpoint untuk Amazon ECR yang hanya mengizinkan gambar didorong ke pendaftar di akun tertentu. AWS Kebijakan seperti ini dapat digunakan untuk mencegah data diekstraksi melalui gambar kontainer sambil tetap memungkinkan pengguna untuk mendorong ke pendaftar ECR Amazon resmi. Untuk informasi selengkapnya, lihat Menggunakan kebijakan titik akhir VPC.

Kebijakan berikut memungkinkan semua AWS prinsipal di akun Anda untuk melakukan semua tindakan terhadap hanya repositori Amazon ECR Anda:

```
{
  "Statement": [
    {
      "Sid": "LimitECRAccess",
      "Principal": "*",
      "Action": "*",
      "Effect": "Allow",
      "Resource": "arn:aws:ecr:region:account_id:repository/*"
    },
  ]
}
```

Anda dapat meningkatkan ini lebih lanjut dengan menetapkan kondisi yang menggunakan `PrincipalOrgID` properti baru. Ini mencegah mendorong dan menarik gambar oleh prinsip

IAM yang bukan bagian dari Anda AWS Organizations. Untuk informasi selengkapnya, lihat aws: PrincipalOrg ID.

Kami merekomendasikan menerapkan kebijakan yang sama untuk kedua titik akhir `com.amazonaws.region.ecr.dkr` dan `com.amazonaws.region.ecr.api` titik akhir.

Pengaturan agen kontainer Amazon ECS

File konfigurasi agen penampung Amazon ECS mencakup beberapa variabel lingkungan yang berhubungan dengan keamanan jaringan. `ECS_AWSVPC_BLOCK_IMDS`dan `ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST` digunakan untuk memblokir akses tugas ke metadata Amazon EC2. `HTTP_PROXY`digunakan untuk mengkonfigurasi agen untuk rute melalui proxy HTTP untuk terhubung ke internet. Untuk petunjuk tentang mengonfigurasi agen dan runtime Docker untuk merutekan melalui proxy, lihat Konfigurasi Proxy HTTP.

> **Important**
>
> Pengaturan ini tidak tersedia saat Anda menggunakan AWS Fargate.

Rekomendasi

Kami menyarankan Anda melakukan hal berikut saat menyiapkan VPC Amazon, penyeimbang beban, dan jaringan Anda.

Gunakan enkripsi jaringan jika berlaku

Anda harus menggunakan enkripsi jaringan jika berlaku. Program kepatuhan tertentu, seperti PCI DSS, mengharuskan Anda mengenkripsi data dalam perjalanan jika data berisi data pemegang kartu. Jika beban kerja Anda memiliki persyaratan serupa, konfigurasikan enkripsi jaringan.

Browser modern memperingatkan pengguna saat menghubungkan ke situs yang tidak aman. Jika layanan Anda dipimpin oleh penyeimbang beban yang menghadap publik, gunakan TLS/SSL untuk mengenkripsi lalu lintas dari browser klien ke penyeimbang beban dan mengenkripsi ulang ke backend jika diperlukan.

Gunakan mode **`awsvpc`** jaringan dan grup keamanan saat Anda perlu mengontrol lalu lintas antara tugas atau antara tugas dan sumber daya jaringan lainnya

Anda harus menggunakan mode `awsvpc` jaringan dan grup keamanan ketika Anda perlu mengontrol lalu lintas antara tugas dan antara tugas dan sumber daya jaringan lainnya. Jika layanan Anda di

belakang ALB, gunakan grup keamanan untuk hanya mengizinkan lalu lintas masuk dari sumber daya jaringan lain menggunakan grup keamanan yang sama dengan ALB Anda. Jika aplikasi Anda berada di belakang NLB, konfigurasikan grup keamanan tugas untuk hanya mengizinkan lalu lintas masuk dari rentang CIDR VPC Amazon dan alamat IP statis yang ditetapkan ke NLB.

Grup keamanan juga harus digunakan untuk mengontrol lalu lintas antara tugas dan sumber daya lain dalam VPC Amazon seperti database Amazon RDS.

Buat cluster di VPC Amazon terpisah saat lalu lintas jaringan perlu diisolasi secara ketat

Anda harus membuat cluster di VPC Amazon terpisah ketika lalu lintas jaringan perlu diisolasi secara ketat. Hindari menjalankan beban kerja yang memiliki persyaratan keamanan ketat pada cluster dengan beban kerja yang tidak harus mematuhi persyaratan tersebut. Jika isolasi jaringan yang ketat diperlukan, buat cluster di VPC Amazon terpisah dan secara selektif mengekspos layanan ke VPC Amazon lainnya menggunakan titik akhir Amazon VPC. Untuk informasi selengkapnya, lihat titik akhir Amazon VPC.

Konfigurasikan AWS PrivateLink titik akhir saat diperlukan

Anda harus mengonfigurasi AWS PrivateLink titik akhir titik akhir saat diperlukan. Jika kebijakan keamanan mencegah Anda melampirkan Internet Gateway (IGW) ke VPC Amazon, konfigurasikan AWS PrivateLink titik akhir untuk Amazon ECS dan layanan lain seperti Amazon ECR AWS Secrets Manager, dan Amazon. CloudWatch

Gunakan Amazon VPC Flow Logs untuk menganalisis lalu lintas ke dan dari tugas yang berjalan lama

Anda harus menggunakan Amazon VPC Flow Logs untuk menganalisis lalu lintas ke dan dari tugas yang berjalan lama. Tugas yang menggunakan mode `awsvpc` jaringan mendapatkan ENI mereka sendiri. Dengan melakukan ini, Anda dapat memantau lalu lintas yang masuk ke dan dari tugas individual menggunakan Amazon VPC Flow Logs. Pembaruan terbaru untuk Amazon VPC Flow Logs (v3), memperkaya log dengan metadata lalu lintas termasuk ID vpc, ID subnet, dan ID instans. Metadata ini dapat digunakan untuk membantu mempersempit penyelidikan. Untuk informasi selengkapnya, lihat Amazon VPC Flow Logs.

> **Note**
>
> Karena sifat kontainer sementara, log aliran mungkin tidak selalu menjadi cara yang efektif untuk menganalisis pola lalu lintas antara kontainer atau kontainer yang berbeda dan sumber daya jaringan lainnya.

Manajemen rahasia

Rahasia, seperti kunci API dan kredensil basis data, sering digunakan oleh aplikasi untuk mendapatkan akses sistem lain. Mereka sering terdiri dari nama pengguna dan kata sandi, sertifikat, atau kunci API. Akses ke rahasia ini harus dibatasi pada prinsipal IAM tertentu yang menggunakan IAM dan disuntikkan ke dalam wadah saat runtime.

Rahasia dapat disuntikkan dengan mulus ke dalam wadah dari AWS Secrets Manager dan Amazon EC2 Systems Manager Parameter Store. Rahasia-rahasia ini dapat direferensikan dalam tugas Anda sebagai salah satu dari berikut ini.

1. Mereka direferensikan sebagai variabel lingkungan yang menggunakan parameter definisi `secrets` kontainer.
2. Mereka direferensikan `secretOptions` seolah-olah platform logging Anda memerlukan otentikasi. Untuk informasi selengkapnya, lihat opsi konfigurasi logging.
3. Mereka direferensikan sebagai rahasia yang ditarik oleh gambar yang menggunakan parameter definisi `repositoryCredentials` wadah jika registri tempat penampung ditarik memerlukan otentikasi. Gunakan metode ini saat menarik gambar dari Galeri Publik Amazon ECR. Untuk informasi selengkapnya, lihat Autentikasi registri pribadi untuk tugas.

Rekomendasi

Kami menyarankan Anda melakukan hal berikut saat mengatur manajemen rahasia.

Gunakan AWS Secrets Manager atau Amazon EC2 Systems Manager Parameter Store untuk menyimpan materi rahasia

Anda harus menyimpan kunci API, kredensyal database, dan materi rahasia lainnya dengan aman di dalam AWS Secrets Manager atau sebagai parameter terenkripsi di Amazon EC2 Systems Manager Parameter Store. Layanan ini serupa karena keduanya merupakan penyimpanan nilai kunci terkelola yang digunakan AWS KMS untuk mengenkripsi data sensitif. AWS Secrets Manager Namun, juga mencakup kemampuan untuk secara otomatis memutar rahasia, menghasilkan rahasia acak, dan berbagi rahasia di seluruh AWS akun. Jika Anda menganggap fitur-fitur penting ini, gunakan AWS Secrets Manager sebaliknya gunakan parameter terenkripsi.

> **Note**
>
> Tugas yang mereferensikan rahasia dari AWS Secrets Manager atau Amazon EC2 Systems Manager Parameter Store memerlukan Peran Eksekusi Tugas dengan kebijakan yang memberikan akses Amazon ECS ke rahasia yang diinginkan dan, jika berlaku, kunci yang digunakan untuk mengenkripsi dan mendekripsi rahasia AWS KMS itu.

> **Important**
>
> Rahasia yang direferensikan dalam tugas tidak diputar secara otomatis. Jika rahasia Anda berubah, Anda harus memaksa penerapan baru atau meluncurkan tugas baru untuk mengambil nilai rahasia terbaru. Untuk informasi selengkapnya, lihat topik berikut:
>
> - AWS Secrets Manager: Menyuntikkan data sebagai variabel lingkungan
> - Amazon EC2 Systems Manager Parameter Store: Menyuntikkan data sebagai variabel lingkungan

Mengambil data dari bucket Amazon S3 terenkripsi

Karena nilai variabel lingkungan dapat bocor secara tidak sengaja di log dan terungkap saat dijalankan`docker inspect`, Anda harus menyimpan rahasia di bucket Amazon S3 terenkripsi dan menggunakan peran tugas untuk membatasi akses ke rahasia tersebut. Ketika Anda melakukan ini, aplikasi Anda harus ditulis untuk membaca rahasia dari ember Amazon S3. Untuk petunjuknya, lihat Menyetel perilaku enkripsi sisi server default untuk bucket Amazon S3.

Pasang rahasia ke volume menggunakan wadah sespan

Karena ada peningkatan risiko kebocoran data dengan variabel lingkungan, Anda harus menjalankan wadah sespan yang membaca rahasia Anda AWS Secrets Manager dan menuliskannya ke volume bersama. Kontainer ini dapat berjalan dan keluar sebelum penampung aplikasi dengan menggunakan pemesanan kontainer Amazon ECS. Ketika Anda melakukan ini, wadah aplikasi kemudian memasang volume tempat rahasia itu ditulis. Seperti metode bucket Amazon S3, aplikasi Anda harus ditulis untuk membaca rahasia dari volume bersama. Karena volume tercakup pada tugas, volume secara otomatis dihapus setelah tugas berhenti. Untuk contoh kontainer sespan, lihat proyeknya aws-secret-sidecar-injector.

> **Note**
>
> Di Amazon EC2, volume tempat rahasia ditulis dapat dienkripsi dengan kunci yang AWS KMS dikelola pelanggan. AWS Fargate Aktif, penyimpanan volume secara otomatis dienkripsi menggunakan kunci yang dikelola layanan.

Sumber daya tambahan

- Meneruskan rahasia ke wadah dalam tugas Amazon ECS
- Chamber adalah pembungkus untuk menyimpan rahasia di Amazon EC2 Systems Manager Parameter Store

Menggunakan kredensial keamanan sementara dengan operasi API

Jika Anda membuat permintaan HTTPS API langsung AWS, Anda dapat menandatangani permintaan tersebut dengan kredensyal keamanan sementara yang Anda dapatkan dari. AWS Security Token Service Untuk informasi selengkapnya, lihat Menandatangani permintaan AWS API di bagian Referensi Umum AWS.

Kepatuhan dan keamanan

Tanggung jawab kepatuhan Anda saat menggunakan Amazon ECS ditentukan oleh sensitivitas data Anda, dan tujuan kepatuhan perusahaan Anda, serta hukum dan peraturan yang berlaku.

AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- Panduan memulai cepat keamanan dan kepatuhan: Panduan penerapan ini membahas pertimbangan arsitektur dan memberikan langkah-langkah untuk menerapkan keamanan dan lingkungan dasar yang berfokus pada kepatuhan. AWS
- Arsitektur untuk Whitepaper Keamanan dan Kepatuhan HIPAA: Whitepaper ini menjelaskan bagaimana perusahaan dapat menggunakan untuk membuat aplikasi yang sesuai dengan HIPAA. AWS
- AWS Layanan dalam Lingkup oleh Program Kepatuhan: Daftar ini berisi AWS layanan dalam lingkup program kepatuhan tertentu. Untuk informasi selengkapnya, lihat Program AWS Kepatuhan.

Standar Keamanan Data Industri Kartu Pembayaran (PCI DSS)

Penting bagi Anda untuk memahami aliran lengkap data pemegang kartu (PJK) dalam lingkungan saat mengikuti PCI DSS. Aliran PJK menentukan penerapan PCI DSS, mendefinisikan batas dan komponen lingkungan data pemegang kartu (CDE), dan oleh karena itu ruang lingkup penilaian PCI DSS. Penentuan cakupan PCI DSS yang akurat adalah kunci untuk mendefinisikan postur keamanan dan akhirnya penilaian yang berhasil. Pelanggan harus memiliki prosedur untuk penentuan ruang lingkup yang menjamin kelengkapannya dan mendeteksi perubahan atau penyimpangan dari ruang lingkup.

Sifat sementara aplikasi kontainer memberikan kompleksitas tambahan saat mengaudit konfigurasi. Akibatnya, pelanggan perlu menjaga kesadaran akan semua parameter konfigurasi kontainer untuk memastikan persyaratan kepatuhan ditangani di semua fase siklus hidup kontainer.

Untuk informasi tambahan tentang mencapai kepatuhan PCI DSS di Amazon ECS, lihat whitepaper berikut.

- Arsitektur di Amazon ECS untuk kepatuhan PCI DSS
- Arsitektur untuk Pelingkupan dan Segmentasi PCI DSS AWS

HIPAA (Undang-Undang Portabilitas dan Akuntabilitas Asuransi Kesehatan AS)

Menggunakan Amazon ECS dengan beban kerja yang memproses informasi kesehatan yang dilindungi (PHI) tidak memerlukan konfigurasi tambahan. Amazon ECS bertindak sebagai layanan orkestrasi yang mengoordinasikan peluncuran kontainer di Amazon EC2. Itu tidak beroperasi dengan atau pada data dalam beban kerja yang diatur. Konsisten dengan peraturan HIPAA dan Adendum Asosiasi AWS Bisnis, PHI harus dienkripsi saat transit dan istirahat saat diakses oleh kontainer yang diluncurkan dengan Amazon ECS.

Berbagai mekanisme untuk mengenkripsi saat istirahat tersedia dengan setiap opsi AWS penyimpanan, seperti Amazon S3, Amazon EBS, dan. AWS KMS Anda dapat menggunakan jaringan overlay (seperti VNS3 atau Weave Net) untuk memastikan enkripsi lengkap PHI yang ditransfer antar kontainer atau untuk menyediakan lapisan enkripsi yang berlebihan. Pencatatan lengkap juga harus diaktifkan dan semua log kontainer harus diarahkan ke Amazon CloudWatch. Untuk mendesain AWS lingkungan Anda menggunakan praktik terbaik untuk keamanan infrastruktur, lihat Perlindungan Infrastruktur dalam Kerangka Kerja yang AWS Diarsiteksikan dengan Baik Pilar Keamanan.

AWS Security Hub

Gunakan AWS Security Hub untuk memantau penggunaan Amazon ECS yang berkaitan dengan praktik terbaik keamanan. Security Hub menggunakan kontrol untuk mengevaluasi konfigurasi sumber daya dan standar keamanan untuk membantu Anda mematuhi berbagai kerangka kerja kepatuhan. Untuk informasi selengkapnya tentang menggunakan Security Hub guna mengevaluasi sumber daya Amazon ECS, lihat kontrol Amazon ECS di AWS Security Hub Panduan Pengguna.

Pemantauan GuardDuty Runtime Amazon

Amazon GuardDuty adalah layanan deteksi ancaman yang membantu melindungi akun, wadah, beban kerja, dan data di AWS lingkungan Anda. Menggunakan model machine learning (ML), serta kemampuan deteksi anomali dan ancaman, GuardDuty terus memantau berbagai sumber log dan aktivitas runtime untuk mengidentifikasi dan memprioritaskan potensi risiko keamanan dan aktivitas berbahaya di lingkungan Anda.

Gunakan Runtime Monitoring GuardDuty untuk mengidentifikasi perilaku berbahaya atau tidak sah. Runtime Monitoring melindungi beban kerja yang berjalan di Fargate dan EC2 dengan terus memantau aktivitas AWS log dan jaringan untuk mengidentifikasi perilaku berbahaya atau tidak sah. Runtime Monitoring menggunakan agen GuardDuty keamanan ringan yang dikelola sepenuhnya yang menganalisis perilaku on-host, seperti akses file, eksekusi proses, dan koneksi jaringan. Ini mencakup masalah termasuk peningkatan hak istimewa, penggunaan kredensyal yang terbuka, atau komunikasi dengan alamat IP berbahaya, domain, dan keberadaan malware di instans Amazon EC2 dan beban kerja kontainer Anda. Untuk informasi selengkapnya, lihat GuardDuty Runtime Monitoring di Panduan GuardDuty Pengguna.

Rekomendasi

Anda harus melibatkan pemilik program kepatuhan dalam bisnis Anda lebih awal dan menggunakan model tanggung jawab AWS bersama untuk mengidentifikasi kepemilikan kontrol kepatuhan agar berhasil dengan program kepatuhan yang relevan.

Pencatatan log dan pemantauan

Pencatatan dan pemantauan merupakan aspek penting dalam menjaga keandalan, ketersediaan, dan kinerja Amazon ECS dan AWS solusi Anda. AWS menyediakan beberapa alat untuk memantau sumber daya Amazon ECS Anda dan menanggapi potensi insiden:

- CloudWatchAlarm Amazon

- CloudWatch Log Amazon
- CloudWatch Acara Amazon
- Log AWS CloudTrail

Anda dapat mengonfigurasi kontainer dalam tugas Anda untuk mengirim informasi log ke Amazon CloudWatch Logs. Jika Anda menggunakan jenis AWS Fargate peluncuran untuk tugas Anda, Anda dapat melihat log dari kontainer Anda. Jika Anda menggunakan jenis peluncuran Amazon EC2, Anda dapat melihat log yang berbeda dari kontainer Anda di satu lokasi yang nyaman. Ini juga mencegah log kontainer Anda mengambil ruang disk pada instance kontainer Anda.

Untuk informasi selengkapnya tentang CloudWatch Log Amazon, lihat Memantau Log dari Instans Amazon EC2 di Panduan Pengguna Amazon CloudWatch . Untuk instruksi pengiriman log kontainer dari tugas Anda ke Amazon CloudWatch Logs, lihat Menggunakan driver `awslogs` log.

Pencatatan kontainer dengan Fluent Bit

AWS menyediakan Fluent Bit gambar dengan plugin untuk Amazon CloudWatch Logs dan Amazon Data Firehose. Gambar ini menyediakan kemampuan untuk merutekan log ke tujuan Amazon CloudWatch dan Amazon Data Firehose (yang mencakup Amazon S3, Layanan Amazon, dan OpenSearch Amazon Redshift). Sebaiknya gunakan Fluent Bit sebagai router log Anda karena memiliki tingkat pemanfaatan sumber daya yang lebih rendah daripada Fluentd. Untuk informasi selengkapnya, lihat Amazon CloudWatch Logs for Fluent Bit dan Amazon Data Firehose untuk. Fluent Bit

Fluent BitGambar AWS untuk tersedia di:

- Amazon ECR di Galeri Publik Amazon ECR
- Repositori Amazon ECR (di sebagian besar Wilayah dengan ketersediaan tinggi)

Berikut ini menunjukkan sintaksis yang digunakan untuk Docker CLI.

```
docker pull public.ecr.aws/aws-observability/aws-for-fluent-bit:tag
```

Misalnya, Anda dapat menarik Fluent Bit gambar terbaru menggunakan AWS perintah CLI Docker ini:

```
docker pull public.ecr.aws/aws-observability/aws-for-fluent-bit:latest
```

Lihat juga posting blog berikut untuk informasi lebih lanjut tentang Fluent Bit dan fitur terkait:

- Fluent Bituntuk Amazon EKS di AWS Fargate
- Logging Kontainer Terpusat dengan Fluent Bit
- Membangun agregator solusi log yang dapat diskalakan dengan AWS Fargate, Fluentd, dan Amazon Data Firehose

Perutean log khusus - FireLens untuk Amazon ECS

Dengan FireLens Amazon ECS, Anda dapat menggunakan parameter definisi tugas untuk merutekan log ke AWS layanan atau tujuan Jaringan AWS Mitra (APN) untuk penyimpanan log dan analitik. FireLens bekerja dengan Fluentd dan. Fluent Bit Kami menyediakan AWS untuk Fluent Bit gambar. Atau, Anda dapat menggunakan Fluentd atau Fluent Bit gambar Anda sendiri.

Anda harus mempertimbangkan kondisi dan pertimbangan berikut saat menggunakan FireLens untuk Amazon ECS:

- FireLens untuk Amazon ECS didukung untuk tugas-tugas yang di-host baik di AWS Fargate maupun Amazon EC2.
- FireLens untuk Amazon ECS didukung dalam AWS CloudFormation template. Untuk informasi selengkapnya, lihat AWS::ECS::TaskDefinition FirelensConfiguration di AWS CloudFormation Panduan Pengguna.
- Untuk tugas yang menggunakan mode `bridge` jaringan, wadah dengan FireLens konfigurasi harus dimulai sebelum salah satu wadah aplikasi yang mengandalkannya dimulai. Untuk mengontrol urutan kontainer Anda mulai, gunakan kondisi ketergantungan dalam definisi tugas Anda. Untuk informasi selengkapnya, lihat Ketergantungan kontainer.

AWS Fargate keamanan

Kami menyarankan Anda mempertimbangkan praktik terbaik berikut saat Anda menggunakannya AWS Fargate. Untuk panduan tambahan, lihat Ikhtisar keamanan AWS Fargate.

Gunakan AWS KMS untuk mengenkripsi penyimpanan sementara

Anda harus memiliki penyimpanan sementara Anda dienkripsi oleh. AWS KMS Untuk tugas Amazon ECS yang di-host saat AWS Fargate menggunakan versi platform `1.4.0` atau yang lebih baru, setiap tugas menerima penyimpanan sementara 20 GiB. Anda dapat meningkatkan jumlah total penyimpanan sementara, hingga maksimum 200 GiB, dengan menentukan `ephemeralStorage` parameter dalam definisi tugas Anda. Untuk tugas-tugas seperti itu yang diluncurkan pada 28

Mei 2020 atau lebih baru, penyimpanan sementara dienkripsi dengan algoritma enkripsi AES-256 menggunakan kunci enkripsi yang dikelola oleh. AWS Fargate

Untuk informasi selengkapnya, lihat Menggunakan volume data dalam tugas.

Contoh: Meluncurkan tugas Amazon ECS pada AWS Fargate platform versi 1.4.0 dengan enkripsi penyimpanan singkat

Perintah berikut akan meluncurkan tugas Amazon ECS pada AWS Fargate platform versi 1.4. Karena tugas ini diluncurkan sebagai bagian dari cluster Amazon ECS, ia menggunakan penyimpanan singkat 20 GiB yang dienkripsi secara otomatis.

```
aws ecs run-task --cluster clustername \
  --task-definition taskdefinition:version \
  --count 1
  --launch-type "FARGATE" \
  --platform-version 1.4.0 \
  --network-configuration
 "awsvpcConfiguration={subnets=[subnetid],securityGroups=[securitygroupid]}" \
  --region region
```

Kemampuan SYS_PTRACE untuk penelusuran syscall kernel

Konfigurasi default kemampuan Linux yang ditambahkan atau dihapus dari container Anda disediakan oleh Docker. Untuk informasi selengkapnya tentang kemampuan yang tersedia, lihat hak istimewa Runtime dan kemampuan Linux dalam dokumentasi Docker run.

Tugas yang diluncurkan AWS Fargate hanya mendukung penambahan kemampuan `SYS_PTRACE` kernel.

Lihat video tutorial di bawah ini yang menunjukkan cara menggunakan fitur ini melalui proyek Sysdig Falco.

ContainersFromTheCouch - Memecahkan masalah AWS Fargate Tugas Anda menggunakan kemampuan SYS_PTRACE

Kode yang dibahas dalam video sebelumnya dapat ditemukan di GitHub sini.

Gunakan Amazon GuardDuty Runtime Monitoring

Amazon GuardDuty adalah layanan deteksi ancaman yang membantu melindungi akun, wadah, beban kerja, dan data di AWS lingkungan Anda. Menggunakan model machine learning (ML), serta

kemampuan deteksi anomali dan ancaman, GuardDuty terus memantau berbagai sumber log dan aktivitas runtime untuk mengidentifikasi dan memprioritaskan potensi risiko keamanan dan aktivitas berbahaya di lingkungan Anda.

Runtime Monitoring in GuardDuty melindungi beban kerja yang berjalan di Fargate dengan terus memantau aktivitas AWS log dan jaringan untuk mengidentifikasi perilaku berbahaya atau tidak sah. Runtime Monitoring menggunakan agen GuardDuty keamanan ringan yang dikelola sepenuhnya yang menganalisis perilaku on-host, seperti akses file, eksekusi proses, dan koneksi jaringan. Ini mencakup masalah termasuk peningkatan hak istimewa, penggunaan kredensyal yang terbuka, atau komunikasi dengan alamat IP berbahaya, domain, dan keberadaan malware di instans Amazon EC2 dan beban kerja kontainer Anda. Untuk informasi selengkapnya, lihat GuardDuty Runtime Monitoring di Panduan GuardDuty Pengguna.

AWS Fargate pertimbangan keamanan

Setiap tugas memiliki kapasitas infrastruktur khusus karena Fargate menjalankan setiap beban kerja pada lingkungan virtual yang terisolasi. Beban kerja yang berjalan di Fargate tidak berbagi antarmuka jaringan, penyimpanan sementara, CPU, atau memori dengan tugas lain. Anda dapat menjalankan beberapa kontainer dalam tugas termasuk wadah aplikasi dan kontainer sespan, atau hanya sespan. Sidecar adalah wadah yang berjalan bersama wadah aplikasi dalam tugas Amazon ECS. Sementara wadah aplikasi menjalankan kode aplikasi inti, proses yang berjalan di sidecars dapat menambah aplikasi. Sidecars membantu Anda memisahkan fungsi aplikasi ke dalam wadah khusus, sehingga memudahkan Anda memperbarui bagian aplikasi Anda.

Kontainer yang merupakan bagian dari tugas berbagi sumber daya yang sama untuk jenis peluncuran Fargate karena kontainer ini akan selalu berjalan di host yang sama dan berbagi sumber daya komputasi. Wadah ini juga berbagi penyimpanan singkat yang disediakan oleh Fargate. Kontainer Linux dalam ruang nama jaringan berbagi tugas, termasuk alamat IP dan port jaringan. Di dalam tugas, kontainer yang termasuk dalam tugas dapat saling berkomunikasi melalui localhost.

Lingkungan runtime di Fargate mencegah Anda menggunakan fitur pengontrol tertentu yang didukung pada instans EC2. Pertimbangkan hal berikut ketika Anda merancang beban kerja yang berjalan di Fargate:

- Tidak ada wadah atau akses istimewa - Fitur seperti wadah istimewa atau akses saat ini tidak tersedia di Fargate. Ini akan memengaruhi kasus penggunaan seperti menjalankan Docker di Docker.
- Akses terbatas ke kemampuan Linux - Lingkungan di mana kontainer berjalan di Fargate dikunci. Kemampuan Linux tambahan, seperti CAP_SYS_ADMIN dan CAP_NET_ADMIN,

dibatasi untuk mencegah eskalasi hak istimewa. Fargate mendukung penambahan kemampuan CAP_SYS_PTRACE Linux ke tugas-tugas untuk memungkinkan observabilitas dan alat keamanan yang digunakan dalam tugas untuk memantau aplikasi container.

- Tidak ada akses ke host yang mendasarinya - Baik pelanggan maupun AWS operator tidak dapat terhubung ke host yang menjalankan beban kerja pelanggan. Anda dapat menggunakan ECS exec untuk menjalankan perintah atau mendapatkan shell ke wadah yang berjalan di Fargate. Anda dapat menggunakan ECS exec untuk membantu mengumpulkan informasi diagnostik untuk debugging. Fargate juga mencegah kontainer mengakses sumber daya host yang mendasarinya, seperti sistem file, perangkat, jaringan, dan runtime kontainer.
- Jaringan - Anda dapat menggunakan grup keamanan dan ACL jaringan untuk mengontrol lalu lintas masuk dan keluar. Tugas Fargate menerima alamat IP dari subnet yang dikonfigurasi di VPC Anda.

Pertimbangan keamanan contoh kontainer EC2

Anda harus mempertimbangkan satu instance kontainer dan aksesnya dalam model ancaman Anda. Misalnya, satu tugas yang terpengaruh mungkin dapat memanfaatkan izin IAM dari tugas yang tidak terinfeksi pada instance yang sama.

Kami menyarankan Anda menggunakan yang berikut ini untuk membantu mencegah hal ini:

- Jangan gunakan hak administrator saat menjalankan tugas Anda.
- Tetapkan peran tugas dengan akses paling tidak istimewa ke tugas Anda.

  Agen kontainer secara otomatis membuat token dengan ID kredensyal unik yang digunakan untuk mengakses sumber daya Amazon ECS.
- Untuk mencegah container yang dijalankan oleh tugas yang menggunakan mode `awsvpc` jaringan mengakses informasi kredensi yang diberikan ke profil instans Amazon EC2, sambil tetap mengizinkan izin yang disediakan oleh peran tugas, tetapkan `ECS_AWSVPC_BLOCK_IMDS` variabel konfigurasi agen ke true dalam file konfigurasi agen dan restart agen.
- Gunakan Amazon GuardDuty Runtime Monitoring untuk mendeteksi ancaman untuk cluster dan container dalam lingkungan Anda AWS . Runtime Monitoring menggunakan agen GuardDuty keamanan yang menambahkan visibilitas runtime ke beban kerja Amazon ECS individual, misalnya, akses file, eksekusi proses, dan koneksi jaringan. Untuk informasi selengkapnya, lihat GuardDutyRuntime Monitoring di Panduan GuardDuty Pengguna.

Keamanan tugas dan kontainer

Anda harus mempertimbangkan gambar kontainer sebagai garis pertahanan pertama Anda terhadap serangan. Gambar yang tidak aman dan dibangun dengan buruk dapat memungkinkan penyerang melarikan diri dari batas-batas wadah dan mendapatkan akses ke host. Anda harus melakukan hal berikut untuk mengurangi risiko terjadinya hal ini.

Rekomendasi

Kami menyarankan Anda melakukan hal berikut saat menyiapkan tugas dan wadah Anda.

Buat minimal atau gunakan gambar distroless

Mulailah dengan menghapus semua binari asing dari gambar kontainer. Jika Anda menggunakan gambar asing dari Amazon ECR Public Gallery, periksa gambar untuk merujuk ke konten setiap layer container. Anda dapat menggunakan aplikasi seperti Dive untuk melakukan ini.

Atau, Anda dapat menggunakan gambar distroless yang hanya menyertakan aplikasi Anda dan dependensi runtime-nya. Mereka tidak berisi manajer paket atau shell. Gambar distroless meningkatkan "sinyal ke noise pemindai dan mengurangi beban menetapkan asal sesuai dengan apa yang Anda butuhkan." Untuk informasi lebih lanjut, lihat GitHub dokumentasi tentang distroless.

Docker memiliki mekanisme untuk membuat gambar dari gambar minimal yang dicadangkan yang dikenal sebagai scratch. Untuk informasi selengkapnya, lihat Membuat gambar induk sederhana menggunakan scratch dalam dokumentasi Docker. Dengan bahasa seperti Go, Anda dapat membuat biner tertaut statis dan mereferensikannya di Dockerfile Anda. Contoh berikut menunjukkan bagaimana Anda dapat mencapai ini.

```
############################
# STEP 1 build executable binary
############################
FROM golang:alpine AS builder
# Install git.
# Git is required for fetching the dependencies.
RUN apk update && apk add --no-cache git
WORKDIR $GOPATH/src/mypackage/myapp/
COPY . .
# Fetch dependencies.
# Using go get.
RUN go get -d -v
```

```
# Build the binary.
RUN go build -o /go/bin/hello
############################
# STEP 2 build a small image
############################
FROM scratch
# Copy our static executable.
COPY --from=builder /go/bin/hello /go/bin/hello
# Run the hello binary.
ENTRYPOINT ["/go/bin/hello"]
This creates a container image that consists of your application and nothing else,
 making it extremely secure.
```

Contoh sebelumnya juga merupakan contoh build multi-tahap. Jenis build ini menarik dari sudut pandang keamanan karena Anda dapat menggunakannya untuk meminimalkan ukuran gambar akhir yang didorong ke registri kontainer Anda. Gambar kontainer tanpa alat build dan binari asing lainnya meningkatkan postur keamanan Anda dengan mengurangi permukaan serangan gambar. Untuk informasi selengkapnya tentang build multi-tahap, lihat membuat build multi-tahap.

Pindai gambar Anda untuk kerentanan

Mirip dengan rekan mesin virtual mereka, gambar kontainer dapat berisi binari dan pustaka aplikasi dengan kerentanan atau mengembangkan kerentanan dari waktu ke waktu. Cara terbaik untuk melindungi terhadap eksploitasi adalah dengan memindai gambar Anda secara teratur dengan pemindai gambar.

Gambar yang disimpan di Amazon ECR dapat dipindai saat push atau on-demand (setiap 24 jam sekali). Pemindaian dasar Amazon ECR menggunakan Clair, solusi pemindaian gambar sumber terbuka. Pemindaian Amazon ECR yang ditingkatkan menggunakan Amazon Inspector. Setelah gambar dipindai, hasilnya dicatat ke aliran acara Amazon ECR di Amazon. EventBridge Anda juga dapat melihat hasil pemindaian dari dalam konsol Amazon ECR atau dengan memanggil DescribeImageScanFindingsAPI. Gambar dengan `CRITICAL` kerentanan `HIGH` atau harus dihapus atau dibangun kembali. Jika gambar yang telah digunakan mengembangkan kerentanan, itu harus diganti sesegera mungkin.

Docker Desktop Edge versi 2.3.6.0 atau yang lebih baru dapat memindai gambar lokal. Pemindaian ini didukung oleh Snyk, layanan keamanan aplikasi. Ketika kerentanan ditemukan, Snyk mengidentifikasi lapisan dan dependensi dengan kerentanan di Dockerfile. Ini juga merekomendasikan alternatif yang aman seperti menggunakan gambar dasar yang lebih ramping dengan kerentanan yang lebih sedikit atau meningkatkan paket tertentu ke versi yang lebih baru.

Dengan menggunakan pemindaian Docker, pengembang dapat menyelesaikan masalah keamanan potensial sebelum mendorong gambar mereka ke registri.

- Mengotomatiskan kepatuhan gambar menggunakan Amazon ECR dan AWS Security Hub menjelaskan cara memunculkan informasi kerentanan dari Amazon ECR AWS Security Hub dan mengotomatiskan remediasi dengan memblokir akses ke gambar yang rentan.

Hapus izin khusus dari gambar Anda

Hak akses menandai `setuid` dan `setgid` memungkinkan menjalankan executable dengan izin pemilik atau grup yang dapat dieksekusi. Hapus semua binari dengan hak akses ini dari gambar Anda karena binari ini dapat digunakan untuk meningkatkan hak istimewa. Pertimbangkan untuk menghapus semua shell dan utilitas seperti `nc` dan `curl` yang dapat digunakan untuk tujuan jahat. Anda dapat menemukan file dengan `setuid` dan hak `setgid` akses dengan menggunakan perintah berikut.

```
find / -perm /6000 -type f -exec ls -ld {} \;
```

Untuk menghapus izin khusus ini dari file-file ini, tambahkan arahan berikut ke gambar kontainer Anda.

```
RUN find / -xdev -perm /6000 -type f -exec chmod a-s {} \; || true
```

Buat satu set gambar yang dikuratori

Daripada mengizinkan pengembang untuk membuat gambar mereka sendiri, buat satu set gambar yang diperiksa untuk tumpukan aplikasi yang berbeda di organisasi Anda. Dengan demikian, pengembang dapat mengabaikan belajar bagaimana menulis Dockerfiles dan berkonsentrasi pada penulisan kode. Saat perubahan digabungkan ke dalam basis kode Anda, pipeline CI/CD dapat secara otomatis mengkompilasi aset dan kemudian menyimpannya dalam repositori artefak. Dan, terakhir, salin artefak ke gambar yang sesuai sebelum mendorongnya ke registri Docker seperti Amazon ECR. Setidaknya Anda harus membuat satu set gambar dasar yang dapat dibuat oleh pengembang Dockerfiles mereka sendiri. Anda harus menghindari menarik gambar dari Docker Hub. Anda tidak selalu tahu apa yang ada dalam gambar dan sekitar seperlima dari 1000 gambar teratas memiliki kerentanan. Daftar gambar-gambar tersebut dan kerentanannya dapat ditemukan di https://vulnerablecontainers.org/.

Pindai paket aplikasi dan pustaka untuk mencari kerentanan

Penggunaan pustaka open source sekarang umum. Seperti halnya sistem operasi dan paket OS, pustaka ini dapat memiliki kerentanan. Sebagai bagian dari siklus hidup pengembangan, pustaka ini harus dipindai dan diperbarui saat kerentanan kritis ditemukan.

Docker Desktop melakukan pemindaian lokal menggunakan Snyk. Ini juga dapat digunakan untuk menemukan kerentanan dan potensi masalah lisensi di perpustakaan open source. Ini dapat diintegrasikan langsung ke dalam alur kerja pengembang yang memberi Anda kemampuan untuk mengurangi risiko yang ditimbulkan oleh pustaka open source. Untuk informasi selengkapnya, lihat topik berikut:

- Open Source Application Security Tools mencakup daftar alat untuk mendeteksi kerentanan dalam aplikasi.

Lakukan analisis kode statis

Anda harus melakukan analisis kode statis sebelum membuat gambar kontainer. Ini dilakukan terhadap kode sumber Anda dan digunakan untuk mengidentifikasi kesalahan pengkodean dan kode yang dapat dieksploitasi oleh aktor jahat, seperti suntikan kesalahan. SonarQubeadalah pilihan populer untuk pengujian keamanan aplikasi statis (SAST), dengan dukungan untuk berbagai bahasa pemrograman yang berbeda.

Jalankan kontainer sebagai pengguna non-root

Anda harus menjalankan container sebagai pengguna non-root. Secara default, container berjalan sebagai `root` pengguna kecuali `USER` direktif disertakan dalam Dockerfile Anda. Kemampuan default Linux yang ditetapkan oleh Docker membatasi tindakan yang dapat dijalankan sebagai`root`, tetapi hanya sedikit. Misalnya, kontainer yang berjalan seperti `root` masih tidak diizinkan untuk mengakses perangkat.

Sebagai bagian dari pipeline CI/CD Anda, Anda harus lint Dockerfiles untuk mencari `USER` arahan dan gagal membangun jika tidak ada. Untuk informasi selengkapnya, lihat topik berikut:

- DockerFile-Lint adalah alat sumber terbuka RedHat yang dapat digunakan untuk memeriksa apakah file tersebut sesuai dengan praktik terbaik.
- Hadolint adalah alat lain untuk membangun gambar Docker yang sesuai dengan praktik terbaik.

Gunakan sistem file root read-only

Anda harus menggunakan sistem file root read-only. Sistem file root container dapat ditulis secara default. Saat Anda mengonfigurasi wadah dengan sistem file root R0 (hanya-baca), itu memaksa Anda untuk secara eksplisit menentukan di mana data dapat disimpan. Ini mengurangi permukaan serangan Anda karena sistem file container tidak dapat ditulis kecuali izin diberikan secara khusus.

> **Note**
>
> Memiliki sistem file root read-only dapat menyebabkan masalah dengan paket OS tertentu yang diharapkan dapat menulis ke sistem file. Jika Anda berencana untuk menggunakan sistem file root read-only, uji secara menyeluruh sebelumnya.

Konfigurasikan tugas dengan batas CPU dan Memori (Amazon EC2)

Anda harus mengkonfigurasi tugas dengan CPU dan batas memori untuk meminimalkan risiko berikut. Batas sumber daya tugas menetapkan batas atas untuk jumlah CPU dan memori yang dapat dicadangkan oleh semua wadah dalam tugas. Jika tidak ada batasan yang ditetapkan, tugas memiliki akses ke CPU dan memori host. Hal ini dapat menyebabkan masalah di mana tugas yang diterapkan pada host bersama dapat membuat tugas-tugas lain dari sumber daya sistem kelaparan.

> **Note**
>
> Amazon ECS pada AWS Fargate tugas mengharuskan Anda menentukan batas CPU dan memori karena menggunakan nilai ini untuk tujuan penagihan. Satu tugas memonopoli semua sumber daya sistem tidak menjadi masalah bagi Amazon ECS Fargate karena setiap tugas dijalankan pada instans khusus sendiri. Jika Anda tidak menentukan batas memori, Amazon ECS mengalokasikan minimal 4MB untuk setiap kontainer. Demikian pula, jika tidak ada batas CPU yang ditetapkan untuk tugas tersebut, agen penampung Amazon ECS menetapkannya minimal 2 CPU.

Gunakan tag yang tidak dapat diubah dengan Amazon ECR

Dengan Amazon ECR, Anda dapat dan harus menggunakan konfigurasi gambar dengan tag yang tidak dapat diubah. Ini mencegah mendorong versi gambar yang diubah atau diperbarui ke repositori gambar Anda dengan tag yang identik. Ini melindungi terhadap penyerang yang mendorong versi

gambar yang dikompromikan di atas gambar Anda dengan tag yang sama. Dengan menggunakan tag yang tidak dapat diubah, Anda secara efektif memaksa diri Anda untuk mendorong gambar baru dengan tag berbeda untuk setiap perubahan.

Hindari menjalankan kontainer sebagai hak istimewa (Amazon EC2)

Anda harus menghindari menjalankan kontainer sebagai hak istimewa. Untuk latar belakang, kontainer dijalankan seperti `privileged` yang dijalankan dengan hak istimewa yang diperluas pada host. Ini berarti wadah mewarisi semua kemampuan Linux yang ditetapkan `root` pada host. Penggunaannya harus sangat dibatasi atau dilarang. Kami menyarankan untuk menyetel variabel lingkungan agen penampung Amazon ECS `ECS_DISABLE_PRIVILEGED` `true` untuk mencegah kontainer berjalan seperti `privileged` pada host tertentu jika `privileged` tidak diperlukan. Atau Anda dapat menggunakan AWS Lambda untuk memindai definisi tugas Anda untuk penggunaan `privileged` parameter.

> **Note**
>
> Menjalankan wadah seperti yang `privileged` tidak didukung di Amazon ECS aktif. AWS Fargate

Hapus kemampuan Linux yang tidak perlu dari wadah

Berikut ini adalah daftar kemampuan Linux default yang ditetapkan untuk kontainer Docker. Untuk informasi selengkapnya tentang setiap kemampuan, lihat Ikhtisar Kemampuan Linux.

```
CAP_CHOWN, CAP_DAC_OVERRIDE, CAP_FOWNER, CAP_FSETID, CAP_KILL,
CAP_SETGID, CAP_SETUID, CAP_SETPCAP, CAP_NET_BIND_SERVICE,
CAP_NET_RAW, CAP_SYS_CHROOT, CAP_MKNOD, CAP_AUDIT_WRITE,
CAP_SETFCAP
```

Jika sebuah container tidak memerlukan semua kemampuan kernel Docker yang tercantum di atas, pertimbangkan untuk menjatuhkannya dari container. Untuk informasi selengkapnya tentang setiap kemampuan kernel Docker, lihat KernalCapabilities. Anda dapat mengetahui kemampuan mana yang digunakan dengan melakukan hal berikut:

- Instal paket OS libcap-ng dan jalankan `pscap` utilitas untuk membuat daftar kemampuan yang digunakan setiap proses.

- Anda juga dapat menggunakan capsh untuk menguraikan kemampuan mana yang digunakan suatu proses.
- Lihat Kemampuan Linux 101 untuk informasi lebih lanjut.

Gunakan kunci terkelola pelanggan (CMK) untuk mengenkripsi gambar yang didorong ke Amazon ECR

Anda harus menggunakan kunci terkelola pelanggan (CMK) untuk mengenkripsi gambar yang didorong ke Amazon ECR. Gambar yang didorong ke Amazon ECR secara otomatis dienkripsi saat istirahat dengan kunci terkelola AWS Key Management Service (AWS KMS). Jika Anda lebih suka menggunakan kunci Anda sendiri, Amazon ECR sekarang mendukung AWS KMS enkripsi dengan kunci terkelola pelanggan (CMK). Sebelum mengaktifkan enkripsi sisi server dengan CMK, tinjau Pertimbangan yang tercantum dalam dokumentasi tentang enkripsi saat istirahat.

Keamanan runtime

Keamanan runtime memberikan perlindungan aktif untuk kontainer Anda saat sedang berjalan. Idenya adalah untuk mendeteksi dan mencegah aktivitas berbahaya terjadi pada wadah Anda. Konfigurasi keamanan runtime berbeda antara wadah Windows dan Linux.

Untuk mengamankan kontainer Microsoft Windows, lihat Kontainer Windows yang aman.

Untuk mengamankan wadah Linux, Anda dapat menambahkan atau menjatuhkan kemampuan kernel Linux menggunakan `linuxParameters` dan menerapkan SELinux`labels`, atau AppArmor profil menggunakan`dockerSecurityOptions`, keduanya per kontainer dalam definisi tugas. SELinux atau AppArmor harus dikonfigurasi pada instance container sebelum dapat digunakan. SELinux dan tidak AppArmor tersedia di. AWS Fargate Untuk informasi selengkapnya, lihat `dockerSecurityOptions`di Referensi API Amazon Elastic Container Service, dan konfigurasi Keamanan di referensi run Docker.

AppArmor adalah modul keamanan Linux yang membatasi kemampuan wadah termasuk mengakses bagian-bagian dari sistem file. Itu dapat dijalankan dalam salah satu `complain` mode `enforcement` atau. Karena membangun AppArmor profil dapat menjadi tantangan, kami sarankan Anda menggunakan alat seperti bane. Untuk informasi selengkapnya AppArmor, lihat AppArmorhalaman resmi.

> **Important**
>
> AppArmor hanya tersedia untuk distribusi Ubuntu dan Debian Linux.

Rekomendasi

Kami menyarankan Anda mengambil tindakan berikut saat mengatur keamanan runtime Anda.

Gunakan Amazon GuardDuty Runtime Monitoring

Amazon GuardDuty adalah layanan deteksi ancaman yang membantu melindungi akun, wadah, beban kerja, dan data di AWS lingkungan Anda. Menggunakan model machine learning (ML), serta kemampuan deteksi anomali dan ancaman, GuardDuty terus memantau berbagai sumber log dan aktivitas runtime untuk mengidentifikasi dan memprioritaskan potensi risiko keamanan dan aktivitas berbahaya di lingkungan Anda.

Runtime Monitoring in GuardDuty melindungi beban kerja yang berjalan di Fargate dengan terus memantau aktivitas AWS log dan jaringan untuk mengidentifikasi perilaku berbahaya atau tidak sah. Runtime Monitoring menggunakan agen GuardDuty keamanan ringan yang dikelola sepenuhnya yang menganalisis perilaku on-host, seperti akses file, eksekusi proses, dan koneksi jaringan. Ini mencakup masalah termasuk peningkatan hak istimewa, penggunaan kredensyal yang terbuka, atau komunikasi dengan alamat IP berbahaya, domain, dan keberadaan malware di instans Amazon EC2 dan beban kerja kontainer Anda. Untuk informasi selengkapnya, lihat GuardDutyRuntime Monitoring di Panduan GuardDuty Pengguna.

Gunakan solusi pihak ketiga untuk pertahanan runtime

Gunakan solusi pihak ketiga untuk pertahanan runtime. Jika Anda terbiasa dengan cara kerja keamanan Linux, buat dan kelola AppArmor profil. Keduanya adalah proyek open source.
Jika tidak, pertimbangkan untuk menggunakan layanan pihak ketiga yang berbeda. Sebagian besar menggunakan pembelajaran mesin untuk memblokir atau memperingatkan aktivitas yang mencurigakan. Untuk daftar solusi pihak ketiga yang tersedia, lihat AWS Marketplace untuk Kontainer.

Praktik terbaik AMI

Amazon ECS menyediakan AMI yang dioptimalkan Amazon ECS yang telah dikonfigurasi sebelumnya dengan persyaratan dan rekomendasi untuk menjalankan beban kerja penampung Anda. Sebaiknya gunakan Amazon ECS yang dioptimalkan Amazon Linux 2023 AMI untuk instans

Amazon EC2 Anda kecuali aplikasi Anda memerlukan instans berbasis GPU Amazon EC2, sistem operasi tertentu, atau versi Docker yang belum tersedia di AMI tersebut. Untuk informasi tentang instans Amazon Linux 2 dan Amazon Linux 2023, lihat Membandingkan Amazon Linux 2 dan Amazon Linux 2023 di Panduan Pengguna Amazon Linux 2023.

Meskipun Anda dapat membuat AMI instans Amazon EC2 Anda sendiri yang memenuhi spesifikasi dasar yang diperlukan untuk menjalankan beban kerja kontainer Anda di Amazon ECS, AMI yang dioptimalkan Amazon ECS telah dikonfigurasi sebelumnya dan diuji di Amazon ECS oleh para insinyur. AWS Ini adalah cara paling sederhana bagi Anda untuk memulai dan menjalankan kontainer Anda AWS dengan cepat.

AWS Mitra

Anda dapat menggunakan salah satu produk AWS Mitra berikut untuk menambahkan keamanan dan fitur tambahan ke beban kerja Amazon ECS Anda. Untuk informasi selengkapnya, lihat Amazon ECS Partners.

Keamanan Aqua

Anda dapat menggunakan Aqua Security untuk mengamankan aplikasi cloud-native Anda dari pengembangan hingga produksi. Aqua Cloud Native Security Platform terintegrasi dengan sumber daya cloud-native dan alat orkestrasi Anda untuk memberikan keamanan yang transparan dan otomatis. Ini dapat mencegah aktivitas dan serangan yang mencurigakan secara real time, dan membantu menegakkan kebijakan dan menyederhanakan kepatuhan terhadap peraturan.

Jaringan Palo Alto

Palo Alto Networks memberikan keamanan dan perlindungan untuk host, kontainer, dan infrastruktur tanpa server Anda di cloud dan di seluruh siklus pengembangan dan perangkat lunak.

Twistlock dipasok oleh Palo Alto Networks dan dapat diintegrasikan dengan Amazon ECS. FireLens Dengan itu, Anda memiliki akses ke log keamanan kesetiaan tinggi dan insiden yang digabungkan dengan mulus ke dalam beberapa layanan. AWS Ini termasuk Amazon CloudWatch, Amazon Athena, dan Amazon Kinesis. Twistlock mengamankan beban kerja yang diterapkan pada layanan kontainer. AWS

Sysdig

Anda dapat menggunakan Sysdig untuk menjalankan beban kerja cloud-native yang aman dan sesuai dalam skenario produksi. DevOps Platform Aman Sysdig memiliki fitur keamanan dan

kepatuhan yang tertanam untuk melindungi beban kerja cloud-native Anda, dan juga menawarkan skalabilitas, kinerja, dan penyesuaian tingkat perusahaan.

Ambil metadata Amazon ECS

Amazon ECS menyediakan berbagai metadata untuk konfigurasi Anda.

Anda dapat menggunakan metadata untuk mendapatkan informasi tugas termasuk:

- Atribut tingkat tugas yang memberikan informasi tentang di mana tugas berjalan.
- Atribut tingkat kontainer yang menyediakan ID Docker, nama, dan detail gambar. Ini memberikan visibilitas ke dalam wadah.
- Pengaturan jaringan seperti alamat IP, subnet, dan mode jaringan. Ini membantu dengan konfigurasi jaringan dan pemecahan masalah.
- Status tugas dan kesehatan

Anda dapat melihat metadata dengan salah satu metode berikut:

- File metadata kontainer

  Dimulai dengan versi 1.15.0 dari agen penampung Amazon ECS, berbagai metadata kontainer tersedia di dalam kontainer Anda atau instans penampung host. Dengan mengaktifkan fitur ini, Anda dapat melakukan kueri informasi tentang tugas, kontainer, dan instans kontainer dari dalam kontainer atau instans kontainer host. File metadata dibuat pada instance host dan dipasang di wadah sebagai volume Docker dan oleh karena itu tidak tersedia saat tugas di-host di Fargate. AWS
- Titik akhir metadata tugas

  Agen kontainer Amazon ECS menyuntikkan variabel lingkungan ke dalam setiap kontainer, yang disebut sebagai titik akhir metadata tugas yang menyediakan berbagai metadata tugas dan statistik Docker ke wadah.
- Introspeksi wadah

  Agen penampung Amazon ECS menyediakan operasi API untuk mengumpulkan detail tentang instance container tempat agen dijalankan dan tugas terkait yang berjalan pada instance tersebut.

File metadata wadah Amazon ECS

Dimulai dengan versi 1.15.0 dari agen penampung Amazon ECS, berbagai metadata kontainer tersedia di dalam kontainer Anda atau instans penampung host. Dengan mengaktifkan fitur ini, Anda dapat melakukan kueri informasi tentang tugas, kontainer, dan instans kontainer dari dalam kontainer atau instans kontainer host. File metadata dibuat pada instance host dan dipasang di wadah sebagai volume Docker dan oleh karena itu tidak tersedia saat tugas di-host di Fargate. AWS

File metadata kontainer dibersihkan pada instans host ketika kontainer dibersihkan. Anda dapat menyesuaikan kapan hal ini terjadi dengan variabel agen kontainer `ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION`. Untuk informasi selengkapnya, lihat Pembersihan tugas dan citra otomatis.

Topik

Lokasi file metadata kontainer

Secara default, file metadata kontainer ditulis ke jalur host dan kontainer berikut.

- Untuk instance Linux:
  - Jalur host: `/var/lib/ecs/data/metadata/`*`cluster_name`*`/`*`task_id`*`/`*`container_name`*`/ecs-container-metadata.json`

    > **Note**
    >
    > Jalur host Linux mengasumsikan bahwa jalur pemasangan direktori data default (`/var/lib/ecs/data`) digunakan saat agen dimulai. Jika Anda tidak menggunakan AMI yang dioptimalkan Amazon ECS (atau `ecs-init` paket untuk memulai dan memelihara agen penampung), pastikan untuk menyetel variabel konfigurasi `ECS_HOST_DATA_DIR` agen ke jalur host tempat file status agen penampung berada. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.

  - Jalur kontainer: `/opt/ecs/metadata/`*`random_ID`*`/ecs-container-metadata.json`
- Untuk contoh Windows:

- Jalur host: `C:\ProgramData\Amazon\ECS\data\metadata\task_id\container_name\ecs-container-metadata.json`
- Jalur kontainer: `C:\ProgramData\Amazon\ECS\metadata\random_ID\ecs-container-metadata.json`

Namun, untuk memudahkan akses, lokasi file metadata kontainer diatur ke variabel lingkungan `ECS_CONTAINER_METADATA_FILE` di dalam kontainer. Anda dapat membaca isi file dari dalam kontainer dengan perintah berikut:

- Untuk instance Linux:

```
cat $ECS_CONTAINER_METADATA_FILE
```

- Untuk instance Windows (PowerShell):

```
Get-Content -path $env:ECS_CONTAINER_METADATA_FILE
```

Mengaktifkan metadata kontainer Amazon ECS

Anda dapat mengaktifkan metadata kontainer di tingkat instance kontainer dengan menyetel variabel agen `ECS_ENABLE_CONTAINER_METADATA` kontainer ke. `true` Anda dapat mengatur variabel ini di file konfigurasi `/etc/ecs/ecs.config` dan mulai ulang agen. Anda juga dapat mengaturnya sebagai variabel lingkungan Docker pada saat waktu aktif ketika kontainer agen dimulai. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.

Jika `ECS_ENABLE_CONTAINER_METADATA` diatur ke `true` ketika agen dimulai, file metadata dibuat untuk setiap kontainer yang dibuat sejak saat itu. Agen penampung Amazon ECS tidak dapat membuat file metadata untuk kontainer yang dibuat sebelum variabel agen `ECS_ENABLE_CONTAINER_METADATA` kontainer disetel ke. `true` Untuk memastikan bahwa semua kontainer menerima file metadata, Anda harus mengatur variabel agen ini pada peluncuran instans kontainer. Berikut ini adalah contoh skrip data pengguna yang akan mengatur variabel ini serta mendaftarkan instans kontainer dengan klaster Anda.

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=your_cluster_name
ECS_ENABLE_CONTAINER_METADATA=true
```

```
EOF
```

Format file metadata wadah Amazon ECS

Informasi berikut disimpan dalam file JSON metadata kontainer.

`Cluster`

Nama klaster tempat tugas kontainer sedang berjalan.

`ContainerInstanceARN`

Nama lrngkap Amazon Resource Name (ARN) dari instans kontainer host.

`TaskARN`

Nama lengkap Amazon Resource Name (ARN) dari tugas milik kontainer.

`TaskDefinitionFamily`

Nama famili ketentuan tugas yag digunakan kontainer.

`TaskDefinitionRevision`

Revisi ketentuan tugas yang digunakan kontainer.

`ContainerID`

ID kontainer Docker (dan bukan ID penampung Amazon ECS) untuk penampung.

`ContainerName`

Nama kontainer dari definisi tugas Amazon ECS untuk penampung.

`DockerContainerName`

Nama kontainer yang digunakan daemon Docker untuk kontainer (misalnya, nama yang muncul di output perintah docker ps).

`ImageID`

SHA digest untuk citra Docker digunakan untuk memulai kontainer.

`ImageName`

Nama citra dan tanda untuk citra Docker yang digunakan untuk memulai kontainer.

`PortMappings`

Setiap pemetaan port yang terkait dengan kontainer.

`ContainerPort`

Port pada kontainer yang diekspos.

`HostPort`

Port pada instans kontainer host yang diekspos.

`BindIp`

Alamat IP yang mengikat yang ditetapkan untuk kontainer oleh Docker. Alamat IP ini hanya diterapkan dengan mode jaringan `bridge`, dan hanya dapat diakses dari instans kontainer.

`Protocol`

Protokol jaringan yang digunakan untuk pemetaan port.

`Networks`

Mode jaringan dan alamat IP untuk kontainer.

`NetworkMode`

Mode jaringan untuk tugas milik kontainer.

`IPv4Addresses`

Alamat IP yang terkait dengan kontainer.

> **Important**
>
> Jika tugas Anda menggunakan mode jaringan `awsvpc`, alamat IP kontainer tidak akan dikembalikan. Dalam hal ini, Anda dapat mengambil alamat IP dengan membaca file /etc/host dengan perintah berikut:
>
> ```
> tail -1 /etc/hosts | awk '{print $1}'
> ```

`MetadataFileStatus`

Status file metadata. Bila statusnya `READY`, file metadata baru dan lengkap. Jika file belum siap (misalnya, saat tugas dimulai), versi format file yang terpotong tersedia. Untuk menghindari kemungkinan kondisi pacu saat kontainer dimulai, namun metadata belum ditulis, Anda dapat mengurai file metadata dan menunggu parameter ini untuk diatur ke `READY` sebelum tergantung

pada metadata. Hal ini biasanya tersedia dalam waktu kurang dari 1 detik sejak saat kontainer dimulai.

`AvailabilityZone`

Availability Zone instans kontainer host berada di.

`HostPrivateIPv4Address`

Alamat IP privat untuk tugas milik kontainer.

`HostPublicIPv4Address`

Alamat IP publik untuk tugas milik kontainer.

Example File metadata wadah Amazon ECS () **`READY`**

Contoh berikut menunjukkan file metadata kontainer di status `READY`.

```
{
    "Cluster": "default",
    "ContainerInstanceARN": "arn:aws:ecs:us-west-2:012345678910:container-instance/
default/1f73d099-b914-411c-a9ff-81633b7741dd",
    "TaskARN": "arn:aws:ecs:us-west-2:012345678910:task/default/2b88376d-
aba3-4950-9ddf-bcb0f388a40c",
    "TaskDefinitionFamily": "console-sample-app-static",
    "TaskDefinitionRevision": "1",
    "ContainerID": "aec2557997f4eed9b280c2efd7afccdcedfda4ac399f7480cae870cfc7e163fd",
    "ContainerName": "simple-app",
    "CreatedAt": "2023-10-08T20:09:11.44527186Z",
    "StartedAt": "2023-10-08T20:09:11.44527186Z",
    "DockerContainerName": "/ecs-console-sample-app-static-1-simple-app-
e4e8e495e8baa5de1a00",
    "ImageID":
 "sha256:2ae34abc2ed0a22e280d17e13f9c01aaf725688b09b7a1525d1a2750e2c0d1de",
    "ImageName": "httpd:2.4",
    "PortMappings": [
        {
            "ContainerPort": 80,
            "HostPort": 80,
            "BindIp": "0.0.0.0",
            "Protocol": "tcp"
        }
    ],
```

```
    "Networks": [
        {
            "NetworkMode": "bridge",
            "IPv4Addresses": ["192.0.2.0"]
        }
    ],
    "MetadataFileStatus": "READY",
    "AvailabilityZone": "us-east-1b",
    "HostPrivateIPv4Address": "192.0.2.0",
    "HostPublicIPv4Address": "203.0.113.0"
}
```

Example File metadata kontainer Amazon ECS tidak lengkap (belum) **`READY`**

Contoh berikut menunjukkan file metadata kontainer yang belum mencapai status `READY`. Informasi dalam file terbatas untuk beberapa parameter yang diketahui dari ketentuan tugas. File metadata kontainer harus siap dalam 1 detik setelah kontainer dimulai.

```
{
    "Cluster": "default",
    "ContainerInstanceARN": "arn:aws:ecs:us-west-2:012345678910:container-instance/
default/1f73d099-b914-411c-a9ff-81633b7741dd",
    "TaskARN": "arn:aws:ecs:us-west-2:012345678910:task/default/
d90675f8-1a98-444b-805b-3d9cabb6fcd4",
    "ContainerName": "metadata"
}
```

Metadata tugas tersedia untuk tugas Amazon ECS di EC2

Agen penampung Amazon ECS menyediakan metode untuk mengambil berbagai metadata tugas dan statistik Docker. Hal Ini disebut sebagai titik akhir metadata tugas. Versi berikut tersedia:

- Titik akhir metadata tugas versi 4 — Menyediakan berbagai metadata dan statistik Docker ke kontainer. Juga dapat menyediakan data tingkat jaringan. Tersedia untuk tugas Amazon ECS yang diluncurkan di instans Amazon EC2 Linux yang menjalankan setidaknya versi agen `1.39.0` penampung Amazon ECS. Untuk instans Windows Amazon EC2 yang menggunakan mode `awsvpc` jaringan, agen penampung Amazon ECS harus setidaknya versi. `1.54.0` Untuk informasi selengkapnya, lihat Titik akhir metadata tugas Amazon ECS versi 4.
- Titik akhir metadata tugas versi 3 — Menyediakan berbagai metadata dan statistik Docker ke kontainer. Tersedia untuk tugas Amazon ECS yang diluncurkan di instans Amazon EC2 Linux yang

menjalankan setidaknya versi agen `1.21.0` penampung Amazon ECS. Untuk instans Windows Amazon EC2 yang menggunakan mode `awsvpc` jaringan, agen penampung Amazon ECS harus setidaknya versi. `1.54.0` Untuk informasi selengkapnya, lihat Titik akhir metadata tugas Amazon ECS versi 3.

- Titik akhir metadata tugas versi 2 — Tersedia untuk tugas Amazon ECS yang diluncurkan di instans Amazon EC2 Linux yang menjalankan setidaknya versi agen penampung Amazon ECS. `1.17.0` Untuk instans Windows Amazon EC2 yang menggunakan mode `awsvpc` jaringan, agen penampung Amazon ECS harus setidaknya versi. `1.54.0` Untuk informasi selengkapnya, lihat Titik akhir metadata tugas Amazon ECS versi 2.

Jika tugas Amazon ECS di-host di Amazon EC2, Anda juga dapat mengakses metadata host tugas menggunakan titik akhir Layanan Metadata Instans (IMDS). Perintah berikut, ketika dijalankan dari dalam instance hosting tugas, mencantumkan ID dari instance host.

```
curl http://169.254.169.254/latest/meta-data/instance-id
```

Informasi yang dapat Anda peroleh dari titik akhir dibagi ke dalam kategori seperti *instance-id*. Untuk informasi selengkapnya tentang berbagai kategori metadata instans host yang dapat Anda peroleh menggunakan titik akhir, lihat Kategori metadata instans.

Titik akhir metadata tugas Amazon ECS versi 4

Agen kontainer Amazon ECS menyuntikkan variabel lingkungan ke dalam setiap kontainer, yang disebut sebagai titik akhir metadata tugas yang menyediakan berbagai metadata tugas dan statistik Docker ke wadah.

Metadata tugas dan statistik tingkat jaringan dikirim ke CloudWatch Wawasan Kontainer dan dapat dilihat di. AWS Management Console Untuk informasi selengkapnya, lihat Pantau kontainer Amazon ECS menggunakan Wawasan Kontainer.

> **Note**
>
> Amazon ECS menyediakan versi sebelumnya dari titik akhir metadata tugas. Untuk menghindari kebutuhan membuat versi titik akhir metadata tugas di masa mendatang, metadata tambahan dapat ditambahkan ke output versi 4. Kami tidak akan menghapus metadata yang ada atau mengubah nama bidang metadata.

Variabel lingkungan disuntikkan secara default ke dalam wadah tugas Amazon ECS yang diluncurkan pada instans Amazon EC2 Linux yang menjalankan setidaknya versi agen `1.39.0` penampung Amazon ECS. Untuk instans Windows Amazon EC2 yang menggunakan mode `awsvpc` jaringan, agen penampung Amazon ECS harus setidaknya versi. `1.54.0` Untuk informasi selengkapnya, lihat Manajemen instance kontainer Linux.

> **Note**
>
> Anda dapat menambahkan dukungan untuk fitur ini di instans Amazon EC2 menggunakan versi lama agen penampung Amazon ECS dengan memperbarui agen ke versi terbaru. Untuk informasi selengkapnya, lihat Memperbarui agen kontainer Amazon ECS.

Jalur titik akhir metadata tugas versi 4

Jalur titik akhir metadata tugas berikut tersedia untuk kontainer.

`${ECS_CONTAINER_METADATA_URI_V4}`

Jalur ini mengembalikan metadata untuk kontainer.

`${ECS_CONTAINER_METADATA_URI_V4}/task`

Jalur ini mengembalikan metadata untuk tugas, termasuk daftar ID kontainer dan nama semua kontainer yang terkait dengan tugas. Untuk informasi selengkapnya tentang respons titik akhir ini, lihat Metadata tugas Amazon ECS V4 Respons JSON.

`${ECS_CONTAINER_METADATA_URI_V4}/taskWithTags`

Jalur ini mengembalikan metadata untuk tugas termasuk dalam titik akhir `/task` selain tugas dan tanda instans kontainer yang dapat diambil menggunakan API `ListTagsForResource`. Setiap kesalahan yang diterima saat mengambil metadata tanda akan disertakan dalam kolom `Errors` sebagai responsnya.

> **Note**
>
> `Errors`Bidang ini hanya dalam respons untuk tugas yang dihosting di instans Amazon EC2 Linux yang menjalankan setidaknya versi agen `1.50.0` penampung. Untuk instans Windows Amazon EC2 yang menggunakan mode `awsvpc` jaringan, agen penampung Amazon ECS harus setidaknya versi `1.54.0`

Titik akhir ini membutuhkan `ecs.ListTagsForResource` izin.

`${ECS_CONTAINER_METADATA_URI_V4}/stats`

Jalur ini mengembalikan statistik Docker untuk kontainer tertentu. Untuk informasi selengkapnya tentang masing-masing statistik yang dikembalikan, lihat ContainerStatsdi dokumentasi Docker API.

Untuk tugas Amazon ECS yang menggunakan mode `bridge` jaringan `awsvpc` atau yang dihosting di instans Amazon EC2 Linux yang menjalankan setidaknya `1.43.0` versi agen penampung, akan ada statistik tarif jaringan tambahan yang disertakan dalam respons. Untuk semua tugas lainnya, respons hanya akan mencakup statistik jaringan kumulatif.

`${ECS_CONTAINER_METADATA_URI_V4}/task/stats`

Jalan ini mengembalikan statistik Docker untuk semua kontainer yang terkait dengan tugas. Hal ini dapat digunakan oleh kontainer sidecar untuk mengekstrak metrik jaringan. Untuk informasi selengkapnya tentang masing-masing statistik yang dikembalikan, lihat ContainerStatsdi dokumentasi Docker API.

Untuk tugas Amazon ECS yang menggunakan mode `bridge` jaringan `awsvpc` atau yang dihosting di instans Amazon EC2 Linux yang menjalankan setidaknya `1.43.0` versi agen penampung, akan ada statistik tarif jaringan tambahan yang disertakan dalam respons. Untuk semua tugas lainnya, respons hanya akan mencakup statistik jaringan kumulatif.

Metadata tugas Amazon ECS V4 Respons JSON

Informasi berikut dikembalikan dari respons JSON (`${ECS_CONTAINER_METADATA_URI_V4}/task`) titik akhir metadata tugas. Hal ini termasuk metadata yang terkait dengan tugas selain metadata untuk setiap kontainer dalam tugas.

`Cluster`

Nama Sumber Daya Amazon (ARN) atau nama pendek dari cluster Amazon ECS tempat tugas tersebut berada.

`ServiceName`

Nama layanan tempat tugas itu berada. ServiceName akan muncul untuk instans penampung Amazon EC2 dan Amazon ECS Anywhere jika tugas dikaitkan dengan layanan.

> **Note**
>
> `ServiceName`Metadata hanya disertakan saat menggunakan versi `1.63.1` agen penampung Amazon ECS atau yang lebih baru.

`VPCID`

ID VPC dari instans penampung Amazon EC2. Bidang ini hanya muncul untuk instans Amazon EC2.

> **Note**
>
> `VPCID`Metadata hanya disertakan saat menggunakan versi `1.63.1` agen penampung Amazon ECS atau yang lebih baru.

`TaskARN`

Nama Sumber Daya Amazon (ARN) lengkap dari tugas yang menjadi milik wadah tersebut.

`Family`

Keluarga definisi tugas Amazon ECS untuk tugas tersebut.

`Revision`

Revisi definisi tugas Amazon ECS untuk tugas tersebut.

`DesiredStatus`

Status yang diinginkan untuk tugas dari Amazon ECS.

`KnownStatus`

Status yang diketahui untuk tugas dari Amazon ECS.

`Limits`

Batas sumber daya yang ditentukan pada tingkat tugas, seperti CPU (dinyatakan dalam vCPU) dan memori. Parameter ini dihilangkan jika tidak ada batasan sumber daya yang ditentukan.

`PullStartedAt`

Stempel waktu saat penarikan gambar kontainer pertama dimulai.

`PullStoppedAt`

Stempel waktu ketika penarikan gambar kontainer terakhir selesai.

`AvailabilityZone`

Availability Zone tempat tugas berada.

> **Note**
>
> Metadata Availability Zone hanya tersedia untuk tugas Fargate menggunakan platform versi 1.4 atau yang lebih baru (Linux) atau 1.0.0 (Windows).

`LaunchType`

Jenis peluncuran yang digunakan tugas. Saat menggunakan penyedia kapasitas cluster, ini menunjukkan apakah tugas tersebut menggunakan infrastruktur Fargate atau EC2.

> **Note**
>
> `LaunchType`Metadata ini hanya disertakan saat menggunakan Amazon ECS Linux versi agen container atau yang lebih baru (Linux) `1.45.0` atau 1.0.0 atau yang lebih baru (Windows).

`Containers`

Daftar metadata kontainer untuk setiap kontainer yang terkait dengan tugas.

`DockerId`

ID Docker untuk wadah.

Saat Anda menggunakan Fargate, id adalah hex 32 digit diikuti dengan angka 10 digit.

`Name`

Nama wadah seperti yang ditentukan dalam definisi tugas.

`DockerName`

Nama wadah yang dipasok ke Docker. Agen penampung Amazon ECS menghasilkan nama unik untuk penampung untuk menghindari tabrakan nama saat beberapa salinan dari definisi tugas yang sama dijalankan pada satu instance.

`Image`

Gambar untuk wadah.

`ImageID`

Intisari SHA-256 untuk gambar.

`Ports`

Port apa pun yang terbuka untuk wadah. Parameter ini dihilangkan jika tidak ada port yang terbuka.

`Labels`

Label apa pun diterapkan pada wadah. Parameter ini dihilangkan jika tidak ada label yang diterapkan.

`DesiredStatus`

Status yang diinginkan untuk wadah dari Amazon ECS.

`KnownStatus`

Status yang diketahui untuk wadah dari Amazon ECS.

`ExitCode`

Kode keluar untuk wadah. Parameter ini dihilangkan jika wadah belum keluar.

`Limits`

Batas sumber daya yang ditentukan pada tingkat kontainer, seperti CPU (dinyatakan dalam unit CPU) dan memori. Parameter ini dihilangkan jika tidak ada batasan sumber daya yang ditentukan.

`CreatedAt`

Cap waktu untuk saat wadah dibuat. Parameter ini dihilangkan jika wadah belum dibuat.

`StartedAt`

Cap waktu untuk saat wadah dimulai. Parameter ini dihilangkan jika wadah belum dimulai.

`FinishedAt`

Cap waktu untuk saat wadah berhenti. Parameter ini dihilangkan jika wadah belum berhenti.

`Type`

Jenis wadahnya. Wadah yang ditentukan dalam definisi tugas Anda adalah tipe`NORMAL`. Anda dapat mengabaikan jenis penampung lain, yang digunakan untuk penyediaan sumber daya tugas internal oleh agen penampung Amazon ECS.

`LogDriver`

Driver log yang digunakan wadah.

> **Note**
>
> `LogDriver`Metadata ini hanya disertakan saat menggunakan Amazon ECS Linux container agent versi `1.45.0` atau yang lebih baru.

`LogOptions`

Opsi driver log yang ditentukan untuk wadah.

> **Note**
>
> `LogOptions`Metadata ini hanya disertakan saat menggunakan Amazon ECS Linux container agent versi `1.45.0` atau yang lebih baru.

`ContainerARN`

Nama Sumber Daya Amazon (ARN) lengkap dari wadah.

> **Note**
>
> `ContainerARN`Metadata ini hanya disertakan saat menggunakan Amazon ECS Linux container agent versi `1.45.0` atau yang lebih baru.

`Networks`

Informasi jaringan untuk wadah, seperti mode jaringan dan alamat IP. Parameter ini dihilangkan jika tidak ada informasi jaringan yang didefinisikan.

`ExecutionStoppedAt`

Cap waktu untuk saat tugas `DesiredStatus` pindah ke`STOPPED`. Ini terjadi ketika wadah penting bergerak ke`STOPPED`.

Contoh metadata tugas Amazon ECS v4

Contoh berikut menampilkan output contoh dari masing-masing titik akhir metadata tugas.

Respons metadata kontainer contoh

Ketika melakukan kueri terhadap titik akhir `${ECS_CONTAINER_METADATA_URI_V4}`, hanya metadata kontainer tersebut yang dikembalikan kepada Anda. Berikut ini adalah output contoh.

```
{
    "DockerId": "ea32192c8553fbff06c9340478a2ff089b2bb5646fb718b4ee206641c9086d66",
    "Name": "curl",
    "DockerName": "ecs-curltest-24-curl-cca48e8dcadd97805600",
    "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
    "ImageID":
 "sha256:d691691e9652791a60114e67b365688d20d19940dde7c4736ea30e660d8d3553",
    "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/8f03e41243824aea923aca126495f665",
        "com.amazonaws.ecs.task-definition-family": "curltest",
        "com.amazonaws.ecs.task-definition-version": "24"
    },
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "Limits": {
        "CPU": 10,
        "Memory": 128
    },
    "CreatedAt": "2020-10-02T00:15:07.620912337Z",
    "StartedAt": "2020-10-02T00:15:08.062559351Z",
    "Type": "NORMAL",
    "LogDriver": "awslogs",
    "LogOptions": {
        "awslogs-create-group": "true",
        "awslogs-group": "/ecs/metadata",
        "awslogs-region": "us-west-2",
```

```
        "awslogs-stream": "ecs/curl/8f03e41243824aea923aca126495f665"
    },
    "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/0206b271-
b33f-47ab-86c6-a0ba208a70a9",
    "Networks": [
        {
            "NetworkMode": "awsvpc",
            "IPv4Addresses": [
                "10.0.2.100"
            ],
            "AttachmentIndex": 0,
            "MACAddress": "0e:9e:32:c7:48:85",
            "IPv4SubnetCIDRBlock": "10.0.2.0/24",
            "PrivateDNSName": "ip-10-0-2-100.us-west-2.compute.internal",
            "SubnetGatewayIpv4Address": "10.0.2.1/24"
        }
    ]
}
```

Respons metadata tugas contoh

Ketika melakukan kueri terhadap titik akhir `${ECS_CONTAINER_METADATA_URI_V4}/task`, metadata dari tugas yang berisi bagian kontainer selain metada untuk masing-masing kontainer dalam tugas dikembalikan kepada Anda. Berikut ini adalah output contoh.

```
{
    "Cluster": "default",
    "TaskARN": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
    "Family": "curltest",
    "ServiceName": "MyService",
    "Revision": "26",
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "PullStartedAt": "2020-10-02T00:43:06.202617438Z",
    "PullStoppedAt": "2020-10-02T00:43:06.31288465Z",
    "AvailabilityZone": "us-west-2d",
    "VPCID": "vpc-1234567890abcdef0",
    "LaunchType": "EC2",
    "Containers": [
        {
            "DockerId":
 "598cba581fe3f939459eaba1e071d5c93bb2c49b7d1ba7db6bb19deeb70d8e38",
```

```
            "Name": "~internal~ecs~pause",
            "DockerName": "ecs-curltest-26-internalecspause-e292d586b6f9dade4a00",
            "Image": "amazon/amazon-ecs-pause:0.1.0",
            "ImageID": "",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RESOURCES_PROVISIONED",
            "KnownStatus": "RESOURCES_PROVISIONED",
            "Limits": {
                "CPU": 0,
                "Memory": 0
            },
            "CreatedAt": "2020-10-02T00:43:05.602352471Z",
            "StartedAt": "2020-10-02T00:43:06.076707576Z",
            "Type": "CNI_PAUSE",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        },
        {
            "DockerId":
 "ee08638adaaf009d78c248913f629e38299471d45fe7dc944d1039077e3424ca",
            "Name": "curl",
            "DockerName": "ecs-curltest-26-curl-a0e7dba5aca6d8cb2e00",
            "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
            "ImageID":
 "sha256:d691691e9652791a60114e67b365688d20d19940dde7c4736ea30e660d8d3553",
            "Labels": {
```

```
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "curl",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RUNNING",
            "KnownStatus": "RUNNING",
            "Limits": {
                "CPU": 10,
                "Memory": 128
            },
            "CreatedAt": "2020-10-02T00:43:06.326590752Z",
            "StartedAt": "2020-10-02T00:43:06.767535449Z",
            "Type": "NORMAL",
            "LogDriver": "awslogs",
            "LogOptions": {
                "awslogs-create-group": "true",
                "awslogs-group": "/ecs/metadata",
                "awslogs-region": "us-west-2",
                "awslogs-stream": "ecs/curl/158d1c8083dd49d6b527399fd6414f5c"
            },
            "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/
abb51bdd-11b4-467f-8f6c-adcfe1fe059d",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        }
    ]
}
```

Tugas contoh dengan respons metadata tanda

Ketika melakukan kueri terhadap titik akhir `${ECS_CONTAINER_METADATA_URI_V4}/taskWithTags`, metadata dari tugas yang berisi tugas dan tanda instans kontainer dikembalikan kepada Anda. Berikut ini adalah output contoh.

```
{
    "Cluster": "default",
    "TaskARN": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
    "Family": "curltest",
    "ServiceName": "MyService",
    "Revision": "26",
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "PullStartedAt": "2020-10-02T00:43:06.202617438Z",
    "PullStoppedAt": "2020-10-02T00:43:06.31288465Z",
    "AvailabilityZone": "us-west-2d",
    "VPCID": "vpc-1234567890abcdef0",
    "TaskTags": {
        "tag-use": "task-metadata-endpoint-test"
    },
    "ContainerInstanceTags":{
        "tag_key":"tag_value"
    },
    "LaunchType": "EC2",
    "Containers": [
        {
            "DockerId":
 "598cba581fe3f939459eaba1e071d5c93bb2c49b7d1ba7db6bb19deeb70d8e38",
            "Name": "~internal~ecs~pause",
            "DockerName": "ecs-curltest-26-internalecspause-e292d586b6f9dade4a00",
            "Image": "amazon/amazon-ecs-pause:0.1.0",
            "ImageID": "",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RESOURCES_PROVISIONED",
```

```
            "KnownStatus": "RESOURCES_PROVISIONED",
            "Limits": {
                "CPU": 0,
                "Memory": 0
            },
            "CreatedAt": "2020-10-02T00:43:05.602352471Z",
            "StartedAt": "2020-10-02T00:43:06.076707576Z",
            "Type": "CNI_PAUSE",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        },
        {
            "DockerId":
 "ee08638adaaf009d78c248913f629e38299471d45fe7dc944d1039077e3424ca",
            "Name": "curl",
            "DockerName": "ecs-curltest-26-curl-a0e7dba5aca6d8cb2e00",
            "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
            "ImageID":
 "sha256:d691691e9652791a60114e67b365688d20d19940dde7c4736ea30e660d8d3553",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "curl",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RUNNING",
            "KnownStatus": "RUNNING",
            "Limits": {
                "CPU": 10,
                "Memory": 128
            },
```

```
            "CreatedAt": "2020-10-02T00:43:06.326590752Z",
            "StartedAt": "2020-10-02T00:43:06.767535449Z",
            "Type": "NORMAL",
            "LogDriver": "awslogs",
            "LogOptions": {
                "awslogs-create-group": "true",
                "awslogs-group": "/ecs/metadata",
                "awslogs-region": "us-west-2",
                "awslogs-stream": "ecs/curl/158d1c8083dd49d6b527399fd6414f5c"
            },
            "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/
abb51bdd-11b4-467f-8f6c-adcfe1fe059d",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        }
    ]
}
```

Tugas contoh dengan tanda yang menggambarkan respons metadata kesalahan

Ketika melakukan kueri terhadap titik akhir `${ECS_CONTAINER_METADATA_URI_V4}/taskWithTags`, metadata dari tugas yang berisi tugas dan tanda instans kontainer dikembalikan kepada Anda. Jika terjadi kesalahan saat mengambil data penandaan, kesalahan dikembalikan dalam bentuk respons. Berikut ini adalah contoh keluaran ketika peran IAM yang terkait dengan instance container tidak memiliki `ecs:ListTagsForResource` izin yang diizinkan.

```
{
    "Cluster": "default",
    "TaskARN": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
    "Family": "curltest",
    "ServiceName": "MyService",
```

```
    "Revision": "26",
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "PullStartedAt": "2020-10-02T00:43:06.202617438Z",
    "PullStoppedAt": "2020-10-02T00:43:06.31288465Z",
    "AvailabilityZone": "us-west-2d",
    "VPCID": "vpc-1234567890abcdef0",
    "Errors": [
        {
            "ErrorField": "ContainerInstanceTags",
            "ErrorCode": "AccessDeniedException",
            "ErrorMessage": "User: arn:aws:sts::111122223333:assumed-
role/ecsInstanceRole/i-0744a608689EXAMPLE is not authorized to perform:
 ecs:ListTagsForResource on resource: arn:aws:ecs:us-west-2:111122223333:container-
instance/default/2dd1b186f39845a584488d2ef155c131",
            "StatusCode": 400,
            "RequestId": "cd597ef0-272b-4643-9bd2-1de469870fa6",
            "ResourceARN": "arn:aws:ecs:us-west-2:111122223333:container-instance/
default/2dd1b186f39845a584488d2ef155c131"
        },
        {
            "ErrorField": "TaskTags",
            "ErrorCode": "AccessDeniedException",
            "ErrorMessage": "User: arn:aws:sts::111122223333:assumed-
role/ecsInstanceRole/i-0744a608689EXAMPLE is not authorized to perform:
 ecs:ListTagsForResource on resource: arn:aws:ecs:us-west-2:111122223333:task/
default/9ef30e4b7aa44d0db562749cff4983f3",
            "StatusCode": 400,
            "RequestId": "862c5986-6cd2-4aa6-87cc-70be395531e1",
            "ResourceARN": "arn:aws:ecs:us-west-2:111122223333:task/
default/9ef30e4b7aa44d0db562749cff4983f3"
        }
    ],
    "LaunchType": "EC2",
    "Containers": [
        {
            "DockerId":
 "598cba581fe3f939459eaba1e071d5c93bb2c49b7d1ba7db6bb19deeb70d8e38",
            "Name": "~internal~ecs~pause",
            "DockerName": "ecs-curltest-26-internalecspause-e292d586b6f9dade4a00",
            "Image": "amazon/amazon-ecs-pause:0.1.0",
            "ImageID": "",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
```

```
                "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RESOURCES_PROVISIONED",
            "KnownStatus": "RESOURCES_PROVISIONED",
            "Limits": {
                "CPU": 0,
                "Memory": 0
            },
            "CreatedAt": "2020-10-02T00:43:05.602352471Z",
            "StartedAt": "2020-10-02T00:43:06.076707576Z",
            "Type": "CNI_PAUSE",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        },
        {
            "DockerId":
 "ee08638adaaf009d78c248913f629e38299471d45fe7dc944d1039077e3424ca",
            "Name": "curl",
            "DockerName": "ecs-curltest-26-curl-a0e7dba5aca6d8cb2e00",
            "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
            "ImageID":
 "sha256:d691691e9652791a60114e67b365688d20d19940dde7c4736ea30e660d8d3553",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "curl",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
```

```
            },
            "DesiredStatus": "RUNNING",
            "KnownStatus": "RUNNING",
            "Limits": {
                "CPU": 10,
                "Memory": 128
            },
            "CreatedAt": "2020-10-02T00:43:06.326590752Z",
            "StartedAt": "2020-10-02T00:43:06.767535449Z",
            "Type": "NORMAL",
            "LogDriver": "awslogs",
            "LogOptions": {
                "awslogs-create-group": "true",
                "awslogs-group": "/ecs/metadata",
                "awslogs-region": "us-west-2",
                "awslogs-stream": "ecs/curl/158d1c8083dd49d6b527399fd6414f5c"
            },
            "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/
abb51bdd-11b4-467f-8f6c-adcfe1fe059d",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        }
    ]
}
```

Respons statistik kontainer contoh

Ketika melakukan kueri terhadap titik akhir `${ECS_CONTAINER_METADATA_URI_V4}/stats`, metrik jaringan untuk kontainer dikembalikan kepada Anda. Untuk tugas Amazon ECS yang menggunakan mode `bridge` jaringan `awsvpc` atau yang dihosting di instans Amazon EC2 yang menjalankan setidaknya `1.43.0` versi agen penampung, akan ada statistik tarif jaringan tambahan

yang disertakan dalam respons. Untuk semua tugas lainnya, respons hanya akan mencakup statistik jaringan kumulatif.

Berikut ini adalah contoh output dari tugas Amazon ECS di Amazon EC2 yang menggunakan mode jaringan`bridge`.


```
{
    "read": "2020-10-02T00:51:13.410254284Z",
    "preread": "2020-10-02T00:51:12.406202398Z",
    "pids_stats": {
        "current": 3
    },
    "blkio_stats": {
        "io_service_bytes_recursive": [

        ],
        "io_serviced_recursive": [

        ],
        "io_queue_recursive": [

        ],
        "io_service_time_recursive": [

        ],
        "io_wait_time_recursive": [

        ],
        "io_merged_recursive": [

        ],
        "io_time_recursive": [

        ],
        "sectors_recursive": [

        ]
    },
    "num_procs": 0,
    "storage_stats": {

    },
    "cpu_stats": {
```

```
        "cpu_usage": {
            "total_usage": 360968065,
            "percpu_usage": [
                182359190,
                178608875
            ],
            "usage_in_kernelmode": 40000000,
            "usage_in_usermode": 290000000
        },
        "system_cpu_usage": 13939680000000,
        "online_cpus": 2,
        "throttling_data": {
            "periods": 0,
            "throttled_periods": 0,
            "throttled_time": 0
        }
    },
    "precpu_stats": {
        "cpu_usage": {
            "total_usage": 360968065,
            "percpu_usage": [
                182359190,
                178608875
            ],
            "usage_in_kernelmode": 40000000,
            "usage_in_usermode": 290000000
        },
        "system_cpu_usage": 13937670000000,
        "online_cpus": 2,
        "throttling_data": {
            "periods": 0,
            "throttled_periods": 0,
            "throttled_time": 0
        }
    },
    "memory_stats": {
        "usage": 1806336,
        "max_usage": 6299648,
        "stats": {
            "active_anon": 606208,
            "active_file": 0,
            "cache": 0,
            "dirty": 0,
            "hierarchical_memory_limit": 134217728,
```

```
            "hierarchical_memsw_limit": 268435456,
            "inactive_anon": 0,
            "inactive_file": 0,
            "mapped_file": 0,
            "pgfault": 4185,
            "pgmajfault": 0,
            "pgpgin": 2926,
            "pgpgout": 2778,
            "rss": 606208,
            "rss_huge": 0,
            "total_active_anon": 606208,
            "total_active_file": 0,
            "total_cache": 0,
            "total_dirty": 0,
            "total_inactive_anon": 0,
            "total_inactive_file": 0,
            "total_mapped_file": 0,
            "total_pgfault": 4185,
            "total_pgmajfault": 0,
            "total_pgpgin": 2926,
            "total_pgpgout": 2778,
            "total_rss": 606208,
            "total_rss_huge": 0,
            "total_unevictable": 0,
            "total_writeback": 0,
            "unevictable": 0,
            "writeback": 0
        },
        "limit": 134217728
    },
    "name": "/ecs-curltest-26-curl-c2e5f6e0cf91b0bead01",
    "id": "5fc21e5b015f899d22618f8aede80b6d70d71b2a75465ea49d9462c8f3d2d3af",
    "networks": {
        "eth0": {
            "rx_bytes": 84,
            "rx_packets": 2,
            "rx_errors": 0,
            "rx_dropped": 0,
            "tx_bytes": 84,
            "tx_packets": 2,
            "tx_errors": 0,
            "tx_dropped": 0
        }
    },
```

```
    "network_rate_stats": {
        "rx_bytes_per_sec": 0,
        "tx_bytes_per_sec": 0
    }
}
```

Respons statistik tugas contoh

Ketika melakukan kueri terhadap titik akhir `${ECS_CONTAINER_METADATA_URI_V4}/task/stats`, metrik jaringan dari tugas yang berisi bagian kontainer dikembalikan kepada Anda. Berikut ini adalah output contoh.

```
{
    "01999f2e5c6cf4df3873f28950e6278813408f281c54778efec860d0caad4854": {
        "read": "2020-10-02T00:51:32.51467703Z",
        "preread": "2020-10-02T00:51:31.50860463Z",
        "pids_stats": {
            "current": 1
        },
        "blkio_stats": {
            "io_service_bytes_recursive": [

            ],
            "io_serviced_recursive": [

            ],
            "io_queue_recursive": [

            ],
            "io_service_time_recursive": [

            ],
            "io_wait_time_recursive": [

            ],
            "io_merged_recursive": [

            ],
            "io_time_recursive": [

            ],
            "sectors_recursive": [
```

```
            ]
        },
        "num_procs": 0,
        "storage_stats": {

        },
        "cpu_stats": {
            "cpu_usage": {
                "total_usage": 177232665,
                "percpu_usage": [
                    13376224,
                    163856441
                ],
                "usage_in_kernelmode": 0,
                "usage_in_usermode": 160000000
            },
            "system_cpu_usage": 13977820000000,
            "online_cpus": 2,
            "throttling_data": {
                "periods": 0,
                "throttled_periods": 0,
                "throttled_time": 0
            }
        },
        "precpu_stats": {
            "cpu_usage": {
                "total_usage": 177232665,
                "percpu_usage": [
                    13376224,
                    163856441
                ],
                "usage_in_kernelmode": 0,
                "usage_in_usermode": 160000000
            },
            "system_cpu_usage": 13975800000000,
            "online_cpus": 2,
            "throttling_data": {
                "periods": 0,
                "throttled_periods": 0,
                "throttled_time": 0
            }
        },
        "memory_stats": {
            "usage": 532480,
```

```
            "max_usage": 6279168,
            "stats": {
                "active_anon": 40960,
                "active_file": 0,
                "cache": 0,
                "dirty": 0,
                "hierarchical_memory_limit": 9223372036854771712,
                "hierarchical_memsw_limit": 9223372036854771712,
                "inactive_anon": 0,
                "inactive_file": 0,
                "mapped_file": 0,
                "pgfault": 2033,
                "pgmajfault": 0,
                "pgpgin": 1734,
                "pgpgout": 1724,
                "rss": 40960,
                "rss_huge": 0,
                "total_active_anon": 40960,
                "total_active_file": 0,
                "total_cache": 0,
                "total_dirty": 0,
                "total_inactive_anon": 0,
                "total_inactive_file": 0,
                "total_mapped_file": 0,
                "total_pgfault": 2033,
                "total_pgmajfault": 0,
                "total_pgpgin": 1734,
                "total_pgpgout": 1724,
                "total_rss": 40960,
                "total_rss_huge": 0,
                "total_unevictable": 0,
                "total_writeback": 0,
                "unevictable": 0,
                "writeback": 0
            },
            "limit": 4073377792
        },
        "name": "/ecs-curltest-26-internalecspause-a6bcc3dbadfacfe85300",
        "id": "01999f2e5c6cf4df3873f28950e6278813408f281c54778efec860d0caad4854",
        "networks": {
            "eth0": {
                "rx_bytes": 84,
                "rx_packets": 2,
                "rx_errors": 0,
```

```
                "rx_dropped": 0,
                "tx_bytes": 84,
                "tx_packets": 2,
                "tx_errors": 0,
                "tx_dropped": 0
            }
        },
        "network_rate_stats": {
            "rx_bytes_per_sec": 0,
            "tx_bytes_per_sec": 0
        }
    },
    "5fc21e5b015f899d22618f8aede80b6d70d71b2a75465ea49d9462c8f3d2d3af": {
        "read": "2020-10-02T00:51:32.512771349Z",
        "preread": "2020-10-02T00:51:31.510597736Z",
        "pids_stats": {
            "current": 3
        },
        "blkio_stats": {
            "io_service_bytes_recursive": [

            ],
            "io_serviced_recursive": [

            ],
            "io_queue_recursive": [

            ],
            "io_service_time_recursive": [

            ],
            "io_wait_time_recursive": [

            ],
            "io_merged_recursive": [

            ],
            "io_time_recursive": [

            ],
            "sectors_recursive": [

            ]
        },
```

```
        "num_procs": 0,
        "storage_stats": {

        },
        "cpu_stats": {
            "cpu_usage": {
                "total_usage": 379075681,
                "percpu_usage": [
                    191355275,
                    187720406
                ],
                "usage_in_kernelmode": 60000000,
                "usage_in_usermode": 310000000
            },
            "system_cpu_usage": 13977800000000,
            "online_cpus": 2,
            "throttling_data": {
                "periods": 0,
                "throttled_periods": 0,
                "throttled_time": 0
            }
        },
        "precpu_stats": {
            "cpu_usage": {
                "total_usage": 378825197,
                "percpu_usage": [
                    191104791,
                    187720406
                ],
                "usage_in_kernelmode": 60000000,
                "usage_in_usermode": 310000000
            },
            "system_cpu_usage": 13975800000000,
            "online_cpus": 2,
            "throttling_data": {
                "periods": 0,
                "throttled_periods": 0,
                "throttled_time": 0
            }
        },
        "memory_stats": {
            "usage": 1814528,
            "max_usage": 6299648,
            "stats": {
```

```
                "active_anon": 606208,
                "active_file": 0,
                "cache": 0,
                "dirty": 0,
                "hierarchical_memory_limit": 134217728,
                "hierarchical_memsw_limit": 268435456,
                "inactive_anon": 0,
                "inactive_file": 0,
                "mapped_file": 0,
                "pgfault": 5377,
                "pgmajfault": 0,
                "pgpgin": 3613,
                "pgpgout": 3465,
                "rss": 606208,
                "rss_huge": 0,
                "total_active_anon": 606208,
                "total_active_file": 0,
                "total_cache": 0,
                "total_dirty": 0,
                "total_inactive_anon": 0,
                "total_inactive_file": 0,
                "total_mapped_file": 0,
                "total_pgfault": 5377,
                "total_pgmajfault": 0,
                "total_pgpgin": 3613,
                "total_pgpgout": 3465,
                "total_rss": 606208,
                "total_rss_huge": 0,
                "total_unevictable": 0,
                "total_writeback": 0,
                "unevictable": 0,
                "writeback": 0
            },
            "limit": 134217728
        },
        "name": "/ecs-curltest-26-curl-c2e5f6e0cf91b0bead01",
        "id": "5fc21e5b015f899d22618f8aede80b6d70d71b2a75465ea49d9462c8f3d2d3af",
        "networks": {
            "eth0": {
                "rx_bytes": 84,
                "rx_packets": 2,
                "rx_errors": 0,
                "rx_dropped": 0,
                "tx_bytes": 84,
```

```
                "tx_packets": 2,
                "tx_errors": 0,
                "tx_dropped": 0
            }
        },
        "network_rate_stats": {
            "rx_bytes_per_sec": 0,
            "tx_bytes_per_sec": 0
        }
    }
}
```

Titik akhir metadata tugas Amazon ECS versi 3

> **Important**
>
> Titik akhir metadata tugas versi 3 tidak lagi dipertahankan secara aktif. Kami menyarankan Anda memperbarui titik akhir metadata tugas versi 4 untuk mendapatkan informasi titik akhir metadata terbaru. Untuk informasi selengkapnya, lihat the section called “Titik akhir metadata tugas versi 4”.
>
> Jika Anda menggunakan tugas Amazon ECS yang dihosting AWS Fargate, lihat Titik akhir metadata tugas versi 3 di Panduan Pengguna Layanan Amazon Elastic Container untuk. AWS Fargate

Dimulai dengan versi 1.21.0 dari agen kontainer Amazon ECS, agen menyuntikkan variabel lingkungan yang dipanggil `ECS_CONTAINER_METADATA_URI` ke setiap kontainer dalam tugas. Ketika Anda melakukan kueri terhadap titik akhir metadata tugas versi 3, berbagai metadata tugas dan Statistik Docker tersedia untuk tugas. Untuk tugas yang menggunakan mode jaringan `bridge`, metrik jaringan tersedia ketika melakukan kueri terhadap titik akhir `/stats`.

Fitur titik akhir metadata tugas versi 3 diaktifkan secara default untuk tugas yang menggunakan jenis peluncuran Fargate pada platform versi v1.3.0 atau yang lebih baru dan tugas yang menggunakan tipe peluncuran EC2 dan diluncurkan di infrastruktur Amazon EC2 Linux yang menjalankan setidaknya versi 1.21.0 dari agen penampung Amazon ECS atau di infrastruktur Windows Amazon EC2 yang menjalankan setidaknya versi agen penampung Amazon ECS dan menggunakan mode jaringan. `1.54.0` `awsvpc` Untuk informasi selengkapnya, lihat Manajemen instance kontainer Linux.

Anda dapat menambahkan dukungan untuk fitur ini pada instans kontainer yang lebih lama dengan memperbarui agen ke versi terbaru. Untuk informasi selengkapnya, lihat Memperbarui agen kontainer Amazon ECS.

> **Important**
>
> Untuk tugas yang menggunakan tipe peluncuran Fargate dan versi platform sebelum v1.3.0, titik akhir metadata tugas versi 2 didukung. Untuk informasi selengkapnya, lihat Titik akhir metadata tugas Amazon ECS versi 2.

Jalur titik akhir Metadata Tugas versi 3

Titik akhir metadata tugas berikut tersedia untuk kontainer:

`${ECS_CONTAINER_METADATA_URI}`

Jalur ini mengembalikan JSON metadata untuk kontainer.

`${ECS_CONTAINER_METADATA_URI}/task`

Jalur ini mengembalikan JSON metadata untuk tugas, termasuk daftar ID kontainer dan nama semua kontainer yang terkait dengan tugas. Untuk informasi selengkapnya tentang respons titik akhir ini, lihat Metadata tugas Amazon ECS v3 respons JSON.

`${ECS_CONTAINER_METADATA_URI}/taskWithTags`

Jalur ini mengembalikan metadata pada tugas yang disertakan dalam titik akhir `/task` selain tugas dan tanda instans kontainer yang dapat diambil menggunakan API `ListTagsForResource`.

`${ECS_CONTAINER_METADATA_URI}/stats`

Jalur ini mengembalikan JSON statistik Docker pada kontainer Docker tertentu. Untuk informasi selengkapnya tentang masing-masing statistik yang dikembalikan, lihat ContainerStatsdi dokumentasi Docker API.

`${ECS_CONTAINER_METADATA_URI}/task/stats`

Jalur ini mengembalikan JSON statistik Docker untuk semua kontainer yang terkait dengan tugas. Untuk informasi selengkapnya tentang masing-masing statistik yang dikembalikan, lihat ContainerStatsdi dokumentasi Docker API.

Metadata tugas Amazon ECS v3 respons JSON

Informasi berikut dikembalikan dari respons JSON titik akhir metadata tugas (`${ECS_CONTAINER_METADATA_URI}/task`).

`Cluster`

Nama Sumber Daya Amazon (ARN) atau nama pendek dari cluster Amazon ECS tempat tugas tersebut berada.

`TaskARN`

Nama Sumber Daya Amazon (ARN) lengkap dari tugas yang menjadi milik wadah tersebut.

`Family`

Keluarga definisi tugas Amazon ECS untuk tugas tersebut.

`Revision`

Revisi definisi tugas Amazon ECS untuk tugas tersebut.

`DesiredStatus`

Status yang diinginkan untuk tugas dari Amazon ECS.

`KnownStatus`

Status yang diketahui untuk tugas dari Amazon ECS.

`Limits`

Batas sumber daya yang ditentukan pada tingkat tugas, seperti CPU (dinyatakan dalam vCPU) dan memori. Parameter ini dihilangkan jika tidak ada batasan sumber daya yang ditentukan.

`PullStartedAt`

Stempel waktu saat penarikan gambar kontainer pertama dimulai.

`PullStoppedAt`

Stempel waktu ketika penarikan gambar kontainer terakhir selesai.

`AvailabilityZone`

Availability Zone tempat tugas berada.

> **Note**
>
> Metadata Availability Zone hanya tersedia untuk tugas Fargate menggunakan platform versi 1.4 atau yang lebih baru (Linux) atau 1.0.0 atau yang lebih baru (Windows).

`Containers`

Daftar metadata kontainer untuk setiap kontainer yang terkait dengan tugas.

`DockerId`

ID Docker untuk wadah.

`Name`

Nama wadah seperti yang ditentukan dalam definisi tugas.

`DockerName`

Nama wadah yang dipasok ke Docker. Agen penampung Amazon ECS menghasilkan nama unik untuk penampung untuk menghindari tabrakan nama saat beberapa salinan dari definisi tugas yang sama dijalankan pada satu instance.

`Image`

Gambar untuk wadah.

`ImageID`

Intisari SHA-256 untuk gambar.

`Ports`

Port apa pun yang terbuka untuk wadah. Parameter ini dihilangkan jika tidak ada port yang terbuka.

`Labels`

Label apa pun diterapkan pada wadah. Parameter ini dihilangkan jika tidak ada label yang diterapkan.

`DesiredStatus`

Status yang diinginkan untuk wadah dari Amazon ECS.

`KnownStatus`

Status yang diketahui untuk wadah dari Amazon ECS.

`ExitCode`

Kode keluar untuk wadah. Parameter ini dihilangkan jika wadah belum keluar.

`Limits`

Batas sumber daya yang ditentukan pada tingkat kontainer, seperti CPU (dinyatakan dalam unit CPU) dan memori. Parameter ini dihilangkan jika tidak ada batasan sumber daya yang ditentukan.

`CreatedAt`

Cap waktu untuk saat wadah dibuat. Parameter ini dihilangkan jika wadah belum dibuat.

`StartedAt`

Cap waktu untuk saat wadah dimulai. Parameter ini dihilangkan jika wadah belum dimulai.

`FinishedAt`

Cap waktu untuk saat wadah berhenti. Parameter ini dihilangkan jika wadah belum berhenti.

`Type`

Jenis wadahnya. Wadah yang ditentukan dalam definisi tugas Anda adalah tipe`NORMAL`. Anda dapat mengabaikan jenis penampung lain, yang digunakan untuk penyediaan sumber daya tugas internal oleh agen penampung Amazon ECS.

`Networks`

Informasi jaringan untuk wadah, seperti mode jaringan dan alamat IP. Parameter ini dihilangkan jika tidak ada informasi jaringan yang didefinisikan.

`ClockDrift`

Informasi tentang perbedaan antara waktu referensi dan waktu sistem. Ini berlaku untuk sistem operasi Linux. Kemampuan ini menggunakan Amazon Time Sync Service untuk mengukur akurasi jam dan memberikan kesalahan jam yang terikat pada kontainer. Untuk informasi selengkapnya, lihat Mengatur waktu untuk instans Linux Anda di Panduan Pengguna Amazon EC2 untuk instans Linux.

`ReferenceTime`

Dasar akurasi jam. Amazon ECS menggunakan standar global Coordinated Universal Time (UTC) melalui NTP, misalnya. `2021-09-07T16:57:44Z`

`ClockErrorBound`

Ukuran kesalahan jam, didefinisikan sebagai offset ke UTC. Kesalahan ini adalah perbedaan milidetik antara waktu referensi dan waktu sistem.

`ClockSynchronizationStatus`

Menunjukkan apakah upaya sinkronisasi terbaru antara waktu sistem dan waktu referensi berhasil.

Nilai yang valid adalah `SYNCHRONIZED` dan `NOT_SYNCHRONIZED`.

`ExecutionStoppedAt`

Cap waktu untuk saat tugas `DesiredStatus` pindah ke`STOPPED`. Ini terjadi ketika wadah penting bergerak ke`STOPPED`.

Contoh metadata tugas Amazon ECS v3

Contoh berikut menampilkan output sampel dari titik akhir metadata tugas.

Respons metadata kontainer contoh

Ketika melakukan kueri terhadap titik akhir `${ECS_CONTAINER_METADATA_URI}`, hanya metadata kontainer tersebut yang dikembalikan kepada Anda. Berikut ini adalah output contoh.

