aws

Panduan Praktik Terbaik

Amazon Elastic Container Service

Amazon Elastic Container Service: Panduan Praktik Terbaik

Table of Contents

Pengantar

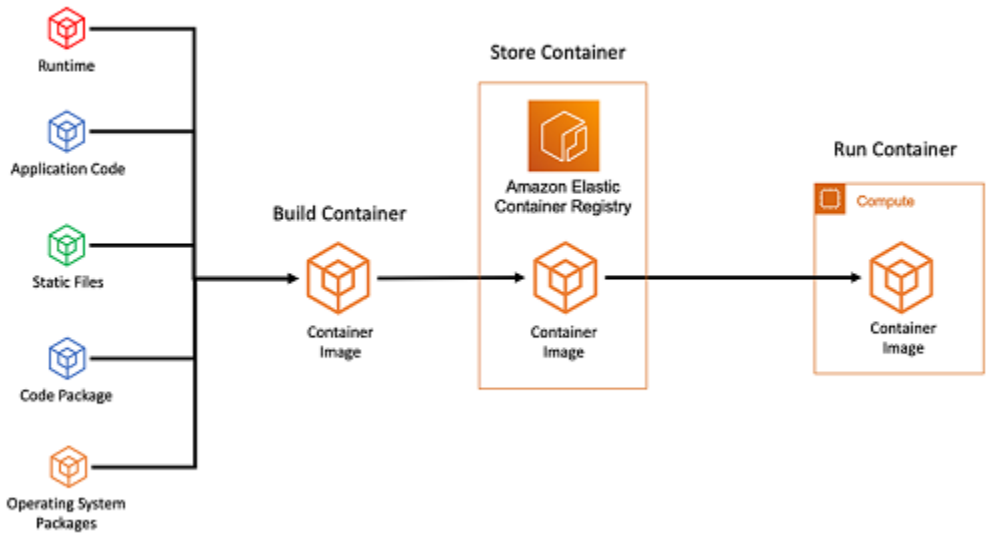
Amazon Elastic Container Service (Amazon ECS) adalah layanan manajemen kontainer yang sangat skalabel dan cepat yang dapat Anda gunakan untuk mengelola kontainer di klaster. Panduan ini mencakup banyak praktik terbaik operasional terpenting untuk Amazon ECS. Ini juga menjelaskan beberapa konsep inti yang terlibat dalam cara kerja aplikasi berbasis Amazon ECS. Tujuannya adalah untuk memberikan pendekatan konkret dan dapat ditindaklanjuti untuk mengoperasikan dan memecahkan masalah aplikasi berbasis Amazon ECS.

Praktik Terbaik - Menjalankan aplikasi Anda dengan Amazon ECS

Sebelum Anda menjalankan aplikasi menggunakan Amazon Elastic Container Service, pastikan Anda memahami cara kerja berbagai aspek aplikasi Anda dengan fitur-fitur di Amazon ECS. Panduan ini mencakup jenis sumber daya Amazon ECS utama, penggunaannya, dan praktik terbaik untuk menggunakan masing-masing jenis sumber daya ini.

Gambar kontainer

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.



Gambar kontainer melalui proses tiga langkah.

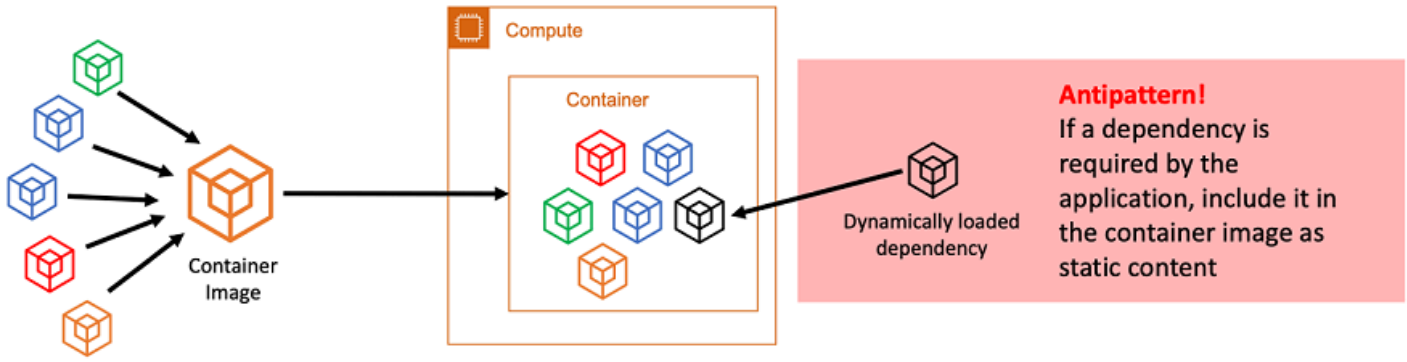
1. Build - Kumpulkan aplikasi Anda dan semua dependensinya menjadi satu image container.
2. Simpan - Unggah gambar kontainer ke registri kontainer.
3. Jalankan - Unduh gambar kontainer pada beberapa komputasi, buka kemasannya, dan mulai aplikasi.

Saat Anda membuat gambar kontainer Anda sendiri, ingatlah praktik terbaik yang dijelaskan di bagian berikut.

Membuat gambar kontainer lengkap dan statis

Idealnya, gambar kontainer dimaksudkan untuk menjadi snapshot lengkap dari semua yang dibutuhkan aplikasi untuk berfungsi. Dengan gambar kontainer yang lengkap, Anda dapat menjalankan aplikasi dengan mengunduh satu gambar kontainer dari satu tempat. Anda tidak perlu mengunduh beberapa bagian terpisah dari lokasi yang berbeda. Oleh karena itu, sebagai praktik terbaik, simpan semua dependensi aplikasi sebagai file statis di dalam image container.

Pada saat yang sama, jangan mengunduh pustaka, dependensi, atau data penting secara dinamis selama startup aplikasi. Sebagai gantinya, sertakan hal-hal ini sebagai file statis dalam gambar kontainer. Nanti, jika Anda ingin mengubah sesuatu di image container, buat image container baru dengan perubahan yang diterapkan padanya.



Ada beberapa alasan mengapa kami merekomendasikan praktik terbaik ini.

- Menyertakan semua dependensi sebagai file statis dalam gambar kontainer mengurangi jumlah peristiwa yang berpotensi merusak yang dapat terjadi selama penerapan. Saat Anda menskalakan hingga puluhan, ratusan, atau bahkan ribuan salinan wadah Anda, mengunduh satu gambar kontainer daripada mengunduh dari dua atau tiga tempat berbeda menyederhanakan beban kerja Anda dengan membatasi potensi titik putus. Misalnya, asumsikan bahwa Anda menggunakan 100 salinan aplikasi Anda, dan setiap salinan aplikasi harus mengunduh potongan dari tiga sumber yang berbeda. Ada 300 unduhan yang bisa gagal. Jika Anda mengunduh gambar kontainer, hanya ada 100 dependensi yang dapat rusak.
- Unduhan gambar kontainer dioptimalkan untuk mengunduh dependensi aplikasi secara paralel. Secara default, gambar kontainer terdiri dari lapisan yang dapat diunduh dan dibongkar secara paralel. Ini berarti bahwa unduhan gambar kontainer bisa mendapatkan semua dependensi Anda

ke komputasi Anda lebih cepat daripada skrip kode tangan yang mengunduh setiap dependensi dalam satu seri.

- Dengan menyimpan semua dependensi Anda di dalam gambar, penerapan Anda lebih andal dan dapat direproduksi. Jika Anda mengubah dependensi yang dimuat secara dinamis, itu mungkin merusak aplikasi di dalam image container. Namun, jika wadah benar-benar mandiri, Anda selalu dapat menerapkannya kembali, bahkan di masa depan. Ini karena sudah memiliki versi yang tepat dan dependensi yang tepat di dalamnya.

Pertahankan waktu peluncuran kontainer yang cepat dengan menjaga gambar kontainer sekecil mungkin

Container lengkap menyimpan semua yang diperlukan untuk menjalankan aplikasi Anda, tetapi mereka tidak perlu menyertakan alat build Anda. Perhatikan contoh ini. Asumsikan bahwa Anda sedang membangun sebuah wadah untuk Node.js aplikasi. Anda harus memiliki manajer NPM paket untuk mengunduh paket untuk aplikasi Anda. Namun, Anda tidak lagi membutuhkan NPM dirinya sendiri saat aplikasi berjalan. Anda dapat menggunakan multistage Docker build untuk menyelesaikan ini.

Berikut ini adalah contoh seperti apa multistage `Dockerfile` seperti itu untuk Node.js aplikasi yang memiliki dependensi. NPM

```
FROM node:14 AS build
WORKDIR /srv
ADD package.json .
RUN npm install

FROM node:14-slim
COPY --from=build /srv .
ADD . .
EXPOSE 3000
CMD ["node", "index.js"]
```

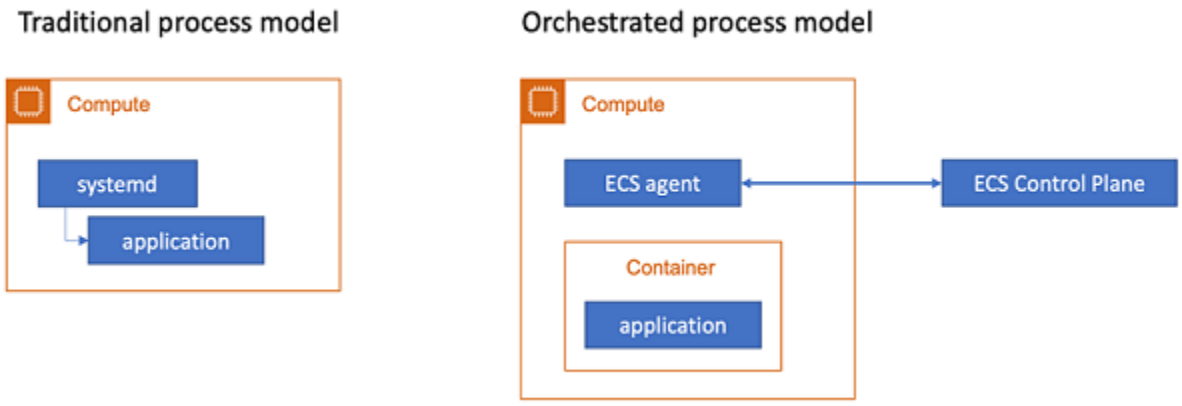
Tahap pertama menggunakan Node.js lingkungan penuh yang memilikiNPM, dan kompiler untuk membangun binding kode asli untuk paket. Tahap kedua tidak mencakup apa pun selain Node.js runtime. Itu dapat menyalin NPM paket yang diunduh dari tahap pertama. Produk akhir adalah gambar minimal yang memiliki Node.js runtime, NPM paket, dan kode aplikasi. Itu tidak termasuk toolchain NPM build lengkap.

Simpan gambar kontainer Anda sekecil mungkin dan gunakan lapisan bersama. Misalnya, jika Anda memiliki beberapa aplikasi yang menggunakan kumpulan data yang sama, Anda dapat membuat gambar dasar bersama yang memiliki kumpulan data tersebut. Kemudian, buat dua varian gambar berbeda dari gambar dasar bersama yang sama. Ini memungkinkan lapisan gambar kontainer dengan kumpulan data diunduh satu kali, bukan dua kali.

Manfaat utama menggunakan gambar kontainer yang lebih kecil adalah gambar-gambar ini dapat diunduh ke perangkat keras komputasi lebih cepat. Ini memungkinkan aplikasi Anda untuk skala lebih cepat dan cepat pulih dari crash atau restart yang tidak terduga.

Jalankan hanya satu proses aplikasi dengan gambar kontainer

Dalam lingkungan mesin virtual tradisional, biasanya menjalankan daemon tingkat tinggi `systemd` seperti proses root. Daemon ini kemudian bertanggung jawab untuk memulai proses aplikasi Anda, dan memulai kembali proses aplikasi jika macet. Kami tidak merekomendasikan ini saat menggunakan wadah. Sebaliknya, jalankan hanya satu proses aplikasi dengan wadah.



Jika proses aplikasi macet atau berakhir, wadah juga berakhir. Jika aplikasi harus dimulai ulang saat crash, biarkan Amazon ECS mengelola aplikasi restart secara eksternal. Agen Amazon ECS melaporkan ke pesawat kontrol Amazon ECS bahwa wadah aplikasi jatuh. Kemudian, bidang kontrol menentukan apakah akan meluncurkan wadah pengganti, dan jika demikian di mana harus meluncurkannya. Wadah pengganti dapat ditempatkan ke host yang sama, atau ke host yang berbeda di cluster.

Perlakukan wadah sebagai sumber daya fana. Mereka hanya bertahan seumur hidup dari proses aplikasi utama. Jangan terus memulai ulang proses aplikasi di dalam wadah, untuk mencoba menjaga wadah tetap aktif dan berjalan. Biarkan Amazon ECS mengganti kontainer sesuai kebutuhan.

Praktik terbaik ini memiliki dua manfaat utama.

- Ini mengurangi skenario di mana aplikasi mogok karena mutasi ke sistem file kontainer lokal. Alih-alih menggunakan kembali lingkungan kontainer bermutasi yang sama, orkestrator meluncurkan wadah baru berdasarkan gambar kontainer asli. Ini berarti Anda dapat yakin bahwa wadah pengganti menjalankan lingkungan dasar yang bersih.
- Proses crash diganti melalui proses pengambilan keputusan terpusat di bidang kontrol Amazon ECS. Pesawat kontrol membuat pilihan yang lebih cerdas tentang di mana harus meluncurkan proses penggantian. Misalnya, bidang kontrol dapat mencoba meluncurkan penggantian ke perangkat keras yang berbeda di Availability Zone yang berbeda. Ini membuat penerapan keseluruhan lebih tangguh daripada jika setiap instance komputasi individu mencoba meluncurkan kembali prosesnya sendiri secara lokal.

Tangani **`SIGTERM`** dalam aplikasi

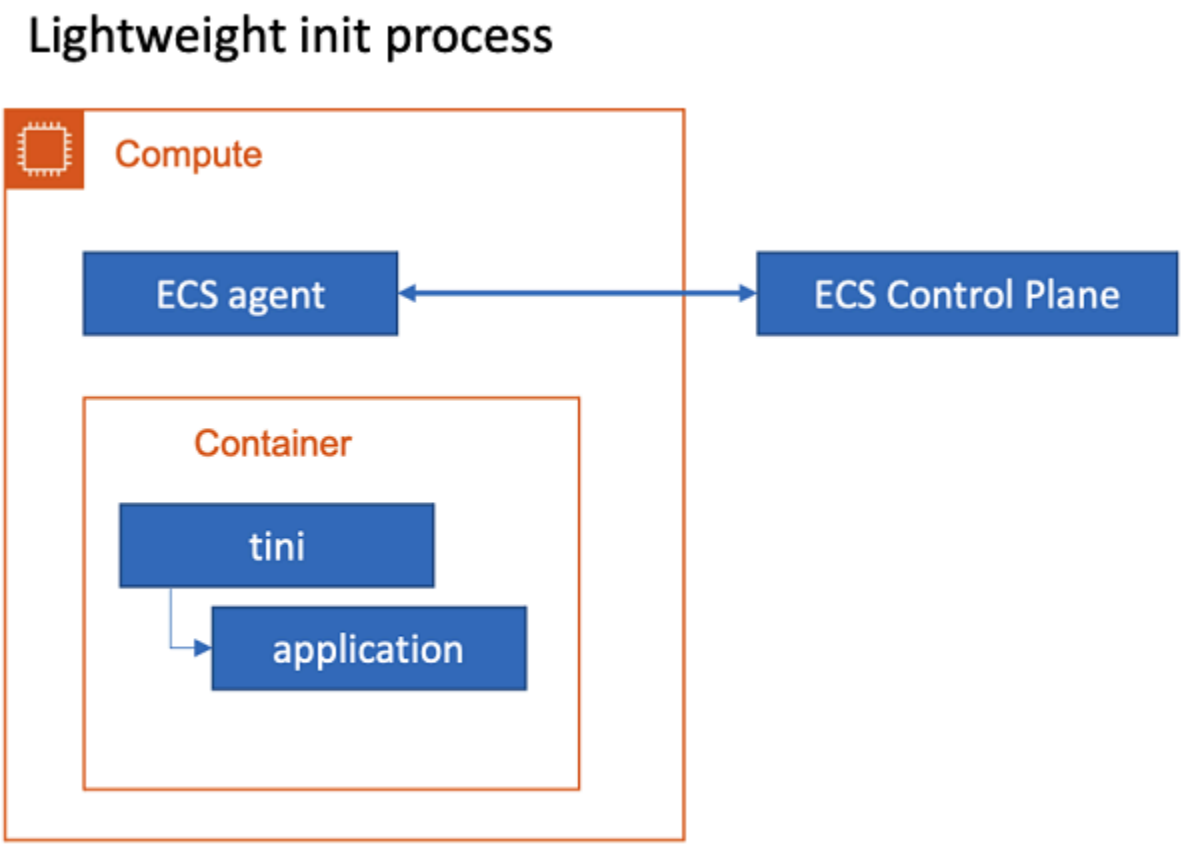
Saat Anda mengikuti panduan dari bagian sebelumnya, Anda mengizinkan Amazon ECS untuk mengganti tugas di tempat lain di cluster, daripada memulai ulang aplikasi yang mogok. Ada saat-saat lain ketika tugas dapat dihentikan yang berada di luar kendali aplikasi. Tugas dapat dihentikan karena kesalahan aplikasi, kegagalan pemeriksaan kesehatan, penyelesaian alur kerja bisnis atau bahkan penghentian manual oleh pengguna.

Ketika tugas dihentikan oleh ECS, ECS mengikuti langkah-langkah dan konfigurasi yang ditunjukkan dalam. Responsif SIGTERM

Untuk mempersiapkan aplikasi Anda, Anda perlu mengidentifikasi berapa lama waktu yang dibutuhkan aplikasi Anda untuk menyelesaikan pekerjaannya, dan memastikan bahwa aplikasi Anda menangani `SIGTERM` sinyal. Dalam penanganan sinyal aplikasi, Anda perlu menghentikan aplikasi dari mengambil pekerjaan baru dan menyelesaikan pekerjaan yang sedang berlangsung, atau menyimpan pekerjaan yang belum selesai ke penyimpanan di luar tugas jika terlalu lama untuk diselesaikan.

Setelah mengirim `SIGTERM` sinyal, Amazon ECS akan menunggu waktu yang ditentukan `StopTimeout` dalam definisi tugas. Kemudian, `SIGKILL` sinyal akan dikirim. Atur cukup `StopTimeout` lama sehingga aplikasi Anda menyelesaikan `SIGTERM` handler dalam semua situasi sebelum `SIGKILL` dikirim.

Untuk aplikasi web, Anda juga perlu mempertimbangkan koneksi terbuka yang menganggur. Lihat halaman berikut dari panduan ini untuk lebih jelasnyaNetwork Load Balancer.



Jika Anda menggunakan proses init dalam wadah Anda, gunakan proses init ringan seperti`tini`. Proses init ini bertanggung jawab untuk menuai proses zombie jika aplikasi Anda memunculkan proses pekerja. Jika aplikasi Anda tidak menangani `SIGTERM` sinyal dengan benar, `tini` dapat menangkap sinyal itu untuk Anda dan menghentikan proses aplikasi Anda. Namun, jika proses aplikasi Anda mogok `tini` tidak memulai ulang. Alih-alih `tini` keluar, memungkinkan wadah berakhir dan diganti dengan orkestrasi kontainer. Untuk informasi lebih lanjut, lihat `tini`di GitHub.

Konfigurasikan aplikasi kontainer untuk menulis log ke dan **`stdoutstderr`**

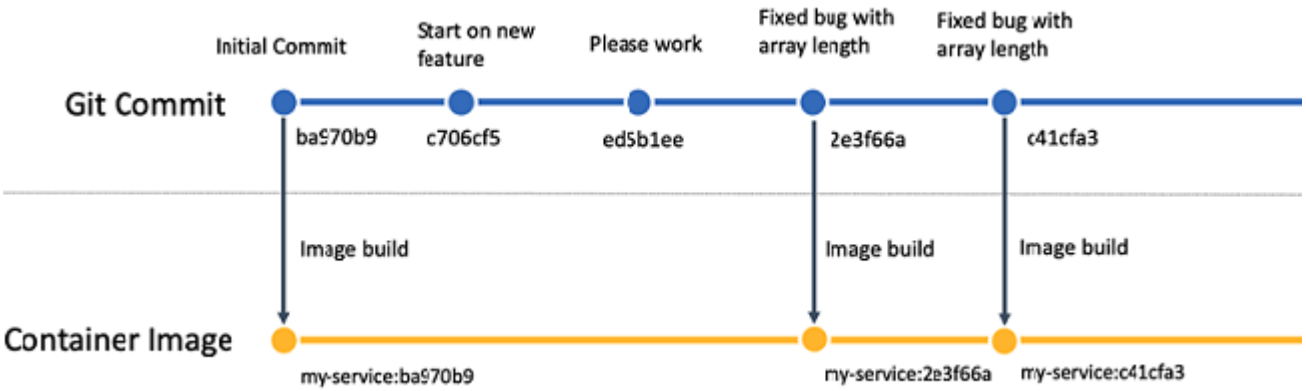
Ada banyak cara berbeda untuk melakukan logging. Untuk beberapa kerangka kerja aplikasi, biasanya menggunakan pustaka logging aplikasi yang menulis langsung ke file disk. Ini juga umum untuk menggunakan salah satu yang mengalirkan log langsung ke tumpukan ELK (OpenSearchLogstash,,Kibana) atau pengaturan logging serupa. Namun, kami menyarankan bahwa, ketika aplikasi dalam kontainerisasi, Anda mengonfigurasinya untuk menulis log aplikasi langsung ke `stdout` dan `stderr` streaming.

Dockermencakup berbagai driver logging yang mengambil `stdout` dan `stderr` mencatat aliran dan menanganinya. Anda dapat memilih untuk menulis stream ke`syslog`, ke disk pada instance lokal yang menjalankan container, atau menggunakan driver logging untuk mengirim log keFluentd,, Splunk CloudWatch, dan tujuan lainnya. Dengan Amazon ECS, Anda dapat memilih untuk mengonfigurasi driver FireLens logging. Driver ini dapat melampirkan metadata Amazon ECS ke log, memfilter log, dan merutekan log ke tujuan yang berbeda berdasarkan kriteria seperti kode status HTTP. Untuk informasi selengkapnya tentang driver Docker logging, lihat Mengkonfigurasi driver logging. Untuk informasi selengkapnyaFireLens, lihat Menggunakan FireLens.

Ketika Anda memisahkan penanganan log dari kode aplikasi Anda, ini memberi Anda fleksibilitas yang lebih besar untuk menyesuaikan penanganan log di tingkat infrastruktur. Asumsikan bahwa Anda ingin beralih dari satu sistem logging ke yang lain. Anda dapat melakukannya dengan menyesuaikan beberapa pengaturan di tingkat orkestrator kontainer, daripada harus mengubah kode di semua layanan Anda, membuat gambar kontainer baru, dan menerapkannya.

Gambar kontainer versi menggunakan tag

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.

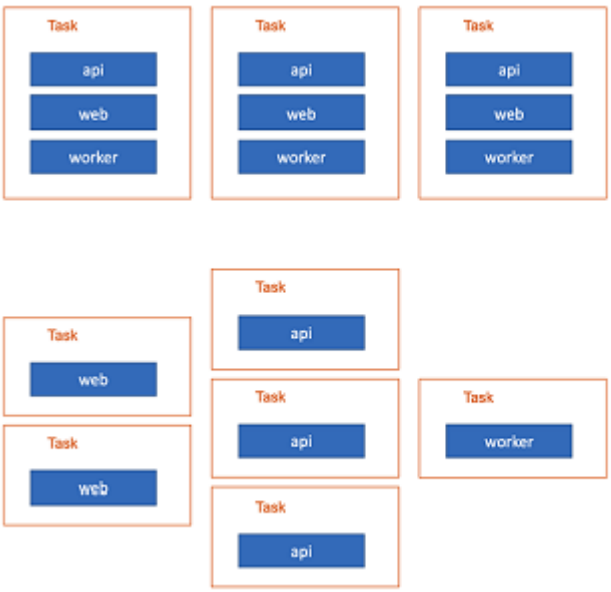
Kami juga menyarankan Anda mengaktifkan tag gambar yang tidak dapat diubah di Amazon Elastic Container Registry. Dengan pengaturan ini, Anda tidak dapat mengubah gambar kontainer yang ditunjuk tag. Sebaliknya Amazon ECR memberlakukan bahwa gambar baru harus diunggah ke tag baru, daripada menimpa tag yang sudah ada sebelumnya. Untuk informasi selengkapnya, lihat Mutabilitas tag gambar di Panduan Pengguna Amazon ECR.

Ketentuan tugas

Definisi tugas adalah dokumen yang menjelaskan gambar kontainer apa yang akan dijalankan bersama, dan pengaturan apa yang akan digunakan saat menjalankan gambar kontainer. Pengaturan ini mencakup jumlah CPU dan memori yang dibutuhkan wadah. Mereka juga menyertakan variabel lingkungan apa pun yang dipasok ke wadah dan volume data apa pun yang dipasang ke wadah. Definisi tugas dikelompokkan berdasarkan dimensi keluarga dan revisi.

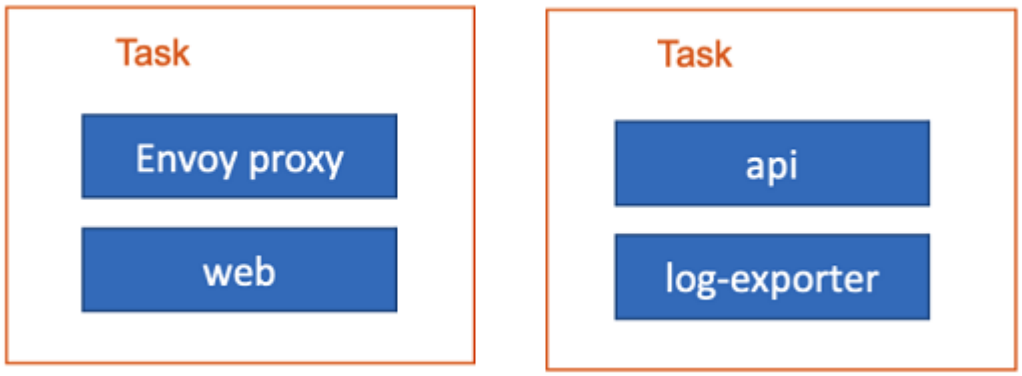
Gunakan setiap keluarga definisi tugas hanya untuk satu tujuan bisnis

Anda dapat menggunakan definisi tugas Amazon ECS untuk menentukan beberapa kontainer. Semua kontainer yang Anda tentukan disebarkan sepanjang kapasitas komputasi yang sama. Jangan gunakan fitur ini untuk menambahkan beberapa wadah aplikasi ke definisi tugas yang sama karena ini mencegah salinan dari setiap penskalaan aplikasi secara terpisah. Misalnya, pertimbangkan situasi ini. Misalnya Anda memiliki container server web, container API, dan container service worker. Sebagai praktik terbaik, gunakan keluarga definisi tugas terpisah untuk masing-masing potongan kode kontainer ini.



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.

Kami menyarankan Anda menskalakan setiap jenis kontainer secara independen berdasarkan permintaan.



Kami tidak menyarankan Anda menggunakan beberapa kontainer dalam satu definisi tugas untuk mengelompokkan berbagai jenis wadah aplikasi. Tujuan memiliki beberapa kontainer dalam satu definisi tugas adalah agar Anda dapat menerapkan sespan, wadah addon kecil yang meningkatkan satu jenis wadah. Sidecar dapat membantu pencatatan dan pengamatan, perutean lalu lintas, atau fitur addon lainnya.

Kami menyarankan Anda menggunakan sidecars untuk melampirkan fungsionalitas tambahan, tetapi tugas tersebut memiliki fungsi bisnis tunggal.