```
{
    "DockerId": "43481a6ce4842eec8fe72fc28500c6b52edcc0917f105b83379f88cac1ff3946",
    "Name": "nginx-curl",
    "DockerName": "ecs-nginx-5-nginx-curl-ccccb9f49db0dfe0d901",
    "Image": "nrdlngr/nginx-curl",
    "ImageID":
 "sha256:2e00ae64383cfc865ba0a2ba37f61b50a120d2d9378559dcd458dc0de47bc165",
    "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "nginx-curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
```

```
        "com.amazonaws.ecs.task-definition-version": "5"
    },
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "Limits": {
        "CPU": 512,
        "Memory": 512
    },
    "CreatedAt": "2018-02-01T20:55:10.554941919Z",
    "StartedAt": "2018-02-01T20:55:11.064236631Z",
    "Type": "NORMAL",
    "Networks": [
        {
            "NetworkMode": "awsvpc",
            "IPv4Addresses": [
                "10.0.2.106"
            ]
        }
    ]
}
```

Respons metadata tugas contoh

Ketika melakukan kueri terhadap titik akhir `${ECS_CONTAINER_METADATA_URI}/task`, metadata dari tugas yang berisi bagian kontainer dikembalikan kepada Anda. Berikut ini adalah output contoh.

Respons JSON berikut adalah untuk tugas kontainer tunggal.

```
{
  "Cluster": "default",
  "TaskARN": "arn:aws:ecs:us-east-2:012345678910:task/9781c248-0edd-4cdb-9a93-
f63cb662a5d3",
  "Family": "nginx",
  "Revision": "5",
  "DesiredStatus": "RUNNING",
  "KnownStatus": "RUNNING",
  "Containers": [
    {
      "DockerId": "731a0d6a3b4210e2448339bc7015aaa79bfe4fa256384f4102db86ef94cbbc4c",
      "Name": "~internal~ecs~pause",
      "DockerName": "ecs-nginx-5-internalecspause-acc699c0cbf2d6d11700",
      "Image": "amazon/amazon-ecs-pause:0.1.0",
      "ImageID": "",
```

```
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RESOURCES_PROVISIONED",
      "KnownStatus": "RESOURCES_PROVISIONED",
      "Limits": {
        "CPU": 0,
        "Memory": 0
      },
      "CreatedAt": "2018-02-01T20:55:08.366329616Z",
      "StartedAt": "2018-02-01T20:55:09.058354915Z",
      "Type": "CNI_PAUSE",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    },
    {
      "DockerId": "43481a6ce4842eec8fe72fc28500c6b52edcc0917f105b83379f88cac1ff3946",
      "Name": "nginx-curl",
      "DockerName": "ecs-nginx-5-nginx-curl-ccccb9f49db0dfe0d901",
      "Image": "nrdlngr/nginx-curl",
      "ImageID":
 "sha256:2e00ae64383cfc865ba0a2ba37f61b50a120d2d9378559dcd458dc0de47bc165",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "nginx-curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": {
```

```
        "CPU": 512,
        "Memory": 512
      },
      "CreatedAt": "2018-02-01T20:55:10.554941919Z",
      "StartedAt": "2018-02-01T20:55:11.064236631Z",
      "Type": "NORMAL",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    }
  ],
  "PullStartedAt": "2018-02-01T20:55:09.372495529Z",
  "PullStoppedAt": "2018-02-01T20:55:10.552018345Z",
  "AavailabilityZone": "us-east-2b"
}
```


Titik akhir metadata tugas Amazon ECS versi 2

> **Important**
>
> Titik akhir metadata tugas versi 2 tidak lagi dipertahankan secara aktif. Kami menyarankan Anda memperbarui titik akhir metadata tugas versi 4 untuk mendapatkan informasi titik akhir metadata terbaru. Untuk informasi selengkapnya, lihat the section called “Titik akhir metadata tugas versi 4”.

Dimulai dengan versi 1.17.0 dari agen penampung Amazon ECS, berbagai metadata tugas dan statistik Docker tersedia untuk tugas yang menggunakan mode `awsvpc` jaringan pada titik akhir HTTP yang disediakan oleh agen penampung Amazon ECS.

Semua kontainer milik tugas yang diluncurkan dengan mode jaringan `awsvpc` menerima alamat IPv4 lokal dalam rentang alamat tautan lokal yang telah ditetapkan. Saat kontainer menanyakan titik akhir metadata, agen penampung Amazon ECS dapat menentukan tugas mana yang dimiliki penampung berdasarkan alamat IP uniknya, serta metadata serta statistik untuk tugas tersebut dikembalikan.

Mengaktifkan metadata tugas

Fitur metadata tugas versi 2 diaktifkan secara default untuk hal berikut:

- Tugas menggunakan jenis peluncuran Fargate yang menggunakan platform versi v1.1.0 atau yang lebih baru. Untuk informasi selengkapnya, lihat Versi platform Fargate Linux.
- Tugas menggunakan jenis peluncuran EC2 yang juga menggunakan mode `awsvpc` jaringan dan diluncurkan pada infrastruktur Amazon EC2 Linux yang menjalankan setidaknya versi 1.17.0 dari agen penampung Amazon ECS atau di infrastruktur Amazon EC2 Windows yang menjalankan setidaknya `1.54.0` versi agen penampung Amazon ECS. Untuk informasi selengkapnya, lihat Manajemen instance kontainer Linux.

Anda dapat menambahkan dukungan untuk fitur ini pada instans kontainer yang lebih lama dengan memperbarui agen ke versi terbaru. Untuk informasi selengkapnya, lihat Memperbarui agen kontainer Amazon ECS.

Jalur titik akhir metadata tugas

Titik akhir API tersedia untuk kontainer:

`169.254.170.2/v2/metadata`

Titik akhir ini mengembalikan JSON metadata untuk tugas, termasuk daftar ID kontainer dan nama semua kontainer yang terkait dengan tugas tersebut. Untuk informasi selengkapnya tentang respons titik akhir ini, lihat Respons JSON metadata tugas.

`169.254.170.2/v2/metadata/<container-id>`

Titik akhir ini mengembalikan JSON metadata untuk ID kontainer Docker ditentukan.

`169.254.170.2/v2/metadata/taskWithTags`

Jalur ini mengembalikan metadata pada tugas yang disertakan dalam titik akhir `/task` selain tugas dan tanda instans kontainer yang dapat diambil menggunakan API `ListTagsForResource`.

`169.254.170.2/v2/stats`

Titik akhir ini mengembalikan JSON statistik Docker pada semua kontainer yang terkait dengan tugas. Untuk informasi selengkapnya tentang masing-masing statistik yang dikembalikan, lihat ContainerStatsdi dokumentasi Docker API.

`169.254.170.2/v2/stats/<container-id>`

Titik akhir ini mengembalikan JSON statistik Docker untuk ID kontainer Docker tertentu. Untuk informasi selengkapnya tentang masing-masing statistik yang dikembalikan, lihat ContainerStatsdi dokumentasi Docker API.

Respons JSON metadata tugas

Informasi berikut dikembalikan dari respons JSON titik akhir metadata tugas (`169.254.170.2/v2/metadata`).

`Cluster`

Nama Sumber Daya Amazon (ARN) atau nama pendek dari cluster Amazon ECS tempat tugas tersebut berada.

`TaskARN`

Nama Sumber Daya Amazon (ARN) lengkap dari tugas yang menjadi milik wadah tersebut.

`Family`

Keluarga definisi tugas Amazon ECS untuk tugas tersebut.

`Revision`

Revisi definisi tugas Amazon ECS untuk tugas tersebut.

`DesiredStatus`

Status yang diinginkan untuk tugas dari Amazon ECS.

`KnownStatus`

Status yang diketahui untuk tugas dari Amazon ECS.

`Limits`

Batas sumber daya yang ditentukan pada tingkat tugas, seperti CPU (dinyatakan dalam vCPU) dan memori. Parameter ini dihilangkan jika tidak ada batasan sumber daya yang ditentukan.

`PullStartedAt`

Stempel waktu saat penarikan gambar kontainer pertama dimulai.

`PullStoppedAt`

Stempel waktu ketika penarikan gambar kontainer terakhir selesai.

`AvailabilityZone`

Availability Zone tempat tugas berada.

> **Note**
>
> Metadata Availability Zone hanya tersedia untuk tugas Fargate menggunakan platform versi 1.4 atau yang lebih baru (Linux) atau 1.0.0 atau yang lebih baru (Windows).

`Containers`

Daftar metadata kontainer untuk setiap kontainer yang terkait dengan tugas.

`DockerId`

ID Docker untuk wadah.

`Name`

Nama wadah seperti yang ditentukan dalam definisi tugas.

`DockerName`

Nama wadah yang dipasok ke Docker. Agen penampung Amazon ECS menghasilkan nama unik untuk penampung untuk menghindari tabrakan nama saat beberapa salinan dari definisi tugas yang sama dijalankan pada satu instance.

`Image`

Gambar untuk wadah.

`ImageID`

Intisari SHA-256 untuk gambar.

`Ports`

Port apa pun yang terbuka untuk wadah. Parameter ini dihilangkan jika tidak ada port yang terbuka.

`Labels`

Label apa pun diterapkan pada wadah. Parameter ini dihilangkan jika tidak ada label yang diterapkan.

`DesiredStatus`

Status yang diinginkan untuk wadah dari Amazon ECS.

`KnownStatus`

Status yang diketahui untuk wadah dari Amazon ECS.

`ExitCode`

Kode keluar untuk wadah. Parameter ini dihilangkan jika wadah belum keluar.

`Limits`

Batas sumber daya yang ditentukan pada tingkat kontainer, seperti CPU (dinyatakan dalam unit CPU) dan memori. Parameter ini dihilangkan jika tidak ada batasan sumber daya yang ditentukan.

`CreatedAt`

Cap waktu untuk saat wadah dibuat. Parameter ini dihilangkan jika wadah belum dibuat.

`StartedAt`

Cap waktu untuk saat wadah dimulai. Parameter ini dihilangkan jika wadah belum dimulai.

`FinishedAt`

Cap waktu untuk saat wadah berhenti. Parameter ini dihilangkan jika wadah belum berhenti.

`Type`

Jenis wadahnya. Wadah yang ditentukan dalam definisi tugas Anda adalah tipe`NORMAL`. Anda dapat mengabaikan jenis penampung lain, yang digunakan untuk penyediaan sumber daya tugas internal oleh agen penampung Amazon ECS.

`Networks`

Informasi jaringan untuk wadah, seperti mode jaringan dan alamat IP. Parameter ini dihilangkan jika tidak ada informasi jaringan yang didefinisikan.

`ClockDrift`

Informasi tentang perbedaan antara waktu referensi dan waktu sistem. Ini berlaku untuk sistem operasi Linux. Kemampuan ini menggunakan Amazon Time Sync Service untuk mengukur akurasi jam dan memberikan kesalahan jam yang terikat pada kontainer. Untuk informasi selengkapnya, lihat Mengatur waktu untuk instans Linux Anda di Panduan Pengguna Amazon EC2 untuk instans Linux.

`ReferenceTime`

Dasar akurasi jam. Amazon ECS menggunakan standar global Coordinated Universal Time (UTC) melalui NTP, misalnya. `2021-09-07T16:57:44Z`

`ClockErrorBound`

Ukuran kesalahan jam, didefinisikan sebagai offset ke UTC. Kesalahan ini adalah perbedaan milidetik antara waktu referensi dan waktu sistem.

`ClockSynchronizationStatus`

Menunjukkan apakah upaya sinkronisasi terbaru antara waktu sistem dan waktu referensi berhasil.

Nilai yang valid adalah `SYNCHRONIZED` dan `NOT_SYNCHRONIZED`.

`ExecutionStoppedAt`

Cap waktu untuk saat tugas `DesiredStatus` pindah ke`STOPPED`. Ini terjadi ketika wadah penting bergerak ke`STOPPED`.

Respons metadata tugas contoh

Respons JSON berikut adalah untuk tugas kontainer tunggal.

```
{
  "Cluster": "default",
  "TaskARN": "arn:aws:ecs:us-east-2:012345678910:task/9781c248-0edd-4cdb-9a93-
f63cb662a5d3",
  "Family": "nginx",
  "Revision": "5",
  "DesiredStatus": "RUNNING",
  "KnownStatus": "RUNNING",
  "Containers": [
    {
      "DockerId": "731a0d6a3b4210e2448339bc7015aaa79bfe4fa256384f4102db86ef94cbbc4c",
      "Name": "~internal~ecs~pause",
      "DockerName": "ecs-nginx-5-internalecspause-acc699c0cbf2d6d11700",
      "Image": "amazon/amazon-ecs-pause:0.1.0",
      "ImageID": "",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
```

```
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RESOURCES_PROVISIONED",
      "KnownStatus": "RESOURCES_PROVISIONED",
      "Limits": {
        "CPU": 0,
        "Memory": 0
      },
      "CreatedAt": "2018-02-01T20:55:08.366329616Z",
      "StartedAt": "2018-02-01T20:55:09.058354915Z",
      "Type": "CNI_PAUSE",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    },
    {
      "DockerId": "43481a6ce4842eec8fe72fc28500c6b52edcc0917f105b83379f88cac1ff3946",
      "Name": "nginx-curl",
      "DockerName": "ecs-nginx-5-nginx-curl-ccccb9f49db0dfe0d901",
      "Image": "nrdlngr/nginx-curl",
      "ImageID":
 "sha256:2e00ae64383cfc865ba0a2ba37f61b50a120d2d9378559dcd458dc0de47bc165",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "nginx-curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": {
        "CPU": 512,
        "Memory": 512
      },
```

```
      "CreatedAt": "2018-02-01T20:55:10.554941919Z",
      "StartedAt": "2018-02-01T20:55:11.064236631Z",
      "Type": "NORMAL",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    }
  ],
  "PullStartedAt": "2018-02-01T20:55:09.372495529Z",
  "PullStoppedAt": "2018-02-01T20:55:10.552018345Z",
  "AvailabilityZone": "us-east-2b"
}
```

Metadata tugas Amazon ECS tersedia untuk tugas di Fargate

Amazon ECS di Fargate menyediakan metode untuk mengambil berbagai metadata, metrik jaringan, dan statistik Docker tentang container Anda dan tugas yang menjadi bagiannya. Ini disebut sebagai titik akhir metadata tugas. Versi titik akhir metadata tugas berikut tersedia untuk Amazon ECS pada tugas Fargate:

- Titik akhir metadata tugas versi 4 — Tersedia untuk tugas yang menggunakan platform versi 1.4.0 atau yang lebih baru.
- Titik akhir metadata tugas versi 3 — Tersedia untuk tugas yang menggunakan platform versi 1.1.0 atau yang lebih baru.

Semua kontainer milik tugas yang diluncurkan dengan mode jaringan `awsvpc` menerima alamat IPv4 lokal dalam rentang alamat tautan lokal yang telah ditetapkan. Ketika sebuah kontainer melakukan kueri terhadap titik akhir metadata, agen kontainer dapat menentukan tugas yang menjadi milik kontainer berdasarkan alamat IP unik, serta metadata dan statistik untuk tugas tersebut dikembalikan.

Topik

- Titik akhir metadata tugas Amazon ECS versi 4 untuk tugas di Fargate
- Titik akhir metadata tugas Amazon ECS versi 3 untuk tugas di Fargate

Titik akhir metadata tugas Amazon ECS versi 4 untuk tugas di Fargate

> **Important**
>
> Jika Anda menggunakan tugas Amazon ECS yang dihosting di instans Amazon EC2, lihat titik akhir metadata tugas Amazon ECS.

Dimulai dengan versi platform Fargate`1.4.0`, variabel lingkungan bernama `ECS_CONTAINER_METADATA_URI_V4` disuntikkan ke setiap wadah dalam tugas. Ketika Anda melakukan kueri terhadap titik akhir metadata tugas versi 4, berbagai metadata tugas dan Statistik Docker tersedia untuk tugas.

Titik akhir metadata tugas versi 4 berfungsi seperti titik akhir versi 3 tetapi menyediakan metadata jaringan tambahan untuk kontainer dan tugas Anda. Metrik jaringan tambahan juga tersedia saat melakukan kueri terhadap titik akhir `/stats`.

Titik akhir metadata tugas aktif secara default untuk semua tugas Amazon ECS yang dijalankan pada versi platform yang menggunakan versi platform atau AWS Fargate yang lebih baru. `1.4.0`

> **Note**
>
> Untuk menghindari kebutuhan membuat versi titik akhir metadata tugas di masa mendatang, metadata tambahan dapat ditambahkan ke output versi 4. Kami tidak akan menghapus metadata yang ada atau mengubah nama bidang metadata.

Jalur titik akhir metadata tugas Fargate versi 4

Titik akhir metadata tugas berikut tersedia untuk kontainer:

`${ECS_CONTAINER_METADATA_URI_V4}`

Jalur ini mengembalikan metadata untuk kontainer.

`${ECS_CONTAINER_METADATA_URI_V4}/task`

Jalur ini mengembalikan metadata untuk tugas, termasuk daftar ID kontainer dan nama semua kontainer yang terkait dengan tugas. Untuk informasi selengkapnya tentang respons titik akhir ini, lihat Metadata tugas Amazon ECS v4 respons JSON untuk tugas di Fargate.

`${ECS_CONTAINER_METADATA_URI_V4}/stats`

Jalur ini mengembalikan statistik Docker untuk kontainer Docker. Untuk informasi selengkapnya tentang masing-masing statistik yang dikembalikan, lihat ContainerStatsdi dokumentasi Docker API.

> **Note**
>
> Tugas Amazon ECS AWS Fargate mengharuskan penampung berjalan selama ~1 detik sebelum mengembalikan statistik penampung.

`${ECS_CONTAINER_METADATA_URI_V4}/task/stats`

Jalur ini mengembalikan statistik Docker untuk semua kontainer yang terkait dengan tugas. Untuk informasi selengkapnya tentang masing-masing statistik yang dikembalikan, lihat ContainerStatsdi dokumentasi Docker API.

> **Note**
>
> Tugas Amazon ECS AWS Fargate mengharuskan penampung berjalan selama ~1 detik sebelum mengembalikan statistik penampung.

Metadata tugas Amazon ECS v4 respons JSON untuk tugas di Fargate

Metadata berikut dikembalikan dalam respons JSON titik akhir metadata tugas (`${ECS_CONTAINER_METADATA_URI_V4}/task`).

`Cluster`

Nama Sumber Daya Amazon (ARN) atau nama pendek dari cluster Amazon ECS tempat tugas tersebut berada.

`VPCID`

ID VPC dari instans penampung Amazon EC2. Bidang ini hanya muncul untuk instans Amazon EC2.

> **Note**
>
> `VPCID`Metadata hanya disertakan saat menggunakan versi `1.63.1` agen penampung Amazon ECS atau yang lebih baru.

`TaskARN`

Nama Sumber Daya Amazon (ARN) lengkap dari tugas yang menjadi milik wadah tersebut.

`Family`

Keluarga definisi tugas Amazon ECS untuk tugas tersebut.

`Revision`

Revisi definisi tugas Amazon ECS untuk tugas tersebut.

`DesiredStatus`

Status yang diinginkan untuk tugas dari Amazon ECS.

`KnownStatus`

Status yang diketahui untuk tugas dari Amazon ECS.

`Limits`

Batas sumber daya yang ditentukan pada tingkat tugas seperti CPU (dinyatakan dalam vCPU) dan memori. Parameter ini dihilangkan jika tidak ada batasan sumber daya yang ditentukan.

`PullStartedAt`

Stempel waktu saat penarikan gambar kontainer pertama dimulai.

`PullStoppedAt`

Stempel waktu ketika penarikan gambar kontainer terakhir selesai.

`AvailabilityZone`

Availability Zone tempat tugas berada.

> **Note**
>
> Metadata Availability Zone hanya tersedia untuk tugas Fargate menggunakan platform versi 1.4 atau yang lebih baru (Linux) atau 1.0.0 (Windows).

`LaunchType`

Jenis peluncuran yang digunakan tugas. Saat menggunakan penyedia kapasitas cluster, ini menunjukkan apakah tugas tersebut menggunakan infrastruktur Fargate atau EC2.

> **Note**
>
> `LaunchType`Metadata ini hanya disertakan saat menggunakan Amazon ECS Linux versi agen container atau yang lebih baru (Linux) `1.45.0` atau 1.0.0 atau yang lebih baru (Windows).

`EphemeralStorageMetrics`

Ukuran cadangan dan penggunaan penyimpanan sementara saat ini dari tugas ini.

> **Note**
>
> Fargate mencadangkan ruang pada disk. Ruang ini hanya digunakan oleh Fargate. Anda tidak akan dikenai biaya untuk ruang tersebut. Ruang ini tidak akan ditampilkan dalam metrik ini. Namun demikian, Anda dapat melihat penyimpanan tambahan ini di alat lain seperti `df`.

`Utilized`

Penggunaan penyimpanan sementara saat ini (dalam MiB) dari tugas ini.

`Reserved`

Penyimpanan sementara yang dicadangkan (dalam MiB) dari tugas ini. Ukuran penyimpanan sementara tidak dapat diubah dalam tugas yang sedang berjalan. Anda dapat menentukan `ephermalStorage` objek dalam definisi tugas Anda untuk mengubah jumlah penyimpanan sementara. `ephermalStorage`Ini ditentukan dalam GiB, bukan MiB. The `ephermalStorage` and the hanya `EphemeralStorageMetrics` tersedia di platform Fargate Linux versi 1.4.0 atau yang lebih baru.

`Containers`

Daftar metadata kontainer untuk setiap kontainer yang terkait dengan tugas.

`DockerId`

ID Docker untuk wadah.

Saat Anda menggunakan Fargate, id adalah hex 32 digit diikuti dengan angka 10 digit.

`Name`

Nama wadah seperti yang ditentukan dalam definisi tugas.

`DockerName`

Nama wadah yang dipasok ke Docker. Agen penampung Amazon ECS menghasilkan nama unik untuk penampung untuk menghindari tabrakan nama saat beberapa salinan dari definisi tugas yang sama dijalankan pada satu instance.

`Image`

Gambar untuk wadah.

`ImageID`

Intisari SHA-256 untuk gambar.

`Ports`

Port apa pun yang terbuka untuk wadah. Parameter ini dihilangkan jika tidak ada port yang terbuka.

`Labels`

Label apa pun diterapkan pada wadah. Parameter ini dihilangkan jika tidak ada label yang diterapkan.

`DesiredStatus`

Status yang diinginkan untuk wadah dari Amazon ECS.

`KnownStatus`

Status yang diketahui untuk wadah dari Amazon ECS.

`ExitCode`

Kode keluar untuk wadah. Parameter ini dihilangkan jika wadah belum keluar.

`Limits`

Batas sumber daya yang ditentukan pada tingkat kontainer seperti CPU (dinyatakan dalam unit CPU) dan memori. Parameter ini dihilangkan jika tidak ada batasan sumber daya yang ditentukan.

`CreatedAt`

Cap waktu untuk saat wadah dibuat. Parameter ini dihilangkan jika wadah belum dibuat.

`StartedAt`

Cap waktu untuk saat wadah dimulai. Parameter ini dihilangkan jika wadah belum dimulai.

`FinishedAt`

Cap waktu untuk saat wadah berhenti. Parameter ini dihilangkan jika wadah belum berhenti.

`Type`

Jenis wadahnya. Wadah yang ditentukan dalam definisi tugas Anda adalah tipe`NORMAL`. Anda dapat mengabaikan jenis penampung lain, yang digunakan untuk penyediaan sumber daya tugas internal oleh agen penampung Amazon ECS.

`LogDriver`

Driver log yang digunakan wadah.

> **Note**
>
> `LogDriver`Metadata ini hanya disertakan saat menggunakan Amazon ECS Linux container agent versi `1.45.0` atau yang lebih baru.

`LogOptions`

Opsi driver log yang ditentukan untuk wadah.

> **Note**
>
> `LogOptions`Metadata ini hanya disertakan saat menggunakan Amazon ECS Linux container agent versi `1.45.0` atau yang lebih baru.

`ContainerARN`

Nama Sumber Daya Amazon (ARN) lengkap dari wadah.

> **Note**
>
> `ContainerARN`Metadata ini hanya disertakan saat menggunakan Amazon ECS Linux container agent versi `1.45.0` atau yang lebih baru.

`Networks`

Informasi jaringan untuk wadah, seperti mode jaringan dan alamat IP. Parameter ini dihilangkan jika tidak ada informasi jaringan yang didefinisikan.

`Snapshotter`

snapshotterYang digunakan oleh containerd untuk mengunduh gambar kontainer ini. Nilai yang valid adalah`overlayfs`, yang merupakan default, dan`soci`, digunakan saat lazy loading dengan indeks SOCI. Parameter ini hanya tersedia untuk tugas-tugas yang berjalan pada versi platform Linux`1.4.0`.

`ClockDrift`

Informasi tentang perbedaan antara waktu referensi dan waktu sistem. Kemampuan ini menggunakan Amazon Time Sync Service untuk mengukur akurasi jam dan memberikan kesalahan jam yang terikat pada kontainer. Untuk informasi selengkapnya, lihat Mengatur waktu untuk instans Linux Anda di Panduan Pengguna Amazon EC2 untuk instans Linux.

`ReferenceTime`

Dasar akurasi jam. Amazon ECS menggunakan standar global Coordinated Universal Time (UTC) melalui NTP, misalnya. `2021-09-07T16:57:44Z`

`ClockErrorBound`

Ukuran kesalahan jam, didefinisikan sebagai offset ke UTC. Kesalahan ini adalah perbedaan milidetik antara waktu referensi dan waktu sistem.

`ClockSynchronizationStatus`

Menunjukkan apakah upaya sinkronisasi terbaru antara waktu sistem dan waktu referensi berhasil.

Nilai yang valid adalah `SYNCHRONIZED` dan `NOT_SYNCHRONIZED`.

`ExecutionStoppedAt`

Cap waktu untuk saat tugas `DesiredStatus` pindah ke`STOPPED`. Ini terjadi ketika wadah penting bergerak ke`STOPPED`.

Contoh metadata tugas Amazon ECS v4 untuk tugas di Fargate

Contoh berikut menunjukkan contoh keluaran dari titik akhir metadata tugas untuk tugas Amazon ECS yang dijalankan. AWS Fargate

Dari wadah, Anda dapat menggunakan curl diikuti oleh titik akhir data meta tugas untuk menanyakan titik akhir misalnya. `curl ${ECS_CONTAINER_METADATA_URI_V4}/task`

Respons metadata kontainer contoh

Ketika melakukan kueri terhadap titik akhir `${ECS_CONTAINER_METADATA_URI_V4}`, hanya metadata kontainer tersebut yang dikembalikan kepada Anda. Berikut ini adalah output contoh.

```
{
    "DockerId": "cd189a933e5849daa93386466019ab50-2495160603",
    "Name": "curl",
    "DockerName": "curl",
    "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
    "ImageID":
 "sha256:25f3695bedfb454a50f12d127839a68ad3caf91e451c1da073db34c542c4d2cb",
    "Labels": {
        "com.amazonaws.ecs.cluster": "arn:aws:ecs:us-west-2:111122223333:cluster/
default",
        "com.amazonaws.ecs.container-name": "curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/default/
cd189a933e5849daa93386466019ab50",
        "com.amazonaws.ecs.task-definition-family": "curltest",
        "com.amazonaws.ecs.task-definition-version": "2"
    },
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "Limits": {
        "CPU": 10,
        "Memory": 128
    },
    "CreatedAt": "2020-10-08T20:09:11.44527186Z",
    "StartedAt": "2020-10-08T20:09:11.44527186Z",
    "Type": "NORMAL",
```

```
    "Networks": [
        {
            "NetworkMode": "awsvpc",
            "IPv4Addresses": [
                "192.0.2.3"
            ],
            "AttachmentIndex": 0,
            "MACAddress": "0a:de:f6:10:51:e5",
            "IPv4SubnetCIDRBlock": "192.0.2.0/24",
            "DomainNameServers": [
                "192.0.2.2"
            ],
            "DomainNameSearchList": [
                "us-west-2.compute.internal"
            ],
            "PrivateDNSName": "ip-10-0-0-222.us-west-2.compute.internal",
            "SubnetGatewayIpv4Address": "192.0.2.0/24"
        }
    ],
    "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/05966557-
f16c-49cb-9352-24b3a0dcd0e1",
    "LogOptions": {
        "awslogs-create-group": "true",
        "awslogs-group": "/ecs/containerlogs",
        "awslogs-region": "us-west-2",
        "awslogs-stream": "ecs/curl/cd189a933e5849daa93386466019ab50"
    },
    "LogDriver": "awslogs",
    "Snapshotter": "overlayfs"
}
```

Contoh metadata tugas Amazon ECS v4 untuk tugas di Fargate

Ketika melakukan kueri terhadap titik akhir `${ECS_CONTAINER_METADATA_URI_V4}/task`, metadata dari tugas yang berisi bagian kontainer dikembalikan kepada Anda. Berikut ini adalah output contoh.

```
{
  "Cluster": "arn:aws:ecs:us-east-1:123456789012:cluster/clusterName",
  "TaskARN": "arn:aws:ecs:us-east-1:123456789012:task/MyEmptyCluster/
bfa2636268144d039771334145e490c5",
  "Family": "sample-fargate",
  "Revision": "5",
```

```
  "DesiredStatus": "RUNNING",
  "KnownStatus": "RUNNING",
  "Limits": {
    "CPU": 0.25,
    "Memory": 512
  },
  "PullStartedAt": "2023-07-21T15:45:33.532811081Z",
  "PullStoppedAt": "2023-07-21T15:45:38.541068435Z",
  "AvailabilityZone": "us-east-1d",
  "Containers": [
    {
      "DockerId": "bfa2636268144d039771334145e490c5-1117626119",
      "Name": "curl-image",
      "DockerName": "curl-image",
      "Image": "curlimages/curl",
      "ImageID":
 "sha256:daf3f46a2639c1613b25e85c9ee4193af8a1d538f92483d67f9a3d7f21721827",
      "Labels": {
        "com.amazonaws.ecs.cluster": "arn:aws:ecs:us-east-1:123456789012:cluster/
MyEmptyCluster",
        "com.amazonaws.ecs.container-name": "curl-image",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-east-1:123456789012:task/
MyEmptyCluster/bfa2636268144d039771334145e490c5",
        "com.amazonaws.ecs.task-definition-family": "sample-fargate",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": { "CPU": 128 },
      "CreatedAt": "2023-07-21T15:45:44.91368314Z",
      "StartedAt": "2023-07-21T15:45:44.91368314Z",
      "Type": "NORMAL",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": ["172.31.42.189"],
          "AttachmentIndex": 0,
          "MACAddress": "0e:98:9f:33:76:d3",
          "IPv4SubnetCIDRBlock": "172.31.32.0/20",
          "DomainNameServers": ["172.31.0.2"],
          "DomainNameSearchList": ["ec2.internal"],
          "PrivateDNSName": "ip-172-31-42-189.ec2.internal",
          "SubnetGatewayIpv4Address": "172.31.32.1/20"
        }
```

```
      ],
      "ContainerARN": "arn:aws:ecs:us-east-1:123456789012:container/MyEmptyCluster/
bfa2636268144d039771334145e490c5/da6cccf7-1178-400c-afdf-7536173ee209",
      "Snapshotter": "overlayfs"
    },
    {
      "DockerId": "bfa2636268144d039771334145e490c5-3681984407",
      "Name": "fargate-app",
      "DockerName": "fargate-app",
      "Image": "public.ecr.aws/docker/library/httpd:latest",
      "ImageID":
 "sha256:8059bdd0058510c03ae4c808de8c4fd2c1f3c1b6d9ea75487f1e5caa5ececa02",
      "Labels": {
        "com.amazonaws.ecs.cluster": "arn:aws:ecs:us-east-1:123456789012:cluster/
MyEmptyCluster",
        "com.amazonaws.ecs.container-name": "fargate-app",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-east-1:123456789012:task/
MyEmptyCluster/bfa2636268144d039771334145e490c5",
        "com.amazonaws.ecs.task-definition-family": "sample-fargate",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": { "CPU": 2 },
      "CreatedAt": "2023-07-21T15:45:44.954460255Z",
      "StartedAt": "2023-07-21T15:45:44.954460255Z",
      "Type": "NORMAL",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": ["172.31.42.189"],
          "AttachmentIndex": 0,
          "MACAddress": "0e:98:9f:33:76:d3",
          "IPv4SubnetCIDRBlock": "172.31.32.0/20",
          "DomainNameServers": ["172.31.0.2"],
          "DomainNameSearchList": ["ec2.internal"],
          "PrivateDNSName": "ip-172-31-42-189.ec2.internal",
          "SubnetGatewayIpv4Address": "172.31.32.1/20"
        }
      ],
      "ContainerARN": "arn:aws:ecs:us-east-1:123456789012:container/MyEmptyCluster/
bfa2636268144d039771334145e490c5/f65b461d-aa09-4acb-a579-9785c0530cbc",
      "Snapshotter": "overlayfs"
    }
```

```
  ],
  "LaunchType": "FARGATE",
  "ClockDrift": {
    "ClockErrorBound": 0.446931,
    "ReferenceTimestamp": "2023-07-21T16:09:17Z",
    "ClockSynchronizationStatus": "SYNCHRONIZED"
  },
  "EphemeralStorageMetrics": {
    "Utilized": 261,
    "Reserved": 20496
  }
}
```

Respons statistik tugas contoh

Ketika melakukan kueri terhadap titik akhir `${ECS_CONTAINER_METADATA_URI_V4}/task/stats`, metrik jaringan dari tugas yang berisi bagian kontainer dikembalikan kepada Anda. Berikut ini adalah output contoh.

```
{
  "3d1f891cded94dc795608466cce8ddcf-464223573": {
    "read": "2020-10-08T21:24:44.938937019Z",
    "preread": "2020-10-08T21:24:34.938633969Z",
    "pids_stats": {},
    "blkio_stats": {
      "io_service_bytes_recursive": [
        {
          "major": 202,
          "minor": 26368,
          "op": "Read",
          "value": 638976
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Write",
          "value": 0
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Sync",
          "value": 638976
```

```
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Async",
          "value": 0
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Total",
          "value": 638976
        }
      ],
      "io_serviced_recursive": [
        {
          "major": 202,
          "minor": 26368,
          "op": "Read",
          "value": 12
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Write",
          "value": 0
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Sync",
          "value": 12
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Async",
          "value": 0
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Total",
          "value": 12
```

```
        }
      ],
      "io_queue_recursive": [],
      "io_service_time_recursive": [],
      "io_wait_time_recursive": [],
      "io_merged_recursive": [],
      "io_time_recursive": [],
      "sectors_recursive": []
    },
    "num_procs": 0,
    "storage_stats": {},
    "cpu_stats": {
      "cpu_usage": {
        "total_usage": 1137691504,
        "percpu_usage": [
          696479228,
          441212276,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0
        ],
        "usage_in_kernelmode": 80000000,
        "usage_in_usermode": 810000000
      },
      "system_cpu_usage": 9393210000000,
      "online_cpus": 2,
      "throttling_data": {
        "periods": 0,
        "throttled_periods": 0,
        "throttled_time": 0
      }
    },
    "precpu_stats": {
      "cpu_usage": {
```

```
        "total_usage": 1136624601,
        "percpu_usage": [
          695639662,
          440984939,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0
        ],
        "usage_in_kernelmode": 80000000,
        "usage_in_usermode": 810000000
      },
      "system_cpu_usage": 9373330000000,
      "online_cpus": 2,
      "throttling_data": {
        "periods": 0,
        "throttled_periods": 0,
        "throttled_time": 0
      }
    },
    "memory_stats": {
      "usage": 6504448,
      "max_usage": 8458240,
      "stats": {
        "active_anon": 1675264,
        "active_file": 557056,
        "cache": 651264,
        "dirty": 0,
        "hierarchical_memory_limit": 536870912,
        "hierarchical_memsw_limit": 9223372036854772000,
        "inactive_anon": 0,
        "inactive_file": 3088384,
        "mapped_file": 430080,
        "pgfault": 11034,
        "pgmajfault": 5,
```

```
        "pgpgin": 8436,
        "pgpgout": 7137,
        "rss": 4669440,
        "rss_huge": 0,
        "total_active_anon": 1675264,
        "total_active_file": 557056,
        "total_cache": 651264,
        "total_dirty": 0,
        "total_inactive_anon": 0,
        "total_inactive_file": 3088384,
        "total_mapped_file": 430080,
        "total_pgfault": 11034,
        "total_pgmajfault": 5,
        "total_pgpgin": 8436,
        "total_pgpgout": 7137,
        "total_rss": 4669440,
        "total_rss_huge": 0,
        "total_unevictable": 0,
        "total_writeback": 0,
        "unevictable": 0,
        "writeback": 0
      },
      "limit": 9223372036854772000
    },
    "name": "curltest",
    "id": "3d1f891cded94dc795608466cce8ddcf-464223573",
    "networks": {
      "eth1": {
        "rx_bytes": 2398415937,
        "rx_packets": 1898631,
        "rx_errors": 0,
        "rx_dropped": 0,
        "tx_bytes": 1259037719,
        "tx_packets": 428002,
        "tx_errors": 0,
        "tx_dropped": 0
      }
    },
    "network_rate_stats": {
      "rx_bytes_per_sec": 43.298687872232854,
      "tx_bytes_per_sec": 215.39347269466413
    }
  }
```

```
}
```

Titik akhir metadata tugas Amazon ECS versi 3 untuk tugas di Fargate

> **Important**
>
> Titik akhir metadata tugas versi 3 tidak lagi dipertahankan secara aktif. Kami menyarankan Anda memperbarui titik akhir metadata tugas versi 4 untuk mendapatkan informasi titik akhir metadata terbaru. Untuk informasi selengkapnya, lihat the section called “Titik akhir metadata tugas versi 4 untuk tugas di Fargate”.

Dimulai dengan versi platform Fargate`1.1.0`, variabel lingkungan bernama `ECS_CONTAINER_METADATA_URI` disuntikkan ke setiap wadah dalam tugas. Ketika Anda melakukan kueri terhadap titik akhir metadata tugas versi 3, berbagai metadata tugas dan Statistik Docker tersedia untuk tugas.

Fitur titik akhir metadata tugas diaktifkan secara default untuk tugas Amazon ECS yang dihosting di Fargate yang menggunakan versi platform atau yang lebih baru. `1.1.0` Untuk informasi selengkapnya, lihat Versi platform Fargate Linux.

Jalur titik akhir metadata tugas untuk tugas di Fargate

Titik akhir API berikut tersedia untuk kontainer:

`${ECS_CONTAINER_METADATA_URI}`

Jalur ini mengembalikan JSON metadata untuk kontainer.

`${ECS_CONTAINER_METADATA_URI}/task`

Jalur ini mengembalikan JSON metadata untuk tugas, termasuk daftar ID kontainer dan nama semua kontainer yang terkait dengan tugas. Untuk informasi selengkapnya tentang respons titik akhir ini, lihat Metadata tugas Amazon ECS v3 Respons JSON untuk tugas di Fargate.

`${ECS_CONTAINER_METADATA_URI}/stats`

Jalur ini mengembalikan JSON statistik Docker untuk kontainer Docker tertentu. Untuk informasi selengkapnya tentang masing-masing statistik yang dikembalikan, lihat ContainerStatsdi dokumentasi Docker API.

`${ECS_CONTAINER_METADATA_URI}/task/stats`

Jalur ini mengembalikan JSON statistik Docker untuk semua kontainer yang terkait dengan tugas. Untuk informasi selengkapnya tentang masing-masing statistik yang dikembalikan, lihat ContainerStatsdi dokumentasi Docker API.

Metadata tugas Amazon ECS v3 Respons JSON untuk tugas di Fargate

Informasi berikut dikembalikan dari respons JSON titik akhir metadata tugas (`${ECS_CONTAINER_METADATA_URI}/task`).

`Cluster`

Nama Sumber Daya Amazon (ARN) atau nama pendek dari cluster Amazon ECS tempat tugas tersebut berada.

`TaskARN`

Nama Sumber Daya Amazon (ARN) lengkap dari tugas yang menjadi milik wadah tersebut.

`Family`

Keluarga definisi tugas Amazon ECS untuk tugas tersebut.

`Revision`

Revisi definisi tugas Amazon ECS untuk tugas tersebut.

`DesiredStatus`

Status yang diinginkan untuk tugas dari Amazon ECS.

`KnownStatus`

Status yang diketahui untuk tugas dari Amazon ECS.

`Limits`

Batas sumber daya yang ditentukan pada tingkat tugas, seperti CPU (dinyatakan dalam vCPU) dan memori. Parameter ini dihilangkan jika tidak ada batasan sumber daya yang ditentukan.

`PullStartedAt`

Stempel waktu saat penarikan gambar kontainer pertama dimulai.

`PullStoppedAt`

Stempel waktu ketika penarikan gambar kontainer terakhir selesai.

`AvailabilityZone`

Availability Zone tempat tugas berada.

> **Note**
>
> Metadata Availability Zone hanya tersedia untuk tugas Fargate menggunakan platform versi 1.4 atau yang lebih baru (Linux) atau 1.0.0 atau yang lebih baru (Windows).

`Containers`

Daftar metadata kontainer untuk setiap kontainer yang terkait dengan tugas.

`DockerId`

ID Docker untuk wadah.

`Name`

Nama wadah seperti yang ditentukan dalam definisi tugas.

`DockerName`

Nama wadah yang dipasok ke Docker. Agen penampung Amazon ECS menghasilkan nama unik untuk penampung untuk menghindari tabrakan nama saat beberapa salinan dari definisi tugas yang sama dijalankan pada satu instance.

`Image`

Gambar untuk wadah.

`ImageID`

Intisari SHA-256 untuk gambar.

`Ports`

Port apa pun yang terbuka untuk wadah. Parameter ini dihilangkan jika tidak ada port yang terbuka.

`Labels`

Label apa pun diterapkan pada wadah. Parameter ini dihilangkan jika tidak ada label yang diterapkan.

`DesiredStatus`

Status yang diinginkan untuk wadah dari Amazon ECS.

`KnownStatus`

Status yang diketahui untuk wadah dari Amazon ECS.

`ExitCode`

Kode keluar untuk wadah. Parameter ini dihilangkan jika wadah belum keluar.

`Limits`

Batas sumber daya yang ditentukan pada tingkat kontainer, seperti CPU (dinyatakan dalam unit CPU) dan memori. Parameter ini dihilangkan jika tidak ada batasan sumber daya yang ditentukan.

`CreatedAt`

Cap waktu untuk saat wadah dibuat. Parameter ini dihilangkan jika wadah belum dibuat.

`StartedAt`

Cap waktu untuk saat wadah dimulai. Parameter ini dihilangkan jika wadah belum dimulai.

`FinishedAt`

Cap waktu untuk saat wadah berhenti. Parameter ini dihilangkan jika wadah belum berhenti.

`Type`

Jenis wadahnya. Wadah yang ditentukan dalam definisi tugas Anda adalah tipe`NORMAL`. Anda dapat mengabaikan jenis penampung lain, yang digunakan untuk penyediaan sumber daya tugas internal oleh agen penampung Amazon ECS.

`Networks`

Informasi jaringan untuk wadah, seperti mode jaringan dan alamat IP. Parameter ini dihilangkan jika tidak ada informasi jaringan yang didefinisikan.

`ClockDrift`

Informasi tentang perbedaan antara waktu referensi dan waktu sistem. Ini berlaku untuk sistem operasi Linux. Kemampuan ini menggunakan Amazon Time Sync Service untuk mengukur akurasi jam dan memberikan kesalahan jam yang terikat pada kontainer. Untuk informasi selengkapnya, lihat Mengatur waktu untuk instans Linux Anda di Panduan Pengguna Amazon EC2 untuk instans Linux.

`ReferenceTime`

Dasar akurasi jam. Amazon ECS menggunakan standar global Coordinated Universal Time (UTC) melalui NTP, misalnya. `2021-09-07T16:57:44Z`

`ClockErrorBound`

Ukuran kesalahan jam, didefinisikan sebagai offset ke UTC. Kesalahan ini adalah perbedaan milidetik antara waktu referensi dan waktu sistem.

`ClockSynchronizationStatus`

Menunjukkan apakah upaya sinkronisasi terbaru antara waktu sistem dan waktu referensi berhasil.

Nilai yang valid adalah `SYNCHRONIZED` dan `NOT_SYNCHRONIZED`.

`ExecutionStoppedAt`

Cap waktu untuk saat tugas `DesiredStatus` pindah ke`STOPPED`. Ini terjadi ketika wadah penting bergerak ke`STOPPED`.

Contoh metadata tugas Amazon ECS v3 untuk tugas di Fargate

Respons JSON berikut adalah untuk tugas kontainer tunggal.

```
{
  "Cluster": "default",
  "TaskARN": "arn:aws:ecs:us-east-2:012345678910:task/9781c248-0edd-4cdb-9a93-
f63cb662a5d3",
  "Family": "nginx",
  "Revision": "5",
  "DesiredStatus": "RUNNING",
  "KnownStatus": "RUNNING",
  "Containers": [
    {
      "DockerId": "731a0d6a3b4210e2448339bc7015aaa79bfe4fa256384f4102db86ef94cbbc4c",
      "Name": "~internal~ecs~pause",
      "DockerName": "ecs-nginx-5-internalecspause-acc699c0cbf2d6d11700",
      "Image": "amazon/amazon-ecs-pause:0.1.0",
      "ImageID": "",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
```

```
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RESOURCES_PROVISIONED",
      "KnownStatus": "RESOURCES_PROVISIONED",
      "Limits": {
        "CPU": 0,
        "Memory": 0
      },
      "CreatedAt": "2018-02-01T20:55:08.366329616Z",
      "StartedAt": "2018-02-01T20:55:09.058354915Z",
      "Type": "CNI_PAUSE",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    },
    {
      "DockerId": "43481a6ce4842eec8fe72fc28500c6b52edcc0917f105b83379f88cac1ff3946",
      "Name": "nginx-curl",
      "DockerName": "ecs-nginx-5-nginx-curl-ccccb9f49db0dfe0d901",
      "Image": "nrdlngr/nginx-curl",
      "ImageID":
 "sha256:2e00ae64383cfc865ba0a2ba37f61b50a120d2d9378559dcd458dc0de47bc165",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "nginx-curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": {
        "CPU": 512,
        "Memory": 512
      },
```

```
      "CreatedAt": "2018-02-01T20:55:10.554941919Z",
      "StartedAt": "2018-02-01T20:55:11.064236631Z",
      "Type": "NORMAL",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    }
  ],
  "PullStartedAt": "2018-02-01T20:55:09.372495529Z",
  "PullStoppedAt": "2018-02-01T20:55:10.552018345Z",
  "AvailabilityZone": "us-east-2b"
}
```

Introspeksi wadah Amazon ECS

Agen penampung Amazon ECS menyediakan operasi API untuk mengumpulkan detail tentang instance container tempat agen dijalankan dan tugas terkait yang berjalan pada instance tersebut. Anda dapat menggunakan curl perintah dari dalam instance container untuk menanyakan agen penampung Amazon ECS (port 51678) dan mengembalikan metadata instans kontainer atau informasi tugas.

> **Important**
>
> Instance container Anda harus memiliki peran IAM yang memungkinkan akses ke Amazon ECS untuk mengambil metadata. Untuk informasi selengkapnya, lihat Peran IAM instans wadah Amazon ECS.

Untuk melihat metadata instans kontainer, masuk ke instans kontainer Anda melalui SSH dan jalankan perintah berikut. Metadata mencakup ID instance container, cluster Amazon ECS tempat instance container terdaftar, dan informasi versi agen penampung Amazon ECS.

```
curl -s http://localhost:51678/v1/metadata | python -mjson.tool
```

Output:

```
{
    "Cluster": "cluster_name",
    "ContainerInstanceArn": "arn:aws:ecs:region:aws_account_id:container-
instance/cluster_name/container_instance_id",
    "Version": "Amazon ECS Agent - v1.30.0 (02ff320c)"
}
```

Untuk melihat informasi tentang semua tugas yang berjalan pada instans kontainer, masuk ke instans kontainer Anda melalui SSH dan jalankan perintah berikut:

```
curl http://localhost:51678/v1/tasks
```

Output:

```
{
    "Tasks": [
        {
            "Arn": "arn:aws:ecs:us-west-2:012345678910:task/default/example5-58ff-46c9-
ae05-543f8example",
            "DesiredStatus": "RUNNING",
            "KnownStatus": "RUNNING",
            "Family": "hello_world",
            "Version": "8",
            "Containers": [
                {
          "DockerId":
 "9581a69a761a557fbfce1d0f6745e4af5b9dbfb86b6b2c5c4df156f1a5932ff1",
          "DockerName": "ecs-hello_world-8-mysql-fcae8ac8f9f1d89d8301",
          "Name": "mysql",
          "CreatedAt": "2023-10-08T20:09:11.44527186Z",
          "StartedAt": "2023-10-08T20:09:11.44527186Z",
          "ImageID":
 "sha256:2ae34abc2ed0a22e280d17e13f9c01aaf725688b09b7a1525d1a2750e2c0d1de"
        },
        {
          "DockerId":
 "bf25c5c5b2d4dba68846c7236e75b6915e1e778d31611e3c6a06831e39814a15",
          "DockerName": "ecs-hello_world-8-wordpress-e8bfddf9b488dff36c00",
          "Name": "wordpress"
        }
    ]
```

```
}
    ]
}
```

Anda dapat melihat informasi untuk tugas tertentu yang berjalan pada instans kontainer. Untuk menentukan tugas atau kontainer tertentu, tambahkan salah satu dari berikut pada permintaan:

- ARN tugas (`?taskarn=`*`task_arn`*)
- ID Docker untuk kontainer (`?dockerid=`*`docker_id`*)

Untuk mendapatkan informasi tugas dengan Docker ID kontainer, masuk ke instans kontainer Anda melalui SSH dan jalankan perintah berikut.

> **Note**
>
> Agen penampung Amazon ECS sebelum versi 1.14.2 memerlukan ID kontainer Docker lengkap untuk API introspeksi, bukan versi pendek yang ditampilkan. docker ps Anda bisa mendapatkan ID Docker lengkap untuk kontainer dengan menjalankan perintah docker ps --no-trunc pada instans kontainer.

```
curl http://localhost:51678/v1/tasks?dockerid=79c796ed2a7f
```

Output:

```
{
    "Arn": "arn:aws:ecs:us-west-2:012345678910:task/default/e01d58a8-151b-40e8-
bc01-22647b9ecfec",
    "Containers": [
        {
            "DockerId":
 "79c796ed2a7f864f485c76f83f3165488097279d296a7c05bd5201a1c69b2920",
            "DockerName": "ecs-nginx-efs-2-nginx-9ac0808dd0afa495f001",
            "Name": "nginx",
            "CreatedAt": "2023-10-08T20:09:11.44527186Z",
            "StartedAt": "2023-10-08T20:09:11.44527186Z",
            "ImageID":
 "sha256:2ae34abc2ed0a22e280d17e13f9c01aaf725688b09b7a1525d1a2750e2c0d1de"
```

```
        }
    ],
    "DesiredStatus": "RUNNING",
    "Family": "nginx-efs",
    "KnownStatus": "RUNNING",
    "Version": "2"
}
```

AWS layanan terintegrasi dengan Amazon ECS

Amazon ECS bekerja sama dengan AWS layanan lain untuk memberikan solusi tambahan untuk tantangan bisnis Anda. Topik ini mengidentifikasi layanan yang menggunakan Amazon ECS untuk menambahkan fungsionalitas, atau layanan yang digunakan Amazon ECS untuk melakukan tugas.

Daftar Isi

Menggunakan Amazon ECR dengan Amazon ECS

Amazon ECR adalah layanan registri AWS Docker terkelola. Pelanggan dapat menggunakan CLI Docker yang sudah dikenal untuk mendorong, menarik, dan mengelola gambar. Amazon ECR menyediakan registri yang aman, terukur, dan andal. Amazon ECR mendukung repositori Docker pribadi dengan izin berbasis sumber daya menggunakan IAM AWS sehingga pengguna tertentu atau instans Amazon EC2 dapat mengakses repositori dan gambar. Developer dapat menggunakan Docker CLI untuk mengotorisasi dan mengelola citra.

Untuk informasi selengkapnya tentang cara membuat repositori, mendorong dan menarik gambar dari Amazon ECR, dan mengatur kontrol akses pada repositori Anda, lihat Panduan Pengguna Amazon Elastic Container Registry.

Menggunakan Gambar Amazon ECR dengan Amazon ECS

Anda dapat menggunakan gambar ECR Anda dengan Amazon ECS, tetapi Anda harus memenuhi prasyarat berikut.

- Instans penampung Anda harus menggunakan setidaknya versi 1.7.0 dari agen penampung Amazon ECS. Versi terbaru dari Amazon ECS — AMI yang dioptimalkan mendukung gambar ECR dalam definisi tugas. Untuk informasi selengkapnya, termasuk ID AMI yang dioptimalkan Amazon ECS terbaru, lihat Versi Agen Kontainer Amazon ECS di Panduan Pengembang Layanan Kontainer Elastis Amazon.
- Peran instans container Amazon ECS (`ecsInstanceRole`) yang Anda gunakan dengan instance container harus memiliki izin kebijakan IAM berikut untuk Amazon ECR.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        }
    ]
}
```

Jika Anda menggunakan kebijakan terkelola `AmazonEC2ContainerServiceforEC2Role` untuk instans kontainer Anda, maka peran anda memiliki izin yang tepat. Untuk memeriksa apakah peran Anda mendukung Amazon ECR, lihat Peran IAM Instance Amazon ECS Container di Panduan Pengembang Layanan Kontainer Elastis Amazon.

- Dalam ketentuan tugas ECS Anda, pastikan bahwa Anda menggunakan penamaan penuh `registry/repository:tag` untuk citra ECR Anda. Misalnya, *`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`my-web-app`*`:`*`latest`*.

AWS Deep Learning Containers di Amazon ECS

AWS Deep Learning Containers menyediakan satu set gambar Docker untuk melatih dan menyajikan model di TensorFlow dan Apache MXNet (Inkubasi) di Amazon ECS. Deep Learning Containers memungkinkan lingkungan yang dioptimalkan dengan TensorFlow library NVIDIA CUDA (untuk instans GPU), dan Intel MKL (untuk instance CPU). Gambar kontainer untuk Deep Learning Containers tersedia di Amazon ECR untuk referensi dalam definisi tugas Amazon ECS. Anda dapat menggunakan Deep Learning Containers bersama Amazon Elastic Inference untuk menurunkan biaya inferensi Anda.

Untuk mulai menggunakan Deep Learning Containers tanpa Elastic Inference di Amazon ECS, lihat Deep Learning Containers di Amazon ECS di Panduan Pengembang.AWS Deep Learning AMI

Deep Learning Containers dengan Elastic Inference di Amazon ECS

> **Note**
>
> Mulai 15 April 2023, tidak AWS akan memasukkan pelanggan baru ke Amazon Elastic Inference (EI), dan akan membantu pelanggan saat ini memigrasikan beban kerja mereka ke opsi yang menawarkan harga dan kinerja yang lebih baik. Setelah 15 April 2023, pelanggan baru tidak akan dapat meluncurkan instans dengan akselerator Amazon EI di Amazon, Amazon ECS, atau SageMaker Amazon EC2. Namun, pelanggan yang telah menggunakan Amazon EI setidaknya sekali selama periode 30 hari terakhir dianggap sebagai pelanggan saat ini dan akan dapat terus menggunakan layanan ini.

AWS Deep Learning Containers menyediakan satu set image Docker untuk melayani model di TensorFlow dan Apache MXNet (Incubating) yang memanfaatkan akselerator Amazon Elastic Inference. Amazon ECS menyediakan parameter definisi tugas untuk memasang akselerator Elastic Inference ke container Anda. Saat Anda menentukan jenis akselerator Elastic Inference dalam definisi tugas, Amazon ECS mengelola siklus hidup, dan konfigurasi untuk, akselerator. Peran terkait layanan Amazon ECS diperlukan saat menggunakan fitur ini. Untuk informasi selengkapnya tentang akselerator Elastic Inference, lihat Dasar-dasar Amazon Elastic Inference.

> **Important**
>
> Instans penampung Amazon ECS Anda memerlukan setidaknya versi `1.30.0` agen kontainer. Untuk informasi tentang memeriksa versi agen Anda dan memperbarui ke versi terbaru, lihat Memperbarui agen kontainer Amazon ECS.

Untuk mulai menggunakan Deep Learning Containers dengan Elastic Inference di Amazon ECS, lihat Deep Learning Containers with Elastic Inference on Amazon ECS di Amazon Elastic Inference di Amazon Elastic Inference Developer Guide.

Menggunakan Notifikasi Pengguna AWS dengan Amazon ECS

Anda dapat menggunakan Notifikasi Pengguna AWSuntuk mengatur saluran pengiriman untuk mendapatkan pemberitahuan tentang peristiwa Amazon ECS. Anda akan menerima notifikasi saat ada sebuah peristiwa yang cocok dengan sebuah aturan yang Anda tentukan. Anda dapat menerima

notifikasi untuk peristiwa melalui beberapa saluran, termasuk email, notifikasi obrolan AWS Chatbot, atau notifikasi push AWS Console Mobile Application. Anda juga dapat melihat notifikasi di Pusat Pemberitahuan Konsol. Notifikasi Pengguna mendukung agregasi, yang dapat mengurangi jumlah pemberitahuan yang Anda terima selama acara tertentu.

Contoh

Pola peristiwa berikut cocok dengan perubahan status tugas pada cluster bernama default.

```
{
  "source": ["aws.ecs"],
  "detail-type": ["ECS Task State Change"]
  "detail": {
    "clusterArn": [
      "default"
    ]
  }
}
```

Tutorial untuk Amazon ECS

Tutorial berikut menunjukkan cara melakukan tugas-tugas umum saat menggunakan Amazon ECS.

Topik

Menggunakan wadah Windows pada AWS Fargate

Mulai menggunakan Amazon ECS AWS Fargate dengan menggunakan jenis peluncuran Fargate untuk tugas Anda di Wilayah tempat Amazon ECS AWS mendukung Fargate.

Selesaikan langkah-langkah berikut untuk memulai Amazon ECS aktif. AWS Fargate

Prasyarat

Sebelum memulai, selesaikan langkah-langkah Siapkan untuk menggunakan Amazon ECS dan bahwa AWS pengguna Anda memiliki izin yang ditentukan dalam contoh kebijakan `AdministratorAccess` IAM.

Konsol mencoba untuk secara otomatis membuat peran IAM eksekusi tugas, yang diperlukan untuk tugas Fargate. Untuk memastikan bahwa konsol dapat membuat peran IAM ini, salah satu dari berikut ini harus benar:

- Pengguna Anda memiliki akses administrator. Untuk informasi selengkapnya, lihat Siapkan untuk menggunakan Amazon ECS.
- Pengguna Anda memiliki izin IAM untuk membuat peran layanan. Untuk informasi selengkapnya, lihat Membuat Peran untuk Mendelegasikan Izin ke Layanan. AWS
- Pengguna dengan akses administrator telah secara manual membuat peran eksekusi tugas sehingga peran itu tersedia pada akun yang akan digunakan. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

> **Important**
>
> Grup keamanan yang Anda pilih saat membuat layanan dengan definisi tugas Anda harus memiliki port 80 terbuka untuk lalu lintas masuk. Tambahkan aturan masuk berikut ke grup keamanan Anda. Untuk informasi tentang cara membuat grup keamanan, lihat Menambahkan aturan ke grup keamanan Anda di Panduan Pengguna Amazon EC2 untuk Instans Linux.
>
> - Jenis: HTTP
> - Protokol: TCP
> - Rentang port: 80
> - Sumber: Di mana saja (`0.0.0.0/0`)

Langkah 1: Buat cluster

Anda dapat membuat cluster baru yang disebut windows yang menggunakan VPC default.

Untuk membuat cluster dengan AWS Management Console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.
3. Di panel navigasi, pilih Klaster.
4. Pada halaman Klaster, pilih Buat klaster.
5. Di bawah konfigurasi Cluster, untuk nama Cluster, masukkan jendela.
6. (Opsional) Untuk mengaktifkan Wawasan Kontainer, perluas Pemantauan, lalu aktifkan Gunakan Wawasan Kontainer.
7. (Opsional) Untuk membantu mengidentifikasi klaster Anda, perluas Tag, lalu konfigurasikan tag Anda.

   [Tambahkan tag] Pilih Tambah tag dan lakukan hal berikut:

   - Untuk Kunci, masukkan nama kunci.
   - Untuk Nilai, masukkan nilai kunci.

   [Hapus tag] Pilih Hapus di sebelah kanan Kunci dan Nilai tag.
8. Pilih Buat.

Langkah 2: Daftarkan ketentuan tugas Windows

Sebelum Anda dapat menjalankan wadah Windows di cluster Amazon ECS Anda, Anda harus mendaftarkan definisi tugas. Contoh ketentuan tugas berikut menampilkan halaman web sederhana pada port 8080 dari instans kontainer dengan citra kontainer `mcr.microsoft.com/windows/servercore/iis`.

Untuk mendaftarkan definisi tugas sampel dengan AWS Management Console

1. Di panel navigasi, pilih Definisi tugas.
2. Pilih Buat definisi tugas baru, Buat definisi tugas baru dengan JSON.
3. Salin dan tempel contoh ketentuan tugas berikut ke dalam kotak dan kemudian pilih Simpan.

```
{
    "containerDefinitions": [
        {
```

```
            "command": ["New-Item -Path C:\\inetpub\\wwwroot\\index.html
 -Type file -Value '<html> <head> <title>Amazon ECS Sample App</title>
 <style>body {margin-top: 40px; background-color: #333;} </style> </head><body>
 <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p>'; C:\\ServiceMonitor.exe w3svc"],
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "essential": true,
            "cpu": 2048,
            "memory": 4096,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "name": "sample_windows_app",
            "portMappings": [
                {
                    "hostPort": 80,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ]
        }
    ],
    "memory": "4096",
    "cpu": "2048",
    "networkMode": "awsvpc",
    "family": "windows-simple-iis-2019-core",
    "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
    "runtimePlatform": {"operatingSystemFamily": "WINDOWS_SERVER_2019_CORE"},
    "requiresCompatibilities": ["FARGATE"]
}
```

4. Verifikasi informasi Anda lalu pilih Buat.

Langkah 3: Buat layanan dengan ketentuan tugas Anda

Setelah Anda mendaftarkan ketentuan tugas, Anda dapat menempatkan tugas di klaster Anda dengan ketentuan tersebut. Prosedur berikut membuat layanan dengan definisi tugas Anda dan menempatkan satu tugas di klaster Anda.

Untuk membuat layanan dari ketentuan tugas Anda dengan konsol

1. Di panel navigasi, pilih Clusters, lalu pilih cluster yang Anda buat. Langkah 1: Buat cluster
2. Dari tab Layanan, pilih Buat.
3. Di bawah konfigurasi Deployment, tentukan cara aplikasi Anda di-deploy.
   a. Untuk definisi Tugas, pilih definisi tugas yang Anda buatLangkah 2: Daftarkan ketentuan tugas Windows.
   b. Untuk nama Layanan, masukkan nama untuk layanan Anda.
   c. Untuk tugas yang diinginkan, masukkan 1.
4. Di bawah Jaringan, Anda dapat membuat grup keamanan atau memilih grup yang ada. Pastikan grup keamanan yang Anda gunakan memiliki aturan masuk yang tercantum di bawahPrasyarat.
5. Pilih Buat.

Langkah 4: Melihat layanan Anda

Setelah layanan Anda meluncurkan tugas ke klaster Anda, Anda dapat melihat layanan dan membuka halaman uji IIS di peramban untuk memverifikasi bahwa kontainer berjalan.

> **Note**
>
> Hal ini dapat memakan waktu hingga 15 menit bagi instans kontainer Anda untuk mengunduh dan mengekstrak lapisan dasar kontainer Windows.

Untuk melihat layanan Anda

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pilih cluster tempat Anda menjalankan layanan.
4. Di tab Layanan, di bawah Nama layanan, pilih layanan yang Anda buatLangkah 3: Buat layanan dengan ketentuan tugas Anda.
5. Pilih tab Tugas, lalu pilih tugas di layanan Anda.
6. Pada halaman tugas, di bagian Konfigurasi, di bawah IP Publik, pilih Buka alamat.

Langkah 5: Bersihkan

Setelah selesai menggunakan kluster Amazon ECS, Anda harus membersihkan sumber daya yang terkait dengannya untuk menghindari biaya untuk sumber daya yang tidak Anda gunakan.

Beberapa sumber daya Amazon ECS, seperti tugas, layanan, cluster, dan instans kontainer, dibersihkan menggunakan konsol Amazon ECS. Sumber daya lain, seperti instans Amazon EC2, penyeimbang beban Elastic Load Balancing, dan grup Auto Scaling, harus dibersihkan secara manual di konsol Amazon EC2 atau dengan menghapus tumpukan yang membuatnya. AWS CloudFormation

1. Pada panel navigasi, silakan pilih Klaster.
2. Pada halaman Clusters, pilih cluster yang Anda buat untuk tutorial ini.
3. Pilih tab Layanan.
4. Pilih layanan, lalu pilih Hapus.
5. Pada prompt konfirmasi, masukkan hapus dan kemudian pilih Hapus.

   Tunggu hingga layanan dihapus.
6. Pilih Hapus klaster. Pada prompt konfirmasi, masukkan hapus ***nama cluster***, lalu pilih Hapus. Menghapus cluster membersihkan sumber daya terkait yang dibuat dengan cluster, termasuk grup Auto Scaling, VPC, atau load balancer.

Membuat cluster dengan tugas Fargate Linux menggunakan AWS CLI

Langkah-langkah berikut membantu Anda menyiapkan klaster, mendaftarkan definisi tugas, menjalankan tugas Linux, dan melakukan skenario umum lainnya di Amazon ECS dengan. AWS CLI Pastikan bahwa Anda menggunakan versi terbaru AWS CLI. Untuk informasi selengkapnya tentang cara meningkatkan ke versi terbaru, lihat Menginstal AWS Command Line Interface.

Topik

- Langkah 4: Buat Layanan
- Langkah 5: Cantumkan Layanan
- Langkah 6: Jelaskan Layanan yang Berjalan
- Langkah 7: Uji
- Langkah 8: Bersihkan

Prasyarat

Tutorial ini mengasumsikan bahwa prasyarat berikut telah diselesaikan.

- Versi terbaru diinstal dan dikonfigurasi. AWS CLI Untuk informasi selengkapnya tentang menginstal atau memutakhirkan AWS CLI, lihat Menginstal. AWS Command Line Interface
- Langkah-langkah di Siapkan untuk menggunakan Amazon ECS telah diselesaikan.
- AWS Pengguna Anda memiliki izin yang diperlukan yang ditentukan dalam contoh kebijakan Amazonecs_ FullAccess IAM.
- Anda memiliki VPC dan grup keamanan yang dibuat untuk digunakan. Tutorial ini menggunakan gambar kontainer yang dihosting di Amazon ECR Public sehingga tugas Anda harus memiliki akses internet. Untuk memberikan tugas Anda rute ke internet, gunakan salah satu pilihan berikut ini.
  - Gunakan subnet privat dengan gateway NAT yang memiliki alamat IP elastis.
  - Gunakan subnet publik dan tetapkan alamat IP publik untuk tugas tersebut.

  Untuk informasi selengkapnya, lihat the section called “Buat virtual private cloud”.

  Untuk informasi tentang grup dan aturan keamanan, lihat, Grup keamanan default untuk VPC Anda dan aturan Contoh di Panduan Pengguna Amazon Virtual Private Cloud.
- Jika Anda mengikuti tutorial ini menggunakan subnet pribadi, Anda dapat menggunakan Amazon ECS Exec untuk berinteraksi langsung dengan penampung Anda dan menguji penerapan. Anda akan perlu untuk membuat peran tugas IAM untuk menggunakan ECS Exec. Untuk informasi selengkapnya tentang peran IAM tugas dan prasyarat lainnya, lihat Menggunakan Amazon ECS Exec untuk debugging.
- (Opsional) AWS CloudShell adalah alat yang memberi pelanggan baris perintah tanpa perlu membuat instance EC2 mereka sendiri. Untuk informasi lebih lanjut, lihat Apa itu AWS CloudShell? dalam AWS CloudShell User Guide.

Langkah 1: Buat Klaster

Secara default, akun Anda menerima klaster `default`.

> **Note**
>
> Manfaat dari penggunaan klaster `default` yang disediakan untuk Anda adalah bahwa Anda tidak perlu menentukan properti pada pilihan `--cluster` *`cluster_name`* di perintah berikutnya. Jika Anda membuat klaster non-default sendiri, Anda harus menentukan `--cluster` *`cluster_name`* untuk setiap perintah yang ingin Anda gunakan dengan klaster itu.

Buat klaster Anda sendiri dengan nama yang unik menggunakan perintah berikut:

```
aws ecs create-cluster --cluster-name fargate-cluster
```

Output:

```
{
    "cluster": {
        "status": "ACTIVE",
        "defaultCapacityProviderStrategy": [],
        "statistics": [],
        "capacityProviders": [],
        "tags": [],
        "clusterName": "fargate-cluster",
        "settings": [
            {
                "name": "containerInsights",
                "value": "disabled"
            }
        ],
        "registeredContainerInstancesCount": 0,
        "pendingTasksCount": 0,
        "runningTasksCount": 0,
        "activeServicesCount": 0,
        "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/fargate-cluster"
    }
}
```

Langkah 2: Daftarkan Definisi Tugas Linux

Sebelum Anda dapat menjalankan tugas di klaster ECS, Anda harus mendaftarkan ketentuan tugas. Ketentuan tugas merupakan daftar kontainer yang dikelompokkan bersama-sama. Contoh berikut adalah ketentuan tugas sederhana yang menciptakan sebuah aplikasi web PHP dengan menggunakan citra kontainer httpd yang di-hosting di Docker Hub. Untuk informasi selengkapnya tentang parameter ketentuan tugas yang tersedia, lihat Definisi tugas Amazon ECS. Untuk tutorial ini, hanya `taskRoleArn` diperlukan jika Anda menerapkan tugas di subnet pribadi dan ingin menguji penerapan. Ganti `taskRoleArn` dengan peran tugas IAM yang Anda buat untuk menggunakan ECS Exec seperti yang disebutkan dalam. Prasyarat

```
{
       "family": "sample-fargate",
       "networkMode": "awsvpc",
       "taskRoleArn": "arn:aws:iam::aws_account_id:role/execCommandRole",
       "containerDefinitions": [
           {
               "name": "fargate-app",
               "image": "public.ecr.aws/docker/library/httpd:latest",
               "portMappings": [
                   {
                       "containerPort": 80,
                       "hostPort": 80,
                       "protocol": "tcp"
                   }
               ],
               "essential": true,
               "entryPoint": [
                   "sh",
                   "-c"
               ],
               "command": [
                   "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon
 ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html && httpd-
foreground\""
               ]
           }
       ],
       "requiresCompatibilities": [
```

```
            "FARGATE"
        ],
        "cpu": "256",
        "memory": "512"
}
```

Simpan definisi tugas JSON sebagai file dan berikan dengan `--cli-input-json file://path_to_file.json` opsi.

Untuk menggunakan file JSON untuk ketentuan kontainer:

```
aws ecs register-task-definition --cli-input-json file://$HOME/tasks/fargate-task.json
```

Parameter perintah register-task-definition mengembalikan penjelasan pada ketentuan tugas setelah menyelesaikan pendaftarannya.

Langkah 3: Cantumkan Ketentuan tugas

Anda dapat mencantumkan daftar ketentuan tugas untuk akun Anda kapan saja dengan perintah list-task-definitions. Output dari perintah ini menunjukkan nilai `family` dan `revision` yang dapat Anda gunakan bersama saat memanggil run-task atau start-task.

```
aws ecs list-task-definitions
```

Output:

```
{
    "taskDefinitionArns": [
        "arn:aws:ecs:region:aws_account_id:task-definition/sample-fargate:1"
    ]
}
```

Langkah 4: Buat Layanan

Setelah Anda telah mendaftarkan tugas untuk akun Anda, Anda dapat membuat layanan untuk tugas yang telah terdaftar di klaster Anda. Pada contoh ini, Anda membuat layanan dengan satu instans dari ketentuan tugas `sample-fargate:1` yang berjalan di klaster Anda. Tugas tersebut membutuhkan rute menuju internet, maka terdapat dua cara untuk Anda agar bisa mencapainya.

Salah satu caranya adalah dengan menggunakan subnet privat yang dikonfigurasikan dengan gateway NAT dengan alamat IP elastis di subnet publik. Cara lainnya adalah dengan menggunakan subnet publik dan menetapkan alamat IP publik untuk tugas Anda. Kami menyediakan kedua contoh di bawah ini.

Contoh menggunakan subnet privat. `enable-execute-command` Opsi ini diperlukan untuk menggunakan Amazon ECS Exec.

```
aws ecs create-service --cluster fargate-cluster --service-name fargate-service --
task-definition sample-fargate:1 --desired-count 1 --launch-type "FARGATE" --network-
configuration "awsvpcConfiguration={subnets=[subnet-abcd1234],securityGroups=[sg-
abcd1234]}" --enable-execute-command
```

Contoh menggunakan subnet publik.

```
aws ecs create-service --cluster fargate-cluster --service-name fargate-service --
task-definition sample-fargate:1 --desired-count 1 --launch-type "FARGATE" --network-
configuration "awsvpcConfiguration={subnets=[subnet-abcd1234],securityGroups=[sg-
abcd1234],assignPublicIp=ENABLED}"
```

Perintah create-service mengembalikan penjelasan pada ketentuan tugas setelah menyelesaikan pendaftarannya.

Langkah 5: Cantumkan Layanan

Cantumkan layanan untuk klaster Anda. Anda seharusnya dapat melihat layanan yang Anda buat di bagian sebelumnya. Anda dapat mengambil nama layanan atau ARN penuh yang telah dikembalikan dari perintah ini serta menggunakannya untuk menjelaskan layanan nantinya.

```
aws ecs list-services --cluster fargate-cluster
```

Output:

```
{
    "serviceArns": [
        "arn:aws:ecs:region:aws_account_id:service/fargate-cluster/fargate-service"
    ]
}
```

Langkah 6: Jelaskan Layanan yang Berjalan

Jelaskan layanan yang menggunakan nama layanan yang telah diambil sebelumnya untuk mendapatkan informasi lebih lanjut tentang tugas.

```
aws ecs describe-services --cluster fargate-cluster --services fargate-service
```

Jika berhasil, tugas akan mengembalikan deskripsi dari layanan yang gagal juga layanan. Misalnya, di `services` bagian ini, Anda akan menemukan informasi tentang penerapan, seperti status tugas sebagai berjalan atau tertunda. Anda juga dapat menemukan informasi tentang ketentuan tugas, konfigurasi jaringan dan peristiwa stempel waktu. Pada bagian kegagalan, Anda akan menemukan informasi tentang kegagalan, jika ada kegagalan tersebut, yang terkait dengan panggilan tersebut. Untuk pemecahan masalah, lihat Pesan Peristiwa Layanan. Untuk informasi selengkapnya tentang deskripsi layanan, lihat Jelaskan layanan.

```
{
    "services": [
        {
            "networkConfiguration": {
                "awsvpcConfiguration": {
                    "subnets": [
                        "subnet-abcd1234"
                    ],
                    "securityGroups": [
                        "sg-abcd1234"
                    ],
                    "assignPublicIp": "ENABLED"
                }
            },
            "launchType": "FARGATE",
            "enableECSManagedTags": false,
            "loadBalancers": [],
            "deploymentController": {
                "type": "ECS"
            },
            "desiredCount": 1,
            "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/fargate-cluster",
            "serviceArn": "arn:aws:ecs:region:aws_account_id:service/fargate-service",
            "deploymentConfiguration": {
                "maximumPercent": 200,
                "minimumHealthyPercent": 100
```

```
            },
            "createdAt": 1692283199.771,
            "schedulingStrategy": "REPLICA",
            "placementConstraints": [],
            "deployments": [
                {
                    "status": "PRIMARY",
                    "networkConfiguration": {
                        "awsvpcConfiguration": {
                            "subnets": [
                                "subnet-abcd1234"
                            ],
                            "securityGroups": [
                                "sg-abcd1234"
                            ],
                            "assignPublicIp": "ENABLED"
                        }
                    },
                    "pendingCount": 0,
                    "launchType": "FARGATE",
                    "createdAt": 1692283199.771,
                    "desiredCount": 1,
                    "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-
definition/sample-fargate:1",
                    "updatedAt": 1692283199.771,
                    "platformVersion": "1.4.0",
                    "id": "ecs-svc/9223370526043414679",
                    "runningCount": 0
                }
            ],
            "serviceName": "fargate-service",
            "events": [
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 53c0de40-ea3b-489f-a352-623bf1235f08) (task d0aec985-901b-488f-9fb4-61b991b332a3).",
                    "id": "92b8443e-67fb-4886-880c-07e73383ea83",
                    "createdAt": 1510811841.408
                },
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 b4911bee-7203-4113-99d4-e89ba457c626) (task cc5853e3-6e2d-4678-8312-74f8a7d76474).",
                    "id": "d85c6ec6-a693-43b3-904a-a997e1fc844d",
                    "createdAt": 1510811601.938
                },
```

```
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 cba86182-52bf-42d7-9df8-b744699e6cfc) (task f4c1ad74-a5c6-4620-90cf-2aff118df5fc).",
                    "id": "095703e1-0ca3-4379-a7c8-c0f1b8b95ace",
                    "createdAt": 1510811364.691
                }
            ],
            "runningCount": 0,
            "status": "ACTIVE",
            "serviceRegistries": [],
            "pendingCount": 0,
            "createdBy": "arn:aws:iam::aws_account_id:user/user_name",
            "platformVersion": "LATEST",
            "placementStrategy": [],
            "propagateTags": "NONE",
            "roleArn": "arn:aws:iam::aws_account_id:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
            "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-definition/
sample-fargate:1"
        }
    ],
    "failures": []
}
```

Langkah 7: Uji

Tugas pengujian dikerahkan menggunakan subnet publik

Jelaskan tugas dalam layanan sehingga Anda bisa mendapatkan Elastic Network Interface (ENI) untuk tugas tersebut.

Pertama, dapatkan tugas ARN.

```
aws ecs list-tasks --cluster fargate-cluster --service fargate-service
```

Outputnya berisi tugas ARN.

```
{
    "taskArns": [
        "arn:aws:ecs:us-east-1:123456789012:task/fargate-service/EXAMPLE
    ]
```

```
}
```

Jelaskan tugas dan temukan ID ENI. Gunakan tugas ARN untuk parameter. `tasks`

```
aws ecs describe-tasks --cluster fargate-cluster --tasks arn:aws:ecs:us-
east-1:123456789012:task/service/EXAMPLE
```

Informasi lampiran tercantum dalam output.

```
{
    "tasks": [
        {
            "attachments": [
                {
                    "id": "d9e7735a-16aa-4128-bc7a-b2d5115029e9",
                    "type": "ElasticNetworkInterface",
                    "status": "ATTACHED",
                    "details": [
                        {
                            "name": "subnetId",
                            "value": "subnetabcd1234"
                        },
                        {
                            "name": "networkInterfaceId",
                            "value": "eni-0fa40520aeEXAMPLE"
                        },
                    ]
                }
…
}
```

Jelaskan ENI untuk mendapatkan alamat IP publik.

```
aws ec2 describe-network-interfaces --network-interface-id  eni-0fa40520aeEXAMPLE
```

Alamat IP publik ada di output.

```
{
    "NetworkInterfaces": [
        {
            "Association": {
```

```
                "IpOwnerId": "amazon",
                "PublicDnsName": "ec2-34-229-42-222.compute-1.amazonaws.com",
                "PublicIp": "198.51.100.2"
            },
…
}
```

Masukkan alamat IP publik di browser web Anda dan Anda akan melihat halaman web yang menampilkan contoh aplikasi Amazon ECS.

Tugas pengujian yang digunakan menggunakan subnet pribadi

Jelaskan tugas dan temukan `managedAgents` untuk memverifikasi bahwa `ExecuteCommandAgent` sedang berjalan. Perhatikan `privateIPv4Address` untuk digunakan nanti.

```
aws ecs describe-tasks --cluster fargate-cluster --tasks arn:aws:ecs:us-
east-1:123456789012:task/fargate-service/EXAMPLE
```

Informasi agen terkelola tercantum dalam output.

```
{
     "tasks": [
        {
            "attachments": [
                {
                    "id": "d9e7735a-16aa-4128-bc7a-b2d5115029e9",
                    "type": "ElasticNetworkInterface",
                    "status": "ATTACHED",
                    "details": [
                        {
                            "name": "subnetId",
                            "value": "subnetabcd1234"
                        },
                        {
                            "name": "networkInterfaceId",
                            "value": "eni-0fa40520aeEXAMPLE"
                        },
                        {
                            "name": "privateIPv4Address",
                            "value": "10.0.143.156"
                        }
                    ]
```

```
                }
            ],
    ...
    "containers": [
        {
        ...
       "managedAgents": [
                        {
                            "lastStartedAt": "2023-08-01T16:10:13.002000+00:00",
                            "name": "ExecuteCommandAgent",
                            "lastStatus": "RUNNING"
                        }
                ],
        ...
   }
```

Setelah memverifikasi bahwa `ExecuteCommandAgent` sedang berjalan, Anda dapat menjalankan perintah berikut untuk menjalankan shell interaktif pada wadah dalam tugas.

```
aws ecs execute-command --cluster fargate-cluster \
    --task  arn:aws:ecs:us-east-1:123456789012:task/fargate-service/EXAMPLE  \
    --container  fargate-app \
    --interactive \
    --command "/bin/sh"
```

Setelah shell interaktif berjalan, jalankan perintah berikut untuk menginstal cURL.

```
apt update
```

```
apt install curl
```

Setelah menginstal cURL, jalankan perintah berikut menggunakan alamat IP pribadi yang Anda peroleh sebelumnya.

```
curl 10.0.143.156
```

Anda akan melihat HTML yang setara dengan halaman web aplikasi sampel Amazon ECS.

```
<html>
```

```
    <head>
     <title>Amazon ECS Sample App</title>
     <style>body {margin-top: 40px; background-color: #333;} </style>
    </head>
      <body>
      <div style=color:white;text-align:center>
      <h1>Amazon ECS Sample App</h1>
      <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p>
      </div>
      </body>
</html>
```

Langkah 8: Bersihkan

Setelah Anda selesai dengan tutorial ini, Anda harus membersihkan sumber daya yang terkait untuk menghindari biaya sumber daya yang tidak terpakai.

Hapus layanan.

```
aws ecs delete-service --cluster fargate-cluster --service fargate-service --force
```

Hapus klaster.

```
aws ecs delete-cluster --cluster fargate-cluster
```

Membuat cluster dengan tugas Fargate Windows menggunakan AWS CLI

Langkah-langkah berikut membantu Anda menyiapkan klaster, mendaftarkan definisi tugas, menjalankan tugas Windows, dan melakukan skenario umum lainnya di Amazon ECS dengan file. AWS CLI Pastikan bahwa Anda menggunakan versi terbaru AWS CLI. Untuk informasi selengkapnya tentang cara meningkatkan ke versi terbaru, lihat Menginstal AWS Command Line Interface.

Topik

- Prasyarat
- Langkah 1: Buat Klaster

Prasyarat

Tutorial ini mengasumsikan bahwa prasyarat berikut telah diselesaikan.

- Versi terbaru diinstal dan dikonfigurasi. AWS CLI Untuk informasi selengkapnya tentang menginstal atau memutakhirkan AWS CLI, lihat Menginstal. AWS Command Line Interface
- Langkah-langkah di Siapkan untuk menggunakan Amazon ECS telah diselesaikan.
- AWS Pengguna Anda memiliki izin yang diperlukan yang ditentukan dalam contoh kebijakan Amazonecs_ FullAccess IAM.
- Anda memiliki VPC dan grup keamanan yang telah dibuat untuk digunakan. Tutorial ini menggunakan citra kontainer yang di-host pada Docker Hub sehingga tugas Anda harus memiliki akses kepada internet. Untuk memberikan tugas Anda rute ke internet, gunakan salah satu pilihan berikut ini.
  - Gunakan subnet privat dengan gateway NAT yang memiliki alamat IP elastis.
  - Gunakan subnet publik dan tetapkan alamat IP publik untuk tugas tersebut.

  Untuk informasi selengkapnya, lihat the section called “Buat virtual private cloud”.

  Untuk informasi tentang grup dan aturan keamanan, lihat, Grup keamanan default untuk VPC Anda dan aturan Contoh di Panduan Pengguna Amazon Virtual Private Cloud.
- (Opsional) AWS CloudShell adalah alat yang memberi pelanggan baris perintah tanpa perlu membuat instance EC2 mereka sendiri. Untuk informasi lebih lanjut, lihat Apa itu AWS CloudShell? dalam AWS CloudShell User Guide.

Langkah 1: Buat Klaster

Secara default, akun Anda menerima klaster `default`.

> **Note**
>
> Manfaat dari penggunaan klaster `default` yang disediakan untuk Anda adalah bahwa Anda tidak perlu menentukan properti pada pilihan `--cluster` *`cluster_name`* di perintah berikutnya. Jika Anda membuat klaster non-default sendiri, Anda harus menentukan `--cluster` *`cluster_name`* untuk setiap perintah yang ingin Anda gunakan dengan klaster itu.

Buat klaster Anda sendiri dengan nama yang unik menggunakan perintah berikut:

```
aws ecs create-cluster --cluster-name fargate-cluster
```

Output:

```
{
    "cluster": {
        "status": "ACTIVE",
        "statistics": [],
        "clusterName": "fargate-cluster",
        "registeredContainerInstancesCount": 0,
        "pendingTasksCount": 0,
        "runningTasksCount": 0,
        "activeServicesCount": 0,
        "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/fargate-cluster"
    }
}
```

Langkah 2: Daftarkan Definisi Tugas Windows

Sebelum Anda dapat menjalankan tugas Windows di cluster Amazon ECS Anda, Anda harus mendaftarkan definisi tugas. Ketentuan tugas adalah daftar kontainer yang dikelompokkan bersama. Contoh berikut adalah definisi tugas sederhana yang membuat aplikasi web. Untuk informasi selengkapnya tentang parameter ketentuan tugas yang tersedia, lihat Definisi tugas Amazon ECS.

```
{
    "containerDefinitions": [
        {
            "command": ["New-Item -Path C:\\inetpub\\wwwroot\\index.html -Type file
 -Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
```

```
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your
 application is now running on a container in Amazon ECS.</p>'; C:\\ServiceMonitor.exe
 w3svc"],
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "essential": true,
            "cpu": 2048,
            "memory": 4096,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "name": "sample_windows_app",
            "portMappings": [
                {
                    "hostPort": 80,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ]
        }
    ],
    "memory": "4096",
    "cpu": "2048",
    "networkMode": "awsvpc",
    "family": "windows-simple-iis-2019-core",
    "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
    "runtimePlatform": {"operatingSystemFamily": "WINDOWS_SERVER_2019_CORE"},
    "requiresCompatibilities": ["FARGATE"]
}
```

Contoh di atas JSON dapat diteruskan ke AWS CLI dalam dua cara: Anda dapat menyimpan definisi tugas JSON sebagai file dan meneruskannya dengan opsi. `--cli-input-json file://`*`path_to_file.json`*

Untuk menggunakan file JSON untuk ketentuan kontainer:

```
aws ecs register-task-definition --cli-input-json file://$HOME/tasks/fargate-task.json
```

Parameter perintah register-task-definition mengembalikan penjelasan pada ketentuan tugas setelah menyelesaikan pendaftarannya.

Langkah 3: Daftar definisi tugas

Anda dapat mencantumkan daftar ketentuan tugas untuk akun Anda kapan saja dengan perintah list-task-definitions. Output dari perintah ini menunjukkan nilai `family` dan `revision` yang dapat Anda gunakan bersama saat memanggil run-task atau start-task.

```
aws ecs list-task-definitions
```

Output:

```
{
    "taskDefinitionArns": [
        "arn:aws:ecs:region:aws_account_id:task-definition/sample-fargate-windows:1"
    ]
}
```

Langkah 4: Buat layanan

Setelah Anda telah mendaftarkan tugas untuk akun Anda, Anda dapat membuat layanan untuk tugas yang telah terdaftar di klaster Anda. Pada contoh ini, Anda membuat layanan dengan satu instans dari ketentuan tugas `sample-fargate:1` yang berjalan di klaster Anda. Tugas tersebut membutuhkan rute menuju internet, maka terdapat dua cara untuk Anda agar bisa mencapainya. Salah satu caranya adalah dengan menggunakan subnet privat yang dikonfigurasikan dengan gateway NAT dengan alamat IP elastis di subnet publik. Cara lainnya adalah dengan menggunakan subnet publik dan menetapkan alamat IP publik untuk tugas Anda. Kami menyediakan kedua contoh di bawah ini.

Contoh menggunakan subnet privat.

```
aws ecs create-service --cluster fargate-cluster --service-name fargate-service
 --task-definition sample-fargate-windows:1 --desired-count 1 --launch-type
 "FARGATE" --network-configuration "awsvpcConfiguration={subnets=[subnet-
abcd1234],securityGroups=[sg-abcd1234]}"
```

Contoh menggunakan subnet publik.

```
aws ecs create-service --cluster fargate-cluster --service-name fargate-service
 --task-definition sample-fargate-windows:1 --desired-count 1 --launch-type
```

```
 "FARGATE" --network-configuration "awsvpcConfiguration={subnets=[subnet-
abcd1234],securityGroups=[sg-abcd1234],assignPublicIp=ENABLED}"
```

Perintah create-service mengembalikan penjelasan pada ketentuan tugas setelah menyelesaikan pendaftarannya.

Langkah 5: Daftar layanan

Cantumkan layanan untuk klaster Anda. Anda seharusnya dapat melihat layanan yang Anda buat di bagian sebelumnya. Anda dapat mengambil nama layanan atau ARN penuh yang telah dikembalikan dari perintah ini serta menggunakannya untuk menjelaskan layanan nantinya.

```
aws ecs list-services --cluster fargate-cluster
```

Output:

```
{
    "serviceArns": [
        "arn:aws:ecs:region:aws_account_id:service/fargate-service"
    ]
}
```

Langkah 6: Jelaskan Layanan yang Berjalan

Jelaskan layanan yang menggunakan nama layanan yang telah diambil sebelumnya untuk mendapatkan informasi lebih lanjut tentang tugas.

```
aws ecs describe-services --cluster fargate-cluster --services fargate-service
```

Jika berhasil, tugas akan mengembalikan deskripsi dari layanan yang gagal juga layanan. Sebagai contoh, pada bagian layanan, Anda akan menemukan informasi tentang deployment, seperti menjelaskan status pada tugas berjalan atau tertundanya tugas tersebut. Anda juga dapat menemukan informasi tentang ketentuan tugas, konfigurasi jaringan dan peristiwa stempel waktu. Pada bagian kegagalan, Anda akan menemukan informasi tentang kegagalan, jika ada kegagalan tersebut, yang terkait dengan panggilan tersebut. Untuk pemecahan masalah, lihat Pesan Peristiwa Layanan. Untuk informasi selengkapnya tentang deskripsi layanan, lihat Jelaskan layanan.

```
{
    "services": [
        {
```

```
            "status": "ACTIVE",
            "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-definition/
sample-fargate-windows:1",
            "pendingCount": 2,
            "launchType": "FARGATE",
            "loadBalancers": [],
            "roleArn": "arn:aws:iam::aws_account_id:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
            "placementConstraints": [],
            "createdAt": 1510811361.128,
            "desiredCount": 2,
            "networkConfiguration": {
                "awsvpcConfiguration": {
                    "subnets": [
                        "subnet-abcd1234"
                    ],
                    "securityGroups": [
                        "sg-abcd1234"
                    ],
                    "assignPublicIp": "DISABLED"
                }
            },
            "platformVersion": "LATEST",
            "serviceName": "fargate-service",
            "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/fargate-cluster",
            "serviceArn": "arn:aws:ecs:region:aws_account_id:service/fargate-service",
            "deploymentConfiguration": {
                "maximumPercent": 200,
                "minimumHealthyPercent": 100
            },
            "deployments": [
                {
                    "status": "PRIMARY",
                    "networkConfiguration": {
                        "awsvpcConfiguration": {
                            "subnets": [
                                "subnet-abcd1234"
                            ],
                            "securityGroups": [
                                "sg-abcd1234"
                            ],
                            "assignPublicIp": "DISABLED"
                        }
                    },
```

```
                    "pendingCount": 2,
                    "launchType": "FARGATE",
                    "createdAt": 1510811361.128,
                    "desiredCount": 2,
                    "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-
definition/sample-fargate-windows:1",
                    "updatedAt": 1510811361.128,
                    "platformVersion": "0.0.1",
                    "id": "ecs-svc/9223370526043414679",
                    "runningCount": 0
                }
            ],
            "events": [
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 53c0de40-ea3b-489f-a352-623bf1235f08) (task d0aec985-901b-488f-9fb4-61b991b332a3).",
                    "id": "92b8443e-67fb-4886-880c-07e73383ea83",
                    "createdAt": 1510811841.408
                },
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 b4911bee-7203-4113-99d4-e89ba457c626) (task cc5853e3-6e2d-4678-8312-74f8a7d76474).",
                    "id": "d85c6ec6-a693-43b3-904a-a997e1fc844d",
                    "createdAt": 1510811601.938
                },
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 cba86182-52bf-42d7-9df8-b744699e6cfc) (task f4c1ad74-a5c6-4620-90cf-2aff118df5fc).",
                    "id": "095703e1-0ca3-4379-a7c8-c0f1b8b95ace",
                    "createdAt": 1510811364.691
                }
            ],
            "runningCount": 0,
            "placementStrategy": []
        }
    ],
    "failures": []
}
```

Langkah 7: Bersihkan

Setelah Anda selesai dengan tutorial ini, Anda harus membersihkan sumber daya yang terkait untuk menghindari biaya sumber daya yang tidak terpakai.

Hapus layanan.

```
aws ecs delete-service --cluster fargate-cluster --service fargate-service --force
```

Hapus klaster.

```
aws ecs delete-cluster --cluster fargate-cluster
```

Membuat cluster dengan tugas EC2 menggunakan AWS CLI

Langkah-langkah berikut membantu Anda menyiapkan klaster, mendaftarkan definisi tugas, menjalankan tugas, dan melakukan skenario umum lainnya di Amazon ECS dengan. AWS CLI Pastikan bahwa Anda menggunakan versi terbaru AWS CLI. Untuk informasi selengkapnya tentang cara meningkatkan ke versi terbaru, lihat Menginstal AWS Command Line Interface.

Topik

Prasyarat

Tutorial ini mengasumsikan bahwa prasyarat berikut telah diselesaikan:

- Versi terbaru diinstal dan dikonfigurasi. AWS CLI Untuk informasi selengkapnya tentang menginstal atau memutakhirkan AWS CLI, lihat Menginstal. AWS Command Line Interface
- Langkah-langkah di Siapkan untuk menggunakan Amazon ECS telah diselesaikan.

- AWS Pengguna Anda memiliki izin yang diperlukan yang ditentukan dalam contoh kebijakan Amazonecs_ FullAccess IAM.
- Anda memiliki VPC dan grup keamanan yang dibuat untuk digunakan. Untuk informasi selengkapnya, lihat the section called “Buat virtual private cloud”.
- (Opsional) AWS CloudShell adalah alat yang memberi pelanggan baris perintah tanpa perlu membuat instance EC2 mereka sendiri. Untuk informasi lebih lanjut, lihat Apa itu AWS CloudShell? dalam AWS CloudShell User Guide.

Langkah 1: Buat Klaster

Secara default, akun Anda menerima klaster `default` ketika Anda meluncurkan instans kontainer pertama Anda.

> **Note**
>
> Manfaat menggunakan klaster `default` yang disediakan untuk Anda adalah Anda tidak perlu menentukan opsi `--cluster` *`cluster_name`* dalam perintah berikutnya. Jika Anda membuat klaster non-default sendiri, Anda harus menentukan `--cluster` *`cluster_name`* untuk setiap perintah yang ingin Anda gunakan dengan klaster itu.

Buat klaster Anda sendiri dengan nama yang unik menggunakan perintah berikut:

```
aws ecs create-cluster --cluster-name MyCluster
```

Output:

```
{
    "cluster": {
        "clusterName": "MyCluster",
        "status": "ACTIVE",
        "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/MyCluster"
    }
}
```

Langkah 2: Luncurkan Instans dengan Amazon ECS AMI

Anda harus memiliki instans penampung Amazon ECS di cluster Anda sebelum dapat menjalankan tugas di dalamnya. Jika Anda tidak memiliki instans kontainer di klaster Anda, lihat Meluncurkan instans penampung Amazon ECS Linux untuk informasi selengkapnya.

Langkah 3: Buat Daftar Instans Kontainer

Dalam beberapa menit setelah meluncurkan instans penampung Anda, agen Amazon ECS mendaftarkan instans tersebut dengan klaster default Anda. Anda dapat membuat daftar instans kontainer dalam klaster dengan menjalankan perintah berikut:

```
aws ecs list-container-instances --cluster default
```

Output:

```
{
    "containerInstanceArns": [
        "arn:aws:ecs:us-east-1:aws_account_id:container-instance/container_instance_ID"
    ]
}
```

Langkah 4: Jelaskan Instans Kontainer Anda

Setelah Anda memiliki ARN atau ID instans kontainer, Anda dapat menggunakan perintah describe-container-instances untuk mendapatkan informasi berharga terkait instans, seperti sumber daya CPU dan memori yang tersisa dan terdaftar.

```
aws ecs describe-container-instances --cluster default --container-
instances container_instance_ID
```

Output:

```
{
    "failures": [],
    "containerInstances": [
        {
            "status": "ACTIVE",
            "registeredResources": [
                {
```

```
                    "integerValue": 1024,
                    "longValue": 0,
                    "type": "INTEGER",
                    "name": "CPU",
                    "doubleValue": 0.0
                },
                {
                    "integerValue": 995,
                    "longValue": 0,
                    "type": "INTEGER",
                    "name": "MEMORY",
                    "doubleValue": 0.0
                },
                {
                    "name": "PORTS",
                    "longValue": 0,
                    "doubleValue": 0.0,
                    "stringSetValue": [
                        "22",
                        "2376",
                        "2375",
                        "51678"
                    ],
                    "type": "STRINGSET",
                    "integerValue": 0
                },
                {
                    "name": "PORTS_UDP",
                    "longValue": 0,
                    "doubleValue": 0.0,
                    "stringSetValue": [],
                    "type": "STRINGSET",
                    "integerValue": 0
                }
            ],
            "ec2InstanceId": "instance_id",
            "agentConnected": true,
            "containerInstanceArn": "arn:aws:ecs:us-west-2:aws_account_id:container-
instance/container_instance_ID",
            "pendingTasksCount": 0,
            "remainingResources": [
                {
                    "integerValue": 1024,
                    "longValue": 0,
```

```
                "type": "INTEGER",
                "name": "CPU",
                "doubleValue": 0.0
            },
            {
                "integerValue": 995,
                "longValue": 0,
                "type": "INTEGER",
                "name": "MEMORY",
                "doubleValue": 0.0
            },
            {
                "name": "PORTS",
                "longValue": 0,
                "doubleValue": 0.0,
                "stringSetValue": [
                    "22",
                    "2376",
                    "2375",
                    "51678"
                ],
                "type": "STRINGSET",
                "integerValue": 0
            },
            {
                "name": "PORTS_UDP",
                "longValue": 0,
                "doubleValue": 0.0,
                "stringSetValue": [],
                "type": "STRINGSET",
                "integerValue": 0
            }
        ],
        "runningTasksCount": 0,
        "attributes": [
            {
                "name": "com.amazonaws.ecs.capability.privileged-container"
            },
            {
                "name": "com.amazonaws.ecs.capability.docker-remote-api.1.17"
            },
            {
                "name": "com.amazonaws.ecs.capability.docker-remote-api.1.18"
            },
```

```
                {
                    "name": "com.amazonaws.ecs.capability.docker-remote-api.1.19"
                },
                {
                    "name": "com.amazonaws.ecs.capability.logging-driver.json-file"
                },
                {
                    "name": "com.amazonaws.ecs.capability.logging-driver.syslog"
                }
            ],
            "versionInfo": {
                "agentVersion": "1.5.0",
                "agentHash": "b197edd",
                "dockerVersion": "DockerVersion: 1.7.1"
            }
        }
    ]
}
```

Anda juga dapat menemukan ID instans Amazon EC2 yang dapat Anda gunakan untuk memantau instans di konsol Amazon EC2 atau dengan perintah. aws ec2 describe-instances --instance-id ***instance_id***

Langkah 5: Daftarkan Definisi Tugas

Sebelum Anda dapat menjalankan tugas di klaster ECS, Anda harus mendaftarkan ketentuan tugas. Ketentuan tugas adalah daftar kontainer yang dikelompokkan bersama. Contoh berikut adalah ketentuan tugas sederhana yang menggunakan citra `busybox` dari Docker Hub dan hanya tidur selama 360 detik. Untuk informasi selengkapnya tentang parameter ketentuan tugas yang tersedia, lihat Definisi tugas Amazon ECS.

```
{
    "containerDefinitions": [
        {
            "name": "sleep",
            "image": "busybox",
            "cpu": 10,
            "command": [
                "sleep",
                "360"
            ],
            "memory": 10,
```

```
            "essential": true
        }
    ],
    "family": "sleep360"
}
```

Contoh di atas JSON dapat diteruskan ke AWS CLI dalam dua cara: Anda dapat menyimpan definisi tugas JSON sebagai file dan meneruskannya dengan opsi. `--cli-input-json file://path_to_file.json` Atau, Anda dapat menghindari tanda kutip di JSON dan meneruskan ketentuan kontainer JSON pada baris perintah seperti pada contoh di bawah ini. Jika Anda memilih untuk meneruskan ketentuan kontainer pada baris perintah, maka perintah Anda juga memerlukan parameter `--family` yang digunakan untuk menjaga beberapa versi ketentuan tugas Anda yang terkait satu sama lain.

Untuk menggunakan file JSON untuk ketentuan kontainer:

```
aws ecs register-task-definition --cli-input-json file://$HOME/tasks/sleep360.json
```

Untuk menggunakan string JSON untuk ketentuan kontainer:

```
aws ecs register-task-definition --family sleep360 --container-definitions "[{\"name
\":\"sleep\",\"image\":\"busybox\",\"cpu\":10,\"command\":[\"sleep\",\"360\"],\"memory
\":10,\"essential\":true}]"
```

register-task-definition mengembalikan deskripsi ketentuan tugas setelah menyelesaikan pendaftarannya.

```
{
    "taskDefinition": {
        "volumes": [],
        "taskDefinitionArn": "arn:aws:ec2:us-east-1:aws_account_id:task-definition/
sleep360:1",
        "containerDefinitions": [
            {
                "environment": [],
                "name": "sleep",
                "mountPoints": [],
                "image": "busybox",
                "cpu": 10,
                "portMappings": [],
```

```
                "command": [
                    "sleep",
                    "360"
                ],
                "memory": 10,
                "essential": true,
                "volumesFrom": []
            }
        ],
        "family": "sleep360",
        "revision": 1
    }
}
```

Langkah 6: Buat Daftar Definisi Tugas

Anda dapat membuat daftar ketentuan tugas untuk akun Anda kapan pun dengan perintah list-task-definitions. Output dari perintah ini menunjukkan nilai `family` dan `revision` yang dapat Anda gunakan bersama saat memanggil run-task atau start-task.

```
aws ecs list-task-definitions
```

Output:

```
{
    "taskDefinitionArns": [
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/sleep300:1",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/sleep300:2",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/sleep360:1",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:3",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:4",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:5",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:6"
    ]
}
```

Langkah 7: Jalankan Tugas

Setelah Anda mendaftarkan tugas untuk akun Anda dan meluncurkan instans kontainer yang didaftarkan ke klaster Anda, Anda dapat menjalankan tugas terdaftar di klaster Anda. Untuk contoh ini, Anda menempatkan satu instans ketentuan tugas `sleep360:1` di klaster default Anda.

```
aws ecs run-task --cluster default --task-definition sleep360:1 --count 1
```

Output:

```
{
    "tasks": [
        {
            "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID",
            "overrides": {
                "containerOverrides": [
                    {
                        "name": "sleep"
                    }
                ]
            },
            "lastStatus": "PENDING",
            "containerInstanceArn": "arn:aws:ecs:us-east-1:aws_account_id:container-
instance/container_instance_ID",
            "clusterArn": "arn:aws:ecs:us-east-1:aws_account_id:cluster/default",
            "desiredStatus": "RUNNING",
            "taskDefinitionArn": "arn:aws:ecs:us-east-1:aws_account_id:task-definition/
sleep360:1",
            "containers": [
                {
                    "containerArn": "arn:aws:ecs:us-
east-1:aws_account_id:container/container_ID",
                    "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID",
                    "lastStatus": "PENDING",
                    "name": "sleep"
                }
            ]
        }
    ]
}
```

Langkah 8: Cantumkan Tugas

Cantumkan tugas untuk klaster Anda. Anda akan melihat tugas yang Anda jalankan di bagian sebelumnya. Anda dapat mengambil ID tugas atau ARN lengkap yang dikembalikan dari perintah tersebut dan menggunakannya untuk menjelaskan tugas berikutnya.

```
aws ecs list-tasks --cluster default
```

Output:

```
{
    "taskArns": [
        "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID"
    ]
}
```

Langkah 9: Jelaskan Tugas yang Sedang Berjalan

Jelaskan tugas menggunakan ID tugas yang diambil sebelumnya untuk mendapatkan informasi selengkapnya tentang tugas tersebut.

```
aws ecs describe-tasks --cluster default --task task_ID
```

Output:

```
{
    "failures": [],
    "tasks": [
        {
            "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID",
            "overrides": {
                "containerOverrides": [
                    {
                        "name": "sleep"
                    }
                ]
            },
            "lastStatus": "RUNNING",
            "containerInstanceArn": "arn:aws:ecs:us-east-1:aws_account_id:container-
instance/container_instance_ID",
            "clusterArn": "arn:aws:ecs:us-east-1:aws_account_id:cluster/default",
            "desiredStatus": "RUNNING",
            "taskDefinitionArn": "arn:aws:ecs:us-east-1:aws_account_id:task-definition/
sleep360:1",
            "containers": [
                {
```

```
                    "containerArn": "arn:aws:ecs:us-
east-1:aws_account_id:container/container_ID",
                    "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID",
                    "lastStatus": "RUNNING",
                    "name": "sleep",
                    "networkBindings": []
                }
            ]
        }
    ]
}
```

Menggunakan penskalaan otomatis cluster dengan AWS Management Console dan konsol Amazon ECS

Tutorial ini memandu Anda menciptakan sumber daya untuk Auto Scaling klaster menggunakan AWS Management Console. Saat sumber daya membutuhkan nama, kita akan menggunakan prefiks `ConsoleTutorial` untuk memastikan mereka semua memiliki nama yang unik dan lebih mudah untuk menemukan mereka.