Cocokkan setiap versi aplikasi dengan revisi definisi tugas dalam keluarga definisi tugas

Definisi tugas dapat dikonfigurasi untuk menunjuk pada tag gambar kontainer apa pun, termasuk tag “terbaru”. Namun, kami tidak menyarankan Anda menggunakan tag “terbaru” dalam definisi tugas Anda. Ini karena tag “terbaru” berfungsi sebagai penunjuk yang dapat berubah, sehingga konten gambar yang ditunjuknya dapat berubah sementara Amazon ECS tidak mengidentifikasi modifikasi.

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.

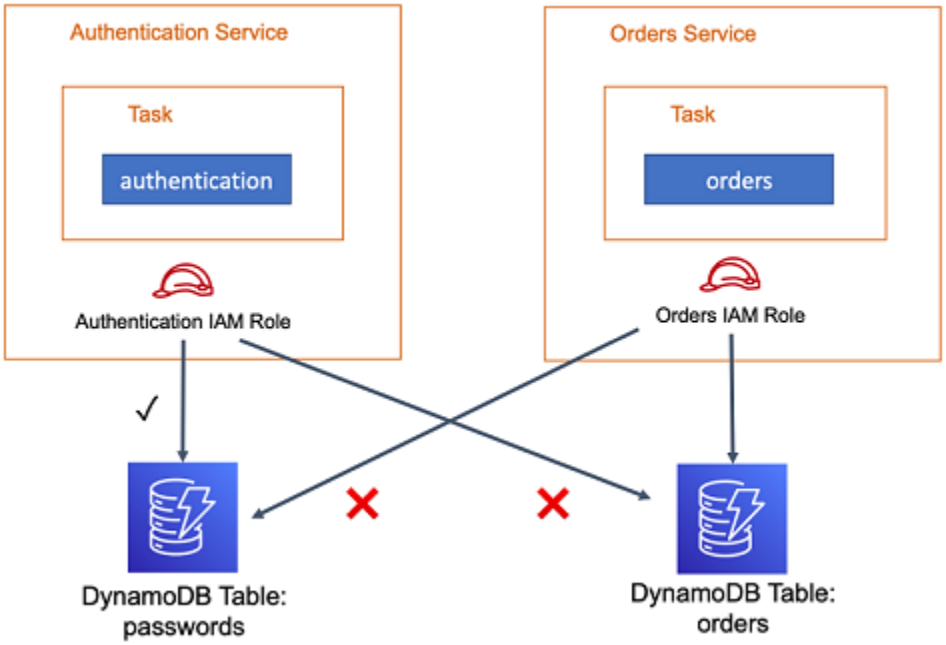
Pastikan ada one-to-one pemetaan antara versi kode aplikasi, tag gambar kontainer, dan revisi definisi tugas. Proses rilis tipikal melibatkan git commit yang diubah menjadi image container yang ditandai dengan git commit SHA. Kemudian, tag gambar kontainer itu mendapatkan revisi

definisi tugas Amazon ECS sendiri. Terakhir, layanan Amazon ECS diperbarui untuk memintanya menerapkan revisi definisi tugas baru.

Dengan menggunakan pendekatan ini, Anda dapat menjaga konsistensi antara pengaturan dan kode aplikasi saat meluncurkan versi baru aplikasi Anda. Misalnya, asumsikan bahwa Anda membuat versi baru aplikasi Anda yang menggunakan variabel lingkungan baru. Definisi tugas baru yang sesuai dengan perubahan itu juga mendefinisikan nilai untuk variabel lingkungan.

Gunakan peran IAM yang berbeda untuk setiap keluarga definisi tugas

Anda dapat menentukan peran IAM yang berbeda untuk tugas yang berbeda di Amazon ECS. Gunakan definisi tugas untuk menentukan peran IAM untuk aplikasi itu. Ketika kontainer dalam definisi tugas tersebut dijalankan, mereka dapat memanggil AWS API berdasarkan kebijakan yang ditentukan dalam peran IAM. Untuk informasi selengkapnya, lihat peran IAM untuk 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 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.

Layanan Amazon ECS

ECS menggunakan sumber daya layanan untuk mengelompokkan, memantau, mengganti, dan menskalakan tugas yang identik. Sumber daya layanan menentukan definisi dan revisi tugas apa yang diluncurkan Amazon ECS. Ini juga menentukan berapa banyak salinan definisi tugas yang diluncurkan dan sumber daya apa yang terhubung ke tugas yang diluncurkan. Sumber daya yang terhubung ini termasuk penyeimbang beban dan penemuan layanan. Sumber daya layanan juga mendefinisikan aturan untuk jaringan dan penempatan tugas pada perangkat keras.

Gunakan mode **`awsvpc`** jaringan dan berikan setiap layanan grup keamanannya sendiri

Kami menyarankan Anda menggunakan mode `awsvpc` jaringan untuk tugas-tugas di Amazon EC2. Ini memungkinkan setiap tugas memiliki alamat IP unik dengan grup keamanan tingkat layanan. Melakukannya menciptakan aturan grup keamanan per layanan, bukan grup keamanan tingkat instans yang digunakan dalam mode jaringan lain. Dengan menggunakan aturan grup keamanan per layanan, Anda dapat, misalnya, mengotorisasi satu layanan untuk berbicara dengan database Amazon RDS. Layanan lain dengan grup keamanan yang berbeda ditolak membuka koneksi ke database Amazon RDS itu.

Aktifkan tag terkelola Amazon ECS dan propagasi tag

Setelah Anda mengaktifkan tag terkelola Amazon ECS dan propagasi tag, Amazon ECS dapat melampirkan dan menyebarkan tag pada tugas yang diluncurkan layanan. Anda dapat menyesuaikan tag ini dan menggunakannya untuk membuat dimensi tag seperti `environment=production` atau `team=web` atau`application=storefront`. Tag ini digunakan dalam laporan penggunaan dan penagihan. Jika Anda mengatur tag dengan benar, Anda dapat menggunakannya untuk melihat berapa jam vCPU atau jam GB yang digunakan oleh lingkungan, tim, atau aplikasi tertentu. Ini dapat membantu Anda memperkirakan biaya keseluruhan infrastruktur Anda di sepanjang dimensi yang berbeda.

Praktik Terbaik - Jaringan

Aplikasi modern biasanya dibangun dari beberapa komponen terdistribusi yang berkomunikasi satu sama lain. Misalnya, aplikasi seluler atau web mungkin berkomunikasi dengan titik akhir API, dan API mungkin didukung oleh beberapa layanan mikro yang berkomunikasi melalui internet.

Panduan ini menyajikan praktik terbaik untuk membangun jaringan di mana komponen aplikasi Anda dapat berkomunikasi satu sama lain dengan aman dan dengan cara yang terukur.

Topik

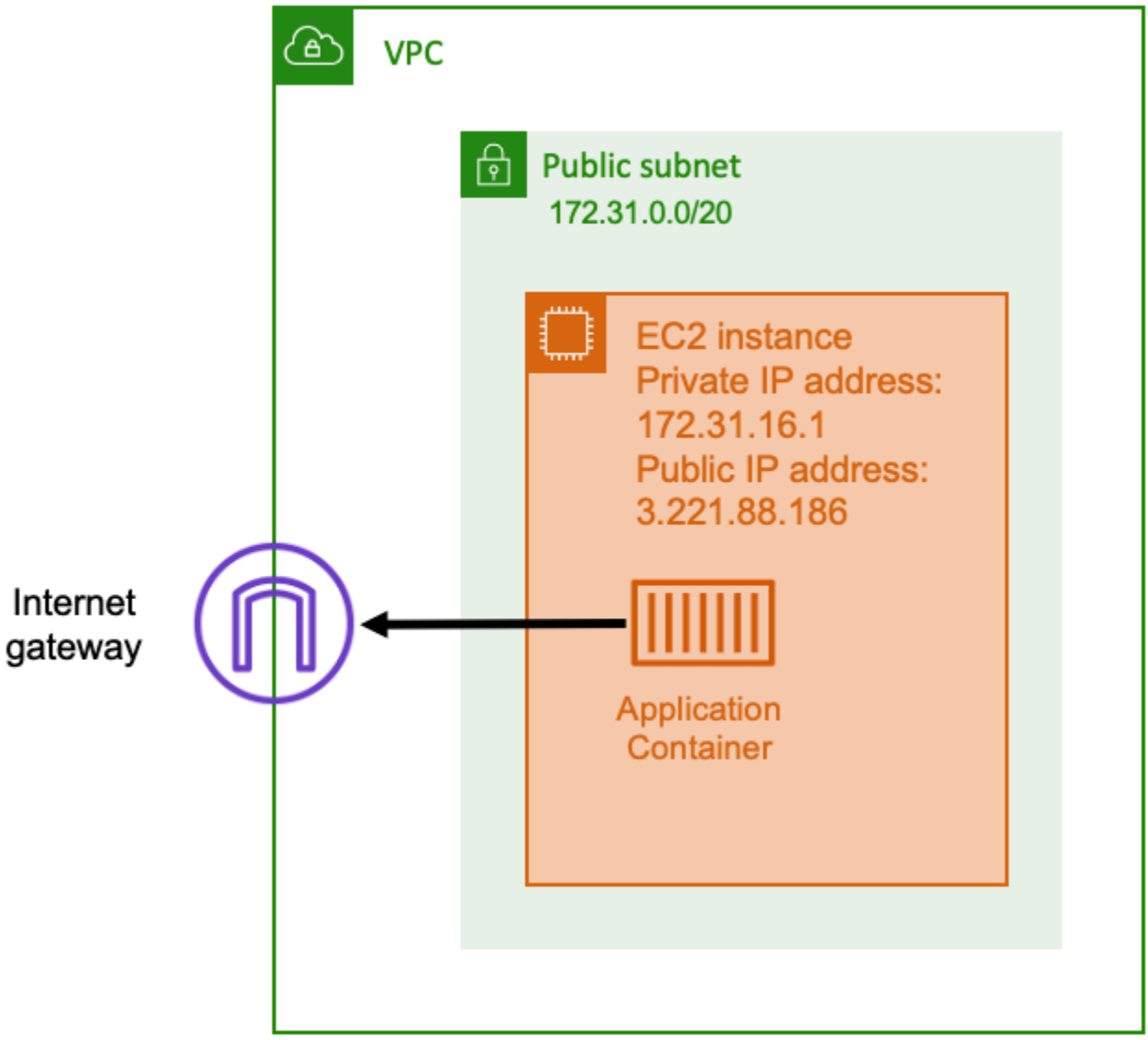
Menghubungkan ke internet

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.

Topik

Menggunakan subnet publik dan gateway internet



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

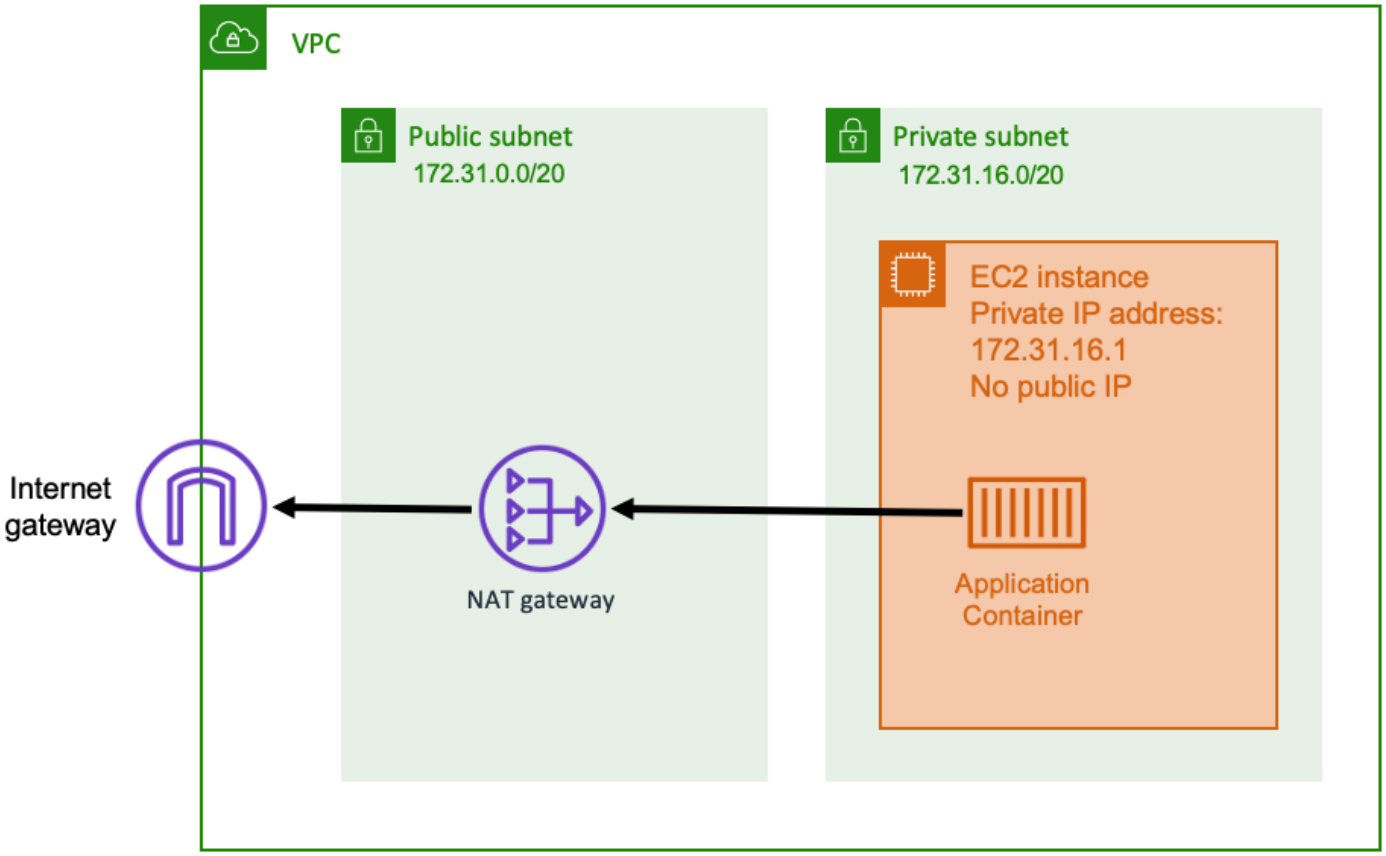
Misalnya, jika aplikasi Anda berjalan di Amazon EC2, pastikan port 22 untuk akses SSH tidak terbuka. Jika tidak, instans Anda dapat menerima upaya koneksi SSH konstan dari bot berbahaya di internet. Bot ini menjaring melalui alamat IP publik. Setelah mereka menemukan port SSH terbuka, mereka mencoba 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

- Menggunakan 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.
- Menggunakan Fargate - Saat Anda membuat layanan Amazon ECS, tentukan subnet publik untuk konfigurasi jaringan layanan Anda, dan pastikan opsi Tetapkan alamat IP publik diaktifkan. Setiap tugas Fargate berjejaring di subnet publik, dan memiliki alamat IP publik sendiri untuk komunikasi langsung dengan internet.

Menggunakan subnet pribadi dan gateway NAT



Dengan menggunakan subnet pribadi dan gateway NAT, Anda dapat menjalankan aplikasi kontainer Anda pada 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 membuat koneksi ke host menggunakan alamat IP pribadinya, tetapi host lain di internet tidak dapat membuat komunikasi masuk ke host.

Dengan subnet pribadi, Anda dapat menggunakan gateway Network Address Translation (NAT) untuk mengaktifkan host di dalam subnet pribadi untuk 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

- Menggunakan 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.
- Menggunakan Fargate - Saat Anda membuat layanan Amazon ECS, tentukan subnet pribadi untuk konfigurasi jaringan layanan Anda, dan jangan aktifkan 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.

Menerima koneksi masuk 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.

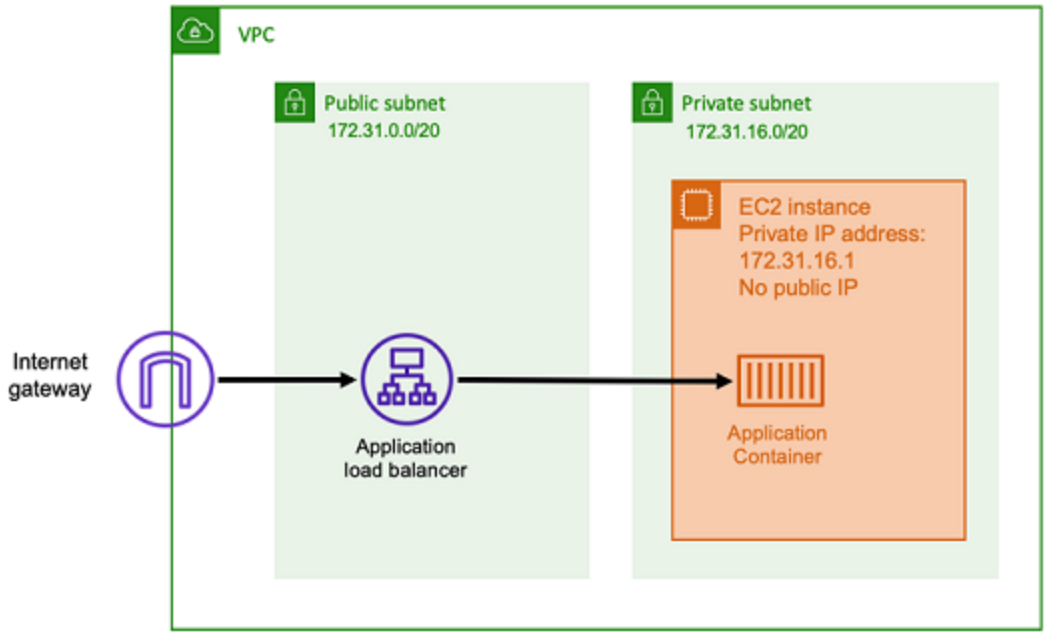
Salah satu pendekatan untuk masalah ini adalah meluncurkan container Anda pada host yang berada di subnet publik dengan alamat IP publik. Namun, kami tidak merekomendasikan ini untuk aplikasi skala besar. Untuk ini, pendekatan yang lebih baik adalah memiliki lapisan input yang dapat diskalakan yang berada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

Topik

- Penyeimbang Beban Aplikasi
- Network Load Balancer
- API HTTP Gerbang Amazon API

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, meneruskan permintaan melalui koneksi internal.

Application Load Balancer memiliki keunggulan sebagai berikut.

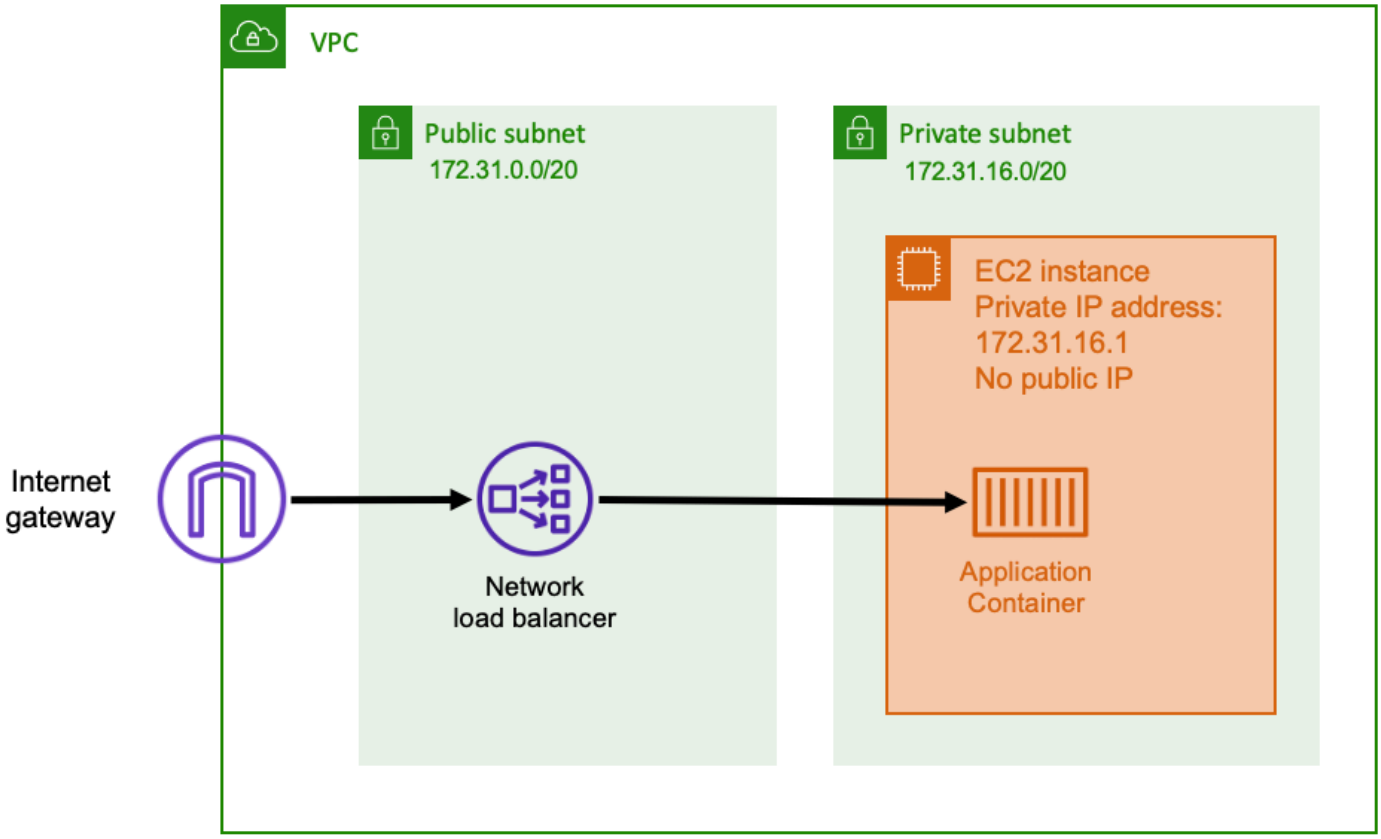
- Pengakhiran SSL/TLS — Application Load Balancer dapat mempertahankan komunikasi HTTPS yang aman dan sertifikat untuk komunikasi dengan klien. Ini secara opsional dapat menghentikan koneksi SSL di tingkat penyeimbang beban sehingga Anda tidak perlu menangani sertifikat dalam aplikasi Anda sendiri.
- 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

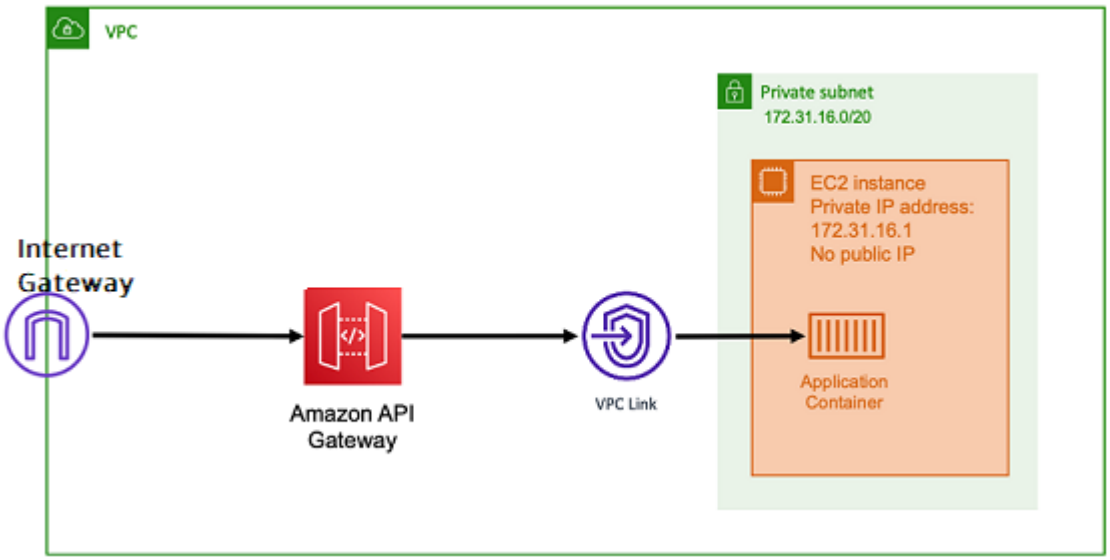
Karena Network Load Balancer tidak mengamati protokol aplikasi pada lapisan yang lebih tinggi dari model OSI, ia tidak dapat mengirim pesan penutupan ke klien dalam protokol tersebut. Berbeda dengan Application Load Balancer, koneksi tersebut harus ditutup oleh aplikasi atau Anda dapat mengkonfigurasi 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 HTTP API adalah ingress tanpa server yang 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 penemuan layanan Amazon ECS.

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.

Memilih mode jaringan

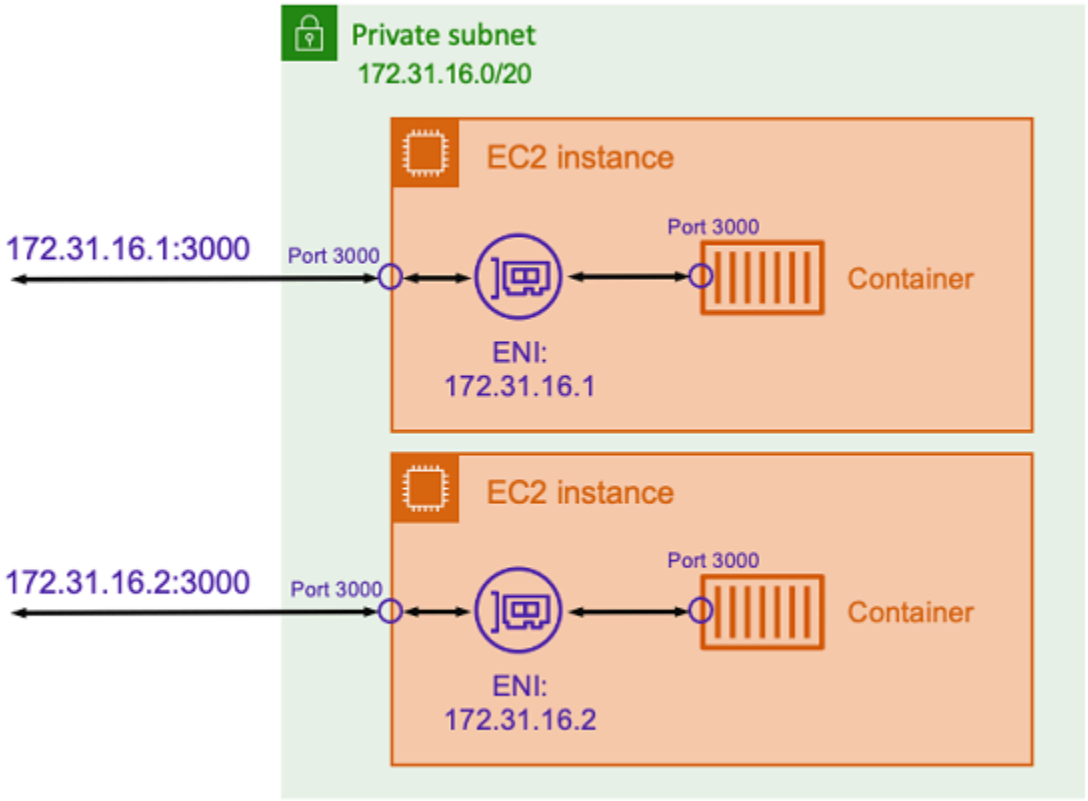
Pendekatan yang disebutkan sebelumnya untuk merancang koneksi jaringan masuk dan keluar dapat diterapkan ke salah satu beban kerja Anda AWS, bahkan jika mereka tidak berada di dalam wadah. Saat menjalankan kontainer aktif AWS, Anda perlu mempertimbangkan tingkat jaringan lain. Salah satu keuntungan utama menggunakan kontainer adalah Anda dapat mengemas beberapa kontainer ke dalam satu host. Saat melakukan ini, Anda harus memilih bagaimana Anda ingin membuat jaringan kontainer yang berjalan pada host yang sama. Berikut ini adalah opsi untuk dipilih.

- the section called “Mode host”- Mode `host` jaringan adalah mode jaringan paling dasar yang didukung di Amazon ECS.
- the section called “Mode jembatan”- Mode `bridge` jaringan memungkinkan Anda menggunakan jembatan jaringan virtual untuk membuat lapisan antara host dan jaringan wadah.

- the section called “AWSVPC modus”- Dengan mode `awsvpc` jaringan, Amazon ECS membuat dan mengelola Antarmuka Jaringan Elastis (ENI) untuk setiap tugas dan setiap tugas menerima alamat IP pribadinya sendiri dalam VPC.

Mode host

Mode `host` jaringan adalah mode jaringan paling dasar yang didukung di Amazon ECS. Menggunakan mode host, jaringan wadah terikat langsung ke host yang mendasari yang menjalankan wadah.



Asumsikan bahwa Anda menjalankan wadah 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.

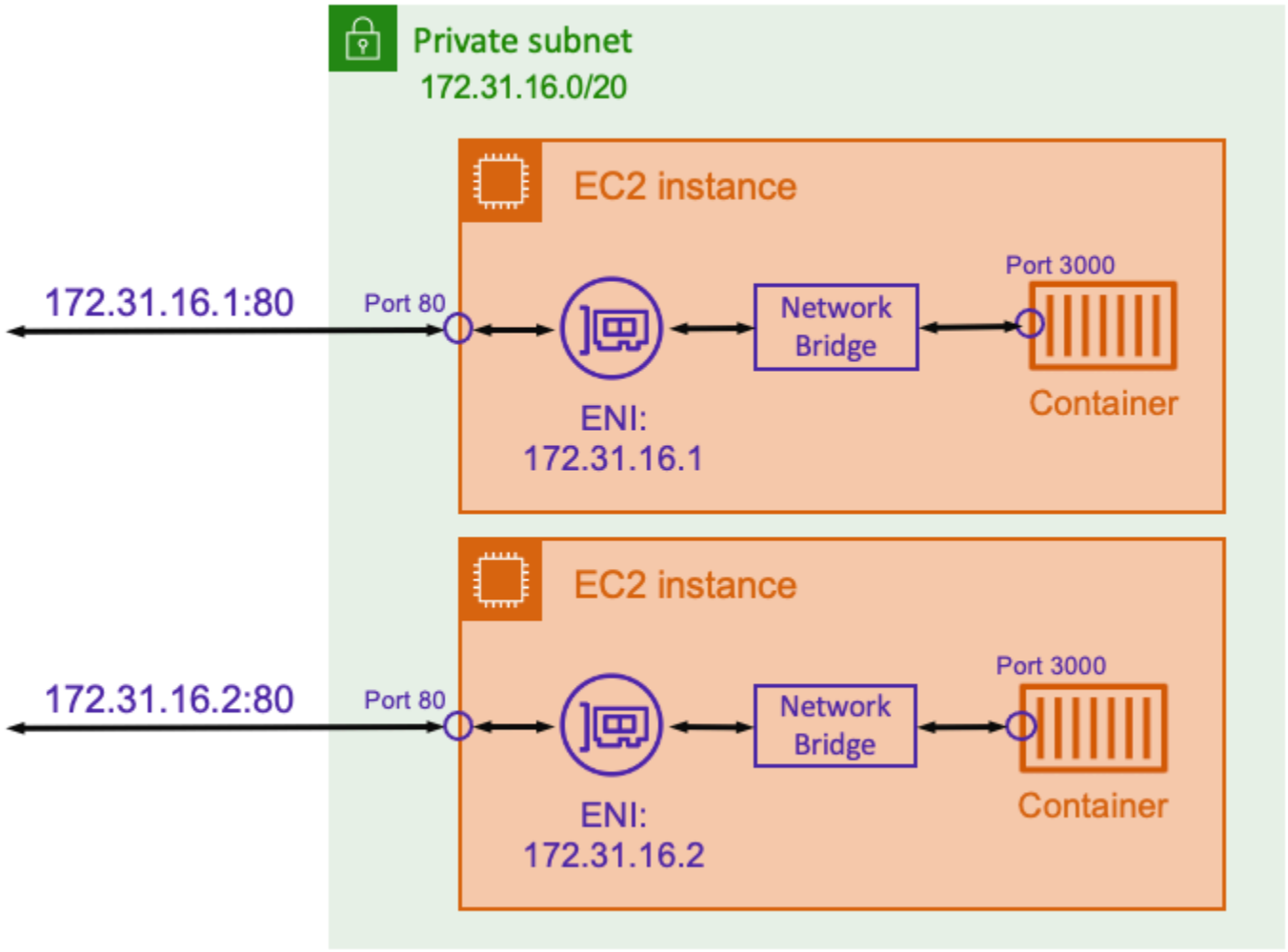
Ada kelemahan signifikan untuk menggunakan mode jaringan ini. Anda tidak dapat menjalankan lebih dari satu instantiasi tugas pada setiap host. Ini karena hanya tugas pertama yang dapat mengikat ke port yang diperlukan pada instans Amazon EC2. Juga tidak ada cara untuk memetakan ulang port kontainer saat 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 `host` jaringan hanya didukung untuk tugas Amazon ECS yang dihosting di instans Amazon EC2. Ini tidak didukung saat menggunakan Amazon ECS di Fargate.

Mode jembatan

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.

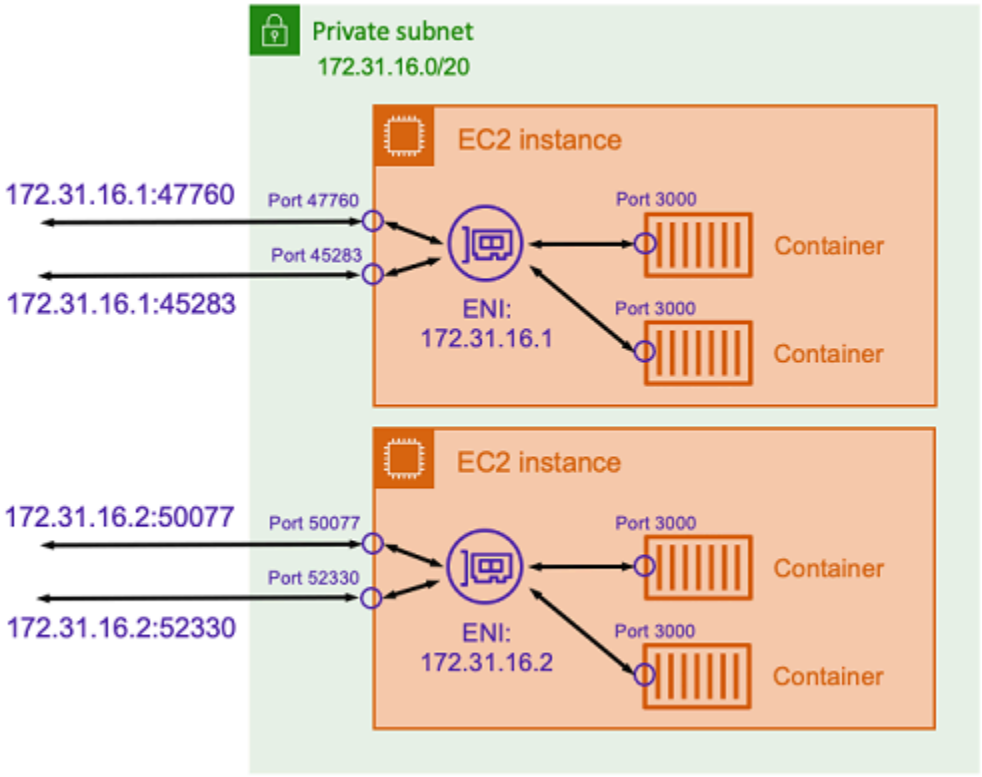


Dengan pemetaan port statis, Anda dapat secara eksplisit menentukan port host mana yang ingin Anda petakan ke port kontainer. Menggunakan contoh di atas, port `80` pada host sedang dipetakan ke port `3000` pada container. Untuk berkomunikasi dengan aplikasi kontainer, Anda mengirim lalu lintas ke port `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.

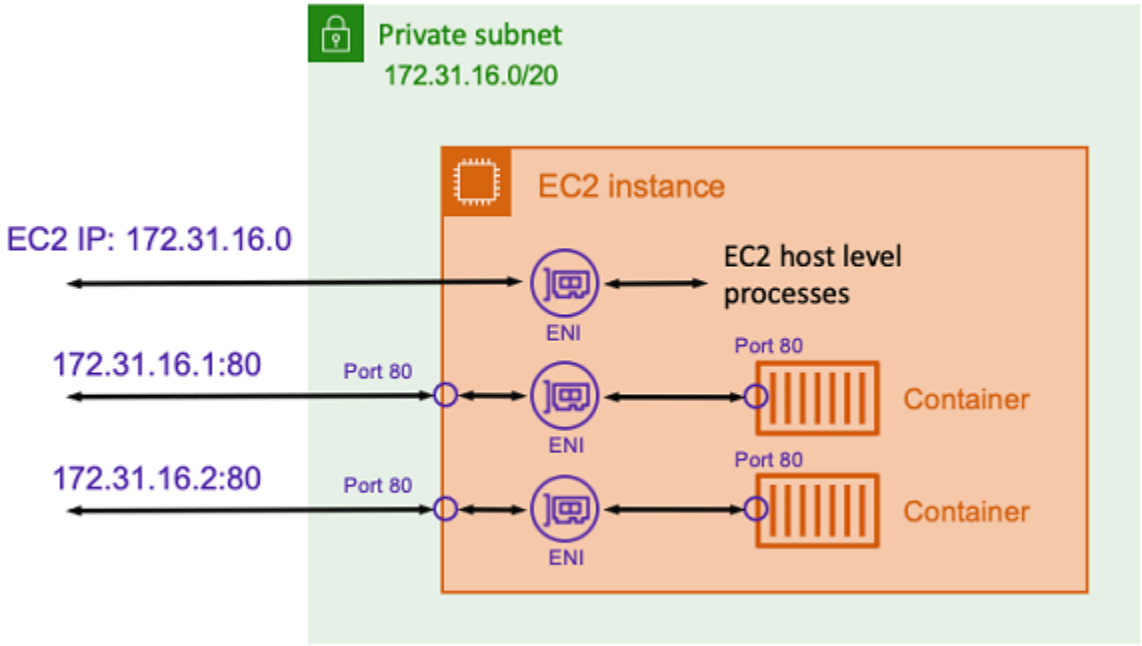
Namun, salah satu kelemahan menggunakan mode `bridge` jaringan adalah sulit untuk mengunci komunikasi layanan ke layanan. Karena layanan mungkin ditetapkan ke port acak dan tidak terpakai,

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.

Mode `bridge` jaringan hanya didukung untuk tugas Amazon ECS yang dihosting di instans Amazon EC2. Ini tidak didukung saat menggunakan Amazon ECS di Fargate.

AWSVPC modus

Dengan mode `awsvpc` jaringan, Amazon ECS membuat dan mengelola Antarmuka Jaringan Elastis (ENI) untuk setiap tugas dan setiap tugas menerima alamat IP pribadinya sendiri dalam VPC. ENI ini terpisah dari host ENI yang mendasarinya. Jika instans Amazon EC2 menjalankan beberapa tugas, ENI setiap tugas juga terpisah.



Pada contoh sebelumnya, instans Amazon EC2 ditetapkan ke ENI. ENI mewakili alamat IP dari instans EC2 yang digunakan untuk komunikasi jaringan di tingkat host. Setiap tugas juga memiliki ENI yang sesuai dan alamat IP pribadi. Karena setiap ENI terpisah, setiap kontainer dapat mengikat port `80` pada tugas ENI. Oleh karena itu, Anda tidak perlu melacak nomor port. Sebagai gantinya, Anda dapat mengirim lalu lintas ke port `80` di alamat IP tugas ENI.

Keuntungan menggunakan mode `awsvpc` jaringan adalah bahwa setiap tugas memiliki grup keamanan terpisah 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

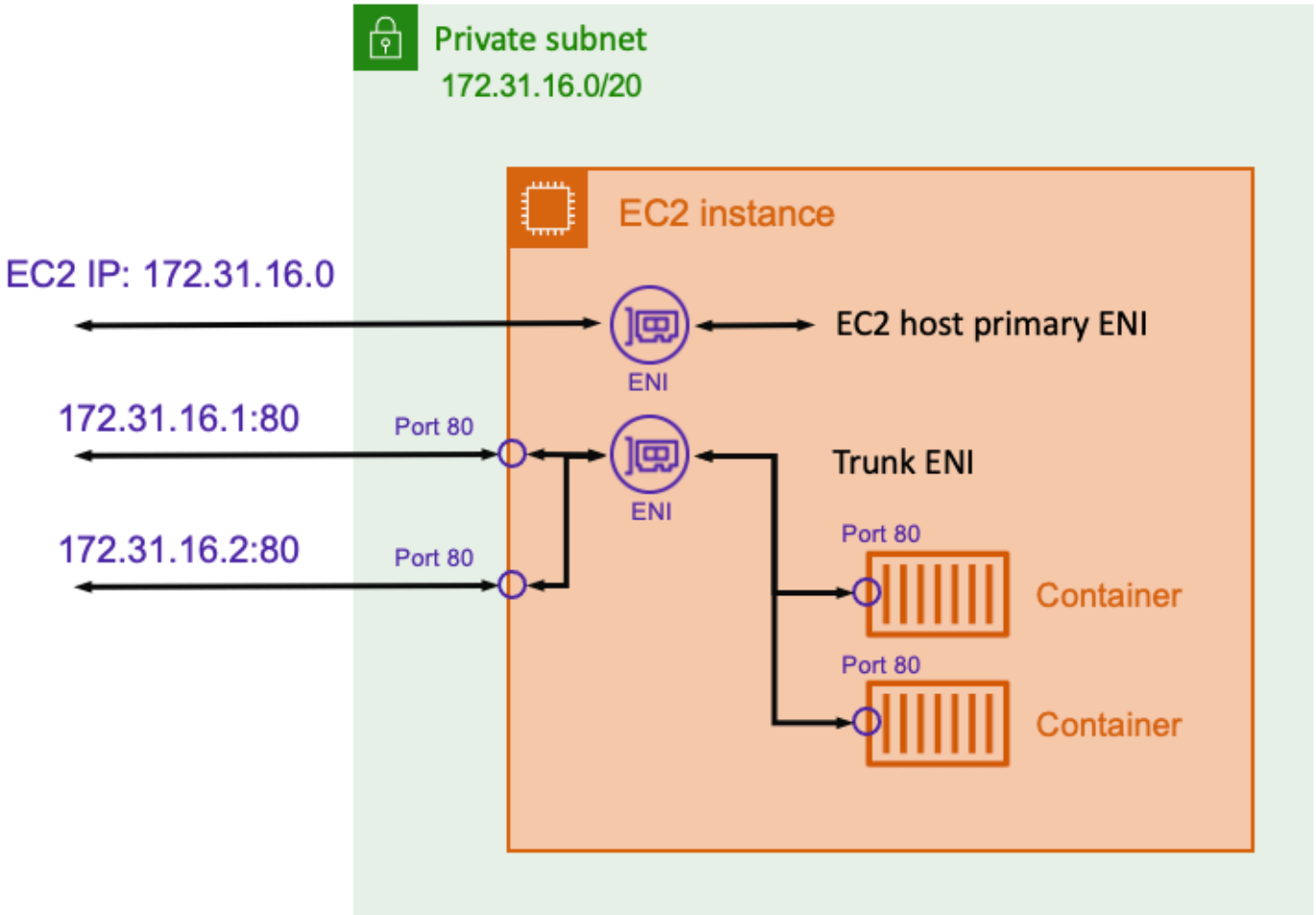
terperinci. Anda juga dapat mengonfigurasi tugas untuk menolak lalu lintas masuk dari tugas lain yang terletak di host yang sama.

Mode `awsvpc` jaringan didukung untuk tugas Amazon ECS yang dihosting di Amazon EC2 dan Fargate. Berhati-hatilah bahwa, saat menggunakan Fargate, `awsvpc` mode jaringan diperlukan.

Saat menggunakan mode `awsvpc` jaringan ada beberapa tantangan yang harus Anda perhatikan.

Meningkatkan kepadatan tugas dengan ENI Trunking

Kerugian terbesar menggunakan mode `awsvpc` jaringan dengan tugas yang di-host di instans Amazon EC2 adalah instans EC2 memiliki batasan jumlah ENI yang dapat dilampirkan padanya. Ini membatasi berapa banyak tugas yang dapat Anda tempatkan pada setiap instance. Amazon ECS menyediakan fitur trunking ENI yang meningkatkan jumlah ENI yang tersedia untuk mencapai kepadatan tugas yang lebih banyak.



Saat menggunakan trunking ENI, dua lampiran ENI digunakan secara default. Yang pertama adalah ENI utama dari instance, yang digunakan untuk setiap proses tingkat host. Yang kedua adalah trunk ENI, yang dibuat Amazon ECS. Fitur ini hanya didukung pada jenis instans Amazon EC2 tertentu.

Perhatikan contoh ini. Tanpa trunking ENI, `c5.large` instance yang memiliki dua vCPU hanya dapat menampung dua tugas. Namun, dengan ENI trunking, `c5.large` instance yang memiliki dua vCPU dapat menampung hingga sepuluh tugas. Setiap tugas memiliki alamat IP dan grup keamanan yang berbeda. Untuk informasi selengkapnya tentang jenis instans yang tersedia dan kepadatannya, lihat Jenis instans Amazon EC2 yang didukung di Panduan Pengembang Layanan Kontainer Elastis Amazon.

Trunking ENI tidak berdampak pada kinerja runtime dalam hal latensi atau bandwidth.

Untuk informasi selengkapnya, lihat Trunking antarmuka jaringan elastis di Panduan Pengembang Layanan Kontainer Elastis Amazon.

Mencegah kelelahan alamat IP

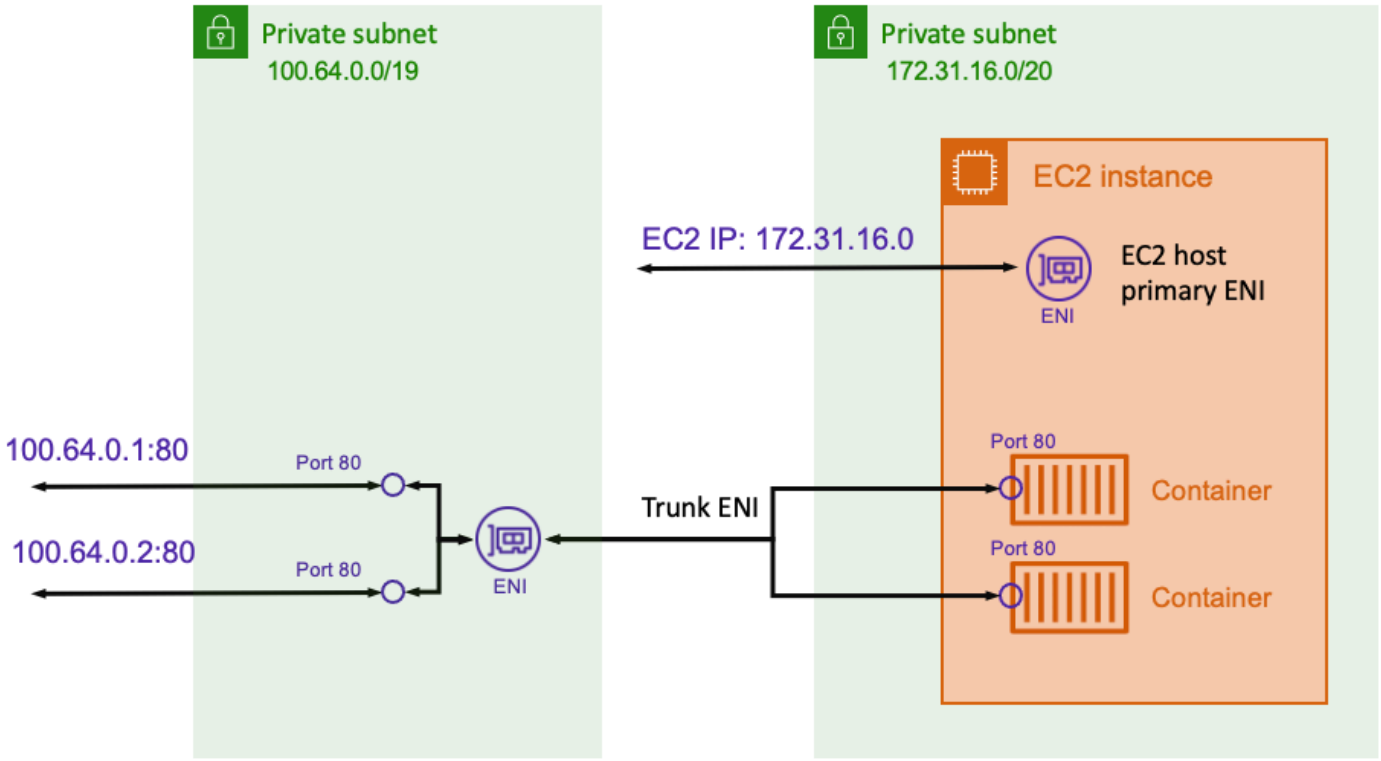
Dengan menetapkan alamat IP terpisah untuk setiap tugas, Anda dapat menyederhanakan infrastruktur Anda secara keseluruhan dan memelihara grup keamanan yang memberikan tingkat keamanan yang tinggi. Namun, konfigurasi ini dapat menyebabkan kelelahan IP.

VPC default di AWS akun Anda memiliki subnet yang telah disediakan sebelumnya yang memiliki rentang CIDR. `/20` Ini berarti setiap subnet memiliki 4.091 alamat IP yang tersedia. Perhatikan bahwa beberapa alamat IP dalam `/20` rentang dicadangkan untuk penggunaan AWS tertentu. Perhatikan contoh ini. Anda mendistribusikan aplikasi Anda di tiga subnet di tiga Availability Zone untuk ketersediaan tinggi. Dalam hal ini, Anda dapat menggunakan sekitar 12.000 alamat IP di ketiga subnet.

Menggunakan trunking ENI, setiap instans Amazon EC2 yang Anda luncurkan memerlukan dua alamat IP. Satu alamat IP digunakan untuk ENI primer, dan alamat IP lainnya digunakan untuk trunk ENI. Setiap tugas Amazon ECS pada instance memerlukan satu alamat IP. Jika Anda meluncurkan beban kerja yang sangat besar, Anda bisa kehabisan alamat IP yang tersedia. Hal ini dapat mengakibatkan kegagalan peluncuran Amazon EC2 atau kegagalan peluncuran tugas. Kesalahan ini terjadi karena ENI tidak dapat menambahkan alamat IP di dalam VPC jika tidak ada alamat IP yang tersedia.

Saat menggunakan mode `awsvpc` jaringan, Anda harus mengevaluasi persyaratan alamat IP Anda dan memastikan bahwa rentang CIDR subnet Anda memenuhi kebutuhan Anda. Jika Anda sudah

mulai menggunakan VPC yang memiliki subnet kecil dan mulai kehabisan ruang alamat, Anda dapat menambahkan subnet sekunder.



Dengan menggunakan trunking ENI, Amazon VPC CNI dapat dikonfigurasi untuk menggunakan ENI di ruang alamat IP yang berbeda dari host. Dengan melakukan ini, Anda dapat memberikan host Amazon EC2 dan tugas Anda rentang alamat IP yang berbeda yang tidak tumpang tindih. Dalam diagram contoh, alamat IP host EC2 berada dalam subnet yang memiliki rentang `172.31.16.0/20` IP. Namun, tugas yang berjalan pada host diberi alamat IP dalam `100.64.0.0/19` kisaran. Dengan menggunakan dua rentang IP independen, Anda tidak perlu khawatir tentang tugas yang menghabiskan terlalu banyak alamat IP dan tidak meninggalkan cukup alamat IP untuk instance.

Menggunakan mode tumpukan ganda IPv6

Mode `awsvpc` jaringan kompatibel dengan VPC yang dikonfigurasi untuk mode tumpukan ganda IPv6. VPC yang menggunakan mode tumpukan ganda dapat berkomunikasi melalui IPv4, IPv6, atau keduanya. Setiap subnet di VPC dapat memiliki rentang IPv4 CIDR dan rentang IPv6 CIDR. Untuk informasi selengkapnya, lihat Pengalamatan IP di VPC Anda di Panduan Pengguna Amazon VPC.

Anda tidak dapat menonaktifkan dukungan IPv4 untuk VPC dan subnet Anda untuk mengatasi masalah kelelahan IPv4. Namun, dengan dukungan IPv6, Anda dapat menggunakan beberapa

kemampuan baru, khususnya gateway internet khusus egres. Sebuah gateway internet egress-only memungkinkan tugas untuk menggunakan alamat IPv6 yang dapat dirutekan secara publik untuk memulai koneksi keluar ke internet. Tetapi gateway internet khusus egres tidak mengizinkan koneksi dari internet. Untuk informasi selengkapnya, lihat gateway internet khusus eGress di Panduan Pengguna Amazon VPC.

Menghubungkan ke AWS layanan dari dalam VPC Anda

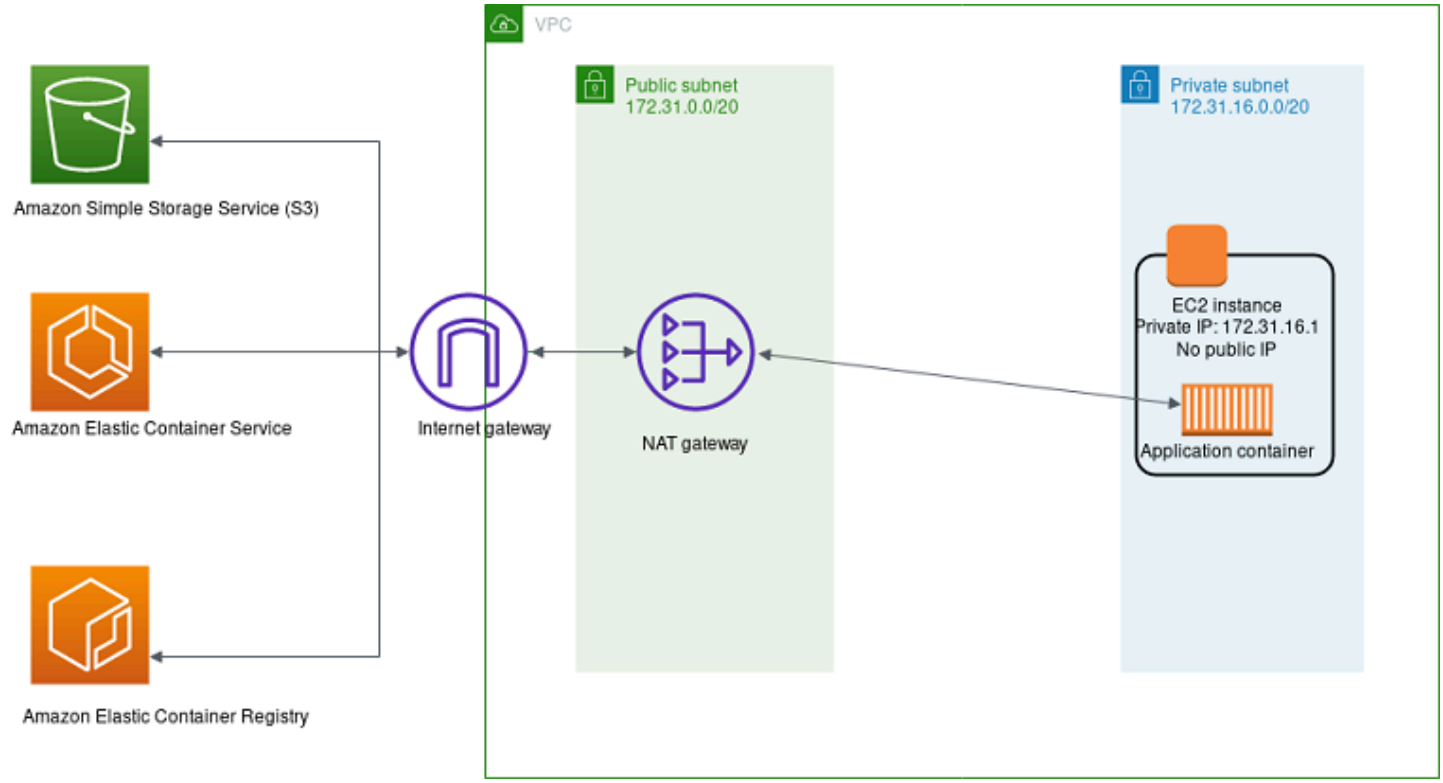
Agar Amazon ECS berfungsi dengan baik, agen kontainer ECS yang berjalan di setiap host harus berkomunikasi dengan bidang kontrol Amazon ECS. Jika Anda menyimpan gambar kontainer di Amazon ECR, host Amazon EC2 harus berkomunikasi ke titik akhir layanan Amazon ECR, dan ke Amazon S3, tempat lapisan gambar disimpan. Jika Anda menggunakan AWS layanan lain untuk aplikasi kontainer Anda, seperti menyimpan data yang disimpan di DynamoDB, periksa kembali apakah layanan ini juga memiliki dukungan jaringan yang diperlukan.