Topik

Prasyarat

Jika mengikuti tutorial ini, berarti prasyarat berikut telah selesai:

- Langkah-langkah di Siapkan untuk menggunakan Amazon ECS telah selesai.
- AWS Pengguna Anda memiliki izin yang diperlukan yang ditentukan dalam contoh kebijakan Amazonecs_ FullAccess IAM.

- Peran IAM instans penampung Amazon ECS dibuat. Untuk informasi selengkapnya, lihat Peran IAM instans wadah Amazon ECS.
- Peran IAM terkait layanan Amazon ECS dibuat. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS.
- Peran IAM terkait layanan Auto Scaling dibuat. Untuk informasi selengkapnya, lihat Peran Tertaut Layanan untuk Penskalaan Otomatis Amazon EC2 di Panduan Pengguna Auto Scaling Amazon EC2.
- Anda memiliki VPC dan grup keamanan yang dibuat untuk digunakan. Untuk informasi selengkapnya, lihat the section called “Buat virtual private cloud”.

Langkah 1: Buat cluster Amazon ECS

Gunakan langkah-langkah berikut untuk membuat cluster Amazon ECS.

Amazon ECS membuat template peluncuran Auto Scaling Amazon EC2 dan grup Auto Scaling atas nama Anda sebagai bagian dari tumpukan. AWS CloudFormation

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Clusters, lalu pilih Create cluster.
3. Di bawah konfigurasi Cluster, untuk nama Cluster, masukkan`ConsoleTutorial-cluster`.
4. Di bawah Infrastruktur, hapus AWS Fargate (tanpa server), lalu pilih instans Amazon EC2. Selanjutnya, konfigurasikan grup Auto Scaling yang bertindak sebagai penyedia kapasitas.
    - Di bawah grup Auto Scaling (ASG). Pilih Buat ASG baru, lalu berikan detail berikut tentang grup:
        - Untuk Sistem Operasi/Arsitektur, pilih Amazon Linux 2.
        - Untuk jenis instans EC2, pilih t3.nano.
        - Untuk Kapasitas, masukkan jumlah minimum dan jumlah maksimum instans yang akan diluncurkan di grup Auto Scaling.
5. (Opsional) Untuk mengelola tag cluster, memperluas Tag, dan kemudian melakukan salah satu operasi berikut:

    [Tambahkan tag] Pilih Tambah tag dan lakukan hal berikut:

    - Untuk Kunci, masukkan nama kunci.

- Untuk Nilai, masukkan nilai kunci.

[Hapus tag] Pilih Hapus di sebelah kanan Kunci dan Nilai tag.

6. Pilih Buat.

Langkah 2: Daftarkan definisi tugas

Sebelum Anda dapat menjalankan tugas di klaster Anda, Anda harus mendaftarkan ketentuan tugas. Ketentuan tugas adalah daftar kontainer yang dikelompokkan bersama. Contoh berikut adalah ketentuan tugas sederhana yang menggunakan citra `amazonlinux` dari Docker Hub dan hanya tidur. Untuk informasi selengkapnya tentang parameter ketentuan tugas yang tersedia, lihat Definisi tugas Amazon ECS.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Definisi tugas.
3. Pilih Buat definisi tugas baru, Buat definisi tugas baru dengan JSON.
4. Di kotak editor JSON, tempel konten berikut.

```
{
    "family": "ConsoleTutorial-taskdef",
    "containerDefinitions": [
        {
            "name": "sleep",
            "image": "amazonlinux:2",
            "memory": 20,
            "essential": true,
            "command": [
                "sh",
                "-c",
                "sleep infinity"
            ]
        }
    ],
    "requiresCompatibilities": [
        "EC2"
    ]
}
```

5. Pilih Buat.

Langkah 3: Jalankan tugas

Setelah Anda mendaftarkan ketentuan tugas dari akun Anda, Anda dapat menjalankan tugas-tugas di klaster Anda. Untuk tutorial ini, Anda menjalankan lima instans dari ketentuan tugas `ConsoleTutorial-taskdef` di perangkat klaster `ConsoleTutorial-cluster` Anda.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih ConsoleTutorial-cluster.
3. Di bawah Tugas, pilih Jalankan tugas baru.
4. Di bagian Lingkungan, di bawah Opsi komputasi, pilih Strategi penyedia kapasitas.
5. Di bawah konfigurasi Deployment, untuk tipe Aplikasi, pilih Tugas.
6. Pilih ConsoleTutorial-taskdef dari daftar dropdown Keluarga.
7. Di bawah Tugas yang diinginkan, masukkan 5.
8. Pilih Buat.

Langkah 4: Verifikasi

Pada titik ini dalam tutorial, Anda harus memiliki cluster dengan lima tugas berjalan dan grup Auto Scaling dengan penyedia kapasitas. Penyedia kapasitas mengaktifkan penskalaan terkelola Amazon ECS.

Kami dapat memverifikasi bahwa semuanya berfungsi dengan baik dengan melihat CloudWatch metrik, pengaturan grup Auto Scaling, dan terakhir jumlah tugas cluster Amazon ECS.

Untuk melihat CloudWatch metrik untuk klaster Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada bilah navigasi di bagian atas layar, pilih Wilayah .
3. Pada panel navigasi, di bawah Metrik, pilih Semua metrik.
4. Pada halaman Semua metrik, di bawah tab Browse, pilih`AWS/ECS/ManagedScaling`.
5. Pilih CapacityProviderName, ClusterName.
6. Pilih kotak centang yang sesuai dengan `ConsoleTutorial-cluster` ClusterName.
7. Di bawah tab Metrik grafik, ubah Periode menjadi 30 detik dan Statistik ke Maksimum.

   Nilai yang ditampilkan dalam grafik menunjukkan nilai kapasitas target untuk penyedia kapasitas. Ini harus dimulai pada `100`, yang merupakan kapasitas target persen yang kami atur. Anda

harus melihatnya menaikkan skala hingga `200`, yang akan memicu alarm kebijakan penskalaan pelacakan target. Alarm kemudian akan memicu grup Auto Scaling untuk skala keluar.

Gunakan langkah-langkah berikut untuk melihat detail grup Auto Scaling untuk mengonfirmasi bahwa tindakan penskalaan telah terjadi.

Untuk memverifikasi grup Auto Scaling yang diperkecil

1. Buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.
2. Pada bilah navigasi di bagian atas layar, pilih Wilayah .
3. Pada panel navigasi, di bawah Auto Scaling, pilih Grup Auto Scaling.
4. Pilih grup `ConsoleTutorial-cluster` Auto Scaling yang dibuat dalam tutorial ini. Lihat nilai di bawah Kapasitas yang diinginkan dan lihat instance di bawah tab Manajemen instans untuk mengonfirmasi grup Anda diskalakan menjadi dua instance.

Gunakan langkah-langkah berikut untuk melihat kluster Amazon ECS untuk mengonfirmasi bahwa instans Amazon EC2 telah terdaftar di klaster dan tugas Anda dialihkan ke status. `RUNNING`

Untuk memverifikasi instans dalam grup Auto Scaling

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih `ConsoleTutorial-cluster` cluster.
4. Pada tab Tugas, konfirmasikan bahwa Anda melihat lima tugas dalam `RUNNING` status.

Langkah 5: Bersihkan

Setelah Anda menyelesaikan tutorial ini, bersihkan sumber daya yang terkait dengannya untuk menghindari biaya yang tidak Anda gunakan. Menghapus penyedia kapasitas dan ketentuan tugas tidak didukung, tetapi tidak ada biaya yang terkait dengan sumber daya ini.

Untuk membersihkan sumber daya tutorial

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih ConsoleTutorial-cluster.

4. Pada halaman ConsoleTutorial-cluster, pilih tab Tasks, lalu pilih Stop, Stop all.
5. Pada panel navigasi, silakan pilih Klaster.
6. Pada halaman Clusters, pilih ConsoleTutorial-cluster.
7. Di kanan atas halaman, pilih Hapus klaster.
8. Di kotak konfirmasi, masukkan delete ConsoleTutorial-cluster dan pilih Delete.
9. Hapus grup Auto Scaling menggunakan langkah-langkah berikut.
    a. Buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.
    b. Pada bilah navigasi di bagian atas layar, pilih Wilayah .
    c. Pada panel navigasi, di bawah Auto Scaling, pilih Grup Auto Scaling.
    d. Pilih grup **`ConsoleTutorial-cluster`** Auto Scaling, lalu pilih Tindakan.
    e. Dari menu Tindakan, pilih Hapus. Masukkan hapus di kotak konfirmasi dan kemudian pilih Hapus.

Menentukan Data Sensitif Menggunakan Rahasia Secrets Manager

Amazon ECS memungkinkan Anda untuk menyuntikkan data sensitif ke dalam container Anda dengan menyimpan data sensitif Anda dalam AWS Secrets Manager rahasia dan kemudian mereferensikannya dalam definisi container Anda. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah.

Tutorial berikut menunjukkan cara membuat rahasia Secrets Manager, mereferensikan rahasia dalam definisi tugas Amazon ECS, dan kemudian memverifikasi itu bekerja dengan menanyakan variabel lingkungan di dalam wadah yang menunjukkan isi rahasia.

Prasyarat

Jika mengikuti tutorial ini, berarti prasyarat berikut telah selesai:

- Langkah-langkah di Siapkan untuk menggunakan Amazon ECS telah selesai.
- AWS Pengguna Anda memiliki izin IAM yang diperlukan untuk membuat Secrets Manager dan sumber daya Amazon ECS yang dijelaskan.

Langkah 1: Buat Rahasia Secrets Manager

Anda dapat menggunakan konsol Secrets Manager untuk membuat rahasia untuk data sensitif Anda. Dalam tutorial ini kita akan menciptakan rahasia dasar untuk menyimpan nama pengguna dan kata sandi untuk mereferensi nanti dalam sebuah kontainer. Untuk informasi lebih lanjut, lihat Membuat Rahasia Dasar dalam Panduan Pengguna AWS Secrets Manager .

Pasangan kunci/nilai yang akan disimpan dalam rahasia ini adalah nilai variabel lingkungan dalam wadah Anda di akhir tutorial.

Simpan ARN Rahasia untuk referensi dalam kebijakan IAM eksekusi tugas Anda dan definisi tugas di langkah selanjutnya.

Langkah 2: Perbarui Peran IAM Eksekusi Tugas Anda

Agar Amazon ECS dapat mengambil data sensitif dari rahasia Secrets Manager Anda, Anda harus memiliki peran eksekusi tugas Amazon ECS dan mereferensikannya dalam definisi tugas Anda. Hal ini memungkinkan agen kontainer untuk menarik sumber daya Secrets Manager yang diperlukan. Jika Anda belum membuat peran IAM eksekusi tugas, lihatPeran IAM eksekusi tugas Amazon ECS.

Langkah-langkah berikut mengasumsikan Anda sudah memiliki peran IAM eksekusi tugas yang dibuat dan dikonfigurasi dengan benar.

Untuk memperbarui peran IAM eksekusi tugas

Gunakan konsol IAM untuk memperbarui peran eksekusi tugas Anda dengan izin yang diperlukan.

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Peran.
3. Cari daftar peran untuk `ecsTaskExecutionRole` dan pilih peran itu.
4. Pilih Izin, Tambahkan kebijakan inline.
5. Pilih tab JSON dan tentukan teks JSON berikut, pastikan Anda menentukan ARN lengkap rahasia Secrets Manager yang Anda buat di langkah 1.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [

 "arn:aws:secretsmanager:region:aws_account_id:secret:username_value"
            ]
        }
    ]
}
```

6. Pilih Tinjau kebijakan. Untuk Nama tentukan `ECSSecretsTutorial`, lalu pilih Buat kebijakan.

Langkah 3: Buat Definisi Tugas Amazon ECS

Anda dapat menggunakan konsol Amazon ECS untuk membuat definisi tugas yang mereferensikan rahasia Secrets Manager.

Untuk membuat ketentuan tugas yang menetapkan rahasia

Gunakan konsol IAM untuk memperbarui peran eksekusi tugas Anda dengan izin yang diperlukan.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Definisi tugas.
3. Pilih Buat definisi tugas baru, Buat definisi tugas baru dengan JSON.
4. Di kotak editor JSON, masukkan teks JSON definisi tugas berikut, pastikan Anda menentukan ARN lengkap rahasia Secrets Manager yang Anda buat di langkah 1 dan peran IAM eksekusi tugas yang Anda perbarui di langkah 2. Pilih Simpan.
5.

```
{
    "executionRoleArn": "arn:aws:iam::aws_account_id:role/ecsTaskExecutionRole",
    "containerDefinitions": [
        {
            "entryPoint": [
                "sh",
                "-c"
            ],
            "portMappings": [
                {
                    "hostPort": 80,
                    "protocol": "tcp",
                    "containerPort": 80
```

```
                }
            ],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
            ],
            "cpu": 10,
            "secrets": [
                {
                    "valueFrom":
 "arn:aws:secretsmanager:region:aws_account_id:secret:username_value",
                    "name": "username_value"
                }
            ],
            "memory": 300,
            "image": "httpd:2.4",
            "essential": true,
            "name": "ecs-secrets-container"
        }
    ],
    "family": "ecs-secrets-tutorial"
}
```

6. Pilih Buat.

Langkah 4: Buat Cluster Amazon ECS

Anda dapat menggunakan konsol Amazon ECS untuk membuat klaster yang berisi instance container untuk menjalankan tugas. Jika Anda memiliki klaster yang ada dengan setidaknya satu instans kontainer yang terdaftar untuk itu dengan sumber daya yang tersedia untuk menjalankan satu instans dari ketentuan tugas yang dibuat untuk tutorial ini Anda dapat melompat ke langkah berikutnya.

Untuk tutorial ini kita akan membuat cluster dengan satu instance `t2.micro` container menggunakan Amazon ECS Amazon Linux 2 AMI yang dioptimalkan Amazon ECS.

Untuk informasi tentang cara membuat cluster untuk jenis peluncuran EC2, lihatthe section called “Membuat cluster untuk jenis peluncuran Amazon EC2 menggunakan konsol”.

Langkah 5: Jalankan Tugas Amazon ECS

Anda dapat menggunakan konsol Amazon ECS untuk menjalankan tugas menggunakan definisi tugas yang Anda buat. Untuk tutorial ini kita akan menjalankan tugas menggunakan tipe peluncuran EC2, menggunakan cluster yang kita buat pada langkah sebelumnya.

Untuk informasi tentang cara menjalankan tugas, lihatthe section called “Jalankan aplikasi sebagai tugas Amazon ECS”.

Langkah 6: Verifikasi

Anda dapat memverifikasi bahwa semua langkah berhasil diselesaikan dan variabel lingkungan dibuat dengan benar dalam kontainer Anda menggunakan langkah-langkah berikut.

Untuk memverifikasi bahwa variabel lingkungan telah dibuat

1. Cari IP publik atau alamat DNS untuk instans kontainer Anda.
   a. Buka konsol di https://console.aws.amazon.com/ecs/v2.
   b. Di panel navigasi, pilih Clusters, lalu pilih cluster yang Anda buat.
   c. Pilih Infrastruktur, lalu pilih instance container.
   d. Rekam IP Publik atau DNS Publik untuk instans Anda.
2. Jika Anda menggunakan komputer macOS atau Linux, sambungkan ke instans Anda dengan perintah berikut, ganti jalur ke kunci pribadi dan alamat publik untuk instans Anda:

   ```
   $ ssh -i /path/to/my-key-pair.pem ec2-user@ec2-198-51-100-1.compute-1.amazonaws.com
   ```

   Untuk informasi selengkapnya tentang menggunakan komputer Windows, lihat Menghubungkan ke Instans Linux Anda dari Windows Menggunakan PuTTY di Panduan Pengguna Amazon EC2 untuk Instans Linux.

   > **Important**
   >
   > Untuk informasi selengkapnya tentang masalah apa pun saat menyambung ke instans, lihat Memecahkan Masalah Menghubungkan ke Instans Anda di Panduan Pengguna Amazon EC2 untuk Instans Linux.

3. Cantumkan kontainer yang berjalan pada instans. Catat ID kontainer untuk kontainer `ecs-secrets-tutorial`.

```
docker ps
```

4. Terhubung ke kontainer `ecs-secrets-tutorial` menggunakan ID kontainer dari output dari langkah sebelumnya.

```
docker exec -it container_ID /bin/bash
```

5. Gunakan perintah `echo` untuk mencetak nilai variabel lingkungan.

```
echo $username_value
```

Jika tutorial berhasil, Anda akan melihat output sebagai berikut:

```
password_value
```

> **Note**
>
> Atau, Anda dapat mencantumkan semua variabel lingkungan dalam wadah Anda menggunakan perintah `env` (atau `printenv`).

Langkah 7: Bersihkan

Setelah selesai dengan tutorial ini, Anda harus membersihkan sumber daya yang terkait untuk menghindari timbulnya biaya untuk sumber daya yang tidak terpakai.

Untuk membersihkan sumber daya

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih cluster.
4. Pilih Hapus klaster.
5. Di kotak konfirmasi, masukkan hapus ***nama cluster***, lalu pilih Hapus.
6. Buka konsol IAM di https://console.aws.amazon.com/iam/.

7. Di panel navigasi, pilih Peran.
8. Cari daftar peran untuk `ecsTaskExecutionRole` dan pilih peran itu.
9. Pilih Izin, lalu pilih X di sebelah ECS SecretsTutorial. Pilih Hapus.
10. Buka konsol Secrets Manager di https://console.aws.amazon.com/secretsmanager/.
11. Pilih rahasia username_value yang Anda buat dan pilih Tindakan, Hapus rahasia.

Membuat layanan menggunakan Service Discovery

Penemuan layanan sekarang diintegrasikan ke dalam wizard Create Service di konsol Amazon ECS. Untuk informasi selengkapnya, lihat Membuat layanan menggunakan konsol.

Tutorial berikut menunjukkan cara membuat layanan ECS yang berisi tugas Fargate yang menggunakan penemuan layanan dengan. AWS CLI

Untuk daftar penemuan layanan dukungan Wilayah AWS tersebut, lihatPenemuan Layanan .

Untuk informasi tentang Daerah yang mendukung Fargate, lihat. the section called “AWS Wilayah Fargate”

Prasyarat

Sebelum Anda memulai tutorial ini, pastikan bahwa prasyarat berikut terpenuhi:

- Versi terbaru diinstal dan dikonfigurasi. AWS CLI Untuk informasi selengkapnya, silakan lihat Menginstal AWS Command Line Interface.
- Langkah-langkah yang dijelaskan Siapkan untuk menggunakan Amazon ECS sudah lengkap.
- AWS Pengguna Anda memiliki izin yang diperlukan yang ditentukan dalam contoh kebijakan Amazonecs_ FullAccess IAM.
- Anda telah membuat setidaknya satu VPC dan satu grup keamanan. Untuk informasi selengkapnya, lihat the section called “Buat virtual private cloud”.

Langkah 1: Buat sumber daya Penemuan Layanan di AWS Cloud Map

Ikuti langkah-langkah berikut untuk membuat namespace penemuan layanan dan layanan penemuan layanan Anda:

1. Buat namespace penemuan layanan Cloud Map pribadi. Contoh ini menciptakan namespace yang disebut. `tutorial` Ganti *vpc-abcd1234* dengan ID salah satu VPC Anda yang ada.

```
aws servicediscovery create-private-dns-namespace \
      --name tutorial \
      --vpc vpc-abcd1234
```

   Output dari perintah ini adalah sebagai berikut.

```
{
    "OperationId": "h2qe3s6dxftvvt7riu6lfy2f6c3jlhf4-je6chs2e"
}
```

2. Menggunakan `OperationId` dari output dari langkah sebelumnya, verifikasi bahwa namespace pribadi berhasil dibuat. Catat ID namespace karena Anda menggunakannya dalam perintah berikutnya.

```
aws servicediscovery get-operation \
      --operation-id h2qe3s6dxftvvt7riu6lfy2f6c3jlhf4-je6chs2e
```

   Outputnya adalah sebagai berikut.

```
{
    "Operation": {
        "Id": "h2qe3s6dxftvvt7riu6lfy2f6c3jlhf4-je6chs2e",
        "Type": "CREATE_NAMESPACE",
        "Status": "SUCCESS",
        "CreateDate": 1519777852.502,
        "UpdateDate": 1519777856.086,
        "Targets": {
           "NAMESPACE": "ns-uejictsjen2i4eeg"
        }
    }
}
```

3. Menggunakan `NAMESPACE` ID dari output dari langkah sebelumnya, buat layanan penemuan layanan. Contoh ini menciptakan layanan bernama`myapplication`. Catat ID layanan dan ARN karena Anda menggunakannya dalam perintah berikutnya.

```
aws servicediscovery create-service \
```

```
      --name myapplication \
      --dns-config "NamespaceId="ns-
uejictsjen2i4eeg",DnsRecords=[{Type="A",TTL="300"}]" \
      --health-check-custom-config FailureThreshold=1
```

Outputnya adalah sebagai berikut.

```
{
    "Service": {
       "Id": "srv-utcrh6wavdkggqtk",
        "Arn": "arn:aws:servicediscovery:region:aws_account_id:service/srv-
utcrh6wavdkggqtk",
        "Name": "myapplication",
        "DnsConfig": {
            "NamespaceId": "ns-uejictsjen2i4eeg",
            "DnsRecords": [
                {
                    "Type": "A",
                    "TTL": 300
                }
            ]
        },
        "HealthCheckCustomConfig": {
            "FailureThreshold": 1
        },
        "CreatorRequestId": "e49a8797-b735-481b-a657-b74d1d6734eb"
    }
}
```

Langkah 2: Buat sumber daya Amazon ECS

Ikuti langkah-langkah berikut untuk membuat klaster Amazon ECS, definisi tugas, dan layanan:

1. Buat cluster Amazon ECS. Contoh ini menciptakan sebuah cluster yang diberi nama`tutorial`.

```
aws ecs create-cluster \
      --cluster-name tutorial
```

2. Daftarkan definisi tugas yang kompatibel dengan Fargate dan gunakan mode `awsvpc` jaringan. Ikuti langkah-langkah ini:

a. Buat file yang diberi nama `fargate-task.json` dengan isi definisi tugas berikut.

```
{
    "family": "tutorial-task-def",
        "networkMode": "awsvpc",
        "containerDefinitions": [
            {
                "name": "sample-app",
                "image": "httpd:2.4",
                "portMappings": [
                    {
                        "containerPort": 80,
                        "hostPort": 80,
                        "protocol": "tcp"
                    }
                ],
                "essential": true,
                "entryPoint": [
                    "sh",
                    "-c"
                ],
                "command": [
                    "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style>
 </head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample
 App</h1> <h2>Congratulations!</h2> <p>Your application is now running on a
 container in Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/
htdocs/index.html && httpd-foreground\""
                ]
            }
        ],
        "requiresCompatibilities": [
            "FARGATE"
        ],
        "cpu": "256",
        "memory": "512"
}
```

b. Daftarkan definisi tugas menggunakan`fargate-task.json`.

```
aws ecs register-task-definition \
      --cli-input-json file://fargate-task.json
```

3. Buat layanan ECS dengan mengikuti langkah-langkah berikut:

   a. Buat file yang diberi nama `ecs-service-discovery.json` dengan konten layanan ECS yang Anda buat. Contoh ini menggunakan definisi tugas yang dibuat pada langkah sebelumnya. `awsvpcConfiguration` diperlukan karena ketentuan tugas contoh menggunakan mode jaringan `awsvpc`.

      Saat Anda membuat layanan ECS, tentukan jenis peluncuran Fargate, dan versi platform `LATEST` yang mendukung penemuan layanan. Ketika layanan penemuan layanan dibuat di AWS Cloud Map , `registryArn` adalah ARN dikembalikan. Itu `securityGroups` dan `subnets` harus milik VPC yang digunakan untuk membuat namespace Cloud Map. Anda dapat memperoleh grup keamanan dan ID subnet dari Konsol VPC Amazon.

```
{
    "cluster": "tutorial",
    "serviceName": "ecs-service-discovery",
    "taskDefinition": "tutorial-task-def",
    "serviceRegistries": [
       {
          "registryArn":
 "arn:aws:servicediscovery:region:aws_account_id:service/srv-utcrh6wavdkggqtk"
       }
    ],
    "launchType": "FARGATE",
    "platformVersion": "LATEST",
    "networkConfiguration": {
       "awsvpcConfiguration": {
          "assignPublicIp": "ENABLED",
          "securityGroups": [ "sg-abcd1234" ],
          "subnets": [ "subnet-abcd1234" ]
       }
    },
    "desiredCount": 1
}
```

   b. Buat layanan ECS Anda menggunakan`ecs-service-discovery.json`.

```
aws ecs create-service \
      --cli-input-json file://ecs-service-discovery.json
```

Langkah 3: Verifikasi Penemuan Layanan di AWS Cloud Map

Anda dapat memverifikasi bahwa semuanya dibuat dengan benar dengan menanyakan informasi penemuan layanan Anda. Setelah penemuan layanan dikonfigurasi, Anda dapat menggunakan operasi AWS Cloud Map API, atau menelepon `dig` dari instance dalam VPC Anda. Ikuti langkah-langkah ini:

1. Dengan menggunakan ID layanan penemuan layanan, cantumkan instance penemuan layanan. Catat ID instance (ditandai dengan huruf tebal) untuk pembersihan sumber daya.

   ```
   aws servicediscovery list-instances \
         --service-id srv-utcrh6wavdkggqtk
   ```

   Outputnya adalah sebagai berikut.

   ```
   {
       "Instances": [
           {
               "Id": "16becc26-8558-4af1-9fbd-f81be062a266",
               "Attributes": {
                   "AWS_INSTANCE_IPV4": "172.31.87.2"
                   "AWS_INSTANCE_PORT": "80",
                   "AVAILABILITY_ZONE": "us-east-1a",
                   "REGION": "us-east-1",
                   "ECS_SERVICE_NAME": "ecs-service-discovery",
                   "ECS_CLUSTER_NAME": "tutorial",
                   "ECS_TASK_DEFINITION_FAMILY": "tutorial-task-def"
               }
           }
       ]
   }
   ```

2. Gunakan namespace penemuan layanan, layanan, dan parameter tambahan seperti nama cluster ECS untuk menanyakan detail tentang instance penemuan layanan.

   ```
   aws servicediscovery discover-instances \
         --namespace-name tutorial \
         --service-name myapplication \
         --query-parameters ECS_CLUSTER_NAME=tutorial
   ```

3. Catatan DNS yang dibuat di zona host Route 53 untuk layanan penemuan layanan dapat ditanyakan dengan perintah berikut: AWS CLI

   a. Menggunakan ID namespace, dapatkan informasi tentang namespace, yang mencakup ID zona yang dihosting Route 53.

   ```
   aws servicediscovery \
         get-namespace --id ns-uejictsjen2i4eeg
   ```

   Outputnya adalah sebagai berikut.

   ```
   {
       "Namespace": {
           "Id": "ns-uejictsjen2i4eeg",
           "Arn": "arn:aws:servicediscovery:region:aws_account_id:namespace/ns-
   uejictsjen2i4eeg",
           "Name": "tutorial",
           "Type": "DNS_PRIVATE",
           "Properties": {
                "DnsProperties": {
                   "HostedZoneId": "Z35JQ4ZFDRYPLV"
               }
           },
           "CreateDate": 1519777852.502,
           "CreatorRequestId": "9049a1d5-25e4-4115-8625-96dbda9a6093"
       }
   }
   ```

   b. Menggunakan ID zona yang dihosting Route 53 dari langkah sebelumnya (lihat teks dalam huruf tebal), dapatkan catatan sumber daya yang ditetapkan untuk zona yang dihosting.

   ```
   aws route53 list-resource-record-sets \
         --hosted-zone-id Z35JQ4ZFDRYPLV
   ```

4. Anda juga dapat menanyakan DNS dari instance dalam `dig` VPC Anda menggunakan.

   ```
   dig +short myapplication.tutorial
   ```

Langkah 4: Membersihkan

Setelah selesai dengan tutorial ini, bersihkan sumber daya terkait untuk menghindari biaya untuk sumber daya yang tidak digunakan. Ikuti langkah-langkah ini:

1. Batalkan pendaftaran instance layanan penemuan layanan menggunakan ID layanan dan ID instans yang Anda catat sebelumnya.

```
aws servicediscovery deregister-instance \
      --service-id srv-utcrh6wavdkggqtk \
      --instance-id 16becc26-8558-4af1-9fbd-f81be062a266
```

   Outputnya adalah sebagai berikut.

```
{
    "OperationId": "xhu73bsertlyffhm3faqi7kumsmx274n-jh0zimzv"
}
```

2. Menggunakan `OperationId` dari output dari langkah sebelumnya, verifikasi bahwa instance layanan penemuan layanan berhasil dideregistrasi.

```
aws servicediscovery get-operation \
      --operation-id xhu73bsertlyffhm3faqi7kumsmx274n-jh0zimzv
```

```
{
  "Operation": {
        "Id": "xhu73bsertlyffhm3faqi7kumsmx274n-jh0zimzv",
        "Type": "DEREGISTER_INSTANCE",
        "Status": "SUCCESS",
        "CreateDate": 1525984073.707,
        "UpdateDate": 1525984076.426,
        "Targets": {
            "INSTANCE": "16becc26-8558-4af1-9fbd-f81be062a266",
            "ROUTE_53_CHANGE_ID": "C5NSRG1J4I1FH",
            "SERVICE": "srv-utcrh6wavdkggqtk"
        }
    }
}
```

3. Hapus layanan penemuan layanan menggunakan ID layanan.

```
aws servicediscovery delete-service \
      --id srv-utcrh6wavdkggqtk
```

4. Hapus namespace penemuan layanan menggunakan ID namespace.

```
aws servicediscovery delete-namespace \
      --id ns-uejictsjen2i4eeg
```

Outputnya adalah sebagai berikut.

```
{
    "OperationId": "c3ncqglftesw4ibgj5baz6ktaoh6cg4t-jh0ztysj"
}
```

5. Menggunakan `OperationId` dari output dari langkah sebelumnya, verifikasi bahwa namespace penemuan layanan berhasil dihapus.

```
aws servicediscovery get-operation \
      --operation-id c3ncqglftesw4ibgj5baz6ktaoh6cg4t-jh0ztysj
```

Outputnya adalah sebagai berikut.

```
{
    "Operation": {
        "Id": "c3ncqglftesw4ibgj5baz6ktaoh6cg4t-jh0ztysj",
        "Type": "DELETE_NAMESPACE",
        "Status": "SUCCESS",
        "CreateDate": 1525984602.211,
        "UpdateDate": 1525984602.558,
        "Targets": {
            "NAMESPACE": "ns-rymlehshst7hhukh",
            "ROUTE_53_CHANGE_ID": "CJP2A2M86XW3O"
        }
    }
}
```

6. Perbarui hitungan yang diinginkan untuk layanan Amazon ECS ke`0`. Anda harus melakukan ini untuk menghapus layanan di langkah berikutnya.

```
aws ecs update-service \
```

```
      --cluster tutorial \
      --service ecs-service-discovery \
      --desired-count 0
```

7. Hapus layanan Amazon ECS.

```
aws ecs delete-service \
      --cluster tutorial \
      --service ecs-service-discovery
```

8. Hapus cluster Amazon ECS.

```
aws ecs delete-cluster \
      --cluster tutorial
```

Membuat layanan menggunakan penerapan biru/hijau

Tutorial berikut menunjukkan cara membuat layanan Amazon ECS yang berisi tugas Fargate yang menggunakan jenis penyebaran biru/hijau dengan. AWS CLI

> **Note**
>
> Dukungan untuk melakukan deployment biru/hijau telah ditambahkan untuk AWS CloudFormation. Untuk informasi selengkapnya, lihat Menjalankan penerapan biru/hijau Amazon ECS melalui CodeDeploy penggunaan AWS CloudFormation di Panduan Pengguna.AWS CloudFormation

Prasyarat

Tutorial ini mengasumsikan bahwa Anda telah menyelesaikan prasyarat berikut:

- Versi terbaru diinstal dan dikonfigurasi. AWS CLI Untuk informasi selengkapnya tentang menginstal atau memutakhirkan AWS CLI, lihat Menginstal. AWS Command Line Interface
- Langkah-langkah di Siapkan untuk menggunakan Amazon ECS telah diselesaikan.
- AWS Pengguna Anda memiliki izin yang diperlukan yang ditentukan dalam contoh kebijakan Amazonecs_ FullAccess IAM.

- Anda memiliki VPC dan grup keamanan yang dibuat untuk digunakan. Untuk informasi selengkapnya, lihat the section called “Buat virtual private cloud”.
- Peran Amazon ECS CodeDeploy IAM dibuat. Untuk informasi selengkapnya, lihat Peran Amazon ECS CodeDeploy IAM.

Langkah 1: Buat Application Load Balancer

Layanan Amazon ECS yang menggunakan tipe penyebaran biru/hijau memerlukan penggunaan Application Load Balancer atau Network Load Balancer. Tutorial ini menggunakan Application Load Balancer.

Untuk membuat Application Load Balancer

1. Gunakan create-load-balancerperintah untuk membuat Application Load Balancer. Tentukan dua subnet yang bukan dari Availability Zone serta grup keamanan yang sama.

```
aws elbv2 create-load-balancer \
     --name bluegreen-alb \
     --subnets subnet-abcd1234 subnet-abcd5678 \
     --security-groups sg-abcd1234 \
     --region us-east-1
```

   Ouput tersebut mencakup Amazon Resource Name (ARN) dari penyeimbang beban, dengan format berikut:

```
arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642
```

2. Gunakan create-target-groupperintah untuk membuat grup target. Grup target ini akan merutekan lalu lintas ke tugas asli yang ditetapkan dalam layanan Anda.

```
aws elbv2 create-target-group \
     --name bluegreentarget1 \
     --protocol HTTP \
     --port 80 \
     --target-type ip \
     --vpc-id vpc-abcd1234 \
     --region us-east-1
```

Output termasuk ARN dari grup target, dengan format berikut:

```
arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget1/209a844cd01825a4
```

3. Menggunakan perintah create-listener untuk membuat listener penyeimbang beban dengan aturan default yang meneruskan permintaan ke grup target.

```
aws elbv2 create-listener \
     --load-balancer-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642 \
     --protocol HTTP \
     --port 80 \
     --default-actions
 Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup
bluegreentarget1/209a844cd01825a4 \
     --region us-east-1
```

Output tersebut mencakup ARN listener, dengan format berikut:

```
arn:aws:elasticloadbalancing:region:aws_account_id:listener/app/bluegreen-alb/
e5ba62739c16e642/665750bec1b03bd4
```

Langkah 2: Buat cluster Amazon ECS

Gunakan perintah create-cluster untuk membuat sebuah klaster bernama `tutorial-bluegreen-cluster` untuk digunakan.

```
aws ecs create-cluster \
     --cluster-name tutorial-bluegreen-cluster \
     --region us-east-1
```

Output tersebut mencakup klaster ARN, dengan format berikut:

```
arn:aws:ecs:region:aws_account_id:cluster/tutorial-bluegreen-cluster
```

Langkah 3: Daftarkan ketentuan tugas

Gunakan register-task-definitionperintah untuk mendaftarkan definisi tugas yang kompatibel dengan Fargate. Hal ini membutuhkan penggunaan mode jaringan `awsvpc`. Berikut ini adalah contoh ketentuan tugas yang digunakan untuk tutorial ini.

Pertama, buat file bernama `fargate-task.json` dengan isi berikut. Pastikan bahwa Anda menggunakan ARN untuk peran eksekusi tugas Anda. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

```
{
    "family": "tutorial-task-def",
        "networkMode": "awsvpc",
        "containerDefinitions": [
            {
                "name": "sample-app",
                "image": "httpd:2.4",
                "portMappings": [
                    {
                        "containerPort": 80,
                        "hostPort": 80,
                        "protocol": "tcp"
                    }
                ],
                "essential": true,
                "entryPoint": [
                    "sh",
                    "-c"
                ],
                "command": [
                    "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #00FFFF;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon
 ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html && httpd-
foreground\""
                ]
            }
        ],
        "requiresCompatibilities": [
            "FARGATE"
        ],
        "cpu": "256",
```

```
        "memory": "512",
        "executionRoleArn": "arn:aws:iam::aws_account_id:role/ecsTaskExecutionRole"
}
```

Kemudian daftarkan ketentuan tugas menggunakan file `fargate-task.json` yang Anda buat.

```
aws ecs register-task-definition \
     --cli-input-json file://fargate-task.json \
     --region us-east-1
```

Langkah 4: Buat layanan Amazon ECS

Gunakan perintah create-service untuk membuat layanan.

Pertama, buat file bernama `service-bluegreen.json` dengan konten berikut.

```
{
    "cluster": "tutorial-bluegreen-cluster",
    "serviceName": "service-bluegreen",
    "taskDefinition": "tutorial-task-def",
    "loadBalancers": [
        {
            "targetGroupArn":
 "arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget1/209a844cd01825a4",
            "containerName": "sample-app",
            "containerPort": 80
        }
    ],
    "launchType": "FARGATE",
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "CODE_DEPLOY"
    },
    "platformVersion": "LATEST",
    "networkConfiguration": {
       "awsvpcConfiguration": {
          "assignPublicIp": "ENABLED",
          "securityGroups": [ "sg-abcd1234" ],
          "subnets": [ "subnet-abcd1234", "subnet-abcd5678" ]
       }
    },
    "desiredCount": 1
```

```
}
```

Kemudian buat layanan Anda menggunakan file `service-bluegreen.json` yang Anda buat.

```
aws ecs create-service \
     --cli-input-json file://service-bluegreen.json \
     --region us-east-1
```

Output tersebut mencakup ARN layanan, dengan format berikut:

```
arn:aws:ecs:region:aws_account_id:service/service-bluegreen
```

Dapatkan nama DNS penyeimbang beban menggunakan perintah berikut.

```
 aws elbv2 describe-load-balancers --name bluegreen-alb  --query
 'LoadBalancers[*].DNSName'
```

Masukkan nama DNS di browser web Anda dan Anda akan melihat halaman web yang menampilkan aplikasi sampel dengan latar belakang biru.

Langkah 5: Buat sumber daya AWS CodeDeploy

Gunakan langkah-langkah berikut untuk membuat CodeDeploy aplikasi Anda, grup target Application Load Balancer untuk grup CodeDeploy deployment, dan grup deployment. CodeDeploy

Untuk membuat CodeDeploy sumber daya

1. Gunakan perintah create-application untuk membuat aplikasi. CodeDeploy Tentukan platform komputasi ECS.

   ```
   aws deploy create-application \
        --application-name tutorial-bluegreen-app \
        --compute-platform ECS \
        --region us-east-1
   ```

   Output termasuk ID aplikasi, dengan format berikut:

   ```
   {
       "applicationId": "b8e9c1ef-3048-424e-9174-885d7dc9dc11"
   }
   ```

2. Gunakan create-target-groupperintah untuk membuat grup target Application Load Balancer kedua, yang akan digunakan saat membuat grup CodeDeploy penyebaran Anda.

```
aws elbv2 create-target-group \
     --name bluegreentarget2 \
     --protocol HTTP \
     --port 80 \
     --target-type ip \
     --vpc-id "vpc-0b6dd82c67d8012a1" \
     --region us-east-1
```

Output termasuk ARN untuk grup target, dengan format berikut:

```
arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget2/708d384187a3cfdc
```

3. Gunakan create-deployment-groupperintah untuk membuat grup CodeDeploy penyebaran.

Pertama, buat file bernama `tutorial-deployment-group.json` dengan konten berikut. Contoh ini menggunakan sumber daya yang Anda buat. Untuk`serviceRoleArn`, tentukan ARN peran Amazon ECS CodeDeploy IAM Anda. Untuk informasi selengkapnya, lihat Peran Amazon ECS CodeDeploy IAM.

```
{
   "applicationName": "tutorial-bluegreen-app",
   "autoRollbackConfiguration": {
      "enabled": true,
      "events": [ "DEPLOYMENT_FAILURE" ]
   },
   "blueGreenDeploymentConfiguration": {
      "deploymentReadyOption": {
         "actionOnTimeout": "CONTINUE_DEPLOYMENT",
         "waitTimeInMinutes": 0
      },
      "terminateBlueInstancesOnDeploymentSuccess": {
         "action": "TERMINATE",
         "terminationWaitTimeInMinutes": 5
      }
   },
   "deploymentGroupName": "tutorial-bluegreen-dg",
   "deploymentStyle": {
      "deploymentOption": "WITH_TRAFFIC_CONTROL",
```

```
      "deploymentType": "BLUE_GREEN"
   },
   "loadBalancerInfo": {
      "targetGroupPairInfoList": [
        {
          "targetGroups": [
             {
                 "name": "bluegreentarget1"
             },
             {
                 "name": "bluegreentarget2"
             }
          ],
          "prodTrafficRoute": {
              "listenerArns": [
                  "arn:aws:elasticloadbalancing:region:aws_account_id:listener/
app/bluegreen-alb/e5ba62739c16e642/665750bec1b03bd4"
              ]
          }
        }
      ]
   },
   "serviceRoleArn": "arn:aws:iam::aws_account_id:role/ecsCodeDeployRole",
   "ecsServices": [
       {
           "serviceName": "service-bluegreen",
           "clusterName": "tutorial-bluegreen-cluster"
       }
   ]
}
```

Kemudian buat grup CodeDeploy penyebaran.

```
aws deploy create-deployment-group \
     --cli-input-json file://tutorial-deployment-group.json \
     --region us-east-1
```

Output termasuk ID grup deployment, dengan format berikut:

```
{
    "deploymentGroupId": "6fd9bdc6-dc51-4af5-ba5a-0a4a72431c88"
}
```

Langkah 6: Buat dan pantau deployment CodeDeploy

Sebelum membuat CodeDeploy penerapan, perbarui definisi `command` tugas `fargate-task.json` sebagai berikut untuk mengubah warna latar belakang aplikasi sampel menjadi hijau.

```
{
    ...
        "containerDefinitions": [
            {
                ...
                "command": [
                    "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #097969;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon
 ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html && httpd-
foreground\""
                ]
            }
        ],
       ...
}
```

Daftarkan definisi tugas yang diperbarui menggunakan perintah berikut.

```
aws ecs register-task-definition \
     --cli-input-json file://fargate-task.json \
     --region us-east-1
```

Sekarang, gunakan langkah-langkah berikut untuk membuat dan mengunggah file spesifikasi aplikasi (AppSpec file) dan CodeDeploy penyebaran.

Untuk membuat dan memantau CodeDeploy penyebaran

1. Buat dan unggah AppSpec file menggunakan langkah-langkah berikut.

    a. Buat file bernama `appspec.yaml` dengan isi grup deployment CodeDeploy . Contoh ini menggunakan definisi tugas yang diperbarui.

    ```
    version: 0.0
    Resources:
      - TargetService:
    ```

```
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "arn:aws:ecs:region:aws_account_id:task-
definition/tutorial-task-def:2"
        LoadBalancerInfo:
          ContainerName: "sample-app"
          ContainerPort: 80
        PlatformVersion: "LATEST"
```

b. Gunakan perintah s3 mb untuk membuat bucket Amazon S3 untuk AppSpec file tersebut.

```
aws s3 mb s3://tutorial-bluegreen-bucket
```

c. Gunakan perintah s3 cp untuk mengunggah AppSpec file ke bucket Amazon S3.

```
aws s3 cp ./appspec.yaml s3://tutorial-bluegreen-bucket/appspec.yaml
```

2. Buat CodeDeploy penyebaran menggunakan langkah-langkah berikut.

a. Buat file bernama `create-deployment.json` dengan isi CodeDeploy penyebaran. Contoh ini menggunakan sumber daya yang Anda buat sebelumnya dalam tutorial.

```
{
    "applicationName": "tutorial-bluegreen-app",
    "deploymentGroupName": "tutorial-bluegreen-dg",
    "revision": {
        "revisionType": "S3",
        "s3Location": {
            "bucket": "tutorial-bluegreen-bucket",
            "key": "appspec.yaml",
            "bundleType": "YAML"
        }
    }
}
```

b. Gunakan perintah create-deployment untuk membuat deployment.

```
aws deploy create-deployment \
     --cli-input-json file://create-deployment.json \
     --region us-east-1
```

Output termasuk ID deployment, dengan format berikut:

```
{
    "deploymentId": "d-RPCR1U3TW"
}
```

3. Gunakan get-deployment-targetperintah untuk mendapatkan rincian penyebaran, menentukan `deploymentId` dari output sebelumnya.

```
aws deploy get-deployment-target \
--deployment-id "d-IMJU3A8TW" \
--target-id tutorial-bluegreen-cluster:service-bluegreen \
--region us-east-1
```

Awalnya, status penyebaran adalah`InProgress`. Lalu lintas diarahkan ke set tugas asli, yang memiliki `taskSetLabel` dari`BLUE`, status`PRIMARY`, dan a `trafficWeight``100.0`. Set tugas pengganti memiliki `taskSetLabel` dari`GREEN`, status`ACTIVE`, dan a `trafficWeight` dari`0.0`. Browser web tempat Anda memasukkan nama DNS masih menampilkan aplikasi sampel dengan latar belakang biru.

```
{
"deploymentTarget": {
"deploymentTargetType": "ECSTarget",
"ecsTarget": {
    "deploymentId": "d-RPCR1U3TW",
    "targetId": "tutorial-bluegreen-cluster:service-bluegreen",
    "targetArn": "arn:aws:ecs:region:aws_account_id:service/service-bluegreen",
    "lastUpdatedAt": "2023-08-10T12:07:24.797000-05:00",
    "lifecycleEvents": [
        {
            "lifecycleEventName": "BeforeInstall",
            "startTime": "2023-08-10T12:06:22.493000-05:00",
            "endTime": "2023-08-10T12:06:22.790000-05:00",
            "status": "Succeeded"
        },
        {
            "lifecycleEventName": "Install",
            "startTime": "2023-08-10T12:06:22.936000-05:00",
            "status": "InProgress"
        },
        {
            "lifecycleEventName": "AfterInstall",
            "status": "Pending"
```

```
        },
        {
            "lifecycleEventName": "BeforeAllowTraffic",
            "status": "Pending"
        },
        {
            "lifecycleEventName": "AllowTraffic",
            "status": "Pending"
        },
        {
            "lifecycleEventName": "AfterAllowTraffic",
            "status": "Pending"
        }
    ],
    "status": "InProgress",
    "taskSetsInfo": [
        {
            "identifer": "ecs-svc/9223370493423413672",
            "desiredCount": 1,
            "pendingCount": 0,
            "runningCount": 1,
            "status": "ACTIVE",
            "trafficWeight": 0.0,
            "targetGroup": {
                "name": "bluegreentarget2"
            },
            "taskSetLabel": "Green"
        },
        {
            "identifer": "ecs-svc/9223370493425779968",
            "desiredCount": 1,
            "pendingCount": 0,
            "runningCount": 1,
            "status": "PRIMARY",
            "trafficWeight": 100.0,
            "targetGroup": {
                "name": "bluegreentarget1"
            },
            "taskSetLabel": "Blue"
        }
    ]
}
}
```

```
}
```

Lanjutkan untuk mengambil rincian penyebaran menggunakan perintah sampai status penyebaran`Succeeded`, seperti yang ditunjukkan pada output berikut. Lalu lintas sekarang dialihkan ke set tugas pengganti, yang sekarang memiliki status `PRIMARY` dan a `trafficWeight` dari. `100.0` Segarkan browser web tempat Anda memasukkan nama DNS penyeimbang beban, dan sekarang Anda akan melihat aplikasi sampel dengan latar belakang hijau.

```
{
"deploymentTarget": {
"deploymentTargetType": "ECSTarget",
"ecsTarget": {
    "deploymentId": "d-RPCR1U3TW",
    "targetId": "tutorial-bluegreen-cluster:service-bluegreen",
    "targetArn": "arn:aws:ecs:region:aws_account_id:service/service-bluegreen",
    "lastUpdatedAt": "2023-08-10T12:07:24.797000-05:00",
    "lifecycleEvents": [
        {
            "lifecycleEventName": "BeforeInstall",
            "startTime": "2023-08-10T12:06:22.493000-05:00",
            "endTime": "2023-08-10T12:06:22.790000-05:00",
            "status": "Succeeded"
        },
        {
            "lifecycleEventName": "Install",
            "startTime": "2023-08-10T12:06:22.936000-05:00",
            "endTime": "2023-08-10T12:08:25.939000-05:00",
            "status": "Succeeded"
        },
        {
            "lifecycleEventName": "AfterInstall",
            "startTime": "2023-08-10T12:08:26.089000-05:00",
            "endTime":  "2023-08-10T12:08:26.403000-05:00",
            "status": "Succeeded"
        },
        {
            "lifecycleEventName": "BeforeAllowTraffic",
            "startTime": "2023-08-10T12:08:26.926000-05:00",
            "endTime":  "2023-08-10T12:08:27.256000-05:00",
            "status": "Succeeded"
        },
```

```
        {
            "lifecycleEventName": "AllowTraffic",
            "startTime": "2023-08-10T12:08:27.416000-05:00",
            "endTime": "2023-08-10T12:08:28.195000-05:00",
            "status": "Succeeded"
        },
        {
            "lifecycleEventName": "AfterAllowTraffic",
            "startTime": "2023-08-10T12:08:28.715000-05:00",
            "endTime":  "2023-08-10T12:08:28.994000-05:00",
            "status": "Succeeded"
        }
    ],
    "status": "Succeeded",
    "taskSetsInfo": [
        {
            "identifer": "ecs-svc/9223370493425779968",
            "desiredCount": 1,
            "pendingCount": 0,
            "runningCount": 1,
            "status": "ACTIVE",
            "trafficWeight": 0.0,
            "targetGroup": {
                "name": "bluegreentarget1"
            },
            "taskSetLabel": "Blue"
        },
        {
            "identifer": "ecs-svc/9223370493423413672",
            "desiredCount": 1,
            "pendingCount": 0,
            "runningCount": 1,
            "status": "PRIMARY",
            "trafficWeight": 100.0,
            "targetGroup": {
                "name": "bluegreentarget2"
            },
            "taskSetLabel": "Green"
        }
    ]
}
}
}
```

Langkah 7: Bersihkan

Bila Anda telah menyelesaikan tutorial ini, bersihkan sumber daya yang terkait dengannya untuk menghindari biaya untuk sumber daya yang tidak Anda gunakan.

Membersihkan sumber daya tutorial

1. Gunakan delete-deployment-groupperintah untuk menghapus grup CodeDeploy penyebaran.

```
aws deploy delete-deployment-group \
     --application-name tutorial-bluegreen-app \
     --deployment-group-name tutorial-bluegreen-dg \
     --region us-east-1
```

2. Gunakan perintah hapus-aplikasi untuk menghapus aplikasi. CodeDeploy

```
aws deploy delete-application \
     --application-name tutorial-bluegreen-app \
     --region us-east-1
```

3. Gunakan perintah delete-service untuk menghapus layanan Amazon ECS. Menggunakan bendera `--force` mengizinkan Anda untuk menghapus layanan bahkan jika itu tidak menurunkan skala ke tugas nol.

```
aws ecs delete-service \
     --service arn:aws:ecs:region:aws_account_id:service/service-bluegreen \
     --force \
     --region us-east-1
```

4. Gunakan perintah delete-cluster untuk menghapus cluster Amazon ECS.

```
aws ecs delete-cluster \
     --cluster tutorial-bluegreen-cluster \
     --region us-east-1
```

5. Gunakan perintah s3 rm untuk menghapus AppSpec file dari bucket Amazon S3.

```
aws s3 rm s3://tutorial-bluegreen-bucket/appspec.yaml
```

6. Gunakan perintah s3 rb untuk menghapus bucket Amazon S3.

```
aws s3 rb s3://tutorial-bluegreen-bucket
```

7. Gunakan delete-load-balancerperintah untuk menghapus Application Load Balancer.

```
aws elbv2 delete-load-balancer \
     --load-balancer-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642 \
     --region us-east-1
```

8. Gunakan delete-target-groupperintah untuk menghapus dua kelompok target Application Load Balancer.

```
aws elbv2 delete-target-group \
     --target-group-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget1/209a844cd01825a4 \
     --region us-east-1
```

```
aws elbv2 delete-target-group \
     --target-group-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget2/708d384187a3cfdc \
     --region us-east-1
```

Mendengarkan Acara Amazon ECS CloudWatch

Dalam tutorial ini, Anda menyiapkan AWS Lambda fungsi sederhana yang mendengarkan peristiwa tugas Amazon ECS dan menuliskannya ke aliran CloudWatch log Log.

Prasyarat: Atur klaster pengujian

Jika Anda tidak memiliki klaster yang berjalan untuk menangkap peristiwa, ikuti langkah-langkah dalam the section called “Membuat cluster untuk jenis peluncuran Fargate dan Eksternal menggunakan konsol” untuk membuat klaster. Di akhir tutorial ini, Anda menjalankan tugas di cluster ini untuk menguji apakah Anda telah mengonfigurasi fungsi Lambda Anda dengan benar.

Langkah 1: Buat fungsi Lambda

Dalam prosedur ini, Anda membuat fungsi Lambda sederhana untuk berfungsi sebagai target pesan aliran acara Amazon ECS.

1. Buka AWS Lambda konsol di https://console.aws.amazon.com/lambda/.
2. Pilih Buat fungsi.
3. Pada layar Tulis dari awal, lakukan hal berikut:
   a. Untuk Nama, masukkan nilai.
   b. Untuk Runtime, pilih versi Python Anda, misalnya, Python 3.9.
   c. Untuk Peran, pilih Buat peran baru dengan izin Lambda dasar.
4. Pilih Buat fungsi.
5. Di bagian Kode fungsi, edit kode sampel untuk mencocokkan contoh berikut:

```
import json

def lambda_handler(event, context):
    if event["source"] != "aws.ecs":
       raise ValueError("Function only supports input from events with a source
 type of: aws.ecs")

    print('Here is the event:')
    print(json.dumps(event))
```

   Ini adalah fungsi Python 3.9 sederhana yang mencetak acara yang dikirim oleh Amazon ECS. Jika semuanya dikonfigurasi dengan benar, di akhir tutorial ini, Anda melihat bahwa detail peristiwa muncul di aliran CloudWatch log Log yang terkait dengan fungsi Lambda ini.
6. Pilih Simpan.

Langkah 2: Mendaftarkan aturan peristiwa

Selanjutnya, Anda membuat aturan CloudWatch acara Acara yang menangkap peristiwa tugas yang berasal dari kluster Amazon ECS Anda. Aturan ini menangkap semua peristiwa dari semua klaster dalam akun yang ditentukan. Pesan tugas sendiri berisi informasi sumber peristiwa, termasuk klaster tempat klaster berada, yang dapat Anda gunakan untuk memfilter dan mengurutkan peristiwa secara terprogram.

> **Note**
>
> Saat Anda menggunakan aturan AWS Management Console untuk membuat acara, konsol secara otomatis menambahkan izin IAM yang diperlukan untuk memberikan izin CloudWatch Acara untuk memanggil fungsi Lambda Anda. Jika Anda membuat aturan acara menggunakan AWS CLI, Anda harus memberikan izin ini secara eksplisit. Untuk informasi selengkapnya, lihat Peristiwa dan Pola Peristiwa di Panduan Pengguna CloudWatch Acara Amazon.

Untuk merutekan acara ke fungsi Lambda Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Peristiwa, Aturan, Buat aturan.
3. Untuk Sumber Peristiwa, pilih ECS sebagai sumber peristiwa. Secara default, aturan ini berlaku untuk semua peristiwa Amazon ECS untuk semua grup Amazon ECS Anda. Atau, Anda dapat memilih acara tertentu atau grup Amazon ECS tertentu.
4. Untuk Target, pilih Tambah target, untuk jenis Target, pilih fungsi Lambda, lalu pilih fungsi Lambda Anda.
5. Pilih Konfigurasikan detail.
6. Untuk Definisi aturan, tuliskan nama dan deskripsi untuk aturan Anda dan pilih Buat aturan.

Langkah 3: Membuat sebuah penetapan tugas

Buat definisi tugas.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Ketentuan Tugas.
3. Pilih Buat Definisi Tugas baru, Buat revisi baru dengan JSON.
4. Salin dan tempel contoh ketentuan tugas berikut ke dalam kotak dan kemudian pilih Simpan.

```
{
    "containerDefinitions": [
        {
            "entryPoint": [
                "sh",
```

```
                "-c"
            ],
            "portMappings": [
                {
                    "hostPort": 80,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
            ],
            "cpu": 10,
            "memory": 300,
            "image": "httpd:2.4",
            "name": "simple-app"
        }
    ],
    "family": "console-sample-app-static"
}
```

5. Pilih Buat.

Langkah 4: Uji aturan Anda

Terakhir, Anda membuat aturan CloudWatch acara Acara yang menangkap peristiwa tugas yang berasal dari kluster Amazon ECS Anda. Aturan ini menangkap semua peristiwa dari semua klaster dalam akun yang ditentukan. Pesan tugas berisi informasi tentang sumber peristiwa, termasuk klaster tempat klaster berada, yang dapat Anda gunakan untuk memfilter dan mengurutkan peristiwa secara terprogram.

Menguji aturan Anda

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pilih Definisi tugas.
3. Pilih console-sample-app-static, lalu pilih Deploy, Run new task.

4. Untuk Cluster, pilih default, lalu pilih Deploy.
5. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
6. Di panel navigasi, pilih Logs (Log) dan pilih grup log untuk fungsi Lambda Anda (misalnya, /aws/lambda/*my-function*).
7. Pilih stream log untuk melihat data peristiwa.

Mengirim peringatan Amazon Simple Notification Service untuk acara yang dihentikan tugas

Dalam tutorial ini, Anda mengonfigurasi aturan EventBridge acara Amazon yang hanya menangkap peristiwa tugas di mana tugas telah berhenti berjalan karena salah satu wadah esensialnya telah dihentikan. Acara hanya mengirimkan peristiwa tugas dengan `stoppedReason` properti tertentu ke topik Amazon SNS yang ditentukan.

Prasyarat: Atur klaster pengujian

Jika Anda tidak memiliki cluster yang sedang berjalan untuk menangkap peristiwa, ikuti langkah-langkah di Memulai konsol menggunakan kontainer Linux AWS Fargate untuk membuatnya. Di akhir tutorial ini, Anda menjalankan tugas di cluster ini untuk menguji bahwa Anda telah mengonfigurasi topik dan EventBridge aturan Amazon SNS Anda dengan benar.

Prasyarat: Konfigurasikan izin untuk Amazon SNS

Untuk memungkinkan mempublikasikan EventBridge ke topik Amazon SNS, gunakan perintah aws sns get-topic-attributes dan aws sns. set-topic-attributes

Untuk informasi tentang cara menambahkan izin, lihat izin Amazon SNS di Panduan Pengembang Layanan Pemberitahuan Sederhana Amazon

Tambahkan izin berikut:

```
{
  "Sid": "PublishEventsToMyTopic",
  "Effect": "Allow",
  "Principal": {
     "Service": "events.amazonaws.com"
  },
  "Action": "sns: Publish",
```

```
  "Resource": "arn:aws:sns:region:account-id:TaskStoppedAlert",
}
```

Langkah 1: Buat dan berlangganan ke topik Amazon SNS

Untuk tutorial ini, Anda mengonfigurasi topik Amazon SNS yang berfungsi sebagai target peristiwa untuk aturan peristiwa baru Anda.

Untuk informasi tentang cara membuat dan berlangganan topik Amazon SNS, lihat Memulai Amazon SNS di Panduan Pengembang Layanan Pemberitahuan Sederhana Amazon dan gunakan tabel berikut untuk menentukan opsi apa yang harus dipilih.

Opsi	Nilai
Tipe	Standar
Nama	TaskStoppedAlert
Protokol	Email
Titik Akhir	Alamat email yang saat ini Anda miliki akses

Langkah 2: Mendaftarkan aturan peristiwa

Selanjutnya, Anda mendaftarkan aturan peristiwa yang hanya menangkap peristiwa yang dihentikan tugas untuk tugas dengan kontainer yang dihentikan.

Untuk informasi tentang cara membuat dan berlangganan topik Amazon SNS, lihat Membuat aturan di Amazon EventBridge di Panduan EventBridge Pengguna Amazon dan gunakan tabel berikut untuk menentukan opsi apa yang harus dipilih.

Opsi	Nilai
Jenis aturan	Aturan dengan pola acara
Sumber peristiwa	AWS acara atau acara EventBridge mitra

Opsi	Nilai
Pola peristiwa	Pola kustom (editor JSON)

Pola peristiwa	<pre>{
	  "source":[
	  "aws.ecs"
	  ],
	  "detail-type":[
	  "ECS Task State Change"
	  ],
	  "detail":{
	  "lastStatus":[
	  "STOPPED"
	  ],
	  "stoppedReason":[
	  "Essential container in task exited"
	  ]
	  }
	}</pre>

Tipe target	AWS layanan
Target	Topik SNS
Topik	TaskStoppedAlert (Topik yang Anda buat di Langkah 1)

Langkah 3: Uji aturan Anda

Verifikasi bahwa aturan bekerja dengan menjalankan tugas yang keluar segera setelah dimulai. Jika aturan peristiwa Anda dikonfigurasi dengan benar, Anda menerima pesan email dalam beberapa menit dengan teks peristiwa. Jika Anda memiliki ketentuan tugas yang sudah ada yang dapat memenuhi persyaratan aturan, jalankan tugas dengan menggunakannya. Jika tidak memilikinya, langkah-langkah berikut akan memandu Anda mendaftarkan ketentuan tugas Fargate dan menjalankannya.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Definisi tugas.
3. Pilih Buat definisi tugas baru, Buat definisi tugas baru dengan JSON.
4. Di kotak editor JSON, edit file JSON Anda, salin yang berikut ini ke editor.

```
{
   "containerDefinitions":[
      {
         "command":[
            "sh",
            "-c",
            "sleep 5"
         ],
         "essential":true,
         "image":"amazonlinux:2",
         "name":"test-sleep"
      }
   ],
   "cpu":"256",
   "executionRoleArn":"arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
   "family":"fargate-task-definition",
   "memory":"512",
   "networkMode":"awsvpc",
   "requiresCompatibilities":[
      "FARGATE"
   ]
}
```

5. Pilih Buat.

Untuk menjalankan tugas dari konsol

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih cluster yang Anda buat dalam prasyarat.
3. Dari tab Tugas, pilih Jalankan tugas baru.
4. Untuk Jenis aplikasi, pilih Tugas.
5. Untuk definisi Tugas, pilih fargate-task-definition.
6. Untuk tugas yang diinginkan, masukkan jumlah tugas yang akan diluncurkan.
7. Pilih Buat.

Menggabungkan pesan log multiline atau stack-trace

Dimulai dengan AWS untuk Fluent Bit versi 2.22.0, filter multiline disertakan. Filter multiline membantu menggabungkan pesan log yang awalnya milik satu konteks tetapi dibagi menjadi beberapa catatan atau baris log. Untuk informasi selengkapnya tentang filter multiline, lihat dokumentasi Fluent Bit.

Contoh umum pesan log terpisah adalah:

- Jejak tumpukan.
- Aplikasi yang mencetak log pada beberapa baris.
- Log pesan yang dibagi karena lebih panjang dari ukuran buffer maks runtime yang ditentukan. Anda dapat menggabungkan pesan log yang dibagi berdasarkan runtime container dengan mengikuti contoh di GitHub: FireLens Contoh: Concatenate Partial/Split Container Logs.

Izin IAM yang diperlukan

Anda memiliki izin IAM yang diperlukan untuk agen penampung untuk menarik gambar kontainer dari Amazon ECR dan wadah untuk merutekan log ke Log. CloudWatch

Untuk izin ini, Anda harus memiliki peran berikut:

- Peran tugas IAM.
- Peran IAM eksekusi tugas Amazon ECS.

Cara menggunakan editor kebijakan JSON untuk membuat kebijakan

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi di sebelah kiri, pilih Kebijakan.

   Jika ini pertama kalinya Anda memilih Kebijakan, akan muncul halaman Selamat Datang di Kebijakan Terkelola. Pilih Memulai.
3. Di bagian atas halaman, pilih Buat kebijakan.
4. Di bagian Editor kebijakan, pilih opsi JSON.
5. Masukkan dokumen kebijakan JSON berikut:

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": [
   "logs:CreateLogStream",
   "logs:CreateLogGroup",
   "logs:PutLogEvents"
  ],
  "Resource": "*"
 }]
}
```

6. Pilih Berikutnya.

> **Note**
>
> Anda dapat beralih antara opsi editor Visual dan JSON kapan saja. Namun, jika Anda melakukan perubahan atau memilih Berikutnya di editor Visual, IAM dapat merestrukturisasi kebijakan Anda untuk mengoptimalkannya bagi editor visual. Untuk informasi selengkapnya, lihat Restrukturisasi kebijakan dalam Panduan Pengguna IAM.

7. Pada halaman Tinjau dan buat, masukkan Nama kebijakan dan Deskripsi (opsional) untuk kebijakan yang Anda buat. Tinjau Izin yang ditentukan dalam kebijakan ini untuk melihat izin yang diberikan oleh kebijakan Anda.
8. Pilih Buat kebijakan untuk menyimpan kebijakan baru Anda.

Tentukan kapan harus menggunakan pengaturan log multiline

Berikut ini adalah contoh cuplikan log yang Anda lihat di konsol CloudWatch Log dengan pengaturan log default. Anda dapat melihat garis yang dimulai dengan `log` untuk menentukan apakah Anda memerlukan filter multiline. Ketika konteksnya sama, Anda dapat menggunakan pengaturan log multiline, Dalam contoh ini, konteksnya adalah”

```
2022-09-20T15:47:56:595-05-00                          {"container_id":
 "82ba37cada1d44d389b03e78caf74faa-EXAMPLE", "container_name": "example-
app", "source=": "stdout", "log": ": "     at com.myproject.modele.
(MyProject.badMethod.java:22)",
    {
```

```
      "container_id":  "82ba37cada1d44d389b03e78caf74faa-EXAMPLE",
      "container_name: ": "example-app",
      "source": "stdout",
      "log": ": "     at com.myproject.model.MyProject.badMethod(MyProject.java:22)",
      "ecs_cluster": "default",
      "ecs_task_arn": "arn:aws:region:123456789012:task/default/
b23c940d29ed4714971cba72cEXAMPLE",
      "ecs_task_definition": "firelense-example-multiline:3"
     }
```

```
2022-09-20T15:47:56:595-05-00                          {"container_id":
 "82ba37cada1d44d389b03e78caf74faa-EXAMPLE", "container_name": "example-app", "stdout",
 "log": ": "     at com.myproject.modele.(MyProject.oneMoreMethod.java:18)",
    {
      "container_id":  "82ba37cada1d44d389b03e78caf74faa-EXAMPLE",
      "container_name: ": "example-app",
      "source": "stdout",
      "log": ": "     at
 com.myproject.model.MyProject.oneMoreMethod(MyProject.java:18)",
      "ecs_cluster": "default",
      "ecs_task_arn": "arn:aws:region:123456789012:task/default/
b23c940d29ed4714971cba72cEXAMPLE,
      "ecs_task_definition": "firelense-example-multiline:3"
     }
```

Setelah Anda menggunakan pengaturan log multiline, output akan terlihat mirip dengan contoh di bawah ini.

```
2022-09-20T15:47:56:595-05-00                          {"container_id":
 "82ba37cada1d44d389b03e78caf74faa-EXAMPLE", "container_name": "example-app",
 "stdout",...
    {
      "container_id":  "82ba37cada1d44d389b03e78caf74faa-EXAMPLE",
      "container_name: ": "example-app",
      "source": "stdout",
      "log:    "September 20, 2022 06:41:48 Exception in thread \"main\"
 java.lang.RuntimeException: Something has gone wrong, aborting!\n
    at com.myproject.module.MyProject.badMethod(MyProject.java:22)\n    at
    at com.myproject.model.MyProject.oneMoreMethod(MyProject.java:18)
 com.myproject.module.MyProject.main(MyProject.java:6)",
      "ecs_cluster": "default",
```

```
      "ecs_task_arn": "arn:aws:region:123456789012:task/default/
b23c940d29ed4714971cba72cEXAMPLE",
      "ecs_task_definition": "firelense-example-multiline:2"
     }
```

Parse dan concatenate pilihan

Untuk mengurai log dan menggabungkan baris yang terbagi karena baris baru, Anda dapat menggunakan salah satu dari dua opsi ini.

- Gunakan file parser Anda sendiri yang berisi aturan untuk mengurai dan menggabungkan baris yang termasuk dalam pesan yang sama.
- Gunakan parser bawaan Fluent Bit. Untuk daftar bahasa yang didukung oleh parser bawaan Fluent Bit, lihat Dokumentasi Bit Lancar.

Tutorial berikut memandu Anda melalui langkah-langkah untuk setiap kasus penggunaan. Langkah-langkahnya menunjukkan cara menggabungkan multiline dan mengirim log ke Amazon. CloudWatch Anda dapat menentukan tujuan yang berbeda untuk log Anda.

Contoh: Gunakan parser yang Anda buat

Dalam contoh ini, Anda akan menyelesaikan langkah-langkah berikut:

1. Bangun dan unggah gambar untuk wadah Fluent Bit.
2. Buat dan unggah gambar untuk aplikasi multiline demo yang berjalan, gagal, dan menghasilkan jejak tumpukan multiline.
3. Buat definisi tugas dan jalankan tugas.
4. Lihat log untuk memverifikasi bahwa pesan yang menjangkau beberapa baris tampak digabungkan.

Buat dan unggah gambar untuk wadah Fluent Bit

Gambar ini akan menyertakan file parser tempat Anda menentukan ekspresi reguler dan file konfigurasi yang mereferensikan file parser.

1. Buat folder dengan nama`FluentBitDockerImage`.

2. Di dalam folder, buat file parser yang berisi aturan untuk mengurai log dan menggabungkan baris yang termasuk dalam pesan yang sama.

   a. Tempel konten berikut di file parser:

```
[MULTILINE_PARSER]
    name          multiline-regex-test
    type          regex
    flush_timeout 1000
    #
    # Regex rules for multiline parsing
    # ---------------------------------
    #
    # configuration hints:
    #
    #  - first state always has the name: start_state
    #  - every field in the rule must be inside double quotes
    #
    # rules |   state name  | regex pattern                  | next state
    # ------|---------------|--------------------------------------------
    rule      "start_state"   "/(Dec \d+ \d+\:\d+\:\d+)(.*)/"  "cont"
    rule      "cont"          "/^\s+at.*/"                     "cont"
```

   Saat Anda menyesuaikan pola regex Anda, kami sarankan Anda menggunakan editor ekspresi reguler untuk menguji ekspresi.

   b. Simpan file sebagai `parsers_multiline.conf`.

3. Di dalam `FluentBitDockerImage` folder, buat file konfigurasi khusus yang mereferensikan file parser yang Anda buat di langkah sebelumnya.

   Untuk informasi selengkapnya tentang file konfigurasi kustom, lihat Menentukan file konfigurasi kustom di Panduan Pengembang Layanan Amazon Elastic Container

   a. Tempel konten berikut dalam file:

```
[SERVICE]
    flush                1
    log_level            info
    parsers_file         /parsers_multiline.conf

[FILTER]
    name                 multiline
```

```
    match                 *
    multiline.key_content log
    multiline.parser      multiline-regex-test
```

> **Note**
>
> Anda harus menggunakan jalur absolut parser.

b. Simpan file sebagai `extra.conf`.

4. Di dalam `FluentBitDockerImage` folder, buat Dockerfile dengan gambar Fluent Bit dan parser dan file konfigurasi yang Anda buat.

   a. Tempel konten berikut dalam file:

```
FROM public.ecr.aws/aws-observability/aws-for-fluent-bit:latest

ADD parsers_multiline.conf /parsers_multiline.conf
ADD extra.conf /extra.conf
```

   b. Simpan file sebagai `Dockerfile`.

5. Menggunakan Dockerfile, buat gambar Fluent Bit kustom dengan parser dan file konfigurasi khusus disertakan.

> **Note**
>
> Anda dapat menempatkan file parser dan file konfigurasi di mana saja di gambar Docker kecuali `/fluent-bit/etc/fluent-bit.conf` karena jalur file ini digunakan oleh. FireLens

   a. Bangun gambar: `docker build -t fluent-bit-multiline-image .`

      Dimana: `fluent-bit-multiline-image` adalah nama untuk gambar dalam contoh ini.

   b. Verifikasi bahwa gambar telah dibuat dengan benar: `docker images –filter reference=fluent-bit-multiline-image`

      Jika berhasil, output menunjukkan gambar dan `latest` tag.

6. Unggah gambar Fluent Bit kustom ke Amazon Elastic Container Registry.

a. Buat repositori Amazon ECR untuk menyimpan gambar: `aws ecr create-repository --repository-name fluent-bit-multiline-repo --region us-east-1`

   Dimana: `fluent-bit-multiline-repo` adalah nama untuk repositori dan `us-east-1` merupakan wilayah dalam contoh ini.

   Outputnya memberi Anda detail repositori baru.

b. Tandai gambar Anda dengan `repositoryUri` nilai dari output sebelumnya: `docker tag fluent-bit-multiline-image repositoryUri`

   Contoh: `docker tag fluent-bit-multiline-image xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-repo`

c. Jalankan docker image untuk memverifikasi itu berjalan dengan benar: `docker images –filter reference=repositoryUri`

   Dalam output, nama repositori berubah dari fluent-bit-multiline-repo ke. `repositoryUri`

d. Otentikasi ke Amazon ECR dengan menjalankan `aws ecr get-login-password` perintah dan menentukan ID registri yang ingin Anda autentikasi: `aws ecr get-login-password | docker login --username AWS --password-stdin registry ID.dkr.ecr.region.amazonaws.com`

   Contoh: `ecr get-login-password | docker login --username AWS --password-stdin xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com`

   Pesan login yang berhasil muncul.

e. Dorong gambar ke Amazon ECR: `docker push registry ID.dkr.ecr.region.amazonaws.com/repository name`

   Contoh: `docker push xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-repo`

Membangun dan meng-upload gambar untuk aplikasi demo multiline

Gambar ini akan mencakup file skrip Python yang menjalankan aplikasi dan file log sampel.

Saat Anda menjalankan tugas, aplikasi mensimulasikan berjalan, lalu gagal dan membuat jejak tumpukan.

1. Buat folder bernama`multiline-app`: `mkdir multiline-app`
2. Buat file skrip Python.
   a. Di dalam `multiline-app` folder, buat file dan beri nama`main.py`.
   b. Tempel konten berikut dalam file:

```
import os
import time
file1 = open('/test.log', 'r')
Lines = file1.readlines()

count = 0

for i in range(10):
    print("app running normally...")
    time.sleep(1)

# Strips the newline character
for line in Lines:
    count += 1
    print(line.rstrip())
print(count)
print("app terminated.")
```

   c. Simpan file `main.py`.
3. Buat file log sampel.
   a. Di dalam `multiline-app` folder, buat file dan beri nama`test.log`.
   b. Tempel konten berikut dalam file:

```
single line...
Dec 14 06:41:08 Exception in thread "main" java.lang.RuntimeException:
 Something has gone wrong, aborting!
    at com.myproject.module.MyProject.badMethod(MyProject.java:22)
    at com.myproject.module.MyProject.oneMoreMethod(MyProject.java:18)
    at com.myproject.module.MyProject.anotherMethod(MyProject.java:14)
    at com.myproject.module.MyProject.someMethod(MyProject.java:10)
    at com.myproject.module.MyProject.main(MyProject.java:6)
another line...
```

c. Simpan file `test.log`.

4. Di dalam `multiline-app` folder, buat Dockerfile.

   a. Tempel konten berikut dalam file:

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest
ADD test.log /test.log

RUN yum upgrade -y && yum install -y python3

WORKDIR /usr/local/bin

COPY main.py .

CMD ["python3", "main.py"]
```

   b. Simpan file `Dockerfile`.

5. Menggunakan Dockerfile, buat gambar.

   a. Bangun gambar: `docker build -t multiline-app-image .`

      Dimana: `multiline-app-image` adalah nama untuk gambar dalam contoh ini.

   b. Verifikasi bahwa gambar telah dibuat dengan benar: `docker images –filter reference=multiline-app-image`

      Jika berhasil, output menunjukkan gambar dan `latest` tag.

6. Unggah gambar ke Amazon Elastic Container Registry.

   a. Buat repositori Amazon ECR untuk menyimpan gambar: `aws ecr create-repository --repository-name multiline-app-repo --region us-east-1`

      Dimana: `multiline-app-repo` adalah nama untuk repositori dan `us-east-1` merupakan wilayah dalam contoh ini.

      Outputnya memberi Anda detail repositori baru. Perhatikan `repositoryUri` nilainya karena Anda akan membutuhkannya di langkah selanjutnya.

   b. Tandai gambar Anda dengan `repositoryUri` nilai dari output sebelumnya: `docker tag multiline-app-image repositoryUri`

Contoh: `docker tag multiline-app-image xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/multiline-app-repo`

c. Jalankan docker image untuk memverifikasi itu berjalan dengan benar: `docker images –filter reference=`*`repositoryUri`*

Dalam output, nama repositori berubah dari `multiline-app-repo` ke nilai. `repositoryUri`

d. Dorong gambar ke Amazon ECR: `docker push` *`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`repository name`*

Contoh: `docker push` *`xxxxxxxxxxxx`*`.dkr.ecr.`*`us-east-1`*`.amazonaws.com/multiline-app-repo`

Buat definisi tugas dan jalankan tugas

1. Buat file definisi tugas dengan nama file`multiline-task-definition.json`.
2. Tempel konten berikut dalam `multiline-task-definition.json` file:

```
{
    "family": "firelens-example-multiline",
    "taskRoleArn": "task role ARN",
    "executionRoleArn": "execution role ARN",
    "containerDefinitions": [
        {
            "essential": true,
            "image": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-
multiline-image:latest",
            "name": "log_router",
            "firelensConfiguration": {
                "type": "fluentbit",
                "options": {
                    "config-file-type": "file",
                    "config-file-value": "/extra.conf"
                }
            },
            "memoryReservation": 50
        },
        {
            "essential": true,
```

```
            "image": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/multiline-app-
image:latest",
            "name": "app",
            "logConfiguration": {
                "logDriver": "awsfirelens",
                "options": {
                    "Name": "cloudwatch_logs",
                    "region": "us-east-1",
                    "log_group_name": "multiline-test/application",
                    "auto_create_group": "true",
                    "log_stream_prefix": "multiline-"
                }
            },
            "memoryReservation": 100
        }
    ],
    "requiresCompatibilities": ["FARGATE"],
    "networkMode": "awsvpc",
    "cpu": "256",
    "memory": "512"
}
```

Ganti yang berikut ini dalam definisi `multiline-task-definition.json` tugas:

a. *task role ARN*

   Untuk menemukan peran tugas ARN, buka konsol IAM. Pilih Peran dan temukan peran `ecs-task-role-for-firelens` tugas yang Anda buat. Pilih peran dan salin ARN yang muncul di bagian Ringkasan.

b. *execution role ARN*

   Untuk menemukan peran eksekusi ARN, buka konsol IAM. Pilih Peran dan temukan `ecsTaskExecutionRole` perannya. Pilih peran dan salin ARN yang muncul di bagian Ringkasan.

c. *aws_account_id*

   Untuk menemukan Anda`aws_account_id`, masuk ke AWS Management Console. Pilih nama pengguna Anda di kanan atas dan salin ID Akun Anda.

d. *us-east-1*

   Ganti wilayah jika perlu.

3. Daftarkan file definisi tugas: `aws ecs register-task-definition --cli-input-json file://multiline-task-definition.json --region region`
4. Buka konsol di https://console.aws.amazon.com/ecs/v2.
5. Di panel navigasi, pilih Definisi Tugas dan kemudian pilih `firelens-example-multiline` keluarga karena kami mendaftarkan definisi tugas ke keluarga ini di baris pertama definisi tugas di atas.
6. Pilih versi terbaru.
7. Pilih tugas Deploy, Run.
8. Pada halaman Jalankan Tugas, Untuk Cluster, pilih cluster, dan kemudian di bawah Jaringan, untuk Subnet, pilih subnet yang tersedia untuk tugas Anda.
9. Pilih Buat.

Verifikasi bahwa pesan log multiline di Amazon CloudWatch tampak digabungkan

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Dari panel navigasi, perluas Log dan pilih Grup log.
3. Pilih grup `multiline-test/applicatio` log.
4. Pilih log. Lihat pesan. Baris yang cocok dengan aturan dalam file parser digabungkan dan muncul sebagai pesan tunggal.

   Cuplikan log berikut menunjukkan baris yang digabungkan dalam satu peristiwa jejak tumpukan Java:

```
{
    "container_id": "xxxxxx",
    "container_name": "app",
    "source": "stdout",
    "log": "Dec 14 06:41:08 Exception in thread \"main\"
 java.lang.RuntimeException: Something has gone wrong, aborting!\n
 at com.myproject.module.MyProject.badMethod(MyProject.java:22)\n    at
 com.myproject.module.MyProject.oneMoreMethod(MyProject.java:18)\n
 at com.myproject.module.MyProject.anotherMethod(MyProject.java:14)\n
 at com.myproject.module.MyProject.someMethod(MyProject.java:10)\n    at
 com.myproject.module.MyProject.main(MyProject.java:6)",
    "ecs_cluster": "default",
    "ecs_task_arn": "arn:aws:ecs:us-east-1:xxxxxxxxxxxx:task/default/xxxxxx",
    "ecs_task_definition": "firelens-example-multiline:2"
```

```
}
```

Cuplikan log berikut menunjukkan bagaimana pesan yang sama muncul hanya dengan satu baris jika Anda menjalankan wadah Amazon ECS yang tidak dikonfigurasi untuk menggabungkan pesan log multiline.

```
{
    "log": "Dec 14 06:41:08 Exception in thread \"main\"
 java.lang.RuntimeException: Something has gone wrong, aborting!",
    "container_id": "xxxxxx-xxxxxx",
    "container_name": "app",
    "source": "stdout",
    "ecs_cluster": "default",
    "ecs_task_arn": "arn:aws:ecs:us-east-1:xxxxxxxxxxxx:task/default/xxxxxx",
    "ecs_task_definition": "firelens-example-multiline:3"
}
```

Contoh: Gunakan parser bawaan Fluent Bit

Dalam contoh ini, Anda akan menyelesaikan langkah-langkah berikut:

1. Bangun dan unggah gambar untuk wadah Fluent Bit.
2. Buat dan unggah gambar untuk aplikasi multiline demo yang berjalan, gagal, dan menghasilkan jejak tumpukan multiline.
3. Buat definisi tugas dan jalankan tugas.
4. Lihat log untuk memverifikasi bahwa pesan yang menjangkau beberapa baris tampak digabungkan.

Buat dan unggah gambar untuk wadah Fluent Bit

Gambar ini akan menyertakan file konfigurasi yang mereferensikan parser Fluent Bit.

1. Buat folder dengan nama`FluentBitDockerImage`.
2. Di dalam `FluentBitDockerImage` folder, buat file konfigurasi khusus yang mereferensikan file parser bawaan Fluent Bit.

   Untuk informasi selengkapnya tentang file konfigurasi kustom, lihat Menentukan file konfigurasi kustom di Panduan Pengembang Layanan Amazon Elastic Container

a. Tempel konten berikut dalam file:

```
[FILTER]
    name                  multiline
    match                 *
    multiline.key_content log
    multiline.parser      go
```

b. Simpan file sebagai `extra.conf`.

3. Di dalam `FluentBitDockerImage` folder, buat Dockerfile dengan gambar Fluent Bit dan parser dan file konfigurasi yang Anda buat.

a. Tempel konten berikut dalam file:

```
FROM public.ecr.aws/aws-observability/aws-for-fluent-bit:latest
ADD extra.conf /extra.conf
```

b. Simpan file sebagai `Dockerfile`.

4. Menggunakan Dockerfile, buat gambar Fluent Bit kustom dengan file konfigurasi khusus yang disertakan.

> **Note**
>
> Anda dapat menempatkan file konfigurasi di mana saja di gambar Docker kecuali `/fluent-bit/etc/fluent-bit.conf` karena jalur file ini digunakan oleh FireLens.

a. Bangun gambar: `docker build -t fluent-bit-multiline-image .`

Dimana: `fluent-bit-multiline-image` adalah nama untuk gambar dalam contoh ini.

b. Verifikasi bahwa gambar telah dibuat dengan benar: `docker images –filter reference=fluent-bit-multiline-image`

Jika berhasil, output menunjukkan gambar dan `latest` tag.

5. Unggah gambar Fluent Bit kustom ke Amazon Elastic Container Registry.

a. Buat repositori Amazon ECR untuk menyimpan gambar: `aws ecr create-repository --repository-name fluent-bit-multiline-repo --region us-east-1`

Dimana: `fluent-bit-multiline-repo` adalah nama untuk repositori dan `us-east-1` merupakan wilayah dalam contoh ini.

Outputnya memberi Anda detail repositori baru.

b. Tandai gambar Anda dengan `repositoryUri` nilai dari output sebelumnya: `docker tag` *`fluent-bit-multiline-image repositoryUri`*

Contoh: `docker tag fluent-bit-multiline-image xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-repo`

c. Jalankan docker image untuk memverifikasi itu berjalan dengan benar: `docker images –filter reference=`*`repositoryUri`*

Dalam output, nama repositori berubah dari fluent-bit-multiline-repo ke. `repositoryUri`

d. Otentikasi ke Amazon ECR dengan menjalankan `aws ecr get-login-password` perintah dan menentukan ID registri yang ingin Anda autentikasi: `aws ecr get-login-password | docker login --username AWS --password-stdin` *`registry ID`*`.dkr.ecr.`*`region`*`.amazonaws.com`

Contoh: `ecr get-login-password | docker login --username AWS --password-stdin xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com`

Pesan login yang berhasil muncul.

e. Dorong gambar ke Amazon ECR: `docker push` *`registry ID`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`repository name`*

Contoh: `docker push xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-repo`

Membangun dan meng-upload gambar untuk aplikasi demo multiline

Gambar ini akan mencakup file skrip Python yang menjalankan aplikasi dan file log sampel.

1. Buat folder bernama`multiline-app`: `mkdir multiline-app`
2. Buat file skrip Python.

   a. Di dalam `multiline-app` folder, buat file dan beri nama`main.py`.

   b. Tempel konten berikut dalam file:

```
import os
import time
file1 = open('/test.log', 'r')
Lines = file1.readlines()

count = 0

for i in range(10):
    print("app running normally...")
    time.sleep(1)

# Strips the newline character
for line in Lines:
    count += 1
    print(line.rstrip())
print(count)
print("app terminated.")
```

c. Simpan file `main.py`.

3. Buat file log sampel.

a. Di dalam `multiline-app` folder, buat file dan beri nama`test.log`.

b. Tempel konten berikut dalam file:

```
panic: my panic

goroutine 4 [running]:
panic(0x45cb40, 0x47ad70)
  /usr/local/go/src/runtime/panic.go:542 +0x46c fp=0xc42003f7b8 sp=0xc42003f710
 pc=0x422f7c
main.main.func1(0xc420024120)
  foo.go:6 +0x39 fp=0xc42003f7d8 sp=0xc42003f7b8 pc=0x451339
runtime.goexit()
  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003f7e0
 sp=0xc42003f7d8 pc=0x44b4d1
created by main.main
  foo.go:5 +0x58

goroutine 1 [chan receive]:
runtime.gopark(0x4739b8, 0xc420024178, 0x46fcd7, 0xc, 0xc420028e17, 0x3)
```

```
  /usr/local/go/src/runtime/proc.go:280 +0x12c fp=0xc420053e30 sp=0xc420053e00
 pc=0x42503c
runtime.goparkunlock(0xc420024178, 0x46fcd7, 0xc, 0x1000f010040c217, 0x3)
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc420053e70 sp=0xc420053e30
 pc=0x42512e
runtime.chanrecv(0xc420024120, 0x0, 0xc420053f01, 0x4512d8)
  /usr/local/go/src/runtime/chan.go:506 +0x304 fp=0xc420053f20 sp=0xc420053e70
 pc=0x4046b4
runtime.chanrecv1(0xc420024120, 0x0)
  /usr/local/go/src/runtime/chan.go:388 +0x2b fp=0xc420053f50 sp=0xc420053f20
 pc=0x40439b
main.main()
  foo.go:9 +0x6f fp=0xc420053f80 sp=0xc420053f50 pc=0x4512ef
runtime.main()
  /usr/local/go/src/runtime/proc.go:185 +0x20d fp=0xc420053fe0 sp=0xc420053f80
 pc=0x424bad
runtime.goexit()
  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc420053fe8
 sp=0xc420053fe0 pc=0x44b4d1

goroutine 2 [force gc (idle)]:
runtime.gopark(0x4739b8, 0x4ad720, 0x47001e, 0xf, 0x14, 0x1)
  /usr/local/go/src/runtime/proc.go:280 +0x12c fp=0xc42003e768 sp=0xc42003e738
 pc=0x42503c
runtime.goparkunlock(0x4ad720, 0x47001e, 0xf, 0xc420000114, 0x1)
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc42003e7a8 sp=0xc42003e768
 pc=0x42512e
runtime.forcegchelper()
  /usr/local/go/src/runtime/proc.go:238 +0xcc fp=0xc42003e7e0 sp=0xc42003e7a8
 pc=0x424e5c
runtime.goexit()
  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003e7e8
 sp=0xc42003e7e0 pc=0x44b4d1
created by runtime.init.4
  /usr/local/go/src/runtime/proc.go:227 +0x35

goroutine 3 [GC sweep wait]:
runtime.gopark(0x4739b8, 0x4ad7e0, 0x46fdd2, 0xd, 0x419914, 0x1)
  /usr/local/go/src/runtime/proc.go:280 +0x12c fp=0xc42003ef60 sp=0xc42003ef30
 pc=0x42503c
runtime.goparkunlock(0x4ad7e0, 0x46fdd2, 0xd, 0x14, 0x1)
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc42003efa0 sp=0xc42003ef60
 pc=0x42512e
runtime.bgsweep(0xc42001e150)
```

```
  /usr/local/go/src/runtime/mgcsweep.go:52 +0xa3 fp=0xc42003efd8
 sp=0xc42003efa0 pc=0x419973
runtime.goexit()
  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003efe0
 sp=0xc42003efd8 pc=0x44b4d1
created by runtime.gcenable
  /usr/local/go/src/runtime/mgc.go:216 +0x58
one more line, no multiline
```

c. Simpan file `test.log`.

4. Di dalam `multiline-app` folder, buat Dockerfile.

   a. Tempel konten berikut dalam file:

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest
ADD test.log /test.log

RUN yum upgrade -y && yum install -y python3

WORKDIR /usr/local/bin

COPY main.py .

CMD ["python3", "main.py"]
```

   b. Simpan file `Dockerfile`.

5. Menggunakan Dockerfile, buat gambar.

   a. Bangun gambar: `docker build -t multiline-app-image .`

      Dimana: `multiline-app-image` adalah nama untuk gambar dalam contoh ini.

   b. Verifikasi bahwa gambar telah dibuat dengan benar: `docker images –filter reference=multiline-app-image`

      Jika berhasil, output menunjukkan gambar dan `latest` tag.

6. Unggah gambar ke Amazon Elastic Container Registry.

   a. Buat repositori Amazon ECR untuk menyimpan gambar: `aws ecr create-repository --repository-name multiline-app-repo --region us-east-1`

Dimana: `multiline-app-repo` adalah nama untuk repositori dan `us-east-1` merupakan wilayah dalam contoh ini.

Outputnya memberi Anda detail repositori baru. Perhatikan `repositoryUri` nilainya karena Anda akan membutuhkannya di langkah selanjutnya.

b. Tandai gambar Anda dengan `repositoryUri` nilai dari output sebelumnya: `docker tag` *`multiline-app-image repositoryUri`*

Contoh: `docker tag multiline-app-image xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/multiline-app-repo`

c. Jalankan docker image untuk memverifikasi itu berjalan dengan benar: `docker images –filter reference=`*`repositoryUri`*

Dalam output, nama repositori berubah dari `multiline-app-repo` ke nilai. `repositoryUri`

d. Dorong gambar ke Amazon ECR: `docker push` *`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`repository name`*

Contoh: `docker push` *`xxxxxxxxxxxx`*`.dkr.ecr.`*`us-east-1`*`.amazonaws.com/multiline-app-repo`

Buat definisi tugas dan jalankan tugas

1. Buat file definisi tugas dengan nama file`multiline-task-definition.json`.
2. Tempel konten berikut dalam `multiline-task-definition.json` file:

```
{
    "family": "firelens-example-multiline",
    "taskRoleArn": "task role ARN,
    "executionRoleArn": "execution role ARN",
    "containerDefinitions": [
        {
            "essential": true,
            "image": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-
multiline-image:latest",
            "name": "log_router",
            "firelensConfiguration": {
                "type": "fluentbit",
```

```
                "options": {
                    "config-file-type": "file",
                    "config-file-value": "/extra.conf"
                }
            },
            "memoryReservation": 50
        },
        {
            "essential": true,
            "image": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/multiline-app-
image:latest",
            "name": "app",
            "logConfiguration": {
                "logDriver": "awsfirelens",
                "options": {
                    "Name": "cloudwatch_logs",
                    "region": "us-east-1",
                    "log_group_name": "multiline-test/application",
                    "auto_create_group": "true",
                    "log_stream_prefix": "multiline-"
                }
            },
            "memoryReservation": 100
        }
    ],
    "requiresCompatibilities": ["FARGATE"],
    "networkMode": "awsvpc",
    "cpu": "256",
    "memory": "512"
}
```

Ganti yang berikut ini dalam definisi `multiline-task-definition.json` tugas:

a. *task role ARN*

   Untuk menemukan peran tugas ARN, buka konsol IAM. Pilih Peran dan temukan peran `ecs-task-role-for-firelens` tugas yang Anda buat. Pilih peran dan salin ARN yang muncul di bagian Ringkasan.

b. *execution role ARN*

Untuk menemukan peran eksekusi ARN, buka konsol IAM. Pilih Peran dan temukan `ecsTaskExecutionRole` perannya. Pilih peran dan salin ARN yang muncul di bagian Ringkasan.

c. *`aws_account_id`*

Untuk menemukan Anda`aws_account_id`, masuk ke AWS Management Console. Pilih nama pengguna Anda di kanan atas dan salin ID Akun Anda.

d. *`us-east-1`*

Ganti wilayah jika perlu.

3. Daftarkan file definisi tugas: `aws ecs register-task-definition --cli-input-json file://multiline-task-definition.json --region us-east-1`
4. Buka konsol di https://console.aws.amazon.com/ecs/v2.
5. Di panel navigasi, pilih Definisi Tugas dan kemudian pilih `firelens-example-multiline` keluarga karena kami mendaftarkan definisi tugas ke keluarga ini di baris pertama definisi tugas di atas.
6. Pilih versi terbaru.
7. Pilih tugas Deploy, Run.
8. Pada halaman Jalankan Tugas, Untuk Cluster, pilih cluster, dan kemudian di bawah Jaringan, untuk Subnet, pilih subnet yang tersedia untuk tugas Anda.
9. Pilih Buat.

Verifikasi bahwa pesan log multiline di Amazon CloudWatch tampak digabungkan

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Dari panel navigasi, perluas Log dan pilih Grup log.
3. Pilih grup `multiline-test/applicatio` log.
4. Pilih log dan lihat pesan. Baris yang cocok dengan aturan dalam file parser digabungkan dan muncul sebagai pesan tunggal.

   Cuplikan log berikut menunjukkan jejak tumpukan Go yang digabungkan menjadi satu peristiwa:

```
{
    "log": "panic: my panic\n\ngoroutine 4 [running]:\npanic(0x45cb40,
 0x47ad70)\n  /usr/local/go/src/runtime/panic.go:542 +0x46c fp=0xc42003f7b8
```

```
 sp=0xc42003f710 pc=0x422f7c\nmain.main.func1(0xc420024120)\n  foo.go:6
 +0x39 fp=0xc42003f7d8 sp=0xc42003f7b8 pc=0x451339\nruntime.goexit()\n  /usr/
local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003f7e0 sp=0xc42003f7d8
 pc=0x44b4d1\ncreated by main.main\n  foo.go:5 +0x58\n\ngoroutine 1 [chan receive]:
\nruntime.gopark(0x4739b8, 0xc420024178, 0x46fcd7, 0xc, 0xc420028e17, 0x3)\n  /usr/
local/go/src/runtime/proc.go:280 +0x12c fp=0xc420053e30 sp=0xc420053e00 pc=0x42503c
\nruntime.goparkunlock(0xc420024178, 0x46fcd7, 0xc, 0x1000f010040c217, 0x3)\n
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc420053e70 sp=0xc420053e30
 pc=0x42512e\nruntime.chanrecv(0xc420024120, 0x0, 0xc420053f01, 0x4512d8)\n
  /usr/local/go/src/runtime/chan.go:506 +0x304 fp=0xc420053f20 sp=0xc420053e70
 pc=0x4046b4\nruntime.chanrecv1(0xc420024120, 0x0)\n  /usr/local/go/src/runtime/
chan.go:388 +0x2b fp=0xc420053f50 sp=0xc420053f20 pc=0x40439b\nmain.main()\n
 foo.go:9 +0x6f fp=0xc420053f80 sp=0xc420053f50 pc=0x4512ef\nruntime.main()\n
  /usr/local/go/src/runtime/proc.go:185 +0x20d fp=0xc420053fe0 sp=0xc420053f80
 pc=0x424bad\nruntime.goexit()\n  /usr/local/go/src/runtime/asm_amd64.s:2337
 +0x1 fp=0xc420053fe8 sp=0xc420053fe0 pc=0x44b4d1\n\ngoroutine 2 [force gc
 (idle)]:\nruntime.gopark(0x4739b8, 0x4ad720, 0x47001e, 0xf, 0x14, 0x1)\n  /
usr/local/go/src/runtime/proc.go:280 +0x12c fp=0xc42003e768 sp=0xc42003e738
 pc=0x42503c\nruntime.goparkunlock(0x4ad720, 0x47001e, 0xf, 0xc420000114, 0x1)\n
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc42003e7a8 sp=0xc42003e768
 pc=0x42512e\nruntime.forcegchelper()\n  /usr/local/go/src/runtime/proc.go:238
 +0xcc fp=0xc42003e7e0 sp=0xc42003e7a8 pc=0x424e5c\nruntime.goexit()\n  /usr/
local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003e7e8 sp=0xc42003e7e0
 pc=0x44b4d1\ncreated by runtime.init.4\n  /usr/local/go/src/runtime/proc.go:227
 +0x35\n\ngoroutine 3 [GC sweep wait]:\nruntime.gopark(0x4739b8, 0x4ad7e0,
 0x46fdd2, 0xd, 0x419914, 0x1)\n  /usr/local/go/src/runtime/proc.go:280 +0x12c
 fp=0xc42003ef60 sp=0xc42003ef30 pc=0x42503c\nruntime.goparkunlock(0x4ad7e0,
 0x46fdd2, 0xd, 0x14, 0x1)\n  /usr/local/go/src/runtime/proc.go:286 +0x5e
 fp=0xc42003efa0 sp=0xc42003ef60 pc=0x42512e\nruntime.bgsweep(0xc42001e150)\n
  /usr/local/go/src/runtime/mgcsweep.go:52 +0xa3 fp=0xc42003efd8 sp=0xc42003efa0
 pc=0x419973\nruntime.goexit()\n  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1
 fp=0xc42003efe0 sp=0xc42003efd8 pc=0x44b4d1\ncreated by runtime.gcenable\n  /usr/
local/go/src/runtime/mgc.go:216 +0x58",
    "container_id": "xxxxxx-xxxxxx",
    "container_name": "app",
    "source": "stdout",
    "ecs_cluster": "default",
    "ecs_task_arn": "arn:aws:ecs:us-east-1:xxxxxxxxxxxx:task/default/xxxxxx",
    "ecs_task_definition": "firelens-example-multiline:2"
}
```

Cuplikan log berikut menunjukkan bagaimana peristiwa yang sama muncul jika Anda menjalankan wadah ECS yang tidak dikonfigurasi untuk menggabungkan pesan log multiline. Bidang log berisi satu baris.

```
{
    "log": "panic: my panic",
    "container_id": "xxxxxx-xxxxxx",
    "container_name": "app",
    "source": "stdout",
    "ecs_cluster": "default",
    "ecs_task_arn": "arn:aws:ecs:us-east-1:xxxxxxxxxxxx:task/default/xxxxxx",
    "ecs_task_definition": "firelens-example-multiline:3"
```

> **Note**
>
> Jika log Anda masuk ke file log alih-alih output standar, kami sarankan untuk menentukan parameter `multiline.parser` dan `multiline.key_content` konfigurasi di plugin input Tail alih-alih Filter.

Menggunakan sistem file Amazon EFS dengan Amazon ECS menggunakan konsol

Amazon Elastic File System (Amazon EFS) menyediakan penyimpanan file yang sederhana dan dapat diskalakan untuk digunakan dengan tugas Amazon ECS Anda. Dengan Amazon EFS, kapasitas penyimpanan menjadi elastis, tumbuh dan menyusut secara otomatis saat Anda menambahkan dan menghapus file. Aplikasi Anda dapat memiliki penyimpanan yang dibutuhkan, kapan pun dibutuhkan.

Anda dapat menggunakan sistem file Amazon EFS dengan Amazon ECS untuk mengakses data sistem file di seluruh armada tugas Amazon ECS Anda. Dengan begitu, tugas Anda memiliki akses ke penyimpanan tetap yang sama, tidak peduli infrastruktur atau instans kontainer di tempat mereka mendarat. Saat Anda mereferensikan sistem file Amazon EFS dan titik pemasangan kontainer dalam definisi tugas Amazon ECS Anda, Amazon ECS menangani pemasangan sistem file di wadah Anda. Bagian berikut membantu Anda mulai menggunakan Amazon EFS dengan Amazon ECS.

Fitur ini didukung oleh tugas yang menggunakan jenis peluncuran EC2 dan Fargate, namun tutorial ini akan menggunakan tugas Amazon ECS yang menggunakan tipe peluncuran EC2. Tutorial ini juga bertujuan agar langkah demi langkah diikuti, namun jika Anda memiliki beberapa sumber daya ini yang telah dibuat pada akun Anda maka Anda mungkin dapat melewatkan beberapa langkah-langkah.

> **Note**
>
> Amazon EFS mungkin tidak tersedia di semua Wilayah. Untuk informasi selengkapnya tentang Wilayah mana yang mendukung Amazon EFS, lihat Titik Akhir dan Kuota Amazon Elastic File System di bagian. Referensi Umum AWS

Langkah 1: Buat cluster Amazon ECS

Gunakan langkah-langkah berikut untuk membuat cluster Amazon ECS.

Untuk membuat cluster baru (konsol Amazon ECS)

Sebelum Anda mulai, tetapkan izin IAM yang sesuai. Untuk informasi selengkapnya, lihat the section called “Contoh klaster”.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.
3. Di panel navigasi, pilih Klaster.
4. Pada halaman Klaster, pilih Buat klaster.
5. Di bawah konfigurasi Cluster, untuk nama Cluster, masukkan `EFS-tutorial` untuk nama cluster.
6. (Opsional) Untuk mengubah VPC dan subnet tempat tugas dan layanan Anda diluncurkan, di bawah Jaringan, lakukan salah satu operasi berikut:
   - Untuk menghapus subnet, di bawah Subnet, pilih X untuk setiap subnet yang ingin Anda hapus.
   - Untuk mengubah ke VPC selain VPC default, di bawah VPC, pilih VPC yang ada, lalu di bawah Subnet, pilih setiap subnet.

7. Untuk menambahkan instans Amazon EC2 ke klaster Anda, perluas Infrastruktur, lalu pilih instans Amazon EC2. Selanjutnya, konfigurasikan grup Auto Scaling yang bertindak sebagai penyedia kapasitas:

    - Untuk membuat grup Auto Scaling, dari grup Auto Scaling (ASG), pilih Buat grup baru, lalu berikan detail berikut tentang grup:

        - Untuk Sistem Operasi/Arsitektur, pilih Amazon Linux 2.
        - Untuk Tipe instans EC2, pilih `t2.micro`.