Topik

- Gateway NAT
- AWS PrivateLink

Gateway NAT

Menggunakan gateway NAT adalah cara termudah untuk memastikan bahwa tugas Amazon ECS Anda dapat mengakses layanan lain AWS . Untuk informasi lebih lanjut tentang pendekatan ini, lihatMenggunakan subnet pribadi dan gateway NAT.



Berikut ini adalah kerugian untuk menggunakan pendekatan ini:

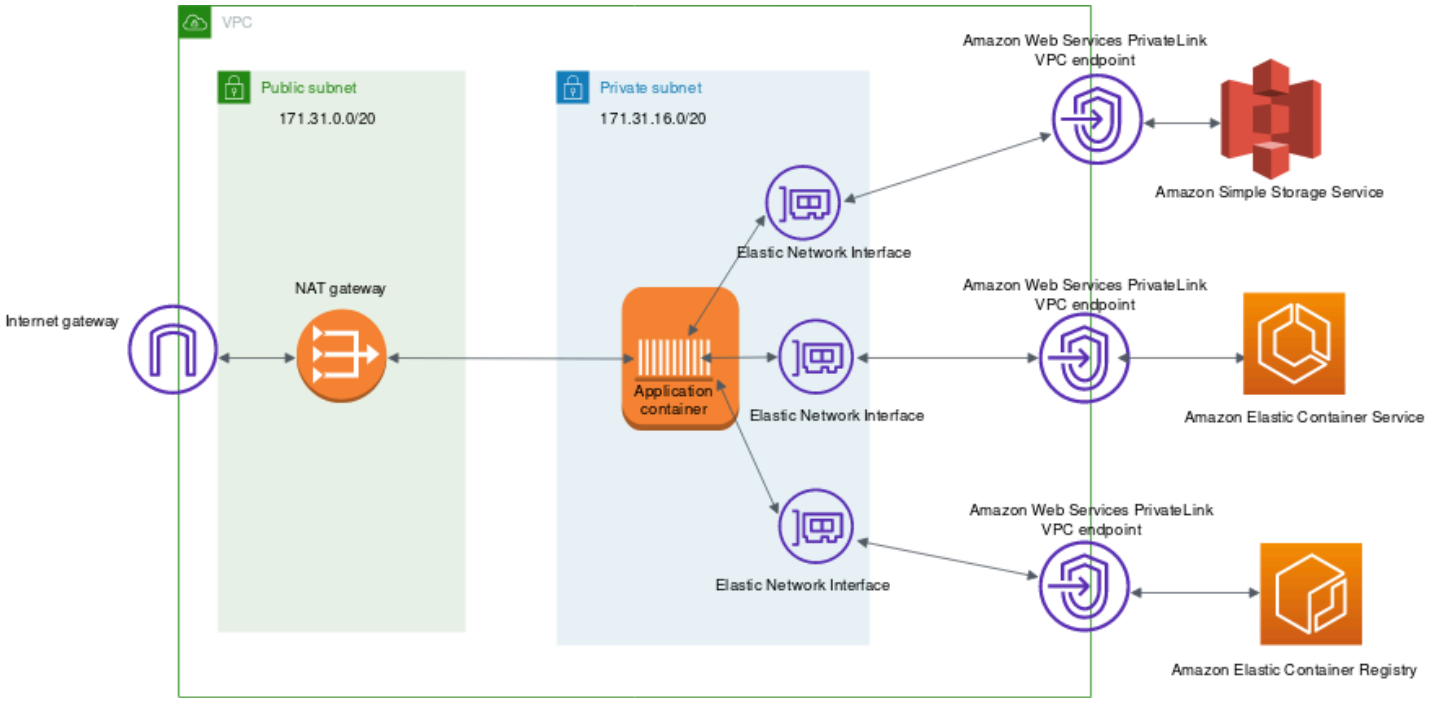
- Anda tidak dapat membatasi tujuan apa yang dapat berkomunikasi dengan gateway NAT. Anda juga tidak dapat membatasi tujuan mana yang dapat dikomunikasikan oleh tingkat backend Anda tanpa mengganggu semua komunikasi keluar dari VPC Anda.
- Gateway NAT mengenakan biaya untuk setiap GB data yang lewat. Jika Anda menggunakan gateway NAT untuk mengunduh file besar dari Amazon S3, atau melakukan kueri database volume tinggi ke DynamoDB, Anda dikenakan biaya untuk setiap GB bandwidth. Selain itu, gateway NAT mendukung bandwidth 5 Gbps dan secara otomatis menskalakan hingga 45 Gbps. Jika Anda merutekan melalui gateway NAT tunggal, aplikasi yang memerlukan koneksi bandwidth yang sangat tinggi mungkin mengalami kendala jaringan. Sebagai solusinya, Anda dapat membagi beban kerja Anda di beberapa subnet dan memberikan masing-masing subnet gateway NAT sendiri.

AWS PrivateLink

AWS PrivateLink menyediakan konektivitas pribadi antara VPC, AWS layanan, dan jaringan lokal Anda tanpa mengekspos lalu lintas Anda ke internet publik.

Salah satu teknologi yang digunakan untuk mencapai ini adalah titik akhir VPC. Titik akhir VPC memungkinkan koneksi pribadi antara VPC Anda dan layanan yang didukung AWS serta layanan titik akhir VPC. Lalu lintas antara VPC Anda dan layanan lainnya tidak meninggalkan jaringan Amazon. Titik akhir VPC tidak memerlukan gateway internet, gateway pribadi virtual, perangkat NAT, koneksi VPN, atau koneksi. AWS Direct Connect Instans Amazon EC2 di VPC Anda tidak memerlukan alamat IP publik untuk berkomunikasi dengan sumber daya dalam layanan.

Diagram berikut menunjukkan cara kerja komunikasi ke AWS layanan saat Anda menggunakan titik akhir VPC alih-alih gateway internet. AWS PrivateLink ketentuan antarmuka jaringan elastis (ENI) di dalam subnet, dan aturan perutean VPC digunakan untuk mengirim komunikasi apa pun ke nama host layanan melalui ENI, langsung ke layanan tujuan. AWS Lalu lintas ini tidak perlu lagi menggunakan gateway NAT atau gateway internet.



Berikut ini adalah beberapa titik akhir VPC umum yang digunakan dengan layanan Amazon ECS.

- Titik akhir VPC gerbang S3
- Titik akhir DynamoDB VPC
- Titik akhir Amazon ECS VPC
- Titik akhir Amazon ECR VPC

Banyak AWS layanan lain yang mendukung titik akhir VPC. Jika Anda menggunakan AWS layanan apa pun secara berlebihan, Anda harus mencari dokumentasi khusus untuk layanan itu dan cara membuat titik akhir VPC untuk lalu lintas tersebut.

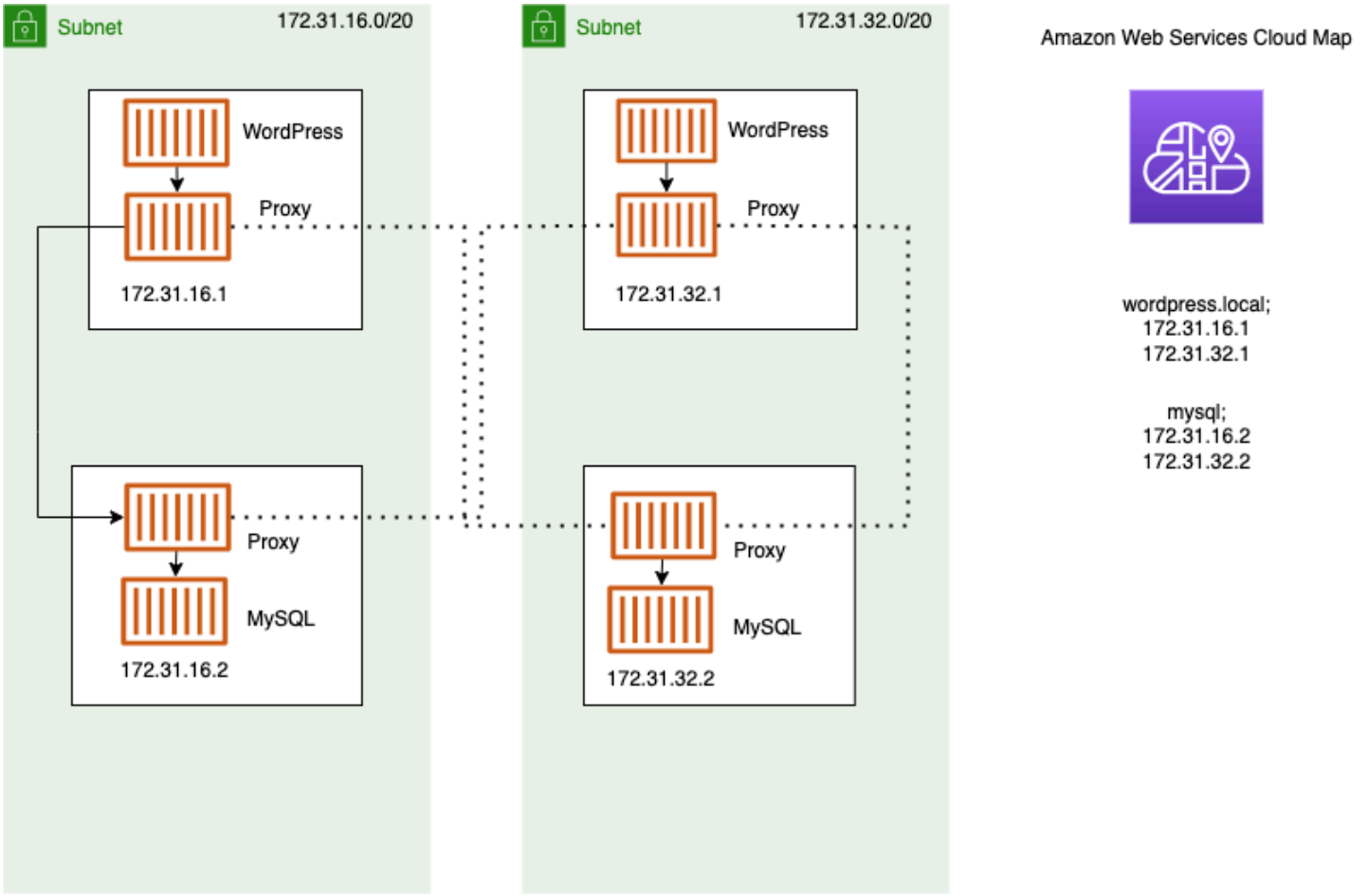
Jaringan antara layanan Amazon ECS dalam VPC

Dengan menggunakan tugas Amazon ECS di VPC, Anda dapat membagi aplikasi monolitik menjadi beberapa bagian terpisah yang dapat digunakan dan diskalakan secara independen di lingkungan yang aman. Arsitektur ini disebut arsitektur berorientasi layanan (SOA) atau layanan mikro. Namun, mungkin sulit untuk memastikan bahwa semua bagian ini, baik di dalam maupun di luar VPC, dapat berkomunikasi satu sama lain. Ada beberapa pendekatan untuk memfasilitasi komunikasi, semuanya dengan kelebihan dan kekurangan yang berbeda.

Menggunakan Service Connect

Kami merekomendasikan Amazon ECS 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 ECS di cluster yang sama, cluster lain, termasuk di seluruh VPC yang sama. Wilayah AWS Untuk informasi selengkapnya, lihat Amazon ECS Service Connect di Panduan Pengembang Layanan Amazon Elastic Container.

Saat Anda menggunakan Service Connect, 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.



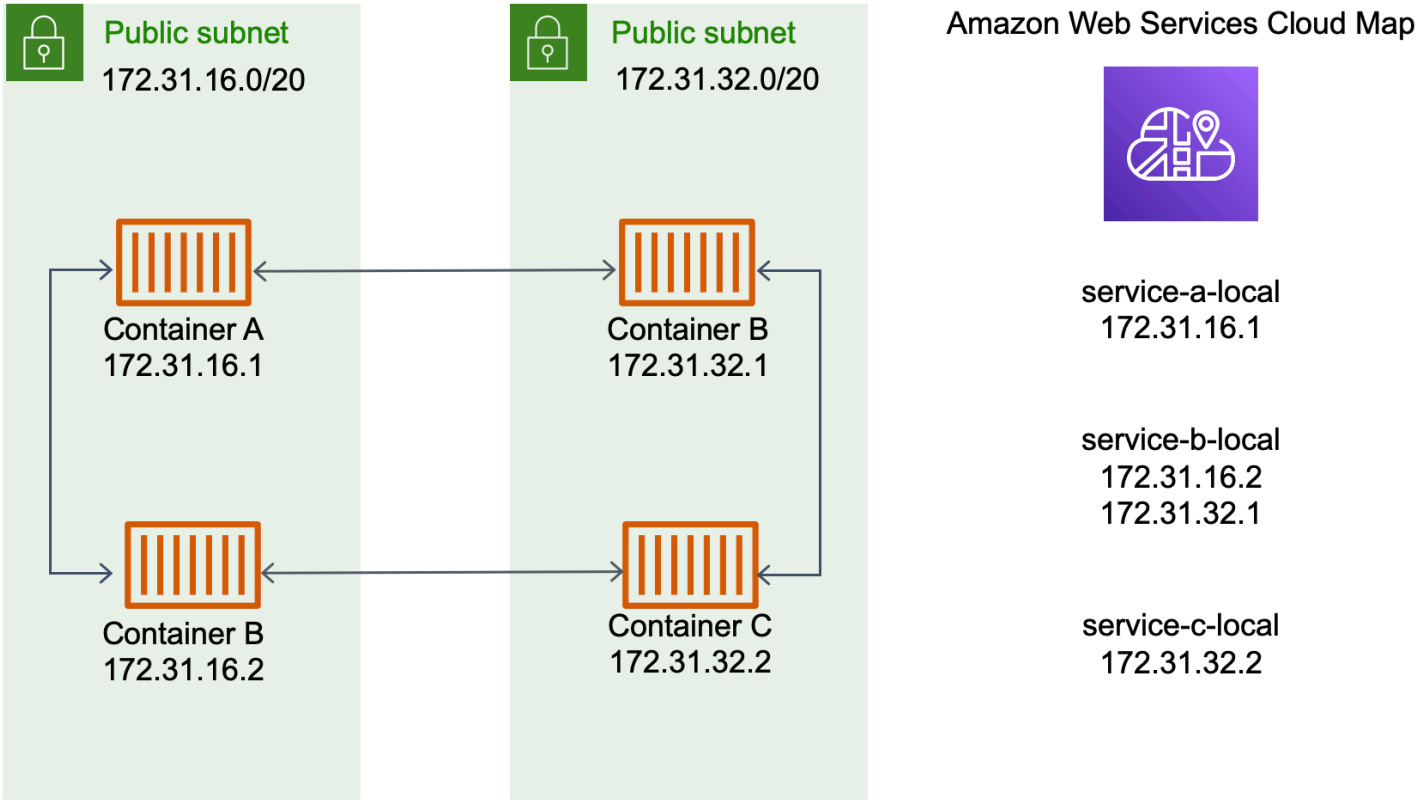
Perubahan hanya terjadi selama penerapan

Anda menyediakan konfigurasi lengkap di dalam setiap layanan ECS dan definisi tugas. ECS mengelola perubahan 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.

Menggunakan penemuan layanan

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. Untuk informasi selengkapnya, lihat Penemuan layanan di Panduan Pengembang Layanan Kontainer Elastis Amazon.



Dalam diagram sebelumnya, ada tiga layanan. `service`Amemiliki satu wadah dan berkomunikasi dengan`service`B, yang memiliki dua wadah. `service`Bjuga harus berkomunikasi dengan`service`C, yang memiliki satu wadah. Setiap kontainer di ketiga layanan ini dapat menggunakan nama DNS internal AWS Cloud Map untuk menemukan alamat IP internal kontainer dari layanan hilir yang perlu dikomunikasikan.

Pendekatan service-to-service komunikasi ini memberikan latensi rendah. Pada pandangan pertama, ini juga sederhana karena 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`

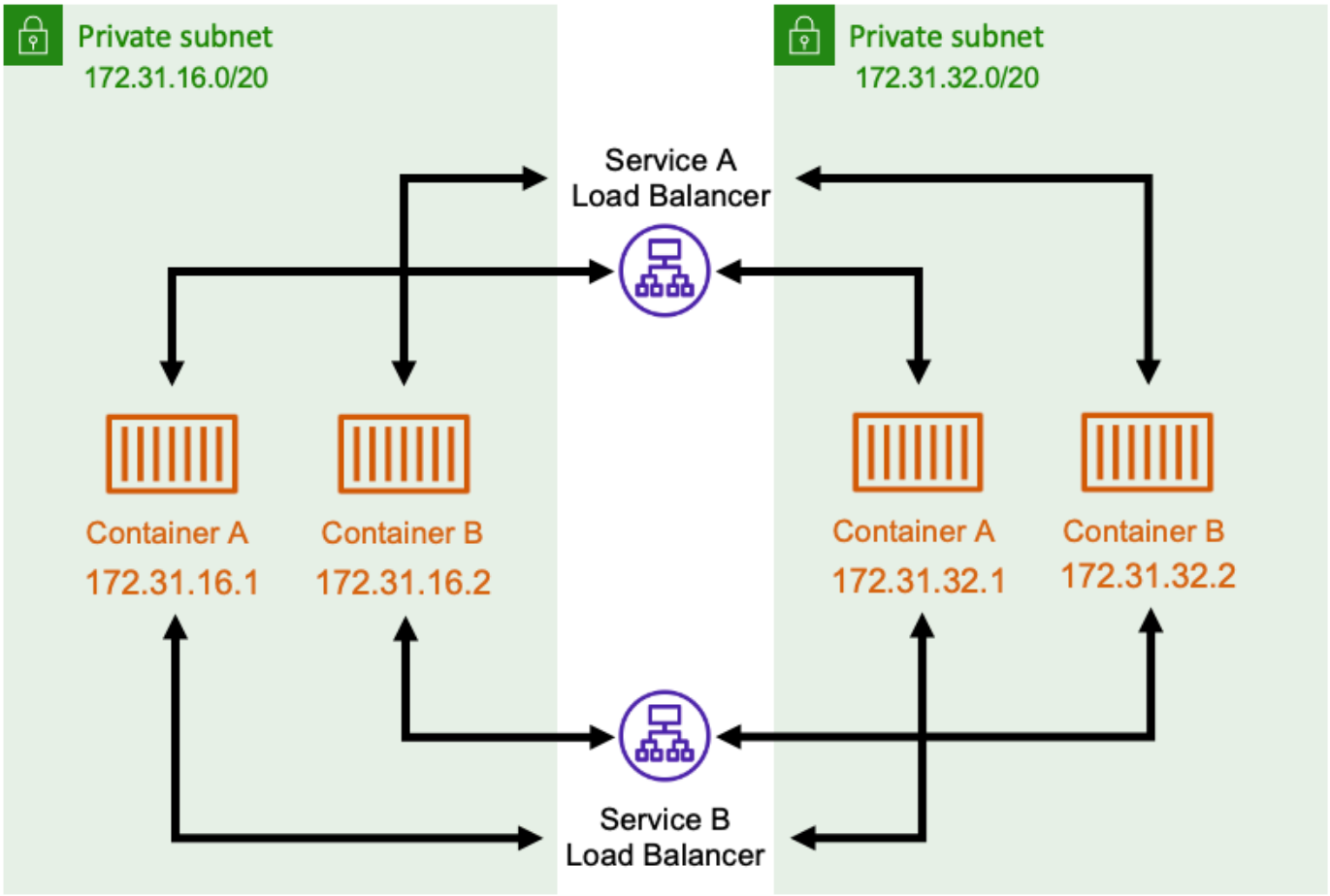
jaringan, beberapa kontainer dapat berbagi alamat IP yang sama. Selain itu, pemetaan port dinamis menyebabkan kontainer diberi nomor port secara acak pada alamat IP tunggal itu. Pada titik ini, `A` catatan tidak lagi cukup untuk penemuan layanan. Anda juga harus menggunakan `SRV` catatan. Jenis catatan 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.

Menggunakan penyeimbang beban internal

Pendekatan lain untuk service-to-service komunikasi adalah dengan menggunakan penyeimbang beban internal. Penyeimbang beban internal ada sepenuhnya di dalam VPC Anda dan hanya dapat diakses oleh layanan di dalam VPC Anda.



Penyeimbang beban mempertahankan ketersediaan tinggi dengan menyebarkan sumber daya yang berlebihan ke setiap subnet. Ketika sebuah wadah dari `serviceA` kebutuhan untuk berkomunikasi dengan wadah dari`serviceB`, itu membuka koneksi ke penyeimbang beban. Penyeimbang beban kemudian membuka koneksi ke wadah dari`service B`. Load balancer berfungsi sebagai tempat terpusat untuk mengelola semua koneksi antara setiap layanan.

Jika wadah `serviceB` berhenti, maka penyeimbang beban dapat mengeluarkan wadah itu dari kolam. Load balancer juga melakukan pemeriksaan kesehatan terhadap setiap target hilir di kolam dan secara otomatis dapat menghapus target buruk dari kolam sampai mereka menjadi sehat kembali. Aplikasi tidak perlu lagi menyadari berapa banyak kontainer hilir yang ada. Mereka hanya membuka koneksi mereka ke penyeimbang beban.

Pendekatan ini menguntungkan untuk semua mode jaringan. Penyeimbang beban dapat melacak alamat IP tugas saat menggunakan mode `awsvpc` jaringan, serta kombinasi alamat IP dan port yang lebih canggih saat menggunakan mode `bridge` jaringan. Ini mendistribusikan lalu lintas secara

merata di semua alamat IP dan kombinasi port, bahkan jika beberapa kontainer benar-benar di-host pada instance Amazon EC2 yang sama, hanya pada port yang berbeda.

Salah satu kelemahan dari pendekatan ini adalah biaya. Agar sangat tersedia, penyeimbang beban harus memiliki sumber daya di setiap Availability Zone. Ini menambah biaya tambahan karena overhead membayar penyeimbang beban dan untuk jumlah lalu lintas yang melewati penyeimbang beban.

Namun, Anda dapat mengurangi biaya overhead dengan memiliki beberapa layanan berbagi penyeimbang beban. Ini sangat cocok untuk layanan REST yang menggunakan Application Load Balancer. Anda dapat membuat aturan perutean berbasis jalur yang merutekan lalu lintas ke layanan yang berbeda. Misalnya, `/api/user/*` mungkin merutekan ke kontainer yang merupakan bagian dari `user` layanan, sedangkan `/api/order/*` mungkin merutekan ke `order` layanan terkait. Dengan pendekatan ini, Anda hanya membayar satu Application Load Balancer, dan memiliki satu URL yang konsisten untuk API Anda. Namun, Anda dapat membagi lalu lintas ke berbagai layanan mikro di backend.

Layanan jaringan di seluruh AWS akun dan VPC

Jika Anda bagian dari organisasi dengan beberapa tim dan divisi, Anda mungkin menyebarkan layanan secara independen ke dalam VPC terpisah di dalam AWS akun bersama atau ke dalam VPC yang terkait dengan beberapa akun individual. AWS Tidak peduli ke arah mana Anda menggunakan layanan Anda, kami sarankan Anda melengkapi komponen jaringan Anda untuk membantu mengarahkan lalu lintas antar VPC. Untuk ini, beberapa AWS layanan dapat digunakan untuk melengkapi komponen jaringan Anda yang ada.

- AWS Transit Gateway — Anda harus mempertimbangkan layanan jaringan ini terlebih dahulu. Layanan ini berfungsi sebagai hub pusat untuk merutekan koneksi Anda antara VPC Amazon, AWS akun, dan jaringan lokal. Untuk informasi selengkapnya, lihat Apa itu gateway transit? di Panduan Gerbang Transit VPC Amazon.
- Dukungan Amazon VPC dan VPN — Anda dapat menggunakan layanan ini untuk membuat koneksi site-to-site VPN untuk menghubungkan jaringan lokal ke VPC Anda. Untuk informasi lebih lanjut, lihat Apa itu AWS Site-to-Site VPN? dalam AWS Site-to-Site VPN User Guide.
- Amazon VPC — Anda dapat menggunakan peering VPC Amazon untuk membantu Anda menghubungkan beberapa VPC, baik di akun yang sama, atau di seluruh akun. Untuk informasi selengkapnya, lihat Apa yang dimaksud peering VPC? di Panduan Peering Amazon VPC.

- VPC Bersama — Anda dapat menggunakan subnet VPC dan VPC di beberapa akun. AWS Untuk informasi selengkapnya, lihat Bekerja dengan VPC bersama di Panduan Pengguna Amazon VPC.

Mengoptimalkan dan memecahkan masalah

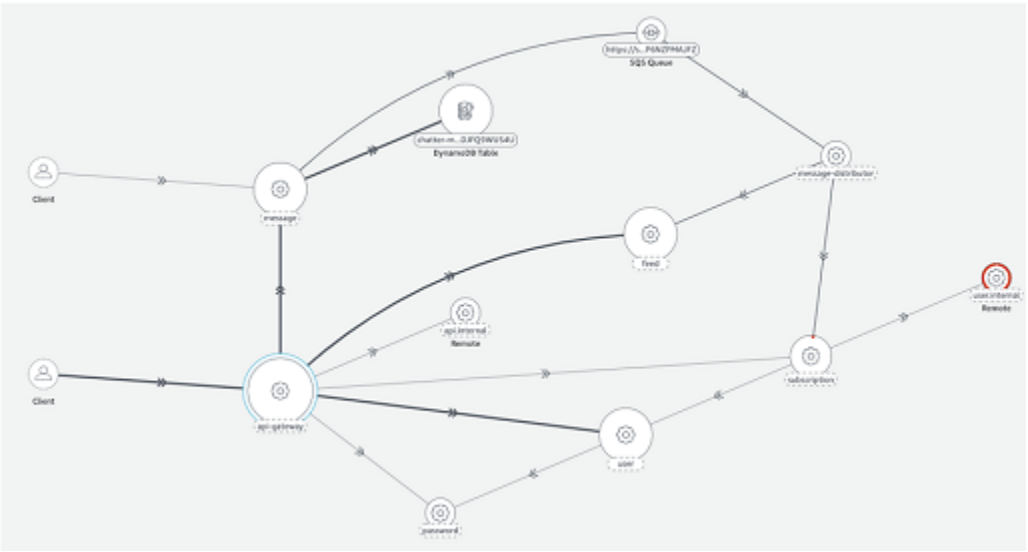
Layanan dan fitur berikut dapat membantu Anda mendapatkan wawasan tentang konfigurasi jaringan dan layanan Anda. Anda dapat menggunakan informasi ini untuk memecahkan masalah jaringan dan mengoptimalkan layanan Anda dengan lebih baik.