          Untuk key pair SSH, pilih pair yang membuktikan identitas Anda saat Anda terhubung ke instance.
        - Untuk Kapasitas, masukkan1.
8. Pilih Buat.

Langkah 2: Buat grup keamanan untuk instans Amazon EC2 dan sistem file Amazon EFS

Pada langkah ini, Anda membuat grup keamanan untuk instans Amazon EC2 yang memungkinkan lalu lintas jaringan masuk pada port 80 dan sistem file Amazon EFS Anda yang memungkinkan akses masuk dari instans kontainer Anda.

Buat grup keamanan untuk instans Amazon EC2 Anda dengan opsi berikut:

- Nama grup keamanan - nama unik untuk grup keamanan Anda.
- VPC - VPC yang Anda identifikasi sebelumnya untuk cluster Anda.
- Aturan ke dalam
    - Jenis - HTTP
    - Sumber - 0.0.0.0/0.

Buat grup keamanan untuk sistem file Amazon EFS Anda dengan opsi berikut:

- Nama grup keamanan - nama unik untuk grup keamanan Anda. Misalnya, `EFS-access-for-sg-`*`dc025fa2`*.
- VPC - VPC yang Anda identifikasi sebelumnya untuk cluster Anda.
- Aturan ke dalam

- Jenis - NFS
- Sumber - Kustom dengan ID grup keamanan yang Anda buat untuk instance Anda.

Untuk informasi tentang cara membuat grup keamanan, lihat Membuat grup keamanan di Panduan Pengguna Amazon EC2 untuk Instans Linux.

Langkah 3: Buat sistem file Amazon EFS

Pada langkah ini, Anda membuat sistem file Amazon EFS.

Untuk membuat sistem file Amazon EFS untuk tugas Amazon ECS.

1. Buka konsol Amazon Elastic File System di https://console.aws.amazon.com/efs/.
2. Pilih Buat sistem file.
3. Masukkan nama untuk sistem file Anda dan kemudian pilih VPC tempat instance container Anda di-host. Secara default, setiap subnet di VPC tertentu menerima target pemasangan yang menggunakan grup keamanan default untuk VPC tersebut. Kemudian, pilih Sesuaikan.

   > **Note**
   >
   > Tutorial ini mengasumsikan bahwa sistem file Amazon EFS, cluster Amazon ECS, instance container, dan tugas Anda berada dalam VPC yang sama. Untuk informasi selengkapnya tentang memasang sistem file dari VPC yang berbeda, lihat Panduan: Memasang sistem file dari VPC yang berbeda di Panduan Pengguna Amazon EFS.

4. Pada halaman Pengaturan sistem file, konfigurasikan pengaturan opsional dan kemudian di bawah Pengaturan kinerja, pilih mode throughput meledak untuk sistem file Anda. Setelah Anda mengonfigurasi pengaturan, pilih Berikutnya.

   a. (Opsional) Tambahkan untuk menandai untuk sistem file Anda. Sebagai contoh, Anda dapat menentukan nama unik untuk sistem file dengan memasukkan nama tersebut di kolom Nilai di samping kunci Nama.

   b. (Opsional) Aktifkan manajemen siklus hidup untuk menghemat uang pada penyimpanan yang jarang diakses. Untuk informasi selengkapnya, lihat EFS Lifecycle Management di Panduan Pengguna Amazon Elastic File System.

   c. (Opsional) Aktifkan enkripsi. Pilih kotak centang untuk mengaktifkan enkripsi sistem file Amazon EFS Anda saat istirahat.

5. Pada halaman Akses jaringan, di bawah Target Mount, ganti konfigurasi grup keamanan yang ada untuk setiap zona ketersediaan dengan grup keamanan yang Anda buat untuk sistem file, Langkah 2: Buat grup keamanan untuk instans Amazon EC2 dan sistem file Amazon EFS lalu pilih Berikutnya.
6. Anda tidak perlu mengkonfigurasi kebijakan sistem File untuk tutorial ini, sehingga Anda dapat melewati bagian dengan memilih Berikutnya.
7. Tinjau opsi sistem file Anda dan pilih Buat untuk menyelesaikan proses.
8. Dari layar Sistem file, rekam ID sistem File. Pada langkah berikutnya, Anda akan mereferensikan nilai ini dalam definisi tugas Amazon ECS Anda.

Langkah 4: Tambahkan konten ke sistem file Amazon EFS

Pada langkah ini, Anda memasang sistem file Amazon EFS ke instans Amazon EC2 dan menambahkan konten ke dalamnya. Proses ini bertujuan untuk menguji tutorial ini, untuk menggambarkan sifat persisten data. Saat menggunakan fitur ini, Anda biasanya memiliki aplikasi atau metode lain untuk menulis data ke sistem file Amazon EFS Anda.

Untuk membuat instans Amazon EC2 dan memasang sistem file Amazon EFS

1. Buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.
2. Pilih Luncurkan Instans.
3. Di bawah Gambar Aplikasi dan OS (Gambar Mesin Amazon), pilih Amazon Linux 2 AMI (HVM).
4. Di bawah tipe Instance, pertahankan tipe instance default,`t2.micro`.
5. Di bawah Key pair (login), pilih key pair untuk akses SSH ke instance.
6. Di bawah Pengaturan jaringan, pilih VPC yang Anda tentukan untuk sistem file Amazon EFS dan kluster Amazon ECS. Pilih subnet dan grup keamanan instance yang dibuat diLangkah 2: Buat grup keamanan untuk instans Amazon EC2 dan sistem file Amazon EFS. Konfigurasikan grup keamanan instans. Pastikan bahwa Auto-assign IP publik diaktifkan.
7. Di bawah Konfigurasi penyimpanan, pilih tombol Edit untuk sistem file dan kemudian pilih EFS. Pilih sistem file yang Anda buatLangkah 3: Buat sistem file Amazon EFS. Anda dapat secara opsional mengubah titik pemasangan atau meninggalkan nilai default.

> **Important**
>
> Anda harus memilih subnet sebelum Anda dapat menambahkan sistem file ke instance.

8. Hapus secara otomatis membuat dan melampirkan grup keamanan. Biarkan kotak centang lainnya dipilih. Pilih Tambahkan sistem file bersama.
9. Di bawah Detail Lanjutan, pastikan bahwa skrip data pengguna diisi secara otomatis dengan langkah-langkah pemasangan sistem file Amazon EFS.
10. Di bawah Ringkasan, pastikan Jumlah instans adalah 1. Pilih Luncurkan instans.
11. Pada halaman Luncurkan instans, pilih Lihat semua instance untuk melihat status instans Anda. Awalnya, status status Instance adalah`PENDING`. Setelah status berubah `RUNNING` dan instance melewati semua pemeriksaan status, instance siap digunakan.

Sekarang, Anda terhubung ke instans Amazon EC2 dan menambahkan konten ke sistem file Amazon EFS.

Untuk terhubung ke instans Amazon EC2 dan menambahkan konten ke sistem file Amazon EFS

1. SSH ke instans Amazon EC2 yang Anda buat. Untuk informasi lebih lanjut, lihat Connect ke Instans Linux Anda dalam Panduan Pengguna Amazon EC2 untuk Instans Linux.
2. Dari jendela terminal, jalankan df -T perintah untuk memverifikasi bahwa sistem file Amazon EFS sudah terpasang. Pada output berikut, kami telah menyoroti pemasangan sistem file Amazon EFS.

```
$ df -T
Filesystem     Type            1K-blocks    Used        Available Use% Mounted on
devtmpfs       devtmpfs           485468       0           485468   0% /dev
tmpfs          tmpfs              503480       0           503480   0% /dev/shm
tmpfs          tmpfs              503480     424           503056   1% /run
tmpfs          tmpfs              503480       0           503480   0% /sys/fs/
cgroup
/dev/xvda1     xfs               8376300 1310952          7065348  16% /
127.0.0.1:/    nfs4     9007199254739968       0 9007199254739968   0% /mnt/efs/fs1
tmpfs          tmpfs              100700       0           100700   0% /run/
user/1000
```

3. Arahkan ke direktori tempat sistem file Amazon EFS dipasang. Dalam contoh di atas, yaitu `/mnt/efs/fs1`.
4. Buat file bernama `index.html` dengan konten berikut:

```
<html>
    <body>
```

```
        <h1>It Works!</h1>
        <p>You are using an Amazon EFS file system for persistent container
 storage.</p>
    </body>
</html>
```

Langkah 5: Buat ketentuan tugas

Ketentuan tugas berikut membuat volume data dengan nama `efs-html`. Pemasangan kontainer `nginx` volume data host pada root NGINX, `/usr/share/nginx/html`.

Untuk membuat definisi tugas baru menggunakan konsol Amazon ECS

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Definisi tugas.
3. Pilih Buat definisi tugas baru, Buat definisi tugas baru dengan JSON.
4. Di kotak editor JSON, salin dan tempel teks JSON berikut, ganti `fileSystemId` dengan ID sistem file Amazon EFS Anda.

```
{
    "containerDefinitions": [
        {
            "memory": 128,
            "portMappings": [
                {
                    "hostPort": 80,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "mountPoints": [
                {
                    "containerPath": "/usr/share/nginx/html",
                    "sourceVolume": "efs-html"
                }
            ],
            "name": "nginx",
            "image": "nginx"
        }
```

```
    ],
    "volumes": [
        {
            "name": "efs-html",
            "efsVolumeConfiguration": {
                "fileSystemId": "fs-1324abcd",
                "transitEncryption": "ENABLED"
            }
        }
    ],
    "family": "efs-tutorial",
    "executionRoleArn":"arn:aws::iam::111122223333:role/ecsTaskExecutionRole"
}
```

> **Note**
>
> Anda dapat menambahkan izin berikut ke peran IAM eksekusi tugas Amazon ECS Anda untuk memungkinkan agen Amazon ECS menemukan dan memasang sistem file Amazon EFS ke tugas saat startup.
>
> - `elasticfilesystem:ClientMount`
> - `elasticfilesystem:ClientWrite`
> - `elasticfilesystem:DescribeMountTargets`
> - `elasticfilesystem:DescribeFileSystems`

5. Pilih Buat.

Langkah 6: Jalankan tugas dan lihat hasilnya

Sekarang setelah sistem file Amazon EFS Anda dibuat dan ada konten web untuk penampung NGINX untuk disajikan, Anda dapat menjalankan tugas menggunakan definisi tugas yang Anda buat. Server web NGINX melayani halaman HTML sederhana Anda. Jika Anda memperbarui konten di sistem file Amazon EFS Anda, perubahan tersebut disebarkan ke wadah apa pun yang juga telah memasang sistem file tersebut.

Tugas berjalan di subnet yang Anda tentukan untuk cluster.

Untuk menjalankan tugas dan melihat hasilnya menggunakan konsol

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman klaster, pilih klaster tempat tugas mandiri dijalankan.

   Tentukan sumber daya dari tempat Anda meluncurkan layanan.

Untuk memulai layanan dari	Langkah-langkah
Klaster	a. Pada halaman Klaster, pilih klaster untuk membuat layanan. b. Dari tab Tugas, pilih Jalankan tugas baru.
Jenis peluncuran	a. Pada halaman Tugas, pilih definisi tugas. b. Jika ada lebih dari satu revisi, pilih revisi. c. Pilih Buat, Jalankan tugas.

3. (Opsional) Pilih bagaimana tugas terjadwal Anda didistribusikan di seluruh infrastruktur klaster Anda. Perluas konfigurasi Compute, lalu lakukan hal berikut:

Metode distribusi	Langkah-langkah
Jenis peluncuran	a. Di bagian Compute options, pilih Launch type. b. Untuk jenis Peluncuran, pilih EC2.

4. Untuk Jenis aplikasi, pilih Tugas.
5. Untuk definisi Tugas, pilih definisi `efs-tutorial` tugas yang Anda buat sebelumnya.
6. Untuk tugas yang diinginkan, masukkan1.
7. Pilih Buat.
8. Pada halaman Cluster, pilih Infrastruktur.

9. Di bawah Instans Kontainer, pilih instance kontainer untuk dihubungkan.
10. Pada halaman Container Instance, di bawah Networking, rekam IP Publik untuk instans Anda.
11. Buka browser dan masukkan alamat IP publik. Anda akan melihat pesan berikut:

```
It works!
You are using an Amazon EFS file system for persistent container storage.
```

> **Note**
>
> Jika Anda tidak melihat pesan, pastikan bahwa grup keamanan untuk instance container Anda mengizinkan lalu lintas jaringan masuk pada port 80 dan grup keamanan untuk sistem file Anda memungkinkan akses masuk dari instance container.

Menggunakan sistem file FSx for Windows File Server dengan Amazon ECS

FSx for Windows File Server menyediakan server file Microsoft Windows yang dikelola sepenuhnya, yang didukung oleh sistem file Windows yang sepenuhnya asli. Saat menggunakan FSx for Windows File Server bersama dengan Amazon ECS, Anda dapat menyediakan tugas Windows Anda dengan penyimpanan file persisten, terdistribusi, bersama, dan statis. Untuk informasi selengkapnya, lihat Apa itu FSx for Windows File Server? di Panduan Pengguna FSx for Windows File Server.

Anda dapat menggunakan FSx for Windows File Server untuk menyebarkan beban kerja Windows yang memerlukan akses ke penyimpanan eksternal bersama, penyimpanan regional yang sangat tersedia, atau penyimpanan throughput tinggi. Anda dapat memasang satu atau beberapa volume sistem file FSx for Windows File Server ke wadah yang berjalan pada instance EC2 Windows. Anda dapat membagikan volume sistem file FSx for Windows File Server di antara beberapa kontainer dalam satu tugas ECS.

> **Note**
>
> FSx for Windows File Server mungkin tidak tersedia di semua Wilayah. Untuk informasi selengkapnya tentang Wilayah mana yang mendukung FSx for Windows File Server, lihat Titik Akhir dan Kuota Amazon FSx di. Referensi Umum AWS

Dalam tutorial ini, Anda meluncurkan instance ECS Optimized Windows yang menghosting sistem file FSx for Windows File Server dan wadah yang dapat mengakses sistem file. Untuk melakukan ini, pertama-tama Anda membuat Direktori Aktif Microsoft AWS Directory Service AWS Terkelola. Kemudian, Anda membuat sistem file Amazon FSx for Windows File Server dan cluster ECS dengan instans Amazon EC2 dan definisi tugas ECS. Anda mengonfigurasi definisi tugas untuk kontainer Anda untuk menggunakan sistem file FSx for Windows File Server. Akhirnya, Anda menguji sistem file.

Dibutuhkan 20 hingga 45 menit setiap kali Anda meluncurkan atau menghapus baik Active Directory atau FSx for Windows File Server sistem file. Bersiaplah untuk menyimpan setidaknya 90 menit untuk menyelesaikan tutorial atau menyelesaikan tutorial selama beberapa sesi.

Prasyarat untuk tutorial

- Pengguna administratif. Lihat Siapkan untuk menggunakan Amazon ECS.
- (Opsional) Sebuah `PEM` key pair untuk menghubungkan ke instans Windows EC2 Anda melalui akses RDP. Untuk informasi tentang cara membuat pasangan kunci, lihat pasangan kunci dan instans Windows Amazon EC2 dalam Panduan Pengguna untuk Instans Windows.
- Sebuah VPC dengan setidaknya satu subnet publik dan satu subnet privat, dan satu grup keamanan. Anda dapat menggunakan VPC default Anda. Anda tidak memerlukan gateway atau perangkat NAT. AWS Directory Service tidak mendukung Network Address Translation (NAT) dengan Direktori Aktif. Agar ini berfungsi, Active Directory, FSx for Windows File Server sistem file, ECS Cluster, dan instans EC2 harus berada di dalam VPC Anda. Untuk informasi lebih lanjut mengenai VPC dan Direktori Aktif, lihat Konfigurasi wizard konsol Amazon VPC dan Prasyarat Microsoft AD Terkelola AWS.
- Izin IAM ecsInstanceRole dan ecsTaskExecution Peran dikaitkan dengan akun Anda. Peran terkait layanan ini memungkinkan layanan melakukan panggilan API dan mengakses kontainer, rahasia, direktori, dan server file atas nama Anda.

Langkah 1: Buat peran akses IAM

Buat klaster dengan AWS Management Console.

1. Lihat Peran IAM instans wadah Amazon ECS untuk memeriksa apakah Anda memiliki ecsInstanceRole dan untuk melihat bagaimana Anda dapat membuatnya jika Anda tidak memilikinya.

2. Kami merekomendasikan bahwa kebijakan peran disesuaikan untuk izin minimum di lingkungan produksi aktual. Untuk tujuan mengerjakan tutorial ini, verifikasi bahwa kebijakan AWS terkelola berikut ini terlampir pada AndaecsInstanceRole. Lampirkan kebijakan jika belum dilampirkan.

   - AmazonEC2 EC2peran ContainerServicefor
   - AmazonSSM ManagedInstanceCore
   - AmazonSSM DirectoryServiceAccess

   Untuk melampirkan kebijakan AWS terkelola.

   a. Buka konsol IAM.
   b. Di panel navigasi, pilih Peran.
   c. Pilih peran yang AWS dikelola.
   d. Pilih Izin, Lampirkan kebijakan.
   e. Untuk mempersempit kebijakan tersedia yang akan dilampirkan, gunakan Filter.
   f. Pilih kebijakan yang sesuai dan pilih Lampirkan kebijakan.

3. Lihat Peran IAM eksekusi tugas Amazon ECS untuk memeriksa apakah Anda memiliki ecsTaskExecution Peran dan untuk melihat bagaimana Anda dapat membuatnya jika Anda tidak memilikinya.

   Kami merekomendasikan bahwa kebijakan peran disesuaikan untuk izin minimum di lingkungan produksi aktual. Untuk tujuan mengerjakan tutorial ini, verifikasi bahwa kebijakan AWS terkelola berikut dilampirkan ke ecsTaskExecution Peran Anda. Lampirkan kebijakan jika mereka belum dilampirkan. Gunakan prosedur yang diberikan di bagian sebelumnya untuk melampirkan kebijakan yang AWS dikelola.

   - SecretsManagerReadWrite
   - AmazonF SxReadOnlyAccess
   - AmazonSSM ReadOnlyAccess
   - AmazonECS TaskExecutionRolePolicy

Langkah 2: Membuat Direktori Aktif Windows (AD)

1. Ikuti langkah-langkah yang dijelaskan dalam Create Your AWS Managed AD Directory di AWS Directory Service Administration Guide. Gunakan VPC yang telah Anda tunjuk untuk tutorial ini.

Pada Langkah 3 Buat Direktori AD Terkelola AWS Anda, simpan nama pengguna dan kata sandi untuk digunakan dalam langkah berikut. Selain itu, perhatikan nama domain yang sepenuhnya memenuhi syarat untuk langkah berikutnya. Anda dapat melanjutkan untuk menyelesaikan langkah berikut selagi Direktori Aktif sedang dibuat.

2. Buat AWS rahasia Secrets Manager untuk digunakan dalam langkah-langkah berikut. Untuk informasi selengkapnya, lihat Memulai AWS Secrets Manager di Panduan Pengguna AWS Secrets Manager.

   a. Buka konsol Secrets Manager.

   b. Klik Menyimpan rahasia baru.

   c. Pilih Tipe rahasia lainnya.

   d. Untuk Kunci/nilai rahasia, di baris pertama, buat kunci **`username`** dengan nilai **`admin`**. Klik pada + Tambahkan baris.

   e. Di baris baru, buat kunci **`password`**. Untuk nilai, ketik kata sandi yang Anda masukkan di Langkah 3 Buat Direktori AD AWS Terkelola Anda.

   f. Klik pada tombol Selanjutnya.

   g. Masukkan nama dan deskripsi. Klik Berikutnya.

   h. Klik Berikutnya. Klik Simpan.

   i. Dari daftar halaman Rahasia, klik pada rahasia yang baru saja Anda buat.

   j. Simpan ARN rahasia baru untuk digunakan dalam langkah-langkah berikut.

   k. Anda dapat melanjutkan ke langkah berikutnya selagi Direktori Aktif sedang dibuat.

Langkah 3: Verifikasi dan perbarui grup keamanan

Pada langkah ini, Anda memverifikasi dan memperbarui aturan untuk grup keamanan yang Anda gunakan. Untuk hal ini, Anda dapat menggunakan grup keamanan default yang dibuat untuk VPC Anda.

Verifikasi dan perbarui grup keamanan.

Anda perlu membuat atau mengedit grup keamanan untuk mengirim data dari dan ke port, yang dijelaskan dalam Grup Keamanan VPC Amazon di Panduan Pengguna FSx for Windows File Server. Anda dapat melakukan ini dengan membuat aturan grup keamanan inbound ditampilkan di baris pertama dari tabel aturan inbound berikut. Mengizinkan lalu lintas masuk dari antarmuka jaringan (dan instans terkait-nya) yang ditugaskan ke grup keamanan. Semua sumber daya cloud yang Anda

buat berada dalam VPC yang sama dan melekat pada grup keamanan yang sama. Oleh karena itu, aturan ini memungkinkan lalu lintas dikirim ke dan dari sistem file FSx for Windows File Server, Active Directory, dan instance ECS sesuai kebutuhan. Aturan inbound lainnya mengizinkan lalu lintas untuk melayani situs web dan akses RDP untuk menghubungkan ke instans ECS Anda.

Tabel berikut menunjukkan aturan grup keamanan inbound mana yang diperlukan untuk tutorial ini.

Tipe	Protokol	Rentang port	Sumber
Semua Lalu lintas	Semua	Semua	*sg-securitygroup*
HTTPS	TCP	443	0.0.0.0/0
RDP	TCP	3389	alamat IP laptop Anda

Tabel berikut menunjukkan aturan grup keamanan inbound mana yang diperlukan untuk tutorial ini.

Tipe	Protokol	Rentang Port	Tujuan
Semua lalu lintas	Semua	Semua	0.0.0.0/0

1. Buka Konsol EC2 dan pilih Grup Keamanan dari menu sebelah kiri.
2. Dari daftar grup keamanan yang sekarang ditampilkan, pilih centang pada kotak centang di sebelah kiri grup keamanan yang Anda gunakan untuk tutorial ini.

   Detail grup keamanan Anda ditampilkan.
3. Edit aturan inbound dan outbound dengan memilih tab Aturan inbound atau Aturan outbound dan memilih tombol Edit aturan inbound atau Edit aturan outbound. Edit aturan untuk mencocokkan yang ditampilkan dalam tabel sebelumnya. Setelah Anda membuat instans EC2 Anda nanti dalam tutorial ini, edit aturan inbound sumber RDP dengan alamat IP publik dari instans EC2 Anda seperti yang dijelaskan dalam Connect ke instans Windows Anda dari Panduan Pengguna Amazon EC2 untuk Instans Windows.

Langkah 4: Buat sistem file FSx for Windows File Server

Setelah grup keamanan Anda diverifikasi dan diperbarui dan Direktori Aktif Anda dibuat dan berada dalam status aktif, buat sistem file FSx for Windows File Server di VPC yang sama dengan Direktori Aktif Anda. Gunakan langkah-langkah berikut untuk membuat sistem file FSx for Windows File Server untuk tugas Windows Anda.

Buat sistem file pertama Anda.

1. Buka konsol Amazon FSx.
2. Pada dasbor, pilih Buat sistem file untuk memulai wizard pembuatan sistem file.
3. Pada halaman Pilih jenis sistem file, pilih FSx for Windows File Server, lalu pilih Berikutnya. Halaman Buat sistem file akan muncul.
4. Pada bagian Detail sistem file, berikan nama untuk sistem file Anda. Penamaan sistem file Anda membuatnya lebih mudah untuk menemukan dan mengelola sistem file tersebut. Anda dapat menggunakan hingga 256 karakter Unicode. Karakter yang diizinkan adalah huruf, angka, spasi, dan karakter khusus plus tanda (+). tanda minus (-), tanda sama dengan (=), titik (.), garis bawah (_), titik dua (:), garis miring (/).
5. Untuk Tipe deployment Pilih Single-AZ untuk men-deploy sistem file yang di-deploy di Availability Zone tunggal. Single-Az 2 adalah generasi terbaru dari sistem file Availability Zone tunggal, dan mendukung penyimpanan SSD dan HDD.
6. Untuk jenis Storage, pilih HDD.
7. Untuk kapasitas penyimpanan, masukkan kapasitas penyimpanan minimum.
8. Pertahankan Kapasitas throughput pada pengaturan default-nya.
9. Di bagian Jaringan & keamanan, pilih VPC Amazon yang sama dengan yang Anda pilih untuk direktori Anda AWS Directory Service .
10. Untuk Grup Keamanan VPC, pilih grup keamanan yang Anda verifikasi di Langkah 3: Verifikasi dan perbarui grup keamanan.
11. Untuk otentikasi Windows, pilih Direktori Aktif Microsoft AWS Terkelola, lalu pilih AWS Directory Service direktori Anda dari daftar.
12. Untuk Enkripsi, simpan default pengaturan Kunci enkripsi dari aws/fsx (default).
13. Simpan pengaturan default untuk Preferensi pemeliharaan.
14. Klik pada tombol Selanjutnya.

15. Tinjau konfigurasi sistem file yang ditampilkan pada halaman Buat sistem file. Untuk referensi Anda, perhatikan pengaturan sistem file mana yang dapat Anda modifikasi setelah sistem file dibuat. Pilih Buat sistem file.
16. Catat ID sistem file. Anda perlu menggunakannya di langkah selanjutnya.

    Anda dapat melanjutkan ke langkah berikutnya untuk membuat cluster dan instans EC2 sementara sistem file FSx for Windows File Server sedang dibuat.

Langkah 5: Buat cluster Amazon ECS

Buat cluster menggunakan konsol Amazon ECS

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.
3. Di panel navigasi, pilih Klaster.
4. Pada halaman Klaster, pilih Buat klaster.
5. Di bawah konfigurasi Cluster, untuk nama Cluster, masukkan windows-fsx-cluster.
6. Perluas Infrastruktur, hapus AWS Fargate (tanpa server) lalu pilih instans Amazon EC2.
    - Untuk membuat grup Auto Scaling, dari grup Auto Scaling (ASG), pilih Buat grup baru, lalu berikan detail berikut tentang grup:
        - Untuk Sistem Operasi/Arsitektur, pilih Windows Server 2019 Core.
        - Untuk jenis instans EC2, pilih t2.medium atau t2.micro.
7. Pilih Buat.

Langkah 6: Buat instans Amazon EC2 Amazon ECS yang dioptimalkan oleh Amazon

Buat instance penampung Amazon ECS Windows.

Untuk membuat instans Amazon ECS

1. Gunakan `aws ssm get-parameters` perintah untuk mengambil nama AMI untuk Wilayah yang menghosting VPC Anda. Untuk informasi selengkapnya, lihat Mengambil metadata AMI yang dioptimalkan Amazon ECS.

2. Gunakan konsol Amazon EC2 untuk meluncurkan instans.

   a. Buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.

   b. Dari bilah navigasi, pilih Wilayah untuk digunakan.

   c. Dari Dasbor EC2, pilih Meluncurkan instans.

   d. Untuk Nama, masukkan nama unik.

   e. Untuk Gambar Aplikasi dan OS (Gambar Mesin Amazon), di bidang pencarian, masukkan nama AMI yang Anda ambil.

   f. Untuk jenis Instance, pilih t2.medium atau t2.micro.

   g. Untuk Key pair (login), pilih key pair. Jika Anda tidak menentukan key pair, Anda

   h. Di bawah Pengaturan jaringan, untuk VPC dan Subnet, pilih VPC Anda dan subnet publik.

   i. Di bawah Pengaturan jaringan, untuk grup Keamanan, pilih grup keamanan yang ada, atau buat yang baru. Pastikan bahwa grup keamanan yang Anda pilih memiliki aturan masuk dan keluar yang ditentukan Prasyarat untuk tutorial

   j. Di bawah Pengaturan jaringan, untuk Auto-assign IP Publik, pilih Aktifkan.

   k. Perluas Detail lanjutan, lalu untuk direktori Gabung Domain, pilih ID Direktori Aktif yang Anda buat. Domain opsi ini bergabung dengan AD Anda ketika instans EC2 diluncurkan.

   l. Di bawah Detail lanjutan, untuk profil instans IAM, pilih ecsInstanceRole.

   m. Konfigurasikan instans penampung Amazon ECS Anda dengan data pengguna berikut. Di bawah Detail Lanjutan, tempelkan skrip berikut ke bidang data Pengguna, ganti *cluster_name dengan nama* cluster Anda.

   ```
   <powershell>
   Initialize-ECSAgent -Cluster windows-fsx-cluster -EnableTaskIAMRole
   </powershell>
   ```

   n. Saat Anda siap, pilih bidang pengakuan, lalu pilih Luncurkan Instans.

   o. Halaman konfirmasi memberi tahu Anda bahwa instans Anda akan diluncurkan. Pilih Lihat Instans untuk menutup halaman konfirmasi dan kembali ke konsol tersebut.

3. Buka konsol di https://console.aws.amazon.com/ecs/v2.

4. Di panel navigasi, pilih Cluster, lalu pilih. windows-fsx-cluster

5. Pilih tab Infrastructure dan verifikasi bahwa instans Anda telah terdaftar di windows-fsx-clustercluster.

Langkah 7: Daftarkan ketentuan tugas Windows

Sebelum Anda dapat menjalankan wadah Windows di cluster Amazon ECS Anda, Anda harus mendaftarkan definisi tugas. Contoh definisi tugas berikut menampilkan halaman web sederhana. Tugas meluncurkan dua kontainer yang memiliki akses ke sistem file FSx. Kontainer pertama menulis file HTML ke sistem file. Kontainer kedua mengunduh file HTML dari sistem file dan menyediakan halaman web.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Definisi tugas.
3. Pilih Buat definisi tugas baru, Buat definisi tugas baru dengan JSON.
4. Di kotak editor JSON, ganti nilai untuk peran eksekusi tugas Anda dan detail tentang sistem file FSx Anda, lalu pilih Simpan.

```
{
    "containerDefinitions": [
        {
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "portMappings": [],
            "command": ["New-Item -Path C:\\fsx-windows-dir\\index.html -ItemType
 file -Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body
 {margin-top: 40px; background-color: #333;} </style> </head><body> <div
 style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1> <h2>It
 Works!</h2> <p>You are using Amazon FSx for Windows File Server file system for
 persistent container storage.</p>' -Force"],
            "cpu": 512,
            "memory": 256,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "essential": false,
            "name": "container1",
            "mountPoints": [
                {
                    "sourceVolume": "fsx-windows-dir",
                    "containerPath": "C:\\fsx-windows-dir",
                    "readOnly": false
                }
            ]
```

```
        },
        {
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "portMappings": [
                {
                    "hostPort": 443,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ],
            "command": ["Remove-Item -Recurse C:\\inetpub\\wwwroot\\* -Force;
 Start-Sleep -Seconds 120; Move-Item -Path C:\\fsx-windows-dir\\index.html -
Destination C:\\inetpub\\wwwroot\\index.html -Force; C:\\ServiceMonitor.exe
 w3svc"],
            "mountPoints": [
                {
                    "sourceVolume": "fsx-windows-dir",
                    "containerPath": "C:\\fsx-windows-dir",
                    "readOnly": false
                }
            ],
            "cpu": 512,
            "memory": 256,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "essential": true,
            "name": "container2"
        }
    ],
    "family": "fsx-windows",
    "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
    "volumes": [
        {
            "name": "fsx-windows-dir",
            "fsxWindowsFileServerVolumeConfiguration": {
                "fileSystemId": "fs-0eeb5730b2EXAMPLE",
                "authorizationConfig": {
                    "domain": "example.com",
                    "credentialsParameter": "arn:arn-1234"
                },
                "rootDirectory": "share"
```

```
            }
        }
    ]
}
```

Langkah 8: Jalankan tugas dan lihat hasilnya

Sebelum menjalankan tugas, verifikasi bahwa status sistem file FSx for Windows File Server Anda Tersedia. Setelah tersedia, Anda dapat menjalankan tugas menggunakan ketentuan tugas yang Anda buat. Tugas dimulai dengan membuat kontainer yang mengacak file HTML di antara mereka menggunakan sistem file. Setelah pengacakan, web server melayani halaman HTML sederhana.

> **Note**
>
> Anda mungkin tidak dapat terhubung ke situs web dari VPN.

Jalankan tugas dan lihat hasilnya dengan konsol Amazon ECS.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Cluster, lalu pilih. windows-fsx-cluster
3. Pilih tab Tugas, lalu pilih Jalankan tugas baru.
4. Untuk Tipe peluncuran, pilih EC2.
5. Di bawah konfigurasi Deployment, untuk Task Definition, pilih fsx-windows, lalu pilih Create.
6. Saat status tugas Anda SEDANG BERJALAN, pilih ID tugas.
7. Di bawah Container, saat status container1 STOPTED, pilih container2 untuk melihat detail kontainer.
8. Di bawah rincian Container untuk container2, pilih Network binding dan kemudian klik pada alamat IP eksternal yang terkait dengan container. Peramban Anda akan membuka dan menampilkan pesan berikut.

```
Amazon ECS Sample App
It Works!
You are using Amazon FSx for Windows File Server file system for persistent
 container storage.
```

> **Note**
>
> Mungkin perlu beberapa menit agar pesan ditampilkan. Jika Anda tidak melihat pesan ini setelah beberapa menit, periksa apakah Anda tidak menjalankan VPN dan pastikan bahwa grup keamanan untuk instance container Anda mengizinkan lalu lintas HTTP jaringan masuk pada port 443.

Langkah 9: Membersihkan

> **Note**
>
> Dibutuhkan 20 hingga 45 menit untuk menghapus sistem file FSx for Windows File Server atau AD. Anda harus menunggu hingga operasi penghapusan sistem file FSx for Windows File Server selesai sebelum memulai operasi penghapusan AD.

Hapus sistem file FSx for Windows File Server.

1. Buka konsol Amazon FSx
2. Pilih tombol radio di sebelah kiri sistem file FSx for Windows File Server yang baru saja Anda buat.
3. Pilih Tindakan.
4. Pilih sistem file untuk dihapus.

Hapus iklan.

1. Buka konsol AWS Directory Service.
2. Pilih tombol radio di sebelah kiri AD yang baru saja Anda buat.
3. Pilih Tindakan.
4. Pilih Hapus direktori.

Hapus klaster .

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.

2. Di panel navigasi, pilih Cluster, lalu pilih. fsx-windows-cluster
3. Pilih Hapus klaster.
4. Masukkan frasa dan kemudian pilih Hapus.

Mengakhiri instans EC2.

1. Buka konsol Amazon EC2.
2. Dari menu sebelah kiri, pilih Instans.
3. Centang kotak di sebelah kiri instans EC2 yang Anda buat.
4. Klik status Instance, Terminate instance.

Hapus rahasia.

1. Buka konsol Secrets Manager.
2. Pilih rahasia yang Anda buat untuk perjalanan ini.
3. Klik Tindakan.
4. Pilih Hapus rahasia.

Menyebarkan Fluent Bit di Amazon ECS untuk wadah Windows

Fluent Bit adalah prosesor log dan router yang cepat dan fleksibel yang didukung oleh berbagai sistem operasi. Ini dapat digunakan untuk merutekan log ke berbagai AWS tujuan seperti Amazon CloudWatch Log, Firehose Amazon S3, dan Amazon Service. OpenSearch Fluent Bit mendukung solusi mitra umum seperti Datadog, Splunk, dan server HTTP kustom. Untuk informasi lebih lanjut tentang Fluent Bit, lihat Fluent Bit situs web.

Gambar AWS for Fluent Bit tersedia di Amazon ECR di Galeri Publik Amazon ECR dan di repositori ECR Amazon di sebagian besar Wilayah untuk ketersediaan tinggi. Untuk informasi lebih lanjut, lihat aws-for-fluent-bit di GitHub situs web.

Tutorial ini memandu Anda melalui cara menerapkan container Fluent Bit pada instance Windows mereka yang berjalan di Amazon ECS untuk mengalirkan log yang dihasilkan oleh tugas Windows ke Amazon CloudWatch untuk pencatatan terpusat.

Tutorial ini menggunakan pendekatan berikut:

- Fluent Bit berjalan sebagai layanan dengan strategi penjadwalan Daemon. Strategi ini memastikan bahwa satu instance Fluent Bit selalu berjalan pada instance container di cluster.
  - Mendengarkan pada port 24224 menggunakan plug-in input forward.
  - Ekspos port 24224 ke host sehingga runtime docker dapat mengirim log ke Fluent Bit menggunakan port yang terbuka ini.
  - Memiliki konfigurasi yang memungkinkan Fluent Bit untuk mengirim catatan log ke tujuan tertentu.
- Luncurkan semua wadah tugas Amazon ECS lainnya menggunakan driver logging lancar. Untuk informasi selengkapnya, lihat Driver logging fluentd di situs web dokumentasi Docker.
  - Docker terhubung ke soket TCP 24224 di localhost di dalam namespace host.
  - Agen Amazon ECS menambahkan label ke wadah yang mencakup nama cluster, nama keluarga definisi tugas, nomor revisi definisi tugas, ARN tugas, dan nama penampung. Informasi yang sama ditambahkan ke catatan log menggunakan opsi label dari driver logging docker fluentd. Untuk informasi selengkapnya, lihat label, labels-regex, env, dan env-regex di situs web dokumentasi Docker.
  - Karena `async` opsi driver logging fluentd disetel `ketrue`, ketika wadah Fluent Bit dimulai ulang, docker menyangga log hingga wadah Fluent Bit dimulai ulang. Anda dapat meningkatkan batas buffer dengan mengatur fluentd-buffer-limit opsi. Untuk informasi selengkapnya, lihat fluentd-buffer-limitdi situs web dokumentasi Docker.

Alur kerja adalah sebagai berikut:

- Kontainer Fluent Bit dimulai dan mendengarkan pada port 24224 yang diekspos ke host.
- Fluent Bit menggunakan kredensyal peran IAM tugas yang ditentukan dalam definisi tugasnya.
- Tugas lain yang diluncurkan pada instance yang sama menggunakan driver logging docker fluentd untuk terhubung ke wadah Fluent Bit pada port 24224.
- Saat wadah aplikasi menghasilkan log, docker runtime menandai catatan tersebut, menambahkan metadata tambahan yang ditentukan dalam label, dan kemudian meneruskannya di port 24224 di namespace host.
- Fluent Bit menerima catatan log pada port 24224 karena terkena namespace host.
- Fluent Bit melakukan pemrosesan internal dan merutekan log seperti yang ditentukan.

Tutorial ini menggunakan konfigurasi CloudWatch Fluent Bit default yang melakukan hal berikut:

- Membuat grup log baru untuk setiap cluster dan keluarga definisi tugas.
- Membuat aliran log baru untuk setiap wadah tugas di grup log yang dihasilkan di atas setiap kali tugas baru diluncurkan. Setiap aliran akan ditandai dengan id tugas tempat wadah tersebut berada.
- Menambahkan metadata tambahan termasuk nama cluster, ARN tugas, nama wadah tugas, keluarga definisi tugas, dan nomor revisi definisi tugas di setiap entri log.

  Misalnya, jika Anda memiliki `task_1` dengan `container_1` dan `container_2` dan t `ask_2` dengan`container_3`, maka berikut ini adalah aliran CloudWatch log:

  - `/aws/ecs/windows.ecs_task_1`

    `task-out.`*`TASK_ID`*`.container_1`

    `task-out.`*`TASK_ID`*`.container_2`
  - `/aws/ecs/windows.ecs_task_2`

    `task-out.`*`TASK_ID`*`.container_3`

Langkah-langkah

Prasyarat

Tutorial ini mengasumsikan bahwa prasyarat berikut telah diselesaikan:

- Versi terbaru diinstal dan dikonfigurasi. AWS CLI Untuk informasi selengkapnya, silakan lihat Menginstal AWS Command Line Interface.
- Gambar `aws-for-fluent-bit` kontainer tersedia untuk sistem operasi Windows berikut:
  - Windows Server 2019 Core
  - Windows Server 2019 Full
  - Windows Server 2022 Inti
  - Windows Server 2022 Lengkap
- Langkah-langkah di Siapkan untuk menggunakan Amazon ECS telah diselesaikan.
- Anda memiliki cluster. Dalam tutorial ini, nama cluster adalah FluentBit-cluster.
- Anda memiliki VPC dengan subnet publik tempat instans EC2 akan diluncurkan. Anda dapat menggunakan VPC default Anda. Anda juga dapat menggunakan subnet pribadi yang memungkinkan CloudWatch titik akhir Amazon mencapai subnet. Untuk informasi selengkapnya tentang CloudWatch titik akhir Amazon, lihat CloudWatch titik akhir dan kuota Amazon di. Referensi Umum AWS Untuk informasi tentang cara menggunakan wizard VPC Amazon untuk membuat VPC, lihat. the section called “Buat virtual private cloud”

Langkah 1: Buat peran akses IAM

Buat peran Amazon ECS IAM.

1. Buat peran instans penampung Amazon ECS bernama "ecsInstanceRole”. Untuk informasi selengkapnya, lihat peran IAM instans penampung Amazon ECS.
2. Buat peran IAM untuk tugas Fluent Bit bernama. `fluentTaskRole` Untuk informasi selengkapnya, lihat the section called “Tugas peran IAM”.

   Izin IAM yang diberikan dalam peran IAM ini diasumsikan oleh wadah tugas. Untuk memungkinkan Bit Lancar mengirim log CloudWatch, Anda harus melampirkan izin berikut ke peran IAM tugas.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "logs:CreateLogStream",
```

```
            "logs:CreateLogGroup",
            "logs:DescribeLogStreams",
            "logs:PutLogEvents"
        ],
        "Resource": "*"
    }
    ]
}
```

3. Lampirkan kebijakan pada peran tersebut.

    a. Simpan konten di atas dalam file bernama`fluent-bit-policy.json`.

    b. Jalankan perintah berikut untuk melampirkan kebijakan inline ke peran `fluentTaskRole` IAM.

    ```
    aws iam put-role-policy --role-name fluentTaskRole --policy-name
     fluentTaskPolicy --policy-document file://fluent-bit-policy.json
    ```

Langkah 2: Buat instance penampung Amazon ECS Windows

Buat instance penampung Amazon ECS Windows.

Untuk membuat instans Amazon ECS

1. Gunakan `aws ssm get-parameters` perintah untuk mengambil ID AMI untuk Wilayah yang menghosting VPC Anda. Untuk informasi selengkapnya, lihat Mengambil metadata AMI yang dioptimalkan Amazon ECS.
2. Gunakan konsol Amazon EC2 untuk meluncurkan instans.

    a. Buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.

    b. Dari bilah navigasi, pilih Wilayah untuk digunakan.

    c. Dari Dasbor EC2, pilih Meluncurkan instans.

    d. Untuk Nama, masukkan nama unik.

    e. Untuk Gambar Aplikasi dan OS (Amazon Machine Image), pilih AMI yang Anda ambil pada langkah pertama.

    f. Untuk jenis Instance, pilih`t3.xlarge`.

    g. Untuk Key pair (login), pilih key pair.

h. Di bawah Pengaturan jaringan, untuk grup Keamanan, pilih grup keamanan yang ada, atau buat yang baru.

i. Di bawah Pengaturan jaringan, untuk Auto-assign IP Publik, pilih Aktifkan.

j. Di bawah Detail lanjutan, untuk profil instans IAM, pilih ecsInstanceRole.

k. Konfigurasikan instans penampung Amazon ECS Anda dengan data pengguna berikut. Di bawah Detail Lanjutan, tempelkan skrip berikut ke bidang data Pengguna, ganti *cluster_name dengan nama* cluster Anda.

```
<powershell>
Import-Module ECSTools
Initialize-ECSAgent -Cluster cluster-name -EnableTaskENI -EnableTaskIAMRole -
LoggingDrivers '["awslogs","fluentd"]'
</powershell>
```

l. Saat Anda siap, pilih bidang pengakuan, lalu pilih Luncurkan Instans.

m. Halaman konfirmasi memberi tahu Anda bahwa instans Anda akan diluncurkan. Pilih Lihat Instans untuk menutup halaman konfirmasi dan kembali ke konsol tersebut.

Langkah 3: Konfigurasikan Bit Lancar

Anda dapat menggunakan konfigurasi default berikut yang disediakan oleh AWS untuk memulai dengan cepat:

- Amazon CloudWatch yang didasarkan pada plug-in Fluent Bit untuk Amazon CloudWatch pada Manual Resmi Fluent Bit.

Atau, Anda dapat menggunakan konfigurasi default lain yang disediakan oleh AWS. Untuk informasi selengkapnya, lihat Mengganti titik masuk untuk gambar Windows di situs web Github`aws-for-fluent-bit`.

Konfigurasi Amazon CloudWatch Fluent Bit default ditunjukkan di bawah ini.

Ganti variabel berikut:

- *wilayah* dengan Wilayah tempat Anda ingin mengirim CloudWatch log Amazon.

```
[SERVICE]
```

```
    Flush                5
    Log_Level            info
    Daemon               off

[INPUT]
    Name                 forward
    Listen               0.0.0.0
    Port                 24224
    Buffer_Chunk_Size    1M
    Buffer_Max_Size      6M
    Tag_Prefix           ecs.

# Amazon ECS agent adds the following log keys as labels to the docker container.
# We would use fluentd logging driver to add these to log record while sending it to
 Fluent Bit.
[FILTER]
    Name                 modify
    Match                ecs.*
    Rename               com.amazonaws.ecs.cluster ecs_cluster
    Rename               com.amazonaws.ecs.container-name ecs_container_name
    Rename               com.amazonaws.ecs.task-arn ecs_task_arn
    Rename               com.amazonaws.ecs.task-definition-family
 ecs_task_definition_family
    Rename               com.amazonaws.ecs.task-definition-version
 ecs_task_definition_version

[FILTER]
    Name                 rewrite_tag
    Match                ecs.*
    Rule                 $ecs_task_arn ^([a-z-:0-9]+)/([a-zA-Z0-9-_]+)/([a-z0-9]+)$
 out.$3.$ecs_container_name false
    Emitter_Name         re_emitted

[OUTPUT]
    Name                 cloudwatch_logs
    Match                out.*
    region               region
    log_group_name       fallback-group
    log_group_template   /aws/ecs/$ecs_cluster.$ecs_task_definition_family
    log_stream_prefix    task-
    auto_create_group    On
```

Setiap log yang masuk ke Fluent Bit memiliki tag yang Anda tentukan, atau dibuat secara otomatis ketika Anda tidak menyediakannya. Tag dapat digunakan untuk merutekan log yang berbeda ke tujuan yang berbeda. Untuk informasi tambahan, lihat Tag di Manual Resmi Fluent Bit.

Konfigurasi Fluent Bit yang dijelaskan di atas memiliki properti berikut:

- Plug-in input maju mendengarkan lalu lintas masuk pada port TCP 24224.
- Setiap entri log yang diterima pada port tersebut memiliki tag yang dimodifikasi oleh plug-in input forward untuk mengawali catatan dengan string. `ecs.`
- Pipeline internal Fluent Bit merutekan entri log untuk memodifikasi filter menggunakan regex Match. Filter ini menggantikan kunci dalam catatan log JSON ke format yang dapat dikonsumsi Fluent Bit.
- Entri log yang dimodifikasi kemudian dikonsumsi oleh filter rewrite_tag. Filter ini mengubah tag catatan log ke format keluar. *`TASK_ID`*. *`CONTAINER_NAME`*.
- Tag baru akan dirutekan ke plug-in cloudwatch_logs keluaran yang membuat grup log dan aliran seperti yang dijelaskan sebelumnya dengan menggunakan opsi dan plug-in keluaran. `log_group_template` `log_stream_prefix` CloudWatch Untuk informasi tambahan, lihat Parameter konfigurasi di Manual Resmi Bit Lancar.

Langkah 4: Daftarkan definisi tugas Windows Fluent Bit yang merutekan log ke CloudWatch

Daftarkan definisi tugas Windows Fluent Bit yang merutekan log ke CloudWatch.

> **Note**
>
> Definisi tugas ini mengekspos port kontainer Fluent Bit 24224 ke port host 24224. Pastikan port ini tidak terbuka di grup keamanan instans EC2 Anda untuk mencegah akses dari luar.

Untuk mendaftarkan ketentuan tugas

1. Buat file yang diberi nama `fluent-bit.json` dengan konten berikut ini.

   Ganti variabel berikut:

   - *`task-iam-role`*dengan Nama Sumber Daya Amazon (ARN) dari peran IAM tugas Anda

- *wilayah* dengan Wilayah tempat tugas Anda berjalan

```
{
  "family": "ecs-windows-fluent-bit",
  "taskRoleArn": "task-iam-role",
  "containerDefinitions": [
    {
      "name": "fluent-bit",
      "image": "public.ecr.aws/aws-observability/aws-for-fluent-
bit:windowsservercore-latest",
      "cpu": 512,
      "portMappings": [
        {
          "hostPort": 24224,
          "containerPort": 24224,
          "protocol": "tcp"
        }
      ],
      "entryPoint": [
        "Powershell",
        "-Command"
      ],
      "command": [
        "C:\\entrypoint.ps1 -ConfigFile C:\\ecs_windows_forward_daemon\
\cloudwatch.conf"
      ],
      "environment": [
        {
          "name": "AWS_REGION",
          "value": "region"
        }
      ],
      "memory": 512,
      "essential": true,
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "/ecs/fluent-bit-logs",
          "awslogs-region": "region",
          "awslogs-stream-prefix": "flb",
          "awslogs-create-group": "true"
        }
```

```
      }
    }
  ],
  "memory": "512",
  "cpu": "512"
}
```

2. Jalankan perintah berikut untuk mendaftarkan definisi tugas.

```
aws ecs register-task-definition --cli-input-json file://fluent-bit.json --
region region
```

Anda dapat membuat daftar definisi tugas untuk akun Anda dengan menjalankan `list-task-definitions` perintah. Output menampilkan nilai keluarga dan revisi yang dapat Anda gunakan bersama dengan `run-task` atau`start-task`.

Langkah 5: Jalankan definisi **`ecs-windows-fluent-bit`** tugas sebagai layanan Amazon ECS menggunakan strategi penjadwalan daemon

Setelah mendaftarkan definisi tugas untuk akun Anda, Anda dapat menjalankan tugas di klaster. Untuk tutorial ini, Anda menjalankan satu contoh definisi `ecs-windows-fluent-bit:1` tugas di `FluentBit-cluster` cluster Anda. Jalankan tugas dalam layanan yang menggunakan strategi penjadwalan daemon, yang memastikan bahwa satu instance Fluent Bit selalu berjalan pada setiap instance container Anda.

Untuk menjalankan tugas

1. Jalankan perintah berikut untuk memulai definisi `ecs-windows-fluent-bit:1` tugas (terdaftar pada langkah sebelumnya) sebagai layanan.

> **Note**
>
> Definisi tugas ini menggunakan driver `awslogs` logging, instance container Anda harus memiliki izin yang diperlukan.

Ganti variabel berikut:

- *`wilayah`* dengan Wilayah tempat layanan Anda berjalan

```
aws ecs create-service \
    --cluster FluentBit-cluster \
    --service-name FluentBitForwardDaemonService \
    --task-definition ecs-windows-fluent-bit:1 \
    --launch-type EC2 \
    --scheduling-strategy DAEMON \
    --region region
```

2. Jalankan perintah berikut untuk membuat daftar tugas Anda.

   Ganti variabel berikut:

   - *wilayah* dengan Wilayah tempat tugas layanan Anda berjalan

```
aws ecs list-tasks --cluster FluentBit-cluster --region region
```

Langkah 6: Daftarkan definisi tugas Windows yang menghasilkan log

Daftarkan definisi tugas yang menghasilkan log. Definisi tugas ini menyebarkan gambar kontainer Windows yang akan menulis angka tambahan untuk `stdout` setiap detik.

Definisi tugas menggunakan driver logging lancar yang terhubung ke port 24224 yang didengarkan oleh plug-in Fluent Bit. Agen Amazon ECS memberi label pada setiap wadah Amazon ECS dengan tag termasuk nama cluster, ARN tugas, nama keluarga definisi tugas, nomor revisi definisi tugas, dan nama wadah tugas. Label nilai kunci ini diteruskan ke Fluent Bit.

> **Note**
>
> Tugas ini menggunakan mode `default` jaringan. Namun, Anda juga dapat menggunakan mode `awsvpc` jaringan dengan tugas tersebut.

Untuk mendaftarkan ketentuan tugas

1. Buat file yang diberi nama `windows-app-task.json` dengan konten berikut ini.

```
{
```

```
  "family": "windows-app-task",
  "containerDefinitions": [
    {
      "name": "sample-container",
      "image": "mcr.microsoft.com/windows/servercore:ltsc2019",
      "cpu": 512,
      "memory": 512,
      "essential": true,
      "entryPoint": [
        "Powershell",
        "-Command"
      ],
      "command": [
        "$count=1;while(1) { Write-Host $count; sleep 1; $count=$count+1;}"
      ],
      "logConfiguration": {
        "logDriver": "fluentd",
        "options": {
          "fluentd-address": "localhost:24224",
          "tag": "{{ index .ContainerLabels \"com.amazonaws.ecs.task-definition-
family\" }}",
          "fluentd-async": "true",
          "labels": "com.amazonaws.ecs.cluster,com.amazonaws.ecs.container-
name,com.amazonaws.ecs.task-arn,com.amazonaws.ecs.task-definition-
family,com.amazonaws.ecs.task-definition-version"
        }
      }
    }
  ],
  "memory": "512",
  "cpu": "512"
}
```

2. Jalankan perintah berikut untuk mendaftarkan definisi tugas.

   Ganti variabel berikut:

   - *wilayah* dengan Wilayah tempat tugas Anda berjalan

```
aws ecs register-task-definition --cli-input-json file://windows-app-task.json --
region region
```

Anda dapat membuat daftar definisi tugas untuk akun Anda dengan menjalankan `list-task-definitions` perintah. Output menampilkan nilai keluarga dan revisi yang dapat Anda gunakan bersama dengan `run-task` atau`start-task`.

Langkah 7: Jalankan definisi **`windows-app-task`** tugas

Setelah Anda mendaftarkan definisi `windows-app-task` tugas, jalankan di `FluentBit-cluster` cluster Anda.

Untuk menjalankan tugas

1. Jalankan definisi `windows-app-task:1` tugas yang Anda daftarkan di langkah sebelumnya.

   Ganti variabel berikut:

   - *`wilayah`* dengan Wilayah tempat tugas Anda berjalan

   ```
   aws ecs run-task --cluster FluentBit-cluster --task-definition windows-app-task:1
    --count 2 --region region
   ```

2. Jalankan perintah berikut untuk membuat daftar tugas Anda.

   ```
   aws ecs list-tasks --cluster FluentBit-cluster
   ```

Langkah 8: Verifikasi log CloudWatch

Untuk memverifikasi penyiapan Fluent Bit Anda, periksa grup log berikut di CloudWatch konsol:

- `/ecs/fluent-bit-logs`- Ini adalah grup log yang sesuai dengan wadah daemon Fluent Bit yang berjalan pada instance container.
- `/aws/ecs/FluentBit-cluster.windows-app-task`- Ini adalah grup log yang sesuai dengan semua tugas yang diluncurkan untuk keluarga definisi `windows-app-task` tugas di dalam `FluentBit-cluster` cluster.

  `task-out.`*`FIRST_TASK_ID`*`.sample-container`- Aliran log ini berisi semua log yang dihasilkan oleh contoh pertama tugas dalam wadah tugas sample-container.

`task-out.SECOND_TASK_ID.sample-container`- Aliran log ini berisi semua log yang dihasilkan oleh instance kedua tugas dalam wadah tugas sample-container.

Aliran `task-out.TASK_ID.sample-container` log memiliki bidang yang mirip dengan berikut ini:

```
{
    "source": "stdout",
    "ecs_task_arn": "arn:aws:ecs:region:0123456789012:task/FluentBit-
cluster/13EXAMPLE",
    "container_name": "/ecs-windows-app-task-1-sample-container-cEXAMPLE",
    "ecs_cluster": "FluentBit-cluster",
    "ecs_container_name": "sample-container",
    "ecs_task_definition_version": "1",
    "container_id": "61f5e6EXAMPLE",
    "log": "10",
    "ecs_task_definition_family": "windows-app-task"
}
```

Untuk memverifikasi pengaturan Fluent Bit

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, pilih Grup log. Pastikan Anda berada di Wilayah tempat Anda menerapkan Fluent Bit ke kontainer Anda.

   Dalam daftar grup log di Wilayah AWS, Anda akan melihat yang berikut:

   - `/ecs/fluent-bit-logs`
   - `/aws/ecs/FluentBit-cluster.windows-app-task`

   Jika Anda melihat grup log ini, pengaturan Bit Lancar diverifikasi.

Langkah 9: Membersihkan

Setelah Anda menyelesaikan tutorial ini, bersihkan sumber daya yang terkait dengannya untuk menghindari biaya yang tidak Anda gunakan.

Untuk membersihkan sumber daya tutorial

1. Hentikan `windows-simple-task` tugas dan `ecs-fluent-bit` tugas. Untuk informasi selengkapnya, lihat the section called “Hentikan tugas Amazon ECS”.
2. Jalankan perintah berikut untuk menghapus grup `/ecs/fluent-bit-logs` log. Untuk informasi selengkapnya, tentang menghapus grup log lihat delete-log-groupdi AWS Command Line Interface Referensi.

```
aws logs delete-log-group --log-group-name /ecs/fluent-bit-logs
aws logs delete-log-group --log-group-name /aws/ecs/FluentBit-cluster.windows-app-task
```

3. Jalankan perintah berikut untuk mengakhiri instance.

```
aws ec2 terminate-instances --instance-ids instance-id
```

4. Jalankan perintah berikut untuk menghapus peran IAM.

```
aws iam delete-role --role-name ecsInstanceRole
aws iam delete-role --role-name fluentTaskRole
```

5. Jalankan perintah berikut untuk menghapus cluster Amazon ECS.

```
aws ecs delete-cluster --cluster FluentBit-cluster
```

Menggunakan GMSAs untuk Windows Container di Amazon EC2

Amazon ECS mendukung otentikasi Active Directory untuk wadah Windows melalui jenis akun layanan khusus yang disebut grup Akun Layanan Terkelola (GMSA).

Aplikasi jaringan berbasis Windows seperti aplikasi NET sering menggunakan Direktori Aktif untuk memfasilitasi autentikasi dan otorisasi manajemen antara pengguna dan layanan. Developer umumnya merancang aplikasi mereka untuk mengintegrasikan dengan Direktori Aktif dan berjalan pada server yang bergabung dengan domain untuk tujuan ini. Karena kontainer Windows tidak dapat bergabung dengan domain, Anda harus mengonfigurasi kontainer Windows untuk berjalan dengan gMSA.

Wadah Windows yang berjalan dengan GMSA bergantung pada instans Amazon EC2 hostnya untuk mengambil kredensyal GMSA dari pengontrol domain Active Directory dan menyediakannya ke instance container. Untuk informasi selengkapnya, lihat Membuat gMSA untuk kontainer Windows.

> **Note**
>
> Fitur ini tidak didukung pada wadah Windows di Fargate.

Topik

Pertimbangan

Hal berikut ini harus dipertimbangkan ketika menggunakan gMSA untuk kontainer Windows:

- Saat menggunakan AMI Penuh Windows Server 2016 yang dioptimalkan Amazon ECS untuk instance container Anda, nama host container harus sama dengan nama akun GMSA yang ditentukan dalam file spesifikasi kredensyal. Untuk menentukan hostname untuk kontainer, gunakan parameter ketentuan kontainer `hostname`. Untuk informasi selengkapnya, lihat Pengaturan jaringan.
- Anda memilih antara domainless gMSA dan menggabungkan setiap instance ke satu domain. Dengan menggunakan domainlessgMSA, instance container tidak digabungkan ke domain, aplikasi lain pada instance tidak dapat menggunakan kredensyal untuk mengakses domain, dan tugas yang menggabungkan domain yang berbeda dapat berjalan pada instance yang sama.

  Kemudian, pilih penyimpanan data untuk CredSpec dan opsional, untuk kredensyal pengguna Active Directory untuk domainless. gMSA

  Amazon ECS menggunakan file spesifikasi kredensi Direktori Aktif ()CredSpec. File ini berisi gMSA metadata yang digunakan untuk menyebarkan konteks gMSA akun ke wadah. Anda menghasilkan CredSpec file dan kemudian menyimpannya di salah satu opsi CredSpec penyimpanan dalam tabel berikut, khusus untuk Sistem Operasi instance kontainer. Untuk menggunakan metode domainless, bagian opsional dalam CredSpec file dapat menentukan kredensyal di salah satu

opsi domainless user credentialspenyimpanan dalam tabel berikut, khusus untuk Sistem Operasi instance kontainer.

gMSAopsi penyimpanan data oleh Sistem Operasi

Lokasi penyimpanan	Linux	Windows
Amazon Simple Storage Service	CredSpec	CredSpec
AWS Secrets Manager	kredensil pengguna tanpa domain	kredensil pengguna tanpa domain
Toko Parameter Amazon EC2 Systems Manager	CredSpec	CredSpec, kredensyal pengguna tanpa domain
Berkas lokal	N/A	CredSpec

Prasyarat

Sebelum Anda menggunakan fitur gMSA untuk Windows container dengan Amazon ECS, pastikan untuk menyelesaikan yang berikut ini:

- Anda menyiapkan domain Direktori Aktif dengan sumber daya yang ingin diakses oleh kontainer Anda. Amazon ECS mendukung pengaturan berikut:
  - Direktori AWS Directory Service Aktif. AWS Directory Service adalah Direktori Aktif AWS terkelola yang di-host di Amazon EC2. Untuk informasi selengkapnya, lihat Memulai dengan Microsoft AD yang AWS Dikelola di Panduan AWS Directory Service Administrasi.
  - Direktori Aktif lokal. Anda harus memastikan bahwa instans penampung Amazon ECS Linux dapat bergabung dengan domain. Untuk informasi selengkapnya, lihat AWS Direct Connect.
- Anda memiliki gMSA akun yang ada di Direktori Aktif. Untuk informasi selengkapnya, lihat Membuat gMSA untuk kontainer Windows.
- Anda memilih untuk menggunakan domainless gMSA atau instans penampung Amazon ECS Windows yang menghosting tugas Amazon ECS harus domain yang digabungkan ke Direktori Aktif dan menjadi anggota grup keamanan Direktori Aktif yang memiliki akses ke akun GMSA.

Dengan menggunakan domainlessgMSA, instance container tidak digabungkan ke domain, aplikasi lain pada instance tidak dapat menggunakan kredensyal untuk mengakses domain, dan tugas yang menggabungkan domain yang berbeda dapat berjalan pada instance yang sama.

- Anda menambahkan izin IAM yang diperlukan. Izin yang diperlukan bergantung pada metode yang Anda pilih untuk kredensyal awal dan untuk menyimpan spesifikasi kredensialnya:
  - Jika Anda menggunakan domainless gMSA untuk kredensyal awal, izin IAM untuk diperlukan AWS Secrets Manager pada peran instans Amazon EC2.
  - Jika Anda menyimpan spesifikasi kredensyal di SSM Parameter Store, izin IAM untuk Amazon EC2 Systems Manager Parameter Store diperlukan pada peran eksekusi tugas.
  - Jika Anda menyimpan spesifikasi kredensyal di Amazon S3, izin IAM untuk Amazon Simple Storage Service diperlukan pada peran eksekusi tugas.

Menyiapkan gMSA untuk Windows Container di Amazon ECS

Untuk mengatur gMSA untuk Windows Containers di Amazon ECS, Anda dapat mengikuti tutorial lengkap yang mencakup konfigurasi prasyarat. Menggunakan Windows Containers dengan Domainless gMSA menggunakan AWS CLI

Bagian berikut mencakup CredSpec konfigurasi secara rinci.

Topik

- Contoh CredSpec
- Pengaturan tanpa domain gMSA
- Referensi File Spek Kredensial dalam Definisi Tugas

Contoh CredSpec

Amazon ECS menggunakan file spesifikasi kredensyal yang berisi gMSA metadata yang digunakan untuk menyebarkan konteks gMSA akun ke wadah Windows. Anda dapat menghasilkan file spesifikasi kredensial dan referensi dalam bidang `credentialSpec` pada ketentuan tugas Anda. File spek kredensial tidak berisi rahasia apa pun.

Berikut ini contoh file spek kredensial:

```
{
```

```
    "CmsPlugins": [
        "ActiveDirectory"
    ],
    "DomainJoinConfig": {
        "Sid": "S-1-5-21-2554468230-2647958158-2204241789",
        "MachineAccountName": "WebApp01",
        "Guid": "8665abd4-e947-4dd0-9a51-f8254943c90b",
        "DnsTreeName": "contoso.com",
        "DnsName": "contoso.com",
        "NetBiosName": "contoso"
    },
    "ActiveDirectoryConfig": {
        "GroupManagedServiceAccounts": [
            {
                "Name": "WebApp01",
                "Scope": "contoso.com"
            }
        ]
    }
}
```

Pengaturan tanpa domain gMSA

Kami merekomendasikan domainless gMSA daripada menggabungkan instance container ke satu domain. Dengan menggunakan domainlessgMSA, instance container tidak digabungkan ke domain, aplikasi lain pada instance tidak dapat menggunakan kredensyal untuk mengakses domain, dan tugas yang menggabungkan domain yang berbeda dapat berjalan pada instance yang sama.

1. Sebelum mengunggah CredSpec ke salah satu opsi penyimpanan, tambahkan informasi ke ARN rahasia di Secrets Manager atau SSM Parameter Store. CredSpec Untuk informasi selengkapnya, lihat Konfigurasi spesifikasi kredensyal tambahan untuk kasus penggunaan host non-domain-joined kontainer di situs web Microsoft Learn.

   Format kredensi tanpa domain gMSA

   Berikut ini adalah format JSON untuk gMSA kredensyal domainless untuk Active Directory Anda. Simpan kredensialnya di Secrets Manager atau SSM Parameter Store.

   ```
   {
       "username":"WebApp01",
       "password":"Test123!",
       "domainName":"contoso.com"
   ```

```
}
```

2. Tambahkan informasi berikut ke CredSpec file di dalam file`ActiveDirectoryConfig`. Ganti ARN dengan rahasia di Secrets Manager atau SSM Parameter Store.

   Perhatikan bahwa `PluginGUID` nilai harus cocok dengan GUID dalam contoh cuplikan berikut dan diperlukan.

```
    "HostAccountConfig": {
        "PortableCcgVersion": "1",
        "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
        "PluginInput": "{\"credentialArn\": \"arn:aws:secretsmanager:aws-
region:111122223333:secret:gmsa-plugin-input\"}"
    }
```

   Anda juga dapat menggunakan rahasia di SSM Parameter Store dengan menggunakan ARN dalam format ini:. `\"arn:aws:ssm:aws-region:111122223333:parameter/gmsa-plugin-input\"`

3. Setelah Anda memodifikasi CredSpec file, itu akan terlihat seperti contoh berikut:

```
{
  "CmsPlugins": [
    "ActiveDirectory"
  ],
  "DomainJoinConfig": {
    "Sid": "S-1-5-21-4066351383-705263209-1606769140",
    "MachineAccountName": "WebApp01",
    "Guid": "ac822f13-583e-49f7-aa7b-284f9a8c97b6",
    "DnsTreeName": "contoso",
    "DnsName": "contoso",
    "NetBiosName": "contoso"
  },
  "ActiveDirectoryConfig": {
    "GroupManagedServiceAccounts": [
      {
        "Name": "WebApp01",
        "Scope": "contoso"
      },
      {
        "Name": "WebApp01",
        "Scope": "contoso"
```

```
      }
    ],
    "HostAccountConfig": {
      "PortableCcgVersion": "1",
      "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
      "PluginInput": "{\"credentialArn\": \"arn:aws:secretsmanager:aws-
region:111122223333:secret:gmsa-plugin-input\"}"
    }
  }
}
```

Referensi File Spek Kredensial dalam Definisi Tugas

Amazon ECS mendukung cara-cara berikut untuk mereferensikan jalur file di `credentialSpecs` bidang definisi tugas. Untuk masing-masing opsi ini, Anda dapat memberikan `credentialspec:` ataudomainlesscredentialspec:, tergantung pada apakah Anda menggabungkan instance penampung ke satu domain, atau menggunakan domainlessgMSA, masing-masing.

Bucket Amazon S3

Tambahkan spesifikasi kredensyal ke bucket Amazon S3, lalu rujuk Nama Sumber Daya Amazon (ARN) bucket Amazon S3 di bidang definisi tugas. `credentialSpecs`

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [
                "credentialspecdomainless:arn:aws:s3:::${BucketName}/${ObjectName}"
            ],
            ...
        }
    ],
    ...
}
```

Anda juga harus menambahkan izin berikut sebagai kebijakan inline ke peran IAM eksekusi tugas Amazon ECS untuk memberikan akses tugas Anda ke bucket Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::{bucket_name}",
                "arn:aws:s3:::{bucket_name}/{object}"
            ]
        }
    ]
}
```

Parameter SSM Menyimpan parameter

Tambahkan spesifikasi kredensyal ke parameter Penyimpanan Parameter SSM, lalu rujuk Nama Sumber Daya Amazon (ARN) parameter Penyimpanan Parameter SSM di bidang definisi tugas. `credentialSpecs`

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [

 "credentialspecdomainless:arn:aws:ssm:region:111122223333:parameter/parameter_name"
            ],
            ...
        }
    ],
    ...
}
```

Anda juga harus menambahkan izin berikut sebagai kebijakan inline ke peran IAM eksekusi tugas Amazon ECS untuk memberikan akses tugas Anda ke parameter Penyimpanan Parameter SSM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameters"
            ],
            "Resource": [
                "arn:aws:ssm:region:111122223333:parameter/parameter_name"
            ]
        }
    ]
}
```

File Lokal

Dengan detail spesifikasi kredensial dalam file lokal, referensi jalur file di bidang ketentuan tugas `credentialSpecs`. Jalur file yang direferensikan harus relatif terhadap `C:\ProgramData\Docker\CredentialSpecs` direktori dan menggunakan garis miring terbalik ('\') sebagai pemisah jalur file.

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [
                "credentialspec:file://CredentialSpecDir\CredentialSpecFile.json"
            ],
            ...
        }
    ],
    ...
}
```

Menggunakan Windows Containers dengan Domainless gMSA menggunakan AWS CLI

Tutorial berikut menunjukkan cara membuat tugas Amazon ECS yang menjalankan wadah Windows yang memiliki kredensyal untuk mengakses Active Directory dengan file. AWS CLI Dengan menggunakan domainlessgMSA, instance container tidak digabungkan ke domain, aplikasi lain pada instance tidak dapat menggunakan kredensyal untuk mengakses domain, dan tugas yang menggabungkan domain yang berbeda dapat berjalan pada instance yang sama.

Topik

Prasyarat

Jika mengikuti tutorial ini, berarti prasyarat berikut telah selesai:

- Langkah-langkah di Siapkan untuk menggunakan Amazon ECS telah selesai.

- AWS Pengguna Anda memiliki izin yang diperlukan yang ditentukan dalam contoh kebijakan Amazonecs_ FullAccess IAM.
- Versi terbaru diinstal dan dikonfigurasi. AWS CLI Untuk informasi selengkapnya tentang menginstal atau memutakhirkan AWS CLI, lihat Menginstal. AWS Command Line Interface
- Anda menyiapkan domain Direktori Aktif dengan sumber daya yang ingin diakses oleh kontainer Anda. Amazon ECS mendukung pengaturan berikut:
  - Direktori AWS Directory Service Aktif. AWS Directory Service adalah Direktori Aktif AWS terkelola yang di-host di Amazon EC2. Untuk informasi selengkapnya, lihat Memulai dengan Microsoft AD yang AWS Dikelola di Panduan AWS Directory Service Administrasi.
  - Direktori Aktif lokal. Anda harus memastikan bahwa instans penampung Amazon ECS Linux dapat bergabung dengan domain. Untuk informasi selengkapnya, lihat AWS Direct Connect.
- Anda memiliki VPC dan subnet yang dapat menyelesaikan nama domain Active Directory.
- Anda memilih antara domainless gMSA dan menggabungkan setiap instance ke satu domain. Dengan menggunakan domainlessgMSA, instance container tidak digabungkan ke domain, aplikasi lain pada instance tidak dapat menggunakan kredensyal untuk mengakses domain, dan tugas yang menggabungkan domain yang berbeda dapat berjalan pada instance yang sama.

  Kemudian, pilih penyimpanan data untuk CredSpec dan opsional, untuk kredensyal pengguna Active Directory untuk domainless. gMSA

  Amazon ECS menggunakan file spesifikasi kredensi Direktori Aktif ()CredSpec. File ini berisi gMSA metadata yang digunakan untuk menyebarkan konteks gMSA akun ke wadah. Anda menghasilkan CredSpec file dan kemudian menyimpannya di salah satu opsi CredSpec penyimpanan dalam tabel berikut, khusus untuk Sistem Operasi instance kontainer. Untuk menggunakan metode domainless, bagian opsional dalam CredSpec file dapat menentukan kredensyal di salah satu opsi domainless user credentialspenyimpanan dalam tabel berikut, khusus untuk Sistem Operasi instance kontainer.

  gMSAopsi penyimpanan data oleh Sistem Operasi

Lokasi penyimpanan	Linux	Windows
Amazon Simple Storage Service	CredSpec	CredSpec
AWS Secrets Manager	kredensil pengguna tanpa domain	kredensil pengguna tanpa domain

Lokasi penyimpanan	Linux	Windows
Toko Parameter Amazon EC2 Systems Manager	CredSpec	CredSpec, kredensyal pengguna tanpa domain
Berkas lokal	N/A	CredSpec

- (Opsional) AWS CloudShell adalah alat yang memberi pelanggan baris perintah tanpa perlu membuat instance EC2 mereka sendiri. Untuk informasi lebih lanjut, lihat Apa itu AWS CloudShell? dalam AWS CloudShell User Guide.

Langkah 1: Buat dan konfigurasikan gMSA akun di Layanan Domain Direktori Aktif (AD DS)

Buat dan konfigurasikan gMSA akun di domain Active Directory.

1. Menghasilkan kunci root Layanan Distribusi Kunci

   > **Note**
   >
   > Jika Anda menggunakan AWS Directory Service, maka Anda dapat melewati langkah ini.

   Kunci root dan gMSA izin KDS dikonfigurasi dengan AWS Microsoft AD terkelola Anda.

   Jika Anda belum membuat Akun gMSA Layanan di domain Anda, Anda harus terlebih dahulu membuat kunci root Key Distribution Service (KDS). KDS bertanggung jawab untuk membuat, memutar, dan melepaskan gMSA kata sandi ke host resmi. Ketika `ccg.exe` kebutuhan untuk mengambil gMSA kredensyal, itu menghubungi KDS untuk mengambil kata sandi saat ini.

   Untuk memeriksa apakah kunci root KDS telah dibuat, jalankan PowerShell cmdlet berikut dengan hak admin domain pada pengontrol domain menggunakan modul. `ActiveDirectory` PowerShell Untuk informasi selengkapnya tentang modul, lihat ActiveDirectory Modul di situs web Microsoft Learn.

   ```
   PS C:\> Get-KdsRootKey
   ```

Jika perintah mengembalikan ID kunci, Anda dapat melewati sisa langkah ini. Jika tidak, buat kunci root KDS dengan menjalankan perintah berikut:

```
PS C:\> Add-KdsRootKey -EffectiveImmediately
```

Meskipun argumen `EffectiveImmediately` untuk perintah menyiratkan kunci efektif segera, Anda harus menunggu 10 jam sebelum kunci root KDS direplikasi dan tersedia untuk digunakan pada semua pengontrol domain.

2. Buat gMSA akun

   Untuk membuat gMSA akun dan memungkinkan `ccg.exe` untuk mengambil gMSA kata sandi, jalankan PowerShell perintah berikut dari Windows Server atau klien dengan akses ke domain. Ganti `ExampleAccount` dengan nama yang Anda inginkan untuk gMSA akun Anda.

   a.
   ```
   PS C:\> Install-WindowsFeature RSAT-AD-PowerShell
   ```

   b.
   ```
   PS C:\> New-ADGroup -Name "ExampleAccount Authorized Hosts" -SamAccountName
    "ExampleAccountHosts" -GroupScope DomainLocal
   ```

   c.
   ```
   PS C:\> New-ADServiceAccount -Name "ExampleAccount" -DnsHostName
    "contoso" -ServicePrincipalNames "host/ExampleAccount", "host/contoso" -
   PrincipalsAllowedToRetrieveManagedPassword "ExampleAccountHosts"
   ```

   d. Buat pengguna dengan kata sandi permanen yang tidak kedaluwarsa. Kredensyal ini disimpan AWS Secrets Manager dan digunakan oleh setiap tugas untuk bergabung dengan domain.

   ```
   PS C:\> New-ADUser -Name "ExampleAccount" -AccountPassword (ConvertTo-
   SecureString -AsPlainText "Test123" -Force) -Enabled 1 -PasswordNeverExpires 1
   ```

   e.
   ```
   PS C:\> Add-ADGroupMember -Identity "ExampleAccountHosts" -Members
    "ExampleAccount"
   ```

   f. Instal PowerShell modul untuk membuat CredSpec objek di Active Directory dan output CredSpec JSON.

   ```
   PS C:\> Install-PackageProvider -Name NuGet -Force
   ```

```
PS C:\> Install-Module CredentialSpec
```

g.

```
PS C:\> New-CredentialSpec -AccountName ExampleAccount
```

3. Salin output JSON dari perintah sebelumnya ke dalam file bernamagmsa-cred-spec.json. Ini adalah CredSpec filenya. Ini digunakan pada Langkah 3,Langkah 3: Ubah CredSpec JSON Anda untuk menyertakan informasi tanpa domain gMSA.

Langkah 2: Unggah Kredensil ke Secrets Manager

Salin kredensyal Active Directory ke dalam sistem penyimpanan kredensyal yang aman, sehingga setiap tugas mengambilnya. Ini adalah metode domainlessgMSA. Dengan menggunakan domainlessgMSA, instance container tidak digabungkan ke domain, aplikasi lain pada instance tidak dapat menggunakan kredensyal untuk mengakses domain, dan tugas yang menggabungkan domain yang berbeda dapat berjalan pada instance yang sama.

Langkah ini menggunakan AWS CLI. Anda dapat menjalankan perintah ini AWS CloudShell di shell default, yaitubash.

- Jalankan AWS CLI perintah berikut dan ganti nama pengguna, kata sandi, dan nama domain agar sesuai dengan lingkungan Anda. Simpan ARN rahasia untuk digunakan pada langkah berikutnya, Langkah 3: Ubah CredSpec JSON Anda untuk menyertakan informasi tanpa domain gMSA

  Perintah berikut menggunakan karakter kelanjutan garis miring terbalik yang digunakan oleh sh dan shell yang kompatibel. Perintah ini tidak kompatibel dengan PowerShell. Anda harus memodifikasi perintah untuk menggunakannya PowerShell.

```
$ aws secretsmanager create-secret \
--name gmsa-plugin-input \
--description "Amazon ECS - gMSA Portable Identity." \
--secret-string "{\"username\":\"ExampleAccount\",\"password\":\"Test123\",
\"domainName\":\"contoso.com\"}"
```

Langkah 3: Ubah CredSpec JSON Anda untuk menyertakan informasi tanpa domain gMSA

Sebelum mengunggah CredSpec ke salah satu opsi penyimpanan, tambahkan informasi ke CredSpec dengan ARN rahasia di Secrets Manager dari langkah sebelumnya. Untuk informasi selengkapnya, lihat Konfigurasi spesifikasi kredensyal tambahan untuk kasus penggunaan host non-domain-joined kontainer di situs web Microsoft Learn.

1. Tambahkan informasi berikut ke CredSpec file di dalam file`ActiveDirectoryConfig`. Ganti ARN dengan rahasia di Secrets Manager dari langkah sebelumnya.

   Perhatikan bahwa `PluginGUID` nilai harus cocok dengan GUID dalam contoh cuplikan berikut dan diperlukan.

   ```
   "HostAccountConfig": {
         "PortableCcgVersion": "1",
         "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
         "PluginInput": "{\"credentialArn\": \"arn:aws:secretsmanager:aws-
   region:111122223333:secret:gmsa-plugin-input\"}"
       }
   ```

   Anda juga dapat menggunakan rahasia di SSM Parameter Store dengan menggunakan ARN dalam format ini:. `\"arn:aws:ssm:aws-region:111122223333:parameter/gmsa-plugin-input\"`

2. Setelah Anda memodifikasi CredSpec file, itu akan terlihat seperti contoh berikut:

   ```
   {
     "CmsPlugins": [
       "ActiveDirectory"
     ],
     "DomainJoinConfig": {
       "Sid": "S-1-5-21-4066351383-705263209-1606769140",
       "MachineAccountName": "ExampleAccount",
       "Guid": "ac822f13-583e-49f7-aa7b-284f9a8c97b6",
       "DnsTreeName": "contoso",
       "DnsName": "contoso",
       "NetBiosName": "contoso"
     },
     "ActiveDirectoryConfig": {
       "GroupManagedServiceAccounts": [
   ```

```
      {
        "Name": "ExampleAccount",
        "Scope": "contoso"
      },
      {
        "Name": "ExampleAccount",
        "Scope": "contoso"
      }
    ],
    "HostAccountConfig": {
      "PortableCcgVersion": "1",
      "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
      "PluginInput": "{\"credentialArn\": \"arn:aws:secretsmanager:aws-
region:111122223333:secret:gmsa-plugin-input\"}"
    }
  }
}
```

Langkah 4: Unggah CredSpec ke Amazon S3

Langkah ini menggunakan AWS CLI. Anda dapat menjalankan perintah ini AWS CloudShell di shell default, yaitu`bash`.

1. Salin CredSpec file ke komputer atau lingkungan tempat Anda menjalankan AWS CLI perintah.
2. Jalankan AWS CLI perintah berikut untuk mengunggah CredSpec ke Amazon S3. Ganti `MyBucket` dengan nama bucket Amazon S3 Anda. Anda dapat menyimpan file sebagai objek di bucket dan lokasi mana pun, tetapi Anda harus mengizinkan akses ke bucket dan lokasi tersebut dalam kebijakan yang Anda lampirkan ke peran eksekusi tugas.

   Perintah berikut menggunakan karakter kelanjutan garis miring terbalik yang digunakan oleh `sh` dan shell yang kompatibel. Perintah ini tidak kompatibel dengan PowerShell. Anda harus memodifikasi perintah untuk menggunakannya PowerShell.

   ```
   $ aws s3 cp gmsa-cred-spec.json \
   s3://MyBucket/ecs-domainless-gmsa-credspec
   ```

Langkah 5: (Opsional) Buat cluster Amazon ECS

Secara default, akun Anda memiliki klaster Amazon ECS bernama`default`. Cluster ini digunakan secara default di AWS CLI, SDK, dan AWS CloudFormation file. Anda dapat menggunakan klaster tambahan untuk mengelompokkan dan mengatur tugas dan infrastruktur, dan menetapkan default untuk beberapa konfigurasi.

Anda dapat membuat cluster dari AWS Management Console, AWS CLI, SDK, atau AWS CloudFormation. Pengaturan dan konfigurasi di cluster tidak mempengaruhigMSA.

Langkah ini menggunakan AWS CLI. Anda dapat menjalankan perintah ini AWS CloudShell di shell default, yaitu`bash`.

```
$ aws ecs create-cluster --cluster-name windows-domainless-gmsa-cluster
```

> **Important**
>
> Jika Anda memilih untuk membuat cluster Anda sendiri, Anda harus menentukan `--cluster clusterName` untuk setiap perintah yang ingin Anda gunakan dengan cluster itu.

Langkah 6: Buat peran IAM untuk instance kontainer

Instance container adalah komputer host untuk menjalankan container dalam tugas ECS, misalnya instans Amazon EC2. Setiap instans kontainer mendaftar ke cluster Amazon ECS. Sebelum meluncurkan instans Amazon EC2 dan mendaftarkannya ke klaster, Anda harus membuat peran IAM agar instans penampung dapat digunakan.

Untuk membuat peran instance container, lihatPeran IAM instans wadah Amazon ECS. Default `ecsInstanceRole` memiliki izin yang cukup untuk menyelesaikan tutorial ini.

Langkah 7: Buat peran eksekusi tugas khusus

Amazon ECS dapat menggunakan peran IAM yang berbeda untuk izin yang diperlukan untuk memulai setiap tugas, bukan peran instance container. Peran ini adalah peran eksekusi tugas. Sebaiknya buat peran eksekusi tugas hanya dengan izin yang diperlukan untuk ECS untuk menjalankan tugas, juga dikenal sebagai izin hak istimewa paling sedikit. Untuk informasi lebih lanjut

tentang prinsip hak istimewa terkecil, lihat SEC03-BP02 Memberikan akses hak istimewa terkecil dalam Kerangka Kerja Well-Architected.AWS

1. Untuk membuat peran eksekusi tugas, lihatMembuat peran eksekusi tugas (ecsTaskExecutionRole). Izin default memungkinkan instance container untuk menarik gambar kontainer dari Amazon Elastic Container Registry `stdout` dan dan `stderr` dari aplikasi Anda untuk dicatat ke Amazon CloudWatch Logs.

   Karena peran membutuhkan izin khusus untuk tutorial ini, Anda dapat memberikan peran nama yang berbeda dari`ecsTaskExecutionRole`. Tutorial ini digunakan `ecsTaskExecutionRole` dalam langkah-langkah selanjutnya.

2. Tambahkan izin berikut dengan membuat kebijakan khusus, baik kebijakan sebaris yang hanya ada untuk peran ini, atau kebijakan yang dapat Anda gunakan kembali. Ganti ARN untuk pernyataan pertama dengan bucket dan lokasi Amazon S3, dan yang kedua `Resource` dengan ARN rahasia di Secrets Manager. `Resource`

   Jika Anda mengenkripsi rahasia di Secrets Manager dengan kunci khusus, Anda juga harus mengizinkan `kms:Decrypt` kunci tersebut.

   Jika Anda menggunakan SSM Parameter Store bukan Secrets Manager, Anda harus mengizinkan `ssm:GetParameter` parameter, bukan. `secretsmanager:GetSecretValue`

```
{
"Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::MyBucket/ecs-domainless-gmsa-credspec/gmsa-cred-
spec.json"
    },
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": "arn:aws:secretsmanager:aws-region:111122223333:secret:gmsa-
plugin-input"
    }
```

```
  ]
}
```

Langkah 8: Buat peran tugas untuk Amazon ECS Exec

Tutorial ini menggunakan Amazon ECS Exec untuk memverifikasi fungsionalitas dengan menjalankan perintah di dalam tugas yang sedang berjalan. Untuk menggunakan ECS Exec, layanan atau tugas harus mengaktifkan ECS Exec dan peran tugas (tetapi bukan peran eksekusi tugas) harus memiliki izin. `ssmmessages` Untuk kebijakan IAM yang diperlukan, lihatIzin IAM diperlukan untuk ECS Exec.

Langkah ini menggunakan AWS CLI. Anda dapat menjalankan perintah ini AWS CloudShell di shell default, yaitu`bash`.

Untuk membuat peran tugas menggunakan AWS CLI, ikuti langkah-langkah ini.

1. Buat file yang disebut `ecs-tasks-trust-policy.json` dengan konten berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

2. Buat peran IAM. Anda dapat mengganti nama `ecs-exec-demo-task-role` tetapi tetap nama untuk langkah-langkah berikut.

   Perintah berikut menggunakan karakter kelanjutan garis miring terbalik yang digunakan oleh `sh` dan shell yang kompatibel. Perintah ini tidak kompatibel dengan PowerShell. Anda harus memodifikasi perintah untuk menggunakannya PowerShell.

```
$ aws iam create-role --role-name ecs-exec-demo-task-role \
--assume-role-policy-document file://ecs-tasks-trust-policy.json
```

Anda dapat menghapus file tersebut`ecs-tasks-trust-policy.json`.

3. Buat file yang disebut `ecs-exec-demo-task-role-policy.json` dengan konten berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateControlChannel",
                "ssmmessages:CreateDataChannel",
                "ssmmessages:OpenControlChannel",
                "ssmmessages:OpenDataChannel"
            ],
            "Resource": "*"
        }
    ]
}
```

4. Buat kebijakan IAM dan lampirkan ke peran dari langkah sebelumnya.

   Perintah berikut menggunakan karakter kelanjutan garis miring terbalik yang digunakan oleh `sh` dan shell yang kompatibel. Perintah ini tidak kompatibel dengan PowerShell. Anda harus memodifikasi perintah untuk menggunakannya PowerShell.

```
$ aws iam put-role-policy \
    --role-name ecs-exec-demo-task-role \
    --policy-name ecs-exec-demo-task-role-policy \
    --policy-document file://ecs-exec-demo-task-role-policy.json
```

   Anda dapat menghapus file tersebut`ecs-exec-demo-task-role-policy.json`.

Langkah 9: Daftarkan definisi tugas yang menggunakan domainless gMSA

Langkah ini menggunakan AWS CLI. Anda dapat menjalankan perintah ini AWS CloudShell di shell default, yaitu`bash`.

1. Buat file yang disebut `windows-gmsa-domainless-task-def.json` dengan konten berikut:

```
{
  "family": "windows-gmsa-domainless-task",
  "containerDefinitions": [
    {
      "name": "windows_sample_app",
      "image": "mcr.microsoft.com/windows/servercore/iis",
      "cpu": 1024,
      "memory": 1024,
      "essential": true,
      "credentialSpecs": [
                "credentialspecdomainless:arn:aws:s3:::ecs-domainless-gmsa-
credspec/gmsa-cred-spec.json"
      ],
      "entryPoint": [
        "powershell",
        "-Command"
      ],
      "command": [
        "New-Item -Path C:\\inetpub\\wwwroot\\index.html -ItemType file -Value
 '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your
 application is now running on a container in Amazon ECS.</p>' -Force ; C:\
\ServiceMonitor.exe w3svc"
      ],
      "portMappings": [
        {
          "protocol": "tcp",
          "containerPort": 80,
          "hostPort": 8080
        }
      ]
    }
  ],
  "taskRoleArn": "arn:aws:iam::111122223333:role/ecs-exec-demo-task-role",
  "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole"
}
```

2. Daftarkan definisi tugas dengan menjalankan perintah berikut:

Perintah berikut menggunakan karakter kelanjutan garis miring terbalik yang digunakan oleh sh dan shell yang kompatibel. Perintah ini tidak kompatibel dengan PowerShell. Anda harus memodifikasi perintah untuk menggunakannya PowerShell.

```
$ aws ecs register-task-definition \
--cli-input-json file://windows-gmsa-domainless-task-def.json
```

Langkah 10: Daftarkan instance wadah Windows ke cluster

Luncurkan instans Windows Amazon EC2 dan jalankan agen kontainer ECS untuk mendaftarkannya sebagai instance container di cluster. ECS menjalankan tugas pada instance kontainer yang terdaftar ke cluster tempat tugas dimulai.

1. Untuk meluncurkan instans Windows Amazon EC2 yang dikonfigurasi untuk Amazon ECS di, lihat. AWS Management ConsoleMeluncurkan instans penampung Amazon ECS Windows Berhenti di langkah untuk data pengguna.
2. UntukgMSA, data pengguna harus mengatur variabel lingkungan `ECS_GMSA_SUPPORTED` sebelum memulai agen kontainer ECS.

   Untuk ECS Exec, agen harus memulai dengan argumen. `-EnableTaskIAMRole`

   Untuk mengamankan peran IAM instance dengan mencegah tugas mencapai layanan web EC2 IMDS untuk mengambil kredensyal peran, tambahkan argumen. `-AwsvpcBlockIMDS` Ini hanya berlaku untuk tugas yang menggunakan mode `awsvpc` jaringan.

   ```
   <powershell>
   [Environment]::SetEnvironmentVariable("ECS_GMSA_SUPPORTED", $TRUE, "Machine")
   Import-Module ECSTools
   Initialize-ECSAgent -Cluster windows-domainless-gmsa-cluster -EnableTaskIAMRole -
   AwsvpcBlockIMDS
   </powershell>
   ```

3. Tinjau ringkasan konfigurasi instans di panel Ringkasan, dan ketika Anda siap, pilih Luncurkan instans.

Langkah 11: Verifikasi contoh kontainer

Anda dapat memverifikasi bahwa ada instance kontainer di cluster menggunakan file AWS Management Console. Namun, gMSA perlu fitur tambahan yang ditunjukkan sebagai atribut. Atribut ini tidak terlihat di AWS Management Console, jadi tutorial ini menggunakan AWS CLI.

Langkah ini menggunakan AWS CLI. Anda dapat menjalankan perintah ini AWS CloudShell di shell default, yaitu`bash`.

1. Buat daftar instance kontainer di cluster. Instance kontainer memiliki ID yang berbeda dari ID instance EC2.

   ```
   $ aws ecs list-container-instances
   ```

   Output:

   ```
   {
       "containerInstanceArns": [
           "arn:aws:ecs:aws-region:111122223333:container-
   instance/default/MyContainerInstanceID"
       ]
   }
   ```

   Misalnya, `526bd5d0ced448a788768334e79010fd` adalah ID instance kontainer yang valid.
2. Gunakan ID instance container dari langkah sebelumnya untuk mendapatkan detail untuk instance container. Ganti `MyContainerInstanceID` dengan ID.

   Perintah berikut menggunakan karakter kelanjutan garis miring terbalik yang digunakan oleh `sh` dan shell yang kompatibel. Perintah ini tidak kompatibel dengan PowerShell. Anda harus memodifikasi perintah untuk menggunakannya PowerShell.

   ```
   $ aws ecs describe-container-instances \
        ----container-instances MyContainerInstanceID
   ```

   Perhatikan bahwa outputnya sangat panjang.
3. Verifikasi bahwa `attributes` daftar memiliki objek dengan kunci yang dipanggil `name` dan nilai`ecs.capability.gmsa-domainless`. Berikut ini adalah contoh objek.

   Output:

```
{
    "name": "ecs.capability.gmsa-domainless"
}
```

Langkah 12: Jalankan tugas Windows

Jalankan tugas Amazon ECS. Jika hanya ada 1 instance kontainer di cluster, Anda dapat menggunakan`run-task`. Jika ada banyak instance kontainer yang berbeda, mungkin lebih mudah untuk menggunakan `start-task` dan menentukan ID instance container untuk menjalankan tugas, daripada menambahkan batasan penempatan ke definisi tugas untuk mengontrol jenis instance kontainer untuk menjalankan tugas ini.

Langkah ini menggunakan AWS CLI. Anda dapat menjalankan perintah ini AWS CloudShell di shell default, yaitu`bash`.

1. Perintah berikut menggunakan karakter kelanjutan garis miring terbalik yang digunakan oleh `sh` dan shell yang kompatibel. Perintah ini tidak kompatibel dengan PowerShell. Anda harus memodifikasi perintah untuk menggunakannya PowerShell.

   ```
   aws ecs run-task --task-definition windows-gmsa-domainless-task \
       --enable-execute-command --cluster windows-domainless-gmsa-cluster
   ```

   Perhatikan ID tugas yang dikembalikan oleh perintah.
2. Jalankan perintah berikut untuk memverifikasi bahwa tugas telah dimulai. Perintah ini menunggu dan tidak mengembalikan shell prompt sampai tugas dimulai. Ganti `MyTaskID` dengan ID tugas dari langkah sebelumnya.

   ```
   $ aws ecs wait tasks-running --task MyTaskID
   ```

Langkah 13: Verifikasi wadah memiliki gMSA kredensyal

Verifikasi bahwa wadah dalam tugas memiliki Kerberos token. gMSA

Langkah ini menggunakan AWS CLI. Anda dapat menjalankan perintah ini AWS CloudShell di shell default, yaitu`bash`.

1. Perintah berikut menggunakan karakter kelanjutan garis miring terbalik yang digunakan oleh `sh` dan shell yang kompatibel. Perintah ini tidak kompatibel dengan PowerShell. Anda harus memodifikasi perintah untuk menggunakannya PowerShell.

   ```
   $ aws ecs execute-command \
   --task MyTaskID \
   --container windows_sample_app \
   --interactive \
   --command powershell.exe
   ```

   Outputnya akan menjadi PowerShell prompt.

2. Jalankan perintah berikut di PowerShell terminal di dalam wadah.

   ```
   PS C:\> klist get ExampleAccount$
   ```

   Dalam output, perhatikan `Principal` adalah salah satu yang Anda buat sebelumnya.

Langkah 14: Bersihkan

Setelah selesai dengan tutorial ini, Anda harus membersihkan sumber daya yang terkait untuk menghindari timbulnya biaya untuk sumber daya yang tidak terpakai.

Langkah ini menggunakan AWS CLI. Anda dapat menjalankan perintah ini AWS CloudShell di shell default, yaitu`bash`.

1. Hentikan tugas. Ganti `MyTaskID` dengan ID tugas dari langkah 12,Langkah 12: Jalankan tugas Windows.

   ```
   $ aws ecs stop-task --task MyTaskID
   ```

2. Mengakhiri instans Amazon EC2. Setelah itu, instance container di cluster akan dihapus secara otomatis setelah satu jam. Untuk informasi selengkapnya, lihat Manajemen kapasitas.

   Anda dapat menemukan dan menghentikan instance dengan menggunakan konsol Amazon EC2. Atau, Anda dapat menjalankan perintah berikut. Untuk menjalankan perintah, temukan ID instans EC2 di output `aws ecs describe-container-instances` perintah dari langkah 1,Langkah 11: Verifikasi contoh kontainer. i-10a64379 adalah contoh ID instans EC2.

```
$ aws ec2 terminate-instances --instance-ids MyInstanceID
```

3. Hapus CredSpec file di Amazon S3. Ganti `MyBucket` dengan nama bucket Amazon S3 Anda.

```
$ aws s3api delete-object --bucket MyBucket --key ecs-domainless-gmsa-credspec/
gmsa-cred-spec.json
```

4. Hapus rahasia dari Secrets Manager. Jika Anda menggunakan SSM Parameter Store sebagai gantinya, hapus parameternya.

   Perintah berikut menggunakan karakter kelanjutan garis miring terbalik yang digunakan oleh `sh` dan shell yang kompatibel. Perintah ini tidak kompatibel dengan PowerShell. Anda harus memodifikasi perintah untuk menggunakannya PowerShell.

```
$ aws secretsmanager delete-secret --secret-id gmsa-plugin-input \
    --force-delete-without-recovery
```

5. Deregister dan hapus definisi tugas. Dengan membatalkan pendaftaran definisi tugas, Anda menandainya sebagai tidak aktif sehingga tidak dapat digunakan untuk memulai tugas baru. Kemudian, Anda dapat menghapus definisi tugas.

   a. Deregister definisi tugas dengan menentukan versinya. ECS secara otomatis membuat versi definisi tugas, yang diberi nomor mulai dari 1. Anda merujuk ke versi dalam format yang sama dengan label pada gambar kontainer, seperti `:1`.

```
$ aws ecs deregister-task-definition --task-definition windows-gmsa-domainless-
task:1
```

   b. Hapus definisi tugas.

```
$ aws ecs delete-task-definitions --task-definition windows-gmsa-domainless-
task:1
```

6. (Opsional) Hapus cluster ECS, jika Anda membuat cluster.

```
$ aws ecs delete-cluster --cluster windows-domainless-gmsa-cluster
```

Debugging Amazon ECS tanpa domain gMSA untuk wadah Windows

Status tugas Amazon ECS

ECS mencoba memulai tugas tepat sekali. Setiap tugas yang memiliki masalah dihentikan, dan disetel ke status`STOPPED`. Ada dua jenis masalah umum dengan tugas. Pertama, tugas yang tidak bisa dimulai. Kedua, tugas di mana aplikasi telah berhenti di dalam salah satu wadah. Di AWS Management Console, lihat bidang Alasan berhenti dari tugas karena alasan mengapa tugas itu dihentikan. Dalam AWS CLI, jelaskan tugas dan lihat`stoppedReason`. Untuk langkah-langkah di AWS Management Console dan AWS CLI, lihatMemeriksa tugas yang telah dihentikan untuk kesalahan.

Peristiwa-peristiwa Windows

Peristiwa Windows untuk gMSA dalam wadah dicatat dalam file `Microsoft-Windows-Containers-CCG` log dan dapat ditemukan di Penampil Acara di bagian Aplikasi dan Layanan di`Logs\Microsoft\Windows\Containers-CCG\Admin`. Untuk tips debugging lainnya, lihat Memecahkan masalah GMSAs untuk wadah Windows di situs web Microsoft Learn.

Plugin agen gMSA ECS

Logging untuk gMSA plugin untuk agen ECS pada instance kontainer Windows ada di direktori berikut,`C:/ProgramData/Amazon/gmsa-plugin/`. Lihat di log ini untuk melihat apakah kredensyal pengguna tanpa domain diunduh dari lokasi penyimpanan, seperti Secrets Manager, dan format kredensialnya telah dibaca dengan benar.

Menggunakan gMSA untuk Linux Wadah di Amazon EC2

Amazon ECS mendukung otentikasi Active Directory untuk container Linux di EC2 melalui jenis akun layanan khusus yang disebut grup Managed Service Account (). gMSA

LinuxAplikasi jaringan berbasis, seperti aplikasi .NET Core, dapat menggunakan Active Directory untuk memfasilitasi otentikasi dan manajemen otorisasi antara pengguna dan layanan. Anda dapat menggunakan fitur ini dengan merancang aplikasi yang terintegrasi dengan Active Directory dan berjalan di server yang bergabung dengan domain. Tetapi, karena Linux kontainer tidak dapat digabungkan dengan domain, Anda perlu mengonfigurasi Linux wadah untuk dijalankan. gMSA

LinuxContainer yang berjalan dengan gMSA mengandalkan `credentials-fetcher` daemon yang berjalan pada instans Amazon EC2 host container. Artinya, daemon mengambil gMSA kredensyal dari pengontrol domain Active Directory dan kemudian mentransfer kredensyal ini ke instance container. Untuk informasi selengkapnya tentang akun layanan, lihat Membuat gMSAs kontainer Windows di situs web Microsoft Learn.

Topik

Pertimbangan

Pertimbangkan hal berikut sebelum Anda gunakan gMSA untuk Linux wadah:

- Jika kontainer Anda berjalan di EC2, Anda dapat menggunakannya gMSA untuk Windows kontainer dan Linux kontainer. Untuk informasi tentang cara menggunakan gMSA untuk wadah Linux di Fargate, lihat. Menggunakan gMSA untuk Linux wadah di Fargate
- Anda mungkin memerlukan Windows komputer yang bergabung dengan domain untuk menyelesaikan prasyarat. Misalnya, Anda mungkin memerlukan Windows komputer yang bergabung dengan domain untuk membuat Active Directory denganPowerShell. gMSA PowerShell Alat Direktur RSAT Aktif hanya tersedia untukWindows. Untuk informasi selengkapnya, lihat Menginstal alat administrasi Direktori Aktif.
- Anda memilih antara domainless gMSA dan menggabungkan setiap instance ke satu domain. Dengan menggunakan domainlessgMSA, instance container tidak digabungkan ke domain, aplikasi lain pada instance tidak dapat menggunakan kredensyal untuk mengakses domain, dan tugas yang menggabungkan domain yang berbeda dapat berjalan pada instance yang sama.

  Kemudian, pilih penyimpanan data untuk CredSpec dan opsional, untuk kredensyal pengguna Active Directory untuk domainless. gMSA

  Amazon ECS menggunakan file spesifikasi kredensi Direktori Aktif ()CredSpec. File ini berisi gMSA metadata yang digunakan untuk menyebarkan konteks gMSA akun ke wadah. Anda menghasilkan CredSpec file dan kemudian menyimpannya di salah satu opsi CredSpec penyimpanan dalam tabel berikut, khusus untuk Sistem Operasi instance kontainer. Untuk menggunakan metode domainless, bagian opsional dalam CredSpec file dapat menentukan kredensyal di salah satu opsi domainless user credentialspenyimpanan dalam tabel berikut, khusus untuk Sistem Operasi instance kontainer.

gMSAopsi penyimpanan data oleh Sistem Operasi

Lokasi penyimpanan	Linux	Windows
Amazon Simple Storage Service	CredSpec	CredSpec
AWS Secrets Manager	kredensil pengguna tanpa domain	kredensil pengguna tanpa domain
Toko Parameter Amazon EC2 Systems Manager	CredSpec	CredSpec, kredensyal pengguna tanpa domain
Berkas lokal	N/A	CredSpec

Prasyarat

Sebelum Anda menggunakan fitur gMSA untuk wadah Linux dengan Amazon ECS, pastikan untuk menyelesaikan yang berikut ini:

- Anda menyiapkan domain Direktori Aktif dengan sumber daya yang ingin diakses oleh kontainer Anda. Amazon ECS mendukung pengaturan berikut:
  - Direktori AWS Directory Service Aktif. AWS Directory Service adalah Direktori Aktif AWS terkelola yang di-host di Amazon EC2. Untuk informasi selengkapnya, lihat Memulai dengan Microsoft AD yang AWS Dikelola di Panduan AWS Directory Service Administrasi.
  - Direktori Aktif lokal. Anda harus memastikan bahwa instans penampung Amazon ECS Linux dapat bergabung dengan domain. Untuk informasi selengkapnya, lihat AWS Direct Connect.
- Anda memiliki gMSA akun yang ada di Direktori Aktif. Untuk informasi selengkapnya, lihat Menggunakan gMSA untuk Linux Wadah di Amazon EC2.
- Anda menginstal dan menjalankan `credentials-fetcher` daemon pada instance container Amazon ECS Linux. Anda juga menambahkan kumpulan kredensyal awal ke `credentials-fetcher` daemon untuk mengautentikasi dengan Active Directory.

> **Note**
>
> `credentials-fetcher`Daemon hanya tersedia untuk Amazon Linux 2023 dan Fedora 37 dan yang lebih baru. Daemon tidak tersedia untuk Amazon Linux 2. Untuk informasi selengkapnya, lihat aws/credentials-fetcher di. GitHub

- Anda mengatur kredensyal untuk `credentials-fetcher` daemon untuk mengautentikasi dengan Active Directory. Kredensyal harus menjadi anggota grup keamanan Direktori Aktif yang memiliki akses ke akun. gMSA Ada beberapa opsi diPutuskan apakah Anda ingin menggabungkan instance ke domain, atau menggunakan gMSA domainless..
- Anda menambahkan izin IAM yang diperlukan. Izin yang diperlukan bergantung pada metode yang Anda pilih untuk kredensyal awal dan untuk menyimpan spesifikasi kredensialnya:
  - Jika Anda menggunakan domainless gMSA untuk kredensyal awal, izin IAM untuk AWS Secrets Manager diperlukan pada peran eksekusi tugas.
  - Jika Anda menyimpan spesifikasi kredensyal di SSM Parameter Store, izin IAM untuk Amazon EC2 Systems Manager Parameter Store diperlukan pada peran eksekusi tugas.
  - Jika Anda menyimpan spesifikasi kredensyal di Amazon S3, izin IAM untuk Amazon Simple Storage Service diperlukan pada peran eksekusi tugas.

Menyiapkan Linux Kontainer gMSA berkemampuan di Amazon ECS

Siapkan infrastruktur

Langkah-langkah berikut adalah pertimbangan dan pengaturan yang dilakukan sekali. Setelah menyelesaikan langkah-langkah ini, Anda dapat mengotomatiskan pembuatan instance kontainer untuk menggunakan kembali konfigurasi ini.

Putuskan bagaimana kredensyal awal disediakan dan konfigurasikan data pengguna EC2 dalam templat peluncuran EC2 yang dapat digunakan kembali untuk menginstal daemon. `credentials-fetcher`

1. Putuskan apakah Anda ingin menggabungkan instance ke domain, atau menggunakan gMSA domainless.
   - Bergabunglah dengan instans EC2 ke domain Active Directory

- Bergabunglah dengan instance berdasarkan data pengguna

  Tambahkan langkah-langkah untuk bergabung dengan domain Active Directory ke data pengguna EC2 Anda dalam template peluncuran EC2. Beberapa grup Auto Scaling Amazon EC2 dapat menggunakan templat peluncuran yang sama.

  Anda dapat menggunakan langkah-langkah ini Bergabung dengan Active Directory atau FreeIPA domain di Fedora Docs.

- Jadikan pengguna Active Directory untuk domainless gMSA

  `credentials-fetcher`Daemon memiliki fitur yang disebut domainless. gMSA Fitur ini memerlukan domain, tetapi instans EC2 tidak perlu digabungkan ke domain. Dengan menggunakan domainlessgMSA, instance container tidak digabungkan ke domain, aplikasi lain pada instance tidak dapat menggunakan kredensyal untuk mengakses domain, dan tugas yang menggabungkan domain yang berbeda dapat berjalan pada instance yang sama. Sebagai gantinya, Anda memberikan nama rahasia di AWS Secrets Manager dalam CredSpec file. Rahasianya harus berisi nama pengguna, kata sandi, dan domain untuk masuk.

  Fitur ini didukung dan dapat digunakan dengan wadah Linux dan Windows.

  Fitur ini mirip dengan gMSA support for non-domain-joined container hostsfitur. Untuk informasi selengkapnya tentang fitur Windows, lihat gMSAarsitektur dan peningkatan di situs web Microsoft Learn.

  a. Buat pengguna di domain Active Directory Anda. Pengguna di Active Directory harus memiliki izin untuk mengakses akun gMSA layanan yang Anda gunakan dalam tugas.

  b. Buat rahasia di AWS Secrets Manager, setelah Anda membuat pengguna di Active Directory. Untuk informasi selengkapnya, lihat Membuat AWS Secrets Manager rahasia.

  c. Masukkan nama pengguna, kata sandi, dan domain ke pasangan nilai kunci JSON yang dipanggil`username`, `password` dan`domainName`, masing-masing.