CloudWatch Wawasan Kontainer

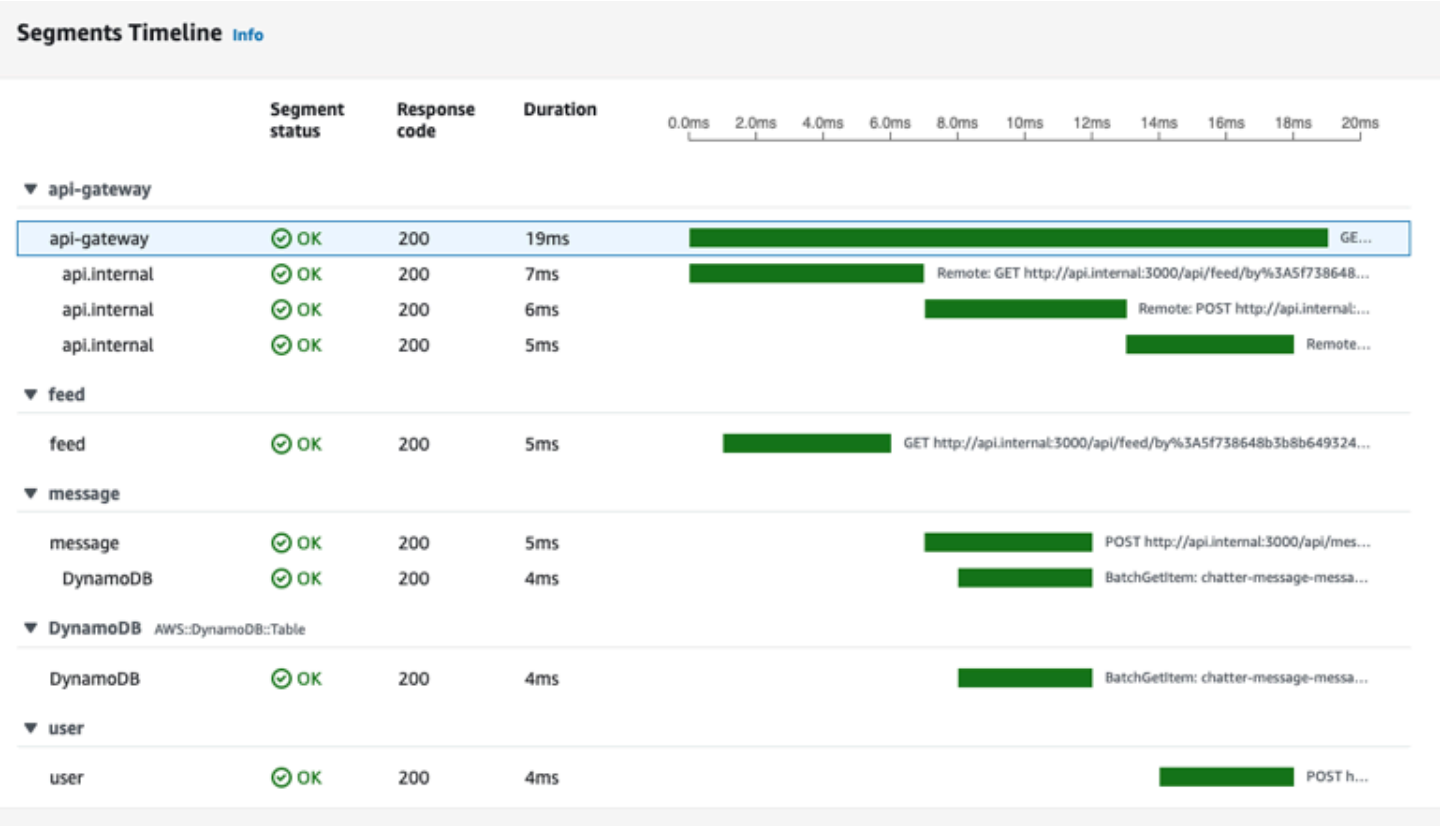
CloudWatch Container Insights mengumpulkan, mengumpulkan, dan merangkum metrik dan log dari aplikasi dan layanan mikro dalam kontainer Anda. Metrik mencakup pemanfaatan sumber daya seperti CPU, memori, disk, dan jaringan. Mereka tersedia di dasbor CloudWatch otomatis. Untuk informasi selengkapnya, lihat Menyiapkan Wawasan Kontainer di Amazon ECS di CloudWatch Panduan Pengguna Amazon.

AWS X-Ray

AWS X-Ray adalah layanan penelusuran yang dapat Anda gunakan untuk mengumpulkan informasi tentang permintaan jaringan yang dibuat aplikasi Anda. Anda dapat menggunakan SDK untuk instrumen aplikasi Anda dan menangkap timing dan kode respons lalu lintas antara layanan Anda, dan antara layanan dan titik akhir AWS layanan Anda. Untuk informasi selengkapnya, lihat Apa yang ada AWS X-Ray di Panduan AWS X-Ray Pengembang.



Anda juga dapat menjelajahi AWS X-Ray grafik tentang bagaimana jaringan layanan Anda satu sama lain. Atau, gunakan mereka untuk mengeksplorasi statistik agregat tentang bagaimana kinerja setiap service-to-service tautan. Terakhir, Anda dapat menyelam lebih dalam ke transaksi tertentu untuk melihat bagaimana segmen yang mewakili panggilan jaringan terkait dengan transaksi tertentu.



Anda dapat menggunakan fitur-fitur ini untuk mengidentifikasi apakah ada hambatan jaringan atau jika layanan tertentu dalam jaringan Anda tidak berfungsi seperti yang diharapkan.

Log Alur VPC

Anda dapat menggunakan log aliran VPC Amazon untuk menganalisis performa jaringan dan masalah konektivitas debug. Dengan log aliran VPC diaktifkan, Anda dapat menangkap log semua koneksi di VPC Anda. Ini termasuk koneksi ke antarmuka jaringan yang terkait dengan Elastic Load Balancing, Amazon RDS, gateway NAT, dan layanan AWS utama lainnya yang mungkin Anda gunakan. Untuk informasi selengkapnya, lihat Log Alur VPC di Panduan Pengguna Amazon VPC.

Kiat penyetelan jaringan

Ada beberapa pengaturan yang dapat Anda selaraskan untuk meningkatkan jaringan Anda.

nofile ulimit

Jika Anda mengharapkan aplikasi Anda memiliki lalu lintas tinggi dan menangani banyak koneksi bersamaan, Anda harus mempertimbangkan kuota sistem untuk jumlah file yang diizinkan. Ketika ada banyak soket jaringan terbuka, masing-masing harus diwakili oleh deskriptor file. Jika kuota deskriptor file Anda terlalu rendah, itu akan membatasi soket jaringan Anda. Ini mengakibatkan koneksi atau kesalahan yang gagal. Anda dapat memperbarui kuota khusus penampung untuk jumlah file dalam definisi tugas Amazon ECS. Jika Anda menjalankan Amazon EC2 (bukan AWS Fargate), Anda mungkin juga perlu menyesuaikan kuota ini pada instans Amazon EC2 yang mendasarinya.

sysctl bersih

Kategori lain dari pengaturan yang dapat disetel adalah pengaturan `sysctl` bersih. Anda harus merujuk ke pengaturan khusus untuk distribusi Linux pilihan Anda. Banyak dari pengaturan ini menyesuaikan ukuran buffer baca dan tulis. Ini dapat membantu dalam beberapa situasi saat menjalankan instans Amazon EC2 skala besar yang memiliki banyak kontainer di dalamnya.

Praktik Terbaik - Penskalaan otomatis dan manajemen kapasitas

Amazon ECS digunakan untuk menjalankan beban kerja aplikasi kontainer dari semua ukuran. Ini termasuk lingkungan pengujian minimal yang ekstrem dan lingkungan produksi besar yang beroperasi pada skala global.

Dengan Amazon ECS, seperti semua AWS layanan, Anda hanya membayar untuk apa yang Anda gunakan. Ketika dirancang dengan tepat, Anda dapat menghemat biaya dengan meminta aplikasi Anda hanya mengkonsumsi sumber daya yang dibutuhkan pada saat dibutuhkan. Panduan praktik terbaik ini menunjukkan cara menjalankan beban kerja Amazon ECS Anda dengan cara yang memenuhi tujuan tingkat layanan Anda sambil tetap beroperasi dengan cara yang hemat biaya.

Topik

Menentukan ukuran tugas

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.

Mengkonfigurasi penskalaan otomatis layanan

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. Kami membahas cara memilih metrik di bagian berikut.

Karakterisasi aplikasi Anda

Penskalaan aplikasi dengan benar membutuhkan mengetahui kondisi di mana aplikasi harus diskalakan dan kapan harus diskalakan. Intinya, aplikasi harus ditingkatkan jika permintaan diperkirakan melebihi kapasitas. Sebaliknya, aplikasi dapat ditingkatkan untuk menghemat biaya ketika sumber daya melebihi permintaan.

Mengidentifikasi metrik pemanfaatan

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, Anda harus mulai dengan membangun 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 CloudWatch Wawasan Kontainer. Atau, lakukan dengan menggunakan Amazon Managed Service untuk Prometheus bersama dengan Amazon Managed Grafana. Selama proses ini, pastikan Anda mengumpulkan dan memplot metrik untuk waktu respons aplikasi atau tingkat penyelesaian pekerjaan Anda.

Saat pengujian beban, 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, periksa mana yang 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 uji 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 untuk itu. Dan, itu juga dapat diskalakan secara horizontal dengan menambahkan lebih banyak replika. Menambahkan lebih banyak replika, atau menggandakan ukuran instans, memotong penggunaan CPU rata-rata 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 kustom 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.

Kapasitas dan ketersediaan

Ketersediaan aplikasi sangat penting untuk memberikan pengalaman bebas kesalahan dan untuk meminimalkan latensi aplikasi. Ketersediaan tergantung pada memiliki sumber daya yang dapat diakses dan memiliki kapasitas yang cukup untuk memenuhi permintaan. AWS menyediakan beberapa mekanisme untuk mengelola ketersediaan. Untuk aplikasi yang dihosting di Amazon ECS, ini termasuk penskalaan otomatis dan Availability Zones (AZ). Penskalaan otomatis mengelola jumlah tugas atau instance berdasarkan metrik yang Anda tentukan, sementara Availability Zone memungkinkan Anda untuk meng-host aplikasi Anda di lokasi yang terisolasi namun dekat secara geografis.

Seperti halnya ukuran tugas, kapasitas dan ketersediaan menyajikan trade-off tertentu yang harus Anda pertimbangkan. Idealnya, kapasitas akan sangat selaras dengan permintaan. Akan selalu ada kapasitas yang cukup untuk melayani permintaan dan memproses pekerjaan untuk memenuhi Tujuan Tingkat Layanan (SLO) termasuk latensi rendah dan tingkat kesalahan. Kapasitas tidak akan pernah terlalu tinggi, menyebabkan biaya yang berlebihan; juga tidak akan pernah terlalu rendah, yang mengarah ke latensi tinggi dan tingkat kesalahan.

Autoscaling adalah proses laten. Pertama, metrik real-time harus dikirimkan ke CloudWatch. Kemudian, mereka perlu dikumpulkan untuk analisis, yang dapat memakan waktu hingga beberapa menit tergantung pada granularitas metrik. CloudWatch membandingkan metrik dengan ambang alarm untuk mengidentifikasi kekurangan atau kelebihan sumber daya. Untuk mencegah ketidakstabilan, konfigurasikan alarm agar ambang batas yang disetel dilintasi selama beberapa menit sebelum alarm berbunyi. Hal ini juga membutuhkan waktu untuk menyediakan tugas-tugas baru dan untuk mengakhiri tugas-tugas yang tidak lagi diperlukan.

Karena potensi penundaan dalam sistem yang dijelaskan ini, penting bagi Anda untuk mempertahankan beberapa ruang kepala dengan penyediaan berlebihan. Melakukan hal ini dapat membantu mengakomodasi ledakan permintaan jangka pendek. Ini juga membantu aplikasi Anda untuk melayani permintaan tambahan tanpa mencapai kejenuhan. Sebagai praktik yang baik, Anda dapat menetapkan target penskalaan Anda antara 60-80% pemanfaatan. Ini membantu aplikasi Anda menangani semburan permintaan ekstra dengan lebih baik sementara kapasitas tambahan masih dalam proses penyediaan.

Alasan lain kami menyarankan Anda untuk menyediakan berlebihan adalah agar Anda dapat dengan cepat menanggapi kegagalan Availability Zone. AWS merekomendasikan agar beban kerja produksi dilayani dari beberapa Availability Zone. Ini karena, jika terjadi kegagalan Availability Zone, tugas Anda yang berjalan di Availability Zone yang tersisa masih dapat memenuhi permintaan. Jika aplikasi

Anda berjalan di dua Availability Zones, Anda perlu menggandakan jumlah tugas normal Anda. Ini agar Anda dapat memberikan kapasitas langsung selama potensi kegagalan. Jika aplikasi Anda berjalan di tiga Availability Zones, kami sarankan Anda menjalankan 1,5 kali jumlah tugas normal Anda. Artinya, jalankan tiga tugas untuk setiap dua tugas yang dibutuhkan untuk penyajian biasa.

Memaksimalkan kecepatan penskalaan

Autoscaling adalah proses reaktif yang membutuhkan waktu untuk diterapkan. Namun, ada beberapa cara untuk membantu meminimalkan waktu yang diperlukan untuk meningkatkan skala.

Minimalkan ukuran gambar. Gambar yang lebih besar membutuhkan waktu lebih lama untuk diunduh dari repositori gambar dan membongkar. Oleh karena itu, menjaga ukuran gambar lebih kecil mengurangi jumlah waktu yang dibutuhkan wadah untuk memulai. Untuk mengurangi ukuran gambar, Anda dapat mengikuti rekomendasi spesifik ini:

- Jika Anda dapat membangun biner statis atau menggunakan Golang, buat `FROM` goresan gambar Anda dan sertakan hanya aplikasi biner Anda dalam gambar yang dihasilkan.
- Gunakan gambar dasar yang diminimalkan dari vendor distro hulu, seperti Amazon Linux atau Ubuntu.
- Jangan sertakan artefak build apa pun di gambar akhir Anda. Menggunakan build multi-tahap dapat membantu dalam hal ini.
- `RUN`Tahapan kompak sedapat mungkin. Setiap `RUN` tahap menciptakan layer gambar baru, yang mengarah ke perjalanan pulang pergi tambahan untuk mengunduh layer. Satu `RUN` tahap yang memiliki beberapa perintah bergabung dengan && memiliki lebih sedikit lapisan daripada satu dengan beberapa `RUN` tahap.
- Jika Anda ingin menyertakan data, seperti data inferensi ML, dalam gambar akhir Anda, sertakan hanya data yang diperlukan untuk memulai dan mulai melayani lalu lintas. Jika Anda mengambil data sesuai permintaan dari Amazon S3 atau penyimpanan lain tanpa memengaruhi layanan, maka simpan data Anda di tempat tersebut.

Jaga agar gambar Anda tetap dekat. Semakin tinggi latensi jaringan, semakin lama waktu yang dibutuhkan untuk mengunduh gambar. Host gambar Anda di repositori di AWS Wilayah yang sama dengan beban kerja Anda. Amazon ECR adalah repositori gambar berkinerja tinggi yang tersedia di setiap Wilayah tempat Amazon ECS tersedia. Hindari melintasi Internet atau tautan VPN untuk mengunduh gambar kontainer. Hosting gambar Anda di Wilayah yang sama meningkatkan keandalan secara keseluruhan. Ini mengurangi risiko masalah konektivitas jaringan dan masalah

ketersediaan di Wilayah yang berbeda. Atau, Anda juga dapat menerapkan replikasi lintas wilayah Amazon ECR untuk membantu hal ini.

Kurangi ambang batas pemeriksaan kesehatan penyeimbang beban. Load balancer melakukan pemeriksaan kesehatan sebelum mengirim lalu lintas ke aplikasi Anda. Konfigurasi pemeriksaan kesehatan default untuk grup target dapat memakan waktu 90 detik atau lebih lama. Selama ini, penyeimbang beban memeriksa status kesehatan dan menerima permintaan. Menurunkan interval pemeriksaan kesehatan dan jumlah ambang batas dapat membuat aplikasi Anda menerima lalu lintas lebih cepat dan mengurangi beban pada tugas lain.

Pertimbangkan kinerja start dingin. Beberapa aplikasi menggunakan runtime seperti Java melakukan kompilasi Just-In-Time (JIT). Proses kompilasi setidaknya saat dimulai dapat menunjukkan kinerja aplikasi. Solusinya adalah menulis ulang bagian penting latensi dari beban kerja Anda dalam bahasa yang tidak memberlakukan penalti kinerja awal yang dingin.

Gunakan penskalaan langkah, bukan kebijakan penskalaan pelacakan target. 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 dilintasi. Dan, yang lebih penting, Anda dapat bereaksi sangat cepat terhadap perubahan permintaan dengan meminimalkan jumlah waktu alarm ambang batas dilanggar. Untuk informasi lebih lanjut, lihat Service Auto Scaling dalam Panduan Pengembang Layanan Amazon Elastic Container.

Jika Anda menggunakan instans Amazon EC2 untuk menyediakan kapasitas klaster, pertimbangkan rekomendasi berikut:

Gunakan instans Amazon EC2 yang lebih besar dan volume Amazon EBS yang lebih cepat. Anda dapat meningkatkan kecepatan pengunduhan dan persiapan gambar dengan menggunakan instans Amazon EC2 yang lebih besar dan volume Amazon EBS yang lebih cepat. Dalam rangkaian instans Amazon EC2 tertentu, throughput maksimum jaringan dan Amazon EBS meningkat seiring bertambahnya ukuran instans (misalnya, dari ke). `m5.xlarge` `m5.2xlarge` Selain itu, Anda juga dapat menyesuaikan volume Amazon EBS untuk meningkatkan throughput dan IOPS mereka. Misalnya, jika Anda menggunakan `gp2` volume, gunakan volume yang lebih besar yang menawarkan lebih banyak throughput dasar. Jika Anda menggunakan `gp3` volume, tentukan throughput dan IOPS saat Anda membuat volume.

Gunakan mode jaringan jembatan untuk tugas yang berjalan di instans Amazon EC2. Tugas yang menggunakan mode `bridge` jaringan di Amazon EC2 dimulai lebih cepat daripada tugas yang menggunakan mode `awsvpc` jaringan. Saat mode `awsvpc` jaringan digunakan, Amazon ECS melampirkan elastic network interface (ENI) ke instance sebelum meluncurkan tugas. Ini memperkenalkan latensi tambahan. Ada beberapa pengorbanan untuk menggunakan jaringan jembatan sekalipun. Tugas-tugas ini tidak mendapatkan grup keamanan mereka sendiri, dan ada beberapa implikasi untuk load balancing. Untuk informasi selengkapnya, lihat Grup target penyeimbang beban di Panduan Pengguna Elastic Load Balancing.

Menangani guncangan permintaan

Beberapa aplikasi mengalami guncangan besar yang tiba-tiba dalam permintaan. Ini terjadi karena berbagai alasan: acara berita, penjualan besar, acara media, atau peristiwa lain yang menjadi viral dan menyebabkan lalu lintas meningkat dengan cepat dan signifikan dalam rentang waktu yang sangat singkat. Jika tidak direncanakan, ini dapat menyebabkan permintaan dengan cepat melampaui sumber daya yang tersedia.

Cara terbaik untuk menangani guncangan permintaan adalah dengan mengantisipasi dan merencanakannya dengan tepat. Karena penskalaan otomatis dapat memakan waktu, kami menyarankan Anda untuk menskalakan aplikasi Anda sebelum kejutan permintaan dimulai. Untuk hasil terbaik, kami sarankan memiliki rencana bisnis yang melibatkan kolaborasi erat antara tim yang menggunakan kalender bersama. Tim yang merencanakan acara harus bekerja sama dengan tim yang bertanggung jawab atas aplikasi terlebih dahulu. Ini memberi tim itu cukup waktu untuk memiliki rencana penjadwalan yang jelas. Mereka dapat menjadwalkan kapasitas untuk skala sebelum acara dan untuk skala setelah acara. Untuk informasi lebih lanjut, lihat Penskalaan terjadwal dalam Panduan Pengguna Application Auto Scaling.

Jika Anda memiliki paket Enterprise Support, pastikan juga untuk bekerja dengan Technical Account Manager (TAM) Anda. TAM Anda dapat memverifikasi kuota layanan Anda dan memastikan bahwa kuota yang diperlukan dinaikkan sebelum acara dimulai. Dengan cara ini, Anda tidak secara tidak sengaja mencapai kuota layanan apa pun. Mereka juga dapat membantu Anda dengan melakukan pemanasan awal seperti penyeimbang beban untuk memastikan acara Anda berjalan lancar.

Menangani guncangan permintaan yang tidak terjadwal adalah masalah yang lebih sulit. Guncangan tak terjadwal, jika amplitudo cukup besar, dapat dengan cepat menyebabkan permintaan melebihi kapasitas. Ini juga dapat melampaui kemampuan autoscaling untuk bereaksi. Cara terbaik untuk mempersiapkan guncangan yang tidak terjadwal adalah dengan menyediakan sumber daya yang

berlebihan. Anda harus memiliki sumber daya yang cukup untuk menangani permintaan lalu lintas maksimum yang diantisipasi setiap saat.

Mempertahankan kapasitas maksimum untuk mengantisipasi guncangan permintaan yang tidak terjadwal bisa mahal. Untuk mengurangi dampak biaya, temukan metrik indikator utama atau peristiwa yang memprediksi guncangan permintaan besar akan segera terjadi. Jika metrik atau peristiwa secara andal memberikan pemberitahuan awal yang signifikan, segera mulailah proses penskalaan saat peristiwa terjadi atau saat metrik melewati ambang batas tertentu yang Anda tetapkan.

Jika aplikasi Anda rentan terhadap guncangan permintaan tak terjadwal yang tiba-tiba, pertimbangkan untuk menambahkan mode kinerja tinggi ke aplikasi Anda yang mengorbankan fungsionalitas non-kritis tetapi mempertahankan fungsionalitas penting bagi pelanggan. Misalnya, asumsikan bahwa aplikasi Anda dapat beralih dari menghasilkan respons khusus yang mahal menjadi menyajikan halaman respons statis. Dalam skenario ini, Anda dapat meningkatkan throughput secara signifikan tanpa menskalakan aplikasi sama sekali.

Terakhir, Anda dapat mempertimbangkan untuk memecah layanan monolitik untuk mengatasi guncangan permintaan dengan lebih baik. Jika aplikasi Anda adalah layanan monolitik yang mahal untuk dijalankan dan lambat untuk skala, Anda mungkin dapat mengekstrak atau menulis ulang bagian penting kinerja dan menjalankannya sebagai layanan terpisah. Layanan baru ini kemudian dapat diskalakan secara independen dari komponen yang kurang kritis. Memiliki fleksibilitas untuk meningkatkan fungsionalitas penting kinerja secara terpisah dari bagian lain dari aplikasi Anda dapat mengurangi waktu yang diperlukan untuk menambah kapasitas dan membantu menghemat biaya.

Kapasitas cluster

Anda dapat menyediakan kapasitas ke cluster Amazon ECS dalam beberapa cara. Misalnya, Anda dapat meluncurkan instans Amazon EC2 dan mendaftarkannya ke cluster saat start-up menggunakan agen penampung Amazon ECS. Namun, metode ini bisa menjadi tantangan karena Anda perlu mengelola penskalaan sendiri. Oleh karena itu, kami menyarankan Anda menggunakan penyedia kapasitas Amazon ECS. Mereka mengelola penskalaan sumber daya untuk Anda. Ada tiga jenis penyedia kapasitas: Amazon EC2, Fargate, dan Fargate Spot. Untuk informasi selengkapnya tentang penyedia kapasitas Fargate, lihat klaster Amazon ECS untuk beban kerja tipe peluncuran Fargate dan untuk jenis peluncuran EC2, lihat klaster Amazon ECS untuk beban kerja tipe peluncuran EC2 di Panduan Pengembang Layanan Kontainer Elastis Amazon.

Penyedia kapasitas Fargate dan Fargate Spot menangani siklus hidup tugas Fargate untuk Anda. Fargate menyediakan kapasitas sesuai permintaan, dan Fargate Spot menyediakan kapasitas Spot.

Saat tugas diluncurkan, ECS menyediakan sumber daya Fargate untuk Anda. Sumber daya Fargate ini dilengkapi dengan memori dan unit CPU yang secara langsung sesuai dengan batas tingkat tugas yang Anda nyatakan dalam definisi tugas Anda. Setiap tugas menerima sumber daya Fargate sendiri, membuat hubungan 1:1 antara tugas dan sumber daya komputasi.

Tugas yang berjalan di Fargate Spot dapat mengalami gangguan. Interupsi datang setelah peringatan dua menit. Ini terjadi selama periode permintaan yang tinggi. Fargate Spot bekerja paling baik untuk beban kerja yang toleran interupsi seperti pekerjaan batch, pengembangan, atau lingkungan pementasan. Mereka juga cocok untuk skenario lain di mana ketersediaan tinggi dan latensi rendah bukanlah persyaratan.

Anda dapat menjalankan tugas Fargate Spot bersama tugas Fargate sesuai permintaan. Dengan menggunakannya bersama-sama, Anda menerima kapasitas “burst” penyediaan dengan biaya lebih rendah.

ECS juga dapat mengelola kapasitas instans Amazon EC2 untuk tugas Anda. Setiap Penyedia Kapasitas Amazon EC2 dikaitkan dengan Grup Auto Scaling Amazon EC2 yang Anda tentukan. Saat Anda menggunakan Penyedia Kapasitas Amazon EC2, penskalaan otomatis cluster ECS mempertahankan ukuran Grup Auto Scaling Amazon EC2 untuk memastikan semua tugas terjadwal dapat ditempatkan.

Memilih ukuran tugas Fargate

Untuk definisi AWS Fargate tugas, Anda harus menentukan CPU dan memori pada tingkat tugas, dan tidak perlu memperhitungkan overhead apa pun. Anda juga dapat menentukan CPU dan memori pada tingkat kontainer untuk tugas Fargate. Namun, melakukannya tidak diperlukan. Batas sumber daya harus lebih besar dari atau sama dengan reservasi apa pun yang Anda nyatakan. Dalam kebanyakan kasus, Anda dapat mengaturnya ke jumlah reservasi dari setiap kontainer yang dideklarasikan dalam definisi tugas Anda. Kemudian, bulatkan angka ke ukuran Fargate terdekat. Untuk informasi selengkapnya tentang ukuran yang tersedia, lihat CPU Tugas dan memori di Panduan Pengembang Layanan Amazon Elastic Container.

Mempercepat penyediaan kapasitas cluster dengan penyedia kapasitas di Amazon EC2

Pelanggan yang menjalankan Amazon ECS di Amazon EC2 dapat memanfaatkan Amazon ECS Cluster Auto Scaling (CAS) untuk mengelola penskalaan grup Auto Scaling Amazon EC2 (ASG).

Dengan CAS, Anda dapat mengonfigurasi Amazon ECS untuk menskalakan ASG Anda secara otomatis, dan hanya fokus menjalankan tugas Anda. Amazon ECS akan memastikan skala ASG masuk dan keluar sesuai kebutuhan tanpa intervensi lebih lanjut yang diperlukan. Penyedia kapasitas Amazon ECS digunakan untuk mengelola infrastruktur di klaster Anda dengan memastikan ada cukup instance kontainer untuk memenuhi permintaan aplikasi Anda. Untuk mempelajari cara kerja Amazon ECS CAS, lihat Deep Dive di Auto Scaling Amazon ECS Cluster.

Karena CAS bergantung pada integrasi CloudWatch berbasis dengan ASG untuk menyesuaikan kapasitas cluster, ia memiliki latensi bawaan yang terkait dengan penerbitan CloudWatch metrik, waktu yang dibutuhkan metrik `CapacityProviderReservation` untuk melanggar CloudWatch alarm (tinggi dan rendah), dan waktu yang dibutuhkan oleh instans Amazon EC2 yang baru diluncurkan untuk pemanasan. Anda dapat mengambil tindakan berikut untuk membuat Amazon ECS CAS lebih responsif untuk penerapan yang lebih cepat:

Ukuran penskalaan langkah penyedia kapasitas

Penyedia kapasitas Amazon ECS pada akhirnya akan menumbuhkan/mengecilkan instans kontainer untuk memenuhi permintaan aplikasi Anda. Jumlah minimum instans yang akan diluncurkan Amazon ECS diatur ke 1 secara default. Ini dapat menambah waktu tambahan untuk penerapan Anda, jika beberapa instance diperlukan untuk menempatkan tugas Anda yang tertunda. Anda dapat meningkatkan `minimumScalingStepSize`melalui Amazon ECS API untuk meningkatkan jumlah minimum instans yang diskalakan atau dikeluarkan Amazon ECS sekaligus. A `maximumScalingStepSize`yang terlalu rendah dapat membatasi berapa banyak instance kontainer yang diskalakan masuk atau keluar sekaligus, yang dapat memperlambat penerapan Anda.