  ```
  {"username":"username","password":"passw0rd", "domainName":"example.com"}
  ```

  d. Tambahkan konfigurasi ke CredSpec file untuk akun layanan. Tambahan `HostAccountConfig` berisi Amazon Resource Name (ARN) dari rahasia di Secrets Manager.

Pada Windows, `PluginGUID` harus cocok dengan GUID dalam contoh cuplikan berikut. Di Linux, `PluginGUID` itu diabaikan. Ganti `MySecret` dengan contoh dengan Amazon Resource Name (ARN) rahasia Anda.

```
    "ActiveDirectoryConfig": {
        "HostAccountConfig": {
            "PortableCcgVersion": "1",
            "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
            "PluginInput": {
                "CredentialArn": "arn:aws:secretsmanager:aws-
region:111122223333:secret:MySecret"
            }
        }
```

e. gMSAFitur tanpa domain memerlukan izin tambahan dalam peran eksekusi tugas. Ikuti langkahnya(Opsional) rahasia tanpa domain gMSA.

2. Konfigurasikan instance dan instal **`credentials-fetcher`** daemon

Anda dapat menginstal `credentials-fetcher` daemon dengan skrip data pengguna di Template Peluncuran EC2 Anda. Contoh berikut menunjukkan dua jenis data pengguna, `cloud-config` YAML atau bash script. Contoh-contoh ini untuk Amazon Linux 2023 (AL2023). Ganti `MyCluster` dengan nama cluster Amazon ECS yang Anda inginkan untuk bergabung dengan instans ini.

- **`cloud-config`** YAML

```
Content-Type: text/cloud-config
package_reboot_if_required: true
packages:
  # prerequisites
  - dotnet
  - realmd
  - oddjob
  - oddjob-mkhomedir
  - sssd
  - adcli
  - krb5-workstation
  - samba-common-tools
  # https://github.com/aws/credentials-fetcher gMSA credentials management for
 containers
  - credentials-fetcher
```

```
write_files:
# configure the ECS Agent to join your cluster.
# replace MyCluster with the name of your cluster.
- path: /etc/ecs/ecs.config
  owner: root:root
  permissions: '0644'
  content: |
    ECS_CLUSTER=MyCluster
    ECS_GMSA_SUPPORTED=true
runcmd:
# start the credentials-fetcher daemon and if it succeeded, make it start after
 every reboot
- "systemctl start credentials-fetcher"
- "systemctl is-active credentials-fetch && systemctl enable credentials-
fetcher"
```

- bashnaskah

  Jika Anda lebih nyaman dengan bash skrip dan memiliki beberapa variabel untuk ditulis`/etc/ecs/ecs.config`, gunakan `heredoc` format berikut. Format ini menulis segalanya di antara baris dimulai dengan cat dan `EOF` pada file konfigurasi.

```
#!/usr/bin/env bash
set -euxo pipefail

# prerequisites
timeout 30 dnf install -y dotnet realmd oddjob oddjob-mkhomedir sssd adcli
 krb5-workstation samba-common-tools
# install https://github.com/aws/credentials-fetcher gMSA credentials
 management for containers
timeout 30 dnf install -y credentials-fetcher

# start credentials-fetcher
systemctl start credentials-fetcher
systemctl is-active credentials-fetch && systemctl enable credentials-fetcher

cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_GMSA_SUPPORTED=true
EOF
```

Ada variabel konfigurasi opsional untuk `credentials-fetcher` daemon yang dapat Anda atur. `/etc/ecs/ecs.config` Kami menyarankan Anda mengatur variabel dalam data pengguna di blok YAMM atau `heredoc` mirip dengan contoh sebelumnya. Melakukannya mencegah masalah dengan konfigurasi sebagian yang dapat terjadi dengan mengedit file beberapa kali. Untuk informasi selengkapnya tentang konfigurasi agen ECS, lihat Amazon ECS Container Agent on. GitHub

- Secara opsional, Anda dapat menggunakan variabel `CREDENTIALS_FETCHER_HOST` jika Anda mengubah konfigurasi `credentials-fetcher` daemon untuk memindahkan soket ke lokasi lain.

Menyiapkan izin dan rahasia

Lakukan langkah-langkah berikut sekali untuk setiap aplikasi dan setiap definisi tugas. Kami menyarankan Anda menggunakan praktik terbaik untuk memberikan hak istimewa paling sedikit dan mempersempit izin yang digunakan dalam kebijakan. Dengan cara ini, setiap tugas hanya dapat membaca rahasia yang dibutuhkannya.

1. (Opsional) rahasia tanpa domain gMSA

   Jika Anda menggunakan metode domainless di mana instance tidak bergabung dengan domain, ikuti langkah ini.

   Anda harus menambahkan izin berikut sebagai kebijakan inline ke peran IAM eksekusi tugas. Melakukan hal itu memberikan akses `credentials-fetcher` daemon ke rahasia Secrets Manager. Ganti `MySecret` contoh dengan Amazon Resource Name (ARN) rahasia Anda dalam daftar. `Resource`

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:ssm:aws-region:111122223333:secret:MySecret"
            ]
```

```
        }
    ]
}
```

> **Note**
>
> Jika Anda menggunakan kunci KMS Anda sendiri untuk mengenkripsi rahasia Anda, Anda harus menambahkan izin yang diperlukan untuk peran ini dan menambahkan peran ini ke kebijakan kunci. AWS KMS

2. Putuskan apakah Anda menggunakan SSM Parameter Store atau S3 untuk menyimpan CredSpec

   Amazon ECS mendukung cara-cara berikut untuk mereferensikan jalur file di `credentialSpecs` bidang definisi tugas.

   Jika Anda menggabungkan instance ke satu domain, gunakan awalan `credentialspec:` di awal ARN dalam string. Jika Anda menggunakan domainlessgMSA, maka gunakan. `credentialspecdomainless:`

   Untuk informasi selengkapnya tentang CredSpec, lihat File spesifikasi kredensi.

   - Bucket Amazon S3

     Tambahkan spesifikasi kredensyal ke bucket Amazon S3. Kemudian, rujuk Nama Sumber Daya Amazon (ARN) dari bucket Amazon S3 di bidang `credentialSpecs` definisi tugas.

     ```
     {
         "family": "",
         "executionRoleArn": "",
         "containerDefinitions": [
             {
                 "name": "",
                 ...
                 "credentialSpecs": [
                     "credentialspecdomainless:arn:aws:s3:::${BucketName}/
     ${ObjectName}"
                 ],
                 ...
             }
         ],
         ...
     ```

```
}
```

Untuk memberikan akses tugas ke bucket S3, tambahkan izin berikut sebagai kebijakan inline ke peran IAM eksekusi tugas Amazon ECS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::{bucket_name}",
                "arn:aws:s3:::{bucket_name}/{object}"
            ]
        }
    ]
}
```

- Parameter SSM Menyimpan parameter

  Tambahkan spesifikasi kredensyal ke parameter Penyimpanan Parameter SSM. Kemudian, rujuk Amazon Resource Name (ARN) dari parameter SSM Parameter Store di `credentialSpecs` bidang definisi tugas.

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [
                "credentialspecdomainless:arn:aws:ssm:aws-
region:111122223333:parameter/parameter_name"
            ],
            ...
        }
```

```
    ],
    ...
}
```

Untuk memberikan akses tugas Anda ke parameter Penyimpanan Parameter SSM, tambahkan izin berikut sebagai kebijakan inline ke peran IAM eksekusi tugas Amazon ECS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameters"
            ],
            "Resource": [
                "arn:aws:ssm:aws-region:111122223333:parameter/parameter_name"
            ]
        }
    ]
}
```

File spesifikasi kredensi

Amazon ECS menggunakan file spesifikasi kredensi Direktori Aktif () CredSpec. File ini berisi gMSA metadata yang digunakan untuk menyebarkan konteks gMSA akun ke wadah. Linux Anda menghasilkan CredSpec dan mereferensikannya di `credentialSpecs` bidang dalam definisi tugas Anda. CredSpecFile tidak mengandung rahasia apa pun.

Berikut ini adalah contoh CredSpec file.

```
{
    "CmsPlugins": [
        "ActiveDirectory"
    ],
    "DomainJoinConfig": {
        "Sid": "S-1-5-21-2554468230-2647958158-2204241789",
        "MachineAccountName": "WebApp01",
        "Guid": "8665abd4-e947-4dd0-9a51-f8254943c90b",
        "DnsTreeName": "example.com",
        "DnsName": "example.com",
```

```
        "NetBiosName": "example"
    },
    "ActiveDirectoryConfig": {
        "GroupManagedServiceAccounts": [
            {
                "Name": "WebApp01",
                "Scope": "example.com"
            }
        ],
        "HostAccountConfig": {
            "PortableCcgVersion": "1",
            "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
            "PluginInput": {
                "CredentialArn": "arn:aws:secretsmanager:aws-
region:111122223333:secret:MySecret"
            }
        }
    }
}
```

MembuatCredSpec

Anda membuat CredSpec dengan menggunakan CredSpec PowerShell modul pada Windows komputer yang bergabung dengan domain. Ikuti langkah-langkah dalam Membuat spesifikasi kredensyal di situs web Microsoft Belajar.

Menggunakan gMSA untuk Linux wadah di Fargate

Amazon ECS mendukung otentikasi Direktori Aktif untuk wadah Linux di Fargate melalui jenis akun layanan khusus yang disebut grup Akun Layanan Terkelola (). gMSA

LinuxAplikasi jaringan berbasis, seperti aplikasi .NET Core, dapat menggunakan Active Directory untuk memfasilitasi otentikasi dan manajemen otorisasi antara pengguna dan layanan. Anda dapat menggunakan fitur ini dengan merancang aplikasi yang terintegrasi dengan Active Directory dan berjalan di server yang bergabung dengan domain. Tetapi, karena Linux kontainer tidak dapat digabungkan dengan domain, Anda perlu mengonfigurasi Linux wadah untuk dijalankan. gMSA

Pertimbangan

Pertimbangkan hal berikut sebelum Anda gunakan gMSA untuk Linux wadah di Fargate:

- Anda harus menjalankan Platform Versi 1.4 atau yang lebih baru.
- Anda mungkin memerlukan Windows komputer yang bergabung dengan domain untuk menyelesaikan prasyarat. Misalnya, Anda mungkin memerlukan Windows komputer yang bergabung dengan domain untuk membuat Active Directory denganPowerShell. gMSA PowerShell Alat Direktur RSAT Aktif hanya tersedia untukWindows. Untuk informasi selengkapnya, lihat Menginstal alat administrasi Direktori Aktif.
- Anda harus menggunakan domainless gMSA.

  Amazon ECS menggunakan file spesifikasi kredensi Direktori Aktif ()CredSpec. File ini berisi gMSA metadata yang digunakan untuk menyebarkan konteks gMSA akun ke wadah. Anda membuat CredSpec file, dan kemudian menyimpannya di ember Amazon S3.
- Sebuah tugas hanya dapat mendukung satu Direktori Aktif.

Prasyarat

Sebelum Anda menggunakan fitur gMSA untuk wadah Linux dengan Amazon ECS, pastikan untuk menyelesaikan yang berikut ini:

- Anda menyiapkan domain Direktori Aktif dengan sumber daya yang ingin diakses oleh kontainer Anda. Amazon ECS mendukung pengaturan berikut:
  - Direktori AWS Directory Service Aktif. AWS Directory Service adalah Direktori Aktif AWS terkelola yang di-host di Amazon EC2. Untuk informasi selengkapnya, lihat Memulai dengan Microsoft AD yang AWS Dikelola di Panduan AWS Directory Service Administrasi.
  - Direktori Aktif lokal. Anda harus memastikan bahwa instans penampung Amazon ECS Linux dapat bergabung dengan domain. Untuk informasi selengkapnya, lihat AWS Direct Connect.
- Anda memiliki gMSA akun yang ada di Direktori Aktif dan pengguna yang memiliki izin untuk mengakses akun gMSA layanan. Untuk informasi selengkapnya, lihat Jadikan pengguna Active Directory untuk domainless gMSA.
- Anda memiliki ember Amazon S3. Lihat informasi yang lebih lengkap di Membuat bucket dalam Panduan Pengguna Amazon S3.

Menyiapkan Linux Kontainer gMSA berkemampuan di Amazon ECS

Siapkan infrastruktur

Langkah-langkah berikut adalah pertimbangan dan pengaturan yang dilakukan sekali.

- Jadikan pengguna Active Directory untuk domainless gMSA

  Saat Anda menggunakan domainlessgMSA, penampung tidak bergabung dengan domain. Aplikasi lain yang berjalan di wadah tidak dapat menggunakan kredensyal untuk mengakses domain. Tugas yang menggunakan domain berbeda dapat berjalan pada wadah yang sama. Anda memberikan nama rahasia di AWS Secrets Manager dalam CredSpec file. Rahasianya harus berisi nama pengguna, kata sandi, dan domain untuk masuk.

  Fitur ini mirip dengan gMSA support for non-domain-joined container hostsfitur. Untuk informasi selengkapnya tentang fitur Windows, lihat gMSAarsitektur dan peningkatan di situs web Microsoft Learn.

  a. Konfigurasikan pengguna di domain Active Directory Anda. Pengguna di Active Directory harus memiliki izin untuk mengakses akun gMSA layanan yang Anda gunakan dalam tugas.
  b. Anda memiliki VPC dan subnet yang dapat menyelesaikan nama domain Active Directory. Konfigurasikan VPC dengan opsi DHCP dengan nama domain yang menunjuk ke nama layanan Active Directory. Untuk informasi tentang cara mengonfigurasi opsi DHCP untuk VPC, lihat Set opsi Bekerja dengan DHCP di Panduan Pengguna Amazon Virtual Private Cloud.
  c. Buat rahasia di AWS Secrets Manager.
  d. Buat file spesifikasi kredensyal.

Menyiapkan izin dan rahasia

Lakukan langkah-langkah berikut satu kali untuk setiap aplikasi dan setiap definisi tugas. Kami menyarankan Anda menggunakan praktik terbaik untuk memberikan hak istimewa paling sedikit dan mempersempit izin yang digunakan dalam kebijakan. Dengan cara ini, setiap tugas hanya dapat membaca rahasia yang dibutuhkannya.

1. Buat pengguna di domain Active Directory Anda. Pengguna di Active Directory harus memiliki izin untuk mengakses akun gMSA layanan yang Anda gunakan dalam tugas.
2. Setelah Anda membuat pengguna Active Directory, buat rahasia di AWS Secrets Manager. Untuk informasi selengkapnya, lihat Membuat AWS Secrets Manager rahasia.
3. Masukkan nama pengguna, kata sandi, dan domain ke pasangan nilai kunci JSON yang dipanggil`username`, `password` dan`domainName`, masing-masing.

   ```
   {"username":"username","password":"passw0rd", "domainName":"example.com"}
   ```

4. Anda harus menambahkan izin berikut sebagai kebijakan inline ke peran IAM eksekusi tugas. Melakukan hal itu memberikan akses `credentials-fetcher` daemon ke rahasia Secrets Manager. Ganti `MySecret` contoh dengan Amazon Resource Name (ARN) rahasia Anda dalam daftar. `Resource`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": [
        "arn:aws:secretsmanager:aws-region:111122223333:secret:MySecret"
      ]
    }
  ]
}
```

> **Note**
>
> Jika Anda menggunakan kunci KMS Anda sendiri untuk mengenkripsi rahasia Anda, Anda harus menambahkan izin yang diperlukan untuk peran ini dan menambahkan peran ini ke kebijakan kunci. AWS KMS

5. Tambahkan spesifikasi kredensyal ke bucket Amazon S3. Kemudian, rujuk Nama Sumber Daya Amazon (ARN) dari bucket Amazon S3 di bidang `credentialSpecs` definisi tugas.

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [
                "credentialspecdomainless:arn:aws:s3:::${BucketName}/${ObjectName}"
            ],
            ...
        }
```

```
    ],
    ...
}
```

Untuk memberikan akses tugas ke bucket S3, tambahkan izin berikut sebagai kebijakan inline ke peran IAM eksekusi tugas Amazon ECS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListObject"
            ],
            "Resource": [
                "arn:aws:s3:::{bucket_name}",
                "arn:aws:s3:::{bucket_name}/{object}"
            ]
        }
    ]
}
```

File spesifikasi kredensi

Amazon ECS menggunakan file spesifikasi kredensi Direktori Aktif () CredSpec. File ini berisi gMSA metadata yang digunakan untuk menyebarkan konteks gMSA akun ke wadah. Linux Anda menghasilkan CredSpec dan mereferensikannya di `credentialSpecs` bidang dalam definisi tugas Anda. CredSpecFile tidak mengandung rahasia apa pun.

Berikut ini adalah contoh CredSpec file.

```
{
    "CmsPlugins": [
        "ActiveDirectory"
    ],
    "DomainJoinConfig": {
        "Sid": "S-1-5-21-2554468230-2647958158-2204241789",
        "MachineAccountName": "WebApp01",
```

```
        "Guid": "8665abd4-e947-4dd0-9a51-f8254943c90b",
        "DnsTreeName": "example.com",
        "DnsName": "example.com",
        "NetBiosName": "example"
    },
    "ActiveDirectoryConfig": {
        "GroupManagedServiceAccounts": [
            {
                "Name": "WebApp01",
                "Scope": "example.com"
            }
        ],
        "HostAccountConfig": {
            "PortableCcgVersion": "1",
            "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
            "PluginInput": {
                "CredentialArn": "arn:aws:secretsmanager:aws-
region:111122223333:secret:MySecret"
            }
        }
    }
}
```

Membuat CredSpec dan mengunggahnya ke Amazon S3

Anda membuat CredSpec dengan menggunakan CredSpec PowerShell modul pada Windows komputer yang bergabung dengan domain. Ikuti langkah-langkah dalam Membuat spesifikasi kredensyal di situs web Microsoft Belajar.

Setelah Anda membuat file spesifikasi kredensyal, unggah ke bucket Amazon S3. Salin CredSpec file ke komputer atau lingkungan tempat Anda menjalankan AWS CLI perintah.

Jalankan AWS CLI perintah berikut untuk mengunggah CredSpec ke Amazon S3. Ganti `MyBucket` dengan nama bucket Amazon S3 Anda. Anda dapat menyimpan file sebagai objek di bucket dan lokasi mana pun, tetapi Anda harus mengizinkan akses ke bucket dan lokasi tersebut dalam kebijakan yang Anda lampirkan ke peran eksekusi tugas.

Untuk PowerShell, gunakan perintah berikut:

```
$ Write-S3Object -BucketName "MyBucket" -Key "ecs-domainless-gmsa-credspec" -File
 "gmsa-cred-spec.json"
```

AWS CLI Perintah berikut menggunakan karakter kelanjutan garis miring terbalik yang digunakan oleh sh dan shell yang kompatibel.

```
$ aws s3 cp gmsa-cred-spec.json \
s3://MyBucket/ecs-domainless-gmsa-credspec
```

Menggunakan EC2 Image Builder untuk membuat AMI yang dioptimalkan Amazon ECS yang disesuaikan

AWS merekomendasikan agar Anda menggunakan AMI yang dioptimalkan Amazon ECS karena AMI tersebut telah dikonfigurasi sebelumnya dengan persyaratan dan rekomendasi untuk menjalankan beban kerja penampung Anda. Mungkin ada saat-saat ketika Anda perlu menyesuaikan AMI Anda untuk menambahkan perangkat lunak tambahan. Anda dapat menggunakan EC2 Image Builder untuk pembuatan, pengelolaan, dan penyebaran gambar server Anda. Anda mempertahankan kepemilikan gambar yang disesuaikan yang dibuat di akun Anda. Anda dapat menggunakan pipeline EC2 Image Builder untuk mengotomatiskan pembaruan dan penambalan sistem untuk gambar Anda, atau menggunakan perintah yang berdiri sendiri untuk membuat gambar dengan sumber daya konfigurasi yang ditentukan.

Anda membuat resep untuk gambar Anda. Resepnya mencakup gambar induk, dan komponen tambahan apa pun. Anda juga membuat pipeline yang mendistribusikan AMI khusus Anda.

Anda membuat resep untuk gambar Anda. Resep image Builder Image adalah dokumen yang mendefinisikan image dasar dan komponen yang diterapkan pada image dasar untuk menghasilkan konfigurasi yang diinginkan untuk gambar AMI keluaran. Anda juga membuat pipeline yang mendistribusikan AMI khusus Anda. Untuk informasi selengkapnya, lihat Cara kerja EC2 Image Builder di Panduan Pengguna EC2 Image Builder.

Kami menyarankan Anda menggunakan salah satu AMI yang dioptimalkan Amazon ECS berikut sebagai “Gambar Induk” Anda di EC2 Image Builder:

- Linux
  - Amazon ECS yang dioptimalkan AL2023 x86
  - Amazon ECS yang dioptimalkan Amazon Linux 2023 (arm64) AMI
  - Amazon ECS yang dioptimalkan Amazon Linux 2 kernel 5 AMI
  - Amazon ECS yang dioptimalkan Amazon Linux 2 x86 AMI

- Windows
  - Amazon ECS-dioptimalkan Windows 2022 Full x86
  - Amazon ECS yang dioptimalkan Windows 2022 Core x86
  - Amazon ECS yang dioptimalkan Windows 2019 Full x86
  - Amazon ECS yang dioptimalkan Windows 2019 Core x86
  - Amazon ECS yang dioptimalkan Windows 2016 Full x86

Kami juga menyarankan Anda memilih “Gunakan versi OS terbaru yang tersedia”. Pipeline akan menggunakan versi semantik untuk gambar induk, yang membantu mendeteksi pembaruan ketergantungan dalam pekerjaan yang dijadwalkan secara otomatis. Untuk informasi selengkapnya, lihat Pembuatan versi semantik di Panduan Pengguna EC2 Image Builder.

AWS secara teratur memperbarui gambar AMI Amazon ECS yang dioptimalkan dengan tambalan keamanan dan versi agen kontainer baru. Saat Anda menggunakan ID AMI sebagai gambar induk Anda dalam resep gambar Anda, Anda perlu secara teratur memeriksa pembaruan pada gambar induk. Jika ada pembaruan, Anda harus membuat versi baru resep Anda dengan AMI yang diperbarui. Ini memastikan bahwa gambar kustom Anda menggabungkan versi terbaru dari gambar induk. Untuk informasi tentang cara membuat alur kerja untuk memperbarui instans EC2 secara otomatis di klaster Amazon ECS dengan AMI yang baru dibuat, lihat Cara membuat pipeline pengerasan AMI dan mengotomatiskan pembaruan ke armada instans ECS Anda.

Anda juga dapat menentukan Amazon Resource Name (ARN) dari gambar induk yang diterbitkan melalui pipeline EC2 Image Builder terkelola. Amazon secara rutin menerbitkan gambar AMI Amazon ECS yang dioptimalkan melalui saluran pipa terkelola. Gambar-gambar ini dapat diakses publik. Anda harus memiliki izin yang benar untuk mengakses gambar. Saat Anda menggunakan ARN gambar alih-alih AMI dalam resep Image Builder, pipeline Anda secara otomatis menggunakan versi terbaru dari gambar induk setiap kali dijalankan. Pendekatan ini menghilangkan kebutuhan untuk membuat versi resep baru secara manual untuk setiap pembaruan.

Menggunakan gambar ARN dengan infrastruktur sebagai kode (IAc)

Anda dapat mengonfigurasi resep menggunakan konsol EC2 Image Builder, atau infrastruktur sebagai kode (misalnya AWS CloudFormation,), atau AWS SDK. Saat Anda menentukan gambar induk dalam resep Anda, Anda dapat menentukan ID AMI EC2, ARN image Builder Image Builder, ID produk AWS Marketplace , atau gambar kontainer. AWS menerbitkan ID AMI dan ARN gambar Image Builder dari AMI Amazon ECS yang dioptimalkan secara publik. Berikut ini adalah format ARN untuk gambar:

```
arn:${Partition}:imagebuilder:${Region}:${Account}:image/${ImageName}/${ImageVersion}
```

Ini `ImageVersion` memiliki format berikut. Ganti *`mayor, minor`*, dan *`patch`* dengan nilai terbaru.

```
<major>.<minor>.<patch>
```

Anda dapat menggantimajor, `minor` dan `patch` ke nilai tertentu atau Anda dapat menggunakan ARN tanpa versi dari sebuah gambar, sehingga pipeline Anda tetap up-to-date dengan versi terbaru dari gambar induk. ARN tanpa versi menggunakan format wildcard 'x.x.x' untuk mewakili versi gambar. Pendekatan ini memungkinkan layanan Image Builder untuk secara otomatis menyelesaikan ke versi terbaru dari gambar. Menggunakan ARN tanpa versi memastikan bahwa referensi Anda selalu mengarah ke gambar terbaru yang tersedia, merampingkan proses mempertahankan gambar terbaru dalam penerapan Anda. Saat Anda membuat resep dengan konsol, EC2 Image Builder secara otomatis mengidentifikasi ARN untuk gambar induk Anda. Saat Anda menggunakan IAc untuk membuat resep, Anda harus mengidentifikasi ARN dan memilih versi gambar yang diinginkan atau menggunakan arn tanpa versi untuk menunjukkan gambar terbaru yang tersedia. Kami menyarankan Anda membuat skrip otomatis untuk memfilter dan hanya menampilkan gambar yang sesuai dengan kriteria Anda. Skrip Python berikut menunjukkan cara mengambil daftar AMI Amazon ECS yang dioptimalkan.

Skrip menerima dua argumen opsional: `owner` dan`platform`, dengan nilai default “Amazon”, dan “Windows” masing-masing. Nilai yang valid untuk argumen pemilik adalah:`Self`,`Shared`,`Amazon`, dan`ThirdParty`. Nilai yang valid untuk argumen platform adalah `Windows` dan`Linux`. Misalnya, jika Anda menjalankan skrip dengan `owner` argumen yang disetel ke `Amazon` dan `platform` disetel ke`Linux`, skrip akan menghasilkan daftar gambar yang diterbitkan oleh Amazon termasuk gambar yang dioptimalkan Amazon ECS.

```
import boto3
import argparse

def list_images(owner, platform):
    # Create a Boto3 session
    session = boto3.Session()

    # Create an EC2 Image Builder client
    client = session.client('imagebuilder')

    # Define the initial request parameters
    request_params = {
```

```
        'owner': owner,
        'filters': [
            {
                'name': 'platform',
                'values': [platform]
            }
        ]
    }

    # Initialize the results list
    all_images = []

    # Get the initial response with the first page of results
    response = client.list_images(**request_params)

    # Extract images from the response
    all_images.extend(response['imageVersionList'])

    # While 'nextToken' is present, continue paginating
    while 'nextToken' in response and response['nextToken']:
        # Update the token for the next request
        request_params['nextToken'] = response['nextToken']

        # Get the next page of results
        response = client.list_images(**request_params)

        # Extract images from the response
        all_images.extend(response['imageVersionList'])

    return all_images

def main():
    # Initialize the parser
    parser = argparse.ArgumentParser(description="List AWS images based on owner and
 platform")

    # Add the parameters/arguments
    parser.add_argument("--owner", default="Amazon", help="The owner of the images.
 Default is 'Amazon'.")
    parser.add_argument("--platform", default="Windows", help="The platform type of the
 images. Default is 'Windows'.")

    # Parse the arguments
    args = parser.parse_args()
```

```
    # Retrieve all images based on the provided owner and platform
    images = list_images(args.owner, args.platform)

    # Print the details of the images
    for image in images:
        print(f"Name: {image['name']}, Version: {image['version']}, ARN:
 {image['arn']}")

if __name__ == "__main__":
    main()
```

Menggunakan gambar ARN dengan AWS CloudFormation

Resep gambar Image Builder adalah cetak biru yang menentukan gambar induk dan komponen yang diperlukan untuk mencapai konfigurasi gambar keluaran yang diinginkan. Anda menggunakan `AWS::ImageBuilder::ImageRecipe` sumber daya. Atur `ParentImage` nilai ke gambar ARN. Gunakan ARN tanpa versi dari gambar yang Anda inginkan untuk memastikan pipeline Anda selalu menggunakan versi gambar terbaru. Misalnya, `arn:aws:imagebuilder:us-east-1:aws:image/amazon-linux-2023-ecs-optimized-x86/x.x.x`. Definisi `AWS::ImageBuilder::ImageRecipe` sumber daya berikut menggunakan ARN image terkelola Amazon:

```
ECSRecipe:
    Type: AWS::ImageBuilder::ImageRecipe
    Properties:
      Name: MyRecipe
      Version: '1.0.0'
      Components:
        - ComponentArn: [<The component arns of the image recipe>]
      ParentImage: "arn:aws:imagebuilder:us-east-1:aws:image/amazon-linux-2023-ecs-
optimized-x86/x.x.x"
```

Untuk informasi selengkapnya tentang `AWS::ImageBuilder::ImageRecipe`sumber daya, lihat di Panduan AWS CloudFormation Pengguna.

Anda dapat mengotomatiskan pembuatan gambar baru di pipeline Anda dengan menyetel `Schedule` properti `AWS::ImageBuilder::ImagePipeline` sumber daya. Jadwal mencakup kondisi awal dan ekspresi cron. Untuk informasi selengkapnya, lihat `AWS::ImageBuilder::ImagePipeline` di AWS CloudFormation Panduan Pengguna.

`AWS::ImageBuilder::ImagePipeline`Contoh berikut ini memiliki pipeline yang menjalankan build pada pukul 10:00 AM Coordinated Universal Time (UTC) setiap hari. Tetapkan `Schedule` nilai-nilai berikut:

- Atur `PipelineExecutionStartCondition` ke `EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE`. Build dimulai hanya jika sumber daya dependen seperti image induk atau komponen, yang menggunakan wildcard 'x' dalam versi semantiknya, diperbarui. Ini memastikan build menggabungkan pembaruan terbaru dari sumber daya tersebut.
- Setel ScheduleExpression ke ekspresi `(0 10 * * ? *)` cron.

```
ECSPipeline:
    Type: AWS::ImageBuilder::ImagePipeline
    Properties:
      Name: my-pipeline
      ImageRecipeArn: <arn of the recipe you created in previous step>
      InfrastructureConfigurationArn: <ARN of the infrastructure configuration
 associated with this image pipeline>
      Schedule:
        PipelineExecutionStartCondition:
 EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE
        ScheduleExpression: 'cron(0 10 * * ? *)'
```

Menggunakan gambar ARN dengan Terraform

Pendekatan untuk menentukan gambar dan jadwal induk pipeline Anda di Terraform sejalan dengan yang ada di. AWS CloudFormation Anda menggunakan `aws_imagebuilder_image_recipe` sumber daya. Atur `parent_image` nilai ke gambar ARN. Gunakan ARN tanpa versi dari gambar yang Anda inginkan untuk memastikan pipeline Anda selalu menggunakan versi gambar terbaru.. Untuk informasi lebih lanjut, lihat `aws_imagebuilder_image_recipe`di dokumentasi Terraform.

Di blok konfigurasi jadwal`aws_imagebuilder_image_pipeline resource`, atur nilai `schedule_expression` argumen ke ekspresi cron pilihan Anda untuk menentukan seberapa sering pipeline berjalan, dan atur `pipeline_execution_start_condition` ke`EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE`. Untuk informasi lebih lanjut, lihat `aws_imagebuilder_image_pipeline`di dokumentasi Terraform.

Pemecahan masalah Amazon ECS

Anda mungkin perlu memecahkan masalah dengan penyeimbang beban, tugas, layanan, atau instance kontainer Anda. Bab ini membantu Anda menemukan informasi diagnostik dari agen penampung Amazon ECS, daemon Docker pada instance container, dan log peristiwa layanan di konsol Amazon ECS.

Topik

Pemecahan masalah dengan ECS Exec

Berikut ini adalah catatan pemecahan masalah untuk membantu mendiagnosis mengapa Anda mungkin mendapatkan kesalahan saat menggunakan ECS Exec.

Verifikasi menggunakan Amazon ECS Exec Checker

Skrip ECS Exec Checker menyediakan cara untuk memverifikasi dan memvalidasi bahwa cluster dan tugas Amazon ECS Anda telah memenuhi prasyarat untuk menggunakan fitur ECS Exec. Skrip ECS Exec Checker memverifikasi AWS CLI lingkungan dan cluster Anda dan tugas siap untuk ECS Exec, dengan memanggil berbagai API atas nama Anda. Alat ini membutuhkan versi terbaru dari AWS CLI dan yang `jq` tersedia. Untuk informasi lebih lanjut, lihat ECS Exec Checker di. GitHub

Terjadi kesalahan saat memanggil **`execute-command`**

Jika terjadi kesalahan `The execute command failed`, berikut ini kemungkinan penyebabnya.

- Tugas tidak memiliki izin yang diperlukan. Verifikasi bahwa definisi tugas yang digunakan untuk meluncurkan tugas Anda memiliki peran IAM tugas yang ditentukan dan peran tersebut memiliki izin yang diperlukan. Untuk informasi selengkapnya, lihat Izin IAM diperlukan untuk ECS Exec.
- Agen SSM tidak diinstal atau tidak berjalan.
- Ada antarmuka titik akhir Amazon VPC untuk Amazon ECS, tetapi tidak ada satu untuk Systems Manager Session Manager.

Memecahkan masalah Amazon ECS Anywhere

Amazon ECS Anywhere menyediakan dukungan untuk mendaftarkan instans eksternal seperti server lokal atau mesin virtual (VM) ke cluster Amazon ECS Anda. Berikut ini adalah permasalahan umum yang mungkin Anda alami serta rekomendasi pemecahan terhadap masalah umum yang mungkin Anda alami.

Topik

Permasalahan registrasi instans eksternal

Saat mendaftarkan instans eksternal dengan kluster Amazon ECS Anda, persyaratan berikut harus dipenuhi:

- AWS Systems Manager Aktivasi, yang terdiri dari ID aktivasi dan kode aktivasi, harus diambil. Anda menggunakannya untuk mendaftarkan instance eksternal sebagai instans terkelola Systems Manager. Ketika aktivasi Systems Manager diminta, tentukan batas pendaftaran dan tanggal kedaluwarsa. Batas pendaftaran tersebut menentukan jumlah maksimum instans yang dapat didaftarkan dengan menggunakan aktivasi. Nilai default untuk batas pendaftaran adalah 1 instance. Tanggal kedaluwarsa menentukan kapan aktivasi akan berakhir. Nilai default adalah 24 jam. Jika aktivasi Systems Manager yang Anda gunakan untuk mendaftarkan instance eksternal Anda tidak valid, mintalah yang baru. Untuk informasi selengkapnya, lihat Pendaftaran instans eksternal ke sebuah klaster.
- Kebijakan IAM digunakan untuk memberikan instans eksternal Anda izin yang diperlukan untuk berkomunikasi dengan operasi AWS API. Jika kebijakan terkelola ini tidak dibuat dengan benar dan tidak memiliki izin yang diperlukan, pendaftaran instans eksternal akan gagal. Untuk informasi selengkapnya, lihat Peran IAM ECS Anywhere.
- Amazon ECS menyediakan skrip instalasi yang menginstal Docker, agen penampung Amazon ECS, dan Agen Systems Manager pada instans eksternal Anda. Jika penulisan pada penginstalan gagal, penulisan kemungkinan tidak dapat dijalankan lagi pada instans yang sama tanpa terjadinya sebuah kesalahan. Jika ini terjadi, ikuti proses pembersihan untuk menghapus AWS sumber daya dari instance sehingga Anda dapat menjalankan skrip instalasi lagi. Untuk informasi selengkapnya, lihat Membatalkan pendaftaran instans eksternal.

  > **Note**
  >
  > Ketahuilah bahwa, jika skrip instalasi berhasil meminta dan menggunakan aktivasi Systems Manager, menjalankan skrip instalasi untuk kedua kalinya menggunakan aktivasi Systems Manager lagi. Hal ini dapat menyebabkan Anda mencapai batas pendaftaran untuk aktivasi tersebut. Jika batas ini tercapai, Anda harus membuat aktivasi baru.

- Saat menjalankan skrip instalasi pada instance eksternal untuk beban kerja GPU, jika driver NVIDIA tidak terdeteksi atau dikonfigurasi dengan benar, kesalahan akan terjadi. Skrip instalasi menggunakan `nvidia-smi` perintah untuk mengkonfirmasi keberadaan driver NVIDIA.

Masalah jaringan instans eksternal

Untuk dapat mengomunikasikan perubahan apa pun, instans eksternal Anda memerlukan koneksi jaringan ke AWS. Jika instans eksternal Anda kehilangan koneksi jaringannya AWS, tugas yang berjalan pada instance Anda tetap berjalan kecuali dihentikan secara manual. Setelah koneksi

ke AWS dipulihkan, AWS kredensyal yang digunakan oleh agen penampung Amazon ECS dan Agen Systems Manager pada instans eksternal diperpanjang secara otomatis. Untuk informasi selengkapnya tentang AWS domain yang digunakan untuk komunikasi antara instans eksternal Anda dan AWS, lihatJaringan dengan ECS Anywhere.

Masalah menjalankan tugas pada instans eksternal

Jika tugas atau kontainer Anda gagal dijalanan pada instans eksternal Anda, penyebab paling umum adalah terkait dengan jaringan atau izin. Jika kontainer Anda menarik gambarnya dari Amazon ECR atau dikonfigurasi untuk mengirim log kontainer ke CloudWatch Log, definisi tugas Anda harus menentukan peran IAM eksekusi tugas yang valid. Tanpa peran IAM eksekusi tugas yang valid, kontainer Anda akan gagal untuk memulai. Untuk informasi selengkapnya tentang permasalahan yang berkaitan dengan jaringan, lihat Masalah jaringan instans eksternal.

> **Important**
>
> Amazon ECS menyediakan alat pengumpulan log Amazon ECS. Anda dapat menggunakannya untuk mengumpulkan catatan dari instans eksternal Anda dengan tujuan memecahkan permasalahan. Untuk informasi selengkapnya, lihat Kolektor log Amazon ECS.

Memeriksa tugas yang telah dihentikan untuk kesalahan

Jika Anda kesulitan memulai tugas, tugas Anda mungkin berhenti karena kesalahan aplikasi atau konfigurasi. Contohnya, Anda menjalankan tugas dan tugas menampilkan status `PENDING` dan kemudian menghilang. Anda dapat melihat kesalahan tugas yang dihentikan seperti ini di konsol Amazon ECS dengan melihat tugas yang dihentikan dan memeriksanya untuk pesan kesalahan.

Jika definisi tugas Anda menggunakan driver `awslogs` log, log aplikasi yang ditulis ke Amazon CloudWatch Logs akan ditampilkan di tab Log di konsol Amazon ECS selama tugas yang dihentikan muncul.

Jika tugas Anda dibuat oleh layanan Amazon ECS, tindakan yang dilakukan Amazon ECS untuk mempertahankan layanan dipublikasikan dalam acara layanan. Anda dapat melihat peristiwa di AWS Management Console, AWS SDK AWS CLI, Amazon ECS API, atau alat yang menggunakan SDK dan API. Peristiwa ini termasuk Amazon ECS menghentikan dan menggantikan tugas karena kontainer dalam tugas telah berhenti berjalan, atau telah gagal terlalu banyak pemeriksaan kesehatan dari Elastic Load Balancing. Untuk informasi selengkapnya, lihat Pesan peristiwa layanan.

Jika tugas Anda berjalan pada instance container di Amazon EC2 atau komputer eksternal, Anda juga dapat melihat log runtime container dan Amazon ECS Agent. Log ini ada di host Amazon EC2 instans atau komputer eksternal. Untuk informasi selengkapnya, lihat Lokasi file log Amazon ECS.

> **Important**
>
> Tugas yang dihentikan hanya muncul di konsol Amazon ECS AWS CLI, dan AWS SDK setidaknya selama 1 jam setelah tugas berhenti. Setelah itu, detail tugas yang dihentikan kedaluwarsa dan tidak tersedia di Amazon ECS.
> Amazon ECS juga mengirimkan peristiwa perubahan status tugas ke Amazon EventBridge. Anda tidak dapat melihat acara di EventBridge. Sebagai gantinya, Anda membuat aturan untuk mengirim peristiwa ke penyimpanan persisten lainnya seperti Amazon CloudWatch Logs. Anda dapat menggunakan penyimpanan untuk melihat detail tugas yang dihentikan setelah kedaluwarsa dari tampilan di konsol Amazon ECS. Untuk informasi selengkapnya, lihat Acara perubahan status tugas Amazon ECS.
> Untuk EventBridge konfigurasi sampel untuk mengarsipkan peristiwa Amazon ECS ke CloudWatch Log Amazon, lihat Tugas yang Dihentikan ECS di CloudWatch Log di situs web. GitHub

Ikuti langkah-langkah ini untuk memeriksa kesalahan tugas yang dihentikan.

Console

AWS Management Console

Langkah-langkah berikut dapat digunakan untuk memeriksa tugas yang dihentikan untuk kesalahan menggunakan yang baru AWS Management Console.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih cluster.
4. Pada halaman Cluster: ***name***, pilih tab Tasks.
5. Konfigurasikan filter untuk menampilkan tugas yang dihentikan. Untuk Filter status yang diinginkan, pilih Dihentikan atau Status apa pun yang diinginkan.

   Opsi Berhenti menampilkan tugas yang dihentikan dan Status apa pun yang diinginkan menampilkan semua tugas Anda.

6. Pilih tugas yang dihentikan untuk diperiksa.
7. Di baris untuk tugas yang dihentikan, di kolom Status Terakhir, pilih Berhenti.

   Jendela pop-up menampilkan alasan berhenti.

AWS CLI

1. Daftar tugas yang berhenti dalam sebuah klaster. Output berisi Amazon Resource Name (ARN) dari tugas, yang Anda butuhkan untuk menjelaskan tugas tersebut.

```
aws ecs list-tasks \
     --cluster cluster_name \
     --desired-status STOPPED \
     --region us-west-2
```

2. Jelaskan tugas yang berhenti untuk mengambil respons `stoppedReason`.

```
aws ecs describe-tasks \
     --cluster cluster_name \
     --tasks arn:aws:ecs:us-west-2:account_id:task/cluster_name/task_ID \
     --region us-west-2
```

Sumber daya tambahan

Halaman-halaman berikut memberikan informasi tambahan tentang kode kesalahan:

- Mengapa tugas Amazon ECS saya dihentikan
- Kode kesalahan tugas yang dihentikan

Kode kesalahan tugas yang berhenti

Berikut ini adalah kemungkinan pesan kesalahan yang mungkin Anda terima ketika tugas Fargate Anda dihentikan secara tak terduga. Kesalahan pesan yang dikembalikan oleh agen kontainer dan prefiks yang tergantung pada versi platform yang digunakan oleh tugas.

Untuk memeriksa tugas yang dihentikan untuk pesan kesalahan menggunakan AWS Management Console, lihatMemeriksa tugas yang telah dihentikan untuk kesalahan.

`ContainerRuntimeTimeoutError`, `DockerTimeoutError`(Platform Linux versi 1.3.0 atau lebih lama)

Kesalahan ini terjadi ketika sebuah kontainer tidak dapat melakukan transisi ke `RUNNING` atau kondisi `STOPPED` dalam periode timeout. Alasan dan nilai timeout yang disediakan dalam kesalahan pesan.

Contoh: `ContainerRuntimeTimeoutError: Could not transition to running; timed out after waiting 1m: <reason>`

`CannotStartContainerError`

Kesalahan ini terjadi ketika kontainer tidak dapat dimulai.

Contoh: `CannotStartContainerError: failed to get container status: <reason>`

`CannotStopContainerError`

Kesalahan ini terjadi ketika wadah tidak dapat dihentikan.

Contoh: `CannotStopContainerError: failed sending SIGTERM to container: <reason>`

`CannotInspectContainerError`

Kesalahan ini terjadi ketika agen kontainer tidak dapat menjelaskan kontainer melalui waktu aktif kontainer.

Saat menggunakan platform versi 1.3 atau sebelumnya, agen Amazon ECS mengembalikan alasannya dari Docker.

Saat menggunakan platform versi 1.4 atau yang lebih baru `1.4.0` atau lebih baru (Linux) `1.0.0` atau yang lebih baru (Windows), agen Fargate mengembalikan alasannya. `containerd`

Contoh: `CannotInspectContainerError: <reason>`

`ResourceInitializationError`

Kesalahan ini terjadi ketika agen penampung untuk tugas Fargate Anda gagal membuat atau mem-bootstrap sumber daya yang diperlukan untuk memulai wadah atau tugas yang dimilikinya.

Penyebab umum untuk kesalahan ini adalah menggunakan VPC yang tidak memiliki resolusi DNS yang diaktifkan.

Kesalahan ini hanya terjadi jika Anda menggunakan versi platform `1.4.0` atau yang lebih baru (Linux) `1.0.0` atau yang lebih baru (Windows).

Contoh: `ResourceInitializationError: failed to initialize logging driver: <reason>`

`CannotPullContainerError`

Kesalahan ini terjadi ketika agen tidak dapat menarik citra kontainer yang ditentukan dalam ketentuan tugas. Untuk informasi selengkapnya, lihat CannotPullContainer kesalahan tugas.

Contoh: `CannotPullContainerError: <reason>`

`CannotCreateVolumeError`

Kesalahan ini terjadi ketika agen tidak dapat membuat volume pemasangan yang ditentukan dalam ketentuan tugas.

Kesalahan ini hanya terjadi jika Anda menggunakan versi platform `1.4.0` atau yang lebih baru (Linux) `1.0.0` atau yang lebih baru (Windows).

Contoh: `CannotCreateVolumeError: <reason>`

`ContainerRuntimeError`

Kesalahan ini terjadi ketika agen menerima kesalahan tak terduga dari `containerd` untuk operasi khusus runtime. Kesalahan ini biasanya disebabkan oleh kegagalan internal pada agen atau `containerd` runtime.

Kesalahan ini hanya terjadi jika Anda menggunakan versi platform `1.4.0` atau yang lebih baru (Linux) `1.0.0` atau yang lebih baru (Windows).

Contoh: `ContainerRuntimeError: failed to create container IO set: <reason>`

`OutOfMemoryError`

Kesalahan ini terjadi ketika kontainer keluar dikarenakan proses dalam kontainer mengonsumsi lebih banyak memori daripada memori yang telah dialokasikan dalam ketentuan tugas.

Contoh: `OutOfMemoryError: container killed due to memory usage`

`InternalError`

Kesalahan ini terjadi ketika agen menghadapi non-waktu aktif terkait dengan kesalahan internal secara tak terduga.

Kesalahan ini hanya terjadi jika menggunakan platform versi 1.4 atau yang lebih baru.

Contoh: `InternalError: <reason>`

`TaskFailedToStart`

Kesalahan ini terjadi ketika lampiran ENI diminta. Amazon EC2 menangani penyediaan ENI secara asinkron. Proses penyediaan membutuhkan waktu. Amazon ECS memiliki batas waktu jika ada waktu tunggu yang lama atau kegagalan yang tidak dilaporkan. Ada kalanya ENI disediakan, tetapi laporan tersebut datang ke Amazon ECS setelah batas waktu kegagalan. Dalam kasus ini, Amazon ECS melihat kegagalan tugas yang dilaporkan dengan ENI yang sedang digunakan.

Contoh: `InternalError: <reason>`

`SpotInterruption`

Kesalahan ini terjadi ketika tidak ada kapasitas Fargate Spot atau ketika Fargate mengambil kembali kapasitas Spot.

Contoh: `SpotInterruption: Your Spot Task was interrupted`

Contoh: `InternalError: <reason>`

`TaskFailedToStart`

Kesalahan ini terjadi ketika subnet yang meng-host instance tidak memiliki cukup alamat IP. Jumlah alamat IP yang tersedia tersedia di halaman detail subnet, atau dengan menggunakan`describe-subnets`. Untuk informasi selengkapnya, lihat Melihat subnet Anda di Panduan Pengguna Amazon VPC.

Contoh: `Unexpected EC2 error while attempting to Create Network Interface with public IP assignment enabled in subnet 'subnet-id': InsufficientFreeAddressesInSubnet`

`TaskFailedToStart`

Kesalahan ini terjadi ketika Anda memilih definisi tugas dengan tipe peluncuran yang tidak cocok dengan tipe kapasitas cluster. Untuk informasi selengkapnya, lihat Jenis peluncuran Amazon ECS.

Contoh: `The selected task definition is not compatible with the selected compute strategy`

CannotPullContainer kesalahan tugas

Kesalahan berikut menunjukkan bahwa, saat membuat sebuah tugas, citra kontainer yang ditentukan tidak dapat diambil.

> **Note**
>
> Versi platform 1.4 Fargate memotong pesan kesalahan yang panjang.

Batas waktu koneksi habis

Ketika tugas Fargate diluncurkan, elastic network interface memerlukan rute ke internet untuk menarik gambar kontainer. Jika Anda menerima pesan kesalahan yang mirip dengan berikut ini saat menjalankan tugas, hal itu disebabkan rute menuju internet tidak ada:

```
CannotPullContainerError: API error (500): Get https://111122223333.dkr.ecr.us-
east-1.amazonaws.com/v2/: net/http: request canceled while waiting for connection"
```

Untuk mengatasi masalah ini, Anda dapat:

- Untuk tugas di subnet publik, tentukan DIAKTIFKAN untuk Tetapkan secara otomatis IP publik saat meluncurkan tugas. Untuk informasi selengkapnya, lihat Jalankan aplikasi sebagai tugas Amazon ECS.
- Untuk tugas di subnet privat, tentukan NONAKTIF untuk Tetapkan secara otomatis IP publik saat meluncurkan tugas, dan mengonfigurasi gateway NAT di VPC Anda untuk merutekan permintaan menuju internet. Untuk informasi lebih lanjut, lihat NAT Gateway di Panduan Pengguna Amazon VPC.

Konteks dibatalkan

Penyebab umum kesalahan ini adalah karena VPC yang digunakan tugas Anda tidak memiliki rute untuk menarik gambar penampung dari Amazon ECR.

Citra tidak ditemukan

Saat menentukan image Amazon ECR dalam definisi container, Anda harus menggunakan URI lengkap repositori Amazon ECR beserta nama image di repositori tersebut. Jika repositori atau citra tidak ditemukan, Anda akan menerima kesalahan berikut:

```
CannotPullContainerError: API error (404): repository 111122223333.dkr.ecr.us-
east-1.amazonaws.com/<repo>/<image> not found
```

Untuk mengatasi masalah ini, verifikasi repositori URI dan nama citra. Pastikan juga bahwa Anda telah mengatur akses yang tepat menggunakan peran IAM eksekusi tugas. Untuk informasi selengkapnya tentang peran pelaksanaan tugas, lihat Peran IAM eksekusi tugas Amazon ECS.

Masalah koneksi titik akhir Amazon ECR

Jika Anda mencoba menarik gambar Amazon ECR dan Anda tidak memiliki izin yang benar untuk titik akhir Amazon ECR, Anda melihat kesalahan yang mirip dengan berikut ini:

```
CannotPullContainerError: API error
```

Untuk mengatasi masalah ini, Amazon ECS harus berkomunikasi dengan titik akhir Amazon ECR. Untuk informasi tentang cara mengatasi masalah ini, lihat Bagaimana cara mengatasi kesalahan Amazon ECR "CannotPullContainerError: Kesalahan API" di Amazon ECS di situs web. AWS Support

Ruang disk tak cukup

Jika volume akar instans kontainer Anda memiliki ruang disk yang cukup saat menarik citra kontainer, Anda akan melihat kesalahan yang serupa dengan hal berikut:

```
CannotPullContainerError: write /var/lib/docker/tmp/GetImageBlob111111111: no space
 left on device
```

Untuk mengatasi masalah ini, kosongkan ruang disk.

Jika Anda menggunakan AMI Amazon ECS yang dioptimalkan, Anda dapat menggunakan perintah berikut untuk mengambil 20 file terbesar di sistem file Anda:

```
du -Sh / | sort -rh | head -20
```

Contoh output:

```
5.7G    /var/lib/docker/
containers/50501b5f4cbf90b406e0ca60bf4e6d4ec8f773a6c1d2b451ed8e0195418ad0d2
1.2G    /var/log/ecs
594M    /var/lib/docker/devicemapper/mnt/
c8e3010e36ce4c089bf286a623699f5233097ca126ebd5a700af023a5127633d/rootfs/data/logs
```

```
...
```

Dalam beberapa kasus, mirip dengan contoh sebelumnya, volume root mungkin diisi oleh kontainer yang sedang berjalan. Jika kontainer menggunakan driver log `json-file` default tanpa batas `max-size`, terdapat kemungkinan bahwa berkas log bertanggung jawab pada sebagian besar ruang yang digunakan. Anda dapat menggunakan perintah `docker ps` untuk memverifikasi kontainer yang menggunakan ruang dengan pemetaan nama direktori dari output di atas untuk ID kontainer. Sebagai contoh:

```
CONTAINER ID   IMAGE                                COMMAND             CREATED
    STATUS              PORTS                               NAMES
50501b5f4cbf   amazon/amazon-ecs-agent:latest   "/agent"            4 days ago
    Up 4 days                                               ecs-agent
```

Secara default, saat menggunakan driver log `json-file`, Docker menangkap output standar (dan kesalahan standar) dari semua kontainer Anda dan menulis output standar dan kesalahan dalam file menggunakan format JSON. Anda dapat mengatur `max-size` sebagai pilihan pengandar catatan, yang dapat mencegah berkas log mengambil terlalu banyak ruang. Untuk informasi selengkapnya, lihat Konfigurasi pengandar pencatatan dalam dokumentasi Docker.

Berikut ini adalah cuplikan ketentuan kontainer yang menunjukkan bagaimana menggunakan pilihan ini:

```
{
    "log-driver": "json-file",
    "log-opts": {
        "max-size": "256m"
    }
}
```

Alternatifnya, jika log kontainer Anda menghabiskan terlalu banyak ruang disk, adalah dengan menggunakan driver `awslogs` log. Driver `awslogs` log mengirimkan log ke CloudWatch, yang membebaskan ruang disk yang seharusnya digunakan untuk log kontainer Anda pada instance kontainer. Untuk informasi selengkapnya, lihat Menggunakan driver log awslogs.

Pembatasan laju Docker Hub

Jika Anda menerima salah satu kesalahan berikut, kemungkinan Anda mencapai tingkat batasan Docker Hub:

```
ERROR: toomanyrequests: Too Many Requests.
```

```
You have reached your pull rate limit. You may increase the limit by authenticating
 and upgrading: https://www.docker.com/increase-rate-limits.
```

Untuk informasi selengkapnya tentang tingkat batasan Docker Hub, lihat Memahami pembatasan pada tingkat Docker Hub.

Jika Anda telah meningkatkan batas laju Docker Hub dan Anda perlu mengautentikasi tarikan Docker untuk instance container, lihat Autentikasi registri pribadi untuk instance container di Panduan Pengembang Layanan Amazon Elastic Container.

Gagal menyalin gambar

Jika Anda menerima kesalahan yang mirip dengan berikut ini saat meluncurkan tugas, itu karena tidak ada akses ke gambar:

```
CannotPullContainerError: ref pull has been retried 1 time(s): failed to copy:
 httpReaderSeeker: failed open: unexpected status code
```

Untuk mengatasi masalah ini, Anda dapat:

- Untuk tugas Fargate, lihat Bagaimana cara mengatasi kesalahan “cannotpullcontainererror” untuk tugas Amazon ECS saya di Fargate.
- Untuk tugas lain, lihat Bagaimana cara mengatasi kesalahan “cannotpullcontainererror” untuk tugas Amazon ECS saya.

Akses tarik ditolak

Jika Anda menerima kesalahan yang mirip dengan berikut ini saat meluncurkan tugas, itu karena tidak ada akses ke gambar:

```
CannotPullContainerError: pull access denied
```

Untuk mengatasi masalah ini, Anda mungkin perlu mengautentikasi klien Docker Anda dengan Amazon ECR Untuk informasi selengkapnya, lihat Autentikasi registri pribadi di Panduan Pengguna Amazon ECR.

Alamat memori tidak valid atau dereferensi pointer nil

Jika Anda menerima kesalahan yang mirip dengan berikut ini saat meluncurkan tugas, itu karena tidak ada akses ke gambar:

```
CannotPullContainerError: containerd: pull command failed: panic: runtime error:
 invalid memory address or nil pointer dereference
```

Untuk menyelesaikan masalah ini:

- Periksa apakah Anda memiliki aturan grup keamanan untuk mencapai Amazon S3.
- Saat Anda menggunakan titik akhir gateway, Anda harus menambahkan rute di tabel rute untuk mengakses titik akhir.

Kesalahan saat menarik konfigurasi gambar

Jika Anda menerima kesalahan yang mirip dengan berikut ini saat meluncurkan tugas, itu karena batas kecepatan yang tercapai atau kesalahan jaringan:

```
CannotPullContainerError: error pulling image conf/error pulling image configuration
```

Untuk mengatasi masalah ini, lihat Bagaimana cara mengatasi kesalahan "CannotPullContainerError" di Tugas Jenis Peluncuran Amazon ECS EC2 saya.

Untuk informasi tambahan tentang error STOPTED, lihat Kode kesalahan tugas berhenti di Panduan Pengguna Amazon Elastic Container Service untuk AWS Fargate.

Pesan peristiwa layanan

Ketika memecahkan permasalahan menggunakan layanan, hal pertama yang harus Anda periksa untuk informasi diagnostik adalah pencatatan peristiwa layanan. Anda dapat melihat peristiwa layanan menggunakan `DescribeServices` API, file AWS CLI, atau dengan menggunakan AWS Management Console.

Saat melihat pesan peristiwa layanan menggunakan Amazon ECS API, hanya peristiwa dari penjadwal layanan yang dikembalikan. Hal Ini termasuk penempatan tugas terbaru dan peristiwa kondisi instans. Namun, konsol Amazon ECS menampilkan peristiwa layanan dari sumber berikut.

- Penempatan tugas dan kejadian kesehatan instans dari penjadwal layanan Amazon ECS. Peristiwa ini memiliki awalan layanan ***(nama layanan)***. Untuk memastikan bahwa tampilan peristiwa ini dapat membantu, kami hanya menampilkan peristiwa terbaru `100` dan pesan peristiwa duplikat yang dihilangkan hingga salah satu penyebab sudah diselesaikan atau sudah melewati waktu enam jam. Jika penyebabnya tidak terselesaikan dalam waktu enam jam, Anda menerima pesan acara layanan lain untuk tujuan itu.

- Peristiwa Auto Scaling layanan. Peristiwa ini memiliki awalan Pesan. Peristiwa penskalaan `10` terbaru ditampilkan. Peristiwa ini hanya terjadi ketika layanan dikonfigurasi dengan kebijakan penskalaan Application Auto Scaling.

Gunakan langkah-langkah berikut untuk melihat pesan peristiwa layanan Anda saat ini.

Console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih cluster.
4. Pilih layanan yang akan diperiksa.
5. Pilih Penerapan dan peristiwa, di bawah Acara, lihat pesan.

AWS CLI

Gunakan perintah jelaskan-layanan untuk melihat pesan peristiwa layanan untuk layanan tertentu.

AWS CLI Contoh berikut menjelaskan layanan *`nama layanan`* di cluster *`default`*, yang akan memberikan pesan acara layanan terbaru.

```
aws ecs describe-services \
    --cluster default \
    --services service-name \
    --region us-west-2
```

Pesan peristiwa layanan

Berikut ini adalah contoh pesan acara layanan yang mungkin Anda lihat di konsol Amazon ECS.

Layanan (***`nama layanan`***) telah mencapai status stabil.

Penjadwal layanan mengirimkan acara `service (`*`service-name`*`) has reached a steady state.` layanan ketika layanan sehat dan pada jumlah tugas yang diinginkan, sehingga mencapai kondisi mapan.

Penjadwal layanan melaporkan status secara berkala, sehingga Anda mungkin menerima pesan ini beberapa kali.

Layanan (*nama layanan*) tidak dapat menempatkan tugas dikarenakan tidak ada instans kontainer yang dapat memenuhi semua persyaratan.

Penjadwal layanan mengirimkan pesan peristiwa ini ketika tidak dapat menemukan sumber daya yang tersedia untuk menambahkan tugas lain. Kemungkinan penyebab untuk ini adalah:

Tidak ada instans kontainer yang ditemukan di klaster Anda

Jika tidak ada instance kontainer yang terdaftar di cluster tempat Anda mencoba menjalankan tugas, Anda menerima kesalahan ini. Anda harus menambahkan instans kontainer untuk klaster Anda. Untuk informasi selengkapnya, lihat Meluncurkan instans penampung Amazon ECS Linux.

Port tidak cukup

Jika tugas Anda menggunakan pemetaan host port tetap (misalnya, tugas Anda menggunakan port 80 pada host untuk server web), Anda harus memiliki setidaknya satu instans kontainer per tugas, karena hanya satu kontainer yang dapat menggunakan port host tunggal pada satu waktu. Anda harus menambahkan instans kontainer ke klaster milik Anda atau mengurangi jumlah tugas yang diinginkan.

Terlalu banyak port yang terdaftar

Instance pencocokan kontainer terdekat untuk penempatan tugas tidak dapat melebihi batas port cadangan maksimum yang diizinkan sebesar 100 port host per instance kontainer. Menggunakan pemetaan port host dinamis dapat memperbaiki masalah.

Port sudah digunakan

Definisi tugas tugas ini menggunakan port yang sama dalam pemetaan portnya sebagai tugas yang sudah berjalan pada instance kontainer yang dipilih. Pesan acara layanan akan memiliki ID instance kontainer yang dipilih sebagai bagian dari pesan di bawah ini.

```
The closest matching container-instance is already using a port required by your
 task.
```

Memori tidak cukup

Jika ketentuan tugas Anda menentukan memori sejumlah 1000 MiB, dan masing-masing klaster Anda pada instans kontainer memiliki memori sejumlah 1024 MiB, Anda hanya dapat menjalankan satu salinan tugas ini per instans kontainer. Anda dapat bereksperimen dengan memori yang lebih sedikit pada ketentuan tugas Anda sehingga Anda bisa meluncurkan lebih dari

satu tugas per instans kontainer, atau meluncurkan lebih banyak instans kontainer menuju klaster Anda.

> **Note**
>
> Jika Anda mencoba memaksimalkan pemanfaatan sumber daya Anda dengan menyediakan memori sebanyak mungkin untuk jenis instans tertentu, lihatManajemen memori instance kontainer.

CPU tidak cukup

Sebuah instans kontainer memiliki 1.024 unit CPU untuk setiap inti CPU. Jika ketentuan tugas Anda menentukan 1.000 unit CPU, dan setiap instans kontainer di klaster Anda memiliki 1.024 unit CPU, Anda hanya dapat menjalankan satu salinan tugas ini per instans kontainer. Anda dapat bereksperimen dengan unit CPU yang lebih sedikit dalam ketentuan tugas Anda sehingga Anda bisa meluncurkan lebih dari satu tugas per instans kontainer, atau meluncurkan lebih banyak instans kontainer menuju klaster Anda.

Poin lampiran ENI yang tersedia tidak cukup

Tugas yang menggunakan mode `awsvpc` jaringan masing-masing menerima elastic network interface (ENI) mereka sendiri, yang dilampirkan ke instance container yang menghostingnya. Instans Amazon EC2 memiliki batasan jumlah ENI yang dapat dilampirkan padanya dan tidak ada instance kontainer di cluster yang memiliki kapasitas ENI yang tersedia.

Batas ENI untuk instans kontainer individu tergantung pada kondisi berikut:

- Jika Anda belum menyertakan untuk Pengaturan akun `awsvpcTrunking`, batas ENI untuk setiap instans kontainer tergantung pada tipe instans. Untuk informasi selengkapnya, lihat Alamat IP Per Antarmuka Jaringan Per Tipe Instans dalam Panduan Pengguna Amazon EC2 untuk Instans Linux.
- Jika Anda telah memilih untuk masuk ke setelan `awsvpcTrunking` akun tetapi Anda belum meluncurkan instance container baru menggunakan tipe instans yang didukung setelah memilih, batas ENI untuk setiap instance container masih pada nilai default. Untuk informasi selengkapnya, lihat Alamat IP Per Antarmuka Jaringan Per Tipe Instans dalam Panduan Pengguna Amazon EC2 untuk Instans Linux.
- Jika Anda sudah menyertakan untuk pengaturan akun `awsvpcTrunking` dan Anda telah meluncurkan instans kontainer yang baru dengan menggunakan tipe instans yang didukung

setelah menyertakannya, ENI tambahan akan tersedia. Untuk informasi selengkapnya, lihat Tipe instans Amazon EC2 yang didukung.

Untuk informasi lebih lanjut tentang penyertaan pada pengaturan akun `awsvpcTrunking`, lihat Pembuatan torso antarmuka jaringan elastis.

Anda dapat menambahkan instans kontainer untuk klaster Anda untuk menyediakan lebih banyak adapter jaringan yang tersedia.

Instans kontainer kehilangan atribut wajib

Beberapa parameter ketentuan tugas memerlukan versi API Docker jarak jauh tertentu untuk diinstal pada instans kontainer. Hal lainnya, seperti opsi pencatatan driver, mengharuskan instans kontainer untuk mendaftarkan pencatatan driver tersebut dengan variabel konfigurasi agen `ECS_AVAILABLE_LOGGING_DRIVERS`. Jika definisi tugas Anda berisi parameter yang memerlukan atribut instance kontainer tertentu, dan Anda tidak memiliki instance kontainer yang tersedia yang dapat memenuhi persyaratan ini, tugas tidak dapat ditempatkan.

Penyebab umum kesalahan ini adalah jika layanan Anda menggunakan tugas yang menggunakan mode `awsvpc` jaringan dan jenis peluncuran EC2. Cluster yang Anda tentukan tidak memiliki instance kontainer yang terdaftar di subnet yang sama yang ditentukan pada `awsvpcConfiguration` saat layanan dibuat.

Untuk informasi lebih lanjut tentang atribut yang diperlukan untuk parameter ketentuan tugas tertentu dan variabel konfigurasi agen, lihat Parameter ketentuan tugas dan Konfigurasi agen kontainer Amazon ECS.

Layanan (***`nama layanan`***) tidak dapat menempatkan tugas dikarenakan tidak ada instans kontainer yang dapat memenuhi semua persyaratan. Instance kontainer yang paling cocok tidak ***`container-instance-id`***memiliki unit CPU yang tersedia.

Instance pencocokan kontainer terdekat untuk penempatan tugas tidak berisi unit CPU yang cukup untuk memenuhi persyaratan dalam definisi tugas. Tinjau persyaratan CPU di kedua parameter ukuran tugas dan ketentuan kontainer dari ketentuan tugas.

Layanan (***`nama layanan`***) tidak dapat menempatkan tugas dikarenakan tidak ada instans kontainer yang dapat memenuhi semua persyaratan. Instance kontainer yang paling cocok ***`container-instance-id`***mengalami kesalahan “AGENT”.

Agen kontainer Amazon ECS pada instance kontainer pencocokan terdekat untuk penempatan tugas terputus. Jika Anda dapat terhubung ke instans kontainer dengan SSH, Anda dapat memeriksa

pencatatan agen; untuk informasi lebih lanjut, lihat Log Agen Kontainer Amazon ECS. Anda juga harus memverifikasi bahwa agen sedang berjalan pada instans. Jika Anda menggunakan AMI Amazon ECS yang dioptimalkan, Anda dapat mencoba menghentikan dan memulai ulang agen dengan perintah berikut.

- Untuk Amazon ECS yang dioptimalkan Amazon Linux 2 AMI dan Amazon ECS yang dioptimalkan Amazon Linux 2023 AMI

```
sudo systemctl restart ecs
```

- Untuk Amazon ECS yang dioptimalkan Amazon Linux AMI

```
sudo stop ecs && sudo start ecs
```

service (***service-name***) (instance ***instance-id***) tidak sehat di (elb ***elb-name***) karena (alasan Instance gagal setidaknya jumlah pemeriksaan kesehatan secara berurutan.) UnhealthyThreshold

Layanan ini terdaftar dengan penyeimbang beban dan pemeriksaan kondisi penyeimbang beban gagal. Untuk informasi selengkapnya, lihat Pemecahan permasalahan terhadap layanan penyeimbang beban.

layanan (***nama layanan***) tidak dapat berhasil memulai tugas secara konsisten.

Layanan ini berisi tugas yang gagal untuk dimulai setelah upaya dilakukan secara berturut-turut. Pada titik ini, penjadwal layanan mulai meningkatkan tambahan waktu antara pengulangan. Anda harus memecahkan masalah mengapa tugas Anda gagal untuk diluncurkan. Untuk informasi selengkapnya, lihat Logika throttle layanan Amazon ECS.

Setelah layanan diperbarui, misalnya dengan ketentuan tugas yang diperbarui, penjadwal layanan melanjutkan perilakunya secara normal.

Operasi layanan (***nama layanan***) sedang berada dalam kondisi throttling. Akan di coba lagi nanti.

Layanan ini tidak dapat meluncurkan lebih banyak tugas dikarenakan batas throttling API. Setelah penjadwal layanan dapat meluncurkan lebih banyak tugas, maka akan dilanjutkan.

Untuk meminta kenaikan tingkat batas kuota API, buka halaman AWS Support Pusat, masuk jika diperlukan, dan pilih Buat kasus. Pilih Peningkatan kuota layanan. Lengkapi dan kirimkan formulir ini.

Layanan (***nama layanan***) tidak dapat menghentikan atau memulai tugas selama deployment dikarenakan konfigurasi layanan deployment. Perbarui nilai minimumHealthyPercent atau MaximumPercent dan coba lagi.

Tugas pada layanan ini tidak dapat dihentikan atau dimulai saat layanan deployment dikarenakan konfigurasi pada deployment. Konfigurasi penyebaran terdiri dari `minimumHealthyPercent` dan `maximumPercent` nilai-nilai, yang didefinisikan ketika layanan dibuat. Nilai-nilai tersebut juga dapat diperbarui pada layanan yang ada.

`minimumHealthyPercent`Ini mewakili batas bawah pada jumlah tugas yang harus dijalankan untuk layanan selama penerapan atau ketika instance kontainer terkuras. Ini adalah persen dari jumlah tugas yang diinginkan untuk layanan ini. Nilai ini dibulatkan ke atas. Misalnya, jika persen sehat minimum adalah `50` dan jumlah tugas yang diinginkan adalah empat, maka penjadwal dapat menghentikan dua tugas yang ada sebelum memulai dua tugas baru. Demikian juga, jika persentase minimum yang sehat adalah 75% dan jumlah tugas yang diinginkan adalah dua, maka penjadwal tidak dapat menghentikan tugas apa pun karena nilai yang dihasilkan juga dua.

`maximumPercent`Ini mewakili batas atas jumlah tugas yang harus dijalankan untuk layanan selama penerapan atau saat instance kontainer terkuras. Ini adalah persentase dari jumlah tugas yang diinginkan untuk suatu layanan. Nilai ini dibulatkan ke bawah. Misalnya, jika persentase maksimum adalah `200` dan jumlah tugas yang diinginkan adalah empat, maka penjadwal dapat memulai empat tugas baru sebelum menghentikan empat tugas yang ada. Demikian juga, jika persentase maksimum adalah `125` dan jumlah tugas yang diinginkan adalah tiga, maka penjadwal tidak dapat memulai tugas apa pun karena nilai yang dihasilkan juga tiga.

Ketika menetapkan jumlah minimum persen kondisi atau persen maksimum, Anda harus memastikan bahwa penjadwal dapat menghentikan atau memulai setidaknya satu tugas ketika deployment telah dipicu.

Layanan (***nama layanan***) tidak dapat menempatkan tugas. Alasan: Anda telah mencapai batas jumlah tugas yang dapat Anda jalankan secara bersamaan

Anda dapat meminta peningkatan kuota untuk sumber daya yang menyebabkan kesalahan. Untuk informasi selengkapnya, lihat Kuota layanan. Untuk meminta penambahan kuota, lihat Meminta penambahan kuota di Panduan Pengguna Service Quotas.

Layanan (***nama layanan***) tidak dapat menempatkan tugas. Alasan: Kesalahan internal.

Berikut ini adalah kemungkinan alasan untuk kesalahan ini:

- Layanan tidak dapat memulai tugas karena subnet berada di Availability Zone yang tidak didukung.

  Untuk informasi tentang Wilayah Fargate dan Zona Ketersediaan yang didukung, lihat. the section called “AWS Wilayah Fargate”

  Untuk informasi tentang cara melihat Zona Ketersediaan subnet, lihat Melihat subnet Anda di Panduan Pengguna Amazon VPC.
- Anda mencoba menjalankan definisi tugas yang menggunakan arsitektur ARM di Fargate Spot.

Layanan (***nama layanan***) tidak dapat menempatkan tugas. Alasan: Konfigurasi CPU yang diminta berada di atas batas Anda.

Anda dapat meminta peningkatan kuota untuk sumber daya yang menyebabkan kesalahan. Untuk informasi selengkapnya, lihat Kuota layanan. Untuk meminta penambahan kuota, lihat Meminta penambahan kuota di Panduan Pengguna Service Quotas.

Layanan (***nama layanan***) tidak dapat menempatkan tugas. Alasan: Konfigurasi MEMORY yang diminta berada di atas batas Anda.

Anda dapat meminta peningkatan kuota untuk sumber daya yang menyebabkan kesalahan. Untuk informasi selengkapnya, lihat Kuota layanan. Untuk meminta penambahan kuota, lihat Meminta penambahan kuota di Panduan Pengguna Service Quotas.

Layanan (***nama layanan***) tidak dapat menempatkan tugas. Alasan: Anda telah mencapai batas jumlah vCPU yang dapat Anda jalankan secara bersamaan

AWS Fargate sedang bertransisi dari kuota berbasis hitungan tugas ke kuota berbasis VCPU.

Anda dapat meminta kenaikan kuota untuk kuota berbasis VCPU Fargate. Untuk informasi selengkapnya, lihat Kuota layanan. Untuk meminta peningkatan kuota Fargate, lihat Meminta kenaikan kuota pada Panduan Pengguna Service Quotas.

layanan (***nama layanan***) tidak dapat mencapai kondisi tunak karena set tugas (***TaskSet-ID***) tidak dapat menskalakan. Alasan: Jumlah tugas yang dilindungi lebih dari jumlah tugas yang diinginkan.

Layanan ini memiliki tugas yang lebih terlindungi daripada jumlah tugas yang diinginkan. Anda dapat melakukan salah satu hal berikut:

- Tunggu hingga perlindungan pada tugas saat ini kedaluwarsa, memungkinkan mereka untuk dihentikan.

- Tentukan tugas mana yang dapat dihentikan dan gunakan `UpdateTaskProtection` API dengan `protectionEnabled` opsi yang disetel `false` ke unset protection untuk tugas-tugas ini.
- Tingkatkan jumlah tugas yang diinginkan dari layanan menjadi lebih dari jumlah tugas yang dilindungi.

layanan (***`nama layanan`***) tidak dapat mencapai kondisi mapan. Alasan: Tidak ada Instans Kontainer yang ditemukan di penyedia kapasitas Anda.

Penjadwal layanan mengirimkan pesan peristiwa ini ketika tidak dapat menemukan sumber daya yang tersedia untuk menambahkan tugas lain. Kemungkinan penyebab untuk ini adalah:

Tidak ada instans kontainer yang ditemukan di klaster Anda

Jika tidak ada instance kontainer yang terdaftar di cluster tempat Anda mencoba menjalankan tugas, Anda menerima kesalahan ini. Anda harus menambahkan instans kontainer untuk klaster Anda. Untuk informasi selengkapnya, lihat Meluncurkan instans penampung Amazon ECS Linux.

Port tidak cukup

Jika tugas Anda menggunakan pemetaan port host tetap (misalnya, tugas Anda menggunakan port 80 pada host untuk server web), Anda harus memiliki setidaknya satu instance kontainer per tugas. Hanya satu kontainer yang dapat menggunakan port host tunggal pada satu waktu. Anda harus menambahkan instans kontainer ke klaster milik Anda atau mengurangi jumlah tugas yang diinginkan.

Terlalu banyak port yang terdaftar

Instance pencocokan kontainer terdekat untuk penempatan tugas tidak dapat melebihi batas port cadangan maksimum yang diizinkan sebesar 100 port host per instance kontainer. Menggunakan pemetaan port host dinamis dapat memperbaiki masalah.

Port sudah digunakan

Definisi tugas tugas ini menggunakan port yang sama dalam pemetaan portnya sebagai tugas yang sudah berjalan pada instance kontainer yang dipilih. Pesan acara layanan akan memiliki ID instance kontainer yang dipilih sebagai bagian dari pesan di bawah ini.

```
The closest matching container-instance is already using a port required by your
 task.
```

Memori tidak cukup

Jika ketentuan tugas Anda menentukan memori sejumlah 1000 MiB, dan masing-masing klaster Anda pada instans kontainer memiliki memori sejumlah 1024 MiB, Anda hanya dapat menjalankan satu salinan tugas ini per instans kontainer. Anda dapat bereksperimen dengan memori yang lebih sedikit pada ketentuan tugas Anda sehingga Anda bisa meluncurkan lebih dari satu tugas per instans kontainer, atau meluncurkan lebih banyak instans kontainer menuju klaster Anda.

> **Note**
>
> Jika Anda mencoba untuk memaksimalkan pemanfaatan sumber daya Anda dengan menyediakan memori pada tugas sebanyak mungkin untuk tipe instans tertentu, lihat Manajemen memori instance kontainer.

Poin lampiran ENI yang tersedia tidak cukup

Tugas yang menggunakan mode `awsvpc` jaringan masing-masing menerima elastic network interface (ENI) mereka sendiri, yang dilampirkan ke instance container yang menghostingnya. Instans Amazon EC2 memiliki batasan jumlah ENI yang dapat dilampirkan padanya, dan tidak ada instance kontainer di cluster yang memiliki kapasitas ENI yang tersedia.

Batas ENI untuk instans kontainer individu tergantung pada kondisi berikut:

- Jika Anda belum menyertakan untuk Pengaturan akun `awsvpcTrunking`, batas ENI untuk setiap instans kontainer tergantung pada tipe instans. Untuk informasi selengkapnya, lihat Alamat IP Per Antarmuka Jaringan Per Tipe Instans dalam Panduan Pengguna Amazon EC2 untuk Instans Linux.
- Jika Anda telah memilih untuk masuk ke setelan `awsvpcTrunking` akun tetapi Anda belum meluncurkan instance container baru menggunakan tipe instans yang didukung setelah memilih, batas ENI untuk setiap instance container masih pada nilai default. Untuk informasi selengkapnya, lihat Alamat IP Per Antarmuka Jaringan Per Tipe Instans dalam Panduan Pengguna Amazon EC2 untuk Instans Linux.
- Jika Anda sudah menyertakan untuk pengaturan akun `awsvpcTrunking` dan Anda telah meluncurkan instans kontainer yang baru dengan menggunakan tipe instans yang didukung setelah menyertakannya, ENI tambahan akan tersedia. Untuk informasi selengkapnya, lihat Tipe instans Amazon EC2 yang didukung.

Untuk informasi lebih lanjut tentang penyertaan pada pengaturan akun `awsvpcTrunking`, lihat Pembuatan torso antarmuka jaringan elastis.

Anda dapat menambahkan instans kontainer untuk klaster Anda untuk menyediakan lebih banyak adapter jaringan yang tersedia.

Instans kontainer kehilangan atribut wajib

Beberapa parameter ketentuan tugas memerlukan versi API Docker jarak jauh tertentu untuk diinstal pada instans kontainer. Hal lainnya, seperti opsi pencatatan driver, mengharuskan instans kontainer untuk mendaftarkan pencatatan driver tersebut dengan variabel konfigurasi agen `ECS_AVAILABLE_LOGGING_DRIVERS`. Jika definisi tugas Anda berisi parameter yang memerlukan atribut instance kontainer tertentu, dan Anda tidak memiliki instance kontainer yang tersedia yang dapat memenuhi persyaratan ini, tugas tidak dapat ditempatkan.

Penyebab umum kesalahan ini adalah jika layanan Anda menggunakan tugas yang menggunakan mode `awsvpc` jaringan dan tipe peluncuran EC2 dan cluster yang Anda tentukan tidak memiliki instance kontainer yang terdaftar di subnet yang sama yang ditentukan pada `awsvpcConfiguration` saat layanan dibuat.

Untuk informasi lebih lanjut tentang atribut yang diperlukan untuk parameter ketentuan tugas tertentu dan variabel konfigurasi agen, lihat Parameter ketentuan tugas dan Konfigurasi agen kontainer Amazon ECS.

Layanan (***`nama layanan`***) tidak dapat menempatkan tugas. Alasan: Kapasitas tidak tersedia saat ini. Silakan coba lagi nanti atau di zona ketersediaan yang berbeda.

Saat ini tidak ada kapasitas yang tersedia untuk menjalankan layanan Anda.

Anda dapat melakukan salah satu hal berikut:

- Tunggu hingga kapasitas Fargate atau instans kontainer EC2 tersedia.
- Luncurkan kembali layanan dan tentukan subnet tambahan.

penerapan ***`layanan (nama layanan`***) gagal: tugas gagal dimulai.

Tugas dalam layanan Anda gagal dimulai.

Untuk informasi tentang cara men-debug tugas yang dihentikan. lihat. Kode kesalahan tugas yang berhenti

layanan (*nama layanan) Waktunya* habis menunggu Amazon ECS Agent dimulai. Silakan periksa log di /var/log/ecs/ecs-agent.log".

Agen kontainer Amazon ECS pada instance kontainer pencocokan terdekat untuk penempatan tugas terputus. Jika Anda dapat terhubung ke instance kontainer dengan SSH, Anda dapat memeriksa log agen. Untuk informasi selengkapnya, lihat Log Agen Kontainer Amazon ECS. Anda juga harus memverifikasi bahwa agen sedang berjalan pada instans. Jika Anda menggunakan AMI Amazon ECS yang dioptimalkan, Anda dapat mencoba menghentikan dan memulai ulang agen dengan perintah berikut.

- Untuk Amazon ECS yang dioptimalkan Amazon Linux 2 AMI

```
sudo systemctl restart ecs
```

- Untuk Amazon ECS yang dioptimalkan Amazon Linux AMI

```
sudo stop ecs && sudo start ecs
```

***set tugas service (service-name) (TaskSet-ID) tidak sehat di target-group (targetGroup-ARN)) karena.*TARGET GROUP IS NOT FOUND**

Tugas yang ditetapkan untuk layanan ini gagal dalam pemeriksaan kesehatan karena kelompok sasaran tidak ditemukan. Anda harus menghapus dan membuat ulang layanan. Jangan menghapus grup target Elastic Load Balancing kecuali layanan Amazon ECS yang sesuai telah dihapus.

***set tugas service (service-name) (TaskSet-ID) tidak sehat di target-group (targetGroup-ARN)) karena.*TARGET IS NOT FOUND**

Tugas yang ditetapkan untuk layanan ini gagal dalam pemeriksaan kesehatan karena target tidak ditemukan.

Penentuan nilai CPU atau memori tidak valid

Saat mendaftarkan definisi tugas menggunakan Amazon ECS API atau AWS CLI, jika Anda menentukan `memory` nilai `cpu` atau tidak valid, kesalahan berikut akan ditampilkan.

```
An error occurred (ClientException) when calling the RegisterTaskDefinition operation:
 Invalid 'cpu' setting for task. For more information, see the Troubleshooting section
 of the Amazon ECS Developer Guide.
```

> **Note**
>
> Saat menggunakan Terraform, kesalahan berikut mungkin dikembalikan.
>
> ```
> Error: ClientException: No Fargate configuration exists for given values.
> ```

Untuk mengatasi masalah ini, Anda harus menentukan nilai yang didukung untuk tugas CPU serta memori dalam ketentuan tugas Anda. `cpu`Nilai dapat dinyatakan dalam unit CPU atau vCPU dalam definisi tugas. Ini dikonversi ke bilangan bulat yang menunjukkan unit CPU ketika definisi tugas terdaftar. `memory`Nilai dapat dinyatakan dalam MiB atau GB dalam definisi tugas. Itu dikonversi ke bilangan bulat yang menunjukkan MiB ketika definisi tugas terdaftar.

Untuk ketentuan tugas yang hanya menentukan `EC2` untuk parameter `requiresCompatibilities`, nilai-nilai CPU yang didukung berada di antara Unit CPU `256` (`0.25` vCPUs) dan Unit CPU `16384` (16 VCPUs). Nilai memori harus berupa bilangan bulat, dan batasnya bergantung pada jumlah memori yang tersedia pada instans Amazon EC2 yang mendasari yang Anda gunakan.

Untuk definisi tugas yang menentukan `FARGATE` `requiresCompatibilities` parameter (bahkan jika juga `EC2` ditentukan), Anda harus menggunakan salah satu nilai dalam tabel berikut. Nilai-nilai ini menentukan rentang nilai yang didukung untuk CPU dan parameter memori.

Untuk tugas yang dihosting di Fargate, tabel berikut menunjukkan kombinasi CPU dan memori yang valid. Nilai memori dalam file JSON ditentukan dalam MiB. Anda dapat mengonversi nilai GB ke MiB dengan mengalikan nilainya dengan 1024. Misalnya 1 GB = 1024 MiB.

Nilai CPU	Nilai memori	Sistem operasi yang didukung untuk AWS Fargate
256 (.25 vCPU)	512 MiB, 1 GB, 2 GB	Linux
512 (.5 vCPU)	1 GB, 2 GB, 3 GB, 4 GB	Linux
1024 (1 vCPU)	2 GB, 3 GB, 4 GB, 5 GB, 6 GB, 7 GB, 8 GB	Linux, Windows

Nilai CPU	Nilai memori	Sistem operasi yang didukung untuk AWS Fargate
2048 (2 vCPU)	Antara 4 GB dan 16 GB dalam peningkatan 1 GB	Linux, Windows
4096 (4 vCPU)	Antara 8 GB dan 30 GB dalam peningkatan 1 GB	Linux, Windows
8192 (8 vCPU) **Note** Opsi ini membutuhkan platform Linux `1.4.0` atau yang lebih baru.	Antara 16 GB dan 60 GB dalam peningkatan 4 GB	Linux
16384 (16vCPU) **Note** Opsi ini membutuhkan platform Linux `1.4.0` atau yang lebih baru.	Antara 32 GB dan 120 GB dalam peningkatan 8 GB	Linux

Untuk tugas yang dihosting di Amazon EC2, nilai CPU tugas yang didukung adalah antara 0,25 vCPU dan 192 vCPU.

Note

Tingkat tugas CPU dan memori parameter diabaikan untuk Windows kontainer.

CannotCreateContainerError: API error (500): devmapper

Kesalahan pada Docker berikut ini menunjukkan bahwa penyimpanan kolam yang tipis pada instans kontainer Anda sudah penuh, dan daemon Docker tidak dapat membuat kontainer baru:

```
CannotCreateContainerError: API error (500): devmapper: Thin Pool has 4350 free data
 blocks which is less than minimum required 4454 free data blocks. Create more free
 space in thin pool or use dm.min_free_space option to change behavior
```

Secara default, Amazon ECS mengoptimalkan Amazon Linux AMI dari versi `2015.09.d` dan kemudian diluncurkan dengan volume 8-GiB untuk sistem operasi yang terpasang `/dev/xvda` dan dipasang sebagai root sistem file. Terdapat tambahan volume 22-GiB yang terlampirkan di `/dev/xvdcz` yang Docker gunakan untuk penyimpanan citra dan metadata. Jika ruang penyimpanan ini terisi, daemon Docker tidak dapat membuat kontainer baru.

Cara termudah untuk menambahkan penyimpanan ke instans kontainer Anda adalah dengan mengakhiri instans yang sudah ada dan meluncurkan instans yang baru dengan volume penyimpanan data yang lebih besar. Namun, jika Anda tidak dapat melakukannya, Anda dapat menambahkan penyimpanan ke grup volume yang Docker gunakan dan memperpanjang logika volume dengan mengikuti prosedur di AMI Amazon ECS yang dioptimalkan.

Jika penyimpanan instans kontainer Anda terlalu cepat terisi, ada beberapa tindakan yang dapat Anda ambil untuk mengurangi efek ini:

- Untuk melihat informasi jajak pendapat tipis, jalankan perintah berikut pada instance container Anda:

  ```
  docker info
  ```

- (Agen kontainer Amazon ECS 1.8.0 dan yang lebih baru) Anda dapat mengurangi jumlah waktu penampung yang berhenti atau keluar tetap pada instance penampung Anda. Parameter variabel konfigurasi agen `ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION` menetapkan durasi waktu untuk menunggu dari saat tugas dihentikan hingga kontainer Docker dihapus (secara default, nilai ini adalah 3 jam). Hal ini akan menghapus data kontainer Docker. Jika nilai ini disetel terlalu rendah, Anda mungkin tidak dapat memeriksa kontainer yang dihentikan atau melihat log sebelum dihapus. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.

- Anda dapat menghapus kontainer yang tidak berjalan dan gambar yang tidak digunakan dari instance penampung Anda. Anda dapat menggunakan contoh perintah berikut ini untuk menghapus kontainer yang berhenti secara manual dan citra yang tidak terpakai. Kontainer yang dihapus nantinya tidak dapat diperiksa, dan citra yang dihapus harus ditarik lagi sebelum memulai kontainer yang baru dari citra tersebut.

  Untuk menghapus kontainer yang tidak berjalan, jalankan perintah berikut pada instance container Anda:

  ```
  docker rm $(docker ps -aq)
  ```

  Untuk menghapus gambar yang tidak digunakan, jalankan perintah berikut pada instance container Anda:

  ```
  docker rmi $(docker images -q)
  ```

- Anda dapat menghapus blok data yang tidak digunakan dalam wadah. Anda dapat menggunakan perintah berikut ini untuk menjalankan fstrim pada setiap kontainer yang berjalan dan membuang setiap blok data yang tidak digunakan oleh sistem file kontainer.

  ```
  sudo sh -c "docker ps -q | xargs docker inspect --format='{{ .State.Pid }}' | xargs -IZ fstrim /proc/Z/root/"
  ```

Pemecahan permasalahan terhadap layanan penyeimbang beban

Layanan Amazon ECS dapat mendaftarkan tugas dengan penyeimbang beban Elastic Load Balancing. Kesalahan konfigurasi terhadap penyeimbang beban adalah penyebab umum kenapa tugas berhenti. Jika tugas Anda yang berhenti dimulai oleh layanan yang menggunakan penyeimbang beban, pertimbangkan kemungkinan penyebab berikut ini.

Peran terkait layanan Amazon ECS tidak ada

> Peran terkait layanan Amazon ECS memungkinkan layanan Amazon ECS mendaftarkan instans kontainer dengan penyeimbang beban Elastic Load Balancing. Peran yang terkait dengan layanan harus dibuat di akun Anda. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS.

Grup keamanan contoh kontainer

Jika kontainer Anda dipetakan kepada port 80 di instans kontainer Anda, grup keamanan instans kontainer harus mengizinkan lalu lintas masuk pada port 80 untuk pemeriksaan kondisi penyeimbang beban agar dapat melalui port tersebut.

Penyeimbang beban Elastic Load Balancing tidak dikonfigurasi untuk semua Availability Zone

Penyeimbang beban Anda harus dikonfigurasikan untuk dapat menggunakan semua Availability Zone pada sebuah Wilayah, atau setidaknya semua Availability Zone tempat instans kontainer berada. Jika layanan menggunakan penyeimbang beban dan memulai tugas pada instance container yang berada di Availability Zone yang tidak dikonfigurasi untuk digunakan oleh penyeimbang beban, tugas tersebut tidak akan pernah melewati pemeriksaan kesehatan. Hal ini mengakibatkan tugas terbunuh.

Pemeriksaan kesehatan penyeimbang beban Elastic Load Balancing salah dikonfigurasi

Parameter pemeriksaan kondisi penyeimbang beban dapat terlalu ketat atau mengarah pada sumber daya yang sudah tidak ada. Jika instance kontainer ditentukan tidak sehat, itu dihapus dari penyeimbang beban. Pastikan untuk memverifikasi bahwa parameter berikut ini dikonfigurasi dengan benar untuk layanan penyeimbang beban Anda.

Port Ping

Parameter nilai Port Ping untuk pemeriksaan kondisi penyeimbang beban adalah port pada instans kontainer bahwa pemeriksaan pada penyeimbang beban bertujuan menentukan apakah itu dalam kondisi yang optimal. Jika port ini salah dikonfigurasikan, maka penyeimbang beban memilik kemungkinan untuk mencabut kembali register instans kontainer Anda dari itu sendiri. Port ini perlu dikonfigurasikan untuk dapat menggunakan nilai `hostPort` untuk kontainer dalam definisi layanan tugas yang Anda gunakan dengan pemeriksaan kondisi.

Jalur Ping

Nilai ini sering diatur ke `index.html`, tetapi jika layanan Anda tidak menanggapi permintaan tersebut, maka pemeriksaan kondisi gagal. Jika kontainer Anda tidak memiliki sebuah file `index.html`, Anda dapat mengaturnya menuju `/` untuk menargetkan dasar URL untuk instans kontainer.

Waktu Respons Habis

Ini adalah jumlah waktu yang kontainer Anda harus dapat kembalikan sebuah respons terhadap ping pemeriksaan kondisi. Jika nilai ini lebih rendah dari jumlah waktu yang diperlukan untuk respon, pemeriksaan kondisi gagal.

Interval Pemeriksaan Kondisi

Ini adalah jumlah waktu antara ping pemeriksaan kondisi. Semakin pendek interval pemeriksaan kondisi Anda, semakin cepat instans kontainer Anda dapat mencapai Ambang Tidak Baik.

Batas Kondisi Tidak Baik

Ini merupakan jumlah berapa kali pemeriksaan kondisi Anda bisa gagal sebelum instans kontainer Anda dianggap tidak optimal. Jika Anda memiliki ambang batas 2 yang tidak sehat, dan interval pemeriksaan kesehatan 30 detik, maka tugas Anda memiliki 60 detik untuk menanggapi ping pemeriksaan kesehatan sebelum dianggap tidak sehat. Anda dapat menaikkan ambang batas yang tidak optimal atau interval pemeriksaan kondisi untuk memberikan tugas Anda lebih banyak waktu untuk merespons.

Tidak dapat memperbarui nama layanan`:` *`Nama`* wadah penyeimbang beban atau port diubah dalam definisi tugas

Jika layanan Anda menggunakan penyeimbang beban, Anda dapat menggunakan AWS CLI atau SDK untuk memodifikasi konfigurasi penyeimbang beban. Untuk informasi tentang cara mengubah konfigurasi, lihat UpdateServicedi Referensi API Amazon Elastic Container Service. Jika Anda memperbarui definisi tugas untuk layanan, nama kontainer dan port kontainer yang ditentukan dalam konfigurasi penyeimbang beban harus tetap dalam definisi tugas.

Anda telah mencapai batas jumlah tugas yang dapat Anda jalankan secara bersamaan.

Untuk akun baru, kuota Anda mungkin lebih rendah dari kuota layanan. Kuota layanan untuk akun Anda dapat dilihat di konsol Service Quotas. Untuk meminta penambahan kuota, lihat Meminta penambahan kuota di Panduan Pengguna Service Quotas.

Memecahkan masalah lampiran volume Amazon EBS

Volume Amazon EBS dapat dikonfigurasi untuk lampiran ke tugas Amazon ECS mandiri saat Anda menjalankan tugas. Mereka juga dapat dikonfigurasi untuk lampiran ke tugas yang diluncurkan melalui layanan Amazon ECS saat Anda membuat atau memperbarui layanan. Paling banyak satu volume dapat dilampirkan per tugas. Jika tugas atau layanan Anda tidak diluncurkan sesuai rencana, Anda dapat memeriksa alasan kegagalan lampiran volume menggunakan AWS Management Console. Anda kemudian dapat menggunakan daftar skenario kegagalan untuk langkah selanjutnya. Untuk informasi selengkapnya tentang volume Amazon EBS untuk tugas Amazon ECS, lihat volume Amazon EBS.

Memeriksa alasan kegagalan lampiran volume

Untuk melihat status lampiran volume dan alasan kegagalan

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih klaster tempat tugas Anda berjalan. Halaman detail cluster muncul.
3. Pada halaman detail klaster, pilih tab Tugas.
4. Untuk Filter status yang diinginkan, pilih Berhenti, lalu pilih tugas yang dihentikan yang ingin Anda pecahkan masalah.
5. Pada halaman detail tugas, pilih tab Volume. Anda akan dapat melihat status lampiran volume Amazon EBS di bawah status Lampiran.
6. Pilih status lampiran untuk melihat alasan kegagalan muncul di popover.

Untuk lebih memahami mengapa lampiran volume gagal, Anda dapat mengatur Amazon EventBridge untuk mengirim peristiwa ke target, seperti CloudWatch grup Amazon. Anda dapat mengirim peristiwa volume Amazon EBS dan peristiwa perubahan status tugas Amazon ECS dan menggunakan peristiwa ini untuk mendiagnosis masalah. Untuk informasi selengkapnya, lihat Bagaimana cara membuat grup CloudWatch log untuk digunakan sebagai target EventBridge aturan? di AWS re:post. Untuk informasi selengkapnya tentang peristiwa perubahan status tugas Amazon ECS, lihat Peristiwa perubahan status tugas. Untuk informasi selengkapnya tentang peristiwa volume Amazon EBS, lihat EventBridge Amazon EBS di Panduan Pengguna Amazon EC2.

Skenario kegagalan lampiran volume Amazon EBS

Gunakan skenario berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah yang mungkin Anda temui saat mengonfigurasi volume Amazon EBS untuk lampiran ke tugas Amazon ECS.

Peran infrastruktur ECS AWS Identity and Access Management (IAM)

Skenario peran IAM berikut dapat mengakibatkan masalah lampiran volume Amazon EBS:

- Anda tidak memberikan peran IAM Nama Sumber Daya Amazon (ARN). Untuk melampirkan volume Amazon EBS ke tugas Amazon ECS, Anda harus memberikan ARN untuk peran IAM dengan izin yang diperlukan Amazon ECS untuk mengelola volume Amazon EBS atas nama Anda.

- Anda memberikan peran IAM tanpa kebijakan kepercayaan yang diperlukan terlampir. Amazon ECS tidak dapat mengakses peran IAM infrastruktur Amazon ECS yang Anda berikan jika peran tersebut tidak memiliki kebijakan kepercayaan yang diperlukan. Anda melihat pesan galat berikut: ECS tidak dapat mengasumsikan Peran Infrastruktur ECS yang dikonfigurasi 'arn:aws:iam*111122223333*:role/*ecsInfrastructureRole*'. Harap verifikasi bahwa peran yang dilewati memiliki hubungan kepercayaan yang tepat dengan Amazon ECS. Tugas itu bisa macet di `DEPROVISIONING` negara bagian. Untuk informasi lebih lanjut tentang kebijakan kepercayaan yang diperlukan, lihatPeran IAM infrastruktur Amazon ECS.

  > **Note**
  >
  > Anda juga dapat melihat pesan kesalahan ini karena penundaan dalam propagasi peran. Jika mencoba kembali menggunakan peran setelah menunggu beberapa menit tidak memperbaiki masalah, Anda mungkin telah salah mengonfigurasi kebijakan kepercayaan untuk peran tersebut.

- Pengguna IAM Anda tidak memiliki izin untuk meneruskan peran infrastruktur Amazon ECS ke Amazon ECS. Tugas itu bisa macet di `DEPROVISIONING` negara bagian. Untuk menghindari masalah ini, Anda dapat melampirkan `PassRole` izin ke pengguna Anda. Untuk informasi selengkapnya, lihat Peran IAM infrastruktur Amazon ECS.
- Peran IAM Anda tidak memiliki izin yang diperlukan untuk lampiran volume Amazon EBS. Tugas itu bisa macet di `DEPROVISIONING` negara bagian. Untuk informasi selengkapnya tentang izin khusus yang diperlukan untuk melampirkan volume Amazon EBS ke tugas, lihat. Peran IAM infrastruktur Amazon ECS

AWS Key Management Service (AWS KMS) kunci

Skenario AWS KMS utama berikut dapat menyebabkan masalah lampiran volume Amazon EBS:

- Anda menentukan kunci KMS ARN, ID, atau alias yang tidak valid. Dalam skenario ini, tugas mungkin tampak berhasil diluncurkan, tetapi tugas akhirnya gagal karena AWS mengautentikasi kunci KMS secara asinkron. Untuk informasi selengkapnya, lihat enkripsi Amazon EBS di Panduan Pengguna Amazon EC2.
- Anda memberikan kunci terkelola pelanggan yang tidak memiliki izin yang memungkinkan peran IAM infrastruktur Amazon ECS menggunakan kunci untuk enkripsi. Untuk menghindari masalah izin kebijakan kunci, lihat contoh kebijakan AWS KMS kunci dalam enkripsi data untuk volume Amazon EBS.

Dalam skenario ini, Anda mungkin menemukan pesan alasan status berikut dalam tampilan Volume dari tugas yang dihentikan di AWS Management Console: IdempotentParameterMismatch“; “Token klien yang Anda berikan dikaitkan dengan sumber daya yang sudah dihapus. Silakan gunakan token klien yang berbeda.”

`configuredAtLaunch`parameter definisi tugas

Anda tidak dapat melampirkan volume Amazon EBS ke tugas jika Anda tidak menentukan `configuredAtLaunch` dalam definisi tugas, atau jika Anda `configuredAtLaunch` `false` menyetelnya. Untuk menghindari masalah ini, setel `configuredAtLaunch` ke `true` dalam definisi tugas Anda.

Nama volume

Skenario nama volume berikut dapat mengakibatkan masalah lampiran volume Amazon EBS:

- Anda tidak memberikan nama volume selama konfigurasi volume.
- Nama volume yang Anda berikan selama konfigurasi tidak cocok dengan nama volume yang Anda tentukan dalam definisi tugas.

Format sistem file

Skenario format sistem file berikut dapat mengakibatkan masalah lampiran volume Amazon EBS:

- Format sistem file yang Anda tentukan selama konfigurasi tidak kompatibel dengan sistem operasi tugas.
- Anda mengonfigurasi volume Amazon EBS yang akan dibuat dari snapshot, dan format sistem file snapshot tidak kompatibel dengan sistem operasi tugas. Untuk volume yang dibuat dari snapshot, Anda harus menentukan jenis sistem file yang sama dengan volume yang digunakan saat snapshot dibuat. Jika ada ketidakcocokan tipe sistem file, tugas gagal dimulai dengan alasan status berikut dalam tampilan Volume tugas yang dihentikan di AWS Management Console: Waktu ECS habis saat mengonfigurasi lampiran volume EBS ke Tugas Anda. Untuk memahami masalah ini dengan lebih baik, Anda dapat melihat log agen kontainer Amazon ECS. Untuk informasi selengkapnya, lihat lokasi file log Amazon ECS dan pengumpul log Amazon ECS.

Tag

Skenario penandaan berikut dapat menyebabkan tugas Amazon ECS gagal diluncurkan:

- Anda mencoba menyebarkan tag dari layanan ke volume yang dikonfigurasi untuk lampiran ke tugas mandiri. Untuk menghindari masalah ini, setel `propagateTags` ke `TASK_DEFINITION` jika Anda mengonfigurasi volume untuk lampiran ke tugas mandiri.

- Anda mencoba memberikan jenis sumber daya tag yang tidak valid. Satu-satunya jenis sumber daya tag yang didukung untuk volume Amazon EBS yang dilampirkan ke tugas adalah`volume`.
- Anda mencoba memberikan nilai untuk `propagateTags` itu tidak valid. Nilai yang valid untuk `propagateTags` adalah`TASK_DEFINITION`,`SERVICE`, dan`NONE`.
- Anda memberikan tag yang tidak valid. Untuk informasi selengkapnya, lihat Menandai sumber daya Amazon ECS Anda.
- Anda memberikan terlalu banyak tag. Amazon ECS menambahkan `AmazonECSManaged` dan `AmazonECSCreated` tag secara otomatis ke volume. Ini berarti 48 tag tambahan yang ditentukan pengguna dan disebarkan dapat ditambahkan ke volume dengan total 50 tag maksimum per volume.
- Anda memberikan tag cadangan yang sudah digunakan oleh Amazon ECS seperti `AmazonECSManaged` atau`AmazonECSCreated`.

Batas volume

Jika Anda mencoba mengonfigurasi volume Amazon EBS untuk lampiran ke tugas Amazon ECS tipe peluncuran EC2, dan jika instance container Nitro berbasis yang terkait dengan cluster berada pada batas volume, Anda tidak dapat menjalankan tugas tersebut. Satu volume Amazon EBS dapat dilampirkan per tugas. Batas volume instance container menentukan jumlah tugas dengan volume Amazon EBS yang dikonfigurasi untuk lampiran yang dapat diluncurkan pada instance. Untuk informasi selengkapnya tentang batas volume, lihat Batas volume instans di Panduan Pengguna Amazon EC2.

Jenis contoh kontainer

Anda tidak dapat melampirkan volume Amazon EBS ke tugas tipe peluncuran EC2 yang diluncurkan pada instance container berdasarkan SistemXen. Anda menemukan pesan galat berikut: service (service-name) tidak dapat menempatkan tugas karena tidak ada instance container yang memenuhi semua persyaratannya. Pencocokan terdekat (container-instance instance-id) tidak memiliki atribut yang diperlukan oleh tugas Anda.

Jenis volume Amazon EBS

Anda tidak dapat melampirkan volume Amazon EBS ke tugas jika Anda mengonfigurasi volume Amazon EBS magnetik (`standard`) untuk lampiran ke tugas Fargate.

Throughput Amazon EBS dan `IOPS`

Anda tidak akan dapat melampirkan volume Amazon EBS ke tugas jika Anda mengonfigurasi throughput atau `IOPS` untuk jenis volume yang tidak mendukung parameter ini. Anda `IOPS`

hanya dapat mengonfigurasi untuk volume SSD IOPS (`io1`dan`io2`) dan SSD Tujuan Umum (`gp2`dan`gp3`). Anda dapat mengonfigurasi throughput hanya untuk volume Throughput Optimized HDD (`st1`), Cold HDD (`sc1`), dan General Purpose SSD (). `gp3`

Titik gunung

Untuk menggunakan volume data dengan tugas Amazon ECS Anda, Anda harus mengonfigurasi titik pemasangan dalam definisi tugas Anda. Untuk menghindari kegagalan, pastikan Anda menentukan titik pemasangan dalam definisi tugas Anda dan merujuk ke volume sumber yang ditentukan dalam definisi tugas selama konfigurasi volume.

Wilayah AWS dan Availability Zone

Anda tidak dapat melampirkan volume Amazon EBS ke tugas jika Anda mencoba mengonfigurasi volume Amazon EBS saat penerapan di Zona Ketersediaan Wilayah AWS atau Availability Zone yang tidak mendukung lampiran volume Amazon EBS. Untuk informasi Wilayah AWS dan Availability Zone, lihat Wilayah AWS dan Availability Zone untuk volume Amazon EBS.

AMI untuk tugas Amazon ECS di Amazon EC2

Jika instans penampung host untuk tugas Amazon ECS di klaster Anda tidak memiliki AMI Amazon ECS yang dioptimalkan, atau jika mereka memiliki AMI Amazon ECS yang dioptimalkan yang tidak mendukung lampiran volume Amazon EBS, lampiran volume akan gagal. Lampiran volume Amazon EBS didukung pada AMI `20231219` yang dioptimalkan Amazon ECS dan yang lebih baru.

Kontainer agen

Anda tidak akan dapat melampirkan volume Amazon EBS ke tugas Amazon ECS tipe peluncuran EC2 jika agen penampung yang berjalan pada instance container tidak `1.79.0` atau lebih baru. Anda menemukan pesan galat berikut: service (service-name) tidak dapat menempatkan tugas karena tidak ada instance container yang memenuhi semua persyaratannya. Pencocokan terdekat (container-instance instance-id) tidak memiliki atribut yang diperlukan oleh tugas Anda.

Versi platform Fargate

Anda tidak akan dapat melampirkan volume Amazon EBS ke tugas Amazon ECS tipe peluncuran Fargate jika versi platform Linux Fargate tidak atau lebih baru. `1.4.0`

Kegagalan layanan dan tugas

Anda mungkin mengalami kegagalan layanan atau tugas yang tidak terkait dengan lampiran volume yang dapat memengaruhi lampiran volume. Untuk informasi selengkapnya tentang

pemecahan masalah dengan layanan, lihat Pesan acara layanan. Untuk informasi selengkapnya tentang pemecahan masalah dengan tugas, lihat Kode kesalahan tugas berhenti. Untuk alasan kegagalan API, lihat alasan kegagalan API.

Pemecahan Masalah layanan Auto Scaling

Application Auto Scaling mematikan proses scale-in saat penerapan Amazon ECS sedang berlangsung, dan proses tersebut dilanjutkan setelah penerapan selesai. Namun, proses penskalaan keluar terus terjadi, kecuali ditangguhkan, selama deployment. Untuk informasi lebih lanjut, lihat Menangguhkan dan melanjutkan penskalaan untuk Application Auto Scaling.

Menggunakan output debug Docker

Jika Anda mengalami masalah dengan wadah atau gambar Docker, Anda dapat mengaktifkan mode debug pada daemon Docker Anda. Mengaktifkan debugging memberikan lebih banyak keluaran verbose dari daemon, dan Anda dapat menggunakan informasi ini untuk mengetahui lebih lanjut tentang mengapa wadah atau gambar Anda mengalami masalah.

Mengaktifkan mode debug Docker dapat sangat berguna dalam mengambil pesan kesalahan yang dikirim dari pendaftar kontainer, seperti Amazon ECR, dan, dalam banyak keadaan, mengaktifkan mode debug adalah satu-satunya cara untuk melihat pesan kesalahan ini.

> **Important**
>
> Prosedur ini ditulis untuk Amazon ECS yang dioptimalkan Amazon Linux AMI. Untuk sistem operasi lain, lihat Aktifkan debugging dan Kontrol dan konfigurasikan Docker dengan systemd dokumentasi Docker.

Untuk mengaktifkan mode debug daemon Docker di Amazon ECS Amazon Linux AMI yang dioptimalkan Amazon

1. Hubungkan menuju instans kontainer Anda.
2. Buka file pilihan Docker dengan editor teks, seperti vi. Untuk Amazon ECS yang dioptimalkan Amazon Linux AMI, file opsi Docker ada di. `/etc/sysconfig/docker`
3. Cari pernyataan opsi Docker dan menambahkan pilihan `-D` untuk string, di dalam tanda kutip.

> **Note**
>
> Jika pernyataan pilihan Docker dimulai dengan #, hapus karakter tersebut untuk menghapus pernyataan dan aktifkan pilihan.

Untuk Amazon ECS yang dioptimalkan Amazon Linux AMI, pernyataan opsi Docker disebut. `OPTIONS` Sebagai contoh:

```
# Additional startup options for the Docker daemon, for example:
# OPTIONS="--ip-forward=true --iptables=true"
# By default we limit the number of open files per container
OPTIONS="-D --default-ulimit nofile=1024:4096"
```

4. Simpan file, dan tutup editor teks Anda.
5. Mulai ulang daemon Docker.

```
sudo service docker restart
```

Outputnya adalah sebagai berikut:

```
Stopping docker:                                           [  OK  ]
Starting docker: .                                   [  OK  ]
```

6. Mulai ulang agen Amazon ECS.

```
sudo service ecs restart
```

Catatan Docker Anda sekarang harus menampilkan lebih banyak output verbose.

```
time="2015-12-30T21:48:21.907640838Z" level=debug msg="Unexpected response from
 server: \"{\\\"errors\\\":[{\\\"code\\\":\\\"DENIED\\\",\\\"message\\\":\\\"User:
 arn:aws:sts::1111:assumed-role/ecrReadOnly/i-abcdefg is not authorized to perform:
 ecr:InitiateLayerUpload on resource: arn:aws:ecr:us-east-1:1111:repository/nginx_test
\\\"}]}\\n\" http.Header{\"Connection\":[]string{\"keep-alive\"}, \"Content-Type\":
[]string{\"application/json; charset=utf-8\"}, \"Date\":[]string{\"Wed, 30 Dec 2015
 21:48:21 GMT\"}, \"Docker-Distribution-Api-Version\":[]string{\"registry/2.0\"},
 \"Content-Length\":[]string{\"235\"}}"
```

Lokasi file log Amazon ECS

Amazon ECS menyimpan log di `/var/log/ecs` folder instans kontainer Anda. Ada log yang tersedia dari agen penampung Amazon ECS dan dari `ecs-init` layanan yang mengontrol status agen (start/stop) pada instance container. Anda dapat melihat berkas log ini dengan menghubungkan ke instans kontainer menggunakan SSH.

> **Note**
>
> Jika Anda tidak yakin cara mengumpulkan semua log pada instans kontainer Anda, Anda dapat menggunakan kolektor log Amazon ECS. Untuk informasi selengkapnya, lihat Kolektor log Amazon ECS.

Log Agen Kontainer Amazon ECS

Agen kontainer Amazon ECS menyimpan log pada instans kontainer Anda.

Untuk agen kontainer versi 1.36.0 dan yang lebih baru, secara default log terletak di `/var/log/ecs/ecs-agent.log` pada instans Linux dan di `C:\ProgramData\Amazon\ECS\log\ecs-agent.log` pada instans Windows.

Untuk agen kontainer versi 1.35.0 dan sebelumnya, secara default log terletak di `/var/log/ecs/ecs-agent.log.`*`timestamp`* pada instans Linux dan di `C:\ProgramData\Amazon\ECS\log\ecs-agent.log.`*`timestamp`* pada instans Windows.

Secara default, log agen diputar per jam dengan maksimum 24 log yang disimpan.

Berikut ini adalah variabel konfigurasi agen kontainer yang dapat digunakan untuk mengubah perilaku pencatatan agen default. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.

`ECS_LOGFILE`

Nilai contoh: `/ecs-agent.log`

Nilai default pada Linux: Null

Nilai default pada Windows: Null

Lokasi di mana log agen harus ditulis. Jika Anda menjalankan agen melaluiecs-init, yang merupakan metode default saat menggunakan AMI yang dioptimalkan Amazon ECS, jalur dalam kontainer adalah/log, dan ecs-init memasangnya di host. /var/log/ecs/

ECS_LOGLEVEL

Contoh nilai:crit,error,warn,info, debug

Nilai default pada Linux: info

Nilai default pada Windows: info

Tingkat detail untuk dicatat.

ECS_LOGLEVEL_ON_INSTANCE

Contoh nilai:none,crit,error,warn,info, debug

Nilai default di Linux:none, if secara eksplisit ECS_LOG_DRIVER disetel ke nilai yang tidak kosong; jika tidak, nilai yang sama dengan ECS_LOGLEVEL

Nilai default pada Windows:none, if secara eksplisit ECS_LOG_DRIVER diatur ke nilai yang tidak kosong; jika tidak nilai yang sama dengan ECS_LOGLEVEL

Dapat digunakan untuk mengganti ECS_LOGLEVEL dan mengatur tingkat detail yang harus dicatat dalam file log on-instance, terpisah dari level yang dicatat di driver logging. Jika driver logging disetel secara eksplisit, log on-instance dimatikan secara default. Mereka dapat dihidupkan kembali dengan variabel ini.

ECS_LOG_DRIVER

Contoh nilai:awslogs,fluentd,gelf,json-file,journald, logentriessyslog, splunk

Nilai default pada Linux: json-file

Nilai default pada Windows: Tidak berlaku

Menentukan driver logging yang digunakan kontainer agen.

ECS_LOG_ROLLOVER_TYPE

Contoh nilai:size, hourly

Nilai default pada Linux: hourly

Nilai default pada Windows: `hourly`

Menentukan apakah file log agen kontainer diputar setiap jam atau berdasarkan ukuran. Secara default, file log agen diputar setiap jam.

`ECS_LOG_OUTPUT_FORMAT`

Contoh nilai:`logfmt`, `json`

Nilai default pada Linux: `logfmt`

Nilai default pada Windows: `logfmt`

Menentukan format keluaran log. Ketika `json` format digunakan, setiap baris dalam log adalah peta JSON terstruktur.

`ECS_LOG_MAX_FILE_SIZE_MB`

Nilai contoh: `10`

Nilai default pada Linux: `10`

Nilai default pada Windows: `10`

Ketika `ECS_LOG_ROLLOVER_TYPE` variabel diatur ke`size`, variabel ini menentukan ukuran maksimum (dalam MB) dari file log sebelum diputar. Jika jenis rollover diatur ke`hourly`, maka variabel ini diabaikan.

`ECS_LOG_MAX_ROLL_COUNT`

Nilai contoh: `24`

Nilai default pada Linux: `24`

Nilai default pada Windows: `24`

Menentukan jumlah file log yang diputar untuk disimpan. File log lama dihapus setelah batas ini tercapai.

Untuk agen kontainer versi 1.36.0 dan yang lebih baru, berikut ini adalah contoh berkas log ketika format `logfmt` digunakan.

```
level=info time=2019-12-12T23:43:29Z msg="Loading configuration" module=agent.go
```

```
level=info time=2019-12-12T23:43:29Z msg="Image excluded from cleanup: amazon/amazon-
ecs-agent:latest" module=parse.go
level=info time=2019-12-12T23:43:29Z msg="Image excluded from cleanup: amazon/amazon-
ecs-pause:0.1.0" module=parse.go
level=info time=2019-12-12T23:43:29Z msg="Amazon ECS agent Version: 1.36.0, Commit:
 ca640387" module=agent.go
level=info time=2019-12-12T23:43:29Z msg="Creating root ecs cgroup: /ecs"
 module=init_linux.go
level=info time=2019-12-12T23:43:29Z msg="Creating cgroup /ecs"
 module=cgroup_controller_linux.go
level=info time=2019-12-12T23:43:29Z msg="Loading state!" module=statemanager.go
level=info time=2019-12-12T23:43:29Z msg="Event stream ContainerChange start
 listening..." module=eventstream.go
level=info time=2019-12-12T23:43:29Z msg="Restored cluster 'auto-robc'" module=agent.go
level=info time=2019-12-12T23:43:29Z msg="Restored from checkpoint file. I
 am running as 'arn:aws:ecs:us-west-2:0123456789:container-instance/auto-
robc/3330a8a91d15464ea30662d5840164cd' in cluster 'auto-robc'" module=agent.go
```

Berikut ini contoh berkas log ketika format JSON digunakan.

```
{"time": "2019-11-07T22:52:02Z", "level": "info", "msg": "Starting Amazon Elastic
 Container Service Agent", "module": "engine.go"}
```

Untuk agen kontainer versi 1.35.0 dan sebelumnya, berikut adalah format berkas log.

```
2016-08-15T15:54:41Z [INFO] Starting Agent: Amazon ECS Agent - v1.12.0 (895f3c1)
2016-08-15T15:54:41Z [INFO] Loading configuration
2016-08-15T15:54:41Z [WARN] Invalid value for task cleanup duration, will be overridden
 to 3h0m0s, parsed value 0, minimum threshold 1m0s
2016-08-15T15:54:41Z [INFO] Checkpointing is enabled. Attempting to load state
2016-08-15T15:54:41Z [INFO] Loading state! module="statemanager"
2016-08-15T15:54:41Z [INFO] Detected Docker versions [1.17 1.18 1.19 1.20 1.21 1.22]
2016-08-15T15:54:41Z [INFO] Registering Instance with ECS
2016-08-15T15:54:41Z [INFO] Registered! module="api client"
```

Log Amazon ECS **`ecs-init`**

Proses `ecs-init` menyimpan log di `/var/log/ecs/ecs-init.log`.

Anda dapat menggunakan perintah berikut untuk melihat file log.

```
cat /var/log/ecs/ecs-init.log
```

Output:

```
2018-02-16T18:13:54Z [INFO] pre-start
2018-02-16T18:13:56Z [INFO] start
2018-02-16T18:13:56Z [INFO] No existing agent container to remove.
2018-02-16T18:13:56Z [INFO] Starting Amazon Elastic Container Service Agent
```

Peran IAM untuk Log Audit Kredensyal Tugas

Ketika penyedia kredensyal untuk peran IAM digunakan untuk memberikan kredensyal untuk tugas, permintaan ini disimpan dalam log audit. Log audit menerapkan pengaturan rotasi log yang sama sebagai log agen kontainer. variabel konfigurasi agen kontainer `ECS_LOG_ROLLOVER_TYPE`, `ECS_LOG_MAX_FILE_SIZE_MB`, dan `ECS_LOG_MAX_ROLL_COUNT` dapat diatur untuk mempengaruhi perilaku log audit. Untuk informasi selengkapnya, lihat Log Agen Kontainer Amazon ECS.

Untuk agen kontainer versi 1.36.0 dan yang lebih baru, log audit terletak di `/var/log/ecs/audit.log`. Ketika log diputar, stempel waktu dalam format *YYYY*-*MM*-*DD*-*HH* ditambahkan ke bagian akhir nama berkas log.

Untuk agen kontainer versi 1.35.0 dan sebelumnya, log audit terletak di `/var/log/ecs/audit.log.`*YYYY*-*MM*-*DD*-*HH*.

Berikut adalah format untuk entri log:

- Stempel Waktu
- Kode tanggapan HTTP
- Alamat IP dan nomor port asal permintaan
- URI relatif penyedia kredensial
- Agen pengguna yang membuat permintaan
- ARN tugas milik kontainer yang meminta
- Nama dan nomor versi API `GetCredentials`
- Nama cluster Amazon ECS tempat instance kontainer terdaftar
- ARN instans kontainer

Contoh entri log ditampilkan di bawah.

Anda dapat menggunakan perintah berikut untuk melihat file log.

```
cat /var/log/ecs/audit.log.2016-07-13-16
```

Output:

```
2016-07-13T16:11:53Z 200 172.17.0.5:52444 "/v1/credentials" "python-requests/2.7.0
 CPython/2.7.6 Linux/4.4.14-24.50.amzn1.x86_64" TASK_ARN GetCredentials
 1 CLUSTER_NAME CONTAINER_INSTANCE_ARN
```

Kolektor log Amazon ECS

Jika Anda tidak yakin cara mengumpulkan semua log yang berbeda pada instans kontainer Anda, Anda dapat menggunakan kolektor log Amazon ECS. Ini tersedia GitHub untuk Linux dan Windows. Skrip mengumpulkan log sistem operasi umum serta log agen kontainer Docker dan Amazon ECS, yang dapat membantu untuk memecahkan masalah kasus. AWS Support Kemudian informasi yang dikumpulkan dikompres dan diarsipkan ke dalam satu file yang dapat dengan mudah dibagikan untuk tujuan diagnostik. Ini juga mendukung mengaktifkan mode debug untuk daemon Docker dan agen kontainer Amazon ECS pada varian Amazon Linux, seperti AMI Amazon ECS yang dioptimalkan. Saat ini, kolektor log Amazon ECS mendukung sistem operasi berikut:

- Amazon Linux
- Red Hat Enterprise Linux 7
- Debian 8
- Ubuntu 14.04
- Ubuntu 16.04
- Ubuntu 18.04
- Windows Server 2016

> **Note**
>
> Kode sumber untuk kolektor log Amazon ECS tersedia GitHub untuk Linux dan Windows. Kami menyarankan Anda agar mengirimkan permintaan pull untuk perubahan yang ingin Anda sertakan. Namun, Amazon Web Services saat ini tidak mendukung menjalankan salinan perangkat lunak yang diubah ini.

Untuk mengunduh dan menjalankan kolektor log Amazon ECS untuk Linux

1. Hubungkan menuju instans kontainer Anda.
2. Unduh skrip kolektor log Amazon ECS.

```
curl -O https://raw.githubusercontent.com/awslabs/ecs-logs-collector/master/ecs-logs-collector.sh
```

3. Jalankan penulisan untuk mengumpulkan catatan dan membuat arsip.

> **Note**
>
> Untuk mengaktifkan mode debug untuk daemon Docker dan agen penampung Amazon ECS, tambahkan `--mode=enable-debug` opsi ke perintah berikut. Ini mungkin memulai ulang daemon Docker, yang membunuh semua kontainer yang berjalan pada instance. Pertimbangkan pengurasan pada instans kontainer dan pemindahan tugas-tugas penting untuk instans kontainer lainnya sebelum mengaktifkan mode debug. Untuk informasi selengkapnya, lihat Pengurasan instans kontainer.

```
[ec2-user ~]$ sudo bash ./ecs-logs-collector.sh
```

Setelah Anda menjalankan penulisannya, Anda dapat memeriksa catatan yang dikumpulkan di folder `collect` yang dibuat oleh penulisan. `collect.tgz`File ini adalah arsip terkompresi dari semua log, yang dapat Anda bagikan AWS Support untuk bantuan diagnostik.

Untuk mengunduh dan menjalankan kolektor log Amazon ECS untuk Windows

1. Hubungkan menuju instans kontainer Anda. Untuk informasi selengkapnya, lihat Hubungkan ke Instans Windows Anda dalam Panduan Pengguna Amazon EC2 untuk Instans Windows.
2. Unduh skrip kolektor log Amazon ECS menggunakan PowerShell.

```
Invoke-WebRequest -OutFile ecs-logs-collector.ps1 https://raw.githubusercontent.com/awslabs/aws-ecs-logs-collector-for-windows/master/ecs-logs-collector.ps1
```

3. Jalankan penulisan untuk mengumpulkan catatan dan membuat arsip.

> **Note**
>
> Untuk mengaktifkan mode debug untuk daemon Docker dan agen penampung Amazon ECS, tambahkan `-RunMode debug` opsi ke perintah berikut. Ini akan memulai kembali daemon Docker, yang menghancurkan semua kontainer yang berjalan pada instans. Pertimbangkan pengurasan instans kontainer dan pemindahan tugas-tugas penting untuk instans kontainer lainnya sebelum mengaktifkan mode debug. Untuk informasi selengkapnya, lihat Pengurasan instans kontainer.

```
.\ecs-logs-collector.ps1
```

Setelah Anda menjalankan penulisannya, Anda dapat memeriksa catatan yang dikumpulkan di folder `collect` yang dibuat oleh penulisan. `collect.tgz`File ini adalah arsip terkompresi dari semua log, yang dapat Anda bagikan dengan AWS Support for diagnostic help.

Diagnostik introspeksi Agen

API introspeksi agen Amazon ECS dapat memberikan informasi diagnostik yang bermanfaat. Misalnya, Anda dapat menggunakan API introspeksi agen untuk mendapatkan Docker ID untuk kontainer yang ada dalam tugas Anda. Anda dapat menggunakan API introspeksi agen dengan menghubungkan ke instans kontainer menggunakan SSH.

> **Important**
>
> Instance container Anda harus memiliki peran IAM yang memungkinkan akses ke Amazon ECS untuk mencapai API introspeksi. Untuk informasi selengkapnya, lihat Peran IAM instans wadah Amazon ECS.

Contoh berikut menunjukkan dua tugas, satu yang sedang berjalan dan satu yang dihentikan.

> **Note**
>
> Perintah berikut disalurkan melalui python -mjson.tool untuk keterbacaan yang lebih besar.

```
curl http://localhost:51678/v1/tasks | python -mjson.tool
```

Output:

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1095  100  1095    0     0   117k      0 --:--:-- --:--:-- --:--:--  133k
{
    "Tasks": [
        {
            "Arn": "arn:aws:ecs:us-west-2:aws_account_id:task/090eff9b-1ce3-4db6-848a-
a8d14064fd24",
            "Containers": [
                {
                    "DockerId":
 "189a8ff4b5f04affe40e5160a5ffadca395136eb5faf4950c57963c06f82c76d",
                    "DockerName": "ecs-console-sample-app-static-6-simple-
app-86caf9bcabe3e9c61600",
                    "Name": "simple-app"
                },
                {
                    "DockerId":
 "f7f1f8a7a245c5da83aa92729bd28c6bcb004d1f6a35409e4207e1d34030e966",
                    "DockerName": "ecs-console-sample-app-static-6-busybox-
ce83ce978a87a890ab01",
                    "Name": "busybox"
                }
            ],
            "Family": "console-sample-app-static",
            "KnownStatus": "STOPPED",
            "Version": "6"
        },
        {
            "Arn": "arn:aws:ecs:us-west-2:aws_account_id:task/1810e302-eaea-4da9-
a638-097bea534740",
            "Containers": [
                {
                    "DockerId":
 "dc7240fe892ab233dbbcee5044d95e1456c120dba9a6b56ec513da45c38e3aeb",
                    "DockerName": "ecs-console-sample-app-static-6-simple-app-
f0e5859699a7aecfb101",
                    "Name": "simple-app"
                },
```

```
                {
                    "DockerId":
 "096d685fb85a1ff3e021c8254672ab8497e3c13986b9cf005cbae9460b7b901e",
                    "DockerName": "ecs-console-sample-app-static-6-
busybox-92e4b8d0ecd0cce69a01",
                    "Name": "busybox"
                }
            ],
            "DesiredStatus": "RUNNING",
            "Family": "console-sample-app-static",
            "KnownStatus": "RUNNING",
            "Version": "6"
        }
    ]
}
```

Dalam contoh sebelumnya, tugas berhenti (*090eff9b-1ce3-4db6-848a-a8d14064fd24*) memiliki dua kontainer. Anda dapat menggunakan docker inspect ***container-ID*** untuk melihat informasi detail pada setiap kontainer. Untuk informasi selengkapnya, lihat Introspeksi wadah Amazon ECS.

Diagnosis Docker

Docker menyediakan beberapa alat diagnostik yang membantu Anda memecahkan masalah dengan kontainer dan tugas Anda. Untuk informasi lebih lanjut tentang semua pemanfaatan baris perintah Docker yang sudah tersedia, lihat topik Baris Perintah Docker dalam dokumentasi Docker. Anda dapat mengakses utilitas baris perintah Docker dengan menghubungkan ke instans kontainer menggunakan SSH.

Kode keluar yang Docker kontainer laporkan juga dapat memberikan beberapa informasi diagnostik (misalnya, kode keluar 137 berarti bahwa kontainer menerima sinyal `SIGKILL`). Untuk informasi selengkapnya, lihat Status Keluar dalam dokumentasi Docker.

Cantumkan kontainer Docker

Anda dapat menggunakan perintah docker ps pada instans kontainer Anda untuk membuat daftar pada kontainer yang berjalan. Dalam contoh berikut, hanya agen kontainer Amazon ECS yang berjalan. Untuk informasi selengkapnya, lihat docker ps di dokumentasi Docker.

```
docker ps
```

Output:

```
CONTAINER ID        IMAGE                            COMMAND             CREATED
      STATUS              PORTS                          NAMES
cee0d6986de0        amazon/amazon-ecs-agent:latest   "/agent"            22 hours ago
      Up 22 hours         127.0.0.1:51678->51678/tcp   ecs-agent
```

Anda dapat menggunakan perintah docker ps -a untuk melihat semua kontainer (bahkan berhenti atau menghancurkan kontainer). Hal ini sangat membantu untuk membuat daftar kontainer yang tiba-tiba berhenti. Pada contoh berikut, kontainer `f7f1f8a7a245` keluar 9 detik yang lalu, sehingga tidak muncul dalam output docker ps tanpa bendera `-a`.

```
docker ps -a
```

Output:

```
CONTAINER ID        IMAGE                                              COMMAND
 CREATED             STATUS                          PORTS                          NAMES
db4d48e411b1        amazon/ecs-emptyvolume-base:autogenerated   "not-applicable"
  19 seconds ago                                                                     ecs-
console-sample-app-static-6-internalecs-emptyvolume-source-c09288a6b0cba8a53700
f7f1f8a7a245        busybox:buildroot-2014.02                   "\"sh -c '/bin/sh -c
  22 hours ago        Exited (137) 9 seconds ago                                     ecs-
console-sample-app-static-6-busybox-ce83ce978a87a890ab01
189a8ff4b5f0        httpd:2                                     "httpd-foreground"
  22 hours ago        Exited (137) 40 seconds ago                                    ecs-
console-sample-app-static-6-simple-app-86caf9bcabe3e9c61600
0c7dca9321e3        amazon/ecs-emptyvolume-base:autogenerated   "not-applicable"
  22 hours ago                                                                       ecs-
console-sample-app-static-6-internalecs-emptyvolume-source-90fefaa68498a8a80700
cee0d6986de0        amazon/amazon-ecs-agent:latest              "/agent"
  22 hours ago        Up 22 hours                 127.0.0.1:51678->51678/tcp   ecs-
agent
```

Lihat Log Docker

Anda dapat melihat STDOUT dan mengalirkan STDERR untuk kontainer dengan perintah docker logs. Pada contoh ini, catatan ditampilkan untuk kontainer *dc7240fe892a* dan disalurkan melalui perintah head untuk ringkasnya. Untuk informasi selengkapnya, kunjungi log docker dalam dokumentasi Docker.

> **Note**
>
> Catatan Docker hanya tersedia pada instans kontainer jika Anda menggunakan log driver `json` default. Jika Anda telah mengonfigurasi tugas Anda untuk menggunakan driver `awslogs` log, maka log kontainer Anda tersedia di CloudWatch Log. Untuk informasi selengkapnya, lihat Menggunakan driver log awslogs.

```
docker logs dc7240fe892a | head
```

Output:

```
AH00558: httpd: Could not reliably determine the server's fully qualified domain name,
 using 172.17.0.11. Set the 'ServerName' directive globally to suppress this message
AH00558: httpd: Could not reliably determine the server's fully qualified domain name,
 using 172.17.0.11. Set the 'ServerName' directive globally to suppress this message
[Thu Apr 23 19:48:36.956682 2015] [mpm_event:notice] [pid 1:tid 140327115417472]
 AH00489: Apache/2.4.12 (Unix) configured -- resuming normal operations
[Thu Apr 23 19:48:36.956827 2015] [core:notice] [pid 1:tid 140327115417472] AH00094:
 Command line: 'httpd -D FOREGROUND'
10.0.1.86 - - [23/Apr/2015:19:48:59 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:48:59 +0000] "GET / HTTP/1.1" 200 348
10.0.1.86 - - [23/Apr/2015:19:49:28 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:49:29 +0000] "GET / HTTP/1.1" 200 348
10.0.1.86 - - [23/Apr/2015:19:49:50 +0000] "-" 408 -
10.0.0.154 - - [23/Apr/2015:19:49:50 +0000] "-" 408 -
10.0.1.86 - - [23/Apr/2015:19:49:58 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:49:59 +0000] "GET / HTTP/1.1" 200 348
10.0.1.86 - - [23/Apr/2015:19:50:28 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:50:29 +0000] "GET / HTTP/1.1" 200 348
time="2015-04-23T20:11:20Z" level="fatal" msg="write /dev/stdout: broken pipe"
```

Periksa Kontainer Docker

Jika Anda memiliki ID Docker dari sebuah kontainer, Anda dapat memeriksanya dengan perintah docker inspect. Memeriksa kontainer memberikan pandangan paling detail tentang lingkungan tempat kontainer telah diluncurkan. Untuk informasi selengkapnya, lihat pemeriksaan docker dalam dokumentasi Docker.

```
docker inspect dc7240fe892a
```

Output:

```
[{
    "AppArmorProfile": "",
    "Args": [],
    "Config": {
        "AttachStderr": false,
        "AttachStdin": false,
        "AttachStdout": false,
        "Cmd": [
            "httpd-foreground"
        ],
        "CpuShares": 10,
        "Cpuset": "",
        "Domainname": "",
        "Entrypoint": null,
        "Env": [
            "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/
local/apache2/bin",
            "HTTPD_PREFIX=/usr/local/apache2",
            "HTTPD_VERSION=2.4.12",
            "HTTPD_BZ2_URL=https://www.apache.org/dist/httpd/httpd-2.4.12.tar.bz2"
        ],
        "ExposedPorts": {
            "80/tcp": {}
        },
        "Hostname": "dc7240fe892a",
...
```

AWS Fargate kuota pelambatan

AWS Fargate membatasi tugas Amazon ECS dan tingkat peluncuran pod Amazon EKS ke kuota (sebelumnya disebut sebagai batas) menggunakan algoritma token bucket untuk setiap AWS akun berdasarkan per wilayah. Dengan algoritme ini, akun Anda memiliki bucket yang memegang sejumlah tertentu token. Jumlah token dalam bucket mewakili kuota tarif Anda pada detik tertentu. Setiap akun pelanggan memiliki bucket token task dan pods yang habis berdasarkan jumlah tugas dan pod yang diluncurkan oleh akun pelanggan. Bucket token ini memiliki bucket maksimum yang memungkinkan Anda membuat jumlah permintaan yang lebih tinggi secara berkala, dan tingkat isi ulang yang memungkinkan Anda mempertahankan tingkat permintaan yang stabil selama diperlukan.

Misalnya, ukuran bucket token task dan pods untuk akun pelanggan Fargate adalah 100 token, dan tingkat isi ulang adalah 20 token per detik. Oleh karena itu, Anda dapat segera meluncurkan hingga 100 tugas Amazon ECS dan pod Amazon EKS per akun pelanggan, dengan tingkat peluncuran berkelanjutan 20 tugas Amazon ECS dan pod Amazon EKS per detik.

Tindakan	Kapasitas maksimum bucket (atau Burst rate)	Tingkat isi ulang bucket (atau Tarif berkelanjutan)
Kuota tarif Fargate Resource untuk tugas Amazon ECS On Demand dan pod Amazon EKS [1]	100	20
Kuota tarif Fargate Resource untuk tugas Spot Amazon ECS	100	20

[1] Akun yang hanya meluncurkan pod Amazon EKS memiliki tingkat burst 20 dengan tingkat peluncuran pod berkelanjutan 20 peluncuran pod per detik saat menggunakan versi platform yang disebut dalam versi platform Amazon EKS.

Melambatkan API **RunTask**

Selain itu, Fargate membatasi tingkat permintaan saat meluncurkan tugas menggunakan Amazon ECS `RunTask` API menggunakan kuota terpisah. Fargate membatasi permintaan Amazon ECS `RunTask` API untuk setiap AWS akun berdasarkan per wilayah. Setiap permintaan yang Anda buat menghapus satu token dari ember. Kami melakukan ini untuk membantu kinerja layanan, dan untuk memastikan penggunaan yang adil untuk semua pelanggan Fargate. Panggilan API tunduk pada kuota permintaan apakah berasal dari konsol Amazon Elastic Container Service, alat baris perintah, atau aplikasi pihak ketiga. Kuota tarif untuk panggilan ke Amazon ECS `RunTask` API adalah 20 panggilan per detik (burst dan berkelanjutan). Namun, setiap panggilan ke API ini dapat meluncurkan hingga 10 tugas. Ini berarti Anda dapat meluncurkan 100 tugas dalam satu detik dengan melakukan 10 panggilan ke API ini, meminta 10 tugas untuk diluncurkan di setiap panggilan. Demikian pula, Anda juga dapat membuat 20 panggilan ke API ini, meminta 5 tugas untuk diluncurkan di setiap panggilan. Untuk informasi selengkapnya tentang pembatasan API untuk Amazon ECS `RunTask` API, lihat pembatasan permintaan API di Referensi API Amazon ECS.

Dalam praktiknya, tingkat peluncuran tugas dan pod juga tergantung pada pertimbangan lain seperti gambar kontainer yang akan diunduh dan dibongkar, pemeriksaan kesehatan dan integrasi lainnya

diaktifkan, seperti mendaftarkan tugas atau pod ke penyeimbang beban. Pelanggan melihat variasi dalam tingkat peluncuran tugas dan pod dibandingkan dengan kuota yang diwakili sebelumnya berdasarkan fitur yang diaktifkan pelanggan.

Kuota tingkat penyesuaian

Anda dapat meminta kenaikan kuota pembatasan tarif Fargate untuk akun Anda. AWS Untuk meminta penyesuaian kuota, hubungi AWS Support Pusat.

Alasan kegagalan API

Jika tindakan API yang dipicu melalui Amazon ECS API, konsol, atau AWS CLI keluar dengan pesan `failures` galat, berikut ini mungkin membantu memecahkan masalah penyebabnya. Kegagalan mengembalikan alasan dan Nama Sumber Daya Amazon (ARN) dari sumber daya yang terkait dengan kegagalan.

Banyak sumber daya merupakan wilayah yang khusus, jadi ketika menggunakan konsol tersebut pastikan bahwa Anda mengatur Wilayah yang benar untuk sumber daya Anda. Saat menggunakan AWS CLI, pastikan AWS CLI perintah Anda dikirim ke Wilayah yang benar dengan `--region` *`region`* parameter.

Untuk informasi selengkapnya tentang struktur tipe `Failure` data, lihat Kegagalan dalam Referensi API Amazon Elastic Container Service.

Berikut ini adalah contoh pesan kegagalan yang mungkin Anda terima saat menjalankan perintah API.

Tindakan API	Alasan kegagalan atau alasan berhenti	Penyebab
`DescribeClusters`	`MISSING`	Klaster yang ditentukan tidak ditemukan. Verifikasi ejaan nama klaster.
`DescribeInstances`	`MISSING`	Instans kontainer yang ditentukan tidak ditemukan . Verifikasi bahwa Anda menetapkan klaster instans kontainer yang terdaftar

Tindakan API	Alasan kegagalan atau alasan berhenti	Penyebab
		dan bahwa kedua instans kontainer ARN atau ID adalah benar.
`DescribeServices`	`MISSING`	Layanan yang ditentukan tidak ditemukan. Verifikasi bahwa klaster yang benar atau wilayah yang ditentukan dan bahwa layanan ARN atau nama adalah valid.
`DescribeTasks`	`MISSING`	Tugas yang ditentukan tidak ditemukan. Verifikasi klaster yang benar atau wilayah yang ditentukan dan bahwa kedua ARN tugas atau ID valid.

Tindakan API	Alasan kegagalan atau alasan berhenti	Penyebab
`DescribeTasks`	`TaskFailedToStart: RESOURCE:*`	Untuk `RESOURCE:CPU` kesalahan, jumlah CPU yang diminta oleh tugas tidak tersedia pada instance container Anda. Ini biasanya terjadi ketika persyaratan unit CPU dalam definisi tugas Anda lebih besar dari ukuran CPU instans Amazon EC2 yang ditentukan dalam grup Auto Scaling yang dipetakan ke penyedia kapasitas. Anda perlu memeriksa konfigura si penyedia kapasitas Anda. Untuk informasi tentang cara menambahkan, melihat, dan memodifikasi penyedia kapasitas Anda, lihatthe section called “Penyedia kapasitas”. Untuk `RESOURCE:MEMORY` kesalahan, jumlah memori yang diminta oleh tugas tidak tersedia pada instance penampung Anda. Ini biasanya terjadi ketika persyaratan jumlah memori dalam definisi tugas Anda lebih besar daripada memori yang didukung pada instans Amazon EC2 yang ditentuka n dalam grup Auto Scaling yang dipetakan ke penyedia

Tindakan API	Alasan kegagalan atau alasan berhenti	Penyebab
		kapasitas. Anda perlu memeriksa konfigurasi penyedia kapasitas Anda. Untuk informasi tentang cara menambahkan, melihat, dan memodifikasi penyedia kapasitas Anda, lihatthe section called “Penyedia kapasitas”.
	`TaskFailedToStart: AGENT`	Instans kontainer tempat Anda mencoba untuk meluncurkan tugas memiliki agen yang saat ini terputus. Untuk mencegah waktu tunggu diperpanjang untuk penempatan tugas, permintaan ditolak. Untuk informasi tentang cara memecahkan masalah agen yang terputus, lihat Bagaimana cara memecahkan masalah agen Amazon ECS yang terputus.
	`TaskFailedToStart: MemberOf placement constraint unsatisfied`	Tidak ada instance kontainer yang memenuhi batasan penempatan yang ditentukan dalam definisi tugas Anda.

Tindakan API	Alasan kegagalan atau alasan berhenti	Penyebab
	`TaskFailedToStart: ATTRIBUTE`	Ketentuan tugas Anda berisi parameter yang memerlukan atribut instans kontainer yang spesifik yang tidak tersedia pada instans kontainer Anda. Contohnya, jika tugas Anda menggunakan mode jaringan `awsvpc`, tetapi tidak ada instans dalam subnet yang Anda tentukan dengan atribut `ecs.capability.task-eni` . Untuk informasi lebih lanjut tentang atribut mana yang diperlukan untuk parameter ketentuan tugas tertentu dan variabel konfigurasi agen, lihat Parameter ketentuan tugas dan Konfigurasi agen kontainer Amazon ECS.
	`TaskFailedToStart: NO ACTIVE INSTANCES`	Tidak ada contoh aktif di penyedia kapasitas Anda. Untuk informasi tentang cara menambahkan, melihat, dan memodifikasi penyedia kapasitas Anda, lihatthe section called “Penyedia kapasitas”. Untuk informasi tentang cara mengelola grup Auto Scaling, lihat grup Auto Scaling di Panduan Pengguna Auto Scaling Amazon EC2.

Tindakan API	Alasan kegagalan atau alasan berhenti	Penyebab
	`TaskFailedToStart: EMPTY CAPACITY PROVIDER`	Tidak ada instance di cluster Anda. Ini kemungkinan besar karena penyedia kapasitas kosong, atau karena instance di penyedia kapasitas tidak terdaftar ke cluster. Untuk informasi tentang cara mengelola penyedia kapasitas Anda, lihatPenyedia kapasitas Amazon ECS. Untuk informasi tentang cara mengelola grup Auto Scaling, lihat grup Auto Scaling di Panduan Pengguna Auto Scaling Amazon EC2.
`GetTaskProtection`	`MISSING`	Tugas yang ditentukan tidak ditemukan. Verifikasi bahwa nama cluster atau ARN dan tugas ARN atau ID valid.
	`TASK_NOT_VALID`	Tugas yang ditentukan bukan bagian dari layanan Amazon ECS. Hanya tugas yang dikelola layanan Amazon ECS yang dapat dilindungi. Verifikasi tugas ARN atau ID dan coba lagi.

Tindakan API	Alasan kegagalan atau alasan berhenti	Penyebab
`RunTask` atau `StartTask`	`RESOURCE:*`	Sumber daya tersebut atau sumber daya yang diminta oleh tugas tidak tersedia pada instans kontainer dalam klaster. Jika sumber daya merupakan CPU, memori, port, atau antarmuka jaringan elastis, Anda mungkin perlu menambahkan instans kontainer tambahan untuk klaster Anda. Untuk `RESOURCE:ENI` kesalahan, klaster Anda tidak memiliki titik lampiran elastic network interface yang tersedia, yang diperlukan untuk tugas yang menggunak an mode `awsvpc` jaringan. Instans Amazon EC2 memiliki batasan jumlah antarmuka jaringan yang dapat dilampirk an padanya, dan antarmuka jaringan utama dihitung sebagai satu. Untuk informasi selengkapnya tentang berapa banyak antarmuka jaringan yang didukung untuk setiap jenis instans, lihat Alamat IP Per Antarmuka Jaringan Per Jenis Instance di Panduan Pengguna Amazon EC2 untuk Instans Linux.

Tindakan API	Alasan kegagalan atau alasan berhenti	Penyebab
		Untuk kesalahan `RESOURCE: GPU` , jumlah GPU yang diminta oleh tugas tidak tersedia dan Anda mungkin perlu menambahkan instans kontainer GPU yang dapat diaktifkan menuju klaster Anda. Untuk informasi selengkapnya, lihat Bekerja dengan GPU di Amazon ECS.
	`AGENT`	Instans kontainer tempat Anda mencoba untuk meluncurkan tugas memiliki agen yang saat ini terputus. Untuk mencegah waktu tunggu diperpanjang untuk penempatan tugas, permintaan ditolak. Untuk informasi tentang cara memecahkan masalah agen yang terputus, lihat Bagaimana cara memecahkan masalah agen Amazon ECS yang terputus.
	`LOCATION`	Instans kontainer tempat Anda coba untuk meluncurkan tugas ada di Availability Zone yang berbeda dari subnet yang Anda tentukan di `awsVpcConfiguration` .

Tindakan API	Alasan kegagalan atau alasan berhenti	Penyebab
	`ATTRIBUTE`	Ketentuan tugas Anda berisi parameter yang memerlukan atribut instans kontainer yang spesifik yang tidak tersedia pada instans kontainer Anda. Contohnya, jika tugas Anda menggunakan mode jaringan `awsvpc`, tetapi tidak ada instans dalam subnet yang Anda tentukan dengan atribut `ecs.capability.task-eni` . Untuk informasi lebih lanjut tentang atribut mana yang diperlukan untuk parameter ketentuan tugas tertentu dan variabel konfigurasi agen, lihat Parameter ketentuan tugas dan Konfigurasi agen kontainer Amazon ECS.
`StartTask`	`MISSING`	Instance container yang Anda coba untuk meluncurkan tugas ke tidak dapat ditemukan. Periksa apakah cluster atau Region yang salah ditentukan, atau contoh kontainer ARN atau ID salah eja.
	`INACTIVE`	Instance container yang Anda coba untuk meluncurkan tugas sebelumnya telah dideregistrasi dengan Amazon ECS dan tidak dapat digunakan.

Tindakan API	Alasan kegagalan atau alasan berhenti	Penyebab
`UpdateTaskProtection`	`DEPLOYMENT_BLOCKED`	Tidak dapat mengatur perlindungan tugas karena satu atau lebih tugas yang dilindungi mencegah penyebaran layanan mencapai kondisi mapan. Hapus perlindungan tugas pada tugas yang ada atau tunggu hingga perlindungan tugas kedaluwarsa.
	`MISSING`	Tugas yang ditentukan tidak ditemukan. Verifikasi bahwa nama cluster atau ARN dan tugas ARN atau ID valid.
	`TASK_NOT_VALID`	Tugas yang ditentukan bukan bagian dari layanan Amazon ECS. Hanya tugas yang dikelola layanan Amazon ECS yang dapat dilindungi. Verifikasi tugas ARN atau ID dan coba lagi.

Note

Selain skenario kegagalan yang dijelaskan di sini, operasi API juga dapat gagal karena pengecualian, yang mengakibatkan respons kesalahan. Untuk daftar pengecualian tersebut, lihat Kesalahan Umum.

Praktik terbaik untuk menangani masalah pelambatan Amazon ECS

Kesalahan pelambatan terbagi dalam dua kategori utama: throttling sinkron dan pelambatan asinkron.

Pelambatan sinkron

Saat pelambatan sinkron terjadi, Anda segera menerima respons kesalahan dari Amazon ECS. Kategori pembatasan ini biasanya terjadi saat Anda memanggil Amazon ECS API saat menjalankan tugas atau membuat layanan. Untuk informasi selengkapnya tentang pembatasan yang terlibat dan batas throttle yang relevan, lihat Meminta pembatasan untuk Amazon ECS API.

Saat aplikasi Anda memulai permintaan API, misalnya, dengan menggunakan AWS CLI atau AWS SDK, Anda dapat memulihkan pelambatan API. Anda dapat melakukan ini dengan merancang aplikasi Anda untuk menangani kesalahan atau dengan menerapkan strategi backoff dan jitter eksponensial dengan logika coba lagi untuk panggilan API. Untuk informasi selengkapnya, lihat Timeout, percobaan ulang, dan backoff dengan jitter.

Jika Anda menggunakan AWS SDK, logika coba ulang otomatis sudah built-in dan dapat dikonfigurasi.

Pelambatan asinkron

Pelambatan asinkron terjadi karena alur kerja asinkron di mana Amazon ECS atau AWS CloudFormation mungkin memanggil API atas nama Anda untuk menyediakan sumber daya. Penting untuk mengetahui AWS API mana yang digunakan Amazon ECS atas nama Anda. Misalnya, `CreateNetworkInterface` API dipanggil untuk tugas yang menggunakan mode `awsvpc` jaringan, dan `DescribeTargetHealth` API dipanggil saat melakukan pemeriksaan kesehatan untuk tugas yang terdaftar ke penyeimbang beban.

Ketika beban kerja Anda mencapai skala yang cukup besar, operasi API ini mungkin akan dibatasi. Artinya, mereka mungkin cukup dibatasi untuk melanggar batas yang diberlakukan oleh Amazon ECS atau Layanan AWS yang sedang disebut. Misalnya, jika Anda menerapkan ratusan layanan, masing-masing memiliki ratusan tugas secara bersamaan yang menggunakan mode `awsvpc` jaringan, Amazon ECS akan memanggil operasi Amazon EC2 API seperti dan operasi API Elastic `CreateNetworkInterface` Load Balancing seperti atau untuk `RegisterTarget` mendaftarkan `DescribeTargetHealth` interface network elastis dan penyeimbang beban, masing-masing.

Panggilan API ini dapat melebihi batas API, yang mengakibatkan kesalahan pelambatan. Berikut ini adalah contoh kesalahan pelambatan Elastic Load Balancing yang disertakan dalam pesan acara layanan.

```
{
   "userIdentity":{
      "arn":"arn:aws:sts::111122223333:assumed-role/AWSServiceRoleForECS/ecs-service-
scheduler",
      "eventTime":"2022-03-21T08:11:24Z",
      "eventSource":"elasticloadbalancing.amazonaws.com",
      "eventName":" DescribeTargetHealth ",
      "awsRegion":"us-east-1",
      "sourceIPAddress":"ecs.amazonaws.com",
      "userAgent":"ecs.amazonaws.com",
      "errorCode":"ThrottlingException",
      "errorMessage":"Rate exceeded",
      "eventID":"0aeb38fc-229b-4912-8b0d-2e8315193e9c"
   }
}
```

Ketika panggilan API ini berbagi batas dengan lalu lintas API lain di akun Anda, mereka mungkin sulit dipantau meskipun dipancarkan sebagai peristiwa layanan.

Pemantauan pelambatan

Penting untuk mengidentifikasi permintaan API mana yang dibatasi dan siapa yang mengeluarkan permintaan ini. Anda dapat menggunakan monitor AWS CloudTrail mana yang membatasi, dan terintegrasi dengan, Amazon CloudWatch Athena, dan Amazon. EventBridge Anda dapat mengonfigurasi CloudTrail untuk mengirim peristiwa tertentu ke CloudWatch Log. CloudWatch Wawasan log log mengurai dan menganalisis peristiwa. Ini mengidentifikasi detail dalam peristiwa pelambatan seperti peran pengguna atau IAM yang melakukan panggilan dan jumlah panggilan API yang dilakukan. Untuk informasi selengkapnya, lihat Memantau file CloudTrail CloudWatch log dengan Log.

Untuk informasi selengkapnya tentang wawasan CloudWatch Log dan petunjuk tentang cara menanyakan file log, lihat Menganalisis data CloudWatch log dengan Wawasan Log.

Dengan Amazon Athena, Anda dapat membuat kueri dan menganalisis data menggunakan SQL standar. Misalnya, Anda dapat membuat tabel Athena untuk mengurai CloudTrail acara. Untuk informasi selengkapnya, lihat Menggunakan CloudTrail konsol untuk membuat tabel Athena untuk CloudTrail log.

Setelah membuat tabel Athena, Anda dapat menggunakan kueri SQL sederhana seperti yang berikut untuk menyelidiki kesalahan. `ThrottlingException`

```
select eventname, errorcode,eventsource,awsregion, useragent,COUNT(*) count
FROM cloudtrail-table-name
where errorcode = 'ThrottlingException'
AND eventtime between '2022-01-14T03:00:08Z' and '2022-01-23T07:15:08Z'
group by errorcode, awsregion, eventsource, username, eventname
order by count desc;
```

Amazon ECS juga memancarkan pemberitahuan acara ke Amazon. EventBridge Ada peristiwa perubahan status sumber daya dan peristiwa tindakan layanan. Mereka termasuk peristiwa pelambatan API seperti `ECS_OPERATION_THROTTLED` dan. `SERVICE_DISCOVERY_OPERATION_THROTTLED` Untuk informasi selengkapnya, lihat Acara tindakan layanan Amazon ECS.

Peristiwa ini dapat dikonsumsi oleh layanan seperti AWS Lambda untuk melakukan tindakan sebagai tanggapan. Untuk informasi selengkapnya, lihat Menangani acara Amazon ECS.

Jika Anda menjalankan tugas mandiri, beberapa operasi API seperti `RunTask` asinkron, dan operasi coba lagi tidak dilakukan secara otomatis. Dalam kasus seperti itu, Anda dapat menggunakan layanan seperti AWS Step Functions EventBridge integrasi untuk mencoba kembali operasi yang dibatasi atau gagal. Untuk informasi selengkapnya, lihat Mengelola tugas penampung (Amazon ECS, Amazon SNS).

Menggunakan CloudWatch untuk memantau pelambatan

CloudWatch menawarkan pemantauan penggunaan API pada `Usage` namespace di bawah By AWS Resource. Metrik ini dicatat dengan API tipe dan nama CallCountmetrik. Anda dapat membuat alarm untuk memulai setiap kali metrik ini mencapai ambang batas tertentu. Untuk informasi selengkapnya, lihat Memvisualisasikan kuota layanan Anda dan menyetel alarm.

CloudWatch juga menawarkan deteksi anomali. Fitur ini menggunakan pembelajaran mesin untuk menganalisis dan menetapkan garis dasar berdasarkan perilaku tertentu dari metrik yang Anda aktifkan. Jika ada aktivitas API yang tidak biasa, Anda dapat menggunakan fitur ini bersama dengan CloudWatch alarm. Untuk informasi lebih lanjut, lihat Menggunakan CloudWatch deteksi anomali.

Dengan secara proaktif memantau kesalahan pelambatan, Anda dapat menghubungi AWS Support untuk meningkatkan batas pelambatan yang relevan dan juga menerima panduan untuk kebutuhan aplikasi unik Anda.

Referensi parameter dan templat sumber daya

Bagian berikut menjelaskan definisi tugas dan parameter definisi layanan.

Topik

Parameter ketentuan tugas

Definisi tugas dibagi menjadi beberapa bagian terpisah: keluarga tugas, peran tugas AWS Identity and Access Management (IAM), mode jaringan, definisi wadah, volume, batasan penempatan tugas, dan jenis peluncuran. Definisi keluarga dan wadah diperlukan dalam definisi tugas. Sebaliknya, peran tugas, mode jaringan, volume, kendala penempatan tugas, dan jenis peluncuran bersifat opsional.

Anda dapat menggunakan parameter ini dalam file JSON untuk mengonfigurasi definisi tugas Anda. Untuk informasi selengkapnya, lihat the section called “Contoh definisi tugas opsi logging”.

Berikut ini adalah deskripsi yang lebih detail untuk setiap parameter ketentuan tugas.

Rangkaian

`family`

Tipe: String

Diperlukan: Ya

Ketika Anda mendaftarkan ketentuan tugas, Anda memberikan sebuah famili, yang mirip dengan nama untuk beberapa versi ketentuan tugas, yang ditentukan dengan nomor revisi. Definisi tugas pertama yang terdaftar ke dalam keluarga tertentu diberikan revisi 1, dan definisi tugas apa pun yang terdaftar setelah itu diberi nomor revisi berurutan.

Jenis peluncuran

Saat mendaftarkan definisi tugas, Anda dapat menentukan jenis peluncuran yang harus divalidasi oleh Amazon ECS terhadap definisi tugas. Jika definisi tugas tidak memvalidasi terhadap

kompatibilitas yang ditentukan, pengecualian klien dikembalikan. Untuk informasi selengkapnya, lihat Jenis peluncuran Amazon ECS.

Parameter berikut diperbolehkan dalam definisi tugas.

`requiresCompatibilities`

Tipe: Array string

Diperlukan: Tidak

Nilai yang Valid: `EC2` | `FARGATE` | `EXTERNAL`

Tipe peluncuran untuk memvalidasi ketentuan tugas. Ini memulai pemeriksaan untuk memastikan bahwa semua parameter yang digunakan dalam definisi tugas memenuhi persyaratan jenis peluncuran.

Peran tugas

`taskRoleArn`

Tipe: String

Wajib: Tidak

Ketika mendaftarkan ketentuan tugas, Anda dapat memberikan peran tugas untuk IAM role yang mengizinkan kontainer dalam izin tugas untuk atas nama Anda memanggil API AWS yang ditentukan dalam kebijakan terkait. Untuk informasi selengkapnya, lihat Tugas peran IAM.

Saat Anda meluncurkan AMI Windows Server Amazon ECS yang dioptimalkan, peran IAM untuk tugas di Windows mengharuskan `-EnableTaskIAMRole` opsi disetel. Container Anda juga harus menjalankan beberapa kode konfigurasi untuk menggunakan fitur tersebut. Untuk informasi selengkapnya, lihat Konfigurasi tambahan untuk peran Windows IAM untuk tugas.

Peran eksekusi tugas

`executionRoleArn`

Tipe: String

Wajib: Bersyarat

Nama Sumber Daya Amazon (ARN) dari peran eksekusi tugas yang memberikan izin agen penampung Amazon ECS untuk melakukan panggilan AWS API atas nama Anda.

> **Note**
>
> IAM role eksekusi tugas diperlukan sesuai dengan persyaratan tugas Anda. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

Mode jaringan

`networkMode`

Tipe: String

Wajib: Tidak

Mode jaringan Docker digunakan untuk kontainer dalam tugas. Untuk tugas Amazon ECS yang di-host di instans Amazon EC2 Linux, nilai yang valid `none` adalah,,`bridge`, `awsvpc` dan. `host` Jika tidak ada mode jaringan yang ditentukan, mode jaringan default adalah `bridge`. Untuk tugas Amazon ECS yang dihosting di instans Windows Amazon EC2, nilai `default` yang valid adalah, dan. `awsvpc` Jika tidak ada mode jaringan yang ditentukan, mode jaringan `default` digunakan. Untuk tugas Amazon ECS yang dihosting di Fargate, `awsvpc` mode jaringan diperlukan.

Jika mode jaringan disetel ke`none`, kontainer tugas tidak memiliki konektivitas eksternal dan pemetaan port tidak dapat ditentukan dalam definisi kontainer.

Jika mode jaringan`bridge`, tugas menggunakan jaringan virtual bawaan Docker di Linux, yang berjalan di dalam setiap instans Amazon EC2 yang menghosting tugas. Jaringan virtual bawaan di Linux menggunakan driver jaringan `bridge` Docker.

Jika mode jaringan`host`, tugas menggunakan jaringan host yang melewati jaringan virtual bawaan Docker dengan memetakan port kontainer langsung ke ENI dari instans Amazon EC2 yang menghosting tugas. Pemetaan port dinamis tidak dapat digunakan dalam mode jaringan ini. Wadah dalam definisi tugas yang menggunakan mode ini harus menentukan `hostPort` nomor tertentu. Nomor port pada host tidak dapat digunakan oleh banyak tugas. Akibatnya, Anda tidak dapat menjalankan beberapa tugas dengan definisi tugas yang sama pada satu instans Amazon EC2.

> **Important**
>
> Saat menjalankan tugas yang menggunakan mode `host` jaringan, jangan jalankan kontainer menggunakan pengguna root (UID 0) untuk keamanan yang lebih baik. Sebagai praktik terbaik keamanan, selalu gunakan pengguna non-root.

Untuk jenis peluncuran Amazon EC2, jika mode jaringan`awsvpc`, tugas dialokasikan sebagai elastic network interface, dan Anda harus menentukan `NetworkConfiguration` kapan Anda membuat layanan atau menjalankan tugas dengan definisi tugas. Untuk informasi selengkapnya, lihat Jaringan tugas untuk tugas di instans Amazon EC2.

Jika mode jaringan`default`, tugas menggunakan jaringan virtual bawaan Docker di Windows, yang berjalan di dalam setiap instans Amazon EC2 yang menghosting tugas. Jaringan virtual bawaan pada Windows menggunakan driver jaringan `nat` Docker.

Untuk jenis peluncuran Fargate, ketika mode jaringan`awsvpc`, tugas dialokasikan sebuah elastic network interface, dan Anda harus menentukan `NetworkConfiguration` kapan Anda membuat layanan atau menjalankan tugas dengan definisi tugas. Untuk informasi selengkapnya, lihat Jaringan Tugas Fargate. Mode `awsvpc` jaringan menawarkan kinerja jaringan tertinggi untuk kontainer karena mereka menggunakan tumpukan jaringan Amazon EC2. Port kontainer yang terbuka dipetakan langsung ke port antarmuka elastic network yang terpasang. Karena itu, Anda tidak dapat menggunakan pemetaan port host dinamis.

Mode `host` dan `awsvpc` jaringan menawarkan kinerja jaringan tertinggi untuk kontainer karena mereka menggunakan tumpukan jaringan Amazon EC2. Dengan mode `host` dan `awsvpc` jaringan, port kontainer yang terbuka dipetakan langsung ke port host yang sesuai (untuk mode `host` jaringan) atau port antarmuka elastis network yang terpasang (untuk mode `awsvpc` jaringan). Karena itu, Anda tidak dapat menggunakan pemetaan port host dinamis.

Jika menggunakan tipe peluncuran Fargate, mode `awsvpc` jaringan diperlukan. Jika menggunakan tipe peluncuran EC2, mode jaringan yang diizinkan bergantung pada sistem operasi instans EC2 yang mendasarinya. Jika Linux, mode jaringan apa pun bisa digunakan. Jika Windows, hanya dapat menggunakan mode jaringan `default`, dan `awsvpc`.

Platform runtime

`operatingSystemFamily`

Tipe: String

Wajib: Bersyarat

Standar: LINUX

Parameter ini diperlukan untuk tugas Amazon ECS yang di-host di Fargate.

Saat Anda mendaftarkan definisi tugas, Anda menentukan keluarga sistem operasi.

Nilai valid untuk tugas Amazon ECS yang di-host di Fargate `LINUX` adalah`WINDOWS_SERVER_2019_FULL`,,, `WINDOWS_SERVER_2019_COREWINDOWS_SERVER_2022_FULL`, dan. `WINDOWS_SERVER_2022_CORE`

Nilai valid untuk tugas Amazon ECS yang dihosting di EC2 adalah`LINUX`,`WINDOWS_SERVER_2022_CORE`,`WINDOWS_SERVER_2022_FULL`,`WINDOWS_SERVER_201` dan`WINDOWS_SERVER_2019_CORE`, `WINDOWS_SERVER_2016_FULLWINDOWS_SERVER_2004_CORE`, dan. `WINDOWS_SERVER_20H2_CORE`

Semua definisi tugas yang digunakan dalam layanan harus memiliki nilai yang sama untuk parameter ini.

Ketika definisi tugas adalah bagian dari layanan, nilai ini harus sesuai dengan `platformFamily` nilai layanan.

`cpuArchitecture`

Tipe: String

Wajib: Bersyarat

Standar: X86_64

Parameter ini diperlukan untuk tugas Amazon ECS yang dihosting di Fargate. Jika parameter dibiarkan sebagai`null`, nilai default secara otomatis ditetapkan pada inisiasi tugas yang dihosting di Fargate.

Saat Anda mendaftarkan definisi tugas, Anda menentukan arsitektur CPU. Nilai yang valid adalah `X86_64` dan `ARM64`.

Semua definisi tugas yang digunakan dalam layanan harus memiliki nilai yang sama untuk parameter ini.

Ketika Anda memiliki tugas Linux baik untuk jenis peluncuran Fargate, atau tipe peluncuran EC2, Anda dapat mengatur nilainya. `ARM64` Untuk informasi selengkapnya, lihat the section called "Bekerja dengan beban kerja ARM 64-bit di Amazon ECS".

Ukuran tugas

Saat Anda mendaftarkan definisi tugas, Anda dapat menentukan total CPU dan memori yang digunakan untuk tugas tersebut. Hal ini terpisah dari nilai `cpu` dan `memory` pada tingkat ketentuan kontainer. Untuk tugas yang di-host di instans Amazon EC2, bidang ini bersifat opsional. Untuk tugas yang di-host di Fargate (baik Linux dan Windows), bidang ini diperlukan dan ada nilai khusus untuk keduanya `cpu` dan `memory` yang didukung.

> **Note**
>
> Tingkat tugas CPU dan memori parameter diabaikan untuk Windows kontainer. Kami merekomendasikan agar Anda menentukan sumber daya tingkat kontainer untuk kontainer Windows.

Parameter berikut diizinkan dalam ketentuan tugas:

`cpu`

Tipe: String

Wajib: Bersyarat

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

Batas tegas unit CPU untuk hadir untuk tugas tersebut. Ini dapat diekspresikan menggunakan unit CPU (misalnya,`1024`) atau menggunakan vCPU (misalnya`1 vCPU`, `1 vcpu` atau) dalam definisi

tugas. Ketika ketentuan tugas terdaftar, nilai vCPU dikonversi ke integer yang menunjukkan unit CPU.

Untuk tugas yang berjalan pada EC2 atau instans eksternal, bidang ini bersifat opsional. Jika klaster Anda tidak memiliki instance kontainer terdaftar dengan unit CPU yang diminta tersedia, tugas gagal. Nilai yang didukung untuk tugas yang berjalan pada EC2 atau instance eksternal adalah antara vCPU dan vCPU`0.125`. `10`

Untuk tugas yang berjalan di Fargate (baik wadah Linux dan Windows), bidang ini diperlukan dan Anda harus menggunakan salah satu nilai berikut, yang menentukan rentang nilai yang didukung untuk parameter. `memory` Tabel berikut menunjukkan kombinasi yang valid antara CPU dan memori tingkat tugas.

Nilai CPU	Nilai memori	Sistem operasi yang didukung untuk AWS Fargate
256 (.25 vCPU)	512 MiB, 1 GB, 2 GB	Linux
512 (.5 vCPU)	1 GB, 2 GB, 3 GB, 4 GB	Linux
1024 (1 vCPU)	2 GB, 3 GB, 4 GB, 5 GB, 6 GB, 7 GB, 8 GB	Linux, Windows
2048 (2 vCPU)	Antara 4 GB dan 16 GB dalam peningkatan 1 GB	Linux, Windows
4096 (4 vCPU)	Antara 8 GB dan 30 GB dalam peningkatan 1 GB	Linux, Windows
8192 (8 vCPU) **Note** Opsi ini membutuhkan platform Linux `1.4.0` atau yang lebih baru.	Antara 16 GB dan 60 GB dalam peningkatan 4 GB	Linux

Nilai CPU	Nilai memori	Sistem operasi yang didukung untuk AWS Fargate
16384 (16vCPU) **Note** Opsi ini membutuhkan platform Linux `1.4.0` atau yang lebih baru.	Antara 32 GB dan 120 GB dalam peningkatan 8 GB	Linux

`memory`

Tipe: String

Wajib: Bersyarat

Note

Parameter ini tidak didukung untuk kontainer Windows.

Batas tegas memori (dalam MiB) hadir untuk tugas tersebut. Ini dapat diekspresikan menggunakan MiB (misalnya`1024`) atau menggunakan GB (misalnya `1GB` atau`1 GB`) dalam definisi tugas. Ketika ketentuan tugas terdaftar, nilai GB dikonversi ke integer menunjukkan MiB.

Untuk tugas yang di-host di instans Amazon EC2, bidang ini bersifat opsional dan nilai apa pun dapat digunakan. Jika nilai memori tingkat tugas ditentukan, maka nilai memori tingkat kontainer adalah opsional. Jika klaster Anda tidak memiliki instance kontainer terdaftar dengan memori yang diminta tersedia, tugas gagal. Anda dapat memaksimalkan pemanfaatan sumber daya Anda dengan menyediakan tugas Anda sebanyak mungkin memori untuk jenis instans tertentu. Untuk informasi selengkapnya, lihat Manajemen memori instance kontainer.

Untuk tugas yang dihosting di Fargate (baik wadah Linux dan Windows), bidang ini diperlukan dan Anda harus menggunakan salah satu nilai berikut, yang menentukan rentang nilai yang didukung untuk parameter: `cpu`

Nilai memori (dalam MiB, dengan perkiraan nilai setara dalam GB)	Nilai CPU	Sistem operasi yang didukung untuk Fargate
512 (0,5 GB), 1024 (1 GB), 2048 (2 GB)	256 (.25 vCPU)	Linux
1024 (1 GB), ,2048 (2 GB), 3072 (3 GB), 4096 (4GB)	512 (.5 vCPU)	Linux
2048 (2 GB), 3072 (3 GB), 4096 (4GB), 5120 (5 GB), 6144 (6 GB), 7168 (7 GB), 8192 (8 GB)	1024 (1 vCPU)	Linux, Windows
Antara 4096 (4 GB) hingga 16384 (16 GB) sebagai tambahan dari 1024 (1 GB)	2048 (2 vCPU)	Linux, Windows
Antara 8192 (8 GB) hingga 30720 (30 GB) sebagai tambahan dari 1024 (1 GB)	4096 (4 vCPU)	Linux, Windows
Antara 16 GB dan 60 GB dalam peningkatan 4 GB **Note** Opsi ini membutuhkan platform Linux `1.4.0` atau yang lebih baru.	8192 (8 vCPU)	Linux
Antara 32 GB dan 120 GB dalam peningkatan 8 GB	16384 (16vCPU)	Linux

Nilai memori (dalam MiB, dengan perkiraan nilai setara dalam GB)	Nilai CPU	Sistem operasi yang didukung untuk Fargate
Note Opsi ini membutuhkan platform Linux `1.4.0` atau yang lebih baru.		

Definisi kontainer

Saat Anda mendaftarkan definisi tugas, Anda harus menentukan daftar definisi kontainer yang diteruskan ke Docker daemon pada instance container. Parameter berikut diizinkan dalam ketentuan kontainer.

Topik

- Parameter definisi wadah standar
- Parameter definisi kontainer tingkat lanjut
- Parameter definisi kontainer lainnya

Parameter definisi wadah standar

Parameter ketentuan tugas berikut diperlukan atau digunakan di sebagian besar ketentuan kontainer.

Topik

- Nama
- Citra
- Memori
- Pemetaan pelabuhan
- Kredensial Repositori Pribadi

Nama

`name`

Tipe: String

Wajib: Tidak

Nama kontainer. Mengizinkan hingga 255 huruf (huruf besar dan huruf kecil), angka, tanda hubung, dan garis bawah. Jika Anda menautkan beberapa kontainer dalam definisi tugas, `name` satu kontainer dapat dimasukkan ke dalam `links` wadah lain. Ini untuk menghubungkan wadah.

Citra

`image`

Tipe: String

Wajib: Tidak

Citra digunakan untuk memulai kontainer. String ini diteruskan langsung ke Docker daemon. Secara default, gambar dalam Docker Hub registri tersedia. Anda juga dapat menentukan repositori lain dengan salah satu atau*repository-url*/*image*:*tag*. *repository-url*/*image*@*digest* Maksimum 255 huruf (huruf besar dan huruf kecil), angka, tanda hubung, garis bawah, titik dua, titik miring ke depan, dan tanda pagar diperbolehkan. Parameter ini sesuai dengan `Image` di bagian Buat kontainer di Docker Remote API dan `IMAGE` parameter docker run.

- Ketika tugas baru dimulai, agen kontainer Amazon ECS menarik versi terbaru dari citra yang ditentukan dan tanda untuk kontainer yang akan digunakan. Namun, pembaruan berikutnya ke gambar repositori tidak disebarkan ke tugas yang sudah berjalan.
- Citra di registri privat didukung. Untuk informasi selengkapnya, lihat Autentikasi registri privat untuk tugas.
- Gambar di repositori Amazon ECR dapat ditentukan dengan menggunakan konvensi lengkap `registry/repository:tag` atau `registry/repository@digest` penamaan (misalnya, *aws_account_id*`.dkr.ecr.`*region*`.amazonaws.com /`*my-web-app*`:`*latest* atau). *aws_account_id*`.dkr.ecr.`*region*`.amazonaws.com /`*my-web-app*`@`*sha256:94afd1f2e64d908bc90dbca0035a5b567EXAMPLE*
- Gambar di repositori resmi Docker Hub menggunakan satu nama (misalnya, `ubuntu` ataumongo).

- Gambar di repositori lain di Docker Hub memenuhi syarat dengan nama organisasi (misalnya,`amazon/amazon-ecs-agent`).
- Gambar di repositori online lainnya memenuhi syarat lebih lanjut berdasarkan nama domain (misalnya, `quay.io/assemblyline/ubuntu`).

Memori

`memory`

Tipe: Integer

Wajib: Tidak

Jumlah (dalam MiB) memori yang akan ditampilkan ke kontainer. Jika kontainer Anda mencoba untuk melebihi memori yang ditentukan di sini, kontainer akan dimatikan. Jumlah total memori yang disimpan untuk semua kontainer dalam tugas harus lebih rendah dari nilai `memory` tugas, jika ada yang ditentukan. Parameter ini memetakan ke `Memory` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--memory` untuk docker run.

Jika Anda menggunakan tipe peluncuran Fargate, parameter ini opsional.

Jika Anda menggunakan tipe peluncuran EC2, Anda harus menentukan nilai memori tingkat tugas atau nilai memori tingkat kontainer. Jika Anda menentukan tingkat kontainer `memory` dan `memoryReservation` nilai, nilainya harus lebih besar dari `memory` nilainya. `memoryReservation` Jika Anda menentukan`memoryReservation`, maka nilai tersebut dikurangi dari sumber daya memori yang tersedia untuk instance wadah tempat penampung ditempatkan. Jika tidak, nilai `memory` digunakan.

Daemon Docker 20.10.0 atau yang lebih baru menyimpan minimal 6 MiB memori untuk sebuah wadah. Jadi, jangan tentukan kurang dari 6 MiB memori untuk wadah Anda.

Docker 19.03.13-ce atau daemon sebelumnya menyimpan minimal 4 MiB memori untuk sebuah wadah. Jadi, jangan tentukan kurang dari 4 MiB memori untuk wadah Anda.

> **Note**
>
> Jika Anda mencoba memaksimalkan pemanfaatan sumber daya Anda dengan menyediakan memori sebanyak mungkin untuk jenis instans tertentu, lihatManajemen memori instance kontainer.

`memoryReservation`

Tipe: Integer

Wajib: Tidak

Batas lunak (di MiB) memori untuk menyimpan kontainer. Ketika memori sistem sedang diperdebatkan, Docker upaya untuk menjaga memori kontainer ke batas lunak ini. Namun, wadah Anda dapat menggunakan lebih banyak memori saat dibutuhkan. Wadah dapat menggunakan hingga batas keras yang ditentukan dengan `memory` parameter (jika ada) atau semua memori yang tersedia pada instance kontainer, mana yang lebih dulu. Parameter ini memetakan ke `MemoryReservation` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--memory-reservation` untuk docker run.

Jika nilai memori tingkat tugas tidak ditentukan, Anda harus menentukan bilangan bulat bukan nol untuk salah satu atau keduanya `memory` atau `memoryReservation` dalam definisi wadah. Jika Anda menentukan keduanya, `memory` harus lebih besar dari `memoryReservation`. Jika Anda menentukan`memoryReservation`, maka nilai tersebut dikurangi dari sumber daya memori yang tersedia untuk instance wadah tempat penampung ditempatkan. Jika tidak, nilai `memory` digunakan.

Misalnya, kontainer Anda biasanya menggunakan memori 128 MiB, tetapi kadang-kadang meledak hingga 256 MiB memori untuk waktu yang singkat. Anda dapat mengatur 128 MiB, dan batas `memory` keras 300 MiB. `memoryReservation` Konfigurasi ini memungkinkan penampung untuk hanya menyimpan 128 MiB memori dari sumber daya yang tersisa pada instance container. Pada saat yang sama, konfigurasi ini juga memungkinkan wadah untuk menggunakan lebih banyak sumber daya memori bila diperlukan.

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

Daemon Docker 20.10.0 atau yang lebih baru menyimpan minimal 6 MiB memori untuk sebuah wadah. Jadi, jangan tentukan kurang dari 6 MiB memori untuk wadah Anda.

Docker 19.03.13-ce atau daemon sebelumnya menyimpan minimal 4 MiB memori untuk sebuah wadah. Jadi, jangan tentukan kurang dari 4 MiB memori untuk wadah Anda.

> **Note**
>
> Jika Anda mencoba memaksimalkan pemanfaatan sumber daya Anda dengan menyediakan memori sebanyak mungkin untuk jenis instans tertentu, lihatManajemen memori instance kontainer.

Pemetaan pelabuhan

`portMappings`

Tipe: Array objek

Diperlukan: Tidak

Pemetaan port mengizinkan kontainer untuk mengakses port pada instans kontainer host untuk mengirim atau menerima lalu lintas.

Untuk definisi tugas yang menggunakan mode `awsvpc` jaringan, hanya tentukan`containerPort`. `hostPort`Dapat tetap kosong atau nilainya sama dengan`containerPort`.

Pemetaan port pada Windows menggunakan alamat gateway `NetNAT` daripada `localhost`. Tidak ada loopback untuk pemetaan port di Windows, jadi Anda tidak dapat mengakses port yang dipetakan kontainer dari host itu sendiri.

Sebagian besar bidang parameter ini (termasuk`containerPort`,`hostPort`,`protocol`) dipetakan ke `PortBindings` bagian Create a container dari Docker Remote API dan `--publish` opsi untuk docker run. Jika mode jaringan dari definisi tugas diatur ke`host`, port host harus tidak ditentukan atau cocok dengan port kontainer dalam pemetaan port.

> **Note**
>
> Setelah tugas mencapai status `RUNNING` , tugas port kontainer dan host manual dan otomatis akan terlihat di lokasi-lokasi berikut:
>
> - Konsol: bagian Pengikatan jaringan dari deskripsi kontainer untuk tugas yang dipilih.
> - AWS CLI: Bagian `networkBindings` dari output perintah describe-tasks .
> - API: Respons `DescribeTasks` .
> - Metadata: Titik akhir metadata tugas.

`appProtocol`

Tipe: String

Wajib: Tidak

Protokol aplikasi yang digunakan untuk pemetaan port. Parameter ini hanya berlaku untuk Service Connect. Kami menyarankan Anda mengatur parameter ini agar konsisten dengan protokol yang digunakan aplikasi Anda. Jika Anda menyetel parameter ini, Amazon ECS akan menambahkan penanganan koneksi khusus protokol ke proxy service connect. Jika Anda menyetel parameter ini, Amazon ECS menambahkan telemetri khusus protokol di konsol Amazon ECS dan. CloudWatch

Jika Anda tidak menetapkan nilai untuk parameter ini, maka TCP digunakan. Namun, Amazon ECS tidak menambahkan telemetri khusus protokol untuk TCP.

Untuk informasi selengkapnya, lihat the section called “Layanan Connect”.

Nilai protokol yang valid: `"HTTP" | "HTTP2" | "GRPC"`

`containerPort`

Jenis: Integer

Diperlukan: Ya, kapan `portMappings` digunakan

Nomor port pada kontainer yang terikat ke port host yang ditentukan pengguna atau secara otomatis ditetapkan.

Jika menggunakan kontainer dalam tugas dengan tipe peluncuran Fargate, port yang terbuka harus ditentukan menggunakan. `containerPort`

Untuk wadah Windows di Fargate, Anda tidak dapat menggunakan port 3150 untuk file. `containerPort` Ini karena sudah dipesan.

Misalkan Anda menggunakan kontainer dalam tugas dengan tipe peluncuran EC2 dan Anda menentukan port kontainer dan bukan port host. Kemudian, kontainer Anda secara otomatis menerima port host dalam rentang port sementara. Untuk informasi selengkapnya, lihat `hostPort`. Pemetaan port yang secara otomatis ditetapkan dengan cara ini tidak dihitung terhadap kuota 100 port cadangan dari instance kontainer.

`containerPortRange`

Tipe: String

Wajib: Tidak

Rentang nomor port pada kontainer yang terikat pada rentang port host yang dipetakan secara dinamis.

Anda hanya dapat mengatur parameter ini dengan menggunakan `register-task-definition` API. Opsi ini tersedia dalam `portMappings` parameter. Untuk informasi selengkapnya, lihat register-task-definition di dalam Referensi AWS Command Line Interface .

Aturan berikut berlaku saat Anda menentukan`containerPortRange`:

- Anda harus menggunakan mode `bridge` jaringan atau mode `awsvpc` jaringan.
- Parameter ini tersedia untuk tipe peluncuran EC2 dan AWS Fargate.
- Parameter ini tersedia untuk sistem operasi Linux dan Windows.
- Instance container harus memiliki setidaknya versi 1.67.0 dari agen kontainer dan setidaknya versi 1.67.0-1 dari paket. `ecs-init`
- Anda dapat menentukan maksimum 100 rentang port untuk setiap kontainer.
- Anda tidak menentukan`hostPortRange`. Nilai `hostPortRange` ditetapkan sebagai berikut:
  - Untuk kontainer dalam tugas dengan mode `awsvpc` jaringan, `hostPort` diatur ke nilai yang sama dengan`containerPort`. Ini adalah strategi pemetaan statis.
  - Untuk kontainer dalam tugas dengan mode `bridge` jaringan, agen Amazon ECS menemukan port host terbuka dari rentang singkat default dan meneruskannya ke docker untuk mengikatnya ke port kontainer.
- Nilai yang `containerPortRange` valid adalah antara 1 dan 65535.
- Sebuah port hanya dapat dimasukkan dalam satu pemetaan port untuk setiap kontainer.
- Anda tidak dapat menentukan rentang port yang tumpang tindih.
- Port pertama dalam kisaran harus kurang dari port terakhir dalam kisaran.
- Dockermerekomendasikan agar Anda mematikan docker-proxy di file konfigurasi Docker daemon ketika Anda memiliki banyak port.

  Untuk informasi selengkapnya, lihat Masalah #11185 di GitHub.

Untuk informasi tentang cara mematikan docker-proxy di file konfigurasi daemon, lihat Docker daemon Docker di Panduan Pengembang Amazon ECS.

Anda dapat menelepon DescribeTasksuntuk melihathostPortRange, yang merupakan port host yang terikat ke port kontainer.

Rentang port tidak disertakan dalam peristiwa tugas Amazon ECS, yang dikirim ke EventBridge. Untuk informasi selengkapnya, lihat the section called “Otomatiskan tanggapan terhadap kesalahan Amazon ECS menggunakan EventBridge”.

`hostPortRange`

Tipe: String

Wajib: Tidak

Rentang nomor port pada host yang digunakan dengan pengikatan jaringan. Ini ditugaskan oleh Docker dan dikirim oleh agen Amazon ECS.

`hostPort`

Tipe: Integer

Wajib: Tidak

Nomor port pada instans kontainer untuk menyimpan kontainer Anda.

Jika menggunakan kontainer dalam tugas dengan tipe peluncuran Fargate, `hostPort` dapat tetap kosong atau nilainya sama dengan. `containerPort`

Misalkan Anda menggunakan kontainer dalam tugas dengan tipe peluncuran EC2. Anda dapat menentukan port host yang tidak dicadangkan untuk pemetaan port kontainer Anda. Ini disebut sebagai pemetaan port host statis. Atau, Anda dapat menghilangkan `hostPort` (atau mengaturnya ke`0`) sambil menentukan a. `containerPort` Container Anda secara otomatis menerima port dalam rentang port sementara untuk sistem operasi dan Docker versi instans kontainer Anda. Ini disebut sebagai pemetaan port host dinamis.

Rentang port fana default Docker versi 1.6.0 dan yang lebih baru tercantum pada instance di bawah. `/proc/sys/net/ipv4/ip_local_port_range` Jika parameter kernel ini tidak tersedia, port sementara default berkisar sejak `49153-65535` digunakan. Jangan mencoba

menentukan port host dalam rentang port sementara. Ini karena ini dicadangkan untuk penugasan otomatis. Secara umum, port di bawah `32768` berada di luar jangkauan port fana.

Port cadangan default adalah `22` untuk SSH, Docker port `2375` dan`2376`, dan port agen kontainer Amazon ECS. `51678-51680` Port host apa pun yang sebelumnya ditentukan pengguna untuk tugas yang sedang berjalan juga dicadangkan saat tugas sedang berjalan. Setelah tugas berhenti, port host dilepaskan. Port cadangan saat ini ditampilkan di `remainingResources` describe-container-instances output. Instance kontainer mungkin memiliki hingga 100 port cadangan sekaligus, termasuk port cadangan default. Port yang ditetapkan secara otomatis tidak dihitung terhadap kuota 100 port cadangan.

`name`

Jenis: String

Wajib: Tidak, diperlukan agar Service Connect dikonfigurasi dalam layanan

Nama yang digunakan untuk pemetaan port. Parameter ini hanya berlaku untuk Service Connect. Parameter ini adalah nama yang Anda gunakan dalam konfigurasi Service Connect suatu layanan.

Untuk informasi selengkapnya, lihat Layanan Connect.

Dalam contoh berikut, kedua bidang wajib untuk Service Connect digunakan.

```
"portMappings": [
    {
        "name": string,
        "containerPort": integer
    }
]
```

`protocol`

Tipe: String

Wajib: Tidak

Protokol yang digunakan untuk pemetaan port. Nilai yang valid adalah `tcp` dan `udp`. Default adalah `tcp`.

> **Important**
>
> Hanya `tcp` didukung untuk Service Connect. Ingat itu `tcp` tersirat jika bidang ini tidak disetel.

> **Important**
>
> Dukungan UDP hanya tersedia pada instance kontainer yang diluncurkan dengan agen kontainer Amazon ECS versi 1.2.0 (seperti `amzn-ami-2015.03.c-amazon-ecs-optimized` AMI) atau yang lebih baru, atau dengan agen kontainer yang telah diperbarui ke versi 1.3.0 atau yang lebih baru. Untuk memperbarui agen kontainer ke versi terbaru, lihat Memperbarui agen kontainer Amazon ECS.

Jika Anda menentukan port host, gunakan sintaks berikut.

```
"portMappings": [
    {
        "containerPort": integer,
        "hostPort": integer
    }
    ...
]
```

Jika Anda menginginkan port host yang ditetapkan secara otomatis, gunakan sintaks berikut.

```
"portMappings": [
    {
        "containerPort": integer
    }
    ...
]
```

Kredensial Repositori Pribadi

`repositoryCredentials`

Tipe: Objek RepositoryCredentials

Diperlukan: Tidak

Kredensi repositori untuk otentikasi registri pribadi.

Untuk informasi selengkapnya, lihat Autentikasi registri privat untuk tugas.

`credentialsParameter`

Tipe: String

Diperlukan: Ya, kapan `repositoryCredentials` digunakan

Amazon Resource Name (ARN) dari rahasia yang berisi kredensial repositori privat.

Untuk informasi selengkapnya, lihat Autentikasi registri privat untuk tugas.

> **Note**
>
> Saat Anda menggunakan Amazon ECS API, AWS CLI, atau AWS SDK, jika rahasia ada di Wilayah yang sama dengan tugas yang Anda luncurkan, maka Anda dapat menggunakan ARN lengkap atau nama rahasianya. Bila Anda menggunakan AWS Management Console, Anda harus menentukan ARN penuh rahasia.

Berikut ini adalah cuplikan definisi tugas yang menunjukkan parameter yang diperlukan:

```
"containerDefinitions": [
    {
        "image": "private-repo/private-image",
        "repositoryCredentials": {
            "credentialsParameter":
 "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name"
        }
    }
]
```

Parameter definisi kontainer tingkat lanjut

Parameter definisi kontainer lanjutan berikut memberikan kemampuan yang diperluas ke docker runperintah yang digunakan untuk meluncurkan kontainer di instans penampung Amazon ECS Anda.

Topik

Pemeriksaan kondisi

`healthCheck`

Perintah pemeriksaan kesehatan kontainer dan parameter konfigurasi terkait untuk wadah. Untuk informasi selengkapnya, lihat Tentukan kesehatan tugas Amazon ECS menggunakan pemeriksaan kesehatan kontainer.

`command`

Array string yang mewakili perintah yang dijalankan kontainer untuk menentukan apakah itu sehat. Array string dapat dimulai dengan `CMD` untuk menjalankan argumen perintah secara langsung, atau `CMD-SHELL` untuk menjalankan perintah dengan shell default container. Jika tidak ada yang `CMD` ditentukan, digunakan.

Saat mendaftarkan definisi tugas di AWS Management Console, gunakan daftar perintah yang dipisahkan koma. Perintah ini dikonversi ke string setelah definisi tugas dibuat. Contoh masukan untuk pemeriksaan kesehatan adalah sebagai berikut.

```
CMD-SHELL, curl -f http://localhost/ || exit 1
```

Saat mendaftarkan definisi tugas menggunakan panel AWS Management Console JSON,, atau API AWS CLI, lampirkan daftar perintah dalam tanda kurung. Contoh masukan untuk pemeriksaan kesehatan adalah sebagai berikut.

```
[ "CMD-SHELL", "curl -f http://localhost/ || exit 1" ]
```

Kode keluar 0, tanpa `stderr` output, menunjukkan keberhasilan, dan kode keluar bukan nol menunjukkan kegagalan. Untuk informasi lebih lanjut, lihat `HealthCheck` di bagian Membuat kontainer dari API Jarak Jauh Docker.

`interval`

Periode waktu (dalam detik) antara setiap pemeriksaan kesehatan. Anda dapat menentukan antara 5 dan 300 detik. Nilai defaultnya adalah 30 detik.

`timeout`

Periode waktu (dalam hitungan detik) untuk menunggu pemeriksaan kesehatan berhasil sebelum dianggap gagal. Anda dapat menentukan antara 2 dan 60 detik. Nilai defaultnya adalah 5 detik.

`retries`

Jumlah raktu untuk mencoba kembali pemeriksaan kondisi yang gagal sebelum kontainer dianggap tidak sehat. Anda dapat menentukan antara 1 dan 10 percobaan ulang. Nilai default-nya adalah tiga kali percobaan ulang.

`startPeriod`

Masa tenggang opsional untuk menyediakan waktu kontainer untuk bootstrap sebelum pemeriksaan kesehatan yang gagal dihitung terhadap jumlah maksimum percobaan ulang. Anda dapat menentukan antara 0 dan 300 detik. Secara default, `startPeriod` dinonaktifkan.

Environment

`cpu`

Tipe: Integer

Wajib: Tidak

Jumlah `cpu` unit cadangan agen kontainer Amazon ECS untuk kontainer. Di Linux, parameter ini memetakan ke `CpuShares` bagian Create a container dari Docker Remote API dan `--cpu-shares` opsi untuk docker run.

Bidang ini opsional untuk tugas yang menggunakan jenis peluncuran Fargate. Jumlah total CPU yang disediakan untuk semua kontainer yang berada dalam tugas harus lebih rendah dari nilai tingkat tugas`cpu`.

> **Note**
>
> Anda dapat menentukan jumlah unit CPU yang tersedia untuk setiap jenis instans Amazon EC2. Untuk melakukan ini, kalikan jumlah vCPU yang terdaftar untuk jenis instans tersebut di halaman detail Instans Amazon EC2 dengan 1.024.

Linuxkontainer berbagi unit CPU yang tidak terisi dengan kontainer lain pada instance kontainer dengan rasio yang sama dengan jumlah yang dialokasikan. Misalnya, asumsikan bahwa Anda menjalankan tugas kontainer tunggal pada tipe instans inti tunggal dengan 512 unit CPU yang ditentukan untuk kontainer tersebut. Selain itu, tugas itu adalah satu-satunya tugas yang berjalan pada instance container. Dalam contoh ini, wadah dapat menggunakan 1.024 pangsa unit CPU penuh pada waktu tertentu. Namun, asumsikan bahwa Anda meluncurkan salinan lain dari tugas yang sama pada instance kontainer itu. Setiap tugas dijamin minimal 512 unit CPU bila diperlukan. Demikian pula, jika wadah lain tidak menggunakan CPU yang tersisa, setiap kontainer dapat mengambang ke penggunaan CPU yang lebih tinggi. Namun, jika kedua tugas tersebut 100% aktif sepanjang waktu, mereka terbatas pada 512 unit CPU.

Pada instance Linux container, Docker daemon pada instance container menggunakan nilai CPU untuk menghitung rasio pangsa CPU relatif untuk menjalankan container. Untuk informasi selengkapnya, lihat batasan pembagian CPU dalam dokumentasi Docker. Nilai pembagian CPU minimum yang valid yang diizinkan kernel Linux adalah 2. Namun, parameter CPU tidak diperlukan, dan Anda dapat menggunakan nilai CPU di bawah dua dalam definisi container Anda. Untuk nilai CPU di bawah dua (termasuk null), perilaku bervariasi berdasarkan versi agen penampung Amazon ECS Anda:

- Versi agen <= 1.1.0: Nilai CPU nol dan nol diteruskan ke Docker 0. Dockerkemudian mengubah nilai ini menjadi 1.024 saham CPU. Nilai CPU dari satu diteruskan ke Docker sebagai satu, yang Linux kernel mengkonversi menjadi dua saham CPU.
- Versi agen >= 1.2.0: Nilai nol, nol, dan CPU dari satu diteruskan Docker sebagai dua pembagian CPU.

Pada instance Windows kontainer, kuota CPU diberlakukan sebagai kuota absolut. Windowscontainer hanya memiliki akses ke jumlah CPU tertentu yang ditentukan dalam definisi tugas. Nilai CPU nol atau nol diteruskan ke Docker as0. Windowskemudian menafsirkan nilai ini sebagai 1% dari satu CPU.

Untuk contoh selengkapnya, lihat Cara Amazon ECS mengelola sumber daya CPU dan memori.

`gpu`

Tipe: Objek ResourceRequirement

Diperlukan: Tidak

Jumlah fisik `GPUs` yang dicadangkan agen kontainer Amazon ECS untuk kontainer. Jumlah GPU yang dicadangkan untuk semua kontainer dalam tugas tidak boleh melebihi jumlah GPU yang tersedia pada instance penampung tempat tugas diluncurkan. Untuk informasi selengkapnya, lihat Bekerja dengan GPU di Amazon ECS.

> **Note**
>
> Parameter ini tidak didukung untuk Windows kontainer atau kontainer yang di-host di Fargate.

`Elastic Inference accelerator`

Tipe: Objek ResourceRequirement

Diperlukan: Tidak

Untuk `InferenceAccelerator` jenisnya, `value` cocok dengan yang `deviceName` `InferenceAccelerator` ditentukan dalam definisi tugas. Untuk informasi selengkapnya, lihat the section called “Nama akselerator Elastic Inference”.

> **Note**
>
> Mulai 15 April 2023, tidak AWS akan memasukkan pelanggan baru ke Amazon Elastic Inference (EI), dan akan membantu pelanggan saat ini memigrasikan beban kerja mereka ke opsi yang menawarkan harga dan kinerja yang lebih baik. Setelah 15 April 2023, pelanggan baru tidak akan dapat meluncurkan instans dengan akselerator Amazon EI di Amazon, Amazon ECS, atau SageMaker Amazon EC2. Namun, pelanggan yang telah menggunakan Amazon EI setidaknya sekali selama periode 30 hari terakhir dianggap sebagai pelanggan saat ini dan akan dapat terus menggunakan layanan ini.

> **Note**
>
> Parameter ini tidak didukung untuk Windows kontainer atau kontainer yang di-host di Fargate.

`essential`

Tipe: Boolean

Wajib: Tidak

Misalkan `essential` parameter wadah ditandai sebagai`true`, dan wadah itu gagal atau berhenti karena alasan apa pun. Kemudian, semua wadah lain yang merupakan bagian dari tugas dihentikan. Jika `essential` parameter wadah ditandai sebagai`false`, maka kegagalannya tidak mempengaruhi sisa kontainer dalam suatu tugas. Jika parameter ini dihilangkan, kontainer diasumsikan menjadi penting.

Semua tugas harus memiliki setidaknya satu kontainer penting. Misalkan Anda memiliki aplikasi yang terdiri dari beberapa kontainer. Kemudian, kelompokkan wadah yang digunakan untuk tujuan bersama menjadi komponen, dan pisahkan komponen yang berbeda menjadi beberapa definisi tugas. Untuk informasi selengkapnya, lihat Merancang aplikasi Anda.

```
"essential": true|false
```

`entryPoint`

> **Important**
>
> Versi awal agen kontainer Amazon ECS tidak menangani `entryPoint` parameter dengan benar. Jika Anda memiliki masalah menggunakan `entryPoint`, perbarui agen kontainer Anda atau masukkan perintah dan argumen Anda sebagai item array `command` sebagai gantinya.

Tipe: Array string

Diperlukan: Tidak

Titik masuk yang diteruskan ke wadah. Parameter ini memetakan ke `Entrypoint` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--entrypoint` untuk docker run. Untuk informasi lebih lanjut tentang Docker `ENTRYPOINT` parameter, lihat https://docs.docker.com/engine/reference/builder/#entrypoint.

```
"entryPoint": ["string", ...]
```

`command`

Tipe: Array string

Diperlukan: Tidak

Perintah yang disampaikan ke kontainer. Parameter ini sesuai dengan `Cmd` di bagian Buat kontainer di Docker Remote API dan parameter `COMMAND` untuk docker run. Untuk informasi lebih lanjut tentang Docker `CMD` parameter, lihat https://docs.docker.com/engine/reference/builder/#cmd. Jika ada beberapa argumen, pastikan bahwa setiap argumen adalah string terpisah dalam array.

```
"command": ["string", ...]
```

`workingDirectory`

Tipe: String

Wajib: Tidak

Direktori kerja untuk menjalankan perintah di dalam wadah di. Parameter ini sesuai dengan `WorkingDir` di bagian Buat kontainer di Docker Remote API dan opsi `--workdir` untuk docker run.

```
"workingDirectory": "string"
```

`environmentFiles`

Tipe: Array objek

Diperlukan: Tidak

Daftar file yang berisi variabel lingkungan untuk diteruskan ke kontainer. Parameter ini memetakan ke opsi `--env-file` ke docker run.

Ini tidak tersedia untuk Windows kontainer.

Anda dapat menentukan hingga 10 file lingkungan. File harus memiliki ekstensi file `.env`. Setiap baris dalam file lingkungan berisi variabel lingkungan dalam `VARIABLE=VALUE` format. Baris yang dimulai dengan # diperlakukan sebagai komentar dan diabaikan. Untuk informasi selengkapnya tentang sintaks file variabel lingkungan yang sesuai, lihat Mendeklarasikan variabel lingkungan default dalam file.

Jika ada variabel lingkungan individu yang ditentukan dalam ketentuan kontainer, maka akan lebih diutamakan daripada variabel yang ada di dalam file lingkungan. Jika beberapa file lingkungan ditentukan yang berisi variabel yang sama, mereka diproses dari atas ke bawah. Kami merekomendasikan agar Anda menggunakan nama variabel yang unik. Untuk informasi selengkapnya, lihat Gunakan parameter definisi tugas untuk meneruskan variabel lingkungan ke wadah.

`value`

Tipe: String

Wajib: Ya

Amazon Resource Name (ARN) dari objek Amazon S3 yang berisi variabel lingkungan file.

`type`

Tipe: String

Wajib: Ya

Tipe file yang akan digunakan. Satu-satunya nilai yang didukung adalah `s3`.

`environment`

Tipe: Array objek

Diperlukan: Tidak

Variabel lingkungan untuk disampaikan ke kontainer. Parameter ini sesuai dengan `Env` di bagian Buat kontainer di Docker Remote API dan opsi `--env` untuk docker run.

> **Important**
>
> Kami tidak merekomendasikan penggunaan variabel lingkungan plaintext untuk informasi sensitif, seperti data kredensial.

`name`

Tipe: String

Diperlukan: Ya, saat `environment` digunakan

Nama variabel lingkungan.

`value`

Tipe: String

Diperlukan: Ya, saat `environment` digunakan

Nilai dari variabel lingkungan.

```
"environment" : [
    { "name" : "string", "value" : "string" },
    { "name" : "string", "value" : "string" }
]
```

`secrets`

Tipe: Object array

Diperlukan: Tidak

Objek yang mewakili rahasia untuk mengekspos ke wadah Anda. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah.

`name`

Tipe: String

Wajib: Ya

Nilai untuk ditetapkan sebagai variabel lingkungan pada kontainer.

`valueFrom`

Tipe: String

Wajib: Ya

Secret untuk mengekspos ke kontainer. Nilai yang didukung adalah Nama Sumber Daya Amazon (ARN) lengkap dari AWS Secrets Manager rahasia atau ARN lengkap parameter di Parameter Store. AWS Systems Manager

> **Note**
>
> Jika parameter Systems Manager Parameter Store ada Wilayah AWS sama dengan tugas yang Anda luncurkan, Anda dapat menggunakan ARN lengkap atau nama rahasia. Jika parameter ada di Wilayah yang berbeda, ARN lengkap harus disebutkan.

```
"secrets": [
    {
        "name": "environment_variable_name",
        "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter/parameter_name"
    }
]
```

Pengaturan jaringan

`disableNetworking`

Tipe: Boolean

Wajib: Tidak

Ketika parameter ini benar, jaringan mati di dalam wadah. Parameter ini memetakan ke `NetworkDisabled` di bagian Buat kontainer API Jarak Jauh Docker.

> **Note**
>
> Parameter ini tidak didukung untuk Windows kontainer atau tugas yang menggunakan mode `awsvpc` jaringan.

Nilai default-nya `false`.

```
"disableNetworking": true|false
```

`links`

Tipe: Array string

Diperlukan: Tidak

Parameter `link` mengizinkan kontainer untuk berkomunikasi satu sama lain tanpa perlu port pemetaan. Parameter ini hanya didukung jika mode jaringan dari definisi tugas diatur ke`bridge`. `name:internalName`Konstruk ini analog dengan dalam tautan. `name:alias` Docker Mengizinkan hingga 255 huruf (huruf besar dan huruf kecil), angka, tanda hubung, dan garis bawah. Untuk informasi selengkapnya tentang menautkan Docker kontainer, lihat https://docs.docker.com/engine/userguide/networking/default_network/dockerlinks/. Parameter ini memetakan ke `Links` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--link` untuk docker run.

> **Note**
>
> Parameter ini tidak didukung untuk Windows kontainer atau tugas yang menggunakan mode `awsvpc` jaringan.

> **Important**
>
> Kontainer yang ditempatkan pada instance kontainer yang sama dapat berkomunikasi satu sama lain tanpa memerlukan tautan atau pemetaan port host. Isolasi jaringan pada instans kontainer dikendalikan oleh grup keamanan dan pengaturan VPC.

```
"links": ["name:internalName", ...]
```

`hostname`

Tipe: String

Wajib: Tidak

Nama host yang digunakan untuk kontainer Anda. Parameter ini memetakan ke `Hostname` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--hostname` untuk docker run.

> **Note**
>
> Jika Anda menggunakan mode `awsvpc` jaringan, `hostname` parameter tidak didukung.

```
"hostname": "string"
```

`dnsServers`

Tipe: Array string

Diperlukan: Tidak

Daftar server DNS yang disajikan untuk kontainer. Parameter ini memetakan ke `Dns` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--dns` untuk docker run.

> **Note**
>
> Parameter ini tidak didukung untuk Windows kontainer atau tugas yang menggunakan mode `awsvpc` jaringan.

```
"dnsServers": ["string", ...]
```

`dnsSearchDomains`

Tipe: Array string

Diperlukan: Tidak

Pola: ^[a-zA-Z0-9-.]{0,253}[a-zA-Z0-9]$

Daftar domain pencarian DNS yang disajikan untuk kontainer. Parameter ini memetakan ke `DnsSearch` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--dns-search` untuk docker run.

> **Note**
>
> Parameter ini tidak didukung untuk wadah Windows atau tugas yang menggunakan mode `awsvpc` jaringan.

```
"dnsSearchDomains": ["string", ...]
```

`extraHosts`

Tipe: Array objek

Diperlukan: Tidak

Daftar nama host dan pemetaan alamat IP untuk ditambahkan ke file `/etc/hosts` di kontainer.

Parameter ini memetakan ke `ExtraHosts` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--add-host` untuk docker run.

> **Note**
>
> Parameter ini tidak didukung untuk wadah Windows atau tugas yang menggunakan mode `awsvpc` jaringan.

```
"extraHosts": [
      {
        "hostname": "string",
        "ipAddress": "string"
      }
      ...
    ]
```

`hostname`

Jenis: String

Diperlukan: Ya, kapan `extraHosts` digunakan

Nama host yang akan digunakan dalam entri `/etc/hosts`.

`ipAddress`

Jenis: String

Diperlukan: Ya, kapan `extraHosts` digunakan

Alamat IP yang akan digunakan dalam `/etc/hosts` entri.

Penyimpanan dan pencatatan

`readonlyRootFilesystem`

Tipe: Boolean

Wajib: Tidak

Jika parameter ini betul, kontainer diberikan akses hanya-baca ke sistem file asalnya. Parameter ini memetakan ke `ReadonlyRootfs` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--read-only` untuk docker run.

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

Nilai default-nya `false`.