> **Note**
>
> Konfigurasi ini saat ini hanya tersedia melalui `UpdateCapacityProvider`API `CreateCapacityProvider`atau.

Periode pemanasan contoh

Periode pemanasan instans adalah periode waktu setelah instans Amazon EC2 yang baru diluncurkan dapat berkontribusi pada metrik CloudWatch untuk grup Auto Scaling. Setelah periode pemanasan yang ditentukan berakhir, instance dihitung terhadap metrik agregat ASG, dan CAS melanjutkan dengan iterasi perhitungan berikutnya untuk memperkirakan jumlah instance yang diperlukan.

Nilai `instanceWarmupPeriod`defaultnya adalah 300 detik, yang dapat Anda konfigurasikan ke nilai yang lebih rendah melalui `UpdateCapacityProvider`API `CreateCapacityProvider`atau untuk penskalaan yang lebih responsif.

Kapasitas cadangan

Jika penyedia kapasitas Anda tidak memiliki instans kontainer yang tersedia untuk menempatkan tugas, maka perlu meningkatkan (skala) kapasitas klaster dengan meluncurkan instans Amazon EC2 dengan cepat, dan menunggu hingga boot sebelum dapat meluncurkan kontainer pada mereka. Ini dapat secara signifikan menurunkan tingkat peluncuran tugas. Anda memiliki dua opsi di sini.

Dalam hal ini, memiliki kapasitas Amazon EC2 cadangan yang sudah diluncurkan dan tugas yang siap dijalankan akan meningkatkan tingkat peluncuran tugas yang efektif. Anda dapat menggunakan `Target Capacity` konfigurasi untuk menunjukkan bahwa Anda ingin mempertahankan kapasitas cadangan di cluster Anda. Misalnya, dengan menetapkan `Target Capacity` 80%, Anda menunjukkan bahwa cluster Anda membutuhkan kapasitas cadangan 20% setiap saat. Kapasitas cadangan ini dapat memungkinkan tugas mandiri apa pun segera diluncurkan, memastikan peluncuran tugas tidak dibatasi. Trade-off untuk pendekatan ini adalah potensi peningkatan biaya untuk menjaga kapasitas cluster cadangan.

Pendekatan alternatif yang dapat Anda pertimbangkan adalah menambahkan ruang kepala ke layanan Anda, bukan ke penyedia kapasitas. Ini berarti bahwa alih-alih mengurangi `Target Capacity` konfigurasi untuk meluncurkan kapasitas cadangan, Anda dapat meningkatkan jumlah replika di layanan Anda dengan memodifikasi metrik penskalaan pelacakan target atau ambang batas penskalaan langkah dari penskalaan otomatis layanan. Perhatikan bahwa pendekatan ini hanya akan membantu untuk beban kerja yang runcing, tetapi tidak akan berpengaruh saat Anda menerapkan layanan baru dan beralih dari 0 ke N tugas untuk pertama kalinya. Untuk informasi selengkapnya tentang kebijakan penskalaan terkait, lihat Kebijakan Penskalaan Pelacakan Target atau Kebijakan Penskalaan Langkah di Panduan Pengembang Layanan Amazon Elastic Container.

Memilih jenis instans Amazon EC2

Jika Anda menggunakan Amazon EC2 untuk menyediakan kapasitas untuk cluster ECS, Anda dapat memilih dari banyak pilihan jenis instans. Semua jenis instans Amazon EC2 dan keluarga kompatibel dengan ECS.

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. 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 pada saat yang 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 agar 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 setiap grup. Terakhir, dalam strategi Penyedia Kapasitas layanan dan tugas Anda, Anda dapat memilih Penyedia Kapasitas yang paling sesuai dengan kebutuhan Anda. Untuk informasi selengkapnya, lihat Jenis instans di Panduan Pengguna Amazon EC2 untuk Instans Linux.

Menggunakan Amazon EC2 Spot dan FARGATE_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.

- Pertama, selama periode permintaan yang sangat tinggi, kapasitas Spot mungkin tidak tersedia. Hal ini dapat menyebabkan tugas Fargate Spot dan peluncuran instans Amazon EC2 Spot tertunda. Dalam peristiwa ini, layanan ECS mencoba lagi meluncurkan tugas, dan grup Auto Scaling Amazon EC2 juga mencoba lagi meluncurkan instans, hingga kapasitas yang diperlukan tersedia. Fargate dan Amazon EC2 tidak menggantikan kapasitas Spot dengan kapasitas sesuai permintaan.
- Kedua, ketika permintaan kapasitas secara keseluruhan 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.

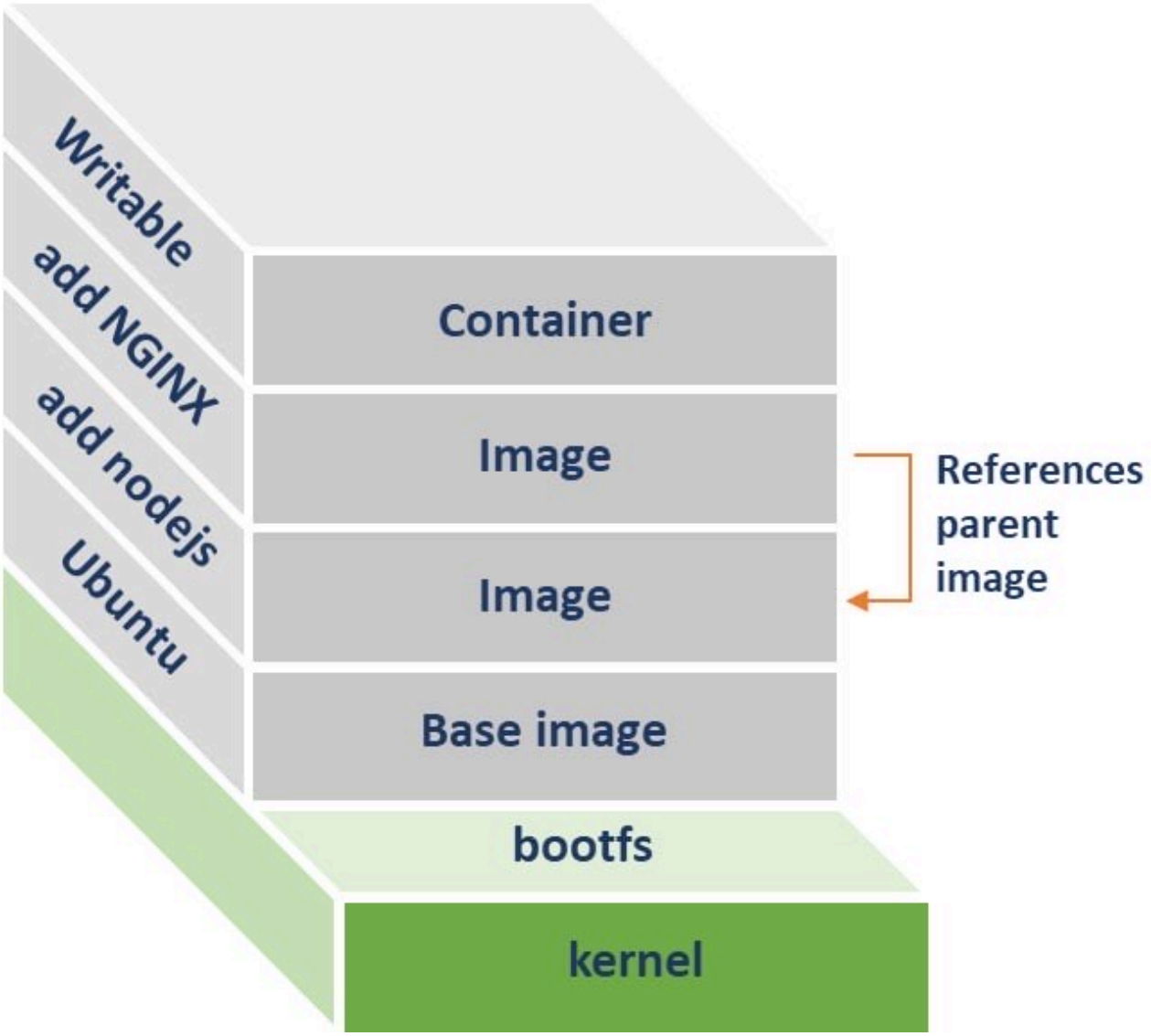
Untuk membantu meminimalkan kekurangan kapasitas Spot, pertimbangkan rekomendasi berikut:

- 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 Anda. 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 yang dioptimalkan kapasitas dan biaya. 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.

Praktik Terbaik - Penyimpanan persisten

Anda dapat menggunakan Amazon ECS untuk menjalankan aplikasi kontainer stateful dalam skala besar dengan menggunakan layanan AWS penyimpanan, seperti Amazon EFS, Amazon EBS, atau FSx for Windows File Server, yang menyediakan persistensi data ke wadah sementara yang inheren. Istilah ketekunan data berarti bahwa data itu sendiri bertahan lebih lama dari proses yang menciptakannya. Persistensi data dalam AWS dicapai dengan memisahkan layanan komputasi dan penyimpanan. Mirip dengan Amazon EC2, Anda juga dapat menggunakan Amazon ECS untuk memisahkan siklus hidup aplikasi kontainer Anda dari data yang mereka konsumsi dan hasilkan. Menggunakan layanan AWS penyimpanan, tugas Amazon ECS dapat mempertahankan data bahkan setelah tugas dihentikan.

Secara default, kontainer tidak menyimpan data yang mereka hasilkan. Ketika sebuah wadah dihentikan, data yang ditulisnya ke lapisan yang dapat ditulis akan dihancurkan dengan wadah. Ini membuat wadah cocok untuk aplikasi stateless yang tidak perlu menyimpan data secara lokal. Aplikasi kontainer yang memerlukan persistensi data memerlukan backend penyimpanan yang tidak dihancurkan saat wadah aplikasi berakhir.



Gambar kontainer dibangun dari serangkaian lapisan. Setiap lapisan mewakili instruksi di Dockerfile tempat gambar dibuat. Setiap lapisan hanya-baca, kecuali untuk wadah. Artinya, ketika Anda membuat wadah, lapisan yang dapat ditulis ditambahkan di atas lapisan yang mendasarinya. File apa pun yang dibuat, dihapus, atau dimodifikasi oleh wadah ditulis ke lapisan yang dapat ditulis. Ketika wadah berakhir, lapisan yang dapat ditulis juga dihapus secara bersamaan. Wadah baru yang menggunakan gambar yang sama memiliki layer yang dapat ditulis sendiri. Lapisan ini tidak menyertakan perubahan apa pun. Oleh karena itu, data kontainer harus selalu disimpan di luar lapisan container yang dapat ditulis.

Dengan Amazon ECS, Anda dapat menjalankan kontainer stateful menggunakan volume dalam empat cara. Amazon ECS untuk wadah Linux terintegrasi dengan Amazon EFS secara asli.

Amazon ECS untuk tugas Windows dan Linux yang dihosting pada instance container atau di Fargate terintegrasi dengan Amazon EBS secara native. Volume Amazon EBS yang dilampirkan ke tugas mandiri dapat dipertahankan dengan menyetelnya. `deleteOnTermination` `false` Instans container Windows dan Linux Amazon EC2 juga dapat menggunakan volume Docker yang terintegrasi dengan Amazon EBS. Untuk wadah Windows, Amazon ECS terintegrasi dengan FSx for Windows File Server untuk menyediakan penyimpanan persisten.

Topik

Memilih jenis penyimpanan yang tepat untuk wadah Anda

Aplikasi yang berjalan di kluster Amazon ECS dapat menggunakan berbagai layanan AWS penyimpanan dan produk pihak ketiga untuk menyediakan penyimpanan persisten untuk beban kerja stateful. Anda harus memilih backend penyimpanan untuk aplikasi kontainer berdasarkan arsitektur dan persyaratan penyimpanan aplikasi Anda. Untuk informasi selengkapnya tentang layanan AWS penyimpanan, lihat Cloud Storage aktif AWS.

Untuk cluster Amazon ECS yang berisi instance Linux atau wadah Linux yang digunakan dengan Fargate, Amazon ECS terintegrasi dengan Amazon EFS untuk menyediakan penyimpanan kontainer. Perbedaan paling khas antara Amazon EFS dan Amazon EBS adalah Anda dapat secara bersamaan memasang sistem file Amazon EFS pada ribuan tugas Amazon ECS. Sebaliknya, volume Amazon EBS tidak mendukung akses bersamaan. Mengingat hal ini, Amazon EFS adalah opsi penyimpanan yang direkomendasikan untuk aplikasi kontainer yang diskalakan secara horizontal. Ini karena mendukung konkurensi. Amazon EFS menyimpan data Anda secara berlebihan di beberapa Availability Zone dan menawarkan akses latensi rendah dari tugas Amazon ECS, terlepas dari Availability Zone.

Misalkan Anda memiliki aplikasi seperti database transaksional yang membutuhkan latensi sub-milidetik dan tidak memerlukan sistem file bersama ketika diskalakan secara horizontal. Untuk aplikasi semacam itu, kami sarankan menggunakan volume Amazon EBS untuk penyimpanan persisten. Amazon ECS mendukung volume Amazon EBS untuk tugas yang dihosting di Amazon EC2 dan Fargate. Sebelum menggunakan volume Amazon EBS dengan tugas Amazon ECS, Anda

dapat melampirkan volume Amazon EBS ke instans penampung dan mengelola volume secara terpisah dari siklus hidup tugas, atau Anda dapat mengonfigurasi volume Amazon EBS untuk lampiran ke tugas Amazon ECS selama penerapan layanan atau tugas mandiri.

Untuk cluster yang berisi instance Windows, FSx for Windows File Server menyediakan penyimpanan persisten untuk kontainer. FSx for Windows File Server sistem file mendukung penerapan Multi-AZ. Melalui penerapan ini, Anda dapat berbagi sistem file dengan tugas Amazon ECS yang berjalan di beberapa Availability Zone.

Anda juga dapat menggunakan penyimpanan instans Amazon EC2 untuk persistensi data untuk tugas Amazon ECS yang dihosting di Amazon EC2 menggunakan pemasangan pengikat atau volume Docker. Saat menggunakan bind mount atau volume Docker, container menyimpan data pada sistem file instance container. Salah satu batasan menggunakan sistem file host untuk penyimpanan kontainer adalah bahwa data hanya tersedia pada satu instance kontainer pada satu waktu. Ini berarti bahwa kontainer hanya dapat berjalan di host tempat data berada. Oleh karena itu, menggunakan penyimpanan host hanya disarankan dalam skenario di mana replikasi data ditangani di tingkat aplikasi.

Volume Amazon EFS

Amazon Elastic File System (Amazon EFS) menyediakan sistem file NFS elastis yang sederhana, dapat diskalakan, dan dikelola sepenuhnya. Ini dibangun untuk dapat menskalakan sesuai permintaan ke petabyte tanpa mengganggu aplikasi. Ini dapat menskalakan masuk atau keluar saat Anda menambahkan dan menghapus file.

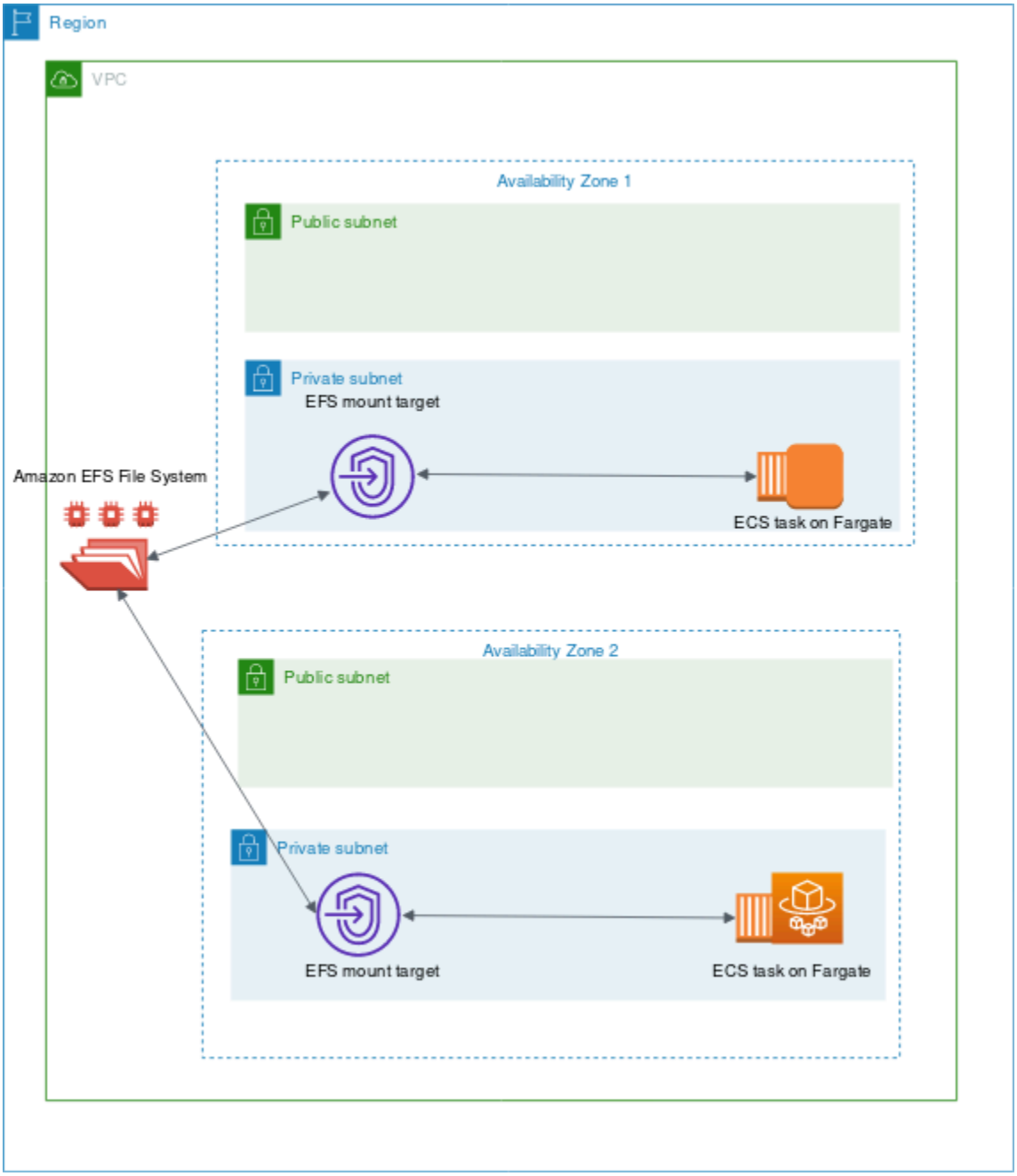
Anda dapat menjalankan aplikasi stateful Anda di Amazon ECS dengan menggunakan volume Amazon EFS untuk menyediakan penyimpanan persisten. Tugas Amazon ECS yang berjalan di instans Amazon EC2 atau di Fargate menggunakan `1.4.0` versi platform dan yang lebih baru dapat memasang sistem file Amazon EFS yang ada. Mengingat bahwa beberapa kontainer dapat memasang dan mengakses sistem file Amazon EFS secara bersamaan, tugas Anda memiliki akses ke kumpulan data yang sama di mana pun mereka di-host.

Untuk memasang sistem file Amazon EFS di container, Anda dapat mereferensikan sistem file Amazon EFS dan titik pemasangan container dalam definisi tugas Amazon ECS. Berikut ini adalah cuplikan definisi tugas yang menggunakan Amazon EFS untuk penyimpanan kontainer.

```
...
"containerDefinitions": [
```

```
  {
    "mountPoints": [
      {
        "containerPath": "/opt/my-app",
        "sourceVolume": "Shared-EFS-Volume"
      }
    ]
  }
]
...
"volumes": [
    {
      "efsVolumeConfiguration": {
        "fileSystemId": "fs-1234",
        "transitEncryption": "DISABLED",
        "rootDirectory": ""
      },
      "name": "Shared-EFS-Volume"
    }
  ]
```

Amazon EFS menyimpan data secara berlebihan di beberapa Availability Zone dalam satu Wilayah. Tugas Amazon ECS memasang sistem file Amazon EFS dengan menggunakan target pemasangan Amazon EFS di Availability Zone. Tugas Amazon ECS hanya dapat memasang sistem file Amazon EFS jika sistem file Amazon EFS memiliki target pemasangan di Availability Zone tempat tugas berjalan. Oleh karena itu, praktik terbaik adalah membuat target pemasangan Amazon EFS di semua Availability Zone yang Anda rencanakan untuk menampung tugas Amazon ECS.



Untuk informasi selengkapnya, lihat volume Amazon EFS di Panduan Pengembang Layanan Kontainer Elastis Amazon.

Kontrol keamanan dan akses

Amazon EFS menawarkan fitur kontrol akses yang dapat Anda gunakan untuk memastikan bahwa data yang disimpan dalam sistem file Amazon EFS aman dan hanya dapat diakses dari aplikasi yang membutuhkannya. Anda dapat mengamankan data dengan mengaktifkan enkripsi saat istirahat dan

dalam perjalanan. Untuk informasi selengkapnya, lihat Enkripsi data di Amazon EFS di Panduan Pengguna Amazon Elastic File System.

Selain enkripsi data, Anda juga dapat menggunakan Amazon EFS untuk membatasi akses ke sistem file. Ada tiga cara untuk menerapkan kontrol akses di EFS.

- Grup keamanan —Dengan target pemasangan Amazon EFS, Anda dapat mengonfigurasi grup keamanan yang digunakan untuk mengizinkan dan menolak lalu lintas jaringan. Anda dapat mengonfigurasi grup keamanan yang dilampirkan ke Amazon EFS untuk mengizinkan lalu lintas NFS (port 2049) dari grup keamanan yang dilampirkan ke instans Amazon ECS Anda atau, saat menggunakan mode `awsvpc` jaringan, tugas Amazon ECS.
- IAM —Anda dapat membatasi akses ke sistem file Amazon EFS menggunakan IAM. Saat dikonfigurasi, tugas Amazon ECS memerlukan peran IAM untuk akses sistem file untuk memasang sistem file EFS. Untuk informasi selengkapnya, lihat Menggunakan IAM untuk mengontrol akses data sistem file di Panduan Pengguna Amazon Elastic File System.

  Kebijakan IAM juga dapat menerapkan kondisi yang telah ditentukan sebelumnya seperti mengharuskan klien untuk menggunakan TLS saat menyambung ke sistem file Amazon EFS. Untuk informasi selengkapnya, lihat kunci kondisi Amazon EFS untuk klien di Panduan Pengguna Amazon Elastic File System.
- Jalur akses Amazon EFS —Jalur akses Amazon EFS adalah titik masuk khusus aplikasi ke dalam sistem file Amazon EFS. Anda dapat menggunakan titik akses untuk menegakkan identitas pengguna, termasuk grup POSIX pengguna, untuk 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 sub-direktorinya.

Pertimbangkan untuk menerapkan ketiga kontrol akses pada sistem file Amazon EFS untuk keamanan maksimum. Misalnya, Anda dapat mengonfigurasi grup keamanan yang dilampirkan ke titik pemasangan Amazon EFS untuk hanya mengizinkan lalu lintas NFS masuk dari grup keamanan yang terkait dengan instance container atau tugas Amazon ECS Anda. Selain itu, Anda dapat mengonfigurasi Amazon EFS agar memerlukan peran IAM untuk mengakses sistem file, meskipun koneksi berasal dari grup keamanan yang diizinkan. Terakhir, Anda dapat menggunakan jalur akses Amazon EFS untuk menerapkan izin pengguna POSIX dan menentukan direktori root untuk aplikasi.

Cuplikan definisi tugas berikut menunjukkan cara memasang sistem file Amazon EFS menggunakan titik akses.

```
"volumes": [
```

```
    {
      "efsVolumeConfiguration": {
        "fileSystemId": "fs-1234",
        "authorizationConfig": {
          "acessPointId": "fsap-1234",
          "iam": "ENABLED"
        },
        "transitEncryption": "ENABLED",
        "rootDirectory": ""
      },
      "name": "my-filesystem"
    }
]
```

Kinerja

Amazon EFS menawarkan dua mode kinerja: Tujuan Umum dan Maks I/O. Tujuan Umum cocok untuk aplikasi yang sensitif terhadap latensi seperti sistem manajemen konten dan alat CI/CD. Sebaliknya, sistem file Max I/O cocok untuk beban kerja seperti analitik data, pemrosesan media, dan pembelajaran mesin. Beban kerja ini perlu melakukan operasi paralel dari ratusan atau bahkan ribuan kontainer dan membutuhkan throughput agregat dan IOPS setinggi mungkin. Untuk informasi selengkapnya, lihat mode kinerja Amazon EFS di Panduan Pengguna Amazon Elastic File System.

Beberapa beban kerja sensitif latensi memerlukan tingkat I/O yang lebih tinggi yang disediakan oleh mode kinerja Max I/O dan latensi yang lebih rendah yang disediakan oleh mode kinerja Tujuan Umum. Untuk jenis beban kerja ini, kami sarankan untuk membuat beberapa sistem file mode kinerja Tujuan Umum. Dengan begitu, Anda dapat menyebarkan beban kerja aplikasi Anda di semua sistem file ini, selama beban kerja dan aplikasi dapat mendukungnya.

Throughput

Semua sistem file Amazon EFS memiliki throughput terukur terkait yang ditentukan oleh jumlah throughput yang disediakan untuk sistem file yang menggunakan Provisioned Throughput atau jumlah data yang disimpan dalam kelas penyimpanan EFS Standard atau One Zone untuk sistem file yang menggunakan Bursting Throughput. Untuk informasi selengkapnya, lihat Memahami throughput terukur di Panduan Pengguna Amazon Elastic File System.

Mode throughput default untuk sistem file Amazon EFS adalah mode bursting. Dengan mode bursting, throughput yang tersedia untuk sistem file masuk atau keluar saat sistem file tumbuh. Karena beban kerja berbasis file biasanya melonjak, membutuhkan tingkat throughput yang tinggi

untuk periode waktu dan tingkat throughput yang lebih rendah sepanjang waktu, Amazon EFS dirancang untuk meledak untuk memungkinkan tingkat throughput yang tinggi untuk periode waktu tertentu. Selain itu, karena banyak beban kerja yang berat baca, operasi baca diukur pada rasio 1:3 terhadap operasi NFS lainnya (seperti menulis).

Semua sistem file Amazon EFS memberikan kinerja dasar yang konsisten sebesar 50 MB/s untuk setiap TB penyimpanan Amazon EFS Standard atau Amazon EFS One Zone. Semua sistem file (terlepas dari ukurannya) dapat meledak hingga 100 MB/s. Sistem file dengan penyimpanan EFS Standard atau EFS One Zone lebih dari 1TB dapat meledak hingga 100 MB/s untuk setiap TB. Karena operasi baca diukur pada rasio 1:3, Anda dapat mendorong hingga 300 MiBs /s untuk setiap TiB throughput baca. Saat Anda menambahkan data ke sistem file, throughput maksimum yang tersedia untuk sistem file akan diskalakan secara linier dan otomatis dengan penyimpanan Anda di kelas penyimpanan Amazon EFS Standard. Jika Anda membutuhkan lebih banyak throughput daripada yang dapat Anda capai dengan jumlah data yang disimpan, Anda dapat mengonfigurasi Throughput yang Disediakan ke jumlah tertentu yang dibutuhkan beban kerja Anda.

Throughput sistem file dibagikan di semua instans Amazon EC2 yang terhubung ke sistem file. Misalnya, sistem file 1TB yang dapat meledak hingga 100 MB/s throughput dapat mendorong 100 MB/s dari satu instans Amazon EC2 dapat masing-masing drive 10 MB/s. Untuk informasi selengkapnya, lihat Performa Amazon EFS di Panduan Pengguna Amazon Elastic File System.

Optimasi Biaya

Amazon EFS menyederhanakan penyimpanan penskalaan untuk Anda. Sistem file Amazon EFS tumbuh secara otomatis saat Anda menambahkan lebih banyak data. Terutama dengan mode Amazon EFS Bursting Throughput, throughput di Amazon EFS meningkat seiring dengan bertambahnya ukuran sistem file Anda di kelas penyimpanan standar. Untuk meningkatkan throughput tanpa membayar biaya tambahan untuk throughput yang disediakan pada sistem file EFS, Anda dapat berbagi sistem file Amazon EFS dengan beberapa aplikasi. Menggunakan jalur akses Amazon EFS, Anda dapat menerapkan isolasi penyimpanan dalam sistem file Amazon EFS bersama. Dengan demikian, meskipun aplikasi masih berbagi sistem file yang sama, mereka tidak dapat mengakses data kecuali Anda mengotorisasi itu.

Seiring pertumbuhan data Anda, Amazon EFS membantu Anda memindahkan file yang jarang diakses secara otomatis ke kelas penyimpanan yang lebih rendah. Kelas penyimpanan Amazon EFS Standard-Infrequent Access (IA) mengurangi biaya penyimpanan untuk file yang tidak diakses setiap hari. Ini dilakukan tanpa mengorbankan ketersediaan tinggi, daya tahan tinggi, elastisitas, dan

akses sistem file POSIX yang disediakan Amazon EFS. Untuk informasi selengkapnya, lihat kelas penyimpanan Amazon EFS di Panduan Pengguna Amazon Elastic File System.

Pertimbangkan untuk menggunakan kebijakan siklus hidup Amazon EFS untuk menghemat uang secara otomatis dengan memindahkan file yang jarang diakses ke penyimpanan Amazon EFS IA. Untuk informasi selengkapnya, lihat Manajemen siklus hidup Amazon EFS di Panduan Pengguna Amazon Elastic File System.

Saat membuat sistem file Amazon EFS, Anda dapat memilih apakah Amazon EFS mereplikasi data Anda di beberapa Availability Zone (Standar) atau menyimpan data Anda secara berlebihan dalam satu Availability Zone. Kelas penyimpanan Amazon EFS One Zone dapat mengurangi biaya penyimpanan dengan margin yang signifikan dibandingkan dengan kelas penyimpanan Amazon EFS Standard. Pertimbangkan untuk menggunakan kelas penyimpanan Amazon EFS One Zone untuk beban kerja yang tidak memerlukan ketahanan Multi-AZ. Anda dapat mengurangi biaya penyimpanan Amazon EFS One Zone dengan memindahkan file yang jarang diakses ke Amazon EFS One Zone-Infrequent Access. Untuk informasi selengkapnya, lihat Amazon EFS Infrequent Access.

Perlindungan data

Amazon EFS menyimpan data Anda secara berlebihan di beberapa Availability Zone untuk sistem file yang menggunakan kelas penyimpanan Standar. Jika Anda memilih kelas penyimpanan Amazon EFS One Zone, data Anda disimpan secara berlebihan dalam satu Availability Zone. Selain itu, Amazon EFS dirancang untuk memberikan daya tahan 99,999999999% (11 9) selama tahun tertentu.

Seperti halnya lingkungan apa pun, ini adalah praktik terbaik untuk memiliki cadangan dan membangun perlindungan terhadap penghapusan yang tidak disengaja. Untuk data Amazon EFS, praktik terbaik tersebut mencakup penggunaan cadangan yang berfungsi dan diuji secara teratur AWS Backup. Sistem file yang menggunakan kelas penyimpanan Amazon EFS One Zone dikonfigurasi untuk secara otomatis mencadangkan file secara default pada pembuatan sistem file kecuali Anda memilih untuk menonaktifkan fungsi ini. Untuk informasi selengkapnya, lihat Perlindungan data untuk Amazon EFS di Panduan Pengguna Amazon Elastic File System.

Kasus penggunaan

Amazon EFS menyediakan akses bersama paralel yang secara otomatis tumbuh dan menyusut saat file ditambahkan dan dihapus. Hasilnya, Amazon EFS cocok untuk aplikasi apa pun yang memerlukan penyimpanan dengan fungsionalitas seperti latensi rendah, throughput tinggi, dan konsistensi. read-after-write Amazon EFS adalah backend penyimpanan yang ideal untuk aplikasi

yang menskalakan secara horizontal dan memerlukan sistem file bersama. Beban kerja seperti analisis data, pemrosesan media, manajemen konten, dan penayangan web adalah beberapa kasus penggunaan Amazon EFS yang umum.

Salah satu kasus penggunaan di mana Amazon EFS mungkin tidak cocok adalah untuk aplikasi yang memerlukan latensi sub-milidetik. Ini umumnya merupakan persyaratan untuk sistem database transaksional. Sebaiknya jalankan pengujian kinerja penyimpanan untuk menentukan dampak penggunaan Amazon EFS untuk aplikasi sensitif latensi. Jika kinerja aplikasi menurun saat menggunakan Amazon EFS, pertimbangkan Amazon EBS io2 Block Express, yang menyediakan latensi I/O sub-milidetik, varians rendah pada instans Nitro. Untuk informasi selengkapnya, lihat Jenis volume Amazon EBS di Panduan Pengguna Amazon EC2 untuk Instans Linux.

Beberapa aplikasi gagal jika penyimpanan dasarnya diubah secara tak terduga. Oleh karena itu, Amazon EFS bukanlah pilihan terbaik untuk aplikasi ini. Sebaliknya, Anda mungkin lebih suka menggunakan sistem penyimpanan yang tidak memungkinkan akses bersamaan dari beberapa tempat.

Volume Docker

Volume Docker adalah fitur runtime kontainer Docker yang memungkinkan kontainer menyimpan data dengan memasang direktori dari sistem file host. Driver volume Docker (juga disebut sebagai plugin) digunakan untuk mengintegrasikan volume kontainer dengan sistem penyimpanan eksternal, seperti Amazon EBS. Volume Docker hanya didukung saat menghosting tugas Amazon ECS di instans Amazon EC2.

Tugas Amazon ECS dapat menggunakan volume Docker untuk mempertahankan data menggunakan volume Amazon EBS. Ini dilakukan dengan melampirkan volume Amazon EBS ke instans Amazon EC2 dan kemudian memasang volume dalam tugas menggunakan volume Docker. Volume Docker dapat dibagikan di antara beberapa tugas Amazon ECS di host.

Keterbatasan volume Docker adalah bahwa sistem file yang digunakan tugas terkait dengan instans Amazon EC2 tertentu. Jika instance berhenti karena alasan apa pun dan tugas ditempatkan pada instance lain, data akan hilang. Anda dapat menetapkan tugas ke instans untuk memastikan volume EBS terkait selalu tersedia untuk tugas.

Untuk informasi selengkapnya, lihat volume Docker di Panduan Pengembang Layanan Amazon Elastic Container.

Siklus hidup volume Amazon EBS

Ada dua pola penggunaan utama dengan penyimpanan kontainer dan Amazon EBS. Yang pertama adalah ketika aplikasi perlu mempertahankan data dan mencegah kehilangan data ketika kontainer berakhir. Contoh dari jenis aplikasi ini akan menjadi database transaksional seperti MySQL. Ketika tugas MySQL berakhir, tugas lain diharapkan untuk menggantikannya. Dalam skenario ini, siklus hidup volume terpisah dari siklus hidup tugas. Saat menggunakan EBS untuk mempertahankan data kontainer, praktik terbaik adalah menggunakan batasan penempatan tugas untuk membatasi penempatan tugas ke satu host dengan volume EBS terpasang.

Yang kedua adalah ketika siklus hidup volume independen dari siklus hidup tugas. Ini sangat berguna untuk aplikasi yang membutuhkan penyimpanan berkinerja tinggi dan latensi rendah tetapi tidak perlu menyimpan data setelah tugas berakhir. Misalnya, beban kerja ETL yang memproses volume data yang besar mungkin memerlukan penyimpanan throughput yang tinggi. Amazon EBS cocok untuk jenis beban kerja ini karena menyediakan volume kinerja tinggi yang menyediakan hingga 256.000 IOPS. Saat tugas berakhir, replika pengganti dapat ditempatkan dengan aman di host Amazon EC2 mana pun di cluster. Selama tugas memiliki akses ke backend penyimpanan yang dapat memenuhi persyaratan kinerjanya, tugas tersebut dapat menjalankan fungsinya. Oleh karena itu, tidak ada kendala penempatan tugas yang diperlukan dalam kasus ini.

Jika instans Amazon EC2 di klaster Anda memiliki beberapa jenis volume Amazon EBS yang dilampirkan padanya, Anda dapat menggunakan batasan penempatan tugas untuk memastikan bahwa tugas ditempatkan pada instans dengan volume Amazon EBS yang sesuai terpasang. Misalnya, sebuah cluster memiliki beberapa instance dengan `gp2` volume, sementara yang lain menggunakan `io1` volume. Anda dapat melampirkan atribut kustom ke instance dengan `io1` volume dan kemudian menggunakan batasan penempatan tugas untuk memastikan tugas intensif I/O Anda selalu ditempatkan pada instance kontainer dengan volume. `io1`

AWS CLI Perintah berikut digunakan untuk menempatkan atribut pada instance container Amazon ECS.

```
aws ecs put-attributes \
     --attributes name=EBS,value=io1,targetId=<your-container-instance-arn>
```

Ketersediaan data Amazon EBS

Kontainer biasanya berumur pendek, sering dibuat, dan dihentikan saat aplikasi masuk dan keluar secara horizontal. Sebagai praktik terbaik, Anda dapat menjalankan beban kerja di beberapa Availability Zone untuk meningkatkan ketersediaan aplikasi Anda. Amazon ECS menyediakan cara

bagi Anda untuk mengontrol penempatan tugas menggunakan strategi penempatan tugas dan batasan penempatan tugas. Saat beban kerja mempertahankan datanya menggunakan volume Amazon EBS, tugasnya harus ditempatkan di Availability Zone yang sama dengan volume Amazon EBS. Kami juga menyarankan agar Anda menetapkan batasan penempatan yang membatasi Availability Zone tempat tugas dapat ditempatkan. Ini memastikan bahwa tugas Anda dan volume yang sesuai selalu berada di Availability Zone yang sama.

Saat menjalankan tugas mandiri, Anda dapat mengontrol Zona Ketersediaan mana tugas ditempatkan dengan menetapkan batasan penempatan menggunakan atribut zona ketersediaan.

```
attribute:ecs.availability-zone == us-east-1a
```

Saat menjalankan aplikasi yang akan mendapat manfaat dari berjalan di beberapa Availability Zone, pertimbangkan untuk membuat layanan Amazon ECS yang berbeda untuk setiap Availability Zone. Ini memastikan bahwa tugas yang membutuhkan volume Amazon EBS selalu ditempatkan di Availability Zone yang sama dengan volume terkait.

Sebaiknya buat instance container di setiap Availability Zone, melampirkan volume Amazon EBS menggunakan template peluncuran, dan menambahkan atribut kustom ke instans untuk membedakannya dari instance container lain di cluster Amazon ECS. Saat membuat layanan, konfigurasikan batasan penempatan tugas untuk memastikan bahwa Amazon ECS menempatkan tugas di Availability Zone dan instance yang tepat. Untuk informasi selengkapnya, lihat Contoh batasan penempatan tugas di Panduan Pengembang Layanan Amazon Elastic Container.

Plugin volume Docker

Plugin Docker seperti Portworx menyediakan abstraksi antara volume Docker dan volume Amazon EBS. Plugin ini dapat secara dinamis membuat volume Amazon EBS saat tugas Anda yang membutuhkan volume dimulai. Portworx juga dapat melampirkan volume ke host baru ketika kontainer berakhir, dan replika berikutnya ditempatkan pada instance kontainer yang berbeda. Ini juga mereplikasi data volume setiap kontainer di antara node Amazon ECS dan di seluruh Availability Zones. Untuk informasi lebih lanjut, lihat Portworx.

FSx for Windows File Server

FSx for Windows File Server menyediakan penyimpanan file yang dikelola sepenuhnya, sangat andal, dan dapat diskalakan yang dapat diakses melalui protokol Server Message Block (SMB) standar industri. Ini dibangun di atas Windows Server, memberikan berbagai fitur administratif seperti

kuota pengguna, pemulihan file pengguna akhir, dan integrasi Microsoft Active Directory (AD). Ini menawarkan opsi penyebaran Single-AZ dan multi-AZ, pencadangan yang dikelola sepenuhnya, dan enkripsi data saat istirahat dan dalam perjalanan.

Amazon ECS mendukung penggunaan FSx for Windows File Server dalam definisi tugas Amazon ECS Windows yang memungkinkan penyimpanan persisten sebagai titik pemasangan melalui protokol SMBv3 menggunakan fitur SMB yang disebut. GlobalMappings

Untuk menyiapkan integrasi FSx for Windows File Server dan Amazon ECS, instance container Windows harus menjadi anggota domain pada Layanan Domain Direktori Aktif (AD DS), yang dihosting oleh Active Directory lokal atau Direktori Aktif AWS Directory Service for Microsoft Active Directory yang dihosting sendiri di Amazon EC2. AWS Secrets Manager digunakan untuk menyimpan data sensitif seperti nama pengguna dan kata sandi dari kredensi Active Directory yang digunakan untuk memetakan share pada instance container Windows.

Untuk menggunakan volume sistem file FSx for Windows File Server untuk container Anda, Anda harus menentukan konfigurasi volume dan mount point dalam definisi tugas Anda. Berikut ini adalah cuplikan definisi tugas yang menggunakan FSx for Windows File Server untuk penyimpanan kontainer.

```
{
 "containerDefinitions": [{
  "name": "container-using-fsx",
  "image": "iis:2",
  "entryPoint": [
   "powershell",
   "-command"
  ],
  "mountPoints": [{
   "sourceVolume": "myFsxVolume",
   "containerPath": "\\mount\\fsx",
   "readOnly": false
  }]
 }],
 "volumes": [{
  "fsxWindowsFileServerVolumeConfiguration": {
   "fileSystemId": "fs-ID",
   "authorizationConfig": {
    "domain": "ADDOMAIN.local",
    "credentialsParameter": "arn:aws:secretsmanager:us-
east-1:111122223333:secret:SecretName"
```

```
    },
    "rootDirectory": "share"
   }
  }]
}
```

Untuk informasi selengkapnya, lihat Volume Amazon FSx for Windows File Server dalam Panduan Developer Amazon Elastic Container Service.

Kontrol keamanan dan akses

FSx for Windows File Server menawarkan fitur kontrol akses berikut yang dapat Anda gunakan untuk memastikan bahwa data yang disimpan dalam sistem file FSx for Windows File Server aman dan hanya dapat diakses dari aplikasi yang membutuhkannya.

Enkripsi data

FSx for Windows File Server mendukung dua bentuk enkripsi untuk sistem file. Mereka adalah enkripsi data dalam perjalanan dan enkripsi saat istirahat. Enkripsi data dalam perjalanan didukung pada berbagi file yang dipetakan pada instance kontainer yang mendukung protokol SMB 3.0 atau yang lebih baru. Enkripsi data at rest diaktifkan secara otomatis saat membuat sistem file Amazon FSx. Amazon FSx secara otomatis mengenkripsi data dalam transit menggunakan enkripsi SMB saat Anda mengakses sistem file Anda tanpa perlu memodifikasi aplikasi Anda. Untuk informasi selengkapnya, lihat Enkripsi data di Amazon FSx di Panduan Pengguna Amazon FSx for Windows File Server.

Kontrol akses tingkat folder menggunakan Windows ACL

Instans Windows Amazon EC2 mengakses berbagi file Amazon FSx menggunakan kredensyal Direktori Aktif. Ini menggunakan daftar kontrol akses Windows standar (ACL) untuk kontrol akses tingkat file dan folder berbutir halus. Anda dapat membuat beberapa kredensyal, masing-masing untuk folder tertentu dalam berbagi yang memetakan ke tugas tertentu.

Dalam contoh berikut, tugas memiliki akses ke folder `App01` menggunakan kredensi yang disimpan di Secrets Manager. Nama Sumber Daya Amazon (ARN) adalah. `1234`

```
"rootDirectory": "\\path\\to\\my\\data\App01",
"credentialsParameter": "arn-1234",
"domain": "corp.fullyqualified.com",
```

Dalam contoh lain, tugas memiliki akses ke folder App02 menggunakan kredensi yang disimpan di Secrets Manager. ARN-nya adalah. 6789

```
"rootDirectory": "\\path\\to\\my\\data\App02",
"credentialsParameter": "arn-6789",
"domain": "corp.fullyqualified.com",
```

Kasus penggunaan

Kontainer tidak dirancang untuk menyimpan data. Namun, beberapa aplikasi .NET kontainer mungkin memerlukan folder lokal sebagai penyimpanan persisten untuk menyimpan output aplikasi. FSx for Windows File Server menawarkan folder lokal dalam wadah. Hal ini memungkinkan beberapa kontainer untuk membaca-tulis pada sistem file yang sama yang didukung oleh SMB Share.

Praktik Terbaik - Mempercepat peluncuran tugas

Ada beberapa perbaikan yang dapat Anda lakukan untuk mempersingkat waktu yang dibutuhkan Amazon ECS untuk meluncurkan tugas Anda.

Alur kerja peluncuran Amazon ECS Task

Memahami cara Amazon ECS menyediakan tugas Anda sangat membantu dalam mempertimbangkan pengoptimalan untuk mempercepat peluncuran tugas Anda. Saat Anda meluncurkan tugas Amazon ECS (tugas mandiri atau layanan Amazon ECS), tugas dibuat dan awalnya dimasukkan ke dalam `PROVISIONING` status sebelum berhasil diluncurkan ke `RUNNING` status (untuk detailnya, lihat Siklus hidup tugas di Panduan Pengembang Amazon ECS). Di `PROVISIONING` negara bagian, 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. Jenis peluncurannya adalah AWS Fargate (Fargate) dan Amazon EC2 aktif AWS, dan `EXTERNAL` jenis yang digunakan dengan Amazon ECS Anywhere. Penyedia kapasitas dan strategi penyedia kapasitas dapat digunakan 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-tugas Anda. Untuk Amazon ECS dengan Amazon EC2, Anda dapat mengelola kapasitas klaster dengan mendaftarkan instans Amazon EC2 ke cluster Anda, atau Anda dapat menggunakan Amazon ECS Cluster Auto Scaling (CAS) untuk menyederhanakan manajemen kapasitas komputasi Anda. CAS menangani penskalaan kapasitas cluster Anda secara dinamis, sehingga Anda dapat fokus hanya 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 detail selengkapnya tentang penempatan tugas, lihat penempatan tugas Amazon ECS.

Setelah menemukan kapasitas untuk menempatkan tugas Anda, Amazon ECS menyediakan lampiran yang diperlukan (misalnya Antarmuka Jaringan Elastis (ENI) untuk tugas dalam `awsvpc` mode), dan menggunakan agen penampung Amazon ECS untuk menarik gambar kontainer Anda dan memulai kontainer Anda. Setelah semua ini selesai dan wadah yang relevan telah diluncurkan, Amazon ECS memindahkan tugas ke `RUNNING` status.

Alur kerja Penjadwal Layanan Amazon ECS

Amazon ECS menyediakan penjadwal layanan untuk mengelola status layanan Anda. Penjadwal layanan memastikan bahwa strategi penjadwalan yang Anda tentukan diikuti dan menjadwal ulang tugas yang gagal. Sebagai contoh, jika infrastruktur pokok gagal, penjadwal layanan dapat menjadwal ulang tugas. Tanggung jawab utama penjadwal layanan adalah memastikan bahwa aplikasi Anda selalu menjalankan jumlah tugas yang diinginkan — berdasarkan jumlah yang diinginkan yang Anda tentukan dalam konfigurasi layanan atau jumlah tugas yang diskalakan secara otomatis berdasarkan pemuatan aplikasi jika Anda menggunakan penskalaan otomatis layanan. Penjadwal layanan menggunakan alur kerja asinkron untuk meluncurkan tugas dalam batch. Untuk memahami bagaimana fungsi penjadwal layanan, bayangkan Anda membuat layanan Amazon ECS untuk API web besar yang menerima lalu lintas padat. Anda mengharapkan layanan ini melayani banyak lalu lintas web, dan menentukan bahwa jumlah yang diinginkan yang sesuai untuk layanan ini adalah 1.000 tugas. Saat Anda menerapkan layanan ini, penjadwal layanan Amazon ECS tidak akan meluncurkan semua 1.000 tugas sekaligus. Sebaliknya, ia akan mulai mengeksekusi siklus alur kerja untuk membawa status saat ini (0 tugas) menuju status yang diinginkan (1.000 tugas), dengan setiap siklus alur kerja meluncurkan sekumpulan tugas baru. Penjadwal layanan dapat menyediakan hingga 500 tugas untuk Fargate, Amazon EC2, dan jenis peluncuran Eksternal per layanan per menit. Untuk informasi selengkapnya tentang tarif dan kuota yang diizinkan di Amazon ECS, lihat kuota layanan Amazon ECS.

Sekarang setelah Anda memahami alur kerja peluncuran tugas Amazon ECS, mari kita bahas bagaimana Anda dapat menggunakan beberapa pengetahuan ini untuk mempercepat peluncuran tugas Anda.

Rekomendasi untuk mempercepat peluncuran tugas

Seperti yang dibahas di bagian sebelumnya, waktu yang dibutuhkan antara pemicu peluncuran tugas (melalui Amazon ECS API atau service scheduler) dan keberhasilan start-up container Anda dipengaruhi oleh berbagai faktor dalam Amazon ECS, konfigurasi Anda, dan container itu sendiri. Untuk mempercepat peluncuran tugas Anda, pertimbangkan rekomendasi berikut.

- Cache gambar wadah dan instance binpack.

  Jika Anda menjalankan Amazon ECS di Amazon EC2, Anda dapat mengonfigurasi agen penampung Amazon ECS untuk menyimpan gambar kontainer yang sebelumnya digunakan untuk 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 — 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 manfaat kecepatan penerapan lebih besar daripada manfaat dari `awsvpc` mode, Anda dapat mempertimbangkan untuk menggunakan `bridge` mode untuk mempercepat peluncuran tugas. Untuk membaca lebih lanjut tentang keuntungan relatif dari setiap mode jaringan, lihat the section called “AWSVPC modus” danthe section called “Mode jembatan”.

- Lacak siklus hidup peluncuran tugas Anda untuk menemukan peluang pengoptimalan.

  Seringkali sulit untuk menyadari jumlah waktu yang dibutuhkan aplikasi Anda untuk memulai. Meluncurkan image container Anda, menjalankan skrip start-up, dan konfigurasi lainnya selama start-up aplikasi dapat memakan waktu yang mengejutkan. Anda dapat menggunakan endpoint ECS Agent Metadata untuk memposting metrik untuk melacak waktu start-up 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.

- Pilih jenis instans yang optimal (saat menggunakan Amazon ECS di Amazon EC2).

  Memilih jenis Instance yang benar didasarkan pada reservasi sumber daya (yaitu CPU, Memori, ENI, GPU) yang Anda konfigurasikan pada tugas Anda, oleh karena itu ketika mengukur instance, Anda dapat menghitung berapa banyak tugas yang dapat ditempatkan pada satu instance.

Contoh sederhana dari tugas yang ditempatkan dengan baik, akan menjadi 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.

- Gunakan penjadwal layanan Amazon ECS untuk meluncurkan layanan secara bersamaan.

  Seperti yang dibahas di bagian sebelumnya, penjadwal layanan dapat secara bersamaan meluncurkan tugas untuk beberapa layanan menggunakan alur kerja asinkron. Dengan demikian, Anda dapat mencapai kecepatan penerapan yang lebih cepat dengan merancang aplikasi Anda sebagai layanan yang lebih kecil dengan tugas yang lebih sedikit daripada layanan besar dengan sejumlah besar tugas. Misalnya, alih-alih memiliki satu layanan dengan 1.000 tugas, memiliki 10 layanan masing-masing dengan 100 tugas akan menghasilkan kecepatan penyebaran yang jauh lebih cepat, karena penjadwal layanan akan memulai penyediaan tugas untuk semua layanan secara paralel.

Praktik Terbaik - Mempercepat penerapan

Anda dapat memilih pembaruan bergulir untuk layanan Amazon ECS Anda. Penerapan mungkin memakan waktu lebih lama dari yang Anda harapkan, tetapi Anda dapat memodifikasi beberapa opsi untuk mempercepat penerapan Anda. Untuk konteks, dengan memilih jenis penerapan ini, Anda memberi tahu penjadwal layanan Amazon ECS untuk mengganti 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 web.
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 penerapan 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. Bagian berikut merinci bagaimana Anda dapat memodifikasi semua ini untuk mempercepat penerapan.

Topik

Parameter pemeriksaan kesehatan penyeimbang beban

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 dan interval antara cek.

- `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.

Mengatur parameter pemeriksaan kesehatan Elastic Load Balancing untuk mempercepat penerapan

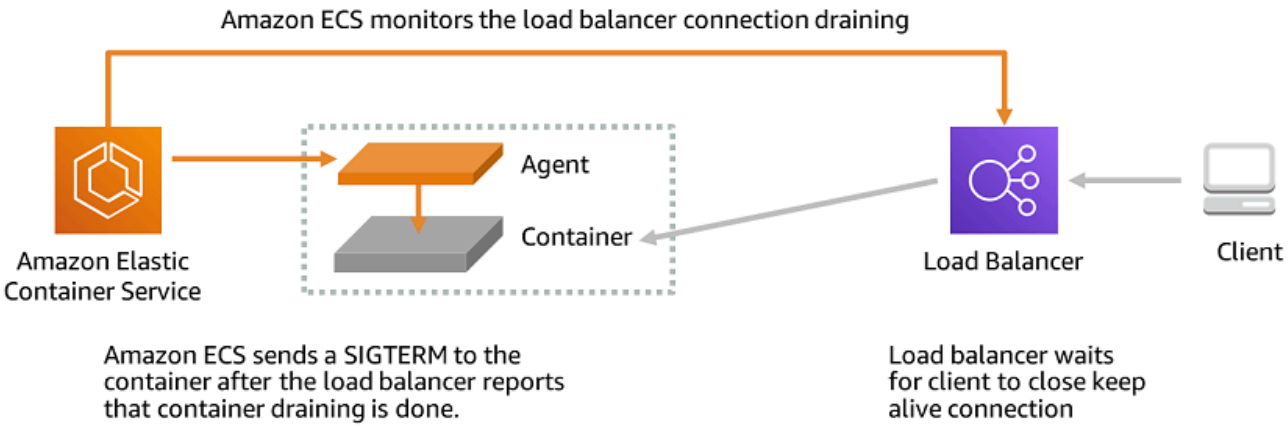
1. Pergi ke https://console.aws.amazon.com/ec2/.
2. Dari navigasi kiri, di bawah Load Balancing, pilih Grup Target.
3. Pada halaman Grup target, pilih grup target.
4. Pada halaman grup target, pilih tab Pemeriksaan Kesehatan.
5. Klik Edit.
6. Perluas Pengaturan pemeriksaan kesehatan lanjutan.
7. Atur parameter pemeriksaan kesehatan.
8. Klik Simpan perubahan.

Untuk informasi selengkapnya tentang parameter pemeriksaan kesehatan Elastic Load Balancing, lihat TargetGroupdi Referensi API Elastic Load Balancing.

Pengeringan koneksi penyeimbang beban

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.

Diagram berikut menjelaskan proses pengeringan koneksi 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 untuk melaporkan bahwa koneksi tetap hidup ditutup.



Jumlah waktu yang menunggu penyeimbang beban adalah penundaan deregistrasi. Anda dapat mengonfigurasi pengaturan penyeimbang beban berikut untuk mempercepat penerapan Anda.