```
"readonlyRootFilesystem": true|false
```

`mountPoints`

Tipe: Array objek

Diperlukan: Tidak

Titik pemasangan untuk volume data dalam penampung Anda. Parameter ini memetakan ke `Volumes` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--volume` untuk docker run.

Kontainer Windows dapat memasang seluruh direktori pada drive yang sama dengan `$env:ProgramData`. Kontainer Windows tidak dapat memasang direktori pada drive yang berbeda, dan titik pemasangan tidak dapat digunakan di seluruh drive.

`sourceVolume`

Jenis: String

Diperlukan: Ya, kapan `mountPoints` digunakan

Nama volume yang akan dipasang.

`containerPath`

Jenis: String

Diperlukan: Ya, kapan `mountPoints` digunakan

Jalur dalam wadah tempat volume akan dipasang.

`readOnly`

Tipe: Boolean

Wajib: Tidak

Jika nilai ini adalah `true`, kontainer memiliki akses hanya-baca ke volume. Jika nilai ini adalah `false`, maka kontainer dapat menulis ke volume. Nilai default-nya adalah `false`.

`volumesFrom`

Tipe: Array objek

Wajib: Tidak

Volume data untuk memasang dari kontainer lain. Parameter ini sesuai dengan `VolumesFrom` di bagian Buat kontainer di Docker Remote API dan opsi `--volumes-from` untuk docker run.

`sourceContainer`

Jenis: String

Diperlukan: Ya, saat `volumesFrom` digunakan

Nama kontainer untuk memasang volume.

`readOnly`

Tipe: Boolean

Wajib: Tidak

Jika nilai ini adalah `true`, kontainer memiliki akses hanya-baca ke volume. Jika nilai ini adalah `false`, maka kontainer dapat menulis ke volume. Nilai default-nya adalah `false`.

```
"volumesFrom": [
```

```
        {
          "sourceContainer": "string",
          "readOnly": true|false
        }
      ]
```

`logConfiguration`

Jenis: LogConfigurationObjek

Diperlukan: Tidak

Spesifikasi konfigurasi log untuk kontainer.

Misalnya definisi tugas yang menggunakan konfigurasi log, lihatContoh ketentuan tugas.

Parameter ini memetakan ke `LogConfig` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--log-driver` untuk `docker run`. Secara default, kontainer menggunakan driver logging yang sama dengan yang digunakan Docker daemon. Namun, penampung mungkin menggunakan driver logging yang berbeda dari Docker daemon dengan menentukan driver log dengan parameter ini dalam definisi container. Untuk menggunakan driver penlog yang berbeda untuk kontainer, sistem log harus dikonfigurasi dengan benar pada instans kontainer (atau pada server log yang berbeda untuk opsi penlog jarak jauh). Untuk informasi selengkapnya tentang opsi untuk driver log yang didukung berbeda, lihat Mengkonfigurasi driver logging dalam Docker dokumentasi.

Pertimbangkan hal berikut saat menentukan konfigurasi log untuk kontainer Anda:

- Amazon ECS mendukung subset driver logging yang tersedia untuk daemon. Docker Driver log tambahan mungkin tersedia di rilis mendatang agen kontainer Amazon ECS.
- Parameter ini memerlukan Docker Remote API versi 1.18 atau yang lebih baru pada instance container Anda.
- Untuk tugas yang menggunakan tipe peluncuran EC2, agen penampung Amazon ECS yang berjalan pada instance container harus mendaftarkan driver logging yang tersedia pada instance tersebut dengan variabel `ECS_AVAILABLE_LOGGING_DRIVERS` lingkungan sebelum container yang ditempatkan pada instance tersebut dapat menggunakan opsi konfigurasi log ini. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.
- Untuk tugas yang menggunakan tipe peluncuran Fargate, karena Anda tidak memiliki akses ke infrastruktur dasar tugas Anda di-host, perangkat lunak tambahan apa pun yang diperlukan

harus diinstal di luar tugas. Misalnya, agregator Fluentd keluaran atau host jarak jauh yang menjalankan Logstash untuk mengirim log ke. Gelf

```
"logConfiguration": {
      "logDriver": "awslogs","fluentd","gelf","json-
file","journald","logentries","splunk","syslog","awsfirelens",
      "options": {"string": "string"
        ...},
 "secretOptions": [{
  "name": "string",
  "valueFrom": "string"
 }]
}
```

`logDriver`

Jenis: String

Nilai yang valid: `"awslogs","fluentd","gelf","json-file","journald","logentries","splunk","syslog","awsfirelens"`

Diperlukan: Ya, saat `logConfiguration` digunakan

Driver log yang digunakan untuk kontainer. Secara default, nilai valid yang tercantum sebelumnya adalah driver log yang dapat berkomunikasi dengan agen penampung Amazon ECS.

Untuk tugas yang menggunakan tipe peluncuran Fargate, driver log yang didukung adalah`awslogs`,`splunk`, dan. `awsfirelens`

Untuk tugas yang menggunakan tipe peluncuran EC2, driver log yang didukung adalah`awslogs`,`fluentd`,`gelf`,`json-file`,`journald`,`logentries`, `syslogsplunk`, dan`awsfirelens`.

Untuk informasi selengkapnya tentang cara menggunakan driver `awslogs` log dalam definisi tugas untuk mengirim log kontainer Anda ke CloudWatch Log, lihatMenggunakan driver log awslogs.

Untuk informasi selengkapnya tentang penggunaan driver log `awsfirelens`, lihat Perutean Log Kustom.

> **Note**
>
> Jika Anda memiliki driver khusus yang tidak terdaftar, Anda dapat melakukan fork proyek agen penampung Amazon ECS yang tersedia GitHub dan menyesuaikannya agar berfungsi dengan driver itu. Kami mendorong Anda untuk mengirim permintaan tarik untuk perubahan yang ingin Anda sertakan. Namun, saat ini kami tidak mendukung menjalankan salinan modifikasi dari perangkat lunak ini.

Parameter ini memerlukan Docker Remote API versi 1.18 atau lebih besar pada instans kontainer Anda.

`options`

Tipe: Peta antar string

Diperlukan: Tidak

Peta kunci/nilai opsi konfigurasi untuk dikirim ke driver log.

Saat Anda menggunakan FireLens untuk merutekan log ke AWS Partner Network tujuan Layanan AWS atau penyimpanan log dan analitik, Anda dapat mengatur `log-driver-buffer-limit` opsi untuk membatasi jumlah peristiwa yang di-buffer dalam memori, sebelum dikirim ke wadah router log. Ini dapat membantu menyelesaikan potensi masalah kehilangan log karena throughput yang tinggi dapat mengakibatkan memori habis untuk buffer di dalamnya. Docker Untuk informasi selengkapnya, lihat the section called “Batas buffer fluentd”.

Parameter ini memerlukan API Docker Remote versi 1.19 atau lebih besar pada instans kontainer Anda.

`secretOptions`

Tipe: Array objek

Diperlukan: Tidak

Objek yang mewakili rahasia untuk diteruskan ke konfigurasi log. Rahasia yang digunakan dalam konfigurasi log dapat mencakup token otentikasi, sertifikat, atau kunci enkripsi. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah.

`name`

Tipe: String

Wajib: Ya

Nilai untuk ditetapkan sebagai variabel lingkungan pada kontainer.

`valueFrom`

Tipe: String

Wajib: Ya

Secret untuk diekspos ke konfigurasi log kontainer.

```
"logConfiguration": {
 "logDriver": "splunk",
 "options": {
  "splunk-url": "https://cloud.splunk.com:8080",
  "splunk-token": "...",
  "tag": "...",
  ...
 },
 "secretOptions": [{
  "name": "splunk-token",
  "valueFrom": "/ecs/logconfig/splunkcred"
 }]
}
```

`firelensConfiguration`

Jenis: FirelensConfigurationObjek

Diperlukan: Tidak

FireLens Konfigurasi untuk wadah. Digunakan untuk menentukan dan mengonfigurasi router log untuk log kontainer. Untuk informasi selengkapnya, lihat Menggunakan perutean log khusus.

```
{
    "firelensConfiguration": {
        "type": "fluentd",
        "options": {
```

```
            "KeyName": ""
        }
    }
}
```

`options`

Tipe: Peta antar string

Diperlukan: Tidak

Peta kunci/nilai opsi untuk digunakan saat mengkonfigurasi router log. Bidang ini opsional dan dapat digunakan untuk menentukan file konfigurasi kustom atau untuk menambahkan metadata tambahan, seperti tugas, ketentuan tugas, klaster, dan detail instans kontainer ke log acara. Jika ditentukan, sintaksis yang digunakan adalah `"options":{"enable-ecs-log-metadata":"true|false","config-file-type:"s3|file","config-file-value":"arn:aws:s3:::mybucket/fluent.conf|filepath"}`. Untuk informasi selengkapnya, lihat Menentukan FireLens konfigurasi dalam definisi tugas.

`type`

Tipe: String

Wajib: Ya

Router log yang akan digunakan. Nilai yang valid adalah `fluentd` atau `fluentbit`.

Keamanan

Untuk informasi selengkapnya tentang keamanan kontainer, lihat Keamanan tugas dan kontainer di Panduan Praktik Terbaik Amazon ECS.

`credentialSpecs`

Tipe: Array string

Diperlukan: Tidak

Daftar ARN di SSM atau Amazon S3 ke file credential spec `CredSpec` () yang mengonfigurasi wadah untuk otentikasi Active Directory. Kami menyarankan Anda menggunakan parameter ini alih-alih`dockerSecurityOptions`. Jumlah maksimum ARN adalah 1.

Ada dua format untuk setiap ARN.

CredentialSpecDomainless:mYarn

Anda gunakan `credentialspecdomainless:MyARN` untuk `CredSpec` menyediakan bagian tambahan untuk rahasia di Secrets Manager. Anda memberikan kredensi login ke domain secara rahasia.

Setiap tugas yang berjalan pada instance kontainer apa pun dapat bergabung dengan domain yang berbeda.

Anda dapat menggunakan format ini tanpa menggabungkan instance container ke domain.

Spesifikasi kredensial:mYarn

Anda gunakan `credentialspec:MyARN` untuk menyediakan `CredSpec` untuk satu domain.

Anda harus menggabungkan instance penampung ke domain sebelum memulai tugas apa pun yang menggunakan definisi tugas ini.

Dalam kedua format, ganti `MyARN` dengan ARN di SSM atau Amazon S3.

`credspec`Harus menyediakan ARN di Secrets Manager untuk rahasia yang berisi nama pengguna, kata sandi, dan domain untuk terhubung. Untuk keamanan yang lebih baik, instance tidak digabungkan ke domain untuk otentikasi tanpa domain. Aplikasi lain pada instance tidak dapat menggunakan kredensyal tanpa domain. Anda dapat menggunakan parameter ini untuk menjalankan tugas pada instance yang sama, bahkan tugas harus bergabung dengan domain yang berbeda. Untuk informasi selengkapnya, lihat Menggunakan GMSAs untuk Kontainer Windows dan Menggunakan GMSAs untuk Kontainer Linux.

`privileged`

Tipe: Boolean

Wajib: Tidak

Jika parameter ini betul, kontainer diberikan hak istimewa yang lebih tinggi pada instans kontainer host misalnya (mirip dengan pengguna `root`). Kami merekomendasikan agar tidak menjalankan kontainer dengan`privileged`. Dalam kebanyakan kasus, Anda dapat menentukan hak istimewa yang tepat yang Anda butuhkan dengan menggunakan parameter tertentu alih-alih menggunakan`privileged`.

Parameter ini memetakan ke `Privileged` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--privileged` untuk docker run.

> **Note**
>
> Parameter ini tidak didukung untuk wadah atau tugas Windows menggunakan tipe peluncuran Fargate.

Default-nya adalah `false`.

```
"privileged": true|false
```

`user`

Tipe: String

Wajib: Tidak

Pengguna yang akan digunakan di dalam kontainer. Parameter ini memetakan ke `User` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--user` untuk docker run.

> **Important**
>
> Saat menjalankan tugas yang menggunakan mode `host` jaringan, jangan jalankan kontainer menggunakan pengguna root (UID 0). Sebagai praktik terbaik keamanan, selalu gunakan pengguna non-root.

Anda dapat menentukan `user` dengan menggunakan format berikut. Jika menentukan UID atau GID, Anda harus menentukannya sebagai bilangan bulat positif.

- `user`
- `user:group`
- `uid`
- `uid:gid`
- `user:gid`
- `uid:group`

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

```
"user": "string"
```

`dockerSecurityOptions`

Tipe: Array string

Nilai yang valid: "no-new-privileges" | “apparmor:profile” | “label: *value*" | “credentialspec:” *CredentialSpecFilePath*

Diperlukan: Tidak

Daftar string untuk menyediakan konfigurasi khusus untuk beberapa sistem keamanan. Untuk informasi selengkapnya tentang nilai yang valid, lihat Konfigurasi Keamanan Dijalankan Docker. Bidang ini tidak valid untuk kontainer dalam tugas menggunakan tipe peluncuran Fargate.

Untuk Linux tugas di EC2, parameter ini dapat digunakan untuk mereferensikan label khusus untuk SELinux dan sistem keamanan AppArmor multi-level.

Untuk tugas apa pun di EC2, parameter ini dapat digunakan untuk mereferensikan file spesifikasi kredensyal yang mengonfigurasi wadah untuk otentikasi Active Directory. Lihat informasi yang lebih lengkap di Menggunakan GMSAs untuk Windows Container di Amazon EC2 dan Menggunakan gMSA untuk Linux Wadah di Amazon EC2.

Parameter ini memetakan ke `SecurityOpt` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--security-opt` untuk docker.

```
"dockerSecurityOptions": ["string", ...]
```

> **Note**
>
> Agen penampung Amazon ECS yang berjalan pada instance container harus mendaftar dengan variabel `ECS_APPARMOR_CAPABLE=true` lingkungan `ECS_SELINUX_CAPABLE=true` atau sebelum container yang ditempatkan pada instance

tersebut dapat menggunakan opsi keamanan ini. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.

Batas sumber daya

`ulimits`

Tipe: Array objek

Diperlukan: Tidak

Daftar nilai `ulimit` yang harus ditentukan untuk kontainer. Nilai ini menimpa pengaturan kuota sumber daya default untuk sistem operasi. Parameter ini memetakan ke `Ulimits` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--ulimit` untuk docker run.

Tugas Amazon ECS yang dihosting di Fargate menggunakan nilai batas sumber daya default yang ditetapkan oleh sistem operasi dengan pengecualian parameter batas sumber daya. `nofile` Batas `nofile` sumber daya menetapkan batasan pada jumlah file terbuka yang dapat digunakan wadah. Di Fargate, batas `nofile` lunak default adalah `1024` dan batas keras adalah. `65535` Anda dapat mengatur nilai dari kedua batas hingga`1048576`. Untuk informasi selengkapnya, lihat Batas sumber daya tugas.

Parameter ini memerlukan Docker Remote API versi 1.18 atau lebih besar pada instans kontainer Anda.

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

```
"ulimits": [
      {
        "name":
 "core"|"cpu"|"data"|"fsize"|"locks"|"memlock"|"msgqueue"|"nice"|"nofile"|"nproc"|"rss"|"rtp
        "softLimit": integer,
        "hardLimit": integer
      }
      ...
    ]
```

`name`

Jenis: String

Nilai yang valid: `"core" | "cpu" | "data" | "fsize" | "locks" | "memlock" | "msgqueue" | "nice" | "nofile" | "nproc" | "rss" | "rtprio" | "rttime" | "sigpending" | "stack"`

Diperlukan: Ya, kapan `ulimits` digunakan

`type` dari `ulimit`.

`hardLimit`

Jenis: Integer

Diperlukan: Ya, kapan `ulimits` digunakan

Batas keras untuk tipe `ulimit`.

`softLimit`

Jenis: Integer

Diperlukan: Ya, kapan `ulimits` digunakan

Batas lunak untuk tipe `ulimit`.

Label docker

`dockerLabels`

Tipe: Peta antar string

Diperlukan: Tidak

Sebuah peta kunci/nilai label untuk ditambahkan ke kontainer. Parameter ini memetakan ke `Labels` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--label` untuk docker run.

Parameter ini memerlukan Docker Remote API versi 1.18 atau lebih besar pada instans kontainer Anda.

```
"dockerLabels": {"string": "string"
```

```
      ...}
```

Parameter definisi kontainer lainnya

Parameter definisi kontainer berikut dapat digunakan saat mendaftarkan definisi tugas di konsol Amazon ECS dengan menggunakan opsi Configure via JSON. Untuk informasi selengkapnya, lihat Membuat definisi tugas menggunakan konsol.

Topik

- Parameter Linux
- Dependensi kontainer
- Waktu habis kontainer
- Kontrol sistem
- Interaktif
- Terminal semu

Parameter Linux

`linuxParameters`

Tipe: Objek LinuxParameters

Diperlukan: Tidak

Linux-opsi spesifik yang diterapkan ke wadah, seperti KernelCapabilities.

> **Note**
>
> Parameter ini tidak didukung untuk Windows kontainer.

```
"linuxParameters": {
      "capabilities": {
        "add": ["string", ...],
        "drop": ["string", ...]
        }
      }
```

`capabilities`

Tipe: Objek KernelCapabilities

Diperlukan: Tidak

LinuxKemampuan untuk wadah yang ditambahkan ke atau dijatuhkan dari konfigurasi default yang disediakan olehDocker. Untuk informasi selengkapnya tentang kemampuan default dan kemampuan lain yang tersedia, lihat hak istimewa Runtime dan kemampuan Linux dalam referensi Dockerrun. Untuk informasi lebih lanjut tentang Linux kemampuan ini, lihat halaman manual kemampuan (7) Linux.

`add`

Tipe: Array string

Nilai yang valid: `"ALL" | "AUDIT_CONTROL" | "AUDIT_READ" | "AUDIT_WRITE" | "BLOCK_SUSPEND" | "CHOWN" | "DAC_OVERRIDE" | "DAC_READ_SEARCH" | "FOWNER" | "FSETID" | "IPC_LOCK" | "IPC_OWNER" | "KILL" | "LEASE" | "LINUX_IMMUTABLE" | "MAC_ADMIN" | "MAC_OVERRIDE" | "MKNOD" | "NET_ADMIN" | "NET_BIND_SERVICE" | "NET_BROADCAST" | "NET_RAW" | "SETFCAP" | "SETGID" | "SETPCAP" | "SETUID" | "SYS_ADMIN" | "SYS_BOOT" | "SYS_CHROOT" | "SYS_MODULE" | "SYS_NICE" | "SYS_PACCT" | "SYS_PTRACE" | "SYS_RAWIO" | "SYS_RESOURCE" | "SYS_TIME" | "SYS_TTY_CONFIG" | "SYSLOG" | "WAKE_ALARM"`

Diperlukan: Tidak

LinuxKemampuan wadah untuk ditambahkan ke konfigurasi default yang disediakan olehDocker. Parameter ini memetakan ke `CapAdd` di bagian Buat kontainer pada API Docker Remote dan opsi `--cap-add` menjadi docker run.

> **Note**
>
> Tugas yang diluncurkan di Fargate hanya mendukung penambahan kemampuan `SYS_PTRACE` kernel.

`add`

Tipe: Array string

Nilai yang valid: "SYS_PTRACE"

Diperlukan: Tidak

LinuxKemampuan kontainer untuk ditambahkan ke konfigurasi default yang disediakan olehDocker. Parameter ini memetakan ke CapAdd di bagian Buat kontainer pada API Docker Remote dan opsi --cap-add menjadi docker run.

drop

Tipe: Array string

Nilai yang valid: "ALL" | "AUDIT_CONTROL" | "AUDIT_WRITE" | "BLOCK_SUSPEND" | "CHOWN" | "DAC_OVERRIDE" | "DAC_READ_SEARCH" | "FOWNER" | "FSETID" | "IPC_LOCK" | "IPC_OWNER" | "KILL" | "LEASE" | "LINUX_IMMUTABLE" | "MAC_ADMIN" | "MAC_OVERRIDE" | "MKNOD" | "NET_ADMIN" | "NET_BIND_SERVICE" | "NET_BROADCAST" | "NET_RAW" | "SETFCAP" | "SETGID" | "SETPCAP" | "SETUID" | "SYS_ADMIN" | "SYS_BOOT" | "SYS_CHROOT" | "SYS_MODULE" | "SYS_NICE" | "SYS_PACCT" | "SYS_PTRACE" | "SYS_RAWIO" | "SYS_RESOURCE" | "SYS_TIME" | "SYS_TTY_CONFIG" | "SYSLOG" | "WAKE_ALARM"

Diperlukan: Tidak

LinuxKemampuan untuk wadah untuk menghapus dari konfigurasi default yang disediakan olehDocker. Parameter ini memetakan ke CapDrop di bagian Buat kontainer pada API Docker Remote dan opsi --cap-drop menjadi docker run.

devices

Perangkat host apa pun yang akan diekspos ke kontainer. Parameter ini memetakan ke Devices di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan --device untuk docker run.

> **Note**
>
> devicesParameter tidak didukung saat Anda menggunakan tipe peluncuran Fargate, atau wadah Windows.

Tipe: Array objek Perangkat

Wajib: Tidak

`hostPath`

Jalur untuk perangkat pada instans kontainer host.

Tipe: String

Diperlukan: Ya

`containerPath`

Jalur di dalam wadah untuk mengekspos perangkat host di.

Tipe: String

Wajib: Tidak

`permissions`

Izin eksplisit untuk diberikan ke kontainer untuk perangkat. Secara default, kontainer memiliki izin untuk`read`, `write`, dan `mknod` pada perangkat.

Tipe: Array string

Nilai yang Valid: `read` | `write` | `mknod`

`initProcessEnabled`

Jalankan proses `init` di dalam kontainer yang meneruskan sinyal dan melakukan proses kembali. Parameter ini dipetakan ke opsi `--init` menuju docker run.

Parameter ini memerlukan API Docker Remote versi 1.25 atau lebih besar pada instans kontainer Anda.

`maxSwap`

Jumlah total memori tukar (dalam MiB) yang dapat digunakan oleh kontainer. Parameter ini diterjemahkan ke opsi `--memory-swap` untuk docker run di mana nilai adalah jumlah memori kontainer ditambah nilai `maxSwap`.

Jika nilai `maxSwap` sebesar `0` ditentukan, kontainer tidak menggunakan swap. Nilai yang diterima adalah `0` atau bilangan bulat positif. Jika `maxSwap` parameter diabaikan, kontainer menggunakan konfigurasi swap untuk instans kontainer tempatnya berjalan. Nilai `maxSwap` harus ditetapkan untuk parameter `swappiness` yang akan digunakan.

> **Note**
>
> Jika Anda menggunakan tugas yang menggunakan tipe peluncuran Fargate, `maxSwap` parameter tidak didukung.

`sharedMemorySize`

Nilai untuk ukuran (dalam MiB) volume `/dev/shm`. Parameter ini sesuai dengan opsi `--shm-size` untuk docker run.

> **Note**
>
> Jika Anda menggunakan tugas yang menggunakan tipe peluncuran Fargate, `sharedMemorySize` parameter tidak didukung.

Jenis: Integer

`swappiness`

Anda dapat menggunakan parameter ini untuk menyetel perilaku pertukaran memori kontainer. `swappiness`Nilai `0` mencegah pertukaran terjadi kecuali diperlukan. `swappiness`Nilai `100` penyebab halaman sering ditukar. Nilai yang diterima adalah bilangan bulat antara `0` dan `100`. Jika Anda tidak menentukan nilai, nilai default `60` digunakan. Selain itu, jika Anda tidak menentukan nilai untuk`maxSwap`, maka parameter ini diabaikan. Parameter ini sesuai dengan opsi `--memory-swappiness` untuk docker run.

> **Note**
>
> Jika Anda menggunakan tugas yang menggunakan tipe peluncuran Fargate, `swappiness` parameter tidak didukung.
> Jika Anda menggunakan tugas di Amazon Linux 2023, `swappiness` parameter tidak didukung.

`tmpfs`

Jalur kontainer, opsi pemasangan, dan ukuran maksimum (dalam MiB) dudukan. tmpfs Parameter ini sesuai dengan opsi `--tmpfs` untuk docker run.

> **Note**
>
> Jika Anda menggunakan tugas yang menggunakan tipe peluncuran Fargate, `tmpfs` parameter tidak didukung.

Tipe: Array objek Tmpfs

Diperlukan: Tidak

`containerPath`

Jalur file absolut tempat tmpfs volume akan dipasang.

Tipe: String

Diperlukan: Ya

`mountOptions`

Daftar opsi pemasangan volume tmpfs.

Tipe: Array string

Wajib: Tidak

Nilai yang Valid: `"defaults" | "ro" | "rw" | "suid" | "nosuid" | "dev" | "nodev" | "exec" | "noexec" | "sync" | "async" | "dirsync" | "remount" | "mand" | "nomand" | "atime" | "noatime" | "diratime" | "nodiratime" | "bind" | "rbind" | "unbindable" | "runbindable" | "private" | "rprivate" | "shared" | "rshared" | "slave" | "rslave" | "relatime" | "norelatime" | "strictatime" | "nostrictatime" | "mode" | "uid" | "gid" | "nr_inodes" | "nr_blocks" | "mpol"`

`size`

Ukuran maksimum (dalam MiB) volume. tmpfs

Jenis: Integer

Wajib: Ya

Dependensi kontainer

`dependsOn`

Tipe: Array objek ContainerDependency

Diperlukan: Tidak

Dependensi yang ditentukan untuk pemulaian dan penonaktifan kontainer. Kontainer dapat berisi beberapa dependensi. Ketika dependensi ditentukan untuk memulai kontainer, maka akan dibalik jika ingin menonaktifkan kontainer. Misalnya, lihat Dependensi kontainer.

> **Note**
>
> Jika kontainer tidak memenuhi batasan ketergantungan atau waktu habis sebelum memenuhi batasan, Amazon ECS tidak memajukan kontainer dependen ke status berikutnya.

Untuk tugas Amazon ECS yang dihosting di instans Amazon EC2, instans memerlukan setidaknya `1.26.0` versi agen penampung untuk mengaktifkan dependensi penampung. Namun, kami menyarankan untuk menggunakan versi agen kontainer terbaru. Untuk informasi tentang memeriksa versi agen Anda dan memperbarui ke versi terbaru, lihat Memperbarui agen kontainer Amazon ECS. Jika Anda menggunakan Amazon Linux AMI Amazon ECS yang dioptimalkan Amazon, instans Anda memerlukan setidaknya `1.26.0-1` versi paket. `ecs-init` Jika instance penampung Anda diluncurkan dari versi `20190301` atau yang lebih baru, instance tersebut berisi versi yang diperlukan dari agen penampung dan`ecs-init`. Untuk informasi selengkapnya, lihat AMI Amazon ECS yang dioptimalkan.

Untuk tugas Amazon ECS yang di-host di Fargate, parameter ini mengharuskan tugas atau layanan menggunakan `1.3.0` versi platform atau yang lebih baru Linux () `1.0.0` atau Windows ().

```
"dependsOn": [
    {
        "containerName": "string",
        "condition": "string"
    }
]
```

`containerName`

Tipe: String

Wajib: Ya

Nama kontainer yang harus memenuhi syarat yang ditentukan.

`condition`

Tipe: String

Wajib: Ya

Syarat dependensi kontainer. Berikut ini adalah syarat yang tersedia dan perilakunya:

- `START`Kondisi ini meniru perilaku link dan volume hari ini. Kondisi memvalidasi bahwa kontainer dependen dimulai sebelum mengizinkan kontainer lain untuk memulai.
- `COMPLETE`— Kondisi ini memvalidasi bahwa kontainer dependen berjalan hingga selesai (keluar) sebelum mengizinkan kontainer lain untuk memulai. Ini dapat berguna untuk wadah non-esensial yang menjalankan skrip dan kemudian keluar. Kondisi ini tidak dapat diatur pada wadah penting.
- `SUCCESS`— Kondisi ini sama dengan`COMPLETE`, tetapi juga mengharuskan wadah keluar dengan `zero` status. Kondisi ini tidak dapat diatur pada wadah penting.
- `HEALTHY`- Kondisi ini memvalidasi bahwa wadah dependen melewati pemeriksaan kesehatan kontainer sebelum mengizinkan wadah lain untuk memulai. Ini mengharuskan wadah dependen memiliki pemeriksaan kesehatan yang dikonfigurasi dalam definisi tugas. Syarat ini dikonfirmasi hanya pada memulai tugas.

Waktu habis kontainer

`startTimeout`

Tipe: Integer

Wajib: Tidak

Nilai contoh: `120`

Durasi waktu (dalam detik) untuk menunggu sebelum menyerah pada penyelesaian dependensi untuk kontainer.

Misalnya, Anda menentukan dua kontainer dalam ketentuan tugas dengan `containerA` yang memiliki dependensi untuk `containerB` yang mencapai status `COMPLETE`, `SUCCESS`, atau `HEALTHY`. Jika `startTimeout` nilai ditentukan untuk `containerB` dan tidak mencapai status yang diinginkan dalam waktu itu, maka `containerA` tidak dimulai.

> **Note**
>
> Jika kontainer tidak memenuhi batasan ketergantungan atau waktu habis sebelum memenuhi batasan, Amazon ECS tidak memajukan kontainer dependen ke status berikutnya.

Untuk tugas Amazon ECS yang di-host di Fargate, parameter ini mengharuskan tugas atau layanan menggunakan `1.3.0` versi platform atau yang lebih baru Linux (). Nilai maksimumnya adalah 120 detik.

`stopTimeout`

Tipe: Integer

Wajib: Tidak

Nilai contoh: `120`

Durasi waktu (dalam detik) untuk menunggu sebelum kontainer dimatikan secara paksa jika tidak keluar dengan sendirinya secara normal.

Untuk tugas Amazon ECS yang di-host di Fargate, parameter ini mengharuskan tugas atau layanan menggunakan `1.3.0` versi platform atau yang lebih baru Linux (). Jika parameter tidak ditentukan, maka nilai default 30 detik digunakan. Nilai maksimumnya adalah 120 detik.

Untuk tugas yang menggunakan tipe peluncuran EC2, jika `stopTimeout` parameter tidak ditentukan, nilai yang ditetapkan untuk variabel konfigurasi agen penampung Amazon ECS akan `ECS_CONTAINER_STOP_TIMEOUT` digunakan. Jika `stopTimeout` parameter atau variabel konfigurasi `ECS_CONTAINER_STOP_TIMEOUT` agen tidak disetel, nilai default 30 detik untuk Linux kontainer dan 30 detik pada Windows kontainer digunakan. instans kontainer memerlukan setidaknya agen kontainer versi 1.26.0 untuk mengaktifkan nilai batas waktu penghentian kontainer. Akan tetapi, kami merekomendasikan untuk menggunakan versi agen kontainer terbaru. Untuk informasi tentang cara memeriksa versi agen Anda dan memperbarui ke versi terbaru, lihatMemperbarui agen kontainer Amazon ECS. Jika Anda menggunakan Amazon

Linux AMI Amazon ECS yang dioptimalkan Amazon, instans Anda memerlukan setidaknya versi 1.26.0-1 paket. `ecs-init` Jika instance penampung Anda diluncurkan dari versi `20190301` atau yang lebih baru, instance tersebut berisi versi yang diperlukan dari agen penampung dan`ecs-init`. Untuk informasi selengkapnya, lihat AMI Amazon ECS yang dioptimalkan.

Kontrol sistem

`systemControls`

Tipe: Objek SystemControl

Diperlukan: Tidak

Daftar parameter kernel namespace untuk diatur dalam wadah. Parameter ini memetakan ke `Sysctls` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--sysctl` untuk docker run. Misalnya, Anda dapat mengonfigurasi `net.ipv4.tcp_keepalive_time` pengaturan untuk mempertahankan koneksi yang lebih lama.

Kami tidak menyarankan Anda menentukan `systemControls` parameter terkait jaringan untuk beberapa kontainer dalam satu tugas yang juga menggunakan mode `awsvpc` atau `host` jaringan. Melakukan hal ini memiliki kelemahan sebagai berikut:

- Untuk tugas yang menggunakan mode `awsvpc` jaringan termasuk Fargate, jika Anda mengatur `systemControls` untuk penampung apa pun, itu berlaku untuk semua kontainer dalam tugas. Jika Anda menyetel berbeda `systemControls` untuk beberapa kontainer dalam satu tugas, penampung yang dimulai terakhir menentukan mana yang `systemControls` berlaku.
- Untuk tugas yang menggunakan mode `host` jaringan, namespace jaringan `systemControls` tidak didukung.

Jika Anda menyetel namespace sumber daya IPC untuk digunakan untuk kontainer dalam tugas, kondisi berikut berlaku untuk kontrol sistem Anda. Untuk informasi selengkapnya, lihat Mode IPC.

- Untuk tugas yang menggunakan mode `host` IPC, namespace `systemControls` IPC tidak didukung.
- Untuk tugas yang menggunakan mode `task` IPC, `systemControls` nilai namespace IPC berlaku untuk semua kontainer dalam tugas.

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

> **Note**
>
> Parameter ini hanya didukung untuk tugas yang di-host AWS Fargate jika tugas menggunakan versi platform `1.4.0` atau yang lebih baru (Linux). Ini tidak didukung untuk wadah Windows di Fargate.

```
"systemControls": [
    {
         "namespace":"string",
         "value":"string"
    }
]
```

`namespace`

Tipe: String

Wajib: Tidak

Parameter kernel namespace untuk menetapkan untuk. `value`

Nilai namespace IPC yang valid:`"kernel.msgmax" | "kernel.msgmnb" | "kernel.msgmni" | "kernel.sem" | "kernel.shmall" | "kernel.shmmax" | "kernel.shmmni" | "kernel.shm_rmid_forced"`, dan `Sysctls` itu dimulai dengan `"fs.mqueue.*"`

Nilai namespace jaringan yang valid: `Sysctls` yang dimulai dengan `"net.*"`

Semua nilai ini didukung oleh Fargate.

`value`

Tipe: String

Wajib: Tidak

Nilai untuk parameter kernel namespace yang ditentukan dalam. `namespace`

Interaktif

`interactive`

Tipe: Boolean

Wajib: Tidak

Ketika parameter ini`true`, Anda dapat menerapkan aplikasi kontainer yang memerlukan `stdin` atau `tty` dialokasikan. Parameter ini memetakan ke `OpenStdin` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--interactive` untuk docker run.

Nilai default-nya `false`.

Terminal semu

`pseudoTerminal`

Tipe: Boolean

Wajib: Tidak

Jika parameter ini `true`, TTY dialokasikan. Parameter ini memetakan ke `Tty` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--tty` untuk docker run.

Nilai default-nya `false`.

Nama akselerator Elastic Inference

> **Note**
>
> Mulai 15 April 2023, tidak AWS akan memasukkan pelanggan baru ke Amazon Elastic Inference (EI), dan akan membantu pelanggan saat ini memigrasikan beban kerja mereka ke opsi yang menawarkan harga dan kinerja yang lebih baik. Setelah 15 April 2023, pelanggan baru tidak akan dapat meluncurkan instans dengan akselerator Amazon EI di Amazon, Amazon ECS, atau SageMaker Amazon EC2. Namun, pelanggan yang telah menggunakan Amazon EI setidaknya sekali selama periode 30 hari terakhir dianggap sebagai pelanggan saat ini dan akan dapat terus menggunakan layanan ini.

Persyaratan sumber daya akselerator Elastic Inference untuk definisi tugas Anda. Untuk informasi lebih lanjut, lihat Apa itu Amazon Elastic Inference? di Panduan Pengembang Amazon Elastic Inference.

Parameter berikut diizinkan dalam definisi tugas:

`deviceName`

Tipe: String

Diperlukan: Ya

Nama perangkat akselerator Elastic Inference. Itu juga `deviceName` harus direferensikan dalam definisi kontainer lihat`Elastic Inference accelerator`.

`deviceType`

Tipe: String

Diperlukan: Ya

Akselerator Elastic Inference untuk digunakan.

Kendala penempatan tugas

Saat mendaftarkan definisi tugas, Anda dapat memberikan batasan penempatan tugas yang menyesuaikan cara Amazon ECS menempatkan tugas.

Jika Anda menggunakan tipe peluncuran Fargate, batasan penempatan tugas tidak didukung. Secara default, tugas Fargate tersebar di Availability Zone.

Untuk tugas yang menggunakan tipe peluncuran EC2, Anda dapat menggunakan batasan untuk menempatkan tugas berdasarkan Availability Zone, tipe instans, atau atribut kustom. Untuk informasi selengkapnya, lihat Tentukan instance kontainer mana yang digunakan Amazon ECS untuk tugas.

Parameter berikut diizinkan dalam ketentuan kontainer:

`expression`

Tipe: String

Wajib: Tidak

Ekspresi bahasa kueri klaster yang akan diterapkan pada batasan. Untuk informasi selengkapnya, lihat Buat ekspresi untuk menentukan instance kontainer untuk tugas Amazon ECS.

`type`

Tipe: String

Diperlukan: Ya

Tipe batasan. Gunakan `memberOf` untuk membatasi pemilihan ke grup kandidat yang valid.

Konfigurasi proxy

`proxyConfiguration`

Tipe: Objek ProxyConfiguration

Diperlukan: Tidak

Detail konfigurasi untuk proxy App Mesh.

Untuk tugas yang menggunakan tipe peluncuran EC2, instance container memerlukan setidaknya versi 1.26.0 dari agen kontainer dan setidaknya versi 1.26.0-1 paket untuk mengaktifkan konfigurasi proxy. `ecs-init` Jika instans kontainer Anda diluncurkan dari AMI yang dioptimalkan Amazon ECS versi `20190301` atau yang lebih baru, maka instans kontainer mengandung versi agen kontainer dan `ecs-init` yang diperlukan. Untuk informasi selengkapnya, lihat AMI Amazon ECS yang dioptimalkan.

Untuk tugas yang menggunakan tipe peluncuran Fargate, fitur ini mengharuskan tugas atau layanan menggunakan platform versi 1.3.0 atau yang lebih baru.

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

```
"proxyConfiguration": {
    "type": "APPMESH",
    "containerName": "string",
    "properties": [
        {
           "name": "string",
```

```
            "value": "string"
        }
    ]
}
```

`type`

Jenis: String

Nilai nilai: `APPMESH`

Wajib: Tidak

Tipe proxy. Satu-satunya nilai yang didukung adalah `APPMESH`.

`containerName`

Tipe: String

Diperlukan: Ya

Nama wadah yang berfungsi sebagai proxy App Mesh.

`properties`

Tipe: Array objek KeyValuePair

Diperlukan: Tidak

Set parameter konfigurasi jaringan berfungsi untuk menyediakan plugin Container Network Interface (CNI), yang ditetapkan sebagai pasangan nilai kunci.

- `IgnoredUID`— (Wajib) ID pengguna (UID) dari wadah proxy sebagaimana didefinisikan oleh `user` parameter dalam definisi kontainer. Ini digunakan untuk memastikan bahwa proksi mengabaikan lalu lintasnya sendiri. Jika `IgnoredGID` ditentukan, bidang ini tidak boleh kosong.
- `IgnoredGID`— (Wajib) ID grup (GID) dari wadah proxy sebagaimana didefinisikan oleh `user` parameter dalam definisi kontainer. Ini digunakan untuk memastikan bahwa proksi mengabaikan lalu lintasnya sendiri. Jika `IgnoredUID` ditentukan, bidang ini tidak boleh kosong.
- `AppPorts`— (Wajib) Daftar port yang digunakan aplikasi. Lalu lintas jaringan ke port ini diteruskan ke `ProxyIngressPort` dan `ProxyEgressPort`.
- `ProxyIngressPort`— (Wajib) Menentukan port yang lalu lintas masuk ke `AppPorts` diarahkan.

- `ProxyEgressPort`— (Wajib) Menentukan port yang lalu lintas keluar dari `AppPorts` diarahkan ke.
- `EgressIgnoredPorts`— (Wajib) Lalu lintas keluar yang menuju ke port yang ditentukan ini diabaikan dan tidak dialihkan ke port. `ProxyEgressPort` Ia bisa berupa daftar kosong.
- `EgressIgnoredIPs`— (Wajib) Lalu lintas keluar yang menuju ke alamat IP yang ditentukan ini diabaikan dan tidak dialihkan ke. `ProxyEgressPort` Ia bisa berupa daftar kosong.

`name`

Tipe: String

Wajib: Tidak

Nama pasangan nilai kunci.

`value`

Tipe: String

Wajib: Tidak

Nilai pasangan nilai kunci.

Volume

Saat Anda mendaftarkan definisi tugas, Anda dapat secara opsional menentukan daftar volume yang akan diteruskan ke Docker daemon pada instance container, yang kemudian tersedia untuk diakses oleh container lain pada instance container yang sama.

Berikut ini adalah jenis volume data yang dapat digunakan:

- Volume Amazon EBS - Menyediakan penyimpanan blok yang hemat biaya, tahan lama, dan berkinerja tinggi untuk beban kerja kontainer intensif data. Anda dapat melampirkan 1 volume Amazon EBS per tugas Amazon ECS saat menjalankan tugas mandiri, atau saat membuat atau memperbarui layanan. Volume Amazon EBS didukung untuk tugas Linux yang dihosting di instans Fargate atau Amazon EC2. Untuk informasi selengkapnya, lihat Volume Amazon EBS.
- Volume Amazon EFS - Menyediakan penyimpanan file yang sederhana, dapat diskalakan, dan persisten untuk digunakan dengan tugas Amazon ECS Anda. Kapasitas penyimpanan Amazon EFS bersifat elastis. Itu tumbuh dan menyusut secara otomatis saat Anda menambah dan menghapus file. Aplikasi Anda dapat memiliki penyimpanan yang mereka butuhkan dan ketika

mereka membutuhkannya. Volume Amazon EFS didukung untuk tugas yang di-host di instans Fargate atau Amazon EC2. Untuk informasi selengkapnya, lihat Volume Amazon EFS.

- Volume FSx for Windows File Server - Menyediakan server file Microsoft Windows yang dikelola sepenuhnya. Server file ini didukung oleh sistem file Windows. Saat menggunakan FSx for Windows File Server bersama dengan Amazon ECS, Anda dapat menyediakan tugas Windows Anda dengan penyimpanan file persisten, terdistribusi, bersama, dan statis. Untuk informasi selengkapnya, lihat Volume FSx for Windows File Server.

  Wadah Windows di Fargate tidak mendukung opsi ini.
- Volume Docker — Volume yang dikelola Docker yang dibuat di bawah instans `/var/lib/docker/volumes` Amazon EC2 host. Driver volume Docker (juga disebut sebagai plugin) digunakan untuk mengintegrasikan volume dengan sistem penyimpanan eksternal, seperti Amazon EBS. Built-in driver volume `local` atau driver volume pihak ke tiga dapat digunakan. Volume Docker hanya didukung saat menjalankan tugas di instans Amazon EC2. Wadah Windows hanya mendukung penggunaan `local` driver. Untuk menggunakan volume Docker, tentukan `dockerVolumeConfiguration` dalam ketentuan tugas Anda. Untuk informasi selengkapnya, lihat Menggunakan volume.
- Bind mount — File atau direktori pada mesin host yang dipasang ke dalam wadah. Volume host mount bind didukung saat menjalankan tugas pada instans AWS Fargate atau Amazon EC2. Untuk menggunakan volume host bind mount, tentukan `sourcePath` nilai `host` dan opsional dalam definisi tugas Anda. Untuk informasi selengkapnya, lihat Menggunakan bind mount.

Untuk informasi selengkapnya, lihat Menggunakan volume data dalam tugas.

Parameter berikut diizinkan dalam ketentuan kontainer.

`name`

Tipe: String

Wajib: Tidak

Nama volume. Hingga 255 huruf (huruf besar dan kecil), angka, tanda hubung (), dan garis bawah (-) diperbolehkan. _ Nama ini direferensikan dalam `sourceVolume` parameter `mountPoints` objek definisi kontainer.

`host`

Diperlukan: Tidak

`host`Parameter ini digunakan untuk mengikat siklus hidup bind mount ke instans Amazon EC2 host, bukan tugas, dan tempat penyimpanannya. Jika `host` parameternya kosong, maka daemon Docker menetapkan jalur host untuk volume data Anda, tetapi data tidak dijamin akan bertahan setelah wadah yang terkait dengannya berhenti berjalan.

Kontainer Windows dapat memasang seluruh direktori pada drive yang sama dengan `$env:ProgramData`.

> **Note**
>
> `sourcePath`Parameter hanya didukung saat menggunakan tugas yang di-host di instans Amazon EC2.

`sourcePath`

Tipe: String

Wajib: Tidak

Saat `host` parameter digunakan, tentukan a `sourcePath` untuk mendeklarasikan jalur pada instans Amazon EC2 host yang disajikan ke wadah. Jika parameter ini kosong, daemon Docker akan menetapkan jalur host untuk Anda. Jika `host` parameter berisi lokasi `sourcePath` file, maka volume data tetap ada di lokasi yang ditentukan pada instans Amazon EC2 host hingga Anda menghapusnya secara manual. Jika `sourcePath` nilai tidak ada pada instans Amazon EC2 host, daemon Docker membuatnya. Jika lokasinya memang ada, konten dari folder jalur sumber diekspor.

`configuredAtLaunch`

Tipe: Boolean

Wajib: Tidak

Menentukan apakah volume dikonfigurasi saat peluncuran. Saat disetel ke`true`, Anda dapat mengonfigurasi volume saat menjalankan tugas mandiri, atau saat membuat atau memperbarui layanan. Saat disetel ke`true`, Anda tidak akan dapat memberikan konfigurasi volume lain dalam definisi tugas. Parameter ini harus diatur `true` untuk mengonfigurasi volume Amazon EBS untuk lampiran ke tugas. Menyetel `configuredAtLaunch` ke `true` dan menunda konfigurasi volume ke fase peluncuran memungkinkan Anda membuat definisi tugas yang tidak dibatasi untuk jenis

volume atau pengaturan volume tertentu. Melakukan hal ini membuat definisi tugas Anda dapat digunakan kembali di berbagai lingkungan eksekusi. Untuk informasi selengkapnya, lihat volume Amazon EBS.

`dockerVolumeConfiguration`

Jenis: DockerVolumeConfigurationObjek

Diperlukan: Tidak

Parameter ini ditentukan saat menggunakan volume Docker. Volume Docker hanya didukung saat menjalankan tugas pada instans EC2. Wadah Windows hanya mendukung penggunaan `local` driver. Untuk menggunakan pemasangan mengikat, tentukan `host` saja.

`scope`

Tipe: String

Nilai yang Valid: `task` | `shared`

Diperlukan: Tidak

Cakupan untuk volume Docker, yang menentukan siklus hidupnya. Volume Docker yang tercakup ke `task` secara otomatis disediakan saat tugas dimulai dan dihancurkan saat tugas berhenti. Volume Docker yang tercakup sebagai `shared` dipertahankan setelah tugas berhenti.

`autoprovision`

Jenis: Boolean

Nilai default: `false`

Diperlukan: Tidak

Jika nilai ini`true`, volume Docker dibuat jika belum ada. Bidang ini hanya digunakan jika `scope` ada`shared`. Jika `scope` ya`task`, maka parameter ini harus dihilangkan atau disetel ke. `false`

`driver`

Tipe: String

Wajib: Tidak

Driver volume Docker yang digunakan. Nilai driver harus sesuai dengan nama driver yang disediakan oleh Docker karena nama ini digunakan untuk penempatan tugas. Jika driver diinstal dengan menggunakan CLI plugin Docker, `docker plugin ls` gunakan untuk mengambil nama driver dari instance container Anda. Jika driver diinstal dengan menggunakan metode lain, gunakan penemuan plugin Docker untuk mengambil nama driver. Untuk informasi lebih lanjut, lihat Penemuan plugin Docker. Parameter ini dipetakan ke `Driver` bagian Create a volume dari Docker Remote API dan `--driver` opsi untuk `docker volume create`.

`driverOpts`

Tipe: String

Wajib: Tidak

Peta opsi khusus driver Docker untuk dilewati. Parameter ini dipetakan ke `DriverOpts` bagian Create a volume dari Docker Remote API dan `--opt` opsi untuk `docker volume create`.

`labels`

Tipe: String

Wajib: Tidak

Metadata kustom untuk ditambahkan ke volume Docker. Parameter ini dipetakan ke `Labels` bagian Create a volume dari Docker Remote API dan `--label` opsi untuk `docker volume create`.

`efsVolumeConfiguration`

Jenis: EFS VolumeConfiguration Object

Diperlukan: Tidak

Parameter ini ditentukan saat menggunakan volume Amazon EFS.

`fileSystemId`

Tipe: String

Diperlukan: Ya

ID sistem file Amazon EFS yang akan digunakan.

`rootDirectory`

Tipe: String

Wajib: Tidak

Direktori dalam sistem file Amazon EFS untuk dipasang sebagai direktori root di dalam host. Jika parameter ini dihilangkan, root volume Amazon EFS akan digunakan. Menentukan `/` memiliki efek yang sama seperti mengabaikan parameter ini.

> **Important**
>
> Jika titik akses EFS ditentukan dalam`authorizationConfig`, parameter direktori root harus dihilangkan atau disetel ke`/`, yang akan menerapkan jalur yang disetel pada titik akses EFS.

`transitEncryption`

Tipe: String

Nilai yang valid: `ENABLED` | `DISABLED`

Diperlukan: Tidak

Menentukan apakah akan mengaktifkan enkripsi untuk data Amazon EFS dalam perjalanan antara host Amazon ECS dan server Amazon EFS. Jika otorisasi Amazon EFS IAM digunakan, enkripsi transit harus diaktifkan. Jika parameter ini diabaikan, nilai default `DISABLED` akan digunakan. Untuk informasi lebih lanjut, lihat Pengenkripsian Data Saat Transit di Panduan Pengguna Amazon Elastic File System.

`transitEncryptionPort`

Tipe: Integer

Wajib: Tidak

Port yang akan digunakan saat mengirim data terenkripsi antara host Amazon ECS dan server Amazon EFS. Jika Anda tidak menentukan port enkripsi transit, tugas akan menggunakan strategi pemilihan port yang digunakan oleh helper mount Amazon EFS. Untuk informasi lebih lanjut, lihat Pembantu Pemasangan EFS di Panduan Pengguna Amazon Elastic File System.

`authorizationConfig`

Jenis: EFS AuthorizationConfiguration Object

Diperlukan: Tidak

Detail konfigurasi otorisasi untuk sistem file Amazon EFS.

`accessPointId`

Tipe: String

Wajib: Tidak

ID titik akses yang akan digunakan. Jika titik akses ditentukan, nilai direktori root di dalam `efsVolumeConfiguration` harus dihilangkan atau disetel ke`/`, yang akan memberlakukan jalur yang ditetapkan pada titik akses EFS. Jika titik akses digunakan, enkripsi transit harus diaktifkan di `EFSVolumeConfiguration`. Untuk informasi lebih lanjut, lihat Bekerja dengan Titik Akses Amazon EFS dalam Panduan Pengguna Amazon Elastic File System.

`iam`

Tipe: String

Nilai yang valid: `ENABLED` | `DISABLED`

Diperlukan: Tidak

Menentukan apakah akan menggunakan peran IAM tugas Amazon ECS yang ditentukan dalam definisi tugas saat memasang sistem file Amazon EFS. Jika diaktifkan, enkripsi transit harus diaktifkan di `EFSVolumeConfiguration`. Jika Anda menghilangkan parameter ini, nilai default `DISABLED` akan digunakan. Untuk informasi selengkapnya, lihat IAM Role pada Tugas.

`FSxWindowsFileServerVolumeConfiguration`

Tipe: F SxWindowsFileServerVolumeConfiguration Objek

Diperlukan: Ya

Parameter ini ditentukan saat Anda menggunakan sistem file Amazon FSx for Windows File Server untuk penyimpanan tugas.

`fileSystemId`

Tipe: String

Diperlukan: Ya

ID sistem file FSx for Windows File Server untuk digunakan.

`rootDirectory`

Tipe: String

Diperlukan: Ya

Direktori dalam sistem file FSx for Windows File Server untuk dipasang sebagai direktori root di dalam host.

`authorizationConfig`

`credentialsParameter`

Tipe: String

Diperlukan: Ya

Opsi kredensi otorisasi.

pilihan:

- Nama Sumber Daya Amazon (ARN) dari sebuah AWS Secrets Managerrahasia.
- ARN dari suatu AWS Systems Managerparameter.

`domain`

Tipe: String

Diperlukan: Ya

Nama domain yang sepenuhnya memenuhi syarat yang dihosting oleh direktori AWS Directory Service for Microsoft Active Directory(AWS Managed Microsoft AD) atau Direktori Aktif EC2 yang dihosting sendiri.

Tag

Ketika mendaftarkan ketentuan tugas, Anda secara opsional dapat menentukan tanda metadata yang diterapkan ke ketentuan tugas. Tanda membantu mengategorikan dan mengatur ketentuan

tugas Anda. Setiap tanda terdiri dari kunci dan nilai opsional. Anda mendefinisikan keduanya. Untuk informasi selengkapnya, lihat Penandaan sumber daya Amazon ECS.

> **Important**
>
> Jangan menambahkan informasi identitas pribadi atau informasi rahasia atau sensitif lainnya dalam tag. Tag dapat diakses oleh banyak AWS layanan, termasuk penagihan. Tag tidak dimaksudkan untuk digunakan untuk data pribadi atau sensitif.

Parameter berikut diperbolehkan dalam objek tanda.

`key`

Tipe: String

Wajib: Tidak

Satu bagian dari pasangan nilai kunci yang membentuk tanda. Kunci adalah label umum yang bertindak seperti kategori untuk nilai tanda yang lebih spesifik.

`value`

Tipe: String

Wajib: Tidak

Bagian opsional pasangan nilai kunci yang membentuk tanda. Nilai bertindak sebagai deskriptor dalam kategori tanda (kunci).

Parameter ketentuan tugas lainnya

Parameter definisi tugas berikut dapat digunakan saat mendaftarkan definisi tugas di konsol Amazon ECS dengan menggunakan opsi Configure via JSON. Untuk informasi selengkapnya, lihat Membuat definisi tugas menggunakan konsol.

Topik

- Penyimpanan sementara
- Mode IPC
- Mode PID

Penyimpanan sementara

`ephemeralStorage`

Tipe: Objek EphemeralStorage

Diperlukan: Tidak

Jumlah penyimpanan sementara (dalam GB) untuk dialokasikan untuk tugas tersebut. Parameter ini digunakan untuk memperluas jumlah total penyimpanan sementara yang tersedia, di luar jumlah default, untuk tugas yang di-host. AWS Fargate Untuk informasi selengkapnya, lihat the section called “Pemasangan terikat”.

> **Note**
>
> Parameter ini hanya didukung untuk tugas-tugas yang di-host AWS Fargate menggunakan versi platform `1.4.0` atau yang lebih baru (Linux) `1.0.0` atau yang lebih baru (Windows).

Mode IPC

`ipcMode`

Tipe: String

Wajib: Tidak

Namespace sumber daya IPC yang digunakan untuk kontainer dalam tugas. Nilai yang valid adalah `host`, `task`, atau `none`. Jika `host` ditentukan, maka semua kontainer yang berada dalam tugas yang menentukan mode `host` IPC pada instance kontainer yang sama berbagi sumber daya IPC yang sama dengan instans Amazon EC2 host. Jika `task` ditentukan, semua kontainer yang berada dalam tugas yang ditentukan berbagi sumber daya IPC yang sama. Jika `none` ditentukan, maka sumber daya IPC dalam kontainer tugas bersifat privat dan tidak dibagi dengan kontainer lain dalam tugas atau pada instans kontainer. Jika tidak ada nilai yang ditentukan, maka berbagi namespace sumber daya IPC tergantung pada pengaturan daemon Docker pada instans kontainer. Untuk informasi selengkapnya, lihat pengaturan IPC di Referensi Docker run.

Jika mode `host` IPC digunakan, ada risiko tinggi eksposur namespace IPC yang tidak diinginkan. Untuk informasi selengkapnya, lihat Keamanan Docker.

Jika Anda menyetel parameter kernel namespace yang digunakan `systemControls` untuk kontainer dalam tugas, hal berikut ini berlaku untuk namespace sumber daya IPC Anda. Untuk informasi selengkapnya, lihat Kontrol sistem.

- Untuk tugas yang menggunakan mode `host` IPC, namespace IPC yang terkait `systemControls` tidak didukung.
- Untuk tugas yang menggunakan mode `task` IPC, `systemControls` yang berhubungan dengan namespace IPC berlaku untuk semua kontainer dalam tugas.

> **Note**
>
> Parameter ini tidak didukung untuk wadah atau tugas Windows menggunakan tipe peluncuran Fargate.

Mode PID

`pidMode`

Tipe: String

Nilai yang Valid: `host` | `task`

Diperlukan: Tidak

Namspace proses yang akan digunakan untuk kontainer dalam tugas. Nilai yang valid adalah `host` atau `task`. Pada Fargate untuk wadah Linux, satu-satunya nilai yang valid adalah. `task` Misalnya, pemantauan sidecar mungkin `pidMode` perlu mengakses informasi tentang kontainer lain yang berjalan dalam tugas yang sama.

Jika `host` ditentukan, semua kontainer dalam tugas yang menentukan mode `host` PID pada instance container yang sama berbagi namespace proses yang sama dengan instans Amazon EC2 host.

Jika `task` ditentukan, semua kontainer dalam tugas tertentu berbagi namespace proses yang sama.

Jika tidak ada nilai yang ditentukan, defaultnya adalah namespace pribadi untuk setiap kontainer. Untuk informasi selengkapnya, lihat Pengaturan PID di Referensi Docker run.

Jika mode `host` PID digunakan, ada risiko tinggi eksposur namespace proses yang tidak diinginkan. Untuk informasi selengkapnya, lihat Keamanan Docker.

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

> **Note**
>
> Parameter ini hanya didukung untuk tugas yang di-host AWS Fargate jika tugas menggunakan versi platform `1.4.0` atau yang lebih baru (Linux). Ini tidak didukung untuk wadah Windows di Fargate.

Templat ketentuan tugas

Templat ketentuan tugas kosong ditampilkan sebagai berikut. Anda dapat menggunakan template ini untuk membuat definisi tugas Anda, yang kemudian dapat ditempelkan ke area input JSON konsol atau disimpan ke file dan digunakan dengan opsi. AWS CLI `--cli-input-json` Untuk informasi selengkapnya, lihat Parameter ketentuan tugas.

Templat jenis peluncuran Amazon EC2

```
{
    "family": "",
    "taskRoleArn": "",
    "executionRoleArn": "",
    "networkMode": "none",
    "containerDefinitions": [
        {
            "name": "",
            "image": "",
            "repositoryCredentials": {
                "credentialsParameter": ""
            },
            "cpu": 0,
            "memory": 0,
            "memoryReservation": 0,
            "links": [
```

```
                ""
            ],
            "portMappings": [
                {
                    "containerPort": 0,
                    "hostPort": 0,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": [
                ""
            ],
            "command": [
                ""
            ],
            "environment": [
                {
                    "name": "",
                    "value": ""
                }
            ],
            "environmentFiles": [
                {
                    "value": "",
                    "type": "s3"
                }
            ],
            "mountPoints": [
                {
                    "sourceVolume": "",
                    "containerPath": "",
                    "readOnly": true
                }
            ],
            "volumesFrom": [
                {
                    "sourceContainer": "",
                    "readOnly": true
                }
            ],
            "linuxParameters": {
                "capabilities": {
                    "add": [
```

```
                    ""
                ],
                "drop": [
                    ""
                ]
            },
            "devices": [
                {
                    "hostPath": "",
                    "containerPath": "",
                    "permissions": [
                        "read"
                    ]
                }
            ],
            "initProcessEnabled": true,
            "sharedMemorySize": 0,
            "tmpfs": [
                {
                    "containerPath": "",
                    "size": 0,
                    "mountOptions": [
                        ""
                    ]
                }
            ],
            "maxSwap": 0,
            "swappiness": 0
        },
        "secrets": [
            {
                "name": "",
                "valueFrom": ""
            }
        ],
        "dependsOn": [
            {
                "containerName": "",
                "condition": "COMPLETE"
            }
        ],
        "startTimeout": 0,
        "stopTimeout": 0,
        "hostname": "",
```

```
            "user": "",
            "workingDirectory": "",
            "disableNetworking": true,
            "privileged": true,
            "readonlyRootFilesystem": true,
            "dnsServers": [
                ""
            ],
            "dnsSearchDomains": [
                ""
            ],
            "extraHosts": [
                {
                    "hostname": "",
                    "ipAddress": ""
                }
            ],
            "dockerSecurityOptions": [
                ""
            ],
            "interactive": true,
            "pseudoTerminal": true,
            "dockerLabels": {
                "KeyName": ""
            },
            "ulimits": [
                {
                    "name": "nofile",
                    "softLimit": 0,
                    "hardLimit": 0
                }
            ],
            "logConfiguration": {
                "logDriver": "splunk",
                "options": {
                    "KeyName": ""
                },
                "secretOptions": [
                    {
                        "name": "",
                        "valueFrom": ""
                    }
                ]
            },
```

```
            "healthCheck": {
                "command": [
                    ""
                ],
                "interval": 0,
                "timeout": 0,
                "retries": 0,
                "startPeriod": 0
            },
            "systemControls": [
                {
                    "namespace": "",
                    "value": ""
                }
            ],
            "resourceRequirements": [
                {
                    "value": "",
                    "type": "InferenceAccelerator"
                }
            ],
            "firelensConfiguration": {
                "type": "fluentbit",
                "options": {
                    "KeyName": ""
                }
            }
        }
    ],
    "volumes": [
        {
            "name": "",
            "host": {
                "sourcePath": ""
            },
            "configuredAtLaunch": true,
            "dockerVolumeConfiguration": {
                "scope": "shared",
                "autoprovision": true,
                "driver": "",
                "driverOpts": {
                    "KeyName": ""
                },
                "labels": {
```

```
                    "KeyName": ""
                }
            },
            "efsVolumeConfiguration": {
                "fileSystemId": "",
                "rootDirectory": "",
                "transitEncryption": "DISABLED",
                "transitEncryptionPort": 0,
                "authorizationConfig": {
                    "accessPointId": "",
                    "iam": "ENABLED"
                }
            },
            "fsxWindowsFileServerVolumeConfiguration": {
                "fileSystemId": "",
                "rootDirectory": "",
                "authorizationConfig": {
                    "credentialsParameter": "",
                    "domain": ""
                }
            }
        }
    ],
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": ""
        }
    ],
    "requiresCompatibilities": [
        "EC2"
    ],
    "cpu": "",
    "memory": "",
    "tags": [
        {
            "key": "",
            "value": ""
        }
    ],
    "pidMode": "task",
    "ipcMode": "task",
    "proxyConfiguration": {
        "type": "APPMESH",
```

```
        "containerName": "",
        "properties": [
            {
                "name": "",
                "value": ""
            }
        ]
    },
    "inferenceAccelerators": [
        {
            "deviceName": "",
            "deviceType": ""
        }
    ],
    "ephemeralStorage": {
        "sizeInGiB": 0
    },
    "runtimePlatform": {
        "cpuArchitecture": "X86_64",
        "operatingSystemFamily": "WINDOWS_SERVER_20H2_CORE"
    }
}
```

Templat jenis peluncuran Fargate

Important

Untuk tipe peluncuran Fargate, Anda harus menyertakan `operatingSystemFamily` parameter dengan salah satu nilai berikut:

- `LINUX`
- `WINDOWS_SERVER_2019_FULL`
- `WINDOWS_SERVER_2019_CORE`
- `WINDOWS_SERVER_2022_FULL`
- `WINDOWS_SERVER_2022_CORE`

```
{
    "family": "",
```

```
    "runtimePlatform": {"operatingSystemFamily": ""},
    "taskRoleArn": "",
    "executionRoleArn": "",
    "networkMode": "awsvpc",
    "platformFamily": "",
    "containerDefinitions": [
        {
            "name": "",
            "image": "",
            "repositoryCredentials": {"credentialsParameter": ""},
            "cpu": 0,
            "memory": 0,
            "memoryReservation": 0,
            "links": [""],
            "portMappings": [
                {
                    "containerPort": 0,
                    "hostPort": 0,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": [""],
            "command": [""],
            "environment": [
                {
                    "name": "",
                    "value": ""
                }
            ],
            "environmentFiles": [
                {
                    "value": "",
                    "type": "s3"
                }
            ],
            "mountPoints": [
                {
                    "sourceVolume": "",
                    "containerPath": "",
                    "readOnly": true
                }
            ],
            "volumesFrom": [
```

```
            {
                "sourceContainer": "",
                "readOnly": true
            }
        ],
        "linuxParameters": {
            "capabilities": {
                "add": [""],
                "drop": [""]
            },
            "devices": [
                {
                    "hostPath": "",
                    "containerPath": "",
                    "permissions": ["read"]
                }
            ],
            "initProcessEnabled": true,
            "sharedMemorySize": 0,
            "tmpfs": [
                {
                    "containerPath": "",
                    "size": 0,
                    "mountOptions": [""]
                }
            ],
            "maxSwap": 0,
            "swappiness": 0
        },
        "secrets": [
            {
                "name": "",
                "valueFrom": ""
            }
        ],
        "dependsOn": [
            {
                "containerName": "",
                "condition": "HEALTHY"
            }
        ],
        "startTimeout": 0,
        "stopTimeout": 0,
        "hostname": "",
```

```
            "user": "",
            "workingDirectory": "",
            "disableNetworking": true,
            "privileged": true,
            "readonlyRootFilesystem": true,
            "dnsServers": [""],
            "dnsSearchDomains": [""],
            "extraHosts": [
                {
                    "hostname": "",
                    "ipAddress": ""
                }
            ],
            "dockerSecurityOptions": [""],
            "interactive": true,
            "pseudoTerminal": true,
            "dockerLabels": {"KeyName": ""},
            "ulimits": [
                {
                    "name": "msgqueue",
                    "softLimit": 0,
                    "hardLimit": 0
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {"KeyName": ""},
                "secretOptions": [
                    {
                        "name": "",
                        "valueFrom": ""
                    }
                ]
            },
            "healthCheck": {
                "command": [""],
                "interval": 0,
                "timeout": 0,
                "retries": 0,
                "startPeriod": 0
            },
            "systemControls": [
                {
                    "namespace": "",
```

```
                    "value": ""
                }
            ],
            "resourceRequirements": [
                {
                    "value": "",
                    "type": "GPU"
                }
            ],
            "firelensConfiguration": {
                "type": "fluentd",
                "options": {"KeyName": ""}
            }
        }
    ],
    "volumes": [
        {
            "name": "",
            "host": {"sourcePath": ""},
            "configuredAtLaunch":true,
            "dockerVolumeConfiguration": {
                "scope": "task",
                "autoprovision": true,
                "driver": "",
                "driverOpts": {"KeyName": ""},
                "labels": {"KeyName": ""}
            },
            "efsVolumeConfiguration": {
                "fileSystemId": "",
                "rootDirectory": "",
                "transitEncryption": "ENABLED",
                "transitEncryptionPort": 0,
                "authorizationConfig": {
                    "accessPointId": "",
                    "iam": "ENABLED"
                }
            }
        }
    ],
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": ""
        }
```

```
    ],
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "",
    "memory": "",
    "tags": [
        {
            "key": "",
            "value": ""
        }
    ],
    "ephemeralStorage": {"sizeInGiB": 0},
    "pidMode": "task",
    "ipcMode": "none",
    "proxyConfiguration": {
        "type": "APPMESH",
        "containerName": "",
        "properties": [
            {
                "name": "",
                "value": ""
            }
        ]
    },
    "inferenceAccelerators": [
        {
            "deviceName": "",
            "deviceType": ""
        }
    ]
}
```

Anda dapat membuat template definisi tugas ini menggunakan AWS CLI perintah berikut.

```
aws ecs register-task-definition --generate-cli-skeleton
```

Parameter ketentuan layanan

Definisi layanan menentukan cara menjalankan layanan Amazon ECS Anda. Parameter berikut dapat ditetapkan dalam ketentuan layanan.

Tipe peluncuran

`launchType`

Tipe: String

Nilai yang valid: `EC2` | `FARGATE` | `EXTERNAL`

Diperlukan: Tidak

Jenis peluncuran untuk menjalankan layanan Anda. Jika jenis peluncuran tidak ditentukan, default `capacityProviderStrategy` digunakan secara default. Untuk informasi selengkapnya, lihat Jenis peluncuran Amazon ECS.

Jika `launchType` ditentukan, parameter `capacityProviderStrategy` harus dihilangkan.

Strategi penyedia kapasitas

`capacityProviderStrategy`

Tipe: Array objek

Wajib: Tidak

Strategi penyedia kapasitas untuk digunakan untuk layanan.

Strategi penyedia kapasitas terdiri dari satu penyedia kapasitas atau lebih bersama dengan `base` dan `weight` yang ditetapkan padanya. Penyedia kapasitas harus dikaitkan dengan klaster sebelum ditentukan dalam strategi penyedia kapasitas. PutClusterCapacityProviders API digunakan untuk mengaitkan penyedia kapasitas dengan cluster. Hanya penyedia kapasitas dengan status `ACTIVE` atau `UPDATING` yang dapat digunakan.

Jika `capacityProviderStrategy` ditetapkan, parameter `launchType` harus dihilangkan. Jika tidak ada `capacityProviderStrategy` atau `launchType` ditentukan, `defaultCapacityProviderStrategy` untuk klaster digunakan.

Jika Anda ingin menentukan penyedia kapasitas yang menggunakan grup Auto Scaling, penyedia kapasitas harus sudah dibuat. Penyedia kapasitas baru dapat dibuat dengan operasi CreateCapacityProvider API.

Untuk menggunakan penyedia kapasitas AWS Fargate, tentukan penyedia `FARGATE_SPOT` kapasitas `FARGATE` atau penyedia. Penyedia kapasitas AWS Fargate tersedia untuk semua akun dan hanya perlu dikaitkan dengan cluster yang akan digunakan.

Operasi PutClusterCapacityProviders API digunakan untuk memperbarui daftar penyedia kapasitas yang tersedia untuk klaster setelah cluster dibuat.

`capacityProvider`

Tipe: String

Diperlukan: Ya

Nama pendek atau nama sumber daya Amazon lengkap (ARN) dari penyedia kapasitas.

`weight`

Jenis: Integer

Kisaran yang valid: Bilangan bulat antara 0 dan 1.000.

Diperlukan: Tidak

Nilai bobot menunjukkan persentase relatif dari jumlah total tugas yang diluncurkan yang menggunakan penyedia kapasitas yang ditentukan.

Misalnya, asumsikan bahwa Anda memiliki strategi yang berisi dua penyedia kapasitas dan keduanya memiliki bobot satu. Ketika basis terpenuhi, tugas-tugas terbagi secara merata di dua penyedia kapasitas. Menggunakan logika yang sama, asumsikan bahwa Anda menentukan bobot 1 untuk CapacityProvidera dan bobot 4 untuk CapacityProviderB. Kemudian, untuk setiap satu tugas yang dijalankan menggunakan CapacityProvidera, empat tugas menggunakan CapacityProviderB.

`base`

Jenis: Integer

Kisaran valid: Bilangan bulat antara 0 dan 100.000.

Wajib: Tidak

Nilai dasar menunjuk berapa banyak tugas, minimal, yang akan berjalan di penyedia kapasitas yang ditentukan. Hanya satu penyedia kapasitas dalam strategi penyedia kapasitas dapat memiliki basis yang ditentukan.

Ketentuan tugas

`taskDefinition`

Tipe: String

Wajib: Tidak

Nama Sumber Daya Amazon `revision` (ARN`family:revision`) `family` dan () atau lengkap dari definisi tugas yang akan dijalankan di layanan Anda. Jika a `revision` tidak ditentukan, `ACTIVE` revisi terbaru dari keluarga yang ditentukan akan digunakan.

Ketentuan tugas harus ditentukan ketika menggunakan pembaruan bergulir (`ECS`) pengendali deployment.

Sistem operasi platform

`platformFamily`

Tipe: string

Wajib: Bersyarat

Default: Linux

Parameter ini diperlukan untuk layanan Amazon ECS yang dihosting di Fargate.

Parameter ini diabaikan untuk layanan Amazon ECS yang dihosting di Amazon EC2.

Sistem operasi pada kontainer yang menjalankan layanan. Nilai yang valid adalah`LINUX`,`WINDOWS_SERVER_2019_FULL`,`WINDOWS_SERVER_2019_CORE`,`WINDOWS_SERVER_202` dan`WINDOWS_SERVER_2022_CORE`.

`platformFamily`Nilai untuk setiap tugas yang Anda tentukan untuk layanan harus sesuai dengan `platformFamily` nilai layanan. Misalnya, jika Anda mengatur `platformFamily` ke`WINDOWS_SERVER_2019_FULL`, `platformFamily` nilai untuk semua tugas harus`WINDOWS_SERVER_2019_FULL`.

Versi platform

`platformVersion`

Tipe: String

Wajib: Tidak

Versi platform tempat tugas Anda dalam layanan berjalan. Versi platform hanya ditentukan untuk tugas yang menggunakan tipe peluncuran Fargate. Jika tidak ditentukan, versi terbaru (`LATEST`) digunakan secara default.

AWS Versi platform Fargate digunakan untuk merujuk ke lingkungan runtime tertentu untuk infrastruktur tugas Fargate. Saat menentukan versi `LATEST` platform saat menjalankan tugas atau membuat layanan, Anda mendapatkan versi platform terbaru yang tersedia untuk tugas Anda. Saat Anda meningkatkan layanan Anda, tugas-tugas tersebut menerima versi platform yang ditentukan pada penerapan layanan saat ini. Untuk informasi selengkapnya, lihat Versi platform Fargate Linux.

> **Note**
>
> Versi platform tidak ditentukan untuk tugas yang menggunakan tipe peluncuran EC2.

Klaster

`cluster`

Tipe: String

Wajib: Tidak

Nama pendek atau Amazon Resource Name (ARN) lengkap dari klaster untuk menjalankan layanan Anda. Jika Anda tidak menentukan cluster, `default` cluster diasumsikan.

Nama layanan

`serviceName`

Tipe: String

Diperlukan: Ya

Nama layanan Anda. Mengizinkan hingga 255 huruf (huruf besar dan huruf kecil), angka, tanda hubung, dan garis bawah. Nama layanan harus unik dalam sebuah klaster, tetapi Anda dapat memiliki layanan yang bernama sama di beberapa klaster dalam satu Wilayah atau lebih.

Strategi penjadwalan

`schedulingStrategy`

Tipe: String

Nilai yang valid: `REPLICA` | `DAEMON`

Wajib: Tidak

Strategi penjadwalan yang boleh digunakan. Jika strategi penjadwalan tidak ditentukan, maka strategi `REPLICA` digunakan. Untuk informasi selengkapnya, lihat Konsep penjadwal layanan.

Ada dua strategi penjadwal layanan yang tersedia:

- `REPLICA`Strategi penjadwalan replika menempatkan dan mempertahankan jumlah tugas yang diinginkan di seluruh klaster Anda. Secara default tugas tersebar di seluruh Availability Zone. Anda dapat menggunakan strategi penempatan tugas dan kendala untuk menyesuaikan keputusan penempatan tugas. Untuk informasi selengkapnya, lihat Replika.
- `DAEMON`—Strategi penjadwalan daemon menerapkan tepat satu tugas pada setiap instance kontainer aktif yang memenuhi semua batasan penempatan tugas yang Anda tentukan di cluster Anda. Saat menggunakan strategi ini, tidak perlu menentukan jumlah tugas yang diinginkan, strategi penempatan tugas, atau menggunakan kebijakan Auto Scaling Layanan. Untuk informasi selengkapnya, lihat Daemon.

> **Note**
>
> Tugas Fargate tidak mendukung strategi `DAEMON` penjadwalan.

Jumlah yang diinginkan

`desiredCount`

Tipe: Integer

Wajib: Tidak

Jumlah instantiasi definisi tugas yang ditentukan untuk ditempatkan dan terus berjalan di layanan Anda.

Parameter ini diperlukan jika strategi penjadwalan `REPLICA` digunakan. Jika layanan menggunakan strategi penjadwalan `DAEMON`, parameter ini bersifat opsional.

Konfigurasi deployment

`deploymentConfiguration`

Tipe: Objek

Wajib: Tidak

Parameter deployment opsional yang mengendalikan berapa banyak tugas yang berjalan selama deployment dan pengurutan tugas berhenti dan memulai.

`maximumPercent`

Tipe: Integer

Wajib: Tidak

Jika layanan menggunakan tipe penerapan rolling update (`ECS`), `maximumPercent` parameter tersebut mewakili batas atas jumlah tugas layanan Anda yang diizinkan dalam `RUNNING`STOPPING, atau `PENDING` status selama penerapan. Hal ini dinyatakan sebagai persentase dari `desiredCount` yang dibulatkan ke bawah ke bilangan bulat terdekat. Anda dapat menggunakan parameter ini untuk menentukan ukuran batch deployment. Misalnya, jika layanan Anda menggunakan penjadwal `REPLICA` layanan dan memiliki `desiredCount` empat tugas dan `maximumPercent` nilai 200%, penjadwal mungkin memulai empat tugas baru sebelum menghentikan empat tugas lama. Ini asalkan sumber daya cluster yang diperlukan untuk melakukan ini tersedia. Nilai `maximumPercent` default untuk layanan yang menggunakan penjadwal layanan `REPLICA` adalah 200%.

Jika layanan Anda menggunakan jenis penjadwal DAEMON layanan, maximumPercent harus tetap pada 100%. Ini adalah nilai default.

Jumlah maksimum tugas selama deployment adalah desiredCount dikalikan dengan maximumPercent/100, dibulatkan ke bawah ke nilai integer terdekat.

Jika layanan menggunakan jenis dan tugas biru/hijau (CODE_DEPLOY) atau EXTERNAL penerapan yang menggunakan tipe peluncuran EC2, nilai persen maksimum disetel ke nilai default dan digunakan untuk menentukan batas atas jumlah tugas dalam layanan yang tetap dalam RUNNING status saat instance kontainer berada dalam status. DRAINING Jika tugas dalam layanan menggunakan tipe peluncuran Fargate, nilai persen maksimum tidak digunakan, meskipun dikembalikan saat mendeskripsikan layanan Anda.

minimumHealthyPercent

Tipe: Integer

Wajib: Tidak

Jika layanan menggunakan tipe penerapan rolling update (ECS), layanan tersebut minimumHealthyPercent mewakili batas bawah pada jumlah tugas layanan Anda yang harus tetap berada dalam RUNNING status selama penerapan. Ini dinyatakan sebagai persentase dari desiredCount yang dibulatkan ke bilangan bulat terdekat. Anda dapat menggunakan parameter ini untuk menyebarkan tanpa menggunakan kapasitas cluster tambahan. Misalnya, jika layanan Anda memiliki desiredCount empat tugas dan 50%, penjadwal layanan mungkin menghentikan dua tugas yang ada untuk membebaskan kapasitas klaster sebelum memulai dua tugas baru. minimumHealthyPercent

Untuk layanan yang tidak menggunakan penyeimbang beban, pertimbangkan hal berikut:

- Sebuah layanan dianggap baik jika semua kontainer penting dalam tugas dalam layanan lulus pemeriksaan kondisi.
- Jika tugas tidak memiliki wadah penting dengan pemeriksaan kesehatan yang ditentukan, penjadwal layanan menunggu selama 40 detik setelah tugas mencapai RUNNING keadaan sebelum tugas dihitung terhadap total persen sehat minimum.
- Jika suatu tugas memiliki satu atau lebih wadah penting dengan pemeriksaan kesehatan yang ditentukan, penjadwal layanan menunggu tugas mencapai status sehat sebelum menghitungnya ke total persen sehat minimum. Sebuah tugas dianggap baik ketika semua kontainer penting dalam tugas telah lulus pemeriksaan kondisi. Jumlah waktu tunggu

penjadwal layanan ditentukan oleh pengaturan pemeriksaan kondisi kontainer. Untuk informasi selengkapnya, lihat Pemeriksaan kondisi.

Untuk layanan yang menggunakan load balancer, perhatikan hal berikut:

- Jika suatu tugas tidak memiliki wadah penting dengan pemeriksaan kesehatan yang ditentukan, penjadwal layanan menunggu pemeriksaan kesehatan kelompok sasaran penyeimbang beban untuk mengembalikan status sehat sebelum menghitung tugas menuju total persen sehat minimum.
- Jika tugas memiliki wadah penting dengan pemeriksaan kesehatan yang ditentukan, penjadwal layanan menunggu tugas untuk mencapai status sehat dan pemeriksaan kesehatan kelompok target penyeimbang beban untuk mengembalikan status sehat sebelum menghitung tugas menuju total persen sehat minimum.

Nilai default untuk layanan replika untuk `minimumHealthyPercent` adalah 100%. `minimumHealthyPercent`Nilai default untuk layanan yang menggunakan jadwal `DAEMON` layanan adalah 0% untuk AWS CLI, AWS SDK, dan API dan 50% untuk. AWS Management Console

Jumlah minimal tugas sehat selama deployment adalah `desiredCount` dikalikan dengan `minimumHealthyPercent`/100, dibulatkan ke atas ke nilai bilangan bulat terdekat.

Jika layanan menggunakan tipe biru/hijau (`CODE_DEPLOY`) atau `EXTERNAL` penerapan dan menjalankan tugas yang menggunakan tipe peluncuran EC2, nilai persen sehat minimum disetel ke nilai default dan digunakan untuk menentukan batas bawah jumlah tugas dalam layanan yang tetap dalam `RUNNING` status saat instance container berada dalam status. `DRAINING` Jika layanan menggunakan tipe biru/hijau (`CODE_DEPLOY`) atau `EXTERNAL` penerapan dan menjalankan tugas yang menggunakan tipe peluncuran Fargate, nilai persen sehat minimum tidak digunakan, meskipun layanan tersebut dikembalikan saat mendeskripsikan layanan Anda.

Pengendali deployment

`deploymentController`

Tipe: Objek

Wajib: Tidak

Pengendali deployment yang bisa digunakan untuk layanan. Jika pengendali deployment tidak ditentukan, maka digunakan pengendali ECS. Untuk informasi selengkapnya, lihat Jenis Penerapan Amazon ECS.

`type`

Tipe: String

Nilai yang valid: `ECS` | `CODE_DEPLOY` | `EXTERNAL`

Wajib: ya

Tipe pengendali deployment yang bisa digunakan. Tersedia tiga tipe pengendali deployment:

`ECS`

Tipe deployment (`ECS`) pembaruan bergulir mencakup penggantian versi yang berjalan sekarang dari kontainer dengan versi terbaru. Jumlah kontainer Amazon ECS yang ditambahkan atau dihapus dari layanan selama pembaruan bergulir dikendalikan dengan menyesuaikan jumlah minimum dan maksimum tugas sehat yang diizinkan selama penerapan layanan, seperti yang ditentukan dalam. deploymentConfiguration

`CODE_DEPLOY`

Tipe penyebaran biru/hijau (`CODE_DEPLOY`) menggunakan model penerapan biru/hijau yang didukung oleh CodeDeploy, yang memungkinkan Anda memverifikasi penerapan layanan baru sebelum mengirim lalu lintas produksi ke sana.

`EXTERNAL`

Gunakan jenis penerapan eksternal saat Anda ingin menggunakan pengontrol penerapan pihak ketiga untuk kontrol penuh atas proses penyebaran untuk layanan Amazon ECS.

Penempatan tugas

`placementConstraints`

Tipe: Array objek

Wajib: Tidak

Aray objek batasan penempatan yang digunakan untuk tugas dalam layanan Anda. Anda dapat menentukan maksimum 10 kendala per tugas. Batas ini mencakup kendala dalam definisi tugas

dan yang ditentukan pada waktu berjalan. Jika Anda menggunakan tipe peluncuran Fargate, batasan penempatan tugas tidak didukung.

`type`

Tipe: String

Wajib: Tidak

Tipe batasan. Gunakan `distinctInstance` untuk memastikan bahwa setiap tugas dalam grup tertentu berjalan di instans kontainer yang berbeda. Gunakan `memberOf` untuk membatasi seleksi ke grup kandidat yang valid. Nilai `distinctInstance` tidak didukung dalam ketentuan tugas.

`expression`

Tipe: String

Wajib: Tidak

Ekspresi bahasa kueri klaster untuk diterapkan pada batasan. Ingat bahwa Anda tidak dapat menentukan ekspresi jika tipe kendalanya adalah `distinctInstance`. Untuk informasi selengkapnya, lihat Buat ekspresi untuk menentukan instance kontainer untuk tugas Amazon ECS.

`placementStrategy`

Tipe: Array objek

Wajib: Tidak

Objek strategi penempatan untuk digunakan untuk tugas di layanan Anda. Anda dapat menentukan maksimal empat aturan strategi per layanan.

`type`

Tipe: String

Nilai yang valid: `random` | `spread` | `binpack`

Wajib: Tidak

Tipe strategi penempatan. Strategi penempatan `random` menempatkan tugas pada kandidat yang tersedia secara acak. Strategi penempatan `spread` menyebarkan penempatan di seluruh kandidat yang tersedia secara merata berdasarkan parameter `field`. `binpack`Strategi menempatkan tugas pada kandidat yang tersedia yang memiliki jumlah

sumber daya paling sedikit yang tersedia yang ditentukan dengan `field` parameter. Misalnya, jika Anda binpack pada memori, tugas ditempatkan pada instance dengan jumlah memori yang tersisa paling sedikit tetapi masih cukup untuk menjalankan tugas.

`field`

Tipe: String

Wajib: Tidak

Bidang tempat menerapkan strategi penempatan. Untuk strategi `spread` penempatan, nilai yang valid adalah `instanceId` (atau`host`, yang memiliki efek yang sama), atau platform atau atribut khusus apa pun yang diterapkan ke instance container, seperti`attribute:ecs.availability-zone`. Untuk strategi penempatan `binpack`, nilai yang valid adalah `cpu` dan `memory`. Untuk strategi penempatan `random`, bidang ini tidak digunakan.

Tag

`tags`

Tipe: Array objek

Diperlukan: Tidak

Metadata yang Anda terapkan ke layanan untuk membantu Anda mengkategorikan dan mengaturnya. Setiap tanda terdiri dari kunci dan nilai opsional, yang keduanya Anda tentukan. Ketika layanan dihapus, tag akan dihapus juga. Maksimal 50 tag dapat diterapkan ke layanan. Untuk informasi selengkapnya, lihat Penandaan sumber daya Amazon ECS.

`key`

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Diperlukan: Tidak

Satu bagian dari pasangan nilai kunci yang membentuk tanda. Kunci adalah label umum yang bertindak seperti kategori untuk nilai tanda yang lebih spesifik.

`value`

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 256.

Diperlukan: Tidak

Bagian opsional pasangan nilai kunci yang membentuk tanda. Nilai bertindak sebagai deskriptor dalam kategori tanda (kunci).

`enableECSManagedTags`

Jenis: Boolean

Nilai yang valid: `true` | `false`

Diperlukan: Tidak

Menentukan apakah akan menggunakan tag terkelola Amazon ECS untuk tugas dalam layanan. Jika tidak ada nilai yang ditentukan, nilai defaultnya adalah `false`. Untuk informasi selengkapnya, lihat Menandai sumber daya Anda untuk penagihan.

`propagateTags`

Tipe: String

Nilai yang valid: `TASK_DEFINITION` | `SERVICE`

Diperlukan: Tidak

Menentukan apakah akan menyalin tag dari definisi tugas atau layanan untuk tugas-tugas dalam layanan. Jika tidak ada nilai yang ditentukan, tag tidak disalin. Tag hanya dapat disalin ke tugas dalam layanan selama pembuatan layanan. Untuk menambahkan tag ke tugas setelah pembuatan layanan atau pembuatan tugas, gunakan tindakan `TagResource` API.

Konfigurasi jaringan

`networkConfiguration`

Tipe: Objek

Wajib: Tidak

Konfigurasi jaringan untuk layanan. Parameter ini diperlukan untuk definisi tugas yang menggunakan mode `awsvpc` jaringan untuk menerima Antarmuka Jaringan Elastis mereka sendiri, dan tidak didukung untuk mode jaringan lainnya. Jika menggunakan tipe peluncuran

Fargate, mode `awsvpc` jaringan diperlukan. Untuk informasi selengkapnya tentang jaringan untuk jenis peluncuran Amazon EC2, lihat. Jaringan tugas untuk tugas di instans Amazon EC2 Untuk informasi selengkapnya tentang jaringan untuk jenis peluncuran Fargate, lihat Jaringan Tugas Fargate.

`awsvpcConfiguration`

Tipe: Objek

Wajib: Tidak

Sebuah objek yang menunjukkan subnet dan grup keamanan untuk tugas atau layanan.

`subnets`

Tipe: Array string

Diperlukan: Ya

Subnet yang terkait dengan tugas atau layanan. Ada batas 16 subnet yang dapat ditentukan sesuai dengan`awsvpcConfiguration`.

`securityGroups`

Tipe: Array string

Wajib: Tidak

Grup keamanan yang terkait dengan tugas atau layanan. Jika Anda tidak menentukan grup keamanan, grup keamanan default untuk VPC akan digunakan. Ada batas lima grup keamanan yang dapat ditentukan berdasarkan`awsvpcConfiguration`.

`assignPublicIP`

Tipe: String

Nilai yang valid: `ENABLED` | `DISABLED`

Wajib: Tidak

Apakah antarmuka jaringan elastis tugas menerima alamat IP publik. Jika nilai tidak ditentukan, maka digunakan nilai default `DISABLED`.

`healthCheckGracePeriodSeconds`

Tipe: Integer

Wajib: Tidak

Periode waktu, dalam hitungan detik, bahwa penjadwal layanan Amazon ECS harus mengabaikan pemeriksaan kesehatan target Elastic Load Balancing yang tidak sehat, pemeriksaan kesehatan kontainer, dan pemeriksaan kesehatan Route 53 setelah tugas memasuki suatu keadaan. `RUNNING` Ini hanya berlaku jika layanan Anda dikonfigurasi untuk menggunakan penyeimbang beban. Jika layanan Anda memiliki penyeimbang beban yang ditentukan dan Anda tidak menentukan nilai tenggang pemeriksaan kesehatan, nilai default akan `0` digunakan.

Jika tugas layanan Anda membutuhkan waktu cukup lama untuk memulai dan menanggapi pemeriksaan kesehatan, Anda dapat menentukan masa tenggang pemeriksaan kesehatan hingga 2.147.483.647 detik selama penjadwal layanan ECS mengabaikan status pemeriksaan kesehatan. Masa tenggang ini dapat mencegah penjadwal layanan ECS menandai tugas sebagai tidak sehat dan menghentikannya sebelum mereka punya waktu untuk muncul.

Jika Anda tidak menggunakan Elastic Load Balancing, kami sarankan Anda menggunakan `startPeriod` parameter pemeriksaan kesehatan definisi tugas. Untuk informasi lebih lanjut, lihat Pemeriksaan Kesehatan.

`loadBalancers`

Tipe: Array objek

Wajib: Tidak

Sebuah objek penyeimbang beban menunjukkan penyeimbang beban untuk digunakan dengan layanan Anda. Untuk layanan yang menggunakan Application Load Balancer atau Network Load Balancer, ada batasan lima grup target yang dapat Anda lampirkan ke layanan.

Setelah Anda membuat layanan, konfigurasi penyeimbang beban tidak dapat diubah dari. AWS Management Console Anda dapat menggunakan AWS Copilot, AWS CloudFormation, AWS CLI atau SDK untuk memodifikasi konfigurasi penyeimbang beban hanya untuk pengontrol penerapan `ECS` bergulir, bukan biru/hijau atau eksternal. AWS CodeDeploy Saat Anda menambahkan, memperbarui, atau menghapus konfigurasi penyeimbang beban, Amazon ECS memulai penerapan baru dengan konfigurasi Elastic Load Balancing yang diperbarui. Hal ini menyebabkan tugas mendaftar dan membatalkan pendaftaran dari penyeimbang beban. Kami menyarankan Anda memverifikasi ini di lingkungan pengujian sebelum memperbarui konfigurasi Elastic Load Balancing. Untuk informasi tentang cara mengubah konfigurasi, lihat UpdateServicedi Referensi API Amazon Elastic Container Service.

Untuk Application Load Balancers dan Network Load Balancers, objek ini harus berisi kelompok target load balancer ARN, nama kontainer (seperti yang muncul dalam definisi kontainer), dan port kontainer untuk mengakses dari penyeimbang beban. Ketika tugas dari layanan ini ditempatkan pada instance kontainer, instance kontainer dan kombinasi port terdaftar sebagai target dalam kelompok target yang ditentukan.

`targetGroupArn`

Tipe: String

Wajib: Tidak

Nama Sumber Daya Amazon (ARN) lengkap dari grup target Elastic Load Balancing yang terkait dengan layanan.

ARN grup target hanya ditentukan saat menggunakan Application Load Balancer atau Penyeimbang Beban Jaringan.

`loadBalancerName`

Tipe: String

Wajib: Tidak

Nama penyeimbang beban untuk dikaitkan dengan layanan.

Jika Anda menggunakan Application Load Balancer atau Network Load Balancer, hilangkan parameter nama load balancer.

`containerName`

Tipe: String

Wajib: Tidak

Nama kontainer (seperti yang muncul dalam ketentuan kontainer) untuk dikaitkan dengan penyeimbang beban.

`containerPort`

Tipe: Bilangan bulat

Wajib: Tidak

Port pada kontainer untuk dikaitkan dengan penyeimbang beban. Port ini harus sesuai dengan `containerPort` dalam ketentuan tugas yang digunakan oleh tugas dalam layanan. Untuk

tugas yang menggunakan tipe peluncuran EC2, instance container harus mengizinkan lalu lintas masuk pada `hostPort` pemetaan port.

`role`

Tipe: String

Wajib: Tidak

Nama pendek atau ARN lengkap dari peran IAM yang memungkinkan Amazon ECS melakukan panggilan ke penyeimbang beban Anda atas nama Anda. Parameter ini hanya diizinkan jika Anda menggunakan penyeimbang beban dengan satu grup target untuk layanan Anda, dan definisi tugas Anda tidak menggunakan mode `awsvpc` jaringan. Jika Anda menentukan `role` parameter, Anda juga harus menentukan objek penyeimbang beban dengan `loadBalancers` parameter.

Jika peran yang Anda tentukan memiliki jalur selain `/`, maka Anda harus menentukan ARN peran penuh (dianjurkan) atau awalan nama peran dengan jalur. Misalnya, jika peran dengan nama `bar` memiliki jalur dari `/foo/` maka Anda akan menentukan `/foo/bar` sebagai nama peran. Untuk informasi selengkapnya, lihat Nama dan Jalur Ramah di Panduan Pengguna IAM.

> **Important**
>
> Jika akun Anda telah membuat peran terkait layanan Amazon ECS, peran tersebut digunakan secara default untuk layanan Anda kecuali Anda menentukan peran di sini. Peran terkait layanan diperlukan jika definisi tugas Anda menggunakan mode jaringan awsvpc, dalam hal ini Anda tidak boleh menentukan peran di sini. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS.

`serviceConnectConfiguration`

Tipe: Objek

Diperlukan: Tidak

Konfigurasi untuk layanan ini untuk menemukan dan terhubung ke layanan, dan ditemukan oleh, dan terhubung dari, layanan lain dalam namespace.

Untuk informasi selengkapnya, lihat Layanan Connect.

`enabled`

Jenis: Boolean

Diperlukan: Ya

Menentukan apakah akan menggunakan Service Connect dengan layanan ini.

`namespace`

Tipe: String

Wajib: Tidak

Nama pendek atau nama sumber daya Amazon lengkap (ARN) dari AWS Cloud Map namespace untuk digunakan dengan Service Connect. Namespace harus Wilayah AWS sama dengan layanan Amazon ECS dan cluster. Jenis namespace tidak memengaruhi Service Connect. Untuk informasi selengkapnya AWS Cloud Map, lihat Bekerja dengan Layanan di Panduan AWS Cloud Map Pengembang.

`services`

Tipe: Array objek

Diperlukan: Tidak

Array objek layanan Service Connect. Ini adalah nama dan alias (juga dikenal sebagai titik akhir) yang digunakan oleh layanan Amazon ECS lainnya untuk terhubung ke layanan ini.

Bidang ini tidak diperlukan untuk layanan Amazon ECS “klien” yang merupakan anggota namespace hanya untuk terhubung ke layanan lain di dalam namespace. Contohnya adalah aplikasi frontend yang menerima permintaan masuk dari penyeimbang beban yang dilampirkan ke layanan atau dengan cara lain.

Objek memilih port dari definisi tugas, menetapkan nama untuk AWS Cloud Map layanan, dan array alias (juga dikenal sebagai titik akhir) dan port untuk aplikasi klien untuk merujuk ke layanan ini.

`portName`

Tipe: String

Diperlukan: Ya

`portName`Harus cocok dengan `name` salah satu `portMappings` dari semua wadah dalam definisi tugas layanan Amazon ECS ini.

`discoveryName`

Tipe: String

Wajib: Tidak

`discoveryName`Itu adalah nama AWS Cloud Map layanan baru yang dibuat Amazon ECS untuk layanan Amazon ECS ini. Ini harus unik di dalam AWS Cloud Map namespace.

Jika bidang ini tidak ditentukan, `portName` digunakan.

`clientAliases`

Tipe: Array objek

Diperlukan: Tidak

Daftar alias klien untuk layanan koneksi layanan ini. Anda menggunakan ini untuk menetapkan nama yang dapat digunakan oleh aplikasi klien. Jumlah maksimum alias klien yang dapat Anda miliki dalam daftar ini adalah 1.

Setiap alias (“endpoint”) adalah nama DNS dan nomor port yang dapat digunakan layanan Amazon ECS lainnya (“klien”) untuk terhubung ke layanan ini.

Setiap kombinasi nama dan port harus unik di dalam namespace.

Nama-nama ini dikonfigurasi dalam setiap tugas layanan klien, bukan di AWS Cloud Map. Permintaan DNS untuk menyelesaikan nama-nama ini tidak meninggalkan tugas, dan tidak dihitung terhadap kuota permintaan DNS per detik per elastic network interface.

`port`

Jenis: Integer

Diperlukan: Ya

Nomor port mendengarkan untuk proxy koneksi layanan. Port ini tersedia di dalam semua tugas dalam namespace yang sama.

Untuk menghindari perubahan aplikasi Anda di layanan Amazon ECS klien, atur ini ke port yang sama yang digunakan aplikasi klien secara default.

`dnsName`

Tipe: String

Wajib: Tidak

Itu `dnsName` adalah nama yang Anda gunakan dalam aplikasi tugas klien untuk terhubung ke layanan ini. Nama harus berupa label DNS yang valid.

Nilai default adalah `discoveryName.namespace` jika bidang ini tidak ditentukan. Jika `discoveryName` tidak ditentukan, definisi `portName` dari tugas digunakan.

Untuk menghindari perubahan aplikasi Anda di layanan Amazon ECS klien, atur ini ke nama yang sama dengan yang digunakan aplikasi klien secara default. Misalnya, beberapa nama umum adalah`database`,db, atau nama huruf kecil dari database, seperti `mysql` atau. `redis`

`ingressPortOverride`

Tipe: Integer

Wajib: Tidak

(Opsional) Nomor port untuk proxy Service Connect untuk didengarkan.

Gunakan nilai bidang ini untuk melewati proxy untuk lalu lintas pada nomor port yang ditentukan dalam nama `portMapping` dalam definisi tugas aplikasi ini, lalu gunakan di grup keamanan Amazon VPC Anda untuk mengizinkan lalu lintas ke proxy untuk layanan Amazon ECS ini.

Dalam `awsvpc` mode, nilai default adalah nomor port kontainer yang ditentukan dalam nama `portMapping` dalam definisi tugas aplikasi ini. Dalam `bridge` mode, nilai default adalah port ephemeral dinamis dari proxy Service Connect.

`logConfiguration`

Jenis: LogConfigurationObjek

Diperlukan: Tidak

Ini menentukan di mana log proxy Service Connect dipublikasikan. Gunakan log untuk debugging selama kejadian tak terduga. Konfigurasi ini menetapkan `logConfiguration` parameter dalam wadah proxy Service Connect di setiap tugas di layanan Amazon ECS ini. Kontainer proxy tidak ditentukan dalam definisi tugas.

Kami menyarankan Anda menggunakan konfigurasi log yang sama dengan wadah aplikasi definisi tugas untuk layanan Amazon ECS ini. Untuk FireLens, ini adalah konfigurasi log dari wadah aplikasi. Bukan wadah router FireLens log yang menggunakan gambar `fluent-bit` atau `fluentd` kontainer.

`serviceRegistries`

Tipe: Array objek

Wajib: Tidak

Detail konfigurasi pencari layanan untuk layanan Anda. Untuk informasi selengkapnya, lihat Penemuan Layanan .

`registryArn`

Tipe: String

Wajib: Tidak

Nama Sumber Daya Amazon (ARN) dari registri layanan. Registri layanan yang saat ini didukung adalah AWS Cloud Map. Untuk informasi selengkapnya, lihat Bekerja dengan Layanan di Panduan AWS Cloud Map Pengembang.

`port`

Tipe: Integer

Wajib: Tidak

Nilai port yang digunakan jika layanan penemuan layanan Anda menentukan catatan SRV. Kolom ini wajib diisi jika mode jaringan `awsvpc` dan catatan SRV digunakan.

`containerName`

Tipe: String

Wajib: Tidak

Nilai nama kontainer yang akan digunakan untuk layanan penemuan layanan Anda. Nilai ini ditentukan dalam definisi tugas. Jika ketentuan tugas yang menentukan tugas layanan Anda menggunakan mode jaringan `bridge` atau `host`, Anda harus menentukan kombinasi `containerName` dan `containerPort` dari ketentuan tugas. Jika ketentuan tugas yang ditentukan oleh tugas layanan Anda menggunakan mode jaringan `awsvpc` dan tipe catatan SRV DNS digunakan, Anda harus menentukan apakah akan menggunakan salah satu kombinasi `containerName` dan `containerPort` atau nilai `port`, tetapi tidak keduanya.

`containerPort`

Tipe: Integer

Wajib: Tidak

Nilai port yang akan digunakan untuk layanan penemuan layanan Anda. Nilai ini ditentukan dalam definisi tugas. Jika ketentuan tugas yang ditentukan oleh tugas layanan Anda

menggunakan mode jaringan `bridge` atau `host`, Anda harus menentukan kombinasi `containerName` dan `containerPort` dari ketentuan tugas. Jika ketentuan tugas yang ditentukan oleh tugas layanan Anda menggunakan mode jaringan `awsvpc` dan tipe catatan SRV DNS digunakan, Anda harus menentukan apakah akan menggunakan salah satu kombinasi `containerName` dan `containerPort` atau nilai `port`, tetapi tidak keduanya.

Token klien

`clientToken`

Tipe: String

Wajib: Tidak

Pengidentifikasi unik dan peka huruf besar/kecil yang Anda berikan untuk memastikan idempotensi permintaan. Panjangnya bisa mencapai 32 karakter ASCII.

Konfigurasi volume

`volumeConfigurations`

Tipe: Objek

Diperlukan: Tidak

Konfigurasi yang akan digunakan untuk membuat volume untuk tugas-tugas yang dikelola oleh layanan. Satu volume dibuat untuk setiap tugas dalam layanan. Hanya volume yang ditandai seperti `configuredAtLaunch` dalam definisi tugas yang dapat dikonfigurasi dengan menggunakan objek ini. Objek ini diperlukan untuk melampirkan volume data Amazon EBS ke tugas yang dikelola oleh layanan. Untuk informasi selengkapnya, lihat volume Amazon EBS.

`name`

Tipe: String

Diperlukan: Ya

Nama volume yang dikonfigurasi saat membuat atau memperbarui layanan. Hingga 255 huruf (huruf besar dan kecil), angka, garis bawah (), dan tanda hubung (_) diperbolehkan. - Nilai ini harus cocok dengan nama volume yang ditentukan dalam definisi tugas.

`managedEBSVolume`

Tipe: Objek

Diperlukan: Tidak

Konfigurasi volume untuk volume Amazon EBS yang dilampirkan ke tugas yang dikelola oleh layanan saat layanan dibuat atau diperbarui.

`encrypted`

Tipe: Boolean

Wajib: Tidak

Nilai yang valid: `true` | `false`

Menentukan apakah volume Amazon EBS yang dilampirkan ke tugas yang dikelola oleh layanan akan dienkripsi. Jika Anda mengaktifkan enkripsi Amazon EBS secara default untuk akun Anda, pengaturan ini akan diganti, dan volume akan dienkripsi. Untuk informasi selengkapnya tentang enkripsi EBS secara default, lihat Enkripsi secara default di Panduan Pengguna Amazon EC2.

`kmsKeyId`

Tipe: String

Wajib: Tidak

Pengidentifikasi kunci AWS Key Management Service (AWS KMS) yang akan digunakan untuk enkripsi Amazon EBS. Jika parameter ini tidak ditentukan, Anda AWS KMS key untuk Amazon EBS digunakan. Jika `KmsKeyId` ditentukan, status terenkripsi harus `true`.

Anda dapat menentukan kunci KMS dengan menggunakan salah satu dari berikut ini:

- ID Kunci — Misalnya,`1234abcd-12ab-34cd-56ef-1234567890ab`.
- Alias kunci — Misalnya,`alias/ExampleAlias`.
- ARN Kunci — Misalnya,. `arn:aws:kms:us-east-1:012345678910:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Alias ARN — Misalnya,. `arn:aws:kms:us-east-1:012345678910:alias/ExampleAlias`

> **Important**
>
> AWS mengautentikasi kunci KMS secara asinkron. Oleh karena itu, jika Anda menentukan ID, alias, atau ARN yang tidak valid, tindakan dapat tampak berhasil, tetapi akhirnya gagal. Untuk informasi selengkapnya, lihat Memecahkan masalah lampiran volume Amazon EBS.

`volumeType`

Tipe: String

Wajib: Tidak

Nilai yang valid: `gp2` `gp3` | `io1` | `io2` | `sc1` | `st1` | `standard`

Jenis volume EBS. Untuk informasi selengkapnya tentang jenis volume, lihat jenis volume Amazon EBS di Panduan Pengguna Amazon EC2. Jenis volume default adalah `gp3`.

> **Note**
>
> Jenis `standard` volume tidak didukung untuk volume Amazon EBS yang dikonfigurasi untuk lampiran ke tugas Fargate.

`sizeInGiB`

Tipe: Integer

Wajib: Tidak

Rentang yang valid: Bilangan bulat antara 1 dan 16.384

Ukuran volume EBS dalam gibibytes (GiB). Jika Anda tidak memberikan ID snapshot untuk mengonfigurasi volume lampiran, Anda harus memberikan nilai ukuran. Jika Anda mengonfigurasi volume untuk lampiran menggunakan snapshot, nilai defaultnya adalah ukuran snapshot. Anda kemudian dapat menentukan ukuran yang lebih besar dari atau sama dengan ukuran snapshot.

Untuk `gp2` dan jenis `gp3` volume, kisaran yang valid adalah 1-16.384.

Untuk `io1` dan jenis `io2` volume, kisaran yang valid adalah 4-16.384.

Untuk `st1` dan jenis `sc1` volume, kisaran yang valid adalah 125-16.384.

Untuk jenis `standard` volume, kisaran yang valid adalah 1-1.024.

`snapshotId`

Tipe: String

Wajib: Tidak

ID snapshot volume EBS yang ada yang digunakan untuk membuat volume baru yang dilampirkan ke tugas ECS.

`iops`

Tipe: Integer

Wajib: Tidak

Jumlah operasi I/O per detik (IOPS). Untuk volume `gp3`, `io1`, dan `io2`, ini mewakili jumlah IOPS yang disediakan untuk volume. Untuk `gp2` volume, nilai ini mewakili kinerja dasar volume dan tingkat di mana volume mengakumulasi kredit I/O untuk meledak. Parameter ini diperlukan untuk volume `io1` dan `io2`. Parameter ini tidak didukung untuk`gp2`,`st1`,`sc1`, atau `standard` volume.

Untuk `gp3` volume, kisaran nilai yang valid adalah 3.000 hingga 16.000.

Untuk `io1` volume, kisaran nilai yang valid adalah 100 hingga 64.000.

Untuk `io2` volume, kisaran nilai yang valid adalah 100 hingga 64.000.

`throughput`

Tipe: Integer

Wajib: Tidak

Throughput untuk penyediaan volume yang melekat pada tugas-tugas yang dikelola oleh layanan.

> **Important**
>
> Parameter ini hanya didukung untuk `gp3` volume.

`roleArn`

Tipe: String

Diperlukan: Ya

Amazon Resource ARN (ARN) dari peran infrastruktur AWS Identity and Access Management (IAM) yang menyediakan izin Amazon ECS untuk mengelola sumber daya Amazon EBS untuk tugas Anda. Untuk informasi selengkapnya, lihat Peran IAM infrastruktur Amazon ECS.

`tagSpecifications`

Tipe: Objek

Diperlukan: Tidak

Spesifikasi tag yang akan diterapkan pada volume Amazon EBS yang dikelola serivce.

`resourceType`

Tipe: String

Diperlukan: Ya

Nilai yang valid: `volume`

Jenis sumber daya untuk menandai pada penciptaan.

`tags`

Tipe: Array objek

Diperlukan: Tidak

Metadata yang Anda terapkan pada volume untuk membantu Anda mengkategorikan dan mengaturnya. Setiap tag terdiri dari kunci dan nilai opsional, yang keduanya Anda tentukan. `AmazonECSCreated`dan `AmazonECSManaged` merupakan tag cadangan yang ditambahkan oleh Amazon ECS atas nama Anda, sehingga Anda dapat menentukan maksimum 48 tag Anda sendiri. Ketika volume dihapus, tag dihapus juga. Untuk informasi selengkapnya, lihat Penandaan sumber daya Amazon ECS.

`key`

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Diperlukan: Tidak

Salah satu bagian dari pasangan kunci-nilai yang membentuk tag. Kunci adalah label umum yang bertindak seperti kategori untuk nilai tanda yang lebih spesifik.

`value`

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 256.

Diperlukan: Tidak

Bagian opsional dari pasangan kunci-nilai yang membentuk tag. Nilai bertindak sebagai deskriptor dalam kategori tanda (kunci).

`propagateTags`

Tipe: String

Nilai yang valid: `TASK_DEFINITION` | `SERVICE` | `NONE`

Diperlukan: Tidak

Menentukan apakah akan menyalin tag dari definisi tugas atau layanan ke volume. Jika `NONE` ditentukan atau tidak ada nilai yang ditentukan, tag tidak disalin.

`fileSystemType`

Tipe: String

Wajib: Tidak

Nilai yang valid: `xfs` | `ext3` | `ext4`

Jenis sistem file pada volume. Jenis sistem file volume menentukan bagaimana data disimpan dan diambil dalam volume. Untuk volume yang dibuat dari snapshot, Anda harus menentukan jenis sistem file yang sama dengan volume yang digunakan saat snapshot dibuat. Jika ada ketidakcocokan tipe sistem file, tugas akan gagal dimulai. Default untuk volume yang dilampirkan ke tugas Linux adalah`XFS`.

Templat definisi layanan

Berikut ini menunjukkan representasi JSON dari definisi layanan Amazon ECS.

Jenis peluncuran Amazon EC2

```
{
    "cluster": "",
    "serviceName": "",
    "taskDefinition": "",
    "loadBalancers": [
        {
            "targetGroupArn": "",
            "loadBalancerName": "",
            "containerName": "",
            "containerPort": 0
        }
    ],
    "serviceRegistries": [
        {
            "registryArn": "",
            "port": 0,
            "containerName": "",
            "containerPort": 0
        }
    ],
    "desiredCount": 0,
    "clientToken": "",
    "launchType": "EC2",
    "capacityProviderStrategy": [
        {
            "capacityProvider": "",
            "weight": 0,
            "base": 0
        }
    ],
    "platformVersion": "",
    "role": "",
    "deploymentConfiguration": {
        "deploymentCircuitBreaker": {
            "enable": true,
            "rollback": true
        },
        "maximumPercent": 0,
        "minimumHealthyPercent": 0,
        "alarms": {
            "alarmNames": [
                ""
```

```
            ],
            "enable": true,
            "rollback": true
        }
    },
    "placementConstraints": [
        {
            "type": "distinctInstance",
            "expression": ""
        }
    ],
    "placementStrategy": [
        {
            "type": "binpack",
            "field": ""
        }
    ],
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": [
                ""
            ],
            "securityGroups": [
                ""
            ],
            "assignPublicIp": "DISABLED"
        }
    },
    "healthCheckGracePeriodSeconds": 0,
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "EXTERNAL"
    },
    "tags": [
        {
            "key": "",
            "value": ""
        }
    ],
    "enableECSManagedTags": true,
    "propagateTags": "TASK_DEFINITION",
    "enableExecuteCommand": true,
    "serviceConnectConfiguration": {
        "enabled": true,
```

```
        "namespace": "",
        "services": [
            {
                "portName": "",
                "discoveryName": "",
                "clientAliases": [
                    {
                        "port": 0,
                        "dnsName": ""
                    }
                ],
                "ingressPortOverride": 0
            }
        ],
        "logConfiguration": {
            "logDriver": "journald",
            "options": {
                "KeyName": ""
            },
            "secretOptions": [
                {
                    "name": "",
                    "valueFrom": ""
                }
            ]
        }
    },
    "volumeConfigurations": [
        {
            "name": "",
            "managedEBSVolume": {
                "encrypted": true,
                "kmsKeyId": "",
                "volumeType": "",
                "sizeInGiB": 0,
                "snapshotId": "",
                "iops": 0,
                "throughput": 0,
                "tagSpecifications": [
                    {
                        "resourceType": "volume",
                        "tags": [
                            {
                                "key": "",
```

```
                                "value": ""
                            }
                        ],
                        "propagateTags": "NONE"
                    }
                ],
                "roleArn": "",
                "filesystemType": ""
            }
        }
    ]
}
```

Jenis peluncuran Fargate

```
{
    "cluster": "",
    "serviceName": "",
    "taskDefinition": "",
    "loadBalancers": [
        {
            "targetGroupArn": "",
            "loadBalancerName": "",
            "containerName": "",
            "containerPort": 0
        }
    ],
    "serviceRegistries": [
        {
            "registryArn": "",
            "port": 0,
            "containerName": "",
            "containerPort": 0
        }
    ],
    "desiredCount": 0,
    "clientToken": "",
    "launchType": "FARGATE",
    "capacityProviderStrategy": [
        {
            "capacityProvider": "",
            "weight": 0,
            "base": 0
```

```
        }
    ],
    "platformVersion": "",
    "platformFamily": "",
    "role": "",
    "deploymentConfiguration": {
        "deploymentCircuitBreaker": {
            "enable": true,
            "rollback": true
        },
        "maximumPercent": 0,
        "minimumHealthyPercent": 0,
        "alarms": {
            "alarmNames": [
                ""
            ],
            "enable": true,
            "rollback": true
        }
    },
    "placementConstraints": [
        {
            "type": "distinctInstance",
            "expression": ""
        }
    ],
    "placementStrategy": [
        {
            "type": "binpack",
            "field": ""
        }
    ],
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": [
                ""
            ],
            "securityGroups": [
                ""
            ],
            "assignPublicIp": "DISABLED"
        }
    },
    "healthCheckGracePeriodSeconds": 0,
```

```
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "EXTERNAL"
    },
    "tags": [
        {
            "key": "",
            "value": ""
        }
    ],
    "enableECSManagedTags": true,
    "propagateTags": "TASK_DEFINITION",
    "enableExecuteCommand": true,
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "",
        "services": [
            {
                "portName": "",
                "discoveryName": "",
                "clientAliases": [
                    {
                        "port": 0,
                        "dnsName": ""
                    }
                ],
                "ingressPortOverride": 0
            }
        ],
        "logConfiguration": {
            "logDriver": "journald",
            "options": {
                "KeyName": ""
            },
            "secretOptions": [
                {
                    "name": "",
                    "valueFrom": ""
                }
            ]
        }
    },
    "volumeConfigurations": [
        {
```

```
            "name": "",
            "managedEBSVolume": {
                "encrypted": true,
                "kmsKeyId": "",
                "volumeType": "",
                "sizeInGiB": 0,
                "snapshotId": "",
                "iops": 0,
                "throughput": 0,
                "tagSpecifications": [
                    {
                        "resourceType": "volume",
                        "tags": [
                            {
                                "key": "",
                                "value": ""
                            }
                        ],
                        "propagateTags": "NONE"
                    }
                ],
                "roleArn": "",
                "filesystemType": ""
            }
        }
    ]
}
```

Anda dapat membuat templat ketentuan layanan ini menggunakan perintah AWS CLI berikut.