- `deregistration_delay.timeout_seconds`: 300 (default)

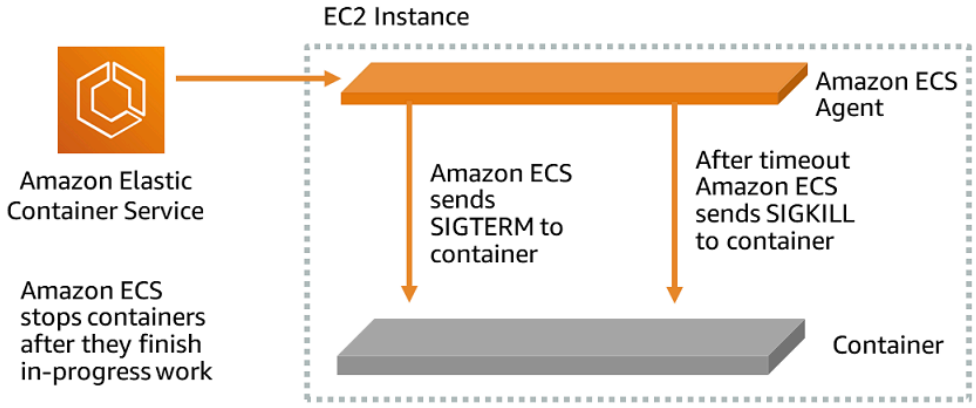
Ketika Anda memiliki layanan dengan waktu respons di bawah satu detik, atur parameter ke nilai berikut agar penyeimbang beban hanya menunggu lima detik sebelum memutuskan koneksi antara klien dan layanan backend:

- `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

Diagram berikut menunjukkan bagaimana Amazon ECS mengakhiri tugas. Amazon ECS pertama-tama mengirimkan sinyal SIGTERM ke tugas untuk memberi tahu aplikasi harus diselesaikan dan dimatikan, dan kemudian Amazon ECS mengirimkan pesan SIGKILL. Ketika aplikasi mengabaikan SIGTERM, layanan Amazon ECS harus menunggu untuk mengirim sinyal SIGKILL untuk menghentikan proses.



Jumlah waktu tunggu Amazon ECS ditentukan oleh opsi agen Amazon ECS berikut:

- `ECS_CONTAINER_STOP_TIMEOUT`: 30 (default)

  Untuk informasi selengkapnya tentang parameter agen kontainer, lihat Konfigurasi agen kontainer di Panduan Pengembang Layanan Kontainer Elastis Amazon.

Untuk mempercepat masa tunggu, atur opsi agen Amazon ECS ke nilai berikut:

> **Note**
>
> Jika aplikasi Anda membutuhkan waktu lebih dari satu detik, kalikan nilainya dengan dua dan gunakan angka itu sebagai nilainya.

- `ECS_CONTAINER_STOP_TIMEOUT`: 2

Dalam hal ini, Amazon ECS menunggu dua 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.

Jenis gambar kontainer

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).

Perilaku tarik gambar kontainer

Perilaku tarik gambar kontainer untuk jenis peluncuran Fargate

Fargate tidak menyimpan gambar, dan oleh karena itu seluruh gambar ditarik dari registri saat tugas berjalan. Berikut ini adalah rekomendasi kami 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.

Perilaku tarik gambar kontainer untuk jenis peluncuran Fargate Windows

Fargate Windows 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:

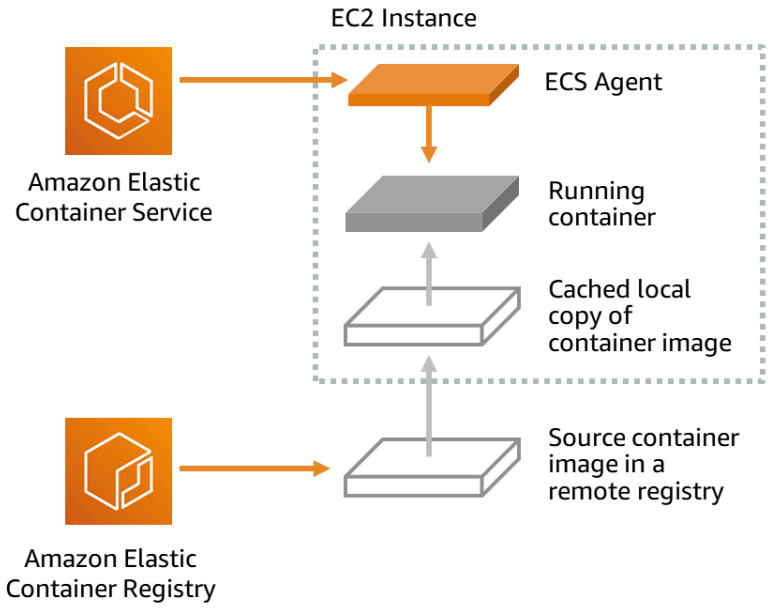
- `mcr.microsoft.com/windows/servercore:ltsc2019`
- `mcr.microsoft.com/windows/servercore:10.0.17763.4737`
- `mcr.microsoft.com/windows/servercore:10.0.17763.4645`

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`

Perilaku tarik gambar kontainer untuk 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 akan 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 dinonaktifkan untuk wadah untuk memastikan bahwa gambar yang di-cache tidak dihapus.

Menyetel parameter ke salah satu nilai sebelumnya dapat menghemat waktu karena agen Amazon ECS menggunakan gambar unduhan yang ada. Untuk gambar Docker yang lebih besar, waktu pengunduhan mungkin memakan waktu 10-20 detik untuk melewati jaringan.

Penyebaran tugas

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 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:

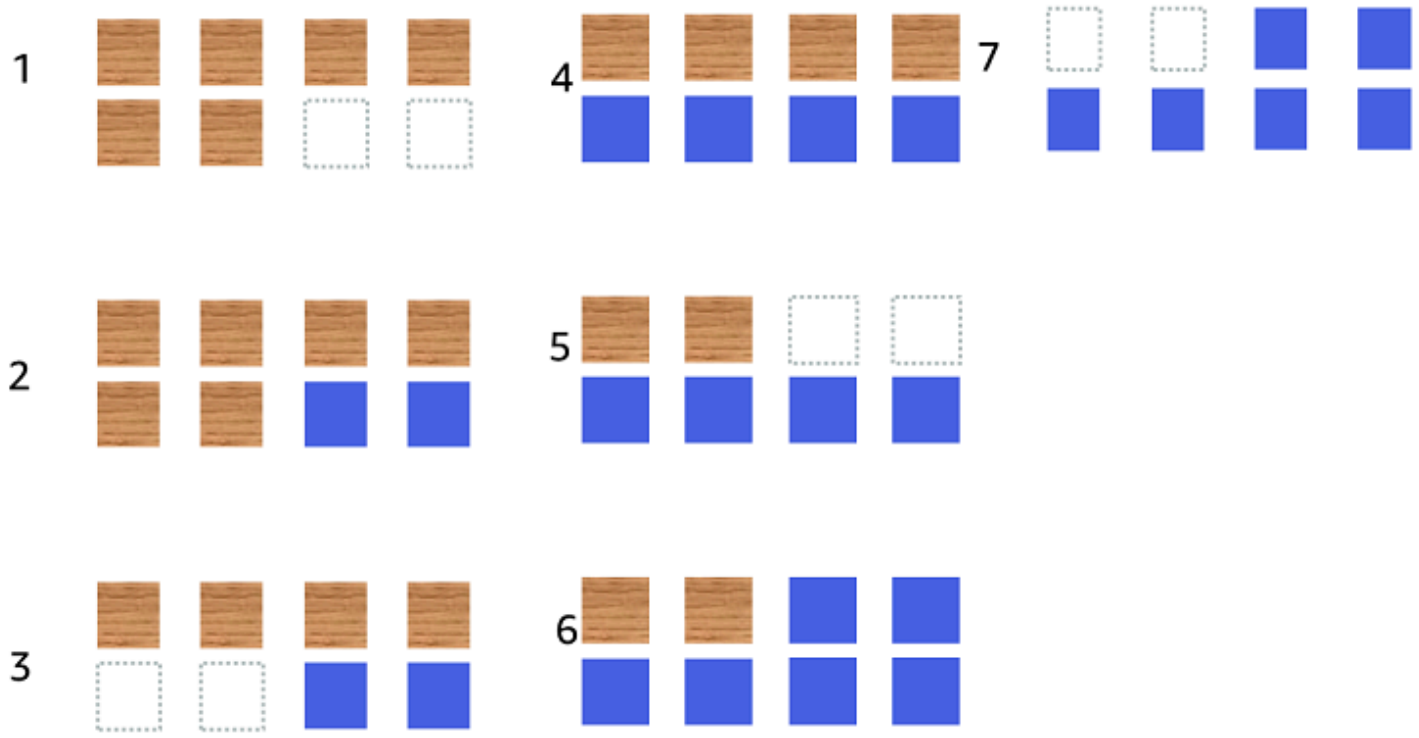
- `minimumHealthyPercent`: 100% (default)

  Batas bawah pada jumlah tugas untuk layanan Anda yang harus tetap dalam `RUNNING` status selama penerapan. Ini direpresentasikan sebagai persentase dari DesiredCount. Itu dibulatkan ke bilangan bulat terdekat. Parameter ini memungkinkan Anda untuk men-deploy tanpa menggunakan kapasitas klaster tambahan.

- `maximumPercent`: 200% (default)

  Batas atas jumlah tugas untuk layanan Anda yang diizinkan di `RUNNING` atau `PENDING` status selama penerapan. Ini direpresentasikan sebagai persentase dari DesiredCount. Ini dibulatkan 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.



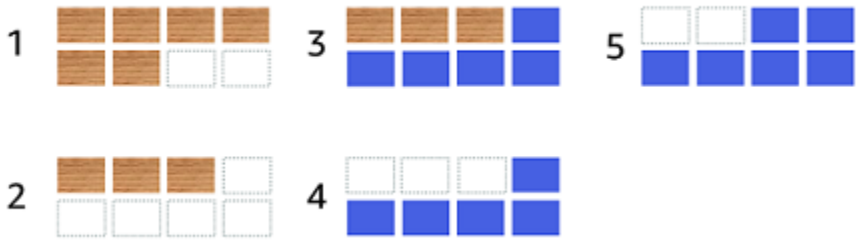
Proses penyebaran adalah sebagai berikut:

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%.

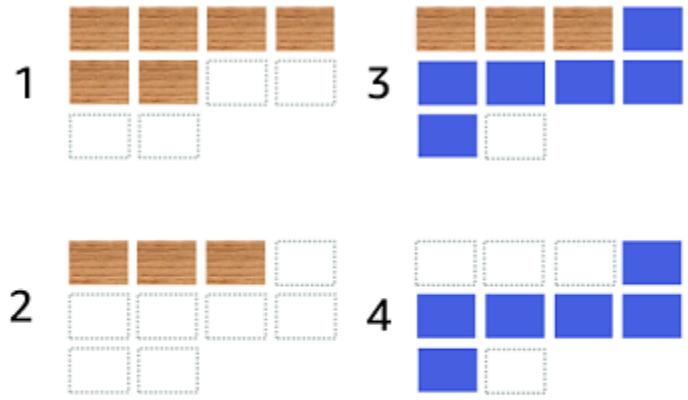
Pertimbangkan layanan berikut yang memiliki enam tugas tan, yang digunakan dalam cluster yang memiliki ruang untuk total delapan tugas.



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 ECS saat tugas Anda menganggur selama beberapa waktu dan tidak memiliki tingkat pemanfaatan yang tinggi.

- `minimumHealthyPercent`: 50%
- `maximumPercent`: 200%

Praktik Terbaik - Mengoperasikan Amazon ECS dalam skala besar

Saat Anda mulai mengoperasikan Amazon ECS dalam skala besar, pertimbangkan bagaimana kuota layanan dan pembatasan API untuk Amazon ECS dan Layanan AWS yang terintegrasi dengan Amazon ECS dapat memengaruhi Anda. Topik ini menjelaskan kuota layanan dan pembatasan API secara rinci, dan juga mencakup pertimbangan penskalaan penting lainnya.

Topik

Kuota layanan 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 penyeimbangan beban layanan, penemuan layanan, jaringan tugas, dan penskalaan otomatis cluster. 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 dari 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 kuota layanan Amazon ECS dan Meminta pembatasan untuk Amazon ECS API.

Saat menskalakan beban kerja Anda di Amazon ECS, kami sarankan Anda mempertimbangkan kuota layanan berikut. Untuk petunjuk tentang cara meminta peningkatan kuota layanan, lihat Mengelola Amazon ECS dan kuota AWS Fargate layanan di. AWS Management Console

- 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 informasi selengkapnya tentang kuota Fargate, lihat kuota layanan di AWS Fargate Panduan Pengguna Layanan Amazon Elastic Container untuk. AWS 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 exponential throttling 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. Bagian berikut mencakup batas tarif utama untuk setiap layanan secara rinci, dan memberikan rekomendasi untuk menangani potensi masalah pelambatan.

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, lihat Pertimbangan penyeimbangan beban layanan dan Parameter pemeriksaan kesehatan penyeimbang beban.

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.

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.

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 pembatasan AWS Cloud Map API Anda. Untuk informasi selengkapnya tentang pemantauan dan pemecahan masalah kesalahan pelambatan tersebut, lihat Menangani masalah pelambatan.

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.

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 Trunking 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.

- 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 menggunakan AWS Cloud Map API untuk mengelola ruang nama untuk layanan Amazon ECS Anda. Jika layanan Anda memiliki banyak tugas, pertimbangkan rekomendasi berikut. Untuk informasi selengkapnya, lihat pertimbangan penemuan layanan Amazon ECS.

AWS Cloud Map kuota layanan

Ketika layanan Amazon ECS dikonfigurasi untuk menggunakan penemuan layanan, `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 secara bersamaan, 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 Menangani masalah pelambatan.

Menangani masalah pelambatan

Bagian ini memberikan ikhtisar mendalam tentang beberapa strategi yang dapat Anda gunakan untuk memantau dan memecahkan masalah kesalahan pelambatan API. 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

Untuk memantau pembatasan, penting untuk mengidentifikasi permintaan API mana yang dibatasi dan siapa yang mengeluarkan permintaan ini. Anda dapat menggunakannya AWS CloudTrail untuk melakukannya. Layanan ini memantau pembatasan, dan dapat diintegrasikan dengan CloudWatch, Amazon Athena, dan Amazon. EventBridge Anda dapat mengonfigurasi CloudTrail untuk mengirim peristiwa tertentu ke CloudWatch Log. Peristiwa ini kemudian diuraikan dan dianalisis menggunakan wawasan CloudWatch log Log. 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 dan petunjuk CloudWatch Log 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 peristiwa tindakan layanan Amazon ECS.

Peristiwa ini dapat dikonsumsi oleh layanan seperti AWS Lambda untuk melakukan tindakan sebagai tanggapan. Untuk informasi selengkapnya, lihat Menangani peristiwa.

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.

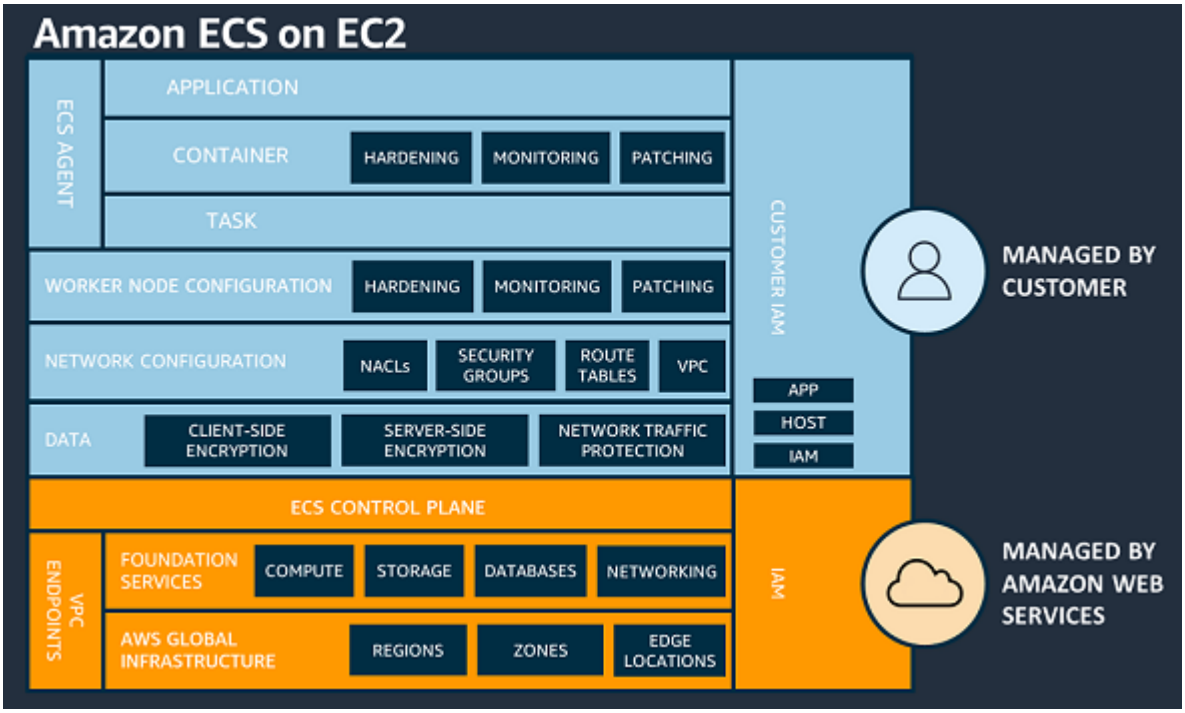
Praktik Terbaik - Keamanan

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.

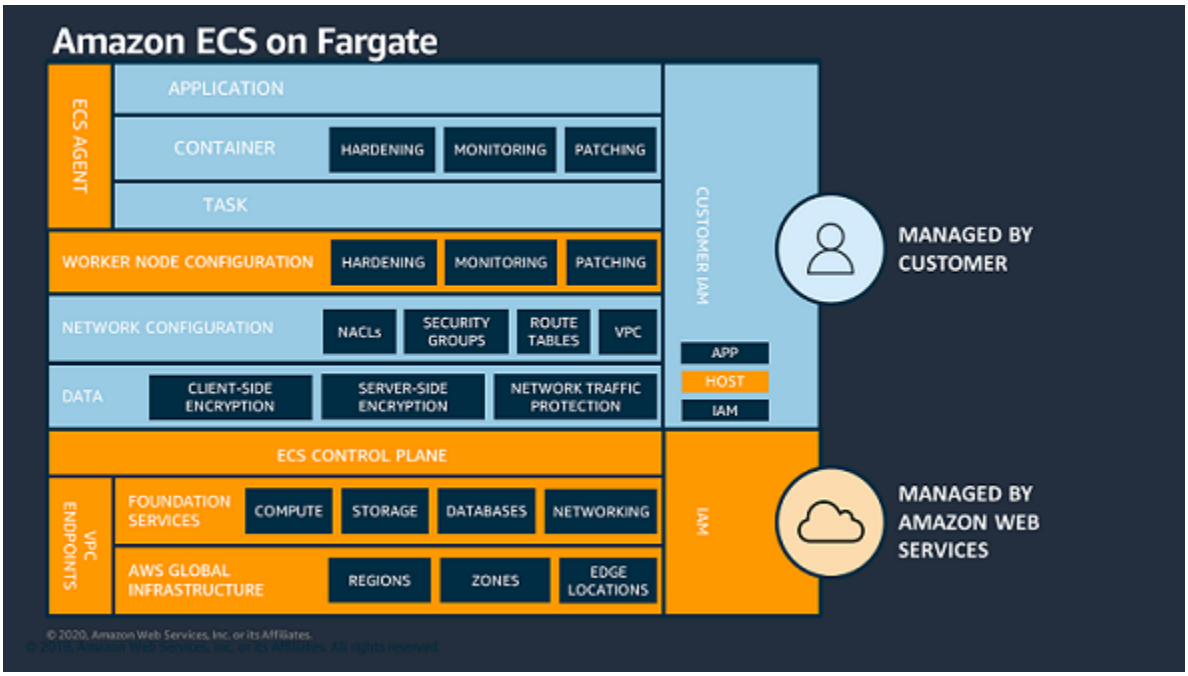
Topik

Model tanggung jawab bersama

Keamanan dan kepatuhan layanan terkelola seperti Amazon ECS adalah tanggung jawab bersama Anda dan AWS. Secara umum, AWS bertanggung jawab atas keamanan “dari” cloud sedangkan Anda, pelanggan, bertanggung jawab atas keamanan “di” cloud. AWS bertanggung jawab untuk mengelola pesawat kontrol Amazon ECS, termasuk infrastruktur yang diperlukan untuk memberikan layanan yang aman dan andal. Dan, Anda sebagian besar bertanggung jawab atas topik dalam panduan ini. Ini termasuk keamanan data, jaringan, dan runtime, serta pencatatan dan pemantauan.



Sehubungan dengan keamanan infrastruktur, AWS memikul lebih banyak tanggung jawab atas AWS Fargate sumber daya daripada untuk contoh yang dikelola sendiri lainnya. Dengan Fargate, AWS mengelola keamanan instance yang mendasarinya di cloud dan runtime yang digunakan untuk menjalankan tugas Anda. Fargate juga secara otomatis menskalakan infrastruktur Anda atas nama Anda.



Sebelum memperluas layanan Anda ke cloud, Anda harus memahami aspek keamanan dan kepatuhan apa yang menjadi tanggung jawab Anda.

Untuk informasi selengkapnya tentang model tanggung jawab bersama, lihat Model Tanggung Jawab Bersama.

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

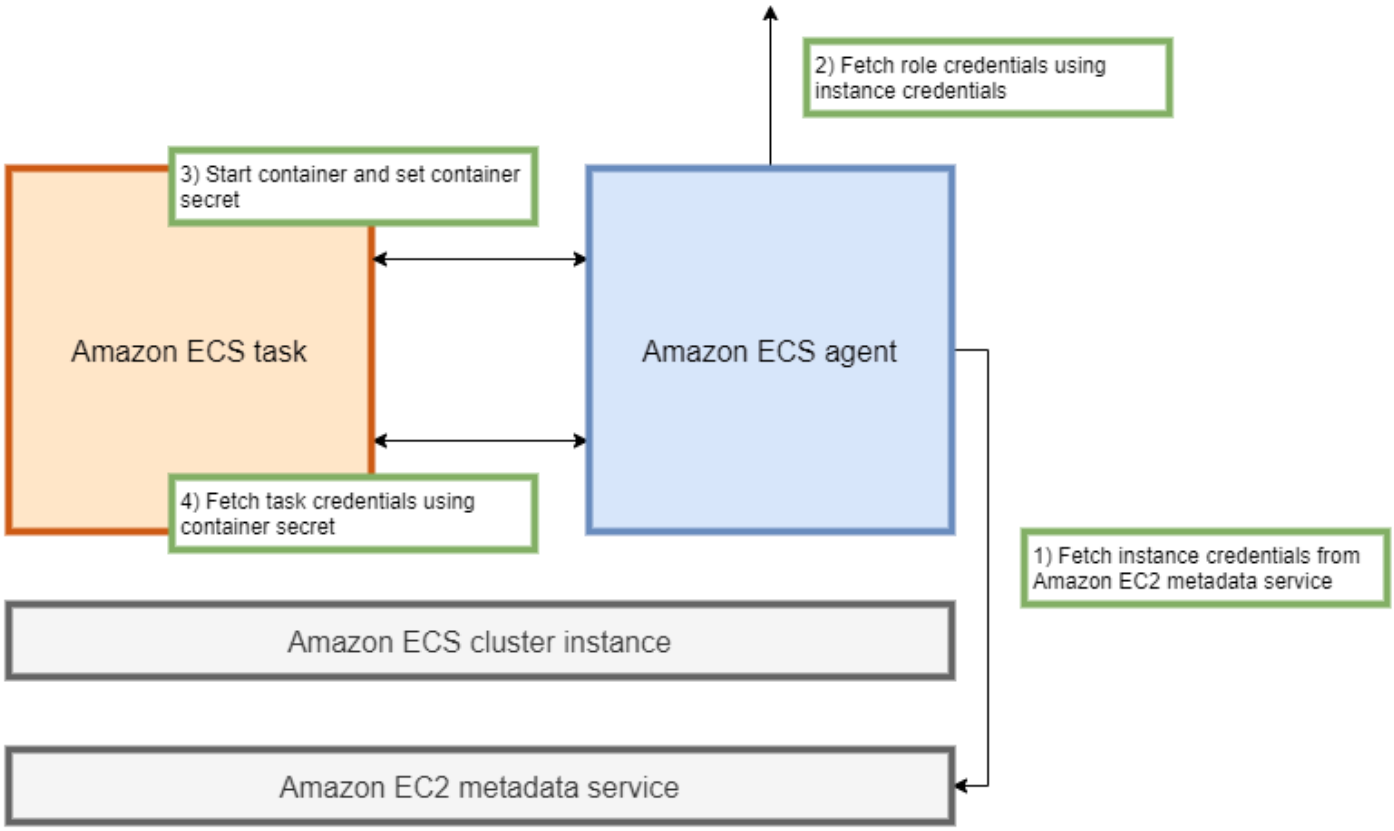
Kami menyarankan Anda menetapkan tugas peran IAM. Perannya dapat dibedakan dari peran instans Amazon EC2 yang sedang berjalan. Menetapkan setiap tugas peran selaras dengan prinsip akses yang paling tidak memiliki hak istimewa dan memungkinkan kontrol terperinci yang lebih besar atas tindakan dan sumber 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 kredenal 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 mengambil kredenal peran sementara secara manual 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 kredensyal 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, kredensyal 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 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 Server Name Indication (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 merujuk 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. Sebaliknya, 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.

Service mesh dan Mutual Transport Layer Security (mTLS)

Anda dapat menggunakan mesh layanan seperti AWS App Mesh untuk mengontrol lalu lintas jaringan. Secara default, node virtual hanya dapat berkomunikasi dengan backend layanan yang dikonfigurasi, seperti layanan virtual yang akan berkomunikasi dengan node virtual. Jika node virtual perlu berkomunikasi dengan layanan di luar mesh, Anda dapat menggunakan filter `ALLOW_ALL` keluar atau dengan membuat node virtual di dalam mesh untuk layanan eksternal. Untuk informasi lebih lanjut, lihat Panduan Cara Jalan Keluar Kubernetes.

App Mesh juga memberi Anda kemampuan untuk menggunakan Mutual Transport Layer Security (mTLS) di mana klien dan server saling diautentikasi menggunakan sertifikat. Komunikasi selanjutnya antara klien dan server kemudian dienkripsi menggunakan TLS. Dengan mewajibkan mTL antar layanan dalam mesh, Anda dapat memverifikasi bahwa lalu lintas berasal dari sumber tepercaya. Untuk informasi selengkapnya, lihat topik berikut.

- otentikasi mTLS
- Panduan Layanan Penemuan Rahasia MtLS (SDS)
- Panduan File mTLS

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 kredensyal 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` wadah.
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 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 dari 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 pada 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.

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 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 FirelensConfigurationdi Panduan AWS CloudFormation 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.

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.

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.

- Alat Keamanan Aplikasi Sumber Terbuka 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 `RO` (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 instance 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 yang 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 kontainer 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 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.

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.

Riwayat dokumen untuk Panduan Praktik Terbaik Amazon ECS

Tabel berikut menjelaskan rilis dokumentasi untuk Panduan Praktik Terbaik Amazon ECS.

Perubahan	Deskripsi	Tanggal
Mengoperasikan Amazon ECS pada praktik terbaik skala	Menambahkan praktik terbaik untuk mengoperasikan Amazon ECS dalam skala besar.	1 Juni 2022
Mempercepat praktik terbaik peluncuran tugas	Menambahkan praktik terbaik untuk mempercepat peluncuran tugas Amazon ECS Anda.	1 April 2022
Praktik terbaik aplikasi	Menambahkan praktik terbaik untuk mengimplementasikan aplikasi Anda dengan Amazon ECS.	28 Oktober 2021
Praktik terbaik penerapan	Menambahkan praktik terbaik untuk mempercepat penerapan Amazon ECS.	Agustus 10, 2021
Praktik terbaik keamanan	Menambahkan praktik terbaik untuk manajemen keamanan untuk beban kerja Amazon ECS.	26 Mei 2021
Praktik terbaik penskalaan otomatis dan manajemen kapasitas	Menambahkan praktik terbaik untuk penskalaan otomatis dan manajemen kapasitas untuk beban kerja Amazon ECS.	14 Mei 2021

Praktik terbaik penyimpanan persisten	Menambahkan praktik terbaik untuk penyimpanan persisten untuk beban kerja Amazon ECS.	7 Mei 2021
Praktik terbaik jaringan	Menambahkan praktik terbaik untuk manajemen jaringan untuk beban kerja Amazon ECS.	6 April 2021
Rilis awal	Rilis awal Panduan Praktik Terbaik Amazon ECS	6 April 2021

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