```
aws ecs create-service --generate-cli-skeleton
```

Kuota layanan Amazon ECS

Tabel berikut menyediakan kuota layanan default, juga disebut sebagai batas, untuk Amazon ECS untuk file. Akun AWSUntuk informasi selengkapnya tentang kuota layanan lainnya Layanan AWS yang dapat Anda gunakan dengan Amazon ECS, seperti Elastic Load Balancing dan Auto Scaling,AWS lihat kuota layanan di. Referensi Umum Amazon Web Services Untuk informasi tentang pembatasan API di Amazon ECS API, lihat Meminta pembatasan untuk Amazon ECS API.

Kuota layanan Amazon ECS

Berikut ini adalah Amazon ECS service quotas.

AWS Akun baru mungkin memiliki kuota awal yang lebih rendah yang dapat meningkat seiring waktu. Amazon ECS terus memantau penggunaan akun di setiap Wilayah, dan kemudian secara otomatis meningkatkan kuota berdasarkan penggunaan Anda. Anda juga dapat meminta peningkatan kuota untuk nilai yang ditampilkan sebagai dapat disesuaikan, lihat Meminta peningkatan kuota dalam Panduan Pengguna Service Quotas.

Nama	Default	Dapat disesuaikan	Deskripsi
Penyedia kapasitas per cluster	Setiap Wilayah yang didukung: 20	Tidak	Jumlah maksimum penyedia kapasitas yang dapat dikaitkan dengan cluster.
Classic Load Balancer per layanan	Setiap Wilayah yang didukung: 1	Tidak	Jumlah maksimum Classic Load Balancer per layanan.
Klaster per akun	Setiap Wilayah yang didukung: 10.000	Ya	Jumlah cluster per akun

Nama	Default	Dapat disesu an	Deskripsi
Instans kontainer per klaster	Setiap Wilayah yang didukung: 5.000	Tidak	Jumlah instance kontainer per cluster
Contoh kontainer per tugas awal	Setiap Wilayah yang didukung: 10	Tidak	Jumlah maksimum instance kontainer yang ditentukan dalam tindakan StartTask API.
Kontainer per definisi tugas	Setiap Wilayah yang didukung: 10	Tidak	Jumlah maksimum ketentuan kontainer dalam ketentuan tugas.
Sesi ECS Exec	Setiap Wilayah yang didukung: 1.000	Tidak	Jumlah maksimum sesi ECS Exec per kontainer.
Tingkat tugas yang diluncurkan oleh layanan di AWS Fargate	Setiap Wilayah yang didukung: 500	Tidak	Jumlah maksimum tugas yang dapat disediakan per layanan per menit di Fargate oleh penjadwal layanan Amazon ECS.
Nilai tugas yang diluncurkan oleh layanan di Amazon EC2 atau instans Eksternal	Setiap Wilayah yang didukung: 500	Tidak	Jumlah maksimum tugas yang dapat disediakan per layanan per menit di Amazon EC2 atau instans Eksternal oleh penjadwal layanan Amazon ECS.

Nama	Default	Dapat disesuaikan	Deskripsi
Revisi per keluarga ketentuan tugas	Setiap Wilayah yang didukung: 1.000.000	Tidak	Jumlah maksimum revisi per keluarga ketentuan tugas. Membatalkan pendaftaran atau menghapus revisi definisi tugas tidak mengecualikannya dari dimasukkan dalam batas ini.
Grup keamanan per AWSVPCConfiguration	Setiap Wilayah yang didukung: 5	Tidak	Jumlah maksimum grup keamanan yang ditentukan dalam AWSVPCConfiguration.
Layanan per klaster	Setiap Wilayah yang didukung: 5.000	Ya	Jumlah maksimum layanan per cluster
Layanan per namespace	Setiap Wilayah yang didukung: 100	Ya	Jumlah maksimum layanan yang dapat berjalan dalam namespace.
Subnet untuk AWSVPCConfiguration	Setiap Wilayah yang didukung: 16	Tidak	Jumlah maksimum subnet yang ditentukan dalam AWSVPCConfiguration.
Tanda per sumber daya	Setiap Wilayah yang didukung: 50	Tidak	Jumlah maksimum tag per sumber daya. Ini berlaku untuk definisi tugas, cluster, tugas, dan layanan.

Nama	Default	Dapat disesu an	Deskripsi
Grup target per layanan	Setiap Wilayah yang didukung: 5	Tidak	Jumlah maksimum grup target per layanan, jika menggunakan Applicati on Load Balancer atau Penyeimbang Beban Jaringan.
Ukuran definisi tugas	Setiap Wilayah yang didukung: 64 Kilobyte	Tidak	Ukuran maksimum ketentuan tugas dalam KiB.
Tugas dalam status PROVISIONING per cluster	Setiap Wilayah yang didukung: 500	Tidak	Jumlah maksimum tugas yang menunggu dalam status PROVISIONING per cluster. Kuota ini hanya berlaku untuk tugas yang diluncurkan menggunakan penyedia kapasitas grup EC2 Auto Scaling.
Tugas diluncurkan per run-task	Setiap Wilayah yang didukung: 10	Tidak	Jumlah maksimum tugas yang dapat diluncurkan per tindakan RunTask API.
Tugas per layanan	Setiap Wilayah yang didukung: 5.000	Tidak	Jumlah maksimum tugas per layanan (jumlah yang diinginkan).

> **Note**
>
> Nilai default adalah kuota awal yang ditetapkan oleh AWS, yang terpisah dari nilai kuota yang diterapkan aktual dan kuota layanan maksimum yang mungkin. Untuk informasi selengkapnya, lihat Terminologi dalam Service Quotas di Panduan Pengguna Service Quotas.

> **Note**
>
> Layanan yang dikonfigurasi untuk menggunakan penemuan layanan Amazon ECS memiliki batas 1.000 tugas per layanan. Hal ini disebabkan oleh AWS Cloud Map service quotas untuk jumlah instans per layanan. Untuk informasi selengkapnya, lihat AWS Cloud Map service quotas dalam Referensi Umum Amazon Web Services.

> **Note**
>
> Dalam praktiknya, tingkat peluncuran tugas juga tergantung pada pertimbangan lain seperti gambar kontainer yang akan diunduh dan dibongkar, pemeriksaan kesehatan dan integrasi lainnya diaktifkan, seperti mendaftarkan tugas dengan penyeimbang beban. Anda mungkin melihat variasi dalam tingkat peluncuran tugas dibandingkan dengan kuota yang diwakili di sini. Variasi ini disebabkan oleh fitur yang telah Anda aktifkan untuk layanan Amazon ECS Anda. Untuk informasi selengkapnya, lihat mempercepat penerapan Amazon ECS di Panduan Praktik Terbaik Amazon ECS.

> **Note**
>
> Layanan yang dikonfigurasi untuk menggunakan Amazon ECS Service Connect memiliki batas 1.000 tugas per layanan. Ini karena kuota AWS Cloud Map layanan untuk jumlah instance per layanan. Untuk informasi selengkapnya, lihat kuota layanan AWS Cloud Map di Referensi Umum Amazon Web Services.

AWS Fargate kuota layanan

Berikut ini adalah Amazon ECS pada kuota AWS Fargate layanan dan terdaftar di bawah layanan di konsol AWS FargateService Quotas.

AWS Akun baru mungkin memiliki kuota awal yang lebih rendah yang dapat meningkat seiring waktu. Fargate terus memantau penggunaan akun di setiap Wilayah, dan kemudian secara otomatis meningkatkan kuota berdasarkan penggunaan Anda. Anda juga dapat meminta peningkatan kuota untuk nilai yang ditampilkan sebagai dapat disesuaikan, lihat Meminta peningkatan kuota dalam Panduan Pengguna Service Quotas.

Nama	Default	Dapat disesuaikan	Deskripsi
Jumlah sumber daya vCPU Sesuai Permintaan Fargate	Setiap Wilayah yang didukung: 6	Ya	Jumlah vCPU Fargate yang berjalan bersamaan dengan Fargate On-Demand di akun ini di Wilayah saat ini.
Jumlah sumber daya vCPU Fargate Spot	Setiap Wilayah yang didukung: 6	Ya	Jumlah vCPU Fargate yang berjalan bersamaan sebagai Fargate Spot di akun ini di Wilayah saat ini.

> **Note**
>
> Nilai default adalah kuota awal yang ditetapkan oleh AWS, yang terpisah dari nilai kuota yang diterapkan aktual dan kuota layanan maksimum yang mungkin. Untuk informasi selengkapnya, lihat Terminologi dalam Service Quotas di Panduan Pengguna Service Quotas.

> **Note**
>
> Fargate juga memberlakukan tugas Amazon ECS dan batas kecepatan peluncuran pod Amazon EKS. Untuk informasi selengkapnya, lihat Batas pelambatan Fargate.

Mengelola Amazon ECS dan kuota AWS Fargate layanan Anda di AWS Management Console

Amazon ECS telah terintegrasi dengan Service Quotas, AWS sebuah layanan yang memungkinkan Anda untuk melihat dan mengelola kuota Anda dari lokasi pusat. Untuk informasi selengkapnya, lihat Apa itu Service Quotas? di Panduan Pengguna Service Quotas.

Service Quotas memudahkan untuk mencari nilai kuota layanan Amazon ECS Anda.

AWS Management Console

Untuk melihat kuota layanan Amazon ECS dan Fargate menggunakan AWS Management Console

1. Buka konsol Service Quotas di https://console.aws.amazon.com/servicequotas/.
2. Di panel navigasi, pilih Layanan AWS .
3. Dari daftar AWS layanan, cari dan pilih Amazon Elastic Container Service (Amazon ECS) atau. AWS Fargate

   Dalam daftar service quotas, Anda dapat melihat nama service quotas, nilai terapan (jika tersedia), kuota default AWS , dan apakah nilai kuota dapat disesuaikan.
4. Untuk melihat informasi tambahan tentang service quotas, seperti deskripsi, pilih nama kuota.
5. (Opsional) Untuk meminta peningkatan kuota, pilih kuota yang ingin Anda tingkatkan, kemudian pilih Meminta peningkatan kuota, masukkan atau pilih informasi yang diperlukan, dan pilih Minta.

Untuk bekerja lebih banyak dengan kuota layanan menggunakan AWS Management Console lihat Panduan Pengguna Service Quotas. Untuk meminta penambahan kuota, lihat Meminta penambahan kuota di Panduan Pengguna Service Quotas.

AWS CLI

Untuk melihat kuota layanan Amazon ECS dan Fargate menggunakan AWS CLI

Jalankan perintah berikut untuk melihat kuota Amazon ECS default.

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code ecs \
    --output table
```

Jalankan perintah berikut untuk melihat kuota Fargate default.

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code fargate \
    --output table
```

Jalankan perintah berikut untuk melihat kuota Fargate yang diterapkan.

```
aws service-quotas list-service-quotas \
    --service-code fargate
```

> **Note**
>
> Amazon ECS tidak mendukung kuota yang diterapkan.

Untuk informasi selengkapnya tentang bekerja dengan kuota layanan menggunakan AWS CLI, lihat Referensi Perintah Service AWS CLI Quotas. Untuk meminta peningkatan kuota, lihat perintah `request-service-quota-increase` dalam Referensi Perintah AWS CLI.

Praktik terbaik untuk kuota layanan Amazon ECS dan batas pembatasan API

Amazon ECS terintegrasi dengan beberapa Layanan AWS, termasuk Elastic Load Balancing AWS Cloud Map,, dan Amazon EC2. Dengan integrasi yang ketat ini, Amazon ECS mencakup

beberapa fitur seperti service load balancing, Service Connect, task networking, dan cluster auto scaling. Amazon ECS dan lainnya Layanan AWS yang terintegrasi dengan semua kuota layanan pemeliharaan dan batas tarif API untuk memastikan kinerja dan pemanfaatan yang konsisten. Kuota layanan ini juga mencegah penyediaan sumber daya yang tidak disengaja daripada yang dibutuhkan dan melindungi terhadap tindakan jahat yang dapat meningkatkan tagihan Anda.

Dengan membiasakan diri dengan kuota layanan dan batas tarif AWS API, Anda dapat merencanakan penskalaan beban kerja Anda tanpa khawatir tentang penurunan kinerja yang tidak terduga. Untuk informasi selengkapnya, lihat Meminta pembatasan untuk Amazon ECS API.

Saat menskalakan beban kerja Anda di Amazon ECS, kami sarankan Anda mempertimbangkan kuota layanan berikut.

- AWS Fargate memiliki kuota yang membatasi jumlah tugas yang berjalan bersamaan di masing-masing tugas. Wilayah AWS Ada kuota untuk tugas On-Demand dan Fargate Spot di Amazon ECS. Setiap kuota layanan juga mencakup semua pod Amazon EKS yang Anda jalankan di Fargate.
- Untuk tugas yang berjalan di instans Amazon EC2, jumlah maksimum instans Amazon EC2 yang dapat Anda daftarkan untuk setiap cluster adalah 5.000. Jika Anda menggunakan penskalaan otomatis klaster Amazon ECS dengan penyedia kapasitas grup Auto Scaling, atau jika Anda mengelola instans Amazon EC2 untuk klaster Anda sendiri, kuota ini mungkin menjadi hambatan penerapan. Jika Anda membutuhkan kapasitas lebih, Anda dapat membuat lebih banyak cluster atau meminta peningkatan kuota layanan.
- Jika Anda menggunakan penskalaan otomatis klaster Amazon ECS dengan penyedia kapasitas grup Auto Scaling, pertimbangkan kuota saat menskalakan layanan Anda. `Tasks in the PROVISIONING state per cluster` Kuota ini adalah jumlah maksimum tugas di `PROVISIONING` negara bagian untuk setiap cluster di mana penyedia kapasitas dapat meningkatkan kapasitas. Ketika Anda meluncurkan sejumlah besar tugas sekaligus, Anda dapat dengan mudah memenuhi kuota ini. Salah satu contohnya adalah jika Anda secara bersamaan menyebarkan puluhan layanan, masing-masing dengan ratusan tugas. Ketika ini terjadi, penyedia kapasitas perlu meluncurkan instance kontainer baru untuk menempatkan tugas ketika cluster memiliki kapasitas yang tidak mencukupi. Sementara penyedia kapasitas meluncurkan instans Amazon EC2 tambahan, penjadwal layanan Amazon ECS kemungkinan akan terus meluncurkan tugas secara paralel. Namun, aktivitas ini mungkin terhambat karena kapasitas cluster yang tidak mencukupi. Penjadwal layanan Amazon ECS mengimplementasikan strategi back-off dan eksponensial untuk mencoba kembali penempatan tugas saat instance container baru diluncurkan. Akibatnya, Anda mungkin mengalami waktu penerapan atau penskalaan yang lebih lambat. Untuk

menghindari situasi ini, Anda dapat merencanakan penerapan layanan Anda di salah satu yang berikut. Menerapkan sejumlah besar tugas tidak memerlukan peningkatan kapasitas cluster, atau mempertahankan kapasitas cluster cadangan untuk peluncuran tugas baru.

Selain mempertimbangkan kuota layanan Amazon ECS saat menskalakan beban kerja Anda, pertimbangkan juga kuota layanan untuk yang lain Layanan AWS yang terintegrasi dengan Amazon ECS.

Penyeimbang Beban Elastis

Anda dapat mengonfigurasi layanan Amazon ECS agar menggunakan Elastic Load Balancing untuk mendistribusikan lalu lintas secara merata di seluruh tugas. Untuk informasi selengkapnya dan praktik terbaik yang direkomendasikan tentang cara memilih penyeimbang beban, lihatMendistribusikan lalu lintas layanan Amazon ECS menggunakan load balancing.

Kuota layanan Elastic Load Balancing

Saat Anda menskalakan beban kerja Anda, pertimbangkan kuota layanan Elastic Load Balancing berikut. Sebagian besar kuota layanan Elastic Load Balancing dapat disesuaikan, dan Anda dapat meminta peningkatan konsol Service Quotas.

Application Load Balancer

Bila Anda menggunakan Application Load Balancer, tergantung pada kasus penggunaan Anda, Anda mungkin perlu meminta peningkatan kuota untuk:

- `Targets per Application Load Balancer`Kuota yang merupakan jumlah target di belakang Application Load Balancer Anda.
- `Targets per Target Group per Region`Kuota yang merupakan jumlah target di belakang Grup Target Anda.

Untuk informasi selengkapnya, lihat Kuota untuk Penyeimbang Beban Aplikasi Anda di Panduan Pengguna untuk Penyeimbang Beban Aplikasi.

Network Load Balancer

Ada batasan yang lebih ketat pada jumlah target yang dapat Anda daftarkan dengan Network Load Balancer. Saat menggunakan Network Load Balancer, Anda sering ingin mengaktifkan

dukungan lintas zona, yang dilengkapi dengan batasan penskalaan tambahan pada `Targets per Availability Zone Per Network Load Balancer` jumlah maksimum target per Availability Zone untuk setiap Network Load Balancer. Untuk informasi selengkapnya, lihat Kuota untuk Network Load Balancer Anda di Panduan Pengguna untuk Network Load Balancers.

Pelambatan API Elastic Load Balancing

Saat Anda mengonfigurasi layanan Amazon ECS untuk menggunakan penyeimbang beban, pemeriksaan kesehatan grup target harus lulus sebelum layanan dianggap sehat. Untuk melakukan pemeriksaan kesehatan ini, Amazon ECS memanggil operasi Elastic Load Balancing API atas nama Anda. Jika Anda memiliki sejumlah besar layanan yang dikonfigurasi dengan penyeimbang beban di akun Anda, Anda mungkin akan memperlambat penerapan layanan karena potensi pembatasan khusus untuk operasi API Elastic Load Balancing, dan `RegisterTarget` Elastic `DeregisterTarget` Load `DescribeTargetHealth` Balancing. Saat pelambatan terjadi, kesalahan pelambatan terjadi di pesan peristiwa layanan Amazon ECS Anda.

Jika mengalami pelambatan AWS Cloud Map API, Anda dapat menghubungi AWS Support untuk panduan tentang cara meningkatkan batas pelambatan AWS Cloud Map API Anda. Untuk informasi selengkapnya tentang pemantauan dan pemecahan masalah kesalahan pelambatan tersebut, lihat. Praktik terbaik untuk menangani masalah pelambatan Amazon ECS

Antarmuka jaringan elastis

Dengan tugas Anda menggunakan mode `awsvpc` jaringan, Amazon ECS menyediakan elastic network interface (ENI) yang unik untuk setiap tugas. Jika layanan Amazon ECS Anda menggunakan penyeimbang beban Elastic Load Balancing, antarmuka jaringan ini juga terdaftar sebagai target ke grup target yang sesuai yang ditentukan dalam layanan.

Kuota layanan antarmuka jaringan elastis

Ketika Anda menjalankan tugas yang menggunakan mode `awsvpc` jaringan, sebuah elastic network interface yang unik dilampirkan ke setiap tugas. Jika tugas-tugas tersebut harus dicapai melalui internet, tetapkan alamat IP publik ke elastic network interface untuk tugas-tugas tersebut. Saat Anda menskalakan beban kerja Amazon ECS Anda, pertimbangkan dua kuota penting ini:

- `Network interfaces per Region`Kuota yang merupakan jumlah maksimum antarmuka jaringan di akun Wilayah AWS Anda.
- `Elastic IP addresses per Region`Kuota yang merupakan jumlah maksimum alamat IP elastis dalam file Wilayah AWS.

Kedua kuota layanan ini dapat disesuaikan dan Anda dapat meminta peningkatan dari konsol Service Quotas Anda untuk ini. Untuk informasi selengkapnya, lihat kuota layanan Amazon VPC di Panduan pengguna Amazon Virtual Private Cloud.

Untuk beban kerja Amazon ECS yang dihosting di instans Amazon EC2, saat menjalankan tugas yang menggunakan `awsvpc` mode jaringan pertimbangkan `Maximum network interfaces` kuota layanan, jumlah maksimum instans jaringan untuk setiap instans Amazon EC2. Kuota ini membatasi jumlah tugas yang dapat Anda tempatkan pada sebuah instance. Anda tidak dapat menyesuaikan kuota dan tidak tersedia di konsol Service Quotas. Untuk informasi selengkapnya, lihat alamat IP per antarmuka jaringan per jenis instans di Panduan Pengguna Amazon EC2.

Meskipun Anda tidak dapat mengubah jumlah antarmuka jaringan yang dapat dilampirkan ke instans Amazon EC2, Anda dapat menggunakan fitur trunking elastic network interface untuk meningkatkan jumlah antarmuka jaringan yang tersedia. Misalnya, secara default sebuah `c5.large` instance dapat memiliki hingga tiga antarmuka jaringan. Antarmuka jaringan utama untuk instance dihitung sebagai satu. Jadi, Anda dapat melampirkan dua antarmuka jaringan tambahan ke instance. Karena setiap tugas yang menggunakan mode `awsvpc` jaringan memerlukan antarmuka jaringan, Anda biasanya hanya dapat menjalankan dua tugas tersebut pada jenis instance ini. Hal ini dapat menyebabkan kurangnya pemanfaatan kapasitas cluster Anda. Jika Anda mengaktifkan trunking elastic network interface, Anda dapat meningkatkan kepadatan antarmuka jaringan untuk menempatkan lebih banyak tugas pada setiap instance. Dengan trunking dihidupkan, sebuah `c5.large` instance dapat memiliki hingga 12 antarmuka jaringan. Instans memiliki antarmuka jaringan utama dan Amazon ECS membuat dan melampirkan antarmuka jaringan “trunk” ke instance. Akibatnya, dengan konfigurasi ini Anda dapat menjalankan 10 tugas pada instance alih-alih dua tugas default. Untuk informasi selengkapnya, lihat Pembuatan torso antarmuka jaringan elastis.

Pelambatan API antarmuka jaringan elastis

Saat Anda menjalankan tugas yang menggunakan mode `awsvpc` jaringan, Amazon ECS bergantung pada API Amazon EC2 berikut. Masing-masing API ini memiliki throttle API yang berbeda. Untuk informasi selengkapnya, lihat Meminta pembatasan untuk Amazon EC2 API di Referensi API Amazon EC2.

- CreateNetworkInterface
- AttachNetworkInterface
- DetachNetworkInterface
- DeleteNetworkInterface
- DescribeNetworkInterfaces

- DescribeVpcs
- DescribeSubnets
- DescribeSecurityGroups
- DescribeInstances

Jika panggilan Amazon EC2 API dibatasi selama alur kerja penyediaan antarmuka jaringan elastis, penjadwal layanan Amazon ECS secara otomatis mencoba ulang dengan back-off eksponensial. Pensiun otomatis ini terkadang dapat menyebabkan penundaan dalam meluncurkan tugas, yang menghasilkan kecepatan penerapan yang lebih lambat. Ketika API throttling terjadi, Anda akan melihat pesan `Operations are being throttled. Will try again later.` pada pesan acara layanan Anda. Jika Anda secara konsisten memenuhi throttle Amazon EC2 API, Anda dapat menghubungi AWS Support untuk panduan tentang cara meningkatkan batas pembatasan API Anda. Untuk informasi selengkapnya tentang pemantauan dan pemecahan masalah kesalahan pelambatan, lihat Menangani masalah pelambatan.

AWS Cloud Map

Penemuan layanan Amazon ECS dan Service Connect menggunakan AWS Cloud Map API untuk mengelola ruang nama untuk layanan Amazon ECS Anda. Jika layanan Anda memiliki banyak tugas, pertimbangkan rekomendasi berikut.

AWS Cloud Map kuota layanan

Ketika layanan Amazon ECS dikonfigurasi untuk menggunakan service discovery atau Service Connect, `Tasks per service` kuota yang merupakan jumlah maksimum tugas untuk layanan, dipengaruhi oleh kuota AWS Cloud Map `Instances per service` layanan yang merupakan jumlah maksimum instans untuk layanan tersebut. Secara khusus, kuota AWS Cloud Map layanan mengurangi jumlah tugas yang dapat Anda jalankan ke paling banyak 1.0000 tugas untuk layanan. Anda tidak dapat mengubah AWS Cloud Map kuota. Untuk informasi selengkapnya, lihat AWS Cloud Map service quotas.

AWS Cloud Map Pelambatan API

Amazon ECS memanggil`ListInstances`,, `GetInstancesHealthStatus`, `RegisterInstance`, dan `DeregisterInstance` AWS Cloud Map API atas nama Anda. Mereka membantu dengan penemuan layanan dan melakukan pemeriksaan kesehatan ketika Anda meluncurkan tugas. Ketika beberapa layanan yang menggunakan penemuan layanan dengan sejumlah besar tugas

diterapkan pada saat yang sama, hal ini dapat mengakibatkan melebihi batas pembatasan AWS Cloud Map API. Ketika ini terjadi, Anda mungkin akan melihat pesan berikut: `Operations are being throttled. Will try again later` di pesan acara layanan Amazon ECS Anda dan penerapan yang lebih lambat dan kecepatan peluncuran tugas. AWS Cloud Map tidak mendokumentasikan batas pelambatan untuk API ini. Jika Anda mengalami pembatasan dari ini, Anda dapat menghubungi AWS Support untuk panduan tentang peningkatan batas pelambatan API Anda. Untuk rekomendasi selengkapnya tentang pemantauan dan pemecahan masalah kesalahan pelambatan tersebut, lihat. Praktik terbaik untuk menangani masalah pelambatan Amazon ECS

Referensi API Amazon ECS

Selain AWS Management Console dan AWS Command Line Interface (AWS CLI), Amazon ECS juga menyediakan API. Anda dapat menggunakan API untuk mengotomatiskan tugas untuk mengelola sumber daya Amazon ECS.

- Untuk daftar operasi API menurut resource Amazon ECS, lihat Actions by Amazon ECS resource.
- Untuk daftar abjad operasi API, lihat Tindakan.
- Untuk daftar abjad jenis data, lihat Jenis data.
- Untuk daftar parameter kueri umum, lihat Parameter umum.
- Untuk deskripsi kode kesalahan, lihat Kesalahan umum.

Untuk informasi selengkapnya tentangAWS CLI, lihat AWS Command Line Interfacereferensi untuk Amazon ECS.

Riwayat dokumen

Tabel berikut menjelaskan pembaruan utama dan fitur baru untuk Amazon Elastic Container Service Developer Guide. Kami juga secara rutin memperbarui dokumentasi untuk menjawab umpan balik yang Anda kirimkan kepada kami.

Perubahan	Deskripsi	Tanggal
GMSA untuk Kontainer Linux pada dukungan Fargate	Amazon ECS mendukung otentikasi Direktori Aktif untuk wadah Linux di Fargate melalui jenis akun layanan khusus yang disebut grup Akun Layanan Terkelola (GMSA). Untuk informasi lebih lanjut, lihat Menggunakan gMSA untuk Linux kontainer di Fargate.	Maret 5, 2024
CloudWatch metrik ditambahkan untuk volume Amazon EBS yang dilampirkan ke tugas	Amazon ECS sekarang menerbitkan CloudWatch metrik untuk pemanfaatan penyimpanan Amazon EBS untuk tugas-tugas yang memiliki volume Amazon EBS terpasang. Untuk informasi selengkapnya, lihat CloudWatch metrik Amazon ECS.	Februari 8, 2024
Layanan Connect TLS	Anda sekarang dapat menggunakan TLS dengan Service Connect.	Januari 22, 2024
Kebijakan terkelola Service Connect TLS	Menambahkan kebijakan AmazonECS InfrastructureRolePolicyForServiceConnectTransportLayerSecurity baru.	Januari 22, 2024
Pembaruan konfigurasi batas waktu tunggu Service Connect	Konfigurasi batas waktu Service Connect sekarang dapat diperbarui dan mencakup dua parameter opsional - `idleTimeout` dan `perRequestTimeout`.	Januari 22, 2024
Pengurasan instans terkelola Amazon ECS	Anda dapat menggunakan pengurasan instans terkelola Amazon ECS untuk memfasilitasi penghentian instans Amazon ECS secara anggun.	Januari 19, 2024

Perubahan	Deskripsi	Tanggal
Dukungan Ubuntu 22 ditambahkan untuk ECS Anywhere	Support untuk sistem operasi Ubuntu 22 ditambahk an ke ECS Anywhere. Untuk informasi selengkapnya, lihat Sistem pengoperasian dan arsitektur sistem yang didukung.	Januari 16, 2024
Tambahkan `AmazonECS Infrastru ctureRole PolicyForVolumes` kebijakan IAM	`AmazonECSInfrastructureolePolicyForV olumes` telah ditambahkan. Kebijakan ini memberika n izin yang diperlukan oleh Amazon ECS untuk melakukan panggilan AWS API guna mengelola volume Amazon EBS yang terkait dengan beban kerja Amazon ECS.	Januari 11, 2024
Volume data Amazon EBS untuk tugas Amazon ECS	Anda dapat mengonfigurasi 1 volume data Amazon EBS per tugas selama penerapan lampiran ke tugas atau tugas Amazon ECS mandiri yang dikelola oleh layanan ECS. Mengonfigurasi volume saat penerapan memungkinkan Anda membuat definisi tugas yang dapat dilanjutkan yang tidak dibatasi untuk jenis atau pengaturan volume tertentu. Volume Amazon EBS menyediakan penyimpanan blok yang sangat tersedia, hemat biaya, tahan lama, dan berkinerja tinggi untuk beban kerja kontainer intensif data.	Januari 11, 2024
Konsol klasik Amazon ECS mencapai akhir masa pakai	Konsol Amazon ECS telah mencapai akhir hayat.	Desember 4, 2023
Kebijakan yang diperbarui	Kebijakan IAM terkelola ServiceRolePolicy AmazonECS diperbarui dengan `events` izin baru dan tambahan serta izin. `autoscaling autoscali ng-plans`	Desember 4, 2023
Dukungan Runtime Monitoring	Anda dapat menggunakan Runtime Monitoring untuk memantau beban kerja Amazon ECS untuk mengident ifikasi perilaku berbahaya atau tidak sah. Untuk informasi selengkapnya, lihat Runtime Monitoring.	26 November 2023

Perubahan	Deskripsi	Tanggal
Kebijakan yang diperbarui	Kebijakan IAM `AmazonECSServiceRolePolicy` terkelola telah diperbarui untuk mengizinkan akses ke AWS Cloud Map `DiscoverInstancesRevision` API.	4 Oktober 2023
AWS Fargate konfigurasi tugas pensiun	Anda dapat mengonfigurasi periode tunggu sebelum tugas Fargate dihentikan Untuk informasi selengkapnya, lihat AWS Fargate pemeliharaan tugas.	5 September 2023
Parameter definisi tugas tambahan di AWS Fargate	AWS Fargate menambahkan dukungan untuk `pidMode` dan `systemControls` dalam versi platform Linux`1.4.0`. Untuk informasi selengkapnya, lihat Definisi tugas.	9 Agustus 2023
Desain ulang halaman definisi tugas konsol Amazon ECS	Halaman definisi tugas di konsol Amazon ECS telah didesain ulang dan berisi opsi tambahan. Untuk informasi selengkapnya, lihat Membuat definisi tugas menggunakan konsol.	26 Juli 2023
Fargate mendukung pemuatan lambat dengan indeks OCI Seekable	AWS Fargate memperkenalkan indeks Seekable OCI (SOCI). Dengan SOCI, kontainer hanya menghabiskan beberapa detik pada penarikan gambar sebelum mereka dapat memulai, menyediakan waktu untuk pengaturan lingkungan dan instantiasi aplikasi saat gambar diunduh di latar belakang. Untuk informasi selengkapnya, lihat Lazy loading image container menggunakan Seekable OCI (SOCI) di Panduan Pengguna Amazon ECS untuk Fargate. AWS	Juli 17, 2023

Perubahan	Deskripsi	Tanggal
Peningkatan dukungan untuk GMSA di Linux dan Windows	Definisi tugas memiliki `credentialSpecs` bidang baru untuk GMSA untuk Linux dan Windows. Tutorial lengkap baru untuk GMSA tanpa domain di Windows telah ditambahkan, lihat Tutorial: Menggunakan Wadah Windows dengan GMSA Tanpa Domain menggunakan file. AWS CLI Untuk informasi selengkapnya, lihat Menggunakan GMSAs untuk Kontainer Linux dan Menggunakan GMSAs untuk Kontainer Windows.	14 Juli 2023
Peningkatan versi ECS Agent dokumentasi	Dokumentasi untuk versi Amazon ECS Agent telah diperbarui. Kami menyarankan Anda menggunakan `v20.10.13` versi atau yang lebih baru dari Docker dengan versi terbaru dari agen penampung Amazon ECS. Versi yang dirilis dan perubahan pada agen tersedia di GitHub. Untuk informasi selengkapnya, lihat versi agen penampung Amazon ECS Linux.	20 Juni 2023
Ketersediaan Wilayah yang diperbarui untuk dukungan Fargate ARM64	Ketersediaan Wilayah untuk dukungan Fargate ARM64 telah diperbarui. Untuk informasi selengkap nya, lihat Pertimbangan.	19 Juni 2023
Tingkatkan dokumenta si penskalaan otomatis cluster	Dokumentasi untuk penskalaan Amazon ECS dari Amazon EC2 Auto Scaling memiliki peningkatan signifikan dalam akurasi dan keterbacaan. Untuk informasi selengkapnya, lihat penskalaan otomatis kluster Amazon ECS.	4 Mei 2023

Perubahan	Deskripsi	Tanggal
Menandai otorisasi untuk pembuatan sumber daya.	Pengguna harus memiliki izin untuk tindakan yang membuat sumber daya, sepertiecsCreateCluster . Saat Anda membuat sumber daya dan menentuka n tag untuk sumber daya tersebut, AWS lakukan otorisasi tambahan untuk memverifikasi bahwa ada izin untuk membuat tag. Untuk informasi selengkap nya, lihat Otorisasi penandaan dan Memberikan izin untuk menandai sumber daya pada pembuatan.	18 April 2023
Support untuk GMSA untuk kontainer Linux di EC2	Anda dapat menggunakan GMSA untuk mengauten tikasi ke Active Directory untuk wadah Linux di EC2. Untuk informasi selengkapnya, lihat Menggunakan GMSAs untuk Kontainer Linux.	April 14, 2023
Support untuk penyimpan an sementara untuk wadah Windows aktif AWS Fargate	Anda dapat menggunakan penyimpanan sementara untuk wadah Windows aktif. AWS Fargate Untuk informasi selengkapnya, lihat Penyimpanan tugas Fargate.	April 14, 2023
AWS Cost Managemen t dukungan untuk data CUR tingkat tugas	Anda dapat mengaktifkan biaya tingkat tugas dan penggunaan sumber daya di Laporan Biaya dan Penggunaan. Ini menambahkan Data Alokasi Biaya Split untuk tugas-tugas yang berjalan di AWS Fargate dan EC2. Untuk informasi selengkapnya, lihat Laporan Biaya dan Penggunaan Tingkat Tugas.	12 April 2023
Amazon Linux 2023 Amazon ECS yang dioptimalkan AMI	Anda dapat menerapkan beban kerja di AMI Amazon Linux 2023 Amazon ECS yang dioptimalkan. Untuk informasi selengkapnya, lihat AMI Amazon ECS yang dioptimalkan.	April 10, 2023
AWS Fargate Standar Pemrosesan Informasi Federal (FIPS) 140	Anda dapat menerapkan beban kerja di AWS Fargate Amazon ECS dengan cara yang sesuai dengan Federal Information Processing Standard (FIPS) 140. Untuk informasi selengkapnya, lihat AWS Fargate Standar Pemrosesan Informasi Federal (FIPS-140).	April 10, 2023

Perubahan	Deskripsi	Tanggal
Penghapusan definisi tugas	Anda dapat menghapus definisi tugas menggunakan konsol Amazon ECS, SDK, dan. AWS CLI Untuk informasi selengkapnya, lihat Menghapus revisi definisi tugas menggunakan konsol dan definisi Tugas.	Februari 24, 2023
AWS Fargate rekomendasi layanan di Compute Optimizer	AWS Compute Optimizer menghasilkan rekomendasi ukuran tugas dan wadah berdasarkan pemanfaatan tugas yang sedang berjalan di layanan Amazon ECS di Fargate. AWS Untuk informasi selengkapnya, lihat Melihat rekomendasi untuk layanan Amazon ECS di Fargate.	27 Januari 2023
Konsol Amazon ECS	Konsol Amazon ECS yang baru sekarang menjadi konsol default. Untuk informasi selengkapnya, lihat Konsol Amazon ECS baru.	19 Januari 2023
Kebijakan `AmazonECS_FullAccess` IAM yang diperbarui	Kebijakan `AmazonECS_FullAccess` IAM diperbarui untuk menyertakan izin untuk menambahkan tag ke penyeimbang beban selama pembuatan. Untuk informasi selengkapnya, lihat Amazonecs_ FullAccess.	4 Januari 2023
Gunakan CloudWatch alarm untuk mendeteksi kegagalan penerapan layanan Amazon ECS	Anda dapat mengonfigurasi Amazon ECS untuk menyetel penerapan menjadi gagal saat mendeteksi bahwa CloudWatch alarm tertentu telah masuk ke status ALARM. Untuk informasi selengkapnya, lihat the section called “Metode deteksi kegagalan”.	19 Desember 2022
Support untuk pemetaan port kontainer	Anda dapat mengatur rentang nomor port pada wadah yang terikat ke rentang port host yang dipetakan secara dinamis. Untuk informasi selengkapnya, lihat the section called “Pemetaan pelabuhan”.	Desember 15, 2022
Ketersediaan umum Amazon ECS Service Connect	Fitur ini menambahkan penemuan layanan dan mesh layanan yang dikendalikan oleh penyebaran layanan Amazon ECS. Untuk informasi selengkapnya, lihat the section called “Layanan Connect”.	27 November 2022

Perubahan	Deskripsi	Tanggal
Pengalaman konsol Amazon ECS baru untuk definisi tugas diperbarui	Pengalaman konsol Amazon ECS yang baru sekarang berisi editor JSON untuk definisi tugas. Untuk informasi selengkapnya, lihat the section called “Membuat definisi tugas menggunakan konsol”.	27 Oktober 2022
Pengalaman konsol Amazon ECS baru untuk definisi tugas diperbarui	Pengalaman konsol Amazon ECS yang baru sekarang berisi editor JSON untuk definisi tugas. Untuk informasi selengkapnya, lihat the section called “Membuat definisi tugas menggunakan konsol”.	27 Oktober 2022
Pengalaman konsol Amazon ECS baru diperbarui	Pengalaman konsol Amazon ECS yang baru telah diperbarui dengan parameter layanan dan tugas tambahan. Lihat informasi yang lebih lengkap di the section called “Membuat sebuah layanan” dan the section called “Jalankan aplikasi sebagai tugas Amazon ECS”.	Oktober 7, 2022
Informasi baru di titik akhir metadata tugas versi 4	Titik akhir metadata tugas versi 4 sekarang menyertakan ID VPC dan nama layanan. Untuk informasi selengkapnya, lihat the section called “Titik akhir metadata tugas versi 4”.	Oktober 7, 2022
Ukuran definisi tugas baru	Amazon ECS di Fargate sekarang mendukung ukuran tugas 8 vCPU dan 16 vCPU. Untuk informasi selengkapnya, lihat the section called “Ukuran tugas”.	September 16, 2022
Halaman ECS CLI diarsipkan	Dokumentasi CLI ECS telah diarsipkan. Kami merekomendasikan menggunakan AWS Copilot untuk kebutuhan alat baris perintah Anda. Untuk informasi selengkapnya, lihat Menggunakan antarmuka baris perintah AWS Copilot.	15 September 2022
Kuota Fargate baru	Fargate sedang beralih dari kuota berbasis hitungan tugas ke kuota berbasis VCPU. Untuk informasi selengkapnya, lihat the section called “AWS Fargate kuota layanan”.	September 8, 2022

Perubahan	Deskripsi	Tanggal
Dukungan untuk kolam hangat Amazon EC2 Auto Scaling.	Anda sekarang dapat menggunakan kolam hangat Amazon EC2 Auto Scaling untuk meningkatkan skala aplikasi Anda lebih cepat dan menghemat biaya. Untuk informasi selengkapnya, lihat Menggunakan kolam hangat untuk grup Auto Scaling Anda.	Maret 23, 2022
Support untuk instance Windows di ECS Anywhere.	ECS Anywhere sekarang mendukung instance Windows. Untuk informasi selengkapnya, lihat Instans eksternal (Amazon ECS Anywhere).	3 Maret 2022
Menambahkan dukungan ECS Exec untuk instans eksternal	ECS Exec sekarang didukung untuk instans eksternal . Untuk informasi selengkapnya, lihat Pantau wadah Amazon ECS dengan ECS Exec.	24 Januari 2022
Pengalaman konsol Amazon ECS baru diperbarui	Pengalaman konsol Amazon ECS yang baru mendukung pembuatan dan penghapusan klaster, memperbarui definisi tugas, dan membatalkan pendaftaran definisi tugas. Untuk informasi lebih lanjut, lihat Membuat cluster untuk jenis peluncuran Fargate dan Eksternal menggunakan konsol, Menghapus cluster menggunakan konsol, Memperbarui definisi tugas menggunakan konsol, dan Membatalkan pendaftaran revisi definisi tugas menggunakan konsol .	8 Desember 2021
Pengalaman konsol Amazon ECS baru diperbarui	Pengalaman konsol Amazon ECS yang baru mendukung pembuatan definisi tugas. Untuk informasi selengkapnya, lihat Membuat definisi tugas menggunakan konsol.	23 November 2021
Amazon ECS mendukung arsitektur ARM 64-bit untuk Linux.	Amazon ECS mendukung arsitektur CPU ARM 64-bit untuk sistem operasi Linux. Untuk informasi selengkap nya, lihat the section called “Bekerja dengan beban kerja ARM 64-bit di Amazon ECS”.	23 November 2021

Perubahan	Deskripsi	Tanggal
Dukungan Amazon ECS untuk opsi lancar log-driver-buffer-limit	Amazon ECS mendukung opsi lancar`log-driver-buffer-limit` . Untuk informasi selengkapnya, lihat Menggunakan perutean log khusus.	22 November 2021
Skrip pembuatan AMI Linux AMI yang dioptimal kan Amazon ECS	Amazon ECS telah membuka skrip build yang digunakan untuk membangun varian Linux dari AMI Amazon ECS yang dioptimalkan. Untuk informasi selengkapnya, lihat Skrip pembuatan AMI Linux AMI yang dioptimalkan Amazon ECS.	November 19, 2021
Kesehatan contoh kontainer	Amazon ECS menambahkan dukungan untuk pemantauan kesehatan instans kontainer. Untuk informasi selengkapnya, lihat Pantau kesehatan instans kontainer Amazon ECS.	November 10, 2021
Support untuk Windows Amazon ECS Exec	Amazon ECS Exec mendukung Windows. Untuk informasi selengkapnya, lihat Pantau wadah Amazon ECS dengan ECS Exec.	November 1, 2021
Support untuk wadah Windows di Fargate.	Amazon ECS mendukung wadah Windows di Fargate. Untuk informasi selengkapnya, lihat Fargate versi platform Windows.	28 Oktober 2021
Dukungan GPU untuk instans eksternal di Amazon ECS Anywhere	Amazon ECS mendukung penetapan persyaratan GPU dalam definisi tugas untuk tugas yang dijalanka n pada instans eksternal. Lihat informasi yang lebih lengkap di Bekerja dengan GPU di Amazon ECS dan Pendaftaran instans eksternal ke sebuah klaster.	Oktober 8, 2021
Dukungan dari mode jaringan `awsvpc` di Windows	Amazon ECS mendukung mode `awsvpc` jaringan pada Windows. Untuk informasi selengkapnya, lihat `awsvpc`modus jaringan.	15 Juli 2021

Perubahan	Deskripsi	Tanggal
Bottlerocket yang umumnya tersedia	Amazon ECS mendukung varian AMI Amazon ECS yang dioptimalkan dari sistem operasi Bottlerocket disediakan sebagai AMI. Untuk informasi selengkap nya, lihat AMI Amazon ECS Bottlerocket yang dioptimalkan.	30 Juni 2021
Pembaruan tugas terjadwal Amazon ECS	Amazon EventBridge menambahkan dukungan untuk parameter tambahan saat membuat aturan yang memicu tugas terjadwal Amazon ECS.	25 Juni 2021
AWS kebijakan terkelola untuk Amazon ECS	Amazon ECS menambahkan dokumentasi kebijakan AWS terkelola untuk peran terkait layanan. Untuk informasi selengkapnya, lihat AWS kebijakan terkelola untuk Amazon Elastic Container Service.	8 Juni 2021
Memulai dengan AWS CDK	Menambahkan panduan memulai untuk menggunak an AWS CDK dengan Amazon ECS. Untuk informasi selengkapnya, lihat Memulai Amazon ECS menggunak an AWS CDK.	27 Mei 2021
Amazon ECS Anywhere	Amazon ECS telah menambahkan dukungan untuk mendaftarkan server on-premise atau mesin virtual (VM) dengan cluster Anda. Untuk informasi selengkap nya, lihat Instans eksternal (Amazon ECS Anywhere).	25 Mei 2021
Amazon ECS yang dioptimalkan Windows Server 20H2 Core AMI	Amazon ECS telah menambahkan dukungan untuk varian AMI Windows Amazon ECS baru yang dioptimal kan untuk Windows Server 20H2 Core. Untuk informasi selengkapnya, lihat AMI Amazon ECS yang dioptimalkan.	19 April 2021
Amazon ECS Exec	Amazon ECS telah merilis alat debugging baru yang disebut ECS Exec. Untuk informasi selengkapnya, lihat Pantau wadah Amazon ECS dengan ECS Exec.	15 Maret 2021

Perubahan	Deskripsi	Tanggal
Dukungan kebijakan VPC endpoint	Amazon ECS sekarang mendukung kebijakan titik akhir VPC. Untuk informasi selengkapnya, lihat Membuat kebijakan titik akhir VPC untuk Amazon ECS.	11 Januari 2021
Pengalaman konsol baru	Amazon ECS telah merilis pengalaman konsol baru yang mendukung pembuatan atau pembaruan layanan atau menjalankan tugas mandiri. Lihat informasi yang lebih lengkap di Membuat layanan menggunakan konsol dan Jalankan aplikasi sebagai tugas Amazon ECS.	28 Desember 2020
Pembaruan penyedia kapasitas	Amazon ECS menambahkan dukungan untuk memperbarui penyedia kapasitas grup Auto Scaling yang ada.	23 November 2020
ECS sekarang mendukung Amazon FSx for Windows File Server untuk tugas Windows	Amazon ECS menambahkan dukungan untuk menentukan volume Amazon FSx for Windows File Server dalam definisi tugas Windows. Untuk informasi selengkapnya, lihat Volume FSx for Windows File Server.	11 November 2020
Dukungan mode tumpukan dobel VPC ditambahkan	Amazon ECS menambahkan dukungan untuk menggunakan VPC dalam mode dual-stack dengan tugas menggunakan mode jaringan, yang menyediakan dukungan `awsvpc` untuk alamat IPv6. Untuk informasi selengkapnya, lihat Menggunakan VPC dalam mode tumpukan dobel.	5 November 2020
Pembaruan v4 titik akhir metadata tugas	Amazon ECS menambahkan metadata tambahan ke output v4 titik akhir metadata tugas. Untuk informasi selengkapnya, lihat Titik akhir metadata tugas Amazon ECS versi 4.	5 November 2020

Perubahan	Deskripsi	Tanggal
Dukungan untuk Local Zones dan Zona Wavelength	Amazon ECS menambahkan dukungan untuk beban kerja di Local Zones dan Wavelength Zones. Untuk informasi selengkapnya, lihat Aplikasi Amazon ECS di subnet bersama, Local Zones, dan Wavelength Zones.	Selasa, 04 September 2020
Varian Amazon ECS dari Bottlerocket AMI	Bottlerocket adalah sistem operasi open source berbasis Linux yang dibuat khusus untuk menjalank an kontainer. AWS Varian AMI yang dioptimal kan Amazon ECS dari sistem operasi Bottlerocket disediakan sebagai AMI yang dapat Anda gunakan saat meluncurkan instans kontainer Amazon ECS. Untuk informasi selengkapnya, lihat AMI Amazon ECS Bottlerocket yang dioptimalkan.	31 Agustus 2020
Titik akhir metadata tugas versi 4 diperbarui untuk statistik tingkat jaringan	Titik akhir metadata tugas versi 4 telah diperbaru i untuk menyediakan statistik laju jaringan untuk tugas Amazon ECS yang menggunakan `awsvpc` mode `bridge` atau jaringan yang dihosting di instans Amazon EC2 yang menjalankan setidaknya versi agen penampung. `1.43.0` Untuk informasi selengkapnya, lihat Titik akhir metadata tugas Amazon ECS versi 4.	10 Agustus 2020
Metrik penggunaan Fargate	AWS Fargate menyediakan metrik CloudWatch penggunaan yang memberikan visibilitas ke akun Anda, penggunaan sumber daya Fargate On-Demand dan Fargate Spot. Untuk informasi selengkapnya, lihat Metrik penggunaan.	3 Agustus 2020
AWS Copilot versi 0.1.0	AWS Copilot CLI baru diluncurkan, menyediakan perintah tingkat tinggi untuk menyederhanakan pemodelan, pembuatan, rilis, dan pengelolaan aplikasi kontainer di Amazon ECS dari lingkungan pengembangan lokal. Untuk informasi selengkapnya, lihat Menggunakan antarmuka baris perintah AWS Copilot.	9 Juli 2020

Perubahan	Deskripsi	Tanggal
AWS Fargate jadwal penghentian versi platform	Jadwal penghentian versi platform Fargate telah ditambahkan. Untuk informasi selengkapnya, lihat AWS Penghentian versi platform Fargate Linux.	8 Juli 2020
AWS Perluasan Wilayah Fargate	Amazon ECS di AWS Fargate telah berkembang ke Wilayah Eropa (Milan).	25 Juni 2020
Amazon ECS dioptimal kan Amazon Linux 2 (Neuron) AMI dirilis	Amazon ECS merilis Amazon ECS yang dioptimal kan Amazon Linux 2 (Neuron) AMI untuk beban kerja inferensial. Untuk informasi selengkapnya, lihat AMI Amazon ECS yang dioptimalkan.	24 Juni 2020
Menambahkan dukungan untuk menghapus penyedia kapasitas	Amazon ECS menambahkan dukungan untuk menghapus penyedia kapasitas grup Auto Scaling.	11 Juni 2020
AWS Pembaruan platform Fargate versi 1.4.0	Mulai 28 Mei 2020, setiap tugas Fargate baru yang diluncurkan menggunakan platform versi 1.4.0 akan memiliki penyimpanan singkat 20 GB yang dienkrips i dengan algoritme enkripsi AES-256 menggunak an kunci enkripsi yang dikelola. AWS Fargate Untuk informasi selengkapnya, lihat Penyimpanan sementara tugas Fargate.	28 Mei 2020
Dukungan file variabel lingkungan	Menambahkan dukungan untuk menentukan file variabel lingkungan dalam ketentuan tugas, yang memungkinkan Anda untuk menambahkan banyak variabel lingkungan ke kontainer Anda. Untuk informasi selengkapnya, lihat Gunakan parameter definisi tugas untuk meneruskan variabel lingkungan ke wadah.	18 Mei 2020
AWS Perluasan Wilayah Fargate	AWS Fargate dengan Amazon ECS telah diperluas ke Wilayah Afrika (Cape Town).	11 Mei 2020

Perubahan	Deskripsi	Tanggal
Service Quotas diperbarui	Service Quotas berikut telah diperbarui: • Klaster per akun ditingkatkan dari `2,000` ke `10,000`. Untuk informasi selengkapnya, lihat Kuota layanan Amazon ECS.	17 April 2020

Perubahan	Deskripsi	Tanggal
AWS Platform Fargate versi 1.4.0	AWS Platform Fargate versi 1.4.0 dirilis, yang berisi fitur-fitur berikut: • Menambahkan dukungan untuk menggunakan volume sistem file Amazon EFS untuk penyimpanan tugas persisten. Untuk informasi selengkapnya, lihat Volume Amazon EFS. • Penyimpanan tugas sementara telah ditingkatkan menjadi 20 GB. Untuk informasi selengkapnya, lihat Penyimpanan sementara tugas Fargate. • Perilaku lalu lintas jaringan ke dan dari tugas telah diperbarui. Dimulai dengan platform versi 1.4, semua tugas Fargate menerima satu elastic network interface (disebut sebagai tugas ENI) dan semua lalu lintas jaringan mengalir melalui ENI tersebut dalam VPC Anda dan akan terlihat oleh Anda melalui log aliran VPC Anda. Untuk informasi selengkapnya, lihat Jaringan Tugas Fargate di Panduan Pengguna Layanan Kontainer Elastis Amazon untuk. AWS Fargate • ENI tugas menambahkan dukungan untuk frame jumbo. Antarmuka jaringan dikonfigurasi dengan unit transmisi maksimum (MTU), yang merupakan ukuran muatan terbesar yang muat dalam satu frame. Semakin besar MTU, semakin banyak muatan aplikasi yang termuat dalam satu frame, yang mengurangi overhead per frame dan meningkatkan efisiensi. Dengan mendukung jumbo frame akan dapat mengurangi overhead ketika jalur jaringan antara tugas dengan tujuan Anda	8 April 2020

Perubahan	Deskripsi	Tanggal
	mendukung jumbo frame, seperti semua lalu lintas yang tetap berada dalam VPC Anda. • CloudWatch Wawasan Kontainer akan menyertakan metrik kinerja jaringan untuk tugas Fargate. Untuk informasi selengkapnya, lihat Pantau kontainer Amazon ECS menggunakan Wawasan Kontainer. • Menambahkan dukungan untuk titik akhir metadata tugas v4 yang menyediakan informasi tambahan untuk tugas Fargate Anda, termasuk statistik jaringan untuk tugas dan Zona Ketersediaan mana tugas sedang berjalan. Untuk informasi selengkap nya, lihat Titik akhir metadata tugas Amazon ECS versi 4. • Menambahkan dukungan untuk parameter Linux SYS_PTRACE dalam ketentuan kontainer. Untuk informasi selengkapnya, lihat Parameter Linux. • Agen kontainer Fargate menggantikan penggunaa n agen kontainer Amazon ECS untuk semua tugas Fargate. Perubahan ini seharusnya tidak berpengar uh pada cara tugas Anda berjalan. • Waktu aktif kontainer kini menggunakan Container d bukan Docker. Perubahan ini seharusnya tidak berpengaruh pada cara tugas Anda berjalan. Anda akan melihat bahwa beberapa pesan kesalahan yang berasal dari waktu aktif kontainer akan berubah dari menyebutkan Docker menjadi kesalahan yang lebih umum. Untuk informasi selengkapnya, lihat Versi platform Fargate Linux.	

Perubahan	Deskripsi	Tanggal
Dukungan sistem file Amazon EFS untuk volume tugas	Sistem file Amazon EFS dapat digunakan sebagai volume data untuk tugas Amazon ECS dan Fargate Anda. Untuk informasi selengkapnya, lihat Volume Amazon EFS.	8 April 2020
Titik Akhir Metadata Tugas Amazon ECS versi 4	Dimulai dengan agen kontainer Amazon ECS versi 1.39.0 dan platform Fargate versi 1.4.0, variabel lingkungan `ECS_CONTAINER_METADATA_URI_V4` bernama disuntikkan ke setiap wadah dalam suatu tugas. Saat Anda melakukan kueri terhadap titik akhir metadata tugas versi 4, berbagai metadata tugas dan statistik Docker tersedia untuk tugas. Untuk informasi selengkapnya, lihat Titik akhir metadata tugas Amazon ECS versi 4.	8 April 2020
Support untuk versi spesifik rahasia Secrets Manager untuk disuntikkan sebagai variabel lingkungan	Menambahkan dukungan untuk menentukan data sensitif menggunakan versi rahasia Secrets Manager tertentu. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah.	24 Februari 2020
Menambahkan opsi konfigurasi CodeDeploy penerapan tambahan untuk penerapan biru/hijau	CodeDeploy Layanan ini menambahkan konfigurasi penerapan canary dan linier baru untuk jenis penyebaran Amazon ECS. Kemampuan untuk menentukan konfigurasi deployment kustom juga tersedia. Untuk informasi selengkapnya, lihat Penerapan Biru/Hijau dengan CodeDeploy.	6 Februari 2020
Ditambahkan parameter definisi efsVolumeConfiguration tugas	Parameter definisi `efsVolumeConfiguration` tugas ada di pratinjau publik, yang memudahkan penggunaan sistem file Amazon EFS dengan tugas Amazon ECS Anda. Untuk informasi selengkapnya, lihat Volume Amazon EFS.	17 Januari 2020

Perubahan	Deskripsi	Tanggal
Perilaku pencatatan agen penampung Amazon ECS diperbarui	Lokasi pencatatan agen kontainer Amazon ECS dan perilaku rotasi telah diperbarui. Untuk informasi selengkapnya, lihat Log Agen Kontainer Amazon ECS.	13 Januari 2020
Spot Fargate	Amazon ECS menambahkan dukungan untuk menjalankan tugas menggunakan Fargate Spot. Untuk informasi selengkapnya, lihat AWS Fargate penyedia kapasitas.	3 Desember 2019
Auto Scaling Klaster	Penskalaan otomatis klaster Amazon ECS memungkinkan Anda memiliki kontrol lebih besar atas cara Anda menskalakan tugas dalam klaster. Untuk informasi selengkapnya, lihat Penskalaan otomatis Klaster Amazon ECS.	3 Desember 2019
Penyedia Kapasitas Klaster	Penyedia kapasitas cluster Amazon ECS menentukan infrastruktur yang akan digunakan untuk tugas Anda. Untuk informasi selengkapnya, lihat Penyedia kapasitas Amazon ECS.	3 Desember 2019
Membuat cluster pada AWS Outposts	Amazon ECS sekarang mendukung pembuatan cluster di file. AWS Outposts Untuk informasi selengkapnya, lihat the section called “Layanan Kontainer Elastis Amazon aktif AWS Outposts”.	3 Desember 2019
Kejadian Tindakan Layanan	Amazon ECS sekarang mengirimkan peristiwa ke Amazon EventBridge ketika tindakan layanan tertentu terjadi. Untuk informasi selengkapnya, lihat Acara tindakan layanan Amazon ECS.	25 November 2019
Amazon ECS GPU AMI yang Dioptimalkan Mendukung Instans G4	Amazon ECS menambahkan dukungan untuk keluarga tipe instans g4 saat menggunakan AMI yang dioptimalkan untuk GPU Amazon ECS. Untuk informasi selengkapnya, lihat Bekerja dengan GPU di Amazon ECS.	8 Oktober 2019

Perubahan	Deskripsi	Tanggal
FireLens untuk Amazon ECS	FireLens untuk Amazon ECS dalam ketersediaan umum. FireLens Amazon ECS memungkinkan Anda menggunakan parameter definisi tugas untuk merutekan log ke AWS layanan atau tujuan mitra untuk penyimpanan log dan analitik. Untuk informasi selengkapnya, lihat Menggunakan perutean log khusus.	30 September 2019
AWS Perluasan wilayah Fargate	AWS Fargate dengan Amazon ECS telah berkembang ke wilayah Eropa (Paris), Eropa (Stockholm), dan Timur Tengah (Bahrain).	30 September 2019
Deep Learning Containers dengan Elastic Inference di Amazon ECS	Amazon ECS mendukung pemasangan akselerator Amazon Elastic Inference ke container Anda untuk membuat menjalankan beban kerja inferensi pembelajaran mendalam menjadi lebih efisien. Untuk informasi selengkapnya, lihat Deep Learning Containers dengan Elastic Inference di Amazon ECS.	3 September 2019
FireLens untuk Amazon ECS	FireLens untuk Amazon ECS ada di pratinjau publik. FireLens Amazon ECS memungkinkan Anda menggunakan parameter definisi tugas untuk merutekan log ke AWS layanan atau tujuan mitra untuk penyimpanan log dan analitik. Untuk informasi selengkapnya, lihat Menggunakan perutean log khusus.	30 Agustus 2019
CloudWatch Wawasan Kontainer	CloudWatch Wawasan Kontainer sekarang tersedia secara umum. Aplikasi ini memungkinkan Anda untuk mengumpulkan, menggabungkan, serta merangkum metrik dan log dari aplikasi dan layanan mikro terkontainerisasi Anda. Untuk informasi selengkapnya, lihat Pantau kontainer Amazon ECS menggunakan Wawasan Kontainer.	30 Agustus 2019

Perubahan	Deskripsi	Tanggal
Konfigurasi Pertukaran Tingkat Kontainer	Amazon ECS menambahkan dukungan untuk mengontrol penggunaan ruang memori swap pada instance container Linux Anda di tingkat container . Dengan menggunakan konfigurasi pertukaran per kontainer, setiap kontainer dalam ketentuan tugas dapat mengaktifkan atau menonaktifkan pertukaran, dan untuk yang diaktifkan, jumlah maksimum ruang tukar yang digunakan dapat dibatasi. Untuk informasi selengkapnya, lihat Mengelola ruang tukar kontainer.	16 Agustus 2019
AWS Perluasan wilayah Fargate	AWS Fargate dengan Amazon ECS telah berkembang ke Wilayah Asia Pasifik (Hong Kong).	6 Agustus 2019
Trunking Antarmuka Jaringan Elastis	Menambahkan jenis instans Amazon EC2 tambahan yang didukung untuk fitur trunking ENI. Untuk informasi selengkapnya, lihat Tipe instans Amazon EC2 yang didukung.	1 Agustus 2019
Mendaftarkan Beberapa Grup Target dengan sebuah Layanan	Menambahkan dukungan untuk menentukan beberapa grup target dalam sebuah definisi layanan. Untuk informasi selengkapnya, lihat Mendaftarkan beberapa grup target dengan layanan.	30 Juli 2019
Menentukan Data Sensitif Menggunakan Rahasia Secrets Manager	Ditambahkan tutorial untuk menentukan data sensitif menggunakan rahasia Secrets Manager. Untuk informasi selengkapnya, lihat Menentukan Data Sensitif Menggunakan Rahasia Secrets Manager.	Juli 20, 2019
CloudWatch Wawasan Kontainer	Amazon ECS telah menambahkan dukungan untuk CloudWatch Container Insights. Untuk informasi selengkapnya, lihat Pantau kontainer Amazon ECS menggunakan Wawasan Kontainer.	9 Juli 2019

Perubahan	Deskripsi	Tanggal
Izin tingkat sumber daya untuk layanan dan set tugas Amazon ECS	Amazon ECS telah memperluas dukungan izin tingkat sumber daya untuk layanan dan tugas Amazon ECS. Untuk informasi selengkapnya, lihat Bagaimana Amazon Elastic Container Service bekerja dengan IAM.	27 Juni 2019
AMI Amazon ECS baru yang dioptimalkan ditambal untuk -2019-005 AWS	Amazon ECS telah memperbarui AMI yang dioptimal kan Amazon ECS untuk mengatasi kerentanan yang dijelaskan dalam -2019-005.AWS	17 Juni 2019
Trunking Antarmuka Jaringan Elastis	Amazon ECS memperkenalkan dukungan untuk meluncurkan instans kontainer menggunakan tipe instans Amazon EC2 yang didukung yang telah meningkatkan kepadatan elastic network interface (ENI). Dengan menggunakan tipe instans ini dan ikut serta dalam pengaturan akun `awsvpcTrunking` akan meningkatkan kepadatan ENI pada instans kontainer yang baru diluncurkan yang mengizinkan Anda untuk menempatkan lebih banyak tugas pada setiap instans kontainer. Untuk informasi selengkap nya, lihat Pembuatan torso antarmuka jaringan elastis.	6 Juni 2019
AWS Pembaruan platform Fargate versi 1.3.0	Mulai 1 Mei 2019, setiap tugas Fargate baru yang diluncurkan mendukung driver `splunk` log selain driver `awslogs` log. Untuk informasi selengkapnya, lihat Penyimpanan dan pencatatan.	1 Mei 2019
AWS Pembaruan platform Fargate versi 1.3.0	Mulai 1 Mei 2019, tugas Fargate baru apa pun yang diluncurkan mendukung referensi data sensitif dalam konfigurasi log wadah menggunakan parameter definisi `secretOptions` wadah. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah.	1 Mei 2019

Perubahan	Deskripsi	Tanggal
AWS Pembaruan platform Fargate versi 1.3.0	Dimulai pada 2 April 2019, tugas Fargate baru apa pun yang diluncurkan mendukung penyuntikan data sensitif ke dalam wadah Anda dengan menyimpan data sensitif Anda baik dalam AWS Secrets Manager rahasia atau AWS Systems Manager parameter Parameter Store dan kemudian merujuknya dalam definisi penampung Anda. Untuk informasi selengkap nya, lihat Meneruskan data sensitif ke wadah.	April 2, 2019
AWS Pembaruan platform Fargate versi 1.3.0	Mulai tanggal 27 Maret 2019, setiap tugas Fargate baru yang diluncurkan dapat menggunakan parameter definisi tugas tambahan yang memungkinkan Anda menentukan konfigurasi proxy, dependensi untuk startup dan shutdown kontainer serta nilai batas waktu mulai dan berhenti per kontainer. Lihat informasi selengkapnya di Konfigurasi proxy, Dependensi kontainer, dan Waktu habis kontainer.	27 Maret 2019
Amazon ECS memperken alkan tipe penerapan eksternal	Jenis penerapan eksternal memungkinkan Anda menggunakan pengontrol penyebaran pihak ketiga untuk kontrol penuh atas proses penyebaran untuk layanan Amazon ECS. Untuk informasi selengkap nya, lihat Menggunakan pengontrol pihak ketiga untuk penerapan eksternal Amazon ECS.	27 Maret 2019
AWS Deep Learning Containers di Amazon ECS	AWS Deep Learning Containers adalah sekumpulan gambar Docker untuk melatih dan menyajikan model TensorFlow di Amazon Elastic Container Service (Amazon ECS). Deep Learning Containers menyediak an lingkungan yang dioptimalkan dengan TensorFlo w library Nvidia CUDA (untuk instans GPU), dan Intel MKL (untuk instans CPU) dan tersedia di Amazon ECR. Untuk informasi selengkapnya, lihat AWS Deep Learning Containers di Amazon ECS.	27 Maret 2019

Perubahan	Deskripsi	Tanggal
Amazon ECS memperkenalkan manajemen ketergantungan kontainer yang disempurnakan	Amazon ECS memperkenalkan parameter definisi tugas tambahan yang memungkinkan Anda menentukan dependensi untuk startup dan shutdown container serta nilai batas waktu mulai dan berhenti per kontainer. Untuk informasi selengkapnya, lihat Dependensi kontainer.	7 Maret 2019
Amazon ECS memperkenalkan API `PutAccountSettingDefault`	Amazon ECS memperkenalkan `PutAccountSettingDefault` API yang memungkinkan pengguna untuk mengatur status opt-in format ARN/ID default untuk semua pengguna dan peran di akun. Sebelumnya, pengaturan status keikutsertaan default akun mengharuskan penggunaan pemilik akun. Untuk informasi selengkapnya, lihat Amazon Resource Name (ARN) dan ID.	8 Februari 2019
Amazon ECS mendukung beban kerja GPU	Amazon ECS memperkenalkan dukungan untuk beban kerja GPU dengan memungkinkan Anda membuat cluster dengan instans kontainer berkemampuan GPU. Dalam ketentuan tugas, Anda dapat menentukan jumlah GPU yang diperlukan dan agen ECS akan menyematkan GPU fisik ke dalam kontainer. Untuk informasi selengkapnya, lihat Bekerja dengan GPU di Amazon ECS.	4 Februari 2019
Amazon ECS memperluas dukungan rahasia	Amazon ECS memperluas dukungan untuk menggunakan AWS Secrets Manager rahasia secara langsung dalam definisi tugas Anda untuk menyuntikkan data sensitif ke dalam wadah Anda. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah.	21 Januari 2019

Perubahan	Deskripsi	Tanggal
VPC endpoint Antarmuka (AWS PrivateLink)	Menambahkan dukungan untuk mengonfigurasi VPC endpoint antarmuka yang didukung oleh AWS PrivateLink. Ini memungkinkan Anda untuk membuat koneksi pribadi antara VPC Anda dan Amazon ECS tanpa memerlukan akses melalui Internet, melalui instans NAT, koneksi VPN, atau. AWS Direct Connect Untuk informasi selengkapnya, lihat VPC Endpoint Antarmuka (AWS PrivateLink).	Desember 26, 2018
AWS Platform Fargate versi 1.3.0	Versi platform AWS Fargate baru dirilis, yang berisi: • Menambahkan dukungan untuk menggunakan AWS Systems Manager parameter Parameter Store untuk menyuntikkan data sensitif ke dalam wadah Anda. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah. • Menambahkan daur ulang tugas untuk tugas Fargate, yang merupakan proses menyegarkan tugas yang merupakan bagian dari layanan Amazon ECS. Untuk informasi selengkapnya, lihat Pemeliharaan tugas di Panduan Pengguna Amazon Elastic Container Service untuk AWS Fargate. Untuk informasi selengkapnya, lihat Versi platform Fargate Linux.	17 Desember 2018

Perubahan	Deskripsi	Tanggal
Kuota layanan diperbarui	Kuota layanan berikut diperbarui: • Jumlah klaster per Wilayah, per akun ditingkatkan dari `1000` ke `2000`. • Jumlah instans kontainer per klaster ditingkatkan dari `1000` ke `2000`. • Jumlah layanan per klaster ditingkatkan dari `500` ke `1000`. Untuk informasi selengkapnya, lihat Kuota layanan Amazon ECS.	14 Desember 2018
AWS Perluasan wilayah Fargate	AWS Fargate dengan Amazon ECS telah berkembang ke Wilayah Asia Pasifik (Mumbai) dan Kanada (Tengah). Untuk informasi selengkapnya, lihat Wilayah yang Didukung untuk Amazon ECS di Fargate AWS.	7 Desember 2018
Penerapan biru/hijau Amazon ECS	Amazon ECS menambahkan dukungan untuk penerapan biru/hijau menggunakan. CodeDeploy Tipe deployment ini mengizinkan Anda untuk memverifikasi deployment layanan baru sebelum mengirim lalu lintas produksi ke dalamnya. Untuk informasi selengkapnya, lihat Penerapan Biru/Hijau dengan CodeDeploy.	27 November 2018
Amazon ECS yang dioptimalkan Amazon Linux 2 (arm64) AMI dirilis	Amazon ECS merilis Amazon Amazon Linux 2 AMI yang dioptimalkan Amazon ECS untuk arsitektur arm64. Untuk informasi selengkapnya, lihat AMI Amazon ECS yang dioptimalkan.	26 November 2018

Perubahan	Deskripsi	Tanggal
Menambahkan dukungan untuk flag Docker tambahan dalam ketentuan tugas	Amazon ECS memperkenalkan dukungan untuk flag Docker berikut dalam definisi tugas: • Mode IPC • Mode PID	16 Novbucket 2018
Dukungan rahasia Amazon ECS	Amazon ECS menambahkan dukungan untuk menggunakan AWS Systems Manager parameter Parameter Store untuk menyuntikkan data sensitif ke dalam wadah Anda. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah.	15 November 2018
Penandaan sumber daya	Amazon ECS menambahkan dukungan untuk menambahkan tag metadata ke layanan, definisi tugas, tugas, kluster, dan instans penampung Anda. Untuk informasi selengkapnya, lihat Penandaan sumber daya Amazon ECS.	15 November 2018
AWS Perluasan Wilayah Fargate	AWS Fargate dengan Amazon ECS telah berkembang ke wilayah AS Barat (California N.) dan Asia Pasifik (Seoul). Untuk informasi selengkapnya, lihat Amazon ECS aktif AWS Fargate.	7 November 2018

Perubahan	Deskripsi	Tanggal
Kuota layanan diperbarui	Kuota layanan berikut diperbarui: • Jumlah tugas yang menggunakan jenis peluncuran Fargate, per Wilayah, per akun dinaikkan dari ke20. 50 • Jumlah alamat IP publik untuk tugas yang menggunakan tipe peluncuran Fargate dinaikkan dari ke20. 50 Untuk informasi selengkapnya, lihat Kuota layanan Amazon ECS.	31 Oktober 2018
AWS Perluasan Wilayah Fargate	AWS Fargate dengan Amazon ECS telah berkembang ke Wilayah Eropa (London). Untuk informasi selengkapnya, lihat Amazon ECS aktif AWS Fargate.	26 Oktober 2018
Amazon ECS yang Dioptimalkan Amazon Linux 2 AMI Dirilis	Amazon ECS menjual AMI Linux yang dioptimalkan untuk layanan dalam dua varian. Versi terbaru dan yang direkomendasikan didasarkan pada x;. Amazon ECS juga menjual AMI yang didasarkan pada, tetapi kami menyarankan Anda memigrasikan beban kerja Anda ke varian Amazon Linux 2, karena dukungan untuk Amazon Linux AMI akan berakhir selambat-lambatnya 30 Juni 2020. Untuk informasi selengkapnya, lihat AMI Amazon ECS yang dioptimalkan.	18 Oktober 2018

Perubahan	Deskripsi	Tanggal
Titik Akhir Metadata Tugas Amazon ECS versi 3	Dimulai dengan versi 1.21.0 dari agen kontainer Amazon ECS, agen menyuntikkan variabel lingkungan yang dipanggil `ECS_CONTAINER_METADATA_URI` ke setiap kontainer dalam tugas. Saat Anda menanyakan titik akhir metadata tugas versi 3, berbagai metadata tugas dan statistik Docker tersedia untuk tugas yang menggunakan mode `awsvpc` jaringan pada titik akhir HTTP yang disediakan oleh agen penampung Amazon ECS. Untuk informasi selengkapnya, lihat Ambil metadata Amazon ECS.	18 Oktober 2018
Penemuan layanan Amazon ECS Perluasan wilayah	Penemuan layanan Amazon ECS telah memperluas dukungan ke Kanada (Tengah), Amerika Selatan (São Paulo), Asia Pasifik (Seoul), Asia Pasifik (Mumbai), dan Wilayah Eropa (Paris). Untuk informasi selengkapnya, lihat Penemuan Layanan .	27 September 2018
Menambahkan dukungan untuk flag Docker tambahan dalam ketentuan kontainer	Amazon ECS memperkenalkan dukungan untuk flag Docker berikut dalam definisi container: • Kontrol sistem • Interaktif • Terminal semu	17 September 2018

Perubahan	Deskripsi	Tanggal
Dukungan otentikasi registri pribadi untuk Amazon ECS menggunakan tugas Fargate AWS	Amazon ECS memperkenalkan dukungan untuk tugas Fargate menggunakan otentikasi registri pribadi menggunakan. AWS Secrets Manager Fitur ini mengizinkan Anda untuk menyimpan kredensial Anda secara aman dan kemudian mereferensikan mereka dalam ketentuan kontainer Anda, yang mengizinkan tugas Anda untuk menggunakan citra privat. Untuk informasi selengkapnya, lihat Autentikasi registri privat untuk tugas.	10 September 2018
Penemuan layanan Amazon ECS Perluasan wilayah	Penemuan layanan Amazon ECS telah memperluas dukungan ke wilayah Asia Pasifik (Singapura), Asia Pasifik (Sydney), Asia Pasifik (Tokyo), UE (Frankfurt), dan Eropa (London). Untuk informasi selengkapnya, lihat Penemuan Layanan .	30 Agustus 2018
Tugas terjadwal dengan dukungan tugas Fargate	Amazon ECS memperkenalkan dukungan untuk tugas terjadwal untuk jenis peluncuran Fargate.	28 Agustus 2018
Otentikasi registri pribadi menggunakan dukungan AWS Secrets Manager	Amazon ECS memperkenalkan dukungan untuk otentikasi registri pribadi menggunakan. AWS Secrets Manager Fitur ini mengizinkan Anda untuk menyimpan kredensial Anda secara aman dan kemudian mereferensikan mereka dalam ketentuan kontainer Anda, yang mengizinkan tugas Anda untuk menggunakan citra privat. Untuk informasi selengkapnya, lihat Autentikasi registri privat untuk tugas.	16 Agustus 2018

Perubahan	Deskripsi	Tanggal
Dukungan volume docker ditambahkan	Amazon ECS memperkenalkan dukungan untuk volume Docker. Untuk informasi selengkapnya, lihat Menggunakan volume data dalam tugas.	9 Agustus 2018
AWS Perluasan Wilayah Fargate	AWS Fargate dengan Amazon ECS telah berkembang ke wilayah Eropa (Frankfurt), Asia Pasifik (Singapura), dan Asia Pasifik (Sydney). Untuk informasi selengkapnya, lihat Amazon ECS aktif AWS Fargate.	19 Juli 2018

Perubahan	Deskripsi	Tanggal
Strategi penjadwal layanan Amazon ECS ditambahkan	Amazon ECS memperkenalkan konsep strategi penjadwal layanan. Ada dua strategi penjadwal layanan yang tersedia: • `REPLICA`Strategi penjadwalan replika menempatkan dan mempertahankan jumlah tugas yang diinginkan di seluruh klaster Anda. Secara default tugas tersebar di seluruh Availability Zone. Anda dapat menggunakan strategi penempatan tugas dan kendala untuk menyesuaikan keputusan penempatan tugas. Untuk informasi selengkapnya, lihat Replika. • `DAEMON`—Strategi penjadwalan daemon menerapkan tepat satu tugas pada setiap instance kontainer aktif yang memenuhi semua batasan penempatan tugas yang Anda tentukan di cluster Anda. Saat menggunakan strategi ini, tidak perlu menentukan jumlah tugas yang diinginkan, strategi penempatan tugas, atau menggunakan kebijakan Auto Scaling Layanan. Untuk informasi selengkapnya, lihat Daemon. **Note** Tugas Fargate tidak mendukung strategi `DAEMON` penjadwalan. Untuk informasi selengkapnya, lihat Konsep penjadwal layanan.	Juni 12, 2018

Perubahan	Deskripsi	Tanggal
Agen kontainer Amazon ECS v1.18.0	Versi baru dari agen penampung Amazon ECS dirilis, yang menambahkan fungsionalitas berikut: • Menambahkan prosedur untuk menginstal agen kontainer secara manual dari URL S3 pada instans EC2 Linux non-Amazon, termasuk metode tanda tangan PGP untuk memverifikasi file instalasi agen penampung Amazon ECS. Untuk informasi selengkapnya, lihat Memasang agen kontainer Amazon ECS. • Menambahkan prosedur untuk menginstal agen kontainer secara manual dari URL S3 pada instans Windows EC2, termasuk metode tanda tangan PGP untuk memverifikasi file instalasi agen penampung Amazon ECS. • Menambahkan dukungan untuk menyesuaikan perilaku penarikan citra agen kontainer menggunak an parameter `ECS_IMAGE_PULL_BEHAVIOR` . Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS. Untuk informasi lebih lanjut, lihat amazon-ecs-agent github.	24 Mei 2018
Menambahkan Dukungan untuk `bridge` dan Mode `host` Jaringan Saat Mengkonfigurasi Penemuan Layanan	Menambahkan dukungan untuk mengonfigurasi penemuan layanan untuk layanan Amazon ECS menggunakan definisi tugas yang menentukan mode `bridge` atau `host` jaringan. Untuk informasi selengkapnya, lihat Penemuan Layanan .	22 Mei 2018

Perubahan	Deskripsi	Tanggal
Menambahkan dukungan untuk parameter metadata AMI tambahan yang dioptimalkan Amazon ECS	Menambahkan subparameter yang memungkinkan Anda mengambil ID AMI, nama gambar, sistem operasi, versi agen kontainer, dan versi runtime yang dioptimalkan Amazon ECS secara terprogram. Kueri metadata menggunakan Systems Manager Parameter Store API. Untuk informasi selengkapnya, lihat Mengambil metadata AMI Amazon ECS yang dioptimalkan.	9 Mei 2018
AWS Perluasan Wilayah Fargate	AWS Fargate dengan Amazon ECS telah berkembang ke Wilayah AS Timur (Ohio), AS Barat (Oregon), dan Uni Eropa Barat (Irlandia). Untuk informasi selengkapnya, lihat Amazon ECS aktif AWS Fargate.	26 April 2018
Pengambilan Metadata AMI yang dioptimalkan Amazon ECS	Menambahkan kemampuan untuk mengambil metadata AMI Amazon ECS yang dioptimalkan secara terprogram menggunakan Systems Manager Parameter Store API. Untuk informasi selengkapnya, lihat Mengambil metadata AMI Amazon ECS yang dioptimalkan.	10 April 2018
AWS Versi platform Fargate	Versi platform AWS Fargate baru dirilis, yang berisi: • Menambahkan dukungan untuk Ambil metadata Amazon ECS. • Menambahkan dukungan untuk Pemeriksaan kondisi. • Menambahkan dukungan untuk Penemuan Layanan Untuk informasi selengkapnya, lihat Versi platform Fargate Linux.	Maret 26, 2018

Perubahan	Deskripsi	Tanggal
Penemuan Layanan Amazon ECS	Menambahkan integrasi dengan Route 53 untuk mendukung penemuan layanan Amazon ECS. Untuk informasi selengkapnya, lihat Penemuan Layanan .	22 Maret 2018
Dukungan shm-size dan tmpfs Docker	Menambahkan dukungan untuk parameter shm-size dan tmpfs Docker dalam definisi tugas Amazon ECS. Untuk informasi selengkapnya tentang sintaksis CLI ECS yang diperbarui, lihat Parameter Linux.	20 Maret 2018
Pemeriksaan Kondisi Kontainer	Menambahkan dukungan untuk pemeriksaan kondisi Docker dalam ketentuan kontainer. Untuk informasi selengkapnya, lihat Pemeriksaan kondisi.	8 Maret 2018
AWS Fargate	Ditambahkan ikhtisar untuk Amazon ECS dengan AWS Fargate. Untuk informasi selengkapnya, lihat Amazon ECS aktif AWS Fargate.	22 Februari 2018
Titik Akhir Metadata Tugas Amazon ECS	Dimulai dengan versi 1.17.0 dari agen penampung Amazon ECS, berbagai metadata tugas dan statistik Docker tersedia untuk tugas yang menggunakan mode `awsvpc` jaringan pada titik akhir HTTP yang disediakan oleh agen penampung Amazon ECS. Untuk informasi selengkapnya, lihat Ambil metadata Amazon ECS.	8 Februari 2018
Auto Scaling Amazon ECS Service menggunakan kebijakan pelacakan target	Menambahkan dukungan untuk Auto Scaling Layanan ECS menggunakan kebijakan pelacakan target di konsol Amazon ECS. Untuk informasi selengkapnya, lihat Skala layanan Amazon ECS Anda menggunakan nilai metrik target. Menghapus tutorial sebelumnya untuk penskalaan langkah di dalam ECS yang pertama kali menjalankan wizard. Tutorial ini diganti dengan tutorial baru untuk pelacakan target.	8 Februari 2018

Perubahan	Deskripsi	Tanggal
Dukungan Docker versi 17.09	Menambahkan dukungan untuk Docker 17.09. Untuk informasi selengkapnya, lihat AMI Amazon ECS yang dioptimalkan.	18 Januari 2018
Perilaku penjadwal layanan baru	Informasi yang diperbarui tentang perilaku untuk tugas layanan yang gagal diluncurkan. Pesan kejadian layanan baru yang terdokumentasi yang terpicu saat tugas layanan mengalami kegagalan berturut-turut. Untuk informasi selengkapnya tentang perilaku yang diperbarui ini, lihat Konsep layanan tambahan.	11 Januari 2018
Periode tunggu inisialisasi pemeriksaan kesehatan Elastic Load Balancing	Menambahkan kemampuan untuk menentukan masa tunggu pemeriksaan kondisi.	Desember 27, 2017
CPU dan memori tingkat tugas	Menambahkan dukungan untuk menentukan CPU dan memori pada tingkat tugas dalam ketentuan tugas. Untuk informasi lebih lanjut, lihat TaskDefinition.	12 Desember 2017
Peran pelaksanaan tugas	Agen penampung Amazon ECS melakukan panggilan ke tindakan Amazon ECS API atas nama Anda, sehingga memerlukan kebijakan dan peran IAM agar layanan mengetahui bahwa agen tersebut milik Anda. Tindakan berikut termasuk peran eksekusi tugas: • Panggilan ke Amazon ECR untuk menarik gambar kontainer • Panggilan CloudWatch untuk menyimpan log aplikasi kontainer Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.	7 Desember 2017

Perubahan	Deskripsi	Tanggal
Kontainer Windows mendukung umumnya tersedia (GA)	Menambahkan dukungan untuk wadah Windows Server 2016. Untuk informasi selengkapnya, lihat Varian AMI yang dioptimalkan Amazon ECS.	5 Desember 2017
AWS Fargate GA	Menambahkan dukungan untuk meluncurkan layanan Amazon ECS menggunakan jenis peluncuran Fargate. Untuk informasi selengkapnya, lihat Jenis peluncuran Amazon ECS.	29 November 2017
Perubahan nama Amazon ECS	Amazon Elastic Container Service diganti namanya (sebelumnya Amazon EC2 Container Service).	21 November 2017
Jaringan tugas	Fitur jaringan tugas yang disediakan oleh mode jaringan awsvpc memberikan tugas Amazon ECS properti jaringan yang sama dengan instans Amazon EC2. Saat Anda menggunakan mode jaringan awsvpc dalam ketentuan tugas, setiap tugas yang diluncurkan dari ketentuan tugas tersebut mendapatkan antarmuka jaringan elastis, alamat IP privat primer, dan nama host DNS internalnya sendiri. Fitur jaringan tugas menyederhanakan jaringan kontainer dan memberi Anda lebih banyak kontrol atas bagaimana aplikasi terkontainer berkomunikasi satu sama lain dan dengan layanan lain dalam VPC Anda. Untuk informasi selengkapnya, lihat Jaringan tugas untuk tugas di instans Amazon EC2.	14 November 2017
Metadata wadah Amazon ECS	Container Amazon ECS sekarang dapat mengakses metadata seperti wadah Docker atau ID gambar, konfigurasi jaringan, atau Amazon ARN. Untuk informasi selengkapnya, lihat File metadata wadah Amazon ECS.	2 November 2017

Perubahan	Deskripsi	Tanggal
Dukungan Docker versi 17.06	Menambahkan dukungan untuk Docker versi 17.06. Untuk informasi selengkapnya, lihat AMI Amazon ECS yang dioptimalkan.	2 November 2017
Dukungan untuk flag Docker: perangkat dan init	Menambahkan dukungan untuk perangkat Docker dan fitur init dalam ketentuan tugas menggunakan parameter `LinuxParameters` (`devices` dan `initProcessEnabled`). Untuk informasi lebih lanjut, lihat LinuxParameters.	2 November 2017
Dukungan untuk flag Docker: cap-add dan cap-drop	Menambahkan dukungan untuk fitur cap-add dan cap-drop Docker dalam ketentuan tugas menggunakan parameter`LinuxParameters` (`capabilities`). Untuk informasi lebih lanjut, lihat LinuxParameters.	22 September 2017
Dukungan Penyeimbang Beban Jaringan	Amazon ECS menambahkan dukungan untuk Network Load Balancers di konsol Amazon ECS.	7 September 2017
RunTask mengesampingkan	Menambahkan dukungan untuk penggantian ketentuan tugas saat menjalankan tugas. Hal ini mengizinkan Anda untuk menjalankan suatu tugas bersamaan dengan mengubah ketentuan tugas tanpa perlu membuat revisi ketentuan tugas yang baru. Untuk informasi selengkapnya, lihat Jalankan aplikasi sebagai tugas Amazon ECS.	27 Juni 2017
Tugas terjadwal Amazon ECS	Menambahkan dukungan untuk tugas penjadwalan menggunakan cron.	7 Juni 2017
Spot Instans di konsol Amazon ECS	Menambahkan dukungan untuk membuat instans kontainer Spot Fleet dalam konsol Amazon ECS. Untuk informasi selengkapnya, lihat Meluncurkan instans penampung Amazon ECS Linux.	6 Juni 2017

Perubahan	Deskripsi	Tanggal
Pemberitahuan Amazon SNS untuk rilis AMI baru yang dioptimalkan Amazon ECS	Menambahkan kemampuan untuk berlangganan pemberitahuan SNS tentang rilis AMI baru yang dioptimalkan Amazon ECS.	Kamis, 23 Maret 2017
Tugas layanan mikro dan batch	Menambahkan dokumentasi untuk dua kasus penggunaan umum untuk Amazon ECS: layanan mikro dan pekerjaan batch. Untuk informasi selengkapnya, lihat Informasi terkait.	Februari 2017
Pengosongan instans kontainer	Menambahkan dukungan untuk pengosongan instans kontainer, yang menyediakan metode untuk menghapus instans kontainer dari klaster. Untuk informasi selengkapnya, lihat Pengurasan instans kontainer.	24 Januari 2017
Dukungan Docker 1.12	Menambahkan dukungan untuk Docker 1.12. Untuk informasi selengkapnya, lihat AMI Amazon ECS yang dioptimalkan.	24 Januari 2017
Strategi penempatan tugas baru	Menambahkan dukungan untuk strategi penempata n tugas: penempatan berbasis atribut, paket bin, penyebaran Availability Zone, dan satu per host. Untuk informasi selengkapnya, lihat Gunakan strategi untuk menentukan penempatan tugas Amazon ECS.	Desember 29, 2016
Dukungan kontainer Windows dalam versi beta	Menambahkan dukungan untuk wadah Windows Server 2016 (beta). Untuk informasi selengkapnya, lihat Varian AMI yang dioptimalkan Amazon ECS.	20 Desember 2016
Dukungan Blox OSS	Menambahkan dukungan untuk Blox OSS, yang mengizinkan penjadwal tugas kustom. Untuk informasi selengkapnya, lihat Menjadwalkan kontainer Anda di Amazon ECS.	1 Desember 2016

Perubahan	Deskripsi	Tanggal
Aliran acara Amazon ECS untuk CloudWatch Acara	Amazon ECS sekarang mengirimkan instance kontainer dan perubahan status tugas ke CloudWatch Acara. Untuk informasi selengkapnya, lihat Otomatisk an tanggapan terhadap kesalahan Amazon ECS menggunakan EventBridge.	21 November 2016
Pencatatan kontainer Amazon ECS ke CloudWatch Log	Menambahkan dukungan untuk driver awslogs untuk mengirim aliran log kontainer ke Log. CloudWatch Untuk informasi selengkapnya, lihat Menggunakan driver log awslogs.	September 12, 2016
Layanan Amazon ECS dengan dukungan Elastic Load Balancing untuk port dinamis	Menambahkan dukungan untuk penyeimbang beban agar dapat mendukung beberapa instans, yaitu: kombinasi port per listener, yang dapat meningkat kan fleksibilitas untuk kontainer. Kini Anda dapat secara dinamis membiarkan Docker menentukan port host kontainer dan penjadwal ECS yang mendaftar kan instans:port dengan penyeimbang beban. Untuk informasi selengkapnya, lihat Mendistribusikan lalu lintas layanan Amazon ECS menggunakan load balancing.	11 Agustus, 2016
Peran IAM untuk tugas Amazon ECS	Menambahkan dukungan untuk mengaitkan peran IAM dengan tugas. Ini memberikan izin yang detail ke kontainer kebalikan dari peran tunggal untuk seluruh instans kontainer. Untuk informasi selengkapnya, lihat Tugas peran IAM.	13 Juli 2016
Dukungan Docker 1.11	Menambahkan dukungan untuk Docker 1.11. Untuk informasi selengkapnya, lihat AMI Amazon ECS yang dioptimalkan.	31 Mei 2016

Perubahan	Deskripsi	Tanggal
Penskalaan otomatis tugas	Amazon ECS menambahkan dukungan untuk secara otomatis menskalakan tugas Anda yang dijalankan oleh layanan. Untuk informasi selengkapnya, lihat Secara otomatis menskalakan layanan Amazon ECS Anda.	18 Mei 2016
Pemfilteran ketentuan tugas pada famili tugas	Menambahkan dukungan untuk memfilter daftar ketentuan tugas berdasarkan famili ketentuan tugas. Untuk informasi lebih lanjut, lihat ListTaskDefinitions.	17 Mei 2016
Kontainer Docker dan pencatatan agen Amazon ECS	Amazon ECS menambahkan kemampuan untuk mengirim agen ECS dan log kontainer Docker dari instance kontainer ke CloudWatch Log untuk menyederhanakan masalah pemecahan masalah.	Mei 5, 2016
AMI yang dioptimalkan ECS sekarang mendukung Amazon Linux 2016.03.	AMI yang dioptimalkan ECS menambahkan dukungan untuk Amazon Linux 2016.03. Untuk informasi selengkapnya, lihat AMI Amazon ECS yang dioptimalkan.	5 April 2016
Dukungan Docker 1.9	Menambahkan dukungan untuk Docker versi 1.9. Untuk informasi selengkapnya, lihat AMI Amazon ECS yang dioptimalkan.	Desember 22, 2015
CloudWatch metrik untuk CPU cluster dan reservasi memori	Amazon ECS menambahkan CloudWatch metrik khusus untuk reservasi CPU dan memori.	Desember 22, 2015
Pengalaman Amazon ECS baru yang dijalankan pertama	Pengalaman pertama kali dijalankan konsol Amazon ECS menambahkan pembuatan peran nol-klik.	November 23, 2015
Penempatan tugas di seluruh Availability Zone	Penjadwal layanan Amazon ECS menambahkan dukungan untuk penempatan tugas di seluruh Availability Zone.	8 Oktober 2015

Perubahan	Deskripsi	Tanggal
CloudWatch metrik untuk kluster dan layanan Amazon ECS	Amazon ECS menambahkan CloudWatch metrik khusus untuk pemanfaatan CPU dan memori untuk setiap instance kontainer, layanan, dan keluarga definisi tugas dalam sebuah cluster. Metrik baru ini dapat digunakan untuk menskalakan instance kontainer di klaster menggunakan grup Auto Scaling atau untuk membuat alarm khusus. CloudWatch	17 Agustus 2015
Dukungan port UDP	Menambahkan dukungan untuk port UDP dalam ketentuan tugas.	Juli 7, 2015
Penggantian variabel lingkungan	Menambahkan dukungan untuk deregisterTaskDefinition dan penggantian variabel lingkungan untuk RunTask.	18 Juni 2015
Pembaruan agen Amazon ECS otomatis	Menambahkan kemampuan untuk melihat versi agen ECS yang berjalan pada instans kontainer. Juga dapat memperbarui agen ECS dari AWS Management Console, AWS CLI, dan SDK.	11 Juni 2015
Penjadwal layanan Amazon ECS dan integrasi Elastic Load Balancing	Menambahkan kemampuan untuk menentukan layanan dan mengaitkan layanan tersebut dengan penyeimbang beban Elastic Load Balancing.	9 April 2015
Amazon ECS GA	Ketersediaan umum Amazon ECS di Wilayah AS Timur (Virginia N.), AS Barat (Oregon), Asia Pasifik (Tokyo), dan Eropa (Irlandia).	9 April 2015

Daftar istilah AWS

Untuk terminologi AWS terbaru, lihat Daftar istilah AWS di Referensi Glosarium AWS.

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